Parallel Optimisation Algorithms for Continuous, Non-Linear Numerical Simulations

by

Andrew Lewis

B.E.(Hons.), University of Newcastle

School of Computing and Information Technology
Faculty of Engineering and Information Technology
Griffith University,
Nathan campus, Brisbane, Australia

Submitted in fulfilment of the requirements
of the degree of Doctor of Philosophy

May, 2004

ABSTRACT

Parallel Optimisation Algorithms for Continuous, Non-Linear Numerical Simulations

Andrew Lewis

Griffith University,

Nathan campus, Brisbane, Australia

In computational science and engineering there are growing numbers of increasingly sophisticated, rigorous and realistic numerical simulations of physical systems. Detailed knowledge of a particular area of enquiry is expressed in mathematical terms, realised in computer programs and run on increasingly powerful computer systems. The use of such simulations is now commonplace in a growing collection of industrial design areas.

Often, users of these models want to understand their behaviour in response to a variety of input stimuli, bounded by various operational parameters. Commonplace in the engineering design process is the need to find the combination of design parameters that minimise (or maximise) some design objective. Optimisation programs seek to apply mathematical techniques and heuristics to guide a computer in choosing trial parameter sets itself in an attempt to satisfy the expressed design objective.

The more realistic the numerical simulations become, the more demanding of computational resources they become. Many of them consume hours, or days, of computing time on supercomputers to deliver a single trial solution. Optimisation algorithms invariably require the model be run more than once, often many times. In the absence of any means to reduce the computational cost of a single run any further, there can be two responses to this dilemma:

1. Reduce the number of model evaluations required by the optimisation algorithm, or
2. reduce the time the whole collection of model evaluations takes by running as many as possible at the same time.

The research in this thesis is directed toward developing methods that use the approach of parallel computing to reduce total optimisation time by exploiting concurrency within the optimisation algorithms developed. For generality it assumes the numerical simulations to which it may be applied will have real-valued parameters, i.e. they are continuous, and that they may be non-linear in nature.

The following contributions are described:

- The idea of developing a set of "sandbox" case studies for effective testing of optimisation algorithms is presented and established as a feasible alternative to the use of artificial test functions. An initial set of problems with varying characteristics is also presented.
- A parallel implementation of the quasi-Newton gradient method with BFGS update and its efficacy in comparison to a corresponding sequential algorithm and widely-used method of simulated annealing is demonstrated.

- The use of a method of parallel line search with the Nelder-Mead simplex algorithm and its advantages compared to the original algorithm, in speed and reliability, are clearly shown.
- New direct search methods, the Reducing Set Concurrent Simplex (RSCS) algorithm with line searching variants, are presented, and their superior performance compared to a variety of direct search methods demonstrated.
- A novel Evolutionary Programming algorithm using concepts of self-organised criticality, EPSOC, is presented, and demonstrated to be superior in performance to a wide variety of gradient, direct search and stochastic methods on a set of test cases drawn from real-world problems. Evidence is presented of its potential for multi-objective optimisation using a novel implementation with multiple, "virtual" archives.
- Methods of preconditioning optimisation problems to reduce the total time taken to achieve an optimal result are presented. Temporal preconditioning, based on the time behaviour of the numerical simulations, is demonstrated to yield substantial speedup.
- Some conclusions have been drawn on the applicability of specific optimisation methods to different classes of real-world problems.

All of the methods described are implemented in the framework of a general-purpose optimisation toolset, Nimrod/O, to provide a sound basis for future work, easy adoption across a wide range of engineering design problems and potential commercial application.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

The undertaking of a PhD does not occur in a vacuum. Numerous people have given assistance, and I should like to mention several in particular.

I can truly say that without Professor David Abramson's constant assistance, guidance and encouragement this work would likely not have been completed. I consider him a valuable mentor and friend, and have enjoyed the opportunity to work with him.

Mr. Tom Peachey, the third member of our research team, has been tireless in his work in programming and bringing to fruition the Nimrod/O framework, which proved invaluable to the progress of this research.

I have been fortunate to collaborate with a number of colleagues in the course of this research, particularly in the application of the methods to a range of case studies. I would like to thank:

Dr. Brenton Hall, for access to the quantum electro-dynamics models from which were drawn the data for the "Laser" test cases.

Professor Rhys Jones, for access to the models and data for the fatigue life engineering applications.

Professor Clive Fletcher, for access to the two-dimensional aerofoil model.

Dr. Seppo Saario, for access to the radio-frequency design cases, "Bead" and the multi-element antenna simulation, and for his enthusiastic cooperation in their investigation.

Dr Martin Cope and Dr Maciej Skierski for their assistance with the photochemical smog modelling.

Dr. Marcus Randall, for his valuable advice on methods of statistical analysis.

Dr. Oliver Sharpe, for a most stimulating discussion on the No Free Lunch theorem.

Several members of staff at Griffith University have provided material assistance and encouragement during the course of this work. I would like to thank Professor Barry Harrison, Associate Professor Liisa von Hellens, Professor Max Standage and Mr. Geoffrey Dengate.

Finally I would like to thank family and friends for their support and forbearance.

STATEMENT OF ORIGINALITY

This work has not previously been submitted for a degree or diploma in any university. To the best of my knowledge and belief, the thesis contains no material previously published or written by another person except where due reference is made in the thesis itself.

_____

Andrew Lewis

1.0 Introduction

In scientific and engineering research and design, a third investigative paradigm has developed to complement the traditional approaches of theory and experiment: *computational* science and engineering. This comparatively recent methodology makes use of computers to investigate the behaviour of complex systems, aid in the design process, and venture where environments may be too hostile for experiment, or intractable for analytical solution.

Underpinning this approach are increasingly sophisticated, rigorous and realistic numerical simulations of physical systems. Detailed knowledge of a particular area of enquiry is expressed in mathematical terms, realised in computer programs and run on increasingly powerful computer systems. The use of such simulations is now commonplace in a growing collection of industrial design areas, with automotive and aeronautical design being notable early adopters.

Often, users of these models want to understand their behaviour in response to a variety of input stimuli, bounded by various operational parameters. A simplistic approach is to construct the set of all possible combinations of the parameters of interest and run the model for all these cases. Even with powerful supercomputers, parallel and distributed resources, this quickly runs into the problem of "combinatorial explosion": as the number of parameters of interest increases, the number of required model experiments needed to adequately investigate the model response increases exponentially, rapidly becoming too many to feasibly contemplate computing. A desire for detailed knowledge of the response, increasing the resolution needed, merely exacerbates the problem.

Many users, particularly design engineers working to the tight deadlines of industry, respond by trying to "outguess" the model, applying experience and specialist knowledge to limit the number of times a model is run. As a result their searches are often truncated and the results they achieve may be sub-optimal. It is the role of optimisation programs to alleviate this problem by applying mathematical techniques and heuristics to guide the computer in choosing trial parameter sets itself in an attempt to satisfy the expressed design objective.

Typically, the more realistic the numerical simulations become, the more demanding of computational resources they become. Many of them consume hours, or days, of computing time on supercomputers to deliver a single trial solution. Optimisation algorithms invariably require the model be run more than once, often many times. In the absence of any means to reduce the computational cost of a single run any further, there can be two responses to this dilemma:

1. Reduce the number of model evaluations required by the optimisation algorithm, or
2. reduce the time the whole collection of model evaluations takes by running as many as possible at the same time.

The dominant architecture in high performance computing in recent times has been parallel, either within a single machine, or through a collection of machines linked by networks, "computational grids". This provides additional motivation for developing a parallel computation approach.

The design engineer who uses the optimisation program, while an expert in the application domain, cannot always also be expected to be an expert in computer science. Ideally, to be useful a general purpose optimisation tool should be easily applicable to a wide range of problems without assuming specialised knowledge in methods of optimisation from the user. The more it can be treated as a "black box", the wider its potential adoption and the greater its end benefit.

The research in this thesis is directed at addressing the main issues briefly outlined above. It uses the approach of parallel computing to reduce total optimisation time by exploiting concurrency

within the optimisation algorithms developed. For generality it assumes the numerical simulations to which it may be applied will have real-valued parameters, i.e. they are continuous, and that they may be non-linear in nature. All the methods described have been implemented in a general-purpose optimisation toolset for their ready application to a wide range of problems.

1.1 Scope

The general optimisation problem can be written in terms of an objective function, $f : \Re^n \rightarrow \Re^1$ where $\Re^n$ denotes a Euclidean space of ordered n-tuples of real numbers, as:

$$\min\{f(x) \,|x \in X \subseteq \Re^n\} \tag{1-1}$$

There appear no satisfactory tests for establishing whether a candidate solution to the problem is (globally) optimal or not (Polak 1971). All the algorithms to be described in this thesis can only be used to construct a series whose limit satisfies some optimality condition. In the absence of convexity assumptions, such a condition is only a necessary condition of optimality, not sufficient condition. Furthermore, there is a broad variety of problems in which uni-extremality cannot be simply postulated or verified (Bongartz, Conn, Gould and Toint 1995), in particular the optimised design and operation of complex "black box" systems, e.g. in diverse engineering applications. Practitioners often resort to local improvement heuristics.

A large class of algorithms finds points:

$$x^* \in \Re^n : \nabla f(x^*) = 0 \tag{1-2}$$

Such a point is a *desirable* or a *stationary point*. Under the assumption that f is convex, points satisfying Equation 1-2 *may* be "optimal". They may not, in fact, be local minimizers, as Equation 1-2 can also be satisfied by maximizers and saddle points. Usually a minimizer corresponds to a positive definite Hessian matrix, a maximizer to a negative definite matrix and a saddle point to an indefinite matrix. The Hessian matrix, **A**, is a matrix of second partial derivatives of f:

$$A = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix} \tag{1-3}$$

For a twice-differentiable function a general approximation in the region of a minimizer can be simply expressed as a quadratic. The most straightforward method of solving such an equation is *Newton's method* (see, for example, Fletcher (1987),Gill, Murray and Wright (1981),Beale (1988) or other introductory texts). This method, however, requires zero, first and second derivatives of the objective function be available at any point. Closely related methods have been derived that only require first derivatives, or approximations to them. Methods that make explicit use of ∇f will be termed *gradient methods*.

Alternatively, algorithms may make no explicit use of gradient information, but instead explore the parameter space relying only on objective function values. Exploratory evaluations may be governed by some defined pattern, "stencil" or structure, in which case they are termed *pattern search*

or *direct search methods*, or the exploratory process may have some (usually carefully controlled) randomness yielding the class of *stochastic methods*.

From these distinctions a simplified taxonomy of optimisation methods can be constructed, and one such overview is illustrated in Figure 1.1. Bäck (1996) draws a preliminary distinction between volume-oriented methods, based on the idea that the whole feasible region must be scanned, and path-oriented methods, that follow a path in the feasible region starting from an arbitrary or "best-so-far" point. Figure 1.1 does not seek to drawn these distinctions, but can be seen as encompassing both. It is interesting to note that volume-oriented methods imply a requirement that the search space be restricted to a finite volume. A review of representative methods from the major classes of Figure 1.1 is given in Chapter 2.

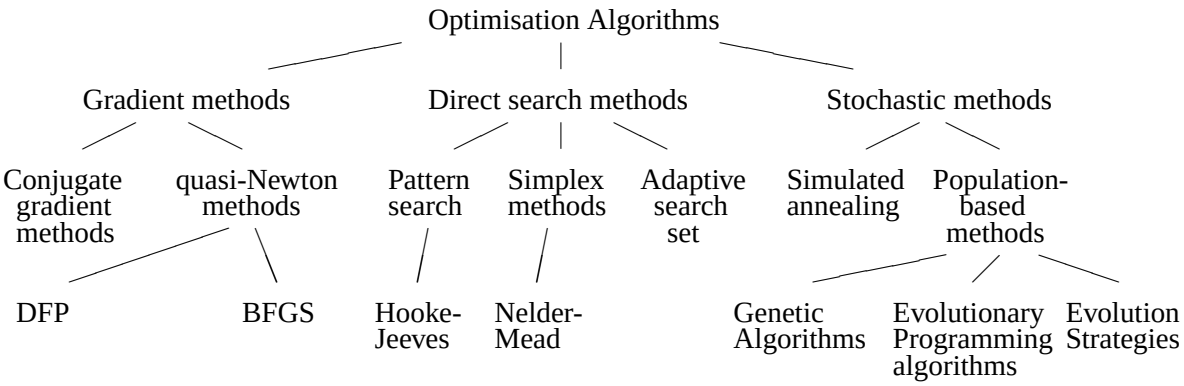


Figure 1.1 A taxonomy of optimisation algorithms

The work in this thesis contains details of new, parallel optimisation algorithms and hybrids of existing methods that expand upon and improve several of the classes in Figure 1.1. This study, primarily motivated by the needs of real design problems, begins by developing a set of test problems drawn from real-world applications, extracting and abstracting their information to provide real challenges, rather than simplified mathematically constructed test cases. Details of the test cases are outlined in Chapter 3. New algorithms, and contributions to existing algorithms, in *gradient methods*, *direct search methods* and *stochastic methods* are discussed in Chapters 4, 5 and 6 respectively. Results of numerical experiments and statistical comparison of methods are also given. Methods of "conditioning" the target problems to improve optimisation performance are discussed in Chapter 7. Finally, in Chapter 8, conclusions are drawn about the new algorithms, contributions to existing methods, and their implementation. Supplementary material is presented on an accompanying CD, including:

- Virtual Reality Modeling Language (VRML) data sets for the case studies described in Chapter 3,
- some animations of the operation of new algorithms discussed in Chapters 5 and 6, and
- an animation of the operation of spatial preconditioning discussed in Chapter 7.

1.2 The Nimrod/O toolset

As outlined above, a fundamental objective of this research is its simple and effective application to real problems. The more the process of optimisation can be treated as a the function of a “black box”, the more readily it may be applied to a wide range of problems. To meet these needs, the algorithms described in this thesis have been implemented as components of a fully integrated optimisation toolset, Nimrod/O (Abramson, Lewis and Peachey 2000, Abramson, Lewis and Peachey 2001, Abramson, Lewis, Peachey and Fletcher 2001a).

Nimrod/O is a development of the Nimrod research project (Abramson, Sosic, Giddy and Hall 1995, Lewis, Abramson, Sosic and Giddy 1995, Abramson, Foster, Giddy, Lewis, Sosic, Sutherst and White 1997), incorporating automatic optimisation into the framework of what was originally a parameter sweep toolset. Nimrod allows a scientist or engineer to succinctly describe their numerical simulation, define parameter ranges and perform automatic explorations of parameter space using enumeration of the cross-product of the defined parameters on parallel or distributed computers. It is a tool specifically designed for the task of performing “parameter sweeps” of simulations, varying parameter values between upper and lower limits in a step-wise manner, and evaluating the simulation at each step. The step size is typically chosen for reasons of desired resolution or feasible computation, given limited time or resources.

While an extremely useful tool, Nimrod suffers from the shortcoming identified in Chapter 3: combinatorial explosion of required model evaluations as problem dimensionality and desired solution resolution increased. Nimrod/O seeks to avoid this problem by extending the toolset to include automatic optimisation in the same, easily usable framework. It is the product of a continuing research project involving principally Abramson, Lewis and Peachey, to which the author has contributed mainly the design and development of parallel optimisation algorithms.

The structure of Nimrod/O and outline of its operation is shown in Figure 1.2. Control is via a declarative plan file, a simple description of the execution of the numerical model and optimisation problem, including parameters, their ranges, and details of the optimisation methods to be used. An example plan file is shown below. Nimrod/O interprets the plan file, sets up the run environment and passes control to the requested optimisation algorithm(s). Nimrod/O explicitly supports simultaneous execution of multiple optimisation runs from different starting points, and simultaneous use of multiple optimisation methods.

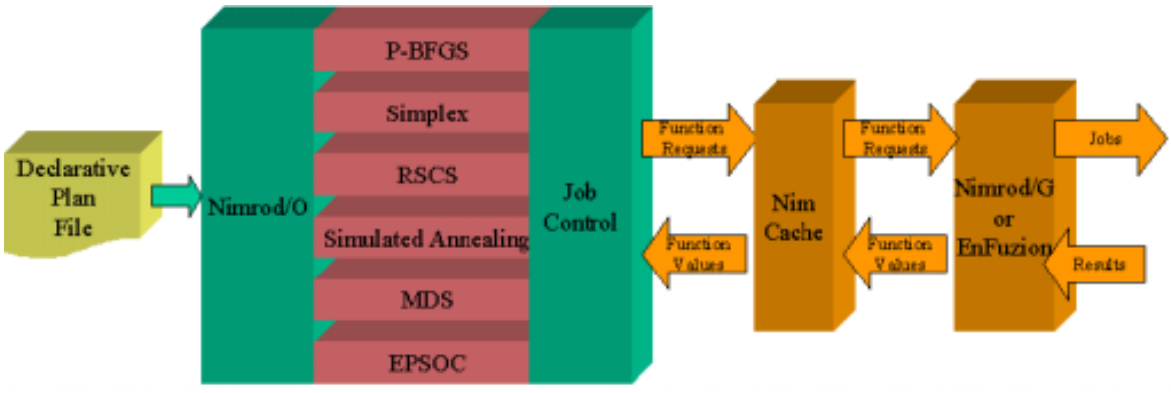


Figure 1.2 The Nimrod/O Architecture

An optimisation algorithm, as it runs, passes requests to Nimrod/O job control for objective function evaluations to be performed. These can be grouped in batches where the algorithm is

capable of generating multiple concurrent tasks. Job control checks the requested set of parameters against a cache of previously computed results and then automatically dispatches evaluation jobs to computing resources provided by the user, whether they be parallel or distributed computers, via one of a number of job distribution mechanisms. Currently two are provided in the standard toolset:

- a Globus-enabled tool for use of computational grid resources (Foster and Kesselman 1997).
- an API to EnFuzion, a commercially available implementation of the original Nimrod tool that allows use of parallel computers or collections of workstations on a LAN.

These tools take care of job submission, execution tracking and input and output file handling.

1.2.1 A Nimrod/O plan file

Following is a simple, example declarative plan file. Further details of the syntax can be found in the Nimrod/O Users' Guide (Peachey 2003a).

```
parameter b float range from 5 to 35
parameter tness float range from 2 to 9
parameter bias float range from -0.02 to 0.02

task main
   copy  runfiles/* node:.
   node:execute ./run.script $b $tness $bias
   copy node:result.dat  output.$jobname
endtask

method simplex
   starts 9 named "simplex"
      starting points specified in starters
      tolerance 0.005
  endstarts
endmethod

method epsoc
  starts 10 named "EPSOC"
    starting points random
    maximum iterations 20
    population 64
    tolerance 0.005
  endstarts
endmethod
```

The plan file starts by defining the parameters. A parameter is named, its type defined, how it will be expressed and, where appropriate, the bounds on its value. For example, parameter "b" is defined as being a floating point value, to be expressed in a range from a lower bound of 5.0 to an upper bound of 35.0:

```
parameter b float range from 5 to 35
```

Supported parameter types are float, integer and text. Float and integer types must be expressed as a range of values, text as a list of values. Where text parameters are used, the listed cases will be evaluated by simple enumeration, i.e. multiple optimisations will be performed, one for each value of the text parameter.

Then, in the section labelled as “task main”, a brief description is given of how to run the numerical simulation. Input and output of the simulation are assumed to be via named files. This has been a common method of interaction with large-scale simulations, and is an easy method independent of the model implementation. For each objective function evaluation task input files are copied to the “node” on which execution is to be scheduled:

```
copy  runfiles/* node:.
```

All other files necessary for execution of the simulation should also be copied, including all necessary scripts and executables. Then a user-provided script is run on the node:

```
node:execute ./run.script $b $tness $bias
```

It is assumed that the script will provide the commands to run the simulation and derive the single, floating-point objective function value. It may be directly output from the model, or derived by post-processing of the model output, in which case the script should also perform the post-porcessing steps. It may be noted that the parameters are provided to the script via named environment variables. These will be substituted by the particular parameter values supplied by the optimisation algorithm at run-time.

The results, written to a file, are copied back to the scheduling node, and placed in a file with the standard name “output” and an extension identifying the particular job:

```
copy node:result.dat  output.$jobname
```

Following the description of the model execution are two sections specifying two different algorithms to be used. These will both be executed simultaneously. Parameters for the algorithms are kept to a minimum, are generally intuitive, and provided with sensible default values. In the example shown, nine simultaneous runs will be performed using a parallel implementation of the Nelder-Mead Simplex algorithm, from starting points specified in a user-supplied file:

```
starts 9 named "simplex"
   starting points specified in starters
```

At the same time, ten runs will be performed using the new EPSOC algorithm, with randomly generated starting seeds, a limit of 20 iterations, and a population size of 64.

```
starts 10 named "EPSOC"
  starting points random
  maximum iterations 20
  population 64
```

Both algorithm descriptions specify a desired solution tolerance:

```
  tolerance 0.005
```

For the Simplex algorithm this specifies the convergence criterion by defining the magnitude of the fractional gradient of the final simplex. As will be described in Chapter 6 EPSOC uses the "maximum iterations" value specified to terminate iteration. In this case, the "tolerance" is used to control the operation of NimCache, by defining the minimum separation between parameter values that will be stored as distinct values in the results cache.

This is sufficient to specify the entire optimisation experiment to Nimrod/O.

Nimrod/O includes a syntax for the imposition of constraints on the parameter space. Both barrier and penalty function constraints can be defined by algebraic expressions. The use and impact of constraints has not been considered in this research, so further information on their use can be found by reference to the Nimrod/O Users' Guide.

Nimrod/O provides a simple, easy to use framework for the application of optimisation to a wide range of problems. No additional programming is required to integrate numerical simulations into the optimisation framework, and no access is generally required to the internal operations of the simulation. Nimrod/O is thus readily usable even with proprietary simulation packages, as is demonstrated in the case study described in Section 3.4. All the algorithms to be described in following chapters have been developed, tested and fully integrated in the Nimrod/O framework.

2.0 Optimisation Methods

When the evaluation of the optimisation objective is computationally very demanding, as is the case with perhaps the majority of the engineering and scientific numerical simulations that constitute the domain for this research, there are two main approaches that can be tried to make the optimisation problem tractable:

1. Choose an optimisation method that minimises the absolute number of function evaluations required to find an optimal solution.

2. Choose an optimisation method that seeks to minimise the total time taken to find an optimal solution by performing as much of the computation required in parallel.

It is the basic intention of this research to find, develop and exploit methods using the latter approach, to build a general purpose toolset that can be deployed on parallel or distributed computing resources. Looking at the taxonomy of methods in Figure 1.1, which methods are best suited to which of these two approaches?

Moving from left to right in the diagram to some extent parallels the historical development of optimisation methods. Early algorithms sought to build on the foundations of mathematical knowledge embodied in Newton's method, least squares minimization and other, manually-executed, largely sequential techniques. The introduction of variable metric methods by Davidon (1959) was a revolutionary step forward in sophistication of methods. In a *very* general sense moving from left to right in the diagram of Figure 1.1 also indicates some degree of increasing parallelism of methods. Until the rise of multi-processor high performance computers in the late 1980's and early 1990's, the accepted wisdom continued to make much of the advantages of sequential processing, using knowledge from earlier iterations of a method to guide later steps. Indeed, in the early 1960's the opinion was expressed that "Naturally, simultaneous schemes are much less effective than sequential plans" (Wilde 1964). Several authors struggled, briefly, with the question of what circumstances might justify the use of simultaneous search methods, and could see little reason *unless working to a deadline in real life* (Wilde 1964, Cooper and Steinberg 1970). Obviously, the limited computing resources available at that time did much to mitigate against ideas of using "speculative" computation. The gradient methods, while rapid to converge on local minima, generally have limited concurrency, i.e. few independent tasks within the algorithm that can be performed simultaneously.

At about this time early direct search methods (Hooke and Jeeves 1961, Spendley, Hext and Himsworth 1962, Nelder and Mead 1965) started to appear. Their emphasis was still on sequential procedures, but the structure of some of them, particularly the simplicial methods, had potentially greater concurrency.

The use of stochastic methods for optimization on a large scale is generally a more recent development. While some precursor studies were made at the same time as the rapid development of other optimisation methods in the late 1950's and early 1960's, it was not until the availability of sufficient computing resources to make the methods practical that they saw widespread adoption.

Simulated annealing is based upon the algorithm of Metropolis et al. (Metropolis, Rosenbluth, Rosenbluth, Teller and Teller 1958), which was originally proposed as a means of finding the equilibrium configuration of a collection of atoms at a given temperature. The connection between this algorithm and mathematical minimization was first noted by Pincus (1970). While having the potential for parallel execution, this can have a large impact on its rate of convergence so the technique is usually employed sequentially.

The population-based methods, practical for real-world optimisation only with the availability of sufficient computing resources to support them, are inherently parallel by their nature, and the most recent to experience attention and active research. Alander (1999) has documented an exponential growth in application of genetic algorithms from about the mid-1970's.

The following sections will examine each of these classes of methods in turn, and later chapters will explore the problem of exploiting parallelism within them, and their application to real optimisation problems.

2.1 Gradient Methods

Gradient methods for optimisation are familiar and widely used. A search on a leading citations database returned over 100,000 references to gradient descent optimisation. For this reason the fundamental topics will be approached by reference to introductory texts.

The gradient descent method can be generalized by the following steps:

1. Supply an initial estimate of the solution, $\mathbf{x}_1$
2. At the kth iteration, determine a search direction, $\mathbf{s}_k$, generally a direction in which $f(\mathbf{x})$ decreases.
3. Find α_k to minimize $f(\mathbf{x}_k + \alpha\mathbf{s}_k)$ with respect to α.
4. Set $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k\mathbf{s}_k$.

Methods differ principally in how they proceed at step (2). Step (3), essentially a *line search*, can be exact in theory and inexact, to a greater or lesser degree, in practice.

A first, intuitive approach to setting the line search direction is to set it in the direction of the local downhill gradient, the *steepest descent method.* It can be trivially demonstrated, however, that even in the simplest cases this is not a good approach, leading to very inefficient, zigzagging trajectories. A more desirable trajectory might be constructed by requiring new search directions to maintain any change in the gradient to be perpendicular to the original search direction, i.e. not to spoil minimization along the first search direction while traversing the new search direction. The two search vectors are said to be *conjugate.*

2.1.1 Conjugate Gradient method

The method of conjugate gradients was introduced by Hestenes and Stiefel (1952) (Hestenes 1956) and applied to optimisation by Fletcher and Reeves (1963). The method is an exact line search method in which:

$$\mathbf{s}_0 = -\mathbf{g}_0 \text{ where } g = \nabla f(x) \tag{2-1}$$

and

$$\mathbf{s}_{k+1} = \text{ component of } -\mathbf{g}_{k+1} \text{ conjugate to } \mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_k \tag{2-2}$$

The conjugacy condition can be written as:

$$s_i^T(g_{j+1} - g_j) \quad j \neq i \tag{2-3}$$

and an update formula for the search direction derived:

$$\mathbf{s}_{k+1} = -\mathbf{g}_{k+1} + \beta_k s_k \tag{2-4}$$

$$\text{with } \beta_0 = 0, \text{ where } \beta_k = \frac{\mathbf{g}_{k+1}^T \mathbf{g}_{k+1}}{\mathbf{g}_k^T \mathbf{g}_k}$$

Applied to a quadratic function, the method converges to an exact solution in a finite number of iterations. For minimization of non-quadratic functions the method cannot be guaranteed to terminate. In practice, the method is periodically reset to the steepest descent search direction.

2.1.2 Quasi-Newton method

An alternative class of gradient methods are the *quasi-Newton* or *variable metric methods*, first introduced by Davidon (1959) and later improved by Fletcher and Powell (1963a) in the *DFP method.* A minor variant of the DFP method proposed by Broyden (1970), Fletcher (1970), Goldfarb (1970) and Shanno (1970) is known as the *BFGS update formula*. In practice it has been found to work well, and is considered by some as the "best" quasi-Newton method (Fletcher 1987).

The main advantage of quasi-Newton methods compared to conjugate gradient methods is that they avoid the need for periodic restarts and the consequent loss of information (Beale 1988). They have also been found to be more efficient (Fletcher 1987). Their main disadvantage is the storage requirement for the $n \times n$ Hessian matrix (or its inverse.) With current computing resources, this is a negligible consideration. Some authors have stated the opinion that today a problem of 50 parameters is small and one of 5000 parameters is large (Sartenaer 1995). It may be instructive, however, to inspect the problems presented, for example, in the CUTE set of standard test functions (Bongartz et al. 1995). Of the 75 problems whose solution has been used in a real application and that have a specified number of parameters, the median number of parameters is 8. It would appear that this smaller dimensionality is more representative of the true scale of typical, real-world problems.

The perception that these methods also require a matrix inversion operation to be performed at each iteration, supposedly another disadvantage in comparison with conjugate gradient methods, is false, as the methods can be easily expressed in terms of operations on the inverse of the Hessian matrix.

The basic idea of variable metric methods is to construct a sequence of matrices, $\mathbf{H}_i$, such that:

$$\lim_{i \to \infty} H_i = A_i^{-1} \tag{2-5}$$

where $A_i^{-1}$is the inverse of the Hessian matrix, i.e. a sequence of approximations is made to A_i^{-1}. The kth iteration of a quasi-Newton method can generally be given by:

1. Set $\mathbf{s}_k$ = -$\mathbf{H}_k \mathbf{g}_k$
2. Line search along $\mathbf{s}_k$ yielding x_{k+1} = $\mathbf{x}_k + \alpha_k \mathbf{s}_k$
3. Update $\mathbf{H}_k$ to $\mathbf{H}_{k+1}$

Initial estimates for $\mathbf{x}_0$ and $\mathbf{H}_0$ must be supplied. In practice, $\mathbf{H}_0$ is often set to the identity matrix.

The BFGS update formula is given by (see, for example, Press, Teukolsky, Vetterling and Flannery (1992)):

$$\begin{aligned} H_{k+1} = H_k &+ \frac{(x_{k+1}-x_k)\otimes(x_{k+1}-x_k)}{(x_{k+1}-x_k)\cdot(g_{k+1}-g_k)} \\ &- \frac{[H_k\cdot(g_{k+1}-g_k)]\otimes[H_k\cdot(g_{k+1}-g_k)]}{(g_{k+1}-g_k)\cdot H_k\cdot(g_{k+1}-g_k)} \\ &+ [(g_{k+1}-g_k)\cdot H_k\cdot(g_{k+1}-g_k)]\, u\otimes u \end{aligned} \tag{2-6}$$

where

$$\begin{aligned} u \equiv H_k &+ \frac{(x_{k+1}-x_k)}{(x_{k+1}-x_k)\cdot(g_{k+1}-g_k)} \\ &- \frac{H_k\cdot(g_{k+1}-g_k)}{(g_{k+1}-g_k)\cdot H_k\cdot(g_{k+1}-g_k)} \end{aligned} \tag{2-7}$$

2.1.2.1 The line search sub-problem. Gradient methods applied to multivariate problems essentially reduce the problem to univariate minimization in some specific search direction, a *line search*. It is this search that forms the heart of the overall method, and largely governs its performance. There can be considerable variation in methods, particularly depending on whether derivatives of the objective function are available or not.

The line search basically generates a sequence of trial solutions, terminating when one of them is an acceptable point. Depending on the methods used, for $f(x)$ in $\Re$, the univariate mapping of $f(\mathbf{x})$ in $\Re^n$ in the search direction, this may be a stationary point as defined by Equation 1-2. There are typically two stages to the line search:

1. The algorithm defines a series of points along the search vector, often by sequentially stepping successively greater distances along it, evaluates them, and continues until a bracket on an interval of acceptable points is detected.

2. Some form of interpolation, e.g. by fitting parabolic approximations to the sampled points, is carried out to give a better approximation to a minimal point.

Methods vary in how they define the sequence of trial points, e.g. how the initial step size and its rate of increase are chosen for iterative stepping methods, and whether interpolation is performed, and how that is achieved.

As outlined in Section 4.1.1 below, by reformulation of the problem it is possible to make use of a method of iterative line interval subdivision. This is readily adapted to use of parallel computing resources. The method of equal interval search is well known (Cooper and Steinberg 1970), but in the past its analysis has been limited to the context of uniprocessor computers, with the result that the three-point equal interval search was considered the most "economical" (Cooper and Steinberg 1970, Kiefer 1959). The method has been re-assessed in the light of using multi-processor, parallel computers and the effectiveness of the search in finding the global minimum of a synthetic, periodic function demonstrated to be proportional to the number of sampled points at each step (Peachey, Abramson and Lewis 2001).

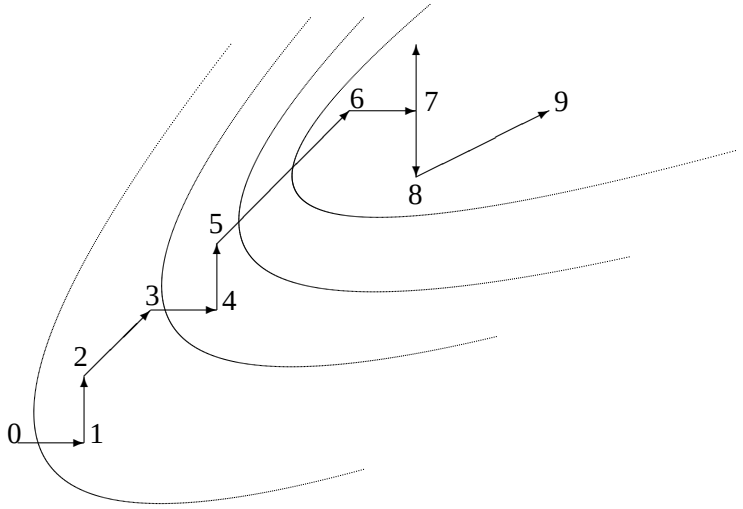


Figure 2.1 The Hooke-Jeeves algorithm in two dimensions

2.2 Direct search methods

Quasi-Newton gradient descent methods typically use finite difference approximations to the gradient if it is not otherwise available. To derive the gradient in this way, a separate step involving evaluation of the objective functions on a finite difference stencil is required. Direct search methods do not explicitly use approximate gradient information, but instead take objective function values from a set of sample points, and use that information to continue sampling.

2.2.1 Hooke-Jeeves algorithm

For example, in an iteration of the Hooke-Jeeves algorithm (Hooke and Jeeves 1961) the objective function is computed on a Cartesian stencil, an "exploratory move" that seeks to accumulate information about the local behaviour of the objective function, and the returned values are used to determine a search direction. A pattern move is then made in the search direction, i.e. the origin of the exploration stencil is relocated based on information from the exploration. An *ad hoc* development of the *alternating variables method*, it sought to make better progress by extending along apparently favourable directions. The process is illustrated in two dimensions in Figure 2.1.

Referring to Figure 2.1, from the starting base point at 0, an exploration probe is made in the first coordinate direction. The point sampled is an improvement on 0, so it is accepted as point 1. A probe is made in the second coordinate direction, is also successful, and the new point, 2, is accepted. All coordinate directions have been probed, so a pattern move is made in the direction of the sum of the probe vectors, minus the (null) vector at 0. This pattern move improves the objective function, so it is accepted as point 3.

The exploration phase is repeated, with successful moves to points 4 and 5. The pattern move is now the sum of the previous pattern move, and the exploratory moves, i.e. it extends in the direction experience has indicated is successful. The new pattern move is also successful, and point 6 is accepted. A probe to point 7 is successful, so it is accepted, but the probe in the second coordinate

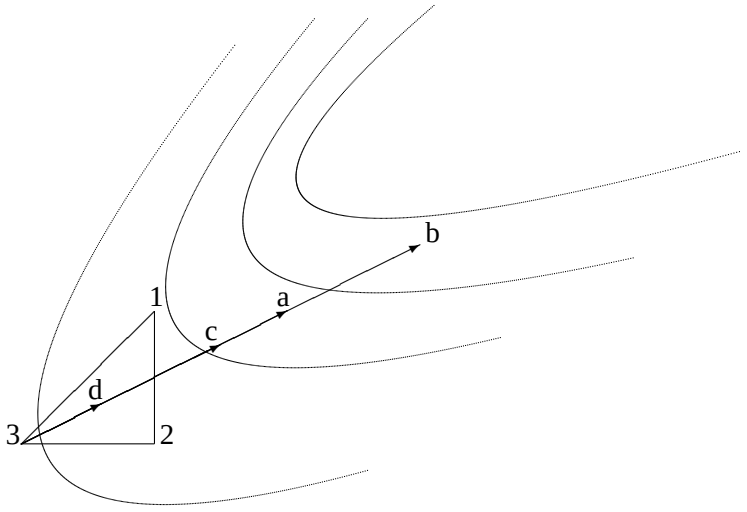


Figure 2.2 The Nelder-Mead algorithm in two dimensions – step 1

direction from point 7 is not. Another probe is tried in the opposite direction, succeeds and point 8 is accepted. Then a pattern move is made to point 9 and the process continues.

If all exploration moves fail, they are repeated with a reduced step size. The method terminates when the step size is reduced below a given threshold.

2.2.2 Simplex methods

In the Nelder-Mead simplex algorithm (Nelder and Mead 1965), the $n + 1$ vertices of a simplex of approximations to an optimal point in n-dimensional space are sampled, ordered by objective function value, and an attempt made to replace the worst vertex by reflection through the convex hull of the remaining vertices, using limited sampling along the search direction so defined. Use of the Nelder-Mead simplex algorithm remains current, largely because, as confirmed during these investigations, on a range of practical engineering problems it is capable of returning a very good result (Wright 1995). It is also robust to small perturbations or inaccuracies in objective function values (Neddermeijer, van Oortmarssen, Piersma, Dekker and Habbema 2000). The Nelder-Mead algorithm in two dimensions is illustrated in Figures 2.2 and 2.3.

In Figure 2.2 the initial simplex is constructed, often aligned with coordinate directions for want of a better guess, the objective function values obtained at the vertices and the vertices ordered by the values. A search vector is constructed from the worst vertex, point 3, through the centroid of the remaining vertices. In two dimensions this lies in the direction through the centre of the simplex edge 1-2.

The original algorithm prescribed a reflection of the worst vertex be tried, i.e. to point *a*. If this was successful, i.e. it produced simple decrease in the objective function value, an extension was tried in the same direction, i.e. to point *b*. If this was successful, it became the new vertex and the process was repeated for the next worst vertex.

If point *a* was unsuccessful, a contracted step was tried. Variants of the algorithm placed this at point *c* or point *d*. If these points also prove unsuccessful, all vertices 2,3,... are contracted toward the best vertex, 1.

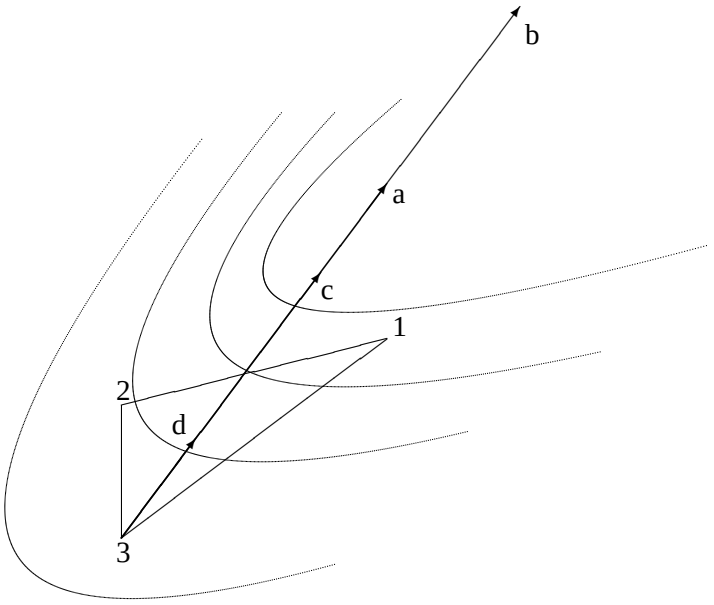


Figure 2.3 The Nelder-Mead algorithm in two dimensions – step 2

In Figure 2.2, the extension to point *b* is successful, it becomes a vertex of the new simplex, shown in Figure 2.3, the vertices are reordered, and a new search begins.

Some criticism has been voiced over the arbitrary choice of the sample points, with no theoretical basis for choosing the expansion and contraction factors (Shekarforoush, Berthod and Zerubia 1995). The simple reflection operation has been found to be advantageous in that it maintains congruency of the sequence of simplices, a desirable property since it prevents degeneration of the simplex (Torczon 1989, Kelley 1999). It may be noted that the Nelder-Mead algorithm always samples at least two points, the location of which are entirely defined by the geometry of the simplex. This inherent concurrency is readily exploitable.

More recently, the work of Dennis and Torczon (Dennis and Torczon 1991, Torczon 1989, Torczon 1991) has yielded abstractions of the simplicial methods with the potential for greatly enhanced parallelism through simultaneous treatment of multiple vertices, and speculative computation of multiple stages of the algorithms. A straightforward implementation of the first of these ideas has been embodied in their Multidirectional Search (MDS) method.

2.3 Stochastic methods

2.3.1 Simulated Annealing

The Simulated Annealing (SA) group of algorithms are based on an analogy to the annealing process used in metals as they cool. Initially a problem is defined with a high “temperature” which allows the next solution point to be relatively random. As the number of trial solution points increases, a probability density function increasingly confines the next solution point to be closer to a known minimum. Depending on the simulation parameters, the optimisation algorithm is able to solve complex problems with some statistical guarantee of finding the global minimum.

The basic components of the SA algorithm are briefly discussed below (Kirkpatrick, Gerlatt and Vecchi 1983):

2.3.1.1 Generating the Probability Density Function. In a D-dimensional parameter space with parameters p^i having ranges $[A_i, B_i]$, about the k'th last saved point (e.g, a local optima), p_k^i, a new point is generated using a distribution defined by the product of distributions of each parameter, $g^i(y^i; T^i)$ in terms of random variables y^i in [-1, 1], where $p_{k+1}^i = p_k^i + y^i(B_i - A_i)$ and "temperatures" T_i.

$$g^i(y^i; T_i) = 1/2(\left|y^i\right| + T_i)\ln(1 + 1/T_i)$$

For each point p_k^i the numerical simulation is run and a cost function, $C(p_k)$, is a measure of the suitability of the solution obtained.

2.3.1.2 Acceptance Probability Density Function. The cost functions, $C(p_{k+1}) - C(p_k)$, are compared using a uniform random generator, U in [0,1] in a "Boltzmann" test: If

$$\exp[-(C(p_{k+1}) - C(p_k))/T_{\cos t}] > U$$

where $T_{\cos t}$ is the "temperature" used for this test, then the new point is accepted as the new saved point for the next iteration. Otherwise, the last saved point is retained.

The parameter "temperatures", normally reduced as some simple function of the algorithm iteration count, may be periodically adaptively reannealed, or increased relative to their previous values, using their relative first derivatives with respect to the cost function, to guide the search evenly among the parameters.

2.3.2 Population-based methods

Population-based methods involve large numbers of objective function evaluations, typically performed in parallel, evolving solutions over several "generations". A particular subset of these methods is evolutionary computation, which seeks to use insight into natural processes to inform computational methods. The resulting Evolutionary Algorithms are popularly differentiated into three main classes (Bäck 1996) namely, Genetic Algorithms, Evolutionary Strategies and Evolutionary Programming.

In the general, population-based method, multiple instances of a problem, each represented by a vector of parameter values, are subject to various operators so that a population of problem instances in a "parent" generation evolve into a "child" population. This process is repeated through a number of generations.

2.3.2.1 Genetic algorithms. Widely known, these methods are generally accepted as having been developed by Holland (1975). Genetic Algorithms, in contrast to Evolutionary Strategies and Evolutionary Programming, work on bitstrings of fixed length. For problems of continuous variable parameters the bitstring has a mapping to the vector of parameters, which generally implies they are capable of returning approximate, rather than exact, global minima. Recent practice has been to use Gray code interpretation of the bitstring segments so that the representations of adjacent integer values have Hamming distance one. Even so, changing a single bit may cause arbitrarily large changes to the integer values. The standard selection methods of Genetic Algorithms require positive fitness values, which are highest for the best population members. This requires some scaling transformation also be used when interpreting bitstring segments.

The bitstring representations are subject to a variety of processes to construct an analogue of genetic inheritance in sexual reproduction, mutation and selection.

The mutation operator, considered by Holland as a "background operator" changes single bits of the bitstring with a given mutation probability. In keeping with the natural model, the mutation probability is usually very small. Small mutation rates in nature guarantee that individuals do not differ greatly from their ancestors in a genetic sense. This does not hold true for encoded real-valued parameters, as a single changed bit can cause large parameter value changes. Mutation serves a useful role by its ability to reintroduce information lost from all members of the population, maintaining genetic diversity.

The crossover operator, emphasised by many as the most important operator in Genetic Algorithms (Bäck 1996), allows useful segments from different parents to be exchanged to yield offspring with the best characteristics of both. Traditionally, the bitstrings of two parents were severed at the same location and corresponding portions of the bitstring exchanged. Empirical studies have indicated this approach is inferior to a more recently developed method of multi-point crossover, that allows simultaneous exchange of several corresponding segments, eliminating positional bias under crossover.

Selection operates on the population using a probabilistic survival rule, survival probabilities being calculated according to relative fitness of individuals. Some sampled subset of the population is copied to the parent population of the next generation, the number of copies of individuals being proportional to the survival probabilities.

These processes of mutation, recombination and selection are repeated, typically until some limit on the number of iterations, or generations, is exceeded.

2.3.2.2 Evolutionary Strategies. These were a joint development of Bienert, Rechenberg and Schwefel in the 1960s (see, for example, Schwefel (1965)), and operate on continuous parameters, using normally distributed mutation and recombination. The earliest applications were experimental, applying small discrete changes sampled from binomial distributions to setups in fluid dynamics problems, performing a physical experiment, measuring the objective criterion, and retaining the new setup if it proved better than the last one used. Schwefel first simulated the method computationally, and this gave rise to what is termed the (1+1)-ES.

Rechenberg introduced the concept of a population by applying recombination of μ parents to produce a single "child". The child undergoes mutation and the whole population is subject to selection, the worst individual being removed. This formed the (μ+1)-ES. Schwefel continued the development of the method, introducing two new variants. In the ($\mu + \lambda$)-ES, μ parents generate λ offspring and the whole population of parents and offspring are subject to selection (with the obvious survival of μ parents of the next generation.) In the (μ,λ)-ES the best μ offspring are selected from λ generated individuals after mutation. This evidently implies $\lambda > \mu$.

An important characteristic of Evolutionary Strategies is that they readily allow the incorporation of important operational parameters, such as the parameters of the mutation probability distributions, into the population, so that optimisation is applied not only to the objective function but also to the strategy, so called *self-adaptation*.

Recombination as used in Evolutionary Strategies can take the same forms as used in Genetic Algorithms, but can also pair a single parent with several other randomly-chosen parents for each segment of a multi-point crossover operation. It may also be intermediate with child parameter values being formed from the (weighted) mean of several parent parameters.

Selection is completely deterministic, governed by the size of μ and λ in the two strategies described, ($\mu + \lambda$)-ES and (μ,λ)-ES.

($\mu + \lambda$)-ES always preserves the best solutions found at each stage, thus guaranteeing monotonic improvement of the population. By way of contrast, (μ,λ)-ES has possible advantages when the objective is time-varying or the objective function has many local minima, because population members adapted to outdated objectives or attracted by local minima may subsequently be discarded.

Termination was originally determined by comparing worst and best fitness values of the parent population, stopping if it was less than some predetermined threshold, either absolute or relative. This is often substituted by the simpler approach of a limit on the maximum iteration count.

2.3.2.3 Evolutionary Programming. This approach was developed by Fogel (1962)(see also Fogel, Owens and Walsh (1966)) and refers to that class of methods in evolutionary computation that apply a uniform random mutation to each member of a population, generating a single offspring. Originally applied to discrete parameters, the method was extended by D.B.Fogel to continuous parameter problems (Fogel 1991, Fogel 1992). However, unlike the other methods previously described, no recombination operators are applied. Population members may be considered as representative of species, rather than individuals, so phenotypic effects are emphasised instead of genetic change. After mutation, selection is applied to the combined population of parents and offspring, and half enter the next generation, i.e. the selection mechanism may be termed ($\mu + \mu$). Termination is usually triggered by exceeding a maximum iteration limit.

Evolutionary Programming algorithms offer a similar self-adaptation capability to Evolutionary Strategies, since the operational parameters controlling the mutation operator can be included in the genotype. Since these parameters undergo mutation at the same time as those defining the objective, the effect of change on them will be delayed.

Evolutionary Programming methods are generally particularly simple, robust and highly parallel.

3.0 Case studies

Where a simulation can be constructed that models the important physical properties of a real system it allows designers to explore a range of scenarios without the need to build a physical prototype. Computational science and engineering has been used extensively in the design processes of aeronautical and automotive industries, electronic and CAD and environmental modelling. Simulations are becoming increasingly sophisticated, complex and computationally challenging, since in order to achieve accurate modelling of a problem, it is important to use a high resolution in the mathematical decomposition.

An important mode of use for these models is in exploring some design space. The model is used as a "black box" to which questions are put such as "which set of input parameters will minimise the output of my model?" This function can be automated by developing optimisation programs that attempt to minimise an objective function value, which is computed either directly by the numerical model, or as a result of post processing the model output. Simple enumeration, i.e. running the model for all possible combinations of the input parameters and selecting the optimal set of parameters by inspecting all the returned model outputs, is not feasible for two reasons:

- The combined computational cost of computing all possible permutations of the parameters is prohibitively excessive. Each evaluation of the model may take several hours of computing time, or more, and the desired resolution of the solution may require thousands of evaluations.
- As the dimensionality of the problem, i.e. the number of independent parameters, *n*, increases, the number of required model evaluations increases as the power of *n* leading to a "combinatorial explosion" in the time to complete the experiment.

The computational techniques of the optimisation programs and the computing machinery used are becoming more complex. Increasingly sophisticated tests are required to evaluate the performance of algorithms, and their likely success tackling the problems of interest to engineers and scientists. Several collections of test problems have been developed and are periodically reviewed and maintained, older, simpler problems making way for more challenging problems to match the state-of-the-art in optimisation programs.

Many of these test problems, however, bear little resemblance to the complexity of the problems the programs are likely to face in the real world. Of more than 1000 test problems in the CUTE test set, only some 75 have actually been used in the solution of a real problem. While mathematical abstractions may serve to test specific aspects of algorithm capability, in many cases they are very poor predictors of the ability to find solutions to real-world problems. A comparison of model outputs from a real problem in Figure 3.13 and a "constructed" problem in Figure 3.17 graphically illustrates the extent of the difference in complexity. When a range of different optimisation methods are to be developed for use on real-world problems employing a range of computational methods, an adequate testing regime using simplified, abstract test cases becomes increasingly difficult to devise. For this reason a starting point for this research has been the development of a series of test problems drawn directly from real applications.

It is generally not practical to directly use complex, real-world problems as test cases. The "black box", when queried, can take a considerable amount of time and computational resource to provide a response. So parameter sweeps were made of a number of the numerical models that form the basis of these test cases, and the output data stored. These pre-computed data are interrogated, and linear interpolation employed to provide realistic responses from what are, in effect, "sandboxes" in which optimisation programs can readily be tested. These "sandboxes" themselves

represent a large investment of time and computational resource – for example, the data acquisition necessary to build the test case described in Section 3.5 required over 2 months of continuous computation on a multi-processor supercomputer.

In summary, the complex, accurate and realistic test cases described in the following sections were used to provide a test domain for optimisation algorithm development that was challenging but computationally tractable. Some of the characteristics of this domain are illustrated in Figure 3.1

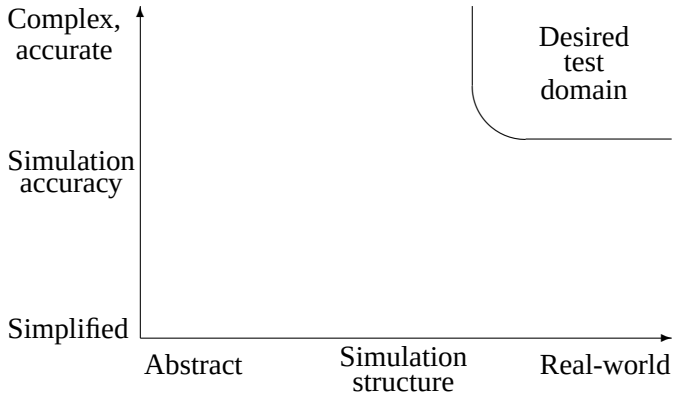


Figure 3.1 Intended characteristics of algorithm test regime

3.1 Photochemical Smog Model

This case study is based on the CIT model developed by McRae, Russell and Harley (1992), to compute the transport and production of photochemical smog within an urban airshed. In the original application it was being used to determine the sensitivity of oxidant photo-chemistry to various input parameters.

In the domain of air quality management, one of the major uses of photochemical airshed models is to compute oxidant concentrations. Oxidants, such as ozone, are generated as a result of the chemical interaction between various precursors such as oxides of nitrogen (NOx) and other reactive organic compounds (ROCs) in the presence of ultra-violet radiation. Ozone is of particular importance because of its health related side effects; ozone levels in Australian urban areas recently have been observed to exceed 0.12 ppm, which is a widely adopted health standard level. Results in this area are of considerable and immediate interest to our environmental research colleagues.

The particular case described in this work was based on detailed meteorological and pollution emission data for a particular scenario collected for the Australian city of Perth. The model behaviour is well understood, as it has served as the basis for earlier work including simple enumeration studies on parallel and distributed computing platforms (Abramson, Cope and McKenzie 1994). For the purposes of this work the objective was to determine the NOx and ROC concentrations, as input parameters, that would produce a minimal peak hourly average ozone concentration over a 24 hour period.

Figure 3.2 shows a sample ozone contour for the model region. It clearly shows the non-linear effect of varying ROCs and NOx on the ozone concentration. In some regions of the control space, increasing one of the precursors can increase the ozone, and in others it can decrease the ozone. Thus, a simplistic strategy of decreasing the precursors may not have the desired effect of decreasing

the oxidants. It is this effect that makes it essential to correctly model the process when evaluating control strategies.

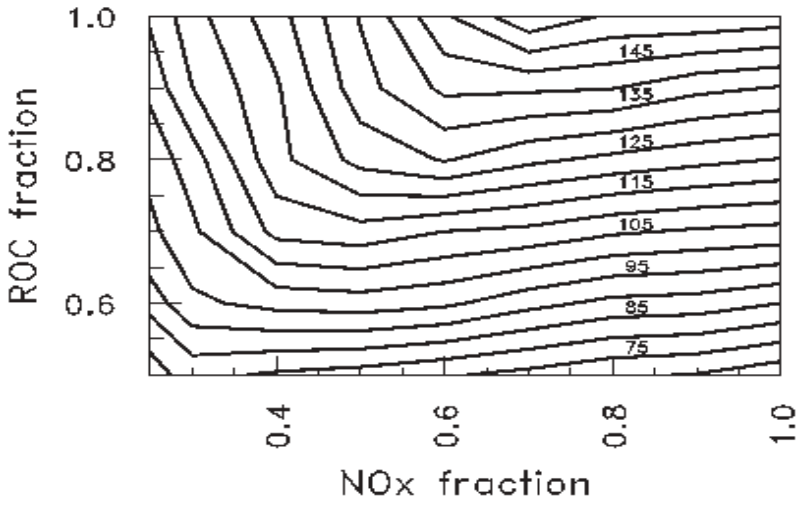


Figure 3.2 Typical surface: Ozone concentration (ppb) as a function of NOx and ROC

The parameter ranges are limited to a subset of the whole domain to exclude the trivial solution of reducing all precursor concentrations to zero. These limits were initially arbitrarily set to 50% in each dimension, but the range permissible for NOx was expanded down to 25% to allow the distinct non-linear feature seen in Figure 1 to extend across the whole domain. The desired solution accuracy is arbitrarily set to $\pm$5% in the input parameters, a reasonable estimate for the accuracy of the simulation.

3.2 Quantum Electrodynamical Problem: Laser

These test cases are derived from a simulation of laser-atom interactions (Hall 1998). A detailed understanding of the collision processes between atoms, electrons and ions is of great interest in the atomic physics community. This knowledge is important in the explanation of laboratory and astrophysical plasmas, spectroscopic and surface collision physics, and scattering dynamics. Applications include fluorescent lamp and gas laser technology, surface science and atmospheric physics (Anderson, Gallagher and Hertel 1988).

Of particular interest is the investigation of electron collisions with a short-lived laser excited target atom. One experimental method for exploration of the electron-excited atom collision process is the electron-superelastic scattering technique. An atom is optically prepared by a laser of known polarisation to an excited state and scattered electrons, which gain energy by collisionally de-exciting the atom, are detected.

This technique requires a detailed understanding of the laser-atom interaction as a function of laser intensity, laser polarisation and laser/atom detunings. It is possible using Quantum Electrodynamic (QED) theory, to generate equations of motion for atomic operator elements representing

atomic populations in the ground and excited state, optical coherences formed between the ground and excited state by the laser and excited state coherences formed by the laser. The QED model generates closed sets of coupled, first order, linear, homogeneous differential equations. These equations are solved using numeric integration, which can be time consuming.

Once the dynamics of the atomic operators are known, it is theoretically possible to predict the line polarisation (K) for linearly polarised excitation, as shown in Figure 3.3. It is how these parameters vary as a function of laser intensity and detuning that is of particular interest to physicists. Introducing integration over the Doppler profile of the atomic beam introduces another complexity, which further lengthens the computing time needed.

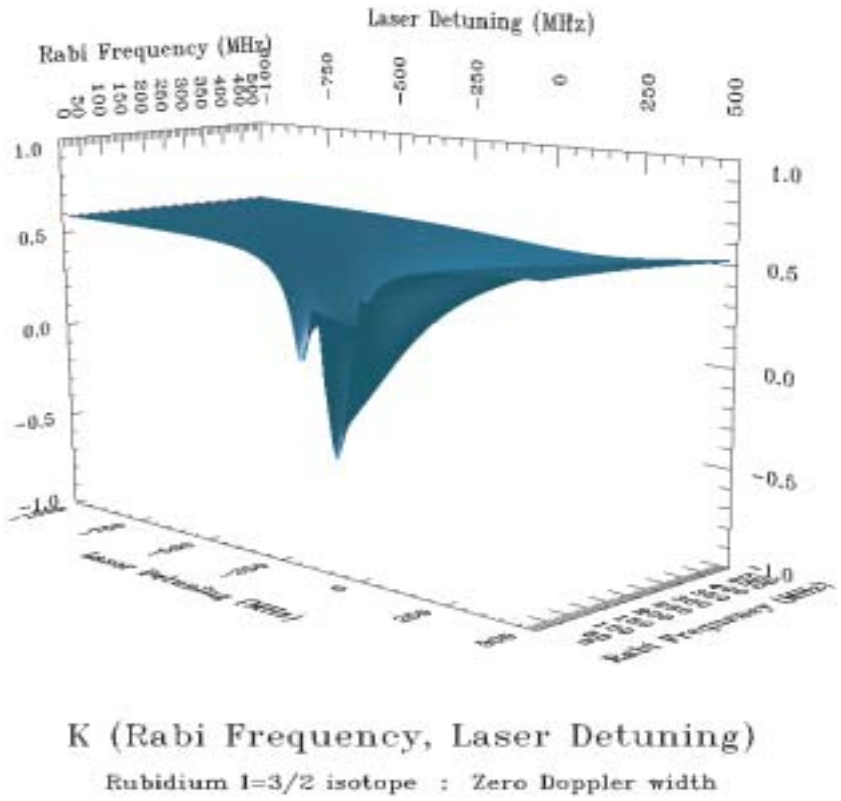


Figure 3.3 Line Polarisation (K) data from laser-atom interaction simulation

The original data were resampled on a 100x100 grid, producing the Laser 1 data set. This base case is quite a smooth surface. At the 100x100 sampling the dataset contains only 4 minima, of which the global minimum is quite dominant, as can be seen in Figure 3.4.

To this surface, additive fractal noise was applied to develop more challenging test surfaces. These surfaces test the ability of algorithms to optimise an objective function f that is a perturbation of a smooth function, f_s by a small function ϕ, i.e.:

$$f(x) = f_s(x) + \phi(x)$$

The perturbation ϕ can be random, based on the results of an experiment, or not even a function, returning different results from subsequent calls with the same arguments.

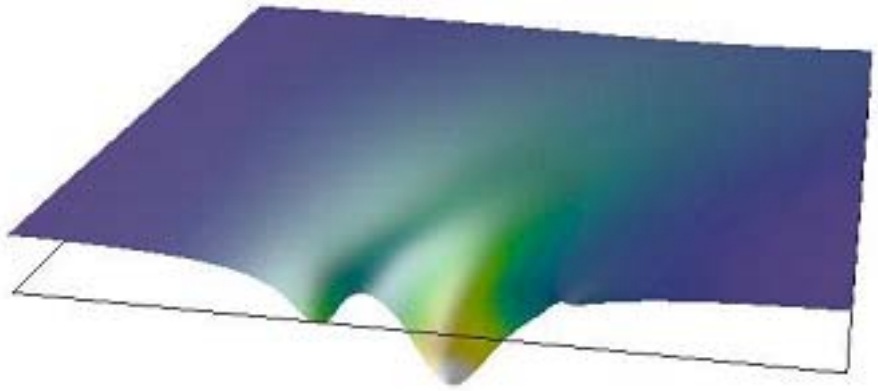


Figure 3.4 Laser 1 test case data

The Laser 2 test case data shown in Figure 3.5 had moderate amounts of noise present and, in contrast to the Laser 1 test case, at the same resolution the Laser 2 data set contains 1157 local minima of varying severity. Other, intermediate cases were generated with more severe interference, culminating in Brownian noise, shown in Figure 3.6. The fractal noise was generated by a process of midpoint displacement (Saupe 1988), with an initial amplitude scaling of 10% of the range of the underlying data. The global minimum for each data set was known from visual inspection of the data.

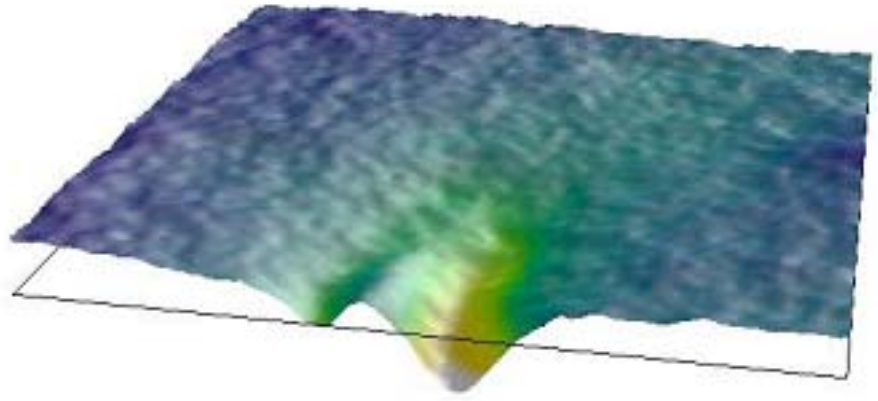


Figure 3.5 Laser 2 test case data

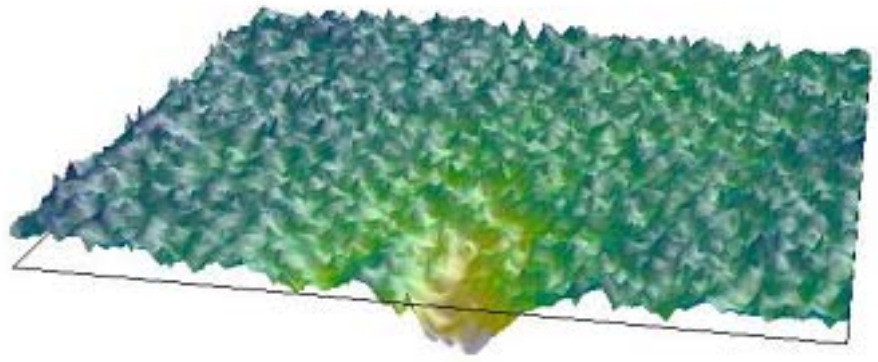


Figure 3.6 Laser test case data with additive Brownian noise

3.3 Design of durable parts

3.3.1 Design of durable parts: minimising stress

In the design of load-bearing structures or mechanical components, competing objectives of minimum weight and maximum strength provide a difficult design challenge. The failure of such components is usually due to slow growth of a pre-existing crack followed by a sudden fracture (Chaperon, Sawyer, Jones and Rose 1999). A common mechanism for crack growth is fatigue due to cyclical loading. As a load is applied the high stress at the crack tip causes plastic deformation that produces an irreversible growth in the crack. This growth is small (typically 10-7 - 10-3 mm) but repeated cycles of loading may extend the crack to a stage where fracture occurs.

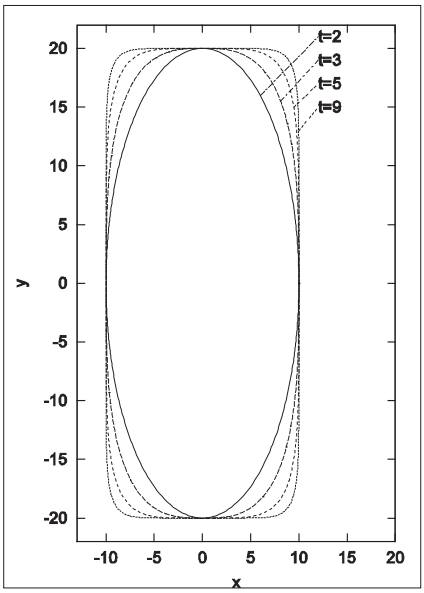
This case study used finite element analysis of a thin plate under cyclic loading that contained a hole of given dimensions and shape, specified by the parameters (Peachey, Abramson, Lewis and Jones 2003).

The dimensions of the hole were given:

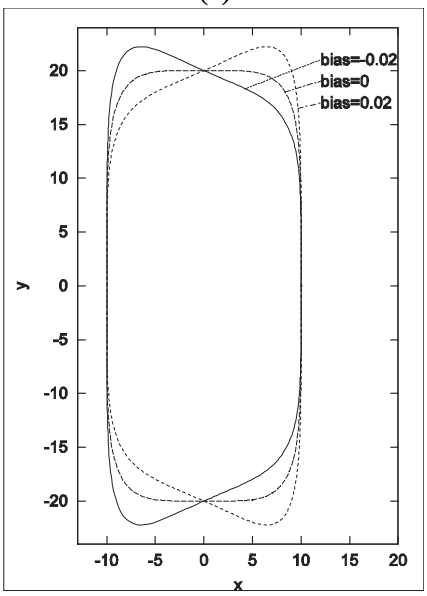
$$\frac{|x-p|^t}{a^t} + \frac{|y-q|^t}{b^t} = 1. \tag{3-1}$$

These are closed curves of width $2a$ height $2b$ where t controls the curvature at the shoulders of the curves. Figure 3.7 shows some of these with $a = 10$, $b = 20$, $p = 0$, $q = 0$ and a variety of values of t. More generally, the curves can be defined by:

$$\frac{|x-p|^t}{a^t} + \frac{|y-q|^t e^{-t\beta(x-p)}}{b^t} = 1 \tag{3-2}$$



(a)



(b)

Figure 3.7 Parametrized curves from (a) Equation 3-1, (b) Equation 3-2

The inclusion of the exponential factor allows for asymmetry about that axis. The parameter β, called the "bias" here, controls the slope at the top. Figure 3.7(b) shows samples with $p = 0$, $q = 0$, $a = 10$, $b = 20$, $t = 5$ and various β.

For the crack model $a = 10$, $p = 20$ and $q = 0$ giving a hole of width 20mm which is 10mm from the boundary. Thus the variables b, t and β are the optimization parameters. The search space used was the domain $5 \leq b \leq 35, \quad 2 \leq t \leq 9, \quad -0.02 \leq \beta \leq 0.02$.

First finite element techniques were used to compute the stress field throughout the component for a given applied load in the absence of cracks. Then cracks are assumed to occur at critical boundaries of the component and a recent modification (Nishioka and Atluri 1983) of the finite element alternating method (Mattheck and Burkhardt 1990) is applied to compute the stress intensity factor at the crack positions. In damage tolerant design common practice has been to minimise the maximum stress under load. Isosurfaces of these stress values are shown in Figure 3.8[1]. These surfaces were reasonably smooth, and only 26 local minima were revealed by a parameter sweep at a resolution of 3%. This dataset became the "Crack 1" case study.

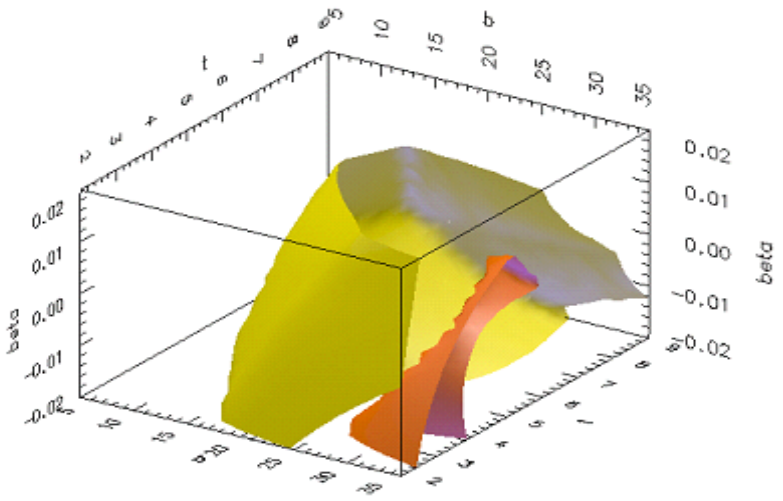


Figure 3.8 Isosurfaces of maximum stress: Crack 1

3.3.2 Design of durable parts: maximising fatigue life

The use of durability as the primary criterion in damage tolerant design is a new development. The crack test case included fatigue cracks at a number of locations, and the objective of this test case was to **maximise** the life of the part as determined by fatigue crack growth to a defined length. Continuing from computation of the stress-intensity factor at the crack locations in the Crack 1 test case a growth law is used to compute the number of loading cycles required for the cracks to grow from its given initial size to a given final size. The number of cycles required at the worst (least cycles) crack is taken as the fatigue life of the designed part.

Isosurfaces at a number of values are shown in Figure 3.9. The dataset was "noisy", and a parameter sweep at a resolution of 3% revealed 540 local maxima. This dataset became the "Crack 2" case study. The data collected from the parameter sweeps of both Crack test cases were each

[1]Throughout the text the value of the objective function is illustrated in a number of places using 3-dimensional isosurfaces. These usually appear as a series of roughly concentric "shells", the smallest, inner shell enclosing the global minimum. Features to observe are the location of this minimum, any discontinuity of similarly coloured isosurfaces indicating significant regions of local minima, and the degree of "smoothness" of the surfaces. Irregularities indicate the presence of divergent local gradients, impeding movement of gradient-based algorithms in directions more or less tangential to the surface.

compiled into a 31x36x21 data set (i.e. 23436 objective function evaluations) for subsequent investigations.

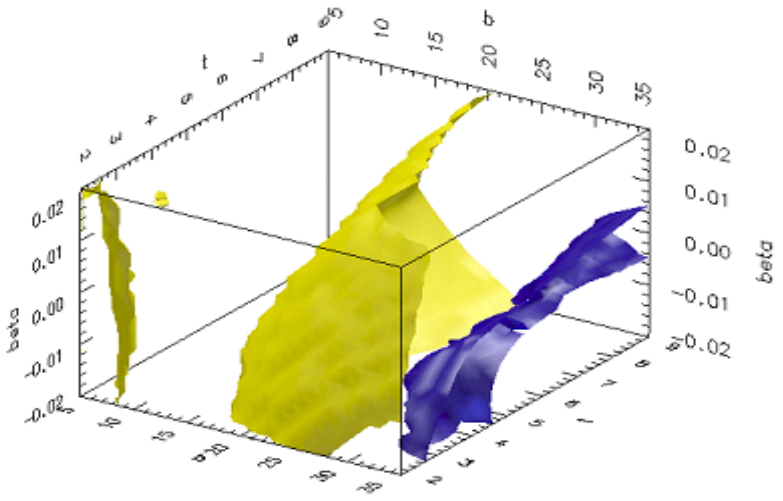


Figure 3.9 Isosurfaces of the design durability: Crack 2

3.4 Design of a two-dimensional aerofoil

The use of computational modelling in design processes is well established in the aerodynamics industry, and the use of optimisation techniques is becoming more widespread. This test case models the aerodynamic properties of an aerofoil cross-section, the objective function to be minimised being the lift-drag ratio (Abramson et al. 2001a). Specifically, a simple two-dimensional aerofoil was modelled using a FLUENT simulation (www.fluent.com). The aerofoil mesh generated by GAMBIT is shown in Figure 3.10. It had 28089 nodes and 49426 elements, made up of 43090 triangular elements and 6336 quad elements. The shape of the simulation mesh was generated from the problem input parameters, and FLUENT was used to compute the flowfield around the aerofoil, from which lift and drag properties were derived. Parameters were the aerofoil angle of attack, its thickness and camber. The angle of attack is the angle with the horizontal formed by a line through the leading and trailing extremities of the aerofoil, the thickness is the greatest width measured normal to the centreline between these extremities, and the camber is the maximum distance the centreline curves away from a straight line between the extremities. The convergence criteria for the FLUENT run were set at 1.0×10^{-4}.

Figure 3.11 shows a number of isosurfaces of the lift-drag ratio in the parameter space investigated. The global minimum, corresponding to low drag and high lift, is located near the bottom, front corner of the domain in the Figure, with a small angle of attack, large camber and small thickness. The isosurfaces for increasing values of the lift-drag ratio can be seen as a series of concentric, largely featureless shells around the global minimum. The dataset was generally “smooth”, and when a parameter sweep was performed at a resolution of 10% only 12 local minima and a dominant global minimum were discovered. The data collected from the parameter sweep were compiled into a 1000-point data set for subsequent investigations.

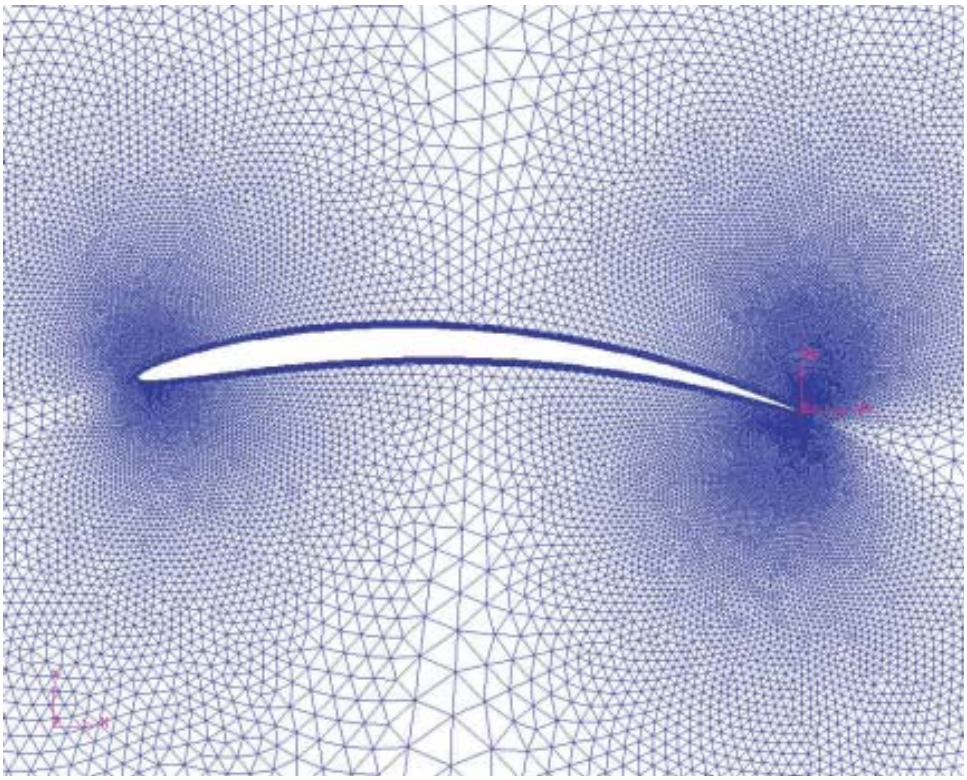


Figure 3.10 Computational mesh for aerofoil simulation

3.5 Radio Frequency Design Problem: Bead

Testing of mobile telecommunication handsets is an important aspect in the design and characterisation of handset performance. Typical performance parameters include input impedance, radiation patterns, gain and efficiency.

The structure of most mobile telephone antennas includes the chassis as an earth reference. This effectively forms an asymmetric dipole structure with the physical characteristics of the chassis (size, shape and orientation) playing a significant role in determining the radiation characteristics of the antenna.

Due to the chassis forming an integral part of the radiating structure, several difficulties are encountered in the measurement of mobile telephone antennas since antennas are generally connected to a network analyser via a coaxial cable. It has been shown that the coaxial cable used to feed a handset has significant effects on both the radiation pattern and resonant frequency (Saario, Thiel, Lu and O'Keefe 1997).

This case study considers the application of high permittivity lossless ceramic suppression beads for minimising the cable effect on measurements (Saario, Thiel and Lu 1999). The bead is optimised for minimum transmission, characterized by the transmission loss, S_{21}. In order to determine the effectiveness of the beads used in the handset characterisation, a Finite Difference Time Domain (FDTD) method was used for numerical analysis of the handset-cable structure. A frequency-domain near-to-far field transformation was used to obtain the far-field radiation patterns. A 4-layer

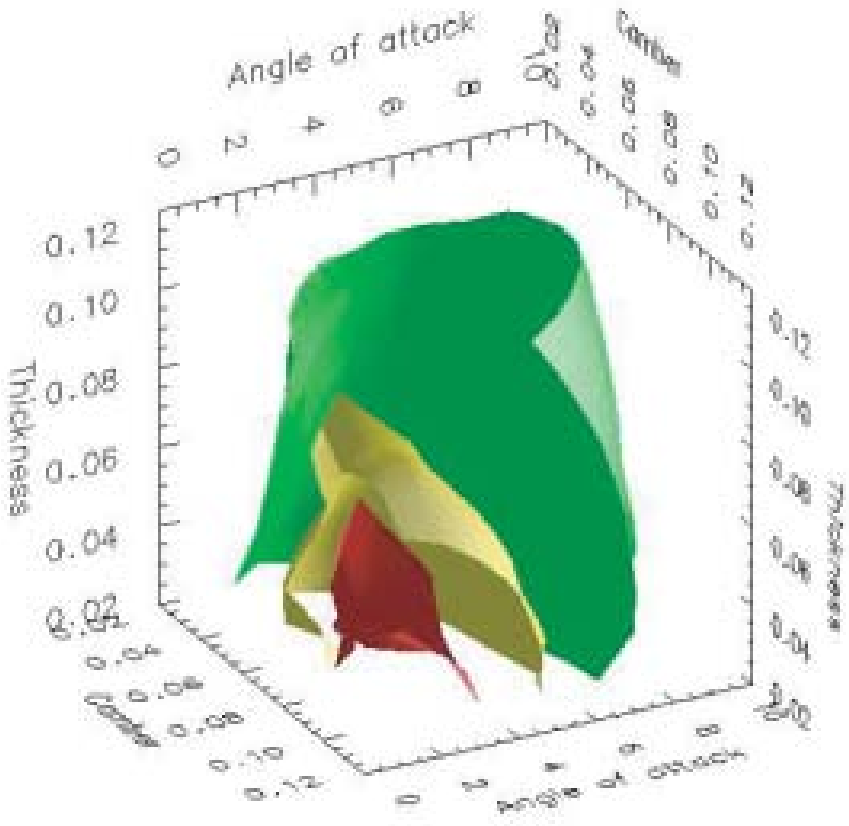


Figure 3.11 Isosurfaces of the aerofoil lift-drag ratio

Perfectly Matched Layer (PML) Absorbing Boundary Condition (ABC) was used for termination of the solution space.

The geometry of the simulation model is shown in Figure 3.12. An infinitely long coaxial cable was modeled by allowing the outer metal conductor of the coaxial cable to penetrate the PML. The PML is effectively a perfect match for the impedance of the coaxial cable since no reflection occurs. The suppression bead was placed centrally on the coaxial cable. Parameters varied were the permittivity, thickness and length of the bead.

A value of -50dB for S_{21} is more than adequate for suppression, so the cost function to drive the optimization is defined as: $C(p_k) = 50 + Avg(S_{21})$ where $Avg(S_{21})$ is the average S_{21} over the 990-1010MHz band, the frequency range of interest.

To obtain a qualitative insight into the nature of the parameter space, the cost function was evaluated at integer intervals in the parameter ranges:

$$35 \leq \varepsilon_r \leq 60$$

$2mm \leq thickness \leq 17mm$ corresponding to diameter, D, $15mm \leq D \leq 90mm$

$$37.5mm \leq length \leq 112.5mm$$

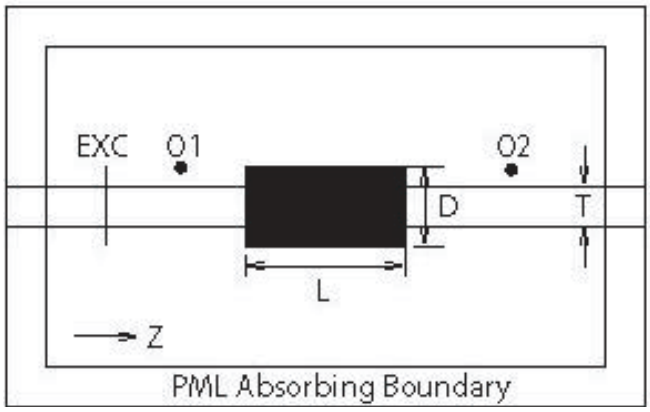


Figure 3.12 Geometry for simulation of a suppression bead on an infinitely long coaxial cable

Figure 3.13 shows isosurfaces in parameter space of the transmission loss, S_{21}, at -35db. While S_{21} at this level is inadequate for the suppression required in the intended application, these isosurfaces can give some guidance as to the structure of the model response in parameter space and location of significant local and global minima. The global minimum, in fact, is located inside the tubular structure evident in the upper right of the Figure.

Figures 3.14, 3.15 and 3.16 show contours of S_{21} on three "cutting planes" through parameter space, each passing through the global minimum, and holding bead dielectric permittivity, bead thickness and length constant, respectively.

Figure 3.15, for constant bead thickness, corresponds to an horizontal plane through Figure 3.13 and clearly shows a slice through the tubular structures. Figures 3.14 and 3.16 show planes cutting through these structures more or less obliquely. Considered together, all the Figures give a picture of considerable complexity and structure. Direct inspection of the sampled data indicates there are approximately 298 local minima. In summary, this is an extremely challenging optimisation problem.

The sampled data set was retained as the Bead test case.

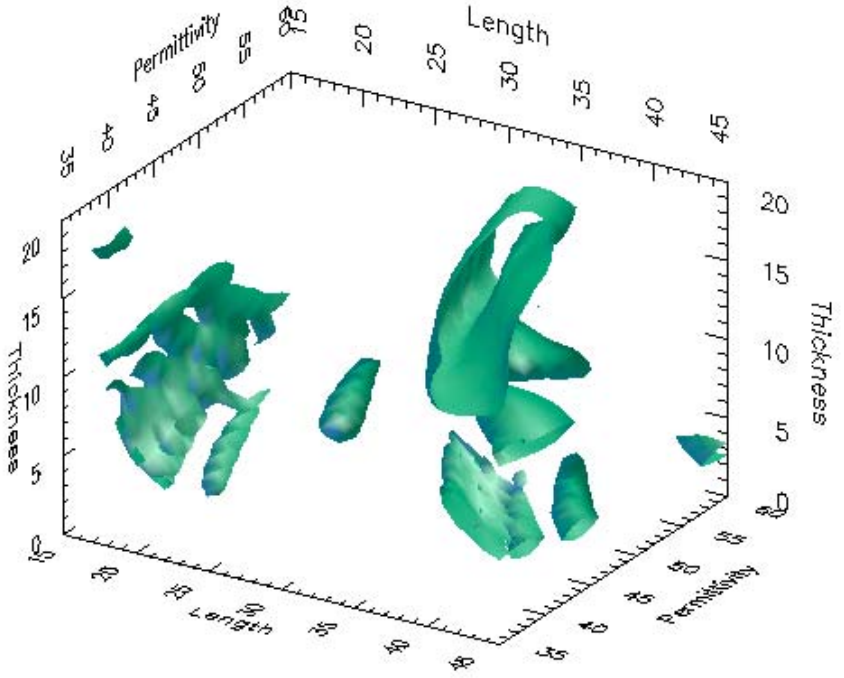


Figure 3.13 Isosurfaces for transmission loss, S_{21} = -35db

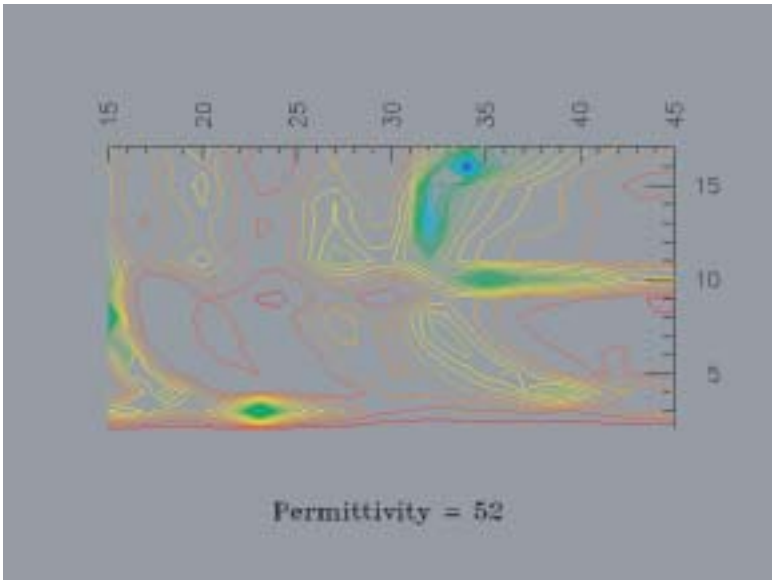


Figure 3.14 Contours of S_{21} on a plane of constant bead permittivity through the global minimum

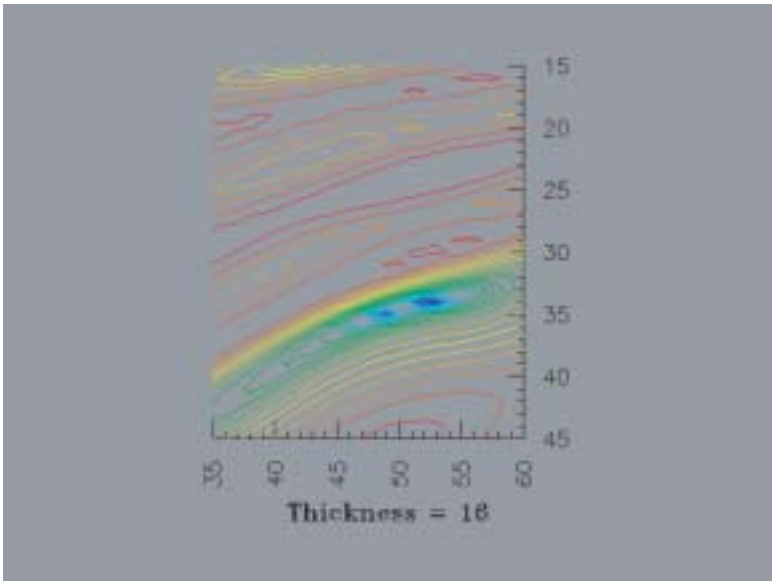


Figure 3.15 Contours of S_{21} on a plane of constant bead thickness through the global minimum

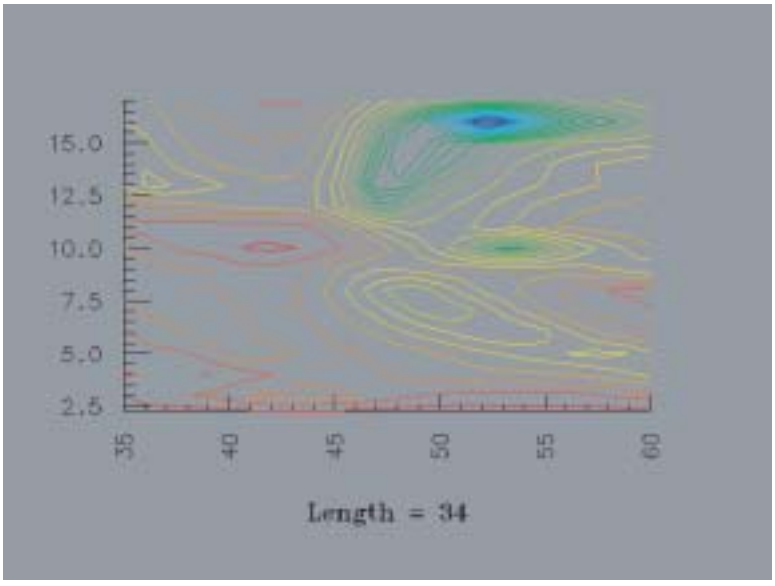


Figure 3.16 Contours of S_{21} on a plane of constant bead length through the global minimum

3.6 Benchmark: Rosenbrock's function

In order to provide a point of comparison with the published literature, the well-known Rosenbrock's function in two dimensions was included. The objective function values for this test case were directly computed from $f(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$ for $x_i \in [-2, 2]$ which has one local minimum at $f(1, 1) = 0$. A contour surface of the function is shown in Figure 3.17.

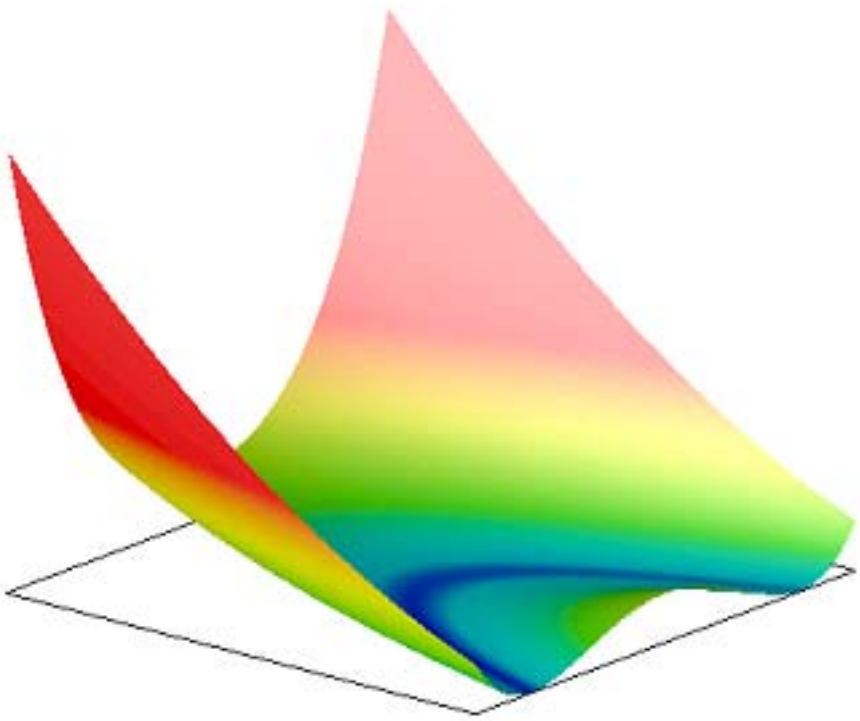


Figure 3.17 Contour surface of Rosenbrock's function in 2D

3.7 Summary

A number of real computational models drawn from problems in a range of physical sciences have been used to test the algorithms investigated in this research. The following compiled datasets derived from parameter sweeps of the respective computational models were retained and used within the testing framework as "sandbox" models for algorithm testing:

- The two Quantum Electro-dynamics cases: Laser 1 and Laser 2

- The two durable component design cases: Crack 1 and Crack 2
- The 2D aerofoil design case: Aerofoil
- The radio-frequency design case: Bead

Linear interpolation was used to provide intermediate results, simulating continuous functions, except for the "Bead" test case. For this case study, unless otherwise noted, real number parameters were truncated to integers internal to the test case, and the corresponding data value returned. The calling optimisation algorithms continued to interpret the parameters as continuous variables, seeing the objective function as having discontinuous first derivatives. This duplicated the treatment of parameters by the original computational model.

The data acquisition in compiling these "sandbox" models represents a considerable investment in time and effort, the return being it allowed algorithms to be tested on real-world problems while minimising the demands of running multiple algorithms on multiple test cases from multiple starting conditions for numerical models with large computational requirements. However, it is this considerable effort – months of computing in some cases – that precludes the use of this approach for practical optimisation of new problems.

3.7.1 Problem characterisation

Generally, the case studies fell into 2 sets:

- Smooth, with a dominant global minimum (Laser 1, Aerofoil, Crack 1, Rosenbrock's function)
- Multiple/many local minima, non-convex (Bead, Laser 2, Crack 2)

In formulating ideas about the most appropriate algorithm for use with a particular problem, a great deal may depend on whether the problem encountered is "noisy" or "smooth". Considerable effort has been expended in the development of methods specifically to address "noisy" problems (Elster and Neumaier 1995, Elster and Neumaier 1997, Anderson and Ferris 2001, Carter, Gablonsky, Patrick, Kelley, Eslinger 2001). The question remains:

> For a new, relatively unknown problem, how may its degree of "smoothness" be determined so that an appropriate choice of algorithm can be made?

One possible approach is to perform some initial sampling of the objective function across the parameter space, as is done during Response Surface Methodology (RSM) optimisation (Khuri and Cornell 1987, Box and Draper 1987). This may reveal the approximate nature of the objective function but, as has been acknowledged for RSM itself (Haftka, Vitali and Sankar 1999, Giunta, Narducci, Burgee, Grossman, Mason, Watson and Haftka 1995, Scotti, Malik, Cheung and Nelder 2000), falls victim to the "curse of dimensionality", i.e. as the dimensionality of the parameter space increases, the number of function evaluations required to adequately characterize the objective function increases greatly. For example, *central composite design (CCD)*, a popular experimental design used with RSM, samples at all the vertices of the search space, the face centers, and the centre of the domain, requiring $2^n + 2n + 1$ analysis points. For a problem of 20 parameters, CCD will require over a million function evaluations. If sufficient points are not sampled, the RSM predictive phase must resort to extrapolation, possibly leading to large errors.

For the task of problem classification, however, it may be sufficient to take a drastically reduced sampling of the objective function, since only a qualitative understanding of the general nature of

the problem is necessary, not a surrogate for the objective function. It is proposed that a locus of points be generated "diagonally" through the parameter space, i.e. starting from a point at the lower bound in all dimensions, for each successive point of the locus all parameter values are uniformly increased, and the objective function then simultaneously sampled at all these points. This procedure was carried out for a 20-point locus for each of the real-world test cases described in preceding sections, and the graphs of returned values against displacement along the loci are shown in Figure 3.18. Comparison of the "smooth" problems (on the left) shows they all have a generally smooth, convex graph, in marked contrast to the "noisy" problems on the right.

These empirical results suggest that a simple, qualitative test may be sufficient to classify problems into "noisy" or "smooth" types. Further, it may be possible to construct an automatic procedure, given the derivation of suitable criteria (for example, number of times the sign of the first derivative of the function on the locus changes.) Such a procedure could be applied at minimal computational cost if sufficient parallel computing resources are available for all necessary function evaluations to be completed concurrently.

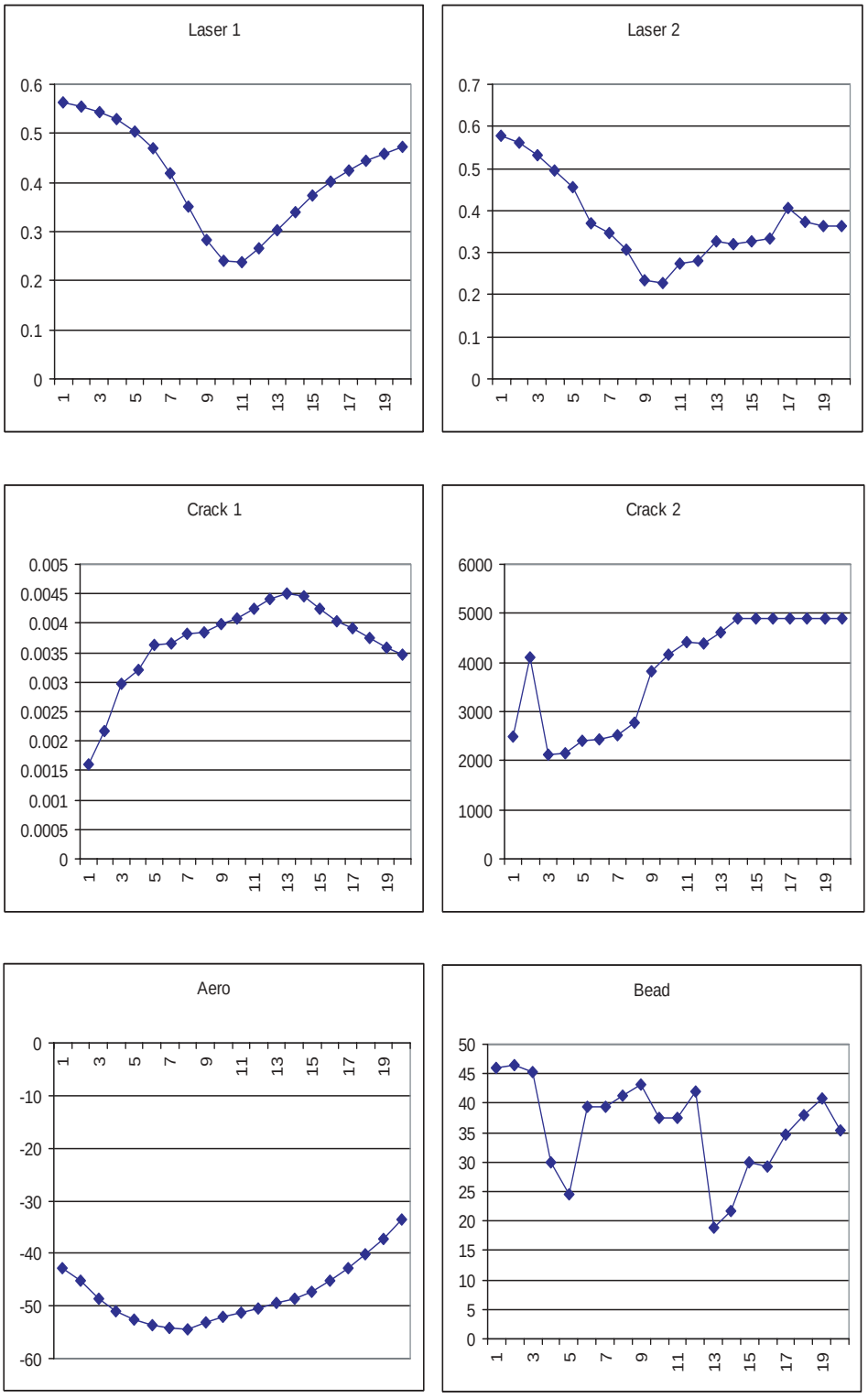


Figure 3.18 Objective function values along sample loci

4.0 Gradient Methods

Gradient methods are among the best established of optimisation methods. They generally have very good convergence properties, particularly on quadratic functions, and some possibility of parallelization of operations within the algorithms. Well understood and widely used, they are eminently suitable to provide a baseline for future comparison, and are indispensable in any complete optimisation toolset. For these reasons they were chosen as the first class of methods to investigate.

As Hough and Meza (2002) clearly point out, for gradient descent methods it is possible to parallelise:

- the function, gradient and constraint evaluations,
- the linear algebra of the method, or
- the optimisation algorithm at a higher level.

As they concluded, often there is no access to the source code for the objective function, precluding the first alternative. For many of the problems of interest the dimensionality is usually small, as was confirmed in Chapter 3, and so parallelisation of the linear algebra can be expected to yield little benefit. In addition, the computation required to derive the objective function values generally rapidly overwhelms the computational needs of the linear algebra. The remaining option is to parallelise the optimisation algorithm at a higher level of abstraction.

As outlined in Section 2.1, there are two major families of gradient descent algorithms: conjugate gradient methods and quasi-Newton methods. For this work, the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm was chosen from the family of quasi-Newton methods. The BFGS algorithm is widely regarded as one of the most efficient and robust gradient descent methods for use with continuous functions (Gill et al. 1981, Fletcher 1987, Press et al. 1992).

For continuous functions, gradient descent methods use the derivative of the cost function, as well as its value, to select a search direction, essentially reducing the multivariate optimization problem to univariate minimization along a search vector; a line search. They thus consist of two main operations which are executed repeatedly, gradient calculations and line searching. When the cost function is the computed result of a numerical simulation, as proposed in this work, finite difference approximations to the derivative must usually be employed.

4.1 A Parallel Gradient Descent Algorithm

The gradient information gathering phase of the algorithm is easily parallelized by concurrent execution of function evaluations on the finite difference stencil, and this rapidly became common practice (van Laarhoven 1985, Schnabel 1987, Byrd, Schnabel and Schulz 1988, Navon, Phua and Ramamurthy 1988, Freeman and Phillips 1992). Parallel computation of finite differences is so obvious it has become an exemplar in the development of parallel algorithms (Foster 1994). The degree of parallelism achievable is limited, however, by the dimensionality of the problem domain. At most, when using central difference approximations, computing the gradient requires $2n + 1$ function evaluations for an *n*-dimensional problem. If forward differences are employed, with their lower accuracy, computing the gradient requires only $n + 1$ function evaluations.

The approach of parallelisation of finite difference gradient approximations has been adopted by many practitioners as a first, and only, recourse. Some even consider parallelisation of line searches

as impossible (see, for example, Koh, Reinbolt, Fregly, and George (2004)). Much of the contemporary research is in the context of unconstrained optimisation, which makes parallelisation of the line search difficult. Several researchers concentrate instead on methods of guaranteeing sufficient decrease in the line search (Moré and Thuente 1994). Even when developing methods for constrained optimisation, some continue to emphasise this aspect of line searching (Byrd, Lu, Nocedal and Zhu 1995). A very few have considered parallel line searching, from speculation (Navon et al. 1988) to practical implementation (van Laarhoven 1985, Lewis, Abramson and Simpson 1997, Phua, Fan and Zeng 1998) and analysis (Peachey et al. 2001). Van Laarhoven (1985) applied parallel gradient evaluation and parallel line searching to early variable metric methods, but claimed parallel execution would give no improvement for the BFGS-method.

4.1.1 Parallel line search

For unconstrained optimization the line search phase is often inherently sequential, but for the class of problems under consideration there will often be restrictions on the expected range of variables determined by physical constraints on their possible values. Equally, in the context of real design problems secondary factors, such as prohibitive retooling costs in manufacture, may serve to limit the range of values of interest, even if they are strictly, physically feasible. The restricted expected range of the variables allows the formulation of these problems as nonlinear optimization problems with **simple bounds**, and so the line search phase of the algorithm may be parallelized by using a method of interval subdivision, rather than the sequential stepping methods typically used. It should be noted, however, that applying simple bounds has the potential to create additional local minima where the objective function intersects the boundaries, as has been alluded to in Elster and Neumaier (1995). The resulting parallel line search is similar to the method of Avriel and Wilde (1966).

Because of the use of an iterative subdivision line search, the parallel algorithm is not trapped in some local minima, unlike the sequential algorithm. This is illustrated in Figure 4.1 which shows the sequential algorithm using a combination of stepping and parabolic interpolation, and the parallel algorithm using interval sub-division. The sequential algorithm terminates in the first minimum found, while the parallel algorithm, extending its search to the boundary, finds the lower minimum.

Two other, quite novel approaches have been proposed for parallelisation of gradient descent methods. Phua et al. (1998) used a method of multiple search directions by simultaneously applying several different update formulae, the BFGS update being just one. This can be considered a form of *speculative* computation, in that it was intended to allow the choice of whichever update was better suited to a particular problem or circumstance. The benefit to the solution time derived from the potential reduction in iterations required to converge. Zavriev and Perunova (2003) used multiple processors to compute finite difference approximations to the gradient with several different spatial discretization constants (yielding finite difference stencils of different sizes) to “smooth” local gradients and avoid small local minima. The next step, of concurrent function evaluation, was overlooked. This method aimed for potential gains in solution quality, rather than acceleration of the algorithm.

4.1.2 Parallel BFGS – Version 1

A parallel BFGS algorithm has been implemented. It uses concurrent evaluation of finite difference approximations to the gradient, but does not extend this to perform speculative gradient calculation for each point in the line search as being too computationally expensive for limited gain. The finite difference approximations are central difference formulations where possible, for

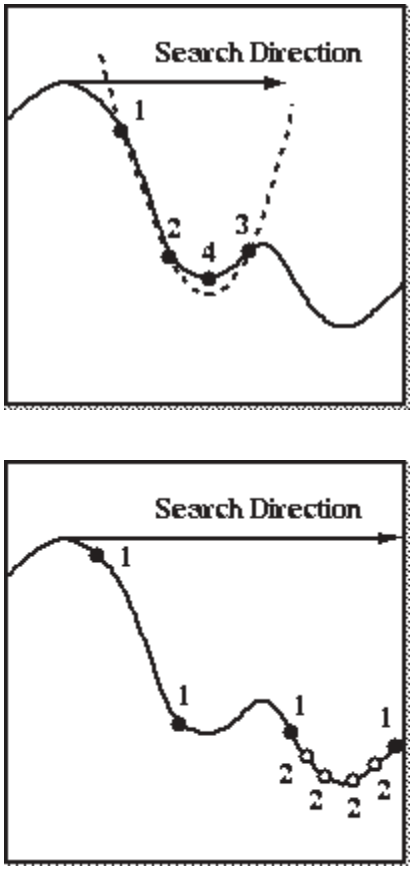


Figure 4.1 Comparison of sequential and parallel line minimization

accuracy, reverting to forward (or backward) differencing when less than the finite difference interval from the domain boundaries. The finite difference interval was set to the desired tolerance multiplied by the range of the parameter in each dimension, for simplicity.

Convergence was determined by iterative change in objective function value and position in parameter space both being proportionally less than the desired tolerance. If the solution space is considered as the (n+1)-dimensional union of the n-dimensional parameter space and the objective function value, this condition constrains the solution within a hyper-sphere with a diameter of the desired tolerance, relative to the parameter range in each dimension and the current value of the objective function.

In addition, the gradient of the objective function at the final point was constrained to be less than an empirically-determined constant, unless the solution point was on or near one or more domain boundaries and on a slope normal to the boundary.

The number of sub-intervals sampled in the parallel line search was set by the user, with a default of 6 to match the 2n finite-difference stencil for a 3-dimensional problem, the largest used in the set of case studies.

For comparison, a sequential BFGS algorithm was also implemented. This algorithm uses a combination of bracketing the minimum by golden section search followed by isolation of the minimum using Brent's method (Brent 1973), instead of the iterative interval subdivision method for

line searching used in the parallel algorithm. Function evaluations for determing the gradient were also performed sequentially. Otherwise the two algorithms were identical.

A brief outline of the first version of the parallel algorithm is given below.

Initialization

- Evaluate the objective function at the starting point
- Calculate the gradient at the starting point by finite difference approximation (parallel function evaluation)
- Set the inverse Hessian to the unit matrix
- Set the initial line direction to the inverse of the gradient

Perform line minimization

- Truncate the line search vector for active constraints
- Determine the nearest boundary in the search direction and set maximum excursion accordingly
- Sub-divide the interval as desired (usually an integer multiple of processors available)
- Evaluate the objective function at the sub-intervals (parallel function evaluation)
- Select a bracket of three points containing the minimum value

  If (bracket width is less than the desired solution tolerance) **then**

  - Line minimization complete
  - **else** Sub-divide the bracket and repeat the evaluation

Test for convergence

If (step change in function value and largest step change in position in any dimension are less than the desired tolerance) **then**

- Calculate the gradient at the “minimum”

If (gradient is less than an empirically determined constant) **then**

- optimization is complete

else if (the “minimum” is at a boundary normal to the gradient) **then**

- optimization is complete

else if (change in function value over previous two steps is less than desired tolerance) **then**

- optimization is complete

else

- Reset the inverse Hessian to the unit matrix, reset the line search direction to the inverse of the gradient and
- **repeat line minimization**

Perform BFGS update

- Calculate a new gradient at the line minimum
- Calculate the step change in the gradient
- Apply the BFGS update to the inverse Hessian
- Calculate the new line search direction and **repeat line minimization**

The version of the parallel algorithm described above has the disadvantage that convergence is governed by the absolute magnitude of the gradient at the final point, x^*. This requires some *a priori* knowledge of the nature of the objective function, and intelligent tuning of the convergence criteria accordingly. If the limit on the magnitude of the gradient for convergence is set too large the algorithm may be prone to premature termination, too small and excessive iterations may ensue. It is unsafe to omit *some* test of the stationarity of x^*, relying only on examination of the differences in successive values of the objective function.

4.1.3 Parallel BFGS – Version 2

In order to construct a robust algorithm simple to use in the context of the tool framework outlined in Section 1.2, a second version of the parallel algorithm was implemented that substituted a test for sufficient decrease in the norm of the gradient, relative to the norm of the gradient at the starting point (Kelley 1999a). For a well-conditioned $\nabla^2 f(x^*)$, a small gradient norm implies a small error norm. "Sufficient decrease" was defined as being a ratio equal to the desired solution tolerance.

These modifications required the interpolation of the following steps in the algorithm outline:

Initialization

- Evaluate the objective function at the starting point
- Calculate the gradient at the starting point by finite difference approximation (parallel function evaluation)
- Calculate the Euclidean norm of the gradient
- Set the inverse Hessian to the unit matrix

. . . (algorithm continues)

Test for convergence

If (step change in function value and largest step change in position in any dimension are less than the desired tolerance) **then**

{Calculate the gradient at the "minimum" and the Euclidean norm of the gradient

If (gradient norm is less than desired tolerance) **then** optimization is complete

else

. . . (algorithm continues)

4.2 Results of Numerical Experiments.

The second version of the parallel implementation of the BFGS algorithm, P-BFGS, was compared with the sequential implementation on the following test cases:

- The photochemical smog model
- The quantum electrodynamical models: Laser 1, 2, and intermediate cases.
- The radio frequency design problem: Bead

For the photochemical smog model, since the computational model was used directly only a single run was performed for each algorithm. For the Laser case studies, results were compiled from 100 runs for each problem, for each algorithm, from uniformly distributed starting points.

4.2.1 Results for photochemical smog model

The photochemical smog model was tested using the first version of P-BFGS, and a tolerance on the gradient of 1.0×10^{-6}. The convergence tolerance on objective function value was $\pm 5\%$, a reasonable estimate for the accuracy of the simulation.

Using P-BFGS, a solution set of input concentrations of NOx and ROC that minimized generated ozone was found at 100% NOx and 50% ROC. Inspection of Figure 3.2 confirms this to be the global minimum for the search domain. This result was achieved in under 14 hours of wall-clock compute time (ignoring queue wait time). This represents 46 evaluations of the cost function, using a simplified chemistry (Azzi, Johnson and Cope 1992) in the airshed model to reduce the computational load in these exploratory tests. Each run of the airshed model took about 1 hour 20 minutes of CPU time. A comparable solution by simple enumeration would have required an estimated 140 hours of computing.

Using a single CPU the sequential algorithm required 51 hours of wall-clock compute time, performing 39 evaluations of the cost function. The prototype parallel code thus achieved a speedup of 3.8 relative to the sequential algorithm. Using an average 4.6 CPUs this represents a parallel efficiency of 83%.

Unlike the parallel algorithm, the sequential algorithm terminated in a local minimum (79% NOx, 50% ROC) which was close to the global minimum actually discovered by the parallel algorithm. The method of line search adopted in the parallel algorithm has been found to more reliably find global minima in a number of test cases.

4.2.2 Results for Laser case studies

The Laser case studies used the later version of P-BFGS. The ability of the parallel algorithm to search beyond intervening local minima, as illustrated in Figure 4.1, suggests it may be more tolerant of noise in the returned values of the objective function. The additional Laser test cases with a range of fractal noise levels imposed were used to test this hypothesis. The global minimum for each Laser data set was known from visual inspection of the data. A search was deemed to have terminated successfully if the fractional difference between final objective function value and the value at the global minimum relative to the range of the data set was less than twice the desired tolerance, which was set at 1.0×10^{-2}.

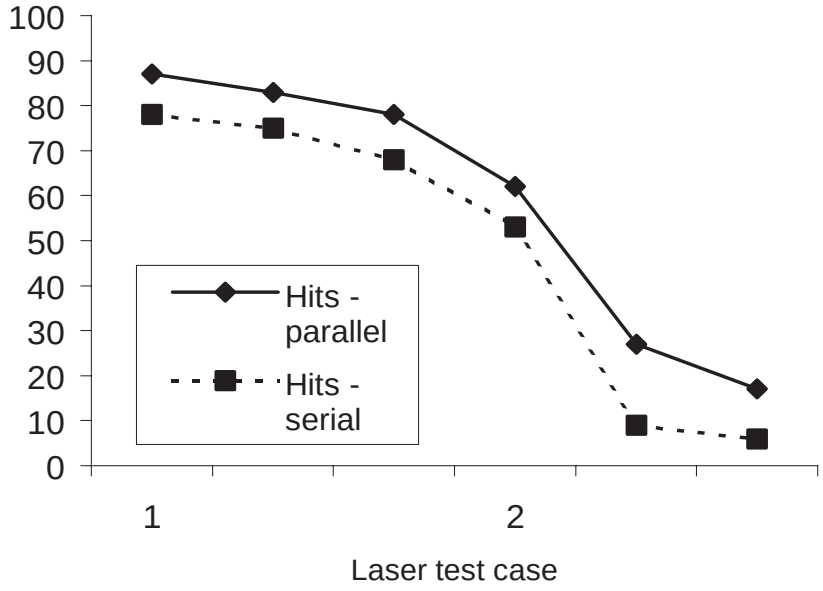


Figure 4.2 Comparison of gradient descent algorithms' success rate

The percentage of runs that were successful in each case, for both parallel and serial algorithms, is shown graphed in Figure 4.2. As can be seen, the parallel algorithm is consistently more successful at finding the global minimum. Not only is it able to avoid gross local minima in the data, such as can be seen in the original data set in Figure 3.3, but it is also able to avoid more of the noise related minima in the surface.

In addition, the concurrent execution of objective function evaluations obtained, averaged over all the runs, provided an average speedup of a factor of 4 compared to the serial code, using an 8-processor computer.

When finding the global minimum is important, as opposed to finding a sufficiently "good" result for some practical purpose, multiple runs can be performed from different starting points. If a 64-processor parallel computer (or cluster) were available, for the original, smooth data the serial algorithm could utilise this resource to improve its success rate from 78%, to a probability of finding the global minimum effectively equal to 1. Using the same 64-processor machine would improve the parallel algorithm's chance of finding the global minimum from 87% to a probability within $1.0 \times 10^{-8}\%$ of 1. It would also achieve this result 4 times faster.

4.2.3 Results for radio frequency design problem

For the Bead case study the later version of P-BFGS was compared with an Adaptive Simulated Annealing (ASA) code (Ingber 1989). For reference, the basics of the simulated annealing method are outlined in Section 2.3.1. For these trials the Bead test case was being directly computed, so results are limited to a single run for each algorithm. The convergence tracks of the algorithms in parameter space, and the location of the global minimum, are shown in Figure 4.3.

On the left side of the Figure, each subsequent point accepted by ASA is shown linked to the preceding point, forming a “track” of points accepted by the algorithm developing through time. Initially, while the “temperature” is high, large excursions can be seen over a significant portion of the domain. Despite this, the right hand side of the domain appears to have been largely unexplored. Then, as the “temperature” cools, the algorithm becomes increasingly constrained in its movement, the tracks become short and densely packed, and the algorithm converges on a local minimum in the middle left of the Figure.

On the right side of the Figure, the track traversed by the P-BFGS algorithm can be seen, as each subsequent line minimum is joined to the preceding track segment. Starting from a randomly chosen point that is below the global minimum – the “arrowhead”-shaped isosurface in the middle right of the Figure – the algorithm spirals toward the viewer and then away again, the spiral growing tighter, until it converges on a local minimum above and behind the global minimum.

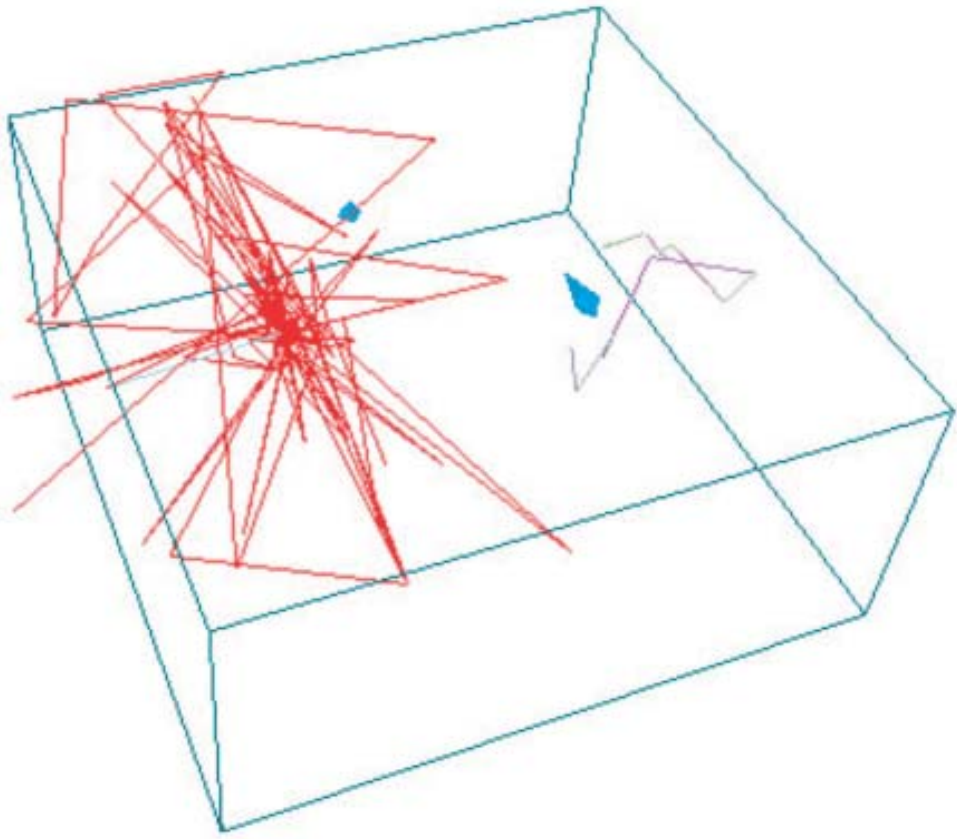


Figure 4.3 Cost function isosurface at -5 (transmission loss = -55dB) with ASA and P-BFGS optimisation points

The ASA algorithm could not be run to convergence due to limitations of time, as it was using a sequential implementation on a uniprocessor machine. However, in 143 function evaluations, it appeared to be converging to a minimum using the default algorithmic parameters. Given a parallel implementation across 8 processors, as was available to the P-BFGS implementation, this result would be equivalent to less than 18 steps. However, this approximation to the parallelism and speedup of ASA ignores any effect of parallel searches on the convergence rate of the algorithm.

By comparison, the P-BFGS algorithm terminated at a point above the global minimum after 44 steps, using 319 function evaluations. The technique adopted in the line search phase of interval subdivision along the entire search vector to the nearest boundary, coupled with iteration to desired solution tolerance on each line segment appears to place a large overhead of unproductive function evaluations on the algorithm, while the non-convex nature of the problem prevents the higher convergence rate of the gradient descent method from being attained. The speed of P-BFGS could perhaps have been increased by allowing inexact line searches for intermediate steps.

By inspection of Figure 4.3 it may be seen that neither algorithm is converging on the global minimum; both are attracted to small, "deep" local minima (neither was resolved by the enumeration and so cannot be seen in the Figure). It should be noted that both algorithms reached values sufficiently low to be of use in the engineering design of the handset measurement setup under consideration, and did so quicker than by enumeration.

The "optimal" solution for a high permittivity bead from ASA was determined to have $\varepsilon_r = 52$, O.D. = 20 mm and length = 57.5 mm, yielding a reflection loss, S_{11} of -10.3dB and transmission loss, S_{21} of -49dB at 1 GHz. The solution delivered by P-BFGS was a bead of $\varepsilon_r = 41.8$, O.D. = 85 mm and length = 102 mm, yielding a reflection loss, S_{11} of –14.7dB and transmission loss, S_{21} of –56.6dB at 1 GHz. These "optimal" beads were then applied to the handset/cable problem to determine the effectiveness in reducing cable perturbation effects.

4.3 Discussion of results

In a realistic problem of air quality management, use of a parallel gradient descent optimization algorithm has shown significant performance gains over other methods of solution. In a simple test it uses less than half the number of evaluations of a computationally demanding numerical simulation than previously used simple enumeration techniques require and is more than four times faster than traditional sequential optimization methods.

As indicated, the photochemical smog problem used an average 4.6 CPUs during the trial. There are practical limits to the number of CPUs this method can profitably utilize. During the gradient determination phase, two CPUs are required for each dimension, to perform the finite difference approximation. So any more than $2n$ processors are wasted for gradient determination. For the photochemical case study n= 2, so the maximum number of processors useful for gradient determination is 4.

During the line minimization phase, interval sub-division is repeated until the sub-intervals are less than the desired tolerance. So for a given fractional tolerance, t, the maximum number of CPUs that can usefully be employed is $1/t$. Since most line searches will be less than the entire span of the search domain, to a first approximation the average number of CPUs usefully employed will be $1/(2t)$. For the benchmark problem t= 0.1 so the number of processors useful for line minimization is 5.

For the photochemical smog problem, at the time of testing the number of processors actually available coincided with the theoretical optimum most of the time, as reflected in the high parallel efficiency. With some code refinements the parallel efficiency could be made very high because of the negligible extent of serial code, most computation being performed in parallel in the objective function evaluation. The algorithm scaling, however, is strictly limited by problem parameters.

The results obtained for the radio frequency test case demonstrate P-BFGS to be comparable with an implemetation of Simulated Annealing. Using parallel computing resources, it was able to return a superior objective function result in less than a third of the time taken by the sequential ASA code.

4.4 Summary

The parallel implementation of the BFGS algorithm has been found to provide moderate speedup on typical problems from the physical sciences using small-scale parallel computing resources, and consistently outperformed a serial implementation of the algorithm in its success at finding global minima in the presence of moderate to severe noise. As expected, the algorithm displayed rapid convergence, finding better solutions faster than, for example, a simulated annealing algorithm.

The scalability of the problem remains a significant shortcoming. The fundamental aim of the parallel optimisation algorithm approach is to reduce the total, "wallclock" time taken for solution of a problem by exploiting concurrency, as opposed to minimising the absolute number of objective function evaluations required. As could be seen, the number of function evaluations required by P-BFGS could exceed those requested by a variety of sequential algorithms. Its advantage lay in that the additional computational cost was performed in parallel, reducing the overall run-time. But the method was severely limited in the degree of concurrency obtainable, governed by problem dimensionality, which the evidence outlined in Section 3 suggests may be limited in many practical applications, and desired solution tolerance, which also can be quite limited, as in the problem described in 3.1.

To overcome these shortcomings, methods with the potential for more concurrency are investigated in the following sections. By reference to Figure 1.1, the next promising approach to be considered is the use of Direct Search methods.

5.0 Direct Search Methods

Gradient methods have generally restricted degrees of concurrency, limiting the extent to which use of parallel computing can speed the process of optimisation. A second reason to consider an alternative approach is the dependence of the methods on the availability of values not only of the objective function, f, but also its derivatives, ∇f. For the large numerical simulations that form the focus of this study, the latter are rarely available, particularly in industrial applications. Conn, Scheinberg and Toint (1997) venture the opinion that for very computationally expensive objective function evaluations the use of finite difference approximations cannot be justified. Another approach may be to apply automatic differentiation techniques, but availability of the complete source code for calculation of f is indispensable (Bischof, Bücker, Lang, and Rasch 2003). Unfortunately, for many problems of interest, particularly in industrial or commercial applications, the source code may not be available at all. As detailed in Section 1.2, Nimrod/O has been designed with the goal of general applicability, and no access to the internal operations of the subject simulation assumed. To continue this flexibility of application consideration may be given to other methods such as direct search.

In the direct search methods, gradient information is implicitly used by constraining moves to be "downhill". The steps of gradient determination and search, separate in gradient descent methods, are effectively merged in direct search methods. The early pattern search methods are, however, of very limited concurrency. The Multidirectional Search (MDS) method of Dennis and Torczon (Dennis and Torczon 1991, Torczon 1989, Torczon 1991) has greatly enhanced parallelism, all vertices of an $n + 1$ simplex being treated simultaneously. The work described in this section seeks to investigate hybrids and variants of Nelder-Mead Simplex and MDS with the key objective of increasing the amount of concurrency, and thus reducing the time to solve a given problem.

5.1 The Algorithms

5.1.1 Nelder-Mead Simplex and variants

When a search direction is chosen by the Nelder-Mead simplex algorithm, defined by the current worst vertex and the centroid of the remaining vertices, the original algorithm tries an exploratory step consisting of reflecting the worst vertex about the centroid. If this step is "successful", i.e. the objective function value is decreased, an expanded step is tried. If it is unsuccessful a contracted step is tried. If this is also unsuccessful, a contraction of the worst vertex toward the centroid, within the current simplex, is tried.

From this brief description, it can be seen that there is a "stencil" of points sampled by the algorithm, with never less than two being evaluated, and their location entirely determined by the geometry of the current simplex. As a basis for this work the original Nelder-Mead simplex algorithm was implemented with concurrent evaluation of all four trial points of this stencil, which provides an immediate improvement in execution speed relative to the original, sequential algorithm. Where subsequent reference is made to the Nelder-Mead algorithm, results are from this parallel implementation.

The work of Parkinson and Hutchinson (1972) suggests including line searching along the Nelder-Mead search direction has the potential to improve the performance of the algorithm. A line search method of iterative subdivision enhances parallelism and, as outlined in Section 4.1.1, has been demonstrated to improve the ability of the line search to avoid entrapment in local minima.

The parallel implementation of the Nelder-Mead algorithm was extended by applying this method of line searching in the work reported here.

It has been demonstrated that the Nelder-Mead Simplex algorithm in some cases exhibits behaviour that causes the algorithm to converge to a non-stationary point (McKinnon 1998). Figure 5.1 shows a test case reported, a function in two variables defined by: $f(x, y) = 360x^2 + y + y^2$ for $x \leq 0$ and $f(x, y) = 6x^2 + y + y^2$ for $x \geq 0$

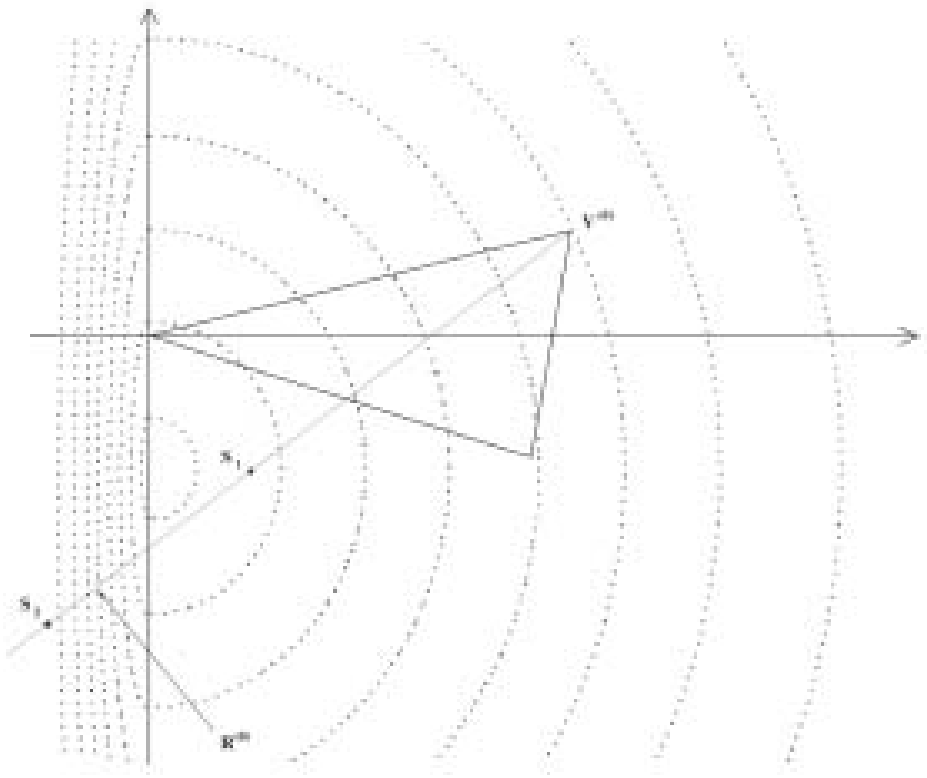


Figure 5.1 Nelder-Mead Simplex for $f(x, y) = 360x^2 + y + y^2$ for $x \leq 0$ and $f(x, y) = 6x^2 + y + y^2$ for $x \geq 0$

When an iterative subdivision line search method is used, it selects a number of sample points along the line segment from the current vertex, $V^{(0)}$, to the nearest external boundary in the search direction. Depending on the interval between sample points, there will be either:

- A sample point in the line segment from the vertex, $V^{(0)}$, to reflection of the vertex, $R^{(0)}$, in the region, for example, of S_1. If this is the case, $V^{(0)}$ will in due course be replaced by a point in this region, and the algorithm will proceed.
- A sample point in the line segment from $R^{(0)}$ to the boundary, in the region, for example, of S_2. In this case, since $V^{(0)}$is the current trial minimum, the interval between $V^{(0)}$ and S_2 will be iteratively subdivided until a minimum is identified, in the region of S_1, and the algorithm proceeds.

In neither case will repeated inside contraction occur in the manner described by McKinnon.

It is widely acknowledged (see, for example, Fletcher (1987), Kelley (1999a) or Gill et al. (1981)) that use of an *exact line search* is generally not worth the extra computational effort and may, in fact, degrade performance. To explore this issue, two variations of line search were implemented:

- Iterative subdivision line search, an exact line search within the limits of the desired tolerance, and
- Subdivision line search, but with only a single iteration.

5.1.2 MDS and variants

As a baseline for comparison the MDS algorithm was implemented, with concurrent evaluation of all trial points.

To implement line searching along the MDS search directions presents a problem. MDS avoids potential ill-conditioning experienced in the Nelder-Mead simplex algorithm by requiring each new simplex to be congruent to the previous one. In a simply-bounded parameter domain, this makes the individual line searches inter-dependent, since each must keep track of whether a proposed new vertex location on one line requires another vertex to be located outside the problem domain to maintain congruency. Such a method could be implemented with additional algorithm complexity by initialising a feasible set of search direction line segments, but this would still entail a limitation in potential expansion of the simplex along desirable search directions. To avoid these complications, line searching was introduced in a variant of the MDS algorithm that allows independent relocation of vertices, i.e. congruency is not enforced.

5.1.3 Hybrid methods

The MDS algorithm raises the idea of concurrent searching along multiple search directions. *Supplementary search directions* drawing on the methods of the Nelder-Mead algorithm and the MDS algorithm have been suggested (Hamma 1997) but, as illustrated in Figure 5.2, a straightforward, concurrent implementation of both sets of search directions is potentially inefficient.

When the search directions of the Nelder-Mead simplex algorithm are applied sequentially, producing new vertices 3a and 2b, the search directions are generally downhill, relative to the local gradient. However, if they are applied concurrently, it may quite often be the case that one or more may be to some degree uphill (for example, the trial vertex 2a). Note: the MDS search directions have been omitted for clarity: they would be from vertices 2 and 3, through vertex 1.

In this research a different approach to generating supplementary search directions for concurrent search is proposed. The search direction from the worst vertex is through the centroid of the remaining n vertices, as in the normal Nelder-Mead algorithm. But the search direction from the next worst vertex is through the $n - 1$ remaining vertices that are better than it, and so on, until the search direction from the second best vertex is reduced to the MDS search direction though the best vertex. All searches are performed concurrently, and all vertices are independently relocatable. The method can be considered as deriving from a hybrid of the Nelder-Mead and MDS algorithms. In this work it will be referred to as the Reducing Set Concurrent Simplex (RSCS) algorithm. The set of search directions generated is illustrated in Figure 5.3. Comparing the search directions illustrated in Figure 5.3 with the concurrent search directions of Figure 5.2, it may be noted that the search directions generated by RSCS are more likely to be downhill.

Line searching variants of this method were also implemented, both iterative subdivision and single-pass versions.

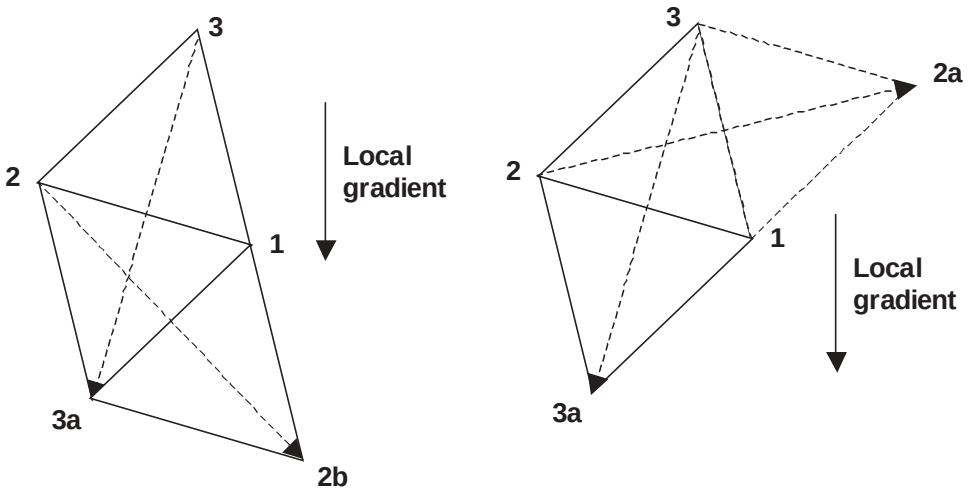


Figure 5.2 Sequential and concurrent Nelder-Mead reflection

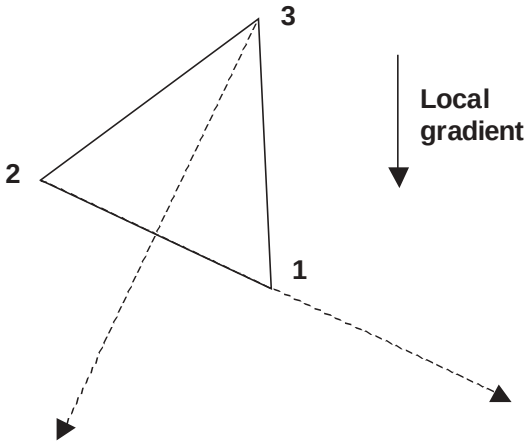


Figure 5.3 RSCS search directions

In summary, the algorithms to be tested were the original two algorithms:

- NM : the Nelder-Mead simplex algorithm (parallel implementation)
- MDS : the MDS algorithm of Dennis and Torczon

and the following variants and hybrids:

- NML : NM with iterative subdivision line search
- NML1 : NM with a single pass line search

- MDS1 : a variant of MDS with individually relocatable vertices
- MDS1L : MDS1 with iterative subdivision line search
- MDS1L1 : MDS1 with a single pass line search
- RSCS : Reducing Set Concurrent Simplex algorithm
- RSCSL : RSCS with iterative subdivision line search
- RSCSL1 : RSCS with a single pass line search

5.2 Results of Numerical Experiments

To evaluate the set of algorithms, seven test cases were used:

- The two Quantum Electro-dynamics cases: Laser 1 and Laser 2
- The two durable component design cases: Crack 1 and Crack 2
- The 2D aerofoil design case: Aerofoil
- The radio-frequency design case: Bead
- Rosenbrock's function in 2 dimensions

Each of the algorithms was run on each of the test cases from 10 randomly distributed start points. For the purposes of comparison, in a given test case the same set of start points were used for each algorithm. The starting simplices were right simplices aligned with the coordinate axes. By default they were scaled to 10% of the parameter range for each coordinate, as use of reasonably large simplices has been shown to enhance performance (Humphrey and Wilson 1998). Convergence criterion for most cases was a fractional step-wise gradient of 10^{-3}. Dolan, Lewis and Torczon (2000) have provided analytic support for the use of this measure by demonstrating pattern size is a reliable measure of stationarity. Despite the possibility raised by Lagarias, Reeds, Wright and Wright (1996) that simplices of zero diameter may have colinear vertices, rather than converging to a point, we prefer to be guided by Elster and Neumaier (1997) that it is more important to have an algorithm that is fast and robust in practice than one which converges in theory but is slow in practice.

Function evaluations are performed concurrently in batches in Nimrod/O, the general purpose optimisation toolset described in Section 1.2 in which the algorithms were implemented. The batch count can be interpreted as equivalent to Effective Serial Function Evaluations (ESFE), a measure of the wall-clock time taken for an algorithm to complete, providing all of the computations can be performed in parallel. This means that the machine must have enough processors to achieve this level of concurrency, which is not an unreasonable assumption given the proliferation of cheap clusters. The parallel job distribution mechanism used for the tests supported concurrent function evaluations in line searches, but not concurrent, multiple line searches. To evaluate the possible parallel performance of each algorithm, an estimate was made of ESFE for those that perform multiple, independent line searches by assuming that the line searches could be performed concurrently. This assumption was made for MDS1L, MDS1L1, RSCSL and RSCSL1.

By inspection of the relevant objective functions values tabulated in Appendix H, and the corresponding Shapiro-Wilk W test statistic for normality it can be seen that in many cases W is below the corresponding percentage point for normality, which for a significance level of 0.05 and n= 10, is 0.842. This leads to the rejection of the null hypothesis that the results are normally distributed. For this reason, median values are used as the basis for comparison of algorithms, and non-parametric, descriptive statistical methods are used for analysis of results.

Tables 5.1, 5.2 and 5.3 show, for each algorithm on each test case over the 10 runs performed, respectively:

- The median objective function value obtained,
- The median number of function evaluations performed (FE) and
- The median Effective Serial Function Evaluations (ESFE).

	Laser 1	*Laser 2*	*Crack 1*	*Crack 2*	*Aerofoil*	*Bead*	*Rosenbrock*
NM	**-0.481**	-0.032	191.9	5299	-67.85	3.66	**0**
NML	-0.459	0.037	196.6	5294	-67.08	-13.82	0.004
NML1	-0.451	-0.12	196.9	5276	-66.14	**-15.68**	0.039
MDS	**-0.481**	0.279	**188.0**	**5319**	-67.92	1.63	0.066
MDS1	-0.100	0.287	190.7	5268	-59.39	8.68	1.744
MDS1L	0.227	0.271	200.3	5249	-51.94	1.61	0.882
MDS1L1	0.041	0.299	201.7	5259	-52.63	1.61	1.120
RSCS	**-0.481**	-0.286	193.5	5310	**-68.56**	2.41	0.186
RSCSL	-0.190	**-0.542**	197.8	5295	-58.00	-13.33	0.124
RSCSL1	-0.467	-0.460	192.6	5291	-60.29	-2.28	0.034

Table 5.1 Median results obtained across 10 runs – Objective function values

	Laser 1	*Laser 2*	*Crack 1*	*Crack 2*	*Aerofoil*	*Bead*	*Rosenbrock*
NM	97	100.5	106	66	88	50	240
NML	971	922.5	690	582	752	179.5	1562
NML1	241	222.5	254.5	157	267	138.5	370
MDS	96	93	94	260.5	121	49	2358
MDS1	95	95	130	88	160	70	139
MDS1L	876.5	1046	1135	2344	974.5	219.5	587
MDS1L1	194	310	314	1096	305.5	170	348.5
RSCS	103	120.5	132	180	118	106	227
RSCSL	725.5	751	760.5	743.5	624.5	164.5	962.5
RSCSL1	298	361	363	312	403.5	120.5	421

Table 5.2 Median results obtained across 10 runs – Function evaluations

For each test case, the **best median objective function value** is highlighted in **bold** type. Also highlighted is the **fastest time,** in terms of ESFE, for an algorithm to achieve a result within 10% of

	Laser 1	*Laser 2*	*Crack 1*	*Crack 2*	*Aerofoil*	*Bead*	*Rosenbrock*
NM	24.5	25.5	26.5	16.5	22	12.5	63.5
NML	58.5	55.5	42	35	45	25	96
NML1	18.5	19.5	16.5	10	17.5	12	30.5
MDS	16.5	16	**11**	**29.5**	14	6	393.5
MDS1	12.5	12.5	11.5	8	14	6.5	18
MDS1L	31	34	29.5	61.5	24	18.5	19
MDS1L1	11.5	16	8	39	8	12.5	18.5
RSCS	13.5	16	12	16	**10.5**	10	29
RSCSL	22.5	23	16.5	16	13.5	**8.5**	31.5
RSCSL1	**12.5**	**17**	8.5	8	9	4.5	**21**

Table 5.3 Median results obtained across 10 runs – Equivalent Serial Function Evaluations

the best median objective function, as a percentage of the range from best median value obtained to worst. Existing algorithms are shown shaded.

The results in Tables 5.1-5.3 show that the MDS algorithm produced the best median result for about half the test cases, but was generally slower than other algorithms. On the other hand, algorithms from the new, RSCS family were the fastest, successful algorithms in 5 of the 7 cases. MDS1 and MDS1L1 were often the outright fastest algorithms, but at the expense of the result obtained.

Table 5.4 shows the best objective function value obtained in 10 runs for each algorithm on each test case. Table 5.5 shows the actual ESFEs required to obtain that result. For each test case, the best objective function value obtained by any algorithm, and the fastest ESFE to obtain that value, are highlighted in **bold** type.

It may be noted from Tables 5.4 and 5.6 that in general the Nelder-Mead algorithms with line searching, and RSCS and its variants, obtain better or similar results to the Nelder-Mead and MDS algorithms, and do so faster.

	Laser 1	*Laser 2*	*Crack 1*	*Crack 2*	*Aerofoil*	*Bead*	*Rosenbrock*
NM	**-0.48**	**-0.56**	187.6	5353	**-68.64**	-26.98	**0**
NML	**-0.48**	**-0.56**	187.6	5332	**-68.64**	**-39.85**	**0**
NML1	**-0.48**	**-0.56**	**187.5**	5331	-68.63	**-39.85**	5e-6
MDS	**-0.48**	**-0.56**	187.6	**5357**	**-68.64**	-16.12	3e-4
MDS1	**-0.48**	**-0.56**	187.7	5348	-68.33	-21.61	12
MDS1L	-0.25	-0.37	191.9	5331	-66.39	-26.91	19
MDS1L1	-0.25	-0.37	198.5	5348	-66.39	-26.91	8
RSCS	**-0.48**	**-0.56**	187.6	5347	**-68.64**	-26.91	**0**
RSCSL	**-0.48**	**-0.56**	**187.5**	**5357**	-67.21	-26.98	6e-6
RSCSL1	**-0.48**	**-0.56**	187.7	5348	-68.62	-26.98	8e-4

Table 5.4 Best objective function values obtained in 10 runs

	Laser 1	*Laser 2*	*Crack 1*	*Crack 2*	*Aerofoil*	*Bead*	*Rosenbrock*
NM	24	24	25	23	20	16	54
NML	55	51	30	46	38	28	112
NML1	15	24	**13**	12	18	**13**	48
MDS	16	**12**	13	1000	14	7	1000
MDS1	**12**	13	12	30	10	6	25
MDS1L	14	63	19	19	36	153	56
MDS1L1	30	4	8	7	11	102	19
RSCS	22	**12**	9	42	**7**	12	**39**
RSCSL	22	24	15	**27**	18	11	88
RSCSL1	13	17	11	8	15	5	33

Table 5.5 Time taken, in ESFE, to achieve best objective function values, across 10 runs

Table 5.6 shows the probability, derived from a binomial trial model, that an algorithm will **not** successfully obtain a "good" result in 10 runs, for each test case. The criteria for success were:

- For all test cases *except* the Bead test case and Rosenbrock's function: an objective function value that was within 10% of the best result obtained by any algorithm, measured in the range of all results obtained by any algorithm.
- Bead test case: all but a very few algorithms experienced considerable difficulty in obtaining a good result on this test case. This difficulty is largely a result of the nature of the global optimum, which exists in a very narrow, deep "notch" among several other local minima. The parameter space also contains several large regions forming local attractors. A final objective function value less than zero (about 50% of the range from best to worst results obtained by any algorithm) was considered "successful" in order to assess the relative merits of most of the algorithms tested. For a conventional measure of success, such as was used for the other test cases, success rates were so low as to make it difficult to separate the algorithms.
- Rosenbrock's function: a threshold of 10% of the range above the global minimum value of zero gave most algorithms tested uniformly high rates of success. It was necessary to reduce the threshold to 1% to separate them.

Results indicating a better than 99% probability of success have been highlighted in **bold** typeface. Whilst these results indicate that almost all algorithms can find solutions almost all of the time, the earlier tables highlight the detailed performance differences between them. Table 5.6 does illustrate that MDS1 and its derivatives have a significant problem returning adequate results. Of all the algorithms, only MDS, NML and RSCS can be guaranteed to return a good result across **all** cases.

To illustrate the time behaviour of the algorithms, the objective function value achieved at each iteration of each algorithm for representative runs on each of the laser problems are plotted in Figures 5.4 and 5.5. It should be noted that iteration count is not a strict measure of time taken. For iterative line searching, several batches of function evaluations may be dispatched sequentially during a single iteration.

As might be expected from the tabulated results, the hybrid algorithms with line search rapidly achieve a good result on both cases. For the Laser 1 problem, a "smooth" test case, the Nelder-Mead

	Laser 1	*Laser 2*	*Crack 1*	*Crack 2*	*Aerofoil*	*Bead*	*Rosenbrock*
NM	**0**	**0.09**	**1.0e-8**	**0.09**	**1.0e-8**	2.8	**0**
NML	**1.0e-5**	**0.6**	**0.6**	**1.0e-8**	**0.09**	**1.0e-5**	**1.0e-5**
NML1	**0.01**	2.8	2.8	**0**	**0.09**	**1.0e-5**	**1.0e-8**
MDS	**0**	**0.6**	**0.01**	**1.0e-8**	**0.0006**	**0.09**	**1.0e-8**
MDS1	10.7	10.7	**0.6**	**0.0006**	34.9	2.8	**0.6**
MDS1L	100	100	100	**0.01**	34.9	**0.09**	**0.09**
MDS1L1	100	100	100	**0.01**	34.9	**0.09**	**0.09**
RSCS	**1.0e-5**	**0.09**	**0.6**	**1.0e-8**	**0.0006**	**0.6**	**0.0006**
RSCSL	**10.7**	2.8	10.7	**1.0e-5**	2.8	**1.0e-5**	**1.0e-8**
RSCSL1	2.8	0.6	2.8	**1.0e-8**	2.8	**0.09**	**1.0e-8**

Table 5.6 Probability of failure in 10 runs

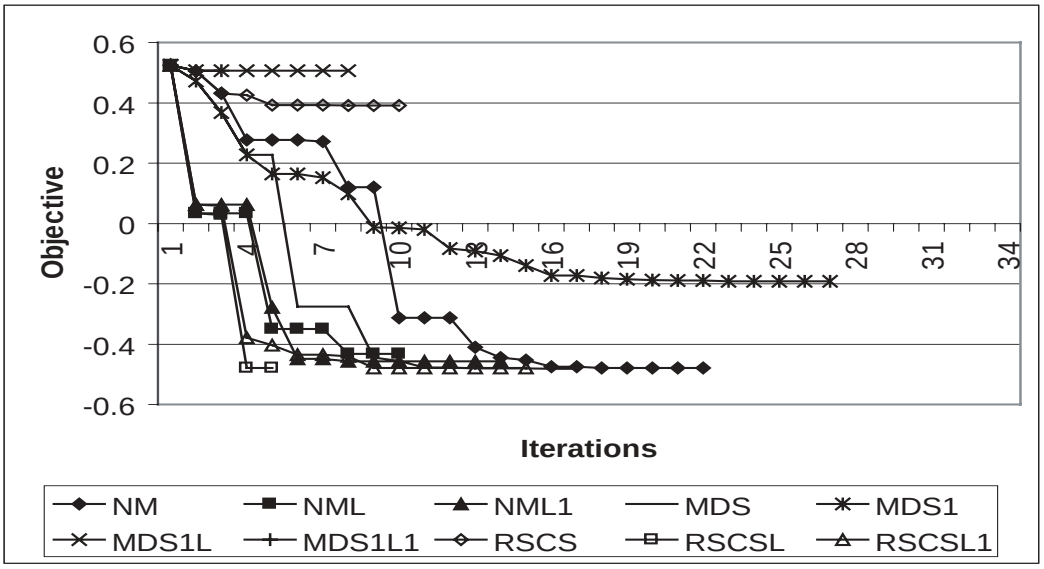


Figure 5.4 Convergence history for Laser 1 problem

and MDS algorithms also achieved a good result, if somewhat slower. For the Laser 2 problem, with many local minima, the progress of the Nelder-Mead algorithm is slower still, and several other algorithms became trapped in local minima. The run chosen to illustrate performance on the Laser 2 test case gave the algorithms a starting point in a region that, with the initial scaling of the simplex, yielded sufficient decrease for the Nelder-Mead and MDS algorithms to “step over” many of the local minima, emphasising the advantage of a sufficiently large initial simplex. If a more distant starting point was chosen, only the line searching variants of RSCS made any significant progress, as illustrated in the example run in Figure 5.6. This echoes the view of Durand and Alliot (1999) that the Nelder-Mead algorithm can only find a local optimum close to the starting point, but simultaneously demonstrates an approach, incorporating a line search, that ameliorates this difficulty.

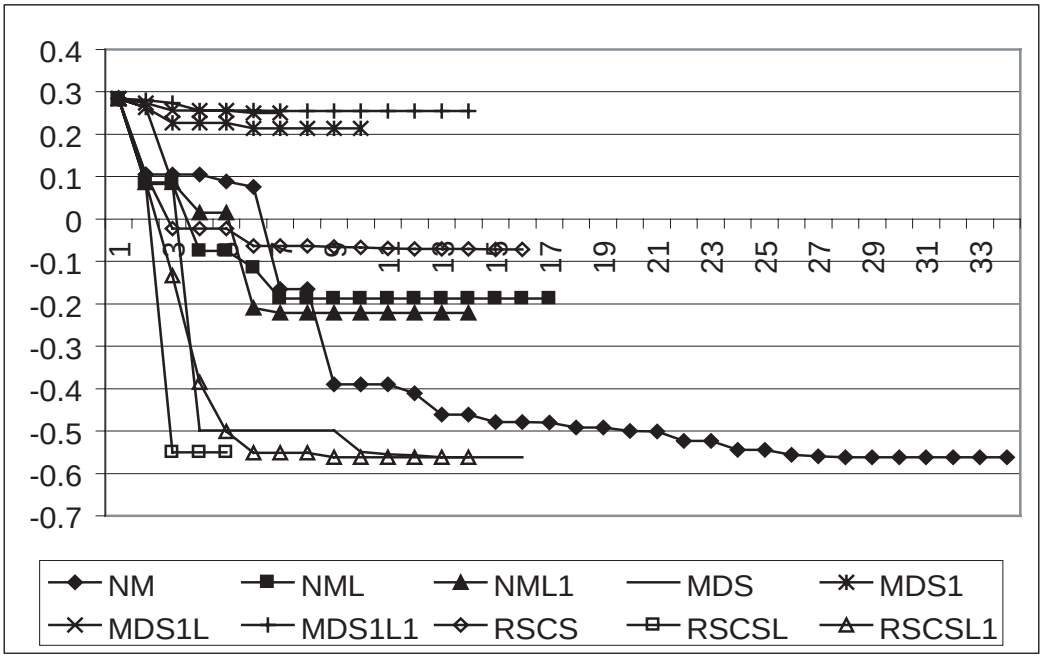


Figure 5.5 Convergence history for Laser 2 problem

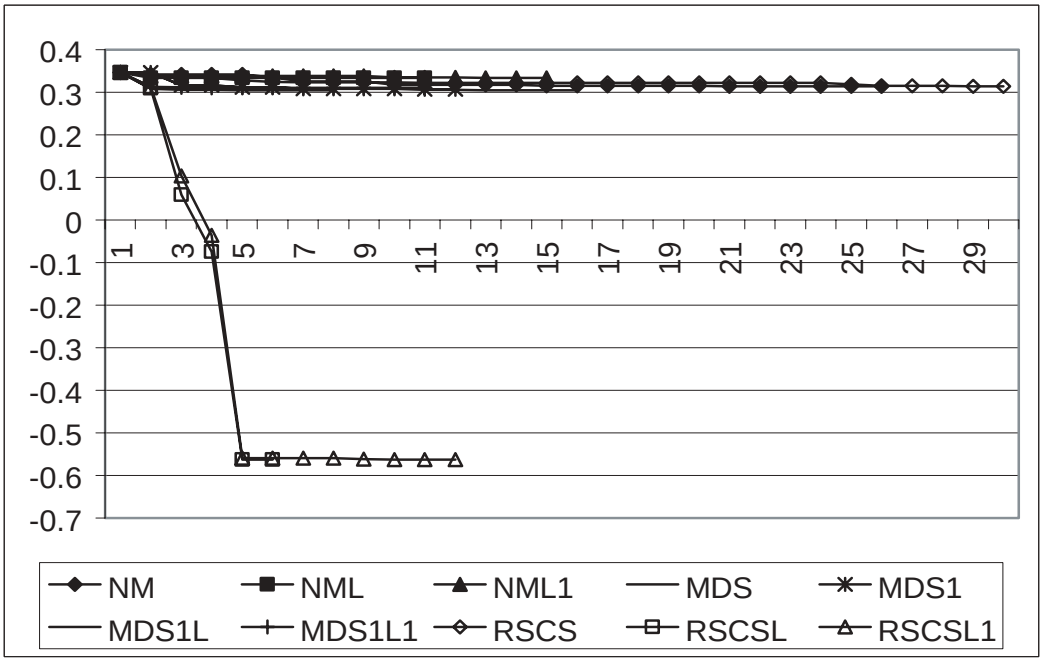


Figure 5.6 Convergence history for Laser 2 problem, from starting point distant from global minimum

In both examples, and in general across the remaining test cases, the line-search variants of the MDS1 algorithm were very prone to premature termination. In these algorithms vertex replacement is entirely free as to location. The diameter of the simplex is not bounded above zero and no congruency of simplices is required. With the MDS-style search directions through the current best vertex, this led to a common mode of failure of the algorithm in which the current best vertex was identified as the best point to which worse vertices should be relocated when the line search did not yield a better, external point. The simplex would effectively collapse onto itself.

The line-search variants of the RSCS algorithm had a similar failing. But with the concurrent use of other, Nelder-Mead-style search directions, collapse of the simplex to a single point was delayed. Instead, the termination phase of the algorithm was beneficially accelerated: few general contractions of the simplex, if any, were observed. RSCSL and RSCSL1 owe some degree of their speed to this accelerated termination. Occasionally, the unbounded replacement of vertices yielded ill-conditioned or degenerate simplices, arbitrarily flat or needle-shaped. This provides a readily understood mechanism, for this algorithm, of a phenomenon already observed for the Nelder-Mead algorithm (Bassiri and Hutchinson 1994, Wright 1995, Lagarias et al. 1996, Lewis, Torczon and Trosset 2000a). RSCS allows a very rapid elongation of the simplex when one of the search directions becomes aligned with a long, downhill run. This elongation will cause a near-loss of linear independence in the search directions. If the required search direction to proceed further is nearly conjugate to this elongation, the concept on which conjugate gradient methods is predicated, failure is likely to occur. It is possible to conjecture a sequence of extensions of the Nelder-Mead simplex that could lead to a similar elongation. A suggestion has been made for a method specifically to protect against this eventuality, by ensuring internal angles of the simplex are bounded above zero (Tseng 1999, Nazareth and Tseng 2002).

Closer examination of run logs for RSCSL1 on the aerofoil test case, in which it returned poor median objective function values, suggests there were problems with the algorithm rapidly approaching, and collapsing the simplex onto, the parameter domain boundaries. It is possible to devise special boundary-handling code to correct this problem. However, several authors have suggested that it is easier for simplex-based, pattern search algorithms to recover from ill-conditioned simplices simply by restarting with rescaled simplices, i.e. construct a new simplex around the best vertex found so far, with a scale appropriate to the stage of optimisation (larger to start with and decreasing in size with iteration count) and orientation reset either to a convenient default or with orthogonal sides having a difference approximation to the local direction of steepest descent (Wright 1995, Kelley 1999).

To quantitatively investigate the effect of introducing line searches, pair-wise comparisons were made between the Nelder-Mead algorithm and its two line-searching variants. The Mann-Whitney U test statistic was computed for each comparison on each test case and results are tabulated in Appendices A and B. These results indicate that the objective function values obtained on the Bead test case by the line searching variants were better than the original algorithm at a significance level of 0.05. On the smooth Laser 1 test case the original algorithm was significantly better than the single-pass line-searching variant. On all other test cases, there was no statistically significant difference between the results obtained by the original algorithm or its line-searching variants. The only difference between them was their speed.

Similarly, comparisons were made between the RSCS algorithms and the Nelder-Mead and MDS algorithms. The Kruskal-Wallis H test statistic was computed across results for the different algorithms, for each test case. Results are tabulated in Appendix C. At a significance level of 0.05, the percentage point of the χ^2-distribution for the 5 algorithms tested, i.e. 4 degrees of freedom, is 9.4877. Comparison of the H test statistic with this percentage point indicates that the results of at least two of the algorithms differed in their distributions in only two cases, the Laser 1 and Aerofoil test cases. These are two “smooth” test cases, and inspection of the rank sums indicates

the Nelder-Mead and MDS algorithms were the best performing. However, the rank sum of RSCS on the Aerofoil test case is equally high.

To confirm the relative performance of RSCS on the Aerofoil test case, a pair-wise comparison was performed with each of Nelder-Mead and MDS, using the Mann-Whitney U test statistic. The results are tabulated in Appendix D. They confirm that RSCS returns results as good as the Nelder-Mead and MDS algorithms. The only remaining difference between the algorithms is speed. Reference to Table 5.3 shows RSCS was generally twice as fast as Nelder-Mead and a third faster than MDS on this test case.

From Table C.6, the results obtained by the iterative line-searching variant of RSCS appear to be better than other algorithms. A pair-wise comparison was performed with each of Nelder-Mead and MDS, using the Mann-Whitney U test statistic. The results are tabulated in Appendix E. They indicate that, to a significance level of 0.05, RSCSL returns better objective function values on the Bead test case.

5.3 Discussion of Results

From the results in Table 5.1, it can be seen that the addition of line searching has improved the performance of the Nelder-Mead simplex algorithm. The single pass line search variant, in particular, is consistently faster, on average by about 53%. The median values obtained were either better, or never more than 10% worse. Analysis of objective function values returned, as shown in Tables A.6 to A.4, indicates there was no statistically significant difference in the quality of results returned by line-searching variants of the algorithm.

The new, hybrid algorithm, RSCS, gave median results and ESFE that were consistently better than the Nelder-Mead algorithm on “noisy” test cases and never more than 5% worse on “smooth” cases. On average, RSCS was 78% faster than the Nelder-Mead algorithm. Analysis of objective function values returned, as shown in Tables B.6 to B.4, indicates there was no statistically significant difference in the quality of results returned by RSCS, Nelder-Mead simplex and MDS algorithms, except for the “smooth” Laser 1 test case. From Table 5.4 it can be seen that RSCS was capable of equalling the best objective function value returned by both the Nelder-Mead and MDS algorithms on this test case. From Table 5.1, it can be seen that RSCS is approximately 20% faster than MDS on average. This excludes the time taken by MDS on Rosenbrock’s function, which was considered a pathological example of the tendency of simplicial methods toward premature convergence (Barton and Ivey 1996).

The single pass line search variant of the new algorithm, RSCSL1, also surpassed the Nelder-Mead simplex algorithm. Among the fastest of the algorithms, it was on average 2.4 times faster than the Nelder-Mead algorithm. The median objective function values obtained were better on two of the “noisy” surfaces. On one of the “smooth” test cases the **median** result was about 30% worse than that of Nelder-Mead, but the **best** objective function values obtained were never more than 0.1% worse than those obtained by the Nelder-Mead algorithm across all test cases, and other median results were within 10%.

RSCSL1 was also faster than MDS, on average by a factor of 6. This was primarily due to very slow runs for MDS on Rosenbrock’s function. If the poor times on this test case are ignored, RSCSL1 is about 35% faster than MDS. Once more, the median result from RSCSL1 on the aerofoil test case was about 32% worse than MDS, but the best objective function values it obtained were in some cases better, and never more than 0.1% worse than those obtained by MDS, and other median results were within 10%.

The MDS algorithm demonstrated a high probability of a "successful" result, as shown by the results in Table 5.6. However, it must be noted that the leniency of the criterion for "success" in the Bead test case contributed to this result: comparison of the best results obtained on this test case, as shown in Table 5.4, show MDS obtained the worst result of all the algorithms. In addition, on a couple of individual runs the algorithm appeared to reduce the diameter of the simplex prematurely, from which it did not recover, resulting in extremely long run times.

The iterative subdivision line search variant of the Nelder-Mead algorithm, NML, also had a high probability of success, but at the expense of speed: Table 5.1 shows it was generally slower than the un-modified algorithm. In contrast, RSCS also had a high probability of success but was faster.

5.4 Summary

In general, the Nelder-Mead and MDS algorithms performed well on the "smooth" test cases, but poorly on the class of cases with noise or many local minima. The new algorithm, RSCS, and its line searching variants, performed reasonably (or very) well on both types of problems. They generally provide better results, and deliver them faster.

The inclusion of line searching in the Nelder-Mead simplex algorithm also proved to be very advantageous. The single-pass line search variant is consistently faster than the unmodified algorithm, and delivered equivalent or better results across almost all test cases.

It may be observed from the results in Table 5.6, that **all** algorithms had a high probability of a successful result using multiple starts on Rosenbrock's function. This would tend to suggest that this function is not really a good predictor of performance on real-world problems. For this reason, results for Rosenbrock's function are included for qualitative comparison, but are not included in detailed analyses.

With the exception of the Nelder-Mead algorithm on "smooth" test cases, all algorithms had at least one run on each test case that yielded an exceptionally poor result. For the purposes of optimisation of industrial or commercial problems, it would appear multiple runs of any chosen algorithm are a practical necessity. Given multiple starts, MDS, RSCS and the iterative line search variant of the Nelder-Mead simplex algorithm could be guaranteed to yield at least one good result across all test cases.

Comparing the direct search methods discussed in this chapter with the gradient methods of the previous chapter all have the advantage that, for most of the time, each step produces some improvement in the objective function. The gradient methods require two steps, gradient determination and a line search, to produce similar improvement. In terms of their use of parallel computing resources they fall into one of a number of categories:

- If methods do not utilise a line search they show somewhat *less* concurrency, of similar order to the gradient determination step of P-BFGS. The (parallel) Nelder-Mead simplex algorithm falls into this category.

- MDS and RSCS fall into an intermediate category. They do not use a line search, but treat multiple vertices simultaneously. The four-point stencil sampled in each search direction allows them the use of twice the parallel computing as the gradient determination step of P-BFGS. Their concurrency relative to the line search step will be dependent on the dimensionality of the problem and the desired tolerance.

- If the methods use a single line search at each step they show about the same level of concurrency, since the line search governs this in each case. The line-searching variants of the Nelder-Mead simplex, NML and NML1 are in this group.
- If several line searches are executed simultaneously, they demonstrate concurrency roughly $\mathcal{O}(n)$ greater than the gradient methods. The new RSCSL and RSCSL1 fall into this category.

While generally utilizing parallel computing resources a little more effectively than the gradient methods, all algorithms were limited to a greater or lesser degree in their maximum concurrency. Line searching increases concurrency, and the MDS and RSCS algorithms also have greater potential, though some have considered this a disadvantage for problems of high dimensionality, preferring the Nelder-Mead algorithm to other methods such as MDS because of the former's parsimony in function evaluation (Kelley 1999). To effectively use more parallel computing resource we must turn to stochastic methods.

6.0 Stochastic methods

Of the optimisation methods outlined in the simple taxonomy in the Introduction, those offering the greatest concurrency are **population-based methods**, in which large numbers of objective function evaluations are typically performed in parallel, evolving solutions over several "generations". It is only over the past decade that the computational capacity available to scientists and engineers has increased to the point where population-based methods of optimisation have become practical for the solution of real-world problems. Moreover, in engineering design, the evaluation of the objective function is so much slower than the rest of the algorithm, that such codes demonstrate excellent speedup in spite of the need for global communication on each iteration. **Genetic Algorithms** (GA) now may be frequently encountered in application to engineering problems (for examples, see Alander (1995)). These have inherent, easily exploitable parallelism, and attractive global convergence probabilities, but have been considered generally slow to converge (Durand and Alliot 1999).

Among the population-based methods, Genetic Algorithms are the most widely used, as is clear from such surveys of the published literature as in Van Veldhuizen (1999) or Coello Coello (1999). However, with the real-valued problems that are the main focus of the work reported here it is unclear why joining, for example, the mantissa of a parameter from one trial solution with the exponent from another should be expected to provide anything more constructive than a random search. Exchange of whole parameter values may beneficially provide a means for exploration of different neighbourhoods in parameter space, and a Genetic Algorithm might be constructed that preserves the integrity of parameter values under its recombination (crossover) operators (Wright 1991), but this work instead concentrates on the purely mutational approach of the Evolutionary Programming methods, while incorporating concepts of self-organised criticality and elitism.

6.1 Self-organised Criticality

The theory of self-organised criticality gives an insight into emergent complexity in nature (Bak 1996, Bak and Sneppen 1993). Systems in stable equilibrium exhibit linear behaviour. The system's response to a disturbance is proportional to the size of the disturbance. Large fluctuations can only occur if several factors simultaneously combine to act in the same direction, which is unlikely to occur. Such a system, then, is also unlikely to adapt rapidly to the demands of an objective function.

At the other end of the spectrum, chaotic systems can react violently to change, as small perturbations of initial values are amplified in the system's response. Chaotic systems have no memory of their past states and cannot evolve.

However, at the transition from a stable system to chaos, complex behaviour can emerge. It is this critical state that may deliver efficient adaptation. Self-organised critical systems evolve to the critical state without any external organising force. This is advantageous because it implies no *a priori* information about the internal functioning of the systems is required to develop an effective means of optimising an objective function expressed in terms of externally exposed parameters and observed system response.

Bak contended that the critical state was "the most efficient state that can actually be reached dynamically". Inspection of many natural phenomena suggests the critical state is capable of efficient adaptation to environmental pressures using simple, robust systems. If optimisation is considered as the adaptation of a system described by its parameters to the selective pressure of an objective function, then it appears developing a critical state may be a highly effective method of optimisation.

Bak sought to model evolution of species with a simple model of inter-species interactions and selection. A number of species were arranged, arbitrarily and randomly, in a ring topology. At each time step, the least fit species and its two neighbours in the ring were replaced by randomly instantiated new species. This model was demonstrated to lead to complex behaviour, with gradual evolution of the fitness of the whole population.

Self-organised criticality is also exhibited by the "sandpile model". In this model, "grains" of sand are modelled numerically, stacking upon each other and toppling onto neighbouring stacks under simple rules. The structure of this simple system can be observed to evolve to an organised state with dynamics characteristic of criticality.

6.2 EPSOC: an Evolutionary Programming algorithm using Self-Organised Criticality

In earlier applications of self-organised criticality to optimisation, it has been proposed that a separately computed power-law extinction rate be imposed on a spatial diffusion model, or cellular GA (Krink and Thomsen 2001). Krink and Thomsen's model used pre-computed, stored dynamics of a sandpile model to control the size of extinction events in a diffusion model. The algorithm apparently does not attempt to evolve a population in a critical state, but indirectly imposes the observed behaviour of such a population.

In both Bak's nearest-neighbour, punctuated equilibrium model, and Krink and Thomsen's spatial diffusion model the population members are artificially arranged spatially: in a ring in the former, and in a toroidal, 2D grid in the latter. In contrast, by considering the trial solution parameter vectors as defining a location in an $n-$dimensional parameter space, the spatial behaviour of Bak's model is realized naturally in EPSOC.

In its operation EPSOC is largely a straightforward implementation of Bak's model as an optimisation algorithm, though using Euclidean parameter space for spatial operators. It also diverges in applying a high degree of greediness to the algorithm, which has been clearly demonstrated to improve the performance of a Genetic Algorithm (Zitzler, Deb and Thiele 2000). Maintaining a large "elite" (in EPSOC, half the total population) can be viewed as a "constructive" operator. An analogous operator is "Maxwell's demon".

Maxwell's Demon is an imaginary creature created by the mathematician James Clerk Maxwell to contradict the second law of thermodynamics. In a thought experiment in gas dynamics, the demon operates a molecule-sized trap door in a partition across a gas-filled cavity. The demon observes the gas molecules and allows faster than average molecules through the door so they end up on one side, and ensures slower than average molecules approaching the door end up on the other side. So after these operations one half of the cavity is filled with all the faster than average gas molecules, and the other half with all the slower than average ones. The demon is decreasing the randomness of the system (by ordering the molecules according to a certain rule). By protecting half of the population from extinction events, applying greediness accretes information in a manner similar to Maxwell's demon, and encourages a gradual improvement in the better half of the population.

6.2.1 EPSOC implementation

Restated, the general optimisation problem for an arbitrary, non-linear function f is:

$$\text{Minimize } f(\mathbf{x}) \text{ where: } f : \Re^n \rightarrow \Re \text{ and } \mathbf{x} = \{x_0,\ldots,x_i,\ldots,x_n\} \tag{6-1}$$

For a population-based method, the population, p, consists of a set of parameter vectors, $\mathbf{x} : p = \{\mathbf{x}_0, ..., \mathbf{x}_i, ..., \mathbf{x}_m\}$. For the real-world engineering design problems being considered, values for $f(\mathbf{x})$ are generally derived from execution of complex numerical simulations requiring considerable computation time.

The steps of the EPSOC algorithm are:

1. Initialise a random, uniformly-distributed population, p, and evaluate each trial solution, $\mathbf{x}_i \quad \forall \quad i$.
2. Sort the population by objective function value, $f(\mathbf{x})$.
3. Select a set, B, of the *nbad* worst members of the population. For each member of B, add to the set its two nearest neighbours in parameter space that are not already members of the set, *or from the best half of the sorted population.*
4. Apply a random, uniformly-distributed mutation to the selected set, B, i.e. re-initialise them. For all other members of the population, generate a "child" by applying a small ($\sim$10% of parameter range), random, uniformly-distributed mutation to the "parent" member.
5. Evaluate each new trial solution, $f(\mathbf{x})$.
6. If a child has a better objective function value than its parent, replace the parent with the child.
7. Repeat from step 2 until a preset number of iterations have been completed.

As each set of parameters defining a trial solution is independent of all others, it is immediately apparent that the evaluation of trial solutions at steps 1 and 5 can be performed concurrently. Since the evaluation of the objective function completely dominates the execution time, from Amdahl's Law we can expect extremely high parallel efficiency.

6.3 Results of Numerical Experiments

To evaluate EPSOC its performance was compared to that of a set of algorithms, including a GA (Genesis 5.0 (Grefenstette 1984)), a parallel gradient descent method (P-BFGS), Dennis and Torczon's MDS, the Simplex method of Nelder and Mead, a Reducing Set Concurrent Simplex (RSCS), and line-searching variants of Simplex and RSCS.

Seven test cases were used: ,

The two Quantum Electro-dynamics cases: Laser 1 and Laser 2

The two durable component design cases: Crack 1 and Crack 2

The 2D aerofoil design case: Aerofoil

The radio-frequency design case: Bead

Rosenbrock's function in 2 dimensions

Each of the algorithms tested was run on each of the test cases from 10 randomly distributed start points. To allow direct comparison, in a given test case the same set of start points were used for each algorithm. The convergence criterion for most cases was a fractional step-wise gradient of 10^{-3}, except for EPSOC and Genesis, which used fixed limits on iteration count and function evaluations, respectively.

The population-based algorithms, EPSOC and Genesis, both used a population of 64 members and elitist strategy. Other operational parameters of the algorithms, such as mutation and crossover rates for the GA, and extinction rate and mutation width for EPSOC, were tuned for each test case. Three different values were chosen for each of these operational parameters, and the two algorithms run with each of the nine resulting combinations. The results of these trials are tabulated in Appendices F and G. Further interpretation of their significance can be found in Section 6.4. The Shapiro-Wilk W test statistic for normality was applied to each set of results for a given set of operational parameters. For a significance level of 0.05, the corresponding percentage point for the W test for normality, for n= 10, is 0.842. By inspection, many of the groups display a W test statistic less than this, leading to the rejection of the null hypothesis that they are normally distributed. For this reason, non-parametric, descriptive statistical methods are used for analysis of results.

Table 6.1 shows the best objective function value obtained in 10 runs for each algorithm on each test case. Table 6.2 shows the Equivalent Serial Function Evaluations (ESFE) required to obtain that result. For each test case, the best objective function value obtained by any algorithm, and the fastest ESFE to obtain the best value, are highlighted. Existing algorithms are again shown shaded.

	Laser 1	*Laser 2*	*Crack 1*	*Crack 2*	*Aerofoil*	*Bead*	*Rosenbrock*
P-BFGS	**-0.48**	**-0.56**	**187.5**	5347	-67.90	-29.71	7e-2
NM	**-0.48**	**-0.56**	187.6	5353	**-68.64**	-26.98	**0**
NML1	**-0.48**	**-0.56**	**187.5**	5331	-68.63	**-39.85**	5e-6
MDS	**-0.48**	**-0.56**	187.6	**5357**	**-68.64**	-16.12	3e-4
RSCS	**-0.48**	**-0.56**	187.6	5347	**-68.64**	-26.91	**0**
RSCSL	**-0.48**	**-0.56**	**187.5**	**5357**	-67.21	-26.98	6e-6
RSCSL1	**-0.48**	**-0.56**	187.7	5348	-68.62	-26.98	8e-4
EPSOC	**-0.48**	**-0.56**	**187.5**	5356	**-68.64**	**-39.85**	1e-3
GA	**-0.48**	**-0.56**	188.5	5346	-68.51	**-39.85**	2e-4

Table 6.1 Best objective functions values obtained in 10 runs

In Table 6.1 it may be noted that on all bar one of the real-world test cases EPSOC achieved the best objective value. For the one case (Crack 2) on which it did not, it came within one part in 5000 of the best result. On half of the cases it was also the fastest algorithm to achieve the best result.

To illustrate the time behaviour of the algorithms, the median objective function value achieved after each iteration of each algorithm across all 10 runs on each of the real-world test cases are plotted in Figures 6.1-6.6. The convergence history of algorithms on Rosenbrock's function is not shown.

Looking at the convergence history for the Laser 1 test case in Figure 6.1, it may be seen that virtually all algorithms converge, with varying degrees of rapidity, to median values at or near the dominant global minimum. This might be expected for such a "smooth" problem. There were two exceptions with poorer results, the reasons for which were found by inspection of numerical experiment log files:

	Laser 1	*Laser 2*	*Crack 1*	*Crack 2*	*Aerofoil*	*Bead*	*Rosenbrock*
P-BFGS	37	39	26	12	26	18	10
NM	24	24	25	23	20	16	54
NML1	15	24	13	12	18	13	48
MDS	16	**12**	13	1000	14	7	1000
RSCS	22	**12**	9	42	**7**	12	**39**
RSCSL	22	24	15	**27**	18	11	88
RSCSL1	13	17	11	8	15	5	33
EPSOC	**10**	**12**	**5**	20	20	14	14
GA	14	19	16	11	9	**7**	23

Table 6.2 Time taken, as ESFE, to achieve best objective functions values across 10 runs

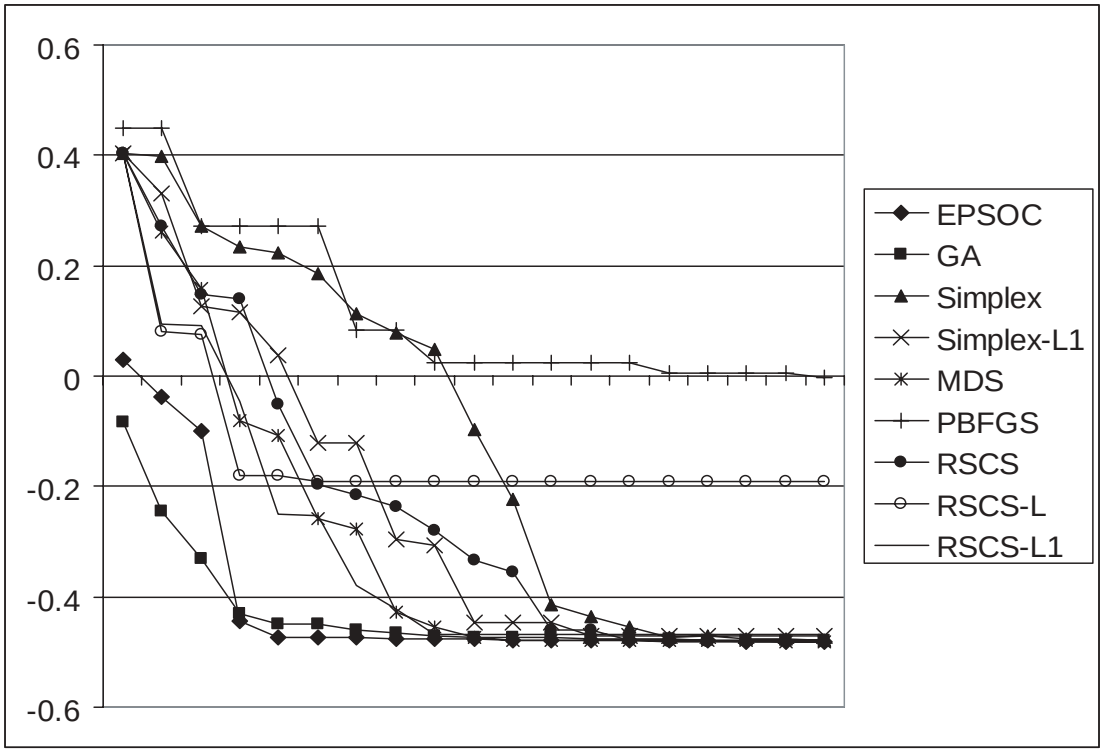


Figure 6.1 Convergence history of median values – Laser 1

- The gradient descent method, P-BFGS, has a final median value that is degraded by several runs that became trapped in the local minima that can be seen in Figure 3.4.
- Similarly, the line-search variant of RSCS had a number of runs that also became trapped in these local minima.

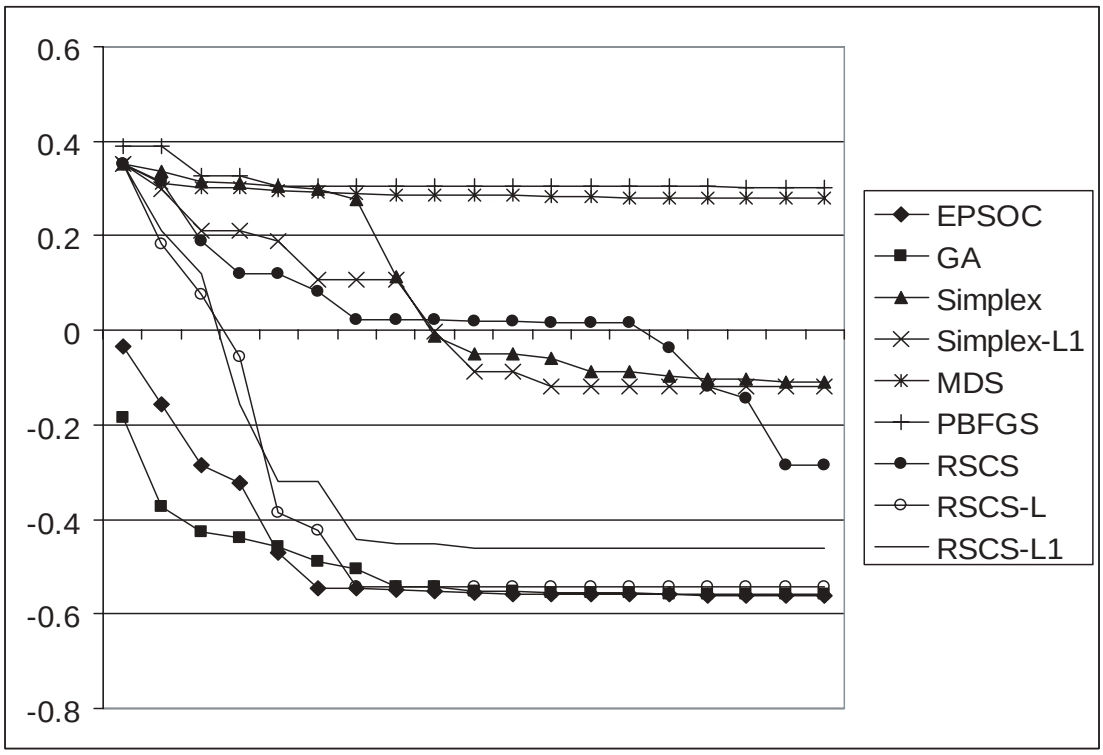


Figure 6.2 Convergence history of median values – Laser 2

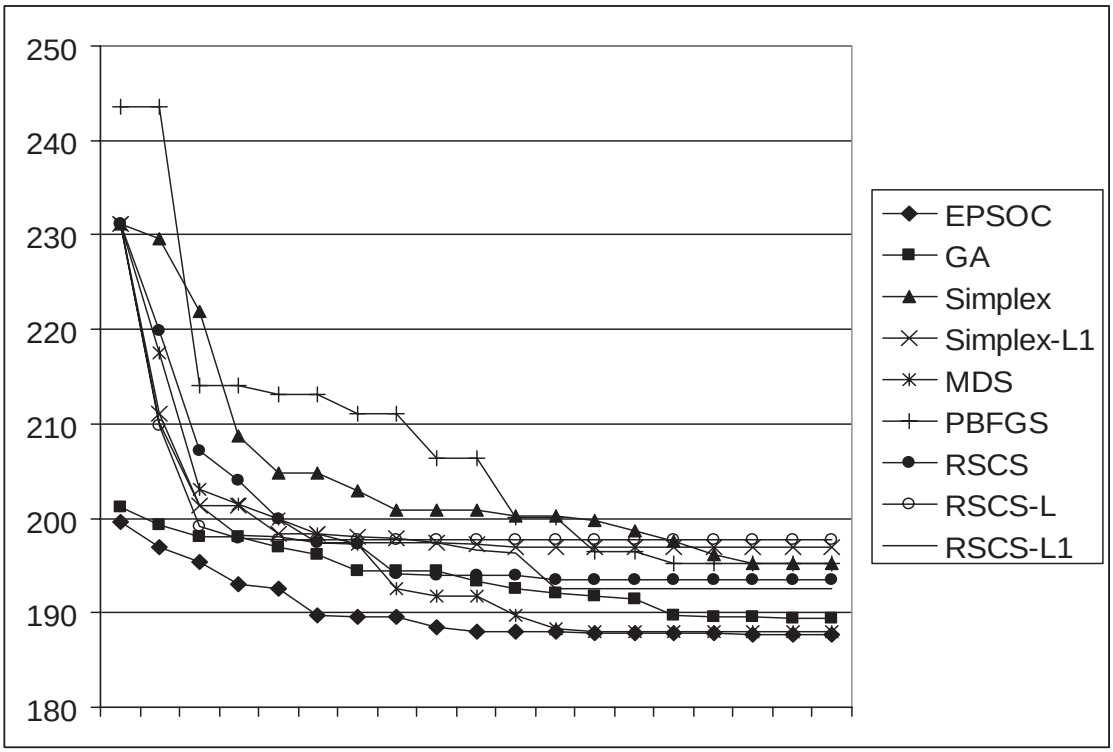


Figure 6.3 Convergence history of median values – Crack 1

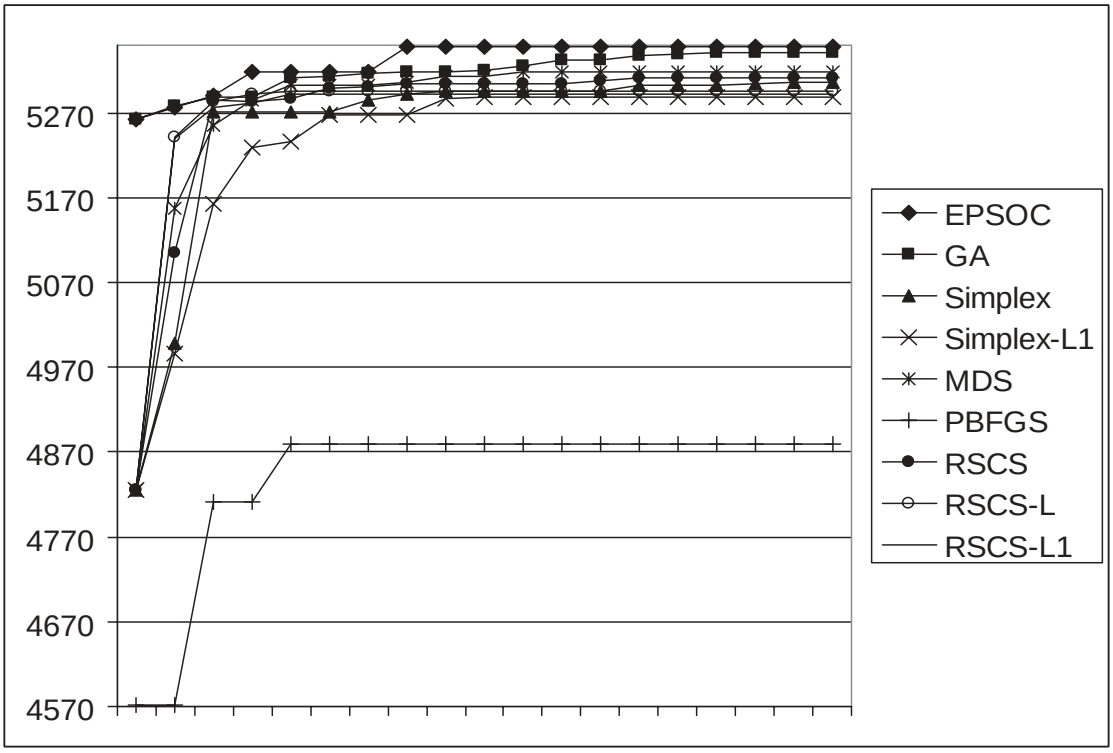


Figure 6.4 Convergence history of median values – Crack 2

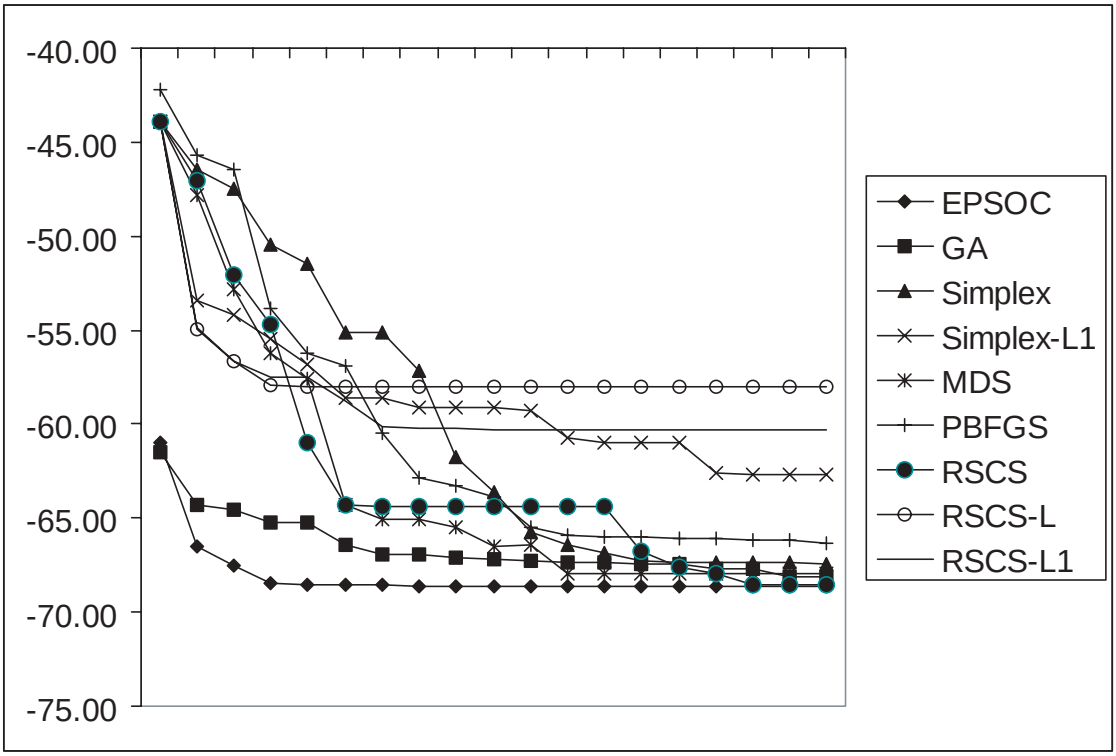


Figure 6.5 Convergence history of median values – Aerofoil

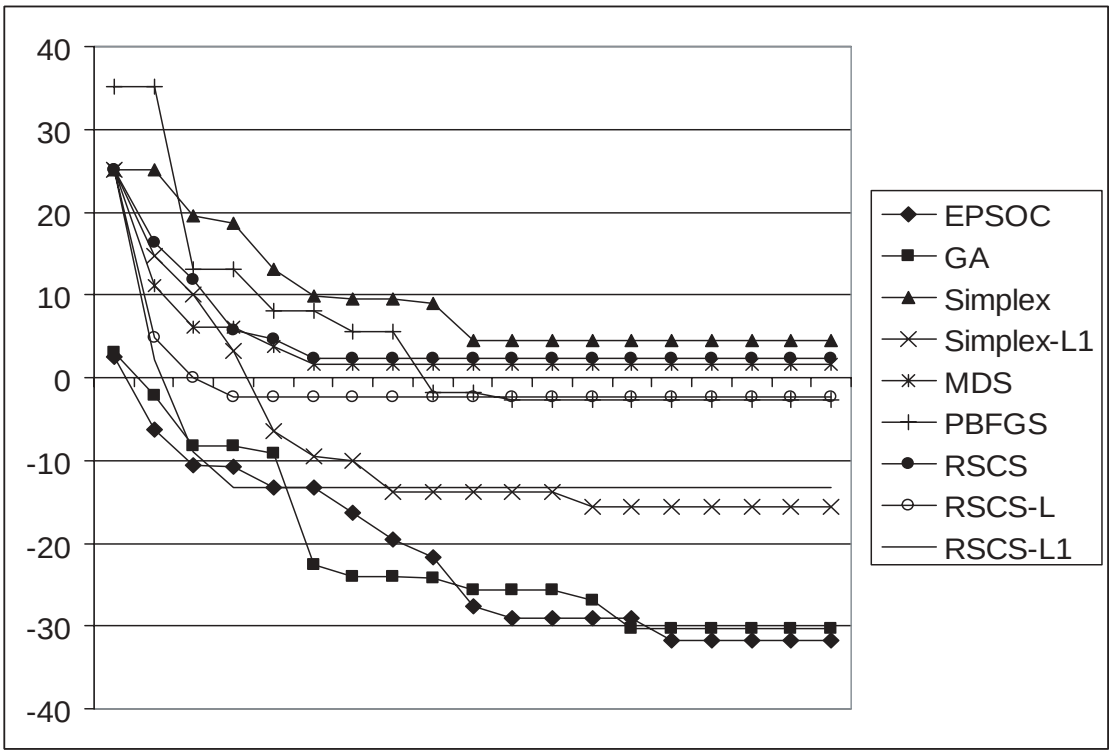


Figure 6.6 Convergence history of median values – Bead

EPSOC and the GA show the benefit of larger sampling populations. The "starting" median values, i.e. the median of the best points returned from the first generation sampling, are better than for other algorithms. All the direct search algorithms start from the same initial simplices and thus have the same initial median value. Even the small sampling of the multiple vertices of the initial simplex confers a slight advantage over the single point evaluation of the gradient descent algorithm.

The median values of EPSOC and the GA are also the most rapid to approach the global minimum. They do this, however, at the expense of significantly greater numbers of objective function evaluations. Without *a priori* knowledge of the problem, it cannot be assumed that this rapidly attained result is the final, best value. A comparison across all the problems (by inspection of the remaining Figures 6.2–6.6) shows the EPSOC and GA values can "plateau" at intermediate stages. (This is particularly evident on, for example, the Bead and Crack problems.)

Turning to the next of the "smooth" problems, inspection of Figure 6.3 shows performance on the Crack 1 problem generally similar to that of the Laser 1 problem in Figure 6.1. There is a slight spread of the final values across all algorithms, but no significant outliers. To explore the reasons for this a further inspection of the shape of the objective function in parameter space is helpful. Figures 3.8 (and also Figure 3.9) show isosurfaces that are generally less sensitive to changes in the value of bias, β. If the objective function is plotted for a constant value of β, as shown in Figure 6.7, a large flat surface adjacent to the global optimum can be seen. It is thus a simple matter for most algorithms to achieve a "reasonably good" result, giving rise to the uniformity of final results seen in Figure 6.3. Of all algorithms, EPSOC achieves the best result earliest.

Considering results for the remaining problem classified in Section 3.7 as a "smooth" problem, the Aerofoil test case shown in Figure 6.5, similar features can be seen:

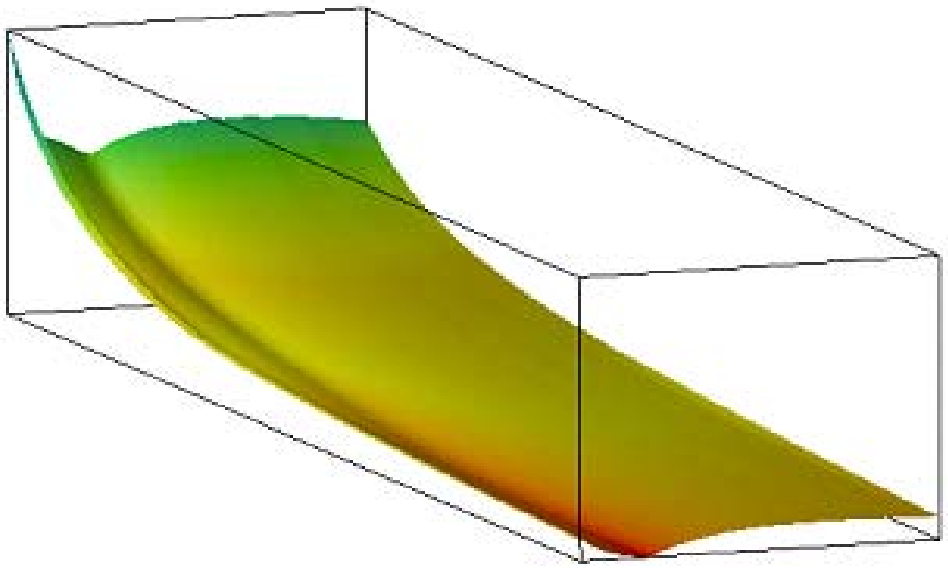


Figure 6.7 Crack 1 isosurface for constant β

- Separation of starting values, with population-based methods leading direct search methods and P-BFGS the poorest starter.
- Clustering of final median values near the global minimum.
- EPSOC the most rapid to achieve the best results.

The line-searching variants of simplex-based methods all fared poorly on this test case. As discussed in Section 5.2, all these variants can yield degenerate simplices, particularly when the local gradient field directs them strongly toward a domain boundary.

In summary, very similar relative behaviour of algorithms was observed on the test cases classified as “smooth”. EPSOC consistently delivered better results faster than other algorithms. However, EPSOC and the GA achieve a median result comparable to the remaining algorithms, and only slightly faster. Without knowledge of the problem, they cannot be assumed to have definitely converged in any less time than the other algorithms and, as has already been noted, utilise many more function evaluations. For these reasons, direct search and gradient descent methods may be seen as preferable for use on “smooth” problems, particularly if the available parallel or distributed computing resources are limited.

Inspecting the median results for the problems classified in Section 3.7 as “noisy” (Figures 6.2, 6.4 and 6.6), there are also similarities in algorithm behaviour:

- The separation and ordering of initial values is the same as observed for “smooth” problems, as is to be expected, since this is purely a function of the initial sampling.
- EPSOC and the GA consistently provide better results more rapidly.

However, except for the Crack 2 test case, there is not the same clustering of algorithms yielding good final results.

Figure 6.8 shows the Crack 2 objective function values plotted for constant β. As was seen for the Crack 1 test case, there are large regions of reasonably good values adjacent to the global maximum. This explains the bunching of algorithms observed in Figure 6.4. In contrast to the Crack 1 test case where these flat surfaces sloped toward the minimum, the flat surfaces in the Crack 2 test case have almost zero gradient. There are also regions of flat, low gradient surfaces at significantly worse function values. As a result P-BFGS, the gradient descent algorithm, fared much worse than others.

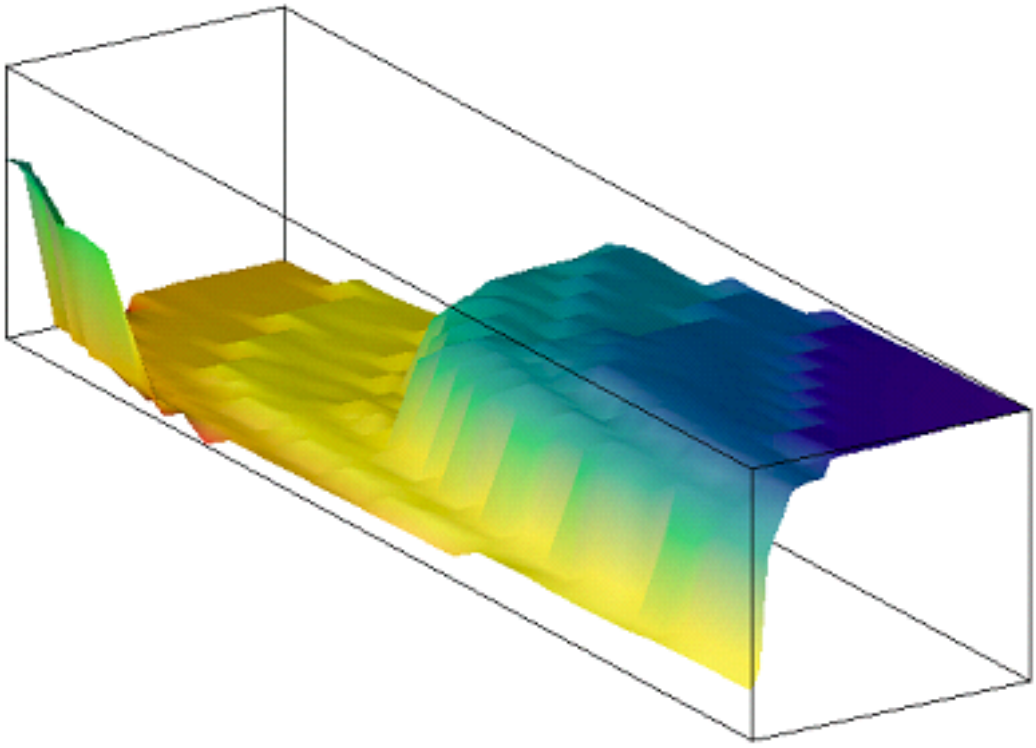


Figure 6.8 Crack 2 isosurface for constant β

On the Laser 2 and Bead test cases EPSOC and the GA in general gave much better results than the other algorithms. The next best were line-searching variants of simplex-based algorithms, with all the remaining algorithms demonstrating similar behaviour to each other.

One point worth noting is that on the Laser 2 test case P-BFGS and MDS both had a tendency to become trapped in local minima if the starting point was remote from the global minimum.

These regions general have a very small "underlying" or average gradient, i.e. they are flat with superimposed, fractal noise dominating the local gradients.

P-BFGS, which is heavily dependent on the local gradient field, performed poorly as a result. MDS, which imposes congruency on successive simplices, was prone to early general contraction of the simplices when search directions failed to reliably return improved points on the generally flat, noisy surface.

In summary, EPSOC and the GA both demonstrated an ability to return much better results more rapidly than other methods on the "noisy" test cases. This amply justifies the additional investment in greater numbers of function evaluations, provided sufficient concurrent computing resources are available.

Comparing the performance of all algorithms on the basis of the convergence history of median results obtained, it may be noted that EPSOC achieved an equal or better median value than any other algorithm on all test cases. In 3 of the 6 cases, the median result obtained by EPSOC lay at the global minimum. On several cases it was also significantly faster than other methods.

6.4 Statistical Analysis of Results

For EPSOC and Genesis, the Kruskal-Wallis H test statistic was computed across results for different operational parameters, for each test case. Results are tabulated in Appendices F and G. The sum of ranks for each group of results for a specific set of operational parameters was compared against the pooled ranks to determine the best set of parameters to use for each test case. At a significance level of 0.05, the percentage point of the χ^2-distribution for the 9 parameter treatments tested, i.e. 8 degrees of freedom, is 15.507. By inspection of H test values, EPSOC has statistically significant differences in its performance ($H > \chi^2_{.05}$) for different operational parameters settings in only two cases, Laser 1 and Aerofoil. In contrast, Genesis has significantly different performance for different operational parameters settings on all test cases. This is in direct contrast to previous comparisons of Evolutionary Programming and Genetic Algorithms (Keane 1996) and indicates a distinct advantage for EPSOC.

To select effective values of operational parameters would generally require some *a priori* understanding of the nature of the problems, and the performance of the algorithm. Close inspection of the results for Genesis, in Appendix G, may indicate some general trends; for example, with lower rates of crossover it appears beneficial to use higher rates of mutation. But in almost all cases best results were obtained with values for these parameters other than their default settings, and no firm rules apply to all cases. The median results obtained on the radio-frequency design case for different settings for Genesis (Table G) shows the difficulty should only one parameter be given the wrong value. Without a detailed understanding of problem and algorithm, multiple trials are needed to ensure the quality of results. This comes at a cost of additional time required, or significant additional computing resources if it is proposed the several permutations be run simultaneously. The relative insensitivity of EPSOC to operational parameters makes it far easier to apply with confidence in the results delivered.

Once the best-tuned EPSOC and Genesis for each test case had been chosen, their results were compared with those of the remaining algorithms. The Kruskal-Wallis H test statistic was computed across results for different algorithms, for each test case. Results are tabulated in Appendix H. For the 9 algorithms tested, comparison of the H test statistic with the percentage point of the χ^2-distribution, at a significance level of 0.05, indicates that the results of at least two of the algorithms differed in their distributions, for all test cases. Put simply, at least one algorithm in each test performed significantly better than others.

Inspection of the rank sums for each algorithm shows that EPSOC was the highest ranked algorithm in 4 of the 6 test cases. In the Bead test case the GA performed marginally better, and the MDS algorithm was marginally better in the Laser 1 test case. In 3 of the remaining cases the GA was ranked second to EPSOC. Simplex ranked second to EPSOC in the Aerofoil test case.

To examine the relative performance of algorithms in more detail, a pair-wise comparison between the first and second ranked algorithms, using the Mann-Whitney U test, was performed for each test case. In the 4 cases in which EPSOC was the best algorithm, it was shown to be significantly better than the next best algorithm, to a significance level of better than 0.05. In the two cases in which another algorithm yielded better results than EPSOC, the difference was statistically insignificant.

To qualitatively evaluate algorithm performance in terms of speed, the time to reach the best objective functions values, measured in ESFE as shown in Table 6.2, was averaged over all the test cases. EPSOC proved to have the fastest mean time to achieve these results, at an average 13.6 iterations. The mean of the averaged times of the remaining algorithms, excluding the outlying results for MDS which skewed the runtime performance, was approximately 21 iterations. It was this result, derived from earlier testing, that was used to determine the "maximum iteration" completion criteria for EPSOC and Genesis.

6.5 EPSOC and Multi-Objective Optimisation

When tackling real-world problems, particularly in the field of engineering design, the desired optimal design may not be expressed in terms of a single objective. Product designers may wish to maximise some element of the performance of a product, while minimising the cost of its manufacture, for example. Different objectives may be conflicting, with little *a priori* knowledge as to how they interact. In the past, common practice was to optimise for a single objective while applying other objectives as constraints, a less than ideal approach. Evolutionary Algorithms (EAs) have recently become more widely used for their ability to work well with large-scale problems (Van Veldhuizen 1999).

In general, there remains a role for a human Decision Maker (DM) in choosing between competing objectives. Multi-objective optimisation algorithms can broadly be categorised by when in the optimisation process the DM intervenes (Zitzler 1999, Van Veldhuizen 2000):

- **Decision making before search**: the DM aggregates the objectives into a single objective, including preference information (or weights). The problem is essentially reduced to a single objective optimisation.
- **Decision making during search**: the DM interactively supplies preference information to guide the search
- **Decision making after search**: the DM selects from a set of candidate solutions resulting from the search.

It may be noted that the first two of these approaches would seem to require some *a priori* knowledge of the problem domain in order to effectively provide preference information, particularly for methods involving aggregation. It has been asserted that the first approach does not return Pareto-optimal solutions in the presence of non-convex search spaces (Coello Coello 1998, Zitzler 1999). The multi-objective EPSOC method to be described (EPSOC-MO) largely falls into the last category.

EAs applied to multi-objective optimisation need to address two main problems:

1. How is fitness assignment and selection performed?
2. How is a diverse population maintained, to aid search space exploration?

Fitness assignment can conceptually be divided into a number of approaches (Zitzler 2002): ,

- Aggregation-based
- Criterion-based
- Pareto-based

A great deal of recent work has concentrated on using pareto-based selection (see Coello Coello (1999), Zitzler (1999) or Zitzler et al. (2000)). EPSOC-MO does not make explicit use of pareto dominance, selection being made according to rankings based on single objective function values, and their combination. It does not do this, however, by aggregation of the objectives. In approach, EPSOC-MO can be seen as a hybrid of criterion-based and Pareto-based fitness assignment and selection. Fitness is determined by objective function values in turn, a criterion-based assignment. But selection, for extinction, is tempered by consideration of ranking against all objectives, an implicit Pareto-based approach.

Diversity of the population is addressed in EPSOC-MO by the same means as for its use for single objective optimisation, i.e. achieving a self-organised critical state through operations of mutation and extinction. Of the common methods (Zitzler 1999):

- Fitness sharing
- Restricted mating
- Isolation by distance
- Overspecification
- Reinitialization
- Crowding

the method of species extinction in EPSOC-MO is closest to a form of reinitialization, though of a carefully selected section of the population.

EPSOC operates on an ordered population, ranked by fitness according to the objective function. However, for it to be used for multi-objective optimisation it would not be possible to sort the population according to two or more objectives simultaneously. The modified EPSOC-MO instead maintains an ordered set for each objective, with a mapping into the original population. An example for two objectives is illustrated in Figure 6.9.

Referring to the outline of the EPSOC algorithm in Section 6.2.1, half of the *nbad* members selected come from the worst members of the first ordered set, and the remainder from the worst members of the second ordered set, *with the proviso* that those chosen from the first set not be among the elite of the second set and *vice versa*.

To implement this restriction, a set of reverse mappings, from the original population back into the ordered sets, was also maintained. Their use is illustrated in Figure 6.10. A member chosen from the first set, for example, would be mapped into the original population, and then into the second set. It would only be chosen for extinction if it were not in the elite of either set. For the

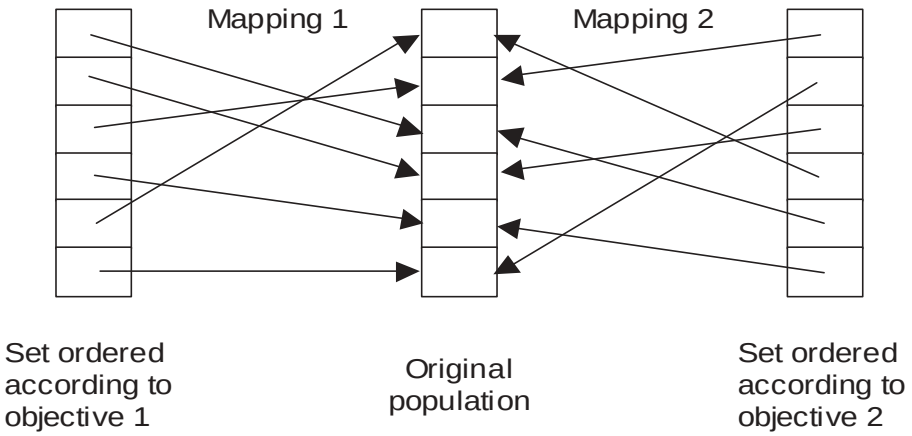


Figure 6.9 Ordered set mapping for EPSOC-MO

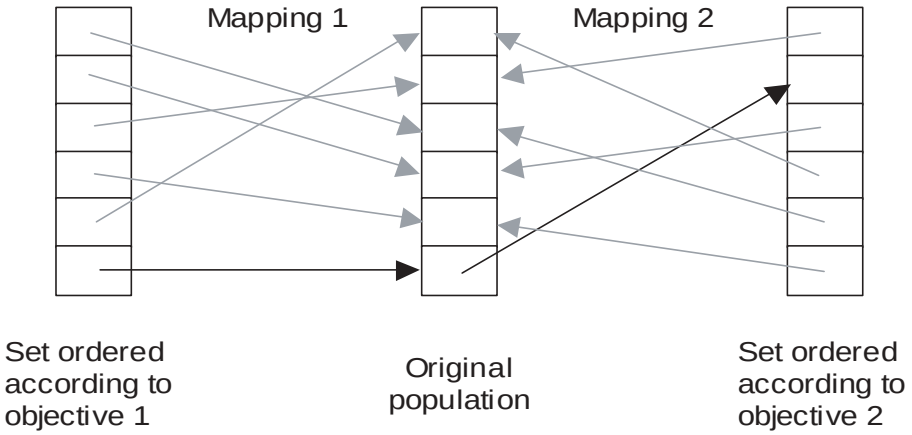


Figure 6.10 Use of reverse mappings in EPSOC-MO

example shown in Figure 6.10, the chosen member from set 1 would be rejected since it is in the elite of set 2. Similar restrictions are imposed on the choice of nearest neighbours at step 3 of the algorithm. The elites of each set were chosen so that their sum was the size of the elite of the single objective implementation of EPSOC.

The use of archives to preserve the Pareto-optimal set is commonplace in multi-objective evolutionary algorithms (MOEAs) (Zitzler 2002) and remains an area of active research (Laumanns, Thiele, Deb and Zitzler 2002). Most algorithms use the archive as a form of “non-volatile” storage, serving only to preserve Pareto-optimal approximations. Only a few use the archive as a dynamic element in the algorithm (Knowles and Corne 1999).

In contrast, in EPSOC-MO the ranked sets play an integral role in the operation of the algorithm. These sets not only preserve the approximations to the Pareto-optimal set, but also mediate between objectives during the search. In this regard they can be considered as playing a role in the decision-making process; in effect *decision making during search* but without the interaction of the Decision Maker. In a sense, through these sets EPSOC-MO maintains multiple, "virtual" archives.

6.5.1 Results of Numerical Experiments

A simple experiment was constructed by formulating a second objective for the Bead test case. Not only was gain to be optimised, but the length of the optimal bead was to be minimised. This is in some sense a realistic goal, in that a shorter, more compact component might be desirable in practice. Ten runs of the multi-objective implementation of EPSOC were made using these objectives. The maximum iteration for the run was set to 500, and a population of 64 was used. The length of the bead was permitted in the range 35mm to 60mm and, as has been shown in earlier numerical experiments, the range of the gain values was from –39.85 to 48.52.

Gain and length values for the points returned at the top of the set ordered by gain for each run are shown in Table 6.3. The median gain for this set of points is –22.35, compared with a median of 34.99 for the whole dataset. The median length of these returned points is 48.1, compared with a median of 47.5 for the whole dataset.

Gain	*Length*
-12.27	50.07
-26.98	48.44
-27.55	51.77
-14.35	41.37
-20.41	41.71
-30.42	51.68
-7.73	41.34
-35.57	51.97
-10.89	42.98
-24.29	47.79

Table 6.3 Gain and length of "best" points from 10 runs

The entire decision space was scanned at integer intervals, and the results are plotted in objective space in Figure 6.11 to illustrate the distribution of solutions in objective space, and give some indication of the location of Pareto-optimal solutions. It should be noted that the test case uses these sample points and linear interpolation between them to provide objective function values, i.e. these are extremal points. The "best" returned values are highlighted. By inspection, it can be seen that the returned points approach the Pareto-optimal front of minimal gain values combined with minimal length.

The top-ranked points in the gain-ranked virtual archive in any particular run also approach the Pareto-optimal front. The top ten points from a representative run are shown in Figure 6.12, superimposed on the same parameter sweep data as Figure 6.11.

In order to compare the effect of optimising for multiple objectives, another 10 runs were performed, the only change being the removal of the second objective function (relating to the length). The median gain of the set of points returned from these runs was –29.97, and the median length of

the optimised beads was 49.0. The length of the returned beads was purely a product of the location of significant minima in the dataset, of which there are two dominant, one in a region of parameter space corresponding to a length of 52mm which also contains the global minimum, and another at length 42mm. The returned points, superimposed on the parameter sweep data, are shown in Figure 6.13. It can be seen that returned points are clustered at higher gain values, unlike the points in Figure 6.11 which are more spread along an approximation to the Pareto-optimal front.

The median gain returned for the multi-objective test case is neither near that returned for the single-objective, gain-only case, nor near the median of the entire dataset. It lies between and

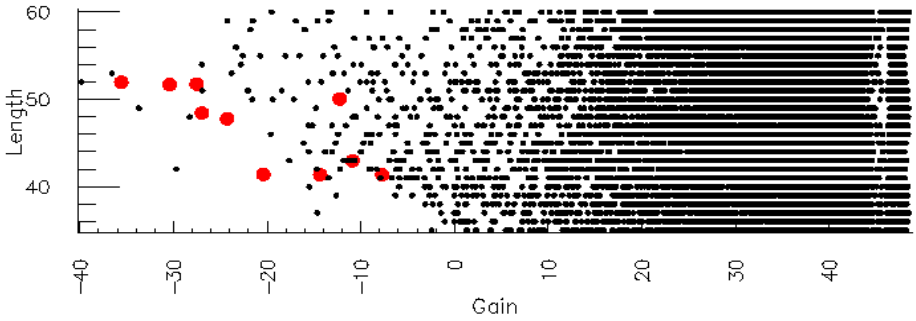


Figure 6.11 Multi-objective test case objective space sampling with “best” points from 10 runs

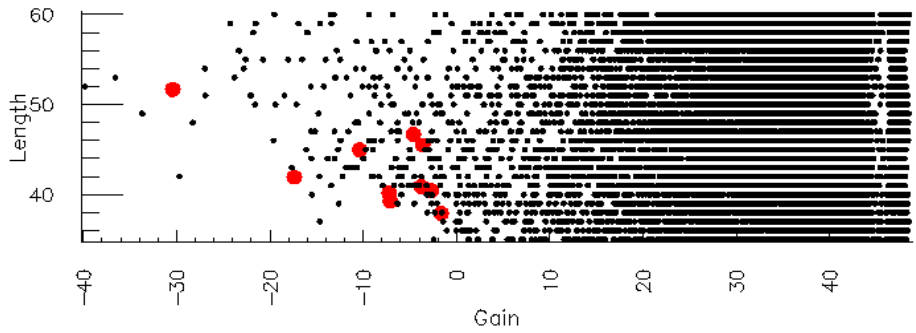


Figure 6.12 Multi-objective test case objective space sampling with points from a single run

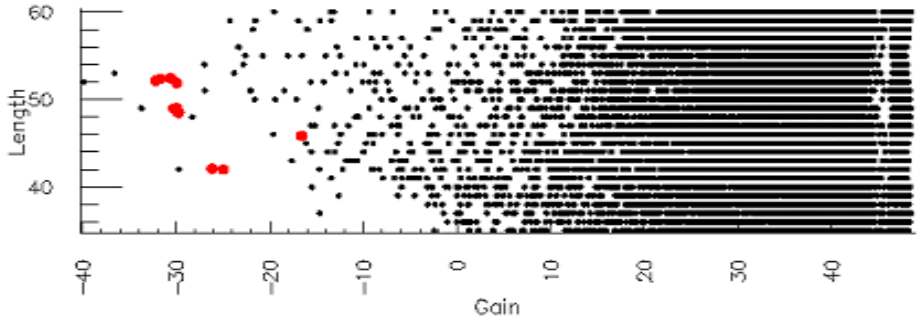


Figure 6.13 Test case objective space sampling with “best” points from 10 runs, optimised for gain only

demonstrates EPSOC-MO is capable of returning a compromise solution, independent of an external Decision Maker supplying preference information after the search.

The median length returned for the multi-objective case lay close to the median for the entire dataset. It could be concluded that behaviour of EPSOC-MO was neutral toward the length objective, but evidence from the single-objective test case indicates this value may also be a compromise between lengths for two solutions of attractive gain in different regions of parameter space.

These are preliminary results, and by no means exhaustive. Several questions remain as to the efficacy of EPSOC-MO for multi-objective optimisation, not the least being the scalability of the method, but the results returned so far are quite promising.

6.6 Summary

A novel Evolutionary Programming algorithm using concepts of Self-Organised Criticality, EPSOC, has been described in this chapter. Tested on a group of real-world problems, it has been demonstrated to deliver significantly better results faster than any of the methods described in previous chapters. It also shows potential for multi-objective optimisation. The multi-objective implementation of the algorithm uses multiple, "virtual" archives both to store optimal results and also to mediate between objectives during selection. A test of the implemented algorithm on a difficult test case with a highly nonlinear objective demonstrated its ability to return reasonable approximations to Pareto-optimal solutions.

The population-base methods of this chapter can make use of far greater parallelism in computers than gradient or direct search methods on the group of problems drawn from scientific research and engineering design processes. For challenging problems, with noisy objectives and multiple local minima about which little may be known, the new EPSOC algorithm proved the clear method of choice.

Looking at the taxonomy in Figure 1.1 we have reached the extremity of concurrency in the stochastic methods of this chapter. To achieve any further improvement in the time to obtain an optimal solution, we must consider a clue given in the Introduction: seeking to reduce the total time taken for optimisation by utilizing more parallel computing resources was necessary *if there were no way to reduce the time taken for a single execution of the numerical simulation*. At no time have we assumed any access to the internals of the simulation to improve its performance, but are there ways the model could be externally manipulated to reduce the time taken to obtain a result? The following chapter will look at two different approaches to this question.

7.0 Preconditioning

The work in preceding chapters has concentrated on reducing the time to achieve optimisation of an objective function by means of iterative methods using concurrent execution of objective function evaluation to a greater or lesser degree. Eventually, however, for a chosen optimisation method applied to a given problem the time taken for its solution reaches some irreducible minimum, determined by the character of the problem and characteristics of the optimisation method such as its degree of concurrency and the computational resources available for its solution.

It is possible that modification of the *problem* may further reduce the time taken to achieve an optimal solution. This work considers two such modifications: those exploiting features of the time behaviour of the numerical simulation, *temporal preconditioning*, and features of the parameter space to which the optimisation effort is confined, *spatial preconditioning*.

The methods used in this chapter to modify optimisation problems can be considered in the context of systematic use of surrogates to reduce the computational cost of the optimisation process. Torczon and Trosset (1998) make the careful distinction between "surrogate models" or auxiliary simulations of less physical accuracy but corresponding less computational cost, and "surrogate approximations" in which algebraic summaries of the response of the physically accurate, computationally expensive models are compiled and used instead of more evaluations of the models. Several other authors use the terms interchangeably, primarily referring to approximations. Current, standard engineering practice is to use approximations (Dennis and Torczon (1996), and see also Booker, Dennis, Frank, Serafini, Torczon and Trosset (1998), Büche, Schraudolph and Koumoutsakos (2003), El-Beltagy and Keane (2001), Liang, Yao and Newton (2000), and Rodríguez, Pérez, Padmanabhan and Renaud (2001)) and the widespread Response Surface Methodology discussed in Section 3.7.1 is founded on this approach. For the reasons stated in Section 3.7.1 the use of "surrogate approximations" is unattractive: the formulation of algebraic approximations is difficult and prey to the "curse of dimensionality" of the problem, as Torczon and Trosset (1998) admit.

A smaller number of authors have investigated "surrogate models" (Bakr, Bandler, Madsen, Rayas-Sánchez and Søndergaard 2000, Vitali, Haftka and Sankar 1999, Alexandrov and Lewis 2000) but generally rely on the explicit construction of these models using detailed knowledge of the problem. In contrast, both the methods described here can be considered "surrogate models", but do not require highly specific knowledge of the problem domain. Temporal preconditioning can be readily understood as a surrogate model or low-fidelity model achieved using user specified solution tolerance, as categorised by Alexandrov and Lewis (2000), though their experiments were confined to the two other categories they mention, multigrid and reduced fidelity physics. Temporal preconditioning also uses a single, step change in accuracy at convergence, similar to the consistency requirements suggested by Alexandrov and Lewis (2000). In the following sections attention is directed to investigating the predictive capability of this surrogate method.

Spatial preconditioning can be considered a surrogate model if the "model" is taken to be the combination of the numerical simulation and the range of the model parameters. The method constructs a sequence of models with gradually differing properties. There are elements of similarity in the method with the approach of Pérez and Renaud (2000). The motivation behind the method is that the final, resulting model will prove quicker to solve.

7.1 Temporal preconditioning

For many problems of practical interest to science and engineering, evaluation of the objective function itself involves an iterative process. Trial solutions to ordinary or partial differential equations are generated and refined, until a solution of some predetermined, acceptable accuracy may be reached. These can be computationally intensive, time-consuming processes. If it were possible to reduce the accuracy with which a trial solution is known but still correctly determine the location, in parameter space, of a minimum of the objective function, the need for several iterations of the numerical simulation might be obviated and the overall time to achieve an optimal solution considerably reduced. This section describes numerical experiments simulating temporal preconditioning applied to two case studies: the Aerofoil case and a multi-dimensional problem in the design of a novel antenna for microwave frequency communications.

As an example to clarify the method of temporal preconditioning, consider the prototypic elliptic equation, the *Poisson* equation in 2 dimensions, which can be written:

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = \rho(x, y)$$

When discretized on a computational grid, the resulting system of equations can be represented in matrix notation as:

$$A \cdot u = b$$

Successive approximations to the matrix of coefficients, **A**, are substituted and a trial balance made with the known information about the problem, for example fixed conditions at the boundary of the model domain, encapsulated in **b**. From an initial guess, the difference between terms on right and left sides of the system of equations is termed the *residual error*, which may itself be used to correct the values in **A**. In a converging solution, the magnitude of the residual error should, on the whole, be progressively decreasing. A question can then be posed:

- At what point is the residual error sufficiently reduced, that the output from the numerical simulation can be used in an optimisation procedure to correctly determine the location of a minimum within the parameter space?

To explore the hypothesis that a numerical simulation truncated before full convergence can perhaps guide optimisation toward the correct region of a local or global minimum, a prototype method was trialled on two real-world problems. The two cases investigated were the Aerofoil test case previously described, and a computationally intensive multi-element antenna simulation. The results are discussed below, for each case.

7.1.1 Case study 1: the Aerofoil case

As outlined in Section 3.4, the Aerofoil case study uses the FLUENT computational fluid dynamics package for evaluation of the objective function. A detailed simulation is made of fluid flow around the 2-dimensional aerofoil whose shape is determined by the case study parameters. This simulation requires the iterative solution of five major equations:

- Momentum equations in the two Cartesian dimensions of the problem
- A continuity equation ensuring conservation of mass

- Two equations modelling the time-averaged turbulence of the flow, deriving the kinetic energy and kinematic rate of dissipation of the turbulence.

Each of these equations is required to have a relative residual error or not more than 1.0×10^{-4}. The model is allowed 10,000 iterations to converge.

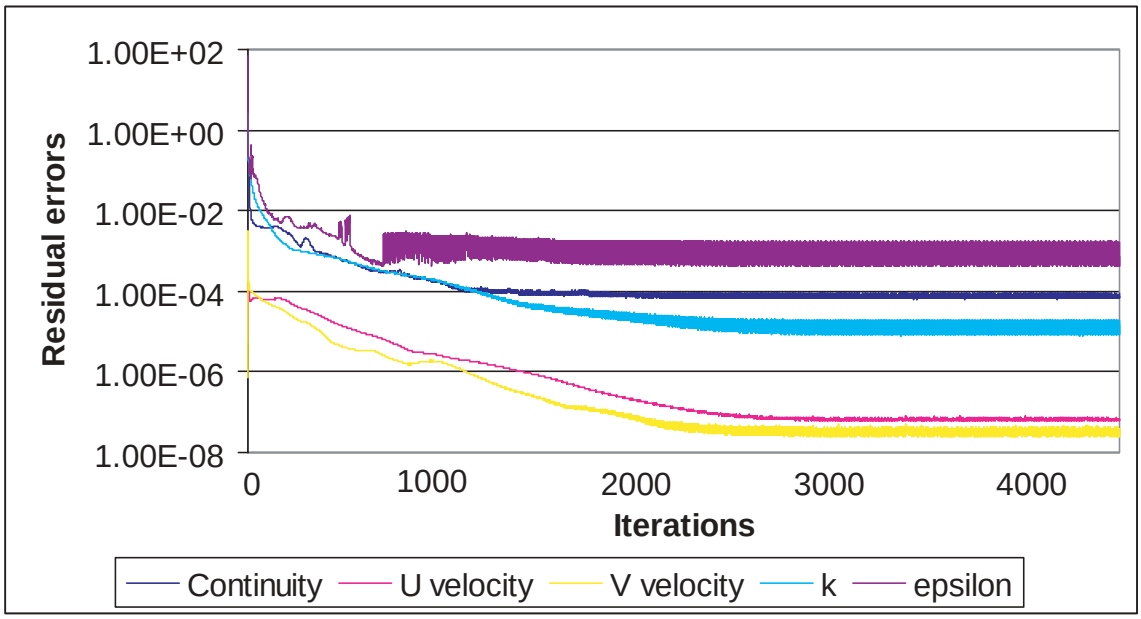


Figure 7.1 Time evolution of residual errors – Aerofoil test case

Figure 7.1 shows the evolution of the residual error for the 5 equations, from 1 to 4200 iterations, plotted on a logarithmic scale. The equations of fluid motion rapidly converge, followed by the turbulence kinetic energy, mass conservation and the turbulence kinematic dissipation coefficient, epsilon, is slowest of all.

The time taken for all equations to converge is obviously quite problem-dependent. Different parameter settings will construct different problems, for which solution time can be expected to be different. For example, the angle of attack in this problem may affect the degree of turbulence in the flow over the upper surface. As can be seen in Figure 7.1, and confirmed by close examination of the log files for this problem run to the maximum 10,000 iterations, no major improvement occurred in the residual errors beyond about 2500-3000 iterations. In particular, the kinematic dissipation equation failed to reach convergence at all.

While it is desirable that all equations relating to quantities of interest be fully converged, it is common practice to ensure that the continuity equation has converged, fully converged turbulence quantities only being required if it is anticipated they will have a significant impact on details of the solution. It thus seems reasonable to hypothesise that the lift-drag ratio derived from a model in which the continuity equation has converged, i.e. after about 1500-2000 iterations, will be adequate to direct an optimisation procedure toward a generally correct minimum. It is anticipated that gross features of the local gradient field will be correctly approximated, with only finer details possibly inaccurate. This would represent an improvement in run-time of a factor of about two over the point at which improvement of the solution effectively ceased in this problem, given that the entire process is completely dominated by the run-time of the numerical simulation.

To test this hypothesis, a parameter sweep of the numerical simulation was performed at 250-iteration intervals up to 1000 iterations, and then at 1500 and 2000 iterations. The derived lift-drag ratio isosurfaces are compared with those derived from a sweep performed on results from a fully converged simulation. The results from these experiments are shown in Figures 7.2–7.7, with results from the fully converged simulation shown in Figure 7.8 (similarly coloured isosurfaces in different figures are at the same value). Examination of these figures confirms that the major features of the objective function in the chosen parameter space derived from a simulation terminated after 1500 iterations are very similar to those derived from a simulation that completed the maximum number of iterations.

In Figure 7.2 the nature of the field in objective space is very simple, almost linear, and shows little more than a tendency for a low angle of attack to yield better results. In Figure 7.3 the "minimum" is already in the correct location, after only 500 iterations, but the field is unstable – by 750 iterations, shown in Figure 7.4 there is a spurious "minimum" appearing for high angle of attack, and large aerofoil thickness. By 1250 iterations, shown in Figure 7.6, this has largely disappeared and the minimum is again settling toward the correct region. By 1500 iterations, shown in Figure 7.7, little major change is occuring and the location of the minimum is essentially the same as for the fully converged case shown in Figure 7.8.

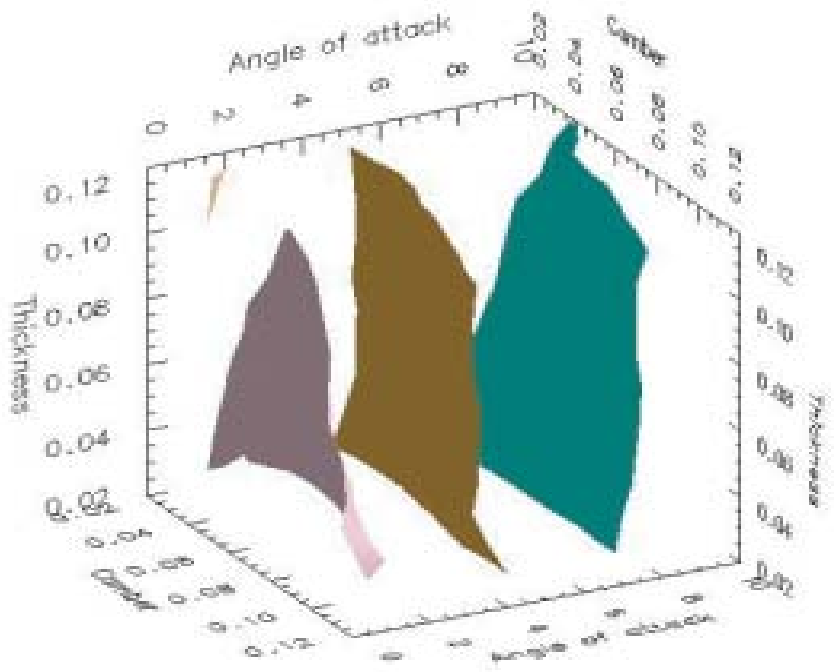


Figure 7.2 Lift-drag ratio isosurfaces at 250 iterations

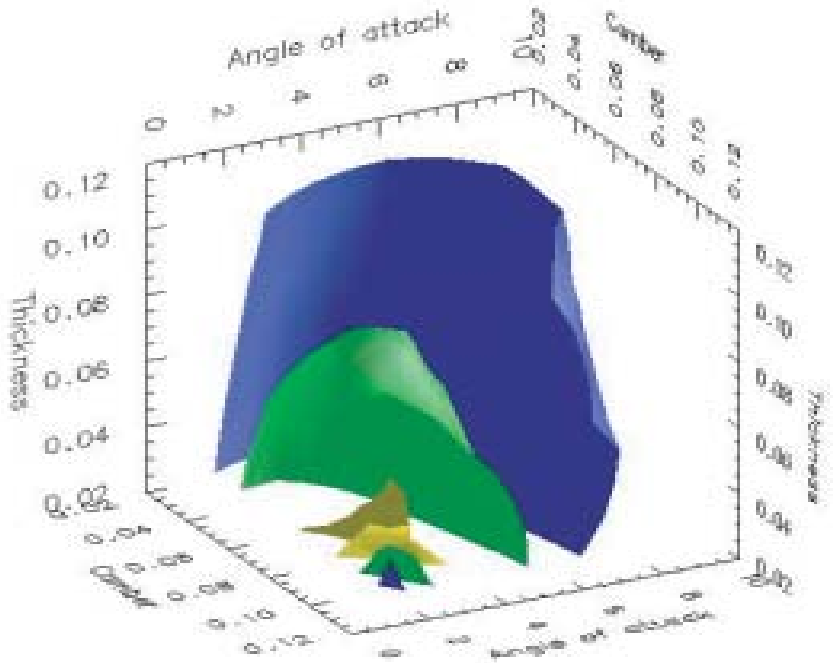


Figure 7.3 Lift-drag ratio isosurfaces at 500 iterations

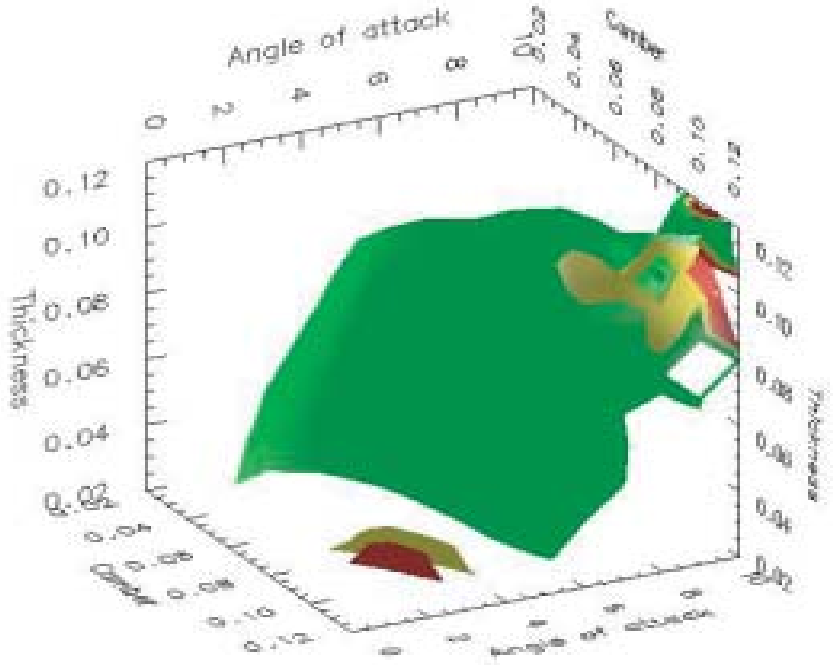


Figure 7.4 Lift-drag ratio isosurfaces at 750 iterations

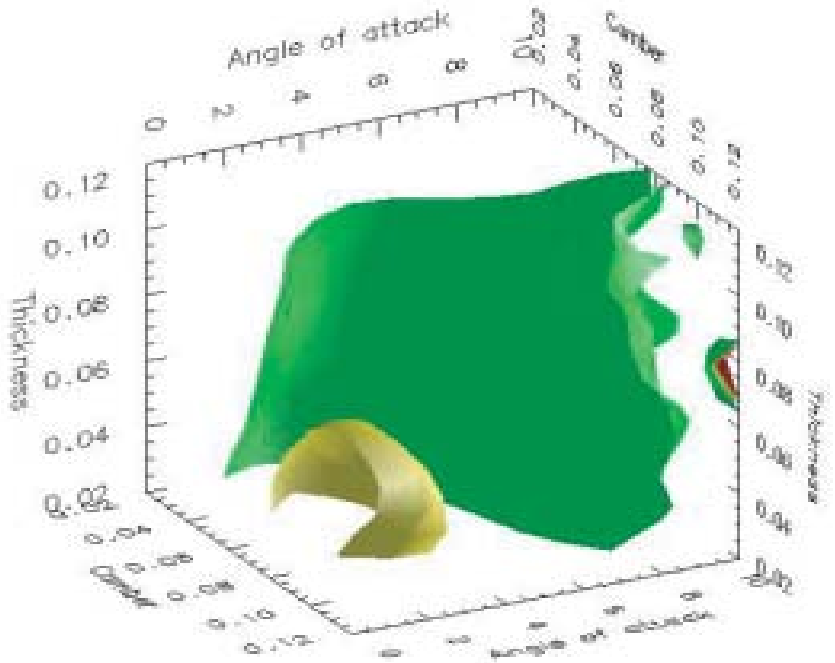


Figure 7.5 Lift-drag ratio isosurfaces at 1000 iterations

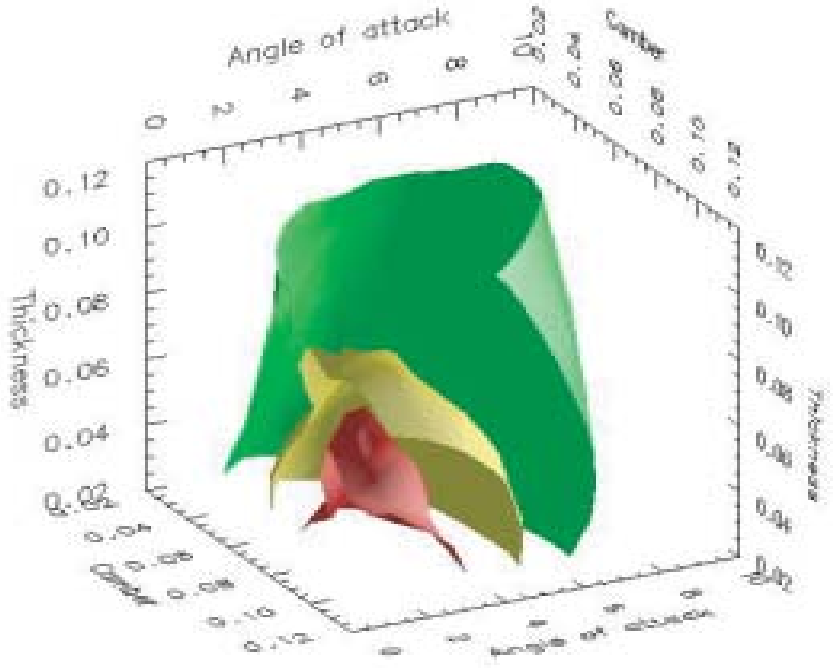


Figure 7.6 Lift-drag ratio isosurfaces at 1500 iterations

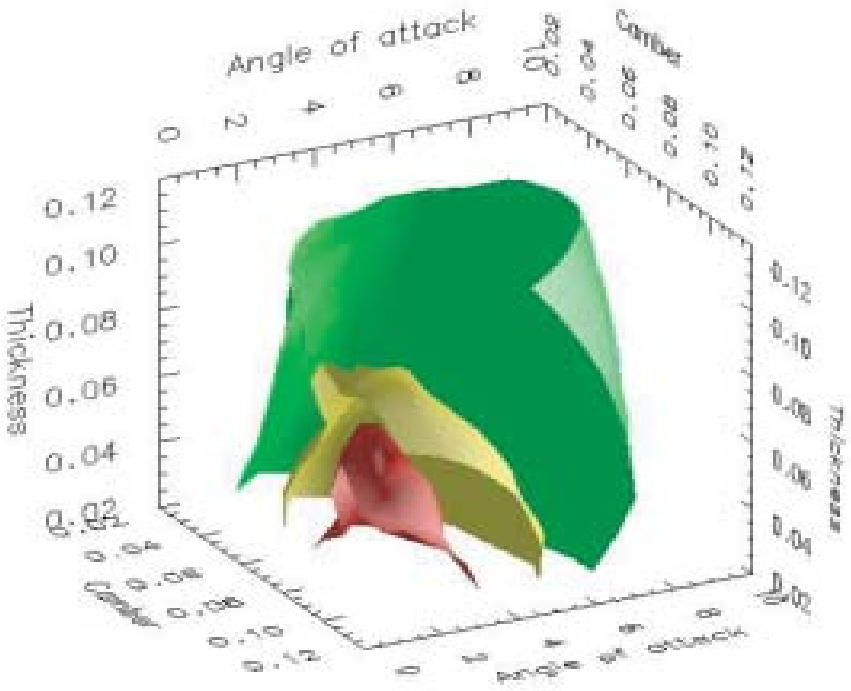


Figure 7.7 Lift-drag ratio isosurfaces at 2000 iterations

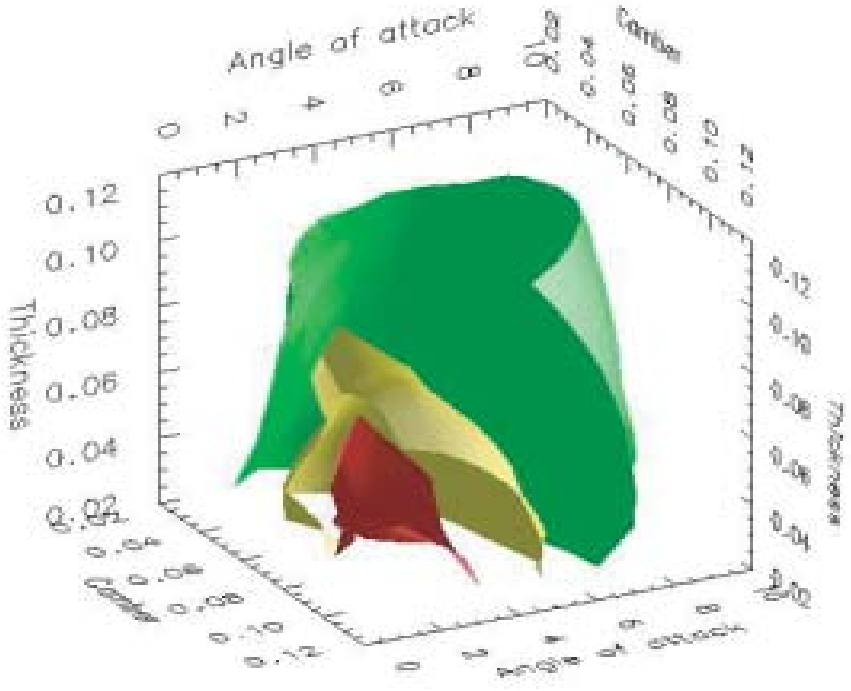


Figure 7.8 Lift-drag ratio isosurfaces at 10,000 iterations

For reference, Figure 7.9 shows the time evolution of the objective function value at a random location in parameter space, as derived after 100 iterations through to 2100 iterations. Once more it appears major changes in the objective function value are complete after about 1500 iterations with minor subsequent refinement. This suggests it is reasonable to assume that an optimisation procedure will correctly identify the location of significant minima within the parameter space using objective function data derived from a simulation terminated after approximately 1500 iterations. This was tested in a series of numerical experiments using a number of representative optimisation algorithms.

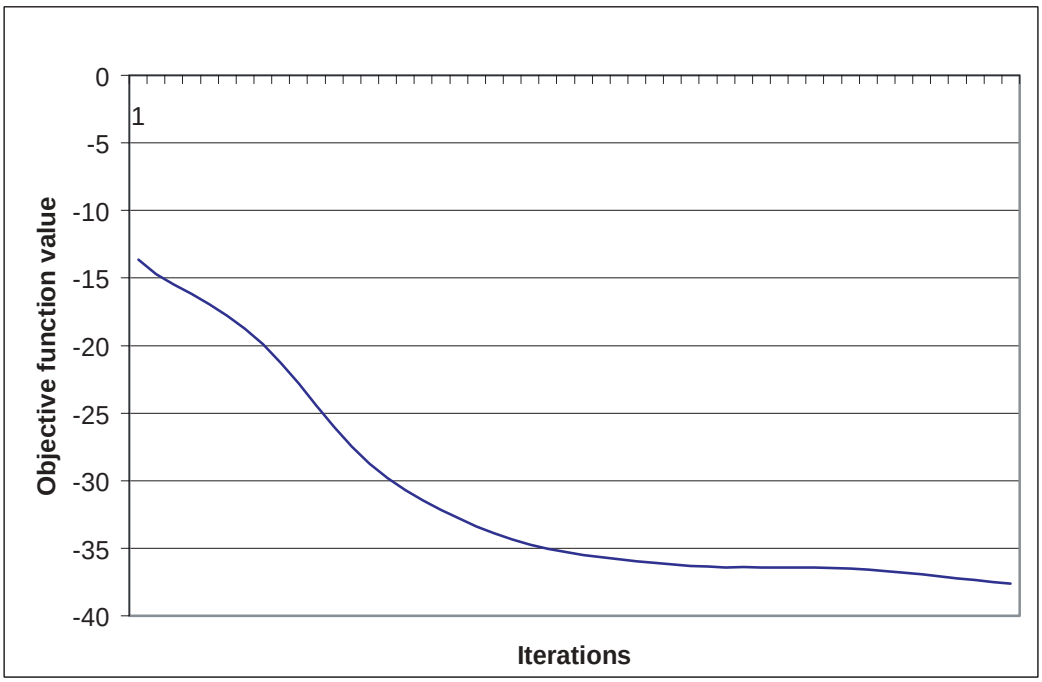


Figure 7.9 Evolution of aerofoil objective function value with simulation iterations

7.1.1.1 Numerical experiments on the truncated Aerofoil case. To confirm that optimisation algorithms can correctly determine the location of significant minima within the parameter space using data from truncated numerical simulations, each of three, representative optimisation algorithms (P-BFGS, RSCS and EPSOC) were run 10 times from random starting points, using data from simulations terminated after 1500 and 10,000 iterations. For each algorithm, a pair-wise comparison was made using the Mann-Whitney U test from each of the two iteration counts. The quantity compared was the dimensionally normalised distance between the endpoints of the optimisation searches and the (known) global minimum of the data set from the fully converged simulation. These results are shown in Table 7.1.

For the P-BFGS and RSCS algorithms there is no statistically significant difference in the ability of the algorithms to find the location of the global minimum for truncated or fully converged simulations. EPSOC, however, shows a statistically significant difference in the endpoint locations. On closer inspection, it may be noted that EPSOC endpoints are much more tightly clustered around the location of the global minimum in each of the two data sets than are the other algorithms: EPSOC is very effective at finding the global minimum. For the data set using truncated simulation, the global minimum is actually slightly displaced from the global minimum from the fully converged simulation. It is this “systematic error” that causes the observed differences. It may be noted that

the distance between the two global minima is less than the spread of distances returned by both P-BFGS and RSCS, i.e. the magnitude of the systematic error of EPSOC is less than the magnitude of random errors of the other two algorithms.

	P-BFGS		*RSCS*		*EPSOC*	
	1500	*10000*	*1500*	*10000*	*1500*	*10000*
	0.0255	0.0322	0.0702	0.1203	0.0102	0.0008
	0.0221	0.0245	0.0545	0.0702	0.0102	0.0007
	0.0172	0.0180	0.0437	0.0457	0.0101	0.0004
	0.0100	0.0172	0.0373	0.0331	0.0100	0.0004
	0.0099	0.0100	0.0340	0.0012	0.0099	0.0004
	0.0099	0.0071	0.0100	0.0005	0.0098	0.0003
	0.0099	0.0040	0.0100	0.0005	0.0092	0.0003
	0.0099	0.0002	0.0100	0.0004	0.0092	0.0003
	0.0098	0.0001	0.0099	0.0003	0.0091	0.0002
	0.0098	0.0000	0.0099	0.0002	0.0090	0.0002
Rank sum	96	114	87.5	122.5	55	155
U	41		32.5		0	
P	0.5288		0.2176		0.0000	

Table 7.1 Statistical comparison of distances to fully converged global minimum

A simple comparison of the truncated simulation iteration count of 1500, and the original iteration count of 10,000 might suggest that temporal preconditioning has yielded a speedup by a factor greater than 6. However, it is more realistic to consider that inspection of residual errors might have led an experienced engineer to ordinarily truncate the simulation at the point at which no further improvement in solution could be expected, i.e. at an iteration count of about 3000. The limit of 10,000 iterations originally imposed can be interpreted as being overly conservative; perhaps useful caution to start with, but altered as experience dictated. On this basis temporal preconditioning has yielded a speedup by a factor of about 2. By way of comparison, this is similar to the two-fold savings with variable resolution (multigrid) models reported by Alexandrov and Lewis (2000), also applied to a 2D airfoil aerodynamic optimisation.

7.1.2 Case study 2: a multi-element antenna simulation

This case study involves the design of a 13 element dielectric embedded electronically switched multiple-beam (DE-ESMB) antenna (Lu, Thiele and Saario 2002). Seven of the design parameters require optimisation, making enumeration of possible configurations infeasible. The size and dimensionality of the parameter space also precluded aggregation of sufficient objective function values to allow meaningful approximation by interpolation into pre-computed data. Numerical experiments required direct computation of objective function values, and were consequently limited in number.

The finite difference time domain (FD-TD) simulation of the antenna was custom-written code, and residual error values were not readily available. Comparison of Figures 7.1 and 7.9 in the previous section suggests it may be possible to determine a reasonable point at which to truncate

iteration of the numerical simulation from inspection of the behaviour of the objective function alone, over the course of a trial evaluation. This behaviour at a randomly selected location in the parameter space for this problem is illustrated in Figure 7.10. Inspection of this Figure suggests that it may be possible to use objective function values from simulations terminated at some point after about 1700 iterations for the purposes of optimisation, since by this time the evidently decaying oscillations in the objective function value have all but disappeared.

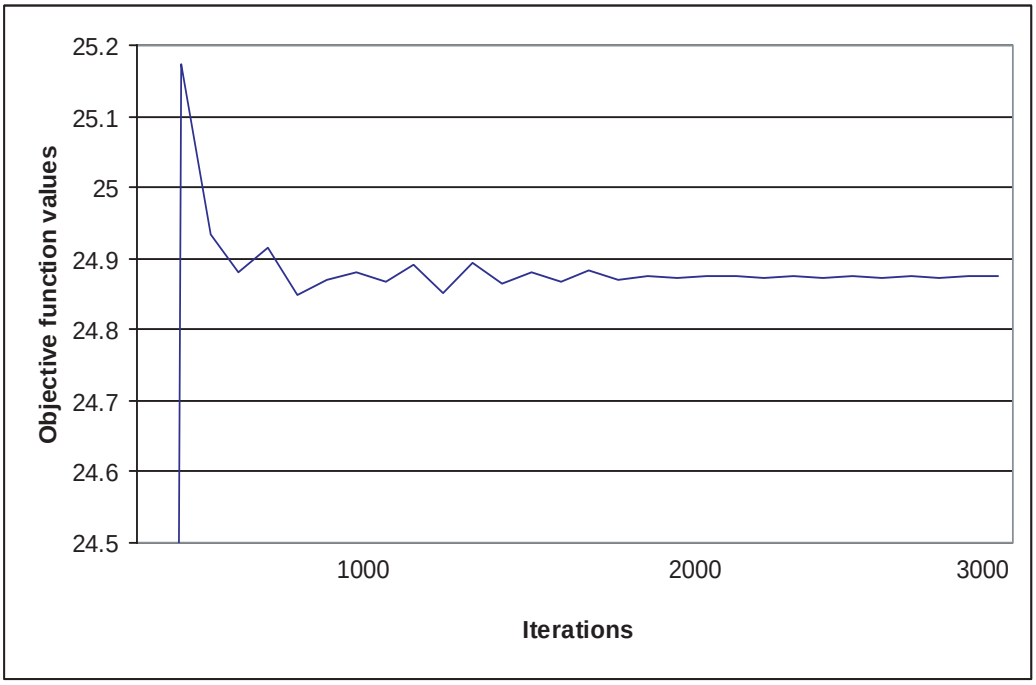


Figure 7.10 Evolution of antenna objective function values with simulation iterations

7.1.2.1 Numerical experiments on the antenna simulation. As a simple test of this assumption, a parameter sweep was performed of the parameter space for the problem, evaluating the objective function at intervals of 100 iterations for cases defined by the cross-product of three parameter values in each dimension, a total of 2187 samples at each iteration count. The minimum value found by this coarse sampling is plotted in Figure 7.11. As can be seen, there is considerably more variation in the minimum value found than there is in the value at the randomly chosen point, but the variation does become smooth and monotonic at about the same iteration count.

The location of the "global" minimum is tabulated, by the index of the sample point in each dimension, against iteration count, in Table 7.2. The location in parameter space ceases to change after 1700 iterations, corresponding approximately to the point at which oscillations in the objective function value at the randomly sampled point become negligible in Figure 7.10, and the variation in the coarsely-sampled "global minimum" value becomes smooth and monotonically changing in Figure 7.11.

From this brief investigation it appears it may be possible to determine an appropriate iteration count for truncation from inspection of objective function value behaviour alone. The advantage in terms of reduced execution time is only a factor of 1.76, slightly less than for the aerofoil case study of the previous section, but over the course of tens of function evaluations, for simulations that may

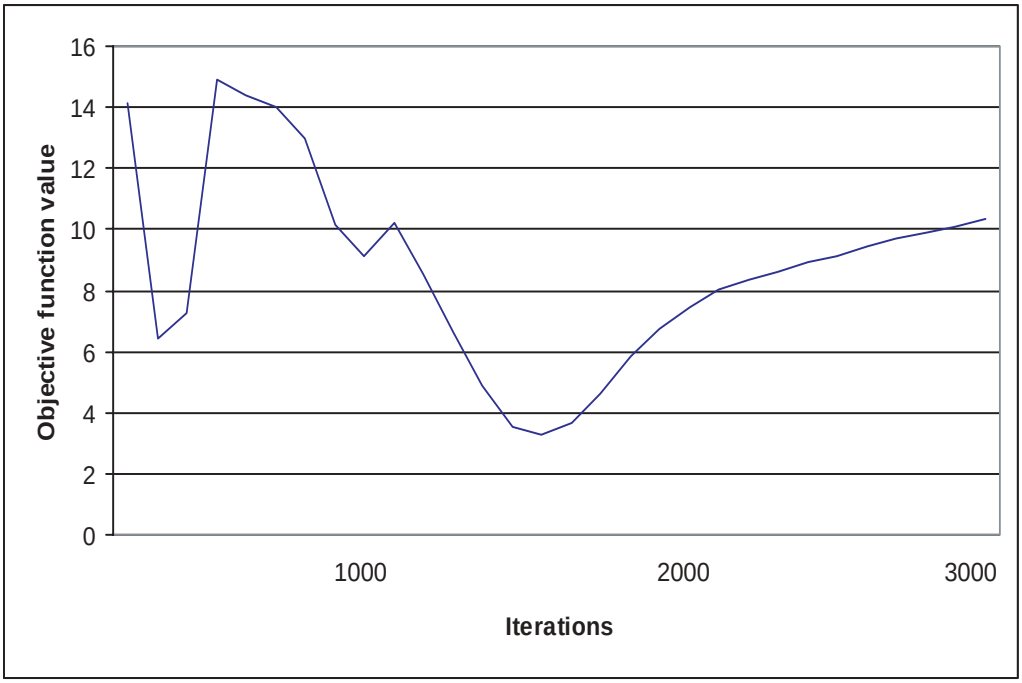


Figure 7.11 Minimum sampled antenna objective function value with simulation iterations

require hours to complete for each function evaluation, the reduction in total time taken could make the difference between a theoretically feasible, and a practically useful tool for engineering design.

7.1.3 Summary: temporal preconditioning

For an important class of optimisation problems in engineering design that derive their objective function from execution of a numerical simulation, the simulation employs iterative methods internally for solution of systems of equations describing the physical problem. A method is proposed of reducing the time taken to locate a minimum in some chosen parameter space, by truncating the iteration of the numerical simulation before convergence of the model. Termed *temporal preconditioning*, the feasibility of the method was investigated for two real-world case studies.

For a problem in design of a two-dimensional aerofoil, with an objective of maximising the aerofoil's lift-drag ratio, it was demonstrated that inspection of the residual errors in the fluid dynamics simulation could identify an appropriate iteration count at which to truncate execution of the simulation. The overall reduction in execution time achieved was a factor of two, compared with the iteration count at which the solution was assumed to have ceased improvement. The iteration count chosen was sufficient to ensure convergence of the continuity equation, required for conservation of mass considerations, while equations for turbulence quantities remained unconverged. Several representative optimisation algorithms were tested using data from truncated simulation execution, and all were shown to be capable of correctly determining the location of the global minimum in a given parameter space, as defined by execution of a fully converged model.

In a problem of the design of a novel communications antenna using finite difference time domain simulation of the electromagnetic fields around the antenna array, it was demonstrated that inspection of the behaviour of the objective function values derived from trial simulations could

Iterations							
100	1	1	1	3	1	3	2
200	2	2	1	1	1	2	3
300	1	2	3	3	1	2	3
400	3	1	1	3	2	2	2
500	3	1	3	3	2	2	2
600	3	1	3	3	2	2	2
700	1	3	1	2	1	2	3
800	1	3	1	2	1	2	3
900	1	3	1	2	1	2	3
1000	1	3	1	2	1	2	3
1100	2	3	3	2	1	2	3
1200	2	3	3	2	1	2	3
1300	2	3	3	2	1	2	3
1400	2	3	3	2	1	2	3
1500	2	3	1	2	1	2	3
1600	3	3	2	2	1	2	3
1700	3	3	1	2	1	2	3
1800	3	3	1	2	1	2	3
1900	3	3	1	2	1	2	3
2000	3	3	1	2	1	2	3
2100	3	3	1	2	1	2	3
2200	3	3	1	2	1	2	3
2300	3	3	1	2	1	2	3
2400	3	3	1	2	1	2	3
2500	3	3	1	2	1	2	3
2600	3	3	1	2	1	2	3
2700	3	3	1	2	1	2	3
2800	3	3	1	2	1	2	3
2900	3	3	1	2	1	2	3
3000	3	3	1	2	1	2	3

Table 7.2 Location indices for antenna “global” minimum with simulation iterations

identify an iteration count at which to truncate execution of the simulation, to achieve a reduction in total execution time of a factor of 1.76. Parameter sweep sampling of the simulation over a defined parameter space confirmed the stability of the coarse location of the global minimum for iteration counts in excess of the chosen threshold. This would suggest the location of a minimum found using the truncated data would coincide with the location found using a fully converged model.

For both problems the choice of iteration count required inspection of time-varying data derived from trial runs of the numerical simulations. Despite this imposition, application of the method would provide substantial gains in completion times for optimisation. While it may be possible to contrive an automatic method of choice of the truncation point, for the fluid dynamics problem in

particular some intelligent application of problem-specific knowledge was required to choose an appropriate threshold for the residual errors to trigger truncation of the simulation.

7.2 Spatial preconditioning

Temporal preconditioning sought to improve optimisation performance by altering the numerical simulation from which an objective function value was derived. In contrast, *spatial preconditioning* seeks to improve performance by altering the parameter space of the problem.

All the techniques investigated in the work described in preceding chapters have been applied to simply bounded parameter spaces. The global minimum of each of these parameter spaces occupies a particular location, and for the purposes of this work this location is defined as being static, i.e. time-varying objective functions have explicitly been excluded. Intuitively, if the optimisation methods could be constrained to the immediate vicinity of the global minimum this could be expected to have an impact on their performance. Of course, this is a "Catch 22", because if we knew where the global minimum was we wouldn't need to perform the optimisation!

In the method proposed, at each step a random sampling is made of the parameter space, multiple objective function evaluations being executed on parallel or distributed computing resources. The values returned are then formulated as a "multi-body" problem. Each sample point is given a "mass" proportional to its objective function value, and a location in the n-space of the parameter space. The values are normalised above the current minimum and exponentially weighted to ensure "clustering" of points does not inordinately outweigh the effect of "good" points. The location of the "centre of mass" of the combined system is then calculated by summing mass components in each dimension. A reduction factor is applied to the extent of the parameter space in each dimension, and a new parameter space, with new upper and lower bounds, constructed about the calculated centre of mass. The process is repeated until a desired reduction in size of the parameter space is achieved.

The motivation behind reducing the size of the parameter space toward regions that accumulate good objective function values rests on two ideas:

- For a convex function, the region surrounding the global minimum can be expected to yield more points with good objective function values than regions further from the minimum. As can be seen from the examples in Chapter 3, real-world problems cannot be assumed to be strictly convex, but the abstraction may still prove useful, especially for local minima.
- All the problems considered have been simply bounded, and the solution found to the limit of a user-supplied, desired tolerance. For the line searching algorithms described, for example, tolerance thresholds will be reached more quickly in the search direction if the domain is more closely bounded. In the toolset architecture outlined in Section 1.2, caching of computed results is employed. For stochastic methods the statistical probability of requesting trial solution points within tolerance limits of a pre-computed, stored result will be greater for a smaller parameter space.

The process of spatial preconditioning is graphically illustrated in Figures 7.12, 7.13 and 7.14, for the radio frequency design case, Bead. Figure 7.12 shows the original parameter space, with isosurfaces of the objective function, and the first set of sampled points. Figure 7.13 shows the parameter space boundaries after the first preconditioning step, and the second step sampled points. The isosurfaces in Figure 7.13 have been reduced to a lower value than appeared in Figure 7.12. Figure 7.14 show the parameters space boundaries further reduced after the second step of preconditioning, and further reduced isosurfaces. It may be noted that the final parameter space, roughly

one eighth the extent of the original, has drawn in toward, and contains, the region of better values defined by the tighter isosurfaces.

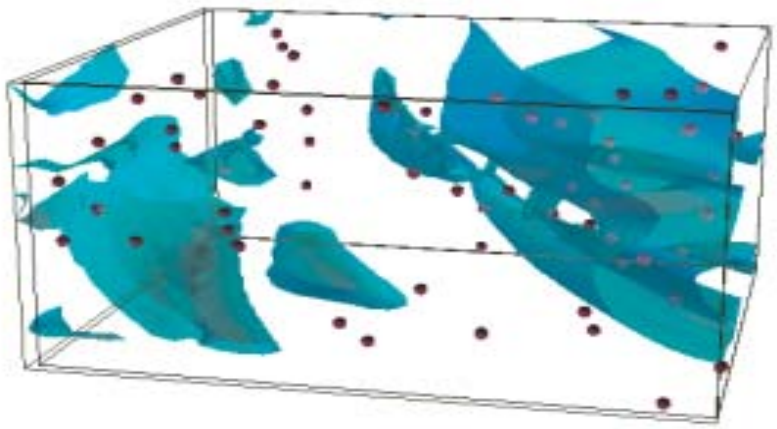


Figure 7.12 Spatial preconditioning, Bead test case, first step sampling

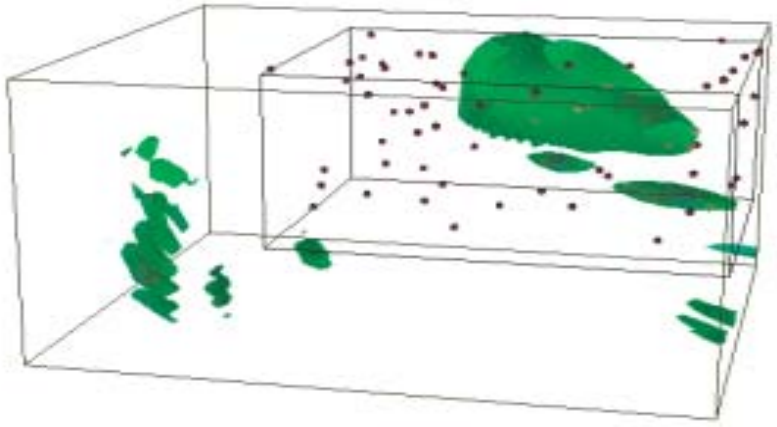


Figure 7.13 Spatial preconditioning, Bead test case, second step sampling

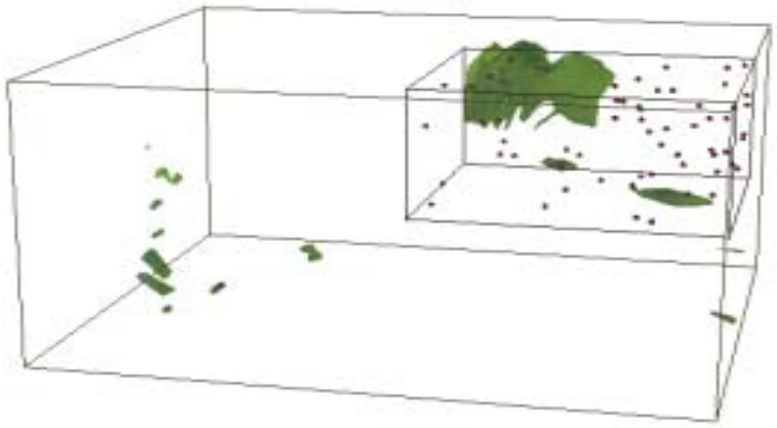


Figure 7.14 Spatial preconditioning, Bead test case, after two steps

7.2.1 Numerical experiments of spatial preconditioning

The method of spatial preconditioning described was implemented with a fixed reduction factor at each step of 0.707. This gives an approximate halving of the parameter space in each dimension in two steps of the preconditioner. The number of samples at each step was fixed at 64, an arbitrarily chosen "population" size corresponding to the number of CPUs available on a local parallel computing cluster used for some of the experiments described in the previous section. The preconditioner was used with each of the following algorithms:

- P-BFGS
- Simplex (Nelder-Mead)
- Simplex with line search
- Simplex with single-pass line search
- MDS
- RSCS
- RSCS with line search
- RSCS with single-pass line search
- EPSOC

on the following test cases:

- The two Quantum Electro-dynamics cases: Laser 1 and Laser 2
- The two durable component design cases: Crack 1 and Crack 2
- The 2D aerofoil design case: Aerofoil
- The radio-frequency design case: Bead

with one and two steps of preconditioning for each case.

Results are tabulated in Appendices J and K. In these tables it may be observed that there are statistically significant differences in the quality of results returned by various algorithms on different test cases. These differences are summarised in Table 7.3, for single-step preconditioning, and Table 7.4, for two-step preconditioning. Where there was a statistically significant (to better than 90% probability) improvement a "+" is shown. Where preconditioning yielded a significant degradation in the result returned, a "-" is shown.

	Laser 1	*Laser 2*	*Crack 1*	*Crack 2*	*Aerofoil*	*Bead*
P-BFGS						
NM			-	-		
NML				-		
NML1				-		
MDS		-		-		
RSCS			-			
RSCSL	+			-		
RSCSL1				-	+	
EPSOC	-		-	-		+

Table 7.3 Statistical differences in quality of results returned after one step spatial preconditioning, by algorithm and test case

	Laser 1	*Laser 2*	*Crack 1*	*Crack 2*	*Aerofoil*	*Bead*
P-BFGS		+				+
NM			-	-		
NML				-		
NML1				-		
MDS			-	-		
RSCS				-		+
RSCSL				-	+	
RSCSL1				-	+	
EPSOC			-	-	+	

Table 7.4 Statistical differences in quality of results returned after two step spatial preconditioning, by algorithm and test case

The results in Tables 7.3 and 7.4 are inconclusive, apart from the observation that preconditioning almost always appears to degrade the results returned for the second Crack test case. Inspection of the log files for the preconditioner for this test case showed that the global minimum for the data set was clipped from the parameter space after a single preconditioner step.

Quality of results returned is only one measure of the possible impact of preconditioning. It may also affect the speed of algorithms. Raw batch counts for each algorithm on each test case were compared before and after preconditioning. The percentage change in the function evaluation batches each algorithm required is tabulated in Table 7.5, for one preconditioning step, and Table 7.6, for two preconditioning steps. These batch counts *include* the batches required to perform the preconditioning. Values for EPSOC are not included, since it has a specified, fixed iteration count, equivalent to the batch count.

	Laser 1	*Laser 2*	*Crack 1*	*Crack 2*	*Aerofoil*	*Bead*
P-BFGS	8.3	-47.5	3.8	-3.2	-39.7	19.5
NM	-10.2	-2.0	-11.3	30.3	29.5	12.0
NML	-15.4	-21.6	-7.1	15.7	-26.7	-14.0
NML1	-10.8	-2.6	-15.2	50.0	2.9	-8.3
MDS	-3.0	0.0	9.1	-5.1	-7.1	16.7
RSCS	3.7	-18.8	8.3	-37.5	0.0	5.0
RSCSL	3.4	-31.1	-8.8	-11.2	12.0	6.7
RSCSL1	-15.0	-16.9	-24.4	-10.3	-18.4	4.8

Table 7.5 Percentage change in batches required, using one preconditioning step

	Laser 1	*Laser 2*	*Crack 1*	*Crack 2*	*Aerofoil*	*Bead*
P-BFGS	-23.8	-3.0	13.2	-22.6	-52.1	-22.0
NM	-2.0	3.9	-39.6	3.0	-6.8	0.0
NML	-47.0	-13.5	-16.7	21.4	-46.7	-26.0
NML1	-18.9	-12.8	-18.2	30.0	14.3	-8.3
MDS	24.2	15.6	-9.1	-35.6	4.6	16.7
RSCS	48.1	-6.3	4.2	-28.1	14.3	-5.0
RSCSL	11.5	-7.8	-15.4	6.7	-25.3	-15.6
RSCSL1	27.5	-18.6	-33.3	-7.7	-32.7	4.8

Table 7.6 Percentage change in batches required, using two preconditioning steps

On average, across all algorithms for all test cases, one preconditioning step led to a reduction in batches of 4.2%. Two preconditioning steps delivered an average reduction in batches of 7.4%. However, there was no uniform trend in speedup, either by algorithm, or for a particular test case.

7.2.2 Summary: spatial preconditioning

A simple method of preconditioning the parameter space for an optimisation problem has been described that uses progressive reduction of simple bounds on the parameter space about a “centre of mass” derived from randomly sampling the objective function values within the parameter space.

It has been shown to deliver modest average reductions in the number of concurrently executed batches of objective function evaluations required by a variety of optimisation algorithms.

The number of steps of preconditioning should be chosen carefully. If this method of random sampling were able to “zero in” on the global minimum with confidence it would make a simple and effective optimisation method in itself. Unfortunately, it is easy for too many steps to eventually remove the global minimum from the region considered. Indeed, in the second durable component design case, Crack 2, this was observed after a single step of the preconditioner. The extent to which this is a significant problem depends on whether the “best” result is required, or just sufficient improvement to suit a particular purpose.

In general, use of spatial preconditioning should be treated with caution, due to the risk in particular cases of degrading the results returned by removing the global optimum from the parameter space considered, and its occasional significant increases in run-time, rather than the intended reduction. It could conceivably be used most effectively as an adjunct used in tandem with un-modified methods, with the possibility it may provide better results sooner.

8.0 Conclusion

The main thrust of the work described in preceding sections has been to develop algorithms with improved performance for use in a general-purpose, "black-box" optimisation toolset, applied to continuous, simply-bounded, non-linear numerical simulations. To a large extent, the focus of the work has already defined many of the characteristics of the problems to be considered:

- They are assumed to be continuous, i.e. differentiable, even if derivatives are not available. This allows consideration of algorithms that may rely on the existence of derivatives, such as quasi-Newton methods. Some discontinuities in objective functions are allowed by assuming piece-wise linearity.
- The objective functions may be multivariate. This implies the methods used, and their implementations, must have the sophistication to operate in multiple dimensions.
- The parameters are simply bounded. As described in Chapters 4 and 5, this allows the use of interval subdivision line searching, with a commensurate increase in available concurrency.
- Linear and, in particular, non-linear constraints have not been considered. Dealing with constraints can be quite difficult. The optimisation toolset Nimrod/O, described in Section 1.2, provides the functionality for application of simple, arithmetic constraints using penalty function methods, but their consideration has been omitted.
- The objective functions are allowed to be non-linear, precluding the use of linear programming methods.

Given these characteristics, algorithms from the methods represented in Chapters 4, 5 and 6 can all be seen as candidates for use on the problems to be considered. Nimrod/O allows the user to choose from a range of different algorithms, or combinations of them, and set the context for their operation on a specific problem at hand. The user is, however, confronted with a question when making these choices:

Which algorithm will perform best on my problem?

A strong interpretation of the No Free Lunch (NFL) theorem (Wolpert and Macready 1997) would have us reply: without some knowledge of the problem there can be no answer, since considered over *all* problems, all algorithms have the same average performance. As Culberson (1996) reminds us, the results of Turing (1936) preclude the existence of a "magic bullet" that can solve any problem. But NFL itself allows the possibility of minimax comparisons between algorithms, *when constrained to a single problem.* It is a matter of common experience, amply demonstrated by the numerical results reported in earlier chapters, that for a given problem some algorithms perform better than others. With cautious extension and a weaker interpretation of NFL, it may be possible to say that for a given *class* of problems, some class of algorithms may be a better choice than others.

This is not a new idea: Sharpe (2000) graphically illustrated this from one viewpoint by suggesting classifying search spaces according to which search algorithms perform well on them. Wolpert and Macready (1997) suggest that years of research on traveling salesman problems have yielded algorithms "aligned" to these problems, which thus perform well on them. Problem landscapes can be artificially constructed to favour one class of algorithms over another (Sharpe 1998).

If, when considering which algorithm is best suited to a problem, an implicit assumption is made that real world problems (RWP) are the union of a number of coherent subsets of problems sharing defining characteristics, then it may be reasonable to assume a matching of algorithms to these

subsets is possible. Figure 8.1 provides one, non-exhaustive illustration of problem classification and a mapping from Sharpe's classification to this can be conjectured.

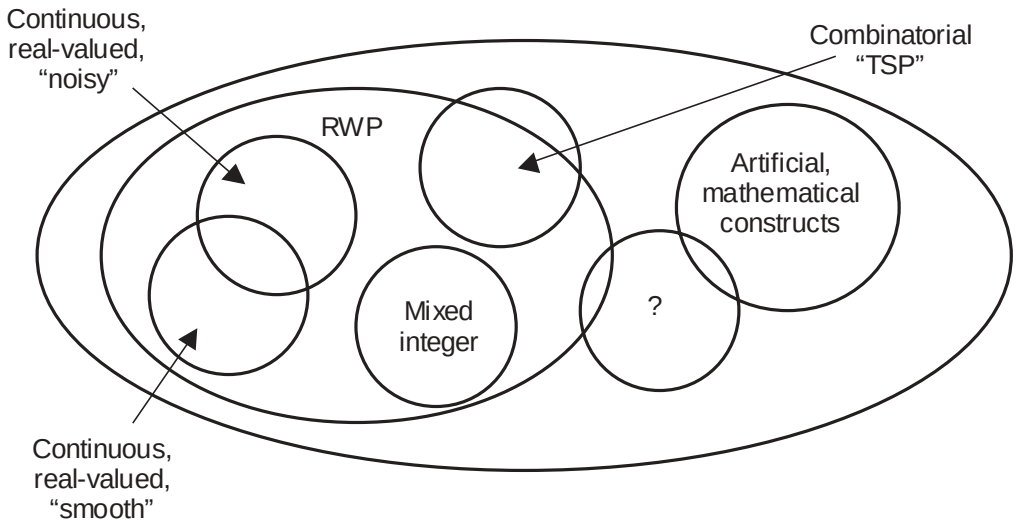


Figure 8.1 Classification of optimization problems by common features

It is generally quite rare that a problem is presented about which there is absolutely no knowledge. At the very least, the numerical methods used in its solution have quite likely been employed on other problems, and optimisation problems in many fields have been explored, some in great detail. (For example, a recent search of the Science Citation Index database using the terms "genetic algorithm" and "aerodynamic" matched over 130,000 documents.) In the event that accurate classification cannot be made, more than one approach can be tried simultaneously, particularly if the toolset used supports this.

In most cases, some general classification is usually possible. Qualitative comparison of the behaviour of the various different methods of optimisation tested suggests that different algorithms may be preferable, depending on quite broadly defined characteristics of the problem. For example, classical, gradient descent algorithms appear to perform poorly when the starting point chosen is distant from a global minimum, particularly when the objective function value returned is uncertain, or "noisy". Population based methods out-perform other algorithms in most cases, but are generally expensive to use, in terms of function evaluations required. It may be possible to summarise some of these observations by suggesting a coarse "mapping" of algorithms depending on two main factors (Sharpe 2003):

- *A priori* knowledge of the problem – how much we know about the "nature" of the problem.
- Problem complexity – how difficult the problem is to solve.

Figure 8.2 shows a suggested mapping drawn from observations of the numerical results of previous chapters. Where a problem is reasonably well understood, simple in structure, and free from complicating factors such as noise, the rapid convergence and efficiency of classical, gradient descent methods are advantageous. For particularly complex problems, "noisy" problems and those about which there is little *a priori* knowledge, the expense of using population-based methods is

justified by their better success rates. Between these two extremes, direct search methods provide reasonably rapid, economical, dependable performance.

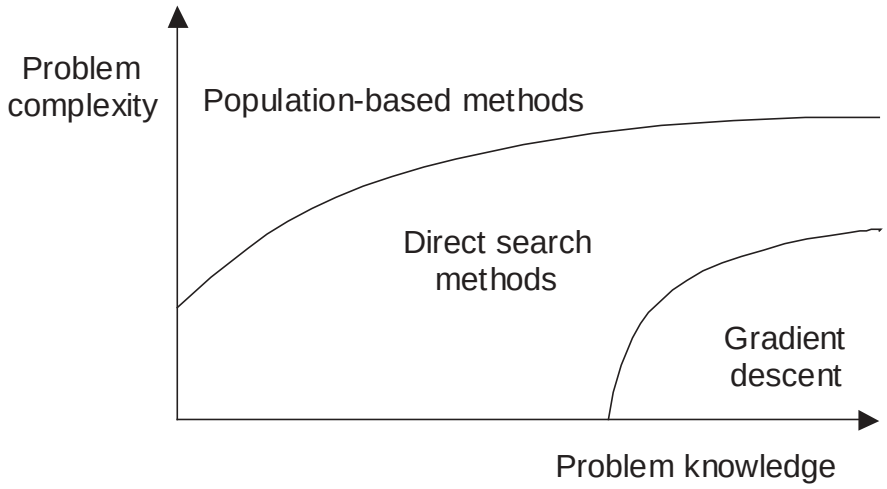


Figure 8.2 Mapping algorithm to problem

Within these broad classes, some further observations may be made.

P-BFGS, the parallel implementation of a quasi-Newton gradient method with BFGS update described in this thesis, is both faster and more reliable at returning good solutions than an equivalent serial algorithm. On a selection of real-world problems, it exhibits speedups of a factor of 3 or 4, and appears less prone to capture by local minima, due to a novel application of an equal interval iterative line search. In comparison with a representative simulated annealing code, it is also faster by a factor of more than 3, and provided better objective function results. On problems with noise, many local minima, or regions of very small gradient, it performed less well than some direct search and stochastic methods.

Among the direct search methods, the Nelder-Mead simplex algorithm and the Multidirectional Search (MDS) method of Dennis and Torczon perform well on “smooth” test cases, but poorly on problems with noise and many local minima. The new hybrid, the Reducing Set Concurrent Simplex (RSCS) algorithm proposed in this thesis, performs better, with median results consistently better than Nelder-Mead on “noisy” problems, and equivalent on the “smooth” cases. It is also almost twice as fast, on average. Compared to MDS, RSCS is also much faster, in particular avoiding the problems of premature convergence MDS experienced on two test cases.

The addition of line searching to the simplex algorithms Nelder-Mead and RSCS, as proposed in Chapter 5, is also clearly very useful. A single-pass line searching variant of RSCS is the fastest of all direct search methods tested to return high quality objective function values. The iterative line-searching variant of RSCS is also very reliable, providing the best objective function value in over half the case studies. Applying line searching strategies to the Nelder-Mead algorithm also yields improvements, with the single-pass variant being consistently faster than the parent algorithm, by more than 50% on average. The iterative line-searching variant is also very reliable: NML, MDS and RSCS are the only algorithms guaranteed to return a good result on all test cases from random, multiple runs.

The new Evolutionary Programming algorithm proposed in Chapter 6, EPSOC, demonstrates the advantages of the stochastic methods. In 4 of 6 cases, EPSOC can deliver statistically significantly better results than any other algorithm tested, and statistically equivalent results on the remaining cases. Contrary to other opinions of stochastic methods and genetic algorithms as being slow to solve local optimisation problems (Durand and Alliot 1999) and requiring thousands of function evaluations even for problems of low dimensionality (Elster and Neumaier 1995), EPSOC provides these results with some 50% fewer iterations, on average, than the other algorithms tested. Given sufficient parallel computing resources, this implies a considerable advantage in time taken. It does use more function evaluations than direct search or gradient methods, in general by an order of magnitude, but not to an extent that its use on real-world problems, as typified by the test cases, becomes infeasible. Compared to a representative Genetic Algorithm, its nearest rival in terms of performance, EPSOC is also more robust, with less dependence on its operational settings.

8.1 Further work

In the course of this research, some promising avenues of further enquiry have become apparent. In particular, the potential for some of the algorithms to be applied to problems of multiple objectives is of great, practical interest. It is often the case, during the design process, that competing objectives require satisfaction. A mechanical part might need to be strong and durable, but also lightweight. In the past, problems of this nature might often have been formulated as optimisation problems subject to constraints. One or more of what are really objectives could be expressed as derived limits on the design, minimum yield strength for example, within which optimisation of the remaining objective, e.g. component weight, was attempted. This is generally unsatisfactory – if the design required is to be "optimal", then all objectives should be the best (feasible) values.

EPSOC has already been identified has having potential for multi-objective optimisation, using a novel implementation with multiple, "virtual" archives as outlined in Section 6.5. Given the expressed demand for a capability of this sort, this is an area deserving of further investigation.

In addition, a common idea in the literature is to combine the differing abilities of different algorithms to develop more powerful hybrids. For example, it may be desirable to precede the rapid convergence to a local optimum of gradient descent or direct search methods with the wide exploration abilities of stochastic methods. Durand and Alliot (1999) provide such an example, with a hybrid GA+Simplex method.

The Nimrod/O general-purpose optimisation toolset described in Section 1.2 provides a prototype means for "chaining" algorithms together sequentially. There is significant potential in exploiting this mechanism to develop and test *ad hoc* hybrids. The idea to improve the performance of one algorithm by re-using the computed results from another also bears investigation. As Nimrod/O employs a cache of computed results, the NimCache shown in Figure 1.2, the impact of "cache seeding" could readily be explored.

Web-based services are becoming an increasingly important mode of service delivery and has already been explicitly applied to remote optimisation tasks (Alotto, Molfino and Molinari 2001). Investigation and implementation of web services for Nimrod/O could possibly broaden its adoption in industry.

It is believed that Nimrod/O, with further development, has commercial potential. Features promoting ease of use, and modification to enhance fault tolerance would greatly benefit its ready application to a wide range of application areas.

8.2 Achievements and significance

This research has presented a number of novel algorithms in the field of optimisation of continuous, non-linear numerical simulations. Parallel algorithms have been developed across a number of classes of optimisation methods. The contributed methods are summarised in Figure 8.3, shown in **bold** typeface.

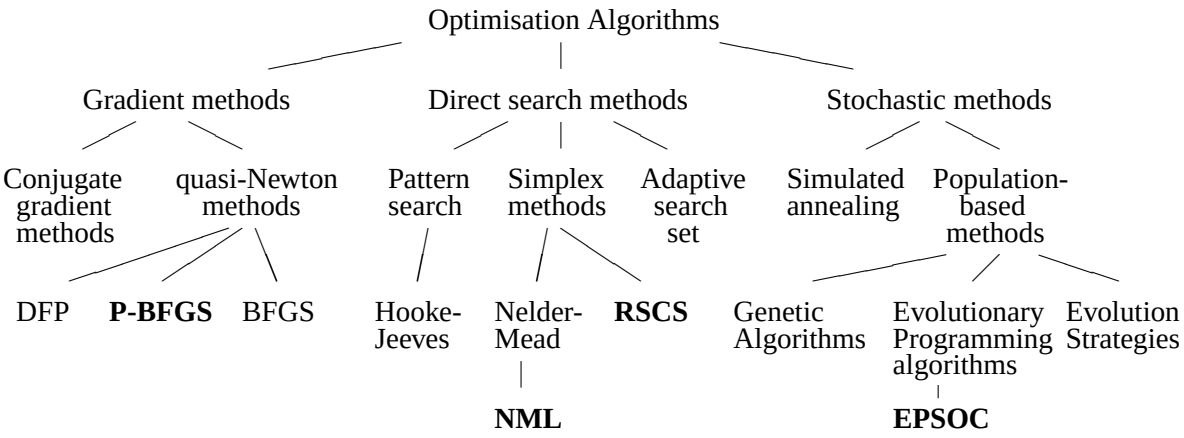


Figure 8.3 A (revised) taxonomy of optimisation algorithms

In order to provide a realistic test of the algorithms abilities, a series of "sandbox" test cases were developed from data collected from parameter sweep execution of several real-world problem simulations. The collected data was encapsulated with linear interpolation routines to give challenging optimisation problems with realistic levels of complexity and structure characteristic of real problems, but with computational overheads a vanishingly small fraction of that of the original models. Without these test cases, the range and extent of testing required to develop and prove new methods of optimisation would have been infeasible without recourse to mathematically contrived test functions, with the limitations that imposes.

With reference to Figure 8.3, it is believed that the work contained in this thesis has contributed to the knowledge of general purpose optimisation systems in the following areas:

- The idea of developing a set of "sandbox" case studies for effective testing of optimisation algorithms has been established as a feasible alternative to the use of artificial test functions, and an initial set of problems with varying characteristics has been presented.
- A parallel implementation of the quasi-Newton gradient method with BFGS update has been developed and its efficacy in comparison to a corresponding sequential algorithm and widely-used method of simulated annealing demonstrated.
- The use of a method of parallel line search with the Nelder-Mead simplex algorithm has been implemented and its advantages compared to the original algorithm, in speed and reliability, clearly shown.

- New direct search methods, the Reducing Set Concurrent Simplex (RSCS) algorithm with its line searching variants, have been presented, and their superior performance compared to a variety of direct search methods demonstrated.
- A novel Evolutionary Programming algorithm using concepts of self-organised criticality, EPSOC, has been presented, and demonstrated to be superior in performance to a wide variety of gradient, direct search and stochastic methods on a set of test cases drawn from real-world problems. It also has evidence of potential for multi-objective optimisation using a novel implementation with multiple, "virtual" archives.
- Methods of preconditioning optimisation problems to reduce the total time taken to achieve an optimal result have been presented. Temporal preconditioning, based on the time behaviour of the numerical simulations, has been demontrated to yield substantial speedup.
- Some conclusions have been drawn on the applicability of specific optimisation methods to different classes of real-world problems.
- All of the methods described have been implemented in the framework of a general-purpose optimisation toolset, Nimrod/O, to provide a sound basis for future work and potential commercial application.

As shown in Section 8.1, the work contained in this thesis has the potential to give rise to a number of future projects.

A.0 Pair-wise comparison of Nelder-Mead Simplex algorithms, with and without iterative line search

The following tables contain final objective function values delivered by 10 simultaneous runs, from random starting points, of the Nelder-Mead Simplex algorithm and a variant with an iterative subdivision line search, for each test case. Also tabulated is the Mann-Whitney U test statistic, and the corresponding significance level, assuming a two-tailed test.

	Laser 1	
	NM	**NML**
	-0.476	0.242
	-0.479	0.045
	-0.480	-0.416
	-0.481	-0.432
	-0.481	-0.455
	-0.481	-0.463
	-0.481	-0.479
	-0.481	-0.482
	-0.482	-0.482
	-0.482	-0.482
Rank sum	123.0	87.0
U	32.0	
P	0.190	

Table A.1 Simplex iterative line search evaluation – Laser 1 test case

	Laser 2	
	NM	**NML**
	0.496	0.333
	0.314	0.310
	0.286	0.278
	0.274	0.274
	0.274	0.262
	-0.561	-0.187
	-0.562	-0.559
	-0.562	-0.559
	-0.562	-0.562
	-0.562	-0.563
Rank sum	108.0	102.0
U	47.0	
P	0.853	

Table A.2 Simplex iterative line search evaluation – Laser 2 test case

	Crack 1	
	NM	**NML**
	199.229	219.539
	199.194	208.181
	199.076	199.522
	199.067	199.191
	195.801	197.574
	188.053	195.622
	187.890	191.217
	187.762	188.547
	187.615	187.695
	187.614	187.553
Rank sum	116	94
U	39.0	
P	0.435	

Table A.3 Simplex iterative line search evaluation – Crack 1 test case

	Crack 2	
	NM	**NML**
	4411	4405
	5264	5239
	5265	5263
	5293	5264
	5297	5291
	5300	5295
	5311	5298
	5319	5322
	5319	5330
	5353	5333
Rank sum	112	98
U	43.0	
P	0.630	

Table A.4 Simplex iterative line search evaluation – Crack 1 test case

	Aerofoil	
	NM	**NML**
	-44.736	-44.767
	-66.251	-54.612
	-67.255	-56.558
	-67.323	-64.967
	-67.323	-65.608
	-68.375	-65.797
	-68.459	-68.362
	-68.626	-68.537
	-68.641	-68.624
	-68.643	-68.642
Rank sum	121	89
U	34.0	
P	0.247	

Table A.5 Simplex iterative line search evaluation – Aerofoil test case

	Bead	
	NM	**NML**
	37.715	24.504
	28.292	10.354
	15.083	-11.795
	13.840	-12.841
	5.086	-14.806
	3.711	-16.547
	3.608	-18.046
	-10.798	-22.615
	-13.005	-24.240
	-26.977	-39.849
Rank sum	78.0	132.0
U	23.0	
P	0.043	

Table A.6 Simplex iterative line search evaluation – Bead test case

B.0 Pair-wise comparison of Nelder-Mead Simplex algorithms, with and without single-pass line search

The following tables contain final objective function values delivered by 10 simultaneous runs, from random starting points, of the Nelder-Mead Simplex algorithm and a variant with a single-pass subdivision line search, for each test case. Also tabulated is the Mann-Whitney U test statistic, and the corresponding significance level, assuming a two-tailed test.

	Laser 1	
	NM	**NML1**
	-0.476	0.129
	-0.479	0.008
	-0.480	-0.051
	-0.481	-0.255
	-0.481	-0.445
	-0.481	-0.457
	-0.481	-0.465
	-0.481	-0.471
	-0.482	-0.473
	-0.482	-0.478
Rank sum	154.0	56.0
U	1.0	
P	0.000	

Table B.1 Simplex single-pass line search evaluation – Laser 1 test case

	Laser 2	
	NM	**NML1**
	0.496	0.334
	0.314	0.253
	0.286	0.120
	0.274	0.008
	0.274	-0.018
	-0.561	-0.2223
	-0.562	-0.237
	-0.562	-0.486
	-0.562	-0.559
	-0.562	-0.562
Rank sum	107.0	103.0
U	48.0	
P	0.911	

Table B.2 Simplex single-pass line search evaluation – Laser 2 test case

	Crack 1	
	NM	**NML1**
	199.229	219.562
	199.194	208.204
	199.076	199.537
	199.067	199.153
	195.801	197.904
	188.053	195.980
	187.890	195.880
	187.762	192.948
	187.615	191.093
	187.614	187.549
Rank sum	121	89
U	34.0	
P	0.247	

Table B.3 Simplex single-pass line search evaluation – Crack 1 test case

	Crack 2	
	NM	**NML1**
	4411	5236
	5264	5253
	5265	5265
	5293	5269
	5297	5283
	5300	5294
	5311	5299
	5319	5319
	5319	5326
	5353	5332
Rank sum	110	100
U	45.0	
P	0.739	

Table B.4 Simplex single-pass line search evaluation – Crack 2 test case

	Aerofoil	
	NM	**NML1**
	-44.736	-44.760
	-66.251	-54.037
	-67.255	-57.182
	-67.323	-60.996
	-67.323	-64.959
	-68.375	-67.323
	-68.459	-68.559
	-68.626	-68.617
	-68.641	-68.627
	-68.643	-68.632
Rank sum	116	94
U	39.0	
P	0.393	

Table B.5 Simplex single-pass line search evaluation – Aerofoil test case

	Bead	
	NM	**NML1**
	37.715	24.504
	28.292	10.354
	15.083	-11.795
	13.840	-12.841
	5.086	-14.806
	3.711	-16.547
	3.608	-18.046
	-10.798	-22.615
	-13.005	-24.240
	-26.977	-39.849
Rank sum	78.0	132.0
U	23.0	
P	0.0432	

Table B.6 Simplex single-pass line search evaluation – Bead test case

C.0 Statistical comparison – Simplex, MDS and RSCS algorithms

The following tables contain the rank sum of each algorithm across the pooled ranks, and Kruskal-Wallis H test statistic. Final objective function values delivered by 10 simultaneous runs of the 5 selected algorithms, from random starting points, median, interquartile range, and Shapiro-Wilk W test statistic can be found in Appendix H.

	NM	MDS	RSCS	RSCSL	RSCSL1
Rank sum	346.5	364.0	220.5	162.0	182.0
H	16.669				

Table C.1 Comparison of direct search algorithms – Laser 1 test case

	NM	MDS	RSCS	RSCSL	RSCSL1
Rank sum	254.5	202.5	263.0	278.0	277.0
H	1.804				

Table C.2 Comparison of direct search algorithms – Laser 2 test case

	NM	MDS	RSCS	RSCSL	RSCSL1
Rank sum	289.5	290.5	271.0	191.0	233.0
H	3.429				

Table C.3 Comparison of direct search algorithms – Crack 1 test case

	NM	MDS	RSCS	RSCSL	RSCSL1
Rank sum	243.0	296.0	301.5	230.0	204.5
H	3.371				

Table C.4 Comparison of direct search algorithms – Crack 2 test case

	NM	MDS	RSCS	RSCSL	RSCSL1
Rank sum	327.0	315.0	314.0	140.0	179.0
H	14.713				

Table C.5 Comparison of direct search algorithms – Aerofoil test case

	NM	MDS	RSCS	RSCSL	RSCSL1
Rank sum	202.0	232.5	214.0	355.0	271.5
H	7.185				

Table C.6 Comparison of direct search algorithms – Bead test case

D.0 Pair-wise comparison of RSCS, Nelder-Mead and MDS algorithms, Aerofoil test case

The following tables contain rank sums across pooled ranks, the Mann-Whitney U test statistic, and the corresponding significance level, assuming a two-tailed test, for the RSCS, Nelder-Mead Simplex and MDS algorithms, for the Aerofoil test case. Final objective function values delivered by the selected algorithms, median, interquartile range, and Shapiro-Wilk W test statistic can be found tabulated in Appendix H.

	Aerofoil	
	RSCS	**NM**
Rank sum	106.0	104.0
U	49.0	
P	0.970	

Table D.1 Pairwise comparison of RSCS and Nelder-Mead Simplex – Aerofoil test case

	Aerofoil	
	RSCS	**MDS**
Rank sum	106	104
U	49.0	
P	0.970	

Table D.2 Pairwise comparison of RSCS and MDS – Aerofoil test case

E.0 Pair-wise comparison of RSCS iterative line-searching, Nelder-Mead and MDS algorithms, Bead test case

The following tables contain the rank sum of each algorithm across the pooled ranks, Mann-Whitney U test statistic and the corresponding significance level, assuming a two-tailed test. Final objective function values delivered by the selected algorithms, median, interquartile range, and Shapiro-Wilk W test statistic can be found tabulated in Appendix H.

	Bead	
	RSCSL	**NM**
Rank sum	134.0	76.0
U	21.0	
P	0.028	

Table E.1 Pairwise comparison of iterative line-search RSCS and Nelder-Mead Simplex – Bead test case

	Bead	
	RSCSL	**MDS**
Rank sum	133.5	76.5
U	21.5	
P	0.032	

Table E.2 Pairwise comparison of iterative line-search RSCS and MDS – Bead test case

F.0 Objective function values for EPSOC

The following tables contain final objective function values delivered by 10 simultaneous runs of EPSOC with given operational parameters, from random starting points. Table column labels are of the format: **b** *nbad* **m** *mutation width*, where mutation width is expressed as a percentage of parameter range. Also tabulated are the median for each group, interquartile range, Shapiro-Wilk W test statistic, rank sum of each group across the pooled ranks, and Kruskal-Wallis H test statistic.

	b3m5	b3m10	b3m20	b6m5	b6m10	b6m20	b9m5	b9m10	b9m20
	-0.471	-0.471	-0.467	-0.476	-0.471	-0.472	-0.476	-0.474	-0.477
	-0.473	-0.473	-0.475	-0.477	-0.473	-0.474	-0.479	-0.475	-0.480
	-0.475	-0.474	-0.477	-0.479	-0.478	-0.475	-0.479	-0.479	-0.481
	-0.476	-0.477	-0.477	-0.479	-0.478	-0.477	-0.480	-0.480	-0.481
	-0.477	-0.478	-0.479	-0.480	-0.480	-0.480	-0.481	-0.480	-0.481
	-0.479	-0.478	-0.480	-0.480	-0.480	-0.480	-0.481	-0.480	-0.481
	-0.479	-0.478	-0.480	-0.481	-0.481	-0.481	-0.481	-0.481	-0.481
	-0.480	-0.480	-0.481	-0.481	-0.481	-0.481	-0.481	-0.481	-0.481
	-0.481	-0.480	-0.481	-0.481	-0.481	-0.481	-0.481	-0.481	-0.482
	-0.481	-0.481	-0.481	-0.482	-0.481	-0.481	-0.482	-0.482	-0.482
Median	-0.478	-0.478	-0.480	-0.480	-0.480	-0.480	-0.481	-0.480	-0.481
IQR	0.005	0.006	0.004	0.002	0.003	0.006	0.002	0.002	0.001
W	0.898	0.890	0.762	0.879	0.784	0.833	0.768	0.746	0.695
Rank sum	309	274	400.5	479.5	453.5	432.5	560.5	523.5	662
H	17.118								

Table F.1 EPSOC evaluation – The quantum electro-dynamical case (Laser 1)

	b3m5	b3m10	b3m20	b6m5	b6m10	b6m20	b9m5	b9m10	b9m20
	-0.550	-0.553	-0.555	-0.558	-0.556	-0.555	-0.558	-0.555	-0.553
	-0.551	-0.558	-0.558	-0.559	-0.556	-0.557	-0.560	-0.558	-0.557
	-0.559	-0.559	-0.560	-0.559	-0.558	-0.560	-0.561	-0.559	-0.558
	-0.560	-0.559	-0.561	-0.561	-0.559	-0.560	-0.561	-0.559	-0.559
	-0.560	-0.561	-0.561	-0.561	-0.560	-0.560	-0.562	-0.560	-0.559
	-0.562	-0.561	-0.561	-0.562	-0.562	-0.560	-0.562	-0.562	-0.561
	-0.562	-0.561	-0.561	-0.562	-0.562	-0.561	-0.562	-0.562	-0.562
	-0.562	-0.562	-0.561	-0.562	-0.562	-0.562	-0.562	-0.562	-0.562
	-0.562	-0.562	-0.561	-0.562	-0.562	-0.562	-0.562	-0.562	-0.562
	-0.562	-0.563	-0.562	-0.562	-0.562	-0.562	-0.562	-0.562	-0.562
Median	-0.561	-0.561	-0.561	-0.561	-0.561	-0.560	-0.562	-0.561	-0.560
IQR	0.003	0.003	0.001	0.003	0.004	0.002	0.001	0.003	0.004
W	0.713	0.819	0.708	0.862	0.824	0.856	0.752	0.846	0.866
Rank sum	410.5	443	411	530.5	472	395	564.5	493	375.5
H	4.894								

Table F.2 EPSOC evaluation – The quantum electro-dynamical case (Laser 2)

	b3m5	b3m10	b3m20	b6m5	b6m10	b6m20	b9m5	b9m10	b9m20
	190.55	188.42	189.71	190.17	187.88	188.20	188.55	188.76	188.07
	189.96	188.30	187.95	188.77	187.85	187.93	188.14	187.86	187.88
	189.11	188.06	187.83	188.16	187.78	187.91	187.78	187.81	187.73
	188.10	188.00	187.80	188.03	187.76	187.89	187.76	187.75	187.70
	187.88	187.99	187.78	187.92	187.72	187.87	187.74	187.74	187.69
	187.85	187.77	187.78	187.69	187.69	187.83	187.68	187.64	187.68
	187.84	187.75	187.66	187.67	187.64	187.74	187.64	187.63	187.67
	187.63	187.65	187.65	187.64	187.64	187.68	187.61	187.60	187.64
	187.63	187.57	187.57	187.61	187.55	187.64	187.58	187.58	187.61
	187.61	187.55	187.57	187.55	187.55	187.59	187.56	187.55	187.60
Median	187.87	187.88	187.78	187.81	187.70	187.85	187.71	187.69	187.68
IQR	1.478	0.411	0.174	0.518	0.147	0.222	0.178	0.210	0.088
W	0.755	0.912	0.542	0.709	0.933	0.914	0.743	0.635	0.782
Rank sum	329	389.5	457.5	399.5	551.5	387.5	509.5	545.5	525.5
H	7.802								

Table F.3 EPSOC evaluation – The durable component design case using stress (Crack 1)

	b3m5	b3m10	b3m20	b6m5	b6m10	b6m20	b9m5	b9m10	b9m20
	5320	5334	5328	5312	5324	5319	5316	5321	5330
	5342	5336	5334	5320	5341	5332	5331	5341	5339
	5342	5339	5336	5337	5344	5341	5348	5343	5339
	5343	5339	5341	5340	5346	5344	5352	5346	5347
	5344	5344	5350	5345	5348	5345	5352	5351	5347
	5346	5351	5351	5351	5350	5346	5355	5352	5351
	5347	5352	5354	5354	5351	5349	5355	5353	5351
	5348	5353	5355	5355	5352	5351	5356	5355	5351
	5353	5353	5356	5355	5354	5353	5356	5356	5352
	5356	5356	5356	5356	5356	5355	5356	5357	5355
Median	5345	5348	5351	5348	5349	5345	5353	5351	5349
IQR	5.46	13.89	19.23	17.75	7.87	9.40	7.76	12.96	11.83
W	0.800	0.863	0.853	0.831	0.814	0.858	0.697	0.793	0.867
Rank sum	396	423	477	436.5	464	375	577	525.5	421
H	4.809								

Table F.4 EPSOC evaluation – The durable component design case using fatigue life (Crack 2)

	b3m5	b3m10	b3m20	b6m5	b6m10	b6m20	b9m5	b9m10	b9m20
	-67.881	-68.547	-68.622	-68.190	-68.586	-68.616	-68.415	-68.595	-68.632
	-68.463	-68.595	-68.629	-68.527	-68.614	-68.624	-68.522	-68.605	-68.633
	-68.522	-68.611	-68.631	-68.556	-68.628	-68.627	-68.535	-68.613	-68.637
	-68.523	-68.614	-68.633	-68.566	-68.628	-68.635	-68.535	-68.617	-68.638
	-68.527	-68.626	-68.634	-68.584	-68.630	-68.636	-68.623	-68.628	-68.638
	-68.565	-68.631	-68.637	-68.610	-68.632	-68.636	-68.625	-68.630	-68.638
	-68.601	-68.633	-68.638	-68.615	-68.634	-68.639	-68.632	-68.632	-68.639
	-68.619	-68.637	-68.641	-68.618	-68.641	-68.639	-68.641	-68.634	-68.640
	-68.620	-68.638	-68.641	-68.641	-68.642	-68.641	-68.641	-68.634	-68.641
	-68.637	-68.641	-68.642	-68.642	-68.643	-68.643	-68.642	-68.638	-68.641
Median	-68.546	-68.628	-68.635	-68.597	-68.631	-68.636	-68.624	-68.629	-68.638
IQR	0.097	0.026	0.010	0.063	0.013	0.013	0.106	0.021	0.003
W	0.594	0.777	0.924	0.623	0.779	0.869	0.788	0.865	0.871
Rank sum	185	424	599.5	300	523	591	395.5	400.5	676.5
H	28.932								

Table F.5 EPSOC evaluation – The 2D aerofoil design case (Aerofoil)

	b3m5	b3m10	b3m20	b6m5	b6m10	b6m20	b9m5	b9m10	b9m20
	-12.84	-23.34	-15.50	-17.66	-15.63	-17.66	-16.12	-18.05	-19.40
	-15.50	-26.91	-15.63	-19.40	-18.05	-19.40	-18.05	-22.80	-21.80
	-15.50	-26.98	-18.46	-21.53	-19.40	-21.53	-20.75	-22.80	-22.62
	-22.62	-26.98	-19.40	-22.80	-23.34	-22.80	-24.24	-24.24	-23.29
	-24.24	-29.72	-19.53	-24.24	-24.24	-24.24	-33.69	-26.91	-23.34
	-29.71	-33.69	-23.34	-24.24	-28.28	-24.24	-36.59	-29.71	-23.34
	-33.69	-33.69	-26.91	-33.69	-28.28	-33.69	-36.59	-33.69	-26.91
	-36.59	-33.69	-28.28	-33.69	-29.71	-33.69	-39.85	-33.69	-29.71
	-39.85	-39.85	-33.69	-39.85	-33.69	-39.85	-39.85	-36.59	-33.69
	-39.85	-39.85	-39.85	-39.85	-36.59	-39.85	-39.85	-39.85	-33.69
Median	-26.97	-31.70	-21.43	-24.24	-26.26	-24.24	-35.14	-28.31	-23.34
IQR	21.08	6.716	9.821	12.16	10.31	12.16	19.10	10.89	7.092
W	0.888	0.887	0.864	0.854	0.936	0.854	0.824	0.930	0.846
Rank sum	434	593.5	335	450.5	402	450.5	544.5	495	390
H	7.430								

Table F.6 EPSOC evaluation – The radio-frequency design case (Bead)

G.0 Objective function values for Genesis 5.0

The following tables contain final objective function values delivered by 10 simultaneous runs of Genesis 5.0 with given operational parameters, from random starting points. Table column labels are of the format: **c** *crossover rate* **m** *mutation rate,* where crossover rates are 0.3, 0.6 and 0.9, and mutation rates are 0.001, 0.01 and 0.1, respectively. Also tabulated for each group are the median, interquartile range, Shapiro-Wilk W test statistic, rank sum across the pooled ranks, and Kruskal-Wallis H test statistic.

	c3m001	c3m01	c3m1	c6m001	c6m01	c6m1	c9m001	c9m01	c9m1
	-0.212	-0.349	-0.468	-0.403	-0.455	-0.476	-0.465	-0.455	-0.472
	-0.324	-0.392	-0.468	-0.436	-0.463	-0.476	-0.469	-0.461	-0.474
	-0.370	-0.422	-0.470	-0.467	-0.466	-0.476	-0.476	-0.474	-0.475
	-0.418	-0.427	-0.473	-0.468	-0.474	-0.478	-0.476	-0.479	-0.477
	-0.418	-0.449	-0.477	-0.477	-0.477	-0.478	-0.479	-0.479	-0.478
	-0.433	-0.458	-0.477	-0.478	-0.478	-0.479	-0.480	-0.479	-0.478
	-0.453	-0.462	-0.479	-0.478	-0.478	-0.479	-0.480	-0.480	-0.479
	-0.465	-0.470	-0.480	-0.479	-0.479	-0.479	-0.480	-0.480	-0.479
	-0.470	-0.474	-0.480	-0.480	-0.480	-0.479	-0.481	-0.480	-0.480
	-0.473	-0.479	-0.480	-0.480	-0.480	-0.480	-0.481	-0.481	-0.481
Median	-0.426	-0.453	-0.477	-0.477	-0.477	-0.479	-0.479	-0.479	-0.478
IQR	0.082	0.041	0.005	0.025	0.008	0.002	0.006	0.009	0.003
W	0.809	0.864	0.835	0.674	0.789	0.856	0.771	0.674	0.914
Rank sum	133	199	507	455	453	570	620	582	576
H	35.626								

Table G.1 GA evaluation – The quantum electro-dynamical case (Laser 1)

	c3m001	c3m01	c3m1	c6m001	c6m01	c6m1	c9m001	c9m01	c9m1
	-0.280	-0.505	-0.552	-0.476	-0.530	-0.556	-0.544	-0.542	-0.547
	-0.351	-0.523	-0.553	-0.507	-0.539	-0.556	-0.545	-0.549	-0.554
	-0.474	-0.534	-0.555	-0.519	-0.548	-0.558	-0.554	-0.554	-0.555
	-0.475	-0.542	-0.556	-0.545	-0.549	-0.558	-0.556	-0.558	-0.555
	-0.503	-0.555	-0.556	-0.546	-0.553	-0.558	-0.558	-0.559	-0.557
	-0.530	-0.555	-0.557	-0.552	-0.554	-0.558	-0.560	-0.560	-0.559
	-0.531	-0.556	-0.559	-0.555	-0.555	-0.558	-0.561	-0.561	-0.559
	-0.552	-0.558	-0.559	-0.556	-0.557	-0.559	-0.561	-0.562	-0.559
	-0.558	-0.562	-0.562	-0.557	-0.557	-0.559	-0.561	-0.562	-0.560
	-0.561	-0.562	-0.562	-0.559	-0.561	-0.559	-0.561	-0.562	-0.561
Median	-0.516	-0.555	-0.557	-0.549	-0.554	-0.558	-0.559	-0.559	-0.558
IQR	0.078	0.024	0.005	0.037	0.009	0.001	0.007	0.008	0.005
W	0.798	0.820	0.936	0.773	0.865	0.792	0.771	0.801	0.860
Rank sum	200	400	550	288	361	588	574	601	533
H	25.355								

Table G.2 GA evaluation – The quantum electro-dynamical case (Laser 2)

	c3m001	c3m01	c3m1	c6m001	c6m01	c6m1	c9m001	c9m01	c9m1
	196.62	195.65	194.01	192.78	195.44	196.36	189.97	192.28	195.39
	193.49	193.84	192.74	192.42	192.59	191.93	189.94	192.20	193.13
	193.00	193.67	191.68	191.96	190.10	190.67	189.72	190.91	191.89
	192.88	193.47	191.66	191.13	189.71	189.89	189.64	190.71	191.44
	192.06	192.98	190.50	190.80	189.58	189.87	189.00	190.35	190.55
	191.88	191.62	190.47	190.63	189.47	189.66	188.56	189.21	190.36
	191.76	189.84	190.20	190.57	188.74	189.25	188.43	189.13	190.32
	190.98	189.37	189.68	190.11	188.54	188.91	188.23	189.01	189.44
	190.10	189.26	189.54	188.47	187.81	188.89	188.04	188.94	189.32
	188.78	188.08	189.08	188.42	187.81	188.34	187.83	188.78	188.88
Median	191.97	192.30	190.48	190.71	189.52	189.76	188.78	189.78	190.46
IQR	2.022	4.303	2.004	1.852	1.559	1.761	1.484	1.904	2.453
W	0.921	0.905	0.911	0.901	0.810	0.742	0.873	0.844	0.885
Rank sum	264	337	388	405	581	518	703	503	396
H	21.18								

Table G.3 GA evaluation – The durable component design case using stress (Crack 1)

	c3m001	c3m01	c3m1	c6m001	c6m01	c6m1	c9m001	c9m01	c9m1
	5290	5284	5294	5291	5286	5299	5298	5305	5308
	5296	5287	5319	5291	5296	5302	5317	5310	5321
	5298	5303	5322	5313	5302	5302	5321	5326	5322
	5300	5309	5324	5318	5306	5307	5323	5328	5328
	5305	5311	5328	5324	5306	5320	5329	5342	5331
	5317	5312	5328	5324	5313	5328	5341	5343	5332
	5319	5317	5329	5337	5325	5334	5342	5344	5332
	5328	5317	5334	5339	5331	5345	5347	5346	5333
	5330	5329	5339	5343	5332	5348	5353	5352	5346
	5350	5333	5344	5351	5347	5350	5354	5353	5348
Median	5311	5312	5328	5324	5310	5324	5335	5342	5332
IQR	29.57	14.20	11.81	25.09	28.47	42.99	25.53	20.67	11.72
W	0.885	0.918	0.861	0.905	0.929	0.864	0.924	0.866	0.917
Rank sum	310	281	491	458	338	464	578	614	561
H	17.293								

Table G.4 GA evaluation – The durable component design case using fatigue life (Crack 2)

	c3m001	c3m01	c3m1	c6m001	c6m01	c6m1	c9m001	c9m01	c9m1
	-64.329	-65.220	-66.458	-66.245	-65.874	-66.160	-66.401	-65.585	-66.302
	-64.606	-65.419	-67.686	-66.675	-66.926	-66.387	-66.924	-66.018	-66.476
	-64.661	-65.822	-67.745	-66.924	-66.975	-66.510	-67.651	-66.564	-66.873
	-65.142	-66.125	-67.908	-67.429	-67.638	-66.620	-67.764	-66.962	-67.374
	-65.900	-66.456	-67.933	-67.471	-67.778	-66.621	-67.919	-67.107	-67.451
	-66.026	-66.782	-68.011	-67.552	-67.841	-66.868	-68.029	-68.126	-67.585
	-66.145	-67.072	-68.179	-67.610	-68.026	-67.998	-68.179	-68.141	-67.794
	-67.065	-67.540	-68.278	-68.244	-68.158	-68.322	-68.198	-68.191	-68.165
	-67.839	-68.071	-68.383	-68.325	-68.233	-68.369	-68.401	-68.216	-68.203
	-68.416	-68.238	-68.556	-68.497	-68.410	-68.439	-68.564	-68.289	-68.271
Median	-65.963	-66.619	-67.972	-67.512	-67.810	-66.745	-67.974	-67.617	-67.518
IQR	2.405	1.718	0.534	1.320	1.183	1.812	0.547	1.627	1.292
W	0.894	0.926	0.810	0.926	0.864	0.801	0.873	0.840	0.902
Rank sum	222	298	613	492	517	444	581	465	463
H	18.355								

Table G.5 GA evaluation – The 2D aerofoil design case (Aerofoil)

	c3m001	c3m01	c3m1	c6m001	c6m01	c6m1	c9m001	c9m01	c9m1
	-6.70	-5.92	-19.40	-10.67	-6.70	-17.66	-21.53	-6.70	-17.66
	-7.90	-8.32	-26.99	-11.01	-8.97	-26.98	-23.34	-9.72	-17.66
	-9.42	-8.97	-28.28	-13.44	-17.66	-26.98	-24.24	-17.66	-24.24
	-10.52	-16.94	-29.71	-22.62	-17.66	-29.71	-28.28	-21.80	-26.98
	-14.70	-17.66	-33.69	-23.78	-19.61	-33.69	-29.71	-24.24	-36.59
	-17.66	-17.66	-39.85	-24.24	-22.07	-39.85	-29.71	-29.71	-39.85
	-21.61	-20.75	-39.85	-26.92	-24.24	-39.85	-29.71	-29.71	-39.85
	-21.80	-22.07	-39.85	-33.69	-29.71	-39.85	-33.69	-33.69	-39.85
	-23.78	-22.62	-39.85	-33.69	-33.69	-39.85	-36.59	-39.85	-39.85
	-39.85	-29.71	-39.85	-39.85	-33.69	-39.85	-39.85	-39.85	-39.85
Median	12.39	-15.65	-33.05	-22.23	-20.03	-32.71	-28.53	-23.67	-31.39
IQR	10.03	7.426	7.334	10.042	9.319	7.832	5.826	11.52	9.577
W	0.082	0.907	0.813	0.902	0.919	0.808	0.916	0.924	0.756
Rank sum	244.5	225.5	647	409.5	340	639	541	445	603.5
H	31.142								

Table G.6 GA evaluation – The radio-frequency design case (Bead)

H.0 Objective function values for 9 algorithms

The following tables contain final objective function values delivered by 10 simultaneous runs of 9 selected algorithms, from random starting points. Also tabulated are the median result for each algorithm, interquartile range, Shapiro-Wilk W test statistic, rank sum of each algorithm across the pooled ranks, and Kruskal-Wallis H test statistic.

	EPSOC	GA	P-BFGS	NM	NML1	MDS	RSCS	RSCSL	RSCSL1
	-0.477	-0.465	0.101	-0.476	0.129	-0.476	0.412	0.411	0.412
	-0.480	-0.469	0.063	-0.479	0.008	-0.479	0.380	0.309	0.119
	-0.481	-0.476	-0.053	-0.480	-0.051	-0.480	0.343	0.145	0.009
	-0.481	-0.476	-0.104	-0.481	-0.255	-0.481	0.096	0.078	-0.285
	-0.481	-0.479	-0.435	-0.481	-0.445	-0.481	-0.356	-0.138	-0.459
	-0.481	-0.480	-0.462	-0.481	-0.457	-0.481	-0.470	-0.242	-0.475
	-0.481	-0.480	-0.469	-0.481	-0.465	-0.481	-0.479	-0.331	-0.478
	-0.481	-0.480	-0.470	-0.481	-0.471	-0.482	-0.482	-0.456	-0.479
	-0.482	-0.481	-0.471	-0.482	-0.473	-0.482	-0.482	-0.479	-0.479
	-0.482	-0.481	-0.480	-0.482	-0.478	-0.482	-0.482	-0.482	-0.481
Median	-0.481	-0.479	-0.448	-0.481	-0.451	-0.481	-0.413	-0.190	-0.467
IQR	0.001	0.004	0.417	0.002	0.419	0.001	0.825	0.601	0.487
W	0.695	0.771	0.747	0.744	0.760	0.648	0.733	0.889	0.737
Rank sum	685.0	515.0	282.0	669.5	267.0	698.0	385.5	261.0	332.0
H	41.674								

Table H.1 Algorithm comparison – The quantum electro-dynamical case (Laser 1)

	EPSOC	GA	P-BFGS	NM	NML1	MDS	RSCS	RSCSL	RSCSL1
	-0.558	-0.542	0.340	0.496	0.334	0.497	0.391	0.280	0.286
	-0.560	-0.549	0.319	0.314	0.253	0.309	0.314	0.273	0.283
	-0.561	-0.554	0.307	0.286	0.120	0.305	0.284	0.242	0.245
	-0.561	-0.558	0.306	0.274	0.008	0.289	0.284	-0.036	-0.276
	-0.562	-0.559	0.302	0.274	-0.018	0.279	-0.024	-0.533	-0.365
	-0.562	-0.560	0.295	-0.561	-0.222	0.278	-0.475	-0.551	-0.555
	-0.562	-0.561	0.012	-0.562	-0.237	-0.560	-0.562	-0.559	-0.556
	-0.562	-0.562	-0.004	-0.562	-0.486	-0.561	-0.562	-0.559	-0.560
	-0.562	-0.562	-0.184	-0.562	-0.559	-0.562	-0.563	-0.559	-0.562
	-0.562	-0.562	-0.559	-0.562	-0.562	-0.562	-0.563	-0.563	-0.563
Median	-0.562	-0.559	0.298	-0.144	-0.120	0.279	-0.250	-0.542	-0.460
IQR	0.001	0.008	0.312	0.848	0.606	0.866	0.847	0.801	0.806
W	0.752	0.801	0.718	0.720	0.906	0.715	0.759	0.701	0.728
Rank sum	686.0	580.0	213.0	464.5	384.0	359.5	469.0	461.0	478.0
H	20.888								

Table H.2 Algorithm comparison – The quantum electro-dynamical case (Laser 2)

	EPSOC	GA	P-BFGS	NM	NML1	MDS	RSCS	RSCSL	RSCSL1
	187.883	189.973	232.298	199.229	219.562	211.372	212.046	204.170	201.833
	187.846	189.939	218.293	199.194	208.204	199.236	199.525	201.957	201.423
	187.783	189.717	208.792	199.076	199.537	199.168	199.067	200.918	200.718
	187.756	189.643	199.442	199.067	199.153	199.092	198.972	200.786	199.821
	187.716	189.000	197.716	195.801	197.904	188.005	195.176	200.387	195.031
	187.689	188.561	195.772	188.053	195.980	187.973	191.794	194.970	190.153
	187.641	188.430	192.939	187.890	195.880	187.787	188.060	194.628	190.136
	187.636	188.233	189.322	187.762	192.948	187.648	187.787	192.641	187.922
	187.552	188.044	187.549	187.615	191.093	187.636	187.603	188.252	187.735
	187.550	187.826	187.547	187.614	187.549	187.587	187.557	187.548	187.654
Median	187.703	188.781	196.744	191.927	196.942	187.989	193.485	197.679	192.592
IQR	0.147	1.484	19.470	11.314	6.589	11.520	11.280	8.277	12.796
W	0.933	0.873	0.849	0.712	0.867	0.737	0.839	0.880	0.809
Rank sum	742.5	534.0	364.5	482.5	316.5	499.0	448.0	320.0	388.0
H	20.766								

Table H.3 Algorithm comparison – The durable component design case using stress (Crack 1)

	EPSOC	GA	P-BFGS	NM	NML1	MDS	RSCS	RSCSL	RSCSL1
	5316	5305	4098	4411	5236	4878	4881	4405	4399
	5331	5310	4464	5264	5253	5253	5293	5007	5271
	5348	5326	4464	5265	5265	5294	5295	5292	5272
	5352	5328	4630	5293	5269	5303	5299	5292	5280
	5352	5342	4808	5297	5283	5319	5310	5294	5291
	5355	5343	4854	5300	5294	5319	5311	5297	5292
	5355	5344	4936	5311	5299	5319	5319	5300	5296
	5356	5346	4975	5319	5319	5319	5333	5304	5319
	5356	5352	5247	5319	5326	5319	5348	5348	5319
	5356	5353	5356	5353	5332	5357	5348	5357	5348
Median	5353	5342	4831	5299	5288	5319	5311	5295	5292
IQR	7.762	20.668	510.839	54.467	53.545	25.175	38.148	12.040	46.806
W	0.697	0.866	0.946	0.453	0.937	0.525	0.500	0.567	0.445
Rank sum	776.5	665.0	164.5	403.0	358.5	484.0	490.0	397.0	356.5
H	37.949								

Table H.4 Algorithm comparison – The durable component design case using fatigue life (Crack 2)

	EPSOC	GA	P-BFGS	NM	NML1	MDS	RSCS	RSCSL	RSCSL1
	-68.632	-66.458	-57.330	-44.736	-44.760	-44.763	-44.656	-44.637	-44.599
	-68.633	-67.686	-63.313	-66.251	-54.037	-66.202	-58.097	-53.356	-54.007
	-68.637	-67.745	-64.148	-67.255	-57.182	-66.218	-61.732	-54.078	-55.778
	-68.638	-67.908	-65.508	-67.323	-60.996	-66.257	-67.323	-55.579	-57.279
	-68.638	-67.933	-66.230	-67.323	-64.959	-67.516	-68.506	-57.428	-58.065
	-68.638	-68.011	-66.441	-68.375	-67.323	-68.332	-68.609	-58.552	-62.521
	-68.639	-68.179	-67.390	-68.459	-68.559	-68.586	-68.619	-65.622	-65.049
	-68.640	-68.278	-68.640	-68.626	-68.617	-68.629	-68.622	-65.988	-67.323
	-68.641	-68.383	-68.642	-68.641	-68.627	-68.636	-68.635	-66.234	-67.323
	-68.641	-68.556	-68.642	-68.643	-68.632	-68.638	-68.643	-67.207	-68.622
Median	-68.638	-67.972	-66.336	-67.849	-66.141	-67.924	-68.558	-57.990	-60.293
IQR	0.003	0.534	4.492	1.372	11.434	2.411	6.889	11.910	11.546
W	0.871	0.810	0.819	0.459	0.804	0.486	0.643	0.888	0.903
Rank sum	789.0	511.0	460.0	519.0	379.5	491.0	500.5	193.0	252.0
H	34.833								

Table H.5 Algorithm comparison – The 2D aerofoil design case (Aerofoil)

	EPSOC	GA	P-BFGS	NM	NML1	MDS	RSCS	RSCSL	RSCSL1
	-23.335	-19.400	22.756	37.715	24.504	15.083	37.715	13.840	15.083
	-26.914	-26.977	15.826	28.292	10.354	13.840	15.083	1.723	13.840
	-26.977	-28.281	11.612	15.083	-11.795	13.451	12.652	-6.282	9.868
	-26.977	-29.707	10.114	13.840	-12.841	3.909	12.570	-11.795	3.909
	-29.707	-33.693	-3.126	5.086	-14.806	3.711	9.868	-12.841	1.723
	-33.693	-39.849	-9.388	3.711	-16.547	-0.455	3.711	-13.818	-6.282
	-33.693	-39.849	-13.083	3.608	-18.046	-2.464	3.608	-13.818	-11.795
	-33.693	-39.849	-14.806	-10.798	-22.615	-7.230	0.756	-14.806	-12.841
	-39.849	-39.849	-16.115	-13.005	-24.240	-11.527	-21.530	-19.610	-19.610
	-39.849	-39.849	-29.707	-26.977	-39.849	-16.115	-26.914	-26.977	-26.977
Median	-31.700	-36.771	-6.257	4.398	-15.676	1.628	6.789	-13.329	-2.279
IQR	6.716	11.568	26.418	25.881	10.820	20.682	11.896	8.524	22.709
W	0.887	0.813	0.916	0.942	0.872	0.907	0.885	0.887	0.915
Rank sum	760.000	807.000	363.000	266.000	499.000	294.000	284.500	462.000	359.500
H	47.943								

Table H.6 Algorithm comparison – The radio-frequency design case (Bead)

I.0 Pair-wise comparison of EPSOC against other algorithms

The following tables contain final objective function values delivered by 10 simultaneous runs, from random starting points, of EPSOC and another algorithm, chosen on the basis of relative rank in tests of multiple algorithms, for each test case. Also tabulated is the Mann-Whitney U test statistic, and the corresponding significance level, assuming a two-tailed test.

	Laser 1	
	EPSOC	**MDS**
	-0.477	-0.476
	-0.480	-0.479
	-0.481	-0.480
	-0.481	-0.481
	-0.481	-0.481
	-0.481	-0.481
	-0.481	-0.481
	-0.481	-0.482
	-0.482	-0.482
	-0.482	-0.482
Rank sum	99.000	111.000
U	44.000	
P	0.6842	

Table I.1 Pair-wise comparison of EPSOC against other algorithms – Laser 1 test case

	Laser 2	
	EPSOC	**GA**
	-0.558	-0.542
	-0.560	-0.549
	-0.561	-0.554
	-0.561	-0.558
	-0.562	-0.559
	-0.562	-0.560
	-0.562	-0.561
	-0.562	-0.562
	-0.562	-0.562
	-0.562	-0.562
Rank sum	132.000	78.000
U	23.000	
P	0.0432	

Table I.2 Pair-wise comparison of EPSOC against other algorithms – Laser 2 test case

	Crack 1	
	EPSOC	**GA**
	187.883	189.972
	187.846	189.938
	187.783	189.717
	187.756	189.642
	187.716	189.000
	187.689	188.561
	187.641	188.430
	187.636	188.233
	187.552	188.043
	187.55	187.825
Rank sum	153	57
U	2.000	
P	0.0000	

Table I.3 Pair-wise comparison of EPSOC against other algorithms – Crack 1 test case

	Crack 2	
	EPSOC	**GA**
	5316	5305
	5331	5310
	5348	5326
	5352	5328
	5352	5342
	5355	5343
	5355	5344
	5356	5346
	5356	5352
	5356	5353
Rank sum	135	75
U	20.000	
P	0.0232	

Table I.4 Pair-wise comparison of EPSOC against other algorithms – Crack 2 test case

	Aerofoil	
	EPSOC	**NM**
	-68.632	-44.736
	-68.6327	-66.251
	-68.6368	-67.255
	-68.6377	-67.323
	-68.6379	-67.323
	-68.6383	-68.375
	-68.6386	-68.459
	-68.6402	-68.626
	-68.6406	-68.641
	-68.6413	-68.643
Rank sum	136	74
U	19.000	
P	0.0186	

Table I.5 Pair-wise comparison of EPSOC against other algorithms – Aerofoil test case

	Bead	
	EPSOC	**GA**
	-23.335	-19.400
	-26.914	-26.977
	-26.977	-28.281
	-26.977	-29.707
	-29.707	-33.693
	-33.693	-39.849
	-33.693	-39.849
	-33.693	-39.849
	-39.849	-39.849
	-39.849	-39.849
Rank sum	85.000	125.000
U	30.000	
P	0.1432	

Table I.6 Pair-wise comparison of EPSOC against other algorithms – Bead test case

J.0 One step spatial preconditioning – results of numerical experiments

	Laser 1		**Laser 2**		**Aerofoil**	
	Original	**Precon**	**Original**	**Precon**	**Original**	**Precon**
	0.016	-0.009	0.296	0.471	-57.32	-53.95
	-0.118	-0.070	0.286	0.302	-63.31	-54.77
	-0.249	-0.130	0.253	0.255	-64.14	-57.87
	-0.387	-0.347	0.215	0.101	-65.50	-59.36
	-0.396	-0.398	-0.031	0.093	-66.23	-65.01
	-0.420	-0.476	-0.242	0.070	-66.44	-65.22
	-0.465	-0.479	-0.267	-0.296	-67.38	-65.45
	-0.480	-0.479	-0.414	-0.381	-68.63	-66.27
	-0.480	-0.480	-0.542	-0.540	-68.64	-68.37
	-0.481	-0.480	-0.562	-0.562	-68.64	-68.64
Rank sum	108.0	102.0	112.0	107.0	123.0	87.0
U	47.0		43.0		32.0	
P	0.8534		0.6306		0.315	

	Bead		**Crack 1**		**Crack 2**	
	Original	**Precon**	**Original**	**Precon**	**Original**	**Precon**
	22.756	21.134	232.2	206.1	4098	4049
	15.826	20.709	218.2	205.4	4464	4329
	11.612	17.019	208.7	204.0	4464	4406
	10.114	-1.536	199.4	199.8	4630	4637
	-3.126	-11.527	197.7	199.6	4808	4926
	-9.388	-16.115	195.7	198.3	4854	4927
	-13.083	-23.782	192.9	197.9	4975	4950
	-14.806	-33.693	189.3	195.8	5007	5000
	-16.115	-39.849	187.5	194.6	5247	5002
	-29.707	-39.849	187.5	192.9	5356	5025
Rank sum	94.5	115.5	114.0	96.0	109.0	101.0
U	39.5		41.0		46.0	
P	0.4358		0.5288		0.796	

Table J.1 Mann-Whitney pair-wise comparison of BFGS results for one step preconditioned and unconditioned test cases

	Laser 1		**Laser 2**		**Aerofoil**	
	Original	**Precon**	**Original**	**Precon**	**Original**	**Precon**
	-0.476	0.301	0.496	0.297	-44.73	-58.53
	-0.479	-0.473	0.314	0.277	-66.25	-64.78
	-0.480	-0.481	0.286	0.275	-67.25	-65.64
	-0.481	-0.481	0.274	-0.555	-67.32	-66.02
	-0.481	-0.481	0.274	-0.562	-67.32	-68.61
	-0.481	-0.481	-0.561	-0.562	-68.37	-68.61
	-0.481	-0.481	-0.562	-0.562	-68.45	-68.62
	-0.481	-0.481	-0.562	-0.562	-68.62	-68.63
	-0.482	-0.481	-0.562	-0.562	-68.64	-68.63
	-0.482	-0.482	-0.562	-0.562	-68.64	-68.64
Rank sum	104.0	106.0	94.0	116.0	105.0	105.0
U	49.0		39.0		50.0	
P	0.9706		0.4358		1.0	
	Bead		**Crack 1**		**Crack 2**	
	Original	**Precon**	**Original**	**Precon**	**Original**	**Precon**
	37.715	32.977	199.2	213.2	4411	5236
	28.292	31.727	199.1	200.1	5264	5236
	15.083	17.662	199.0	199.4	5265	5236
	13.840	17.019	199.0	199.1	5293	5248
	5.086	16.671	195.8	199.1	5297	5248
	3.711	16.022	188.0	199.1	5300	5248
	3.608	-6.311	187.8	199.0	5311	5248
	-10.798	-10.520	187.7	195.7	5319	5278
	-13.005	-28.281	187.6	193.6	5319	5279
	-26.977	-29.707	187.6	192.9	5353	5287
Rank sum	111.0	99.0	131.0	79.0	139.0	71.0
U	44.0		24.0		16.0	
P	0.6842		0.0524		0.0090	

Table J.2 Mann-Whitney pair-wise comparison of Nelder-Mead Simplex results for one step pre-conditioned and unconditioned test cases

	Laser 1		**Laser 2**		**Aerofoil**	
	Original	**Precon**	**Original**	**Precon**	**Original**	**Precon**
	0.242	0.323	0.333	-0.022	-44.76	-58.04
	0.045	0.293	0.310	-0.368	-54.61	-62.44
	-0.416	0.020	0.278	-0.369	-56.55	-64.58
	-0.432	-0.218	0.274	-0.405	-64.96	-67.55
	-0.455	-0.269	0.262	-0.507	-65.60	-67.59
	-0.463	-0.406	-0.187	-0.526	-65.79	-67.72
	-0.479	-0.479	-0.559	-0.536	-68.36	-67.96
	-0.482	-0.482	-0.559	-0.558	-68.53	-68.33
	-0.482	-0.482	-0.562	-0.559	-68.62	-68.40
	-0.482	-0.482	-0.563	-0.563	-68.64	-68.49
Rank sum	113.0	97.0	93.0	117.0	102.0	108.0
U	42.0		38.0		47.0	
P	0.5788		0.3930		0.8534	

	Bead		**Crack 1**		**Crack 2**	
	Original	**Precon**	**Original**	**Precon**	**Original**	**Precon**
	24.504	24.424	219.5	209.8	4405	5000
	10.354	17.662	208.1	202.0	5239	5000
	-11.795	17.019	199.5	199.6	5263	5181
	-12.681	12.461	199.1	199.5	5264	5182
	-12.841	-9.374	197.5	199.4	5291	5236
	-14.806	-11.843	195.6	199.4	5295	5243
	-16.547	-23.782	191.2	199.0	5298	5244
	-18.046	-26.977	188.5	197.9	5322	5245
	-22.615	-28.281	187.6	197.0	5330	5247
	-39.849	-33.693	187.5	195.1	5333	5291
Rank sum	110.0	100.0	122.0	88.0	137.5	72.5
U	45.0		33.0		17.5	
P	0.7394		0.2176		0.0146	

Table J.3 Mann-Whitney pair-wise comparison of Nelder-Mead Simplex with line search results for one step preconditioned and unconditioned test cases

	Laser 1		**Laser 2**		**Aerofoil**	
	Original	**Precon**	**Original**	**Precon**	**Original**	**Precon**
	0.129	0.323	0.334	0.254	-44.76	-59.51
	0.008	0.311	0.253	0.253	-54.04	-64.89
	-0.051	-0.006	0.120	-0.100	-57.18	-66.90
	-0.255	-0.016	0.008	-0.539	-61.00	-67.50
	-0.445	-0.095	-0.018	-0.548	-64.96	-67.70
	-0.457	-0.354	-0.222	-0.557	-67.32	-67.85
	-0.465	-0.445	-0.237	-0.560	-68.56	-68.03
	-0.471	-0.454	-0.486	-0.560	-68.62	-68.21
	-0.473	-0.454	-0.559	-0.562	-68.63	-68.36
	-0.478	-0.477	-0.562	-0.563	-68.63	-68.44
Rank sum	120.0	90.0	86.0	124.0	101.0	109.0
U	35.0		31.0		46.0	
P	0.2798		0.1654		0.7960	

	Bead		**Crack 1**		**Crack 2**	
	Original	**Precon**	**Original**	**Precon**	**Original**	**Precon**
	24.504	24.424	219.6	210.0	5236	5000
	10.354	17.662	208.2	201.7	5253	5226
	-11.795	17.019	199.5	199.9	5265	5236
	-12.841	12.461	199.2	199.7	5269	5236
	-14.806	-13.049	197.9	199.4	5283	5244
	-16.547	-23.782	196.0	199.3	5294	5244
	-18.046	-26.977	195.9	199.0	5299	5244
	-22.615	-26.977	192.9	198.9	5319	5246
	-24.240	-28.281	191.1	198.0	5326	5248
	-39.849	-29.707	187.5	195.0	5332	5291
Rank sum	103.0	107.0	123.0	87.0	144.0	66.0
U	48.0		32.0		11.0	
P	0.9198		0.1904		0.0020	

Table J.4 Mann-Whitney pair-wise comparison of Nelder-Mead Simplex with single-pass line search results for one step preconditioned and unconditioned test cases

	Laser 1		**Laser 2**		**Aerofoil**	
	Original	**Precon**	**Original**	**Precon**	**Original**	**Precon**
	-0.476	-0.481	0.497	0.277	-44.76	-54.53
	-0.479	-0.481	0.309	-0.558	-66.20	-59.16
	-0.480	-0.481	0.305	-0.559	-66.22	-66.08
	-0.481	-0.481	0.289	-0.561	-66.26	-66.18
	-0.481	-0.481	0.279	-0.562	-67.52	-67.44
	-0.481	-0.481	0.278	-0.562	-68.33	-68.54
	-0.481	-0.481	-0.560	-0.562	-68.59	-68.58
	-0.482	-0.481	-0.561	-0.562	-68.63	-68.61
	-0.482	-0.482	-0.562	-0.562	-68.64	-68.62
	-0.482	-0.482	-0.562	-0.563	-68.64	-68.64
Rank sum	97.0	113.0	75.0	135.0	112.0	98.0
U	42.0		20.0		43.0	
P	0.5788		0.0232		0.6306	

	Bead		**Crack 1**		**Crack 2**	
	Original	**Precon**	**Original**	**Precon**	**Original**	**Precon**
	15.083	33.978	211.4	206.1	4878	5004
	13.840	32.998	199.2	205.1	5253	5208
	13.451	25.337	199.2	199.8	5294	5232
	3.909	24.329	199.1	199.8	5294	5236
	3.711	18.145	188.0	199.7	5303	5236
	-0.455	17.019	188.0	199.2	5319	5236
	-2.464	-11.588	187.8	199.1	5319	5236
	-7.230	-13.049	187.6	199.1	5319	5248
	-11.527	-15.595	187.6	199.1	5319	5263
	-16.115	-29.707	187.6	193.1	5319	5263
Rank sum	118.0	92.0	133.0	77.0	143.0	67.0
U	37.0		22.0		12.0	
P	0.3526		0.0354		0.0028	

Table J.5 Mann-Whitney pair-wise comparison of MDS results for one step preconditioned and unconditioned test cases

	Laser 1		**Laser 2**		**Aerofoil**	
	Original	**Precon**	**Original**	**Precon**	**Original**	**Precon**
	0.412	-0.481	0.355	0.277	-44.66	-58.20
	0.390	-0.481	0.314	-0.024	-58.10	-59.45
	-0.476	-0.481	0.295	-0.041	-61.73	-65.50
	-0.481	-0.481	0.094	-0.554	-67.32	-67.32
	-0.481	-0.482	-0.071	-0.557	-68.51	-67.39
	-0.482	-0.482	-0.501	-0.562	-68.61	-68.55
	-0.482	-0.482	-0.555	-0.562	-68.62	-68.62
	-0.482	-0.482	-0.562	-0.562	-68.62	-68.63
	-0.482	-0.482	-0.563	-0.562	-68.64	-68.63
	-0.482	-0.482	-0.563	-0.563	-68.64	-68.64
Rank sum	94.0	116.0	91.0	119.0	102.5	107.5
U	39.0		36.0		47.5	
P	0.4358		0.3150		0.9118	
	Bead		**Crack 1**		**Crack 2**	
	Original	**Precon**	**Original**	**Precon**	**Original**	**Precon**
	37.715	35.118	212.0	201.6	4881	5236
	15.083	31.727	199.5	199.4	5293	5248
	12.652	17.662	199.1	199.2	5295	5248
	9.916	12.697	199.0	199.1	5299	5263
	3.608	5.919	195.2	199.1	5310	5263
	1.214	-11.588	191.8	195.0	5311	5263
	-11.795	-14.242	188.1	193.1	5319	5265
	-13.818	-16.115	187.8	192.9	5333	5291
	-21.801	-29.707	187.6	192.9	5348	5291
	-26.914	-33.693	187.6	192.9	5348	5291
Rank sum	104.0	106.0	120.0	90.0	145.0	65.0
U	49.0		35.0		10.0	
P	0.9706		0.2798		0.0016	

Table J.6 Mann-Whitney pair-wise comparison of RSCS results for one step preconditioned and unconditioned test cases

	Laser 1		**Laser 2**		**Aerofoil**	
	Original	**Precon**	**Original**	**Precon**	**Original**	**Precon**
	0.411	0.178	0.280	0.266	-44.64	-60.21
	0.309	-0.077	0.273	0.124	-53.36	-62.11
	0.145	-0.149	0.242	0.093	-54.08	-62.43
	0.078	-0.332	-0.036	-0.071	-55.58	-63.35
	-0.138	-0.448	-0.533	-0.243	-57.43	-63.70
	-0.242	-0.449	-0.551	-0.531	-58.55	-64.07
	-0.331	-0.482	-0.559	-0.553	-65.62	-65.44
	-0.456	-0.482	-0.559	-0.559	-65.99	-66.03
	-0.479	-0.482	-0.559	-0.559	-66.23	-68.10
	-0.482	-0.482	-0.563	-0.563	-67.21	-68.29
Rank sum	84.5	142.0	103.0	107.0	85.0	125.0
U	13.0		48.0		30.0	
P	0.0038		0.9118		0.1432	

	Bead		**Crack 1**		**Crack 2**	
	Original	**Precon**	**Original**	**Precon**	**Original**	**Precon**
	13.840	24.424	204.2	203.2	4405	5000
	1.723	17.832	202.0	200.5	5007	5000
	-6.282	17.662	200.9	200.2	5292	5000
	-11.795	16.022	200.8	199.5	5292	5245
	-12.841	-11.014	200.4	199.4	5294	5245
	-13.818	-11.843	195.0	199.3	5297	5248
	-13.818	-17.153	194.6	196.2	5300	5263
	-14.806	-19.400	192.6	195.9	5304	5269
	-19.610	-28.281	188.3	195.8	5348	5278
	-26.977	-39.849	187.5	195.6	5357	5291
Rank sum	112.0	98.0	110.0	100.0	138.0	72.0
U	43.0		45.0		17.0	
P	0.6306		0.7394		0.0114	

Table J.7 Mann-Whitney pair-wise comparison of RSCS with line search results for one step preconditioned and unconditioned test cases

	Laser 1		**Laser 2**		**Aerofoil**	
	Original	**Precon**	**Original**	**Precon**	**Original**	**Precon**
	0.412	0.212	0.286	0.148	-44.60	-60.71
	0.119	0.187	0.283	0.079	-54.01	-60.94
	0.009	-0.435	0.245	-0.542	-55.78	-61.35
	-0.285	-0.446	-0.276	-0.547	-57.28	-62.80
	-0.459	-0.448	-0.365	-0.548	-58.07	-64.09
	-0.475	-0.463	-0.555	-0.554	-62.52	-66.57
	-0.478	-0.471	-0.556	-0.560	-65.05	-68.20
	-0.479	-0.475	-0.560	-0.561	-67.32	-68.64
	-0.479	-0.481	-0.562	-0.562	-67.32	-68.64
	-0.481	-0.481	-0.563	-0.562	-68.62	-68.64
Rank sum	107.0	103.0	97.0	113.0	82.0	128.0
U	48.0		42.0		27.0	
P	0.9118		0.5788		0.0892	

	Bead		**Crack 1**		**Crack 2**	
	Original	**Precon**	**Original**	**Precon**	**Original**	**Precon**
	15.083	24.424	201.8	200.3	4399	4993
	13.840	17.832	201.4	200.0	5271	4993
	9.868	17.662	200.7	199.7	5272	4993
	3.909	16.022	199.8	199.6	5280	5242
	1.723	-9.991	195.0	199.4	5291	5244
	-6.282	-10.520	190.2	199.4	5292	5248
	-11.795	-11.014	190.1	199.1	5296	5268
	-12.841	-11.843	187.9	197.2	5319	5281
	-19.610	-19.400	187.7	195.8	5319	5291
	-26.977	-28.281	187.7	194.2	5348	5291
Rank sum	112.0	98.0	116.0	94.0	134.5	75.5
U	43.0		39.0		20.5	
P	0.6306		0.4358		0.0288	

Table J.8 Mann-Whitney pair-wise comparison of RSCS with single-pass line search results for one step preconditioned and unconditioned test cases

	Laser 1		Laser 2		Aerofoil	
	Original	**Precon**	**Original**	**Precon**	**Original**	**Precon**
	-0.477	-0.477	-0.558	-0.558	-68.632	-68.631
	-0.480	-0.477	-0.560	-0.561	-68.633	-68.635
	-0.481	-0.479	-0.561	-0.561	-68.637	-68.635
	-0.481	-0.479	-0.561	-0.561	-68.638	-68.636
	-0.481	-0.480	-0.562	-0.562	-68.638	-68.638
	-0.481	-0.480	-0.562	-0.562	-68.638	-68.639
	-0.481	-0.480	-0.562	-0.562	-68.639	-68.642
	-0.481	-0.481	-0.562	-0.562	-68.640	-68.642
	-0.482	-0.481	-0.562	-0.562	-68.641	-68.642
	-0.482	-0.481	-0.562	-0.562	-68.641	-68.643
Rank sum	133.0	77.0	105.0	105.0	96.5	113.5
U	22.0		50.0		41.5	
P	0.0354		1.0000		0.6842	

	Bead		Crack 1		Crack 2	
	Original	**Precon**	**Original**	**Precon**	**Original**	**Precon**
	-23.335	-36.585	187.9	193.4	5316	5283
	-26.914	-36.585	187.8	193.3	5331	5283
	-26.977	-39.849	187.8	193.3	5348	5287
	-26.977	-39.849	187.8	193.3	5348	5288
	-29.707	-39.849	187.7	193.2	5352	5289
	-33.693	-39.849	187.7	193.1	5352	5289
	-33.693	-39.849	187.6	193.1	5355	5289
	-33.693	-39.849	187.6	193.1	5355	5290
	-39.849	-39.849	187.6	193.1	5356	5291
	-39.849	-39.849	187.6	193.0	5356	5291
Rank sum	67.0	143.0	155.0	55.0	155.0	55.0
U	12.0		0.0		0.0	
P	0.0028		0.0000		0.0000	

Table J.9 Mann-Whitney pair-wise comparison of EPSOC results for one step preconditioned and unconditioned test cases

K.0 Two step spatial preconditioning – results of numerical experiments

	Laser 1		**Laser 2**		**Aerofoil**	
	Original	**Precon**	**Original**	**Precon**	**Original**	**Precon**
	0.016	0.268	0.296	-0.141	-57.32	-60.89
	-0.118	0.009	0.286	-0.296	-63.31	-62.18
	-0.249	-0.008	0.253	-0.317	-64.14	-62.53
	-0.387	-0.085	0.215	-0.351	-65.50	-64.40
	-0.396	-0.407	-0.031	-0.524	-66.23	-65.05
	-0.420	-0.408	-0.242	-0.526	-66.44	-65.15
	-0.465	-0.456	-0.267	-0.550	-67.38	-65.22
	-0.480	-0.476	-0.414	-0.553	-68.63	-66.03\
	-0.480	-0.476	-0.542	-0.558	-68.64	-68.30
	-0.481	-0.476	-0.562	-0.561	-68.64	-68.64
Rank sum	115.0	95.0	77.0	133.0	119.0	91.0
U	40.0		22.0		36.0	
P	0.4812		0.0354		0.3150	

	Bead		**Crack 1**		**Crack 2**	
	Original	**Precon**	**Original**	**Precon**	**Original**	**Precon**
	22.756	22.956	232.2	203.3	4098	4405
	15.826	-9.991	218.2	201.3	4464	4505
	11.612	-16.115	208.7	199.4	4464	4673
	10.114	-16.115	199.4	199.0	4630	4695
	-3.126	-29.707	197.7	199.0	4808	4878
	-9.388	-29.707	195.7	199.0	4854	4902
	-13.083	-29.707	192.9	197.9	4975	4929
	-14.806	-33.693	189.3	197.4	5007	4950
	-16.115	-33.693	187.5	197.4	5247	4981
	-29.707	-33.693	187.5	195.8	5356	5000
Rank sum	73.5	136.5	114.0	96.0	105.0	105.0
U	18.5		41.0		50.0	
P	0.0186		0.5288		1.0	

Table K.1 Mann-Whitney pair-wise comparison of BFGS results for two step preconditioned and unconditioned test cases

	Laser 1		**Laser 2**		**Aerofoil**	
	Original	**Precon**	**Original**	**Precon**	**Original**	**Precon**
	-0.476	0.168	0.496	0.107	-44.73	-61.57
	-0.479	0.167	0.314	0.067	-66.25	-64.94
	-0.480	-0.009	0.286	0.066	-67.25	-65.18
	-0.481	-0.009	0.274	-0.024	-67.32	-66.12
	-0.481	-0.481	0.274	-0.560	-67.32	-66.21
	-0.481	-0.481	-0.561	-0.561	-68.37	-68.61
	-0.481	-0.481	-0.562	-0.562	-68.45	-68.62
	-0.481	-0.482	-0.562	-0.562	-68.62	-68.62
	-0.482	-0.482	-0.562	-0.562	-68.64	-68.63
	-0.482	-0.482	-0.562	-0.563	-68.64	-68.63
Rank sum	108.0	102.0	92.0	118.0	112.0	98.0
U	47.0		37.0		43.0	
P	0.8534		0.3526		0.6306	

	Bead		**Crack 1**		**Crack 2**	
	Original	**Precon**	**Original**	**Precon**	**Original**	**Precon**
	37.715	20.019	199.2	199.7	4411	4663
	28.292	8.671	199.1	199.4	5264	4673
	15.083	8.164	199.0	199.2	5265	4695
	13.840	-2.666	199.0	199.2	5293	4993
	5.086	-9.991	195.8	199.1	5297	5000
	3.711	-10.520	188.0	199.1	5300	5164
	3.608	-10.520	187.8	199.0	5311	5181
	-10.798	-11.527	187.7	198.0	5319	5181
	-13.005	-29.707	187.6	195.0	5319	5239
	-26.977	-33.693	187.6	195.0	5353	5239
Rank sum	89.0	121.0	133.0	77.0	145.0	65.0
U	34.0		22.0		10.0	
P	0.2474		0.0354		0.0016	

Table K.2 Mann-Whitney pair-wise comparison of Nelder-Mead Simplex results for two step preconditioned and unconditioned test cases

	Laser 1		**Laser 2**		**Aerofoil**	
	Original	**Precon**	**Original**	**Precon**	**Original**	**Precon**
	0.242	0.219	0.333	0.156	-44.77	-63.32
	0.045	0.191	0.310	-0.041	-54.61	-64.97
	-0.416	0.046	0.278	-0.420	-56.56	-65.11
	-0.432	0.013	0.274	-0.456	-64.97	-65.23
	-0.455	0.011	0.262	-0.520	-65.61	-66.73
	-0.463	-0.355	-0.187	-0.521	-65.80	-67.15
	-0.479	-0.445	-0.559	-0.544	-68.36	-68.48
	-0.482	-0.478	-0.559	-0.559	-68.54	-68.57
	-0.482	-0.482	-0.562	-0.559	-68.62	-68.57
	-0.482	-0.482	-0.563	-0.563	-68.64	-68.62
Rank sum	119.5	90.5	91.0	119.0	96.0	114.0
U	35.5		36.0		41.0	
P	0.3150		0.3150		0.5288	

	Bead		**Crack 1**		**Crack 2**	
	Original	**Precon**	**Original**	**Precon**	**Original**	**Precon**
	24.504	20.019	219.5	205.1	4405	4950
	10.354	2.679	208.2	202.8	5239	5000
	-11.795	1.132	199.5	199.8	5263	5000
	-12.681	-1.524	199.2	199.5	5264	5000
	-12.841	-9.991	197.6	199.2	5291	5000
	-14.806	-11.527	195.6	199.2	5295	5008
	-16.547	-11.843	191.2	199.0	5298	5025
	-18.046	-29.707	188.5	198.8	5322	5104
	-22.615	-29.707	187.7	198.1	5330	5155
	-39.849	-33.693	187.6	196.6	5333	5236
Rank sum	114.0	96.0	123.0	87.0	145.0	65.0
U	41.0		32.0		10.0	
P	0.5288		0.1904		0.0016	

Table K.3 Mann-Whitney pair-wise comparison of Nelder-Mead Simplex with line search results for two step preconditioned and unconditioned test cases

	Laser 1		**Laser 2**		**Aerofoil**	
	Original	**Precon**	**Original**	**Precon**	**Original**	**Precon**
	0.129	0.219	0.334	0.156	-44.76	-63.29
	0.008	0.191	0.253	0.101	-54.04	-64.30
	-0.051	0.160	0.120	0.074	-57.18	-64.95
	-0.255	-0.003	0.008	0.039	-61.00	-65.20
	-0.445	-0.109	-0.018	-0.239	-64.96	-65.23
	-0.457	-0.184	-0.222	-0.456	-67.32	-67.15
	-0.465	-0.467	-0.237	-0.456	-68.56	-68.47
	-0.471	-0.469	-0.486	-0.558	-68.62	-68.57
	-0.473	-0.479	-0.559	-0.559	-68.63	-68.57
	-0.478	-0.481	-0.562	-0.563	-68.63	-68.62
Rank sum	113.0	97.0	97.0	113.0	100.0	110.0
U	42.0		42.0		45.0	
P	0.5788		0.5788		0.7394	

	Bead		**Crack 1**		**Crack 2**	
	Original	**Precon**	**Original**	**Precon**	**Original**	**Precon**
	24.504	20.019	219.6	205.1	5236	4677
	10.354	8.164	208.2	202.8	5253	4993
	-11.795	-1.524	199.5	199.9	5265	5000
	-12.841	-6.698	199.2	199.8	5269	5000
	-14.806	-8.996	197.9	199.1	5283	5009
	-16.547	-9.991	196.0	199.1	5294	5012
	-18.046	-29.707	195.9	199.0	5299	5013
	-22.615	-29.707	192.9	198.5	5319	5165
	-24.240	-29.707	191.1	197.5	5326	5172
	-39.849	-33.693	187.5	196.9	5332	5238
Rank sum	108.0	102.0	121.0	89.0	154.0	56.0
U	47.0		34.0		1.0	
P	0.8534		0.2474		0.0000	

Table K.4 Mann-Whitney pair-wise comparison of Nelder-Mead Simplex with single-pass line search results for two step preconditioned and unconditioned test cases

	Laser 1		**Laser 2**		**Aerofoil**	
	Original	**Precon**	**Original**	**Precon**	**Original**	**Precon**
	-0.476	-0.481	0.497	0.096	-44.76	-65.12
	-0.479	-0.481	0.309	0.082	-66.20	-66.17
	-0.480	-0.481	0.305	0.066	-66.22	-66.18
	-0.481	-0.481	0.289	-0.173	-66.26	-66.19
	-0.481	-0.481	0.279	-0.173	-67.52	-66.25
	-0.481	-0.482	0.278	-0.557	-68.33	-67.71
	-0.481	-0.482	-0.560	-0.558	-68.59	-68.61
	-0.482	-0.482	-0.561	-0.562	-68.63	-68.62
	-0.482	-0.482	-0.562	-0.562	-68.64	-68.63
	-0.482	-0.482	-0.562	-0.562	-68.64	-68.64
Rank sum	83.0	127.0	84.0	126.0	111.5	98.5
U	28.0		29.0		43.5	
P	0.1052		0.1230		0.6842	

	Bead		**Crack 1**		**Crack 2**	
	Original	**Precon**	**Original**	**Precon**	**Original**	**Precon**
	15.083	22.919	211.4	202.8	4878	4673
	13.840	14.415	199.2	00.2	5253	4695
	13.451	13.473	199.2	199.6	5294	4926
	3.909	8.164	199.1	199.2	5294	5000
	3.711	-5.137	188.0	199.2	5303	5005
	-0.455	-11.527	188.0	199.0	5319	5162
	-2.464	-11.588	187.8	199.0	5319	5181
	-7.230	-19.400	187.6	198.2	5319	5181
	-11.527	-21.607	187.6	198.0	5319	5238
	-16.115	-29.707	187.6	195.1	5319	5240
Rank sum	94.5	115.5	128.0	82.0	147.0	63.0
U	39.5		27.0		8.0	
P	0.4812		0.0892		0.0008	

Table K.5 Mann-Whitney pair-wise comparison of MDS results for two step preconditioned and unconditioned test cases

	Laser 1		**Laser 2**		**Aerofoil**	
	Original	**Precon**	**Original**	**Precon**	**Original**	**Precon**
	0.412	-0.009	0.355	0.095	-44.66	-61.17
	0.390	-0.009	0.314	0.067	-58.10	-65.22
	-0.476	-0.169	0.295	-0.171	-61.73	-67.34
	-0.481	-0.459	0.094	-0.552	-67.32	-68.60
	-0.481	-0.470	-0.071	-0.560	-68.51	-68.60
	-0.482	-0.482	-0.501	-0.562	-68.61	-68.62
	-0.482	-0.482	-0.555	-0.562	-68.62	-68.62
	-0.482	-0.482	-0.562	-0.563	-68.62	-68.63
	-0.482	-0.482	-0.563	-0.563	-68.64	-68.64
	-0.482	-0.482	-0.563	-0.563	-68.64	-68.64
Rank sum	107.5	102.5	88.0	122.0	95.0	115.0
U	47.5		33.0		40.0	
P	0.9118		0.2176		0.4812	
	Bead		**Crack 1**		**Crack 2**	
	Original	**Precon**	**Original**	**Precon**	**Original**	**Precon**
	37.715	20.019	212.0	201.3	4881	4695
	15.083	-1.834	199.5	199.4	5293	4695
	12.652	-19.400	199.1	199.2	5295	5006
	9.916	-19.400	199.0	199.2	5299	5008
	3.608	-19.610	195.2	199.1	5310	5172
	1.214	-21.607	191.8	199.0	5311	5172
	-11.795	-28.281	188.1	197.4	5319	5172
	-13.818	-29.707	187.8	195.0	5333	5236
	-21.801	-29.707	187.6	195.0	5348	5236
	-26.914	-33.693	187.6	195.0	5348	5239
Rank sum	76.0	134.0	124.0	86.0	147.0	63.0
U	21.0		31.0		8.0	
P	0.0288		0.1654		0.0008	

Table K.6 Mann-Whitney pair-wise comparison of RSCS results for two step preconditioned and unconditioned test cases

	Laser 1		**Laser 2**		**Aerofoil**	
	Original	**Precon**	**Original**	**Precon**	**Original**	**Precon**
	0.411	0.053	0.280	-0.285	-44.64	-65.77
	0.309	0.034	0.273	-0.291	-53.36	-66.07
	0.145	-0.011	0.242	-0.339	-54.08	-66.44
	0.078	-0.041	-0.036	-0.356	-55.58	-66.83
	-0.138	-0.445	-0.533	-0.459	-57.43	-67.48
	-0.242	-0.468	-0.551	-0.471	-58.55	-67.55
	-0.331	-0.482	-0.559	-0.553	-65.62	-67.99
	-0.456	-0.482	-0.559	-0.563	-65.99	-68.01
	-0.479	-0.482	-0.559	-0.563	-66.23	-68.18
	-0.482	-0.482	-0.563	-0.563	-67.21	-68.47
Rank sum	84.0	126.0	96.0	114.0	62.0	148.0
U	29.0		41.0		7.0	
P	0.1230		0.5288		0.0004	

	Bead		**Crack 1**		**Crack 2**	
	Original	**Precon**	**Original**	**Precon**	**Original**	**Precon**
	13.840	20.019	204.2	202.8	4405	4695
	1.723	8.164	202.0	201.1	5007	4978
	-6.282	8.164	200.9	200.5	5292	5000
	-11.795	-5.426	200.8	199.4	5292	5000
	-12.841	-5.426	200.4	199.2	5294	5002
	-13.818	-9.374	195.0	199.2	5297	5161
	-13.818	-13.049	194.6	198.9	5300	5162
	-14.806	-14.699	192.6	197.4	5304	5190
	-19.610	-29.707	188.3	195.9	5348	5236
	-26.977	-33.693	187.5	195.0	5357	5240
Rank sum	114.0	96.0	113.0	97.0	140.0	70.0
U	41.0		42.0		15.0	
P	0.5288		0.5788		0.0068	

Table K.7 Mann-Whitney pair-wise comparison of RSCS with line search results for two step preconditioned and unconditioned test cases

	Laser 1		**Laser 2**		**Aerofoil**	
	Original	**Precon**	**Original**	**Precon**	**Original**	**Precon**
	0.412	0.053	0.286	0.075	-44.60	-66.14
	0.119	0.025	0.283	0.066	-54.01	-66.46
	0.009	-0.008	0.245	-0.420	-55.78	-67.46
	-0.285	-0.009	-0.276	-0.533	-57.28	-67.55
	-0.459	-0.469	-0.365	-0.537	-58.07	-67.83
	-0.475	-0.470	-0.555	-0.537	-62.52	-67.98
	-0.478	-0.472	-0.556	-0.555	-65.05	-68.01
	-0.479	-0.475	-0.560	-0.557	-67.32	-68.18
	-0.479	-0.481	-0.562	-0.561	-67.32	-68.47
	-0.481	-0.481	-0.563	-0.562	-68.62	-68.64
Rank sum	105.0	105.0	101.0	109.0	68.0	142.0
U	50.0		46.0		13.0	
P	1.0000		0.7960		0.0038	

	Bead		**Crack 1**		**Crack 2**	
	Original	**Precon**	**Original**	**Precon**	**Original**	**Precon**
	15.083	20.019	201.8	202.8	4399	4689
	13.840	8.164	201.4	199.8	5271	4980
	9.868	8.164	200.7	199.4	5272	5000
	3.909	-5.037	199.8	199.2	5280	5004
	1.723	-5.426	195.0	199.2	5291	5161
	-6.282	-6.698	190.2	198.3	5292	5164
	-11.795	-13.049	190.1	196.9	5296	5167
	-12.841	-14.699	187.9	195.9	5319	5181
	-19.610	-29.707	187.7	195.1	5319	5236
	-26.977	-33.693	187.7	195.1	5348	5239
Rank sum	97.0	113.0	119.0	91.0	145.0	65.0
U	42.0		36.0		10.0	
P	0.5788		0.3150		0.0016	

Table K.8 Mann-Whitney pair-wise comparison of RSCS with single-pass line search results for two step preconditioned and unconditioned test cases

	Laser 1		Laser 2		Aerofoil	
	Original	**Precon**	**Original**	**Precon**	**Original**	**Precon**
	-0.477	-0.480	-0.558	-0.560	-68.632	-68.624
	-0.480	-0.480	-0.560	-0.560	-68.633	-68.638
	-0.481	-0.481	-0.561	-0.560	-68.637	-68.639
	-0.481	-0.481	-0.561	-0.561	-68.638	-68.640
	-0.481	-0.481	-0.562	-0.561	-68.638	-68.640
	-0.481	-0.481	-0.562	-0.562	-68.638	-68.642
	-0.481	-0.481	-0.562	-0.562	-68.639	-68.642
	-0.481	-0.481	-0.562	-0.562	-68.640	-68.642
	-0.482	-0.481	-0.562	-0.562	-68.641	-68.643
	-0.482	-0.481	-0.562	-0.562	-68.641	-68.643
Rank sum	105.0	105.0	100.0	110.0	79.5	131.5
U	50.0		45.0		23.5	
P	1.0000		0.7394		0.0524	
	Bead		**Crack 1**		**Crack 2**	
	Original	**Precon**	**Original**	**Precon**	**Original**	**Precon**
	-23.335	-29.707	187.9	195.7	5316	5240
	-26.914	-33.693	187.8	195.2	5331	5240
	-26.977	-33.693	187.8	195.1	5348	5240
	-26.977	-33.693	187.8	195.1	5348	5240
	-29.707	-33.693	187.7	195.1	5352	5240
	-33.693	-33.693	187.7	195.1	5352	5240
	-33.693	-33.693	187.6	195.1	5355	5240
	-33.693	-33.693	187.6	195.0	5355	5240
	-39.849	-33.693	187.6	195.0	5356	5240
	-39.849	-33.693	187.6	195.0	5356	5240
Rank sum	92.0	118.0	155.0	55.0	155.0	55.0
U	37.0		0.0		0.0	
P	0.3472		0.0000		0.0000	

Table K.9 Mann-Whitney pair-wise comparison of EPSOC results for two step preconditioned and unconditioned test cases

REFERENCES

REFERENCES

Abramson, D., Cope, M., and McKenzie, R., (1994) "Modelling Photochemical Pollution using Parallel and Distributed Computing Platforms", *Proc. PARLE-94*, Athens, Greece, pp. 478-489.

Abramson, D., Sosic, R., Giddy, J., and Hall, B., (1995) "Nimrod: A Tool for Performing Parametised Simulations using Distributed Workstations", *4th IEEE Symnposium on High Performance Distributed Computing*, Virginia.

Abramson, D., Foster, I., Giddy, J., Lewis, A., Sosic, R., Sutherst, R., and White, N., (1997) "The Nimrod Computational Workbench: A Case Study in Desktop Metacomputing", *Australian Computer Science Conference (ACSC97)*, Sydney, Australia.

Abramson, D., Lewis, A., and Peachey, T., (2000) "Nimrod/O: A Tool for Automatic Design Optimization", *The 4th International Conference on Algorithms & Architectures for Parallel Processing (ICA3PP 2000)*, Hong Kong, China.

Abramson, D., Lewis, A., and Peachey, T., (2001) "Case Studies in Automatic Design Optimisation using the P-BFGS Algorithm", *Proc. 2001 High Performance Computing Symposium (HPC'01), Advanced Simulation Technologies Conference*, Seattle, WA, pp. 104-109.

Abramson, D., Lewis, A., Peachey, T., Fletcher, C., (2001a) "An Automatic Design Optimization Tool and its Application to Computational Fluid Dynamics", *Proc. ACM/IEEE SC2001 Conf.*, Denver, CO.

Alander, J.T., (1995) "An Indexed Bibliography of Genetic Algorithms in Computer Aided Design", Report 94-1-CAD, Department of Information Technology and Production Economics, University of Vaasa, Finland.

Alander, J.T., (1999) "An Indexed Bibliography of Genetic Algorithms Papers Available via ftp and www", Report 94-1-FTP, Department of Information Technology and Production Economics, University of Vaasa, Finland.

Alexandrov, N.M., and Lewis, R.M., (2000) "First-Order Frameworks for Managing Models in Engineering Optimization", in *Modeling and Space Mapping for Engineering Optimization*, Lyngby, Denmark.

Alotto, P., Molfino, P., and Molinari, G., (2001) "A WWW-based Tool for the Remote Optimisation of Electromagnetic Devices", *IEEE Trans. Magnetics*, **37**(5)1, pp. 3592-5.

Anderson, N., Gallagher, J.W., and Hertel, I.V., (1988) *Physics Reports (Review Section of Phys. Rev. Lett.)*, **165**(1-2), pp. 1-188.

Anderson, E.J., and Ferris, M.C., (2001) "A Direct Search Algorithm for Optimization with Noisy Function Evaluations", *SIAM J. Optim.*, **11**(3), pp. 837-857.

Avriel, M., and Wilde, D.J., (1966) "Optimal search for a maximum with sequences of simultaneous function evaluations", *Management Science*, **12**, pp. 722-731.

Azzi, M., Johnson, G., and Cope, M., (1992) "An Introduction to the Generic Reaction Set photochemical mechanism", *Proc. 11th Int. Conf. of the Clean Air Society of Australia and New Zealand*, Brisbane, Australia.

Bak P. and Sneppen K. (1993) “Punctuated equilibrium and criticality in a simple model of evolution”, *Phys. Rev. Let.*, **71**, pp. 4083-4086.

Bak, P. (1996) “How Nature Works”, Springer-Verlag, New York.

Bakr, M.H., Bandler, J.W., Madsen, K., Rayas-Sánchez, J.E., and Søndergaard, J., (2000) “Space-Mapping Optimization of Microwave Circuits Exploiting Surrogate Models”, *IEEE Trans. Microwave Theory and Techniques*, **48**(12), pp. 2297-2306.

Bäck, T. (1996) “Evolutionary Algorithms in Theory and Practice”, Oxford University Press, New York, NY.

Barton, R.R., and Ivey, J.S., Jr., (1996) “Nelder-Mead Simplex Modifications for Simplex Optimization”, *Management Science*, **42**(7), pp. 954-973.

Bassiri, K., and Hutchinson, D., (1994) “New Parallel Variants on Parallel Multi-Dimensional Search for Unconstrained Optimisation”, Research Report 94.28, University of Leeds, School of Computer Studies.

Beale, E.M.L., (1988) “Introduction to Optimization”, John Wiley & Sons, Chichester, 1988.

Bischof, C.H., Bücker, H.M., Lang, B., and Rasch, A., (2003) “Automated Gradient Calculation”, In J. Ballmann (ed.), *Flow Modulation and Fluid-Structure Interaction at Airplane Wings*, number 84 in Notes on Numerical Fluid Mechanics and Multidisciplinary Design, pp. 205-224. Springer, Berlin.

Bongartz, I., Conn, A.R., Gould, N. and Toint, P.L., (1995) “{CUTE}: Constrained and Unconstrained Testing Environment”, *ACM Trans.Math.Software*, **21**(1), pp. 123-160.

Booker, A.J., Dennis, Jr., J.E., Frank, P.D., Serafini, D.B., Torczon, V., and Trosset, M.W., (1998) “Optimization using surrogate objectives on a helicopter test example”, in *Computational Methods in Optimal Design and Control*, Borggaard, et al. (eds.), Birkhauser, Boston, MA, pp. 49-58.

Box, G.E.P., and Draper, N.R., (1987) “Empirical Model-Building and Response Surfaces”, Wiley.

Brent, R.P., (1973) “Algorithms for Minimization without Derivatives”, Prentice-Hall, Englewood Cliffs, NJ.

Broyden, C.G., (1970) “The convergence of a class of double rank minimization algorithms, parts I and II, *J.Inst.Maths.Applns.*, **6**, pp. 76-90 and 222-231.

Büche, D., Schraudolph, N.N., and Koumoutsakos, P., (2003) “Accelerating Evolutionary Algorithms Using Fitness Function Models”, *Proc. Bird of a Feather Workshops, Genetic and Evolutionary Computation Conference (GECCO 2003)*, Chicago, IL, pp. 166-169.

Byrd, R.H., Schnabel, R.B., and Schulz, G.A., (1988) “Parallel quasi-Newton Methods for Unconstrained Optimization”, *Math. Prog.*, **42**, pp. 273-306.

Byrd, R.H., Lu, P., Nocedal, J., and Zhu, C., (1995) “A limited memory algorithm for bound constrained optimization”, *SIAM J. Opt.*, **16**(5), pp. 1190-1208.

Carter, R., Gablonsky, J.M, Patrick, A., Kelley, C.T., and Eslinger, O.J., (2001) “Algorithms for Noisy Problems in Gas Transmission Pipeline Optimization”, *Optimization and Engineering*, (2), pp. 139-157.

Chaperon, P., Sawyer, J., Jones, R., and Rose, F., (1999) “Structural Optimization with Damage Tolerance Constraints”, *Proceedings of the 20th Symposium of the International Committee on Aeronautical Fatigue*, Bellevue, WA.

Coello Coello, C.A., (1998) “An Updated Survey of Evolutionary Multiobjective Optimization Techniques” State of the Art and Future Trends”, Technical Report Lania-RD-98-08, Laboratorio Nacional de Informática Avanzada (LANIA), Xalapa, Veracruz, México.

Coello Coello, C.A., (1999) “A Comprehensive Survey of Evolutionary-Based Multiobjective Optimization Techniques”, *Knowledge and Information Systems*, **1**(3), pp. 269-308.

Conn, A.R., Scheinberg, K., and Toint, Ph.L., (1997) “Recent Progress in Unconstrained Nonlinear Optimization Without Derivatives”, *ISMP97*, Lausanne, Switzerland.

Cooper, L., and Steinberg, D., (1970) “Introduction to Methods of Optimization”, W.B.Saunders, Philadelphia, PA.

Culberson, J., (1996) “On the Futility of Blind Search”, Technical Report TR 96-18, University of Alberta, Department of Computer Science.

Davidon, W.C., (1959) “Variable metric method for minimization”, *AEC Res. and Dev. Report ANL-5990* (revised).

Dennis, J.E., and Torczon, V., (1991) “Direct search methods on parallel machines”, *SIAM J. Optim.*, **1**, pp. 448-474.

Dennis, J.E., and Torczon, V., (1996) “Managing Approximation Models in Optimization”, *Proceedings of the 6th AIAA/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, Bellevue, WA.

Dolan, E.D., Lewis, R.M., and Torczon, V., (2000) “On the Local Convergence of Pattern Search”, *SIAM Journal on Optimization*, **14**(2), pp. 567-583.

Durand, N., and Alliot, J-M., (1999) “A Combined Nelder-Mead Simplex and Genetic Algorithm”, *GECCO’99: Proceedings of the Genetic and Evolutional Computation Conference*, Orlando, FL.

El-Beltagy, M.A., and Keane, A., (2001) “Evolutionary optimization for computationally expensive problems using Gaussian processes”, *Proc. Int. Conf. on Artificial Intelligence IC-AI’2001*, Las Vegas, pp. 708-714.

Elster, C., and Neumaier, A., (1995) “A grid algorithm for bound constrained optimisation of noisy functions”, *IMA J. Numerical Analysis*, **15**(4), pp. 585-608.

Elster, C., and Neumaier, A., (1997) “A trust region method for the optimisation of noisy functions”, *Computing*, **58**, pp. 31-46.

Fletcher, R. and Reeves, C.M., (1963) “Function minimization by conjugate gradients”, *Computer Journal*, **7**, pp. 149-54.

Fletcher, R. and Powell, M.J.D., (1963) “A rapidly convergent descent method for minimization”, *Computer Journal*, **6**, pp. 163-8.

Fletcher, R., (1970) “A new approach to variable metric algorithms”, *Computer Journal*, **13**, pp. 317-322.

Fletcher, R., (1987) "Practical Methods of Optimization", 2^{nd} Ed., John Wiley & Sons, Chichester.

http://www.fluent.com/

Fogel, L.J., (1962) "Autonomous automata", *Industrial Research*, **4**, pp. 14-19.

Fogel, L.J., Owens, A.J., and Walsh, M.J., (1966) "Artificial Intelligence through Simulated Evolution", Wiley, New York.

Fogel, D.B., (1991) "System Identification through Simulated Evolution: A Machine Learning Approach to Modeling", Ginn Press, Needham Heights.

Fogel, D.B., (1992) "Evolving Artificial Intelligence", PhD thesis, University of California, San Diego, CA.

Foster, I., (1994) *Designing and Building Parallel Programs*, Addison-Wesley, Reading, MA.

Foster, I., Kesselman, C., (1997) "Globus: A Metacomputing Infrastructure Toolkit", *International Journal of Supercomputer Applications*, **11**(2), pp. 115-128.

Freeman, T.L., and Phillips, C., (1992) *Parallel Numerical Algorithms*, Prentice Hall International, Hemel Hempstead, UK.

Gill, P.E., Murray, W., and Wright, M.H., (1981) "Practical Optimization", Academic Press, London and New York.

Giunta, A. A, Narducci R., Burgee, S., Grossman, B., Mason, W. H., Watson, L. T. and Haftka, R. T., (1995) "Variable-Complexity Response Surface Aerodynamic Design of an HSCT Wing", AIAA 95–1886, *Proceedings of 13th AIAA Applied Aerodynamics Conference*, San Diego, CA, pp. 994-1002.

Goldfarb, D., (1970) "A family of variable metric methods derived by variational means", *Maths.Comp.*, **24**, pp. 23-26.

Grefenstette, J.J., (1984) "GENESIS: A system for using genetic search procedures", *Proceedings of the 1984 Conference on Intelligent Systems and Machines*, pp. 161-165.

Haftka, R.T., Vitali, R., and Sankar, B.V., (1999) "Optimization of Composite Structures using Response Surface Approximations," NATO Advanced Study Institute, July 1998, In *Mechanics of Composite Materials and Structures*, A. Mota Soares, et al. (eds.), Kluwer Academic Publishers, pp. 409-430.

Hall, B.V., (1989) "Superelastic Electron Scattering from the Laser Excited $5^2P_{3/2}$ State of Rubidium", PhD thesis, Griffith University, Australia.

Hamma, B.S., (1997) "Local and Global Behavior of Moving Polytope Algorithms", *CERFACS Tech. Report TR/PA/97/39*, Toulouse, France.

Hestenes, M.R. and Stiefel, E., (1952) "Methods of conjugate gradients for solving linear equations", *J.Res.Nat.Bur.Stand.*, **49**, pp. 409-36.

Hestenes, M.R., (1956) "The conjugate gradient method for solving linear systems", In *Proc.Symposia on Appl.Math.*, **VI** "Numerical Analysis", McGraw-Hill, New York, pp. 83-102.

Holland, J.H. (1975) "Adaptation in natural and artificial systems", University of Michigan Press, Ann Arbor, MI.

Hooke, R., and Jeeves, T.A., (1961) "'Direct Search' solution of numerical and statistical problems", *J. Assoc. Comput. Mach.*, **8**, pp. 212-229.

Hough, P.D., and Meza, J.C., "A class of trust-region methods for parallel optimization", *SIAM J. Opt.*, **13**(1), pp. 264-282.

Humphrey, D.G., and Wilson, J.R., (1998) "A Revised Simplex Search Procedure for Stochastic Simulation Response-Surface Optimization", *Proc. 1998 Winter Simulation Conference*, D.J. Medeiros, et al. (eds.), IEEE Press, Piscataway, N.J., pp. 751-759.

Ingber, L., (1989) "Very fast simulated re-annealing", *Journal of Mathematical Computer Modeling*, **12**(8), pp. 967-973.

Keane, A.J., (1996) "A Brief Comparison of Some Evolutionary Optimization Methods", In *Modern Heuristic Search Methods*, V. Rayward-Smith, et al. (eds.), J. Wiley, pp. 255-272.

Kelley, C.T., (1999) "Detection and Remediation of Stagnation in the Nelder-Mead Algorithm using a Sufficient Decrease Condition", *SIAM J. Optim.*, **10**(1), pp. 48-55.

Kelley, C.T., (1999) "Iterative Methods of Optimization", SIAM, Philadelphia, PA.

Khuri, A.I., and Cornell, J.A., (1987) "Response surfaces: design and analyses", Dekker, Inc., New York, NY.

Kiefer, J., (1959) "Optimum sequential search and approximation methods under minimum regularity assumptions", *J.SIAM*, **5**(3).

Kirkpatrick, S., Gerlatt, C. D. Jr., and Vecchi, M.P., (1983)" Optimization by Simulated Annealing", *Science*, **220**, pp. 671-680.

Knowles, J.D. and Corne, D.W., (1999) "The Pareto Archived Evolution Strategy: A New Baseline Algorithm for Multiobjective Optimisation", In *1999 Congress on Evolutionary Computation*, Washington, DC, pp. 98-105.

Koh, B.I., Reinbolt, J.A., Fregly, B.J., and George, A.D. (2004) "Parallel decomposition methods for biomechanical optimization", in *Proceedings of the Eighth International Symposium on the 3D Analysis of Human Movement*, Tampa, FL.

Krink, T., and Thomsen, R., (2001) "Self-Organized Criticality and Mass Extinction in Evolutionary Algorithms", *Proceedings of the 2001 Congress on Evolutionary Computation (CEC2001)*, Seoul, Korea.

van Laarhoven, P.J.M., (1985) "Parallel variable metric algorithms for unconstrained optimization", *Math. Prog.*, **39**, pp. 68-81.

Lagarias, J.C., Reeds, J.A., Wright, M.H., and Wright, P.E., (1996) "Convergence Properties of the Nelder-Mead Simplex Algorithm in Low Dimensions", Technical Report 96-4-07, Bell Laboratories, Computing Science Research Center.

Laumanns, M., Thiele, L., Deb, K. and Zitzler, E., (2002) "Archiving with guaranteed convergence and diversity in multi-objective optimization", in *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 439-447.

Lewis, A., Abramson, D., Sosic, R., and Giddy, J., (1995) "Tool-based Parameterisation: An Application Perspective", *Proc. Computational Techniques and Applications Conference (CTAC95)*, pp. 463-469, Melbourne, Australia.

Lewis, A., Abramson, D., and Simpson, R., (1997) “Parallel non-linear optimization: Towards the design of a decision support system for air quality management”, *Proceedings of the 1997 ACM/IEEE conference on Supercomputing*, San Jose, CA.

Lewis, R.M., Torczon, V., and Trosset, M.W., (2000) “Direct Search Methods: Then and Now”, *J. Computational and Applied Mathematics* **124**(1-2), pp. 191-207.

Liang, K-H., Yao, X., and Newton, C., (2000) “Evolutionary Search of Approximated N-Dimensional Landscapes”, *Int. J. Knowledge-Based Intelligent Eng. Sys.*, **4**(3), pp. 172-183.

Lu, J.W., Thiel, D.V., and Saario, S.A., (2002) “FDTD analysis of dielectric-embedded electronically switched multiple-beam (DE-ESMB) antenna array”, *IEEE Trans. Magnetics*, **38**(2), pp. 701-704.

Mattheck, C., and Burkhardt, S., (1990) “A new method of structural shape optimization based on biological growth”, *Int. J. Fatigue*, **12**(3), pp. 185-190.

McKinnon, K.I.M., (1998) “Convergence of the Nelder-Mead Simplex Method to a Non-Stationary Point”, *SIAM J. Optim.*, **9**, pp. 148-158.

McRae, G.J., Russell, A.G., and Harley, R.A., (1992) “CIT photochemical airshed model – users manuals”, Carnegie Mellon University, Pittsburgh, PA and California Intitute of Technology, Pasadena, CA.

Metropolis, N., Rosenbluth, A.W., Rosenbluth, M. N., Teller, A.H. and Teller, E., (1958) “Equations of State Calculations by Fast Computing Machines”, *J. Chem. Phys.* **21**, pp. 1087-1092.

Moré, J.J., and Thuente, D.J., (1994) “Line search algorithms with guaranteed sufficient decrease”, *ACM Transactions on Mathematical Software*, **20**(3), pp. 286-307.

Navon, I.M., Phua, P.K.H., and Ramamurthy, M., (1988) “Vectorization of conjugate-gradient methods for large scale minimization”, *Proceedings of the 1988 ACM/IEEE conference on Supercomputing*, pp. 410-418.

Nazareth, L, and Tseng, P., (2002) “Gilding the Lily: a Variant of the Nelder-Mead Algorithm based on Golden-Section Search”, *Comp.Optim.Appl.*, **22**, pp. 133-144.

Neddermeijer, H.G., van Oortmarssen, G.J., Piersma, N., Dekker, R., and Habbema, J.D.F., (2000) “Adaptive extensions of the Nelder and Mead Simplex Method for optimisation of stochastic simulation models”, *Econometric Institute Report EI2000-22/A*, Faculty of Economics, Erasmus University Rotterdam, The Netherlands.

Nelder, J.A., and Mead, R., (1965) “A simplex method for function minimization”, *Comput. J.*, **7**, pp. 308-313.

Nishioka, T. and Atluri, S.N., (1983) “Analytical Solution for Embedded Elliptical Cracks, and Finite Element Alternating Method for Elliptical Surface Cracks, Subject to Arbitrary Loadings”, *Engng. Fracture Mechanics*, **17**(3), pp. 247-268.

Parkinson, J.M., and Hutchinson, D., (1972) “An Investigation into the Efficiency of Variants on the Simplex Method”, In *Numerical Methods for Non-linear Optimization* (F.A. Lootsma, ed.) Academic Press, London and New York, pp. 115-135.

Peachey, T., Abramson, D., and Lewis, A., (2001) “Parallel Line Search”, To appear in *Proc. ASOR2001 Optimization Day*, Adelaide, Australia.

Peachey, T., Abramson, D., Lewis, A. and Jones, R., (2003) "Distributed Optimization using Nimrod/O and its Application to Fault Tolerant Structures", *Fifth International Conference on Parallel Processing and Applied Mathematics (PPAM 2003)*, Czestochowa, Poland.

Peachey, T., (2003a) "Nimrod/O User's Guide: for Version 2.1.x", Monash University, Australia.

Pérez, V.M., and Renaud, J.E., (2000) "Decoupling the design sampling region from the trust region in approximate optimization", *International Mechanical Engineering Congress & Exposition (IMECE'00)*, Orlando, FL.

Phua, P.K-H., Fan, W., and Zeng, Y., (1998) "Parallel Algorithms for Large-scale Nonlinear Optimization", *Int. Trans. in Operational Res.*, **5**(1), pp. 67-77.

Pincus, M., (1970) "A Monte Carlo Method for the Approximate Solution of Certain Types of Constrained Optimization Problems", *Oper. Res.*, **18**, pp. 1225-1228.

Polak, E., (1971) Computational Methods in Optimization, Academic Press, New York.

Press, W.H., Teukolsky, S.A., Vetterling, W.T., and Flannery, B.P., (1992) "Numerical Recipes in Fortran 77: the art of scientific computing", 2^{nd} Ed., Cambridge University Press.

Rodríguez, J.F., Pérez, V.M., Padmanabhan, D., Renaud, J.E., (2001) "Sequential Approximate Optimization Using Variable Fidelity Response Surface Approximations", *Structural Optimization*, **22**,(1), pp. 24-34.

Saario, S., Thiel, D.V., Lu, J.W., and O'Keefe, S.G., (1997) "An assessment of cable radiation effects on mobile communications antenna measurements", *IEEE APS Symp. Dig.*, pp. 550-553.

Saario, S., Thiel, D.V., and Lu, J.W., (1999) "Application and optimisation of high permittivity ceramic beads for RF isolation on straight wire transmission lines", *CSIRO, Symp. on Antennas* Sydney, Australia.

Sartenaer, A., (1995) "Large-scale nonlinear optimization and the LANCELOT package", *Belgian Journal of Operations Research, Statistics and Computer Science*, **35**(3-4), pp. 61-79.

Saupe, D., (1988) "Algorithms for random fractals", In *The Science of fractal images*, (H.-O. Peitgen and D. Saupe, eds.), Springer-Verlag, New York.

Schwefel, H.-P. (1965) "Kybernetische Evolution als Strategie der experimentellen Forschung in der Strömungstechnik", Diplomarbeit, Technische Universität Berlin.

Scotti, M.V., Malik, Z., Cheung, P.Y.K., and Nelder, J., (2000) "Optimisation of full custom logic cells using response surface methodology", *Electronics Letters*, **36**(1).

Shanno, D.F., (1970) "Conditioning of quasi-Newton methods for function minimization", *Maths.Comp.*, **24**, pp. 647-656.

Sharpe, O.J., (1998) "Beyond NFL : A few tentative steps", *Proceedings of the Third Annual Genetic Programming Conference*, (Koza, J.R., ed.).

Sharpe, O.J., (2000) "Towards a Rational Methodology for Using Evolutionary Search Algorithms", PhD thesis, University of Sussex.

Sharpe, O.J., (2003) private communication.

Shekarforoush, H., Berthod, M., and Zerubia, J., (1995) “Direct Search Generalized Simplex Algorithm for Optimizing Non-Linear Functions”, INRIA Research Report RR-2535, Sophia Antipolis, France.

Schnabel, R.B., (1987) “Concurrent function evaluations in local and global optimization”, *Computer Methods in Applied Mechanics and Engineering*, **64**, pp. 537-552.

Spendley, W., Hext, G.R., and Himsworth, F.R., (1962) “Sequential application of simplex designs in optimization and evolutionary operation”, *Technometrics*, **4**, pp. 441-461.

Torczon, V., (1989) “Multidirectional Search”, Ph.D. thesis, Rice University, Houston, TX.

Torczon, V., (1991) “On the convergence of the multidirectional search algorithm”, *SIAM J. Optim.*, **1**, pp. 123-145.

Torczon, V., and Trosset, M.W., (1998) “Using approximations to accelerate engineering design optimization”, *Proceedings of the 7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, St. Louis, MO.

Tseng, P., (1999) “Fortified-Descent Simplicial Search Method: A General Approach”, *SIAM J. Optim.*, **10**(1), pp. 269-288.

Turing, A.M., (1936) “On computable numbers with an application to the entscheidungs-problem”, *Proc. London Mathematical Society*, **2**(42), pp. 230-265, *ibid.*, (43), pp. 544-546.

Van Veldhuizen, D.A., (1999) “Multiobjective Evolutionary Algorithms: Classifications, Analyses and New Innovations”, PhD Thesis, Air Force Institute of Technology, Air University, USA.

Van Veldhuizen, D.A., and Lamont, G.B., (2000) “Multiobjective Evolutionary Algorithms: Analyzing the State-of-the-Art”, *Evolutionary Computation*, **8**(2) pp. 125-147.

Vitali, R., Haftka, R.T., and Sankar, B.V., (1999) “Multifidelity Design of Stiffened Composite Panel with a Crack”, *World Congress of Structural and Multidisciplinary Optimization Congress*, Buffalo, New York.

Wilde, D.J., (1964) “Optimum Seeking Methods”, Prentice-Hall, Englewood Cliffs, NJ.

Wolpert, D., and Macready, W., (1997) “No free lunch theorems for search”, *IEEE Transactions on Evolutionary Computation*, **1**(1), pp. 67-82.

Wright, A.H., (1991) “Genetic Algorithms for Real Parameter Optimization”, in *Foundations of Genetic Algorithms*, (G.J.E. Rawlins, ed.), Morgan Kaufman.

Wright, M.H., (1995)“Direct Search Methods: Once Scorned, Now Respectable”, in *Numerical Analysis 1995 (Proceedings of the 1995 Dundee Biennial Conference in Numerical Analysis)*, (D.F. Griffiths, and G.A. Watson, eds.), Addison Wesley Longman, Harlow, UK, pp. 191-208.

Zavriev, S.K., and Perunova, Y.N., (2003) “Parallel Versions of the Modified Coordinate and Gradient Descent Methods and Their Application to a Class of Global Optimization Problems”, *Computational Mathematics and Modeling*, **14**(2), pp. 108-122.

Zitzler, E., (1999) “Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications”, D.Tech.Sci. Dissertation, Swiss Federal Institute of Technology, Zurich, Switzerland.

Zitzler, E., Deb, K., and Thiele, L., (2000) “Comparison of Multiobjective Evolutionary Algorithms: Empirical Results”, *Evolutionary Computation*, **8**(2) pp. 173-195.

Zitzler, E., (2002) “Evolutionary Algorithms for Multiobjective Optimization”, *EUROGEN-2001, Evolutionary Methods for Design, Optimisation and Control*, Barcelona, Spain.