



Studying the Rate of Convergence of Gradient  
Optimisation Algorithms via the Theory of  
Optimal Experimental Design

Rebecca Jane Haycroft,  
Cardiff University

May, 2008

A thesis submitted for the degree of Doctor of Philosophy

UMI Number: U585088

All rights reserved

INFORMATION TO ALL USERS

The quality of this reproduction is dependent upon the quality of the copy submitted.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if material had to be removed, a note will indicate the deletion.



UMI U585088

Published by ProQuest LLC 2013. Copyright in the Dissertation held by the Author.  
Microform Edition © ProQuest LLC.

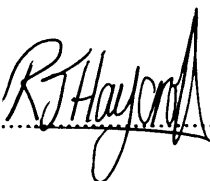
All rights reserved. This work is protected against  
unauthorized copying under Title 17, United States Code.



ProQuest LLC  
789 East Eisenhower Parkway  
P.O. Box 1346  
Ann Arbor, MI 48106-1346


## DECLARATION

This work has not previously been accepted in substance for any degree and is not being concurrently submitted in candidature for any degree.

Signed .....  ..... Date ..... 07/05/2008 .....

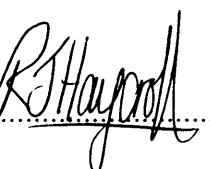
## STATEMENT 1

This thesis is the result of my own investigations, except where otherwise stated. Other sources are acknowledged explicitly. A bibliography is appended.

Signed .....  ..... Date ..... 07/05/2008 .....

## STATEMENT 2

I hereby give my consent for my thesis, if accepted, to be available for photocopying and for inter-library loan, and for the title and summary to be made available to outside organisations.

Signed .....  ..... Date ..... 07/05/2008 .....

## ACKNOWLEDGEMENTS

I would like to thank my supervisor, Professor Anatoly Zhigljavsky, for his continuous support throughout my period of study and for providing me with invaluable opportunities to broaden my skills along the way. Without him none of this would have been possible.

I would also like to extend my gratitude to my family and friends; to my parents, David and Janet, for their unconditional love and encouragement, to Jonny for being my emotional support and to all of my friends of the last few years who have put up with me and my talk of maths. I would also like to express thanks to my colleagues in the School of Mathematics for making my time as a postgraduate student a very enjoyable experience.

Finally I would like to thank the EPSRC for providing me with the financial means to partake in this research.

## PUBLICATIONS

Haycroft, R. J. - *Studying the Rate of Convergence of the Steepest Descent Optimisation Algorithm with Relaxation*, in: *Computer Aided Methods in Optimal Design and Operations*, (Ser. Series on Computers and Operations Research - Vol 7). World Scientific Publishing Co. Inc. (2006) p.49-58, ISBN 981-256-909-X

Haycroft R, Pronzato L, Wynn H. and Zhigljavsky A - *Optimal Experimental Design and Quadratic Optimization*, Tatra Mountains Mathematical Publications. (**To appear**)

Haycroft R, Pronzato L, Wynn H. and Zhigljavsky A - *Studying Convergence of Gradient Algorithms via Optimal Experimental Design Theory*, in: “W-optimality and related statistical issues”. (**To appear**)

## SUMMARY

The most common class of methods for solving quadratic optimisation problems is the class of gradient algorithms, the most famous of which being the Steepest Descent algorithm. The development of a particular gradient algorithm, the Barzilai-Borwein algorithm, has sparked a lot of research in the area in recent years and many algorithms now exist which have faster rates of convergence than that possessed by the Steepest Descent algorithm. The technology to effectively analyse and compare the asymptotic rates of convergence of gradient algorithms is, however, limited and so it is somewhat unclear from literature as to which algorithms possess the faster rates of convergence.

In this thesis methodology is developed to enable better analysis of the asymptotic rates of convergence of gradient algorithms applied to quadratic optimisation problems. This methodology stems from a link with the theory of optimal experimental design. It is established that gradient algorithms can be related to algorithms for constructing optimal experimental designs for linear regression models. Furthermore, the asymptotic rates of convergence of these gradient algorithms can be expressed through the asymptotic behaviour of multiplicative algorithms for constructing optimal experimental designs.

The described connection to optimal experimental design has also been used to influence the creation of several new gradient algorithms which would not have otherwise been intuitively thought of. The asymptotic rates of convergence of these algorithms are studied extensively and insight is given as to how some gradient algorithms are able to converge faster than others. It is demonstrated that the worst rates are obtained when the corresponding multiplicative procedure for updating the designs converges to the optimal design. Simulations reveal that the asymptotic rates of convergence of some of these new algorithms compare favourably with those of existing gradient-type algorithms such as the Barzilai-Borwein algorithm.

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Convex Optimisation . . . . .	1
1.1.1	Convexity and Concavity . . . . .	3
1.1.2	Quadratic Optimisation . . . . .	4
1.1.3	Iterative Methods . . . . .	4
1.2	Steepest Descent . . . . .	7
1.3	The Conjugate Gradient Method . . . . .	11
1.4	Iterative Methods for Solving Linear Systems . . . . .	13
1.4.1	Stationary Iterative Methods . . . . .	14
1.4.2	Krylov Subspace Methods . . . . .	16
1.5	Recent Successful Gradient Algorithms . . . . .	18
1.5.1	The Barzilai-Borwein Algorithm . . . . .	18
1.5.2	Generalisations of the Barzilai-Borwein Algorithm . . . . .	19
1.5.3	Other New Gradient Methods . . . . .	26
1.6	Motivation of Thesis . . . . .	30
<b>2</b>	<b>Gradient Algorithms and their Relation to Optimal Design Theory</b>	<b>32</b>
2.1	Renormalised Versions of Gradient Algorithms . . . . .	32
2.1.1	Renormalisation of the Steepest Descent Algorithm . . . . .	32
2.1.2	Renormalisation of a General Gradient Algorithm . . . . .	35
2.2	Asymptotic Behaviour of the Steepest Descent Algorithm . . . . .	35
2.3	Relation to Optimal Experimental Design . . . . .	44
2.3.1	Optimal Experimental Design Terminology . . . . .	45

2.3.2	Constructing Gradient Algorithms which Correspond to Given Optimality Criteria . . . . .	48
2.3.3	Rate of Convergence of Gradient Algorithms Corresponding to Optimal Designs . . . . .	50
<b>3</b>	<b>The <math>\gamma</math>-Steepest Descent Algorithm</b>	<b>53</b>
3.1	The Renormalised $\gamma$ -Steepest Descent Algorithm . . . . .	54
3.1.1	Concavity of the Optimality Criterion . . . . .	55
3.1.2	Convergence to an Optimal Design . . . . .	56
3.1.3	Speed of Convergence to the Optimum Design . . . . .	60
3.1.4	Behaviour of the Sequence $\{\Phi(\xi^{(k)})\}$ . . . . .	61
3.2	Asymptotic Rate of Convergence . . . . .	71
3.2.1	Dependence on $\gamma$ . . . . .	72
3.2.2	Dependence on $\rho$ . . . . .	75
3.2.3	Dependence on $d$ . . . . .	79
3.2.4	Behaviour of $r^{(k)}$ . . . . .	83
3.3	$\gamma$ -Steepest Descent Summary . . . . .	92
<b>4</b>	<b>The <math>\beta</math>-Root Algorithm</b>	<b>94</b>
4.1	The Square Root Algorithm . . . . .	94
4.1.1	The Renormalised Square Root Algorithm . . . . .	95
4.1.2	Asymptotic Rate of Convergence . . . . .	96
4.2	The $\gamma$ -Square Root Algorithm . . . . .	97
4.2.1	Concavity of the Optimality Criterion . . . . .	98
4.2.2	Convergence to an Optimum Design . . . . .	99
4.2.3	Speed of Convergence to the Optimum Design . . . . .	105
4.2.4	Asymptotic Rate of Convergence . . . . .	105
4.3	Generalisation to the $\beta$ -Root Algorithm . . . . .	109
4.3.1	The Renormalised $\beta$ -Root Algorithm . . . . .	109
4.3.2	Concavity of the Optimality Criterion . . . . .	110
4.3.3	Behaviour of the Sequence $\{\Phi(\xi^{(k)})\}$ . . . . .	111
4.3.4	Asymptotic Rate of Convergence . . . . .	118



4.4	$(\gamma, \beta)$ -Root Algorithm . . . . .	134
4.4.1	Generalisation of the $\beta$ -Root Algorithm . . . . .	134
4.4.2	Asymptotic Rate of Convergence . . . . .	135
4.5	$\beta$ -Root Algorithm Summary . . . . .	138
<b>5</b>	<b><math>\Phi_p</math>-optimality</b> . . . . .	<b>140</b>
5.1	A-Optimality . . . . .	140
5.1.1	The A-Optimality Criterion and the Corresponding Gradient Algorithm . . . . .	140
5.1.2	Behaviour of the Sequence $\{\Phi(\xi^{(k)})\}$ . . . . .	142
5.1.3	Asymptotic Rate of Convergence . . . . .	147
5.2	A-Optimality with Relaxation . . . . .	155
5.2.1	The Renormalised $(\gamma, A)$ -Optimality Gradient Algorithm . . . . .	155
5.2.2	Asymptotic Rate of Convergence . . . . .	155
5.3	$\Phi_2$ -Optimality . . . . .	161
5.3.1	The $\Phi_2$ -Optimality Criterion and the Corresponding Gradient Algorithm . . . . .	161
5.3.2	Behaviour of the Sequence $\{\Phi(\xi^{(k)})\}$ . . . . .	163
5.3.3	Asymptotic Rate of Convergence . . . . .	163
5.4	$\Phi_3$ -Optimality . . . . .	172
5.4.1	The $\Phi_3$ -Optimality Criterion and the Corresponding Gradient Algorithm . . . . .	172
5.4.2	Behaviour of the Sequence $\{\Phi(\xi^{(k)})\}$ . . . . .	174
5.4.3	Asymptotic Rate of Convergence . . . . .	174
5.5	$\Phi_p$ -Optimality Summary . . . . .	184
<b>6</b>	<b>Conclusions and Further Work</b> . . . . .	<b>185</b>
6.1	Summary . . . . .	185
6.2	Further Work . . . . .	188
6.2.1	The Optimum 2-Gradient Algorithm . . . . .	188
6.2.2	The Relaxed Optimum 2-Gradient Algorithm . . . . .	189
6.2.3	Future Investigations . . . . .	193

6.3	Comparison of Algorithms . . . . .	195
6.4	Conclusions . . . . .	198
<b>A</b>	<b>Example Programmes</b>	<b>200</b>
<b>B</b>	<b>Summary Tables</b>	<b>202</b>
<b>C</b>	<b>Extra Graphs</b>	<b>207</b>

# Chapter 1

## Introduction

### 1.1 Convex Optimisation

Finding the extreme points of a function cannot always be achieved via the classical approach of calculus. In some situations calculating the partial derivatives of a function is an arduous task and it is often necessary to resort to numerical optimisation methods to find approximate solutions to the extreme points. Differing numerical methods exist, however the goal in each method invariably involves determining extrema, be they minima or maxima, of the objective function. It is potentially the case that a given objective function could possess more than one extreme point and so the crux of the problem usually lies in establishing whether a found local extremum is also the overall, or global extremum. Methods which are successful in accomplishing this are termed *global optimisation methods* whereas methods which concentrate on identifying local extrema are named *local optimisation methods*. Local optimisation methods are usually used as part of the inner workings of a global optimisation technique.

First two important definitions are given.

**Definition 1.1.1.**  $x^*$  is a local minimum of a function  $f$ , given on  $X$  such that  $f : X \rightarrow \mathbb{R}^1$ , if for some  $\varepsilon > 0$ ,  $f(x^*) \leq f(x)$  for all  $x$  where  $\|x - x^*\| < \varepsilon$ .

The global minimum is then defined as follows:

**Definition 1.1.2.**  $x^*$  is a global minimum of a function  $f$  if,  $f(x^*) \leq f(x) \forall x \in X$ .

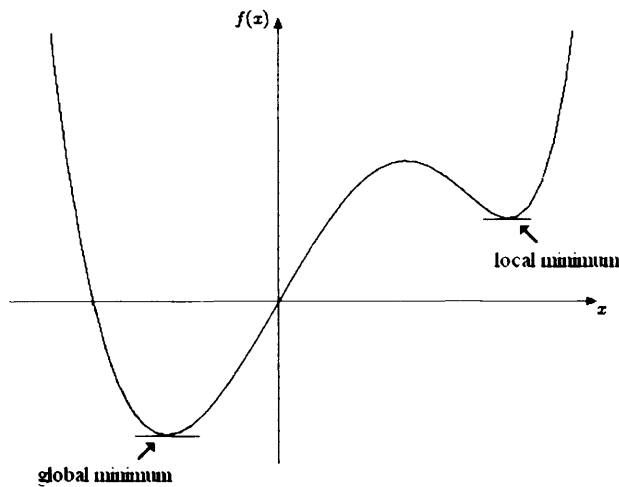


Figure 1.1: Graph showing a typical function  $f(x)$  with more than one minimum.

Figure 1.1 illustrates a local and global minimum of an objective function. Note that the definition of a local and global maximum can be achieved by reversing the inequalities in the above definitions.

Global optimisation is inevitably a much harder problem than local optimisation as, if the function being optimised has more than one turning point, it is possible that the progress of an algorithm might be halted at the discovery of a local minimum and thus the global minimum may not be found. One solution to the problem is to calculate all local minima in turn, (perhaps by selecting different starting points from which to run a local optimisation algorithm) and then, by process of comparison, determine which amongst them is the global minimum. This procedure can be very time-costly and an alternative is desirable. Other approaches exist such as stochastic techniques e.g. simulated annealing where a step in a direction away from a local solution is accepted with some probability. This avoids the certainty of the algorithm terminating prematurely at a local solution. There is, however, a group of functions with a specific property for which this problem is not an issue, namely convex (and concave) functions.

### 1.1.1 Convexity and Concavity

A definition of a convex function is as follows.

**Definition 1.1.3.** A function  $f(x)$  is said to be convex over a convex set  $\mathbb{S}$  if, for any two points  $x_1, x_2 \in \mathbb{S}$  and for all  $\alpha$ ,  $0 \leq \alpha \leq 1$ ,

$$f(\alpha x_2 + (1 - \alpha)x_1) \leq \alpha f(x_2) + (1 - \alpha)f(x_1). \quad (1.1)$$

The definition of a concave function can be obtained simply by reversing the inequality (1.1) i.e. a function is concave if

$$f(\alpha x_2 + (1 - \alpha)x_1) \geq \alpha f(x_2) + (1 - \alpha)f(x_1), \quad \forall \alpha, 0 \leq \alpha \leq 1. \quad (1.2)$$

If the inequalities in (1.1) and (1.2) are replaced with strict inequalities then the function  $f(x)$  is said to be a strictly convex function, or a strictly concave function, respectively. For the proofs of several theorems on the subject of convexity see, for example, [69].

Figure 1.2 shows examples of convex and concave functions of one variable. Here the property translates to the curve lying below (conversely above for concavity) the chord joining any two points  $(x_1, f(x_1))$  and  $(x_2, f(x_2))$ . The advantage of knowing

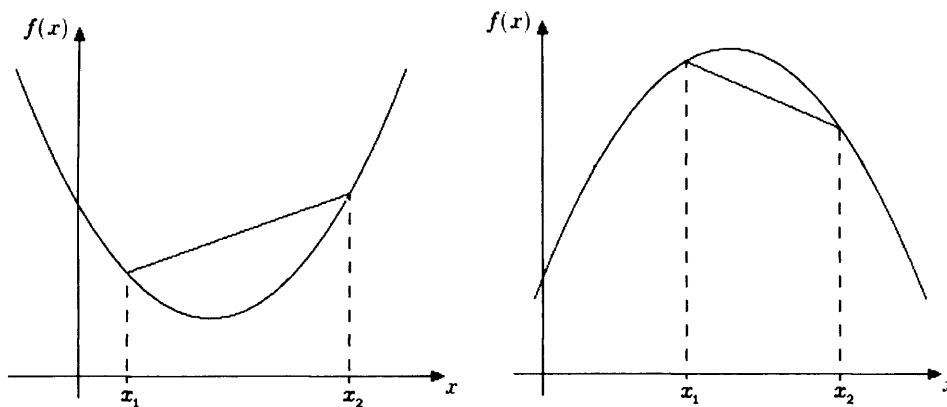


Figure 1.2: Example of (a) a convex function and (b) a concave function.

that the objective function is strictly convex is that, for a function of this kind, there exists at most one minimum, thus any local minimum found is also the global minimum. Similarly with concave functions, any maximum found will be the only

maximum of the function and thus the global maximum. There are many real-life problems where the objective function is known to be convex and for these problems local optimisation techniques can be applied.

### 1.1.2 Quadratic Optimisation

Consider the task of finding the minimum point  $x^*$  of a quadratic objective function of the form

$$f(x) = \frac{1}{2}x^T Ax - x^T b + c \quad (1.3)$$

where  $A$  is a positive definite symmetric  $d \times d$  matrix with eigenvalues

$$0 < m = \lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_d = M < \infty$$

and  $b$  is a  $d \times 1$  vector. Since  $A$  is positive definite it follows that the function  $f$  is strictly convex. Algorithms developed for optimising quadratic functions of this form can thus take advantage of the desirable property of convexity; any local extremum found will also be the global extremum. Quadratic optimisation has been studied extensively. One reason why it has proven so popular is due to the ability to adapt algorithms of this kind to solve systems of linear equations. This application is introduced in Section 1.4. Another reason is that many non-quadratic functions can be approximated well by functions of the form (1.3) in the region of their minimum point; so in many cases it is sufficient to simply minimise the quadratic approximation of the function. In this thesis only the problem of quadratic optimisation is considered.

Most numerical methods developed for solving optimisation problems, quadratic optimisation being no exception, take an iterative form.

### 1.1.3 Iterative Methods

Starting from either an initial guess or a random vector  $x^{(0)}$ , an iterative method will produce a sequence of approximations, in the form of vectors  $x^{(k)}$ ,  $k = 1, 2, \dots$  that are expected to move increasingly closer to the exact solution. The procedure is only said to converge if  $\lim_{k \rightarrow \infty} \|x^* - x^{(k)}\| = 0$ . The iterations are ceased when some predefined stopping criterion is reached.

The majority of iterative algorithms can be written in the following general form

$$x^{(k+1)} = x^{(k)} - \alpha^{(k)} \delta^{(k)}, \quad (1.4)$$

where  $\delta^{(k)}$  is the direction from  $x^{(k)}$  along which the next point is selected. For fixed  $\delta^{(k)}$ ,  $\alpha^{(k)}$  is the step length. Different optimisation algorithms vary in their choice of  $\alpha^{(k)}$  and  $\delta^{(k)}$ .

It must be noted at this point that iterative methods work just as well for the minimisation or maximisation of more general non-linear objective functions; however it is often much harder to calculate the step length for objective functions of higher degree. It is straightforward to obtain results for maximisation of quadratic functions by adapting those results described for minimisation.

### Direct Search Methods

*Direct Search Methods* is the name given to those methods which rely entirely on the value of the objective function  $f(x)$  at iteration  $k$  and information gained from previous iterations. These methods do not require the explicit evaluation of any partial derivatives of  $f(x)$ .

When dealing with functions of one variable a number of simple search methods can be employed. Techniques belonging to this category include the method of Bisection, Fibonacci search and Golden Section search. These methods all involve determining an increasingly smaller interval in which the minimum lies. The process ceases when the interval which contains the minimum point is sufficiently narrow or a pre-specified number of function evaluations are made.

For multi-variate functions, several direct search methods also exist. A generalisation of the Fibonacci search for one variable functions can be applied. In this method a series of nested uni-variate Fibonacci searches is carried out in order to reduce the problem to one which is readily solvable.

Another technique, this one not of the form (1.4), applied to multi-variate functions is the Simplex method. A simplex is a  $d$ -dimensional polytope made up of  $d + 1$  vertices; if  $d = 2$  then it is an equilateral triangle, if  $d = 3$  it takes the form of a regular tetrahedron and so on. The objective function is evaluated at each of the

$d + 1$  vertices. For minimisation, the vertex where the objective function is found to have the highest value is then reflected in the centroid of the remaining  $d$  vertices. The objective function is then evaluated at this new vertex and the process continues by reflecting whichever vertex is now evaluated to be the one with the largest function value. Modifications exist to prevent oscillations and to aid the method in cases where convergence is slow, such as when descent down a narrow valley is required.

An extension of the Simplex method gives rise to the method of Nelder and Mead. Here the basic step of reflection of the largest vertex is accompanied by either an expansion (if this improves upon the vertex obtained by simply reflecting) or contraction (if the point obtained by reflecting is worse than the original vertex).

Other direct search methods developed include Hooke and Jeeves', Rosenbrock's and Davies, Swann and Campey's method. For a description of the main existing direct methods see [10, 11].

## Gradient Methods

In contrast to direct search methods, gradient methods utilise the partial derivatives of the objective function, coupled with information obtained from earlier iterations, to select the direction along which the next point in the iterative process is to be chosen.

For a general smooth function  $f(x)$  in  $\mathbb{R}^d$  gradient algorithms can be written as

$$x^{(k+1)} = x^{(k)} - \alpha^{(k)} \nabla f(x^{(k)}), \quad (1.5)$$

where

$$\nabla f(x^{(k)}) = g(x^{(k)}) = \left( \frac{\partial f}{\partial x_1}, \dots, \frac{\partial f}{\partial x_d} \right)^T$$

and  $\alpha^{(k)}$  is the step length.

It is generally concluded that if information regarding the first derivatives of the objective function is readily obtainable and not too costly to compute then this information should be used. Methods which also make use of the second derivatives usually converge faster but these are not necessarily the most efficient algorithms as the cost associated with computing the matrix of second derivatives and inverting it



can outweigh the advantage gained through this extra knowledge. Also, algorithms of this kind are very sensitive to computational inaccuracies.

Amongst the gradient algorithms are the Newton-Raphson method, the Davidon-Fletcher-Powell method, see [30], the Fletcher-Reeves method, see [31] and the Polak-Ribière method (a modification of the Fletcher-Reeves method), see [52]. The two most important algorithms, to which the next two sections are dedicated are the methods of steepest descent and conjugate gradients.

## 1.2 Steepest Descent

The steepest descent algorithm is the most famous of all the gradient algorithms. Nowadays it is generally regarded as having a poor convergence rate; however it is popular on account of its easy application and stability as an algorithm. Many algorithms have been created by adapting the steepest descent algorithm in some way, with a view to improving upon the rate of convergence. For this reason the method of steepest descent is widely regarded as the gold standard against which all other algorithms of this type are compared.

The method of steepest descent, also known as the gradient descent method, dates back as far as Cauchy, see [12] where it was first suggested as a method for solving systems of linear equations, see Section 1.4.

For a general convex function the steepest descent algorithm takes the following form:

$$x^{(k+1)} = x^{(k)} - \alpha^{(k)} \nabla f(x^{(k)}),$$

where

$$\alpha^{(k)} = \arg \min_{\alpha} f(x^{(k)} - \alpha \nabla f(x^{(k)})). \quad (1.6)$$

The step length  $\alpha^{(k)}$  is chosen so that, at iteration  $(k+1)$ , the function takes on the minimum possible value along the anti-gradient  $-\nabla f(x^{(k)})$ . In other words, the point  $x^{(k+1)}$  is determined by travelling from the previous point  $x^{(k)}$  down the direction of the negative gradient until the minimum point along this line is located. The direction  $-\nabla f(x^{(k)})$  is chosen because the initial rate of decrease of the objective function from  $x^{(k)}$  is greatest in this direction, hence the name ‘steepest descent’.

In the two-dimensional case, the method approaches the minimum point in a zigzag manner. This is due to the fact that, at the minimum point along any search direction, the direction of steepest descent is always at right angles to the previous search direction thus making successive directions orthogonal. Figure 1.3 shows an example of the approach to the minimum point of a function of two variables made by the steepest descent algorithm. The level sets of a function are  $d$ -dimensional ellipsoids (in this figure  $d = 2$  thus the contours are ellipses) and show regions of constant value of the function. For ill-conditioned problems, i.e. if the corresponding

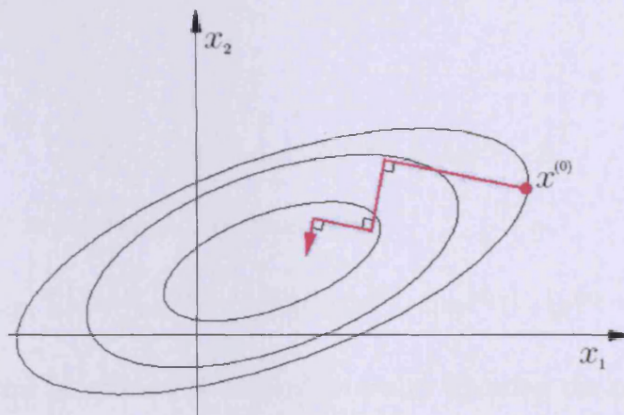


Figure 1.3: Graph showing a typical path of descent made by the steepest descent algorithm for a function of two variables.

maximum and minimum eigenvalues of the problem are considerably different from each other so that the condition number (i.e. the ratio  $\rho = M/m$  of the largest to smallest eigenvalues of the matrix  $A$ ) is large, then the method of steepest descent exhibits poor rates of convergence. At the outset steady convergence is achieved but progress becomes steadily slower, the closer to the minimum the approximations get. Figure 1.4 depicts such a situation where the large condition number causes slow convergence to occur. In this example the minimum point lies in a narrow valley of a quadratic function. If either  $\lambda_1 = 0$  or  $\lambda_d = \infty$  then the method of steepest descent is slow to converge.

For the quadratic case, the value of  $\alpha^{(k)}$ , such that (1.6) holds true, can be calculated explicitly. The derivative of a general quadratic function of the form (1.3)

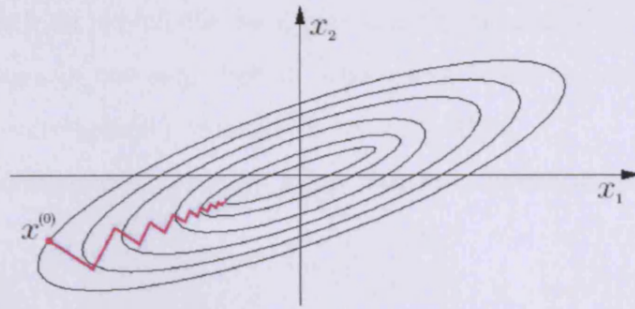


Figure 1.4: Graph showing a typical path of descent made by the steepest descent algorithm when  $x^*$  lies in a narrow valley.

is

$$g^{(k)} = g(x^{(k)}) = \nabla f(x^{(k)}) = Ax^{(k)} - b.$$

The value of  $\alpha^{(k)}$  is determined by minimising

$$f(x^{(k)} - \alpha g^{(k)}) = \frac{1}{2}(x^{(k)} - \alpha g^{(k)})^T A(x^{(k)} - \alpha g^{(k)}) - (x^{(k)} - \alpha g^{(k)})^T b.$$

Differentiating this function with respect to  $\alpha$  and equating the resulting derivative to zero gives

$$\alpha^{(k)} = \frac{(g^{(k)}, g^{(k)})}{(Ag^{(k)}, g^{(k)})},$$

where  $(a, b)$  denotes the Euclidean inner product. The steepest descent algorithm for the quadratic case can thus be written as

$$x^{(k+1)} = x^{(k)} - \frac{(g^{(k)}, g^{(k)})}{(Ag^{(k)}, g^{(k)})} g^{(k)}. \quad (1.7)$$

Multiplying by  $A$  and subtracting  $b$  gives

$$Ax^{(k+1)} - b = Ax^{(k)} - b - \frac{(g^{(k)}, g^{(k)})}{(Ag^{(k)}, g^{(k)})} Ag^{(k)},$$

which is equivalent to

$$g^{(k+1)} = g^{(k)} - \frac{(g^{(k)}, g^{(k)})}{(Ag^{(k)}, g^{(k)})} Ag^{(k)}. \quad (1.8)$$

It can be shown that the method of steepest descent converges to the unique minimum point of a convex quadratic function; see [44] for a proof of this. The rate of convergence is, however, dependent on certain conditions. In the case when  $m = M$ ,

i.e. all eigenvalues are equal, the level sets are spherical and the steepest descent algorithm converges in one step. For all other cases, however, some investigation is needed into the convergence properties of the algorithm.

A rate of convergence at iteration  $k$  for gradient algorithms can be defined as

$$r^{(k)} = \frac{(g^{(k+1)}, g^{(k+1)})}{(g^{(k)}, g^{(k)})}. \quad (1.9)$$

First note that

$$(g^{(k+1)}, g^{(k+1)}) = (g^{(k)}, g^{(k)}) - 2\alpha^{(k)}(Ag^{(k)}, g^{(k)}) + (\alpha^{(k)})^2(A^2g^{(k)}, g^{(k)}), \quad (1.10)$$

where  $\alpha^{(k)} = (g^{(k)}, g^{(k)})/(Ag^{(k)}, g^{(k)})$  for the steepest descent algorithm. Dividing (1.10) through by  $(g^{(k)}, g^{(k)})$  gives

$$r^{(k)} = 1 - 2\alpha^{(k)} \frac{(Ag^{(k)}, g^{(k)})}{(g^{(k)}, g^{(k)})} + (\alpha^{(k)})^2 \frac{(A^2g^{(k)}, g^{(k)})}{(g^{(k)}, g^{(k)})}$$

which for the SD algorithm is equal to

$$r_{SD}^{(k)} = \frac{(A^2g^{(k)}, g^{(k)})(g^{(k)}, g^{(k)})}{(Ag^{(k)}, g^{(k)})^2} - 1. \quad (1.11)$$

The asymptotic rate of convergence for gradient algorithms can then be defined as

$$R = \lim_{k \rightarrow \infty} \left( \prod_{i=1}^k r^{(i)} \right)^{1/k} \quad (1.12)$$

The steepest descent algorithm converges  $R$ -linearly, i.e.  $R_{SD} < 1$ . Incidentally, if a gradient algorithm is such that its rate of convergence  $r^{(k)} \rightarrow 0$  as  $k \rightarrow \infty$  then it is said to have  $R$ -superlinear convergence. In order to compare convergence rates of two algorithms, both with linear convergence, it is necessary to evaluate their respective asymptotic rates of convergence. Whilst it is not possible to get an exact analytic formula for the asymptotic rate of convergence of the steepest descent algorithm, an upper bound for this convergence rate is obtainable by employing the Kantorovich inequality, see [41]. This can be considered as the worst-case rate for the algorithm and is equal to

$$R_{\text{ref}} = \left( \frac{M - m}{M + m} \right)^2 = \left( \frac{\rho - 1}{\rho + 1} \right)^2. \quad (1.13)$$

The worst-case rate of convergence of the steepest descent algorithm thus depends on the condition number of the matrix  $A$  in (1.3) i.e. the ratio  $\rho = m/M$  of the smallest to largest eigenvalues  $m$  and  $M$ .

The actual rate of convergence is known to depend on the starting vector  $x^{(0)}$  in a complicated manner and is studied in [1, 50, 53] with the aid of a useful technique, renormalisation. For detailed discussion on this and other results concerning the asymptotic behaviour of the steepest descent algorithm see Section 2.1.

There are, of course, ways of improving the rate of convergence of the steepest descent algorithm. Even a small adaptation to the algorithm can result in a dramatic improvement in convergence rate. For example, Booth [9] suggested that for ill-conditioned systems adding a relaxation coefficient of 0.9 to the algorithm for some iterations will improve the performance of the algorithm. The iterative formula would thus become

$$x^{(k+1)} = x^{(k)} - 0.9 \left( \frac{g^{(k)T} g^{(k)}}{g^{(k)T} A g^{(k)}} \right) g^{(k)}$$

for, for instance, four iterations and then a single iteration using the exact formula would be made. There exist many other algorithms that have been created from modifying the steepest descent algorithm, see Section 1.5 for discussion on some of these.

### 1.3 The Conjugate Gradient Method

Presently, the conjugate gradient method is a very popular algorithm for the minimisation of convex quadratic functions. It was first introduced as a method for solving systems of linear equations whose matrices are symmetric and positive-definite, see Sec 1.4. [38].

The algorithm can be defined as follows. Starting with an initial vector  $x^{(0)}$ , the method of conjugate gradients first takes a step in the direction of steepest descent i.e.  $\delta^{(0)} = -g^{(0)}$ . Then the method follows the iterative procedure

$$x^{(k+1)} = x^{(k)} - \alpha^{(k)} \delta^{(k)},$$

with

$$\alpha^{(k)} = \frac{(g^{(k)}, \delta^{(k)})}{(A\delta^{(k)}, \delta^{(k)})}, \text{ and } \delta^{(k+1)} = -g^{(k+1)} + \beta^{(k)}\delta^{(k)},$$

where  $\beta^{(k)} = (g^{(k+1)}, A\delta^{(k)})/(\delta^{(k)}, \delta^{(k)})$ . See [44] for a proof that  $\beta^{(k)}$  and  $\alpha^{(k)}$  have the alternative formulae

$$\alpha^{(k)} = (g^{(k)}, g^{(k)})/(A\delta^{(k)}, \delta^{(k)}) \text{ and } \beta^{(k)} = (g^{(k+1)}, g^{(k+1)})/(g^{(k)}, g^{(k)})$$

respectively.

The method of conjugate gradients is advantageous for several reasons; the formula is relatively simple, being only slightly more complicated than that of the method of steepest descent, the advances towards the minimum point are generally fairly uniform at each iteration and the direction at each step is always linearly independent of all other step directions.

A disadvantage of the method of steepest descent is that it often finds itself repeating descent directions during the course of its descent to the solution. The method of conjugate gradients on the other hand only travels in each necessary direction once. The way the method achieves this is by forming  $d$  conjugate search directions  $\delta^{(0)}, \delta^{(1)}, \dots, \delta^{(d-1)}$ , where a definition of conjugacy is given by Definition 1.3.1.

**Definition 1.3.1.** Two vectors  $\delta^{(1)}$  and  $\delta^{(2)}$  are said to be conjugate to each other if for some matrix  $A$ ,

$$\delta^{(1)T} A \delta^{(2)} = 0.$$

These directions can be produced by using the conjugate Gram-Schmidt process, see [60]. After each iteration, one of the  $d$  elements of  $x^*$  will be found and thus after  $d$  iterations the solution will be known. Such algorithms which locate the minimum point of the function in  $d$  iterations are said to be quadratically convergent. It is possible that the solution can be found in less than  $d$  iterations if, in any particular iteration, the value of  $\alpha^{(k)}$  is 0.

The method is not free from potential problems however. For the conjugate gradient algorithm to converge to the solution in  $d$  iterations each computation needs to be precise. Even small errors may result in the search vectors losing conjugacy

which would result in more than  $d$  iterations being required. In reality it is not always possible to maintain this level of accuracy whilst still retaining computational efficiency; thus often a play off between accuracy and speed is needed.

### Convergence

Whilst the method of conjugate gradients converges to the solution in  $d$  iterations in ideal conditions, the method could be applied to a problem so large that it is not practical to run the algorithm for  $d$  iterations. For this reason, and also because in practice the algorithm is generally restarted frequently to avoid loss of conjugacy, convergence analysis for this algorithm is still required.

It must be noted that ill-conditioned problems will have a similar impact on the convergence of the conjugate gradient method to the effect they have on the method of steepest descent. As with the method of steepest descent, there exist many modifications of the algorithm which seek to improve on the efficiency and reliability of the algorithm. One such method is the Fletcher-Reeves algorithm, suggested in [31].

For problems where  $d$  is large, the search directions may begin to become inefficient after a few iterations. One way to get the process back on track is to restart the process with an iteration of steepest descent and continue in this manner thus operating the method in cycles.

## 1.4 Iterative Methods for Solving Linear Systems

The task of solving a system of linear equations  $Ax = b$ , where  $A$  is a  $n \times n$  non-singular matrix and  $b$  is a given vector, is a common one, arising in many fields. The solution to such a system is  $x^* = A^{-1}b$ . In some circumstances, calculating the inverse  $A^{-1}$  can be time consuming and so a more efficient means in which to find the solution  $x^*$  is sought.

An application of particular interest is the solution of linear systems obtained from the discretisation of partial differential equations. Many problems in areas such as physics and engineering require the solution of systems of partial differential

equations, for example the study of elasticity and fluid flow. In the majority of cases these equations are non-linear; however, after the discretisation process coupled with some iterative methods to tackle the problem of non-linearity, a linear system of equations remains. These systems can be large and are often sparse.

Whilst it is possible to solve linear systems via direct methods, those based on Gaussian elimination perhaps being the most well-known, these methods frequently prove to be too expensive, with respect to both computer time and storage required, particularly when the matrices are large and sparse. For a description of several direct methods see [47]. Often direct methods cannot take full advantage of the properties of sparse or specially structured matrices. Conversely, matrix-multiplication can benefit greatly from sparseness. The number of operations needed to compute a product involving a matrix with several zero entries is much reduced and only storage of those entries that are non-zero is required. For this reason iterative methods that use a moderate number of matrix multiplications can often outperform direct methods.

Due to their nature, iterative methods will only provide an approximation to the exact solution. In reality, however, even direct methods can only find a solution to a certain degree of accuracy (due to the limited precision of the floating point operations carried out by the computer) so the approximate solution of iterative methods is not seen as a disadvantage.

Iterative methods are thought to date back as far as Gauss (1777–1855), where he suggested solving a four-dimensional system of equations by means of repeatedly solving the component which contained the largest residual. Since then many iterative methods have been suggested. These methods can be split into two main groups; stationary iterative methods and Krylov subspace methods. For a chronological review of some of the more important advances in the field of iterative methods see [72].

### 1.4.1 Stationary Iterative Methods

Stationary iterative methods include amongst them, the Jacobi method and the Gauss-Seidel method. While they are generally easy to implement and analyse,



these methods have the main drawback that their convergence is only guaranteed under fairly restrictive conditions and the rate of convergence is generally slow.

Most methods of this form involve splitting the matrix  $A$  in some way. In the general case  $A$  is denoted as

$$A = M - N ,$$

where  $M$  is a non-singular matrix. The solution of  $Ax = b$  will then be equal to  $x = M^{-1}Nx + M^{-1}b$  and thus the general updating formula for algorithms of this type can be written as

$$x^{(k+1)} = M^{-1}Nx^{(k)} + M^{-1}b .$$

The Jacobi method partitions  $A$  as follows

$$A = D + E + F ,$$

where  $D$  is the diagonal of matrix  $A$ , and  $E$  and  $F$  are the strictly upper and lower triangular parts of  $A$  respectively. The matrix  $A$  is then split into two with  $M$  being set equal to the diagonal of  $A$ , i.e.  $M = D$  and  $N = -(E + F)$ . Since  $M$  is diagonal it is easy to invert. Although this method is simple, convergence is very slow.

Seidel, a student of Jacobi, suggested a variation on this method, namely the Gauss-Seidel method. Seidel himself however recommended not to use it. The Gauss-Seidel algorithm differs from the Jacobi method in the manner in which  $A$  is split. For this algorithm  $M$  is set equal to the lower triangle of  $A$ , i.e.  $M = D + E$  and  $N = -F$ . This algorithm makes use of updated values as soon as they become available. Convergence is faster than that of the Jacobi algorithm but it is still relatively slow.

Another algorithm, improving on the performance of both the Jacobi and the Gauss-Seidel algorithm is that of the Successive Over Relaxation (SOR) Method. This faster algorithm, which can be derived from the Gauss-Seidel method by introducing an extrapolation parameter, provoked a great deal of research in the 1950s and 1960s. For a detailed study of the SOR method see [71]. For results concerning the stability of stationary iterative algorithms as well as a survey of error analysis see [39].

### 1.4.2 Krylov Subspace Methods

A Krylov subspace  $K$ , for a symmetric matrix  $A$  and vector  $b$ , is defined as

$$K = \text{Span}(b, Ab, \dots, A^{n-1}b).$$

Krylov Subspace Methods for solving systems of linear equations create an orthonormal basis of  $K$ . The vectors  $x^{(k+1)}$  are constructed by minimising the residual,  $r^{(k)} = b - Ax^{(k)}$  over the subspace formed. Krylov subspace methods suffer from the vectors  $x^{(k)}$  becoming virtually linearly dependent and for this reason methods of this type often require the aid of an orthogonalisation process such as Arnoldi orthogonalisation, see [6]. Algorithms belonging to this group include the Generalised Minimal Residual Algorithm (or GMRES), the Stabilised Biconjugate Gradient Method (or BiCGSTAB), the Lanczos algorithm and, perhaps the most widely used of them all, the Conjugate Gradient Method.

The GMRES method, see [59], is a generalised version of Paige and Saunders' MINRES method (see [51]) which minimises the norm of the residual vector over a Krylov subspace at every iteration. A description of some extensions of the GMRES algorithms are given in Chapter 7 of [47].

The Lanczos method was developed as an algorithm for determining the eigenvalues of a square matrix, see [43]; however it can also be used to solve linear systems. In fact the conjugate gradient method is a particular case of the Lanczos method.

#### Method of Steepest Descent

The problem of solving a system of linear equations  $Ax = b$  corresponds exactly to that of finding the minimum (or maximum) point  $x^*$  of a quadratic function of the form (1.3). As was stated earlier, the gradient of a quadratic function can be written as  $g(x) = Ax - b$ . At the minimum point,  $g(x^*) = 0$  thus the equality  $Ax - b = 0$ , or in other words  $Ax = b$ , holds. Consequently some iterative methods for solving quadratic optimisation problems can be adapted to the context of solving linear systems. The steepest descent algorithm, sometimes known as the optimum gradient method in this context, is a prime example of this. In fact, the method of steepest descent was first suggested in the context of the solution of linear systems

of equations, see [12]. Mention of this application of the algorithm was also given by Curry in [14].

### Conjugate Gradient Method

Apart from being a very successful optimisation algorithm, as discussed in section 1.3, the conjugate gradient method is a popular iterative method for solving sparse systems of linear equations where  $A$  is a positive definite symmetric matrix. Indeed the conjugate gradient method was first suggested as a means to solve linear systems; it was not until much later that its full potential as an optimisation algorithm was realised. It is generally fast and requires minimal storage as only the derivative and the computation must be kept after each iteration. The algorithm can however suffer from a loss of conjugacy and in the worst situations can fail to converge. The method was first developed independently by Hestenes and Steifel, see [38] and since its proposal much research has been dedicated to this algorithm and its convergence rate. Several modifications have been suggested over the years to improve upon its convergence rates, mostly involving preconditioning of some kind. The algorithm has also been extended to solve non-linear systems and generalised to deal with nonsymmetric matrices and indefinite matrices. For a chronological survey of developments to both the conjugate gradient algorithm and the Lanczos method see [35].

### Preconditioning

For ill-conditioned systems, i.e. those where the condition number  $\rho = \lambda_d/\lambda_1$  of the matrix  $A$  is large, it is often beneficial to first precondition the system before performing an iteration. The rate of convergence of algorithms designed to solve systems of linear equations worsens when the condition number is large but if the system can be multiplied by a matrix  $B^{-1}$  which reduces the condition number and the system  $B^{-1}Ax = B^{-1}b$  is solved in place of the original system, improved convergence rates can be achieved.

A simple preconditioner is  $B = D$  where  $D$  is a matrix made up of only the diagonal elements of  $A$ . This is known as the Jacobi preconditioner. A more complicated

example of a preconditioner is the successive over relaxation preconditioner

$$B = \left( \frac{D}{\omega} + L \right) \frac{\omega}{2 - \omega} D^{-1} \left( \frac{D}{\omega} + U \right),$$

where  $D$  is the diagonal of  $A$ ,  $L$  is the lower triangle,  $U$  is the upper triangle and  $0 < \omega < 2$  is a relaxation parameter. If  $A$  is symmetric then  $U = L^T$  and  $B$  becomes the so-called symmetric successive over relaxation (SSOR) preconditioner. The introduction of preconditioning gives rise to the preconditioned conjugate gradient method. In [7] a SSOR preconditioner is used in conjunction with the conjugate gradient method and it is shown that in some circumstances the preconditioner has the effect of reducing the condition number by up to the power  $1/2$ .

## 1.5 Recent Successful Gradient Algorithms

In this section a review of some recent developments in the area of gradient algorithms, mainly through modifications of the classical method of steepest descent, is given.

### 1.5.1 The Barzilai-Borwein Algorithm

Possibly the most important recent advance in the field of gradient descent algorithms is due to Barzilai and Borwein. In their paper [8], an adaptation of the classical steepest descent method is suggested whereby a two-point step size is used. The idea came from finding a two-point approximation to the secant equation which underlies quasi-Newton methods (see for example [25] for details on quasi-Newton methods). The search direction in this new algorithm, now commonly known as the Barzilai-Borwein (BB) algorithm, is the same as for steepest descent but two new step lengths are suggested;

$$\alpha_{\text{BB1}}^{(k)} = \frac{(\Delta x, \Delta g)}{(\Delta g, \Delta g)} \quad \text{and} \quad \alpha_{\text{BB2}}^{(k)} = \frac{(\Delta x, \Delta x)}{(\Delta x, \Delta g)}, \quad (1.14)$$

where  $\Delta x = x^{(k)} - x^{(k-1)}$  and  $\Delta g = g^{(k)} - g^{(k-1)}$ . In the quadratic case these correspond to

$$\alpha_{\text{BB1}}^{(k)} = \frac{(g^{(k-1)}, g^{(k-1)})}{(Ag^{(k-1)}, g^{(k-1)})} \quad \text{and} \quad \alpha_{\text{BB2}}^{(k)} = \frac{(Ag^{(k-1)}, g^{(k-1)})}{(A^2g^{(k-1)}, g^{(k-1)})}, \quad (1.15)$$

the step length  $\alpha_{BB1}^{(k)}$  being slightly easier to implement in practice. This step length is exactly the step length for the standard steepest descent algorithm but at the previous iteration. For the positive definite quadratic case the BB algorithm is proved to be  $R$ -superlinearly convergent in two dimensions and the algorithm is shown numerically to have a significantly faster rate of convergence when compared with the original steepest descent algorithm. Proof of convergence of the BB algorithm for the strictly convex quadratic case is extended to any dimensional problems by Raydan in [57] although for dimensions greater than two the rate of convergence is no longer  $R$ -superlinearly convergent. Indeed, in [19], it is established that the BB algorithm converges  $R$ -linearly when applied to a strongly convex quadratic function of greater than two dimensions.

It must be noted that the behaviour of the BB algorithm is completely non-monotonic i.e. it is possible that an iteration of this algorithm could cause the gradient norm,  $\|g^{(k)}\|$ , to increase. Due to this property the algorithm is less sensitive to ill-conditioning since unlike monotonic algorithms, such as the original steepest descent algorithm, where the algorithm is often limited to small step-sizes when the condition number is large, the BB algorithm is not. An improvement to the algorithm is suggested by Barzilai and Borwein in [8] for situations where the gradient norm does not decrease; the point calculated by means of the original steepest descent algorithm should be compared with that obtained via the BB algorithm after a set number of iterations (say 2 or 3) and whichever point is lowest should be taken. There are however no numerical results to support this theory and since their suggestion will destroy the very nature of the algorithm, the improvement in convergence rate gained by the BB algorithm is likely to be lost.

### 1.5.2 Generalisations of the Barzilai-Borwein Algorithm

The massive improvement in convergence rate the BB method demonstrates when compared with the steepest descent algorithm has provoked a great deal of interest in the algorithm and many adaptations and generalisations of the algorithm have been suggested. In [48] the BB algorithm is used in conjunction with pre-conditioning techniques in an attempt to improve further upon the rate of convergence. Here the

pre-conditioned BB (PBB) method is applied to a large sparse positive definite system of linear equations and it is seen that, when compared with the pre-conditioned conjugate gradient method, the PBB method is competitive and in some cases better.

In [48] the BB method is embedded in a non-monotone line search strategy in order to ensure global convergence for the general non-linear case. In many global strategies the step length of the underlying algorithm is accepted if the so-called Armijo-Goldstein-Wolfe conditions are met (see [5, 34, 70] for details on these conditions) however, if these conditions are applied to the BB method, a decrease in the objective function value is enforced at every iteration. This would destroy the non-monotone nature of the algorithm and reduce the convergence rate to that of the method of steepest descent. For this reason the BB method is combined with the non-monotone line search suggested in [36], which has previously been combined with several optimisation algorithms such as Newton's method. This non-monotone line search technique consists of computing  $\alpha^{(k)}$  such that

$$f(x^{(k)} + \alpha^{(k)}g^{(k)}) \leq \max_{0 \leq i \leq s(k)} f(x^{(k-i)}) + \delta \alpha^{(k)}(g^{(k)}, g^{(k)}), \quad (1.16)$$

where  $s(k) = \min(k, S - 1)$ ,  $S$  is an integer and  $\delta$  is small. This method enforces much weaker conditions and allows for an increase in objective function value thus allowing the BB method to keep its non-monotone behaviour. The combined BB line search technique is referred to as the Global Barzilai-Borwein (GBB) algorithm. The convergence of this algorithm is established and the method is seen to compare favourably with some conjugate gradient methods except in ill-conditioned problems. It must be noted however, that problems to which algorithms of this nature are generally applied to tend to be ill-conditioned. The method is also heavily dependent on the choice of the parameter  $S$ . It is stated in [19] that the BB algorithm is locally  $R$ -linearly convergent when applied to general non-linear objective functions which means that when combined with the non-monotone line search of [36] to form the GBB algorithm, the BB step size will always be accepted by the non-monotone line search when the current iterate is in the vicinity of the solution.

Adapting the approach in [36], new stabilisation schemes that combine watchdog

techniques (see [13] for details of the watchdog technique) with non-monotone line search procedures are proposed in [37]. It is demonstrated that when the BB algorithm is combined with one of the suggested non-monotone line search techniques, global convergence is guaranteed. An advantage of the suggested technique is that a sequence of steps is permitted where different formulae can be used to compute the step size at each step.

In [40] an inexact Barzilai-Borwein method is used where, instead of using the exact residual at the  $k$ -th iteration  $g^{(k)}$ , some approximation of the linear system  $\bar{g}^{(k)}$  is used instead. It is proven, under the assumption  $\|\bar{g}^{(k)} - g^{(k)}\| \leq \eta \|\bar{g}^{(k)}\|$  for some small  $\eta > 0$ , that the algorithm converges  $R$ -linearly. The suggested algorithm can be adapted to solve saddle point problems of the form

$$\begin{pmatrix} A & B^T \\ B & -C \end{pmatrix} \begin{pmatrix} u \\ p \end{pmatrix} = \begin{pmatrix} f \\ h \end{pmatrix},$$

where  $A \in \mathbb{R}^{n \times n}$  is a symmetric positive definite matrix and  $C \in \mathbb{R}^{m \times m}$  is symmetric positive semi-definite.  $R$ -linear convergence for this adapted algorithm is obtained by extending the previous result.

A detailed review of the Barzilai-Borwein algorithm and its adaptations is given by Fletcher in [29]. Here some insight into the behaviour of the BB algorithm is given. The possibility of non-monotonicity is partially explained by the fact that if  $1/\alpha_{BB1}^{(k)} < \lambda_d/2$  then  $|g_d^{(k+1)}| > |g_d^{(k)}|$ . Also if  $1/\alpha_{BB1}^{(k)}$  (known as the Rayleigh quotient) is close to the smallest eigenvalue,  $\lambda_1$ , then  $|g_d^{(k+1)}|/|g_d^{(k)}|$  can approach  $(\lambda_d/\lambda_1) - 1$ , so that the degree to which the non-monotonicity occurs is dependent on the condition number  $\rho$ .

In most circumstances the conjugate gradient method outperforms the BB method however the CG method relies on the search direction being from a quadratic model. If the calculation deviates from the quadratic model, through round-off errors for example, then the method can become inefficient thus if less precision is used the BB method is much more competitive. This phenomenon is shown numerically to be true.

It is also noted in [29] that a comparatively large increase in the gradient norm,  $\|g^{(k)}\|$ , is usually followed by a desirable large decrease in  $\|g^{(k+1)}\|$  which overall gives

the net effect of reducing the value of the gradient norm. It is thought that this phenomenon occurs when  $\rho$  is large and the step size is relatively small which has the effect of diminishing the large components of the gradient (i.e.  $g_i^{(k)}$  for large  $i$ ) so that the small components dominate which then gives rise to a large  $\alpha^{(k+1)}$  which subsequently causes the large components of  $g^{(k+1)}$  to dominate thus causing a spike of activity. The GBB method suggested in [48] limits these large increases in  $\|g^{(k)}\|$  which will prevent the large decreases in  $\|g^{(k+1)}\|$  which follow. This is suggested as a reason why the GBB method has poor convergence for ill-conditioned problems. As a remedy Fletcher recommends selecting much larger values of the parameter  $S$  in (2.17) if an ill-conditioned problem is thought likely but he also states that this could cause difficulty for problems which are far from quadratic.

In the paper by Friedlander et al, [33], a generalisation of the steepest descent and Barzilai-Borwein algorithms for the solution of large scale symmetric positive definite systems is given. This generalised algorithm is defined as

$$x^{(k+1)} = x^{(k)} - \frac{(g^{(\nu(k))}, g^{(\nu(k))})}{(Ag^{(\nu(k))}, g^{(\nu(k))})} g^{(k)},$$

with  $\nu(k)$  chosen arbitrarily from the set  $\{k, k-1, \dots, \max\{0, k-t\}\}$  where  $t$  is some positive integer; i.e. the step length is chosen to be one of the last  $t$  step lengths. If  $\nu(k) = k$  then the step length is simply that of the steepest descent algorithm and if  $\nu(k) = k-1$  is chosen then the step length corresponds to that of the BB method, however, there are now many other choices of step length as well.

This new method is proven to be globally convergent through a generalisation of the proof in [57] and numerical experiments show convergence rates are favourable when compared to the CG method with respect to both storage requirements and computations if precision needed is low.

Some choices of  $\nu(k)$  suggested in the paper [33] are given below.

1.  $\nu(k)$  is a random integer between  $\bar{k}$  and  $k$  where  $\bar{k} = \max\{0, k-t\}$
2.  $\nu(k) = k$  if  $\nu(k-1) < \bar{k}$  or  $k = 0$ ,  $\nu(k) = \nu(k-1)$  otherwise
3.  $\nu(k) = \bar{k}$
4.  $\nu(k) = \arg \max \left( \frac{(g^{(k)}, g^{(k)})}{(Ag^{(k)}, g^{(k)})}, \dots, \frac{(g^{(\bar{k})}, g^{(\bar{k})})}{(Ag^{(\bar{k})}, g^{(\bar{k})})} \right)$



5.  $\nu(k) = \bar{k}$  if  $k$  is even,  $\nu(k) = k$  if  $k$  is odd
6.  $\nu(k)$  is a random integer between  $\bar{k}$  and  $k - 1$

An advantage of these methods, named gradient methods with retards (GMR), is that the step length does not need to be calculated at every iteration, in particular the 2nd method described above only requires 1 step length to be computed every  $t + 1$  iterations. A pre-conditioned version of the gradient method with retards (PGMR) is also presented in [33] (a follow on from the PBB method of [48]).

The gradient method with retards has faster convergence the larger the retard parameter  $t$ ; however, as  $t$  increases the method suffers from a loss of precision due to the non-monotone behaviour of the algorithm. In [42] a technique is used to smooth down this non-monotone behaviour in order to make a larger choice of  $t$  viable. It is also shown in this paper that the parameter  $t$  can be chosen adaptively in order to combat loss of precision. In essence the adaptive choice of  $t$  is a compromise between speed of convergence and the stability of the algorithm. The adaptive method involves choosing an aggressive value of  $t$  as often as possible but after a preset number of consecutive increases in  $\|g^{(k)}\|$  has occurred the method is kept in check by a preset number of iterations in which the BB step length is used.

In [18] the gradient method with retards, with  $\nu(k) = k$  if  $\nu(k - 1) < \bar{k}$  or  $k = 0$ ,  $\nu(k) = \nu(k - 1)$  otherwise, is discussed further and is referred to as the cyclic Barzilai-Borwein method. Since, for this algorithm, the same BB step length is reused for  $t$  iterations in a row, the amount of calculations of the step length required is reduced by  $1/t$ . Global convergence of the cyclic BB algorithm is ascertained in [33] for strongly convex quadratic objective functions and in [21] the algorithm is shown to converge  $R$ -linearly for this case. In [16] it is discovered that the cyclic BB method is locally linearly convergent at a local minimiser with a positive definite hessian matrix  $A$ . It is shown numerically that the cyclic BB algorithm can be locally superlinearly convergent for strongly convex quadratic functions if  $t > d/2 \geq 3$ . The algorithm is extended by incorporating a non-monotone line search, in order to produce a globally convergent algorithm in the non-quadratic case, and since the choice of  $t$  has an impact on performance, an adaptive method for choosing suitable

cycle lengths is proposed. Numerical results show this algorithm to be better than the standard BB method.

In [24] an adaptive non-monotone line search is proposed which when combined with the BB method is globally convergent. This is an adaptive version of the technique suggested in [48] with the reasoning being that if the non-monotone line search is so dependent on the parameter  $S$  in (2.17) then it would be beneficial if the value of this parameter could be re-selected at each iteration. It is shown numerically here that the line search method proposed is particularly well suited to the BB method.

The original Barzilai-Borwein step length was derived from the quasi-Newton secant equation but, as is shown in [15], it can also be obtained from an interpolation point of view. In [15] two modified BB algorithms are proposed with step lengths:

$$\alpha^{(k)} = \frac{(\Delta x, \Delta x)}{2(f^{(k-1)} - f^{(k)} + (g^{(k)}, \Delta x))}$$

and

$$\alpha^{(k)} = \frac{(\Delta x, \Delta x)}{6(f^{(k-1)} - f^{(k)}) + 4(g^{(k)}, \Delta x) + 2(g^{(k-1)}, \Delta x)},$$

where  $\Delta x = x^{(k-1)} - x^{(k)}$ . Both are shown to be globally convergent when combined with the non-monotone line search technique in [36]. These new methods are compared with the GBB method via numerical experiments and are shown to perform better, requiring fewer storage points and less iterations to converge.

In [16] an algorithm is suggested in which the step length alternates between that of the SD algorithm and that of the BB algorithm. In this way

$$\alpha_{AS}^{(k)} = \begin{cases} \alpha_{SD}^{(k)} & \text{for odd } k \\ \alpha_{BB}^{(k)} & \text{for even } k \end{cases}$$

and the method avoids the zigzagging behaviour which the SD algorithm exhibits. This method is referred to as the alternate step (AS) method and requires less step length computations than either the SD or BB methods since the BB step length at iteration  $k$  is exactly the SD step length at iteration  $k - 1$ . It is a particular case of the GMR method 5 described above and is suggested in [33] where  $t = 1$ . The AS method can be adapted so that a BB step length is taken after every  $t - 1$  iterations

of the SD step length and conversely it can be adapted so that a SD step length is taken after every  $t - 1$  iterations of the BB step length.  $R$ -linear convergence for the AS algorithm is established for the symmetric positive definite case and numerical experiments show the AS method to be a good alternative and rival to the BB algorithm. The AS method is a different formulation of the Cyclic-Barzilai-Borwein algorithm, see [18].

Another modification of the BB algorithm is introduced in [58] and is named the Cauchy-Barzilai-Borwein (CBB) algorithm. For this algorithm, each iteration can be viewed as two consecutive steepest descent iterations where the step length only needs to be computed once but is used twice. The computational cost is thus still the same as that of the standard SD algorithm despite there being, in effect, twice the number of iterations. The algorithm is defined as follows:

set

$$y^{(k)} = x^{(k)} - \frac{(g^{(k)}, g^{(k)})}{(Ag^{(k)}, g^{(k)})} g^{(k)},$$

then

$$x^{(k+1)} = y^{(k)} - \frac{(g^{(k)}, g^{(k)})}{(Ag^{(k)}, g^{(k)})} g^{(k)}.$$

This can be equivalently written as

$$x^{(k+1)} = x^{(k)} - 2 \frac{(g^{(k)}, g^{(k)})}{(Ag^{(k)}, g^{(k)})} g^{(k)} + \left( \frac{(g^{(k)}, g^{(k)})}{(Ag^{(k)}, g^{(k)})} \right)^2 Ag^{(k)}, \quad (1.17)$$

and in a similar way to the BB algorithm exhibits non-monotonic behaviour in its descent to the minimum point. It is shown that this algorithm converges  $Q$ -linearly, i.e. its convergence rate is such that

$$\limsup_{k \rightarrow \infty} r^{(k)} < 1,$$

( $Q$ -linear convergence implies  $R$ -linear convergence). Numerical comparison with the BB algorithm shows that, for the test problems chosen, the CBB algorithm converges in fewer iterations.

The asymptotic behaviour of some of these gradient methods, including the BB algorithm and others which are competitive with conjugate gradient methods, is studied in [17]. It is observed in the quadratic case that a transition between superlinear and linear convergence occurs at a certain dimension which depends on

the method. Simplifying the algorithms, by neglecting some terms in the recurrence relations, enables analysis into why this transition from superlinear to linear convergence occurs when it does.

### 1.5.3 Other New Gradient Methods

In [58] the standard SD algorithm is modified by introducing a relaxation parameter  $\gamma^{(k)}$  which can vary between 0 and 2. If  $\gamma^{(k)} = 1$  the step length is reduced to that of the standard SD algorithm and if  $\gamma^{(k)} = 2$  then  $f(x^{(k+1)}) = f(x^{(k)})$ . Convergence of this method, named the relaxed steepest descent (RSD) method, is established under a mild assumption and the question of what are good choices for the relaxation parameter is posed. A numerical experiment is undertaken comparing the standard SD method with the RSD method where the relaxation parameter is chosen randomly and it is seen that the RSD method outperforms standard SD although its rate of convergence is not as fast as the BB algorithm.

In [22] a gradient algorithm is developed in which the step sizes alternately minimise the function value  $f(x)$  and the gradient norm  $\|g^{(k)}\|$ , i.e.

$$\alpha^{(2k-1)} = \arg \min_{\alpha} \|g(x^{(2k-1)} - \alpha g^{(2k-1)})\|$$

and

$$\alpha^{(2k)} = \arg \min_{\alpha} (f(x^{(2k)} - \alpha g^{(2k)}))$$

which in the quadratic case equates to

$$\alpha^{(2k-1)} = \frac{(Ag^{(k)}, g^{(k)})}{(A^2g^{(k)}, g^{(k)})} \quad \text{and} \quad \alpha^{(2k)} = \frac{(g^{(k)}, g^{(k)})}{(Ag^{(k)}, g^{(k)})}.$$

The idea behind this method is to produce a monotone algorithm (unlike the BB algorithm) with a better rate of convergence than the steepest descent algorithm. It is proven in the convex quadratic case that the algorithm (called the alternate minimisation gradient algorithm) is  $Q$ -linear convergent and numerical results for a problem with a symmetric positive definite hessian matrix show the algorithm to have a superior rate of convergence to the steepest descent algorithm and, if low precision is required, the method is even competitive with the BB algorithm.

Two variants of this method, named here as shortened SD step gradient method 1 and 2 (SS1 and SS2) are suggested. In SS1 the step length is

$$\alpha_{SS1}^{(k)} = \gamma_1 \alpha_{SD}^{(k)}$$

and in SS2

$$\alpha_{SS2}^{(k)} = \begin{cases} \gamma_2 \alpha_{SD}^{(k)} & \text{odd } k \\ \alpha_{SD}^{(k)} & \text{even } k \end{cases}$$

where  $\gamma_1$  and  $\gamma_2$  are constants less than 1. These two methods have a similar performance to the alternate minimisation algorithm but they are much more readily extendable to unconstrained global minimization problems with inexact line search. It is pointed out that SS1 and SS2 are less effected by the condition number than the alternate minimisation algorithm as the shortened step lengths help prevent the occurrence of zigzags in the decent path.

Two new gradient methods are proposed in [74] and are given the names; Adaptive Steepest Descent algorithm (ASD) and Adaptive Barzilai-Borwein algorithm (ABB). The ASD method is a monotone algorithm which, like the alternate minimisation algorithm of [22], uses a combination of step lengths using the steepest descent formula  $\alpha_{SD}^{(2k)}$  and the step length formula that minimises the gradient norm. In [74] this step size is named the minimal gradient (MG) step length and for quadratic minimisation is equal to

$$\alpha_{MG}^{(k)} = \frac{(Ag^{(k)}, g^{(k)})}{(A^2g^{(k)}, g^{(k)})}.$$

Unlike the alternate minimization algorithm where the step length alternates systematically between  $\alpha_{MG}^{(2k-1)}$  and  $\alpha_{SD}^{(2k)}$ , the choice for the step length at iteration  $k$  for the ASD algorithm is determined as follows:

$$\alpha_{ASD}^{(k)} = \begin{cases} \alpha_{MG}^{(k)} & \text{if } \alpha_{MG}^{(k)}/\alpha_{SD}^{(k)} > k \\ \alpha_{SD}^{(k)} & \text{otherwise,} \end{cases}$$

where  $k \in (0, 1)$  is a parameter close to 0.5. The SD step length can be shortened slightly by subtracting  $\delta \alpha_{MG}^{(k)}$ , where  $\delta \in (0, 1)$ , to further improve upon convergence rates. By combining these two methods in this way, the worst case behaviour of both the SD algorithm and the MG method are prevented and it is shown that ASD algorithm converges  $Q$ -linearly.

The ABB algorithm, on the other hand, is a non-monotone algorithm and uses both the step sizes originally suggested by Barzilai and Borwein in [8]. Noting that  $\alpha_{BB1}^{(k)} = \alpha_{SD}^{(k-1)}$  and  $\alpha_{BB2}^{(k)} = \alpha_{MG}^{(k-1)}$ , in a similar way to the ASD algorithm, the ABB algorithm's step length at iteration  $k$  is chosen according to

$$\alpha_{ABB}^{(k)} = \begin{cases} \alpha_{BB2}^{(k)} & \text{if } \alpha_{BB2}^{(k)}/\alpha_{BB1}^{(k)} < k \\ \alpha_{BB1}^{(k)} & \text{otherwise,} \end{cases}$$

where  $k \in (0, 1)$ . It is shown that the ABB algorithm converges  $R$ -linearly.

In both of these algorithms there is a longer and shorter step length. It is suggested that the smaller step size might induce a desirable direction of descent whilst the larger step size brings about good reductions in  $\|g^{(k)}\|$ . Numerical examples show that both of these algorithms require fewer iterations to converge than the BB, AS and AM algorithms, the ABB method seems to perform particularly well in comparison with the rest when the condition number is large and high precision is required.

Another adaptation of the steepest descent algorithm is suggested in [73]. Again the motivation behind the adaptation is to produce an algorithm which yields faster rates of convergence than the SD method but one which still possesses the monotone property of the original algorithm. The reason for desiring a monotone algorithm is that an algorithm of this kind can be generalised to general non-linear functions without the need for the specialised non-monotone line search procedures of [36] that the BB algorithms requires. The proposed algorithm has the steepest descent step length in odd iterations and in even iterations the step length

$$\alpha^{(2k)} = \frac{2}{\sqrt{(1/\alpha_{SD}^{(2k-1)} - 1/\alpha_{SD}^{(2k)})^2 + 4\|g^{(2k)}\|^2/\|s^{(2k-1)}\|^2 + 1/\alpha_{SD}^{(2k-1)} + 1/\alpha_{SD}^{(2k)}}}, \quad (1.18)$$

where  $s^{(k-1)} = x^{(k)} - x^{(k-1)}$  and  $\alpha_{SD}^{(k)}$  indicates the steepest descent step length at iteration  $k$ . A further modification to this algorithm is made in which an iteration of the new step size (1.18) is made after every two consecutive steepest descent step lengths. Both variations are an improvement upon the classical SD algorithm but the second variation is the better of the two and works particularly well for small scale problems. It is also competitive with the BB algorithm for large scale

problems.

A gradient method with step size  $\alpha_{OPT}^{(k)} = 2/(m+M) = \arg \min \|I - \alpha A\|$  can be called an optimal step length in the sense that it minimises  $\|I - \alpha A\|$ . In practice, however,  $m$  and  $M$  are rarely known so this optimal step size cannot be used. In [20] a step size is suggested for symmetric positive definite problems which tends to the optimal step length,  $\alpha_{OPT}^{(k)}$  as  $k \rightarrow \infty$ . This new step length is

$$\alpha_{OPT2}^{(k)} = \frac{\|g^{(k)}\|}{\|Ag^{(k)}\|}. \quad (1.19)$$

It is proven that this method converges  $Q$ -linearly and through numerical results it is claimed that the algorithm performs slightly better than the SD algorithm; a possible reason for this being that  $\alpha_{OPT2}^{(k)} \leq \alpha_{SD}^{(k)}$ . It will, however, be shown in Chapter 4 (under the name the ‘square-root algorithm’) that this algorithm has exactly the same worse rate of convergence as the SD algorithm. A way to improve upon the proposed algorithm is suggested whereby the step length for the SD algorithm is used in odd iterations and the step length  $\alpha_{OPT2}^{(k)}$  is used in even iterations. Numerical results show that this combined method only requires half the number of iterations of the first method to converge.

A review of existing monotone gradient algorithms can be found in [23]. The original steepest decent algorithm, the relaxed SD algorithm of [58], the Alternate Minimisation algorithm, the shortened SD step gradient methods 1 and 2 of [22], the two methods described in [73] as well as the method presented in [20] are discussed and a study of their numerical behaviour is enabled by employing a long term observation technique. In many situations these algorithms only require very few iterations to converge to the solution; so to enable long term observations the gradient  $g^{(k)}$  is normalised by setting  $g^{(k)} = g^{(k)}/\|g^{(k)}\|$  before computing  $g^{(k+1)}$ . Examples are given to show that the Alternate Minimisation algorithm and the first method suggested in [73] may sometimes fall into a cycle. It is seen that the second method suggested in [73] shares a common property with the BB algorithm in so much that the gradient components with respect to the eigenvalues of  $A$  decrease together in both these algorithms. It is thought that this could be the reason why this method has a similar rate of convergence to the BB algorithm while the other

algorithms are worse.

In [4] a modification of the steepest descent algorithm is proposed by Andrei in which the classical step length is multiplied by a positive parameter,  $\theta^{(k)}$ , less than 1 which is recalculated at each iteration by means of backtracking (an inexact line search technique). Linear convergence in the convex case is established for the algorithm and it is shown to have a faster rate of convergence than the original steepest descent algorithm. Another gradient method which uses the steepest descent direction but which involves a backtracking method to calculate the step length is presented in [2]. It is shown to have a similar rate of convergence to the BB algorithm for well-conditioned convex problems.

In [3] the relaxed steepest descent method, as discussed in [58], is extended to general non-linear convex well-conditioned functions where the step length is calculated by means of backtracking. Linear convergence for this case is proved. The algorithm suggested by Andrei in [2] is compared with the relaxed SD algorithm for convex well-conditioned functions and it is shown numerically to have a superior rate of convergence.

## 1.6 Motivation of Thesis

To summarise this chapter, there exists a lot of literature on gradient algorithms but, amongst this literature, no set method for studying the rates of convergence of these algorithms is present. This has led to confusion when comparing the rates of different algorithms with many papers presenting differing opinions on which algorithm is superior depending on the testing conditions used. Many existing convergence rate studies have been limited to small dimensional cases as little can be proved for larger dimensions.

In this thesis a technique for studying rates of convergence is developed based on renormalisation, dynamical systems and optimal design theory. This technique enables an in depth analysis of the convergence rates of gradient algorithms. It allows improved comparisons between algorithms to be made and gives a better insight into how a successful algorithm achieves faster rates of convergence.



In Chapter 2 the renormalisation process is introduced and it is shown how, when gradient algorithms are in their renormalised form, they can be related to algorithms for constructing optimal experimental designs. A detailed review of the steepest descent algorithm and its rate of convergence is given. It is demonstrated that the renormalised steepest descent algorithm is related to an optimal design algorithm based on the D-optimum criterion.

In Chapter 3 the steepest descent algorithm is generalised to the  $\gamma$ -steepest descent algorithm through the addition of a fixed relaxation parameter,  $\gamma$ . Investigations into the effect the choice of the relaxation parameter  $\gamma$  has on the asymptotic rate of convergence are undertaken and the best value for  $\gamma$  for various parameters is determined. The algorithm is compared with the relaxed steepest descent algorithm suggested in [58].

The steepest descent algorithm is generalised in a different manner to produce a family of algorithms named the  $\beta$ -root family in Chapter 4 whose performance depends on the value of the parameter  $\beta$ . A particular member of this family, named the square root algorithm is examined and is shown to be exactly the algorithm suggested in [20] with step size  $\alpha_{OPT2}^{(k)} = \|g^{(k)}\|/\|Ag^{(k)}\|$ . The  $\beta$ -root family is further generalised by the addition of the relaxation coefficient  $\gamma$  to try to gain even faster convergence rates.

In Chapter 5 a family of well-known optimality criteria are used to inspire the creation of new gradient algorithms which otherwise would not intuitively be thought of. This family includes the well known A-optimality criterion. Their asymptotic rates of convergence are subsequently analysed and compared with the rate of the steepest descent algorithm. Finally in Chapter 6 the asymptotic rates of convergence of the algorithms studied in this thesis are compared both amongst themselves and with the asymptotic rates of convergence of the Barzilai-Borwein (1.15) and Cauchy-Barzilai-Borwein (1.17) algorithms. Those algorithms developed whose rate of convergence is better than that of the BB and CBB algorithm are deemed viable alternatives to existing gradient algorithms. Further work is suggested including the further generalisation of Forsythe's  $s$ -dimensional optimum gradient method, see [32]. Finally the main findings of the thesis are summarised.

## Chapter 2

# Gradient Algorithms and their Relation to Optimal Design

## Theory

The steepest descent algorithm is introduced in Section 1.2 and modifications of the algorithm are described in Section 1.5. In this chapter a review of the steepest descent algorithm and its asymptotic rate of convergence is given. In order to achieve this, the technique of renormalisation is first introduced which enables a link between gradient optimisation algorithms and algorithms for constructing optimal experimental designs to be established.

### 2.1 Renormalised Versions of Gradient Algorithms

#### 2.1.1 Renormalisation of the Steepest Descent Algorithm

A useful tool in the study of asymptotic rates of convergence of gradient algorithms, such as the steepest descent algorithm, is that of renormalisation. In many situations gradient algorithms converge to a local extremum very quickly making analysis of their asymptotic rate of convergence hard. Renormalising an algorithm's gradient facilitates a better analysis of the algorithm's asymptotic rate of convergence as the long term behaviour of the algorithm can be studied.

As was discussed in Section 1.2, the steepest descent method for the minimisation of quadratic functions of the form (1.3) can be written as an iterative formula in terms of the gradient  $g^{(k)}$  as follows

$$g^{(k+1)} = g^{(k)} - \frac{(g^{(k)}, g^{(k)})}{(Ag^{(k)}, g^{(k)})} Ag^{(k)}. \quad (2.1)$$

It can be assumed without loss of generality that

$$A = \begin{pmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \dots & \lambda_d \end{pmatrix}, \quad (2.2)$$

since the step lengths of gradient methods are invariant under any orthogonal transformation. The eigenvalues  $0 < m = \lambda_1 < \lambda_2 < \dots < \lambda_d = M$  can be assumed to be distinct as it is possible to join together the gradient components if there exist any multiple eigenvalues, see [29] for further explanation. In view of the fact that it is the asymptotic rate of convergence of the gradient algorithms that is being studied and the actual value of  $x^*$  that the algorithm converges to does not effect this rate, it is also possible to assume that  $x^* = 0$ , otherwise the substitution  $x - x^* \rightarrow x$  can be made.

Using the assumption that  $A$  is diagonal, the iterative formula (2.1) can be rewritten component-wise as

$$g_i^{(k+1)} = g_i^{(k)} - \frac{\sum_{j=1}^d (g_j^{(k)})^2}{\sum_{j=1}^d \lambda_j (g_j^{(k)})^2} \lambda_i g_i^{(k)} \quad \text{for } i = 1, \dots, d. \quad (2.3)$$

The step of renormalisation is now possible by setting

$$z^{(k)} = \frac{g^{(k)}}{\|g^{(k)}\|} = \frac{g^{(k)}}{(g^{(k)}, g^{(k)})^{1/2}}$$

(so that  $\|z^{(k)}\| = 1$  for all  $k = 1, 2, \dots$ ). Also set  $p_i^{(k)} = (z_i^{(k)})^2$  so that

$$p_i^{(k)} = \frac{(g_i^{(k)})^2}{\sum_j (g_j^{(k)})^2}.$$

Squaring both sides of (2.3) gives

$$(g_i^{(k+1)})^2 = \left(1 - \frac{\lambda_i \sum_{j=1}^d (g_j^{(k)})^2}{\sum_{j=1}^d \lambda_j (g_j^{(k)})^2}\right)^2 (g_i^{(k)})^2 \quad \text{for } i = 1, \dots, d, \quad (2.4)$$

which implies

$$p_i^{(k+1)} = \frac{(\sum_{j=1}^d \lambda_j p_j^{(k)} - \lambda_i)^2}{\sum_{l=1}^d (\sum_{j=1}^d \lambda_j p_j^{(k)} - \lambda_l)^2} p_i^{(k)} \quad \text{for } i = 1, \dots, d. \quad (2.5)$$

Note that  $\sum_{i=1}^d p_i^{(k)} = 1$  and  $p_i^{(k)} \geq 0$  so that  $p_i^{(k)}$  can be conceived as the weight associated with eigenvalue  $\lambda_i$  and (2.5) can be considered as an updating formula

$$P^{(k+1)} = \Psi(P^{(k)})$$

for discrete probability distributions supported at the points  $\{\lambda_1, \dots, \lambda_d\}$ . The moments of these probability distributions are given by

$$\mu_l = \mu_l(P^{(k)}) = \sum_{i=1}^d \lambda_i^l p_i.$$

The updating formula (2.5) for the weights  $p_i^{(k)}$  can thus be written in terms of the moments as follows:

$$p_i^{(k+1)} = \frac{(\mu_1 - \lambda_i)^2}{\mu_2 - \mu_1^2} p_i^{(k)} \quad \text{for } i = 1, \dots, d. \quad (2.6)$$

Dividing the numerator and denominator of (2.6) by  $\mu_1^2$  gives the alternative form

$$p_i^{(k+1)} = \frac{(1 - \lambda_i/\mu_1)^2}{\mu_2/\mu_1^2 - 1} p_i^{(k)} \quad \text{for } i = 1, \dots, d. \quad (2.7)$$

Since

$$\mu_l = \sum_{i=1}^d \lambda_i^l p_i = \frac{\sum_{i=1}^d \lambda_i^l p_i}{\sum_{i=1}^d p_i} = \frac{(A^l g, g)}{(g, g)}$$

the rate of convergence  $r_{SD}^{(k)}$  defined by (1.11) can also be written in terms of the moments  $\mu_\alpha$  as follows:

$$\begin{aligned} r_{SD}^{(k)} &= \frac{(A^2 g^{(k)}, g^{(k)})(g^{(k)}, g^{(k)})}{(A g^{(k)}, g^{(k)})^2} - 1 = \frac{(A^2 g^{(k)}, g^{(k)})}{(g^{(k)}, g^{(k)})} \left( \frac{(g^{(k)}, g^{(k)})}{(A g^{(k)}, g^{(k)})} \right)^2 - 1 \\ &= \frac{\mu_2}{\mu_1^2} - 1, \end{aligned} \quad (2.8)$$

which is exactly the form of the denominator of the updating formula (2.7).

Now the algorithm is in a renormalised form, a link can be made between it and multiplicative algorithms for constructing optimal experimental designs. This connection will be explained in Section 2.3.

### 2.1.2 Renormalisation of a General Gradient Algorithm

It is possible to renormalise any gradient algorithm of the form (1.5) using the same method as described above for the steepest descent algorithm. A general formula for renormalised gradient algorithms can thus be obtained. First note that, under the same assumptions as used for the steepest descent algorithm above (i.e. that  $A$  is diagonal with distinct eigenvalues  $0 < m = \lambda_1 < \dots < \lambda_d = M$  and  $x^* = 0$ ), the updating formula for the general gradient algorithm can be written component-wise as

$$g_i^{(k+1)} = g_i^{(k)} - \alpha^{(k)} \lambda_i g_i^{(k)} \quad \text{for } i = 1, \dots, d.$$

By renormalising the gradient  $g^{(k)}$  so that

$$p_i^{(k)} = \frac{(g_i^{(k)})^2}{\sum_j (g_j^{(k)})^2},$$

and by setting  $\alpha^{(k)} = \alpha$ , the updating formula for  $p_i^{(k)}$  can be written as

$$p_i^{(k+1)} = \frac{(1 - \alpha \lambda_i)^2}{1 - 2\alpha \mu_1 + \alpha^2 \mu_2} p_i^{(k)}. \quad (2.9)$$

If  $\alpha = (g^{(k)}, g^{(k)}) / (Ag^{(k)}, g^{(k)}) = 1/\mu_1$  the updating formula for the renormalised steepest descent algorithm (2.5) is obtained. The rate for the general gradient algorithm has the form

$$\begin{aligned} r^{(k)} &= \frac{(g^{(k+1)}, g^{(k+1)})}{(g^{(k)}, g^{(k)})} = \frac{(g^{(k)}, g^{(k)}) - 2\alpha (Ag^{(k)}, g^{(k)}) + \alpha (A^2 g^{(k)}, g^{(k)})}{(g^{(k)}, g^{(k)})} \\ &= 1 - 2\alpha \mu_1 + \alpha^2 \mu_2, \end{aligned}$$

which, as was the case with  $r_{SD}^{(k)}$ , is equal to the denominator of the corresponding updating formula (2.9) of the algorithm.

## 2.2 Asymptotic Behaviour of the Steepest Descent Algorithm

Analysis of the steepest descent algorithm using the method of renormalisation described above was first carried out by Akaike in [1]. In this paper the limiting

behaviour of repeatedly applying the transformation  $\Psi(P^{(k)})$  to a probability distribution is studied. The methods of this study are applied to the steepest descent algorithm and it is shown that the behaviour of the sequence of probability distributions  $\{p^{(k)}\}$  is oscillatory in nature. Furthermore it is shown that the sequences  $\{p^{(2k)}; k = 1, 2, \dots\}$  and  $\{p^{(2k+1)}; k = 0, 1, 2, \dots\}$  converge to some limiting distributions

$$p^{(\infty)} = \left( \frac{1}{1+c^2}, 0, \dots, 0, \frac{c^2}{1+c^2} \right) \quad \text{and} \quad p^{*(\infty)} = \left( \frac{c^2}{1+c^2}, 0, \dots, 0, \frac{1}{1+c^2} \right), \quad (2.10)$$

where  $c$  depends (in a complicated manner) on the starting vector  $x^{(0)}$ , the condition number  $\rho$  and the co-ordinate system defined by the eigenvectors  $\xi_1$  and  $\xi_d$  corresponding to the eigenvalues  $\lambda_1$  and  $\lambda_d$ . Note that these limiting distributions have their total probability attached only to the extremal points  $m$  and  $M$ . An example of this phenomenon is shown in Figure 2.1 where, as  $k$  increases, the middle weights, i.e.  $p_i^{(k)}$ ,  $i = 2, \dots, d-1$ , tend to zero while  $p_1^{(k)}$  and  $p_d^{(k)}$  reach their limiting 2-point cycle.

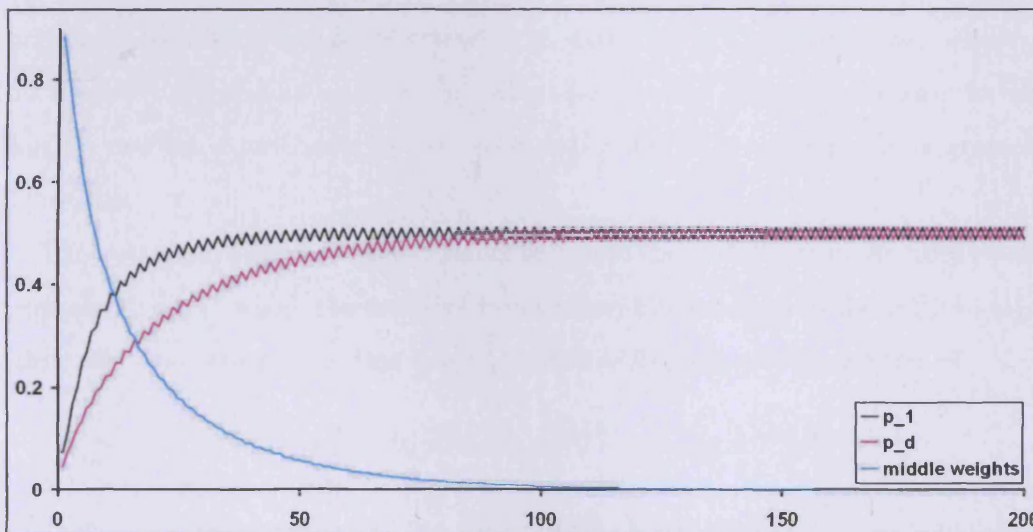


Figure 2.1: Weights  $p_1^{(k)}$ ,  $p_d^{(k)}$  and  $\sum_{i=2}^{d-1} p_i^{(k)}$  as a function of  $k$  ( $\rho = 10$ ,  $d = 100$ ,  $\lambda_i$  equally spaced).

Relating back to the steepest descent algorithm in non-normalised form the er-

ror,  $\epsilon^{(k)} = x^{(k)} - x^*$ , at iteration  $k$  alternates asymptotically between two fixed directions generated by the eigenvectors  $\xi_1$  and  $\xi_d$ , i.e. the steepest descent method is asymptotically reduced to a search in the plane generated by these eigenvectors.

The asymptotic rate of convergence of the steepest descent algorithm is shown in [1] to equal

$$R(c) = \frac{(M - m)^2}{(m + M)^2 + (c - c^{-1})^2 m M} = \frac{c^2(\rho - 1)^2}{(c^2 + \rho)(1 + c^2\rho)} \quad (2.11)$$

which is exactly the worst case rate  $R_{\text{ref}}$  defined by (1.13) when  $c^2 = 1$ . The corresponding limiting distribution for the worst case rate is

$$p^{(\infty)} = p^{*(\infty)} = \left( \frac{1}{2}, 0, \dots, 0, \frac{1}{2} \right)$$

i.e. in this case  $\{p^{(k)}\}$  will not exhibit oscillatory behaviour and instead will converge to a single point. It is stated that the asymptotic rate of convergence of the steepest descent algorithm will tend towards its worst rate when  $\rho$  is large especially if there exists an eigenvalue of  $A$  close to  $(M + m)/2$ .

The main result of [1], that the renormalised steepest descent algorithm (2.5) in  $\mathbb{R}^d$  converges to a two-point attractor, is extended in [54] to the case where  $A$  is a bounded self-adjoint operator in Hilbert space. The Hilbert space case for the steepest descent algorithm is further extended in [55] to a whole family of gradient algorithms.

The optimum  $s$ -gradient algorithm is an adaptation of the original steepest descent algorithm in which the iterative formula combines  $s$ -steps of the original algorithm into one iteration. In this way the iterative formula can be written as

$$x_s^{(k+1)} = x_s^{(k)} - \sigma_1 g^{(k)} - \sigma_2 A g^{(k)} - \dots - \sigma_s A^{s-1} g^{(k)},$$

where  $\sigma_1, \dots, \sigma_s$  are coefficients, the values of which are calculated so as to minimise the error function  $\epsilon^{(k)}$ . If  $s = 1$  there is only one coefficient and the formula collapses to that of the standard steepest descent algorithm. Further explanation of this algorithm is given in, for example, [27]. In [32] the asymptotic behaviour of the renormalised sequence  $\{z^{(k)}\}$  is studied for the optimum  $s$ -gradient method and the main results of [1] are extended to the case where  $s > 1$ . As  $s \rightarrow \infty$  the asymptotic

rate of convergence of this algorithm tends to

$$R_{\min} = \frac{(\sqrt{\rho} - 1)^2}{(\sqrt{\rho} + 1)^2}. \quad (2.12)$$

This convergence rate is a distinct improvement on the original case where  $s = 1$  and, as will be seen in the coming chapters, is hard to improve upon with any gradient algorithm. For this reason  $R_{\min}$  can, in some sense, be taken as a lower bound for  $R$  and new gradient algorithms developed should be done so with the aim of achieving an asymptotic rate of convergence as close to  $R_{\min}$  as possible.

The work of Akaike in [1] was extended independently by Nocedal et al. in [49] and Pronzato et al. in [53]. In [49] a greater insight into the value of  $c$  in (2.11) is given. It is demonstrated that  $c$  satisfies

$$c = \lim_{k \rightarrow \infty} \frac{z_d^{(2k)}}{z_1^{(2k)}} = - \lim_{k \rightarrow \infty} \frac{z_1^{(2k+1)}}{z_d^{(2k+1)}}$$

and some restrictions on what value  $c$  can take are established.

A necessary condition for a minimum, or indeed a maximum, of a function  $f(x)$  is that

$$\frac{\partial f}{\partial x_1} = \frac{\partial f}{\partial x_2} = \dots = \frac{\partial f}{\partial x_n} = 0,$$

thus the value of the gradient norm of any successful optimisation algorithm must converge to zero. Due to the oscillatory nature of the steepest descent algorithm, however, at some iterations the value of  $\|g^{(k)}\|$  will increase. By employing the Kantorovich inequality an upper bound for the growth of  $\|g^{(k)}\|$  can be given as

$$\frac{\|g^{(k+1)}\|^2}{\|g^{(k)}\|^2} \leq \frac{(\rho - 1)^2}{4\rho}$$

and in [49] this is proved to hold both asymptotically and at each iteration.

The occurrence of large oscillations in the gradient norm is attributable to the matrix  $A$  being ill-conditioned; however the fact that  $A$  is ill-conditioned will not necessarily imply large oscillations. The choice of starting vector  $x^{(0)}$  will also affect the magnitude of the oscillations in  $\|g^{(k)}\|$  in some way but it is not the case that a bad starting point will give rise to large oscillations. It is stated that the maximum possible oscillation in  $\|g^{(k)}\|$  will occur if  $c^2 = \rho$  or  $1/\rho$ .



It is also proven in this paper that

$$\lim_{k \rightarrow \infty} \frac{\|g^{(2k+1)}\|^2}{\|g^{(2k)}\|^2} = \frac{c^2(\rho-1)^2}{(1+c^2\rho)^2} \quad \text{and} \quad \lim_{k \rightarrow \infty} \frac{\|g^{(2k+2)}\|^2}{\|g^{(2k+1)}\|^2} = \frac{c^2(\rho-1)^2}{(c^2+\rho)^2}, \quad (2.13)$$

these limits being the same and equal to  $R_{\text{ref}}$  if  $c^2 = 1$ . In addition, it is proven that

$$\lim_{k \rightarrow \infty} \frac{f^{(k+1)}}{f^{(k)}} = \lim_{k \rightarrow \infty} \frac{\|g^{(k+2)}\|}{\|g^{(k)}\|},$$

i.e. the one-step asymptotic rate of convergence of the function value is equal to the two-step asymptotic rate of convergence of the gradient norm. Figure 2.2 shows an example of the rate  $r^{(k)}$  progressing to the limiting oscillatory behaviour given in (2.13) as  $k$  increases.

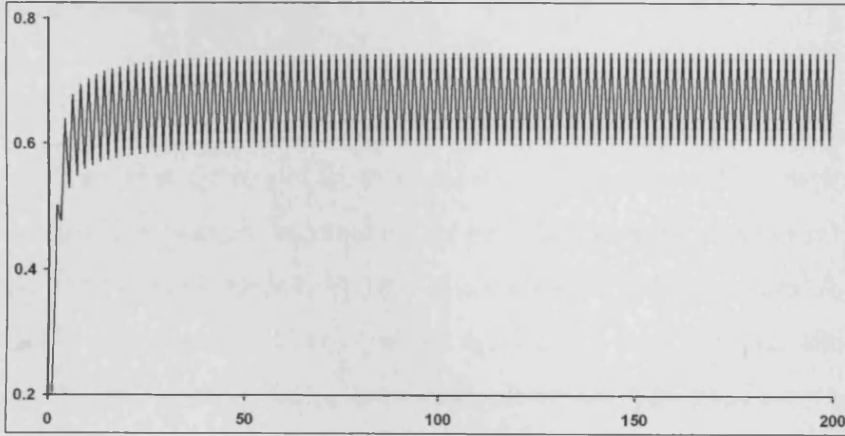


Figure 2.2: Rate  $r^{(k)}$  as a function of  $k$  ( $\rho = 10$ ,  $d = 100$ ,  $\lambda_i$  equally spaced).

The asymptotic behaviour of the steepest descent algorithm is also studied in detail in [53]. Here the 2-point limiting cycle of the renormalised sequence  $\{p^{(k)}\}$  is written as

$$p^{(\infty)} = \lim_{k \rightarrow \infty} p^{(2k)} = q\xi_1 + (1-q)\xi_d \quad \text{and} \quad p^{*(\infty)} = \lim_{k \rightarrow \infty} p^{(2k+1)} = (1-q)\xi_1 + q\xi_d, \quad (2.14)$$

where  $q \in (0, 1)$  is exactly equal to

$$q = \frac{1}{1+c^2}.$$

It is proved that, in order for the 2-point attractor to be stable, the value of  $q$  must lie in the range  $q \in \left[1/2 - b(\lambda_{i^*}), 1/2 + b(\lambda_{i^*})\right]$ , where

$$b(\lambda) = \frac{\sqrt{(M - \lambda)^2 + (m - \lambda)^2}}{2(M - m)}$$

and  $i^*$  is such that  $|\lambda_{i^*} - (m + M)/2|$  is minimum over all  $\lambda_i$ ,  $i = 2, \dots, d - 1$ . The shortest achievable interval occurs when  $\lambda_{i^*} = (m + M)/2$  for some  $i^* \in \{2, \dots, d - 1\}$  and is equal to  $[1/2 \pm \sqrt{2}/4]$ . A change of variables gives the asymptotic rate, defined by (2.11), in terms of  $q$  as

$$R(q) = \frac{q(1 - q)(\rho - 1)^2}{(q + \rho(1 - q))((1 - q) + \rho q)}$$

for almost all starting vectors  $x^{(0)}$ .

### Dependence on $\lambda_2, \dots, \lambda_{d-1}$

It is known that the worst rate of the steepest descent algorithm,  $R_{\text{ref}}$ , depends solely on the smallest and largest eigenvalues  $m$  and  $M$ ; however the actual observed asymptotic rate of convergence, (1.12), of an algorithm differs from  $R_{\text{ref}}$  in this respect as  $x^{(0)}$  and the middle eigenvalues  $\lambda_2, \dots, \lambda_{d-1}$  also have an effect on the rate of convergence. A three-dimensional example of this dependence in the steepest descent algorithm is shown in Figure 2.3 where the smallest and largest eigenvalues are fixed to be  $m = 1$  and  $M = 10$  and the middle eigenvalue  $\lambda_2$  is varied between these limits. The graph shows the average (with respect to  $x^{(0)}$ ) asymptotic rate of convergence  $R$  as a function of  $\lambda_2$ . It can clearly be seen that the asymptotic rate of convergence is best when  $\lambda_2$  is close to either  $m$  or  $M$  and worst when  $\lambda_2 = 11/2 = (m + M)/2$ , i.e. when  $\lambda_2$  is the midpoint of  $m$  and  $M$  so that the eigenvalues are equally spaced apart. A similar dependence can be seen in the four-dimensional case, see Figure 2.4. In the left hand graph  $\lambda_2$  and  $\lambda_3$  are varied between  $m = 1$  and  $M = 10$  symmetrically so that  $A = \text{diag}(1, X, 11 - X, 10)$ . The graph shows the asymptotic rate of convergence as a function of  $X$ . When  $X = 1 = m$  or  $X = 10 = M$  there are two repeated eigenvalues which has the effect of reducing the dimension to  $d = 2$  since the weights,  $\xi(\lambda_i)$ , associated with the repeated eigenvalues can be combined and the eigenvalues joined. When  $X = 11/2$  there is also a repeated

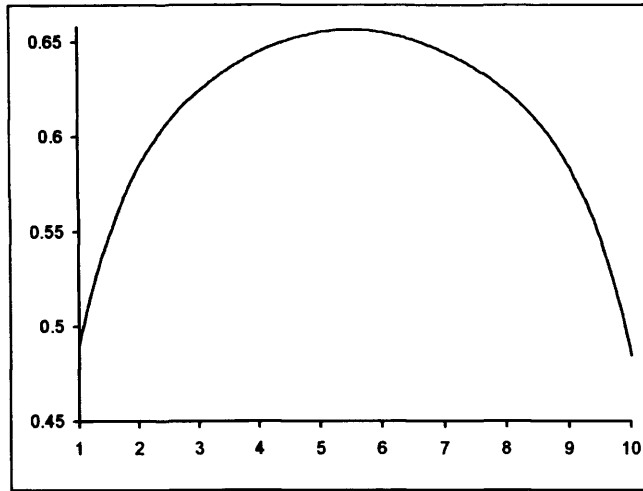


Figure 2.3: Average asymptotic rate of convergence as a function  $\lambda_2$  with  $d = 3$ ,  $\rho = 10$ .

eigenvalue thus reducing the dimension to  $d = 3$ . On the graph these three values of  $X$  represent points where the asymptotic rate of convergence is comparatively favourable indicating that, at least in small dimensions, the asymptotic rate of convergence worsens as the dimension of the problem grows. As  $X$  moves further and further away from a repeated eigenvalue the asymptotic rate worsens until  $X$  is such that a situation where the eigenvalues are evenly spread between  $m$  and  $M$  occurs; here the asymptotic rate is at its worst.

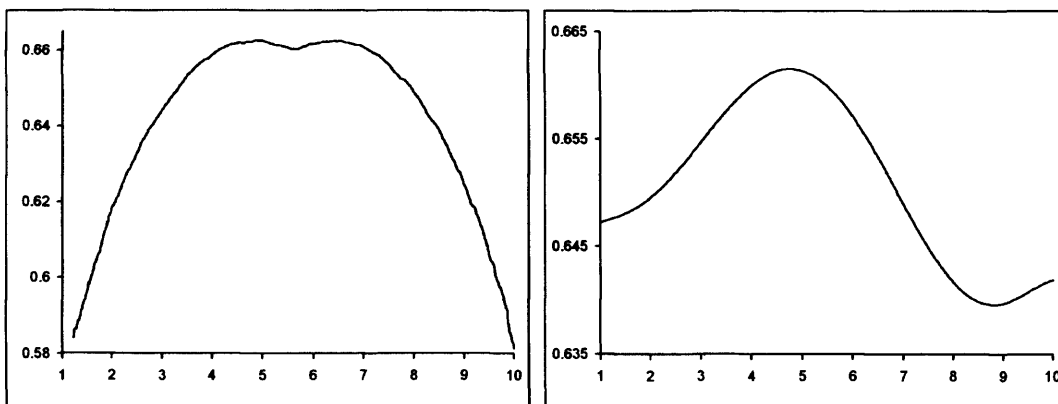


Figure 2.4: Average asymptotic rate of convergence as a function  $X$  where (left)  $A = \text{diag}(1, X, 11 - X, 10)$  and (right)  $A = \text{diag}(1, X, 7, 10)$ ;  $d = 4$ ,  $\rho = 10$ .

In larger dimensions, the difference in the average asymptotic rate of convergence

in situations where the eigenvalues of  $A$  are evenly spaced or not evenly spaced is less pronounced. Figure 2.5 shows the average asymptotic rate of convergence of the steepest descent algorithm as a function of  $\rho$  in the case where the eigenvalues are evenly spaced and the case where the eigenvalues are randomly chosen from a uniform distribution. The dimension was fixed as  $d = 50$ . The graph shows there to be virtually no difference between the two cases. This phenomenon is true of all algorithms studied as part of this thesis and for that reason all results given in this thesis from this point on are from simulations run with evenly spaced eigenvalues unless otherwise stated. A characteristic worthy of mention can be demonstrated

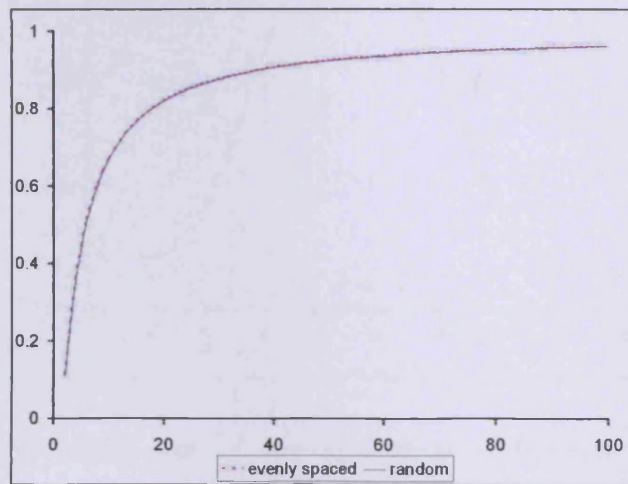


Figure 2.5: Average asymptotic rate of convergence as a function  $\rho$ ; for (a) evenly spaced eigenvalues and (b) randomly generated eigenvalues;  $d = 50$ .

in the five-dimensional case. By fixing four of the eigenvalues to be equally spaced between the smallest and largest eigenvalues and varying the fifth between  $m$  and  $M$  shows that the worst-case scenario here is when the varied eigenvalue is equal to  $\lambda = (m + M)/2$  (see Fig 2.6). It is true of all dimensions that if any eigenvalue of the matrix  $A$  is equal to  $\lambda_i = (m + M)/2$  then the asymptotic rate of convergence is worse than if this value is not contained within the eigenvalues of  $A$ . In the right hand graph of Figure 2.4 a four-dimensional example is shown where the second eigenvalue  $\lambda_2$  is varied between 1 and 10. The average asymptotic rate of convergence is worst when  $4 \lesssim \lambda_2 \lesssim 6$  i.e. when the eigenvalues are most evenly spread or

$\lambda_2 \cong (m + M)/2$ . For this reason, in odd dimensions, where the equal spacing of eigenvalues naturally includes the midpoint  $(m + M)/2$ , a slightly worse asymptotic rate of convergence is observed compared with even dimensions where equal spacing does not incorporate the midpoint. The difference between even and odd dimensions is, however, negligible in larger dimensions since, when  $d$  is even, for any  $\rho$ , the more eigenvalues between  $m$  and  $M$  there are, the closer to the midpoint the middle two eigenvalues in the range will be.

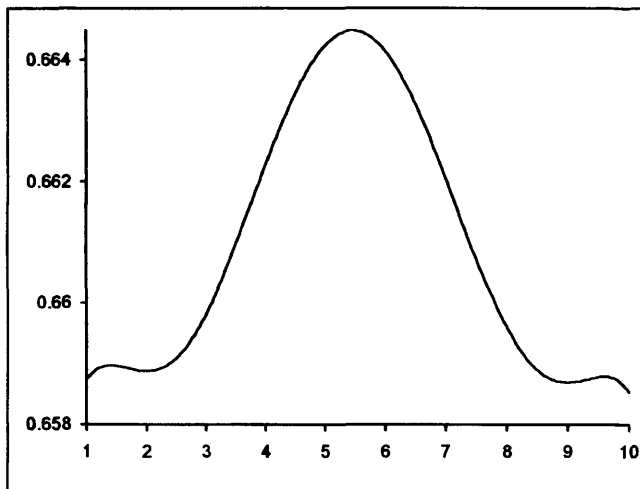


Figure 2.6: Asymptotic rate of convergence as a function  $X$  where  $A = \text{diag}(1, 4, X, 7, 10)$ ,  $d = 5$ ,  $\rho = 10$ .

### Dependence on $\rho$

Figure 2.7 shows the dependence of the asymptotic rate of convergence  $R$  of the steepest descent algorithm on the condition number  $\rho$  of the matrix  $A$  for a problem of dimension  $d = 100$ . As the condition number increases, the asymptotic rate of convergence worsens, a well known phenomenon; see for example [44]. It can also be seen from Figure 2.7 that the average rate of convergence,  $R$ , of the steepest descent algorithm is only slightly less if not equal to the worst case rate  $R_{\text{ref}}$ . This confirms the fact, observed in [1], that close to the worst rate is achieved for the majority of starting vectors  $x^{(0)}$  if the set  $\{\lambda_i\}$  contains an eigenvalue close to  $(m + M)/2$ . Figure 2.7 also compares the asymptotic rate of convergence,  $R$ , of the steepest descent

algorithm with that of the ideal asymptotic rate of convergence  $R_{\min}$  achieved by the optimum  $s$ -gradient algorithm with  $s = \infty$  and it can be concluded that there is a great deal of room for improvement upon the original steepest descent algorithm.

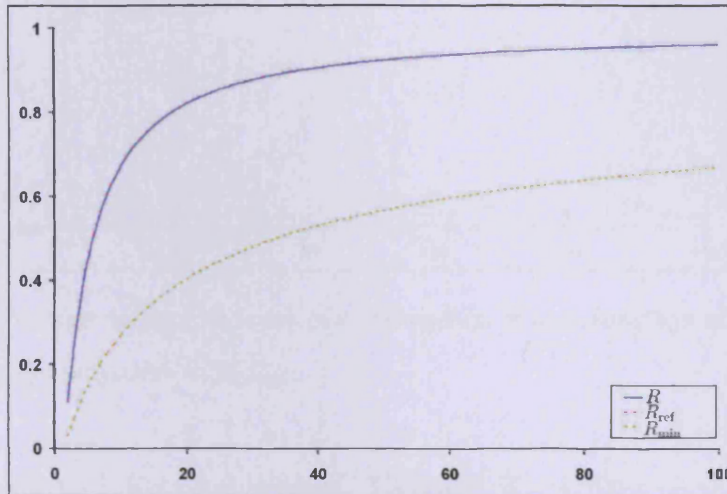


Figure 2.7: Average asymptotic rate of convergence  $R$  of the steepest descent algorithm (terminated after 1000 iterations) as a function of  $\rho$  ( $d = 100$ ,  $\lambda_i$  equally spaced, 300 repetitions) compared with  $R_{\text{ref}}$  and  $R_{\text{min}}$ .

### Dependence on $d$

While the asymptotic rate of convergence is shown to depend on the condition number of the matrix  $A$ , for  $d \gtrsim 10$ , the dimension of  $A$  has virtually no effect on the asymptotic rate of convergence,  $R$ . An example of this is shown in Figure 2.8. Simulations have shown that a dimension of  $d = 100$  is representative of the case  $d = \infty$  and will therefore be used to demonstrate this case throughout this thesis.

## 2.3 Relation to Optimal Experimental Design

It will emerge that gradient algorithms can be related to multiplicative algorithms for constructing optimal experimental designs. First a short overview of the required optimal experimental design notation is given.

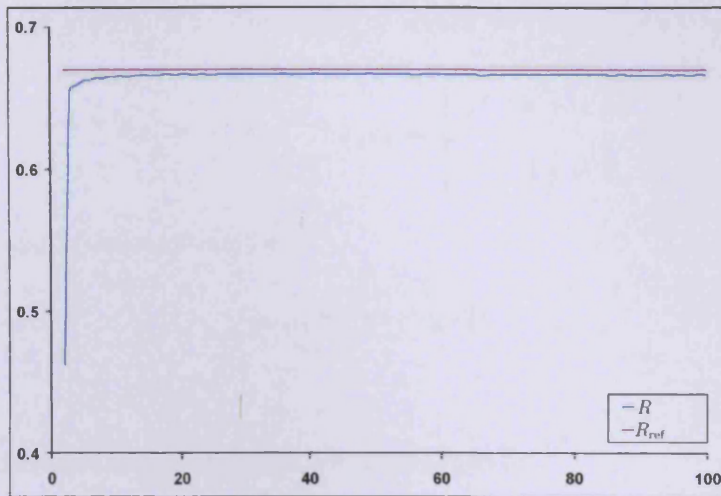


Figure 2.8: Average asymptotic rate of convergence  $R$  as a function of  $d$  ( $\rho = 10$ ,  $\lambda_i$  equally spaced) compared with  $R_{\text{ref}}$ .

### 2.3.1 Optimal Experimental Design Terminology

Consider the regression model

$$y_j = \sum_{i=1}^m \theta_i f_i(x_j) + \varepsilon_j, \quad j = 1, \dots, N. \quad (2.15)$$

The number of parameters in the model (2.15) is denoted by  $m$ ,  $N$  signifies the number of experiments performed and the vector of unknown parameters is represented by  $\theta = (\theta_1, \dots, \theta_m)^T$ . Denote by  $X$  the design matrix:

$$X = (f_i(x_j))_{j,i=1}^{N,m} = \begin{pmatrix} f_1(x_1) & \dots & f_m(x_1) \\ \vdots & \ddots & \vdots \\ f_1(x_N) & \dots & f_m(x_N) \end{pmatrix} = \begin{pmatrix} f^T(x_1) \\ \vdots \\ f^T(x_N) \end{pmatrix},$$

where  $f(x) = (f_1(x), \dots, f_m(x))^T$ . The matrix

$$M(\hat{\theta}) = X^T X = \sum_{j=1}^N f(x_j) f^T(x_j)$$

is called the information matrix and

$$D(\hat{\theta}) = \sigma^2 (X^T X)^{-1}$$

is the covariance matrix of  $\hat{\theta}$ . The vector of errors  $\varepsilon = (\varepsilon_1, \dots, \varepsilon_N)^T$  satisfies

$$E\varepsilon_j = 0 \quad \text{and} \quad E\varepsilon_j\varepsilon_k = \begin{cases} \sigma^2 & \text{if } j = k, \\ 0 & \text{if } j \neq k, \end{cases}$$

so that the least squares estimator

$$\hat{\theta} = (X^T X)^{-1} X^T Y$$

is valid.

An approximate weighted design can be defined as

$$\xi = \xi(x) = \begin{cases} x_1, & \dots, & x_d \\ p_1, & \dots, & p_d \end{cases}, \quad (2.16)$$

where  $p_1, \dots, p_d$  are the weights assigned to the  $d$  distinct design points  $x_1, \dots, x_d$  such that  $0 \leq p_i \leq 1$ ,  $i = 1, \dots, d$  and  $\sum_{i=1}^d p_i = 1$ . If two design points  $x_i, x_j$  are equal then their weights  $p_i$  and  $p_j$  can be summed and the points combined as one so that the presence of  $d$  distinct design points can be assumed. For the case where  $f(x) = (1, x, \dots, x^{K-1})^T$ , the corresponding information matrix for a weighted design of the form (2.16) is defined as

$$M(\xi) = \sum_{j=1}^d p_j f(x_j)^T f(x_j) = \{m_{ij} : m_{ij} = \mu_{i+j}; 0 \leq i, j \leq K-1\},$$

where  $\mu_\alpha = \mu_\alpha(\xi) = \sum_{j=1}^d x_j^\alpha p_j$  are the  $\alpha$ -th moments of the measure  $\xi$ . If  $K = 2$ , for example, the information matrix would be

$$M(\xi) = \begin{pmatrix} \mu_0 & \mu_1 \\ \mu_1 & \mu_2 \end{pmatrix}, \quad (2.17)$$

where  $\mu_0 = 1$ . The covariance per observation matrix of the approximate design (2.16) is equal to

$$D(\xi) = (M(\xi))^{-1}$$

if  $\sigma^2 = 1$ .

The purpose of experimental design in regression is to select the design points  $x_1, \dots, x_d$  and corresponding weights  $p_1, \dots, p_d$  so that the design  $\xi$  is optimal for



some criterion. Different optimality criteria exist but generally they are created with the aim of minimising some convex functional of the covariance matrix  $D(\xi)$  of the design or, equivalently, maximising some concave functional of the information matrix  $M(\xi)$ .

A useful tool in optimum experimental design theory is the directional derivative (also known as the Fréchet derivative) ‘towards’ a discrete measure  $\xi_x$  of mass 1 at a point  $x$ . This is

$$\left. \frac{\partial}{\partial a} \Phi \left( M[(1-a)\xi + a\xi_x] \right) \right|_{a=0} = \text{tr} \left( \overset{\circ}{\Phi}(\xi) M(\xi_x) \right) - \text{tr} \left( \overset{\circ}{\Phi}(\xi) M(\xi) \right), \quad (2.18)$$

where

$$\overset{\circ}{\Phi}(\xi) = \left. \frac{\partial \Phi}{\partial M} \right|_{M=M(\xi)}.$$

Here  $\Phi$  is a functional of the  $K \times K$  information matrix  $M(\xi)$  and is usually considered as the optimality criterion to be maximized with respect to  $\xi$ . The first term on the right hand side of (2.18) is equal to

$$\varphi(x, \xi) = f^T(x) \overset{\circ}{\Phi}(\xi) f(x). \quad (2.19)$$

Perhaps the most well-known of the optimality criteria is  $D$ -optimality. Here the criterion is defined to be

$$\Phi(\xi) = \det M(\xi)$$

and a design  $\xi^*$  is called  $D$ -optimum if

$$\det M(\xi^*) = \max_{\xi} \det M(\xi).$$

The determinant of a matrix is a natural characteristic of how big that matrix is, so if the determinant of  $M(\xi)$  is large then it follows that the inverse of  $M(\xi)$ , which is proportional to the covariance matrix  $D(\hat{\theta})$ , will be small. A  $D$ -optimum design is hence considered desirable. More technically, a  $D$ -optimum design will provide minimum volume confidence regions for  $\theta$ . (See [62] for a review on  $D$ -optimality for regression designs and [26] for a review of optimal experimental design in general).

### 2.3.2 Constructing Gradient Algorithms which Correspond to Given Optimality Criteria

It will be seen that gradient algorithms belonging to the family (1.5) can be constructed, albeit indirectly, from optimality criteria, such as the  $D$ -optimality criterion. In order to accomplish this, a connection between multiplicative optimal design algorithms and renormalised gradient algorithms of the form (2.9) is first described.

A family of optimal design algorithms is based on the multiplicative updating of the weights of the current design measure  $\xi^{(k)}$  with some function of  $\varphi(x, \xi)$ . With the intention of formulating a particular member of this family the following assumptions are made: the design measure  $\xi$  is discrete and concentrated on the interval  $[m, M]$  and the  $(K, K)$ th-element of the matrix  $\partial\Phi(M)/\partial M$  is positive, i.e.  $\partial\Phi(M)/\partial\mu_{2K-2} > 0$ . Under these assumptions the function  $\varphi(x, \xi)$  has a well-defined minimum

$$c(\xi) = \min_{x \in \mathbb{R}} \varphi(x, \xi) > -\infty.$$

Let  $\xi(x)$  be the mass at a point  $x$ , the re-weighting at  $x$  can then be defined by

$$\xi'(x) = \frac{\varphi(x, \xi) - c(\xi)}{b(\xi)} \xi(x), \quad (2.20)$$

where  $b(\xi)$  is a normalising constant equal to

$$\begin{aligned} b(\xi) &= \int_m^M (\varphi(x, \xi) - c(\xi)) \xi(dx) = \int_m^M \varphi(x, \xi) \xi(dx) - c(\xi) \\ &= \text{tr} [M(\xi) \overset{\circ}{\Phi}(\xi)] - c(\xi). \end{aligned} \quad (2.21)$$

The normalisation constant is required to ensure that the measure  $\xi'$  is still a probability distribution. The first term on the left hand side of (2.21) is (ignoring the difference in sign) the second term in the directional derivative (2.18).

The algorithm (2.20), considered as an algorithm for constructing  $\Phi$ -optimal designs, is a particular case of the family of algorithms considered in, for example, [45, 46, 68]. In the optimal design arena there is interest in optimising a design criterion with respect to the weights assigned to the support points  $\lambda_1, \dots, \lambda_d$ . When using a multiplicative algorithm of this kind, at each iteration the current weights,

$\xi_i^{(k)}$ , are multiplied by a factor (in the form of a function  $f(d_i^{(k)})$  of the current partial derivatives) and are then scaled to sum to unity. In [63], Titterton proved monotonicity for the case where  $f(d) = d$  for  $D$ -optimality and in [64], Torsney proved monotonicity for the case where  $f(d) = d^{1/2}$  for  $A$ -optimality. In fact, this proof extended a result given in [28] for  $c$ -optimality, but the focus in [28] was not on algorithms. An empirical exploration of  $f(d) = d^\delta$ ,  $\delta > 0$ , for  $D$ -optimality is given in [61]. Other choices of  $f(d)$  are needed if it is possible that the criterion function has negative derivatives (as is the case in some maximum likelihood estimation problems, or if partial derivatives are replaced by vertex directional derivatives); see [65, 67, 66].

To relate algorithms of the kind (2.20) to gradient algorithms such as the steepest descent algorithm it is necessary to consider the case where  $K = 2$ . For this case,

$$f(x) = \begin{pmatrix} 1 \\ x \end{pmatrix}, \quad \overset{\circ}{\Phi}(\xi) = \begin{pmatrix} \frac{\partial \Phi}{\partial \mu_0} & \frac{1}{2} \frac{\partial \Phi}{\partial \mu_1} \\ \frac{1}{2} \frac{\partial \Phi}{\partial \mu_1} & \frac{\partial \Phi}{\partial \mu_2} \end{pmatrix}$$

and the function  $\varphi(x, \xi)$ , quadratic in  $x$ , is equal to

$$\varphi(x, \xi) = \begin{pmatrix} 1 & x \end{pmatrix} \begin{pmatrix} \frac{\partial \Phi}{\partial \mu_0} & \frac{1}{2} \frac{\partial \Phi}{\partial \mu_1} \\ \frac{1}{2} \frac{\partial \Phi}{\partial \mu_1} & \frac{\partial \Phi}{\partial \mu_2} \end{pmatrix} \begin{pmatrix} 1 \\ x \end{pmatrix} = \frac{\partial \Phi}{\partial \mu_0} + x \frac{\partial \Phi}{\partial \mu_1} + x^2 \frac{\partial \Phi}{\partial \mu_2}. \quad (2.22)$$

It then follows that

$$c(\xi) = \frac{\partial \Phi}{\partial \mu_0} - B(\xi),$$

where

$$B(\xi) = \frac{1}{4} \left( \frac{\partial \Phi}{\partial \mu_1} \right)^2 / \left( \frac{\partial \Phi}{\partial \mu_2} \right),$$

and the numerator on the right-hand side of (2.20) is

$$\varphi(x, \xi) - c(\xi) = \frac{\partial \Phi}{\partial \mu_2} \left( x + \frac{1}{2} \frac{\partial \Phi / \partial \mu_1}{\partial \Phi / \partial \mu_2} \right)^2 = B(\xi) \left( 1 + 2 \frac{\partial \Phi / \partial \mu_1}{\partial \Phi / \partial \mu_2} x \right)^2. \quad (2.23)$$

Define  $\alpha = \alpha(\xi) = \alpha(\mu_1, \mu_2)$  as

$$\alpha = \alpha(\xi) = -2 \frac{\partial \Phi / \partial \mu_1}{\partial \Phi / \partial \mu_2}. \quad (2.24)$$

The numerator (2.23) can then be written as

$$\varphi(x, \xi) - c(\xi) = B(\xi) (1 - \alpha(\xi)x)^2 \quad (2.25)$$

and, through algebraic manipulation, the re-weighting formula (2.20) can be written in terms of  $\alpha$  as

$$\xi'(x) = \frac{(1 - \alpha x)^2}{1 - 2\alpha\mu_1 + \alpha^2\mu_2} \xi(x). \quad (2.26)$$

This is exactly the same form as the renormalised general gradient algorithm (2.9).

This can be seen by re-writing the updating formula (2.26) iteratively as

$$\xi^{(k+1)}(x) = \frac{(1 - \alpha^{(k)}x)^2}{1 - 2\alpha^{(k)}\mu_1^{(k)} + (\alpha^{(k)})^2\mu_2^{(k)}} \xi^{(k)}(x). \quad (2.27)$$

and setting  $\xi^{(k)} = P^{(k)}$  and  $x_i = \lambda_i$  for  $i = 1, \dots, d$ .

Consider the  $D$ -optimality criterion  $\Phi(\xi) = \det M(\xi)$ , which for the case where  $K = 2$  is equal to

$$\Phi(\xi) = \det \begin{pmatrix} \mu_0 & \mu_1 \\ \mu_1 & \mu_2 \end{pmatrix} = \mu_0\mu_2 - \mu_1^2. \quad (2.28)$$

For this criterion

$$\overset{\circ}{\Phi}(\xi) = \begin{pmatrix} \mu_2 & -\mu_1 \\ -\mu_1 & \mu_0 \end{pmatrix}, \quad \varphi(x, \xi) = \mu_2 - 2\mu_1x + \mu_0x^2, \quad \text{and} \quad c = b = \mu_2 - \mu_1^2.$$

Consequently, the corresponding multiplicative optimal design algorithm will have the form

$$\xi'(x) = \frac{(1 - x/\mu_1)^2}{\mu_2/\mu_1^2 - 1} \xi(x)$$

which when rewritten iteratively with the change of variables  $\xi^{(k)} = P^{(k)}$  and  $x_i = \lambda_i$  for  $i = 1, \dots, d$ , will yield the steepest descent algorithm in renormalised form (2.5).

Using this link between multiplicative optimal design algorithms and renormalised gradient optimisation algorithms, any optimality criterion  $\Phi(\xi)$  can be used to generate a corresponding gradient algorithm. This opens up the possibility of new gradient algorithms whose step lengths  $\alpha^{(k)}$ , created from optimality criteria, would not necessarily be intuitively thought of otherwise.

### 2.3.3 Rate of Convergence of Gradient Algorithms Corresponding to Optimal Designs

A design  $\xi^*$  is said to be optimum for a given criterion  $\Phi$  on  $[m, M]$  if

$$\Phi(M(\xi^*)) = \max_{\xi} \Phi(M(\xi)),$$

where the maximum is taken over all probability measures supported on  $[m, M]$ . Consider the case where  $\Phi = \Phi(M(\xi))$  is an optimality criterion, with  $M(\xi)$  as defined in (2.17) and associate with it a gradient algorithm with step-length  $\alpha(\mu_1, \mu_2)$  as given by (2.24). The rate of the gradient algorithm corresponding to a particular design  $\xi$  is given by

$$r(\xi) = 1 - 2\alpha\mu_1 + \alpha^2\mu_2 = \frac{b(\xi)}{B(\xi)}.$$

If a design is optimum then it is invariant under one iteration of the updating formula (2.27) i.e. if  $\xi = \xi^*$  then  $\xi'(x_i) = \xi(x_i)$  for  $i = 1 \dots, d$ . Assume that  $\xi^*$  is non-degenerate, i.e. it is supported at at least two points, then since  $\xi^*$  is optimum it follows that  $\Phi(M(\xi))$  is at its maximum and thus all directional derivatives

$$\frac{\partial}{\partial a} \Phi \left[ M((1-a)\xi^* + a\xi(x)) \right] \Big|_{a=0^+} \leq 0,$$

i.e. are non-positive for all  $x \in [m, M]$ . Since

$$\frac{\partial}{\partial \alpha} \Phi \left( M[(1-a)\xi^* + a\xi_x] \right) \Big|_{\alpha=0} = \varphi(x, \xi^*) - \text{tr} \left( \overset{\circ}{\Phi}(\xi^*) M(\xi^*) \right),$$

see (2.18), the following inequality is implied

$$\max_{x \in [m, M]} \varphi(x, \xi^*) \leq t^* = \text{tr} \left[ M(\xi^*) \overset{\circ}{\Phi}(\xi^*) \right].$$

Equivalently the following inequalities hold;  $\varphi(m, \xi^*) \leq t^*$ ,  $\varphi(M, \xi^*) \leq t^*$ , since  $\varphi(x, \xi^*)$  is quadratic and convex with respect to  $x$ . The fact that

$$\int_m^M \varphi(x, \xi^*) \xi^*(dx) = t^*$$

implies that  $\xi^*$  is supported at the extreme points  $m$  and  $M$ , thus  $\xi^*$  must have positive masses at  $m$  and  $M$ , as it is assumed that  $\xi^*$  is non-degenerate, and

$$\varphi(m, \xi^*) = \varphi(M, \xi^*) = t^*.$$

As  $\varphi(x, \xi^*)$  is quadratic in  $x$  with its minimum at  $1/\alpha$ , see (2.25), it implies that

$$\alpha^* = \alpha(\mu_1(\xi^*), \mu_2(\xi^*)) = \frac{2}{m+M}.$$

The rate  $r(\xi^*)$ , associated with an optimum design is therefore

$$r(\xi^*) = \frac{b(\xi^*)}{B(\xi^*)} = \frac{t^* - c(\xi^*)}{B(\xi^*)} = (1 - m\alpha^*)^2 = (1 - M\alpha^*)^2 = R_{\text{ref}},$$

If the optimum design  $\xi^*$  is degenerate, i.e. it is supported at a single point  $x^*$  then  $x^*$  must be either  $m$  or  $M$  since the function  $\varphi(x, \xi^*)$  is quadratic and convex. The optimum design  $\xi^*$  is invariant under one iteration of the updating formula (2.27) therefore  $\alpha^*$  is constant and

$$\max_{\xi} r(\xi) = \max [(1 - m\alpha^*)^2, (1 - M\alpha^*)^2] \geq R_{\text{ref}},$$

with  $\max_{\xi} r(\xi)$  only equalling  $R_{\text{ref}}$  if  $\alpha^* = 2/(m + M)$ .

In conclusion, any gradient algorithm corresponding to an optimum design  $\xi^*$  will possess the worst case rate of convergence of the steepest descent algorithm. A multiplicative optimal design algorithm of the form (2.26) which converges to an optimum design does not, therefore, give rise to a gradient optimisation algorithm with an asymptotic rate of convergence any better than that of the classical steepest descent algorithm.

## Chapter 3

# The $\gamma$ -Steepest Descent Algorithm

The standard steepest descent algorithm is known to have a poor rate of convergence. Any modification of the steepest descent algorithm which might lead to an improvement in the asymptotic rate of convergence of the algorithm would therefore be viewed as beneficial. It will be seen in this section that the introduction of a relaxation coefficient,  $\gamma$ , to the standard steepest descent algorithm will completely change the behaviour of the algorithm and, for certain values of  $\gamma$ , will dramatically improve upon the rate of convergence of the original algorithm.

Let the modified steepest descent algorithm be defined as follows:

$$x^{(k+1)} = x^{(k)} - \gamma \frac{(g^{(k)}, g^{(k)})}{(Ag^{(k)}, g^{(k)})} g^{(k)} .$$

This algorithm will be known as the  $\gamma$ -steepest descent algorithm hereafter. The algorithm was discussed in Chapter 7 of [53], however analysis there was restricted to the 2-dimensional case. The difference between the  $\gamma$ -steepest descent algorithm and the original steepest descent algorithm (1.7) is that in the modified algorithm the usual steepest descent step length  $\alpha_{SD}^{(k)}$  is multiplied by a relaxation coefficient  $\gamma$ . If  $\gamma = 1$  the step length of the original algorithm is thus regained. It is well known that, for ill-conditioned problems, the step length for the steepest descent algorithm is slightly too long (see discussion in [9];) thus intuitively it is possible that values of  $\gamma < 1$  may lead to improved rates of convergence.

### 3.1 The Renormalised $\gamma$ -Steepest Descent Algorithm

To complete analysis of the  $\gamma$ -steepest descent ( $\gamma$ SD) algorithm, the algorithm is first renormalised. The step length of the  $\gamma$ SD algorithm can be rewritten in terms of the moments as

$$\alpha^{(k)} = \gamma \frac{(g^{(k)}, g^{(k)})}{(Ag^{(k)}, g^{(k)})} = \frac{\gamma}{\mu_1}.$$

Substituting this value of  $\alpha$  into the general equation for renormalised gradient algorithms (2.9) gives the updating formula

$$\xi_i^{(k+1)} = \frac{(1 - \gamma\lambda_i/\mu_1)^2}{1 - 2\gamma + \gamma^2\mu_2/\mu_1^2} \xi_i^{(k)} \quad \text{for } i = 1, \dots, d. \quad (3.1)$$

The rate at iteration  $k$  of the  $\gamma$ -steepest descent algorithm is equal to

$$r_{\gamma SD}^{(k)} = 1 - 2\gamma + \gamma^2\mu_2/\mu_1^2. \quad (3.2)$$

As with the renormalised steepest descent algorithm, the renormalised  $\gamma$ -steepest descent algorithm can also be viewed as a multiplicative algorithm for constructing optimal designs. The optimality criterion for this algorithm is

$$\Phi(\xi) = \gamma\mu_0\mu_2 - \mu_1^2 = \gamma\mu_2 - \mu_1^2, \quad (3.3)$$

which can be seen as a generalisation of the  $D$ -optimality criterion where, instead of taking the determinant of the information matrix (2.17) as the criterion, the determinant

$$\Phi(\xi) = \det \left[ M(\xi) + \begin{pmatrix} \tau & 0 \\ 0 & 0 \end{pmatrix} \right],$$

with  $\tau = \gamma - 1$ , is used in its place. For this criterion

$$\overset{\circ}{\Phi}(\xi) = \begin{pmatrix} \gamma\mu_2 & -\mu_1 \\ -\mu_1 & \gamma\mu_0 \end{pmatrix}, \quad \varphi(x, \xi) = \gamma\mu_2 - 2\mu_1x + \gamma x^2,$$

$$c = \gamma\mu_2 - \frac{\mu_1^2}{\gamma}, \quad \text{and} \quad b = \gamma^2\mu_1 + \mu_1^2 \left( \frac{1}{\gamma} - 2 \right).$$



### 3.1.1 Concavity of the Optimality Criterion

For the purpose of the maximisation of an optimality criterion in experimental design, it is often important that the optimality criterion  $\Phi(\xi)$  is concave, see Def. 1.1.3. In other words, it is necessary that

$$\Phi((1-a)\xi + a\xi') \geq (1-a)\Phi(\xi) + a\Phi(\xi'),$$

for  $0 \leq a \leq 1$  and any designs  $\xi, \xi'$ , where

$$\Phi(\xi) = \Phi(\mu_1(\xi), \mu_2(\xi)) = \Phi(M(\xi))$$

and

$$\begin{aligned} \mu_1((1-a)\xi + a\xi') &= (1-a)\mu_1(\xi) + a\mu_1(\xi'), \\ \mu_2((1-a)\xi + a\xi') &= (1-a)\mu_2(\xi) + a\mu_2(\xi'). \end{aligned}$$

A necessary and sufficient condition for the concavity of a functional is that the second derivative is non-positive i.e.

$$\frac{\partial^2}{\partial a^2} \Phi(\xi_a) \leq 0 \quad \forall \xi_a = (1-a)\xi + a\xi'$$

(indicating that the turning point is a maximum). For a general criterion  $\Phi$ , depending on the first two moments of the design, the second derivative is

$$\begin{aligned} \frac{\partial^2}{\partial a^2} \Phi(\xi_a) &= \frac{\partial}{\partial a} \left[ \frac{\partial}{\partial a} \Phi(\mu_1(\xi_a), \mu_2(\xi_a)) \right] \\ &= \frac{\partial}{\partial a} \left[ \frac{\partial \Phi(\mu_1, \mu_2)}{\partial \mu_1} \left( \frac{\partial \mu_1(\xi_a)}{\partial a} \right) + \frac{\partial \Phi(\mu_1, \mu_2)}{\partial \mu_2} \left( \frac{\partial \mu_2(\xi_a)}{\partial a} \right) \right] \\ &= \frac{\partial}{\partial a} \left[ \frac{\partial \Phi(\mu_1, \mu_2)}{\partial \mu_1} (\mu_1(\xi') - \mu_1(\xi)) + \frac{\partial \Phi(\mu_1, \mu_2)}{\partial \mu_2} (\mu_2(\xi') - \mu_2(\xi)) \right], \end{aligned} \quad (3.4)$$

note  $\mu_1 = \mu_1(\xi_a)$  and  $\mu_2 = \mu_2(\xi_a)$ .

The following theorem shows that the optimality criterion (3.3) is a concave function.

**Theorem 3.1.1.** *The functional,  $\Phi(\xi) = \gamma\mu_2 - \mu_1^2$ , is a concave functional for all  $\gamma$ .*

*Proof.* The first derivatives of  $\Phi = \gamma\mu_2 - \mu_1^2$  are

$$\frac{\partial \Phi}{\partial \mu_1} = -2\mu_1(\xi_a), \quad \frac{\partial \Phi}{\partial \mu_2} = \gamma.$$

Substituting these values into the general formula for the second derivative (3.4) gives

$$\begin{aligned}\frac{\partial^2}{\partial a^2}\Phi(\xi_a) &= \frac{\partial}{\partial a} \left[ -2\mu_1(\xi_a)(\mu_1(\xi') - \mu_1(\xi)) + \gamma(\mu_2(\xi') - \mu_2(\xi)) \right] \\ &= -2(\mu_1(\xi') - \mu_1(\xi))^2 \leq 0.\end{aligned}$$

The necessary and sufficient condition that the second derivative  $\partial^2\Phi(\xi)/\partial a^2$  is negative has been met and does not depend on the value of  $\gamma$  thus  $\Phi(\xi) = \gamma\mu_2 - \mu_1^2$  is a concave functional for all  $\gamma$ .  $\square$

### 3.1.2 Convergence to an Optimal Design

If  $2m/(m+M) < \gamma < 4Mm/(m+M)^2$  or  $1 < \gamma < 2M/(m+M)$  the renormalised  $\gamma$ -steepest descent algorithm (3.1) will always converge to an optimum design  $\xi^*$  and hence, for these ranges of  $\gamma$ , the asymptotic rate of convergence of the algorithm is equal to the worst possible rate of the steepest descent algorithm,  $R_{\text{ref}}$ . In order to prove this it is first shown that the optimality criterion (3.3) is a monotonically increasing sequence which converges to a limiting point  $\Phi^*$ .

**Theorem 3.1.2.** *Let  $\xi^{(0)}$  be any non-degenerate probability measure with support  $\{\lambda_1, \dots, \lambda_d\}$  and let the sequence of probability measures  $\{\xi^{(k)}\}$  be defined via the updating formula (3.1). The sequence*

$$\Phi^{(k)} = \Phi(\xi^{(k)}) = \gamma\mu_2(\xi^{(k)}) - \mu_1^2(\xi^{(k)})$$

*then monotonically increases and converges for  $\gamma < 4mM/(m+M)^2$  and  $\gamma > 1$ ; that is  $\Phi^{(0)} \leq \Phi^{(1)} \leq \dots \leq \Phi^{(k)} \leq \dots$  and the limit  $\Phi^*(\xi^{(0)}) = \lim_{k \rightarrow \infty} \Phi^{(k)}$  exists.*

*Proof.* Note that  $\Phi^{(k)} \geq 0$  for any  $\xi$  in view of the Cauchy-Schwartz inequality.

The sequence  $\Phi^{(k)}$  is non-decreasing if  $\Phi^{(k+1)} - \Phi^{(k)} \geq 0$  holds for any distribution  $\xi = \xi^{(k)}$ . If the distribution  $\xi^{(k)}$  is degenerate (that is, has mass 1 at one point) then  $\Phi^{(k+1)} = \Phi^{(k)}$  and the statement of the theorem holds. It is assumed below that  $\xi$  is non-degenerate. In particular this implies  $\mu_2 > \mu_1^2$ . The problem can be formulated as:

$$\Phi^{(k+1)} - \Phi^{(k)} \geq 0 \iff \left( \gamma\mu_2' - (\mu_1')^2 \right) - (\gamma\mu_2 - \mu_1^2) \geq 0, \quad (3.5)$$

where  $\mu'_1$  and  $\mu'_2$  are respectively the first and second moments associated with the distribution  $\xi^{(k+1)}$  respectively. It is possible to express the moments of  $\xi^{(k+1)}$  through the moments of  $\xi = \xi^{(k)}$ . First it is verified that  $\mu'_0 = \sum \xi_i^{(k+1)} = 1$ . Indeed,

$$\begin{aligned}\mu'_0 &= \sum_{i=1}^d \xi_i^{(k+1)} = \frac{1}{1 - 2\gamma + \gamma^2 \mu_2 / \mu_1^2} \sum_{i=1}^d (1 - \gamma \lambda_i / \mu_1)^2 \xi_i \\ &= \frac{1}{1 - 2\gamma + \gamma^2 \mu_2 / \mu_1^2} \left[ \sum_{i=1}^d \xi_i + 2 \frac{\gamma}{\mu_1} \sum_{i=1}^d \lambda_i \xi_i + \frac{\gamma^2}{\mu_1^2} \sum_{i=1}^d \lambda_i^2 \xi_i \right] \\ &= \frac{1}{1 - 2\gamma + \gamma^2 \mu_2 / \mu_1^2} \left[ 1 - 2\gamma + \gamma^2 \frac{\mu_2}{\mu_1^2} \right] = 1.\end{aligned}$$

In a similar way

$$\mu'_1 = \sum_{i=1}^d \lambda_i \xi_i^{(k+1)} = \frac{1}{1 - 2\gamma + \gamma^2 \mu_2 / \mu_1^2} \left[ \mu_1 - 2\gamma \frac{\mu_2}{\mu_1} + \gamma^2 \frac{\mu_3}{\mu_1^2} \right]$$

and

$$\mu'_2 = \sum_{i=1}^d \lambda_i^2 \xi_i^{(k+1)} = \frac{1}{1 - 2\gamma + \gamma^2 \mu_2 / \mu_1^2} \left[ \mu_2 - 2\gamma \frac{\mu_3}{\mu_1} + \gamma^2 \frac{\mu_4}{\mu_1^2} \right].$$

It is possible to represent the left hand side of (3.5) as

$$\left( \gamma \mu'_2 - (\mu'_1)^2 \right) - \left( \gamma \mu_2 - \mu_1^2 \right) = \gamma \frac{U}{W}$$

with,

$$\begin{aligned}U &= -\mu_1^2 \mu_2^2 \gamma^2 + 4 \gamma^2 \mu_1 \mu_3 \mu_2 - 2 \gamma^3 \mu_1 \mu_3 \mu_2 - 4 \mu_1^2 \mu_2^2 \gamma + 4 \mu_1^4 \mu_2 - 4 \gamma \mu_1^3 \mu_3 \\ &+ \gamma^2 \mu_4 \mu_1^2 + 4 \mu_1^4 \mu_2 \gamma + 4 \gamma^2 \mu_1^3 \mu_3 + \gamma^4 \mu_4 \mu_2 - 2 \gamma^3 \mu_4 \mu_1^2 + 5 \mu_1^2 \mu_2^2 \gamma^3 \\ &- 8 \mu_1^4 \mu_2 \gamma^2 - \gamma^3 \mu_3^2 - 4 \mu_1^6 + 4 \mu_1^6 \gamma - \gamma^4 \mu_2^3\end{aligned}$$

and

$$W = \left( -\gamma^2 \mu_2 - \mu_1^2 + 2 \gamma \mu_1^2 \right)^2.$$

It is clear  $W$  will always remain positive thus the problem is reduced to determining whether or not the numerator,  $U$ , is non-negative. One way of proving that an expression is non-negative is to relate it to a variance of random variable, as it is known that variances are always non-negative. Consider the variance

$$V = \text{var}(a\eta + b\eta^2)$$

where  $\eta$  is a random variable with distribution  $\xi$  and  $a, b$  are some parameters that can be chosen.

$$\begin{aligned} V = \text{var}(a\eta + b\eta^2) &= E(a\eta + b\eta^2)^2 - [E(a\eta + b\eta^2)]^2 \\ &= a^2 E(\eta^2) + 2ab E(\eta^3) + b^2 E(\eta^4) - [aE(\eta) + bE(\eta^2)]^2 \\ &= a^2 \mu_2 + 2ab \mu_3 + b^2 \mu_4 - (a\mu_1 + b\mu_2)^2 . \end{aligned}$$

Subtract the variance  $V$  from  $U$  and consider this as a function of  $a, b$ :

$$\begin{aligned} F(a, b) &= U - V = 4\gamma^2 \mu_1 \mu_3 \mu_2 - \gamma^3 \mu_3^2 - 4\mu_1^6 + 4\mu_1^6 \gamma - 4\mu_1^2 \mu_2^2 \gamma + 4\mu_1^4 \mu_2 - 4\gamma \mu_1^3 \mu_3 \\ &+ \gamma^2 \mu_4 \mu_1^2 - \mu_1^2 \mu_2^2 \gamma^2 + 4\mu_1^4 \mu_2 \gamma + 4\gamma^2 \mu_1^3 \mu_3 + \gamma^4 \mu_4 \mu_2 - 2\gamma^3 \mu_4 \mu_1^2 + 5\mu_1^2 \mu_2^2 \gamma^3 \\ &- 8\mu_1^4 \mu_2 \gamma^2 - 2\gamma^3 \mu_1 \mu_3 \mu_2 - \gamma^4 \mu_2^3 - a^2 \mu_2 - 2ab \mu_3 - b^2 \mu_4 + a^2 \mu_1^2 + 2a\mu_1 b \mu_2 + b^2 \mu_2^2 . \end{aligned}$$

If  $a, b$  are selected so that  $F(a, b) = 0$ , this would imply that  $U = V$ . First, choose  $b$  to eliminate the  $\mu_4$  term:

$$b = b_0 = \gamma \sqrt{\gamma^2 \mu_2 + \mu_1^2 - 2\gamma \mu_1^2} .$$

As  $\mu_2 \geq \mu_1^2$ , the inequality  $\gamma^2 \mu_2 + \mu_1^2 - 2\gamma \mu_1^2 \geq 0$  holds and so  $b$  is always correctly defined. Substitute this value into  $F(a, b)$ , then solve  $F(a, b_0) = 0$  with respect to  $a$ .  $F(a, b_0)$  is a quadratic function in  $a$ . Let  $D$  be the discriminant of  $F(a, b_0)$ . This discriminant can be simplified to

$$D = (\gamma - 1) (\gamma \mu_2 - \mu_1^2) (\mu_3 \gamma + 2\mu_1^3 - \gamma \mu_1 \mu_2 - 2\mu_2 \mu_1)^2 .$$

This is clearly non-negative when  $\gamma < 4mM/(m + M)^2 = \min_{\xi}(\mu_1^2(\xi)/\mu_2(\xi))$  and when  $\gamma \geq 1$  since  $\mu_2 - \mu_1^2 \geq 0$  by the Cauchy-Schwarz inequality. There is therefore a solution of the equation  $F(a, b) = 0$  and hence the conclusion can be drawn that  $\Phi^{(k)}$  monotonously increases and converges to a limit, i.e.  $\lim_{k \rightarrow \infty} \Phi^{(k)}$  exists for any initial design  $\xi^{(0)}$ .  $\square$

It has been established that the sequence  $\Phi^{(k)}$  converges to a limit, the value of that limit can now be calculated.

**Theorem 3.1.3.** *Let  $\xi^{(0)}$  be any non-degenerate probability measure with support  $\{\lambda_1, \dots, \lambda_d\}$  and let the sequence of probability measures  $\{\xi^{(k)}\}$  be defined via the*

updating formula (3.1). If  $\xi^{(0)}$  is such that  $\xi^{(0)}(\lambda_1) > 0$  and  $\xi^{(0)}(\lambda_d) > 0$ , then the limit  $\lim_{k \rightarrow \infty} \Phi^{(k)}$  of the sequence  $\Phi^{(k)} = \Phi(\xi^{(k)}) = \gamma\mu_2(\xi^{(k)}) - \mu_1^2(\xi^{(k)})$  does not depend on  $\xi^{(0)}$  and

$$\Phi^*(\xi^{(0)}) = \Phi^* = \frac{\gamma^2}{4}(m+M)^2 - \gamma mM;$$

moreover, the sequence of probability measures  $\{\xi^{(k)}\}$  converges (as  $k \rightarrow \infty$ ) to the probability measure  $\xi^*$  supported at the points  $m$  and  $M$  with weights

$$\xi^*(m) = \frac{2M - \gamma(m+M)}{2(M-m)} \quad \text{and} \quad \xi^*(M) = \frac{\gamma(m+M) - 2m}{2(M-m)},$$

where  $\frac{2m}{m+M} \leq \gamma \leq \frac{2M}{m+M}$ .

*Proof.* A limiting probability measure  $\xi$  will exist if applying the transformation  $\Psi(\xi)$  results in the original measure  $\xi$ , i.e. if  $\Psi(\xi) = \xi$  for some  $\xi$ . For the  $\gamma$ -steepest descent algorithm this equates to finding a measure  $\xi(x)$  such that

$$\xi(x) = \frac{(1 - \frac{\gamma m}{\mu_1})^2}{1 - 2\gamma + \gamma^2 \frac{\mu_2}{\mu_1^2}} \xi(x).$$

As was discussed in Section 2.3.3, an optimal design,  $\xi^*$  will be supported at the minimum and maximum eigenvalues  $m$  and  $M$ , so that

$$\xi^* = \left\{ \begin{array}{cc} m & M \\ p & q \end{array} \right\},$$

where  $q = 1 - p$ . The first and second moments of the limiting probability measure  $\xi^*$  are

$$\mu_1 = mp + M(1-p) \quad \text{and} \quad \mu_2 = m^2p + M^2(1-p)$$

respectively. Solving the equation

$$p = \frac{(1 - \frac{\gamma m}{\mu_1})^2}{1 - 2\gamma + \gamma^2 \frac{\mu_2}{\mu_1^2}} p$$

for  $p$  gives a weight at  $m$  for  $\xi^*$  of

$$\xi^*(m) = p^* = \frac{2M - \gamma(m+M)}{2(M-m)}$$

and therefore the weight at  $M$  will be

$$\xi^*(M) = q^* = 1 - p^* = \frac{\gamma(m+M) - 2m}{2(M-m)}.$$

Since  $\xi^*(m)$  and  $\xi^*(M)$  are weights they must be between 0 and 1, i.e.

$$0 \leq \frac{2M - \gamma(m + M)}{2(M - m)}, \frac{\gamma(m + M) - 2m}{2(M - m)} \leq 1$$

therefore  $\gamma$  is restricted to the range

$$\frac{2m}{m + M} \leq \gamma \leq \frac{2M}{m + M}.$$

Substituting the weights of the probability measure  $\xi^*$  into

$\Phi(\xi^{(k)}) = \gamma\mu_2(\xi^{(k)}) - \mu_1^2(\xi^{(k)})$  gives the limit of the sequence  $\Phi^{(k)}$ ;

$$\Phi^* = \frac{\gamma^2}{4}(m + M)^2 - \gamma mM$$

as required. □

It can be verified that the rate  $r(\xi^*)$  associated with the optimum design is the worst possible rate of the steepest descent algorithm,  $R_{\text{ref}}$ , by further substituting the value of  $p^*$  into the equation for the rate of convergence of the  $\gamma$ -steepest descent algorithm, (3.2).

If  $\gamma < 2m/(m + M)$  then the probability measure becomes degenerate and the design is supported solely at the smallest eigenvalue  $m$ . The consequence is that  $\Phi(M(\xi^*)) = m^2(\gamma - 1)$  and the rate  $r(\xi^*) = (1 - \gamma)^2$ . Similarly, if  $\gamma > 2M/(m + M)$  the design is supported entirely at the maximum eigenvalue  $M$ ,  $\Phi(M(\xi^*)) = M^2(\gamma - 1)$  and the rate in this case equals  $r(\xi^*) = (\gamma - 1)^2$ . These results are summarised in Table 3.1.

### 3.1.3 Speed of Convergence to the Optimum Design

It has been established that when the chosen relaxation coefficient  $\gamma$  lies within either  $2/(1 + \rho) < \gamma < 4\rho/(1 + \rho)^2$  or  $1 < \gamma < 2\rho/(1 + \rho)$  the renormalised  $\gamma$ -steepest descent algorithm converges to an optimum design and  $r(\xi^*) = R_{\text{ref}}$ . An investigation into how the choice of  $\gamma$  affects the speed with which the sequence  $\{\xi^{(k)}\}$  reaches the optimum design  $\xi^*$  is therefore of interest. This can be measured in one of two ways; either the number of iterations required to reach the optimum design within a certain degree of accuracy can be measured, or alternatively, the

	$0 < \gamma \leq \frac{2m}{m+M}$	$\frac{2m}{m+M} \leq \gamma \leq \frac{2M}{m+M}$	$\gamma \geq \frac{2M}{m+M}$
$\xi^*(m)$	1	$\frac{2M - \gamma(M+m)}{2(M-m)}$	0
$\Phi(M(\xi^*))$	$m^2(\gamma - 1)$	$\frac{1}{4}\gamma^2(m+M)^2 - \gamma mM$	$M^2(\gamma - 1)$
$r(\xi^*)$	$(1 - \gamma)^2$	$R_{\text{ref}} = \frac{(M-m)^2}{(m+M)^2}$	$(\gamma - 1)^2$

Table 3.1: Values of  $\xi^*(m)$ ,  $\Phi(\xi^*)$  and  $r(\xi^*)$ 

degree of accuracy to which the optimum design has been reached after a preset number of iterations can be measured. Using the latter option Table 3.2 indicates the speed with which the rate  $r^{(k)}$  reaches  $r^* = R_{\text{ref}}$  and thus also the speed with which  $\xi^{(k)}$  reaches  $\xi^*$ . The number of correct decimal places attained by  $r^{(k)}$  after 200 iterations of the  $\gamma$ -steepest descent algorithm for different  $\rho$  is given. Blank entries in the table reflect parameter choices where an optimum design is not attained due to  $\gamma$  being outside of both the intervals where convergence to an optimum design occurs. With the exception of when  $\gamma$  is near the boundary of one of the feasible intervals (where convergence to the optimum design takes longer) the speed of convergence of the sequence  $\{r^{(k)}\}$  is approximately the same for all feasible values of  $\gamma$ .

### 3.1.4 Behaviour of the Sequence $\{\Phi(\xi^{(k)})\}$

It was shown in the proof of Theorem 3.1.3 that for regions of  $\gamma$  where convergence to an optimal design occurs, i.e. where  $2m/(m+M) < \gamma < 4mM/(m+M)^2$  or  $1 < \gamma < 2M/(m+M)$ , the sequence  $\{\Phi(\xi^{(k)})\}$  converges to  $\Phi^* = \frac{\gamma^2}{4}(m+M)^2 - \gamma mM$ . For the values of  $\gamma$  which lie between these two ranges, the behaviour of the sequence  $\{\Phi(\xi^{(k)})\}$  is completely different and indeed for much of this range of  $\gamma$  the optimality

$\gamma$	$\rho = 2$	$\rho = 4$	$\rho = 10$	$\rho = 19$	$\rho = 49$	$\rho = 99$
0.05	-	-	-	-	1	-
0.1	-	-	-	10	-	-
0.15	-	-	-	14	-	-
0.2	-	-	2	-	-	-
0.25	-	-	17	-	-	-
0.3	-	-	18	-	-	-
0.35	-	-	-	-	-	-
0.4	-	-	-	-	-	-
0.45	-	19	-	-	-	-
0.5	-	18	-	-	-	-
0.55	-	17	-	-	-	-
0.6	-	16	-	-	-	-
0.65	-	-	-	-	-	-
0.7	18	-	-	-	-	-
0.75	18	-	-	-	-	-
0.8	18	-	-	-	-	-
0.85	13	-	-	-	-	-
<hr/>						
1.05	15	12	11	12	12	13
1.1	20	18	19	20	17	19
1.15	19	20	19	18	19	18
1.2	19	19	19	18	18	19
1.25	19	19	19	20	18	19
1.3	19	18	18	18	18	18
1.35	-	18	18	19	19	20
1.4	-	19	19	19	19	19
1.45	-	18	19	19	18	19
1.5	-	19	20	19	20	17
1.55	-	17	19	19	19	18
1.6	-	-	19	20	18	19
1.65	-	-	19	19	18	19
1.7	-	-	19	19	18	20
1.75	-	-	14	19	18	19
1.8	-	-	4	19	19	19
1.85	-	-	-	10	20	20
1.9	-	-	-	3	10	13
1.95	-	-	-	-	2	5

Table 3.2: Speed with which  $\{r^{(k)}\}$  converges to  $r^*$ , measured as the average number of decimal places of accuracy achieved by  $r^{(k)}$  after 200 iterations of the  $\gamma$ -steepest descent algorithm;  $d = 100$ .



criterion does not always converge to a single point regardless of the number of iterations the sequence is run for. Figure 3.1 and Figure 3.2 show the attractors of  $\Phi(\xi^{(k)})$  as a function of  $\gamma$  for various values of  $\rho$  for both a 2-dimensional and 50-dimensional problem. For much of the range  $4mM/(m+M)^2 < \gamma < 1$ , the limiting behaviour of the sequence of the criterion shows chaos to be present; however, for some values of  $\gamma$  in amongst the chaos, the attractors of  $\Phi(\xi^{(k)})$  form cycles of varying size. The value of  $\gamma$  chosen thus has a huge impact on the nature of the asymptotic behaviour of the sequence  $\{\Phi(\xi^{(k)})\}$ . Cyclic behaviour appears more often when  $d = 2$  and when the condition number,  $\rho$  is small.

To further demonstrate the limiting behaviour of the sequence  $\{\Phi(\xi^{(k)})\}$  Figure 3.3 and Figure 3.4 show the normalised values of  $\Phi(\xi^{(k)})$  as a function of  $k$ . For equivalent graphs for different values of the relaxation coefficient,  $\gamma$ , see Appendix C. For  $\gamma = 0.985$  a clear pattern can be made out where, for a particular iteration number, the value of the optimality criterion shoots up dramatically from its previous value before then descending slowly back down to more typical values as  $k$  increases. The sequence then enjoys an unpredictable phase where low values of  $\Phi(\xi^{(k)})$  are witnessed before again rising sharply for the process to repeat itself. The length and frequency of each phase of this behaviour are not constant however it can be observed that as  $\gamma$  moves closer and closer to 1, the graph of  $\Phi(\xi^{(k)})$  as a function of  $k$  shows that the length of each phase generally increases.

Figure 3.5 shows the spread of values of  $\{\Phi(\xi^{(k)})\}$  for  $k = 1, \dots, 10000$ , for different values of the relaxation coefficient  $\gamma$ . The impact the choice of the relaxation coefficient has on the distribution of  $\{\Phi(\xi^{(k)})\}$  can clearly be seen. For example, when  $\gamma = 0.8$ , a distribution is produced where the majority of values of  $\Phi(\xi^{(k)})$  are located at the centre of the range of possible values the optimality criterion can take on. When  $\gamma = 0.999$ , on the other hand, the distribution of  $\{\Phi(\xi^{(k)})\}$  shows the optimality criterion to take on the extreme values of the range more often than more central values.

Figure 3.6 shows the relationship between the value of the optimality criterion at consecutive iterations, i.e.  $\Phi(\xi^{(k)})$  and  $\Phi(\xi^{(k+1)})$ , for various values of the relaxation coefficient  $\gamma$ . For some values of  $\gamma$  the progression from iteration to iteration is more

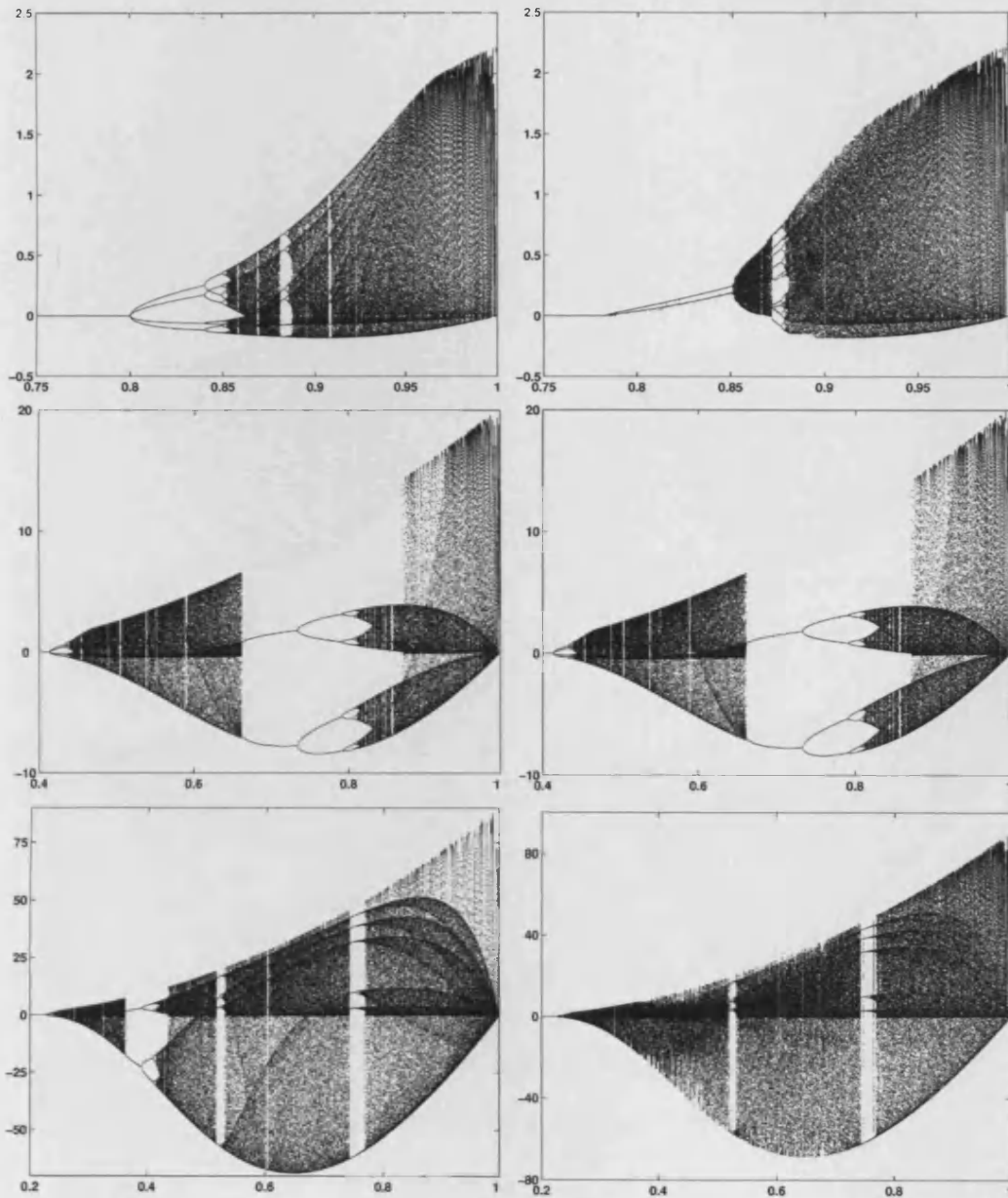


Figure 3.1:  $\Phi(\xi^{(k)})$  for  $k = 750, \dots, 1000$ ; plotted as a function of  $\gamma$  for (left)  $d = 2$  and (right)  $d = 50$  and with, from top to bottom:  $\rho = 4, \rho = 10, \rho = 20$ .

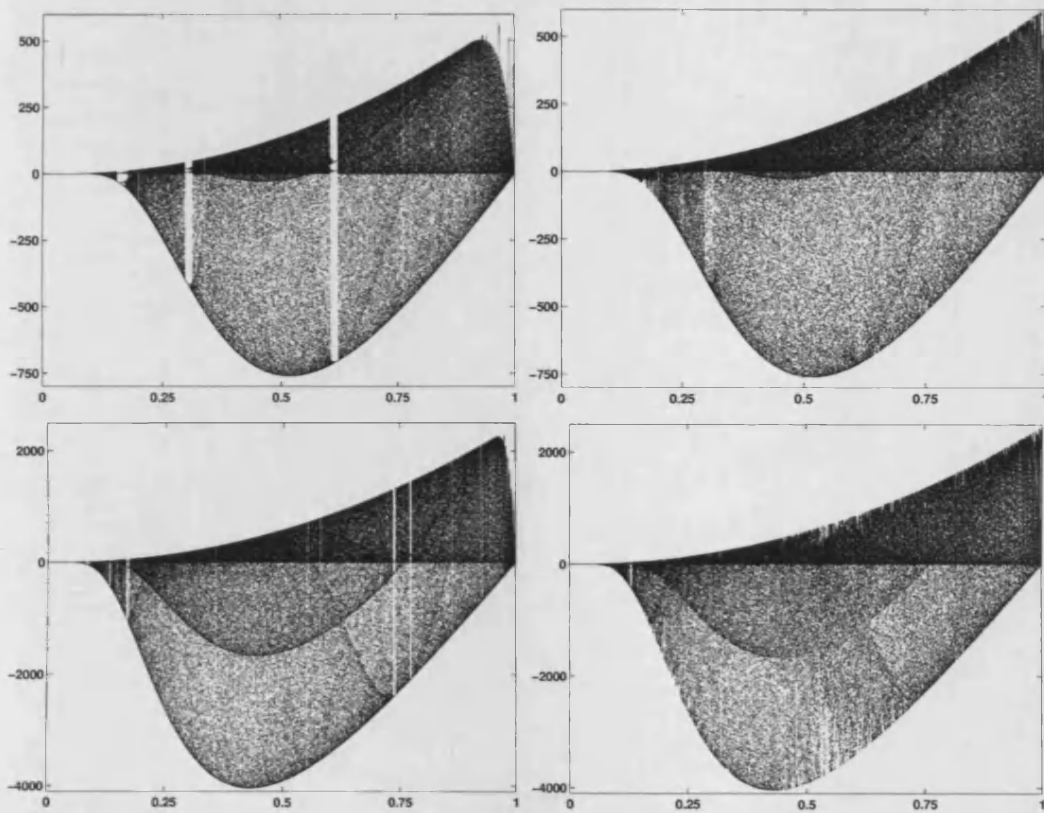


Figure 3.2:  $\Phi(\xi^{(k)})$  for  $k = 750, \dots, 1000$ ; plotted as a function of  $\gamma$  for (left)  $d = 2$  and (right)  $d = 50$  and with, from top to bottom:  $\rho = 50, \rho = 100$ .

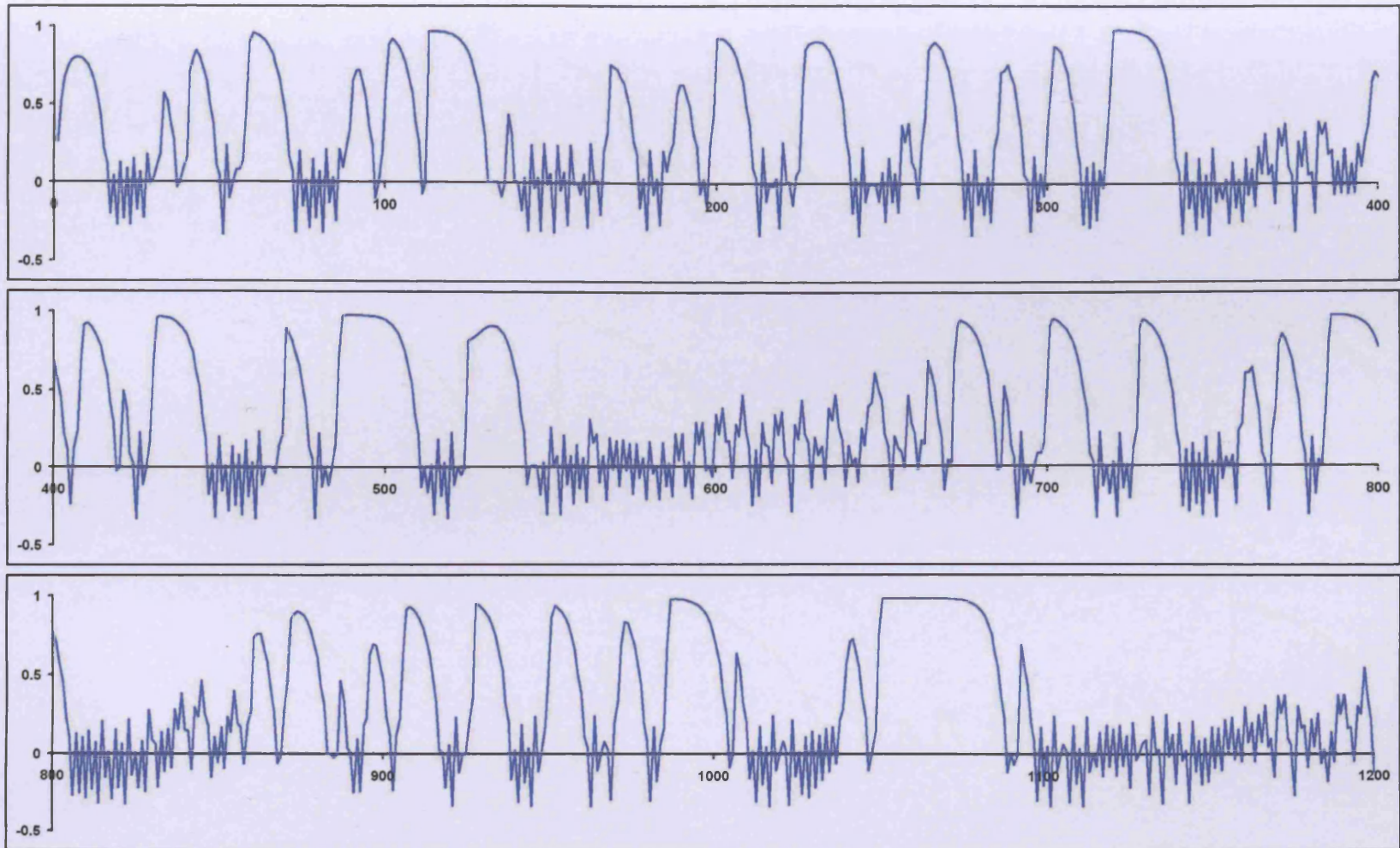


Figure 3.3: Trajectory of the efficiency  $\Phi(\xi^{(k)})/\max_{\xi} \Phi(\xi)$  plotted as a function of  $k$  for a single realisation of the  $\gamma$ -steepest descent algorithm with  $\gamma = 0.9$ ,  $\rho = 10$ ,  $d = 100$ .

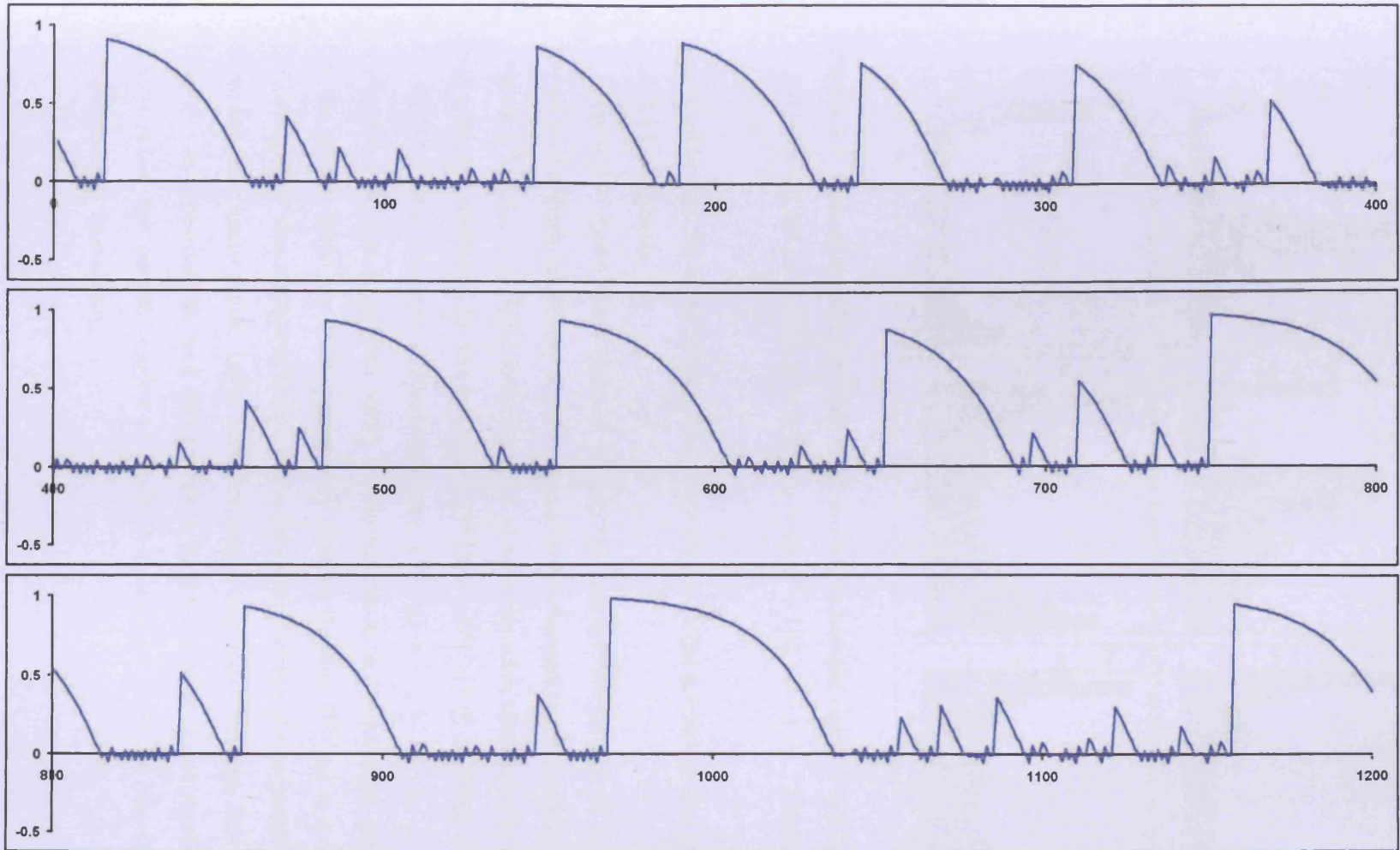


Figure 3.4: Trajectory of the efficiency  $\Phi(\xi^{(k)}) / \max_{\xi} \Phi(\xi)$  plotted as a function of  $k$  for a single realisation of the  $\gamma$ -steepest descent algorithm with  $\gamma = 0.985$ ,  $\rho = 10$ ,  $d = 100$ .

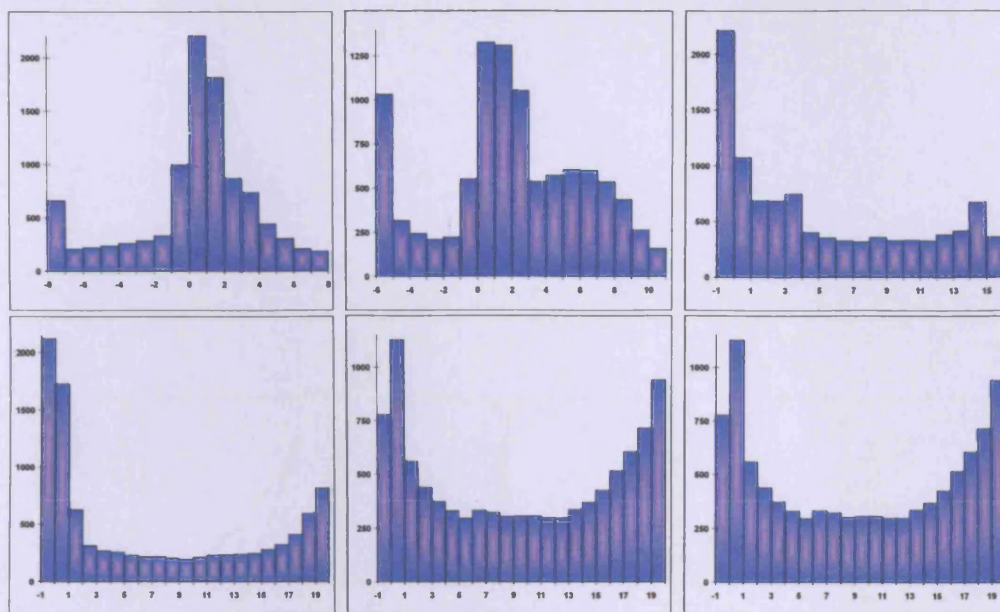


Figure 3.5: Histogram of  $\Phi(\xi^{(k)})$  for, from top left to bottom right:  $\gamma = 0.7$ ,  $\gamma = 0.8$ ,  $\gamma = 0.9$ ,  $\gamma = 0.985$ ,  $\gamma = 0.99$ ,  $\gamma = 0.999$ ;  $\rho = 10$ ;  $d = 100$ ,  $k = 1, \dots, 10000$ .

predictable than for other values, with chaos clearly being present in the process for  $\gamma = 0.9$  for example.

Figure 3.7 shows the weight at  $\lambda_1$  plotted against the weight at  $\lambda_d$  at each iteration of a single trajectory of the renormalised  $\gamma$ -steepest descent algorithm. The weights  $\xi^{(k)}(\lambda_1), \dots, \xi^{(k)}(\lambda_d)$  must sum to 1 at each iteration therefore the points on each plot are restricted by the inequality  $\xi^{(k)}(\lambda_d) + \xi^{(k)}(\lambda_1) \leq 1$ . When  $\xi^{(k)}(\lambda_d) \cong 1 - \xi^{(k)}(\lambda_1)$  this indicates a situation where,  $\xi^{(k)}(\lambda_2) = \dots = \xi^{(k)}(\lambda_{d-1}) \cong 0$ , i.e. where the design is supported solely at the minimum and maximum eigenvalues. On the graphs there are many points very close to the line  $\xi^{(k)}(\lambda_d) = 1 - \xi^{(k)}(\lambda_1)$  indicating that this situation occurs quite frequently within the sequence however there are also many points below this line suggesting that although from time to time  $\xi^{(k)}$  is supported at only two points, a limiting design is not reached, since after a while the middle weights re-establish themselves and the algorithm once again descends into chaos.

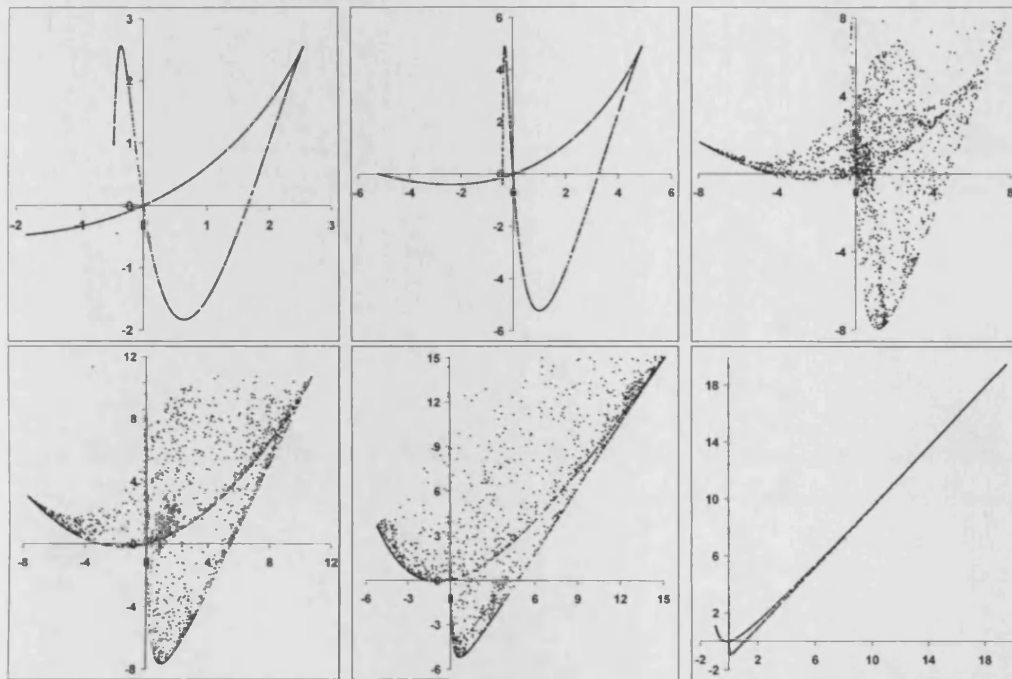


Figure 3.6: Plots of the pairs  $(\Phi(\xi^{(k)}), \Phi(\xi^{(k+1)}))$  for, from top left to bottom right  $\gamma = 0.5, 0.6, 0.7, 0.8, 0.9, 0.985$ . Points plotted are the last 2000 of 10000 iterations;  $d = 100, \rho = 10$ .

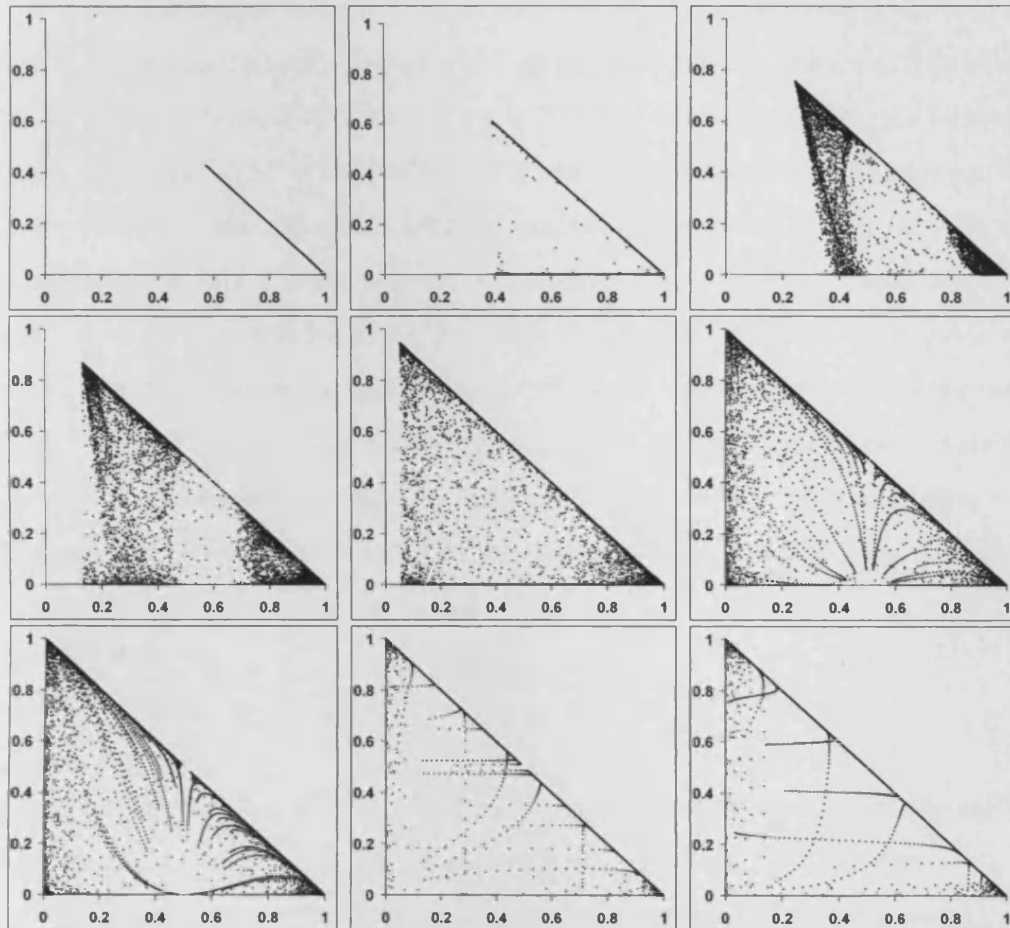


Figure 3.7: Plots of the pairs  $(\xi^{(k)}(\lambda_1), \xi^{(k)}(\lambda_d))$  for the  $\gamma$ -steepest descent algorithm with (from top left to bottom right):  $\gamma = 0.5, 0.6, 0.7, 0.8, 0.9, 0.985, 0.99, 0.999, 0.9995$ ;  $\rho = 10$ ;  $d = 100$ . Points plotted are the last 8000 of 10000 iterations.



## 3.2 Asymptotic Rate of Convergence

In this section a detailed analysis of the asymptotic rate of convergence of the  $\gamma$ -steepest descent algorithm and the factors which affect this convergence rate are given.

It is possible to give an analytical formula for the asymptotic rate of convergence when  $\gamma$  is in regions where convergence to an optimal design occurs, i.e. if  $2m/(m+M) < \gamma < 4mM/(m+M)^2$  or  $1 < \gamma < 2M/(m+M)$ ; then the rate is simply  $R_{\text{ref}} = (M-m)^2/(M+m)^2$ . When  $\gamma$  is outside of these regions, however, the manner in which the algorithm behaves entirely changes and it will be seen that the algorithm exhibits either chaotic behaviour or forms cycles of some length for some values of  $\gamma$  in this interval. An analytical formula for the convergence rate cannot, therefore, always be specified since  $r^{(k)}$  is not constant for much of the range  $4mM/(m+M)^2 < \gamma < 1$ . In order to study the asymptotic rate of convergence in this case, it is instead necessary to simulate the behaviour of the algorithm with different starting vectors and calculate an approximation to the asymptotic rate of convergence (1.12) by using the formula

$$R = \frac{1}{N} \sum_{j=1}^N \left[ \left( \prod_{i=1}^k r^{(i)} \right)^{1/k} \right] = \frac{1}{N} \sum_{j=1}^N \left[ \exp \left\{ \frac{1}{k} \sum_{i=1}^k \log r^{(i)} \right\} \right]$$

where  $N$  is the number of different random starting vectors over which the asymptotic rate is averaged and  $k$ , the number of iterations performed, is large.

With fair comparisons in mind, all approximate average asymptotic rates given in this thesis were calculated from simulations run for 1000 iterations ( $k = 1000$ ) and averaged over 300 replicates. The starting vector  $x^{(0)}$  for each replicate was generated randomly, however if  $x_i^{(0)} = 0$  for any  $i$  then the complexity of the problem is reduced yielding unrepresentative rates so these zero entries were replaced by a non-zero number. Simulations were run for the most part in MatLab 13 although Maple 10 was used in situations where this better facilitated the analysis. Examples of a typical programme in MatLab and Maple can be found in Appendix A.

### 3.2.1 Dependence on $\gamma$

In order to achieve the best asymptotic rates of convergence when using the  $\gamma$ -steepest descent algorithm it is first necessary to find the optimum value of the relaxation parameter  $\gamma$ . It is important to see whether the optimum value of  $\gamma$  changes for different values of parameters, such as the condition number  $\rho$  or the dimension  $d$ , in order to recommend a useful value of  $\gamma$  in any situation.

Figure 3.8 shows the average asymptotic rate of convergence of the  $\gamma$ -steepest descent algorithm as a function of  $\gamma$  for typical parameter values. As was discussed in the previous section, if  $1 < \gamma < 2M/(m+M)$  or  $2m/(m+M) < \gamma < 4mM/(m+M)^2$  then the renormalised  $\gamma$ -steepest descent algorithm converges to an optimal design and the rate of convergence in these regions is equal to  $R_{\text{ref}}$ . If  $0 < \gamma < m/(m+M)$  or  $M/(m+M) < \gamma < 2$  then the algorithm converges with a rate worse than that of  $R_{\text{ref}}$  and if  $\gamma < 0$  or  $\gamma > 2$  then  $R > 1$  indicating that divergence occurs if  $\gamma$  is not contained within  $0 \leq \gamma \leq 2$ . The region of interest, however, is when  $\gamma$  lies in the range  $4mM/(m+M)^2 < \gamma < 1$ . In this area of the graph it is plain to see that much better rates of convergence are present, the best rate occurring when  $\gamma$  is slightly less than 1. To gain a better insight into exactly how the algorithm behaves when these better rates of convergence are produced, a more informative plot is needed. Figure 3.9 shows the rate,  $r^{(k)}$ , at iteration  $k$  for  $k = 750, \dots, 1000$  for a single trajectory plotted as a function of  $\gamma$ . By displaying each individual rate  $r^{(k)}$  for every value of  $\gamma$  sampled, a clear picture can be gained as to both how varied the rates can be and also what the nature of the algorithm is like e.g. whether any cyclic or chaotic behaviour is revealed. From Figure 3.9 it can be observed that, as  $\gamma$  increases, the nature of the algorithm evolves from converging at a constant rate equal to  $R_{\text{ref}}$  (denoted  $R_{\text{max}}$  in the figure), to converging at a constant rate better than  $R_{\text{ref}}$ , to forming cycles of increasing size and finally descending into chaos. It is clear therefore that varying choices of  $\gamma$  produce optimisation algorithms with totally different behaviours.

Referring back to Figure 3.8, it can be seen in Figure 3.9, that the area of the graph corresponding to the best average asymptotic rates of convergence is that where chaos occurs. Here the rates  $r^{(k)}$  are incredibly varied and at first glance it

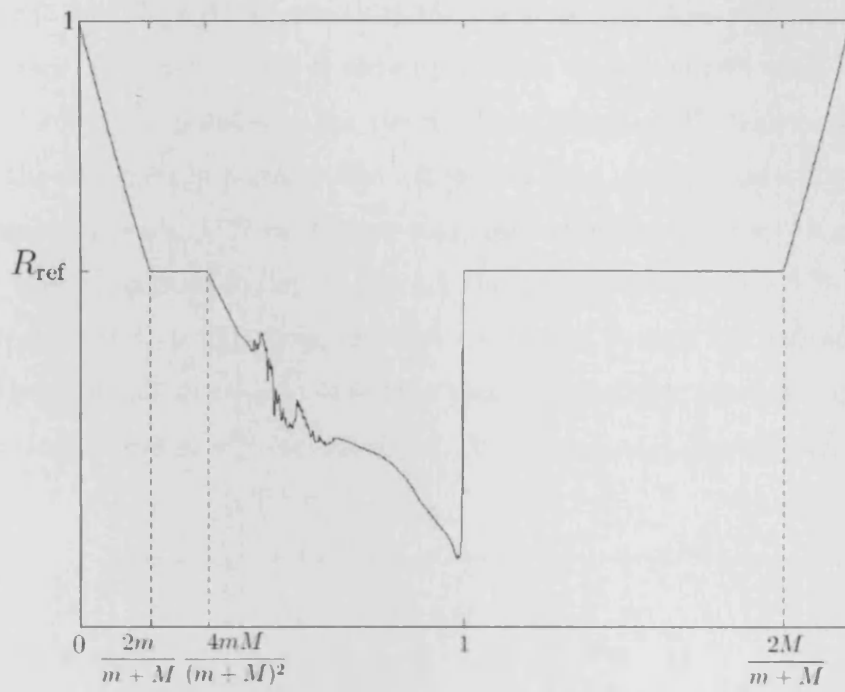


Figure 3.8: Average asymptotic rate of convergence as a function of  $\gamma$  for the  $\gamma$ -steepest descent algorithm.

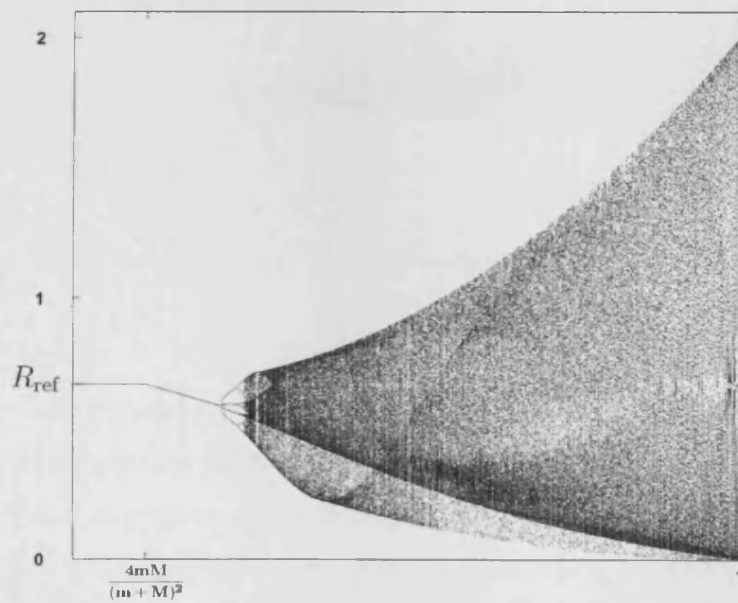


Figure 3.9: Attractors of  $r^{(k)}$  as a function of  $\gamma$  for the  $\gamma$ -steepest descent algorithm.

is not obvious that the asymptotic rate of convergence in these parts is attractive at all since it is clear  $r^{(k)}$  is extremely high in some iterations. The reason for the fast average asymptotic rates of convergence lies with those  $r^{(k)}$  close to zero. In Figure 3.9 it is not possible to see clearly those values of  $r^{(k)}$  close to zero, partly due to the denseness of points in the region (which in itself indicates that there are many more values of  $r^{(k)}$  close to zero compared with larger values). A more useful plot in this situation is Figure 3.10 where the attractors of  $(-\ln(r^{(k)}))$  are plotted as a function of  $\gamma$ . In this form, the higher values of  $(-\ln(r^{(k)}))$  indicate the more desirable rates and vice versa. It is thus apparent that the advantage gained from these better values of  $r^{(k)}$  far outweighs the disadvantage gained from the worse values.

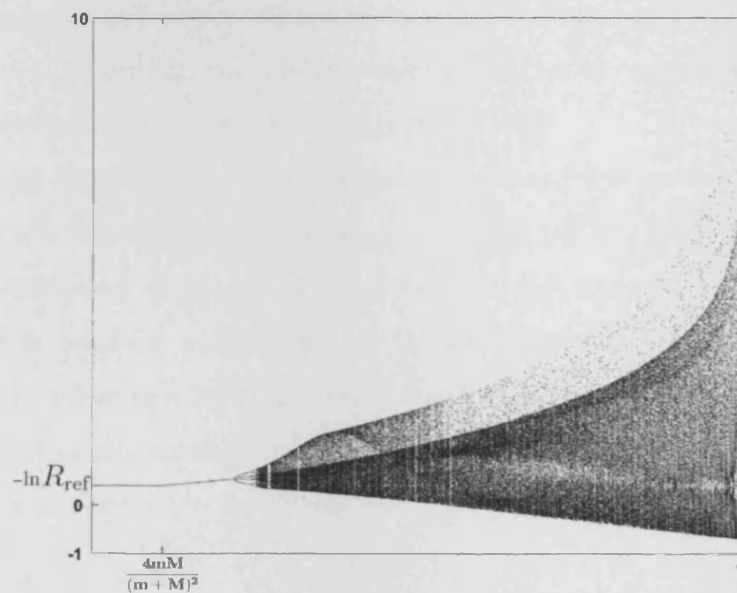


Figure 3.10: Attractors of  $(-\ln r^{(k)})$  as a function of  $\gamma$  for the  $\gamma$ -steepest descent algorithm.

While the best average asymptotic rates of convergence occur when  $\gamma$  is just less than one, in this region the algorithm exhibits chaotic behaviour and the actual rate at iteration  $k$  can vary quite dramatically. In fact, it is even possible that at some iterations  $r^{(k)} > 1$  which corresponds to a situation where an iteration produces a new approximation to the minimum point which is further away from  $x^*$  than the previous approximation. While on average, these varied rates produce a much more

desirable overall asymptotic rate of convergence than the classical steepest descent algorithm, it is not possible to guarantee a fast asymptotic rate of convergence for a single trajectory due to the chaotic nature of the algorithm. The situation is the same for the BB algorithm whose step length is defined in (1.15) and the CBB algorithm defined in (1.17). There is, however, a region of values of  $\gamma$  just before the bifurcation to chaos procedure starts, for which an asymptotic rate of convergence better than that of the steepest descent algorithm is guaranteed to be achieved. In this region the rate  $r^{(k)} = r \quad \forall k$ , i.e. the rate is constant and no chaotic behaviour comes into play as can be seen in Figure 3.9. To guarantee a faster asymptotic rate of convergence than the steepest descent algorithm, the best value of  $\gamma$  to choose is thus the point at which the bifurcation to chaos phenomenon starts, since the rate is decreasing up to this point. Although this rate is considerably better than the steepest descent algorithm, the possible asymptotic rate of convergence achievable when  $\gamma$  is just less than one is considerably better still.

From Figure 3.11 it can be observed that the value of  $\gamma$  at which the bifurcation to chaos procedure starts is smallest when  $\lambda_2 = (m+M)/2$ , i.e. when the midpoint of  $\lambda_1$  and  $\lambda_d$  is contained within the eigenvalue set. In this situation the region where the rate  $r^{(k)}$  is constant ends sooner and so the asymptotic rate of convergence attained at the point at which bifurcation starts is at its worst for that particular value of  $\rho$ . This phenomenon is true for all dimensions  $d$  and condition numbers  $\rho$  but is more prominent when  $d$  is small.

### 3.2.2 Dependence on $\rho$

In a similar manner to the classical steepest descent algorithm, and in fact all gradient algorithms studied in this thesis, the asymptotic rate of convergence of the  $\gamma$ -steepest descent algorithm will worsen as the condition number  $\rho$  increases, regardless of the value of  $\gamma$ . Figure 3.12 shows the relationship between the average asymptotic rate of convergence and the condition number  $\rho$  for several values of  $\gamma$  for a small dimensional problem ( $d = 4$ ) and a larger dimensional problem ( $d = 100$ ). While the relationship between the two is more erratic for some values of  $\gamma$  compared to others this outcome is analogous for small and larger dimensions. Figure 3.13

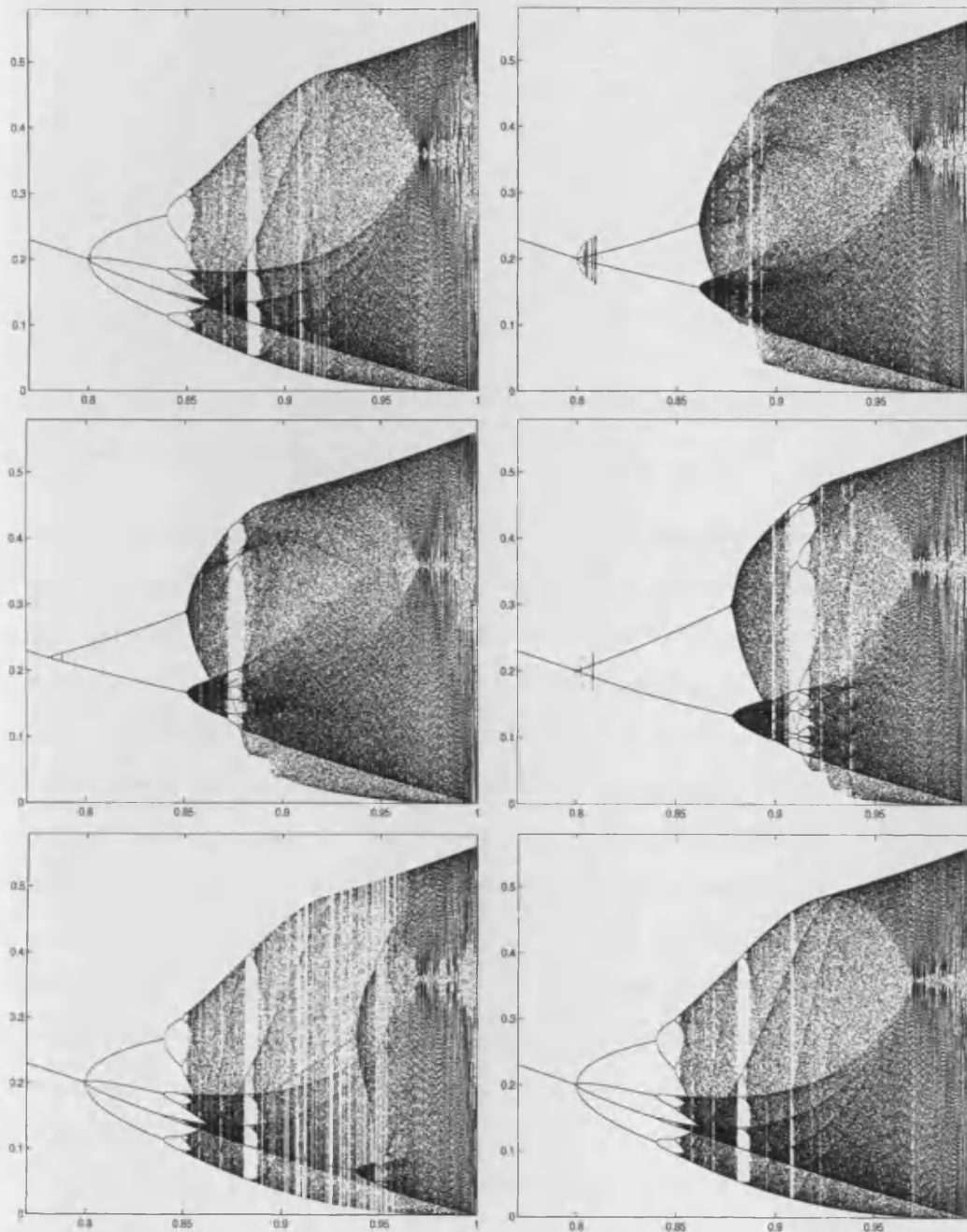


Figure 3.11: Attractors of  $r^{(k)}$  as a function of  $\gamma$  for a 3-dimensional problem with  $\lambda_1 = 1$ ,  $\lambda_3 = 4$  and, from top left to bottom right:  $\lambda_2 = 3/2, 2, 5/2 = (m + M)/2, 3, 7/2, 4$ .

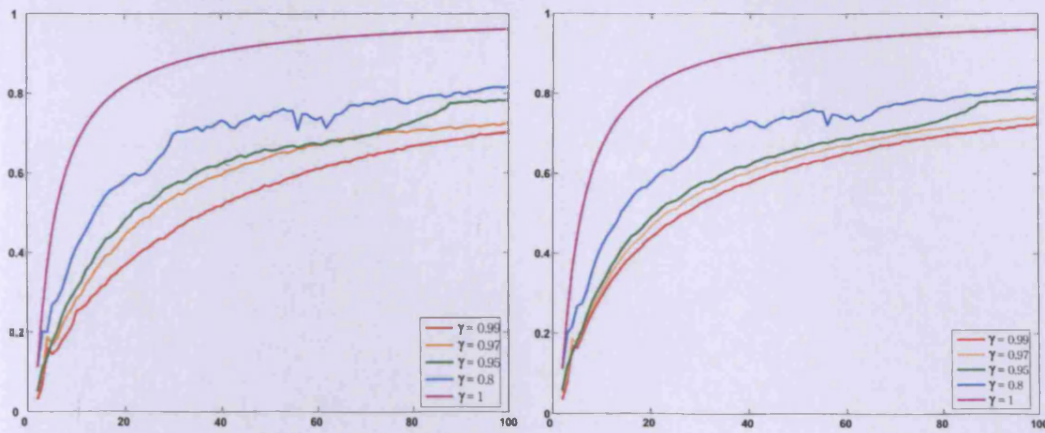


Figure 3.12: Average asymptotic rate of convergence as a function of  $\rho$  for (left)  $d = 4$  and (right)  $d = 100$ .

shows the average asymptotic rate of convergence of the  $\gamma$ -steepest descent algorithm as a function of  $\gamma$  for various values of  $\rho$ . For  $\rho \gtrsim 10$  the optimum value of  $\gamma$  is approximately 0.985 however as  $\rho$  increases the trough in which the best asymptotic rate of convergence is found becomes narrower and narrower and so the importance of precision in selecting the value of  $\gamma$  to be used in a certain situation becomes more profound, as a small deviation either side of the optimum value will result in a relatively large increase in  $R$ . It must be noted however, that in reality the condition number of the matrix  $A$  is generally unknown and so precise selection of  $\gamma$  is not possible. For  $\rho < 10$  the optimum value of  $\gamma$  is smaller. Fortunately, here the trough is not so deep so using a value of  $\gamma$  slightly larger will not spoil the lucrative rate greatly. A value of  $\gamma = 0.985$  thus appears to be the most sensible value for the relaxation parameter for a generic large dimensional situation where the condition number is unknown.

Figure 3.14 and Figure 3.15 show the corresponding plots of the attractors of  $r^{(k)}$  as a function of  $\gamma$  for  $d = 50$  in the right column and similar plots for  $d = 2$  in the left column. As  $\rho$  increases, the region in which better asymptotic rates of convergence are demonstrated,  $4mM/(m+M)^2 < \gamma < 1$ , naturally increases and thus chaotic behaviour is present in the algorithm's convergence rates for a greater range of  $\gamma$ .

The overall shape of the graph is comparable when  $d = 2$  and  $d = 50$  for

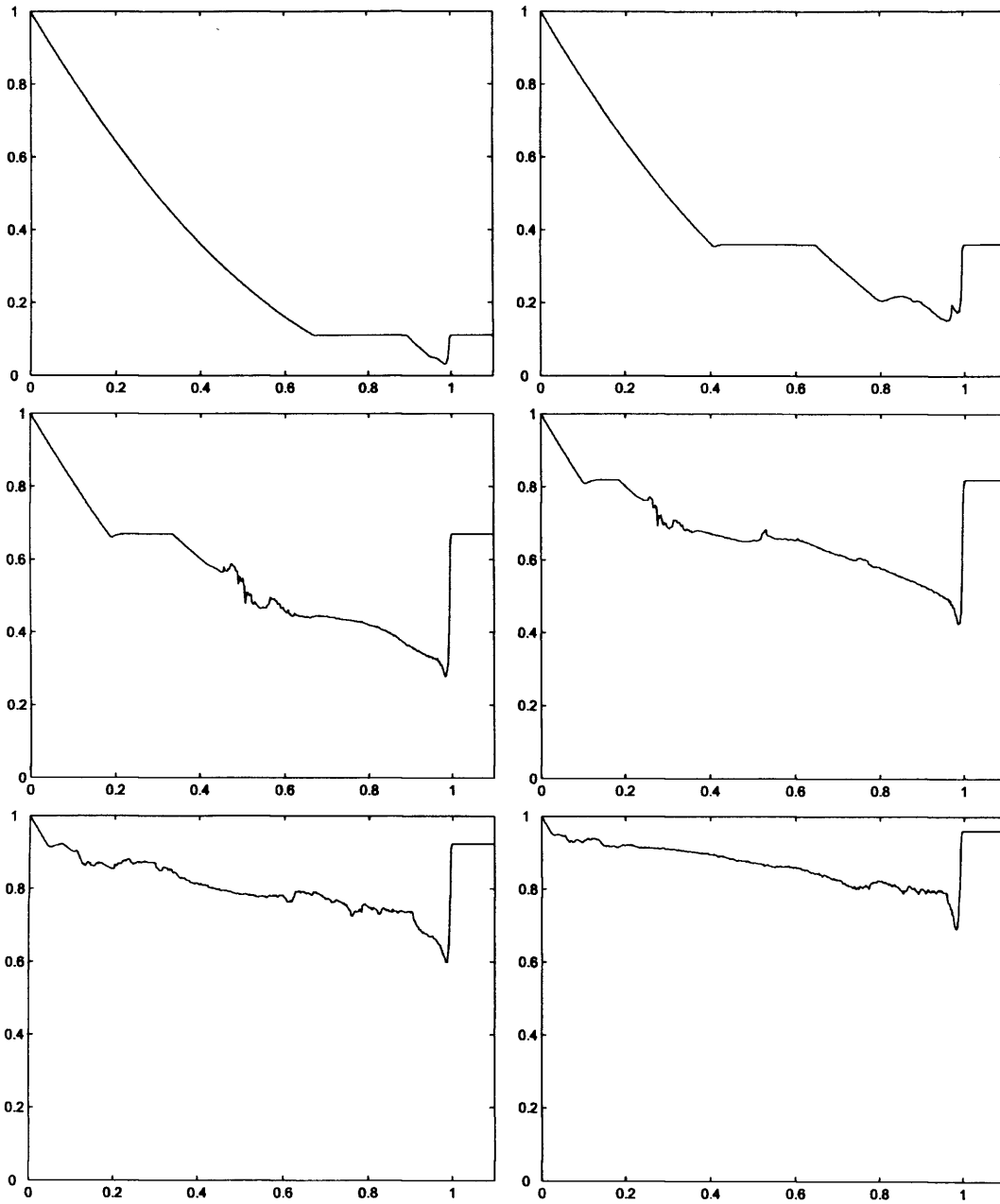


Figure 3.13: Average asymptotic rate of convergence as a function of  $\gamma$  for, from top left to bottom right:  $\rho = 2, 4, 10, 20, 50, 100$ ;  $d = 50$ .



$\rho \cong 10$  onwards. The main difference, however, in the 2-dimensional case is that, where in larger dimensions chaotic behaviour is exhibited, in the 2-dimensional case cycles are often formed of varying size. Cyclic behaviour is also present in larger dimensions but to a much lesser extent. To demonstrate the frequency and size of cycles present, Figure 3.16 shows the natural logarithm of the number of attractors of  $r^{(k)}$  as a function of  $\gamma$  in the 2 and 3-dimensional case with  $\rho = 4$ . It must be noted, however, that in situations where chaos is occurring, due to limitations on the number of iterations run, in this case 2000, the natural logarithm of the number of attractors is artificially limited to  $\ln(2000) \cong 7.6$ . For those values of  $\gamma$ , the attractor is thus better thought of as having an infinite number of points.

### 3.2.3 Dependence on $d$

It was seen in the previous chapter that, for the steepest descent algorithm, the number of dimensions in the problem does not have a significant effect on the asymptotic rate of convergence of the algorithm provided the number of dimensions is greater than approximately 10. The situation is not dissimilar for the  $\gamma$ -steepest descent algorithm, although with this algorithm, dimensionality does have a more pronounced effect on the asymptotic rate of convergence for a small range of  $\gamma$ . More precisely, the observed relationship between the number of dimensions of the problem and the asymptotic rate of convergence is approximately constant for all values of  $\gamma$  in the range  $0 < \gamma < 2$  with the exception of when  $\gamma$  is marginally less than 1, i.e. in the region where the best asymptotic rates are exhibited. When  $\gamma$  is in this zone the dimension has a more marked effect on the rate of convergence as can be seen in Figure 3.17, however, even here the difference between the rate with  $d = 50$  and  $d = 100$  is very marginal.

From Figure 3.17 it is evident that the value of  $d$  also effects the optimum value of  $\gamma$  when  $d$  is less than approximately 50. The fewer the number of dimensions, the closer to 1 the optimum value of  $\gamma$  becomes. Since the size of matrix  $A$  is always known in advance of applying the algorithm it is possible to adjust the value of  $\gamma$  used to account for this however, in general gradient optimisation algorithms would normally be used in situations where  $d$  is large and from  $d = 50$  upwards the number

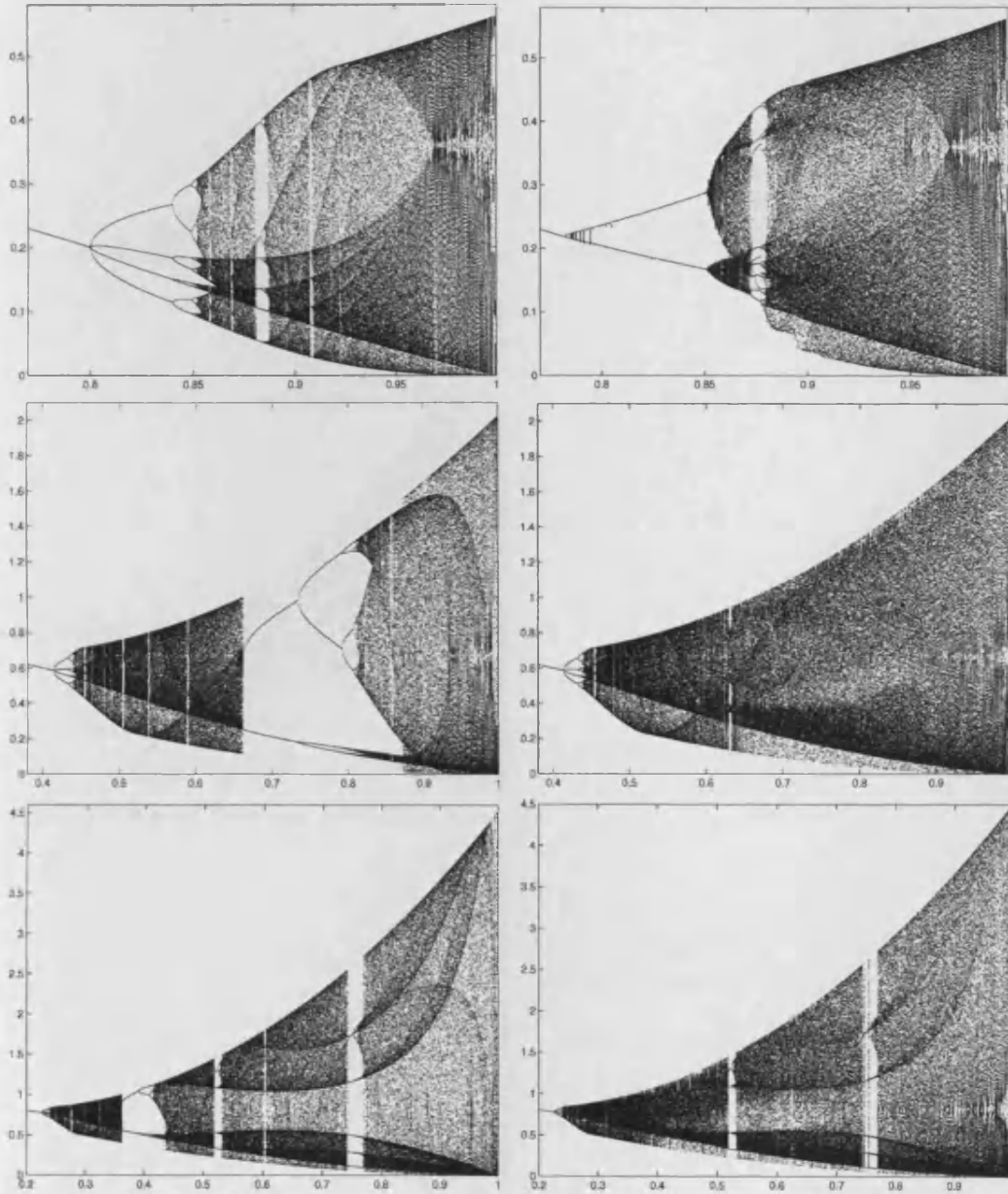


Figure 3.14: Attractors of  $r^{(k)}$  as a function of  $\gamma$  for  $d = 2$  (left) and  $d = 50$  (right) and from top to bottom;  $\rho = 4, 10, 20$ .

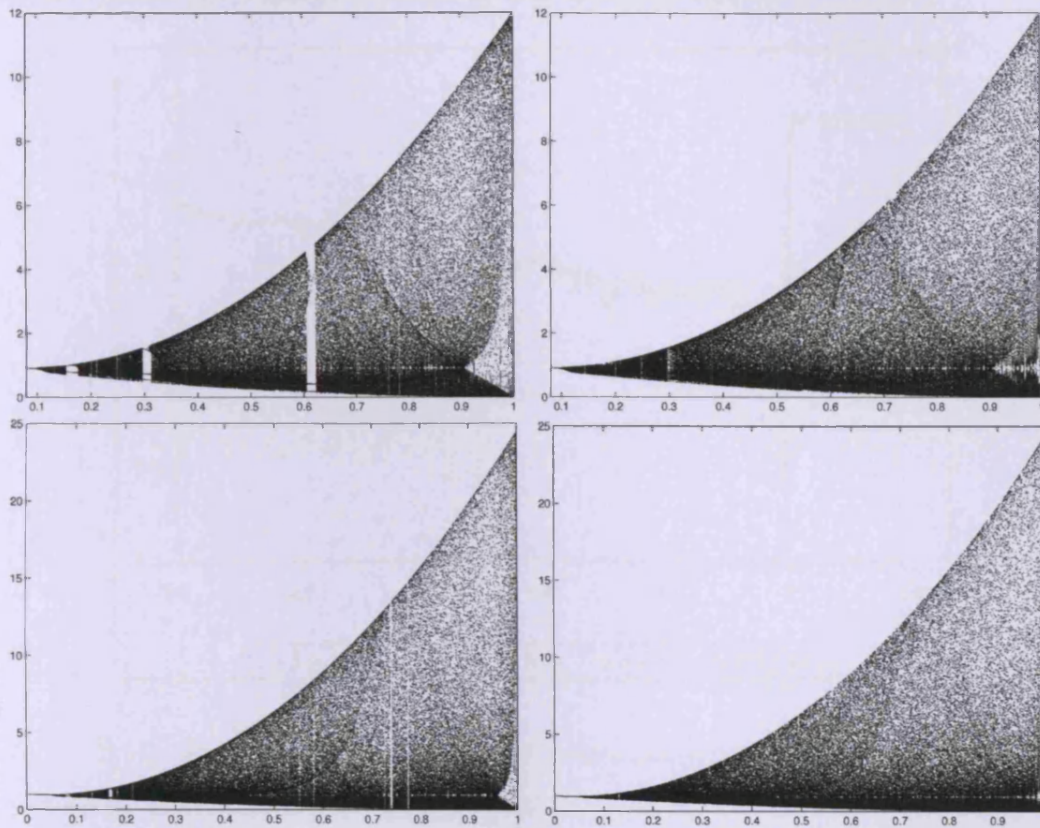


Figure 3.15: Attractors of  $r^{(k)}$  as a function of  $\gamma$  for  $d = 2$  (left) and  $d = 50$  (right) and from top to bottom;  $\rho = 50, 100$ .

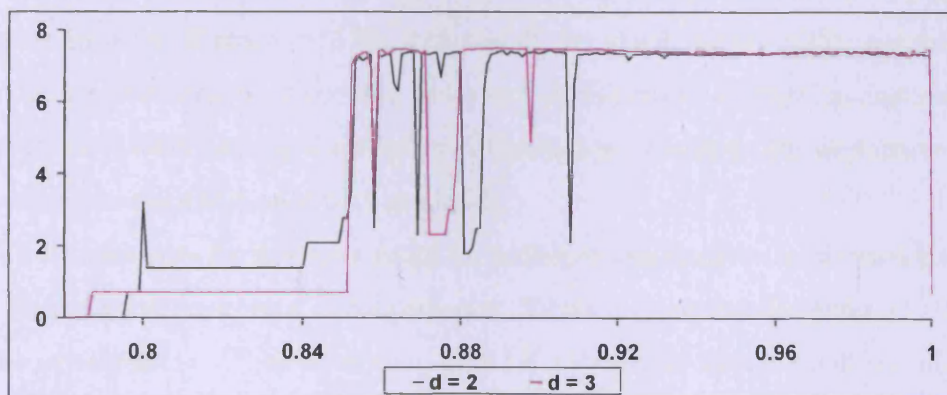


Figure 3.16:  $\ln(\text{number of attractors})$  as a function of  $\gamma$  for the  $\gamma$ -steepest descent algorithm with  $d = 2$  and  $d = 3$ ;  $\rho = 4$ .

of dimensions has virtually no effect on the optimum value of  $\gamma$ .

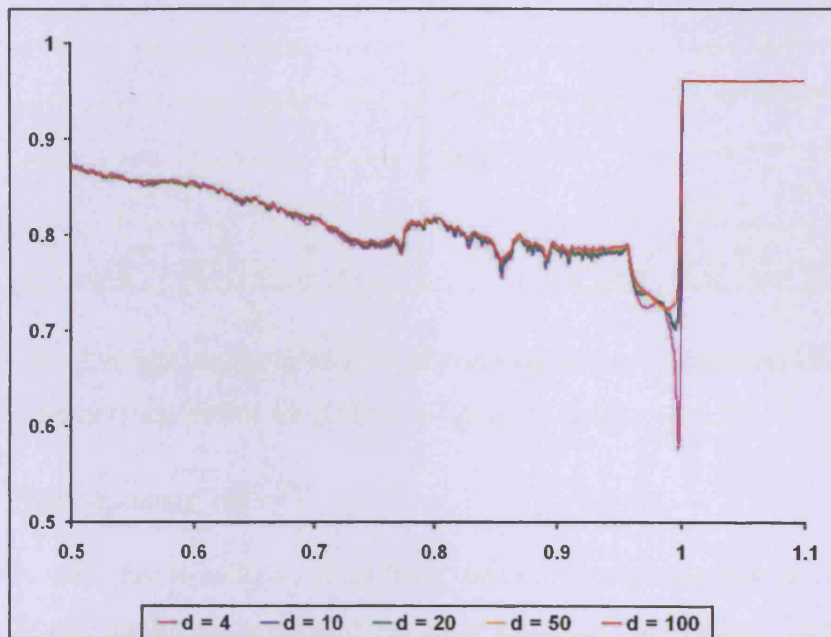


Figure 3.17: Average asymptotic rate of convergence as a function of  $\gamma$  for the  $\gamma$ -steepest descent algorithm with  $\rho = 100$ .

Figure 3.18 illustrates the difference in effect the dimensionality of the problem has on the average asymptotic rate of convergence of the  $\gamma$ -steepest descent algorithm with  $\gamma = 0.9$  and  $\gamma = 0.99$ . The left graph is typical of most values of  $\gamma$  and shows that the rate of convergence is virtually independent of the number of dimensions of  $A$ . When  $\gamma = 0.99$ , depicted in the graph on the right, a somewhat more erratic relationship is present; however fluctuations in average asymptotic rate are relatively small and, as the number of dimensions increases, the asymptotic rate of convergence worsens only very gradually.

To demonstrate further how small an influence the number of dimensions of  $A$  has on the asymptotic rate of convergence, Figure 3.19 shows, for different sizes of  $d$ , the attractors of  $r^{(k)}$  as a function of  $\gamma$  for a situation with a condition number of  $\rho = 4$ . For  $d = 10$  and larger the graphs are almost identical.

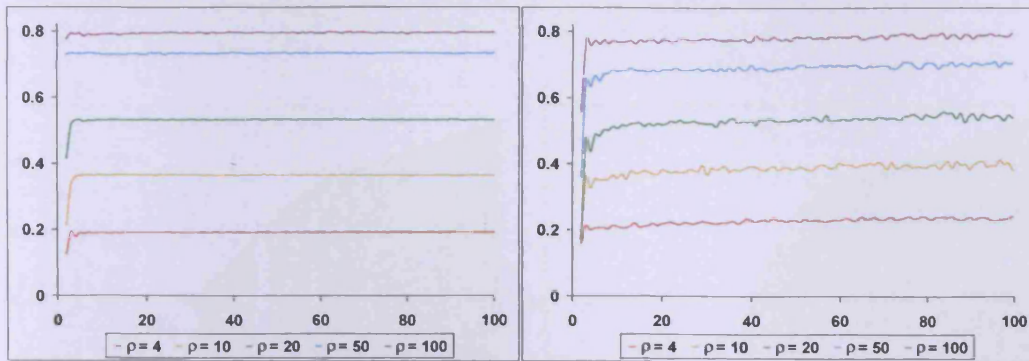


Figure 3.18: Average asymptotic rate of convergence as a function of  $d$  for the  $\gamma$ -steepest descent algorithm for (left)  $\gamma = 0.9$  and (right)  $\gamma = 0.99$ .

### 3.2.4 Behaviour of $r^{(k)}$

It has been seen that in order to profit from the best asymptotic rate of convergence possible for this algorithm, a value of  $\gamma$  slightly less than 1 is required. In this region the rate at each iteration varies dramatically and for some iterations even indicates that the approximation to the minimum point is worse than at the previous iteration. In order to understand how the algorithm is benefiting from taking steps in the wrong direction a more detailed look at the rate  $r^{(k)}$  at each iteration is necessary.

Figure 3.20 and Figure 3.21 show the rate  $r^{(k)}$  as a function of  $k$  for  $\gamma = 0.99$  and  $\gamma = 0.995$  respectively. Both start with an oscillatory period similar to that of the steepest descent algorithm, with similarly poor rates as a consequence. After some iterations the oscillations grow steadily in size culminating in producing extremely high rates (greater than 1) followed by extremely low rates (close to zero). The asymptotic rate of convergence is calculated by taking the geometric mean of the rates  $r^{(k)}$ , in this case a rate of around 2, followed by a rate close to zero would average out at a profitable figure. From time to time the algorithm breaks out of this oscillatory behaviour and periods of chaos ensue where valuable lower values of  $r^{(k)}$  are produced. It is these regions of chaos which have the greatest effect on improving the asymptotic rate of convergence. It must be noted, therefore, that the algorithm will only produce a desirable asymptotic rate of convergence if it is left to run for enough iterations to reach these lower values of  $r^{(k)}$ .

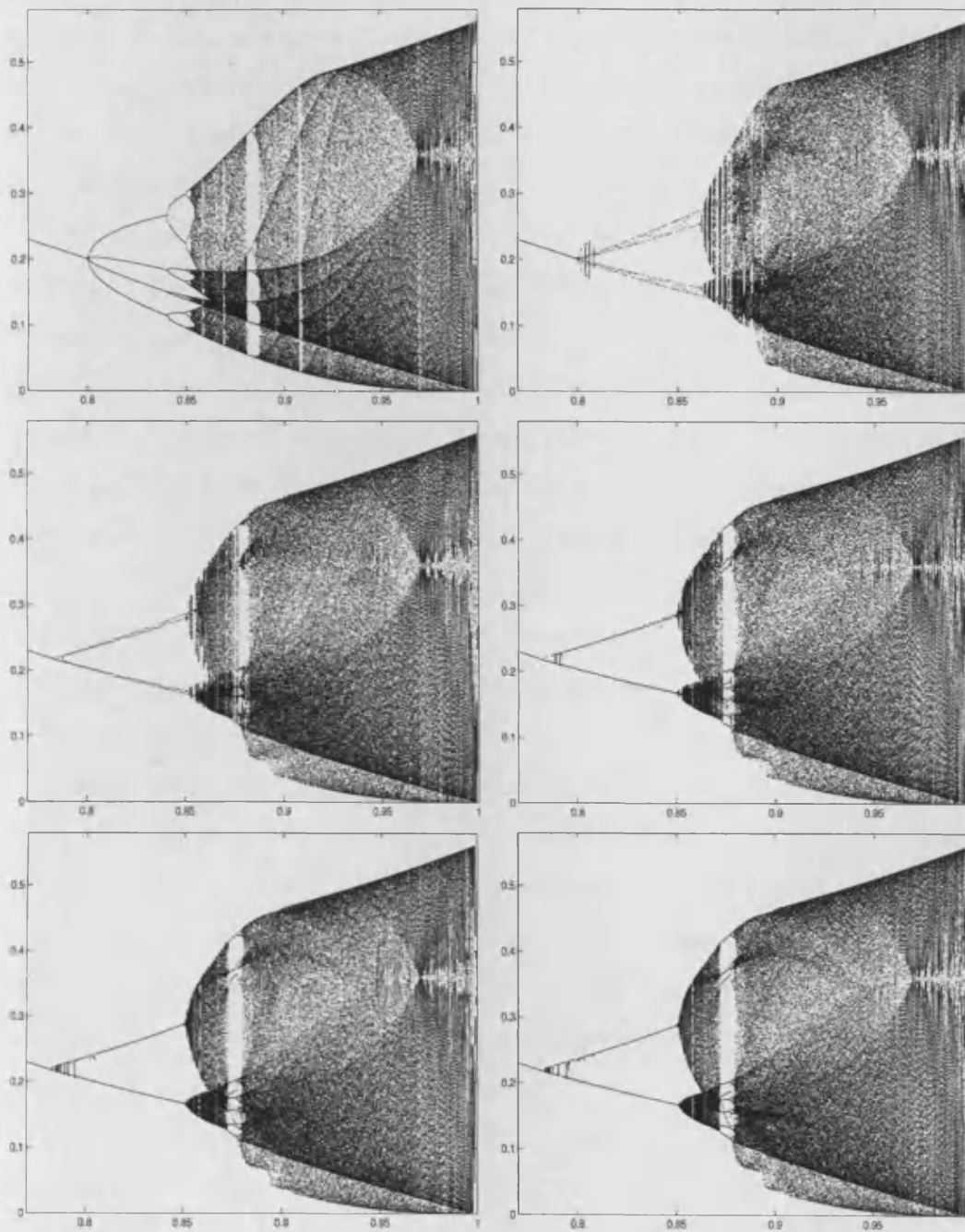


Figure 3.19: Attractors of  $r^{(k)}$  as a function of  $\gamma$  for, from top left to bottom right:  $d = 2, 4, 10, 20, 50, 100$ ;  $\rho = 4$ .

As  $\gamma$  approaches 1, the algorithm takes more and more iterations before it breaks out of its oscillatory phase and reaches the stage where more beneficial rates are achieved. In fact, when  $\gamma = 0.999$ , after 1000 iterations the algorithm is still exhibiting an oscillatory nature. Realistically, optimisation algorithms would not be run for as many iterations as would be required to reach superior rates so using a value of  $\gamma$  too close to 1 would not be wise.

Corresponding graphs for other values of  $\gamma$  can be found in Appendix C and show that for smaller values of  $\gamma$  little pattern can be found within the rates indicating the presence of true chaos. While the rates for these values of  $\gamma$  regularly exceed 1, there are no extremely bad rates and thus there are no remarkably good rates to follow, leading to a less varied picture on the whole. This can be seen from Figure 3.22 and Figure 3.23 where the distribution of  $r^{(k)}$  and  $-\ln(r^{(k)})$ , for various  $\gamma$ , clearly shows that, for the values of  $\gamma$  which relate to the better asymptotic rates of convergence, the rates  $r^{(k)}$  are much more varied than less advantageous values of  $\gamma$ . It can also be seen for those values of  $\gamma$  close to 1 that there are more extreme values of  $r^{(k)}$  than there are middle values, demonstrating further the algorithms tendency to oscillate from good rate to bad rate and back again.

Further analysis of the rate,  $r^{(k)}$ , is enabled by exploring patterns in the transition of  $r^{(k)}$  from one iteration to the next. Figure 3.24 depicts plots of the pairs  $(r^{(k)}, r^{(k+1)})$  for  $k = 5000, \dots, 10000$  with  $\gamma = 0.9995$  and shows, that for this value of  $\gamma$ , there is a clear relationship between the rates in consecutive iterations. Most importantly, it illustrates that if the rate at iteration  $k$  is large then the rate in the following iteration is close to zero. It is also seen here that the number of dimensions does not effect the shape of the plot.

Figure 3.25 shows the effect of  $\rho$  on the transition of  $r^{(k)}$ . It can be seen that there is a clear transitional pattern no matter what the condition number may be but for larger values of  $\rho$ , the situation becomes more extreme due to  $r^{(k)}$  becoming very large in some iterations; however the general pattern of a bad rate being followed by a good rate is still true.

Finally, Figure 3.26 shows how patterns in the transition from  $r^{(k)}$  to  $r^{(k+1)}$  change for different choices of  $\gamma$ . Chaos is clearly visible for some values of  $\gamma$ , making

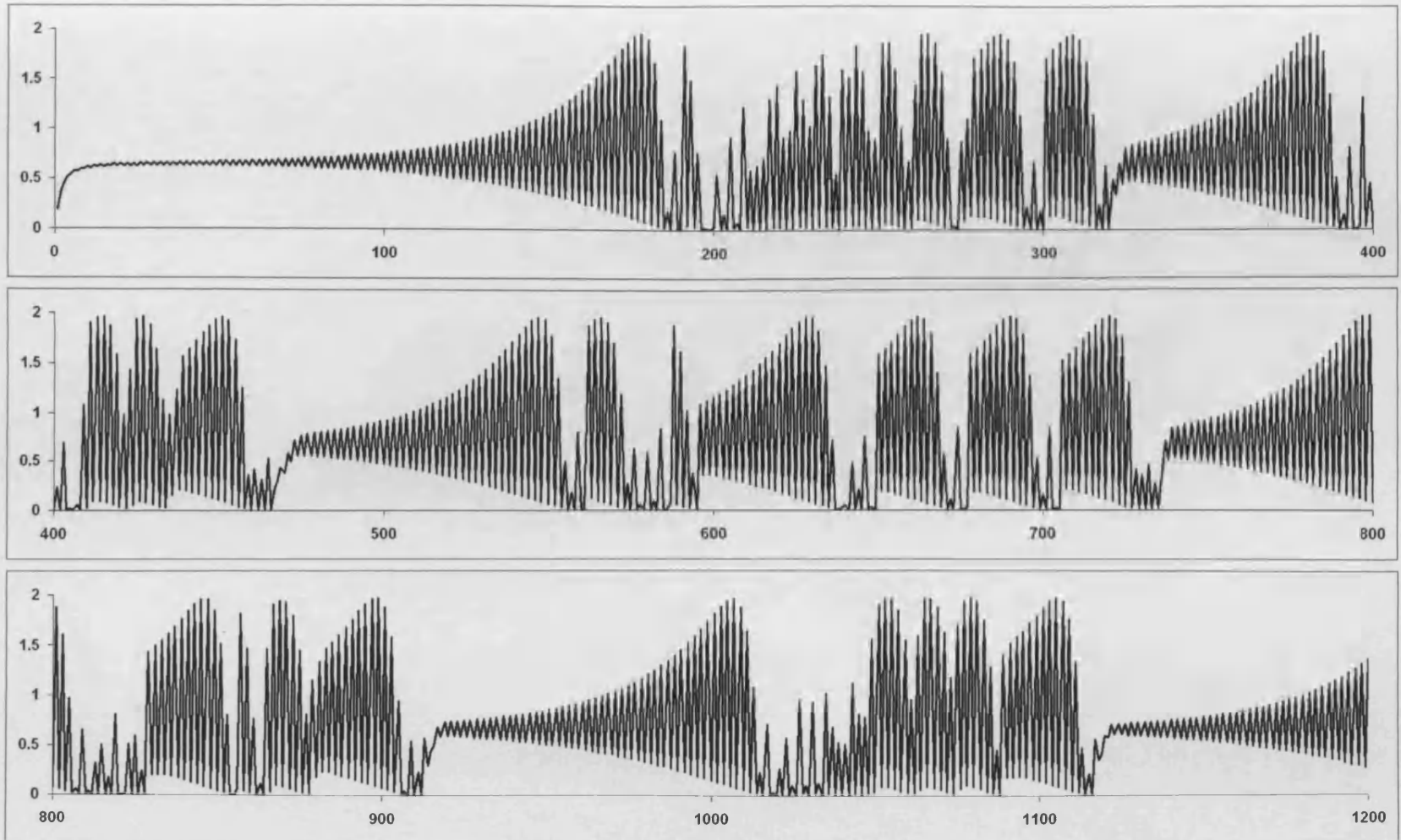


Figure 3.20: Rate,  $r^{(k)}$ , as a function of  $k$  for  $d = 100$ ,  $\rho = 10$  and  $\gamma = 0.985$ .



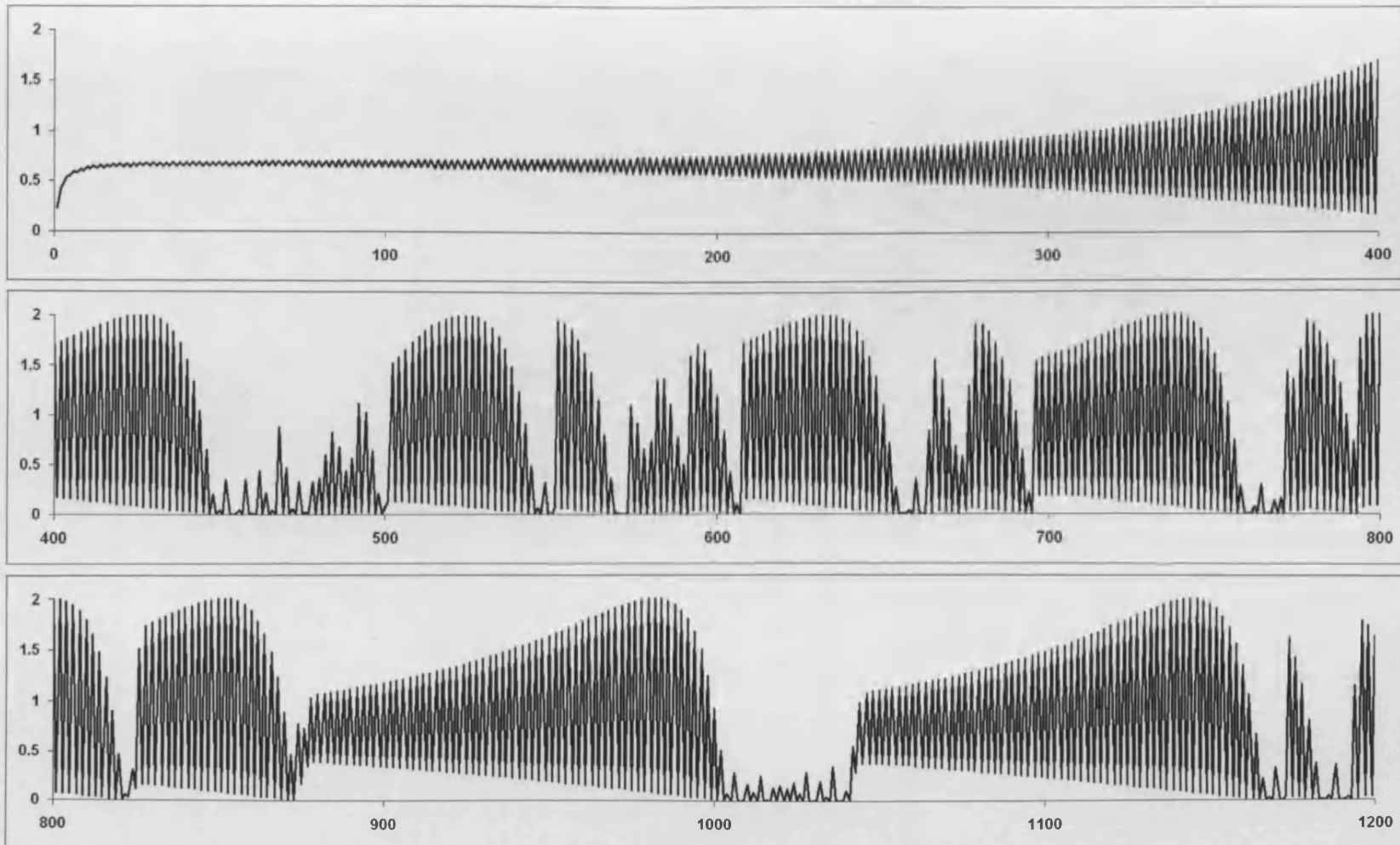


Figure 3.21: Rate,  $r^{(k)}$ , as a function of  $k$  for  $d = 100$ ,  $\rho = 10$  and  $\gamma = 0.995$ .

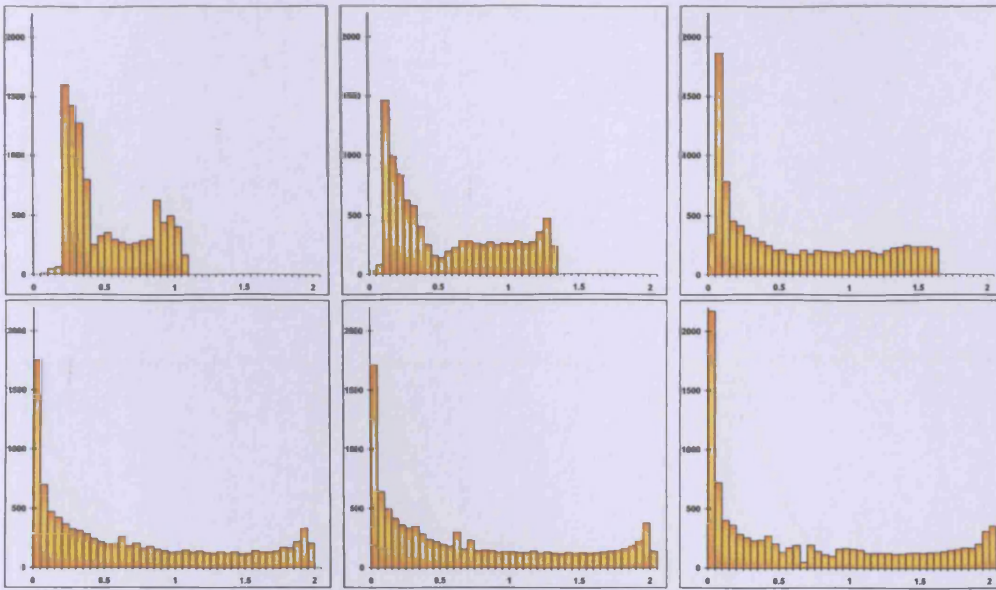


Figure 3.22: Histogram of  $r^{(k)}$  for, from top left to bottom right:  $\gamma = 0.7$ ,  $\gamma = 0.8$ ,  $\gamma = 0.9$ ,  $\gamma = 0.985$ ,  $\gamma = 0.995$ ,  $\gamma = 0.999$ ;  $\rho = 10$ ;  $d = 100$ ,  $k = 1, \dots, 10000$ .

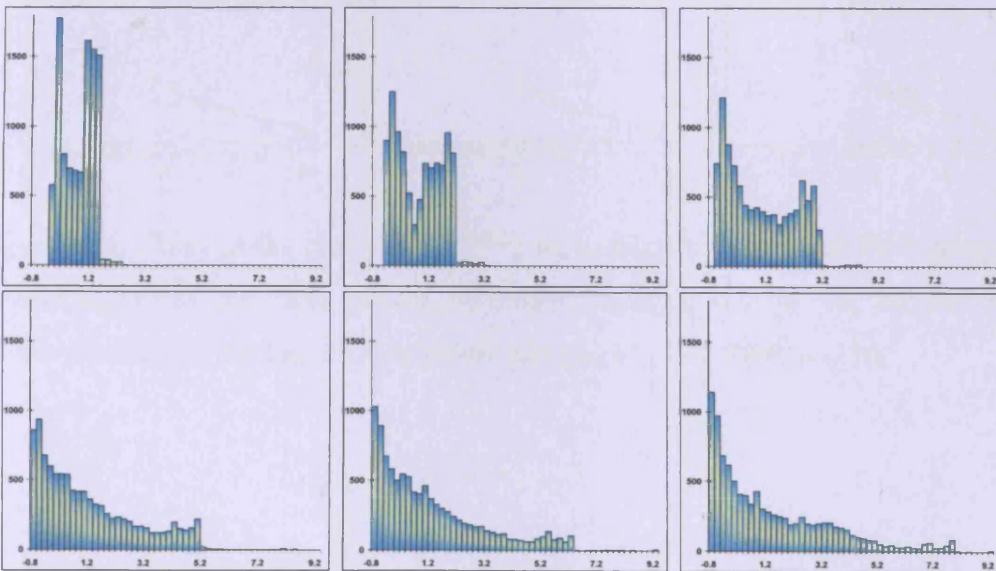


Figure 3.23: Histogram of  $-\ln(r^{(k)})$  for, from top left to bottom right:  $\gamma = 0.7$ ,  $\gamma = 0.8$ ,  $\gamma = 0.9$ ,  $\gamma = 0.985$ ,  $\gamma = 0.995$ ,  $\gamma = 0.999$ ;  $\rho = 10$ ;  $d = 100$ ,  $k = 1, \dots, 10000$ .

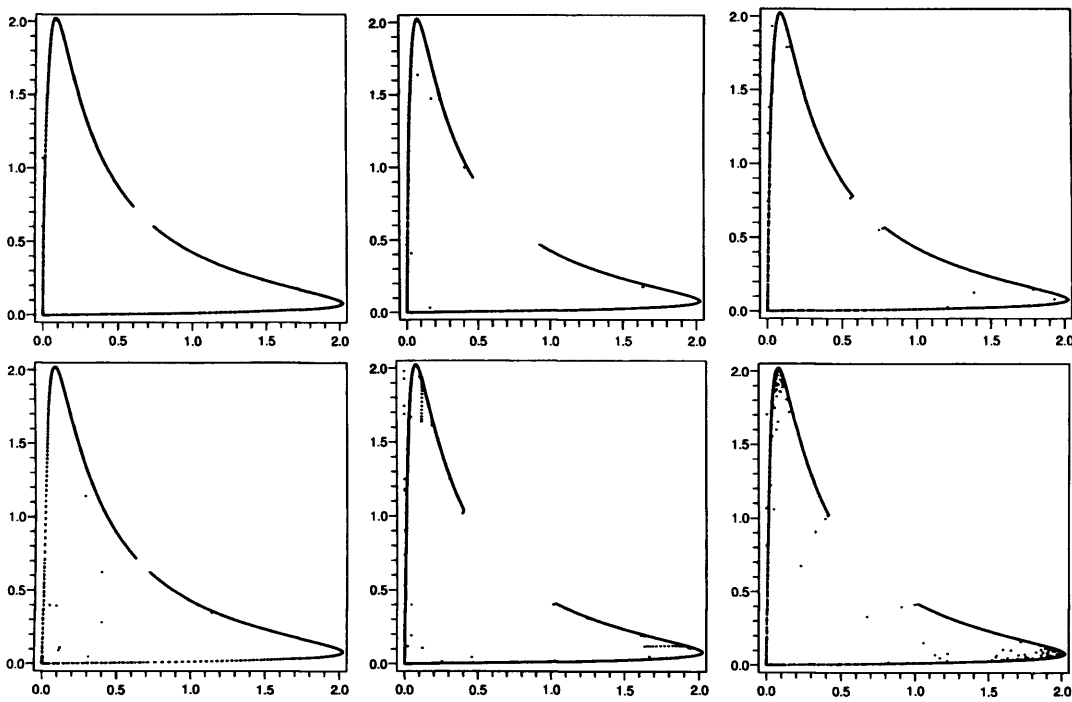


Figure 3.24: Plots of the pairs  $(r^{(k)}, r^{(k+1)})$  for a single trajectory of the  $\gamma$ -steepest descent algorithm for, from top left to bottom right:  $d = 2, 4, 10, 20, 50, 100$ . Points plotted are the last 5000 of 10000 iterations;  $\gamma = 0.9995, \rho = 10$ .

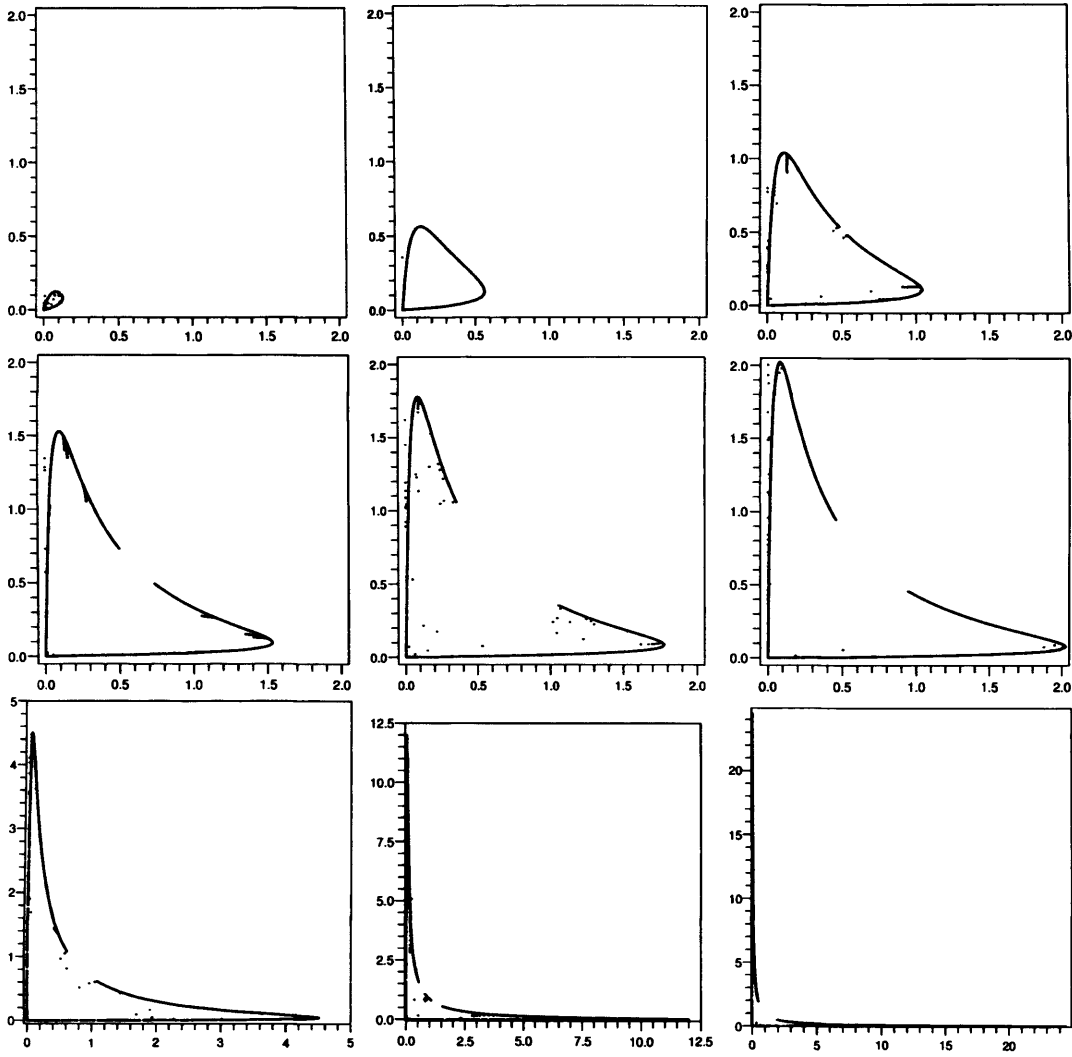


Figure 3.25: Plots of the pairs  $(r^{(k)}, r^{(k+1)})$  for, from top left to bottom right:  $\rho = 2, 4, 6, 8, 9, 10, 20, 50, 100$ . Points plotted are the last 5000 of 10000 iterations;  $\gamma = 0.9995, d = 100$ .

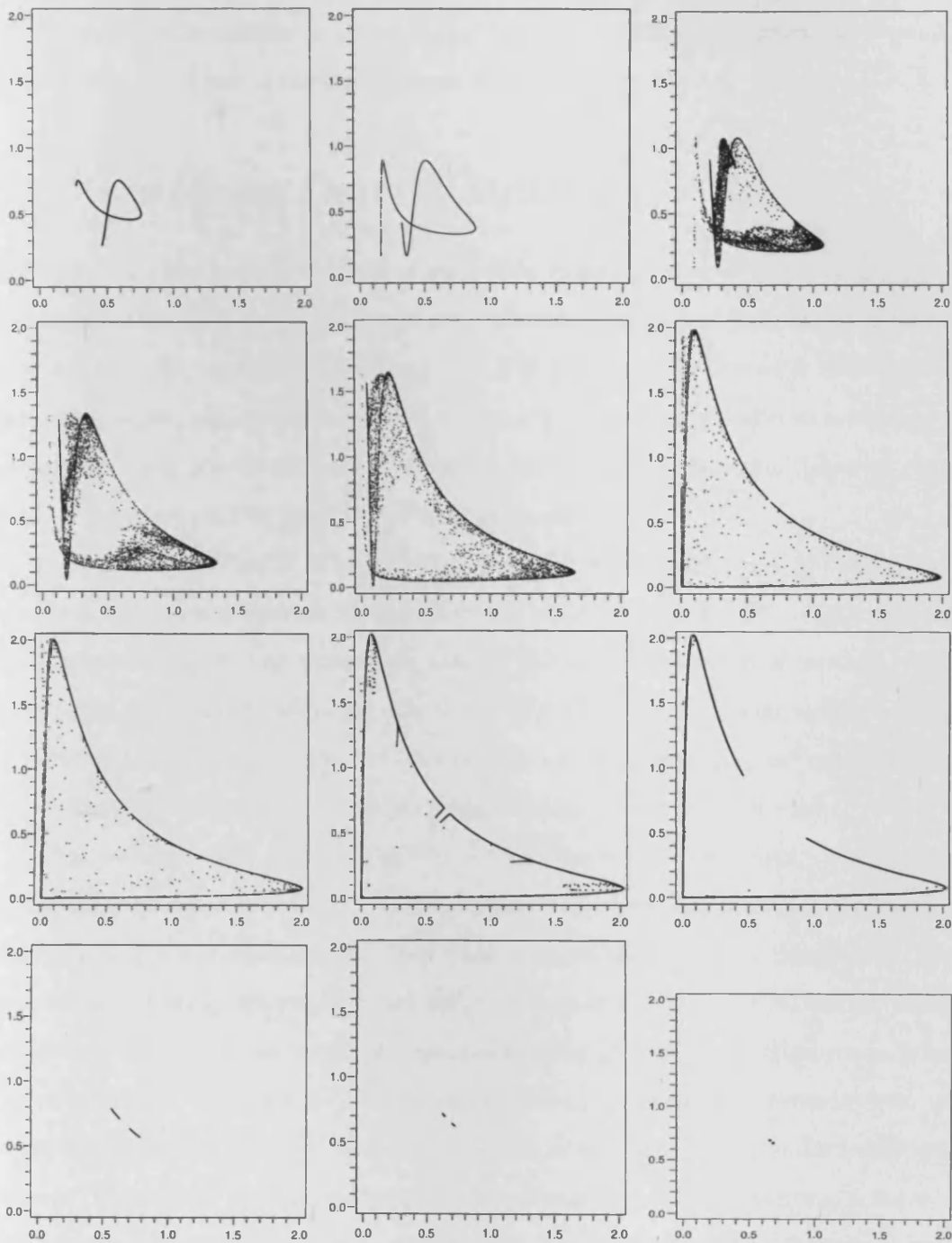


Figure 3.26: Plots of the pairs  $(r^{(k)}, r^{(k+1)})$  for, from top left to bottom right:  $\gamma = 0.5, 0.6, 0.7, 0.8, 0.9, 0.99, 0.995, 0.999, 0.9995, 0.9999, 0.99995, 1$ . Points plotted are the last 5000 of 10000 iterations;  $d = 100, \rho = 10$ .

the transition of rates less predictable. For values of  $\gamma$  approaching 1 the attractors of  $r^{(k)}$  diminish in number and the behaviour of the rates approaches the 2-point cycle present in classical steepest descent (i.e. when  $\gamma = 1$ ).

### 3.3 $\gamma$ -Steepest Descent Summary

In conclusion, the  $\gamma$ -steepest descent algorithm looks to be a fairly promising gradient algorithm with the best asymptotic rates of convergence demonstrated when the value of the relaxation coefficient  $\gamma$  is just less than 1. As with the original steepest descent algorithm, increasing the condition number  $\rho$  leads to a worsened asymptotic rate of convergence. The dimensionality of the problem, however, has little or no effect on this rate from  $d \cong 10$  onwards.

If  $1 < \gamma < 2M/(m + M)$  or  $2m/(m + M) < \gamma < 4mM/(m + M)^2$  then the gradient algorithm converges to the minimum point  $x^*$  with an asymptotic rate of convergence equal to the worst case rate of the steepest descent algorithm,  $R_{\text{ref}}$ . This corresponds to the situation where convergence to an optimum design occurs. Whilst this is bad from the point of view of creating an efficient gradient optimisation algorithm it is advantageous from the experimental design point of view.

The optimum value of  $\gamma$  (minimising the asymptotic rate of convergence) occurs in the interval  $4mM/(m + M)^2 < \gamma < 1$  where the algorithm does not necessarily converge to the optimum design. It is in this region that chaos is exhibited in the algorithm. Chaotic behaviour gives rise to the rate  $r^{(k)}$  in some iterations being greater than one i.e. at some iterations the point  $x^{(k)}$  moves further away from the minimum  $x^*$  meaning the algorithm is no longer monotonic. Nevertheless, at other iterations the rate  $r^{(k)}$  is much better than when  $\gamma = 1$  (standard steepest descent) and overall an improved average asymptotic rate of convergence is achieved. This non-monotonic behaviour is shared with, amongst others, the BB algorithm. If  $2M/(m + M) < \gamma < 2$  or  $0 < \gamma < 2m/(m + M)$  then the gradient algorithm converges with a rate worse than  $R_{\text{ref}}$  and if  $\gamma < 0$  or  $\gamma > 2$  then the algorithm diverges.

In Section 1.5 an algorithm, introduced in [58] and named the relaxed steepest

descent algorithm was described. This algorithm differs from the  $\gamma$ -steepest descent algorithm in so much that in this algorithm the standard steepest descent algorithm is relaxed with the inclusion of a random relaxation parameter  $\gamma^{(k)}$  which can vary between 0 and 2 at each iteration. This is in contrast to the  $\gamma$ -steepest descent algorithm where the relaxation parameter is fixed throughout all iterations. Figure 3.27 below shows that the  $\gamma$ -steepest descent algorithm with  $\gamma = 0.985$  compares very favourably with the relaxed steepest descent algorithm. There is, however, still room for improvement since the desirable rate  $R_{\min}$  has not yet been reached.

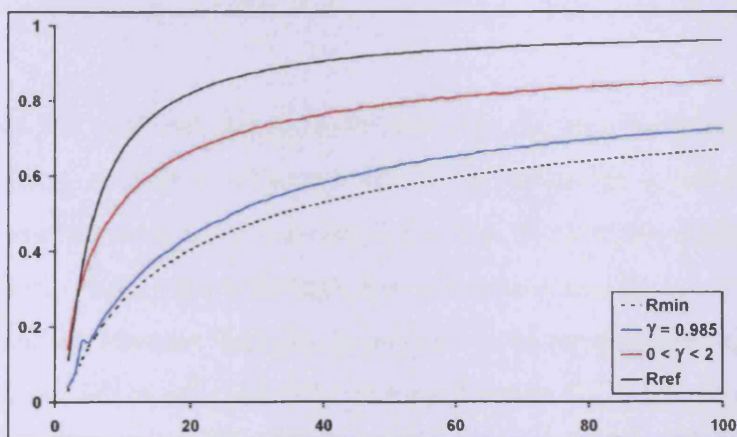


Figure 3.27: Average asymptotic rate of convergence as a function of  $\rho$  for the relaxed steepest descent algorithm and the  $\gamma$ -steepest descent algorithm with  $\gamma = 0.985$ ;  $d = 100$ .

# Chapter 4

## The $\beta$ -Root Algorithm

It was seen in the previous chapter that adapting the step length of the steepest descent algorithm, so that it includes a relaxation parameter  $\gamma$ , can generate much improved rates of convergence if  $\gamma$  is selected wisely. A value of  $\gamma$  slightly less than 1 is recommended. This method of adaptation is however, merely one of many ways in which the standard steepest descent algorithm can be modified. Keeping the same descent direction,  $\delta^{(k)} = g^{(k)}$ , but changing the formula for the step length,  $\alpha^{(k)}$ , in some way, gives rise to a whole host of possible new algorithms with the potential of having better asymptotic rate of convergence than the original steepest descent algorithm. In this chapter a particular family of algorithms, all with related step lengths, will be considered. These algorithms will collectively be referred to as the  $\beta$ -root algorithm. Moreover, it will emerge that the steepest descent algorithm is a particular case of this more generalised family of algorithms.

### 4.1 The Square Root Algorithm

The first algorithm in the  $\beta$ -root family to be considered in this chapter is that with step length

$$\sqrt{\frac{(g^{(k)}, g^{(k)})}{(A^2 g^{(k)}, g^{(k)})}}. \quad (4.1)$$



This algorithm shall be referred to individually as the square root algorithm and has the formula

$$x^{(k+1)} = x^{(k)} - \sqrt{\frac{(g^{(k)}, g^{(k)})}{(A^2 g^{(k)}, g^{(k)})}} g^{(k)}.$$

As was discussed in Chapter 1 the square root step length (4.1) can be seen to be exactly the step length (1.19), suggested by Dai in [20], since

$$\alpha_{OPT2}^{(k)} = \frac{\|g^{(k)}\|}{\|Ag^{(k)}\|} = \frac{\sqrt{(g^{(k)}, g^{(k)})}}{\sqrt{(Ag^{(k)}, Ag^{(k)})}} = \sqrt{\frac{(g^{(k)}, g^{(k)})}{(A^2 g^{(k)}, g^{(k)})}}.$$

### 4.1.1 The Renormalised Square Root Algorithm

As was the case with the steepest descent algorithm, the best way to analyse the asymptotic convergence rate of the square root algorithm is by first converting the algorithm to its renormalised form.

The step length of the square root algorithm can be re-written in terms of moments as

$$\alpha^{(k)} = \frac{1}{\sqrt{\mu_2}} \quad (4.2)$$

and, by substituting (4.2) into the general updating formula for renormalised gradient algorithms defined in (2.9), it can be seen that the updating formula for the renormalised square root algorithm takes the following form

$$\xi_i^{(k+1)} = \frac{(1 - \lambda_i/\sqrt{\mu_2})^2}{2(1 - \mu_1/\sqrt{\mu_2})} \xi_i^{(k)} \quad \text{for } i = 1, \dots, d. \quad (4.3)$$

The rate,  $r^{(k)}$ , for the square root algorithm can be written as

$$r^{(k)} = \frac{(g^{(k+1)}, g^{(k+1)})}{(g^{(k)}, g^{(k)})} = 2 \left( 1 - \frac{(Ag^{(k)}, g^{(k)})}{(g^{(k)}, g^{(k)})} \sqrt{\frac{(g^{(k)}, g^{(k)})}{(A^2 g^{(k)}, g^{(k)})}} \right) = 2 \left( 1 - \frac{\mu_1}{\sqrt{\mu_2}} \right)$$

and corresponds exactly to the denominator of the updating formula (4.3).

In a similar manner to the steepest descent algorithm, the renormalised square root algorithm corresponds exactly to a multiplicative algorithm for constructing optimal experimental designs. In this case, the optimality criterion to be maximised,

$$\Phi(\xi) = \sqrt{\mu_0 \mu_2} - \mu_1 = \sqrt{\mu_2} - \mu_1,$$



can be seen as an extension of the  $D$ -optimality criterion where the determinant of the matrix

$$M = \begin{pmatrix} \sqrt{\mu_0} & \sqrt{\mu_1} \\ \sqrt{\mu_1} & \sqrt{\mu_2} \end{pmatrix}$$

is taken in place of the determinant of (2.17). The multiplicative algorithm,

$$\xi'(x) = \frac{\varphi(x, \xi) - c(\xi)}{b(\xi)} \xi(x),$$

formed from the square root criterion benefits from the relatively simple formulae:

$$\overset{\circ}{\Phi}(\xi) = \begin{pmatrix} \frac{1}{2}\sqrt{\frac{\mu_2}{\mu_0}} & -\frac{1}{2} \\ -\frac{1}{2} & \frac{1}{2}\sqrt{\frac{\mu_0}{\mu_2}} \end{pmatrix},$$

$$\varphi(x, \xi) = f^T(x) \overset{\circ}{\Phi}(\xi) f(x) = \frac{\sqrt{\mu_2}}{2\sqrt{\mu_0}} - x + \frac{\sqrt{\mu_0}}{2\mu_2} x^2 = \frac{\sqrt{\mu_2}}{2} \left(1 - \frac{x}{\sqrt{\mu_2}}\right)^2,$$

$$c = \min_x \varphi(x, \xi) = 0 \quad \text{and} \quad b = \text{tr} \left[ M(\xi) \overset{\circ}{\Phi}(\xi) \right] - c(\xi) = \Phi(\xi) = \sqrt{\mu_2} - \mu_1.$$

#### 4.1.2 Asymptotic Rate of Convergence

Figure 4.1 shows the average asymptotic rate of convergence of the square root algorithm as a function of  $\rho$  (left) and as a function of  $d$  (right). As with the  $\gamma$ -steepest descent algorithm, the rate of convergence of the square root algorithm seems not to depend on the number of dimensions of the problem, shown in the right hand graph by the constant rate. The average asymptotic rate of convergence will, however, worsen with an increase in  $\rho$ , as seen in the left hand graph. In fact, the average asymptotic rate of convergence of the square root algorithm is exactly the worst rate of the steepest descent algorithm,  $R_{\text{ref}}$  defined in (1.13), indicating that this algorithm is no better than the standard steepest descent algorithm, in fact, on average it is slightly worse than the steepest descent algorithm. The fact that the square root algorithm converges with an asymptotic rate of convergence equal to  $R_{\text{ref}}$  also indicates that the corresponding multiplicative algorithm for constructing optimal designs will converge to an optimum design. The proof is a particular case of Theorem 4.2.2 in the next section.

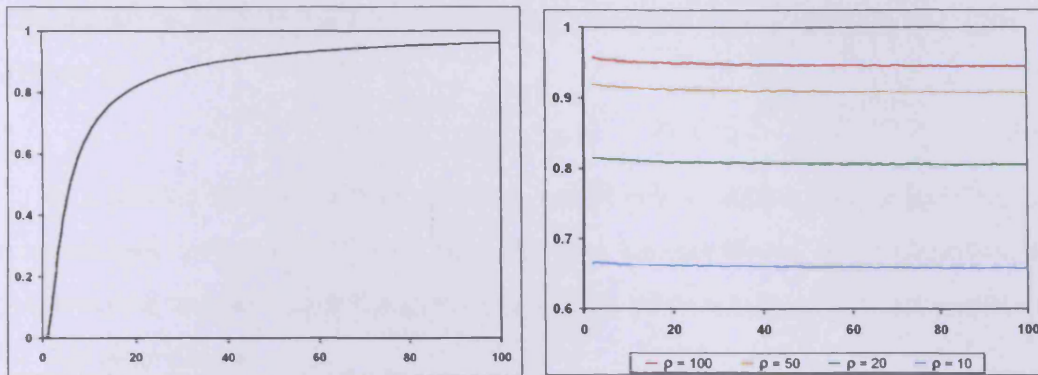


Figure 4.1: Asymptotic rate of convergence as a function of (left)  $\rho$  with  $d = 100$  and (right)  $d$  with various  $\rho$ ; eigenvalues equally distributed.)

## 4.2 The $\gamma$ -Square Root Algorithm

In the previous section, it was shown that the square root algorithm does not improve upon the asymptotic rate of convergence of the original steepest descent algorithm. In an attempt to produce a gradient algorithm with a faster asymptotic rate of convergence, the square root step length can be multiplied by a relaxation parameter,  $\gamma$ , in the same way the steepest descent algorithm was generalised in Chapter 3. The generalised square root step length, which from now on shall be referred to as the  $\gamma$ -square root step length, thus becomes

$$\alpha^{(k)} = \gamma \sqrt{\frac{(g^{(k)}, g^{(k)})}{(A^2 g^{(k)}, g^{(k)})}}. \quad (4.4)$$

If  $\gamma = 1$  the step length (4.4) is exactly the step length of the original square root algorithm.

In order to create the renormalised version of the  $\gamma$ -square root algorithm it is noted that the step length (4.4) can be represented in terms of the moments of  $\xi$  as

$$\alpha^{(k)} = \gamma / \sqrt{\mu_2}$$

and the rate at iteration  $k$  for the algorithm can be formulated as:

$$r^{(k)} = \frac{(g^{(k+1)}, g^{(k+1)})}{(g^{(k)}, g^{(k)})} = 1 - 2\gamma \frac{(Ag^{(k)}, g^{(k)})}{(g^{(k)}, g^{(k)})} \sqrt{\frac{(g^{(k)}, g^{(k)})}{(A^2 g^{(k)}, g^{(k)})}} + \gamma^2 = 1 - 2\gamma \frac{\mu_1}{\sqrt{\mu_2}} + \gamma^2. \quad (4.5)$$

The updating formula for the renormalised  $\gamma$ -square root algorithm can thus be written as

$$\xi_i^{(k+1)} = \frac{(1 - \lambda_i \gamma / \sqrt{\mu_2})^2}{1 - 2\gamma\mu_1 / \sqrt{\mu_2} + \gamma^2} \xi_i^{(k)} \quad \text{for } i = 1, \dots, d. \quad (4.6)$$

In a similar manner to the standard renormalised square root algorithm, the renormalised  $\gamma$ -square root algorithm can also be considered as an algorithm for constructing optimal experimental designs. The corresponding optimality criterion for this algorithm is

$$\Phi(\xi) = \gamma\sqrt{\mu_0\mu_2} - \mu_1 = \gamma\sqrt{\mu_2} - \mu_1. \quad (4.7)$$

This is a further generalisation of the  $D$ -optimality criterion where the determinant

$$\Phi(\xi) = \det \left[ \begin{pmatrix} \sqrt{\mu_0} & \sqrt{\mu_1} \\ \sqrt{\mu_1} & \sqrt{\mu_2} \end{pmatrix} + \begin{pmatrix} \tau & 0 \\ 0 & 0 \end{pmatrix} \right],$$

with  $\tau = \gamma - 1$ , is taken as the criterion in place of the determinant of (2.17).

For this criterion

$$\overset{\circ}{\Phi}(\xi) = \begin{pmatrix} \frac{\gamma\sqrt{\mu_2}}{2} & -\frac{1}{2} \\ -\frac{1}{2} & \frac{\gamma}{2\sqrt{\mu_2}} \end{pmatrix}, \quad \varphi(x, \xi) = \frac{\gamma\sqrt{\mu_2}}{2} - x + \frac{\gamma x^2}{2\sqrt{\mu_2}},$$

$$c = \frac{\mu_2}{2\gamma}(\gamma^2 - 1), \quad \text{and} \quad b = \sqrt{\mu_2} \left( \frac{\gamma}{2} + \frac{1}{2\gamma} \right) - \mu_1.$$

### 4.2.1 Concavity of the Optimality Criterion

The  $\gamma$ -square root optimality criterion (4.7) can be shown to be a concave functional for all designs  $\xi$  in a similar fashion to the way in which the  $\gamma$ -steepest descent optimality criterion was shown to be concave in Theorem 3.1.1.

**Theorem 4.2.1.** *The functional,  $\Phi(\xi) = \gamma\sqrt{\mu_2} - \mu_1$ , is a concave functional provided  $\gamma \geq 0$ .*

*Proof.* To prove concavity it is necessary and sufficient to show that

$$\frac{\partial^2}{\partial a^2} \Phi(\xi_a) \leq 0$$

holds for all  $\xi_a = (1 - a)\xi + a\xi'$ , where  $\xi, \xi'$  are any two designs and  $0 \leq a \leq 1$ .

The first derivatives of  $\Phi(\xi_a) = \gamma\sqrt{\mu_2(\xi_a)} - \mu_1(\xi_a)$  are

$$\frac{\partial\Phi(\xi_a)}{\partial\mu_1(\xi_a)} = -1, \quad \frac{\partial\Phi(\xi_a)}{\partial\mu_2(\xi_a)} = \frac{\gamma}{2\sqrt{\mu_2(\xi_a)}}.$$

Substituting these values into the general formula for the second derivative (3.4) gives

$$\begin{aligned} \frac{\partial^2}{\partial a^2}\Phi(\xi_a) &= \frac{\partial}{\partial a} \left[ -(\mu_1(\xi') - \mu_1(\xi)) + \frac{\gamma}{2\sqrt{\mu_2(\xi_a)}}(\mu_2(\xi') - \mu_2(\xi)) \right] \\ &= -\frac{\gamma}{4(\mu_2(\xi_a))^{3/2}}(\mu_2(\xi') - \mu_2(\xi))^2 \leq 0 \quad \forall \gamma \geq 0. \end{aligned}$$

The necessary and sufficient condition that the second derivative  $\partial^2\Phi(\xi)/\partial a^2$  is negative has been met provided  $\gamma \geq 0$  and thus for  $\gamma \geq 0$  the optimality criterion  $\Phi(\xi) = \gamma\mu_2 - \mu_1^2$  is a concave functional. This implies that the optimality criterion for the standard square root algorithm is also a concave functional.  $\square$

## 4.2.2 Convergence to an Optimum Design

**Theorem 4.2.2.** *Let  $\xi^{(0)}$  be any non-degenerate probability measure with support  $\{\lambda_1, \dots, \lambda_d\}$  and let the sequence of probability measures  $\{\xi^{(k)}\}$  be defined via the updating formula (4.6). Then the sequence*

$$\Phi^{(k)} = \Phi(\xi^{(k)}) = \gamma\sqrt{\mu_2(\xi^{(k)})} - \mu_1(\xi^{(k)})$$

*monotonically increases and converges for  $\gamma \geq 1$ ; that is  $\Phi^{(0)} \leq \Phi^{(1)} \leq \dots \leq \Phi^{(k)} \leq \dots$  and the limit  $\Phi^*(\xi^{(0)}) = \lim_{k \rightarrow \infty} \Phi^{(k)}$  exists.*

*Proof.* Note that  $\Phi^{(k)} \geq 0$  for any  $\xi$  in view of the Cauchy-Schwartz inequality. The sequence  $\Phi^{(k)}$  is non-decreasing if  $\Phi^{(k+1)} - \Phi^{(k)} \geq 0$  holds for any distribution  $\xi$ . If the distribution  $\xi$  is degenerate (that is, has mass 1 at one point) then  $\Phi^{(k+1)} = \Phi^{(k)}$  and the statement of the theorem holds. It is assumed below that  $\xi$  is non-degenerate. In particular this implies  $\mu_2 > \mu_1^2$ .

The problem can be formulated as

$$\begin{aligned} \Phi^{(k+1)} - \Phi^{(k)} \geq 0 &\iff \left( \gamma\sqrt{\mu'_2} - \mu'_1 \right) - \left( \gamma\sqrt{\mu_2} - \mu_1 \right) \geq 0 \\ &\iff \gamma^2\mu'_2 - \left( \gamma\sqrt{\mu_2} + \mu'_1 - \mu_1 \right)^2 \geq 0, \end{aligned}$$

where  $\mu'_1$  and  $\mu'_2$  are respectively the first and second moments associated with the distribution  $\xi^{(k+1)}$  respectively. The inequality that will be proved to hold true is

$$\gamma^2 \mu'_2 - \left( \gamma \sqrt{\mu_2} + \mu'_1 - \mu_1 \right)^2 \geq 0. \quad (4.8)$$

As  $(\mu_2 - \mu_1)$  and  $\mu'_1$  are both positive, this inequality is equivalent to the fact that  $\Phi^{(k)}$  are non-decreasing.

It is possible to express the moments of  $\xi^{(k+1)}$  through the moments of  $\xi = \xi^{(k)}$ .

First it is verified that  $\mu'_0 = \sum \xi_i^{(k+1)} = 1$ . Indeed,

$$\begin{aligned} \mu'_0 &= \sum_{i=1}^d \xi_i^{(k+1)} = \frac{1}{1 - 2\gamma\mu_1/\sqrt{\mu_2} + \gamma^2} \sum_{i=1}^d \left( 1 - \lambda_i \frac{\gamma}{\sqrt{\mu_2}} \right)^2 \xi_i \\ &= \frac{1}{1 - 2\gamma\mu_1/\sqrt{\mu_2} + \gamma^2} \left[ \sum_{i=1}^d \xi_i + 2 \frac{\gamma}{\sqrt{\mu_2}} \sum_{i=1}^d \lambda_i \xi_i + \frac{\gamma^2}{\mu_2} \sum_{i=1}^d \lambda_i^2 \xi_i \right] \\ &= \frac{1}{1 - 2\gamma\mu_1/\sqrt{\mu_2} + \gamma^2} \left[ 1 - 2\gamma \frac{\mu_1}{\sqrt{\mu_2}} + \gamma^2 \right] = 1. \end{aligned}$$

In a similar way

$$\mu'_1 = \sum_{i=1}^d \lambda_i \xi_i^{(k+1)} = \frac{1}{1 - 2\gamma\mu_1/\sqrt{\mu_2} + \gamma^2} \left[ \mu_1 - 2\gamma\sqrt{\mu_2} + \gamma^2 \frac{\mu_3}{\mu_2} \right]$$

and

$$\mu'_2 = \sum_{i=1}^d \lambda_i^2 \xi_i^{(k+1)} = \frac{1}{1 - 2\gamma\mu_1/\sqrt{\mu_2} + \gamma^2} \left[ \mu_2 - 2\gamma \frac{\mu_3}{\sqrt{\mu_2}} + \gamma^2 \frac{\mu_4}{\mu_2} \right].$$

The left-hand side of (4.8) can be represented as

$$\gamma^2 \mu'_2 - \left( \gamma \sqrt{\mu_2} + \mu'_1 - \mu_1 \right)^2 = \frac{U}{W} \quad (4.9)$$

with,

$$\begin{aligned} U &= \gamma^2 (-8\mu_2^{7/2} \gamma \mu_1 + 3\mu_2^4 \gamma^2 + 10\gamma^2 \mu_3 \mu_2^2 \mu_1 - 4\gamma^3 \mu_3 \mu_2^{5/2} + \mu_2^2 \gamma^2 \mu_4 \\ &\quad - 2\mu_2^{3/2} \gamma^3 \mu_4 \mu_1 + \mu_2^2 \gamma^4 \mu_4 - 4\mu_2^{3/2} \gamma \mu_3 \mu_1^2 - 4\mu_1^4 \mu_2^2 - 13\gamma^2 \mu_2^3 \mu_1^2 \\ &\quad + 6\gamma^3 \mu_2^{7/2} \mu_1 + 12\gamma \mu_2^{5/2} \mu_1^3 + 4\mu_2^3 \mu_1^2 - \mu_2 \gamma^2 \mu_3^2 - \gamma^4 \mu_2^4) \end{aligned} \quad (4.10)$$

and

$$W = (\sqrt{\mu_2} - 2\gamma\mu_1 + \gamma^2 \sqrt{\mu_2})^2 \mu_2^2. \quad (4.11)$$

As  $\xi$  is non-degenerate,  $W$ , the denominator of (4.9), is always positive.

It thus remains to prove that  $U$ , the numerator of (4.9), is also non-negative. One way of proving that an expression is non-negative is to relate it to a variance of a random variable, as it is known that variances are always non-negative. Consider the variance

$$V = \text{var}(a\eta + b\eta^2)$$

where  $\eta$  is a random variable with distribution  $\xi$  and  $a, b$  are some parameters.

$$\begin{aligned} V = \text{var}(a\eta + b\eta^2) &= E(a\eta + b\eta^2)^2 - [E(a\eta + b\eta^2)]^2 \\ &= a^2 E(\eta^2) + 2ab E(\eta^3) + b^2 E(\eta^4) - (aE(\eta) + bE(\eta^2))^2 \\ &= a^2 \mu_2 + 2ab \mu_3 + b^2 \mu_4 - (a\mu_1 + b\mu_2)^2. \end{aligned}$$

Subtract the variance,  $V$ , from  $U$  and consider this as a function of  $a, b$  :

$$\begin{aligned} F(a, b) &= U - V = 3\mu_2^3 \gamma^2 - 8\mu_2^{5/2} \gamma \mu_1 + 10\gamma^2 \mu_3 \mu_2 \mu_1 - 4\gamma^3 \mu_3 \mu_2^{3/2} + \mu_2 \gamma^2 \mu_4 \\ &\quad - 2\sqrt{\mu_2} \gamma^3 \mu_4 \mu_1 + \mu_2 \gamma^4 \mu_4 + 6\gamma^3 \mu_2^{5/2} \mu_1 - 13\gamma^2 \mu_2^2 \mu_1^2 + 12\gamma \mu_2^{3/2} \mu_1^3 + 4\mu_2^2 \mu_1^2 \\ &\quad - \gamma^2 \mu_3^2 - \gamma^4 \mu_2^3 - 4\mu_1^4 \mu_2 - 4\sqrt{\mu_2} \gamma \mu_3 \mu_1^2 \\ &\quad - a^2 \mu_2 - 2ab \mu_3 - b^2 \mu_4 + a^2 \mu_1^2 + 2a\mu_1 b \mu_2 + b^2 \mu_2^2. \end{aligned}$$

If  $a, b$  can be selected so that  $F(a, b) = 0$  this would mean that  $U = V$ . First, choose  $b$  to eliminate the  $\mu_4$  term:

$$b = b_0 = \gamma \sqrt{\mu_2 - 2\gamma \sqrt{\mu_2} \mu_1 + \gamma^2 \mu_2} ;$$

note that  $b_0 > 0$  for  $\gamma \geq 1$ . Substitute this value into  $F(a, b)$ , then solve  $F(a, b_0) = 0$  with respect to  $a$ . Note that  $F(a, b_0)$  is a quadratic function in  $a$ .

Let  $D$  be the discriminant of  $F(a, b_0)$ . (For a quadratic function  $Ax^2 + Bx + C$  in  $x$  the discriminant is  $D = B^2 - 4AC$ ; the condition  $D \geq 0$  guarantees that real roots of this equation exist). In this particular case, the discriminant is

$$\begin{aligned} D &= 4\mu_2^4 \gamma^2 + 8\gamma^2 \mu_3 \mu_2^2 \mu_1 - 8\mu_2^{7/2} \gamma \mu_1 - 4\gamma^3 \mu_3 \mu_2^{5/2} - 8\mu_1^4 \mu_2^2 + 4\mu_2^3 \mu_1^2 \\ &\quad - 6\mu_1^3 \gamma^3 \mu_2^{5/2} - 16\gamma^2 \mu_2^3 \mu_1^2 + 4\gamma^3 \mu_2^{7/2} \mu_1 + 20\gamma \mu_2^{5/2} \mu_1^3 - 4\mu_2^{3/2} \gamma \mu_3 \mu_1^2 \\ &\quad - 10\mu_1^3 \gamma^2 \mu_3 \mu_2 + 8\mu_1^2 \gamma^3 \mu_3 \mu_2^{3/2} + 4\mu_1^4 \sqrt{\mu_2} \gamma \mu_3 + 13\mu_1^4 \gamma^2 \mu_2^2 + \mu_1^2 \gamma^2 \mu_3^2 \\ &\quad + \mu_1^2 \gamma^4 \mu_2^3 - 12\mu_1^5 \gamma \mu_2^{3/2} + 4\mu_1^6 \mu_2 + \mu_3^2 \mu_2 \gamma^4 - 2\mu_1 \mu_2^2 \mu_3 \gamma^4 - 2\mu_3^2 \sqrt{\mu_2} \gamma^3 \mu_1. \end{aligned}$$

If  $D \geq 0$  then there is always a solution of the equation  $F(a, b) = 0$  in  $a$  and  $b$  meaning that there are always values of  $a, b$  such that

$$U = \text{var}(a\eta + b\eta^2) .$$

Split  $D$  into two parts, one with fractional powers of  $\mu_2$  and the other with whole powers of  $\mu_2$  . We then have  $D = D_1 + D_2$  with

$$D_1 = 2\gamma\sqrt{\mu_2}(2\mu_2^2 - 3\mu_2\mu_1^2 + \mu_3\mu_1)(\mu_2\gamma^2\mu_1 - 2\mu_2\mu_1 + 2\mu_1^3 - \gamma^2\mu_3) \quad (4.12)$$

and

$$\begin{aligned} D_2 = & \mu_1^2\gamma^2\mu_3^2 - 2\mu_1\mu_2^2\mu_3\gamma^4 - 10\mu_1^3\gamma^2\mu_3\mu_2 + 4\mu_1^6\mu_2 + \mu_1^2\gamma^4\mu_2^3 + 13\mu_1^4\gamma^2\mu_2^2 \\ & + \mu_3^2\mu_2\gamma^4 + 4\mu_2^3\mu_1^2 + 4\mu_2^4\gamma^2 + 8\gamma^2\mu_3\mu_2^2\mu_1 - 16\gamma^2\mu_2^3\mu_1^2 - 8\mu_1^4\mu_2^2 . \end{aligned} \quad (4.13)$$

For the moment assume that  $D_2 \geq 0$ ,  $D_1 \leq 0$  (see end of proof for assumptions).

Then the fact that  $D$ , the discriminant, is non-negative, is equivalent to

$$|D_2| \geq |D_1| \quad \text{or} \quad (D_2)^2 \geq (D_1)^2 .$$

The difference of squares,  $(D_2)^2 - (D_1)^2$ , can be represented as a product of two squares as follows:

$$(D_2)^2 - (D_1)^2 = (\mu_1^2 - \gamma^2\mu_2)^2(4\mu_1^4\mu_2 - 8\mu_2^2\mu_1^2 - \gamma^2\mu_2^2\mu_1^2 + 2\gamma^2\mu_3\mu_2\mu_1 - \gamma^2\mu_3^2 + 4\mu_2^3)^2 \geq 0 .$$

This implies that  $(D_2)^2 - (D_1)^2 \geq 0$  and that  $|D_2| = |D_1|$  (that is,  $D = 0$ ) if and only if

$$(4\mu_1^4\mu_2 - 8\mu_2^2\mu_1^2 - \gamma^2\mu_2^2\mu_1^2 + 2\gamma^2\mu_3\mu_2\mu_1 - \gamma^2\mu_3^2 + 4\mu_2^3)^2 = 0$$

It will now be proved that  $D_1 \leq 0$ , where  $D_1$  is defined in (4.12). Consider each factor in (4.12) separately.

- (i) The factor  $\mu_1\mu_3 + 2\mu_2^2 - 3\mu_1^2\mu_2$  is non-negative since  $\mu_1\mu_3 \geq \mu_2^2$  by the Jensen inequality and therefore

$$\begin{aligned} \mu_1\mu_3 + 2\mu_2^2 - 3\mu_1^2\mu_2 & \geq 3\mu_2^2 - 3\mu_1^2\mu_2 = 3\mu_2(\mu_2 - \mu_1^2) \geq 0 \quad \text{implying} \quad \mu_1\mu_3 + \\ & 2\mu_2^2 - 3\mu_1^2\mu_2 \geq 0 . \end{aligned}$$



(ii) The factor  $\gamma^2\mu_3 - \gamma^2\mu_2\mu_1 + 2\mu_2\mu_1 - 2\mu_1^3$  is non-negative for  $\gamma \geq 1$  since  $\mu_3 \geq \mu_1^3$  by the Jensen inequality and therefore  $\gamma^2\mu_3 - \gamma^2\mu_2\mu_1 + 2\mu_2\mu_1 - 2\mu_1^3 = (\gamma^2\mu_3 - \mu_1^3) + \mu_1(\gamma^2\mu_2 + 2\mu_2 - \mu_1^2) \geq 0$

This implies that  $D_1 = -2\gamma\sqrt{\mu_2}(2\mu_2^2 - 3\mu_2\mu_1^2 + \mu_3\mu_1)(\gamma^2\mu_3 - \gamma^2\mu_2\mu_1 + 2\mu_2\mu_1 - 2\mu_1^3) \leq 0$ .

Finally consider  $D_2$ , defined in (4.13), and prove that  $D_2 > 0$ . Consider this expression as a quadratic equation in  $\mu_3$  and compute the roots. This is a quadratic equation of the form  $D_2 = A\mu_3^2 + B\mu_3 + C$  with

$$A = \gamma^2\mu_1^2 + \gamma^4\mu_2, \quad B = -2\gamma^2\mu_2\mu_1(\gamma^2\mu_2 + 5\mu_1^2 - 4\mu_2),$$

$$C = \mu_2(\mu_1^2\gamma^4\mu_2^2 + 13\mu_1^4\gamma^2\mu_2 - 16\gamma^2\mu_2^2\mu_1^2 + 4\mu_1^6 + 4\mu_2^3\gamma^2 + 4\mu_2^2\mu_1^2 - 8\mu_1^4\mu_2).$$

Let  $T$  be the discriminant of  $D_2$ :

$$T = B^2 - 4AC = -\mu_2(\mu_2 - \mu_1^2)^2(\gamma^2\mu_2 - \mu_1^2)^2 < 0.$$

Consequently  $D_2$  has no real roots and, as a result,  $D_2 > 0$  as required.  $\square$

It has been proven that a limit to the sequence  $\{\Phi(\xi^{(k)})\}$  exists for the  $\gamma$ -square root algorithm when  $\gamma \geq 1$ . It is therefore possible to find the limit of this sequence.

**Theorem 4.2.3.** *Let  $\xi^{(0)}$  be any non-degenerate probability measure with support  $\{\lambda_1, \dots, \lambda_d\}$  and let the sequence of probability measures  $\{\xi^{(k)}\}$  be defined via the updating formula (4.6). If  $\xi^{(0)}$  is such that  $\xi^{(0)}(\lambda_1) > 0$  and  $\xi^{(0)}(\lambda_d) > 0$ , then the limit  $\lim_{k \rightarrow \infty} \Phi^{(k)}$  of the sequence  $\Phi^{(k)} = \Phi(\xi^{(k)}) = \gamma\sqrt{\mu_2(\xi^{(k)})} - \mu_1(\xi^{(k)})$  does not depend on  $\xi^{(0)}$  and*

$$\lim_{k \rightarrow \infty} \Phi(\xi^{(k)}) = \Phi^* = \frac{\gamma^2}{4}(m + M) - \frac{mM}{m + M};$$

moreover, the sequence of probability measures  $\{\xi^{(k)}\}$  converges (as  $k \rightarrow \infty$ ) to the probability measure  $\xi^*$  supported at the points  $m$  and  $M$  with weights

$$\xi^*(m) = \frac{4M^2 - \gamma^2(m + M)^2}{4(M^2 - m^2)} \quad \text{and} \quad \xi^*(M) = \frac{\gamma^2(m + M)^2 - 4m^2}{4(M^2 - m^2)},$$

where  $\gamma \geq 1$ .

*Proof.* In a similar fashion to the proof of Theorem 3.1.3, the limiting point of the probability measure  $\xi$  can be found by solving the equation

$$\xi(x) = \frac{(1 - \gamma x / \sqrt{\mu_2})^2}{1 - 2\gamma\mu_1 / \sqrt{\mu_2} + \gamma^2} \xi(x),$$

for  $\xi(x)$  i.e. by finding a measure which is invariant under the  $\gamma$ -square root updating formula (4.6).

As was discussed in Section 2.3.3, an optimal design,  $\xi^*$  will be supported at the minimum and maximum eigenvalues  $m$  and  $M$ , so that

$$\xi^* = \left\{ \begin{array}{cc} m & M \\ p & q \end{array} \right\},$$

where  $q = 1 - p$ . The first and second moments of the limiting probability measure  $\xi^*$  are

$$\mu_1 = mp + M(1 - p) \quad \text{and} \quad \mu_2 = m^2p + M^2(1 - p)$$

respectively. Solving the equation

$$\xi(m) = \frac{(1 - \gamma m / \sqrt{\mu_2})^2}{1 - 2\gamma\mu_1 / \sqrt{\mu_2} + \gamma^2} \xi(m)$$

for  $\xi(m)$  gives a weight at  $m$  for  $\xi^*$  of

$$\xi^*(m) = \frac{4M^2 - \gamma^2(m + M)^2}{4(M^2 - m^2)}$$

and therefore the weight at  $M$  will be

$$\xi^*(M) = 1 - \xi^*(m) = \frac{\gamma^2(m + M)^2 - 4m^2}{4(M^2 - m^2)}.$$

Since  $\xi^*(m)$  and  $\xi^*(M)$  are weights they must be between 0 and 1, i.e.

$$0 \leq \frac{4M^2 - \gamma^2(m + M)^2}{4(M^2 - m^2)}, \frac{\gamma^2(m + M)^2 - 4m^2}{4(M^2 - m^2)} \leq 1$$

and it was shown in the previous Theorem that  $\gamma$  must be greater than 1, therefore

$$1 \leq \gamma \leq \frac{2M}{m + M}.$$

Substituting the weights of the probability measure  $\xi^*$  into  $\Phi(\xi^{(k)}) = \gamma\sqrt{\mu_2(\xi^{(k)})} - \mu_1(\xi^{(k)})$  gives

$$\Phi^* = \frac{\gamma^2}{4}(m + M) - \frac{mM}{m + M}$$

as required. □

It is possible to confirm that the rate  $r(\xi^*)$  associated with the optimum design is equal to  $R_{\text{ref}}$  by substituting the weight  $p^*$  into the equation for the rate of convergence (4.5) for the  $\gamma$ -square root algorithm. If  $\gamma > 2M/(m + M)$  then the probability measure becomes degenerate and the design is supported solely at the maximum eigenvalue  $M$ .

### 4.2.3 Speed of Convergence to the Optimum Design

The renormalised  $\gamma$ -square root algorithm has been shown to converge to an optimum design if the relaxation coefficient  $\gamma$  is in the range  $1 \leq \gamma \leq \frac{2\rho}{1+\rho}$ . The exact value of  $\gamma$  used could however have an effect on the speed with which the optimum design is reached. In order to verify whether or not this is the case, the same technique that was utilised to measure the speed of convergence for the  $\gamma$ -steepest descent algorithm has also been applied to the  $\gamma$ -square root algorithm. The results of this investigation can be found in Table 4.1.

Whilst it was shown in Theorem 4.2.2 and Theorem 4.2.3 that the  $\gamma$ -square root algorithm will converge to an optimum design if  $1 \leq \gamma \leq 2\rho/(1 + \rho)$ , it can be seen from Table 4.1 that the range of  $\gamma$  for which convergence occurs is actually more extensive than this. In fact there is only a small range of  $\gamma$  (the location of which depends on the condition number  $\rho$ ) within the interval  $[2/(1 + \rho), 2\rho/(1 + \rho)]$  for which convergence does not occur. This range of non-convergence can be identified in the table by the entries which show that a significantly reduced number of decimal places of accuracy were observed. Blank entries in the table indicate parameter values for which the  $\gamma$ -square root algorithm does not converge at all (due to the fact that  $\gamma < 2/(1 + \rho)$  or  $\gamma > 2\rho/(1 + \rho)$ ). The speed of convergence of the sequence  $\{r^{(k)}\}$  is approximately equal for all feasible values of  $\gamma$  except those values of  $\gamma$  close to regions where non-convergence occurs, there the convergence is slower.

### 4.2.4 Asymptotic Rate of Convergence

The average asymptotic rate of convergence of the  $\gamma$ -square root algorithm is shown in Figure 4.2 as a function of  $\gamma$  for various condition numbers. In concurrence with

$\gamma$	$\rho = 2$	$\rho = 4$	$\rho = 10$	$\rho = 19$	$\rho = 49$	$\rho = 99$
0.05	-	-	-	-	1	2
0.1	-	-	-	11	9	4
0.15	-	-	-	15	7	4
0.2	-	-	2	17	3	2
0.25	-	-	16	14	2	3
0.3	-	-	17	7	1	3
0.35	-	-	18	4	2	5
0.4	-	-	13	3	3	6
0.45	-	18	7	1	6	8
0.5	-	18	4	3	7	10
0.55	-	19	2	4	10	12
0.6	-	19	1	6	13	14
0.65	-	16	3	9	15	18
0.7	18	8	5	12	18	19
0.75	18	3	7	15	19	19
0.8	18	1	11	18	19	20
0.85	3	2	15	19	18	19
0.9	4	6	19	19	18	19
0.95	1	11	18	19	19	18
1	6	18	19	19	19	19
1.05	18	19	19	18	18	18
1.1	19	19	19	19	19	17
1.15	19	18	19	17	18	19
1.2	17	19	20	19	19	18
1.25	3	18	18	19	19	20
1.3	3	17	19	19	19	19
1.35	-	19	19	19	17	18
1.4	-	18	18	19	19	18
1.45	-	18	19	19	20	17
1.5	-	19	19	19	18	19
1.55	-	17	19	19	19	19
1.6	-	-	18	18	19	18
1.65	-	-	17	19	19	19
1.7	-	-	19	19	19	20
1.75	-	-	17	19	19	19
1.8	-	-	5	20	19	19
1.85	-	-	-	10	19	19
1.9	-	-	-	3	3	2
1.95	-	-	-	-	2	4

Table 4.1: Speed with which  $\{r^{(k)}\}$  converges to  $r^*$ , measured as the average number of decimal places of accuracy achieved by  $r^{(k)}$  after 200 iterations of the  $\gamma$ -square root algorithm;  $d = 100$ .

the findings of the previous section it can be seen that for the range  $2/(1 + \rho) \leq \gamma \leq 2\rho/(1 + \rho)$ , apart from a small region where a marginally better asymptotic rate of convergence occurs, the  $\gamma$ -square root algorithm demonstrates an average convergence rate equal to the worst rate of the steepest descent algorithm,  $R_{\text{ref}}$ .

With regards to the asymptotic rate of convergence of the algorithm, the range of  $\gamma$  of most interest is the, albeit small, interval for which the average asymptotic rate of convergence is slightly better than the worst rate of the steepest descent algorithm. In order to gain a greater understanding as to why this improvement in rate occurs, Figure 4.3 shows the attractors of the sequence  $\{r^{(k)}\}$  in the region where this improvement in asymptotic rate of convergence occurs. As was the case with the  $\gamma$ -steepest descent algorithm, the areas of improved asymptotic rate of convergence correspond to regions of  $\gamma$  where chaos is present in the algorithm. This improvement in asymptotic rate of convergence is, however, negligible when compared with the asymptotic rates of convergence achievable by the  $\gamma$ -steepest descent algorithm with  $\gamma$  slightly less than 1.

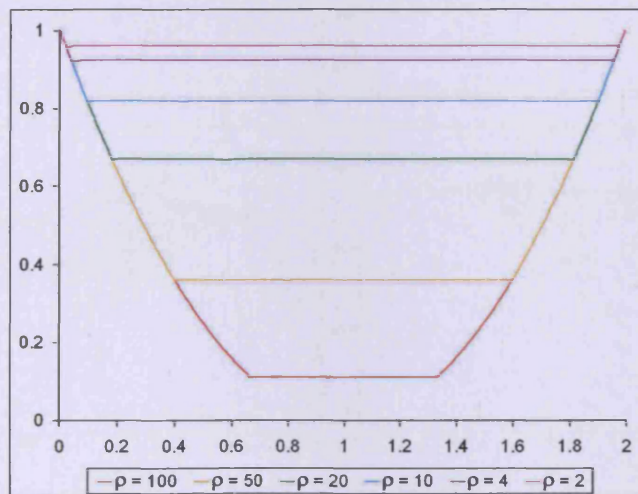


Figure 4.2: Average asymptotic rate of convergence as a function of  $\gamma$  for the  $\gamma$ -square root algorithm;  $d = 100$ .

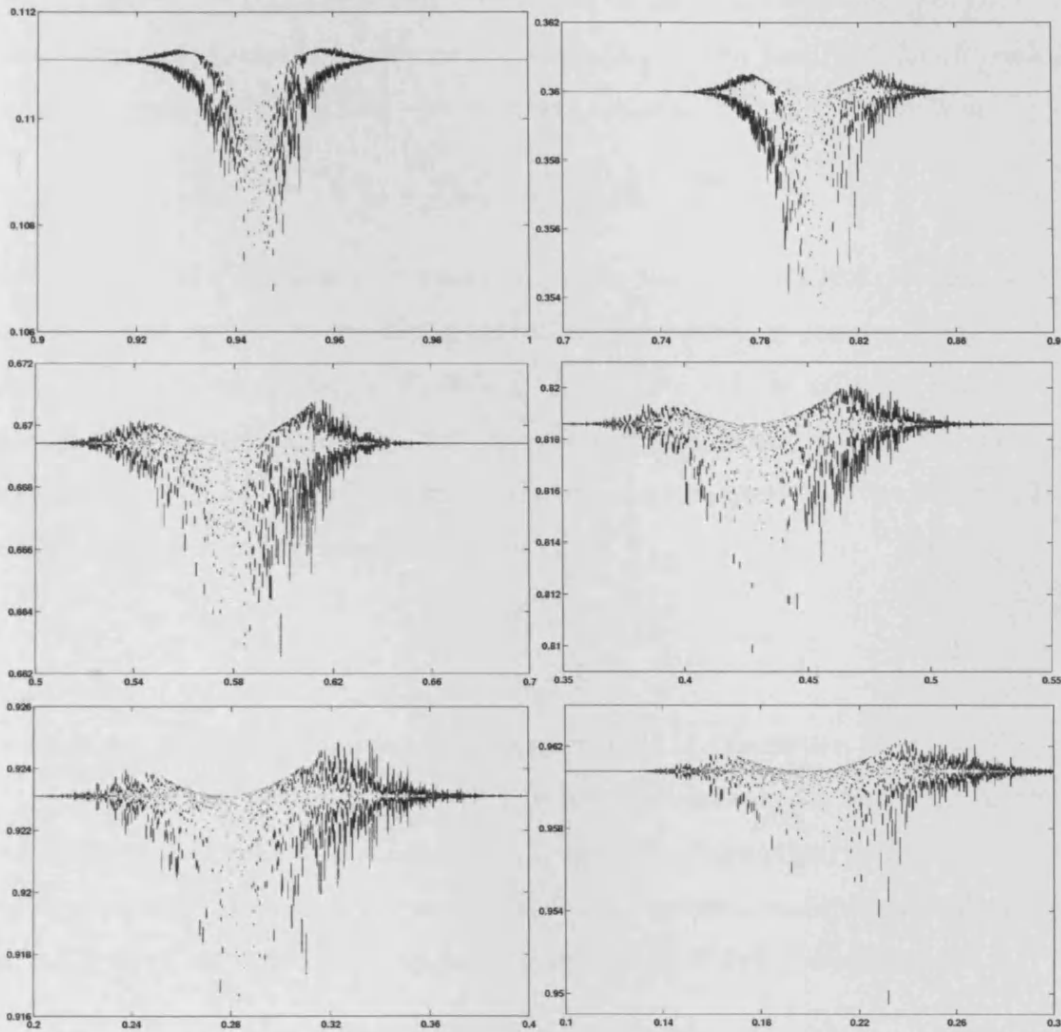


Figure 4.3: Attractors of  $r^{(k)}$  as a function of  $\gamma$  for the  $\gamma$ -square root algorithm with, from top left to bottom right;  $\rho = 2, 4, 10, 20, 50, 100$ ;  $d = 100$ .

## 4.3 Generalisation to the $\beta$ -Root Algorithm

### 4.3.1 The Renormalised $\beta$ -Root Algorithm

While it may not be immediately obvious from their step lengths, the steepest descent and square root algorithms both belong to the same larger family of gradient algorithms. Algorithms are classified as belonging to this family if the optimality criterion corresponding to their renormalised updating formula has the form

$$\Phi(\xi) = \mu_0^\beta \mu_2^\beta - \mu_1^{2\beta} = \mu_2^\beta - \mu_1^{2\beta}. \quad (4.14)$$

When  $\beta = 1$  the optimality criterion is exactly the  $D$ -optimality criterion which was shown in Sec 2.3 to produce a multiplicative algorithm corresponding to the renormalised steepest descent algorithm. When  $\beta = 1/2$  the criterion is precisely the optimality criterion described for the square root algorithm, see Sec 4.1. The optimality criterion (4.14) can therefore be seen as a generalisation of the  $D$ -optimality criterion where the determinant of the matrix

$$M(\xi) = \begin{pmatrix} \mu_0^\beta & \mu_1^\beta \\ \mu_1^\beta & \mu_2^\beta \end{pmatrix}$$

is taken as the criterion in place of the determinant of the matrix (2.17).

Using (4.14) as a basis, many more gradient algorithms, which would not be intuitively thought of without this connection to optimal experimental design theory, can be constructed. The relevant formulae needed to create a multiplicative algorithm of the form (2.20), which will maximise the criterion (4.14), are as follows:

$$\begin{aligned} \overset{\circ}{\Phi}(\xi) &= \begin{pmatrix} \frac{\partial \Phi}{\partial \mu_0} & \frac{1}{2} \frac{\partial \Phi}{\partial \mu_1} \\ \frac{1}{2} \frac{\partial \Phi}{\partial \mu_1} & \frac{\partial \Phi}{\partial \mu_2} \end{pmatrix} = \begin{pmatrix} \beta \mu_0^{\beta-1} \mu_2^\beta & -\beta \mu_1^{2\beta-1} \\ -\beta \mu_1^{2\beta-1} & \beta \mu_0^\beta \mu_2^{\beta-1} \end{pmatrix}, \\ \varphi(x, \xi) &= f^T(x) \overset{\circ}{\Phi}(\xi) f(x) = \beta \left( \mu_2^\beta - 2\mu_1^{2\beta-1} x + \mu_2^{\beta-1} x^2 \right), \\ c &= \min_x \varphi(x, \xi) = \frac{\beta \left( \mu_2^{2\beta-1} - \mu_1^{2(2\beta-1)} \right)}{\mu_2^{\beta-1}}, \\ b &= \text{tr} \left[ M(\xi) \overset{\circ}{\Phi}(\xi) \right] - c(\xi) = \beta \left( \mu_2^\beta - 2\mu_1^{2\beta} + \frac{\mu_1^{2(2\beta-1)}}{\mu_2^{\beta-1}} \right), \end{aligned}$$

$$\alpha = -2 \frac{\partial \Phi}{\partial \mu_2} / \frac{\partial \Phi}{\partial \mu_1} = \frac{\mu_2^{\beta-1}}{\mu_1^{2\beta-1}}. \quad (4.15)$$

The resulting updating formula is

$$\xi_i^{(k+1)} = \frac{\left(1 - \lambda_i \mu_2^{(\beta-1)} / \mu_1^{2\beta-1}\right)^2}{1 - 2 \left(\frac{\mu_2}{\mu_1}\right)^{\beta-1} + \left(\frac{\mu_2}{\mu_1}\right)^{2\beta-1}} \xi_i^{(k)} \quad \text{for } i = 1, \dots, d \quad (4.16)$$

and has a rate of convergence equal to

$$r^{(k)} = 1 - 2 \left(\frac{\mu_2}{\mu_1}\right)^{\beta-1} + \left(\frac{\mu_2}{\mu_1}\right)^{2\beta-1}. \quad (4.17)$$

This algorithm shall be referred to as the (renormalised)  $\beta$ -root algorithm.

To obtain the non-renormalised version of the  $\beta$ -root algorithm it is noted that the step length (4.15) can be re-written as

$$\alpha^{(k)} = \frac{\mu_2^{\beta-1}}{\mu_1^{2\beta-1}} = \frac{(g^{(k)}, g^{(k)})^\beta (A^2 g^{(k)}, g^{(k)})^{\beta-1}}{(A g^{(k)}, g^{(k)})^{2\beta-1}}$$

and hence the  $\beta$ -root algorithm takes the form

$$x^{(k+1)} = x^{(k)} - \frac{(g^{(k)}, g^{(k)})^\beta (A^2 g^{(k)}, g^{(k)})^{\beta-1}}{(A g^{(k)}, g^{(k)})^{2\beta-1}} g^{(k)}. \quad (4.18)$$

For the purpose of studying the asymptotic rate of convergence associated with this algorithm, the renormalised version (4.16) will be used.

### 4.3.2 Concavity of the Optimality Criterion

Following the same method set out in Theorem 3.1.1, the  $\beta$ -root optimality criterion (4.14) can be shown to be a concave functional for a specific range of  $\beta$ .

**Theorem 4.3.1.** *The functional,  $\Phi(\xi) = \mu_2^\beta - \mu_1^{2\beta}$ , is a concave functional for  $1/2 \leq \beta \leq 1$ .*

*Proof.* To prove concavity it is necessary and sufficient to show that

$$\frac{\partial^2}{\partial a^2} \Phi(\xi_a) \leq 0$$

holds for all  $\xi_a = (1 - a)\xi + a\xi'$ , where  $\xi, \xi'$  are any two designs and  $0 \leq a \leq 1$ .



The first derivatives of  $\Phi(\xi_a) = \mu_2^\beta(\xi_a) - \mu_1^{2\beta}(\xi_a)$ , are

$$\frac{\partial \Phi(\xi_a)}{\partial \mu_1(\xi_a)} = -2\beta \mu_1^{2\beta-1}(\xi_a), \quad \frac{\partial \Phi(\xi_a)}{\partial \mu_2(\xi_a)} = \beta \mu_2^{\beta-1}(\xi_a).$$

Substituting these values into the general formula for the second derivative (3.4) gives

$$\begin{aligned} \frac{\partial^2}{\partial a^2} \Phi(\xi_a) &= \frac{\partial}{\partial a} \left[ -2\beta \mu_1^{2\beta-1}(\xi_a)(\mu_1(\xi') - \mu_1(\xi)) + \beta \mu_2^{\beta-1}(\xi_a)(\mu_2(\xi') - \mu_2(\xi)) \right] \\ &= -2\beta(2\beta-1) \mu_1^{2\beta-2}(\xi_a)(\mu_1(\xi') - \mu_1(\xi))^2 + \beta(\beta-1) \mu_2^{\beta-2}(\xi_a)(\mu_2(\xi') - \mu_2(\xi))^2. \end{aligned}$$

Both terms will be non-positive and hence  $\frac{\partial^2}{\partial a^2} \Phi(\xi_a)$  will definitely be non-positive if  $\beta(2\beta-1) \geq 0$  and  $\beta(\beta-1) \leq 0$  i.e. if  $\beta$  lies in the range  $1/2 \leq \beta \leq 1$ .  $\square$

### 4.3.3 Behaviour of the Sequence $\{\Phi(\xi^{(k)})\}$

Performing similar analysis to that undertaken for the  $\gamma$ -steepest descent algorithm, the behaviour of the sequence

$$\{\Phi(\xi^{(k)})\} = \{\mu_2^\beta(\xi^{(k)}) - \mu_1^{2\beta}(\xi^{(k)})\} \quad (4.19)$$

is studied. It has already been shown in this chapter that when  $\beta = 1/2$  the sequence converges to a limiting design (see Theorem 4.2.2). For other values of  $\beta$  the behaviour of the sequence (4.19) is quite different. In particular, when  $\beta > 1$  the sequence can become chaotic. Figure 4.4 and Figure 4.5 show the attractors of  $\Phi(\xi^{(k)})$  as a function  $\beta$ , for various values of  $\rho$ , for both 2-dimensional and 100-dimensional problems. For most  $\beta > 1$  chaos is clearly present regardless of the condition number and number of dimensions. For those values of  $\beta$  where chaotic behaviour is not exhibited small cycles are formed and, as was observed with the  $\gamma$ -steepest descent algorithm, there is a greater abundance of values of  $\beta$  at which cyclic behaviour is demonstrated when  $d = 2$ . It is also evident from the  $y$ -axis in these figures that the sequence (4.19) is much more varied than the equivalent sequence for the  $\gamma$ -steepest descent algorithm.

In an attempt to further understand the progression of the sequence (4.19), Figure 4.6 and Figure 4.7 show the normalised values of  $\Phi(\xi^{(k)})$  as a function of  $k$  for various values of  $\beta$ . For  $1 < \beta \lesssim 1.1$  the sequence tends to progress in a relatively

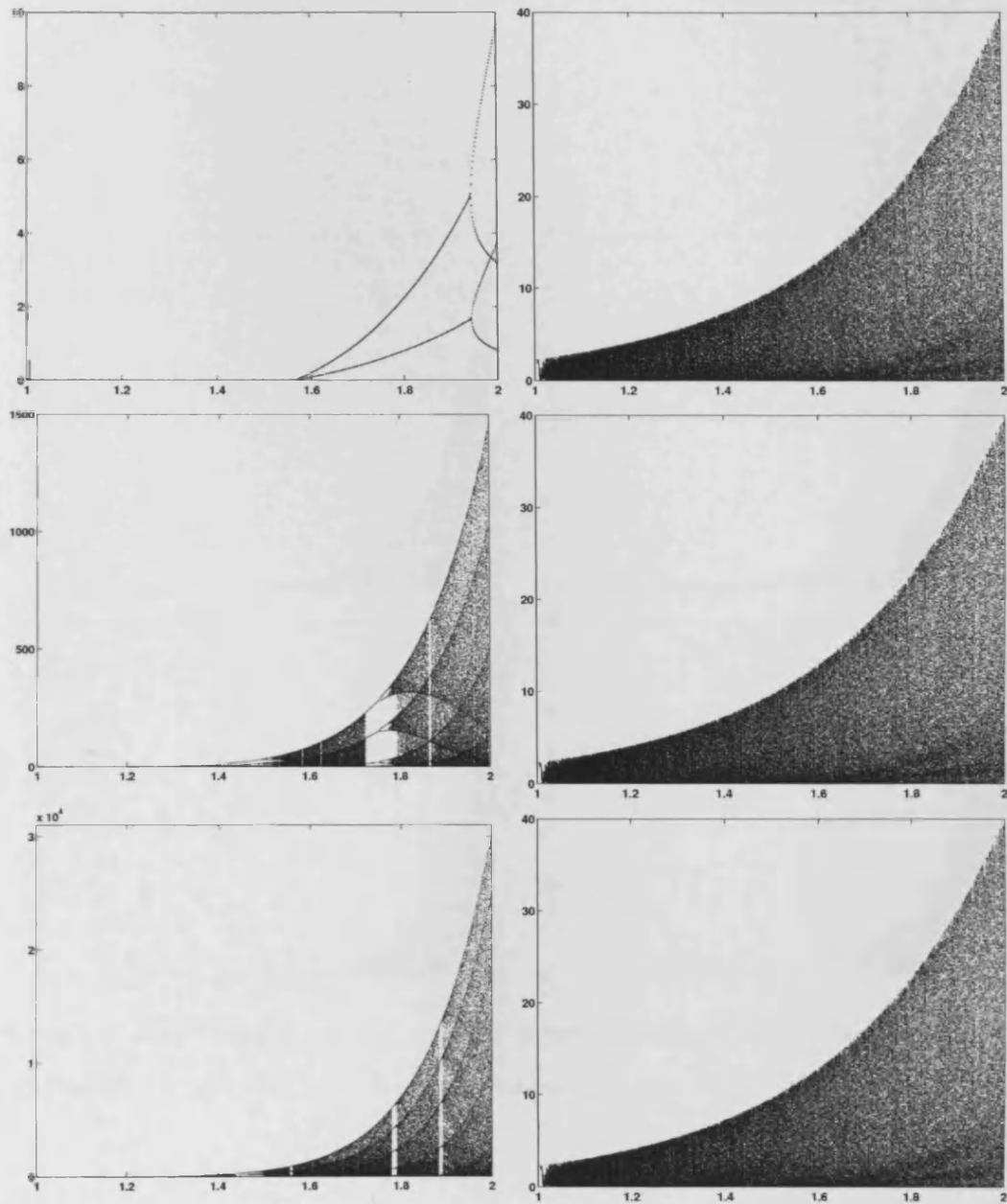


Figure 4.4:  $\Phi(\xi^{(k)})$  for  $k = 750, \dots, 1000$ ; plotted as a function of  $\beta$  for (left)  $d = 2$  and (right)  $d = 100$  and with, from top to bottom:  $\rho = 4$ ,  $\rho = 10$ ,  $\rho = 20$ .

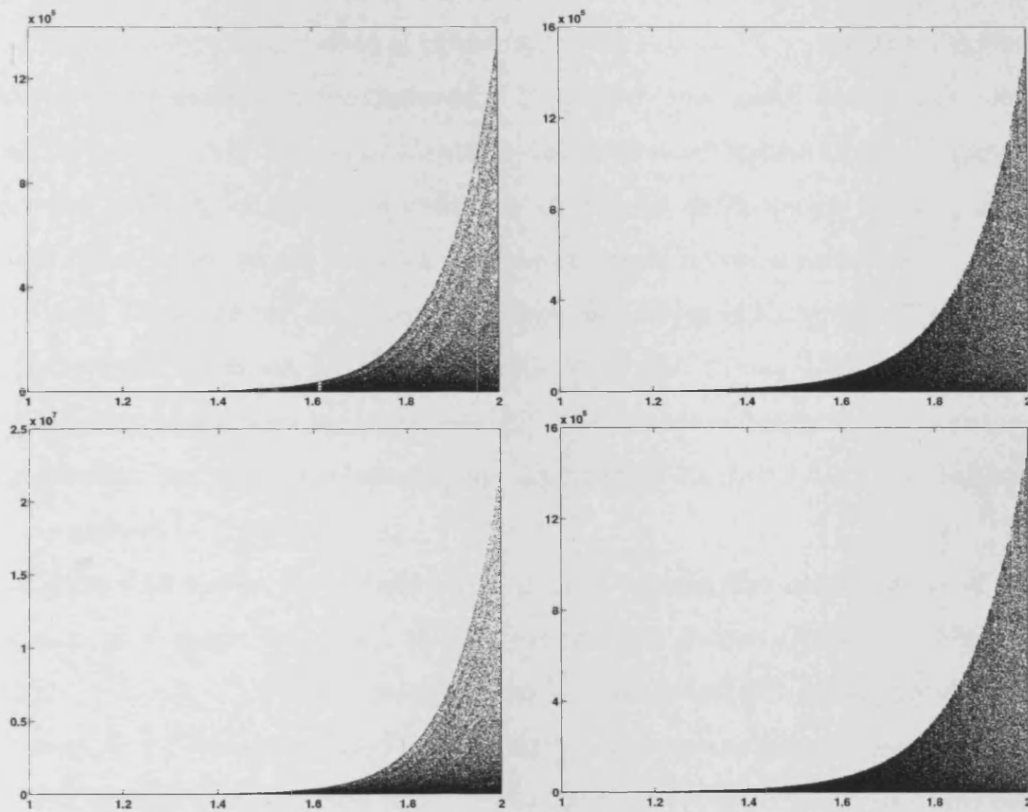


Figure 4.5:  $\Phi(\xi^{(k)})$  for  $k = 750, \dots, 1000$ ; plotted as a function of  $\beta$  for (left)  $d = 2$  and (right)  $d = 100$  and with, from top to bottom:  $\rho = 50, \rho = 100$ .

stable fashion, slowly decreasing in value until a near zero figure is reached, at which point a zigzagging period ensues until a near zero figure is again reached which causes the sequence to jump to a much larger value for the behaviour to then repeat itself. This is, however, only the general trend of the sequence, the length and frequency of each phase of behaviour is not constant. For  $\beta \gtrsim 1.1$  the sequence progresses in a more hap-hazard fashion with  $\Phi(\xi^{(k)})$  zigzagging from low to high values and back again with a higher frequency when  $\beta$  is larger.

Figure 4.8 shows the spread of values of  $\Phi(\xi^{(k)})$  for  $k = 1, \dots, 10000$ , for different values of  $\beta$ . Regardless of the choice of  $\beta$ , the distribution peaks at near zero values. For  $\beta$  close to 1, the histograms confirm the oscillatory nature seen in Figure 4.6 since the majority of values are located at either end of the range. This behaviour is less apparent for larger values of  $\beta$  as the sequence becomes more erratic.

Figure 4.9 shows the relationship between the values of the optimality criterion at consecutive iterations, i.e.  $\Phi(\xi^{(k)})$  and  $\Phi(\xi^{(k+1)})$ , for various values  $\beta$ . For  $\beta$  close to 1 the progression from iteration to iteration is more predictable but as  $\beta$  increases the situation becomes more chaotic, in particular when  $\beta = 1.5$  or  $\beta = 2$  chaos is clearly present in the transition.

Figure 4.10 shows the weight at  $\lambda_1$  plotted against the weight at  $\lambda_d$  at each iteration of a single trajectory of the renormalised  $\beta$ -root algorithm. Since the weights,  $\xi^{(k)}(\lambda_1), \dots, \xi^{(k)}(\lambda_d)$ , must sum to 1,  $\xi^{(k)}(\lambda_1)$  and  $\xi^{(k)}(\lambda_d)$  are restricted by the inequality  $\xi^{(k)}(\lambda_d) + \xi^{(k)}(\lambda_1) \leq 1$ . On the graph corresponding to each value of  $\beta$  studied, there are many points very close to the line  $\xi^{(k)}(\lambda_d) = 1 - \xi^{(k)}(\lambda_1)$  indicating that at many iterations the design is supported solely at the minimum and maximum eigenvalues. There are however points located beneath this line, indicating that after a time the middle weights re-establish themselves sending the algorithm back into chaos. As  $\beta$  increases, the density of points below the line increases which indicates that the design is supported entirely at the end points ( $\lambda_1$  and  $\lambda_d$ ) less and less.

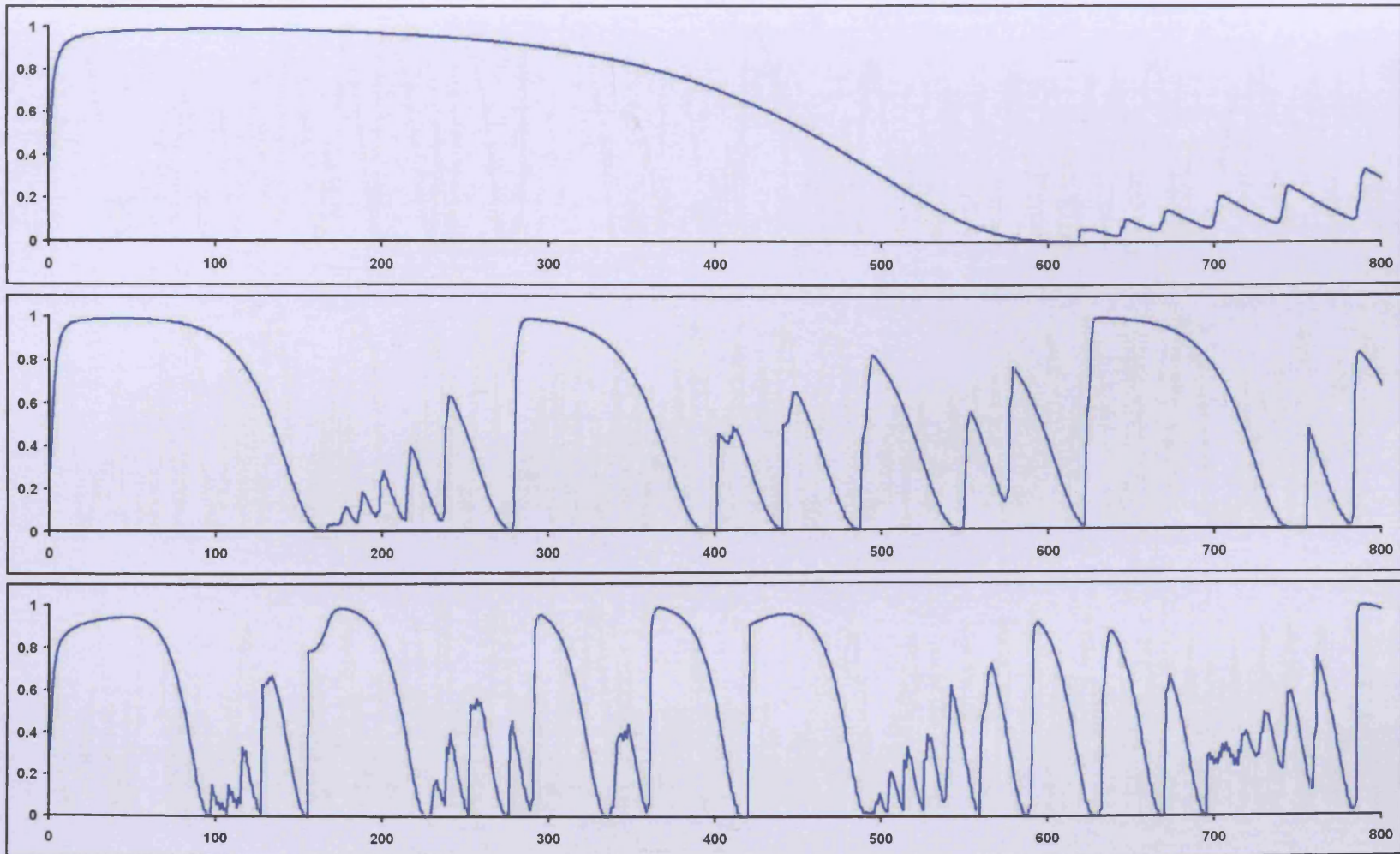


Figure 4.6: Trajectory of the efficiency  $\Phi(\xi^{(k)}) / \max_{\xi} \Phi(\xi)$  plotted as a function of  $k$  for single realisations of the  $\beta$ -root algorithm with, from top to bottom;  $\beta = 1.01$ ,  $\beta = 1.05$ ,  $\beta = 1.1$ ;  $\rho = 10$ ,  $d = 100$ .

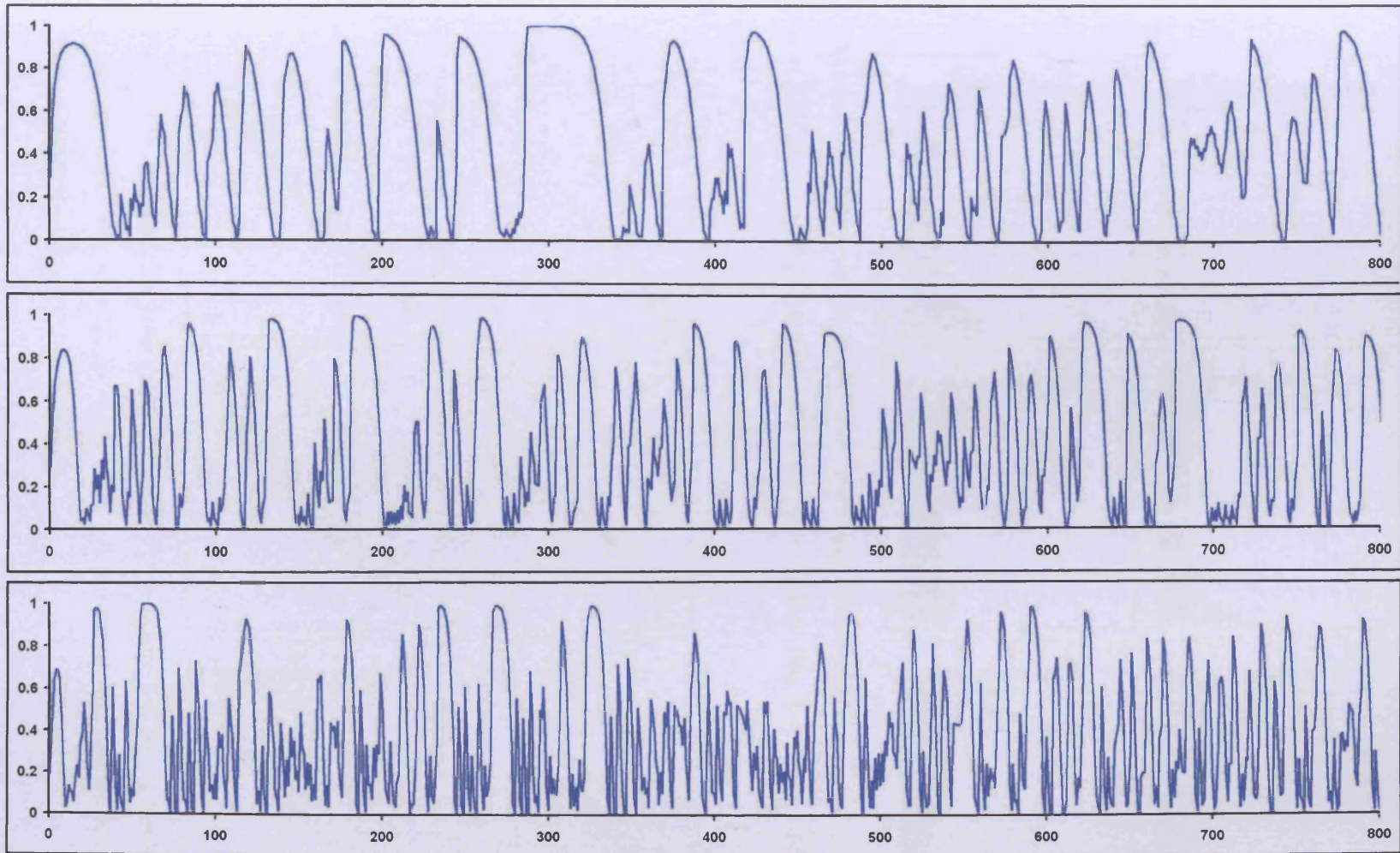


Figure 4.7: Trajectory of the efficiency  $\Phi(\xi^{(k)})/\max_{\xi} \Phi(\xi)$  plotted as a function of  $k$  for single realisations of the  $\beta$ -root algorithm with, from top to bottom;  $\beta = 1.2$ ,  $\beta = 1.5$ ,  $\beta = 2$ ;  $\rho = 10$ ,  $d = 100$ .

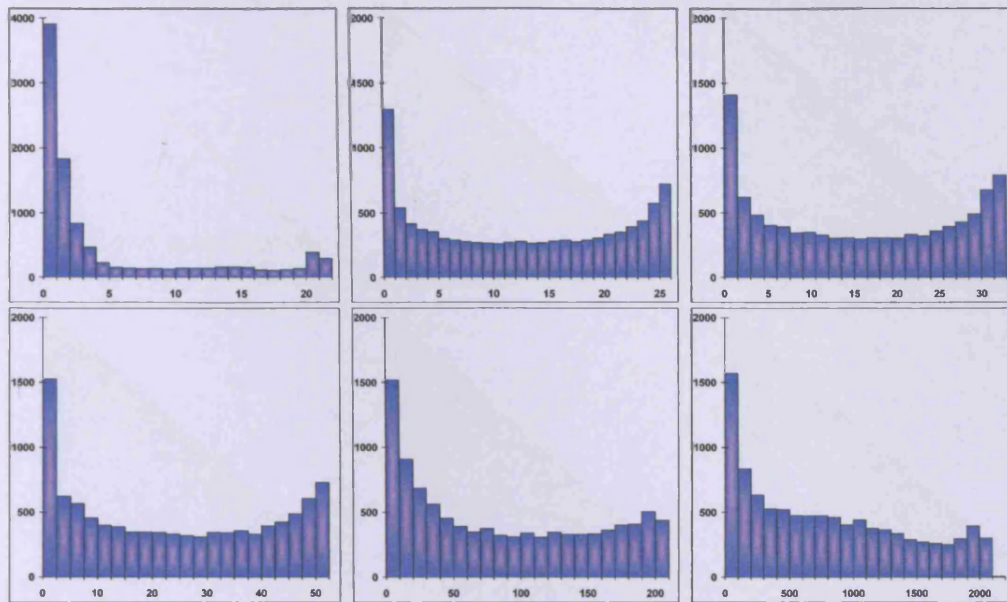


Figure 4.8: Histogram of  $\Phi(\xi^{(k)})$  for, from top left to bottom right:  $\beta = 1.01$ ,  $\beta = 1.05$ ,  $\beta = 1.1$ ,  $\beta = 1.2$ ;  $\beta = 1.5$ ,  $\beta = 2$ ;  $\rho = 10$ ;  $d = 100$ ,  $k = 1, \dots, 10000$ .

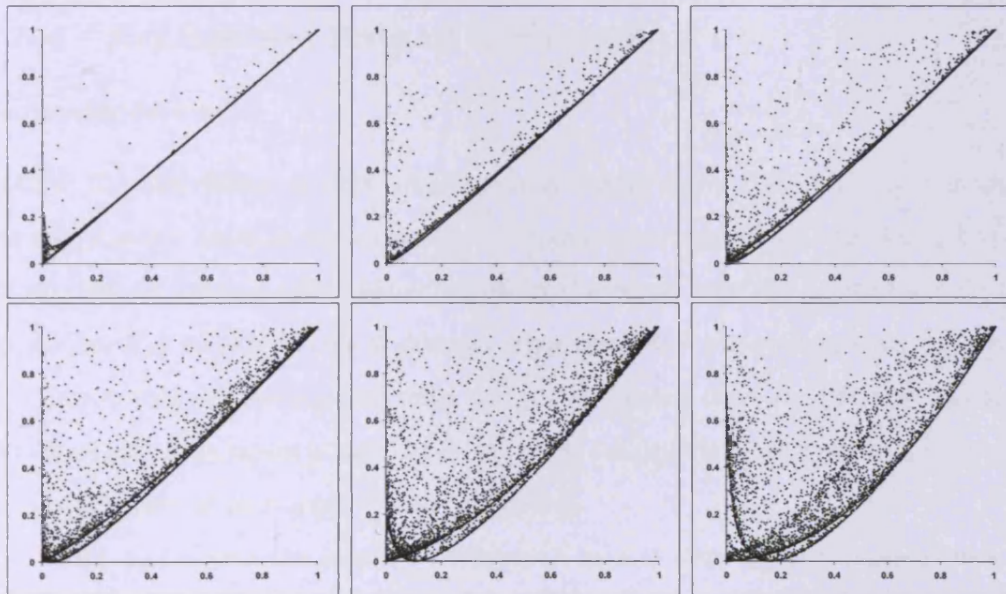


Figure 4.9: Plots of the pairs  $(\Phi(\xi^{(k)}), \Phi(\xi^{(k+1)}))$  for, from top left to bottom right  $\beta = 1.01$ ,  $\beta = 1.05$ ,  $\beta = 1.1$ ,  $\beta = 1.2$ ;  $\beta = 1.5$ ,  $\beta = 2$ . Points plotted are the last 2000 of 10000 iterations;  $d = 100$ ,  $\rho = 10$ .

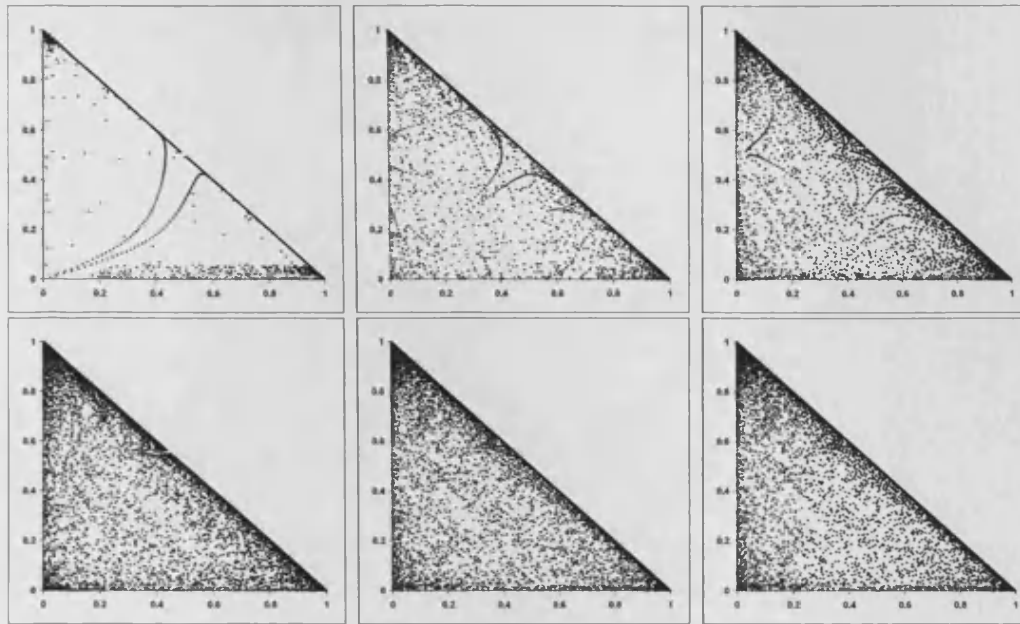


Figure 4.10: Plots of the pairs  $(\xi^{(k)}(\lambda_1), \xi^{(k)}(\lambda_d))$  for the  $\beta$ -root algorithm with (from top left to bottom right):  $\beta = 1.01$ ,  $\beta = 1.05$ ,  $\beta = 1.1$ ,  $\beta = 1.2$ ;  $\beta = 1.5$ ,  $\beta = 2$ ;  $\rho = 10$ ;  $d = 100$ . Points plotted are the last 8000 of 10000 iterations.

#### 4.3.4 Asymptotic Rate of Convergence

##### Dependence on $\beta$

Both of the algorithms studied in the  $\beta$ -root family so far have had comparatively poor asymptotic rates of convergence. The square-root algorithm was shown to have an asymptotic rate of convergence equal to the worst rate of the steepest descent algorithm,  $R_{\text{ref}}$  and the steepest descent algorithm has been seen to have an average asymptotic rate of convergence only marginally better than  $R_{\text{ref}}$ . It remains to be seen whether there exists a value of  $\beta$  for which a significant improvement in average asymptotic rate of convergence can be yielded.

Figure 4.11 shows the average asymptotic rate of convergence of the  $\beta$ -root algorithm as a function of  $\beta$ . For  $\beta \leq 1$  the average asymptotic rate of convergence is unvarying and equal to  $R_{\text{ref}}$  but when  $\beta > 1$  the rate is no longer constant and it is evident that much improved average asymptotic rates of convergence are possible. For these particular parameter settings, the average asymptotic rate of convergence



achieved when  $\beta$  is fractionally larger than one is especially promising.

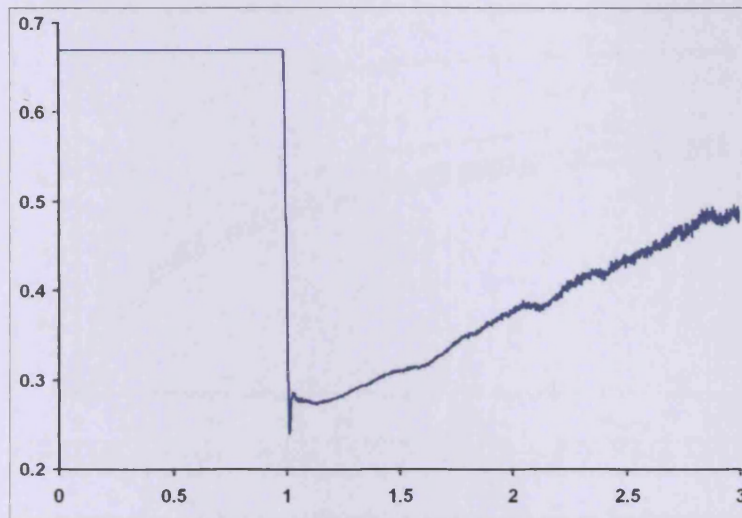


Figure 4.11: Average asymptotic rate of convergence as a function of  $\beta$  for the  $\beta$ -root algorithm.

It may not be the case that the optimum value of  $\beta$  is the same for all condition numbers or for problems of any number of dimensions. For the parameters used in Figure 4.11 the valley in which the optimum value of  $\beta$  lies is very steep. A shift of as little as 0.01 in the value of  $\beta$  will affect the asymptotic rate of convergence significantly, indicating the importance of selecting exactly the right value of  $\beta$  for the situation. In order to be able to choose the most appropriate value of  $\beta$  further analysis is needed.

### Dependence on $\rho$

Now that it is known that competitive convergence rates are possible with the correct choice of  $\beta$ , it is necessary to discover to what extent the average asymptotic rate of convergence depends on the value of the condition number  $\rho$ . Figure 4.12 shows the average asymptotic rate of convergence as a function of  $\rho$  for several different  $\beta$ . Similarly to what has already been observed for other algorithms, the average asymptotic rate of convergence worsens as  $\rho$  increases. For some values of  $\beta$  the increase in  $R$  occurs in a more stable fashion than others. In particular, when  $\beta = 1.01$  or  $\beta = 2$  the increase in asymptotic rate,  $R$ , is less steady than for

intermediate values. Figure 4.13 shows the average asymptotic rate of convergence

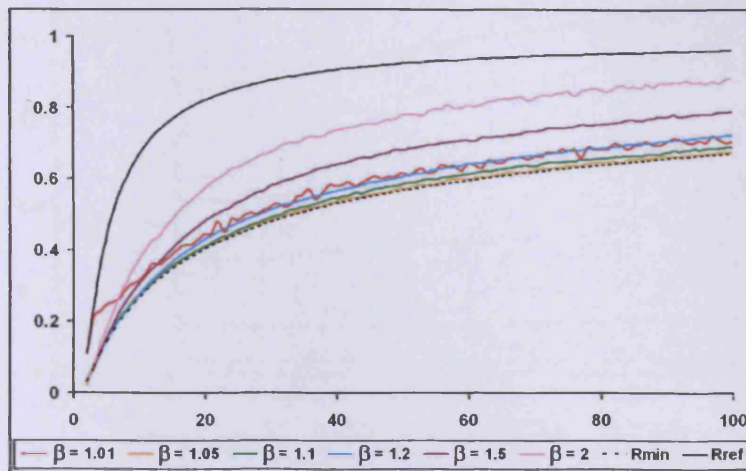


Figure 4.12: Average asymptotic rate of convergence as a function of  $\rho$  for various values of  $\beta$ ;  $d = 100$ .

as a function of  $\beta$  for several different values of  $\rho$ . It can be seen that the shape of the curve is different for the different condition numbers used. The value of  $\beta$  that will produce the best asymptotic rates of convergence will therefore be slightly different in each case. In reality, the condition number of an optimisation problem is not usually known in advance so it is not possible to pre-specify a value of  $\beta$  for a particular condition number. It is therefore of use to recommend a value of  $\beta$  that will work well for any condition number. A value of  $\beta = 1.05$  seems to be best.

To gain a greater insight into how these advantageous rates are achieved, Figure 4.14 and Figure 4.15 show the attractors of  $r^{(k)}$  as a function of  $\beta$  for both a 2-dimensional and 100-dimensional case. As was noted when studying the behaviour of the sequence (4.19), the algorithm generally exhibits chaos when  $\beta > 1$ . This corresponds to the region where faster asymptotic rates of convergence occur.

In the 2-dimensional case, for certain values of  $\beta$ , the  $\beta$ -root algorithm converges  $R$ -superlinearly. The regions of  $\beta$  for which this is the case can be identified in the graphs by values of  $\beta$  where all attractors of the sequence  $\{r^{(k)}\}$  equal zero. The range of  $\beta$  for which  $R$ -superlinear convergence occurs shrinks as  $\rho$  increases. Another phenomenon observable in the 2-dimensional case is that cycles are formed with a much higher prevalence compared to the 100-dimensional case.

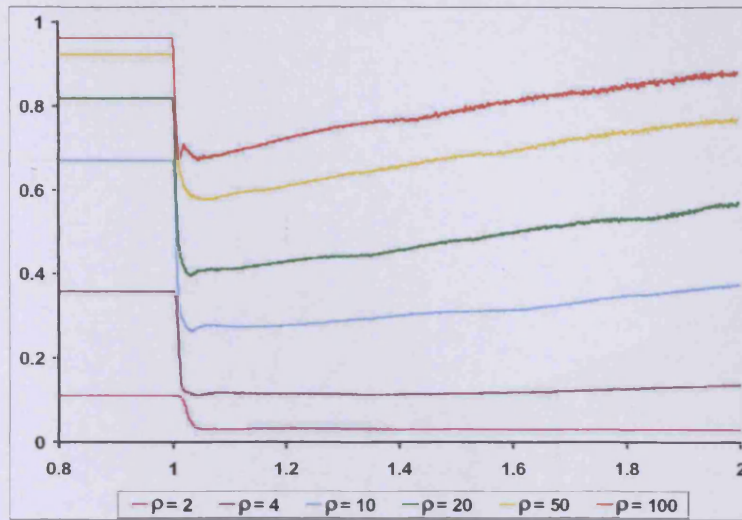


Figure 4.13: Average asymptotic rate of convergence of a function of  $\beta$  for various  $\rho$ ;  $d = 100$ .

It can be understood from the denseness of points close to the  $x$ -axis in Figure 4.14 that at many iterations  $r^{(k)}$  is very close to zero. Near zero values have a desirable effect on the average asymptotic rate of convergence since  $R$  is the geometric mean of the sequence  $\{r^{(k)}\}$ . A value of  $r^{(k)} = 0.001$  compared to a value of  $r^{(k)} = 0.01$  would not have that much of a difference in effect on an arithmetic mean but on a geometric mean the difference is much more significant. The scales in Figure 4.14 and Figure 4.15 are such that it is not easy to see, with any degree of accuracy, exactly how close to zero these points really are. A small difference in  $r^{(k)}$  can be much more clearly distinguished in Figure 4.16 and Figure 4.17 where the attractors of  $(-\ln r^{(k)})$  are plotted as a function of  $\beta$ . It should be noted that, due to limitations in accuracy with the floating point calculations completed in the computer simulations used to generate these figures, a value of  $10^{-10} \cong 0$  and therefore values of  $\beta$  where  $R$ -superlinear convergence occurs should be identified by points at which  $-\ln r^{(k)} = -\ln(10^{-10}) \cong 23$ .

When  $\beta = 1$  the sequence  $\{r^{(k)}\}$  converges to a 2-point cycle. For  $\beta > 1$  the sequence no longer converges and instead exhibits chaos. The transition from cyclic behaviour to chaotic behaviour can be seen more closely in Figure 4.18 which shows  $r^{(k)}$  as a function of  $\beta$  for  $1 \leq \beta \leq 1.01$ .

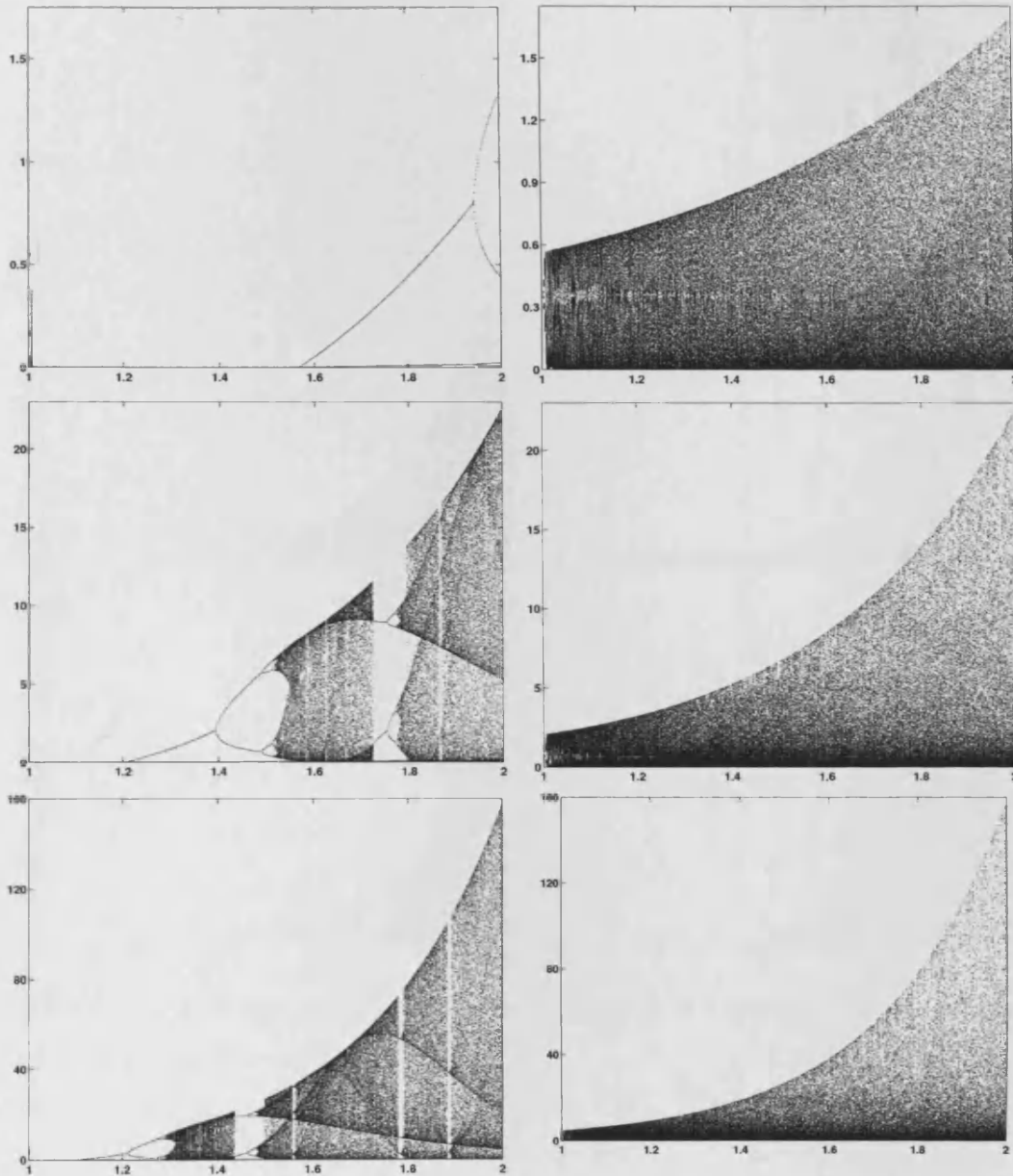


Figure 4.14: Attractors of  $r^{(k)}$  as a function of  $\beta$  for  $d = 2$  (left) and  $d = 100$  (right) and from top to bottom;  $\rho = 4, 10, 20$ .

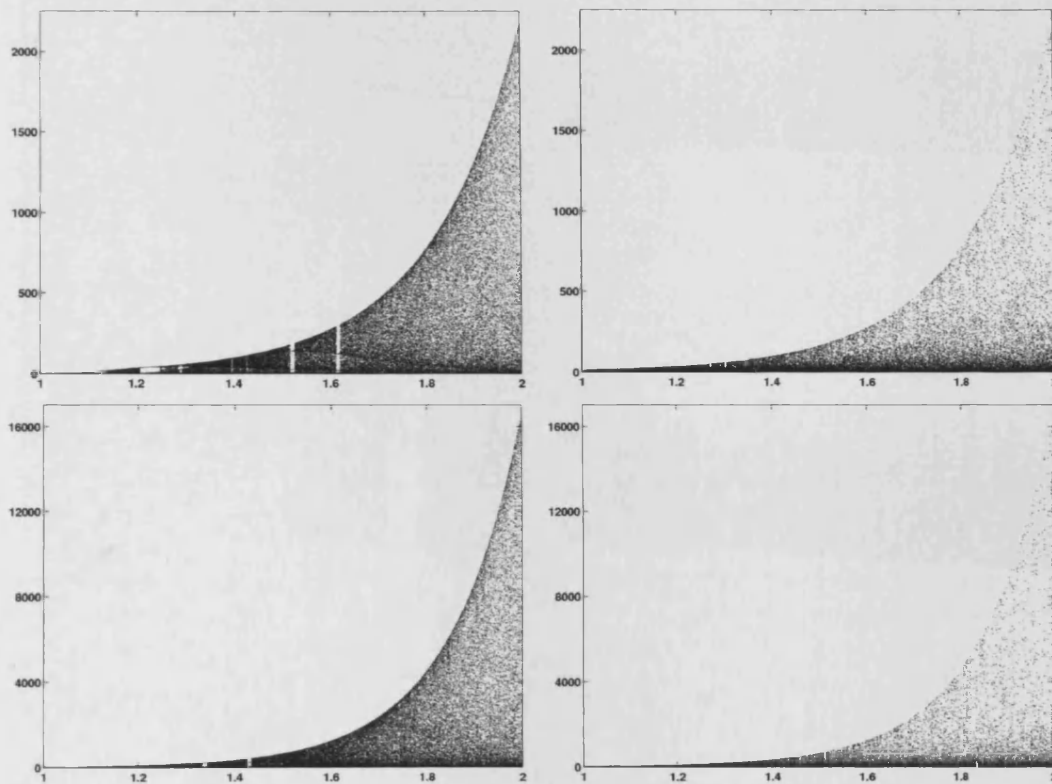


Figure 4.15: Attractors of  $r^{(k)}$  as a function of  $\beta$  for  $d = 2$  (left) and  $d = 100$  (right) and from top to bottom;  $\rho = 50, 100$ .

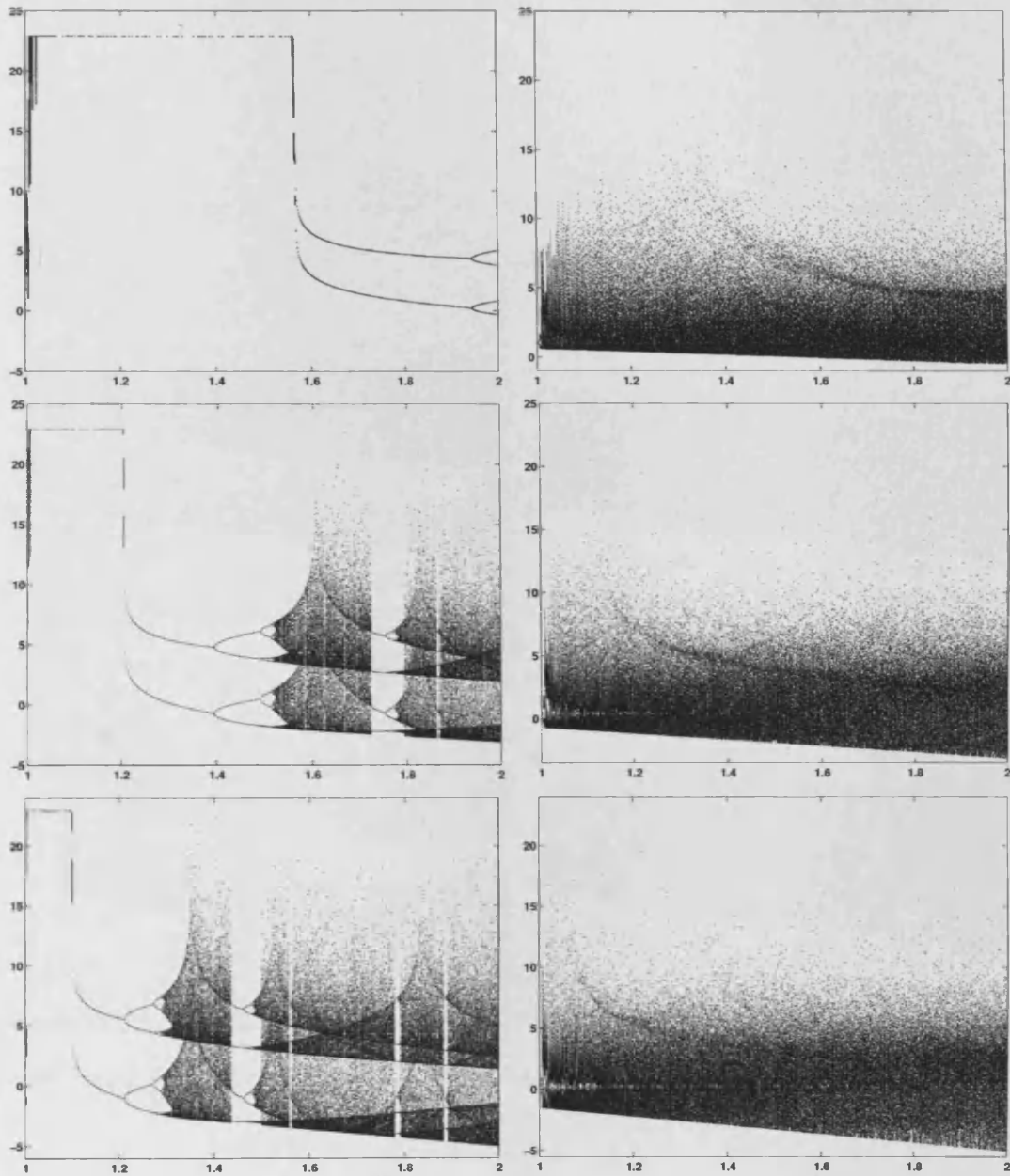


Figure 4.16: Attractors of  $(-\ln r^{(k)})$  as a function of  $\beta$  for  $d = 2$  (left) and  $d = 100$  (right) and from top to bottom;  $\rho = 4, 10, 20$ .

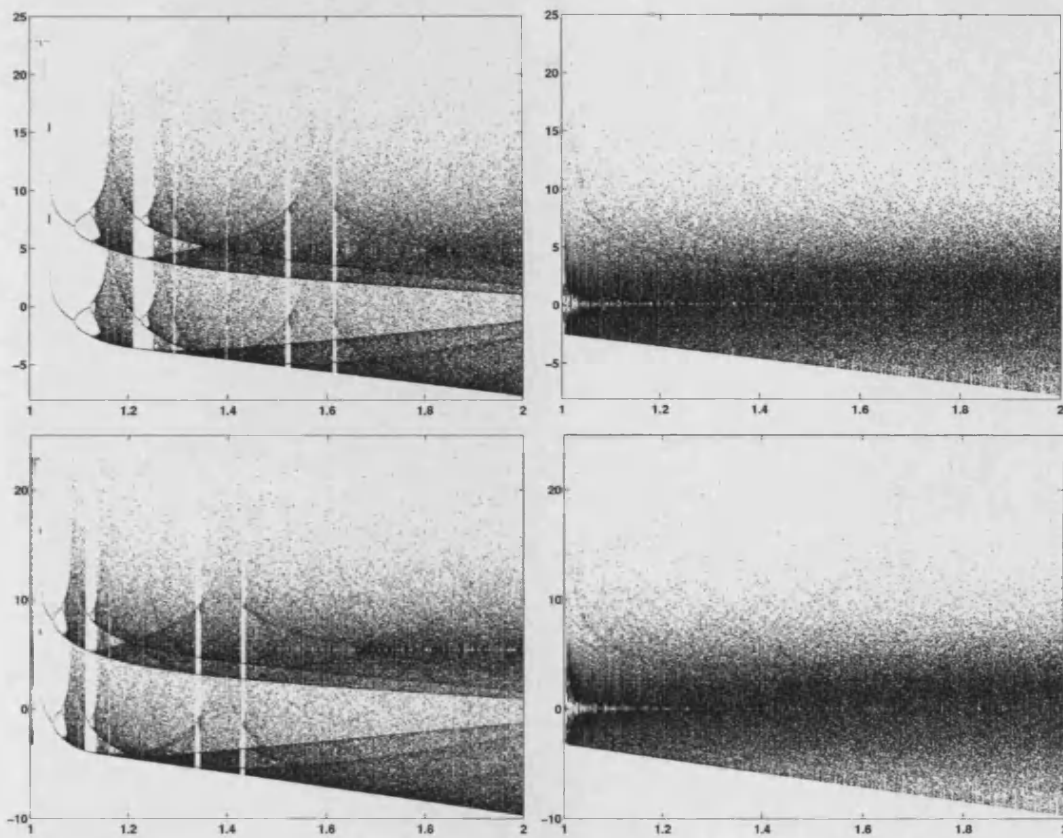


Figure 4.17: Attractors of  $(-\ln r^{(k)})$  as a function of  $\beta$  for  $d = 2$  (left) and  $d = 100$  (right) and from top to bottom;  $\rho = 50, 100$ .

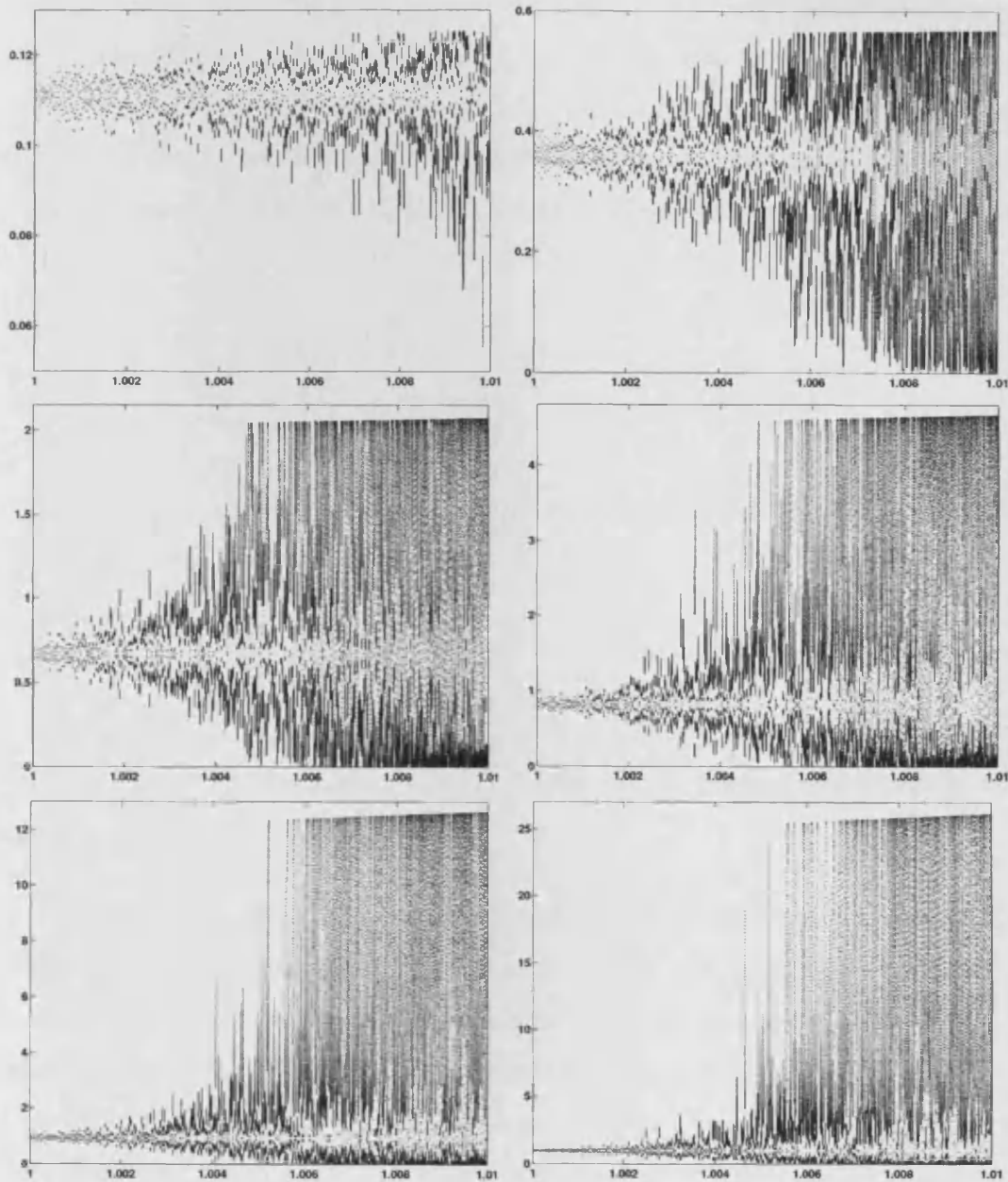


Figure 4.18: Attractors of  $r^{(k)}$  as a function of  $\beta$  for  $1 \leq \beta \leq 1.01$ , with from top left to bottom right;  $\rho = 2, 4, 10, 20, 50, 100$ .



### Dependence on $d$

The effect of  $d$  on the average asymptotic rate of convergence of the  $\beta$ -root algorithm must also be analysed. Figure 4.19 shows the average asymptotic rate of convergence as a function of  $d$  for various values of  $\beta$ . It can be seen that the relationship between the asymptotic rate and the number of dimensions is approximately constant for  $d \cong 10$  onwards indicating that increasing the number of dimensions of the problem does not worsen the rate at which the algorithm converges.

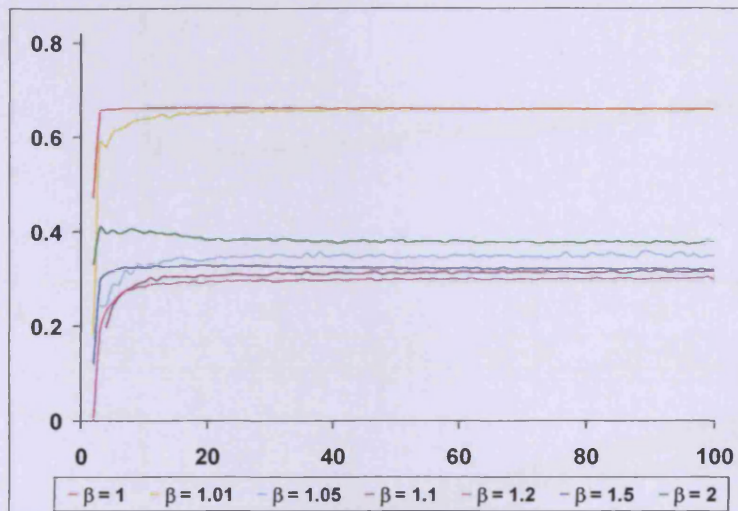


Figure 4.19: Average asymptotic rate of convergence as a function of  $d$  for the  $\beta$ -root algorithm for various  $\beta$ , with  $\rho = 10$ .

While increasing the number of dimensions has little effect on the asymptotic rate of convergence of the  $\beta$ -root algorithm once  $\beta$  has been selected, the number of dimensions does have an effect on the selection of the optimum value of  $\beta$  in the first place. Figure 4.20 shows the average asymptotic rate of convergence of the  $\beta$ -root algorithm as a function of  $\beta$  for different values of  $d$ . The minimum point of each curve is slightly different and hence the optimum value of  $\beta$  depends on  $d$  as well as on the condition number  $\rho$ . In both the graphs in Figure 4.20 the line corresponding to the 4-dimensional case is broken in several places. At the values of  $\beta$  where the line is broken, the  $\beta$ -root algorithm has  $R$ -superlinear convergence. It should be noted however that algorithms of this type would generally be applied to problems with a large number of dimensions.

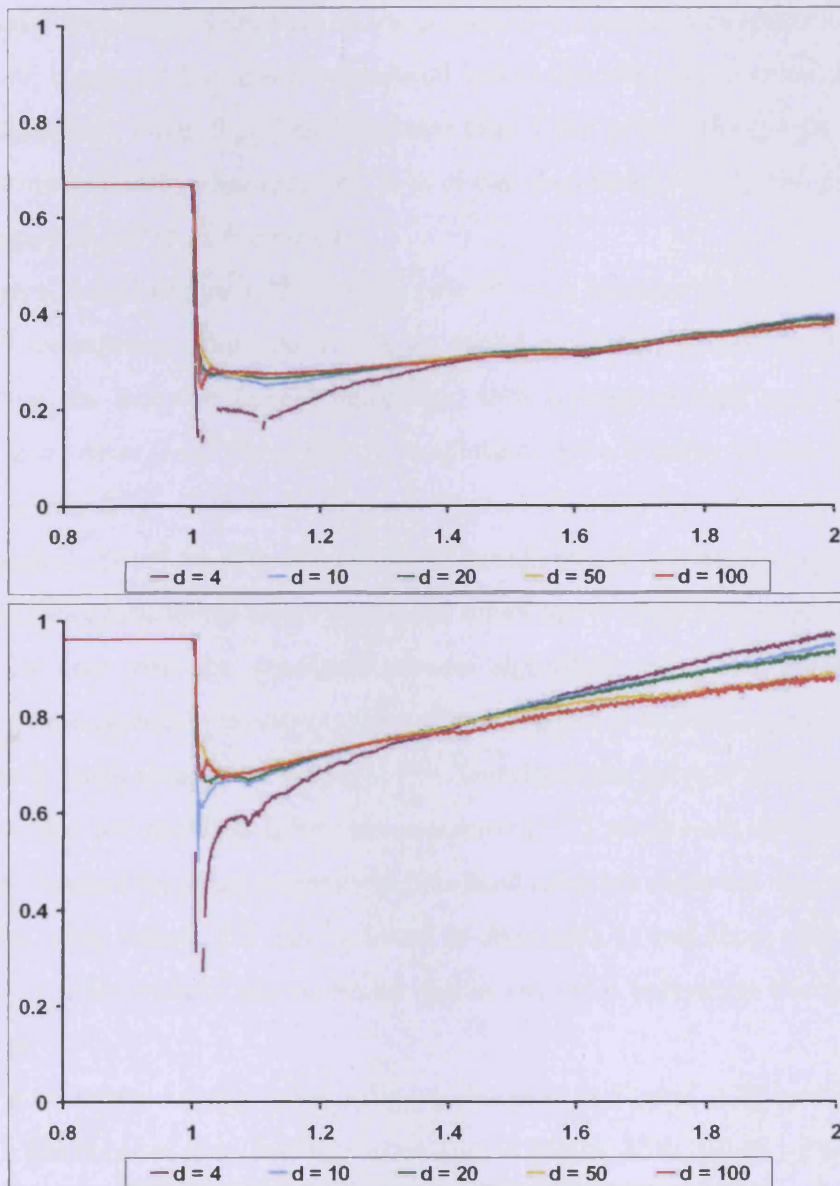


Figure 4.20: Average asymptotic rate of convergence as a function of  $\beta$  for the  $\beta$ -root algorithm with various  $d$  and  $\rho = 10$  (top) and  $\rho = 100$  (bottom).

### Behaviour of $r^{(k)}$

It has been seen that the best asymptotic rates of convergence occur when  $\beta$  is slightly greater than 1. The exact value of  $\beta$  for which the best asymptotic rates of convergence are achieved depends on the parameters  $\rho$  and  $d$ . The condition number  $\rho$ , however, is not usually known beforehand and in this situation a value of  $\beta = 1.05$  is recommended. When  $\beta$  is slightly greater than 1 the rate of the  $\beta$ -root algorithm at each iteration varies dramatically. It is of use therefore to study the progression of the sequence  $\{r^{(k)}\}$  as  $k$  increases.

Figure 4.21 and Figure 4.22 show the rate  $r^{(k)}$  as a function of  $k$  for  $\beta = 1.01$  and  $\beta = 1.05$  respectively. Both start with an initial oscillatory period similar to that observed in the steepest descent algorithm, with correspondingly poor rates as a consequence. After some iterations the oscillations grow steadily in size, eventually producing extremely high rates (greater than 1) followed by extremely low rates (close to zero). Every so often the algorithm breaks out of this oscillatory behaviour and periods of chaos follow where profitable values of  $r^{(k)}$  close to zero are produced. As was the case with the  $\gamma$ -steepest descent algorithm, the  $\beta$ -root algorithm will only produce a desirable asymptotic rate of convergence if it is left to run for enough iterations to reach these lower values of  $r^{(k)}$ . The closer the value of  $\beta$  is to 1 the more iterations that are required before the sequence  $\{r^{(k)}\}$  breaks out of its oscillatory phase and reaches the stage where more beneficial rates are achieved. Corresponding graphs for other values of  $\beta$  can be found in Appendix C and show that for larger values of  $\beta$ , little pattern can be found within the rates indicating the presence of true chaos.

The distribution of the rates,  $r^{(k)}$ , can be seen in Figure 4.23 and the corresponding distribution of  $(-\ln r^{(k)})$  can be seen in Figure 4.24. While on average the asymptotic rate of convergence is desirable for the  $\beta$ -root algorithm with suitable choice of  $\beta$ , it can be seen from these histograms that at some iterations the rate will be much worse and, of course, at some iterations the rate of convergence will be much better. It is recalled that if  $r^{(k)} > 1$  at a particular iteration this corresponds to a situation where the approximation to the minimum point is further away from  $x^*$  than at the previous iteration. This indicates that, just like the  $\gamma$ -steepest descent

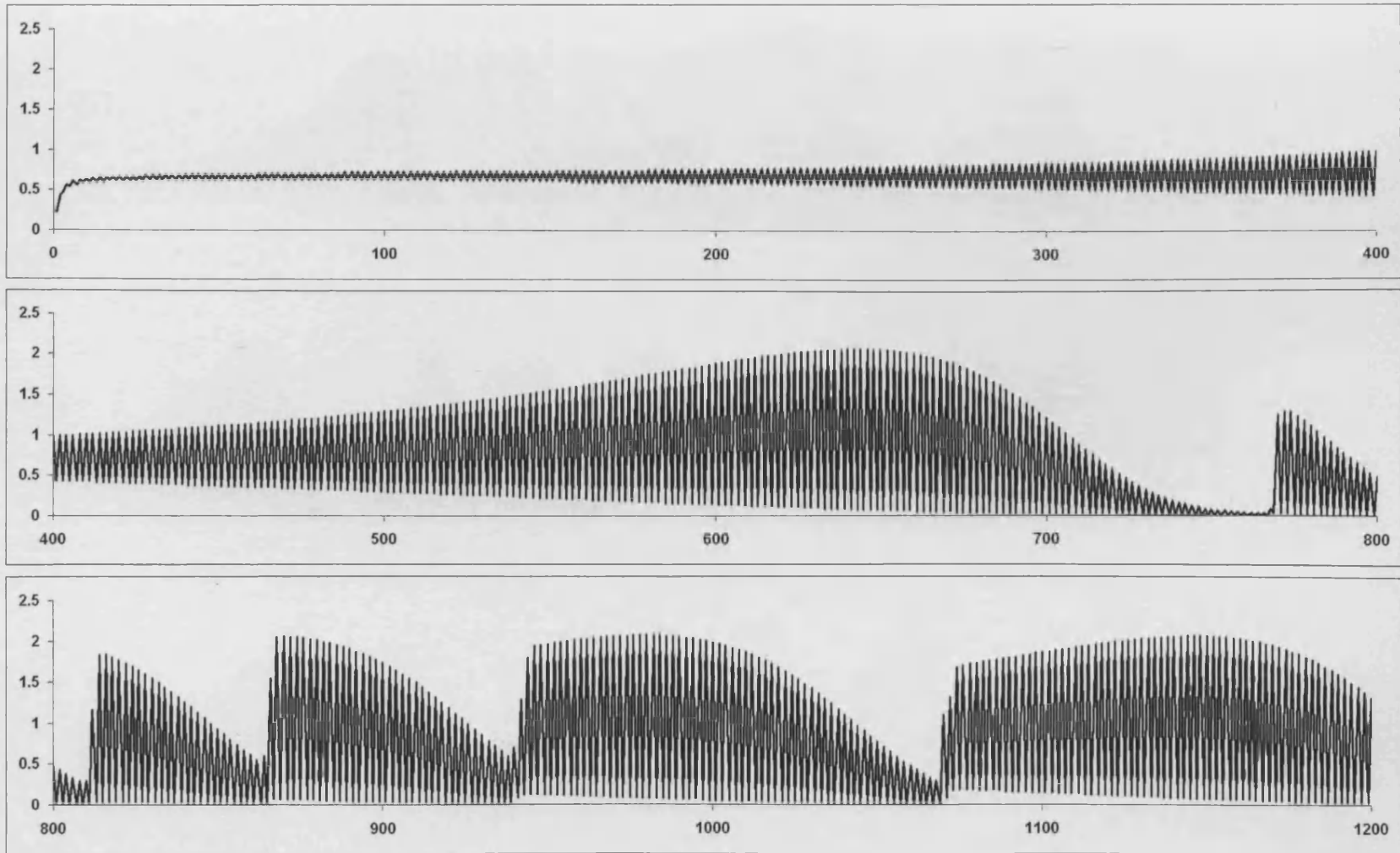


Figure 4.21: Rate,  $r^{(k)}$ , as a function of  $k$  for  $d = 100$ ,  $\rho = 10$  and  $\beta = 1.01$ .

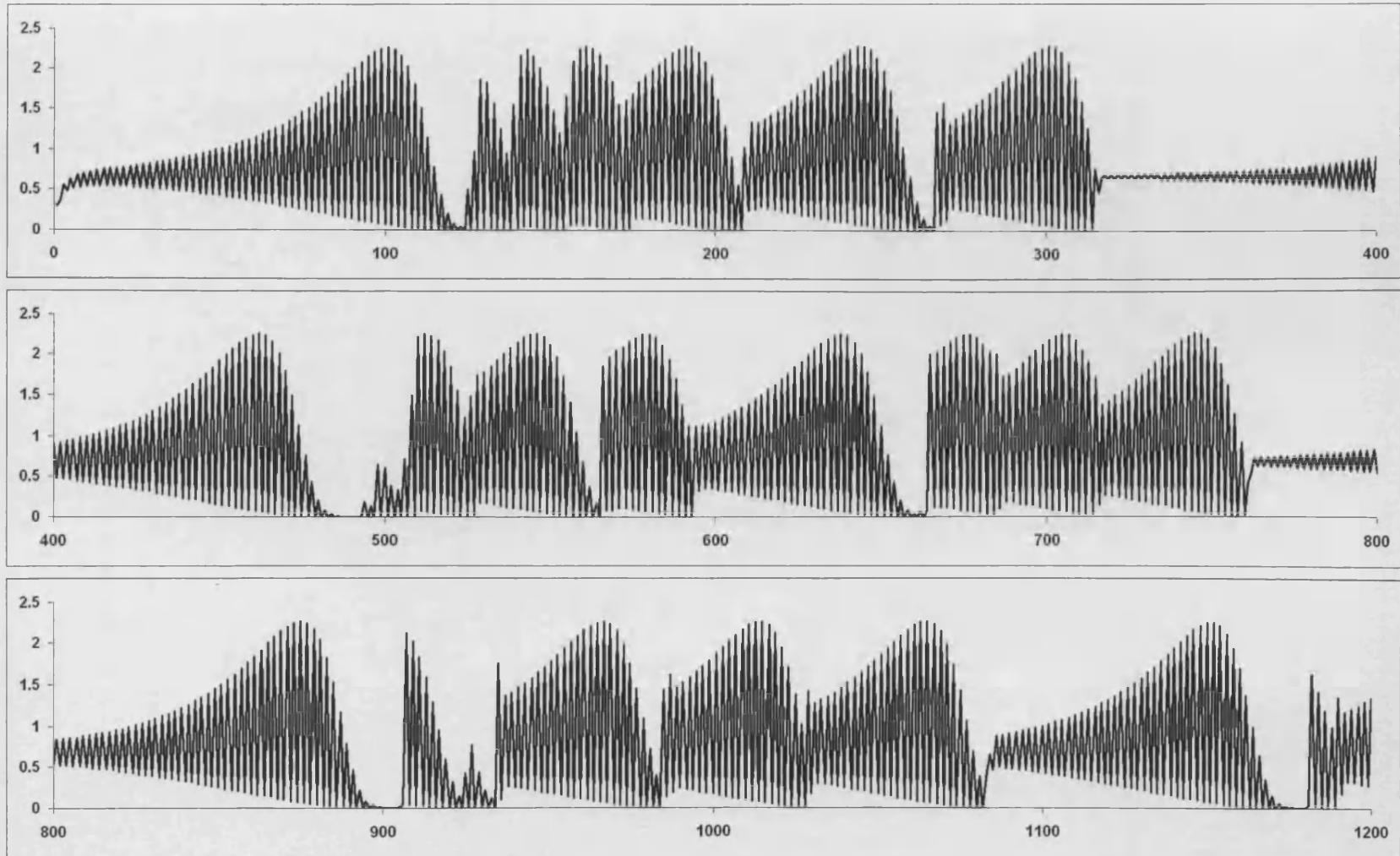


Figure 4.22: Rate,  $r^{(k)}$ , as a function of  $k$  for  $d = 100$ ,  $\rho = 10$  and  $\beta = 1.05$ .

and Barzilai-Borwein algorithms, the  $\beta$ -root algorithm is non-monotonic in nature.

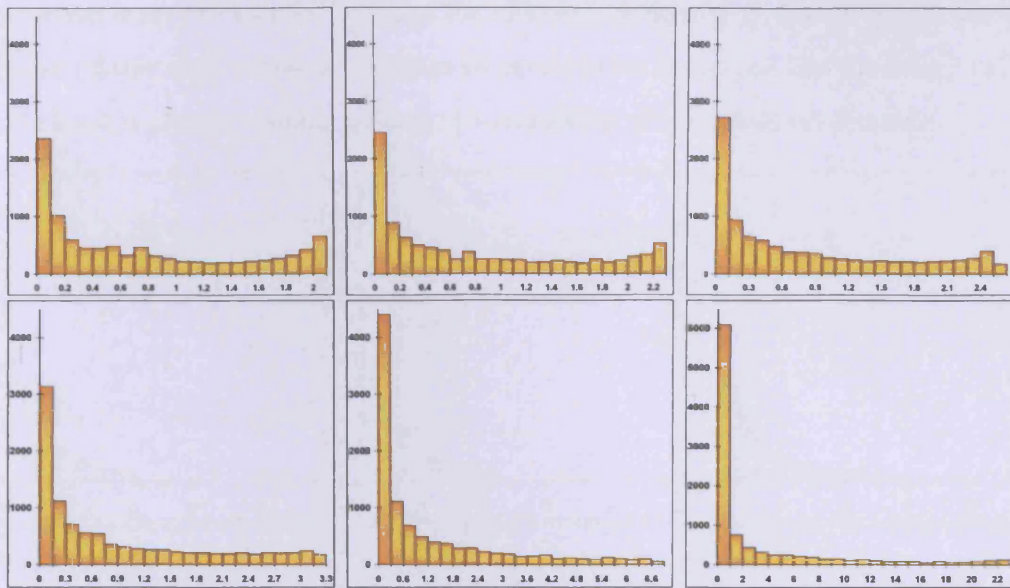


Figure 4.23: Histogram of  $r^{(k)}$  for, from top left to bottom right:  $\beta = 1.01$ ,  $\beta = 1.05$ ,  $\beta = 1.1$ ,  $\beta = 1.2$ ,  $\beta = 1.5$ ,  $\beta = 2$ ;  $\rho = 10$ ;  $d = 100$ ,  $k = 1, \dots, 10000$ .

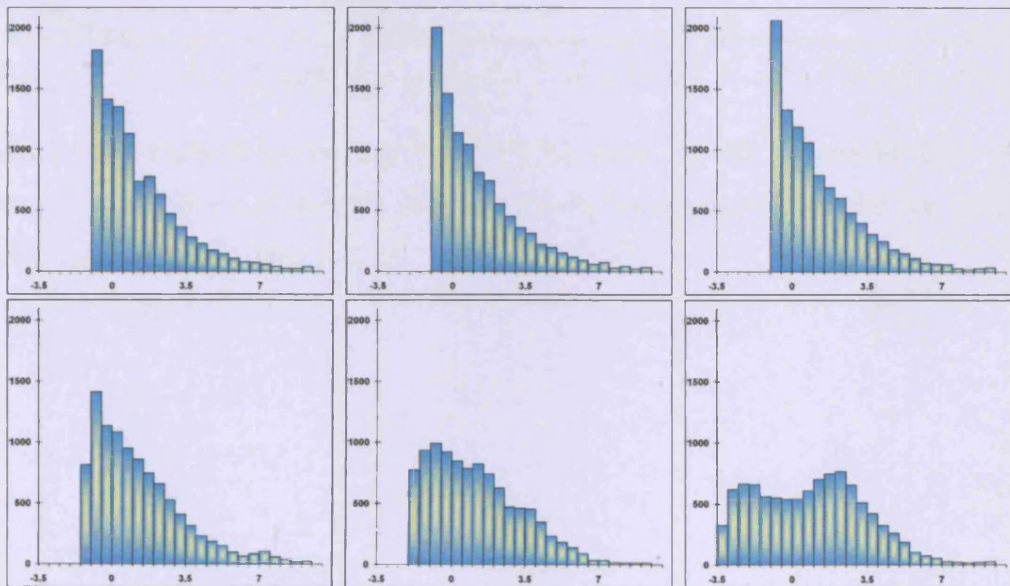


Figure 4.24: Histogram of  $(-\ln(r^{(k)}))$  for, from top left to bottom right:  $\beta = 1.01$ ,  $\beta = 1.05$ ,  $\beta = 1.1$ ,  $\beta = 1.2$ ,  $\beta = 1.5$ ,  $\beta = 2$ ;  $\rho = 10$ ;  $d = 100$ ,  $k = 1, \dots, 10000$ .

Further analysis of the rate,  $r^{(k)}$ , is enabled by exploring patterns in the transi-

tion of  $r^{(k)}$  from one iteration to the next. Figure 4.25 shows how patterns in the transition from  $r^{(k)}$  to  $r^{(k+1)}$  change for different choices of  $\beta$ . For  $\beta = 1.05$  there is a clear relationship between the rates in consecutive iterations but for larger values of  $\beta$  chaos is clearly visible, making the transition of rates less predictable.

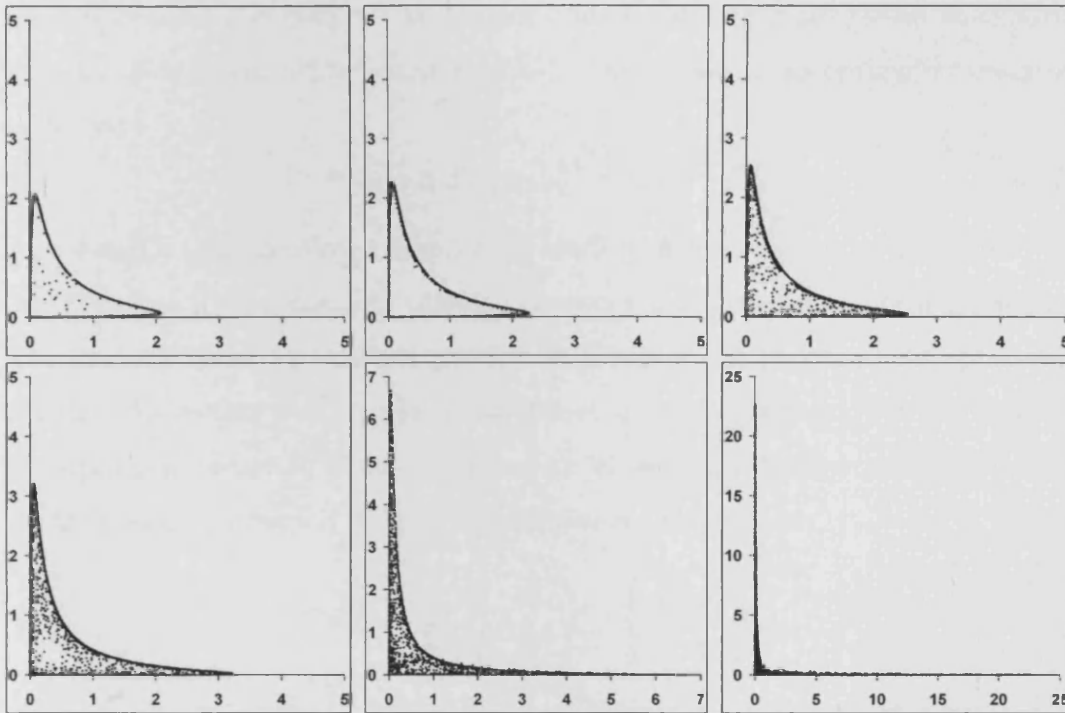


Figure 4.25: Plots of the pairs  $(r^{(k)}, r^{(k+1)})$  for, from top left to bottom right:  $\beta = 1.01$ ,  $\beta = 1.05$ ,  $\beta = 1.1$ ,  $\beta = 1.2$ ,  $\beta = 1.5$ ,  $\beta = 2$ ; Points plotted are the last 5000 of 10000 iterations;  $d = 100$ ,  $\rho = 10$ .

## 4.4 $(\gamma, \beta)$ -Root Algorithm

### 4.4.1 Generalisation of the $\beta$ -Root Algorithm

The  $\beta$ -root family of algorithms can be further extended to incorporate the  $\gamma$ -steepest descent and  $\gamma$ -square root algorithms by adding a relaxation coefficient,  $\gamma$ , to the  $\beta$ -root optimality criterion (4.14). This produces an optimality criterion of the form

$$\Phi(\xi) = \gamma\mu_0^\beta\mu_2^\beta - \mu_1^{2\beta} = \gamma\mu_2^\beta - \mu_1^{2\beta} \quad (4.20)$$

from which a corresponding renormalised gradient algorithm can be produced.

When  $\beta = 1$  the optimality criterion is exactly the  $\gamma$ -steepest descent optimality criterion and when  $\beta = 1/2$  the criterion is precisely the  $\gamma$ -square root optimality criterion. By setting  $\gamma = 1$  the standard  $\beta$ -root optimality criterion can be regained. The optimality criterion (4.20) can therefore be seen as a further generalisation of the D-optimality criterion where the determinant of the matrix

$$M(\xi) = \begin{pmatrix} \mu_0^\beta & \mu_1^\beta \\ \mu_1^\beta & \mu_2^\beta \end{pmatrix} + \begin{pmatrix} \tau & 0 \\ 0 & 0 \end{pmatrix},$$

where  $\tau = \gamma - 1$ , is taken as the criterion in place of the determinant of the matrix (2.17).

The relevant formulae required to produce the multiplicative algorithm which maximises the optimality criterion (4.20) and hence also produces a renormalised gradient algorithm are as follows:

$$\begin{aligned} \overset{\circ}{\Phi}(\xi) &= \begin{pmatrix} \frac{\partial\Phi}{\partial\mu_0} & \frac{1}{2}\frac{\partial\Phi}{\partial\mu_1} \\ \frac{1}{2}\frac{\partial\Phi}{\partial\mu_1} & \frac{\partial\Phi}{\partial\mu_2} \end{pmatrix} = \begin{pmatrix} \gamma\beta\mu_0^{\beta-1}\mu_2^\beta & -\beta\mu_1^{2\beta-1} \\ -\beta\mu_1^{2\beta-1} & \gamma\beta\mu_0^\beta\mu_2^{\beta-1} \end{pmatrix}, \\ \varphi(x, \xi) &= f^T(x) \overset{\circ}{\Phi}(\xi) f(x) = \beta \left( \gamma\mu_2^\beta - 2\mu_1^{2\beta-1}x + \gamma\mu_2^{\beta-1}x^2 \right), \\ c &= \min_x \varphi(x, \xi) = \beta \left( \gamma\mu_2^\beta - \frac{\mu_1^{2(2\beta-1)}}{\gamma\mu_2^{\beta-1}} \right), \\ b &= \text{tr} \left[ M(\xi) \overset{\circ}{\Phi}(\xi) \right] - c(\xi) = \frac{\beta}{\gamma} \left( \gamma^2\mu_2^\beta - 2\gamma\mu_1^{2\beta} + \mu_1^{2(2\beta-1)}\mu_2^{(1-\beta)} \right), \\ \alpha &= -2 \frac{\partial\Phi}{\partial\mu_2} / \frac{\partial\Phi}{\partial\mu_1} = \frac{\gamma\mu_2^{\beta-1}}{\mu_1^{2\beta-1}}. \end{aligned} \quad (4.21)$$



The resulting updating formula has the form

$$\xi^{(k+1)} = \frac{(1 - \lambda_i \gamma \mu_2^{\beta-1} / \mu_1^{2\beta-1})^2}{1 - 2\gamma \left(\frac{\mu_2}{\mu_1}\right)^{\beta-1} + \gamma^2 \left(\frac{\mu_2}{\mu_1}\right)^{2\beta-1}} \xi^{(k)} \quad \text{for } i = 1, \dots, d \quad (4.22)$$

and has a rate of convergence equal to

$$r^{(k)} = 1 - 2\gamma \left(\frac{\mu_2}{\mu_1}\right)^{\beta-1} + \gamma^2 \left(\frac{\mu_2}{\mu_1}\right)^{2\beta-1}.$$

The algorithm (4.22) shall be referred to as the  $(\gamma, \beta)$ -root algorithm.

To obtain the non-renormalised version it is noted that the step length (4.21) can be rewritten as

$$\alpha^{(k)} = \frac{\gamma \mu_2^{\beta-1}}{\mu_1^{2\beta-1}} = \frac{\gamma (g^{(k)}, g^{(k)})^\beta (A^2 g^{(k)}, g^{(k)})^{\beta-1}}{(A g^{(k)}, g^{(k)})^{2\beta-1}}$$

and hence the  $(\gamma, \beta)$ -root algorithms has the form

$$x^{(k+1)} = x^{(k)} - \frac{\gamma (g^{(k)}, g^{(k)})^\beta (A^2 g^{(k)}, g^{(k)})^{\beta-1}}{(A g^{(k)}, g^{(k)})^{2\beta-1}} g^{(k)}.$$

#### 4.4.2 Asymptotic Rate of Convergence

Figure 4.26 shows the average asymptotic rate of convergence of the  $(\gamma, \beta)$ -root algorithm as a function of  $\gamma$  for various values of  $\beta$ . For those values of  $\beta$  which have been observed to produce a  $\beta$ -root algorithm with the fastest asymptotic rates of convergence, the addition of a relaxation parameter,  $\gamma$ , does not improve the rate further. This can be seen in the top graph by the fact that the minimum point of each curve is at  $\gamma = 1$ .

For other values of  $\beta$ , a wise choice of relaxation coefficient,  $\gamma$ , will improve upon the rates observed for the same value of  $\beta$  when no relaxation parameter is used. The improvements yielded are not, however, enough to produce an algorithm with a faster asymptotic rate of convergence than the standard  $\beta$ -root algorithm with  $\beta$  slightly larger than 1.

For more insight into the behaviour of the rate of the  $(\gamma, \beta)$ -root algorithm, Figure 4.27 shows the corresponding plots of the attractors of  $r^{(k)}$  as a function of  $\gamma$ .

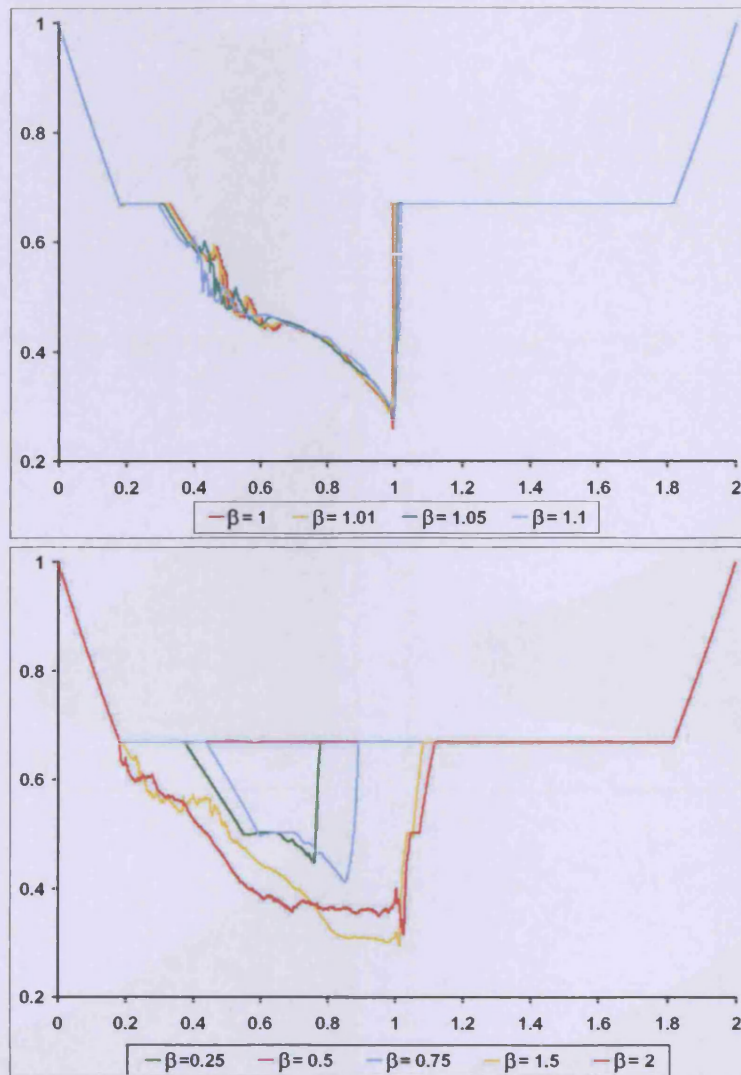


Figure 4.26: Average asymptotic rate of convergence as a function of  $\gamma$  for the  $(\gamma, \beta)$ -root algorithm with  $\beta = 1, 1.01, 1.05, 1.1$  (top); and  $\beta = 0.25, 0.5, 0.75, 1.2, 1.5, 2$  (bottom);  $d = 100, \rho = 10$ .

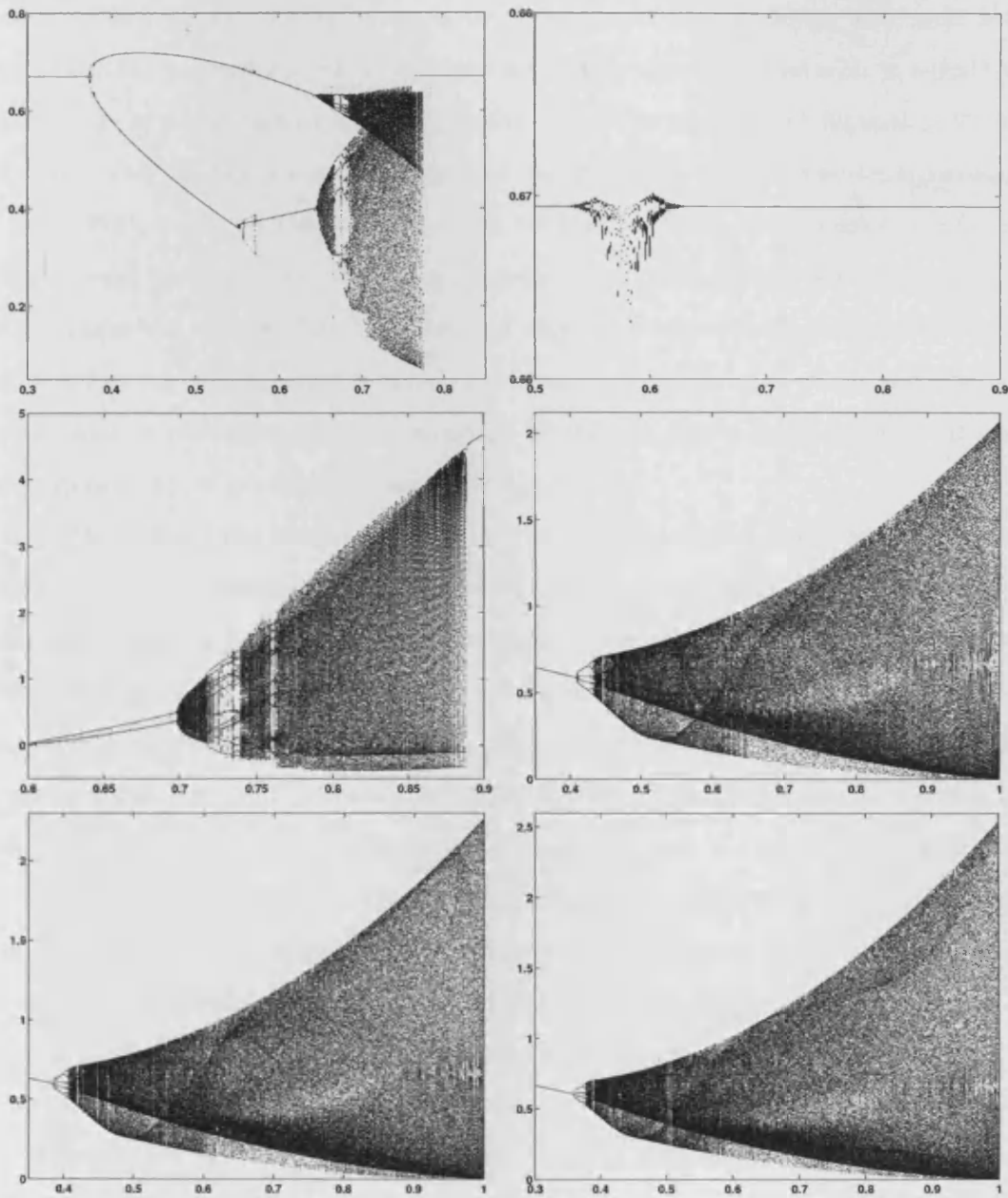


Figure 4.27: Attractors of  $r^{(k)}$  as a function of  $\gamma$  for the  $(\gamma, \beta)$ -root algorithm with from top to bottom  $\beta = 0.25, 0.5, 0.75, 1.01, 1.05, 1.1$ ;  $d = 100$ ,  $\rho = 10$ .

## 4.5 $\beta$ -Root Algorithm Summary

The square-root algorithm was introduced in Section 4.1. It was shown that the renormalised square-root algorithm converges to an optimum design and, as a consequence, the asymptotic rate of convergence of this algorithm was seen to equal the worst rate of the steepest descent algorithm,  $R_{\text{ref}}$ . The square-root algorithm therefore provides no improvement in rate over the standard steepest descent algorithm.

In Section 4.2, an attempt was made to improve upon the asymptotic rate of convergence by adding a relaxation coefficient  $\gamma$  to the square-root algorithm. It was shown that the renormalised  $\gamma$ -square root algorithm also converges to an optimum design for  $\gamma \geq 1$  and, apart from a small region of values of  $\gamma$  where the asymptotic rates of convergence were marginally better, the algorithm converges with an asymptotic rate of convergence equal to  $R_{\text{ref}}$ .

In Section 4.3 the steepest descent and square-root algorithms were both identified, by the structures of the optimality criterion corresponding to their renormalised forms, as belonging to a larger family of algorithms. This family, collectively known as the  $\beta$ -root algorithm, was shown to possess similar characteristics to the  $\gamma$ -steepest descent algorithm in the sense that for certain values of  $\beta$  the rate  $r^{(k)}$  converges to  $R_{\text{ref}}$ ; for other values of  $\beta$  the attractors of  $r^{(k)}$  form cycles of varying size and for many  $\beta$  the algorithm exhibits chaos. As was the case with the  $\gamma$ -steepest descent algorithm, the best asymptotic rates achievable occur in regions where chaos is demonstrated. The value of  $\beta$  which will form the  $\beta$ -root algorithm with the best possible asymptotic rate of convergence depends on the parameters  $\rho$  and  $d$ . In the case where  $\rho$  is unknown, it is advised to use  $\beta = 1.05$  since the asymptotic rate of convergence is relatively fast for all condition numbers when  $\beta = 1.05$ . It is noted that the  $\beta$ -root algorithm with  $\beta = 1.05$  is non-monotonic in nature.

Finally, in Section 4.4 the  $\beta$ -root algorithm is extended further to the  $(\gamma, \beta)$ -root algorithm by adding a relaxation coefficient  $\gamma$ . The  $\gamma$ -steepest descent and  $\gamma$ -square root algorithms both belong to the  $(\gamma, \beta)$ -root family. It is shown that an improvement in asymptotic rate over the  $\beta$ -root algorithm is possible but only for values of  $\beta$  where the  $\beta$ -root algorithm does not already demonstrate fast asymptotic

rates of convergence. For  $1 < \beta \lesssim 1.1$ , i.e. where the fastest asymptotic rates have been witnessed, no improvement in asymptotic rate of convergence is achievable by adding a relaxation coefficient  $\gamma$ . The best asymptotic rates observed so far have thus been produced by the  $\beta$ -root algorithm with  $\beta$  slightly greater than 1.

# Chapter 5

## $\Phi_p$ -optimality

In the previous two chapters the step lengths of the algorithms under scrutiny have been created from modifications and generalisations of the steepest descent step length. There is, however, another source of inspiration from which new step lengths for gradient algorithms can be generated. Due to the link, established in Chapter 2, between renormalised gradient algorithms and multiplicative algorithms for constructing optimal experimental designs, it is possible to create new step lengths for gradient algorithms based upon existing optimality criteria. In this chapter gradient algorithms created from the family of  $\Phi_p$ -optimality criteria will be developed and their asymptotic rates of convergence analysed.

### 5.1 A-Optimality

#### 5.1.1 The A-Optimality Criterion and the Corresponding Gradient Algorithm

A well-known optimality criterion in the field of optimal experimental design is that of A-optimality. A design  $\xi^*$  is said to be A-optimum if

$$\text{tr } M(\xi^*)^{-1} = \min_{\xi} \text{tr } M^{-1}(\xi).$$

Essentially an A-optimum design is one where the sum of the variances of the variables in the linear regression model  $y_j = \sum_{i=1}^m \theta_i f_i(x_j) + \varepsilon_j$ ,  $j = 1, \dots, N$ , is min-

imised. The corresponding criterion, as a functional of the information matrix, is

$$\Phi(M(\xi)) = \text{tr}M^{-1}(\xi).$$

For the base function  $f(x) = (1, x)^T$  this corresponds to

$$\Phi_{\text{Aopt}}(\xi) = \text{tr}M^{-1}(\xi) = \frac{\mu_0 + \mu_2}{\mu_0\mu_2 - \mu_1^2}. \quad (5.1)$$

In order to be consistent with  $D$ -optimality where the problem is one of maximisation, here the  $A$ -optimality criterion (5.1) is inverted so that

$$\Phi(\xi) = \frac{1}{\text{tr}M^{-1}(\xi)} = \frac{\mu_0\mu_2 - \mu_1^2}{\mu_0 + \mu_2} = \frac{\mu_2 - \mu_1^2}{1 + \mu_2}. \quad (5.2)$$

For this criterion

$$\begin{aligned} \overset{\circ}{\Phi}(\xi) &= \begin{pmatrix} \frac{\mu_2^2 + \mu_1^2}{(\mu_2 + \mu_0)^2} & -\frac{\mu_1}{\mu_2 + \mu_0} \\ -\frac{\mu_1}{\mu_2 + \mu_0} & \frac{\mu_0^2 + \mu_1^2}{(\mu_2 + \mu_0)^2} \end{pmatrix}, \\ \varphi(x, \xi) &= f^T(x) \overset{\circ}{\Phi}(\xi) f(x) = \frac{(x - \mu_1)^2 + (x\mu_1 - \mu_2)^2}{(\mu_2 + 1)^2}, \\ c &= \min_x \varphi(x, \xi) = \frac{(\mu_2 - \mu_1^2)^2}{(\mu_2 + 1)^2 (1 + \mu_1^2)} \end{aligned}$$

and

$$b = \text{tr} \left[ M(\xi) \overset{\circ}{\Phi}(\xi) \right] - c(\xi) = \frac{(\mu_1^2 \mu_2 + 1 + 2\mu_1^2)(\mu_2 - \mu_1^2)}{(\mu_2 + 1)^2 (1 + \mu_1^2)}.$$

Using (2.24) the resultant step length can be formulated as

$$\begin{aligned} \alpha_{\text{Aopt}}^{(k)} &= \alpha(\xi) = -2 \frac{\partial \Phi}{\partial \mu_2} / \frac{\partial \Phi}{\partial \mu_1} \\ &= \frac{1 + \mu_1^2}{\mu_1(1 + \mu_2)} \\ &= \frac{(g^{(k)}, g^{(k)})^2 + (Ag^{(k)}, g^{(k)})^2}{(Ag^{(k)}, g^{(k)}) [(g^{(k)}, g^{(k)}) + (A^2g^{(k)}, g^{(k)})]}. \end{aligned}$$

The gradient algorithm corresponding to the  $A$ -optimality criterion can thus be written

$$x^{(k+1)} = x^{(k)} - \frac{(g^{(k)}, g^{(k)})^2 + (Ag^{(k)}, g^{(k)})^2}{(Ag^{(k)}, g^{(k)}) [(g^{(k)}, g^{(k)}) + (A^2g^{(k)}, g^{(k)})]} g^{(k)}. \quad (5.3)$$

The algorithm (5.3) shall hereafter be referred to as the  $A$ -optimality gradient algorithm.

For the purpose of studying the asymptotic rate of convergence of the  $A$ -optimality gradient algorithm, the renormalised version of the algorithm will be used. Substituting  $\alpha_{\text{Aopt}}^{(k)}(\mu_1, \mu_2)$  into the general equation for renormalised gradient algorithms (2.9), gives the updating formula for the renormalised gradient algorithm corresponding to the  $A$ -optimality criterion as

$$\xi_i^{(k+1)} = \frac{(1 - \lambda_i(1 + \mu_1^2)/(\mu_1(1 + \mu_2)))^2}{1 - 2(1 + \mu_1^2)/(1 + \mu_2) + \mu_2(1 + \mu_1^2)^2/(\mu_1^2(1 + \mu_2)^2)} \xi_i^{(k)} \quad \text{for } i = 1, \dots, d. \quad (5.4)$$

The rate at iteration  $k$  associated with this algorithm is

$$r_{\text{Aopt}}^{(k)} = \frac{(1 + 2\mu_1^2 + \mu_1^2\mu_2)(\mu_2 - \mu_1^2)}{\mu_1^2(1 + \mu_2)^2}.$$

### 5.1.2 Behaviour of the Sequence $\{\Phi(\xi^{(k)})\}$

The behaviour of the sequence  $\{\Phi(\xi^{(k)})\}$  for the  $A$ -optimality criterion (5.2) is completely different to the behaviour of the sequence  $\{\Phi(\xi^{(k)})\}$  corresponding to the  $D$ -optimality criterion (2.28) in the sense that, instead of converging to a two-point cycle, the sequence exhibits chaos. In that respect the sequence behaves more like that of the  $\gamma$ -steepest descent criterion (3.3).

Figure 5.1 and Figure 5.2 show  $\Phi(\xi^{(k)})$  as a function of  $k$  for the  $A$ -optimality criterion for various values of the condition number  $\rho$ . As with the  $\gamma$ -steepest descent algorithm, the value of  $\Phi(\xi^{(k)})$  alternates between low and high values in a seemingly haphazard fashion. For  $A$ -optimality however, the range of values taken by the sequence is much smaller. The frequency with which the sequence jumps between high and low values is not constant but it can be observed that generally this frequency is higher when  $\rho$  is larger.

Figure 5.3 shows the distribution of  $\Phi(\xi^{(k)})$  for various values of  $\rho$ . The histograms demonstrate, that for small  $\rho$ , the majority of values of  $\Phi(\xi^{(k)})$  are located at either end of the range of possible values  $\Phi(\xi^{(k)})$  can take on with relatively few points falling more centrally. This is similar behaviour to the  $\gamma$ -steepest descent algorithm with  $\gamma$  close to one. For larger values of  $\rho$  the distribution shows the sequence to assume more and more values from the upper end of the range of possible values.

Plots of the pairs  $(\Phi(\xi^{(k)}), \Phi(\xi^{(k+1)}))$  in Figure 5.4 show chaos to be clearly



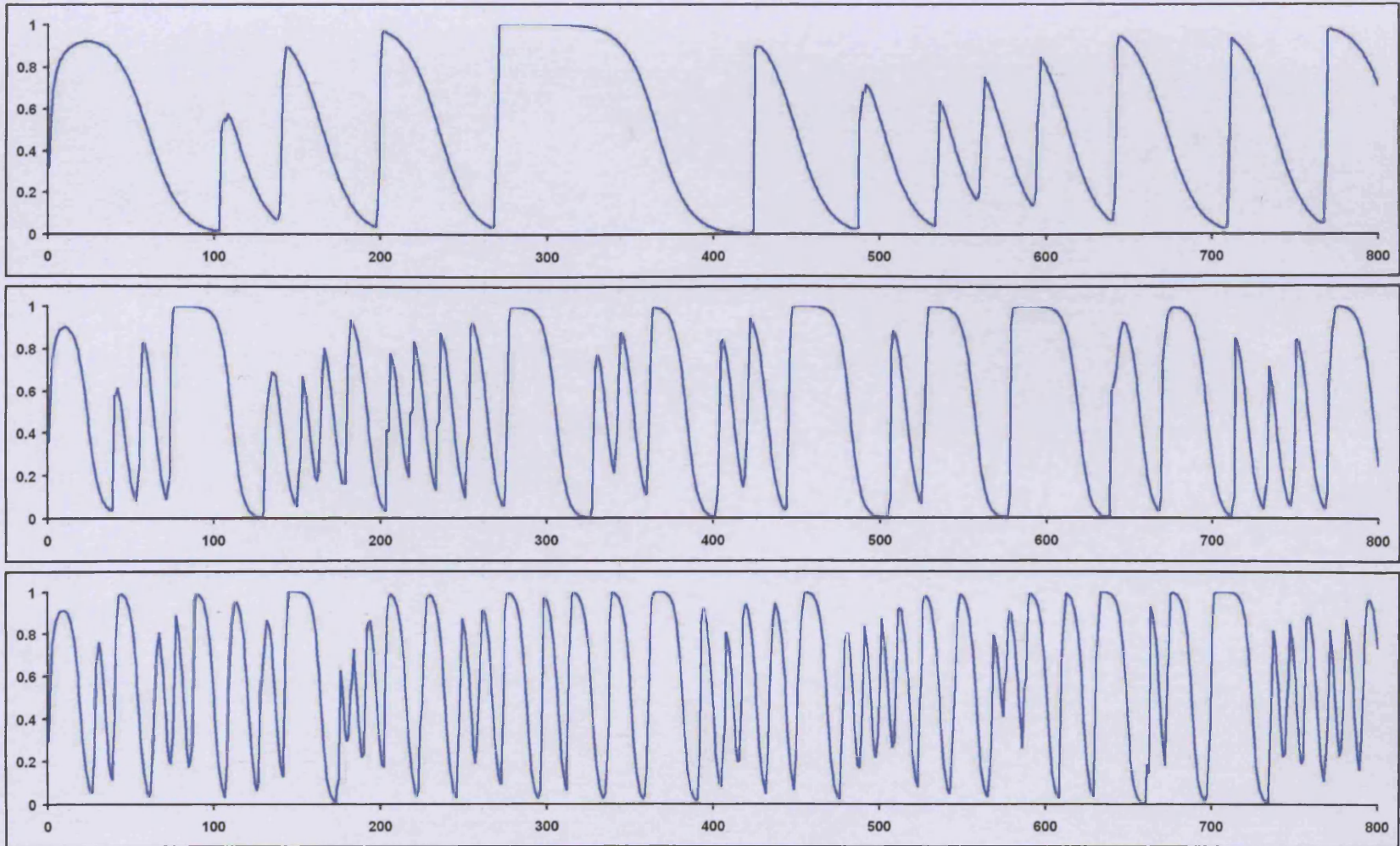


Figure 5.1:  $\Phi(\xi^{(k)})$  as a function of  $k$  for from top to bottom:  $\rho = 2$ ,  $\rho = 4$ ,  $\rho = 10$ ;  $d = 100$ .

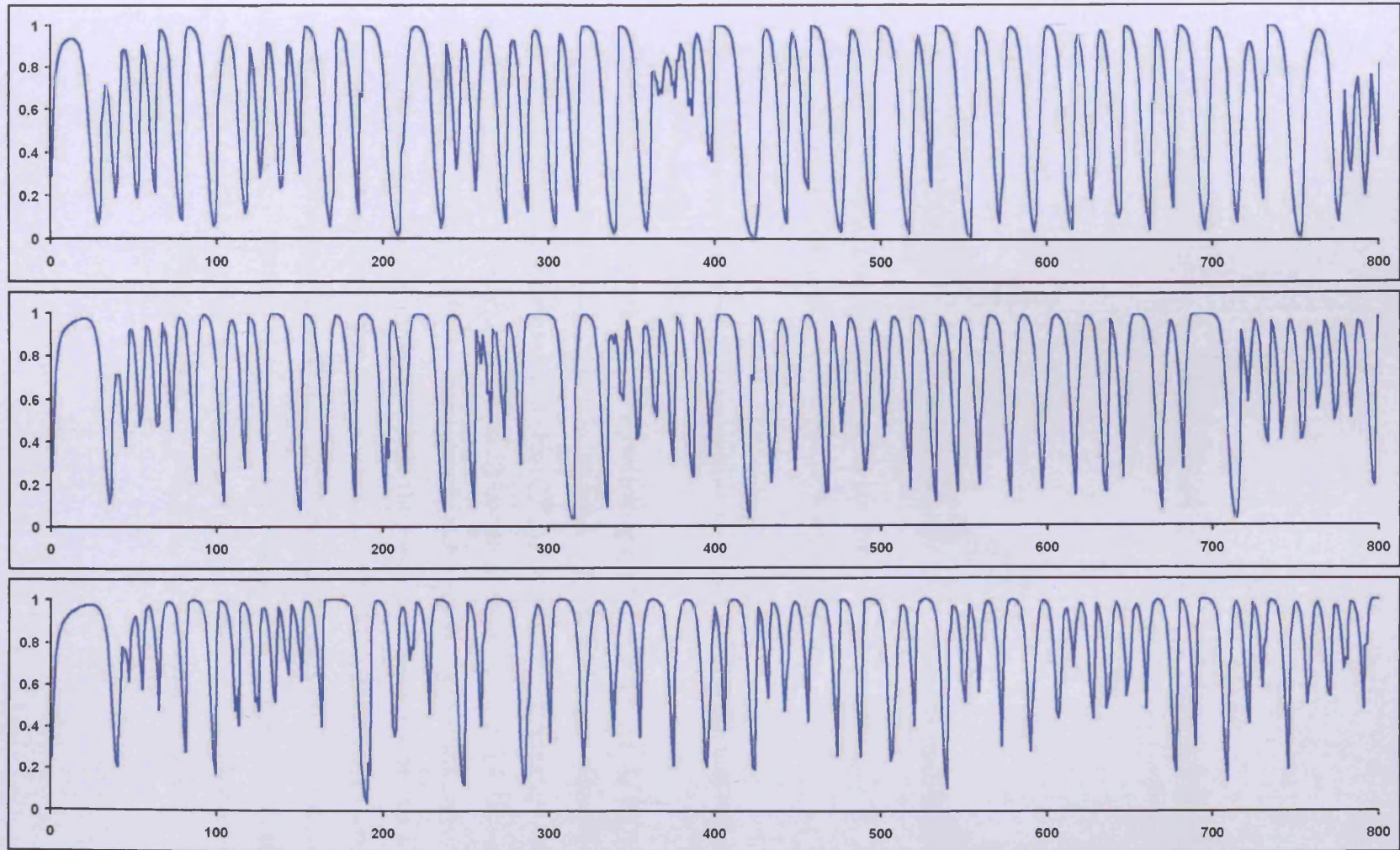


Figure 5.2:  $\Phi(\xi^{(k)})$  as a function of  $k$  for from top to bottom:  $\rho = 20$ ,  $\rho = 50$ ,  $\rho = 100$ ;  $d = 100$ .

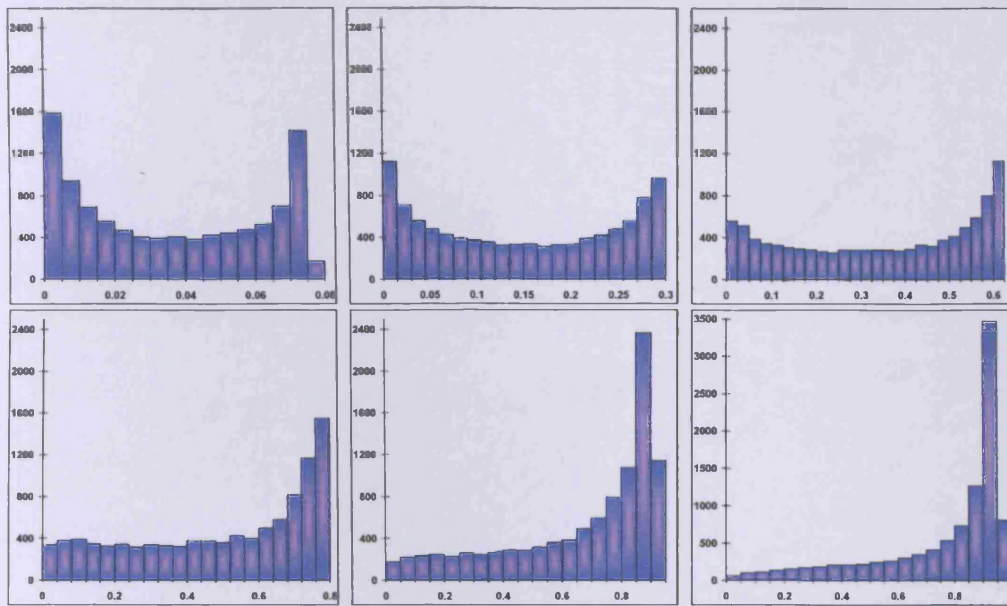


Figure 5.3: Histogram of  $\Phi(\xi^{(k)})$  for, from top left to bottom right:  $\rho = 2$ ,  $\rho = 4$ ,  $\rho = 10$ ,  $\rho = 20$ ;  $\rho = 50$ ,  $\rho = 100$ ;  $d = 100$ ,  $k = 1, \dots, 10000$ .

present in the  $A$ -optimality gradient algorithm making the transition between  $\Phi(\xi^{(k)})$  and  $\Phi(\xi^{(k+1)})$  unpredictable.

Figure 5.5 shows the weight at  $\lambda_1$  plotted against the weight at  $\lambda_d$  for various  $\rho$ . Akin to the situation with the renormalised  $\gamma$ -steepest descent algorithm, the dense scattering of points close to the line  $\xi^{(k)}(\lambda_d) = 1 - \xi^{(k)}(\lambda_1)$  suggests that the middle weights, i.e.  $\xi^{(k)}(\lambda_2), \dots, \xi^{(k)}(\lambda_{d-1})$  are often reduced to near 0 throughout the sequence. In fact the renormalised  $A$ -optimality gradient algorithm attempts to attract to the two-dimensional plane with the basis  $e_1, e_d$  by reducing the weights of the designs  $\xi^{(k)}$  at the rest of the eigenvalues. When the plane has almost been reached, the convergence rate of the algorithm accelerates and the updating rule quickly regains the weights of the other components. The process then restarts, essentially at random, which creates chaos. Figure 5.6 demonstrates the change in the weights  $\xi^{(k)}(\lambda_1)$ ,  $\xi^{(k)}(\lambda_d)$ , and  $\sum_{i=2}^{d-1} \xi^{(k)}(\lambda_i)$  as the iterations progress.

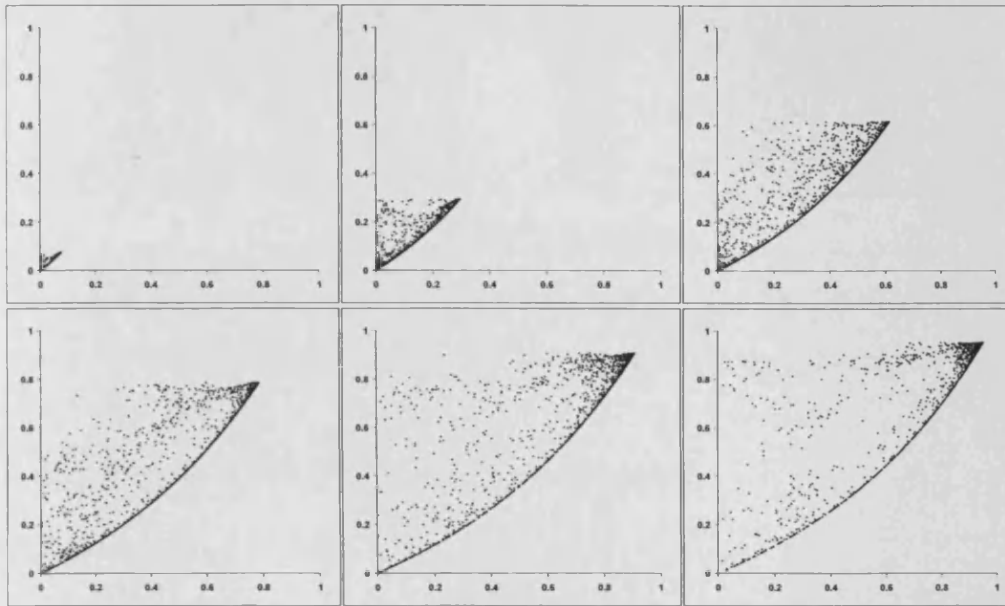


Figure 5.4: Plots of the pairs  $(\Phi(\xi^{(k)}), \Phi(\xi^{(k+1)}))$  for, from top left to bottom right  $\rho = 2, \rho = 4, \rho = 10, \rho = 20, \rho = 50, \rho = 100$ . Points plotted are the last 2000 of 10000 iterations;  $d = 100$ .

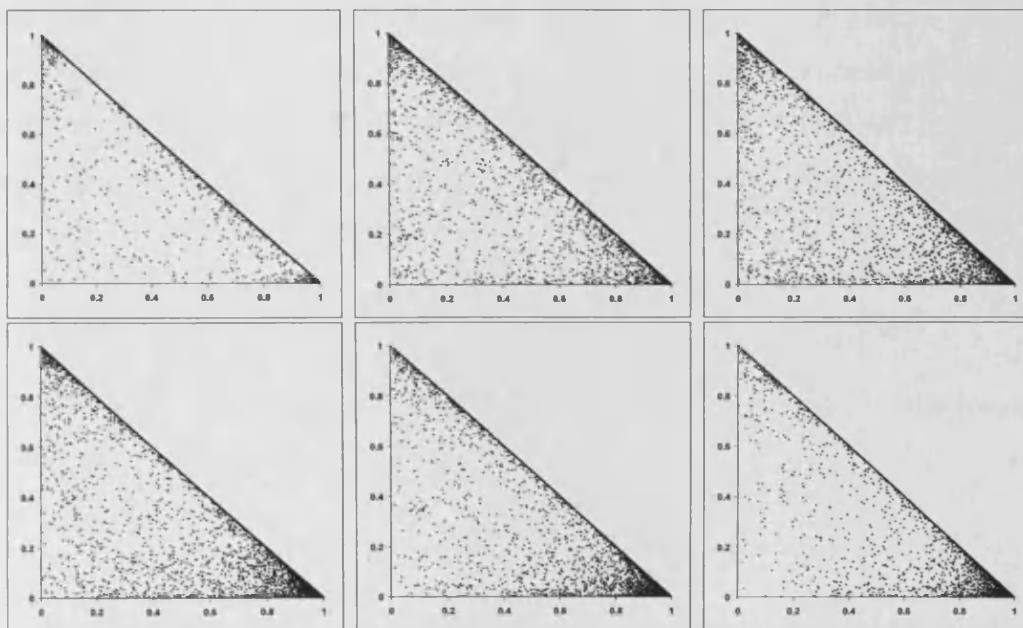


Figure 5.5: Plots of the pairs  $(\xi^{(k)}(\lambda_1), \xi^{(k)}(\lambda_d))$  for the  $A$ -optimality gradient algorithm with (from top left to bottom right):  $\rho = 2, 4, 10, 20, 50, 100$ ;  $d = 100$ . Points plotted are the last 8000 of 10000 iterations.

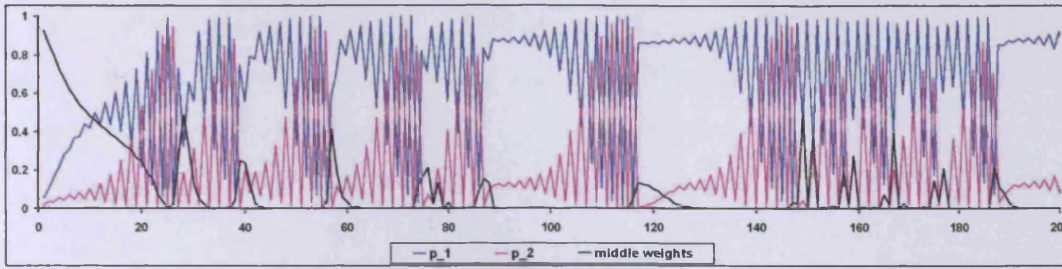


Figure 5.6:  $\xi^{(k)}(\lambda_1)$  (denoted  $p_1$ ),  $\xi^{(k)}(\lambda_d)$  (denoted  $p_2$ ) and  $\sum_{i=2}^{d-1} \xi^{(k)}(\lambda_i)$  as a function of  $k$  for the  $A$ -optimality gradient algorithm;  $\rho = 10$ ,  $d = 100$ .

### 5.1.3 Asymptotic Rate of Convergence

#### The 2-Dimensional Case

The  $A$ -optimality gradient algorithm has super-linear convergence when  $d = 2$ . Consider the behaviour of the gradient algorithm generated by the  $A$ -optimality criterion in the two-dimensional case; that is, when  $d = 2$ ,  $\lambda_1 = m$  and  $\lambda_2 = M$ . Assume that the initial point  $x^{(0)}$  is such that  $0 < \xi^{(0)}(m) < 1$  (otherwise the initial design  $\xi^{(0)}$  would be degenerate as would all other designs  $\xi^{(k)}$ ,  $k \geq 1$ ).

Denote  $\xi_1^{(k)} = \xi^{(k)}(m)$  for  $k = 0, 1, \dots$ . As  $d = 2$ , all the designs  $\xi^{(k)}$  are fully described by the corresponding values of  $\xi_1^{(k)}$  since  $\xi_d^{(k)} = \xi^{(k)}(M) = 1 - \xi_1^{(k)}$ . The updating formula for the weight  $\xi_1^{(k)}$  is  $\xi_1^{(k+1)} = f(\xi_1^{(k)})$  where, for the renormalised  $A$ -optimality gradient algorithm,

$$f(\xi_1) = \left(1 - \frac{(1 + \mu_1^2)}{\mu_1(1 + \mu_2)} m\right)^2 \frac{\mu_1^2(1 + \mu_2)^2}{(1 + 2\mu_1^2 + \mu_1^2\mu_2)(\mu_2 - \mu_1^2)} \xi_1,$$

$\mu_1 = \xi_1 m + (1 - \xi_1)M$  and  $\mu_2 = \xi_1 m^2 + (1 - \xi_1)M^2$ . The fixed point of the transformation  $\xi_1^{(k+1)} = f(\xi_1^{(k)})$  is

$$\xi_1^* = \frac{M^2 + 1 - \sqrt{(M^2 + 1)(m^2 + 1)}}{M^2 - m^2}.$$

For this point we have  $\xi_1^* = f(\xi_1^*)$ . The design with the mass  $\xi_1^*$  at  $m$  and mass  $(1 - \xi_1^*)$  at  $M$  is the  $A$ -optimum design for the linear regression model  $y_j = \theta_0 + \theta_1 x_j + \varepsilon_j$  on the interval  $[m, M]$ . This fixed point  $\xi_1^*$  is unstable for the mapping  $\xi_1 \rightarrow f(\xi_1)$  as  $|f'(\xi_1^*)| > 1$ .

For the transformation  $f^2(\cdot) = f(f(\cdot))$ , see Figure 5.7 for an illustration of this map, there are two stable fixed points which are 0 and 1. The fact that the points 0 and 1 are stable for the mapping  $\xi_1 \rightarrow f^2(\xi_1)$  follows from

$$(f(f(\xi_1)))' \Big|_{\xi_1=0} = (f(f(\xi_1)))' \Big|_{\xi_1=1} = f'(0)f'(1) = \frac{(Mm+1)^4}{(m^2+1)^2(M^2+1)^2};$$

the right-hand side of this formula is always positive and less than 1. There is a third fixed point for the mapping  $\xi_1 \rightarrow f^2(\xi_1)$ ; this is of course  $\xi_1^*$  which is clearly unstable.

This implies that in the two-dimensional case the sequence of measures  $\xi^{(k)}$  attracts (as  $k \rightarrow \infty$ ) to a cycle of oscillations between two degenerate measures, one is concentrated at  $m$  and the other one is concentrated at  $M$ . The super-linear convergence of the corresponding gradient algorithm follows from the fact that the rates  $r(\xi)$  at these two degenerate measures are 0 (implying  $r^{(k)} \rightarrow 0$  as  $k \rightarrow \infty$  for the sequence of rates  $r^{(k)}$ ).

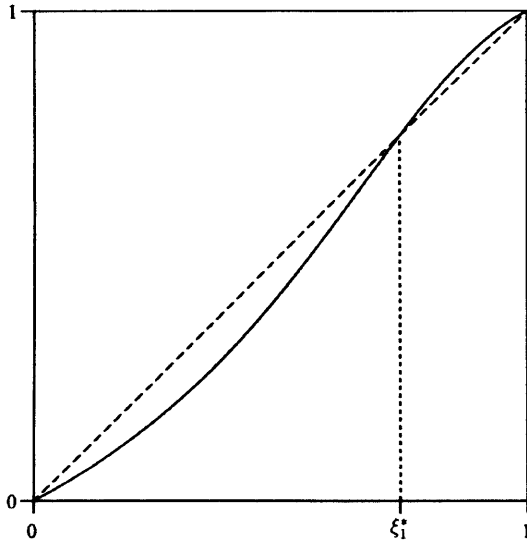


Figure 5.7: Graph depicting the transformation  $f^2(\cdot)$  for the A-optimality gradient algorithm with  $m = 1$ ,  $M = 4$ ;  $d = 2$ .

### Asymptotic Rate of Convergence when $d > 2$

For almost all starting points  $x^{(0)}$ , the gradient algorithm corresponding to the A-optimality criterion for  $d = 2$  has super-linear convergence in the sense that the

sequence of rates  $r^{(k)}$  tends to 0 as  $k \rightarrow \infty$ . If the dimension  $d$  is larger than 2, however, the convergence of the optimisation algorithm generated by the  $A$ -optimality criterion is no longer super-linear.

The asymptotic rate of convergence of the  $A$ -optimality gradient algorithm is shown as a function of  $d$  in Figure 5.8 and as a function of  $\rho$  in Figure 5.9. All algorithms presented in this thesis have been shown to have little dependence on the dimension  $d$  and the  $A$ -optimality gradient algorithm is no exception, as can be seen by the approximately constant asymptotic rate of convergence for all values of  $d \gtrsim 10$ . The asymptotic rate of convergence worsens as  $\rho$  increases however a considerable improvement over the standard steepest descent algorithm is apparent when this rate is compared with the worst case rate of the steepest descent algorithm,  $R_{\text{ref}}$ .

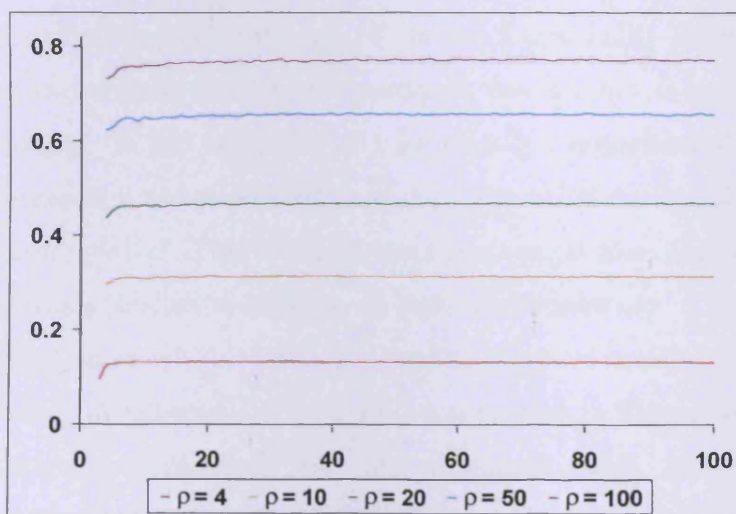


Figure 5.8: Average asymptotic rate of convergence as a function of  $d$  for the  $A$ -optimality gradient algorithm.

Figure 5.10 and Figure 5.11 show  $r^{(k)}$  as a function of  $k$  for various different condition numbers  $\rho$ . Unlike the  $\gamma$ -steepest descent algorithm with  $\gamma$  close to 1, where an initial oscillatory period takes place before the algorithm reaches the more desirable lower rates, the sequence  $\{r^{(k)}\}$  for the  $A$ -optimality gradient algorithm descends straight into chaotic behaviour. The advantageous rates close to 0 are not reached with the same frequency as they are in the  $\gamma$ -steepest descent algorithm with

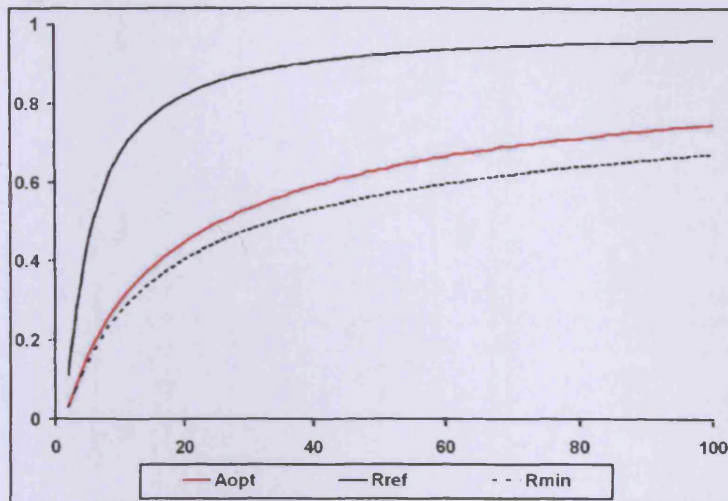


Figure 5.9: Average asymptotic rate of convergence as a function of  $\rho$  for the A-optimality gradient algorithm.

suitable choice of  $\gamma$ , however, the rates  $r^{(k)}$  in the A-optimality gradient algorithm are much less varied than those in the  $\gamma$ -steepest descent and so no extremely bad rates are produced. In fact, since  $r^{(k)} < 1$  for all  $k \geq 1$  regardless of the condition number of the problem, the A-optimality gradient algorithm monotonically decreases to the minimum point  $x^*$ . This is not necessarily a sought-after characteristic since the best algorithms studied so far have all been non-monotonic.

The distribution of  $r^{(k)}$  for different values of  $\rho$  can be seen in Figure 5.12 and the corresponding distributions of  $(-\ln r^{(k)})$  can be seen in Figure 5.13. For larger values of  $\rho$  the range of values  $r^{(k)}$  can take on is much wider. Also, the larger the value of  $\rho$ , the greater the proportion of  $r^{(k)}$  there is taking on high values (close to 1). This is true of all gradient algorithms and shows why the overall asymptotic rate of convergence always worsens as  $\rho$  increases.

The transition from  $r^{(k)}$  to  $r^{(k+1)}$  is shown in Figure 5.14. It is evident that there is chaos present in the sequence  $\{r^{(k)}\}$  however there is also a pattern present in the form of a denser line of points. This indicates that at least in some places throughout the course of the sequence a particular value of  $r^{(k)}$  will give rise to a slightly smaller value of  $r^{(k+1)}$  gradually decreasing the rate until chaos enters the sequence again and eventually larger values of  $r^{(k)}$  are produced for the pattern to then start again.



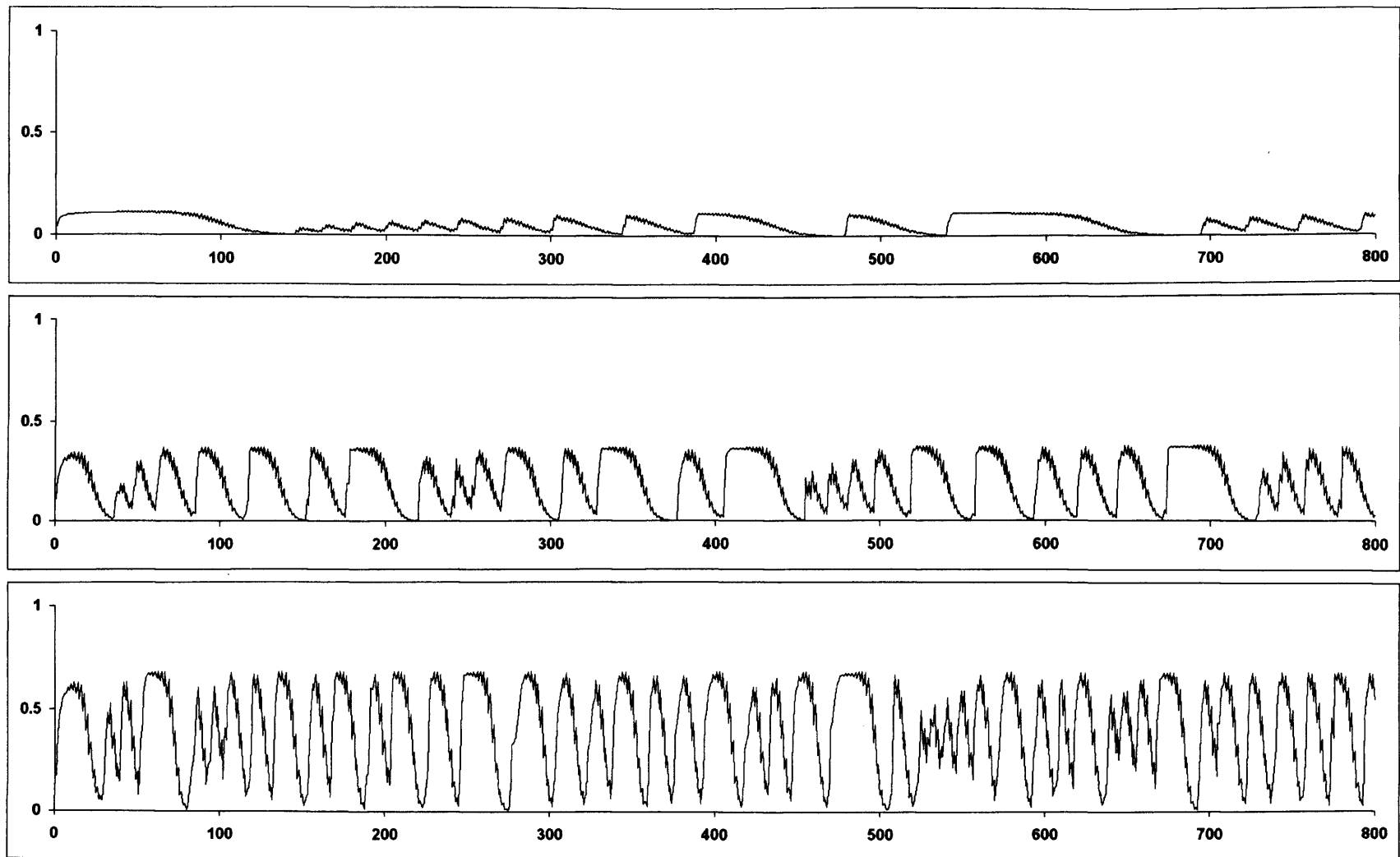


Figure 5.10: Rate,  $r^{(k)}$ , as a function of  $k$  for from top to bottom:  $\rho = 2$ ,  $\rho = 4$ ,  $\rho = 10$ ;  $d = 100$ .

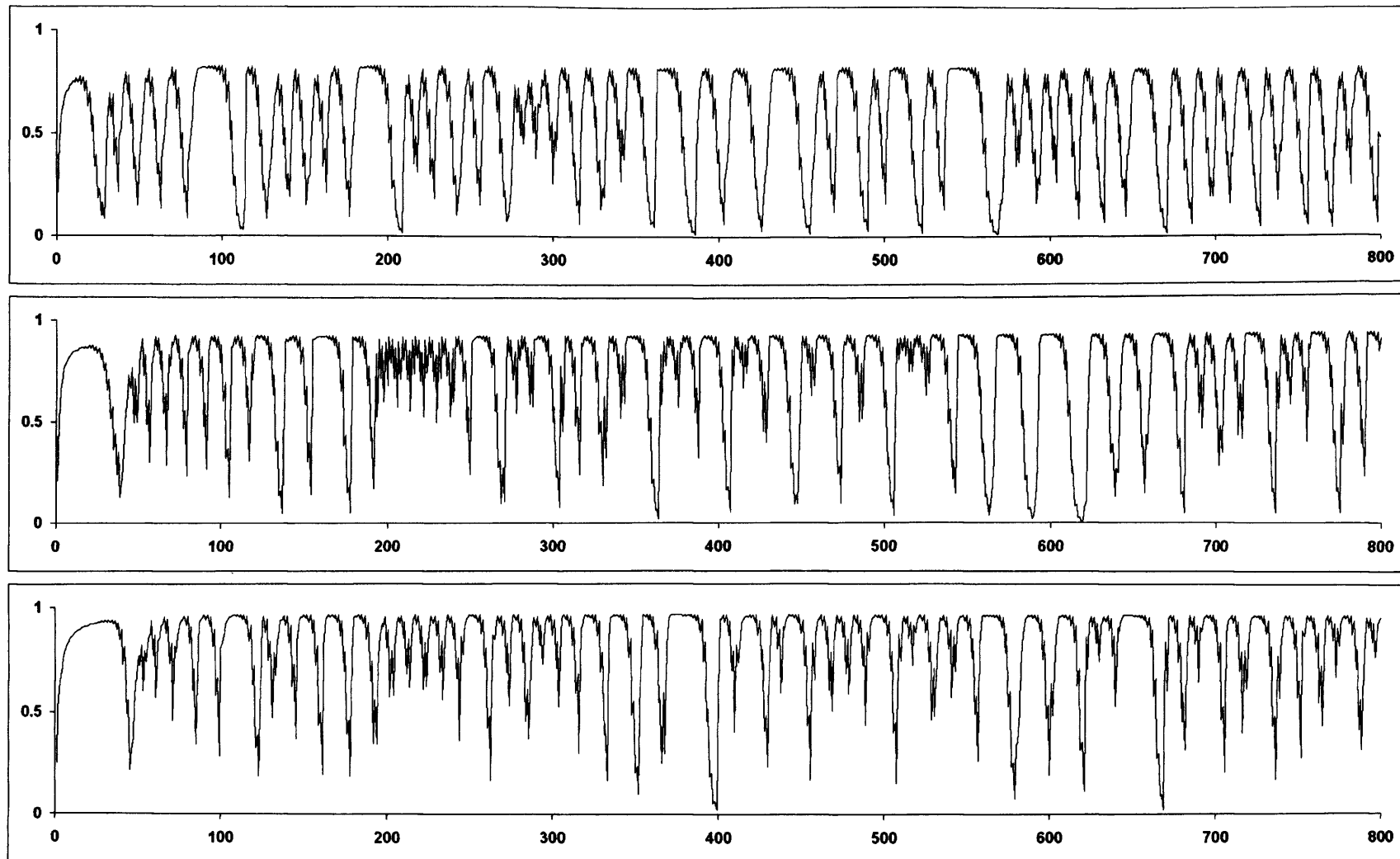


Figure 5.11: Rate,  $r^{(k)}$ , as a function of  $k$  for from top to bottom:  $\rho = 20$ ,  $\rho = 50$ ,  $\rho = 100$ ;  $d = 100$ .

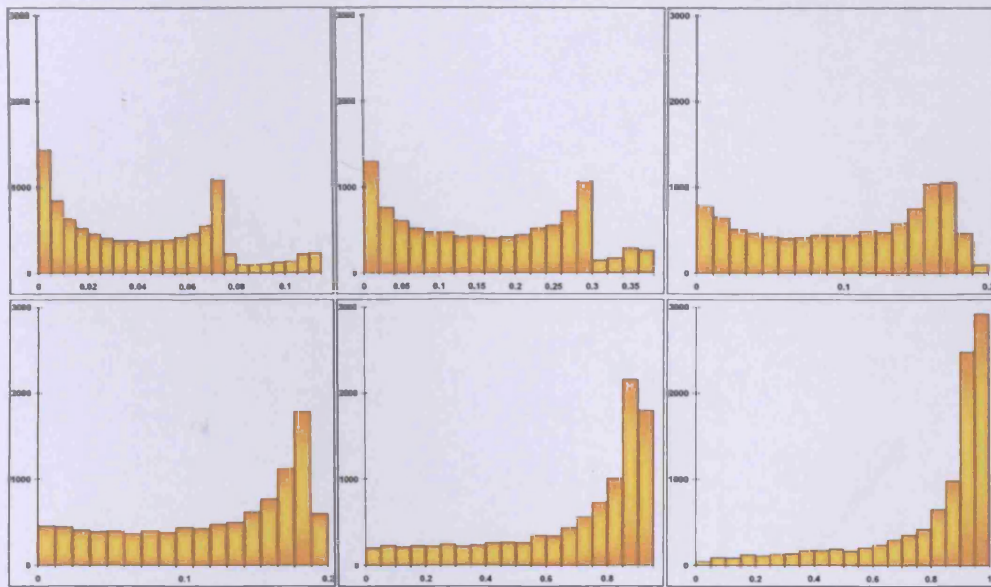


Figure 5.12: Histogram of  $r^{(k)}$  for, from top left to bottom right:  $\rho = 2$ ,  $\rho = 4$ ,  $\rho = 10$ ,  $\rho = 20$ ,  $\rho = 50$ ,  $\rho = 100$ ;  $d = 100$ ,  $k = 1, \dots, 10000$ .

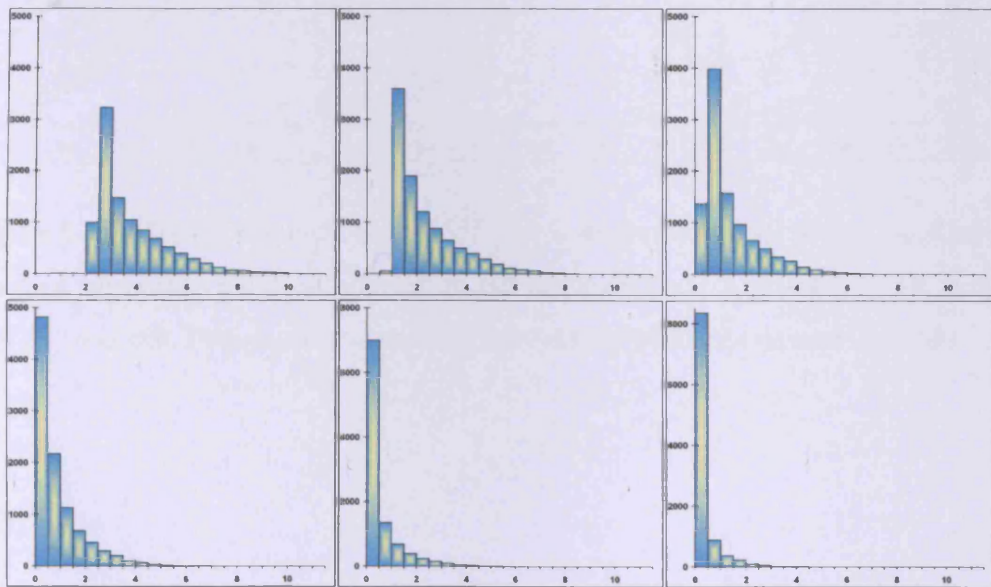


Figure 5.13: Histogram of  $(-\ln(r^{(k)}))$  for, from top left to bottom right:  $\rho = 2$ ,  $\rho = 4$ ,  $\rho = 10$ ,  $\rho = 20$ ,  $\rho = 50$ ,  $\rho = 100$ ;  $d = 100$ ,  $k = 1, \dots, 10000$ .

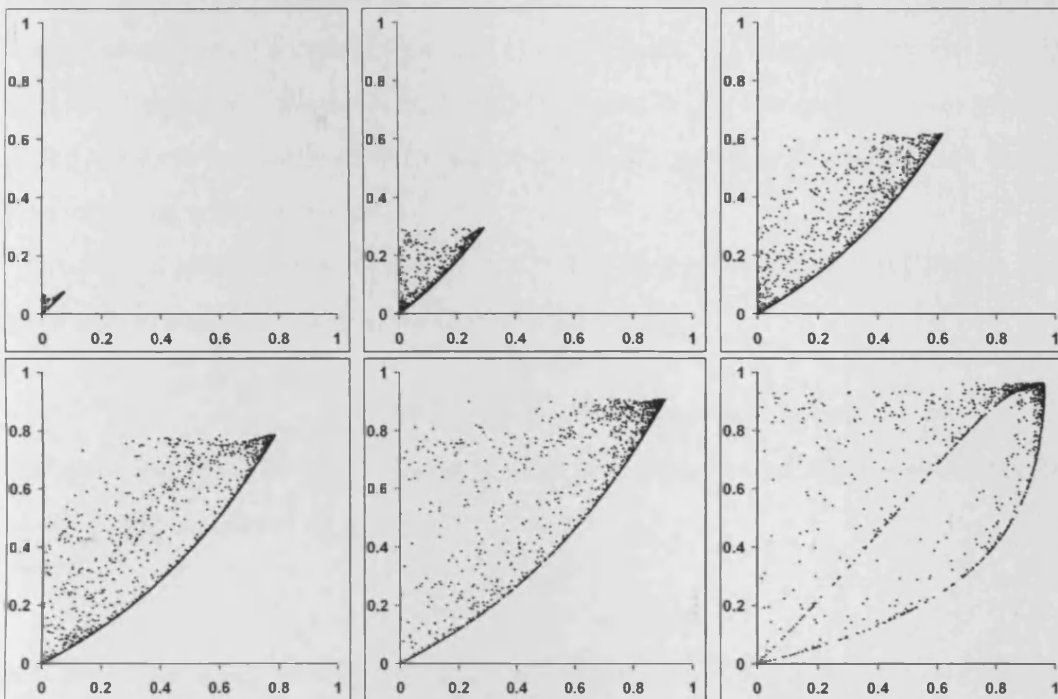


Figure 5.14: Plots of the pairs  $(r^{(k)}, r^{(k+1)})$  for a single trajectory of the  $A$ -optimality gradient algorithm for, from top left to bottom right:  $\rho = 2$ ,  $\rho = 4$ ,  $\rho = 10$ ,  $\rho = 20$ ,  $\rho = 50$ ,  $\rho = 100$ . Points plotted are the last 5000 of 10000 iterations;  $d = 100$ .

## 5.2 A-Optimality with Relaxation

### 5.2.1 The Renormalised $(\gamma, A)$ -Optimality Gradient Algorithm

The addition of an appropriate relaxation coefficient,  $\gamma$ , to the steepest descent algorithm has proved to be greatly beneficial in terms of the improvement in asymptotic rate of convergence it yielded. The  $A$ -optimality gradient algorithm (5.3) has already demonstrated an impressive rate of convergence in comparison to the original steepest descent algorithm without added relaxation, so it is possible that incorporating a relaxation coefficient into the  $A$ -optimality gradient algorithm will further ameliorate its convergence rate.

Adding a relaxation coefficient  $\gamma$  to the standard  $A$ -optimality step length (5.3) gives rise to the  $(\gamma, A)$ -optimality gradient algorithm

$$x^{(k+1)} = x^{(k)} - \gamma \frac{(g^{(k)}, g^{(k)})^2 + (Ag^{(k)}, g^{(k)})^2}{(Ag^{(k)}, g^{(k)}) [(g^{(k)}, g^{(k)}) + (A^2g^{(k)}, g^{(k)})]} g^{(k)}. \quad (5.5)$$

The step length of the  $(\gamma, A)$ -optimality gradient algorithm (5.5) can be written in terms of the moments,  $\mu_1$  and  $\mu_2$ , as

$$\alpha_{\gamma A \text{opt}}^{(k)} = \gamma \frac{1 + \mu_1^2}{\mu_1(1 + \mu_2)}$$

and hence, using the general formula for renormalised gradient algorithms (2.9), the renormalised version of the  $(\gamma, A)$ -optimality gradient algorithm can be written as

$$\xi_i^{(k+1)} = \xi_i^{(k)} \left( 1 - \lambda_i \frac{\gamma(1 + \mu_1^2)}{\mu_1(1 + \mu_2)} \right)^2 \Big/ \left( \frac{\gamma^2 \mu_2(1 + \mu_1^2)^2}{\mu_1^2(1 + \mu_2)^2} - 2\gamma \frac{(1 + \mu_1^2)}{1 + \mu_2} + 1 \right),$$

for  $i = 1, \dots, d$ . The formula for the updating of the weights for the renormalised  $A$ -optimality gradient algorithm can be recovered by setting  $\gamma = 1$ . The rate at iteration  $k$  corresponding to the  $(\gamma, A)$ -optimality gradient algorithm is

$$r^{(k)} = \gamma^2 \frac{\mu_2(1 + \mu_1^2)^2}{\mu_1^2(1 + \mu_2)^2} - 2\gamma \frac{(1 + \mu_1^2)}{1 + \mu_2} + 1.$$

### 5.2.2 Asymptotic Rate of Convergence

Figure 5.15 and Figure 5.16 show the average asymptotic rate of convergence for the  $(\gamma, A)$ optimality algorithm as a function of the relaxation coefficient  $\gamma$ , for varying

values of the condition number  $\rho$  and dimension  $d$ . As can be seen from all of these graphs, the minimum point and thus the fastest rate of convergence occurs in the region of  $\gamma = 1$ . The exact value of  $\gamma$  for which the convergence rate is fastest appears to depend on the value of  $\rho$  but only varies very slightly, being either marginally less than 1, 1, or marginally greater than 1. The increase in convergence rate gained by using this optimum value of  $\gamma$  as opposed to simply taking  $\gamma = 1$  is however seemingly negligible; the extra speed of convergence gained is probably not worth the cumbersome calculations needed due to having a more complicated and less natural updating formula. The evidence does not therefore suggest the addition of a relaxation coefficient to the  $A$ -optimality algorithm is worthwhile. Having said that, there is nevertheless a range of  $\gamma$ , the width of which grows with  $\rho$ , for which the  $(\gamma, A)$ -optimality gradient algorithm still gives rise to a better asymptotic rate of convergence than the worst case rate of the steepest descent algorithm,  $R_{\text{ref}}$ .

From Figure 5.15 it can be seen that, similar to the all algorithms considered so far, the asymptotic rate of convergence worsens as  $\rho$  increases. Furthermore, consistent with what has been observed in other algorithms, the dimensionality does not seem to play much of a part in effecting the convergence rate. For  $d = 10$  and higher there does not seem to be any significant difference in the asymptotic rate of convergence observed for the whole range of  $\gamma$  studied, see Figure 5.16.

In order to obtain a clearer picture of how the value of the relaxation parameter affects the asymptotic behaviour of the algorithm Figure 5.17 and Figure 5.18 show the rate  $r^{(k)}$  at iteration  $k$  for a single trajectory plotted as a function of  $\gamma$  for various values of  $\rho$  and  $d$ . For the  $\gamma$ -steepest descent algorithm, the presence of chaos indicated areas where faster convergence rates were achieved, the situation is same with the  $(\gamma, A)$ -optimality gradient algorithm. Comparing Figure 5.15 with Figure 5.17 it can be seen that the range of  $\gamma$  for which chaotic behaviour is exhibited coincides with a region where a rate superior to the worst case rate of the steepest descent algorithm is attained.

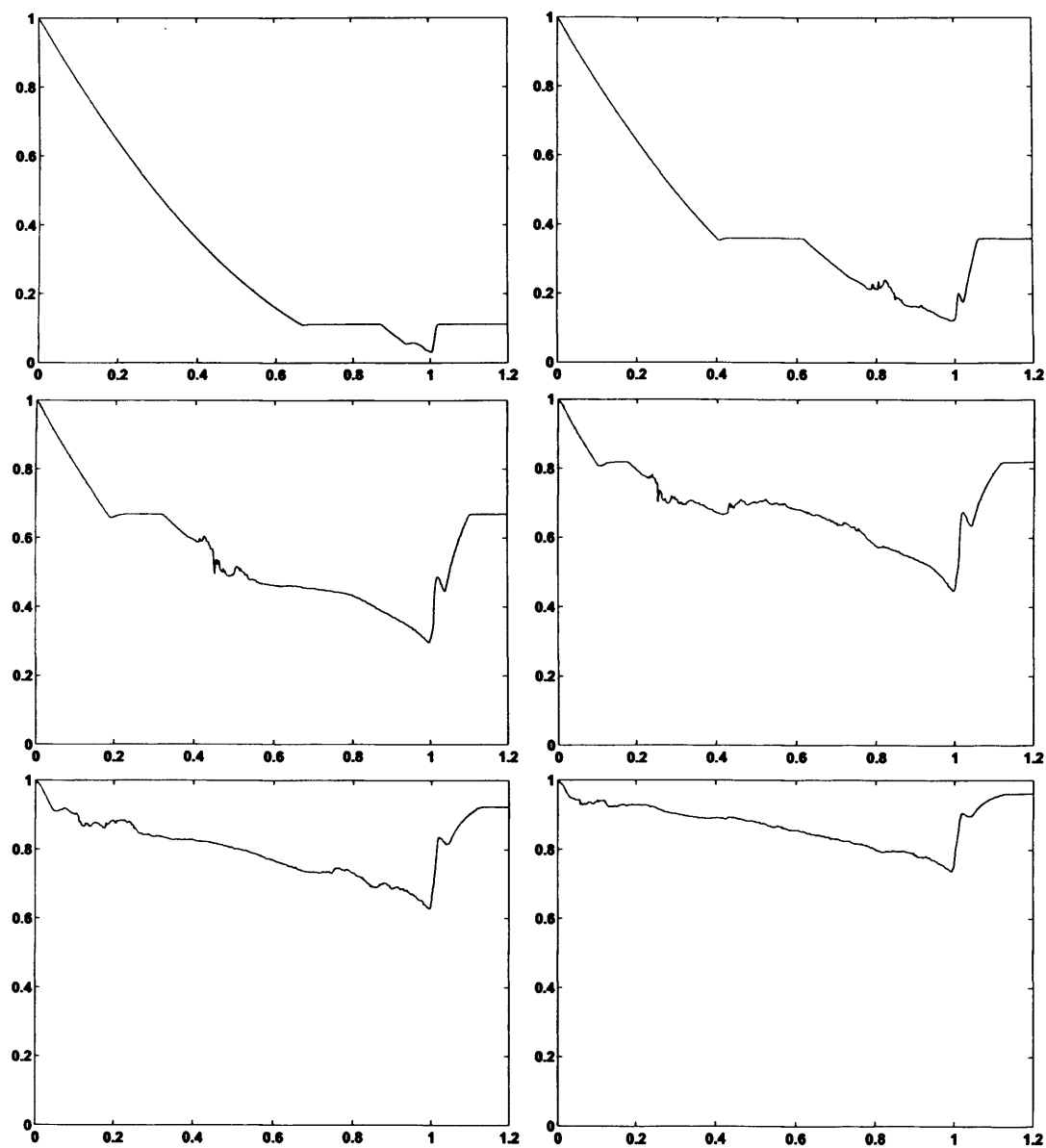


Figure 5.15: Average asymptotic rate of convergence as a function of  $\gamma$  for the  $(\gamma, A)$ -optimality gradient algorithm with  $d = 50$  and (from top left to bottom right)  $\rho = 2$ ,  $\rho = 4$ ,  $\rho = 10$ ,  $\rho = 20$ ,  $\rho = 50$ ,  $\rho = 100$ .

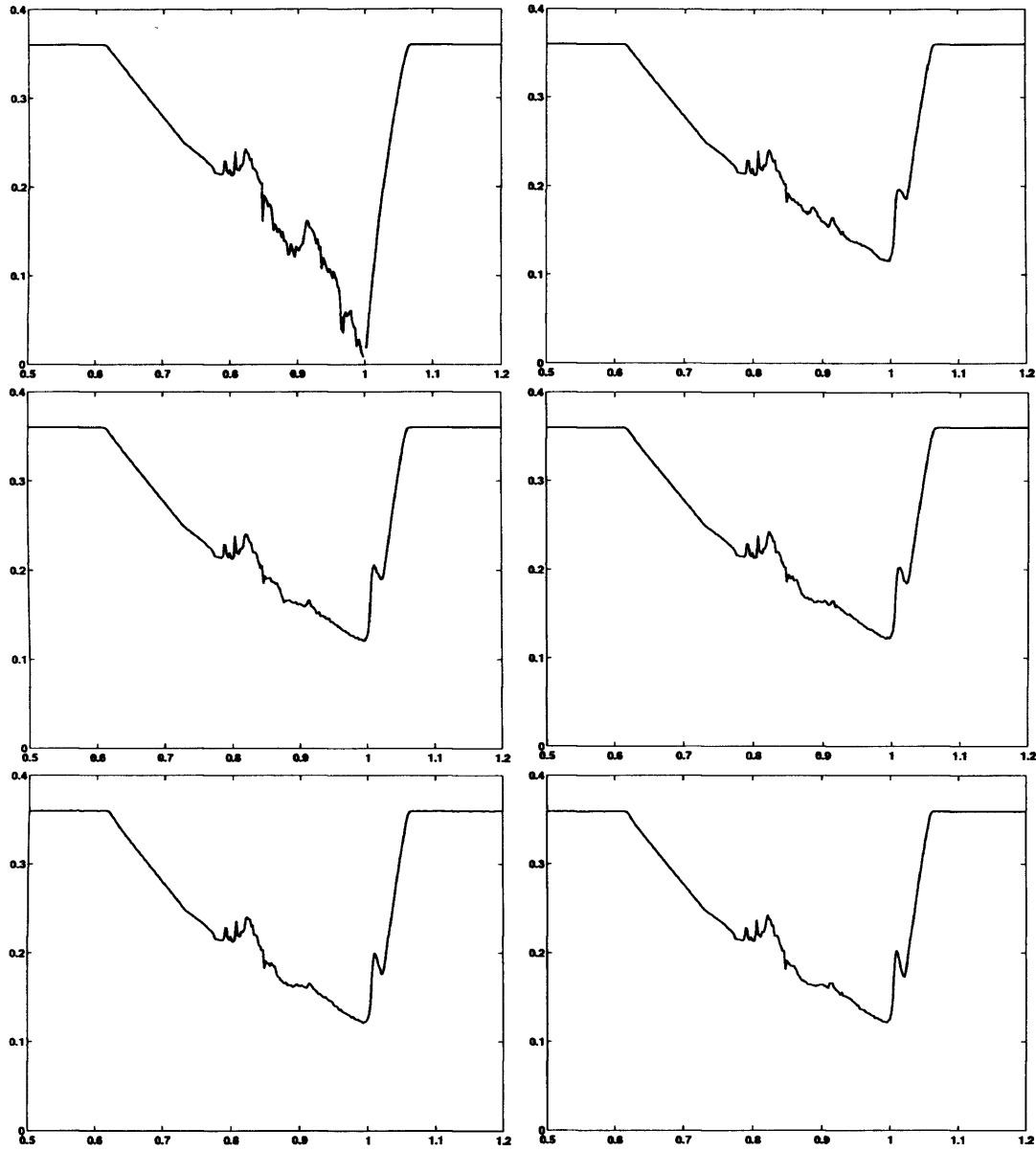


Figure 5.16: Average asymptotic rate of convergence as a function of  $\gamma$  for the  $(\gamma, A)$ -optimality gradient algorithm with  $\rho = 4$ , (from top left to bottom right)  $d = 2, d = 4, d = 10, d = 20, d = 50, d = 100$ .



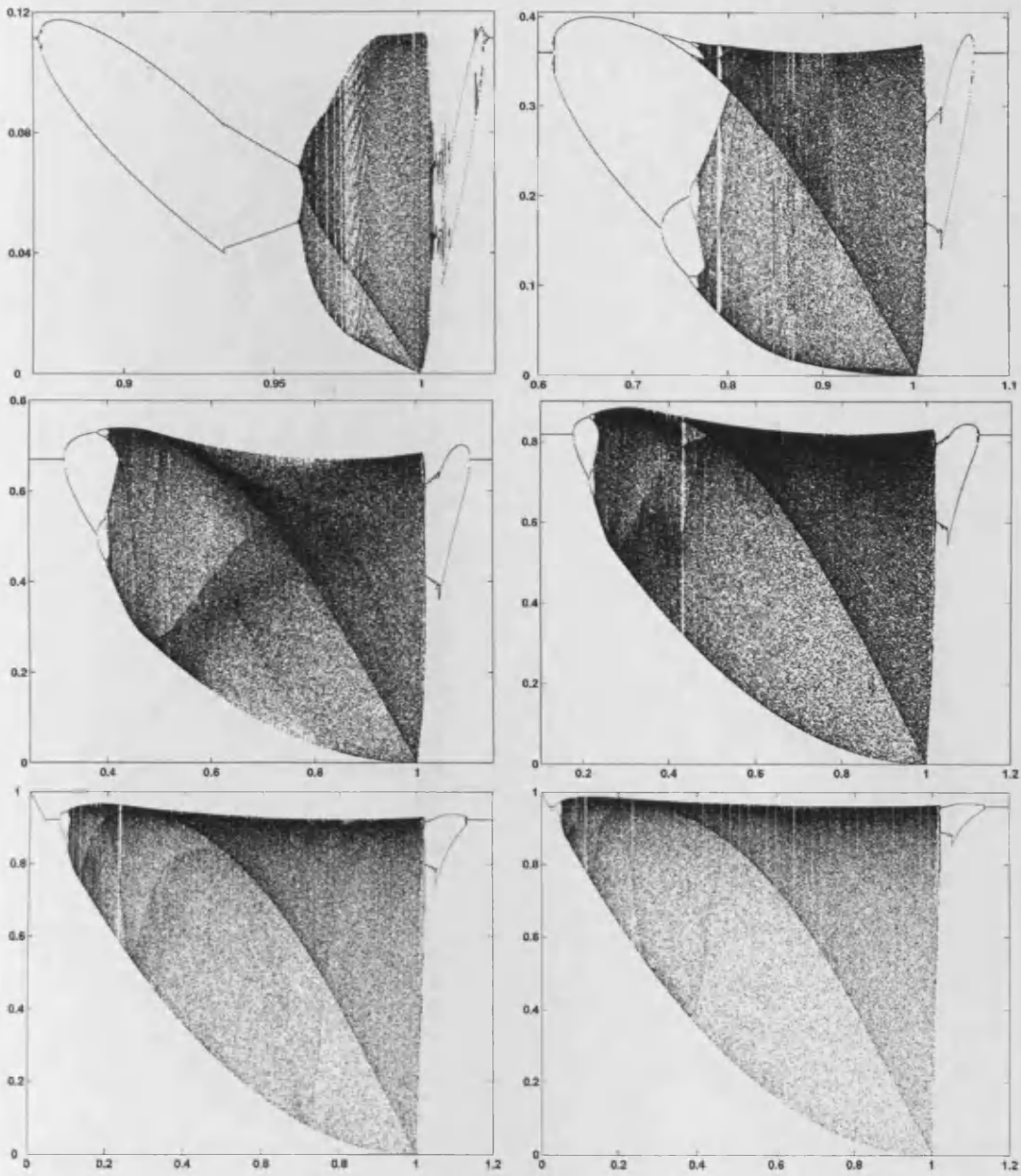


Figure 5.17: Attractors of  $r^{(k)}$  as a function of  $\gamma$  for the  $(\gamma, A)$ -optimality gradient algorithm with (from top left to bottom right)  $\rho = 2$ ,  $\rho = 4$ ,  $\rho = 10$ ,  $\rho = 20$ ,  $\rho = 50$ ,  $\rho = 100$ .

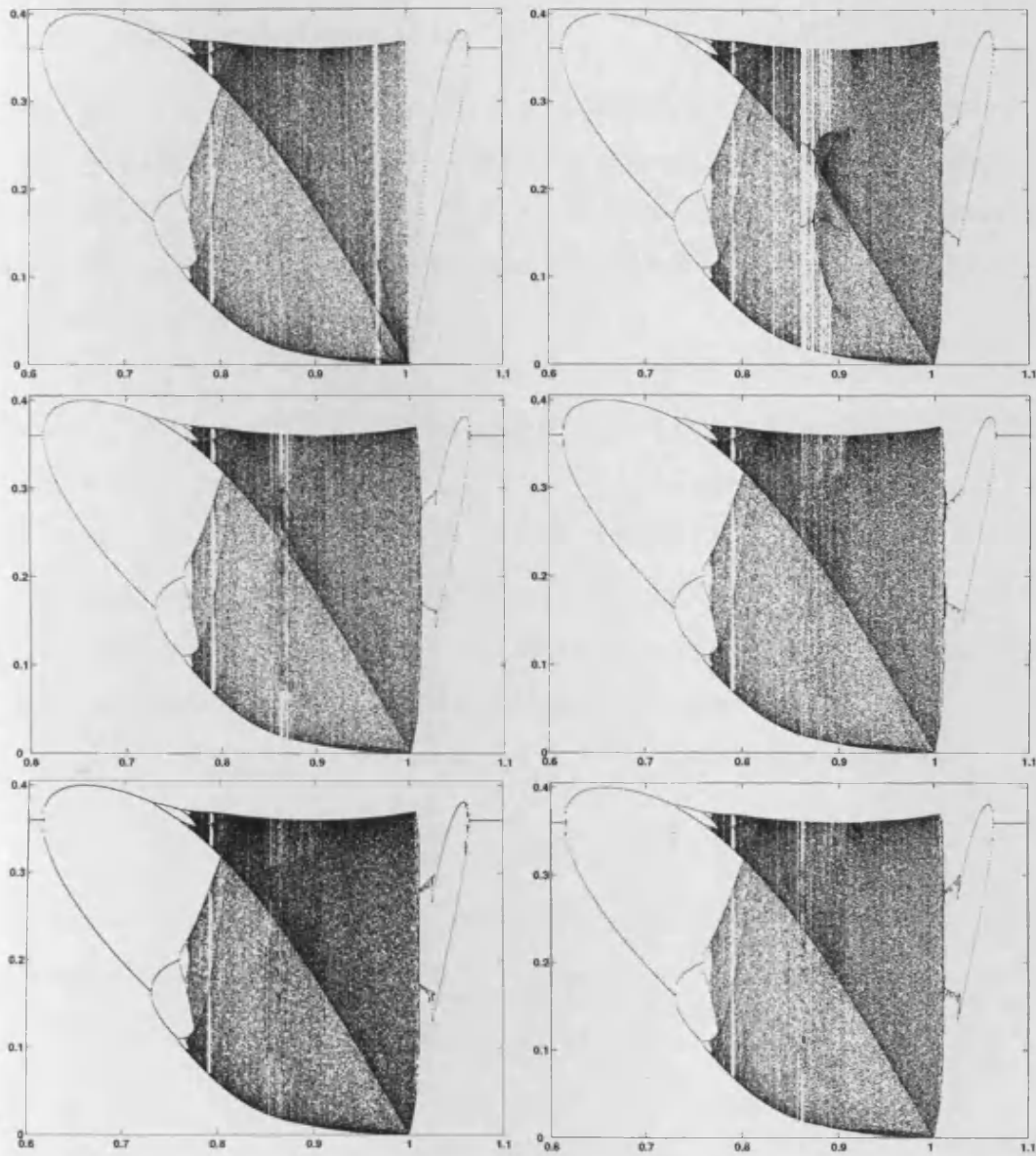


Figure 5.18: Attractors of  $r^{(k)}$  as a function of  $\gamma$  for the  $(\gamma, A)$ -optimality gradient algorithm with (from top left to bottom right)  $d = 2$ ,  $d = 4$ ,  $d = 10$ ,  $d = 20$ ,  $d = 50$ ,  $d = 100$ .

## 5.3 $\Phi_2$ -Optimality

### 5.3.1 The $\Phi_2$ -Optimality Criterion and the Corresponding Gradient Algorithm

It has been concluded that trying to modify the so-called  $A$ -optimality gradient algorithm by introducing a relaxation coefficient  $\gamma$ , does not give rise to any substantial improvement in asymptotic rate. There is, however, another way in which the  $A$ -optimality gradient algorithm can be generalised in order to create more prospective gradient algorithms.

The  $A$ -optimality criterion defined in (5.1) is one in a family of so-called  $\Phi_p$ -optimality criteria, see [26]. Here the general formula for the  $\Phi_p$ -optimality criterion is defined as

$$\Phi_p(M(\xi)) = (\text{tr}M^{-p}(\xi))^{\frac{1}{p}}, \quad (5.6)$$

where  $0 \leq p \leq \infty$ . When  $p = 1$  this criterion is that of  $A$ -optimality. As the  $A$ -optimality criterion has provided a very useful gradient optimisation algorithm, a logical progression would therefore be to consider the case where  $p = 2$ .

Substituting  $p = 2$  into (5.6) yields the so-called  $\Phi_2$ -optimality criterion

$$\Phi_2(M(\xi)) = (\text{tr}M^{-2}(\xi))^{\frac{1}{2}}.$$

In keeping with the  $D$ -optimality criterion of Chapter 2,  $\Phi(M(\xi)) = 1/\text{tr}M^{-2}(\xi)$  is considered as the optimality criterion to be maximised by the optimum design.

For the base function  $f(x) = (1, x)^T$

$$M^{-2}(\xi) = \frac{1}{(\mu_2 - \mu_1^2)^2} \begin{pmatrix} \mu_2^2 - \mu_1^2 & -\mu_1(\mu_2 + 1) \\ -\mu_1(\mu_2 + 1) & \mu_1^2 + 1 \end{pmatrix}$$

and therefore

$$\text{tr}(M^{-2}(\xi)) = \frac{\mu_2^2 + 2\mu_1^2 + 1}{(\mu_2 - \mu_1^2)^2}.$$

It follows that the criterion for  $\Phi_2$ -optimality can be written as

$$\Phi_2(M(\xi)) = \frac{\sqrt{(\mu_2^2 + 2\mu_1^2 + 1)}}{(\mu_2 - \mu_1^2)}$$

and thus

$$\Phi(\xi) = 1/(\Phi_2(M(\xi)))^2 = \frac{(\mu_2 - \mu_1^2)^2}{\mu_2^2 + 2\mu_1^2 + 1} \quad (5.7)$$

becomes the considered optimality criterion to be maximised.

In a similar way to previous algorithms generated, the updating formula for the  $\Phi_2$ -optimality gradient algorithm can be obtained from the general re-weighting formula

$$\xi'(x) = \frac{\varphi(x, \xi) - c(\xi)}{\text{tr}[M(\xi) \overset{\circ}{\Phi}(\xi)] - c(\xi)} \xi(x),$$

where in this case

$$\varphi(x, \xi) = \frac{2(\mu_2 - \mu_1^2)(\mu_2^3 + 2\mu_1^2\mu_2 + \mu_1^2 - 2x\mu_1\mu_2^2 - 2x\mu_1^3 - 2x\mu_1\mu_2 - 2x\mu_1 + x^2\mu_1^2\mu_2 + 2x^2\mu_1^2 + x^2)}{(\mu_2^2 + 1 + 2\mu_1^2)^2},$$

$$c(\xi) = \frac{2(\mu_2 - \mu_1^2)^4}{(\mu_1^2\mu_2 + 2\mu_1^2 + 1)(\mu_2^2 + 1 + 2\mu_1^2)^2}$$

and

$$\text{tr}[M(\xi) \overset{\circ}{\Phi}(\xi)] = 2\Phi(\xi).$$

The resulting updating formula has the form

$$\xi_i^{(k+1)} = \xi_i^{(k)} \left( 1 - \lambda_i \left( \frac{1 + 2\mu_1^2 + \mu_1^2\mu_2}{\mu_1(1 + \mu_2 + \mu_1^2 + \mu_2^2)} \right) \right)^2 / r^{(k)}, \quad \text{for } i = 1, \dots, d, \quad (5.8)$$

where

$$r^{(k)} = \frac{(\mu_2 - \mu_1^2)(\mu_1^2\mu_2^3 + 2\mu_1^2\mu_2^2 + 3\mu_2\mu_1^2 + 2\mu_1^4\mu_2 + 4\mu_1^2 + 1 + 3\mu_1^4)}{\mu_1^2(\mu_2 + \mu_1^2 + 1 + \mu_2^2)^2}.$$

Algorithm (5.8) shall be referred to as the  $\Phi_2$ -optimality gradient algorithm (in renormalised form). The step length of the  $\Phi_2$ -optimality gradient algorithm has the form

$$\alpha_{\Phi_2}^{(k)} = \frac{\mu_2\mu_1^2 + 2\mu_1^2 + 1}{\mu_1(\mu_2 + \mu_1^2 + \mu_2^2 + 1)}$$

$$= \frac{(Ag^{(k)}, g^{(k)})^2 [(A^2g^{(k)}, g^{(k)}) + 2(g^{(k)}, g^{(k)})] + (g^{(k)}, g^{(k)})^3}{(Ag^{(k)}, g^{(k)})[(g^{(k)}, g^{(k)})(A^2g^{(k)}, g^{(k)}) + (Ag^{(k)}, g^{(k)})^2 + (A^2g^{(k)}, g^{(k)})^2 + (g^{(k)}, g^{(k)})^2]}.$$

### 5.3.2 Behaviour of the Sequence $\{\Phi(\xi^{(k)})\}$

In a similar manner to the  $A$ -optimality criterion, the sequence  $\{\Phi(\xi^{(k)})\}$  for the  $\Phi_2$ -optimality gradient algorithm also does not converge and instead exhibits chaos. This chaotic behaviour can be seen in Figure 5.19 and Figure 5.20 where the sequence  $\{\Phi(\xi^{(k)})\}$  is shown for various values of  $\rho$ .

Figure 5.21 gives the distribution of  $\Phi(\xi^{(k)})$  for different condition numbers  $\rho$  and shows that, as with the  $A$ -optimality gradient algorithm,  $\Phi(\xi^{(k)})$  takes on extreme values in the range more often than more central values. This confirms the oscillatory behaviour from higher to lower values and back again seen in Figure 5.19 and Figure 5.20. The distribution of  $\Phi(\xi^{(k)})$  is shown in Figure 5.21. Figure 5.22 shows the transition from  $\Phi(\xi^{(k)})$  to  $\Phi(\xi^{(k+1)})$  and Figure 5.23 shows the weight pairings  $(\xi^{(k)}(\lambda_1), \xi^{(k)}(\lambda_d))$  express a similar behaviour to those of the  $A$ -optimality gradient algorithm.

### 5.3.3 Asymptotic Rate of Convergence

Figure 5.24 compares the average asymptotic rate of convergence of the  $\Phi_2$ -optimality gradient algorithm with that of the  $A$ -optimality gradient algorithm. It can be seen that the  $\Phi_2$ -optimality gradient algorithm outperforms the  $A$ -optimality gradient algorithm, with respect to the asymptotic rate of convergence, by some considerable margin. In fact the  $\Phi_2$ -optimality gradient algorithm has an asymptotic rate of convergence close to that achieved by the optimum  $s$ -gradient algorithm with  $s = \infty$ , see (2.12).

The asymptotic rate of convergence of the  $\Phi_2$ -optimality gradient algorithm is shown to have a greater dependence on  $d$ , particularly for large  $\rho$ , in Figure 5.25. For small  $\rho$  there is little dependence on  $d$  but as  $\rho$  increases, the number of dimensions required before the asymptotic rate starts to plateau also increases and for  $\rho = 100$  the dimensionality of the problem has an effect on the rate up until approximately  $d = 20$ .

In order to understand how the improvement in asymptotic rate of convergence over the  $A$ -optimality algorithm has occurred it is useful to compare the behaviour

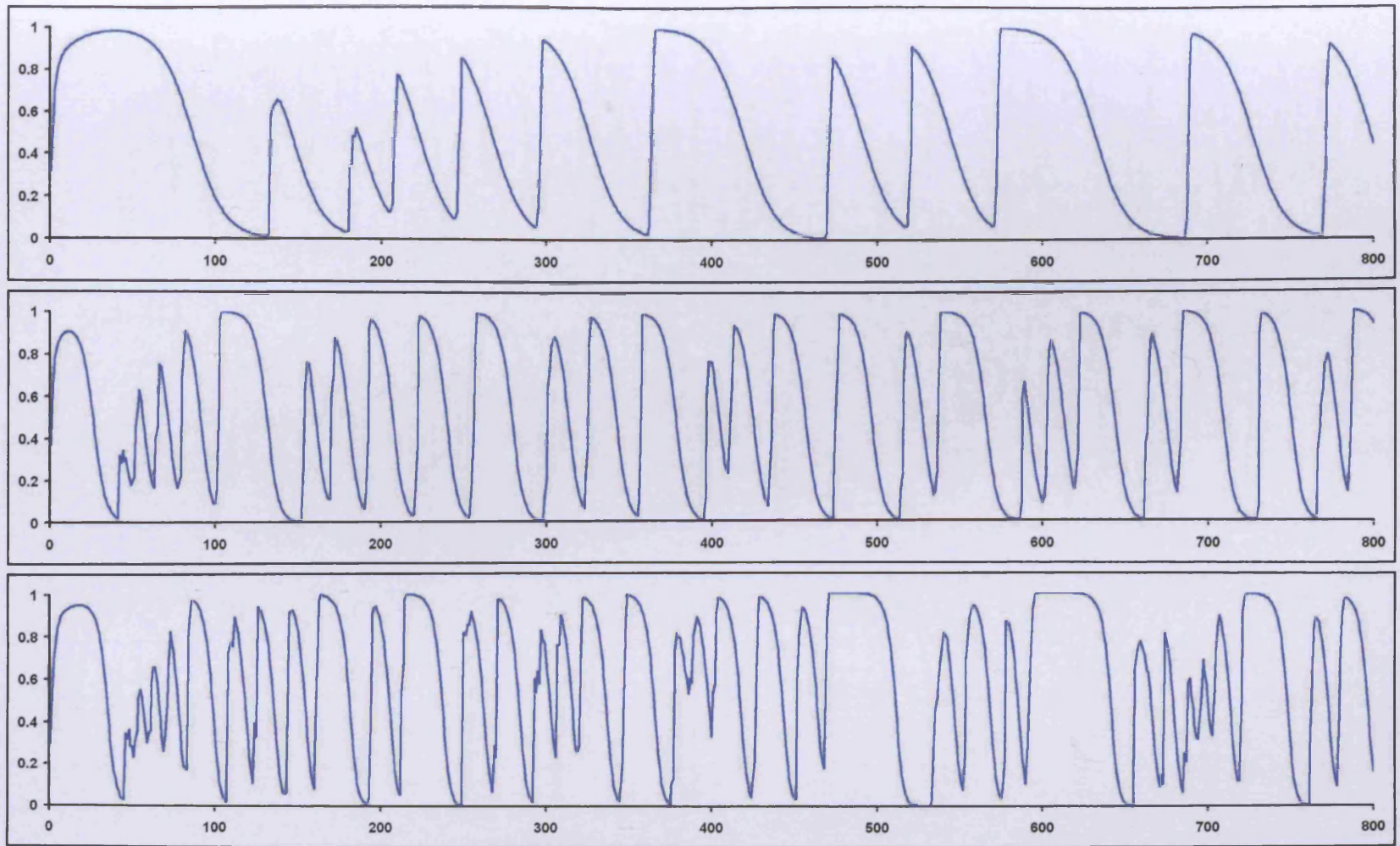


Figure 5.19: Trajectory of  $\Phi(\xi^{(k)})$ , plotted as a function of  $k$  for a single realisation of the  $\Phi_2$ -optimality gradient algorithm with, from top to bottom:  $\rho = 2$ ,  $\rho = 4$ ,  $\rho = 10$ ;  $d = 100$ .

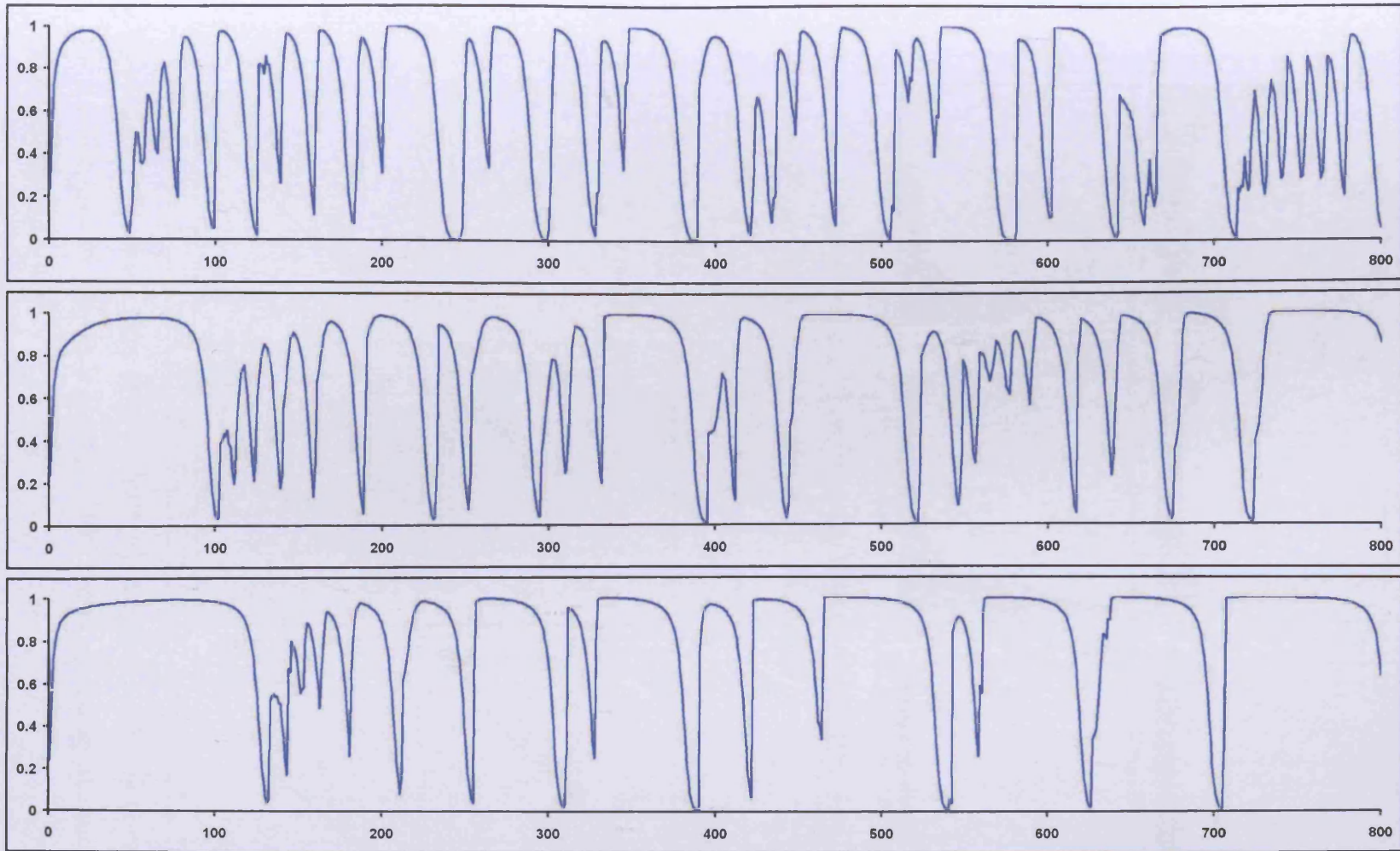


Figure 5.20: Trajectory of  $\Phi(\xi^{(k)})$ , plotted as a function of  $k$  for a single realisation of the  $\Phi_2$ -optimality gradient algorithm with, from top to bottom:  $\rho = 20$ ,  $\rho = 50$ ,  $\rho = 100$ ;  $d = 100$ .

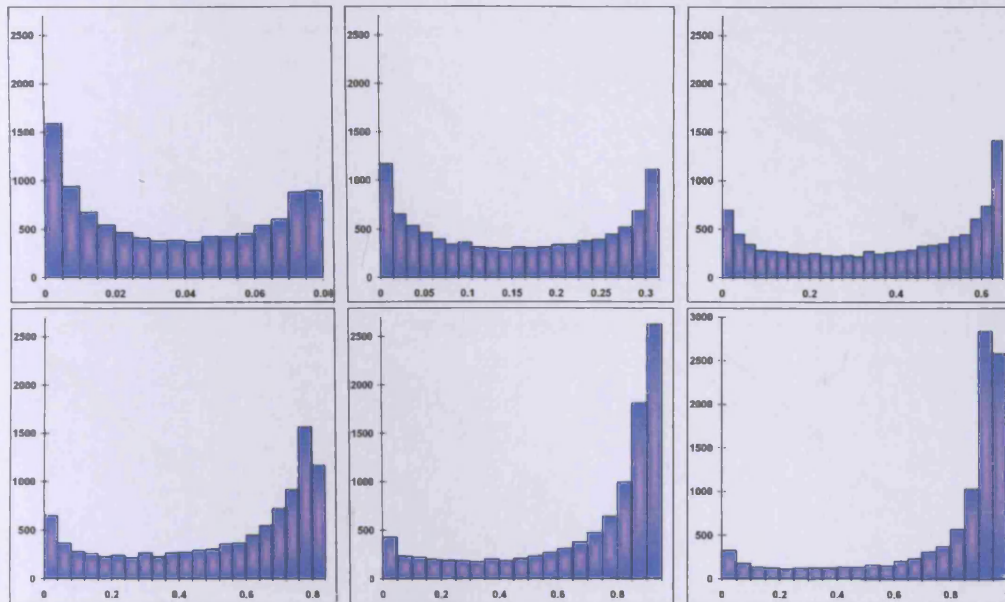


Figure 5.21: Histogram of  $\Phi(\xi^{(k)})$  for, from top left to bottom right:  $\rho = 2$ ,  $\rho = 4$ ,  $\rho = 10$ ,  $\rho = 20$ ;  $\rho = 50$ ,  $\rho = 100$ ;  $d = 100$ ,  $k = 1, \dots, 10000$ .

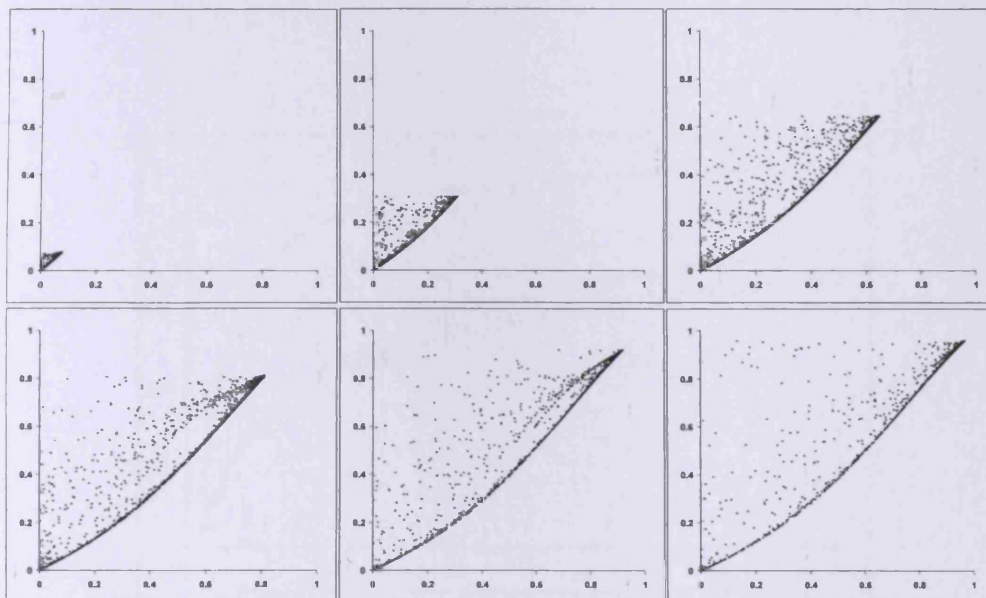


Figure 5.22: Plots of the pairs  $(\Phi(\xi^{(k)}), \Phi(\xi^{(k+1)}))$  for, from top left to bottom right  $\rho = 2$ ,  $\rho = 4$ ,  $\rho = 10$ ,  $\rho = 20$ ,  $\rho = 50$ ,  $\rho = 100$ . Points plotted are the last 2000 of 10000 iterations;  $d = 100$ .



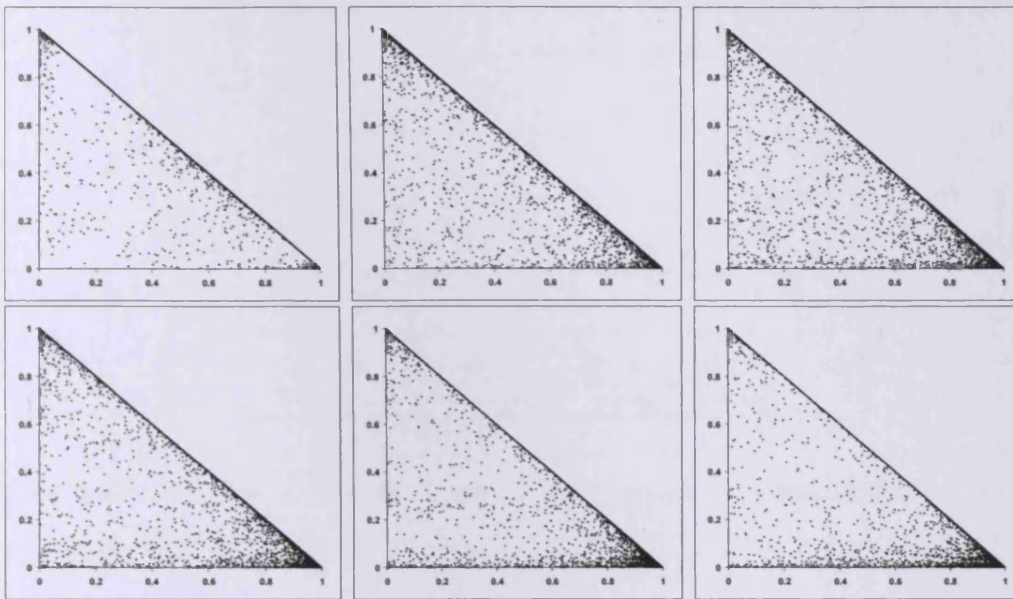


Figure 5.23: Plots of the pairs  $(\xi^{(k)}(\lambda_1), \xi^{(k)}(\lambda_d))$  for the  $\Phi_2$ -optimality gradient algorithm with (from top left to bottom right):  $\rho = 2, 4, 10, 20, 50, 100$ ;  $d = 100$ . Points plotted are the last 8000 of 10000 iterations.

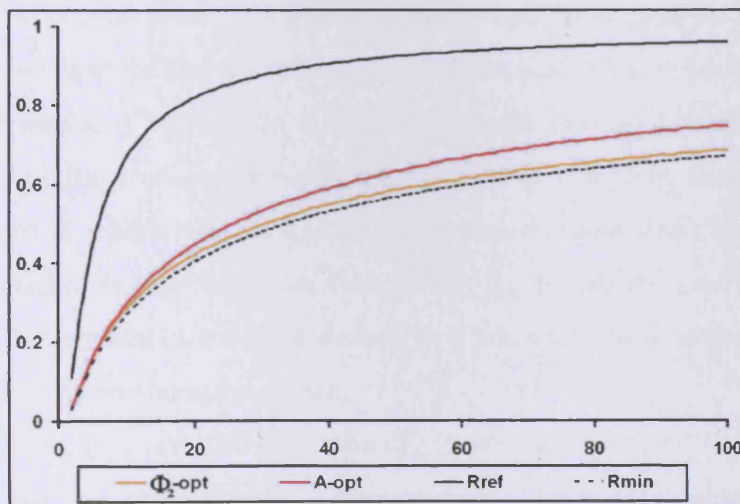


Figure 5.24: Average asymptotic rate of convergence as a function of  $\rho$  for the A-optimality and  $\Phi_2$ -optimality gradient algorithms;  $d = 100$ .

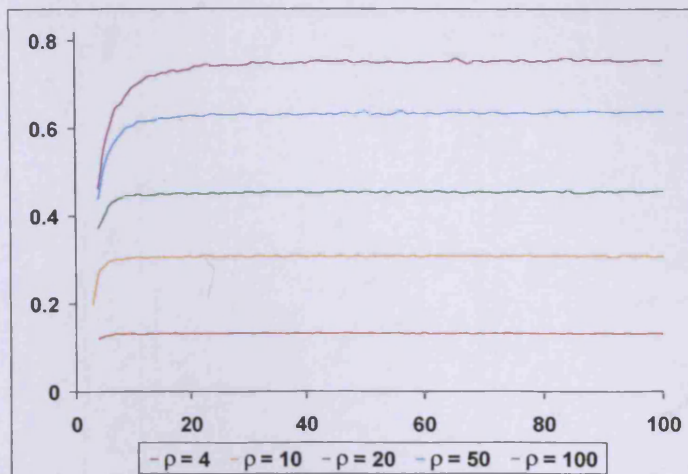


Figure 5.25: Average asymptotic rate of convergence as a function of  $d$  for the  $\Phi_2$ -optimality gradient algorithm.

of the rate  $r^{(k)}$  of each of the algorithms. Figure 5.26 and Figure 5.27 show the rate  $r^{(k)}$  for the  $\Phi_2$ -optimality gradient algorithm as a function of  $k$  for several values of  $\rho$ . Again, regardless of the value of  $\rho$ , the rate at each iteration of the  $\Phi_2$ -optimality gradient algorithm remains below 1 indicating that this algorithm is also monotonic in its approach to the minimum point  $x^*$ . As with the  $A$ -optimality gradient algorithm, the rate of the  $\Phi_2$ -optimality gradient algorithm fluctuates from extremely good points, close to 0 to comparatively bad rates, close to 1. The number of iterations required for the  $\Phi_2$ -optimality gradient algorithm to progress from bad rate to good rate and back again is however greater making this algorithm more systematic than its predecessor since less periods of complete chaos ensue. For larger values of  $\rho$ , where the most pronounced improvement over the  $A$ -optimality gradient algorithm is observed, the rate of the  $\Phi_2$ -optimality gradient algorithm still reaches the extremely good rates close to 0 whereas the  $A$ -optimality gradient algorithm does not to the same extent.

Figure 5.28 and Figure 5.29 show the distributions of  $r^{(k)}$  and  $(-\ln r^{(k)})$ . These graphs show the rate of the  $\Phi_2$ -optimality gradient algorithm to behave similarly to that of the  $A$ -optimality gradient algorithm where more values of  $r^{(k)}$  are located at either end of the range and less occur in the middle. The  $\Phi_2$ -optimality gradient algorithm's rate is more extreme in this shape of distribution with markedly more

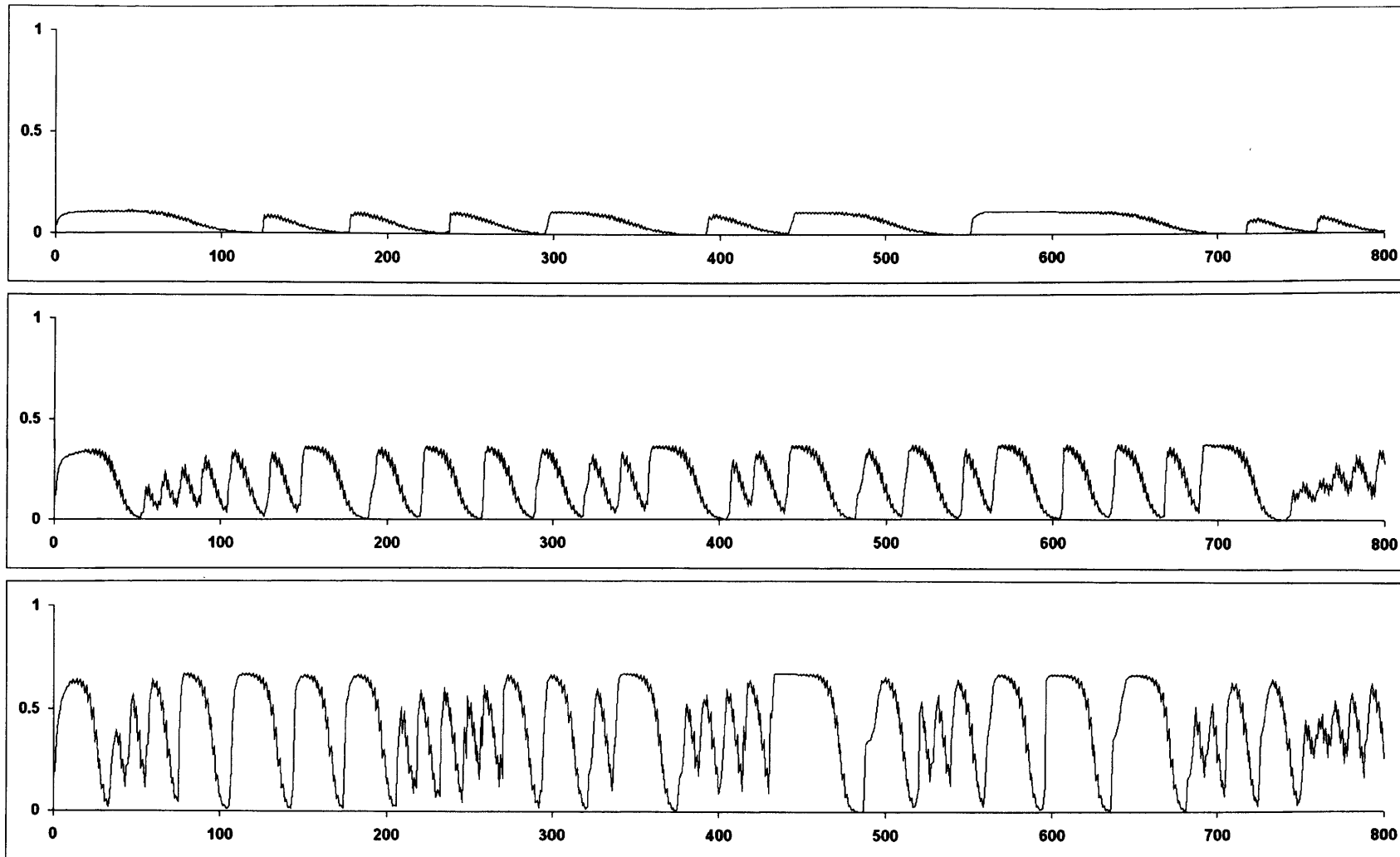


Figure 5.26: Rate,  $r^{(k)}$ , as a function of  $k$  for from top to bottom:  $\rho = 2$ ,  $\rho = 4$ ,  $\rho = 10$ ;  $d = 100$ .

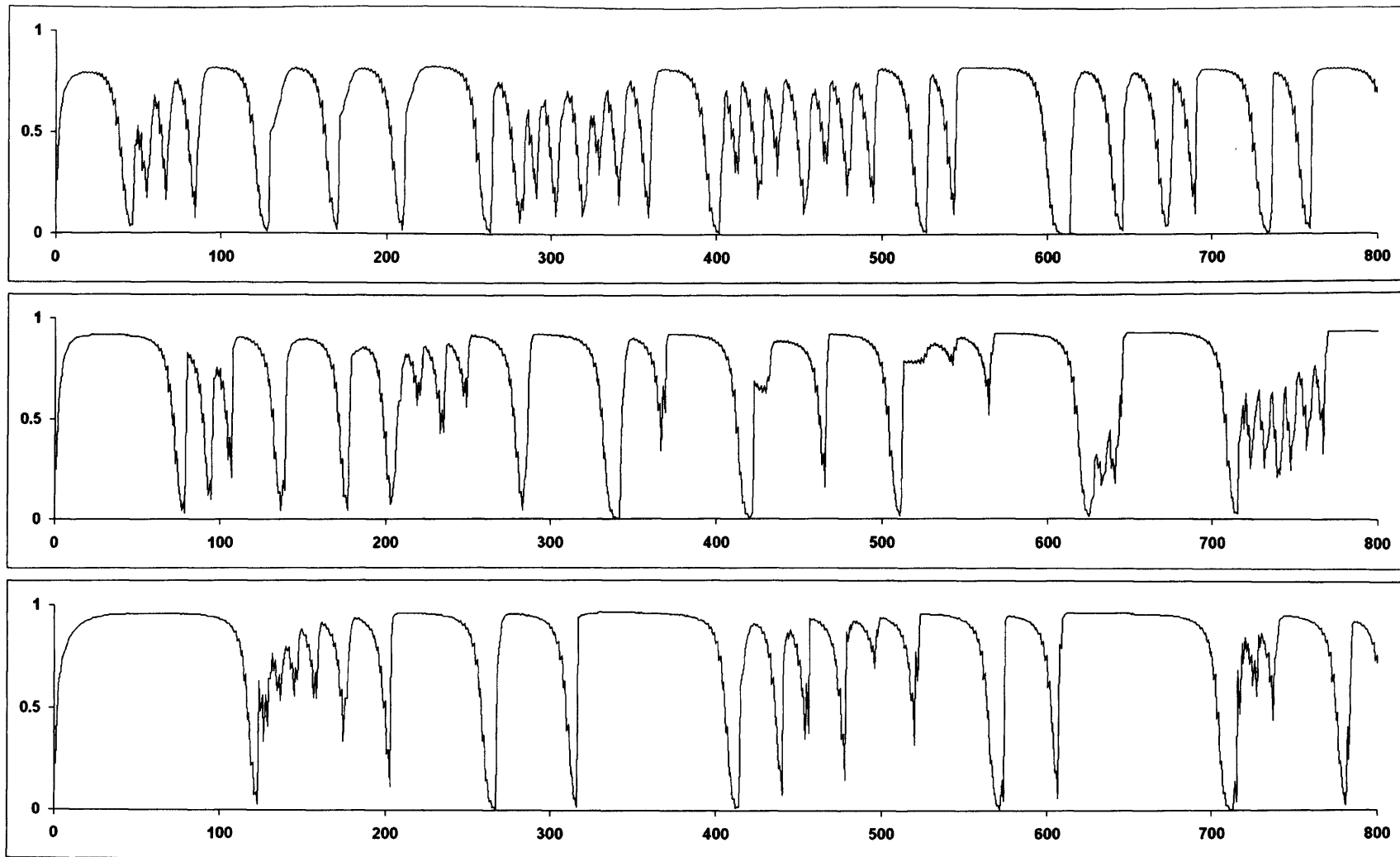


Figure 5.27: Rate,  $r^{(k)}$ , as a function of  $k$  for from top to bottom:  $\rho = 20$ ,  $\rho = 50$ ,  $\rho = 100$ ;  $d = 100$ .

rates falling in the lower end of the range for larger values of  $\rho$ .

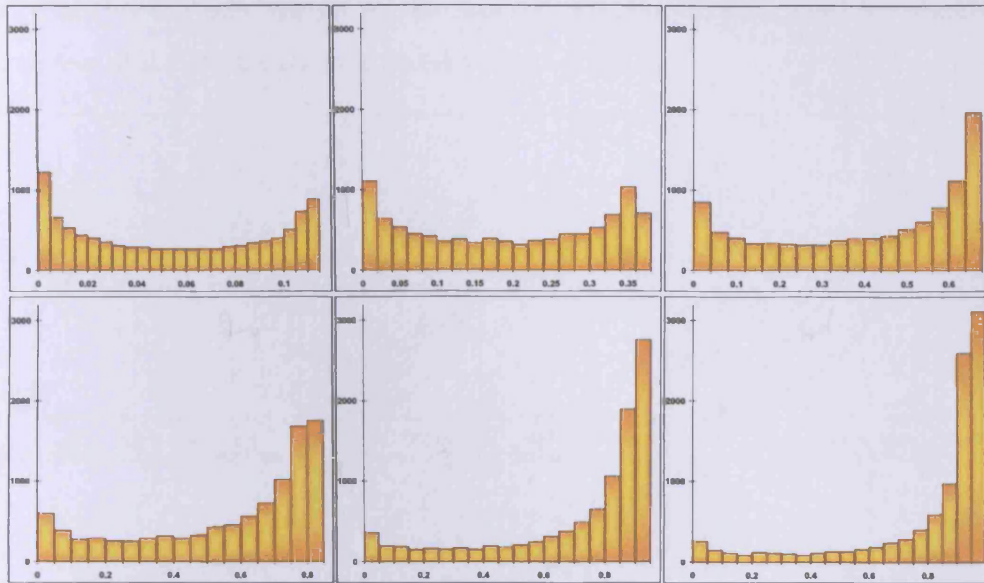


Figure 5.28: Histogram of  $r^{(k)}$  for, from top left to bottom right:  $\rho = 2$ ,  $\rho = 4$ ,  $\rho = 10$ ,  $\rho = 20$ ,  $\rho = 50$ ,  $\rho = 100$ ;  $d = 100$ ,  $k = 1, \dots, 10000$ .

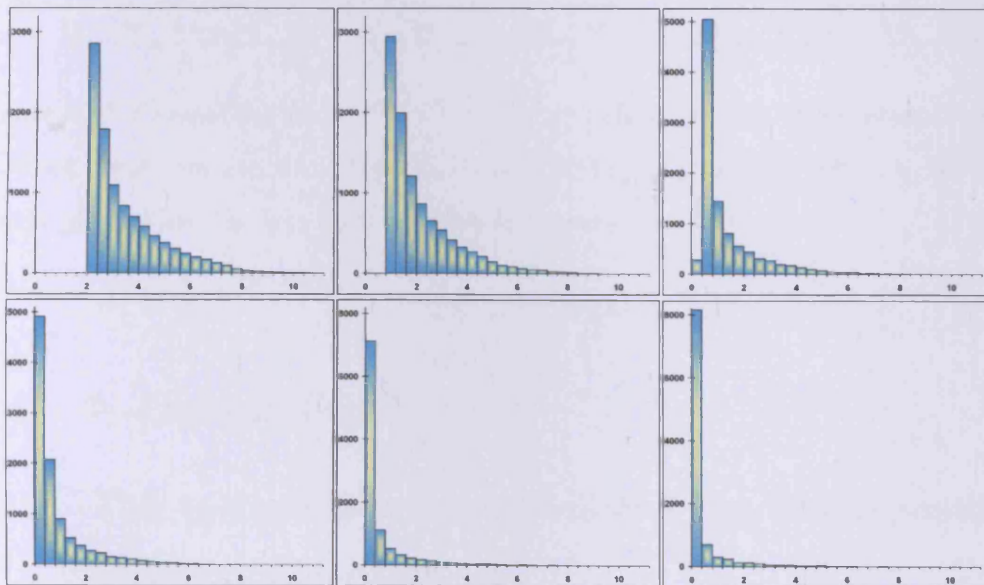


Figure 5.29: Histogram of  $(-\ln(r^{(k)}))$  for, from top left to bottom right:  $\rho = 2$ ,  $\rho = 4$ ,  $\rho = 10$ ,  $\rho = 20$ ,  $\rho = 50$ ,  $\rho = 100$ ;  $d = 100$ ,  $k = 1, \dots, 10000$ .

Figure 5.30 shows the transition from  $r^{(k)}$  to  $r^{(k+1)}$  for the  $\Phi_2$ -optimality gradient algorithm. While chaos is evidently still present in the transition from one rate to

the next there is also a definite pattern suggesting that the algorithm does not lapse into chaos to the same extent as the rates of the  $A$ -optimality gradient algorithm where less of a clear transitional pattern can be seen.

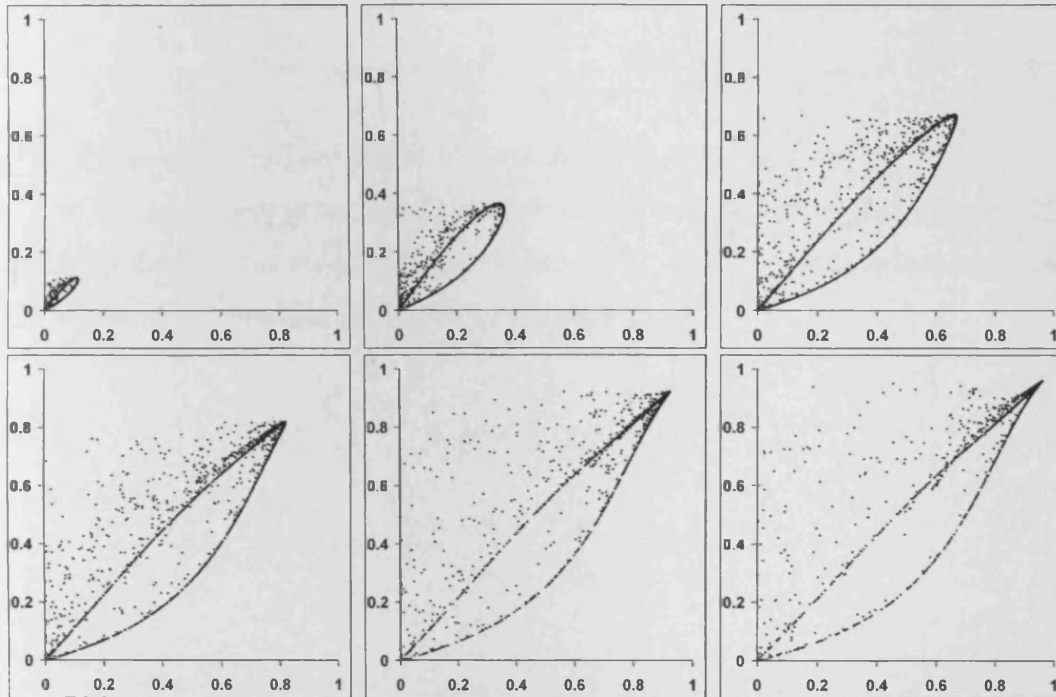


Figure 5.30: Plots of the pairs  $(r^{(k)}, r^{(k+1)})$  for a single trajectory of the  $\Phi_2$ -optimality gradient algorithm for, from top left to bottom right:  $\rho = 2, 4, 10, 20, 50, 100$ . Points plotted are the last 5000 of 10000 iterations;  $d = 100$ .

## 5.4 $\Phi_3$ -Optimality

### 5.4.1 The $\Phi_3$ -Optimality Criterion and the Corresponding Gradient Algorithm

Since the  $\Phi_2$ -optimality gradient algorithm gave rise to a faster asymptotic rate of convergence than the  $A$ -optimality algorithm it is sensible to progress further and consider the possibility of constructing an algorithm from the  $\Phi_3$ -optimality criterion. Substituting  $p = 3$  into the general formula (5.6) for the  $\Phi_p$ -optimality

criterion gives

$$\Phi_3(M(\xi)) = \sqrt[3]{\frac{3\mu_2\mu_1^2 + \mu_2^3 + 3\mu_1^2 + 1}{(\mu_2 - \mu_1^2)^3}}.$$

For the sake of simplicity, the function

$$\Phi(\xi) = 1/(\Phi_3(M(\xi)))^3 = \frac{(\mu_2 - \mu_1^2)^3}{3\mu_2\mu_1^2 + \mu_2^3 + 3\mu_1^2 + 1} \quad (5.9)$$

will be taken to form the basis of the new gradient algorithm.

In the same way as with the previous algorithms formed in this chapter, the updating formula for the algorithm corresponding the the  $\Phi_3$ -optimality criterion can be obtained from the general re-weighting formula

$$\xi'(x) = \frac{\varphi(x, \xi) - c(\xi)}{\text{tr}[M(\xi)\overset{\circ}{\Phi}(\xi)] - c(\xi)}\xi(x).$$

In this case

$$\varphi(x, \xi) = \frac{\partial\Phi}{\partial\mu_0} + x \frac{\partial\Phi}{\partial\mu_1} + x^2 \frac{\partial\Phi}{\partial\mu_2},$$

where

$$\frac{\partial\Phi}{\partial\mu_0} = 3 \frac{(\mu_0\mu_2 - \mu_1^2)^2 (2\mu_0\mu_2\mu_1^2 + \mu_1^4 + \mu_1^2\mu_0^2 + \mu_2^4 + 3\mu_2^2\mu_1^2)}{(\mu_2^3 + 3\mu_2\mu_1^2 + 3\mu_1^2\mu_0 + \mu_0^3)^2},$$

$$\frac{\partial\Phi}{\partial\mu_1} = -6 \frac{\mu_1 (\mu_0\mu_2 - \mu_1^2)^2 (\mu_0^2 + 2\mu_1^2 + \mu_2^2)}{(\mu_0^2 - \mu_0\mu_2 + 3\mu_1^2 + \mu_2^2) (\mu_2^3 + 3\mu_2\mu_1^2 + 3\mu_1^2\mu_0 + \mu_0^3)^2},$$

and

$$\frac{\partial\Phi}{\partial\mu_2} = 3 \frac{(\mu_0\mu_2 - \mu_1^2)^2 (2\mu_0\mu_2\mu_1^2 + \mu_2^2\mu_1^2 + \mu_1^4 + 3\mu_1^2\mu_0^2 + \mu_0^4)}{(\mu_2^3 + 3\mu_2\mu_1^2 + 3\mu_1^2\mu_0 + \mu_0^3)^2};$$

$$c = \frac{3(\mu_2 - \mu_1^2)^6}{(\mu_2 + 1)^2 (2\mu_2\mu_1^2 + \mu_2^2\mu_1^2 + \mu_1^4 + 3\mu_1^2 + 1) (1 - \mu_2 + 3\mu_1^2 + \mu_2^2)^2}$$

and

$$\text{tr}[M(\xi)\overset{\circ}{\Phi}(\xi)] = 3\Phi(\xi).$$

The resulting updating formula has the form

$$\xi_i^{(k+1)} = \frac{(1 - \lambda_i \alpha_{\Phi_3}^{(k)})^2}{r^{(k)}} \xi_i^{(k)}, \quad \text{for } i = 1, \dots, d, \quad (5.10)$$

where

$$\alpha_{\Phi_3}^{(k)} = \frac{2\mu_2\mu_1^2 + \mu_2^2\mu_1^2 + \mu_1^4 + 3\mu_1^2 + 1}{\mu_1(\mu_2 + 1)(1 + 2\mu_1^2 + \mu_2^2)}$$

and

$$r^{(k)} = 1 - 2 \frac{2\mu_2\mu_1^2 + \mu_2^2\mu_1^2 + \mu_1^4 + 3\mu_1^2 + 1}{(\mu_2 + 1)(1 + 2\mu_1^2 + \mu_2^2)} + \frac{\mu_2(2\mu_2\mu_1^2 + \mu_2^2\mu_1^2 + \mu_1^4 + 3\mu_1^2 + 1)^2}{\mu_1^2(\mu_2 + 1)^2(1 + 2\mu_1^2 + \mu_2^2)^2}.$$

In line with the rest in this family of algorithms, the algorithm (5.10) shall be referred to as the  $\Phi_3$ -optimality gradient algorithm.

### 5.4.2 Behaviour of the Sequence $\{\Phi(\xi^{(k)})\}$

Figure 5.31 and Figure 5.32 show the sequence  $\{\Phi(\xi^{(k)})\}$ , for the  $\Phi_3$ -optimality criterion for various  $\rho$ . Figure 5.33 shows the distribution of  $\Phi(\xi^{(k)})$  in the form of histograms also for various  $\rho$  and Figure 5.34 shows the transition from  $\Phi(\xi^{(k)})$  to  $\Phi(\xi^{(k+1)})$ . By comparing these figures with the corresponding graphs for the  $\Phi_2$ -optimality criterion in the previous section, it can be seen that all of these figures show the behaviour of the  $\Phi_3$ -optimality criterion (5.9) to be virtually identical to that of the  $\Phi_2$ -optimality criterion (5.7). Likewise, Figure 5.35 shows the weight pairings  $(\xi^{(k)}(\lambda_1), \xi^{(k)}(\lambda_d))$  to behave in exactly the same fashion as those of the  $\Phi_2$ -optimality gradient algorithm.

### 5.4.3 Asymptotic Rate of Convergence

The  $\Phi_3$ -optimality criterion was shown to behave in an almost identical fashion to the  $\Phi_2$ -optimality criterion. It is of no surprise therefore that the asymptotic rate of convergence of the  $\Phi_3$ -optimality gradient algorithm is also approximately the same as that of the  $\Phi_2$ -optimality gradient algorithm, see Figure 5.36. The dimensionality of the problem has the same pronounced effect on the asymptotic rate in lower dimensions, see Figure 5.37 with the rate not becoming constant with increasing  $d$  until  $d \gtrsim 20$  for larger values of  $\rho$ . Since both algorithms share the same desirable asymptotic rate of convergence it is advisable to select the  $\Phi_2$ -optimality gradient algorithm as a preference between the two as the step length formula for this algorithm is simpler. Figure 5.38 and Figure 5.39 show the rate  $r^{(k)}$  as a function of  $k$  for different values of  $\rho$ . Figure 5.40 and Figure 5.41 show the distributions of  $r^{(k)}$  and  $(-\ln r^{(k)})$  respectively and Figure 5.42 shows the transition from  $r^{(k)}$  to  $r^{(k+1)}$ .



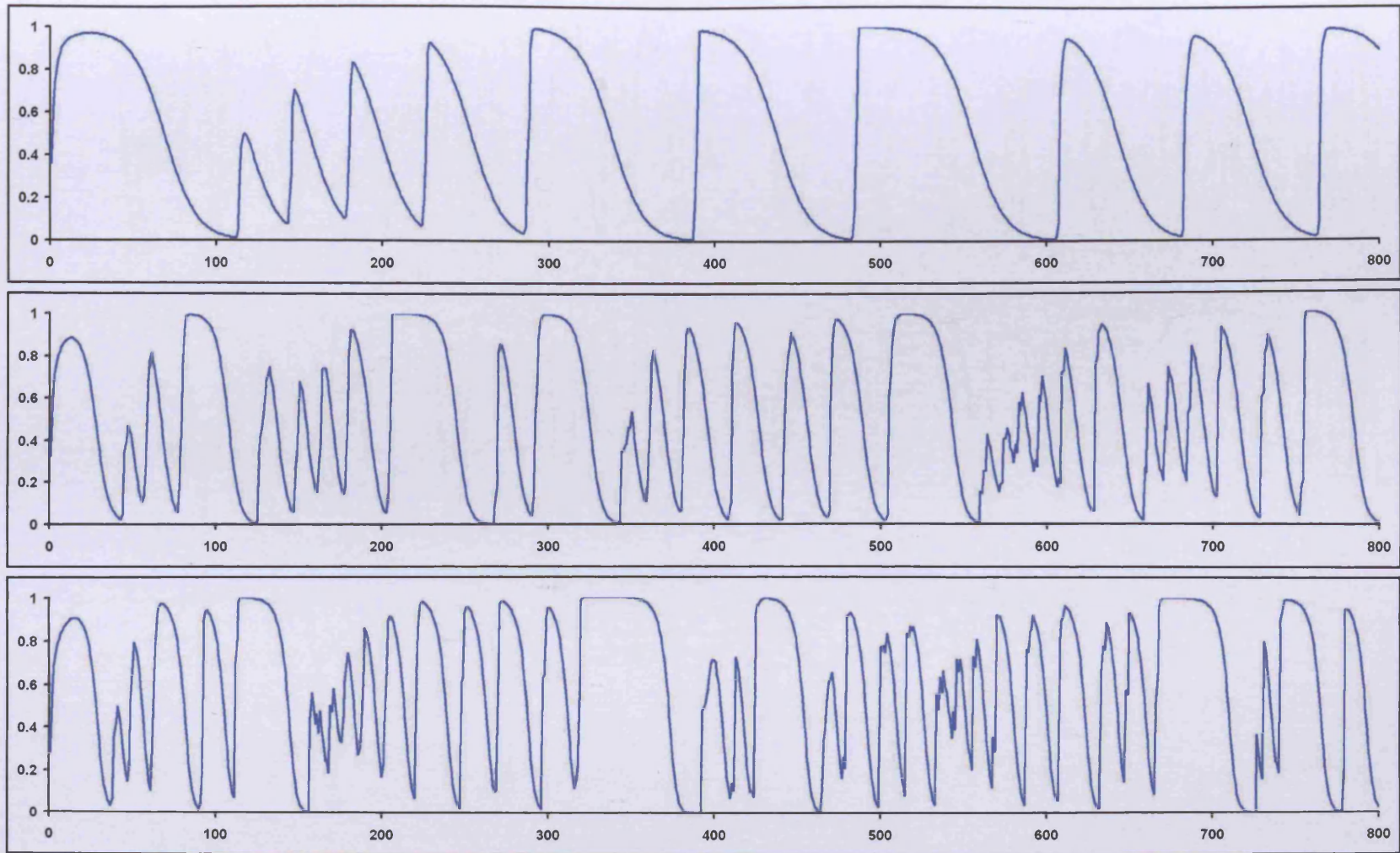


Figure 5.31: Trajectory of  $\Phi(\xi^{(k)})$ , as a function of  $k$  for from top to bottom:  $\rho = 2$ ,  $\rho = 4$ ,  $\rho = 10$ ;  $d = 100$ .

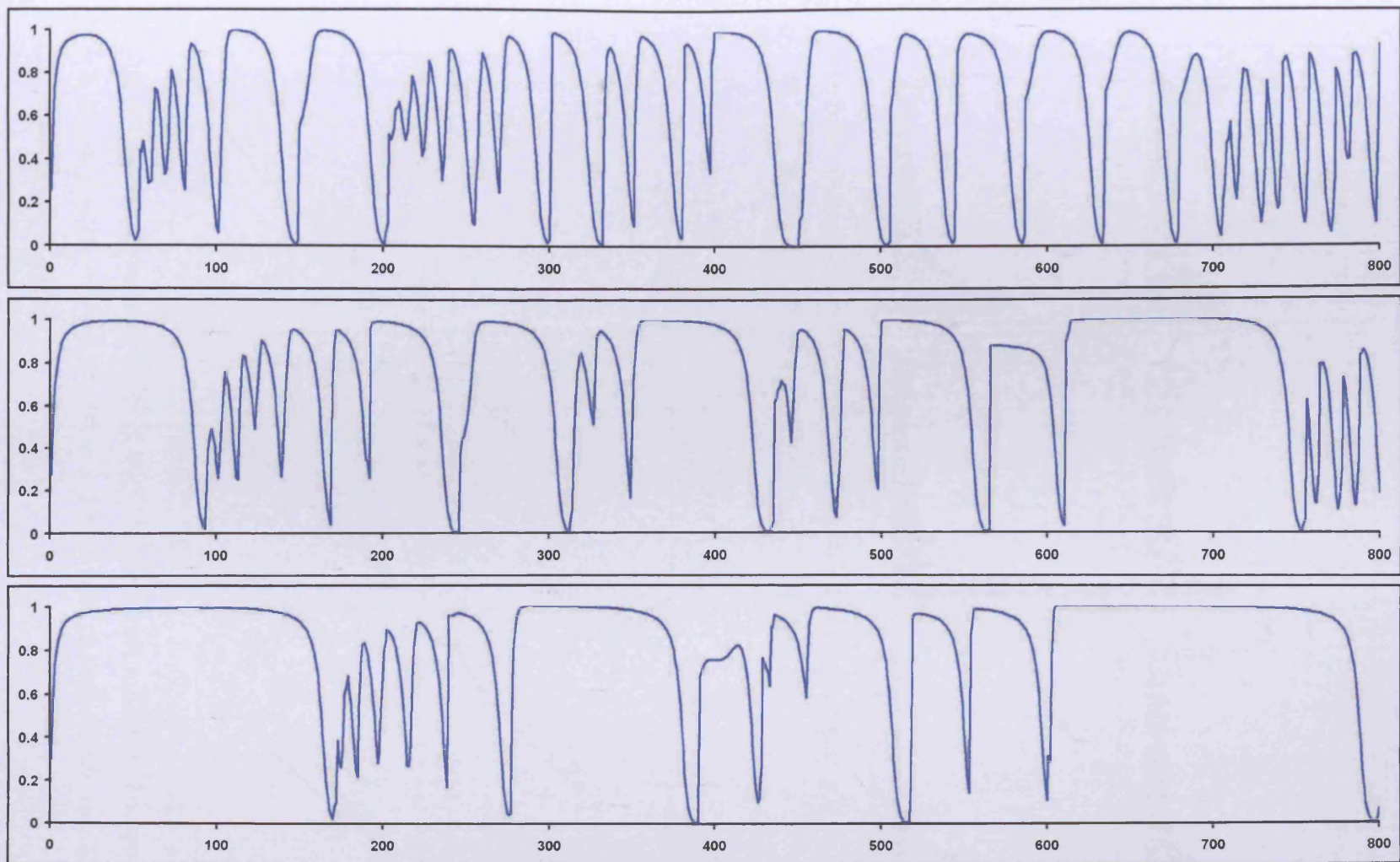


Figure 5.32: Trajectory of  $\Phi(\xi^{(k)})$ , as a function of  $k$  for from top to bottom:  $\rho = 20$ ,  $\rho = 50$ ,  $\rho = 100$ ;  $d = 100$ .

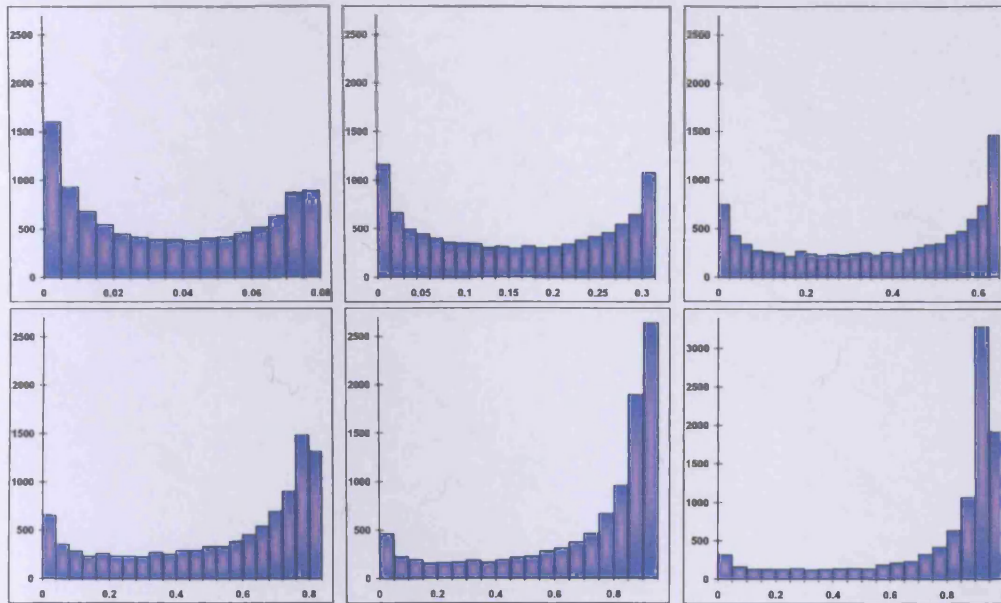


Figure 5.33: Histogram of  $\Phi(\xi^{(k)})$  for, from top left to bottom right:  $\rho = 2$ ,  $\rho = 4$ ,  $\rho = 10$ ,  $\rho = 20$ ,  $\rho = 50$ ,  $\rho = 100$ ;  $d = 100$ ,  $k = 1, \dots, 10000$ .

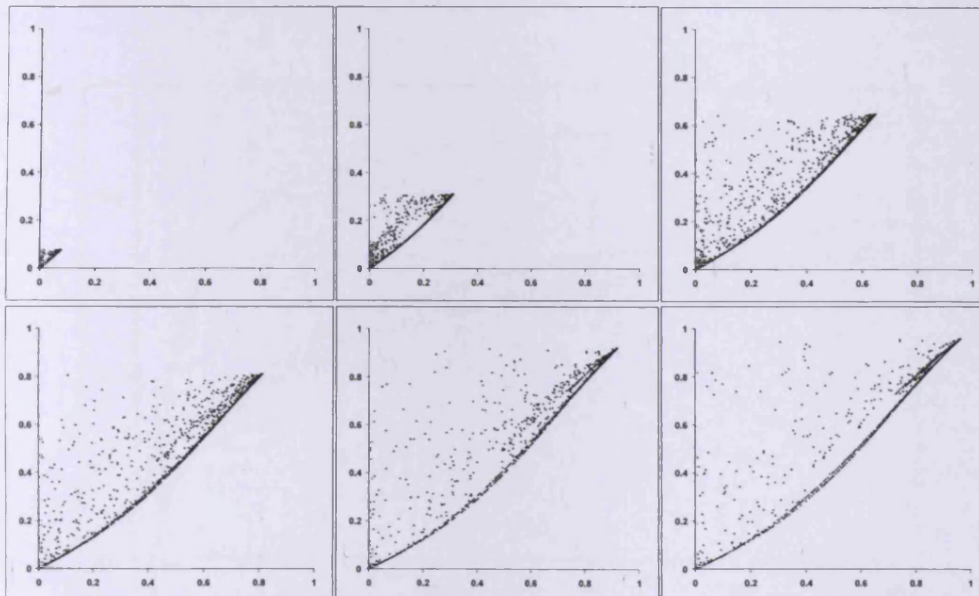


Figure 5.34: Plots of the pairs  $(\Phi(\xi^{(k)}), \Phi(\xi^{(k+1)}))$  for, from top left to bottom right  $\rho = 2$ ,  $\rho = 4$ ,  $\rho = 10$ ,  $\rho = 20$ ,  $\rho = 50$ ,  $\rho = 100$ . Points plotted are the last 2000 of 10000 iterations;  $d = 100$ .

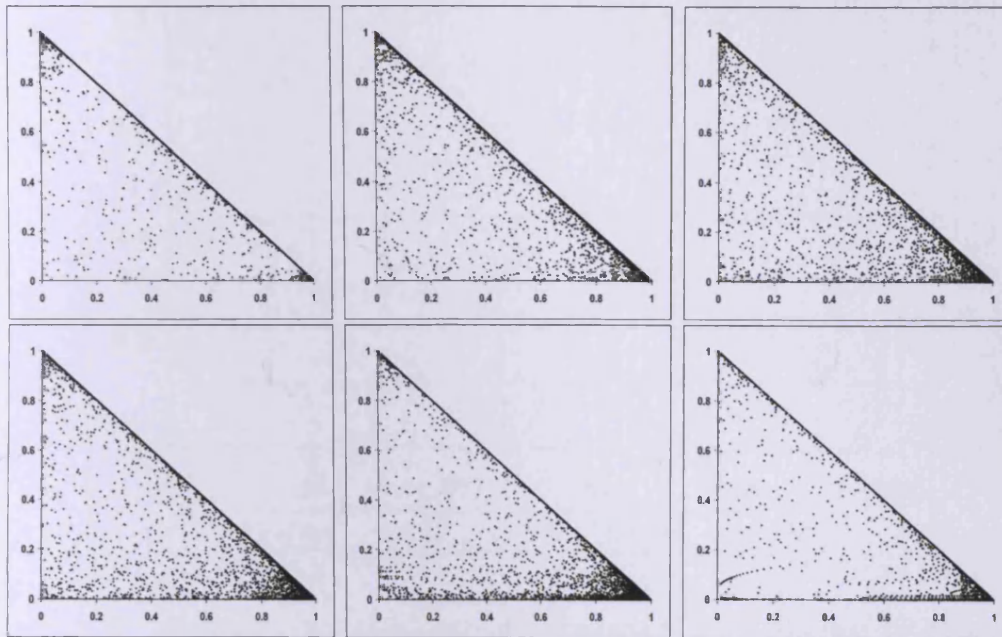


Figure 5.35: Plots of the pairs  $(\xi^{(k)}(\lambda_1), \xi^{(k)}(\lambda_d))$  for the  $\Phi_3$ -optimality gradient algorithm with (from top left to bottom right):  $\rho = 2, 4, 10, 20, 50, 100$ ;  $d = 100$ . Points plotted are the last 8000 of 10000 iterations.

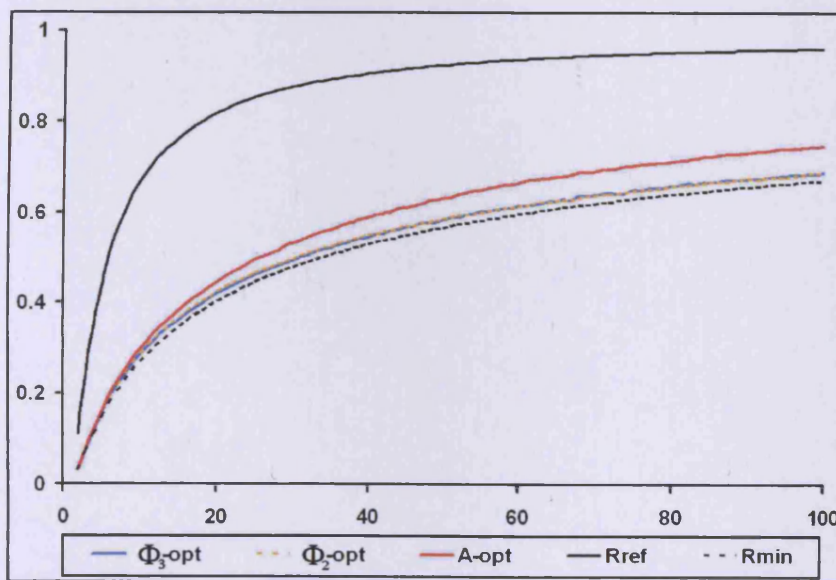


Figure 5.36: Average asymptotic rate of convergence as a function of  $\rho$  for the  $A$ -optimality,  $\Phi_2$ -optimality and  $\Phi_3$ -optimality algorithms.

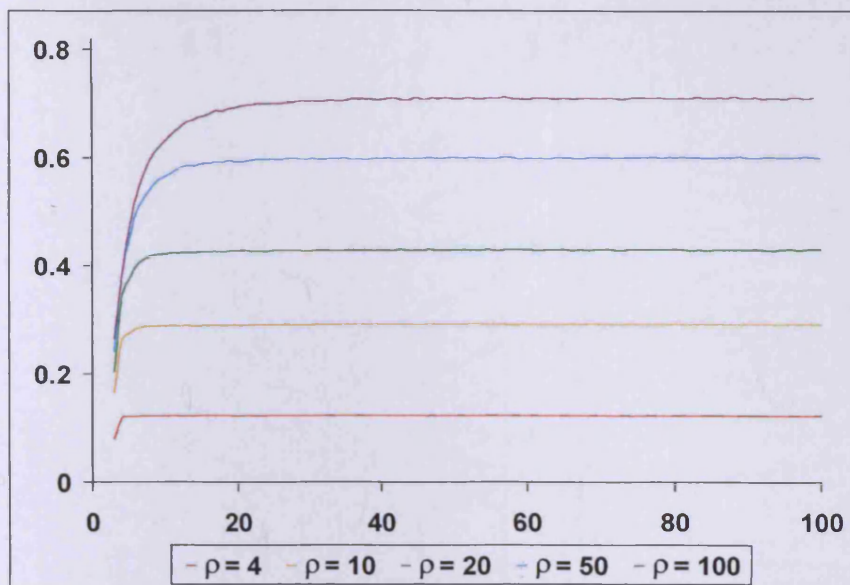


Figure 5.37: Average asymptotic rate of convergence as a function of  $d$  for the  $\Phi_3$ -optimality gradient algorithm.

All these figures reveal the behaviour of the rate  $r^{(k)}$  for the  $\Phi_3$ -optimality gradient algorithm to be almost identical to that of the  $\Phi_2$ -optimality gradient algorithm.

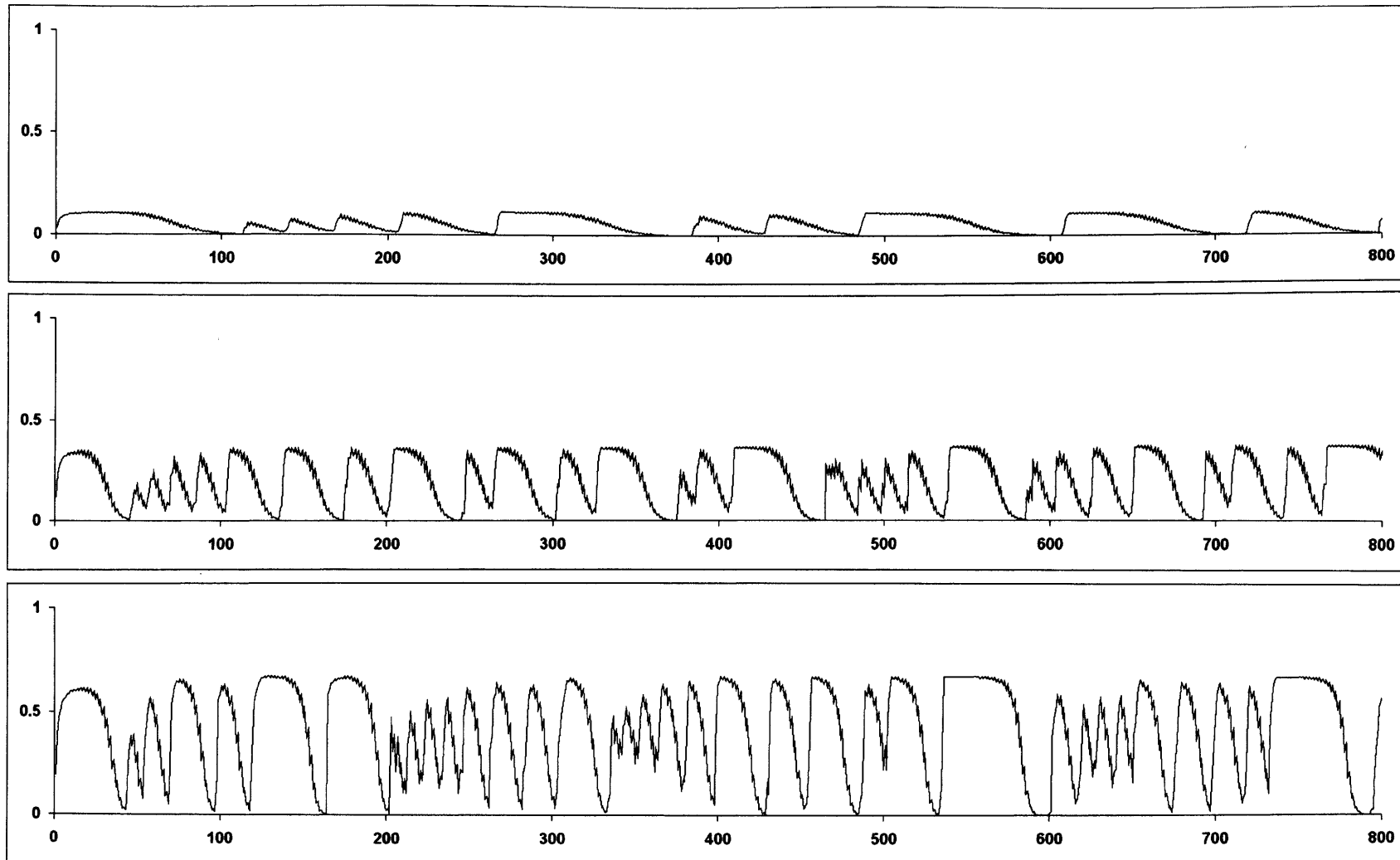


Figure 5.38: Rate,  $r^{(k)}$ , as a function of  $k$  for from top to bottom:  $\rho = 2$ ,  $\rho = 4$ ,  $\rho = 10$ ;  $d = 100$ .

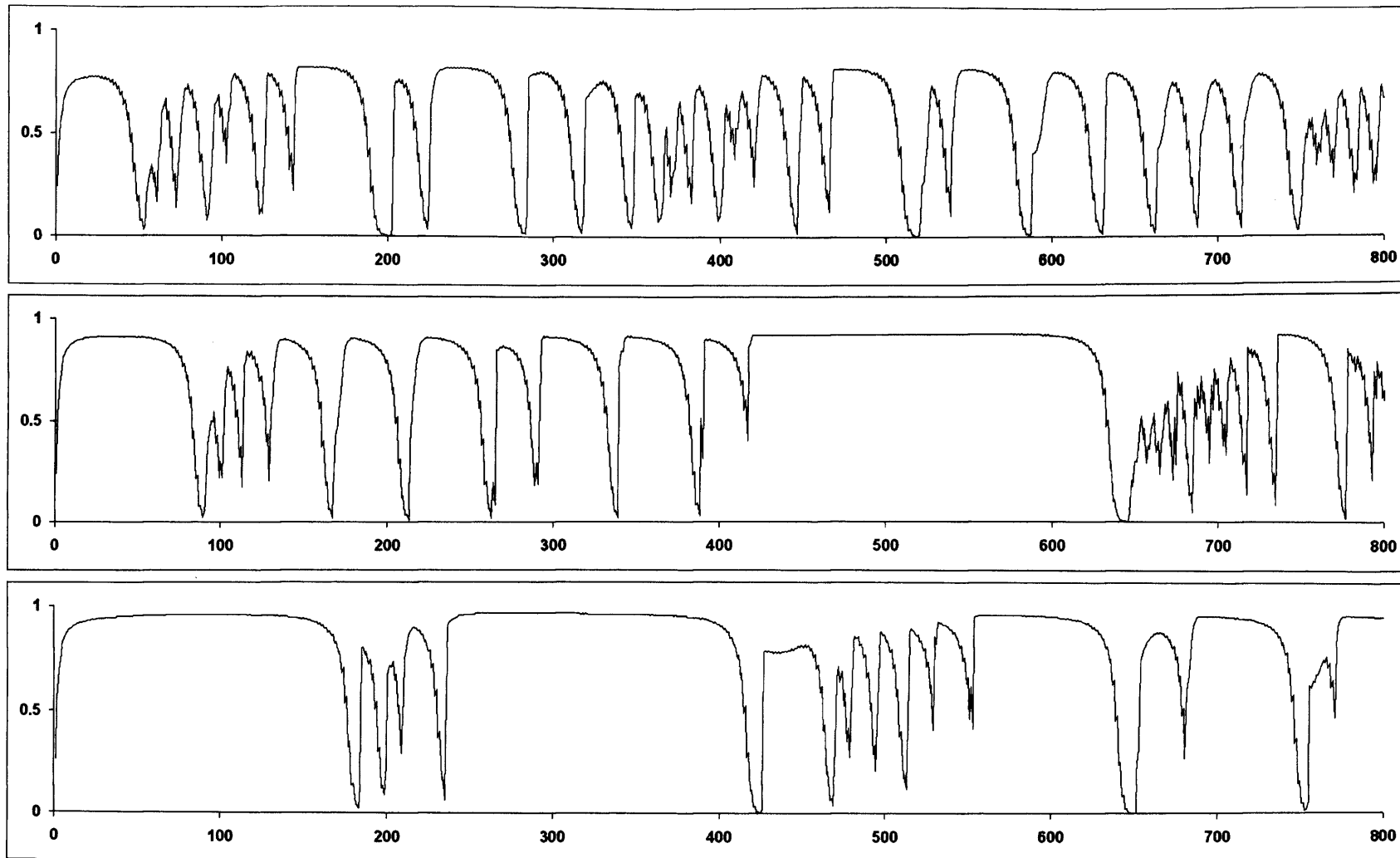


Figure 5.39: Rate,  $r^{(k)}$ , as a function of  $k$  for from top to bottom:  $\rho = 20$ ,  $\rho = 50$ ,  $\rho = 100$ ;  $d = 100$ .

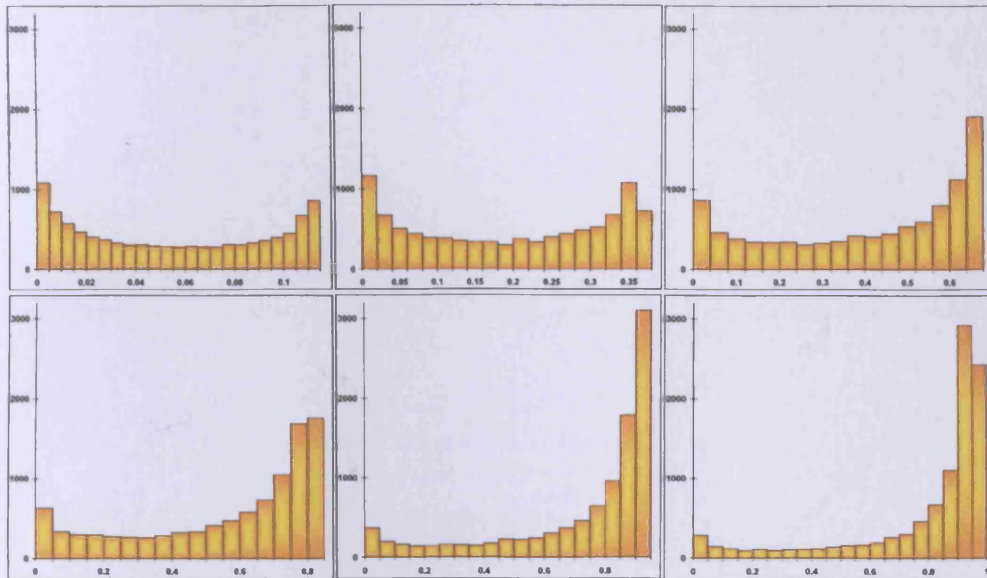


Figure 5.40: Histogram of  $r^{(k)}$  for, from top left to bottom right:  $\rho = 2$ ,  $\rho = 4$ ,  $\rho = 10$ ,  $\rho = 20$ ,  $\rho = 50$ ,  $\rho = 100$ ;  $d = 100$ ,  $k = 1, \dots, 10000$ .

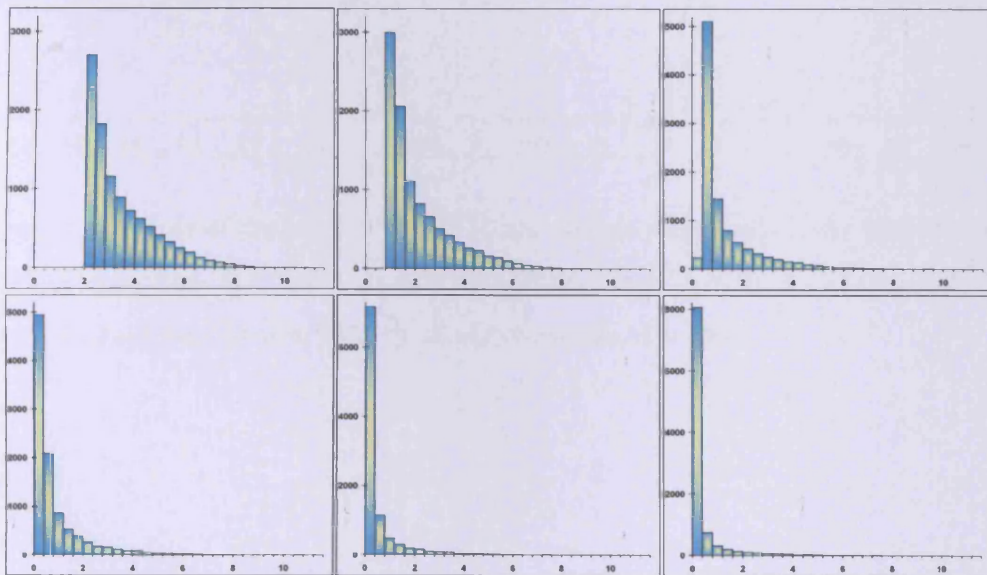


Figure 5.41: Histogram of  $(-\ln(r^{(k)}))$  for, from top left to bottom right:  $\rho = 2$ ,  $\rho = 4$ ,  $\rho = 10$ ,  $\rho = 20$ ,  $\rho = 50$ ,  $\rho = 100$ ;  $d = 100$ ,  $k = 1, \dots, 10000$ .



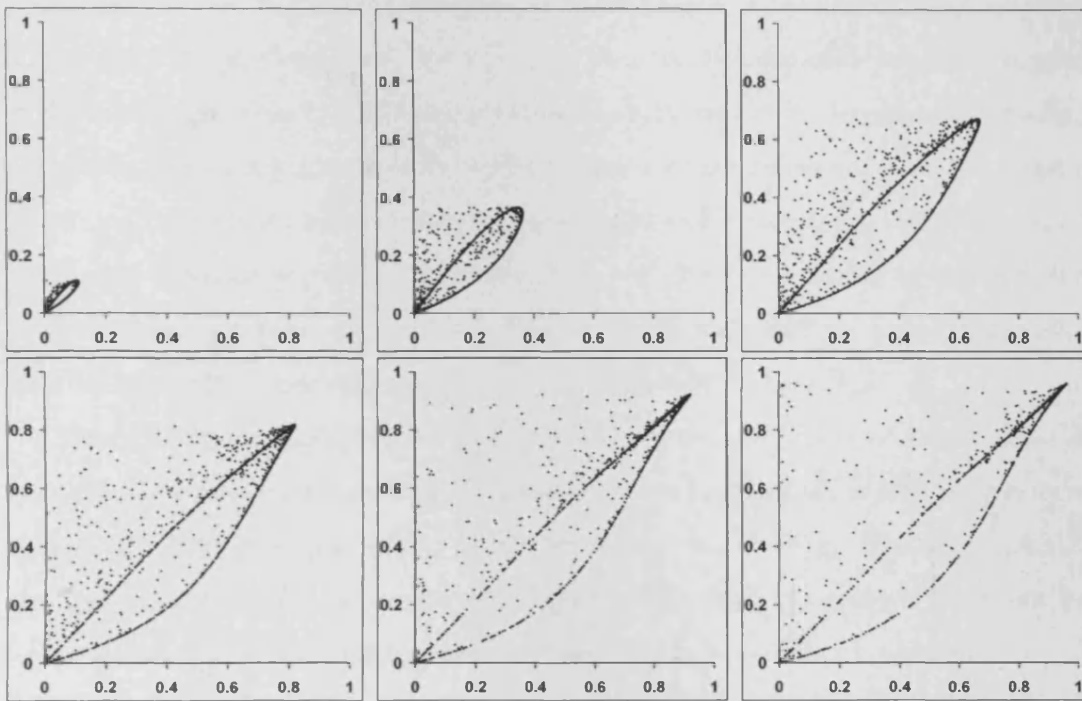


Figure 5.42: Plots of the pairs  $(r^{(k)}, r^{(k+1)})$  for a single trajectory of the  $\Phi_3$ -optimality gradient algorithm for, from top left to bottom right:  $\rho = 2, 4, 10, 20, 50, 100$ . Points plotted are the last 5000 of 10000 iterations;  $d = 100$ .

## 5.5 $\Phi_p$ -Optimality Summary

In this chapter, four new gradient optimisation algorithms were proposed. All four algorithms were derived from existing optimal experimental design criteria, namely the family of  $\Phi_p$ -optimality criteria.

The algorithm created from the  $A$ -optimality criterion showed itself to have a good asymptotic rate of convergence, faster than the worst case rate of the steepest descent algorithm,  $R_{\text{ref}}$ . This algorithm also has the additional attractive quality that all the rates  $\tau^{(k)}$  are such that  $0 < \tau^{(k)} < 1$ . This implies the monotone convergence of the algorithm. Due to this impressive asymptotic rate, the  $A$ -optimality gradient algorithm was then generalised by adding a relaxation coefficient,  $\gamma$ , in the hope of creating an algorithm with a faster asymptotic rate of convergence still. The  $(\gamma, A)$ -optimality gradient algorithm, however, does not produce a better asymptotic rate of convergence than the original  $A$ -optimality gradient algorithm. For certain ranges of  $\gamma$  the algorithm does still converge with a rate better than  $R_{\text{ref}}$ .

The next algorithm developed was based on the  $\Phi_2$ -optimality criterion. This algorithm demonstrates a very fast asymptotic rate of convergence which is superior to that of the  $A$ -optimality gradient algorithm and is close to  $R_{\text{min}}$ . The  $\Phi_2$ -optimality gradient algorithm also possesses the characteristic that the rates  $\tau^{(k)}$  remain between 0 and 1. The algorithm therefore converges in a monotonic fashion.

The success of the  $\Phi_2$ -optimality gradient algorithm in comparison to the  $A$ -optimality gradient algorithm prompted the creation of an algorithm based on the  $\Phi_3$ -optimality criterion. The figures of Section 5.4 show that the behaviour of this algorithm closely mirrors that of the  $\Phi_2$ -optimality gradient algorithm resulting in an extremely similar asymptotic rate of convergence. Since no improvement in the rate of convergence can be yielded from an algorithm with much longer formulae it is concluded that the creation of gradient algorithms from  $\Phi_p$ -optimality criteria with  $p \geq 4$  is not worthwhile due to even more complicated formulae and little chance of further amelioration in convergence rates.

# Chapter 6

## Conclusions and Further Work

### 6.1 Summary

This section will briefly outline the principal findings of each chapter in this thesis. For a more detailed review of Chapters 3-5, refer to the end-of-chapter summaries.

In Chapter 1 the principles of quadratic optimisation were introduced and descriptions of some of the most famous iterative methods for solving quadratic optimisation problems were given. The most common class of methods applied to quadratic optimisation problems is the class of gradient algorithms. It was explained that gradient algorithms can also be applied to the solution of linear equations and it was revealed that in fact two of the most famous optimisation algorithms, the steepest descent algorithm and the conjugate gradient algorithm were both first thought of in terms of solving linear systems. Other methods for solving linear systems were also outlined. At the end of Chapter 1 a comprehensive literature review was given which explored recent developments in the field of gradient optimisation algorithms. Most of the gradient algorithms that have been developed in recent times are modifications of the Barzilai-Borwein algorithm which in itself is a modification of the steepest descent algorithm. The asymptotic rates of convergence of the algorithms developed in this thesis are compared to that of the Barzilai-Borwein algorithm in Section 6.3.

The technique of renormalisation was used by Akaike in [1] to study the rate of convergence of the steepest descent algorithm. In Chapter 2 the methodology used

by Akaike is reviewed and a general formula for renormalised gradient algorithms is obtained. Once in renormalised form, it was shown that a link can be established between gradient optimisation algorithms and multiplicative algorithms for constructing optimal experimental designs. It was further demonstrated that the asymptotic rate of convergence of gradient algorithms can be expressed through the asymptotic behaviour of multiplicative algorithms for constructing optimal experimental designs for linear regression models. A general framework was established which enables new gradient algorithms to be created from any optimality criterion which takes the form of a functional of the moment matrix

$$M(\xi) = \begin{pmatrix} \mu_0 & \mu_1 \\ \mu_1 & \mu_2 \end{pmatrix}.$$

It was revealed that an algorithm which converges to an optimum design possesses the worst case asymptotic rate of convergence of the steepest descent algorithm.

In Chapter 3 the steepest descent algorithm was generalised by incorporating a relaxation coefficient,  $\gamma$ , into the original step length. The techniques used to study the behaviour of the rate of convergence of all the algorithms generated in this thesis were introduced in this chapter and were applied to the  $\gamma$ -steepest descent algorithm. It was shown that for  $1 < \gamma < 2M/(m + M)$  and  $2m/(m + M) < \gamma < 4mM/(m + M)^2$  convergence to an optimum design occurs and thus for these ranges of  $\gamma$ , the gradient algorithm possesses the worst case rate of convergence of the steepest descent algorithm,  $R_{\text{ref}}$ . For much of the range  $4mM/(m + M)^2 < \gamma < 1$  chaos is exhibited and it is within this range that improvements in the asymptotic rates of convergence are demonstrated. It was concluded that the best asymptotic rate of convergence is obtained when  $\gamma$  is slightly less than 1. The  $\gamma$ -steepest descent algorithm with  $\gamma$  slightly less than 1 is non-monotonic in its descent to the minimum point. With a sensible choice of  $\gamma$ , the  $\gamma$ -steepest descent algorithm was shown to possess an asymptotic rate of convergence faster than the relaxed steepest descent algorithm suggested in [58].

In Chapter 4 the  $\beta$ -root family of gradient algorithms was created through generalising the  $D$ -optimality criterion. The steepest descent algorithm belongs to this family of algorithms. The first  $\beta$ -root algorithm considered was the square-root

algorithm and it was shown that this algorithm possesses the worst case rate of convergence of the steepest descent algorithm since the renormalised square-root algorithm converges to an optimum design. Adding a relaxation coefficient to the step length of the square-root algorithm did not improve upon the asymptotic rate of convergence, and indeed for a large range of  $\gamma$  convergence to an optimum design was shown to occur. Other algorithms belonging to this family, however, possess much faster rates of convergence. Some values of  $\beta$  allow cycles of varying size to form in the sequence of rates  $\{r^{(k)}\}$  while others induce chaos in the sequence. The best asymptotic rates of convergence are obtained when  $\beta$  is such that chaos is present. The value of  $\beta$  which produces the fastest asymptotic rate of convergence was seen to depend on  $\rho$  but is always slightly greater than 1. Similarly to the  $\gamma$ -steepest descent and Barzilai-Borwein algorithms, the  $\beta$ -root algorithms with the fastest asymptotic rates are non-monotonic in nature. Adding a relaxation coefficient to the algorithm did not produce any further improvements in the asymptotic rate of convergence.

Using the general methodology outlined in Chapter 2, four new gradient algorithms, inspired by the so-called  $\Phi_p$ -optimality criteria, were created in Chapter 5. The first algorithm was formed from the  $A$ -optimality criterion. This algorithm was shown to exhibit chaos and has an asymptotic rate of convergence that is much better than  $R_{\text{ref}}$ . A relaxed version of the  $A$ -optimality gradient algorithm was created but it was concluded that no improvement in asymptotic rate of convergence could be yielded in this way. The  $\Phi_2$ -optimality criterion and the  $\Phi_3$ -optimality criterion were also used to create new gradient algorithms. These algorithms were seen to behave in an almost identical manner to each other and both demonstrated a substantial improvement in asymptotic rate of convergence over the gradient algorithm constructed from the  $A$ -optimality criterion. It was seen that the gradient algorithms created from the  $A$ -optimality,  $\Phi_2$ -optimality and  $\Phi_3$ -optimality criteria have the additional attractive quality that all the rates  $r^{(k)}$  for each algorithm are such that  $0 < r^{(k)} < 1$ . This implies the monotone convergence of the algorithms.

Tables containing the main formulae required to produce each of the algorithms constructed in this thesis can be found in Appendix B.

## 6.2 Further Work

All gradient algorithms constructed in this thesis have been restricted to the form

$$x^{(k+1)} = x^{(k)} - \alpha^{(k)} g^{(k)},$$

where  $\alpha^{(k)}$  is the step length at iteration  $k$  and  $g^{(k)} = \left( \frac{\partial f}{\partial x_1}, \dots, \frac{\partial f}{\partial x_d} \right)^T$  is the direction of descent. In renormalised form these algorithms have been shown to be exactly equivalent to a multiplicative algorithm for constructing optimal experimental designs for the linear regression model  $y = \theta_0 + \theta_1 x + \varepsilon$ . There are, however, other classes of algorithms whose asymptotic rates of convergence can be studied using the methodology developed in this thesis. One of these classes of algorithms is briefly considered below.

### 6.2.1 The Optimum 2-Gradient Algorithm

A logical progression in the study of the asymptotic rates of convergence of gradient algorithms would be to consider the convergence rates of those algorithms of the form

$$x^{(k+1)} = x^{(k)} - \alpha_1^{(k)} g^{(k)} - \alpha_2^{(k)} A g^{(k)}. \quad (6.1)$$

Gradient algorithms conforming to the structure set out in (6.1) can be related to multiplicative procedures for constructing optimal experimental designs for the quadratic regression model  $y = \theta_0 + \theta_1 x + \theta_2 x^2 + \varepsilon$ . In this case the optimality criterion to be maximised for each multiplicative algorithm is a functional of the moment matrix

$$M(\xi) = \begin{pmatrix} \mu_0 & \mu_1 & \mu_2 \\ \mu_1 & \mu_2 & \mu_3 \\ \mu_2 & \mu_3 & \mu_4 \end{pmatrix}.$$

The optimum  $s$ -gradient algorithm (see [32, 56]) with  $s = 2$  is an example of a gradient algorithm of this type. There, the parameters  $\alpha_1^{(k)}$  and  $\alpha_2^{(k)}$  are chosen so as to minimise  $f(x^{(k+1)})$  at each iteration. Brief analysis of the optimum 2-gradient

algorithm,

$$\begin{aligned}
x^{(k+1)} &= x^{(k)} - \frac{(g^{(k)}, g^{(k)})(A^3 g^{(k)}, g^{(k)}) - (Ag^{(k)}, g^{(k)})(A^2 g^{(k)}, g^{(k)})}{(Ag^{(k)}, g^{(k)})(A^3 g^{(k)}, g^{(k)}) - (A^2 g^{(k)}, g^{(k)})^2} g^{(k)} \\
&\quad - \frac{(g^{(k)}, g^{(k)})(A^2 g^{(k)}, g^{(k)}) - (Ag^{(k)}, g^{(k)})^2}{(Ag^{(k)}, g^{(k)})(A^3 g^{(k)}, g^{(k)}) - (A^2 g^{(k)}, g^{(k)})^2} Ag^{(k)}, \quad (6.2)
\end{aligned}$$

has been performed.

In order to study the asymptotic rate of convergence of the optimum 2-gradient algorithm it is first necessary to renormalise the algorithm. The updating formula for the renormalised optimum 2-gradient algorithm has the form

$$\xi_i^{(k+1)} = \xi_i^{(k)} \left( 1 - \lambda_i \left( \frac{\mu_3 - \mu_1 \mu_2}{\mu_1 \mu_3 - \mu_2^2} \right) + \lambda_i^2 \left( \frac{\mu_2 - \mu_1^2}{\mu_1 \mu_3 - \mu_2^2} \right) \right)^2 / r^{(k)},$$

for  $i = 1, \dots, d$ , where

$$r^{(k)} = \frac{(g^{(k+1)}, g^{(k+1)})}{(g^{(k)}, g^{(k)})} = \frac{(\mu_2 - \mu_1^2) (2\mu_2 \mu_3 \mu_1 - \mu_4 \mu_1^2 - \mu_3^2 + \mu_4 \mu_2 - \mu_2^3)}{(\mu_1 \mu_3 - \mu_2^2)^2}.$$

The asymptotic rate of convergence of the optimum 2-gradient algorithm in relation to  $R_{\min}$  and  $R_{\text{ref}}$  is depicted in Figure 6.1. It can clearly be seen that the asymptotic rate of convergence of this algorithm is much better than  $R_{\text{ref}}$  and hence is also an improvement over the asymptotic rate of convergence of the standard steepest descent algorithm. The optimum 2-gradient algorithm does not, however, possess an asymptotic rate of convergence which would compete with that of some of the other algorithms studied in this thesis, for example the  $\gamma$ -steepest descent algorithm with  $\gamma$  slightly less than 1. There is nevertheless the possibility of improving upon the asymptotic rate of convergence of the optimum 2-gradient algorithm by incorporating relaxation into the updating formula.

### 6.2.2 The Relaxed Optimum 2-Gradient Algorithm

To create the  $\gamma$ -steepest descent algorithm a relaxation coefficient,  $\gamma$ , was added to the step length,  $\alpha^{(k)}$ , of the standard steepest descent algorithm. A similar method can be applied to the optimum 2-gradient algorithm but since this algorithm possesses two parameters,  $\alpha_1^{(k)}$  and  $\alpha_2^{(k)}$  there is scope to add two relaxation coefficients,

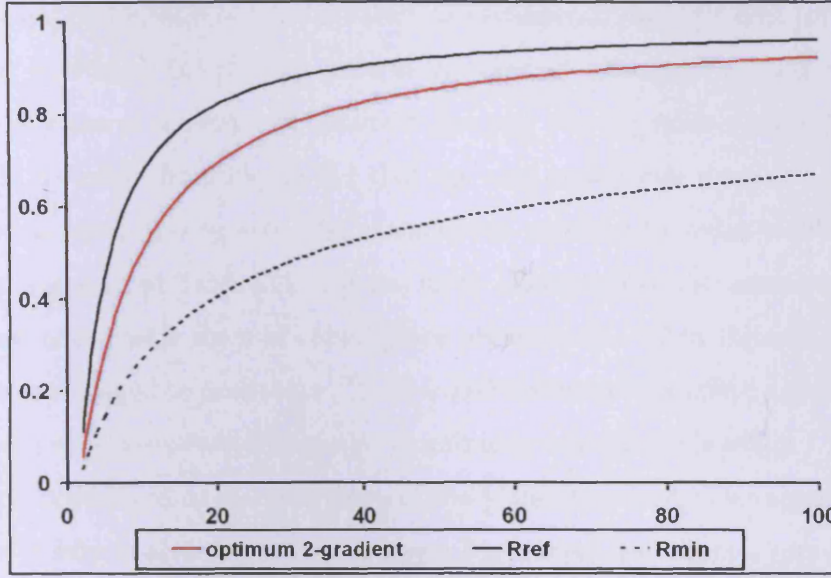


Figure 6.1: Average asymptotic rate of convergence as a function of  $\rho$  for the optimum 2-gradient algorithm with  $d = 100$ .

$\gamma_1, \gamma_2$ , to the algorithm. The result is a relaxed algorithm of the form

$$\begin{aligned} x^{(k+1)} = & x^{(k)} - \gamma_1 \left( \frac{(g^{(k)}, g^{(k)})(A^3 g^{(k)}, g^{(k)}) - (Ag^{(k)}, g^{(k)})(A^2 g^{(k)}, g^{(k)})}{(Ag^{(k)}, g^{(k)})(A^3 g^{(k)}, g^{(k)}) - (A^2 g^{(k)}, g^{(k)})^2} \right) g^{(k)} \\ & - \gamma_2 \left( \frac{(g^{(k)}, g^{(k)})(A^2 g^{(k)}, g^{(k)}) - (Ag^{(k)}, g^{(k)})^2}{(Ag^{(k)}, g^{(k)})(A^3 g^{(k)}, g^{(k)}) - (A^2 g^{(k)}, g^{(k)})^2} \right) Ag^{(k)}. \end{aligned} \quad (6.3)$$

This algorithm shall be referred to as the relaxed optimum 2-gradient algorithm.

The renormalised updating formula for the relaxed optimum 2-gradient algorithm has the form

$$\xi^{(k+1)} = \xi^{(k)} \left( 1 - \lambda_i \gamma_1 \left( \frac{\mu_3 - \mu_1 \mu_2}{\mu_1 \mu_3 - \mu_2^2} \right) + \lambda_i^2 \gamma_2 \left( \frac{\mu_2 - \mu_1^2}{\mu_1 \mu_3 - \mu_2^2} \right) \right)^2 / r^{(k)}, \quad (6.4)$$

for  $i = 1, \dots, d$ . The rate,  $r^{(k)}$ , and hence also the denominator of the updating formula (6.4) is equal to  $r^{(k)} = (g^{(k+1)}, g^{(k+1)}) / (g^{(k)}, g^{(k)}) = r_{\text{num}}^{(k)} / (\mu_1 \mu_3 - \mu_2^2)$ , where

$$\begin{aligned} r_{\text{num}}^{(k)} = & \mu_4 \gamma_2^2 \mu_1^4 + (-2\mu_2 \gamma_2 \mu_3 - 2\mu_3 \gamma_1 \gamma_2 \mu_2 + 2\gamma_1 \mu_2 \mu_3) \mu_1^3 \\ & + (\mu_2^3 \gamma_1^2 - 2\gamma_1 \mu_2^3 + \mu_3^2 + 2\mu_2^3 \gamma_2 + 2\mu_3^2 \gamma_1 \gamma_2 - 2\gamma_1 \mu_3^2 - 2\mu_4 \gamma_2^2 \mu_2) \mu_1^2 \\ & + (2\gamma_1 \mu_3 \mu_2^2 - 2\mu_3 \mu_2^2 + 2\mu_3 \gamma_1 \gamma_2 \mu_2^2 - 2\mu_2^2 \gamma_1^2 \mu_3 + 2\mu_2^2 \gamma_2 \mu_3) \mu_1 \\ & - 2\mu_2^4 \gamma_2 - 2\mu_3^2 \gamma_1 \gamma_2 \mu_2 + \mu_2^4 + \mu_2 \gamma_1^2 \mu_3^2 + \mu_4 \gamma_2^2 \mu_2^2. \end{aligned}$$

In the relaxed optimum 2-gradient algorithm there are two relaxation coefficients, thus in order to find the relaxed optimum 2-gradient algorithm which yields the best



possible asymptotic rates of convergence it is necessary to find the best combination of  $\gamma_1$  and  $\gamma_2$ . Figure B.1 in Appendix B contains an extract of a table where the asymptotic rates of convergence achieved through varying both  $\gamma_1$  and  $\gamma_2$  are displayed. It is evident from Figure B.1 that the best asymptotic rates of convergence are achieved when  $\gamma_1 = \gamma_2 = \gamma$ . The analysis was repeated for other values of  $\rho$  and  $d$  than those used in Table B.1 and the same phenomenon was seen to occur i.e. the fastest asymptotic rates of convergence always occur when the two relaxation parameters are equal to each other. The relaxed optimum 2-gradient algorithm with  $\gamma_1 = \gamma_2 = \gamma$  shall be referred to as the  $\gamma$ -optimum 2-gradient algorithm.

The asymptotic rates of convergence of the  $\gamma$ -optimum 2-gradient algorithm have also been briefly analysed. Figure 6.2 shows the average asymptotic rate of convergence of the  $\gamma$ -optimum 2-gradient algorithm as a function of  $\gamma$  for several values of  $\rho$ . Regardless of the value of the condition number  $\rho$ , the worst asymptotic rates of convergence occur when  $\gamma = 1$ , i.e. when the algorithm is equal to the standard optimum 2-gradient algorithm. Choosing a value of  $\gamma$  either slightly less, or even slightly larger, than 1 will result in a dramatic improvement in the asymptotic rate but the value of  $\gamma$  which yields the best asymptotic rates of convergence appears to be fractionally less than one.

In order to see with greater precision the value of  $\gamma$  which produces the  $\gamma$ -optimum 2-gradient algorithm with best asymptotic rate of convergence possible, Figure 6.3 shows the average asymptotic rate of convergence as a function of  $\gamma$  in the vicinity of  $\gamma = 1$ . It seems that the optimal value for the relaxation coefficient is  $\gamma \cong 0.998$ , however the exact value of  $\gamma$  which produces the best asymptotic rates of convergence will depend on the number of iterations the algorithm is run for. As the number of iterations is increased the optimal value of  $\gamma$  moves closer to 1.

Figure 6.4 shows the attractors of  $r^{(k)}$  as a function of  $\gamma$  for several values of  $\rho$ . As has been the case with every gradient algorithm that has demonstrated a comparatively fast asymptotic rate of convergence, the sequence  $\{r^{(k)}\}$  for the  $\gamma$ -optimum 2-gradient algorithm with  $\gamma$  slightly less than 1, is chaotic in nature.

From Figure 6.5 it can be seen that the asymptotic rate of convergence of the  $\gamma$ -optimum 2-gradient algorithm with  $\gamma = 0.998$  is not only considerably better

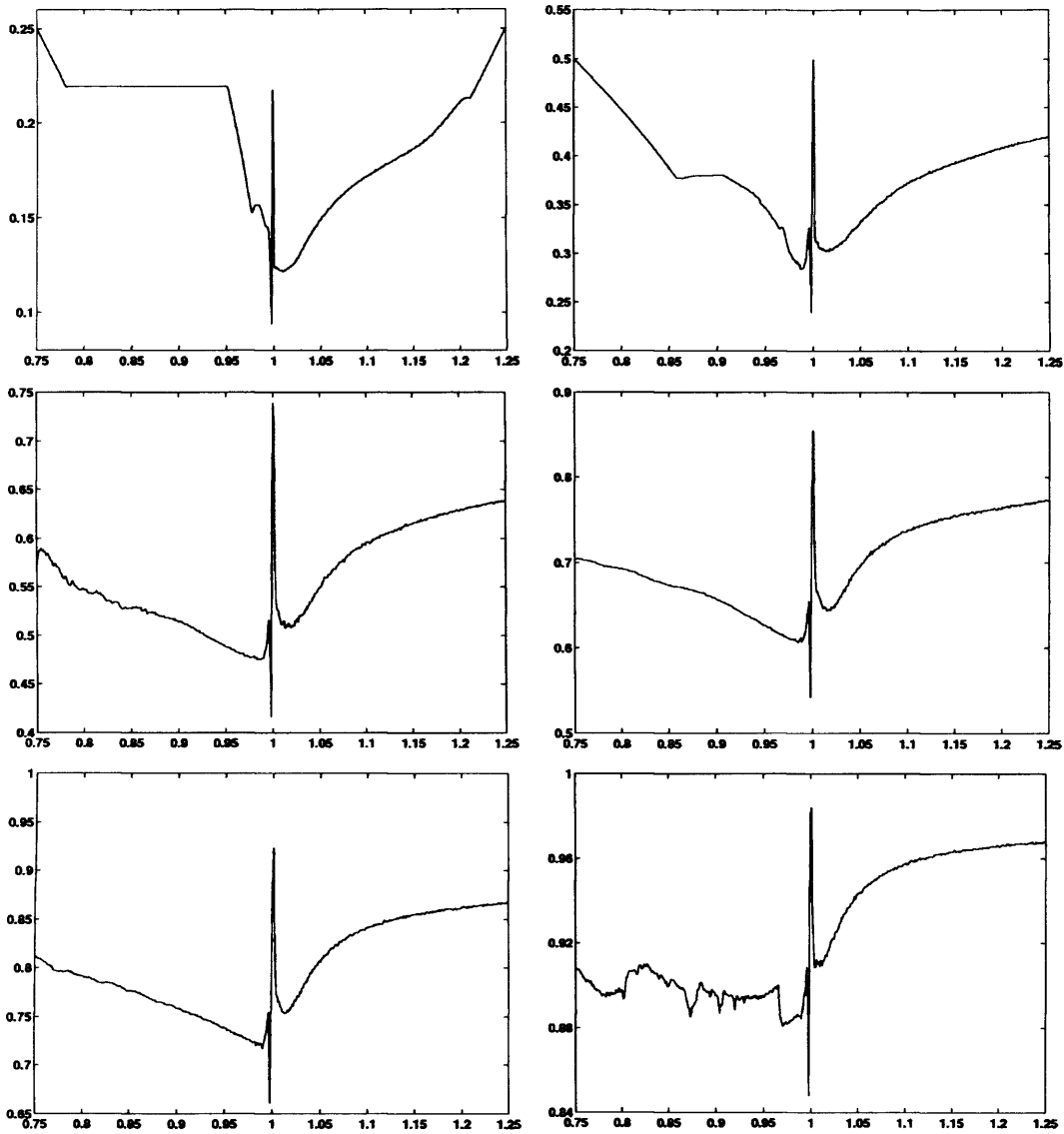


Figure 6.2: Average asymptotic rate of convergence as a function of  $\gamma$  for the  $\gamma$ -optimum 2-gradient algorithm with, from top left to bottom right,  $\rho = 4, 10, 25, 50, 100, 500$ ;  $d = 100$ .

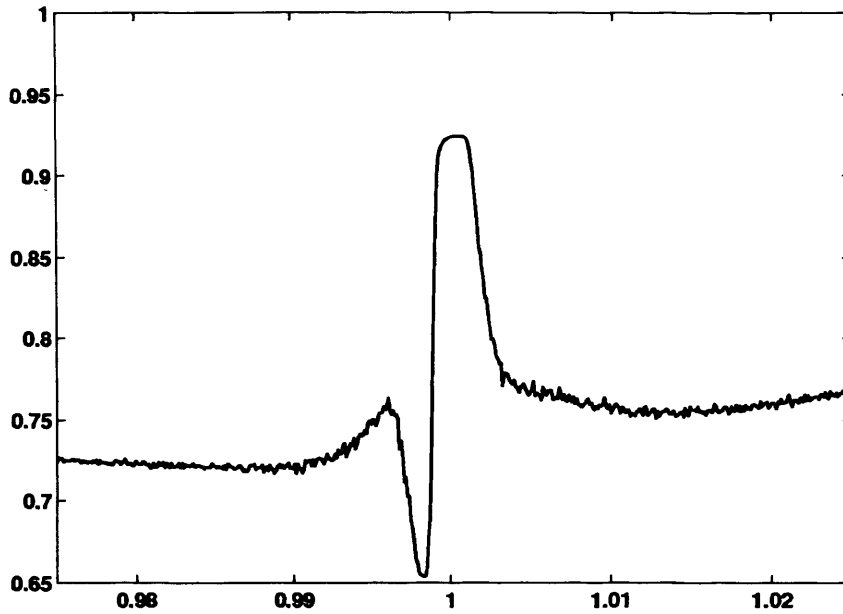


Figure 6.3: Average asymptotic rate of convergence as a function of  $\gamma$  for the  $\gamma$ -optimum 2-gradient algorithm;  $d = 100$ .

than the standard optimum 2-gradient algorithm but is also slightly better than  $R_{\min}$ , which was previously deemed to be the best asymptotic rate of convergence possible.

### 6.2.3 Future Investigations

The asymptotic rate of convergence of the  $\gamma$ -optimum 2-gradient algorithm with  $\gamma$  slightly less than 1 is better than that of all the other gradient algorithms studied in the previous chapters. For this reason alone it was felt that the  $\gamma$ -optimum 2-gradient algorithm was worthy of an albeit short discussion in this thesis. The preliminary analysis performed on the  $\gamma$ -optimum 2-gradient algorithm has shown the algorithm to be very promising and a full investigation into how and why these extremely fast asymptotic rates of convergence arise is warranted.

There is also the potential to develop other 2-gradient algorithms, i.e. algorithms of the form (6.1) with different choices of  $\alpha_1^{(k)}$  and  $\alpha_2^{(k)}$ , perhaps influenced by the  $A$ -optimality criterion for example. It is not clear at present, however, how to produce a general methodology for relating any multiplicative algorithm for constructing

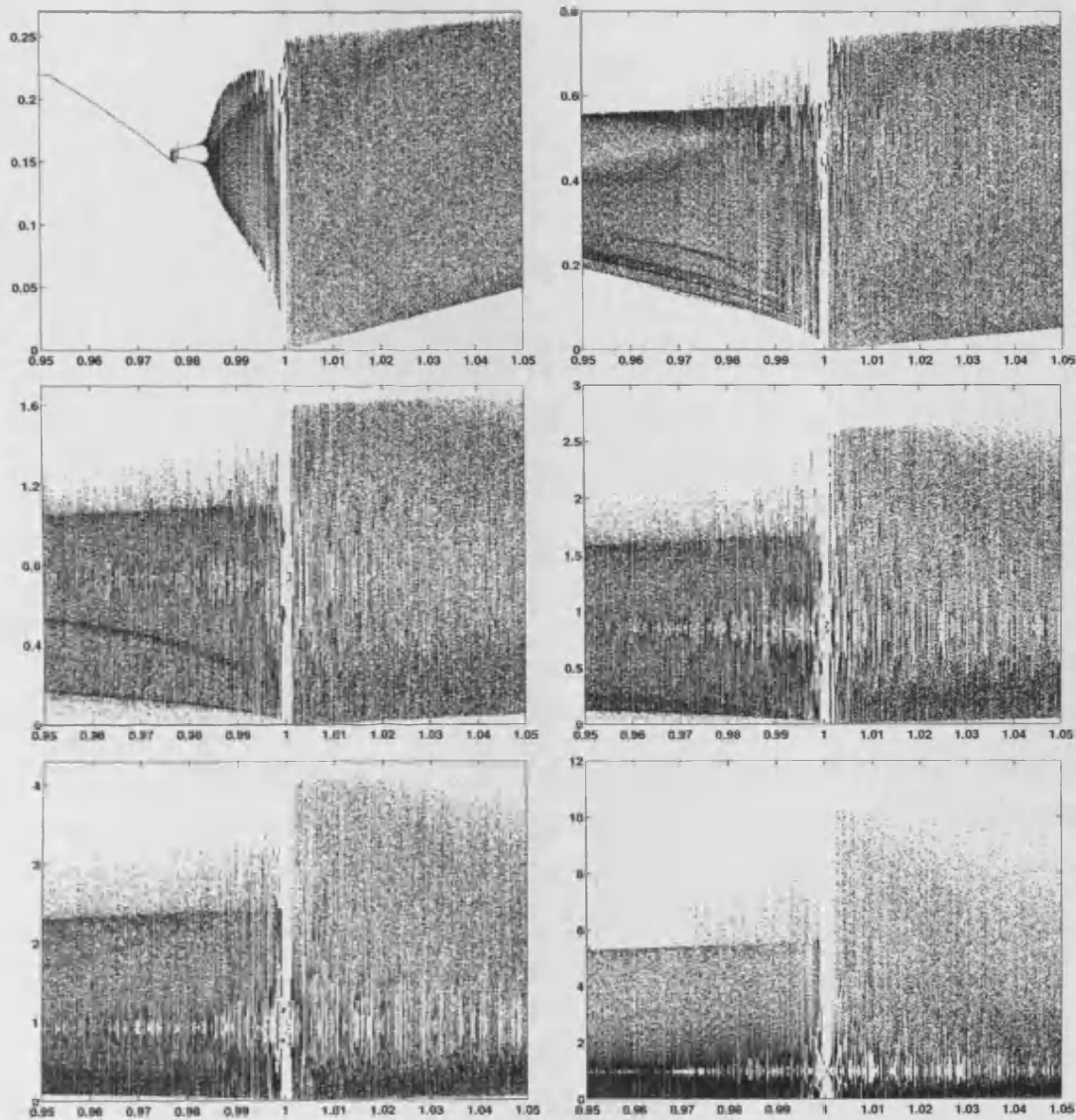


Figure 6.4: Attractors of  $\{r^{(k)}\}$  as a function of  $\gamma$  for the  $\gamma$ -optimum 2-gradient algorithm with, from top left to bottom right,  $\rho = 4, 10, 25, 50, 100, 500$ ;  $d = 100$ .

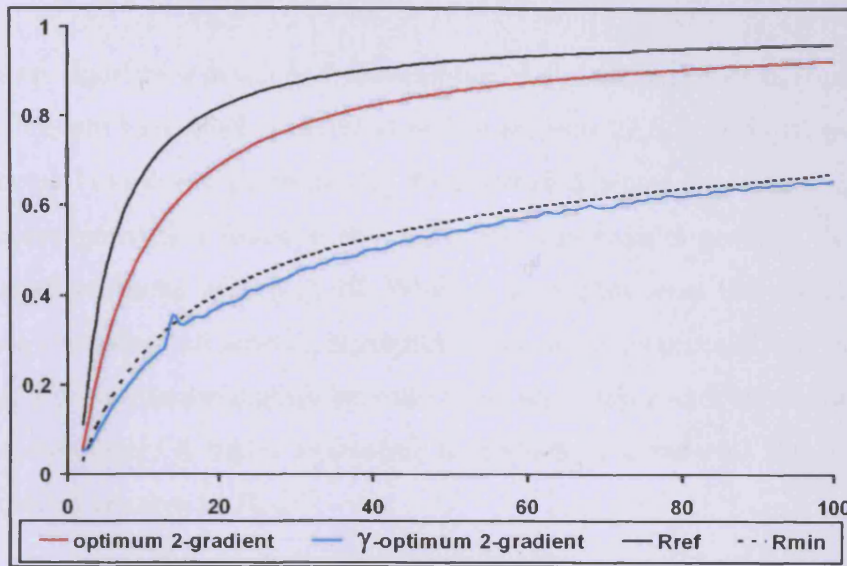


Figure 6.5: Average asymptotic rate of convergence as a function of  $\rho$  for the optimum 2-gradient and  $\gamma$ -optimum 2-gradient algorithms with  $\gamma = 0.998$ ;  $d = 100$ .

optimal experimental designs for quadratic regression models to a corresponding 2-gradient optimisation algorithm.

It has been seen that the best asymptotic rates of convergence are achieved in algorithms where chaos is present. It is not true, however, that all algorithms which behave chaotically produce fast asymptotic rates of convergence. The  $\gamma$ -steepest descent algorithm with  $\gamma = 0.5$ , for example, also exhibits chaos but the rate of convergence of this algorithm is not as fast as when  $\gamma = 0.99$ . Further research is required into discovering why some types of chaos produce faster asymptotic rates of convergence than others.

### 6.3 Comparison of Algorithms

In this section all the new algorithms constructed in this thesis which possess competitive asymptotic rates of convergence are compared, both with each other and with the Barzilai-Borwein algorithm (see [8]) and the Cauchy-Barzilai-Borwein algorithm (see [58]). When comparing those algorithms which have a relaxation coefficient inbuilt into their step lengths, the optimum choice of that relaxation coefficient

is used.

For most algorithms developed, the number of dimensions,  $d$ , of the optimisation problem does not have much of an effect on the asymptotic rate of convergence of the algorithm used to solve it, provided  $d \gtrsim 10$ . Figure 6.6, which depicts the asymptotic rate of convergence as a function of  $\rho$  for a 100-dimensional problem, is therefore typical of all problems with  $d \gtrsim 10$ . While it is evident from this figure that the algorithms considered all have an asymptotic rate of convergence that is must faster than  $R_{\text{ref}}$ , it is hard to distinguish between those algorithms with similar asymptotic rates of convergence. A better evaluation is enabled by comparing the efficiency of each algorithm relative to  $R_{\text{min}}$ .

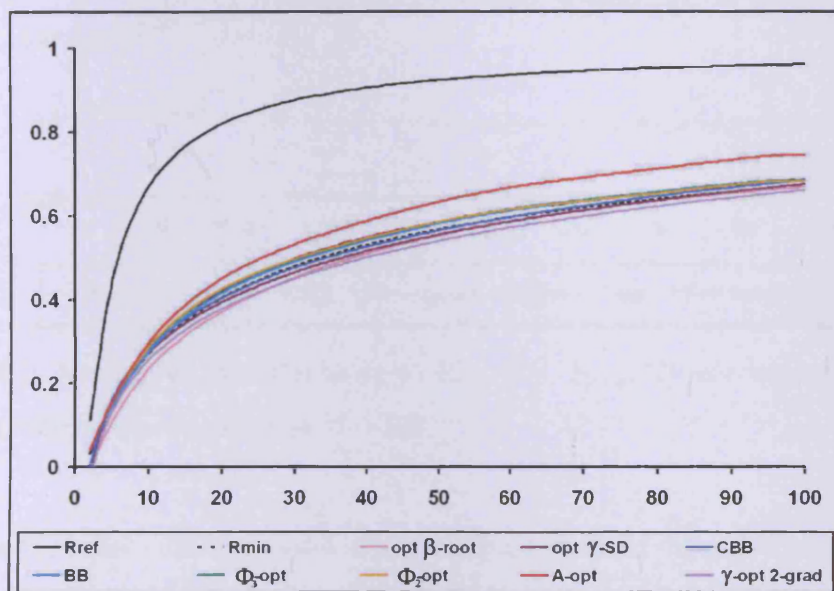


Figure 6.6: Average asymptotic rate of convergence as a function of  $\rho$  for several gradient-type algorithms;  $d = 100$ .

The average efficiency,  $(R_{\text{min}}/R)$ , of each algorithm is plotted as a function of  $\rho$  in Figure 6.7. It is necessary to mention at this point that for the  $\beta$ -root algorithm in particular, the optimum choice for the relaxation coefficient will depend on  $\rho$  and so the values of the relaxation coefficients used in Figure 6.6 and Figure 6.7 change slightly as  $\rho$  increases. The optimum  $\gamma$ -optimum 2-gradient, optimum  $\beta$ -root and optimum  $\gamma$ -steepest descent algorithms all have an efficiency greater than that of the BB and CBB algorithms which signifies that they have asymptotic rates

of convergence faster than those of the BB and CBB algorithms. As  $\rho$  increases the gradient algorithms corresponding to  $\Phi_2$ -optimality and  $\Phi_3$ -optimality approach the same level of efficiency as that of the BB algorithm whilst the gradient algorithm corresponding to the  $A$ -optimality criterion is the least efficient algorithm included in the figure.

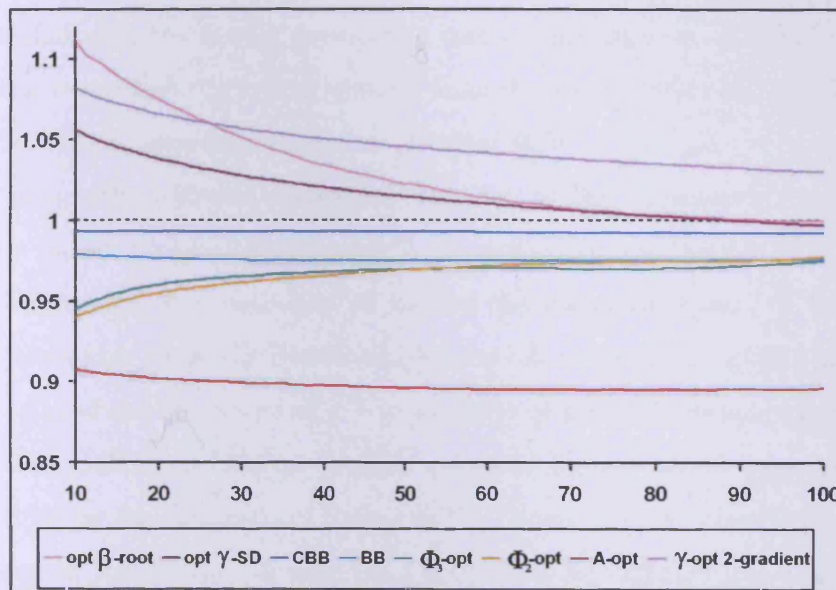


Figure 6.7: Average efficiency relative to  $R_{\min}$  (i.e.  $R_{\min}/R$ ) as a function of  $\rho$  for several gradient-type algorithms;  $d = 100$ .

For  $\rho \lesssim 85$  the optimum  $\beta$ -root and optimum  $\gamma$ -steepest descent algorithms both have an efficiency greater than 1 indicating that the asymptotic rates of convergence of these algorithms are better than  $R_{\min}$ , which was initially thought to be the best rate a gradient algorithm could achieve. For  $\rho \gtrsim 85$ , however, the efficiency is reduced to below 1 suggesting that, at least as  $\rho \rightarrow \infty$ ,  $R_{\min}$  is still the best rate a gradient algorithm can possess. The optimum  $\gamma$ -optimum 2-gradient algorithm on the other hand has an efficiency greater than 1 regardless of the condition number. It is unclear therefore, whether the asymptotic rates of convergence of 2-gradient algorithms, such as the  $\gamma$ -optimum 2-gradient algorithm, are restricted by  $R_{\min}$  in the same way as standard gradient algorithms.

## 6.4 Conclusions

Several new gradient algorithms have been created as part of this thesis, both through adapting the steepest descent algorithm and through exploiting the link established with optimal experimental design. Out of those algorithms developed, the  $\beta$ -root algorithm with  $\beta$  slightly greater than 1 has proven to be the gradient algorithm with the fastest asymptotic rate of convergence. A brief look at the  $\gamma$ -optimum 2-gradient algorithm, showed that the asymptotic rate of convergence achievable with a 2-gradient algorithm is better still.

The asymptotic rates of convergence of each of the algorithms constructed in this thesis have been studied extensively. In particular, the behaviour of the rates  $r^{(k)}$ , as  $k$  increases, was examined as well as the transition from  $r^{(k)}$  to  $r^{(k+1)}$ . A very useful plot for those algorithms which involved a relaxation coefficient was that which displayed the attractors of  $r^{(k)}$  as function of the relaxation coefficient. This enabled the quick identification of those values of the relaxation parameter which induce chaos in the sequence of rates,  $\{r^{(k)}\}$ . Some general observations on the characteristics possessed by an algorithm with a fast asymptotic rate of convergence are given below to summarise proceedings.

In their renormalised form some of those algorithms developed in this thesis converge to optimum designs. These algorithms have been shown to converge with the worst possible rate of convergence of the steepest descent algorithm. The rates of other algorithms have converged to cycles of varying size. The standard steepest-descent algorithm is an example of an algorithm whose rate,  $r^{(k)}$ , converges to a two-point cycle. The algorithms which have demonstrated the fastest asymptotic rates of convergence have all exhibited chaotic behaviour. It is not the case, however, that all algorithms which behave in a chaotic manner have fast asymptotic rates of convergence. More investigation into why some types of chaos induce faster asymptotic rates of convergence than others has been suggested as further work.

It should be noted that those algorithms which possess the fastest asymptotic rates of convergence do not produce desirable rates, close to zero, from the first iteration. Examining the sequence of rates,  $\{r^{(k)}\}$ , revealed that most algorithms



---

need a warm up period, where oscillatory behaviour is displayed, before better rates are reached. The length of the warm up period depends on the algorithm.

Those algorithms developed in this thesis which possess the best asymptotic rates of convergence are all non-monotonic in nature. Non-monotonicity is easily identifiable from examining the sequence of rates,  $\{r^{(k)}\}$ . If  $r^{(k)}$  is greater than 1 at some iterations then this corresponds to a situation where the approximation to the minimum point is further away from  $x^*$  than at the previous iteration. A step in the wrong direction seems a counter-intuitive approach to producing faster asymptotic rate of convergence, however it was observed in many of the sequences,  $\{r^{(k)}\}$ , that an extremely bad rate, greater than 1, was often followed by a period in the sequence where extremely good rates close to zero were observed. The overall effect is an improvement in the asymptotic rate of convergence.

The Barzilai-Borwein algorithm is also non-monotonic in nature. This algorithm was successfully incorporated into a non-monotonic line search technique (see [36]) in order to produce a global optimisation algorithm. It is therefore possible that the non-monotonic algorithms presented in this thesis, such as the  $\gamma$ -steepest descent and  $\beta$ -root algorithms, could also be incorporated into a non-monotone line search technique in a similar manner.

# Appendix A

## Example Programmes

### Example MatLab Programme

```
Niter=1000; % no. of iterations
NRepeat=300; % no. of repetitions
d=10; % dimension
rho=10; % condition no. of matrix A
t=(rho-1)/(d-1);
lambda=[1;1+t;1+2*t;1+3*t;1+4*t;1+5*t;1+6*t;1+7*t;1+8*t;rho]; % (equally spaced) eigenvalues of matrix A
a=-100; b=100;
row=1;
for gam=0:0.001:2, % range of relaxation coefficient
    for rep=1:NRepeat,
        rate=0;
        RATE(rep,1)=0;
        x0=a+(b-a)*rand(d,1); % random starting vector
        for i5=1:d,
            if abs(x0(i5,1))<0.5 % replacing zero components of starting vector
                x0(i5,1)=1;
            end
        end
        s=1/sum(x0(:,1).^2);
        p0(:,1)=s*x0(:,1).^2; % renormalising
        for k=1:Niter,
            mu1=sum(lambda(:,1).*p0(:,1)); mu2=sum(lambda(:,1).^2.*p0(:,1)); % calculating moments
            L=(gam^2*(mu2/mu1^2)-2*gam+1); % rate r^(k)
            pl(:,1)=p0(:,1).*(1-lambda(:,1).*gam/mu1).^2./L; % updating formula for the weights
            ss=1/sum(pl(:,1));
            p0(:,1)=pl(:,1).*ss;
            if k>100
                rate=rate+log(L);
            end
        end
        reps(rep)=rep;
        RATE(rep,1)=RATE(rep,1)+exp(rate/(Niter-100)); % asymptotic rate of convergence for 1 repetition
    end
    mean=mean(RATE(:,1))/NRepeat; % calculating average asymptotic rate for each value of gamma
    means(row)=mean; gamma(row)=gam;
    row=row+1;
end
```

Above: MatLab programme to generate average asymptotic rates of convergence for the  $\gamma$ -steepest descent algorithm for varying  $\gamma$ . The remaining parameters are fixed.

## Example Maple Programme

```

restart: Digits:=40:
with(linalg):with(stats):randomize():
rho:=100: # Condition no. of matrix A
Niter:=1000: # No. of iterations
N_Repeat:=300: # No. of repetitions
gam:=0.98: beta:=1.05: # Relaxation parameters
for d from 2 to 100 do lambda:=array(1..d): # Eigenvalues of A
i:='i': for i from 1 to d do lambda[i]:=1+(i-1)*(rho-1)/(d-1): od: # Even eigenvalue spacing
i:='i': RATE:=[]: for rep from 1 to N_Repeat do rate:=0:
# Starting vector
x0:=randmatrix(d,1): # Replacing 0 components of x0
for i5 from 1 to d do if x0[i5,1]=0 then x0[i5,1]:=1: fi: od:
s:=evalf(1/(sum((x0[i,1])^2,i=1..d))):
p0:=seq(s*(x0[i,1])^2,i=1..d): # Renormalisation
i:='i': for k from 1 to Niter do
mu[1]:=sum(lambda[i]*p0[i],i=1..d): # Calculating first moment
mu[2]:=sum((lambda[i]^2*p0[i],i=1..d): # Calculating second moment
L:=(gam^2*(mu[2]/mu[1]^2)^(2*beta-1)-2*gam*(mu[2]/mu[1]^2)^(beta-1)+1): # Rate r^(k)
p1:=seq(p0[i]*(1-lambda[i]*(gam*mu[2]^(beta-1)/(mu[1]^(2*beta-1))))^2/L,i=1..d):
ss:=1/sum(p1[i],i=1..d): # Above: updating formula for the weights
p0:=seq(p1[i]*ss,i=1..d): # Precautionary steps to ensure weights sum to 1
rate:=evalf(rate+log(L)): od:
RATE:=[op(RATE),exp(rate/Niter)]: od: # Calculating asymptotic rate of convergence
print(d,mean(RATE)): od: # Calculating average asymptotic rate

```

Above: Maple 10 programme to generate average asymptotic rates of convergence for the  $(\gamma, \beta)$ -root algorithm for varying dimensions  $d$ . The remaining parameters are fixed.

# Appendix B

## Summary Tables

Formulae for  $r^{(k)}$

Algorithm	$r^{(k)} = \frac{(g^{(k+1)}, g^{(k+1)})}{(g^{(k)}, g^{(k)})}$
Steepest Descent	$\frac{\mu_2}{\mu_1^2} - 1$
$\gamma$ -Steepest Descent	$1 - 2\gamma + \gamma^2 \frac{\mu_2}{\mu_1}$
Square-Root	$2 \left(1 - \frac{\mu_1}{\sqrt{\mu_2}}\right)$
$\gamma$ -Square Root	$1 - 2\gamma \frac{\mu_1}{\sqrt{\mu_2}} + \gamma^2$
$\beta$ -root	$\left(\frac{\mu_2}{\mu_1}\right)^{2\beta-1} - 2 \left(\frac{\mu_2}{\mu_1}\right)^{\beta-1} + 1$
$(\gamma, \beta)$ -root	$\gamma^2 \left(\frac{\mu_2}{\mu_1}\right)^{2\beta-1} - 2\gamma \left(\frac{\mu_2}{\mu_1}\right)^{\beta-1} + 1$
A-optimality	$\frac{(1+2\mu_1^2+\mu_1^2\mu_2)(\mu_2-\mu_1^2)}{\mu_1^2(1+\mu_2)^2}$
$\Phi_2$ -optimality	$\frac{(\mu_2-\mu_1^2)(\mu_1^2\mu_2^3+2\mu_1^2\mu_2^2+3\mu_2\mu_1^2+2\mu_1^4\mu_2+4\mu_1^2+1+3\mu_1^4)}{\mu_1^2(\mu_2+\mu_1^2+1+\mu_2^2)^2}$
$\Phi_3$ -optimality	$1 - 2 \frac{2\mu_2\mu_1^2+\mu_2^2\mu_1^2+\mu_1^4+3\mu_1^2+1}{(\mu_2+1)(1+2\mu_1^2+\mu_2^2)} + \frac{\mu_2(2\mu_2\mu_1^2+\mu_2^2\mu_1^2+\mu_1^4+3\mu_1^2+1)^2}{\mu_1^2(\mu_2+1)^2(1+2\mu_1^2+\mu_2^2)^2}$

Table B.1: Formulae for the rate,  $r^{(k)}$ .

Step Length Formulae

Algorithm	Step Length $\alpha^{(k)}$	$\alpha^{(k)}(\mu_1, \mu_2)$
Steepest Descent	$\frac{(g^{(k)}, g^{(k)})}{(Ag^{(k)}, g^{(k)})}$	$\frac{1}{\mu_1}$
$\gamma$ -Steepest Descent	$\gamma \frac{(g^{(k)}, g^{(k)})}{(Ag^{(k)}, g^{(k)})}$	$\frac{\gamma}{\mu_1}$
Square-Root	$\sqrt{\frac{(g^{(k)}, g^{(k)})}{(A^2g^{(k)}, g^{(k)})}}$	$\frac{1}{\sqrt{\mu_2}}$
$\gamma$ -Square Root	$\gamma \sqrt{\frac{(g^{(k)}, g^{(k)})}{(A^2g^{(k)}, g^{(k)})}}$	$\frac{\gamma}{\sqrt{\mu_2}}$
$\beta$ -root	$\frac{(g^{(k)}, g^{(k)})^\beta (A^2g^{(k)}, g^{(k)})^{\beta-1}}{(Ag^{(k)}, g^{(k)})^{2\beta-1}}$	$\frac{\mu_2^{\beta-1}}{\mu_1^{2\beta-1}}$
$(\gamma, \beta)$ -root	$\gamma \frac{(g^{(k)}, g^{(k)})^\beta (A^2g^{(k)}, g^{(k)})^{\beta-1}}{(Ag^{(k)}, g^{(k)})^{2\beta-1}}$	$\gamma \frac{\mu_2^{\beta-1}}{\mu_1^{2\beta-1}}$
A-optimality	$\frac{(g^{(k)}, g^{(k)})^2 + (Ag^{(k)}, g^{(k)})^2}{(Ag^{(k)}, g^{(k)}) [(g^{(k)}, g^{(k)}) + (A^2g^{(k)}, g^{(k)})]}$	$\frac{(1+\mu_1^2)}{\mu_1(1+\mu_2)}$
$\Phi_2$ -optimality	$\frac{(Ag^{(k)}, g^{(k)})^2 [(A^2g^{(k)}, g^{(k)}) + 2(g^{(k)}, g^{(k)})] + (g^{(k)}, g^{(k)})^3}{(Ag^{(k)}, g^{(k)}) [(g^{(k)}, g^{(k)}) (A^2g^{(k)}, g^{(k)}) + (Ag^{(k)}, g^{(k)}) 2(A^2g^{(k)}, g^{(k)}) 2(g^{(k)}, g^{(k)})^2]}$	$\frac{1+2\mu_1^2+\mu_1^2\mu_2}{\mu_1(1+\mu_2+\mu_1^2+\mu_2^2)}$
$\Phi_3$ -optimality	$\frac{2(g^{(k)}, g^{(k)}) (A^2g^{(k)}, g^{(k)}) (Ag^{(k)}, g^{(k)})^2 + (Ag^{(k)}, g^{(k)})^2 (A^2g^{(k)}, g^{(k)})^2 + (Ag^{(k)}, g^{(k)})^4 + 3(g^{(k)}, g^{(k)})^2 (Ag^{(k)}, g^{(k)})^2 + (g^{(k)}, g^{(k)})^4}{(Ag^{(k)}, g^{(k)}) ((g^{(k)}, g^{(k)}) + (A^2g^{(k)}, g^{(k)})) ((g^{(k)}, g^{(k)})^2 + 2(Ag^{(k)}, g^{(k)})^2 + (A^2g^{(k)}, g^{(k)})^2)}$	$\frac{\mu_2^2\mu_1^2+2\mu_1^2\mu_2+\mu_1^4+3\mu_1^2+1}{\mu_1(\mu_2+1)(\mu_2^2+2\mu_1^2+1)}$

Table B.2: Formulae for  $\alpha^{(k)}$  (the step length at iteration  $k$ .)

## Optimal Design Formulae

Algorithm	$\Phi(\xi)$	Limiting Behaviour of $\{\Phi(\xi^{(k)})\}$
Steepest Descent	$\mu_2 - \mu_1^2$	cycle
$\gamma$ -Steepest Descent	$\gamma\mu_2 - \mu_1^2$	depends on $\gamma$
Square-Root	$\sqrt{\mu_2} - \mu_1$	optimum design
$\gamma$ -Square Root	$\gamma\sqrt{\mu_2} - \mu_1$	depends on $\gamma$
$\beta$ -root	$\mu_2^\beta - \mu_1^{2\beta}$	depends on $\beta$
$(\gamma, \beta)$ -root	$\gamma\mu_2^\beta - \mu_1^{2\beta}$	depends on $\gamma, \beta$
A-optimality	$\frac{\mu_2 - \mu_1^2}{1 + \mu_2}$	chaos
$\Phi_2$ -optimality	$\frac{(\mu_2 - \mu_1^2)^2}{1 + \mu_2^2 + 2\mu_1^2}$	chaos
$\Phi_3$ -optimality	$\frac{(\mu_2 - \mu_1^2)^3}{(\mu_2 + 1)(1 - \mu_2 + 3\mu_1^2 + \mu_2^2)}$	chaos

Table B.3: The optimality criterion  $\Phi(\xi)$  and the corresponding limiting behaviour of the sequence  $\{\Phi(\xi^{(k)})\}$  for each algorithm.

Algorithm	$c = \min \varphi(x, \xi)$	$\varphi(x, \xi) = f^T(x) \overset{\circ}{\Phi}(\xi) f(x)$	$\text{tr}M(\xi) \overset{\circ}{\Phi}(\xi)$
Steepest Descent	$\Phi(\xi)$	$\mu_2 - 2x\mu_1 + x^2$	$2\Phi(\xi)$
$\gamma$ -Steepest Descent	$\gamma\mu_2 - \frac{\mu_1^2}{\gamma}$	$\gamma\mu_2 - 2\mu_1x + \gamma x^2$	$2\Phi(\xi)$
Square-Root	0	$\frac{\sqrt{\mu_2}}{2} \left(1 - \frac{x}{\sqrt{\mu_2}}\right)^2$	$\Phi(\xi)$
$\gamma$ -Square Root	$\sqrt{\mu_2} \left(\frac{\gamma}{2} - \frac{1}{2\gamma}\right)$	$\frac{\gamma\mu_2 - 2x\sqrt{\mu_2} + \gamma x^2}{2\sqrt{\mu_2}}$	$\Phi(\xi)$
$\beta$ -root	$\beta \left(\frac{\mu_2^{2\beta-1} - \mu_1^{4\beta-2}}{\mu_2^{\beta-1}}\right)$	$\beta \left(\mu_2^\beta - 2\mu_1^{2\beta-1}x + \mu_2^{\beta-1}x^2\right)$	$2\beta\Phi(\xi)$
$(\gamma, \beta)$ -root	$\beta \left(\frac{\gamma^2\mu_2^{2\beta-1} - \mu_1^{4\beta-2}}{\gamma\mu_2^{\beta-1}}\right)$	$\beta \left(\gamma\mu_2^\beta - 2\mu_1^{2\beta-1}x + \gamma\mu_2^{\beta-1}x^2\right)$	$2\beta\Phi(\xi)$
A-optimality	$\frac{(\mu_2 - \mu_1^2)^2}{(\mu_1^2 + 1)(\mu_2 + 1)^2}$	$\frac{(x - \mu_1)^2 + (x\mu_1 - \mu_2)^2}{(\mu_2 + 1)^2}$	$\Phi(\xi)$
$\Phi_2$ -optimality	$\frac{2(\mu_2 - \mu_1^2)^4}{(\mu_1^2\mu_2 + 2\mu_1^2 + 1)(\mu_2^2 + 1 + 2\mu_1^2)^2}$	$\frac{\phi_2}{(\mu_2^2 + 1 + 2\mu_1^2)^2}$	$2\Phi(\xi)$
$\Phi_3$ -optimality	$\frac{3(\mu_2 - \mu_1^2)^6}{(\mu_2 + 1)^2(2\mu_2\mu_1^2 + \mu_2^2\mu_1^2 + \mu_1^4 + 3\mu_1^2 + 1)(1 - \mu_2 + 3\mu_1^2 + \mu_2^2)^2}$	$\frac{\phi_3}{(\mu_2 + 1)^2(1 - \mu_2 + 3\mu_1^2 + \mu_2^2)^2}$	$3\Phi(\xi)$

Table B.4: Formulae for  $c$ ,  $\varphi(x, \xi)$  and  $\text{tr}M(\xi) \overset{\circ}{\Phi}(\xi)$  for each algorithm.

$$\phi_2 = 2(\mu_2 - \mu_1^2)(\mu_2^3 + 2\mu_1^2\mu_2 + \mu_1^2 - 2x\mu_1\mu_2^2 - 2x\mu_1^3 - 2x\mu_1\mu_2 - 2x\mu_1 + x^2\mu_1^2\mu_2 + 2x^2\mu_1^2 + x^2)$$

$$\phi_3 = 3(\mu_2^4 + 3\mu_2^2\mu_1^2 + 2\mu_1^2\mu_2 + \mu_1^4 + \mu_1^2 - 2x\mu_1\mu_2^3 - 2x\mu_1\mu_2^2 - 4x\mu_1^3\mu_2 - 4x\mu_1^3 - 2x\mu_1\mu_2 - 2x\mu_1 + x^2\mu_2^2\mu_1^2 + 2x^2\mu_1^2\mu_2 + x^2\mu_1^4 + 3x^2\mu_1^2 + x^2)(-\mu_2 + \mu_1^2)^2$$

$e_1 \backslash e_2$	0.95	0.952	0.954	0.956	0.958	0.96	0.962	0.964	0.966	0.968	0.97	0.972	0.974	0.976	0.978	0.98	0.982	0.984	0.986	0.988	0.99	0.992	0.994	0.996	0.998	1
0.95	0.7376	0.7564	0.794	0.816	0.8059	0.7962	0.793	0.7894	0.777	0.7709	0.7691	0.7661	0.7654	0.7667	0.7666	0.7671	0.769	0.769	0.771	0.7699	0.7745	0.7747	0.775	0.7781	0.7827	0.7862
0.952	0.756	0.7372	0.759	0.799	0.8164	0.8054	0.7962	0.7926	0.784	0.7756	0.7692	0.7667	0.7653	0.766	0.7667	0.7669	0.7687	0.769	0.771	0.7716	0.7715	0.7726	0.777	0.7749	0.7798	0.7836
0.954	0.7629	0.7557	0.735	0.761	0.7979	0.8176	0.803	0.7944	0.789	0.7815	0.7713	0.7679	0.7663	0.7648	0.765	0.766	0.7665	0.768	0.768	0.7699	0.7727	0.7731	0.774	0.7754	0.777	0.7811
0.956	0.7679	0.7631	0.754	0.734	0.7615	0.8032	0.8145	0.7983	0.794	0.7882	0.777	0.7697	0.7667	0.7652	0.7654	0.7641	0.7652	0.767	0.767	0.7692	0.7704	0.7719	0.773	0.7739	0.778	0.7792
0.958	0.7702	0.768	0.764	0.755	0.7345	0.7608	0.8074	0.8125	0.797	0.793	0.7857	0.7737	0.7675	0.765	0.7642	0.7645	0.7638	0.766	0.766	0.7682	0.7685	0.7702	0.772	0.7738	0.7736	0.7768
0.96	0.7708	0.7698	0.767	0.762	0.7553	0.7332	0.7671	0.8141	0.811	0.7968	0.7923	0.7825	0.7698	0.7648	0.7642	0.7628	0.7646	0.763	0.763	0.7654	0.769	0.7693	0.771	0.7709	0.7741	0.7756
0.962	0.7734	0.7714	0.769	0.766	0.7628	0.7548	0.7339	0.7676	0.813	0.808	0.7941	0.7887	0.779	0.7689	0.7642	0.7632	0.7621	0.762	0.764	0.7668	0.7663	0.7692	0.769	0.7705	0.7743	0.775
0.964	0.7741	0.774	0.771	0.769	0.766	0.7635	0.7551	0.7322	0.769	0.8188	0.8074	0.7928	0.7873	0.7726	0.7648	0.7632	0.7615	0.763	0.763	0.7635	0.7662	0.7673	0.768	0.771	0.7715	0.7731
0.966	0.7747	0.7745	0.773	0.772	0.7693	0.7684	0.7642	0.7553	0.773	0.7694	0.8166	0.8024	0.7908	0.7848	0.7691	0.7643	0.7634	0.761	0.762	0.7628	0.7645	0.7672	0.766	0.7692	0.769	0.7714
0.968	0.7762	0.7747	0.774	0.774	0.7715	0.77	0.7682	0.7637	0.755	0.7288	0.7751	0.8197	0.7973	0.7902	0.7781	0.7669	0.7631	0.761	0.761	0.762	0.7628	0.7647	0.765	0.7682	0.7687	0.7715
0.97	0.7772	0.7762	0.775	0.774	0.7734	0.7721	0.7706	0.769	0.763	0.7557	0.7273	0.7877	0.8217	0.7963	0.7886	0.7713	0.7646	0.76	0.76	0.7599	0.7623	0.7631	0.766	0.766	0.7676	0.7694
0.972	0.777	0.7774	0.775	0.775	0.7754	0.7726	0.7735	0.7711	0.768	0.7647	0.7554	0.7286	0.7894	0.8112	0.7945	0.7837	0.7671	0.762	0.76	0.759	0.7596	0.7625	0.762	0.7652	0.7664	0.7689
0.974	0.7778	0.7771	0.778	0.777	0.7777	0.7745	0.7739	0.7725	0.77	0.7687	0.7655	0.7553	0.7261	0.7982	0.8092	0.7915	0.7783	0.764	0.759	0.7594	0.7584	0.7616	0.763	0.7643	0.7645	0.7664
0.976	0.7776	0.7771	0.778	0.778	0.7764	0.7758	0.7744	0.7741	0.772	0.7705	0.77	0.7662	0.7571	0.7259	0.8079	0.8067	0.7923	0.769	0.761	0.7577	0.7564	0.7584	0.761	0.7603	0.7649	0.7648
0.978	0.7793	0.7781	0.779	0.777	0.7765	0.7765	0.7754	0.7756	0.774	0.7717	0.7721	0.7711	0.7661	0.7577	0.7245	0.8114	0.7978	0.786	0.765	0.7589	0.7569	0.7582	0.759	0.7606	0.7618	0.7642
0.98	0.7783	0.778	0.778	0.778	0.7763	0.7762	0.7756	0.7744	0.776	0.7742	0.7749	0.7728	0.7697	0.7672	0.7596	0.7231	0.8131	0.795	0.778	0.7605	0.7586	0.7566	0.757	0.7583	0.7599	0.7613
0.982	0.7777	0.7786	0.778	0.777	0.7785	0.7773	0.7774	0.7755	0.775	0.7756	0.7756	0.7749	0.7736	0.771	0.7682	0.7587	0.7215	0.816	0.795	0.7674	0.7565	0.7541	0.756	0.7561	0.7579	0.76
0.984	0.7776	0.7765	0.776	0.777	0.7774	0.7773	0.7764	0.777	0.777	0.7762	0.7754	0.7745	0.7751	0.7737	0.7718	0.7708	0.7601	0.723	0.825	0.7928	0.7615	0.7538	0.756	0.7547	0.7564	0.7575
0.986	0.7776	0.778	0.778	0.777	0.7764	0.777	0.7786	0.7761	0.777	0.7763	0.7761	0.7755	0.7764	0.7756	0.7754	0.7741	0.7698	0.765	0.722	0.8182	0.7829	0.7566	0.754	0.7548	0.7556	0.7571
0.988	0.7777	0.7789	0.777	0.777	0.7781	0.7771	0.7769	0.7771	0.778	0.7756	0.7761	0.7767	0.7766	0.7753	0.7764	0.7741	0.7751	0.772	0.767	0.7173	0.8104	0.7644	0.754	0.7523	0.7525	0.7529
0.99	0.7778	0.776	0.777	0.775	0.7783	0.7779	0.777	0.7772	0.779	0.7763	0.7772	0.7786	0.7782	0.7766	0.7759	0.777	0.776	0.776	0.775	0.7693	0.719	0.8026	0.756	0.7526	0.7516	0.7508
0.992	0.7771	0.7779	0.779	0.778	0.777	0.7766	0.7768	0.7771	0.778	0.7771	0.7769	0.7769	0.777	0.7782	0.7769	0.7774	0.7768	0.776	0.777	0.776	0.7702	0.7273	0.801	0.7524	0.7505	0.7498
0.994	0.7766	0.777	0.778	0.779	0.7767	0.7785	0.7766	0.7783	0.778	0.7759	0.7779	0.777	0.7781	0.7764	0.7777	0.7784	0.7778	0.778	0.777	0.7787	0.7747	0.7732	0.749	0.7742	0.748	0.7483
0.996	0.7776	0.7763	0.778	0.778	0.7777	0.779	0.777	0.7783	0.778	0.777	0.7776	0.7765	0.7784	0.7777	0.7774	0.7783	0.7773	0.778	0.778	0.7812	0.7783	0.778	0.78	0.7661	0.7553	0.7492
0.998	0.7775	0.7763	0.778	0.777	0.7775	0.7776	0.7783	0.7774	0.777	0.7785	0.7756	0.7763	0.7767	0.777	0.7779	0.7784	0.7799	0.777	0.778	0.7795	0.7802	0.7798	0.779	0.7954	0.6246	0.7525
1	0.777	0.7784	0.778	0.777	0.7766	0.7772	0.7768	0.7759	0.778	0.7778	0.7764	0.7773	0.777	0.7781	0.7782	0.7774	0.7782	0.781	0.78	0.7782	0.7795	0.7807	0.781	0.7878	0.7926	0.9232

Figure B.1: Above: Table showing the asymptotic rates of convergence of the relaxed optimum 2-gradient algorithm for varying relaxation parameters  $\gamma_1$  (denoted  $e_1$  above) and  $\gamma_2$  (denoted  $e_2$  above). Green values indicate where the fastest asymptotic rates of convergence occur, the yellow indicates the fastest of all. Here  $\rho = 100$ ,  $d = 100$ .



# Appendix C

## Extra Graphs

### The $\gamma$ -Steepest Descent Algorithm

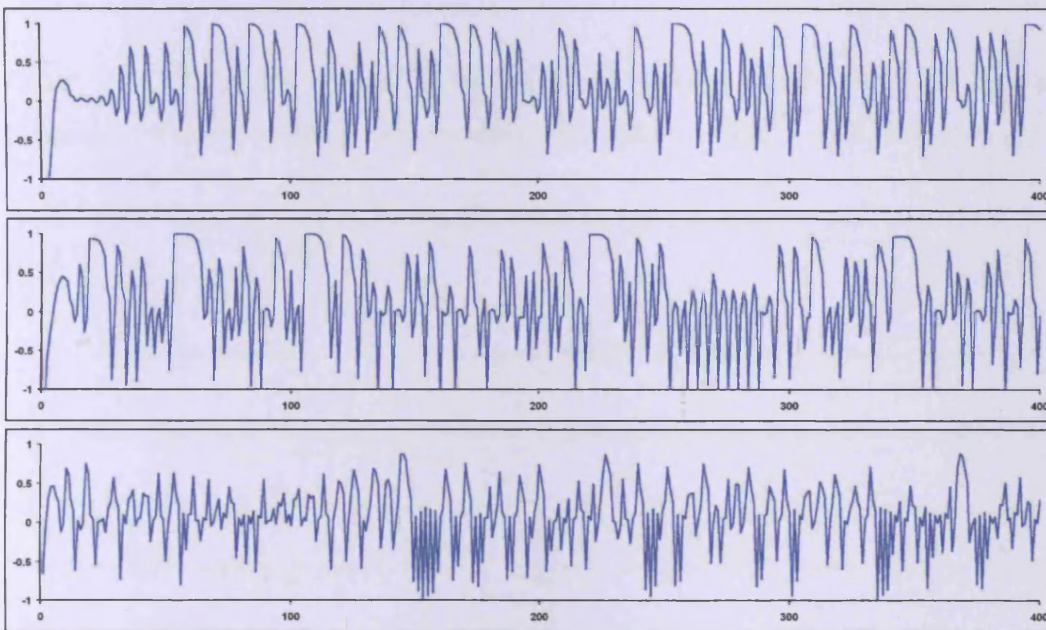


Figure C.1: Trajectory of  $\Phi(\xi^{(k)})/\max_{\xi} \Phi(\xi)$  plotted as a function of  $k$  for a single realisation of the  $\gamma$ -steepest descent algorithm with (from top to bottom)  $\gamma = 0.5$ ,  $\gamma = 0.6$ ,  $\gamma = 0.7$ ;  $\rho = 10$ ,  $d = 100$ .

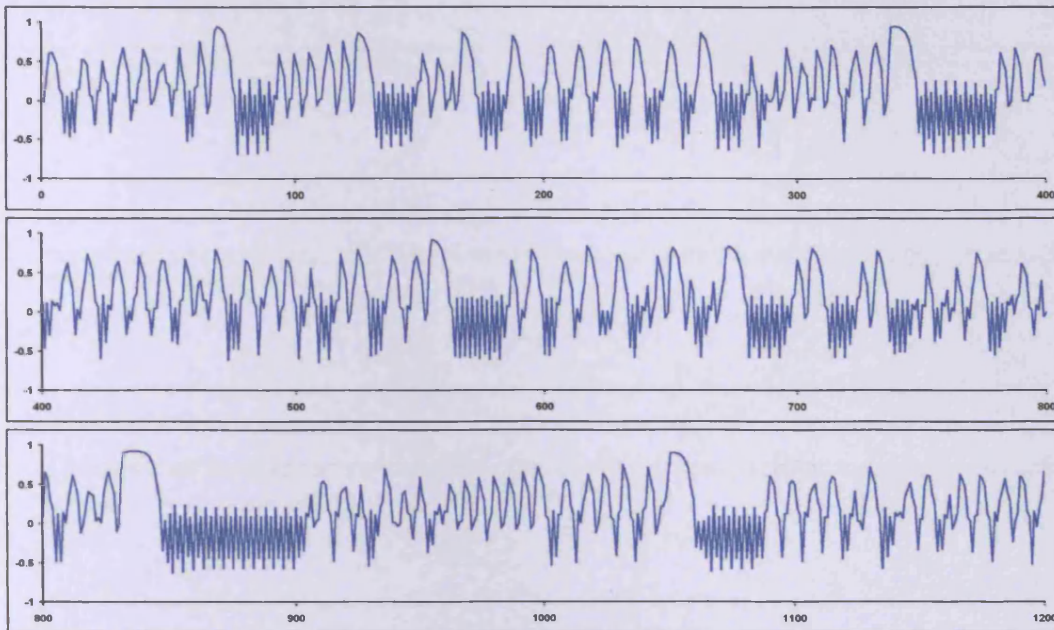


Figure C.2: Trajectory of  $\Phi(\xi^{(k)})/\max_{\xi}\Phi(\xi)$  plotted as a function of  $k$  for a single realisation of the  $\gamma$ -steepest descent algorithm with  $\gamma = 0.8$ ;  $\rho = 10$ ,  $d = 100$ .

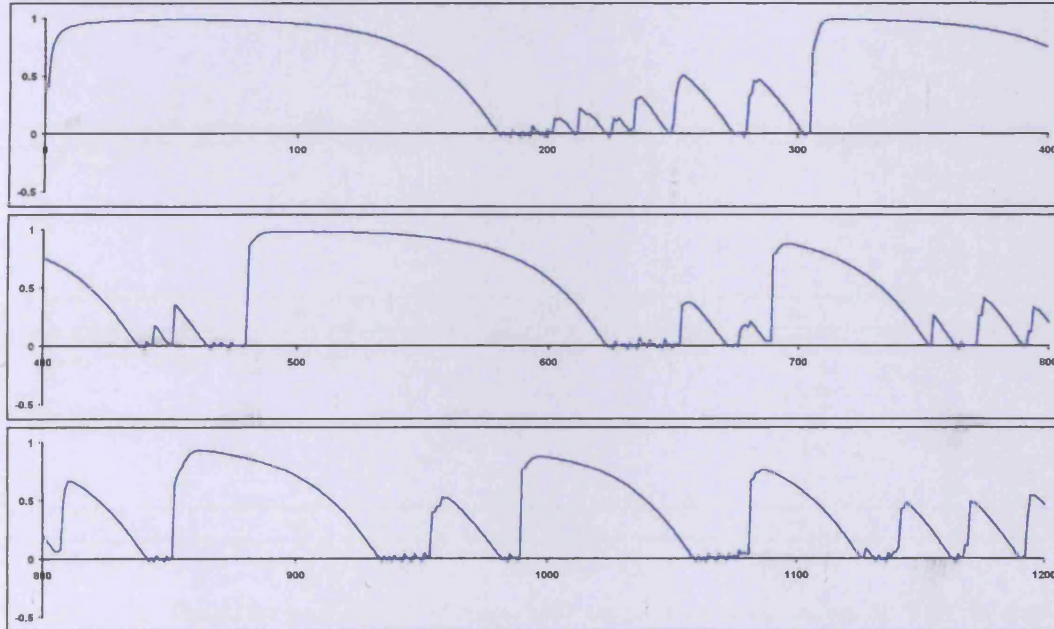


Figure C.3: Trajectory of  $\Phi(\xi^{(k)})/\max_{\xi}\Phi(\xi)$  plotted as a function of  $k$  for a single realisation of the  $\gamma$ -steepest descent algorithm with  $\gamma = 0.99$ ;  $\rho = 10$ ,  $d = 100$ .

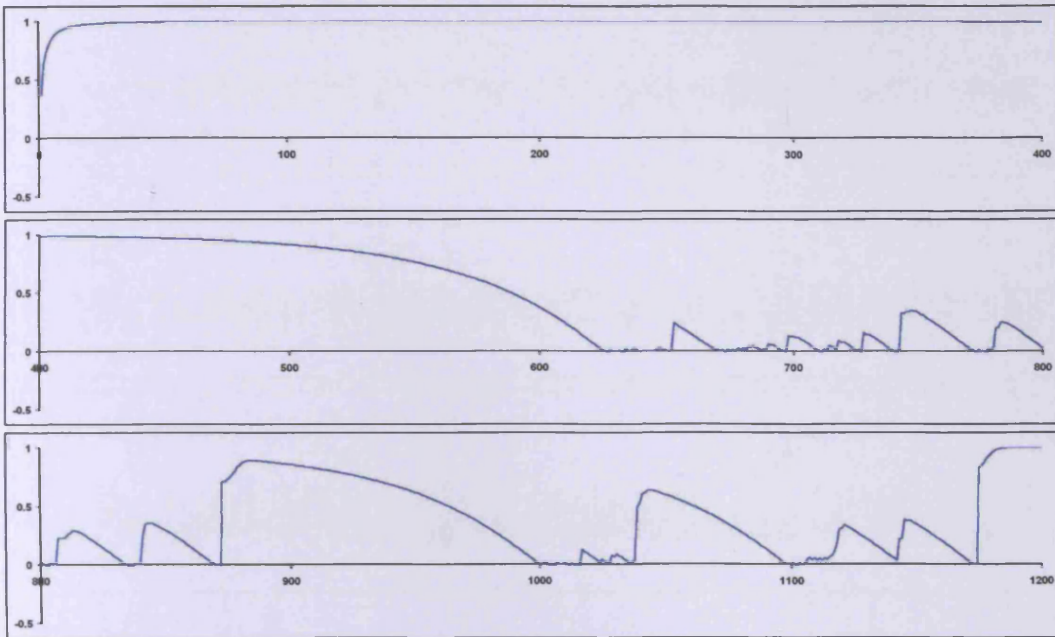


Figure C.4: Trajectory of  $\Phi(\xi^{(k)})/\max_{\xi} \Phi(\xi)$  plotted as a function of  $k$  for a single realisation of the  $\gamma$ -steepest descent algorithm with  $\gamma = 0.995$ ;  $\rho = 10$ ,  $d = 100$ .

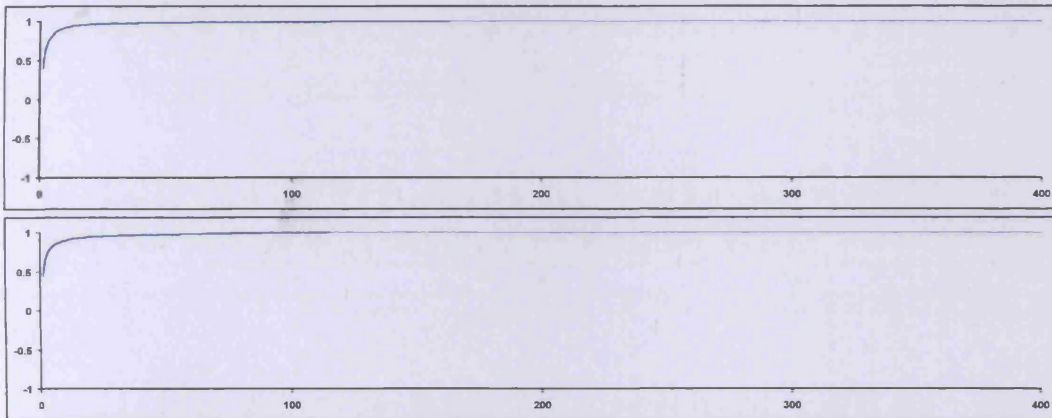


Figure C.5: Trajectory of  $\Phi(\xi^{(k)})/\max_{\xi} \Phi(\xi)$  plotted as a function of  $k$  for a single realisation of the  $\gamma$ -steepest descent algorithm with (from top to bottom)  $\gamma = 1$ ,  $\gamma = 1.1$ ,  $\rho = 10$ ,  $d = 100$ .

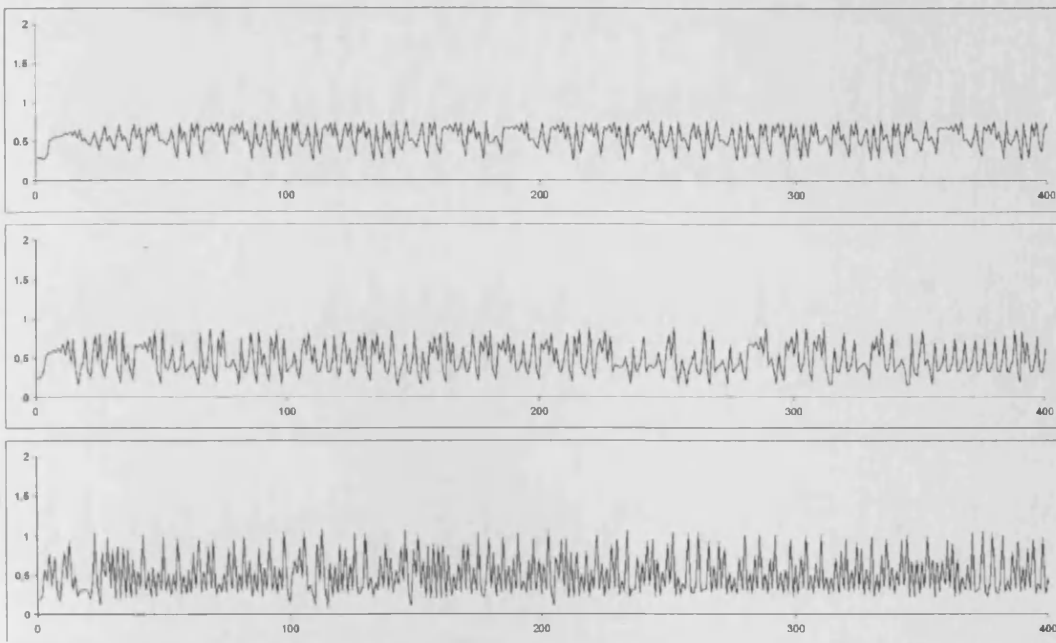


Figure C.6: Rate,  $r^{(k)}$ , as a function of  $k$  for the  $\gamma$ -steepest descent algorithm with  $d = 100$ ,  $\rho = 10$  and  $\gamma = 0.5$  (top),  $\gamma = 0.6$  (middle),  $\gamma = 0.7$  (bottom).

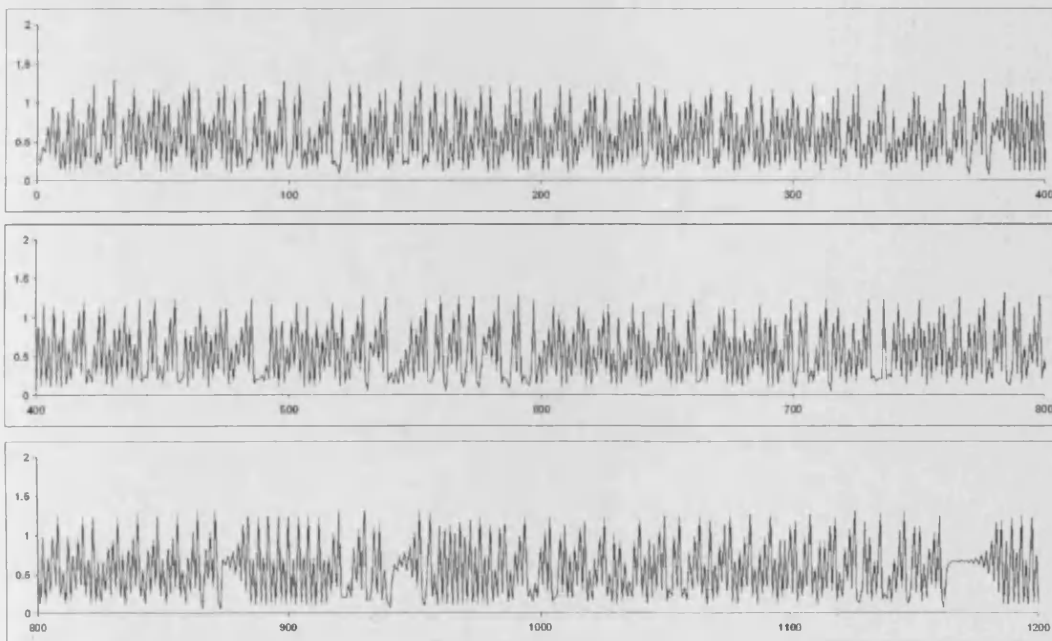


Figure C.7: Rate,  $r^{(k)}$ , as a function of  $k$  for the  $\gamma$ -steepest descent algorithm with  $d = 100$ ,  $\rho = 10$  and  $\gamma = 0.8$ .

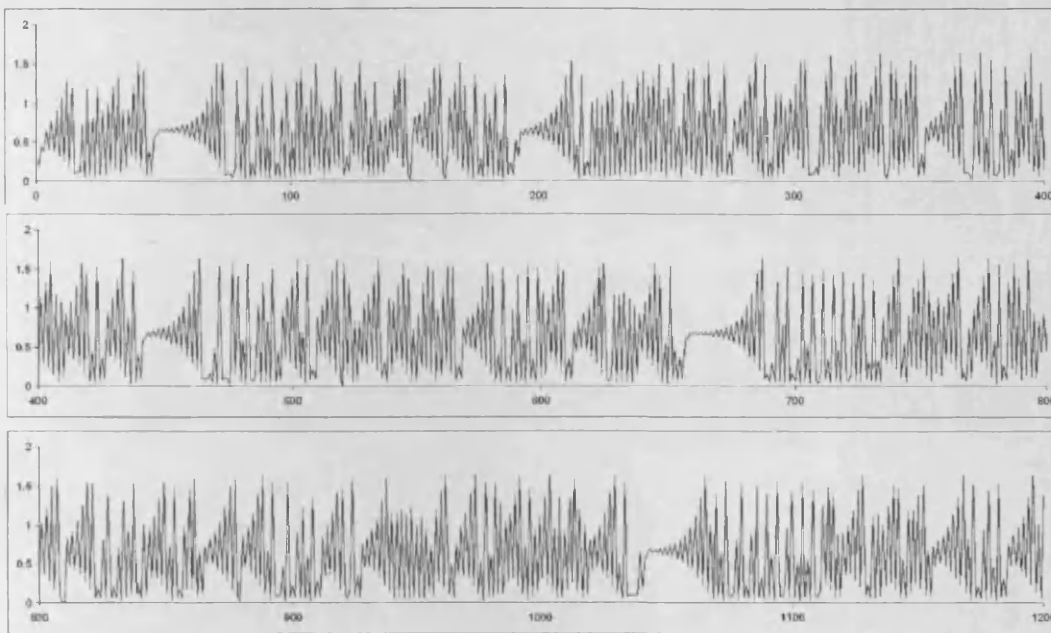


Figure C.8: Rate,  $r^{(k)}$ , as a function of  $k$  for the  $\gamma$ -steepest descent algorithm with  $d = 100$ ,  $\rho = 10$  and  $\gamma = 0.9$ .

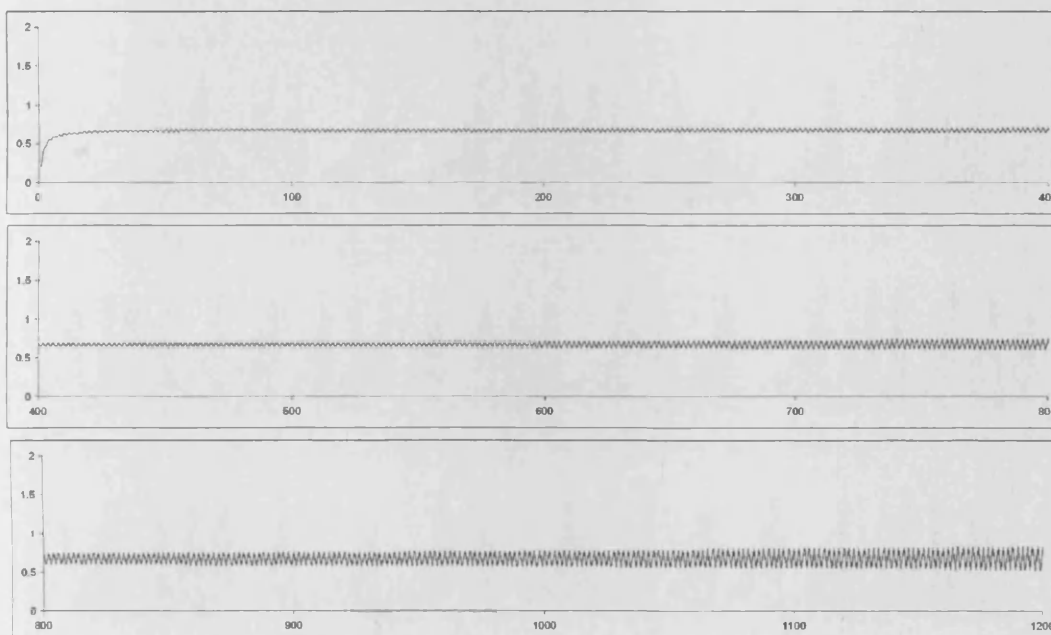


Figure C.9: Rate,  $r^{(k)}$ , as a function of  $k$  for the  $\gamma$ -steepest descent algorithm with  $d = 100$ ,  $\rho = 10$  and  $\gamma = 0.999$ .

### The $\beta$ -Root Algorithm

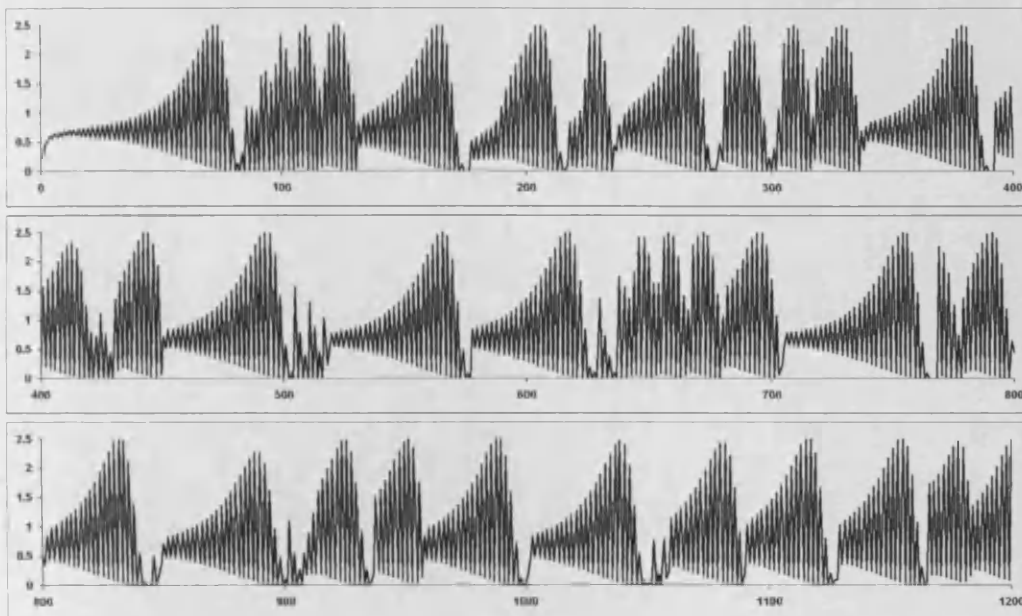


Figure C.10: Rate,  $r^{(k)}$ , as a function of  $k$  for the  $\beta$ -root algorithm with  $d = 100$ ,  $\rho = 10$  and  $\beta = 1.1$ .

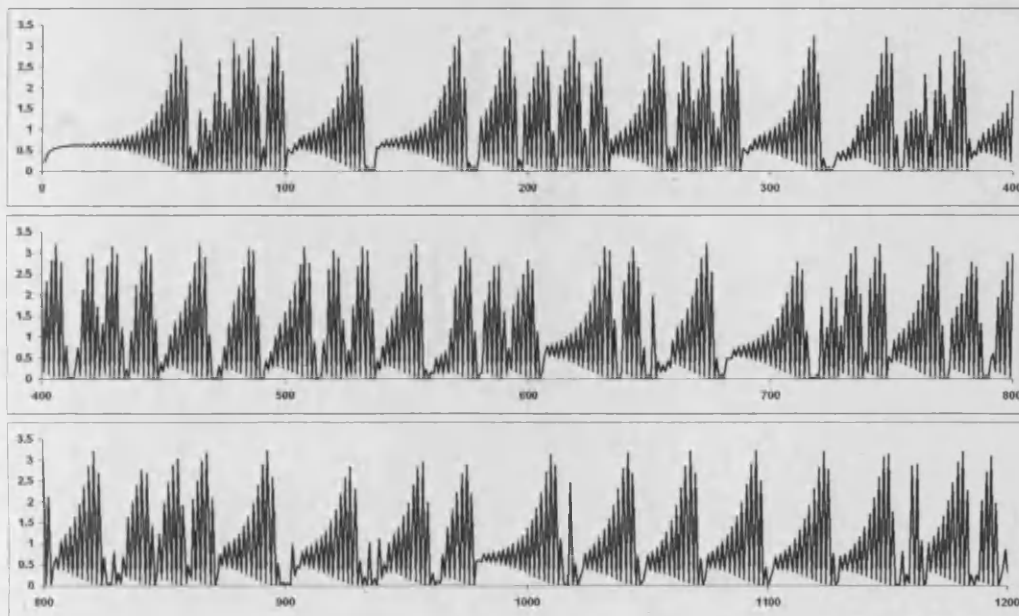


Figure C.11: Rate,  $r^{(k)}$ , as a function of  $k$  for the  $\beta$ -root algorithm with  $d = 100$ ,  $\rho = 10$  and  $\beta = 1.2$ .

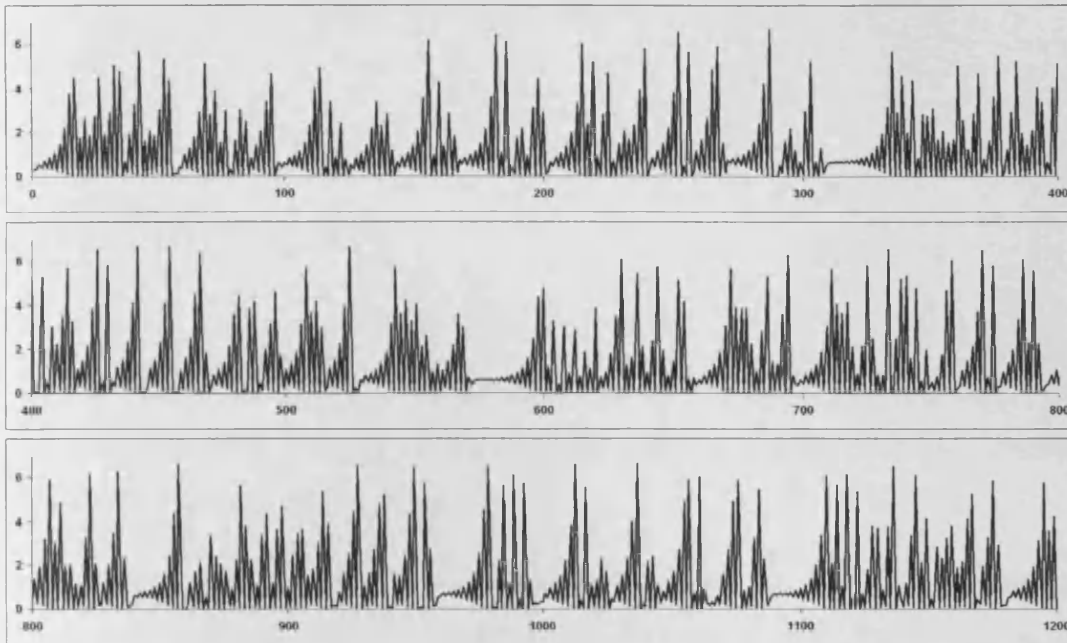


Figure C.12: Rate,  $r^{(k)}$ , as a function of  $k$  for the  $\beta$ -root algorithm with  $d = 100$ ,  $\rho = 10$  and  $\beta = 1.5$ .

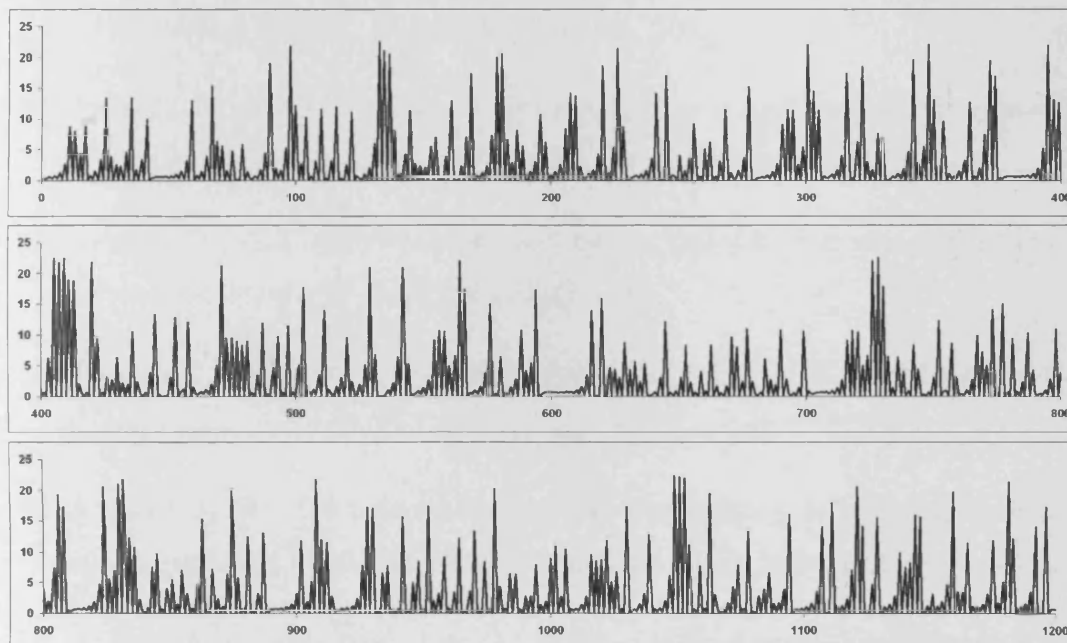


Figure C.13: Rate,  $r^{(k)}$ , as a function of  $k$  for the  $\beta$ -root algorithm with  $d = 100$ ,  $\rho = 10$  and  $\beta = 2$ .

# Bibliography

- [1] AKAIKE, H. On a successive transformation of probability distribution and its application to the analysis of the optimum gradient method. *Ann. Inst. Statist. Math. Tokyo* 11 (1959), 1–16.
- [2] ANDREI, N. A new gradient descent method for unconstrained optimization. Tech. rep., Research Institute for Informatics, 8-10 Averescu Avenue, Bucharest, Romania, 2004.
- [3] ANDREI, N. Relaxed gradient descent and a new gradient descent methods for unconstrained optimization. Tech. rep., Research Institute for Informatics, 8-10 Averescu Avenue, Bucharest, Romania, 2004.
- [4] ANDREI, N. An acceleration of gradient descent algorithm with backtracking for unconstrained optimization. *Numer. Algorithms* 42, 1 (2006), 63–73.
- [5] ARMIJO, L. Minimization of functions having Lipschitz continuous first partial derivatives. *Pacific J. Math.* 16 (1966), 1–3.
- [6] ARNOLDI, W. E. The principle of minimized iteration in the solution of the matrix eigenvalue problem. *Quart. Appl. Math.* 9 (1951), 17–29.
- [7] AXELSSON, O. On preconditioning and convergence acceleration in sparse matrix problems. Tech. rep., CERN, Data Handling Division, Geneva, 1974.
- [8] BARZILAI, J., AND BORWEIN, J. M. Two-point step size gradient methods. *IMA J. Numer. Anal.* 8, 1 (1988), 141–148.
- [9] BOOTH, A. D. *Numerical methods*. Academic Press Inc., New York, 1955.



- 
- [10] BOX, M. J., D. D., AND SWANN, W. H. *Non-Linear Optimization Techniques*. Oliver & Boyd Ltd., Edinburgh, 1969.
- [11] BUNDAY, B. D. *Basic Optimisation Methods*. Edward Arnold Ltd., London, 1984.
- [12] CAUCHY, A. Méthode générale pour la résolution des systèmes d'équations simulatanees. *Comp. Rend. Sci.* 25 (1847), 536–538.
- [13] CHAMBERLAIN, R. M., POWELL, M. J. D., LEMARECHAL, C., AND PEDERSEN, H. C. The watchdog technique for forcing convergence in algorithms for constrained optimization. *Math. Programming Stud.*, 16 (1982), 1–17. Algorithms for constrained minimization of smooth nonlinear functions.
- [14] CURRY, H. B. The method of steepest descent for non-linear minimization problems. *Quart. Appl. Math.* 2 (1944), 258–261.
- [15] DAI, Y., YUAN, J., AND YUAN, Y.-X. Modified two-point stepsize gradient methods for unconstrained optimization. *Comput. Optim. Appl.* 22, 1 (2002), 103–109.
- [16] DAI, Y.-H. Alternate step gradient method. *Optimization* 52, 4-5 (2003), 395–415. Theory, methods and applications of optimization.
- [17] DAI, Y.-H., AND FLETCHER, R. On the asymptotic behaviour of some new gradient methods. *Math. Program.* 103, 3, Ser. A (2005), 541–559.
- [18] DAI, Y.-H., HAGER, W. W., SCHITTKOWSKI, K., AND ZHANG, H. The cyclic Barzilai-Borwein method for unconstrained optimization. *IMA J. Numer. Anal.* 26, 3 (2006), 604–627.
- [19] DAI, Y.-H., AND LIAO, L.-Z. R-linear convergence of the Barzilai and Borwein gradient method. *IMA J. Numer. Anal.* 22, 1 (2002), 1–10.
- [20] DAI, Y. H., AND YANG, X. Q. A new gradient method with an optimal stepsize property. *Comput. Optim. Appl.* 33, 1 (2006), 73–88.

- [21] DAI, Y. H., AND YUAN, Y. Some properties of a new conjugate gradient method. In *Advances in nonlinear programming (Beijing, 1996)*, vol. 14 of *Appl. Optim.* Kluwer Acad. Publ., Dordrecht, 1998, pp. 251–262.
- [22] DAI, Y.-H., AND YUAN, Y.-X. Alternate minimization gradient method. *IMA J. Numer. Anal.* *23*, 3 (2003), 377–393.
- [23] DAI, Y.-H., AND YUAN, Y.-X. Analysis of monotone gradient methods. *J. Ind. Manag. Optim.* *1*, 2 (2005), 181–192.
- [24] DAI, Y.-H., AND ZHANG, H. Adaptive two-point stepsize gradient algorithm. *Numer. Algorithms* *27*, 4 (2001), 377–385.
- [25] DENNIS, JR., J. E., AND MORÉ, J. J. Quasi-Newton methods, motivation and theory. *SIAM Rev.* *19*, 1 (1977), 46–89.
- [26] DODGE, Y., F. V. V., AND WYNN, H. P. Optimal design of experiments: An overview. In *Optimal Design and Analysis of Experiments*. Elsevier Science Publishers B.V., North Holland, 1988, pp. 1–9.
- [27] FADDEEV, D. K., AND FADDEEVA, V. N. *Computational methods of linear algebra*. Translated by Robert C. Williams. W. H. Freeman and Co., San Francisco, 1963.
- [28] FELLMAN, J. On the allocation of linear observations. *Phys. Math.* *44* (1974), 27–78.
- [29] FLETCHER, R. On the Barzilai-Borwein method. Tech. rep., Department of Mathematics, University of Dundee, Department of Mathematics, University of Dundee, Dundee DD1 4HN, October 2001.
- [30] FLETCHER, R., AND POWELL, M. J. D. A rapidly convergent descent method for minimization. *Comput. J.* *6* (1963/1964), 163–168.
- [31] FLETCHER, R., AND REEVES, C. M. Function minimization by conjugate gradients. *Comput. J.* *7* (1964), 149–154.

- [32] FORSYTHE, G. E. On the asymptotic directions of the  $s$ -dimensional optimum gradient method. *Numer. Math.* 11 (1968), 57–76.
- [33] FRIEDLANDER, A., MARTÍNEZ, J. M., MOLINA, B., AND RAYDAN, M. Gradient method with retards and generalizations. *SIAM J. Numer. Anal.* 36, 1 (1999), 275–289 (electronic).
- [34] GOLDSTEIN, A. A. Cauchy’s method of minimization. *Numer. Math.* 4 (1962), 146–150.
- [35] GOLUB, G. H., AND O’LEARY, D. P. Some history of the conjugate gradient and Lanczos algorithms: 1948–1976. *SIAM Rev.* 31, 1 (1989), 50–102.
- [36] GRIPPO, L., LAMPARIELLO, F., AND LUCIDI, S. A nonmonotone line search technique for Newton’s method. *SIAM J. Numer. Anal.* 23, 4 (1986), 707–716.
- [37] GRIPPO, L., AND SCIANDRONE, M. Nonmonotone globalization techniques for the Barzilai-Borwein gradient method. *Comput. Optim. Appl.* 23, 2 (2002), 143–169.
- [38] HESTENES, M. R., AND STIEFEL, E. Methods of conjugate gradients for solving linear systems. *J. Research Nat. Bur. Standards* 49 (1952), 409–436 (1953).
- [39] HIGHAM, N. J. *Accuracy and stability of numerical algorithms*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1996.
- [40] HU, Y.-Q., AND DAI, Y.-H. Inexact Barzilai-Borwein method for saddle point problems. *Numer. Linear Algebra Appl.* 14, 4 (2007), 299–317.
- [41] KANTOROVICH, L. V., AND AKILOV, G. P. *Functional analysis*, second ed. Pergamon Press, Oxford, 1982. Translated from the Russian by Howard L. Silcock.
- [42] LAMOTTE, J.-L., MOLINA, B., AND RAYDAN, M. Smooth and adaptive gradient method with retards. *Math. Comput. Modelling* 36, 9–10 (2002), 1161–1168.

- [43] LANCZOS, C. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *J. Research Nat. Bur. Standards* 45 (1950), 255–282.
- [44] LUENBERGER, D. G. *Linear and Nonlinear Programming*, second ed. Addison-Wesley Publishing Company, US, 1989.
- [45] MANDAL, S., AND TORSNEY, B. Construction of optimal designs using a clustering approach. *J. Statist. Plann. Inference* 136, 3 (2006), 1120–1134.
- [46] MANDAL, S., TORSNEY, B., AND CARRIERE, K. C. Constructing optimal designs with constraints. *J. Statist. Plann. Inference* 128, 2 (2005), 609–621.
- [47] MEURANT, G. *Computer solution of large linear systems*, vol. 28 of *Studies in Mathematics and its Applications*. North-Holland Publishing Co., Amsterdam, 1999.
- [48] MOLINA, B., AND RAYDAN, M. Preconditioned Barzilai-Borwein method for the numerical solution of partial differential equations. *Numer. Algorithms* 13, 1-2 (1996), 45–60.
- [49] NOCEDAL, J., SARTENAER, A., AND ZHU, C. On the behavior of the gradient norm in the steepest descent method. *Comput. Optim. Appl.* 22, 1 (2002), 5–35.
- [50] NOCEDAL, J., S. A., AND ZHU, C. On the accuracy of nonlinear optimization algorithms. Tech. rep., ECE Department, Northwestern University, Evanston IL 60208, November 1998.
- [51] PAIGE, C. C., AND SAUNDERS, M. A. Solutions of sparse indefinite systems of linear equations. *SIAM J. Numer. Anal.* 12, 4 (1975), 617–629.
- [52] POLAK, E., AND RIBIÈRE, G. Note sur la convergence de méthodes de directions conjuguées. *Rev. Française Informat. Recherche Opérationnelle* 3, 16 (1969), 35–43.

- [53] PRONZATO, L., WYNN, H. P., AND ZHIGLJAVSKY, A. A. *Dynamical search*. Chapman & Hall/CRC, Boca Raton, FL, 2000. Applications of dynamical systems in search and optimization, Interdisciplinary statistics.
- [54] PRONZATO, L., WYNN, H. P., AND ZHIGLJAVSKY, A. A. Renormalised steepest descent in Hilbert space converges to a two-point attractor. *Acta Appl. Math.* 67, 1 (2001), 1–18.
- [55] PRONZATO, L., WYNN, H. P., AND ZHIGLJAVSKY, A. A. Asymptotic behaviour of a family of gradient algorithms in  $\mathbb{R}^d$  and Hilbert spaces. *Math. Program.* 107, 3, Ser. A (2006), 409–438.
- [56] PRONZATO, L., WYNN, H. P., AND ZHIGLJAVSKY, A. A. Asymptotic behaviour of optimum s-gradient algorithm ish. In *W-optimality*. Springer, To appear.
- [57] RAYDAN, M. On the Barzilai and Borwein choice of steplength for the gradient method. *IMA J. Numer. Anal.* 13, 3 (1993), 321–326.
- [58] RAYDAN, M., AND SVAITER, B. F. Relaxed steepest descent and Cauchy-Barzilai-Borwein method. *Comput. Optim. Appl.* 21, 2 (2002), 155–167.
- [59] SAAD, Y., AND SCHULTZ, M. H. GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Statist. Comput.* 7, 3 (1986), 856–869.
- [60] SHEWCHUK, J. An introduction to the conjugate gradient method without the agonizing pain. Tech. rep., Carnegie Mellon University, Pittsburgh, PA 15213, August 1984.
- [61] SILVEY, S. D., T. D. M., AND TORSNEY, B. An algorithm for optimal designs on a finite design space. *Communications in Statistics A* 7 (1978), 1379–1389.
- [62] ST. JOHN, R. C., AND DRAPER, N. R. *D*-optimality for regression designs: a review. *Technometrics* 17 (1975), 15–23.

- 
- [63] TITTERINGTON, D. M. Algorithms for computing  $d$ -optimal designs on a finite design space. In *Proc. 1976 Conference on Information Sciences and Systems*. 1976.
- [64] TORSNEY, B. A moment inequality and monotonicity of an algorithm. In *Lecture Notes in Economics and Mathematical Systems*, A. F. . K. K. Eds, Ed., vol. 215. Springer Verlag, 1983, pp. 249–260.
- [65] TORSNEY, B. Computing optimizing distributions with applications in design, estimation and image processing. In *Optimal Design and Analysis of Experiments*, H. W. E. Y. Dodge, V.V. Fedorov, Ed. North Holland, 1988, pp. 316–370.
- [66] TORSNEY, B. Fitting bradley terry models using a multiplicative algorithm. In *Proceedings in Computational Statistics*, J. Antoch, Ed. Physica Verlag, Prague, 2004, pp. 214–226.
- [67] TORSNEY, B., AND ALAHMADI, A. Further developments of algorithms for constructing optimizing distributions. In *Model Orientated Data Analysis*, I. V. E. V.V. Fedorov, W.G. Muller, Ed. Physica Verlag, Bulgaria, 1992, pp. 121–129.
- [68] TORSNEY, B., AND MANDAL, S. Multiplicative algorithms for constructing optimizing distributions: further developments. In *mODa 7—Advances in model-oriented design and analysis*, Contrib. Statist. Physica, Heidelberg, 2004, pp. 163–171.
- [69] WALSH, G. R. *Methods of Optimization*. John Wiley & Sons Ltd., 1975.
- [70] WOLFE, P. Convergence conditions for ascent methods. *SIAM Rev.* 11 (1969), 226–235.
- [71] YOUNG, D. M. *Iterative solution of large linear systems*. Academic Press, New York, 1971.

- [72] YOUNG, D. M. A historical overview of iterative methods. *Comput. Phys. Comm.* 53, 1-3 (1989), 1–17. Practical iterative methods for large scale computations (Minneapolis, MN, 1988).
- [73] YUAN, Y.-X. A new stepsize for the steepest descent method. Tech. rep., Chinese Academy of Sciences, 2004.
- [74] ZHOU, B., GAO, L., AND DAI, Y.-H. Gradient methods with adaptive step-sizes. *Comput. Optim. Appl.* 35, 1 (2006), 69–86.

