Non-Linear Semi-Infinite Programming.

A thesis
submitted in partial fulfillment
of the requirements for
the degree of
Doctor of Philosophy in Mathematics
at the
University of Canterbury,
by C. J. Price.

August, 1992.
University of Canterbury,
Christchurch,
New Zealand.

Contents

1 INTRODUCTION AND OVERVIEW OF THE TOPIC. 1

1.1 The Semi-Infinite Programming Problem. 1

1.2 Characterizing SIP solutions. 3

1.3 Locally Convergent Methods 7

1.4 Discretization methods. 9

1.5 Multi-Phase Methods. 13

1.6 Exact Penalty Function Methods. 15

1.6.1 SOLVER-like Methods. 17

1.6.2 Trust Region Methods. 20

1.7 Remarks. 22

2 THE THEORETICAL BASIS FOR THE ALGORITHM. 25

2.1 The penalty function problem. 26

2.2 Existence of an approximating L_∞QP. 28

2.3 An L_∞-norm algorithm for SIP. 32

2.4 Convergence. 34

2.5 Concluding Remarks. 37

3 MULTI—LOCAL OPTIMISATION. 39

3.1 Introduction. 39

3.2 Outline of the MOS Algorithm. 42

3.3 Halton sequences. 43

3.4 The Stochastic Process. 46

3.4.1 Links and Clusters. 56

3.4.2 Estimation of c. 57

3.5 Convergence. 61

3.6 Other Remarks. 63

4 IMPLEMENTATION OF THE ALGORITHM. 67

4.1 Implementation of the SIP algorithm. 67

4.2 The Multi-Dimensional MOS Sub-algorithm. 73

4.2.1 Choosing the maximum link length. 78

4.2.2 Choosing the prime and increment lists for the Halton sequence. 80

4.3 The One Dimensional Multi-Local Optimisation Algorithm. 80

5 NUMERICAL RESULTS. 83

5.1 One Dimensional Problems. 84

5.2 Two Dimensional Problems. 88

5.3 Other Results. 92

5.4 Comparison of the Various Algorithms. 99

5.5 Higher Dimensional Problems. 103

5.6 Using a NLP First Phase. 111

5.7 Advantages of an Extra Penalty Parameter. 114

5.8 Other Comments. 120

6 DISCUSSION AND CONCLUSION. 123

List of Tables

5.1 *A comparison of results with those obtained by Tanaka et al.* 100
5.2 *A comparison of results with those obtained by Watson, and those obtained by Coope and Watson.* 101
5.3 *A comparison of results for the extended version of problem 4 with those obtained by Bell.* 104
5.4 *A comparison of the results for the extended form of problem 5 with those obtained by Tanaka et al., and those obtained by Coope and Watson.* 104
5.5 *Results for the higher T dimensional problems.* 105
5.6 *Results for a two phase algorithm on problem S with $p = 4$. Here the number of constraints in the first phase has been fixed at 160, and the accuracy to which the NLP was solved has been varied.* 112
5.7 *Results for a two phase algorithm on problem S with $p = 4$. Here the number of constraints in the NLP has been varied, and each NLP was solved to an accuracy of 1.0E-5.* 112
5.8 *Variations of the algorithms performance on problem 6 with respect to changes in $\theta_{crossover}$. When θ exceeds $\theta_{crossover}$, ν is altered, otherwise μ is altered. The first and last rows are for a single parameter exact non-differentiable penalty function ($\nu \equiv 0$), and for a quadratic penalty function ($\mu \equiv 0$) respectively.* 117
5.9 *The hybrid PF, and the SPEPF with unlimited increases in the penalty parameters permitted.* 118
5.10 *The hybrid PF, and the SPEPF with unlimited increases, and with decreases in the penalty parameters permitted.* 118
5.11 *Results for problem K with various values of the penalty parameters. Both penalty parameters were fixed during each run.* 119

Acknowledgements.

Firstly, I would like to thank the University Grants Committee of New Zealand for financial support in the form of a UGC postgraduate scholarship, without which this thesis would not have been possible. I would also like to acknowledge financial support from the Mathematics Department of the University of Canterbury in the form of a part-time tutor's position. I also wish to thank the New Zealand Mathematics Society, and the Royal Society of New Zealand for assistance in attending a conference.

On the academic side of things, I would firstly and foremostly like to thank my supervisor, Dr. Ian Coope, for his assistance, guidance, and wisdom. I would like to thank Prof. M. J. D. Powell for many helpful comments on an early version of chapter 2, and in particular for furnishing a simple proof of proposition 2.6, which appears herein. I would also like to thank the numerous friends, colleagues and family for their support, and for expressing a surprising degree of interest in what, to many of them, must have been an obscure and abstract subject.

Finally, I would like to thank my father for proof-reading the English in much of this thesis. In doing so, he came up with the following little gem:

> "Reading the 19 pages reminded me of driving 'battle wagons,' vehicles without headlights, through road tunnels in Italy. Stroll in at 40 mph and come out the other end at 4 mph, with a 'what the hell' expression on the face."

A List of the Symbols used throughout the Thesis.

The following is an incomplete list of symbols used in the thesis. The symbols which have been omitted are those which are defined and used only in the space of a few pages.

LATIN LETTERS.

H	The matrix containing (approximated) curvature information appearing in each L_∞QP. (Ch. 1) the Hessian of the Lagrangian.
I_n	The $n \times n$ identity matrix.
N	The number of test points generated by the MOS.
S_b	The upper bound imposed on the magnitude of each element of s.
T	The semi-infinite constraint's index set.
$b_j(t)$	(Ch. 1) The finite set of constraints $\{b_j(t) \leq 0\}$ defines the set T.
c	The variance parameter for the Brownian Motion Process.
$\overline{c}$	The estimate of the variance parameter for the Brownian Motion Process.
$c^{(k)}$	The Maratos effect correction vector (or second order correction).
$c(x)$	This symbol is used (mostly in Ch. 1) to denote an ordinary constraint.
f	The objective function of the semi-infinite programme.
g	The semi-infinite constraint function.
k	The SIP iteration number. In chapter 5 k is used to denote the number of SIP iterations performed in solving a SIP.
l_i	The (simple) lower bound on the i^{th} component of x.
n	The dimension of x.
p	The dimension of t.
$q^{(k)}(\alpha)$	$= \alpha s^{(k)} + \alpha^2 c^{(k)}$. The quadratic arc along which a search for the next iterate is conducted.
s	The proposed step (or search direction) from the current iterate.
t	The vector of index variables of the semi-infinite constraint.
u_i	The (simple) upper bound on the i^{th} component of x.
x	The vector of variables with respect to which the objective function $f(x)$ is minimized.
x_i	The i^{th} element of x.

Greek Letters.

Δ	Governs the size of the trust region.
$\Gamma(x)$	The set of global maximisers of $g(x,t)$ with respect to t, where $t \in T$ is required.
Ξ	(Ch. 2) $\Xi(x)$ is used to denote the set of extensions of a local maximiser of $g(x_0,.)$ evaluated at some point x.
Φ'	The magnitude of the most negative directional derivative of the L_1 exact penalty function at the final iterate $x^\sharp$.
α	The variable used to conduct the line (or arc) search.
β	The ratio of successive trial values of α used in the arc search.
δ_i	The i^{th} increment used in generating the Halton sequence.
$\zeta(s)$	An approximation to $\theta(x+s)$ formed using linearizations of $g(x,t)$ with respect to x for each t in $\mathcal{A}$. At iteration k, $\zeta^{(k)}(s) = \max\{[g(x^{(k)},t) + s^T\nabla_x g(x^{(k)},t)]_+ : t \in \mathcal{A}^{(k)}\}$.
θ	$\theta(x)$ is the maximum value the semi-infinite constraint function g takes on T, or 0, whichever is larger: $\theta(x) = \max_{t\in T}[g(x,t)]_+$.
θ_{cap}	When θ exceeds θ_{cap}, an extra constraint is imposed on the L_∞QP. The effect of this extra constraint is to ensure that the search direction s is not one of ascent for θ. Additionally, in the corresponding arc search, any α value which produces an increase in θ is automatically rejected.
$\theta_{\text{crossover}}$	When θ exceeds $\theta_{\text{crossover}}$, any increases in the penalty parameters are made to ν. Otherwise they are made to μ.
λ	The vector of Lagrange multipliers for the SIP.
λ^*_{est}	Estimates of the SIP's optimal Lagrange multipliers.
μ	Penalty parameter.
μ_{min}	(Ch. 5) When decreases in the penalty parameters are permitted, this variable is the lower limit for μ.
ν	Penalty parameter.
ν_{min}	(Ch. 5) When decreases in the penalty parameters are permitted, this variable is the lower limit for ν.
ξ	(Ch. 1) A Lagrange multiplier for the problem of finding a local maximiser of $g(x,t)$ with respect to t.
π_i	The i^{th} prime used in generating the Halton sequence.

ρ	This parameter is used in the arc search. It is the minimum acceptable ratio of the actual descent to that predicted by the L_∞QP.
τ	A point in T which is a local, and sometimes a global maximiser of g.
ϕ	The exact non-differentiable penalty function: $\phi(\mu,\nu;x) = f(x) + \mu\theta(x) + \frac{1}{2}\nu\theta^2(x)$.
ψ	The strictly convex piecewise quadratic local approximation to ϕ in the neighbourhood of the current iterate. $\psi(x^{(k)},\mathcal{A}^{(k)};\mu,\nu;s) = f(x^{(k)}) + s^T\nabla f(x^{(k)}) + \frac{1}{2}s^THs + \mu\zeta(s) + \frac{1}{2}\nu\zeta^2(s)$.

Calligraphic Letters and Other Symbols.

$\mathcal{A}$	The (finite) subset of T from which the approximation $\zeta(s)$ to the maximum constraint violation $\theta(x^{(k)}+s)$ is formed.
$\mathcal{A}_{\mathrm{soc}}$	The (finite) subset of T found by the MOS subalgorithm. $\mathcal{A}^{(k)}_{\mathrm{soc}}$ is used in calculating the second order correction $c^{(k)}$.
$\mathcal{H}$	The Halton sequence.
$\mathcal{H}_m$	The first m points of the Halton sequence.
$\mathcal{Q}^{(k)}$	The subset of $\mathcal{A}^{(k)}$ which gives rise to the L_∞QP constraints which are active at the solution of the L_∞QP formed in the k^{th} iteration.
$\mathcal{T}$	The sequence of test points generated by the MOS.
$\mathcal{T}_N$	The first N points of the sequence $\mathcal{T}$.
$\emptyset$	The empty set.
$\ell_{\max}$	The maximum length, measured using the ∞-norm, of a link between two test points.
$\wp(i)$	The linkage parameter of the link upwards from test point i.
$\wp_{\max}$	The maximum value $\wp(i)$ may take.
$\clubsuit$	Denotes the end of a proof, definition, or assumption.

Superscripts and Subscripts.

$?^{(0)}$	The initial value of ?.
$?^{(k)}$	The value ? takes at the iterate $x^{(k)}$.
$?^*$	The value the variable ? takes at the point x^*, where x^* is a

	solution point of the SIP.
$?^\sharp$	The value the variable ? takes at the point $x^\sharp$, where $x^\sharp$ is the final iterate generated by the quasi-Newton algorithm.

Acronyms.

SIP	Semi-Infinite Programme.
PFP	Penalty Function Problem.
NLP	Non-Linear Programme.
MOS	Multi-local Optimisation Subproblem.
QP	Quadratic Programme.
SQP	Sequential Quadratic Programming.
KKT	Karush Kuhn Tucker (ie. the first order optimality conditions).
L_∞QP	The problem of minimising the sum of a quadratic and the infinity norm of the violations of a finite number of linear constraints.
BMP	Brownian Motion Process.
IBMP	Integrated Brownian Motion Process.
SPEPF	(Ch. 5) Single Parameter Exact Penalty Function.
QPF	(Ch. 5) Quadratic Penalty Function.

Chapter 1

INTRODUCTION AND OVERVIEW OF THE TOPIC.

Optimisation problems occur in many branches of science, engineering, and economics, as well as in other areas. The diversity of the various types of optimisation problems is extremely large, and so a unified approach is not attempted here. This thesis concentrates on a specific type of problem: non-linear semi-infinite programming.

1.1 The Semi-Infinite Programming Problem.

The specific Semi Infinite Programming problem (SIP) considered herein is of the following form:

$$\min_{x \in R^n} f(x) \tag{1.1}$$

$$\text{subject to } \; g(x,t) \leq 0 \;\; \forall t \in T, \;\; \text{where } \; T \subset R^p. \tag{1.2}$$

The semi-infinite programming problem can be viewed as a generalisation of the finite Non-Linear Programming problem (NLP):

$$\min_{x \in R^n} f(x) \quad \text{subject to} \quad c_i(x) \leq 0, \;\; \forall i = 1, \ldots, m.$$

The variable t corresponds to the index variable i, and the set T corresponds to the index set $\{1, \ldots, m\}$. The set T typically contains an infinite number of points. For example, T could be an interval such as $[0, 1]$. With this choice of T the constraint (1.2) can be viewed as an infinite set of ordinary constraints indexed by the variable t. It is from this infinite number of constraints that the 'infinite' half of the name 'semi-infinite' arises. The 'semi' refers to the fact that x is finite dimensional. Optimisation problems which involve minimising over an infinite dimensional set subject to an infinite number of constraints are known as Infinite Programming Problems.

The SIP listed in (1.1) and (1.2) is by no means the most general form of semi-infinite programming problem. A general SIP may involve several constraints of the form (1.2) as well as a finite number of auxillary constraints each of the form $c(x) \leq 0$. Such SIP problems are in principle no more difficult than the SIP (1.1,1.2). For convenience attention will be restricted to problems of the form (1.1,1.2).

SIPs arise in many practical applications. One well known example in the literature is the air pollution problem [60, 42, 55]. Other authors [9, 77] report semi-infinite programming problems arising in control system design, electronic circuit design, and the like.

A linear SIP problem is of the form (1.1,1.2) except that $f(x)$ and $g(x,t)$ are affine functions of x. In general $g(x,t)$ is not required to be an affine function of t. These problems have been investigated rather more than non-linear SIP problems. Good review papers on linear SIP include those by Hettich [51, 52] and the rather more theoretical one by Gustafson and Kortanek [42]. The texts by Glashoff and Gustafson [37] and by Krabs [60] are also to be recommended. The topic of this thesis is non-linear, and non-convex SIP, and so the methods for linear SIP are only of passing interest, except where they may be adapted to solve non-linear SIP problems.

Hereafter, the following restrictions will be made on the SIP (1.1,1.2). The objective function $f(x)$, mapping R^n into R, and the constraint function $g(x,t)$, mapping $R^n \times T$ into R, are both continuously differentiable in all arguments. No assumptions concerning linearity or convexity are made for f or g. The set T is compact and connected. It is assumed that T is defined by a finite number of continuously differentiable constraints

$$b_j(t) \leq 0 \ \ \forall j = 1, \ldots, k_{\mathrm{b}}. \tag{1.3}$$

As it will be necessary to find maximisers of g with respect to t, it is assumed that the constraints (1.3) satisfy an appropriate constraint qualification. Frequently T is a Cartesian product of intervals.

Unless specifically stated otherwise, the term local maximiser refers to a local maximiser of $g(x,t)$ with respect to t, where x is fixed at some value which will be clear from the context. Similarly, the term global maximiser refers to a global maximiser of $g(x,t)$.

1.2 Characterizing SIP solutions.

Before discussing methods of solving SIPs, a workable definition of precisely what constitutes a solution is needed. As for finite Non-Linear Programmes (NLP), solutions are characterised as points at which the pertinent Karush-Kuhn-Tucker (KKT) conditions hold. The first order necessary KKT conditions for the SIP are listed in (1.5) and in (1.6) in the following theorem.

Theorem 1.1 *Let x^* be any optimal point of the SIP, and let the following regularity assumption hold at x^*:*

$$\exists u \in R^n \ \ \textit{such that} \ \ g(x^*,t) + u^T\nabla_x g(x^*,t) < 0, \ \ \forall t \in T. \tag{1.4}$$

Then for some $m^* \leq n$, there exists m global maximisers τ_i^* of $g(x^*,t)$, each with an associated Lagrange multiplier λ_i^*, for which

$$\nabla f(x^*) + \sum_{i=1}^{m^*} \lambda_i^* \nabla_x g(x^*,\tau_i^*) = 0 \tag{1.5}$$

$$\textit{where} \ \ g(x^*,\tau_i^*) = 0, \ \ \textit{and} \ \ \lambda_i^* \geq 0, \ \forall i = 1,\ldots,m^*. \tag{1.6}$$

PROOF: By lemmas 2 and 3 of Gustafson [41]. For completeness, a proof based on the Lebesgue integral rather than the Riemann integral (as in [41]) is given here.

If $\nabla f^* = 0$, then $m^* = 0$, and the theorem is obvious. Whenceforth assume $\nabla f^* \neq 0$.

Let $C(T)$ denote the space of real continuous functions on T. If L is a linear functional on $C(T)$, and is continuous with respect to the infinity norm, then L may be written as

$$L(h) = \int_T h(t)\, d\mu, \ \ \text{where} \ \ h(t) \in C(T),$$

and where μ is a bounded measure on T. In particular if L is a positive linear functional, then μ is a non-negative measure. Regularity assumption (1.4), and theorem 1 on page 249 of Luenberger [61] imply that the first order necessary conditions for optimality are:

$$\nabla f^* + \int_T \nabla_x g(x^*,t)\, d\mu^* = 0 \tag{1.7}$$

$$\text{and} \quad \int_T g(x^*,t)\, d\mu^* = 0. \tag{1.8}$$

Here μ^* is a bounded non-negative measure.

Select some fixed vector u which satisfies the regularity assumption (1.4). Equations (1.7) and (1.8) imply

$$-u^T\nabla f^* = \int_T g(x^*,t) + u^T\nabla_x g(x^*,t)\, d\mu^*. \tag{1.9}$$

Clearly $u^T\nabla f^* \geq 0$, by the regularity assumption (1.4). In light of (1.4), the compactness of T and the C^1 continuity of g imply the integrand in (1.9) achieves its supremum, and it is negative. Hence, if $u^T\nabla f^* = 0$, the non-negativity of μ^* implies that T is a set of μ^* measure zero. Equation (1.7) then implies $\nabla f^* = 0$. As this possibility has already been dealt with, henceforth assume $u^T\nabla f^* > 0$.

Let $\mathcal{K}$ denote the set of all vectors ψ of the form

$$\psi = \int_T \nabla_x g(x^*,t)\, d\mu \tag{1.10}$$

$$\text{where} \quad u^T\psi = -u^T\nabla f^*,$$

$$\text{where} \quad \int_T g(x^*,t)\, d\mu = 0, \tag{1.11}$$

and where μ is any (bounded) non-negative measure. Clearly $\mathcal{K}$ is a convex set, and $-\nabla f^* \in \mathcal{K}$.

Define $T_0 = \{t \in T : g(x^*,t) = 0\}$. As $g(x^*,t) \leq 0$ for all $t \in T$, (1.11) implies $T - T_0$ is a set of μ measure zero. Hence, without loss of generality, the integrals in (1.7,1.8,1.10) and in (1.11) may be taken over T_0 rather than T. The continuity of g, and the compactness of T imply T_0 is also compact. The compactness of T_0 implies $u^T\nabla_x g(x^*,t)$ achieves its supremum on T_0, which must be strictly negative by (1.4). Therefore the set

$$\mathcal{G} = \left\{ -\nabla_x g(x^*,t) \frac{u^T\nabla f^*}{u^T\nabla_x g(x^*,t)} : t \in T_0 \right\}$$

is also compact, as g is a C^1 function.

Now, because $\nabla_x g(x^*,t)$ is continuous, any member of $\mathcal{K}$ may be approximated arbitrarily closely by convex combinations of elements of $\mathcal{G}$. Let $\text{co}(\mathcal{G})$ denote the convex hull of $\mathcal{G}$. The compactness of $\mathcal{G}$ implies $\text{co}(\mathcal{G})$ is also compact (see, eg., [95]). Hence $\mathcal{K} \subseteq \text{co}(\mathcal{G})$.

Also $\text{co}(\mathcal{G})$ is a subset of the hyperplane

$$\mathcal{H} = \{h \in R^n : u^T h = -u^T\nabla f^*\}.$$

As $\mathcal{H}$ is of dimension $n-1$, and as $-\nabla f^* \in \mathcal{K}$, Caratheodory's theorem implies μ^* may be chosen as a positive finite point measure which is non-zero at not more than n points in T. Denoting the finite set of points at which μ^* is non-zero by $\{\tau_i^* : i = 1, \ldots, m^*\}$, and denoting the corresponding weights by $\{\lambda_i^* : i = 1, \ldots, m^*\}$ yields the required result. ♣

The regularity assumption (1.4) is not the only constraint qualification under which the first order KKT conditions can be derived. In particular, if there are a number of auxillary equality constraints, then a different constraint qualification than (1.4) must be used.

The form the first order optimality conditions take is important in that only a *finite* number of points in T are required. This admits numerous solution methods based on replacing the semi-infinite constraint by a finite set of ordinary constraints. This is achieved by replacing T with a finite subset of T.

Actually (1.5) and (1.6) are identical to the first order KKT conditions for the NLP:

$$\min_{x \in R^n} f(x) \quad \text{subject to} \quad g(x, \tau_i^*) \leq 0 \quad \forall i = 1, \ldots, m^*.$$

Under rather stricter assumptions than (1.4) alone, the SIP is locally equivalent to an NLP. It can be shown that at any fixed point x_0 which satisfies these assumptions, each local maximiser τ_i of $g(x_0, t)$ gives rise to a continuous function $\tau_i(x)$ on some neighbourhood $\mathcal{N}_0$ of x_0. For all $x \in \mathcal{N}_0$, each $\tau_i(x)$ is a local maximiser of $g(x,t)$. Given the number of local maximisers of $g(x_0, .)$ is finite (there are, say, m_0 of them), and given that perturbing x_0 does not change any stationary point t_s of $g(x_0, t)$ into a local maximiser at the perturbed value of x, then under these conditions the SIP is locally equivalent to:

$$\min_{x \in R^n} f(x) \quad \text{subject to} \tag{1.12}$$

$$c_i(x) = g(x, \tau_i(x)) \leq 0 \quad i = 1, \ldots, m_0. \tag{1.13}$$

The condition that a stationary point t_s of $g(x_0, t)$ can not be changed into a local maximiser by perturbing x can be stated more precisely as:

$$\exists \epsilon, \delta > 0 \text{ such that } \forall x,$$

$$\|x - x_0\| < \delta \Rightarrow \text{ every local maximiser } t \text{ of } g(x, .) \text{ satisfies } \|t_s - t\| \geq \epsilon.$$

Sufficient conditions for a local maximiser τ_i of $g(x_0, .)$ to give rise to a continuous function $\tau_i(x)$ for x near x_0 are given in the following theorem.

Theorem 1.2 *Given:*

1. *g is twice continuously differentiable.*

2. *t_0 is a local maximiser of $g(x_0, .)$, and t_0 lies on j_0 of the bounding constraints of T, where $j_0 \leq p$. Without loss of generality, let these active constraints be $\{b_i(t) \leq 0 : i = 1, \ldots, j_0\}$.*

3. *The constraint normals $\{\nabla b_i(t_0) : i = 1, \ldots, j_0\}$ are linearly independent.*

4. *Strict complementarity holds at t_0.*

5. *$\nabla^2_{tt} g(x_0, t_0)$ is negative definite on the subspace $\mathcal{M}$, where $\mathcal{M}$ is the subspace of R^p orthogonal to span$\{\nabla b_i(t_0) : i = 1, \ldots, j_0\}$.*

Then on some open neighbourhood $\mathcal{N}_0$ of x_0, there exists a continuous function $t(x)$, with $t(x_0) = t_0$, such that $t(x)$ is a local maximiser of $g(x, .)$ lying on the bounding constraints $\{b_i(t) = 0 : i = 1, \ldots, j_0\}$ for all $x \in \mathcal{N}_0$.

Proof. The set of constraints (1.3) defining T has been assumed to satisfy a constraint qualification which ensures stationary points of g satisfy the first order Karush-Kuhn-Tucker (KKT) conditions:

$$\nabla_t g(x_0, t_0) + \sum_{i=1}^{j_0} \xi_i \nabla b_i(t_0) = 0,$$

$$b_i(t_0) = 0 \text{ and } \xi_i \geq 0 \ \forall i = 1, \ldots, j_0.$$

Here the ξ_i are the Lagrange multipliers. The Jacobian of this system of equations is continuous with respect to x. If it is also non-singular then the result follows immediately from the implicit function theorem. The Jacobian $J(x_0, t_0)$ at x_0 is

$$\begin{pmatrix} \nabla^2_{tt} g(x_0, t_0) & B \\ B^T & 0 \end{pmatrix} \text{ where } B = \begin{pmatrix} \nabla b_1(t_0) & \nabla b_2(t_0) & \ldots & \nabla b_{j_0}(t_0) \end{pmatrix}.$$

To show $J(x_0, t_0)$ is non-singular, consider

$$J(x_0) \begin{pmatrix} t \\ \xi \end{pmatrix} = 0.$$

The bottom j_0 rows imply $t \in \mathcal{M}$. Hence $t^T B \xi = 0$, and so the top p rows imply $t^T \nabla_{tt} g(x_0, t_0) t = 0$. However $\nabla_{tt} g(x_0, t_0)$ is negative definite on $\mathcal{M}$, and so $t = 0$.

As B is of full rank, $\xi = 0$. As $J(x_0, t_0)$ is continuous in all arguments, there exists a continuous function $t(x)$ satisfying the above requirements. ♣

In light of this theorem, it can be shown (eg. [97, 79]) that all functions in the NLP (1.12,1.13) are C^2. The first and second derivatives of $c_i(x)$ being respectively

$$\nabla_x c(x) = \nabla_x g(x, \tau_i)$$

and

$$\nabla^2_{xx} c(x) = \nabla^2_{xx} g(x, \tau_i) - \nabla^2_{xt} g(x, \tau_i) \left(\nabla_{tt} g(x, \tau_i)\right)^{-1} \nabla^2_{tx} g(x, \tau_i). \tag{1.14}$$

If the second order sufficiency conditions do not hold for some $\tau_i(x)$ at x_0 but the set of local maximisers $\{\tau_i(x)\}_{i=1}^{m_0}$ still contains the local maximisers of $g(x, .)\ \forall x \in \mathcal{N}_0$, then the local equivalence to an NLP still holds. However, in general the NLP is now only C^1 (this follows from the results developed in chapter 2). If $\nabla^2_{tt} g$ is singular at x_0, then its inverse, which appears in (1.14), is undefined. If strict complementarity does not hold for some τ_i, then the rank of the last term in (1.14) can change, which may lead to a discontinuous change in the second derivative of c_i at x_0.

If extra local maximisers appear out of the stationary points of $g(x_0, .)$, or if a local maximiser of g splits into several local maximisers at x_0, then a locally equivalent NLP can usually still be constructed in a similar fashion. The exceptions to this are quite unusual. In any case the analysis becomes more complex without any real gain.

One could solve the SIP by constructing a locally equivalent NLP about each iterate. Applying one iteration of a standard NLP method to any such NLP would then yield a prospective step for the SIP. A more fruitful approach is to bypass the NLP altogether, and construct an approximating quadratic programme (QP) directly. This avoids the difficulties caused by the appearances of extra local maximisers. It forms the main thrust of chapter 2.

1.3 Locally Convergent Methods

For a finite NLP a locally convergent superlinear method is obtained by applying Newton's method to the first order KKT conditions. A similar approach to SIP problems can be taken [43, 54, 99], although the system of equations to which Newton's method must be applied is much larger: it consists of the first order KKT conditions for the SIP together with the first order KKT conditions for each global maximiser τ_i^*. This system is

$$\nabla f(x) + \sum_{i=1}^{m^*} \lambda_i \nabla_x g(x, \tau_i) = 0 \tag{1.15}$$

$$g(x, \tau_i) = 0 \ \ \forall i = 1, \ldots, m^* \tag{1.16}$$

$$\nabla_t g(x, \tau_i) + \sum_{j \in \mathcal{B}_T(i)} \xi_{ij} \nabla b_j(\tau_i) = 0 \ \ \forall i = 1, \ldots, m^* \tag{1.17}$$

$$b_j(\tau_i) = 0 \ \ \forall i = 1, \ldots, m^* \text{ and } \forall j \in \mathcal{B}_T(i) \tag{1.18}$$

where $\mathcal{B}_T(i)$ is the set of index values of the bounding constraints (1.3) of T on which τ_i^* lies. The ξ_{ij} variables are the Lagrange multipliers for the constraints $b_j(t) \leq 0$ which are active at the corresponding global maximiser τ_i^*. Strict complementarity for the SIP at x^* and for the local maximisation problem at each τ_i^* is assumed. Second order sufficiency conditions (including linearly independent active constraint normals) are assumed to hold for x^*, and also for each global maximiser τ_i^*. These conditions ensure that the Newton step for the system of equations (1.15,1.16,1.17,1.18) is well-defined for x, λ, ξ_{ij}, and τ_i sufficiently close to their optimal values. The special structure of the system permits considerable reduction in the computational effort required for each iteration over that needed if no structure were present [98].

An alternative approach to using Newton's method on (1.15,1.16,1.17,1.18) is to make use of the locally equivalent NLP (1.12,1.13) at $x_0 = x^*$. Applying Newton's method to the first order KKT conditions of this NLP yields a locally convergent method. To obtain constraint values and gradients at each iterate, it is necessary to calculate exactly the local maximisers of g at each iterate.

The Lagrangian for this NLP is

$$L = f(x) + \sum_{i=1}^{m} \lambda_i c_i(x). \tag{1.19}$$

Applying Newton's method to the system of equations $\nabla_{x,\lambda} L = 0$ yields the following linear system of equations:

$$\begin{pmatrix} H & B \\ B^T & 0 \end{pmatrix} \begin{pmatrix} \delta x \\ \delta \lambda \end{pmatrix} = - \begin{pmatrix} \nabla f - B^T \lambda \\ -\underline{c} \end{pmatrix}$$

where $\underline{c} = (c_1(x_0), c_2(x_0), \ldots, c_m(x_0))^T$ and $B = [\nabla c_1(x_0), \nabla c_2(x_0), \ldots, \nabla c_m(x_0)]$.

Here δx is the increment to the current iterate, $\delta\lambda$ is the corresponding increment to the Lagrange multiplier estimates, and the matrix H is the Hessian of the Lagrangian (1.19).

Provided effort is spent in looking for *all* local maximisers at each iterate, and not merely those which are continuations $\tau_i(x)$ of the local maximisers τ_i known from the previous iterate, then this method is far more tolerant of changes in the number of local maximisers than Newton's method applied directly to the first order KKT conditions (1.15,1.16,1.17,1.18) for the SIP.

In the special case that the number of global maximisers is n, the step δx is completely determined by the subsystem

$$B^T \delta x = \underline{c}.$$

There is no need to find H, or to find the changes in the Lagrange multipliers to update x. However, the Lagrange multipliers are useful for determining if a constraint should be dropped. This is the basis of the exchange methods for linear SIP problems.

Other local superlinearly convergent methods have been proposed. For instance, van Honstede [55] gives a method which uses a QP with an affine (in x) semi-infinite constraint as a subproblem.

These local methods are superlinearly convergent on problems with the requisite degree of continuity. Unfortunately they are only locally convergent in general. Because of the excellent asymptotic convergence properties of these methods, many global methods have been created by modifying these local methods.

1.4 Discretization methods.

The main theme of these methods is that T is replaced by a finite subset (T_0 say) of itself. The effect of this is to substitute a finite number of ordinary constraints for the semi-infinite constraint. With this modification the SIP is turned into a finite NLP, which can be solved by existing methods. Of course the solution to the NLP will not (usually) be a solution of the SIP. In general the semi-infinite constraint will be violated at some t in $T - T_0$. In practice a succession of NLPs are solved. The NLPs differ from one another in the choices of T_0. This sequence of subsets of

T is chosen so as to force the violation of the semi-infinite constraint to zero. Under mild conditions this forces the sequence of NLP solutions to converge to a solution of the SIP.

Methods adapted from linear SIP.

Hu [56] proposes an algorithm which solves the SIP by solving a succession of finite Non Linear Programming (NLP) problems. The first problem in the sequence contains no constraints. Each subsequent problem is formed from its predecessor by adding one constraint of the form

$$g(x, t_0) \leq 0,$$

where t_0 is a point in T at which the semi-infinite constraint is violated at the solution to the preceding NLP. When g is affine with respect to x this method is essentially a cutting plane method. One disadvantage is that some of the NLPs encountered may be unbounded.

A second drawback is the risk of ill-conditioned NLPs or associated subproblems even when the SIP is well conditioned. This may arise in the following way. Let τ_0 be a global maximiser of g at a solution to which the sequence of iterates $\{x^{(k)}\}$ generated by the algorithm is converging. It is very likely that several approximations to τ_0 will be made by the algorithm. Each of these approximations will give rise to a constraint, and each one of these constraints will appear in all successive NLP subproblems. These constraints, and their linearizations are likely to be very similar, possibly leading to ill-conditioned subproblems in some NLPs.

Hettich [50], and Hettich and Gramlich [53] give discretization algorithms for linear SIP problems, and for convex quadratic SIP problems respectively. These algorithms are easily generalised to general non-linear SIP problems, and will be described here in those terms. These algorithms solve a general non-linear SIP by solving a succession of NLP problems. The i^{th} NLP in the sequence involves minimising the objective function subject to g being less than zero on a finite subset $\{T_i\}$, where $T_i \subseteq T$. Typically, each subset $\{T_i\}$ is chosen as the intersections of a grid. In any case, $T_i \subseteq T_{i+1}$ is required for all positive i: this forces each NLP to mimic the SIP at least as well as its predecessor.

As the discretizing grids become finer, the number of constraints in each NLP can become very large. To avoid excessive computation, the NLP problems are not solved as they stand. Some of the constraints arising from the discretization are initially omitted from the NLP. The NLP is then solved with this reduced constraint set. The solution so obtained is then checked to see if it violates any constraint which was omitted. If all such constraints are satisfied, then the original NLP has been solved. If at least one omitted constraint has been violated, then at least one of the omitted constraints is placed back in the NLP constraint set, and the process is repeated.

This appears to be an effective linearly convergent algorithm. It is unclear just how much effort is typically involved in solving each NLP with the full constraint set by solving a sequence of NLPs with partial constraint sets. The risk of ill-conditioning from several similar constraints is still present, although it is less than for Hu's algorithm.

Methods adapted from Convex SIP.

The algorithm of Ašić and Kovačević-Vujčić [6] is of interest. It does not require a convex objective function, although $g(x, t)$ must still be convex in x for all t. This algorithm is one of many which employs successively finer discretizations of T. A global Lipschitz constant for $g(x, t)$ with respect to t is also employed. This Lipschitz constant is global in the sense that for each value of x, it bounds the change in $g(x, .)$ between all pairs of points in T. This means that, for particular values of x, the actual value of the constant used may be far in excess of the minimum value the Lipschitz constant could take at that particular x. This can lead to a very significant loss of efficiency when finding the local and global maximisers.

The algorithm requires the existence of a Slater point. This is needed because each iterate generated by the algorithm lies in feasible region's interior. Feasibility of each iterate is made possible by exploiting the knowledge of a global Lipschitz constant. To generate the iterates, a sequence of NLP problems are solved. The objective functions of each NLP and the SIP are the same. For the k^{th} iteration, the NLP constraints are formed by considering a discrete subset $T_0^{(k)}$ of T, and then tightening the bound on g from 0 to $\eta^{(k)}$. That is:

$$g(x^{(k)}, t) \leq \eta^{(k)} \leq 0, \quad \forall t \in T_0^{(k)} \tag{1.20}$$

is used. By a suitable choice of the discretizing subset $T_0^{(k)}$ at each iteration, the global Lipschitz constant ensures that the solution to each NLP is also a feasible point of the SIP.

For this approach to be effective, every feasible point of the SIP must be a cluster point of the subsets of the feasible region defined by the sequence of the tightened discretized semi-infinite constraints (1.20). In [6], the convexity of g with respect to x for each value of t is used to establish this. Most non-convex SIP problems will also satisfy this condition.

Other Methods.

Mine, Fukushima and Tanaka [65] describe an algorithm which uses second order information. The problem is discretized in a very simple manner: the set T is replaced by the points on a regular grid. This grid is not altered throughout the solution process. In effect the semi-infinite constraint is replaced by a large number of ordinary constraints. The resultant NLP is solved by applying the Lagrange-Newton algorithm in conjunction with a trust region. The ℓ_∞ exact penalty function is employed as a merit function. The number of constraints included in each QP sub-problem is reduced by including only those which take a value not less than $-\epsilon$ at the relevant iterate, where ϵ is small and positive. This strategy may be ineffective at infeasible points: almost all of the NLP constraints may be included in the QP subproblem. Also, if the discretization grid is very fine, then the possibility of ill-conditioning from nearly identical constraints occurs.

Panier and Tits [72] propose a sequential 'QP' method using adaptive discretization of the constraint index set T. Their algorithm is described explicitly in terms of a one dimensional constraint index set. Their QP is comprised of linear approximations to the discretized semi-infinite constraint, together with a linear approximation to f. The QP does not include second order information about the objective or constraint functions. The quadratic term in the QP objective function is simply half the square of the proposed step's length.

The QP is:

$$\min_s \tfrac{1}{2}\|s\|^2 + v$$

subject to

$$s^T \nabla f - \gamma[\max_{t \in T_0} g(x_0, t)]_+ \leq v$$

$$\text{and} \quad s^T \nabla_x g(x_0, \tau) + g(x_0, \tau) - [\max_{t \in T_0} g(x_0, t)]_+ \leq v \qquad \forall \tau \in T_\epsilon,$$

where γ is a positive parameter. The set T_0 is an increasing sequence $\{t_i\}_{i=0}^N$ of $N+1$ equally spaced test points, such that $T = [t_0, t_N]$. The set T_ϵ is a subset of T_0, and contains all points which gave rise to constraints which were active at the previous QP's solution. It also contains all left local maximisers at which $g(x_0,.) > -\epsilon$, where ϵ is small and positive. The term left local maximiser denotes a point t_i for which $g(x_0, t_{i-1}) < g(x_0, t_i)$ provided $i \neq 0$, and also for which $g(x_0, t_{i+1}) \leq g(x_0, t_i)$ provided $i \neq N$. This strategy can reduce markedly the number of constraints in each NLP. It also reduces the propensity toward ill-conditioned NLP problems caused by several very similar constraints in the NLP.

The discretization of T is adaptive in the sense that whenever the descent predicted by the QP is small, or a prescribed number of iterations is reached, the discretization is refined. This refinement consists of doubling the number of points in the discrete subset of T used.

The algorithm has two phases: the first seeks feasibility whilst ignoring any changes in the objective function values, and the second seeks optimality without relinquishing feasibility. This avoids the need for a merit function, however it means this algorithm is not capable of handling non-linear equality constraints.

As no second order information is used the algorithm is only linearly convergent. Indeed, when well inside the interior of the feasible region, the set T_ϵ is empty, and the search direction is one of steepest descent.

The algorithm of Panier and Tits is a development of the algorithm of Gonzaga, Polak, and Trahan [39], which is in turn a development of the algorithm of Polak and Mayne [78]. Similar remarks can be made about these other two algorithms.

1.5 Multi-Phase Methods.

The methods listed in section 1.3 are local ones; they are guaranteed to converge to a stationary point only if the starting point is a sufficiently accurate approximation to that stationary point. In particular, knowledge of the number of global maximisers active at the solution is vital. If $p > 1$ the positional information of each global maximiser on the constraints defining T is also important, especially for the method based on (1.15,1.16,1.17,1.18). Unless a good approximation is known, these methods cannot be applied with any degree of certainty. On the other hand, methods such as those described in section 1.4 are globally convergent, but usually possess only a linear rate of convergence. Experimental results for linearly convergent algorithms for SIP and NLP problems show that these methods may be intolerably slow in practice. Various ways of hybridizing these two types of methods have been considered. Some of these are looked at in this section.

Gustafson [41] proposed creating a global superlinearly convergent method by using a two phase approach. The first phase consists of a global method, such as those outlined in section 1.4. This provides an approximation to the solution. The second phase refines this approximation using a local superlinearly convergent method.

At the end of the first phase, the algorithm must identify the number, and approximate position of each global maximiser, and possibly also the bounding constraints of T which are active for each global maximiser. The local method is then applied. If it fails to converge, or converges to a point which is not stationary, then phase 1 must be repeated to obtain a better starting point for the local method.

If a good initial estimate is needed for phase 2, then phase 1 may involve a considerable amount of work. There is no guarantee that the potential difficulties associated with linearly convergent methods will not be encountered in phase 1 well before a phase 2 starting point of adequate accuracy is reached. Even if a suitable approximate solution is found in phase 1, determining the number of global maximisers active at the solution, and the pertinent active bounding constraints of T for each such global maximiser is not completely straightforward — especially if $p > 1$.

Polak and Tits [79] also propose an algorithm which combines a local and a global method. The local method is the Lagrange-Newton method applied to the locally equivalent NLP (1.12,1.13). The global method is that of Gonzaga, Polak, and Trahan [39]. In both of these methods approximations to the local maximisers of the same accuracy as the current iterate approximates the SIP solution are used. In contrast to Gustafson, the local method is tried first at each iterate. If the step chosen by the local method is unacceptable, or it does not exist, then one iteration of the global method is applied. The step generated using the local method is accepted if its length is less than $K\eta^i$, where K and η are constants satisfying $K > 0$ and $0 < \eta < 1$, and where i is the number of times the local step has been accepted in the past.

This approach does make the Lagrange-Newton method globally convergent. However other difficulties with the Lagrange-Newton method remain. For instance, if the initial point is close to a local maximum of the SIP, then the Lagrange-Newton step may be chosen at each iteration — resulting in convergence to that local maximiser. This may be avoided by making K sufficiently small, but this increases the susceptibility of the algorithm to the deficiencies of the linearly convergent global method.

A completely different approach to globalizing locally convergent algorithms is proposed by Gfrerer, Guddat, Wacker, and Zulehner in [33]. Their algorithm is a modification of the continuation method for solving non-linear equations. It is described in terms of a finite NLP, but could easily be extended to a SIP problem. The SIP is replaced by a continuous family of SIP problems. This family $\mathcal{G}_\xi$ is indexed by a parameter ξ, which ranges over the interval $[0, 1]$. The SIP $\mathcal{G}_0$ is a SIP problem with a known solution. The SIP $\mathcal{G}_1$ is the original SIP problem. Briefly, an increasing sequence (in ξ) of SIP problems $\mathcal{G}_\xi$ is solved. The first member of the sequence is $\mathcal{G}_0$ and the last is the original SIP $\mathcal{G}_1$. The solution of each $\mathcal{G}_\xi$ is used as the initial point for the next SIP in the sequence. Each SIP is solved using a locally convergent algorithm.

The next section discusses methods of making the second phase more robust. In some cases these methods are globally convergent, in which case the first phase is no longer necessary.

1.6 Exact Penalty Function Methods.

In unconstrained optimisation, the algorithm created by applying Newton's method to the first order KKT conditions is easily globalised by using, for example, a line search or a trust region. In the unconstrained case comparing the relative merits of the current iterate and its prospective successor is easy; one simply compares the two pertinent function values. Unfortunately, as in all constrained optimisation problems, in solving the SIP (1.1,1.2) there are two (often conflicting) aims: minimising the objective function, and satisfaction of the constraints. If globalisation of a local method for solving the SIP is to be achieved, some means of reconciling advance toward one aim at the expense of the other is needed. A common approach to this for both SIPs and NLPs has been to use an exact penalty function as a merit function [46, 97, 21]. For NLPs the ℓ_1 exact penalty function

$$\Phi_1(x) = f(x) + \mu \sum_{i=1}^{m} [c_i(x)]_+ \qquad (1.21)$$

is the current favourite. Here $[c]_+$ denotes the maximum of c and 0. For SIPs, the (apparently) corresponding penalty function is identical, where each $c_i(x)$ is a local maximum of g at x as given by (1.13). Clearly if Φ_1 is to be finite, the number of local maxima of g taking positive values must be finite for all x in the region of interest.

In Charalambous [15] conditions for a local minimum of the SIP to be a local minimum of the L_1 exact penalty function are given. Specifically, if the SIP is locally equivalent to a NLP at a solution, and if second order sufficiency conditions hold at that solution then

$$\mu > \|\lambda^*\|_\infty \tag{1.22}$$

ensures the SIP solution under discussion is also a local minimum of the L_1 exact penalty function.

Another exact penalty function which appears in the NLP literature is the ℓ_∞ exact penalty function. Its equivalent for SIP problems is the L_∞ exact penalty function:

$$\Phi_\infty(x) = f(x) + \mu \max_{t \in T}[g(x,t)]_+. \tag{1.23}$$

In contrast to the L_1 exact penalty function, the L_∞ exact penalty function is always continuous. Tanaka et al [94] give an example of a SIP with a discontinuous L_1 exact penalty function. In case another example is of interest, here is one. Let $n = p = 1$, $T = [-10, 10]$, $f \equiv 0$, $\mu = 1$, and

$$g(x,t) = \frac{25x^2 - 1}{25x^2 + 1} + t^2(\tfrac{1}{4} - x^2) - 8t^4.$$

Then Φ_1 and Φ_∞ are as pictured in figures 1a and 1b respectively.

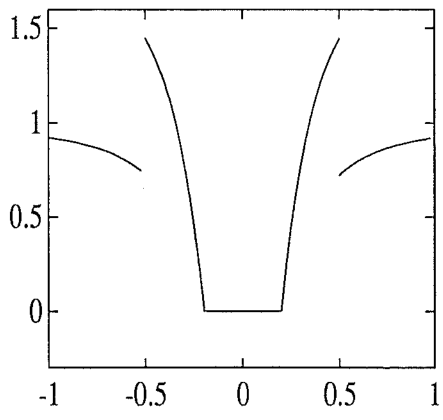

Figure 1a. The L_1 penalty function.

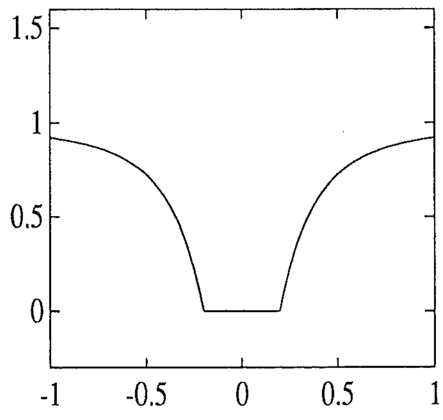

Figure 1b. The L_∞ penalty function.

The potential for discontinuities in the L_1 exact penalty function to prevent convergence to a solution of the SIP is obvious. Such discontinuities can only occur at infeasible points, and are caused by a change in the number of local maxima taking positive values other than by the value of the local maximum changing sign. Such a change occurs for example when two local maxima combine into one (as in the above example) or when a local maximum changes into a stationary point.

1.6.1 SOLVER-like Methods.

Hettich and van Honstede [54] considered adapting the locally convergent SOLVER method of Wilson [100] for finite NLPs to SIP problems. They discuss locally convergent methods for SIP problems. Watson [97, 99] considers a similar generalization together with the L_1 exact penalty function (1.21). The SOLVER-like method proposed in [97, 99] exploits the local equivalence of the SIP to an NLP. The usual way this equivalence is obtained requires the assumption that g is twice continuously differentiable. With this local equivalence, an exact penalty function based SOLVER-like method for SIP problems may be modelled closely on such methods for finite NLP problems such as those of Han, and of others [46, 47, 84, 48]. At each iterate the following sub-problem is solved to yield a search direction:

$$\min_{s \in R^n} s^T \nabla f(x_0) + \frac{1}{2} s^T H s \tag{1.24}$$

$$\text{subject to} \quad c_i(x_0) + s^T \nabla c_i(x_0) \leq 0 \;\; \forall i = 1, \ldots, m. \tag{1.25}$$

where H is a positive definite $n \times n$ matrix. Once the search direction is selected, the penalty parameter μ is adjusted via (1.22) to ensure the search direction is one of descent for the exact penalty function. An Armijo style line search is then done to determine the next iterate. The criterion for acceptance being that the actual reduction in the penalty function is at least a given fraction of that predicted by the directional derivative of the penalty function along the search direction.

The algorithm does not seek to minimise the exact penalty function as such: it merely uses it to adjudicate between the two aims of minimising f, and ensuring feasibility.

If H is indefinite, then the solution of (1.24,1.25) may be a direction of ascent for the penalty function. This restricts the choice of H. Ideally the matrix H would be the Hessian of the Lagrangian (1.19) evaluated at x_0, with λ set at a suitable estimate λ_0 of the optimal Lagrange multiplier vector. For sufficiently accurate estimates of a stationary point of the SIP, and of the associated Lagrange multiplier vector, solving (1.24,1.25) is, under mild conditions, equivalent to applying Newton's method to the system of equations

$$\nabla_{x,\lambda}\left[f(x)+\sum_i \lambda_i c_i(x)\right]=0$$

where the sum is only over values of i pertaining to constraints which are active at the pertinent stationary point. Unfortunately such a choice of H may not be positive definite. Watson [99] imposes positive definiteness on H by choosing $H=\nabla^2_{xx}L+\gamma I$, for some suitable positive value of γ. Initially $\gamma=0$ is always tried. If $\nabla^2_{xx}L$ is not positive definite at the solution then second order convergence may be inhibited by the γI term.

The method of Coope and Watson [21] is also a SQP method employing an L_1 exact penalty function. The objective and constraint functions are taken to be twice continuously differentiable. As before, local equivalence to an NLP is obtained as described in section 1.2. The search direction s_0 at the iterate x_0 is chosen as the solution of the IQP (1.24,1.25).

Watson's method requires that the matrix H to be positive definite, whereas the second order sufficient KKT conditions only guarantee that H is positive definite on the subspace orthogonal to the constraint normals $\{\nabla_x g(x^*,t) : t \in \Gamma^*\}$.

If $\nabla_{xx}L^*$ is not positive definite then forcing positive definiteness in the manner described in [99] will generally result in a loss of superlinear convergence. This is circumvented by solving (1.24,1.25) as a series of Equality constrained Quadratic Programmes (EQP), each of the form

$$\min_{s\in R^n} s^T\nabla f(x_0)+\frac{1}{2}s^T H s \tag{1.26}$$

$$\text{subject to}\quad c_i(x_0)+s^T\nabla c_i(x_0)=0,\ \ \forall i\in\mathcal{A}, \tag{1.27}$$

where $\mathcal{A}$ is the set of indices of the constraints which feature in the EQP. Let $\mathcal{M}(\mathcal{A})$ denote the subspace span$\{\nabla c_i(x_0) : i \in \mathcal{A}\}$, and let $\mathcal{A}_E$ be the intersection of the class of sets $\mathcal{A}$ of all EQPs encountered in solving the IQP (1.24,1.25). The set of constraints indexed by $\mathcal{A}_E$ can be taken as equality constraints, in which case H need only be positive definite on the subspace orthogonal to $\mathcal{M}(\mathcal{A}_E)$. This can be imposed on H in the way described for Watson's algorithm [97]. The reason positive definiteness is only required on a subspace is that if H is replaced by $H+\sigma CC^T$, where the columns of C lie in the subspace $\mathcal{M}(\mathcal{A}_E)$, and where $\sigma \geq 0$, then the solution to the IQP (1.24,1.25) remains unchanged. For sufficiently large σ and a suitable choice of C, $H + \sigma CC^T$ is positive definite.

Near a solution at which second order sufficient KKT conditions hold for the SIP, the Hessian of the Lagrangian will be positive definite on $[\mathcal{M}(\mathcal{A}_E)]^\perp$. Hence $H = \nabla^2_{xx}L$ will be used, and the connection with Newton's method for the first order KKT conditions is retained.

As the replacement of H by $H+\sigma CC^T$ is not done explicitly, the minimal value of the penalty parameter is no longer given by (1.22). It can be shown that, in the limit $x \longrightarrow x^*$, (1.22) does give the minimal value of μ. However, far from the solution, a much higher value than that indicated by (1.22) may be required. Accordingly μ is first calculated using (1.22), and then repeatedly doubled until the solution of the IQP (1.24,1.25) is a direction of descent for the penalty function.

Having chosen the search direction, and a suitable value of the penalty parameter, an Armijo type line search is performed, where the criterion for sufficient descent is that the ratio of the actual descent to the predicted descent exceeds ρ, where $0 < \rho < \frac{1}{2}$. The predicted descent is calculated from the gradient of the penalty function along the search direction. Like Watson's algorithm, this algorithm does not directly minimise the non-differentiable penalty function: it uses it only as a merit function in the line search.

Both of these SOLVER-like methods assume that the IQP (1.24,1.25) generated at each iterate has a solution. It is quite possible that, far from a solution of the SIP, the IQP formed as indicated is infeasible. If this is so these two methods, in the above form, fail. They may be globalised in the way described by Polak and Tits, although some of the drawbacks involved in the Polak and Tits algorithm will still be present.

Conn and Gould [19] observe that the L_1 exact penalty function (1.21) is more closely related to the ℓ_∞ exact penalty function for an NLP than the ℓ_1 penalty function for an NLP. For each $t \in T$, $g(x,t) \leq 0$ is a constraint on x. In effect (1.21) divides the set of constraints $\{g(x,t) \leq 0 : t \in T\}$ up into a collection of disjoint subsets. Each of these subsets is the region of attraction of a stationary point of $g(x,t)$ with respect to the local search algorithm used in finding each local maximiser. The L_1 exact penalty function is the sum over this collection of subsets of the maximum constraint violation on each subset.

Pietrzykowski [76] proposed a direct generalisation of the L_1 exact penalty function for a NLP, viz

$$\Psi(x,\mu) = \mu f(x) + \int_T [g(x,t)]_+ \, dt,$$

where μ is a positive scalar. Unfortunately, as pointed out in [19], simple examples can be constructed for which Ψ is not an exact penalty function. For instance, using $n = p = 1$, $T = [-1,1]$, and

$$f(x) = -x, \quad g(x,t) = t - x - 1$$

then

$$\Psi(x,\mu) = -\mu x - \tfrac{1}{2}[x]_+^2$$

and $\mu \longrightarrow 0$ is needed if $x = 0$ is to be a stationary point of Ψ.

Conn and Gould use a modified form of Ψ, specifically

$$\Psi = \mu f(x) + \frac{\int_T [g(x,t)]_+ \, dt}{\int_T [\operatorname{sign}(g(x,t))]_+ \, dt}. \tag{1.28}$$

The second term in this equation is the average value of the constraint violations. The integral in its denominator is the Lebesgue measure of the part of T on which $g(x,.)$ exceeds its bound 0.

To avoid discontinuities in Ψ at infeasible points, Conn and Gould require that $\forall x \in R^n$, no subset of T of strictly positive Lebesgue measure exists on which $g(x,t) = 0$. If this is not true then the denominator of the second term of (1.28) may be discontinuous. Second order sufficiency, and strict complementarity are also required for both the local minimiser of the SIP, and the global maximisers of the constraint function at that x value. The requirement that the number of global maximisers is finite at each iterate is not made by Conn and Gould. This is quite special: almost all other algorithms except those which discretize the semi-infinite constraint require this assumption. The discretization algorithms avoid the need for this assumption by simply solving each NLP subproblem outright. The solution of each NLP is treated as an approximation to that of the SIP; no descent of a penalty function for the SIP is required.

1.6.2 Trust Region Methods.

The second method proposed by Watson [97] employs a trust region, with an L_1 exact penalty function. Using a suitable value of the penalty parameter, a local minimum of the exact penalty function is sought directly. The prospective step at x_0

is chosen as as the minimiser of the following L_1QP approximation to the exact penalty function:

$$\min_{s\in R^n} s^T\nabla f(x_0) + \frac{1}{2}s^T Hs + \mu \sum_{i=1}^{m}[c(x_0) + s^T\nabla c_i(x_0)]_+ \tag{1.29}$$

$$\text{subject to } \|s\|_\infty \leq \Delta, \tag{1.30}$$

where Δ defines the size of the trust region. If the prospective step results in a reduction of the exact penalty function, it is accepted. Otherwise, the value of Δ is reduced, and the L_1QP is re-solved at the current iterate. Once an acceptable step has been found, Δ is decreased, left unaltered, or increased depending on the ratio of the actual decrease of the exact penalty function to that predicted by the L_1QP. Another iteration is then begun.

Unlike the IQP (1.24,1.25) it is clear that the L_1QP always has a solution. Furthermore, as no line search is employed, there is no need for the prospective step to be a direction of descent for the exact penalty function. Hence H may be chosen as the Hessian of the Lagrangian irrespective of whether or not it is positive definite.

Tanaka, Fukushima and Ibaraki [94] also propose a trust region method. Unlike Watson [97], the L_∞ exact penalty function (1.23) is used. Both f and g are taken to be C^2 functions. At each x the local equivalence of the SIP to an NLP is assumed. At each iterate x_0 the prospective step s_0 is chosen as the solution to the following L_∞QP approximation to Φ_∞ at x_0:

$$\min_{s\in R^n} s^T\nabla f(x_0) + \frac{1}{2}s^T Hs + \mu \max_{i\in\{1,\ldots,m\}}[c(x_0) + s^T\nabla c_i(x_0)]_+ \tag{1.31}$$

$$\text{subject to } \|s\|_\infty \leq \Delta \tag{1.32}$$

where Δ is the size of the trust region, and H is the Hessian of the Lagrangian at x_0.

In practice the algorithm works only with local maximisers at which g exceeds $-\eta$, where η is positive. These local maximisers are henceforth referred to as prominent local maximisers. All other local maximisers are ignored.

For the prospective step to be accepted it must result in a decrease of the penalty function. The size of the trust region is adjusted in the usual manner. Tanaka et al impose the additional requirement that each prominent local maximiser of $g(x_0 + s_0, t)$ must be identifiable as a continuation of some prominent local maximiser at x_0, where the identification process is one-to-one. If some prominent local maximisers at $x_0 + s_0$ can not be matched, then each such local maximiser is added to the list of points in T from which the linear constraint approximations are formed. The L_∞QP is then re-solved at the current iterate for a new prospective step.

If a good reduction in the penalty function is obtained with some s_0, it would appear that rejecting s_0 on these grounds, and recalculating it with an augmented set $\mathcal{A}$ is wasteful. Each such recalculation results in an extra global optimisation, so the extra computation may be considerable. However, if no reduction in the penalty function is obtained, then augmenting $\mathcal{A}$ requires little or no extra effort.

The algorithm of Tanaka et al [94] is a development of those by Mine, Fukushima, and Tanaka [65], and by Tanaka, Fukushima and Hasegawa [93].

Bell [11] gives an algorithm which replaces the SIP with the problem of minimising its L_∞ exact penalty function. Bell's algorithm is developed in a wider context: that of minimising a locally Lipschitz function using approximate function values, where the cost of calculating the function values increases with the accuracy required of these values. Because of this, the algorithm does not need exact values for the global maximisers at each iteration. Instead approximations to those maximisers are used; the accuracy required of these approximations is increased as the steps taken at each iteration become smaller. A major advantage of this approach is that the algorithm can cope with other sources of uncertainty in the function values provided the uncertainty can be made arbitrarily small with enough effort. A Levenberg-Marquadt parameter is used to implement a trust region strategy. Curvature information is included in the form of an approximation to the Hessian of the Lagrangian, and is updated using a quasi-Newton method.

1.7 Remarks.

Semi-infinite programming is an extension of finite non-linear programming. The relationship between these two types of problems is close: this is reflected in the various methods of solving them.

Under certain conditions, a SIP is locally equivalent to a NLP. This fact can be exploited directly to obtain local methods for SIP problems. Local methods for semi-infinite programming, such as the Lagrange-Newton method, have all the disadvantages of their NLP counterparts. In addition, they are susceptible to changes in the number of local maximisers. This is especially true of Newton's method when applied to the combined first order KKT conditions for the SIP, and for each global maximiser of the constraint function.

Various globally convergent algorithms for SIP have also been developed. These often involve repeatedly discretizing the semi-infinite constraint by replacing T with a succession of finite subsets of itself. Many of the discretization algorithms are generalisations of algorithms created for linear or convex SIP problems. Typically these algorithms are linearly convergent, and so risk all the usual practical convergence problems of such methods. Linear convergence is not an inherent feature of discretization algorithms — Bell's algorithm (for example) uses second order information in choosing each step.

Many multi-phase methods have been developed by hybridizing a locally convergent superlinear method with a global linearly convergent one. The general idea being to use the global method to get close to the solution and then switch over to the local superlinear one to obtain a very accurate estimate of the solution. When successful, this avoids the deficiencies of both methods. The changeover from one method to the other is not entirely trivial, and if not done correctly then the flaws of one or other method may affect the algorithm's performance. Also, there is no guarantee that the local method can be used effectively before the linearly convergent global method encounters difficulties.

These snags can be avoided if the superlinearly convergent method used in the second phase is also globally convergent. When the first phase method strikes difficulties, the second phase method can be used safely from then on. There are still advantages in the two phase approach: for a good choice of algorithm for the first phase, the work required in each iteration of the first phase will be much less than that taken by an equivalent iteration of the second phase method.

As with NLP problems, good theoretical properties are obtained by using a SQP method backed up by an exact penalty function. In contrast to the NLP case, the L_1 exact penalty function may possess discontinuities in the infeasible region even if the objective and constraint functions are continuous everywhere. The L_∞ exact penalty function avoids these discontinuities, and thus is a better choice for SIP problems of the form (1.1,1.2). In light of the current preference for L_1 exact penalty functions for NLP problems, this would suggest a mixed $L_1 L_\infty$ exact penalty function would be a good choice for SIPs involving several semi-infinite constraints, or auxillary finite constraints, or both. The constraint term of the mixed exact penalty function would consist of the sum of the maximum constraint violations of each semi-infinite constraint, plus the sum of the auxillary constraints' violations.

The use of an L_1QP together with a trust region guarantees that a prospective step can be generated at each iteration. The existence of a feasible region for the L_1QP is guaranteed. The trust region ensures unboundedness can not be a problem. Similar remarks apply to L_∞QP subproblems.

Even with discretization the L_∞ exact penalty function retains some of its advantages over its L_1 counterpart. At highly infeasible points the L_1 exact penalty function may require a very large number of constraints in the QP subproblem. With the L_∞ penalty function the constraints arising from many of the infeasible points which take values appreciably below the global maximum may be omitted.

In conclusion it appears that a good method would possess two phases. The first would calculate an approximate solution using one, or more discretized versions of the SIP. The second phase would refine this approximation using an algorithm based on an L_∞ exact penalty function. This second phase algorithm would be along the lines of either Coope and Watson's algorithm or Tanaka et al's algorithm.

Chapter 2

THE THEORETICAL BASIS FOR THE ALGORITHM.

In this chapter various theoretical properties of semi-infinite programmes are established, where the semi-infinite programmes are required to satisfy some mild assumptions which make the SIP tractable. A quasi-Newton algorithm for SIP is described, and a convergence proof for this algorithm is presented. The work which appears in this chapter is an expansion of that presented in Price and Coope [85, 86].

A common approach to SIP which yields global convergence is the use of Sequential Quadratic Programming (SQP) techniques in conjunction with an exact penalty function [19, 21, 94, 98]. The methods given in [21, 94, 98] use an implicit function theorem on each semi-infinite constraint to establish convergence. Use of the implicit function theorem requires that the constraint function be C^2. The L_1 exact penalty function algorithm of Conn and Gould [19] is along somewhat different lines, but makes use of similarly restrictive assumptions. The purpose of this chapter is to show that provided the exact penalty function is based on the infinity norm, a much weaker condition than that required for the implicit function theorem to hold is sufficient to ensure convergence for C^1 problems. The algorithm presented can take second order information into account, yielding superlinear convergence on problems with the requisite degree of continuity.

2.1 The penalty function problem.

The approach taken is to replace the SIP with the rather more tractable problem of minimizing a non-differentiable penalty function ϕ. This penalty function is chosen so that solutions of the SIP are also solutions of the Penalty Function Problem (PFP):

$$\min_{x \in R^n} \phi(x).$$

The exact penalty function used is:

$$\phi(\mu, \nu; x) = f(x) + \mu\theta + \tfrac{1}{2}\nu\theta^2 \quad \text{where} \quad \theta = \max_{t \in T} [g(x,t)]_+ .$$

The penalty parameters μ and ν are restricted to $\mu > 0$, and $\nu \geq 0$. Clearly $\theta(x)$ is the infinity norm of the constraint violations, so ϕ is continuous $\forall x \in R^n$.

The quadratic term in θ has been included to reduce the risk that μ will be set at a value far in excess of that required in theorem 2.2. If this does occur it may have a detrimental effect on the algorithm's performance [20]. If the initial point used by the algorithm is highly infeasible, very high values of one or other penalty parameter may be needed if feasibility is eventually to be attained. For instance, let $f \approx -\exp(x_1)$ in the x_1 direction, with a constraint $x_1 \leq 0$, and with the initial value $x_1 = 10$. If $\nu \equiv 0$ is used, and if the sequence of iterates is to converge to a feasible point, then the minimum value μ can (initially) take is over 10,000. On the other hand if ν is set at an extremely high value, when θ becomes small, the quadratic term in θ is much smaller than the linear term. The effect of a high ν is therefore much less pronounced.

The algorithm to be described uses only first derivatives: accordingly it is desirable that the algorithm be capable of solving C^1 problems. This precludes the use of second order optimality conditions in specifying solutions of an arbitrary problem of the form (1.1,1.2). Consequently stationary points of the SIP will be regarded as valid solution points. The first order optimality conditions, together with an appropriate regularity assumption, are given in inequality (1.4), and equations (1.5,1.6) respectively.

A definition of what constitutes a solution of the PFP is needed. In accordance with the remarks in the previous paragraph, only first derivatives are used.

Definition 2.1 *For fixed values μ_0 and ν_0 of μ and ν, a point x_0 is a* ***critical point*** *of $\phi(\mu_0, \nu_0; x)$ iff at x_0 the directional derivative of $\phi(\mu_0, \nu_0; x)$ with respect to x along every direction is non-negative.*

The solution set of the PFP for $\mu = \mu_0$, and $\nu = \nu_0$ is defined as the set of critical points of $\phi(\mu_0, \nu_0; x)$.

If solving the PFP is to yield a solution of the SIP, it is highly desirable that the PFP's solution set be contained in (and ideally be equal to) the SIP's solution set. This can be achieved to a limited extent by a suitable choice of μ, for any ν.

Theorem 2.2 *Let x^* be an optimal point of the SIP (1.1,1.2) at which the regularity assumption (inequality (1.4)) holds, and let λ^* be the vector of Lagrange multipliers as specified in (1.5,1.6). If μ satisfies*

$$\mu > \|\lambda^*\|_1 \tag{2.1}$$

then x^* is a critical point of $\phi(\mu, \nu; x)$.

Conversely, if x^* is both feasible, and a critical point of $\phi(\mu, \nu; x)$ for some $\mu > 0$, and $\nu \geq 0$, then x^* is a solution point of the SIP.

PROOF. The first item follows from theorem 2.1 of [8], and from theorem (1.1). For the second item, if x^* is a critical point of ϕ for some μ, and ν, then

$$\forall x \text{ near } x^*, \quad \phi(\mu, \nu; x) \geq \phi(\mu, \nu; x^*) + o(\|x - x^*\|).$$

Now $\phi \equiv f$ on the SIP's feasible region, and so x^* is a solution of the SIP. ♣

This theorem implies the set of feasible critical points of the PFP are a subset of the set of stationary points of the SIP. The relationship between the two solution sets falls short of the ideal in two respects.

Firstly, there may be critical points which are not feasible, and therefore not solutions to the SIP. This admits the possibility that the algorithm may fail to solve the SIP by (in essence) failing to find a feasible point. This is characteristic of any algorithm attempting to attain feasibility from an arbitrary initial point by seeking a local minimum of the constraint violations. If the algorithm fails for this reason a common response is to consider other initial points.

Secondly, there may be solution points of the SIP which are not feasible critical points. This problem is circumvented by automatically adjusting μ so that any SIP solution is a critical point provided it is sufficiently close to some iterate.

2.2 Existence of an approximating L_∞QP.

It has been shown in the previous section that the SIP may be replaced by the problem of locating feasible solutions of the PFP. The PFP is tackled as follows. At each iterate linear approximations to all global (and some local) maximal values of the constraint function are formed. From these a local approximation to $\theta(x)$ can be constructed. This, together with an approximation to the objective function, yields an approximation to ϕ, and hence an L_∞ Quadratic Programme (hereafter L_∞QP) locally approximating the PFP. The solution of this L_∞QP yields a search direction along which the next iterate is sought, using an Armijo type line search.

All global maximisers of $g(x_0, t)$ are needed to obtain an adequate approximation to $\theta(x)$ for x near x_0. In order to ensure each iteration of the algorithm listed in section 2.3 is a finite computational process, the following assumption is made.

Assumption 2.3 *For each $x \in R^n$, the number of global maximizers of $g(x,t)$ over T is finite.* ♣

This, together with the other usual assumptions, ensures the convergence of the algorithm; use of an implicit function theorem is superfluous. Actually, it is sufficient that the number of global maximizers of g is finite for each x at which approximations to the global maximizers are calculated explicitly, and at each cluster point of the sequence of iterates. For convenience, assumption 2.3 is used.

The existence of an approximating L_∞QP is shown by examining the behaviour of the set of global maximizers $\Gamma(x)$ of $g(x,t)$ at points x near some point x_c, where x_c satisfies assumption 2.3. The first result states $\Gamma(x)$ is semi-continuous with respect to x.

Proposition 2.4 *Let C be a compact subset of T, and let $\Omega(x_c)$ be the set of global maximizers of $g(x_c,t)$ on C. If $\Omega(x_c)$ is a subset of the interior of C relative to T (hereafter int(C)), then firstly*

$$\forall \epsilon > 0, \;\; \exists \eta(\epsilon) > 0 \;\; : \;\; \forall x \in R^n, \;\; \|x - x_c\| < \eta \;\; \Rightarrow \;\; \Omega(x) \subset \mathcal{N}_\epsilon(\Omega(x_c)),$$

$$\textit{where } \mathcal{N}_\epsilon(\Omega(x_c)) = \{t \in T \; : \; \exists \gamma \in \Omega(x_c) \;\; \textit{satisfying} \;\; \|t - \gamma\| < \epsilon\},$$

and secondly, each element of $\Omega(x)$ is a local maximizer of $g(x,t)$ over T, for all x sufficiently near x_c.

PROOF. Use the topology on T induced by the standard topology on R^p. Let g_c be the global maximal value of $g(x_c, t)$ on C. For all small positive ϵ, as $C - \mathcal{N}_\epsilon(\Omega(x_c))$ is compact and non-empty, $g(x_c, t)$ achieves its supremum on $C - \mathcal{N}_\epsilon(\Omega(x_c))$, which must be strictly less than g_c. Define

$$m(\epsilon) = g_c - \max_{t \in C - \mathcal{N}_\epsilon(\Omega(x_c))} g(x_c, t).$$

Now the continuity of $\nabla_x g$ with respect to all arguments, and the compactness of T imply the set of functions $\{g(x,t)\}_{t \in T}$ is equicontinuous with respect to x. Therefore

$$\forall \epsilon > 0, \ \ \exists \eta(\epsilon) > 0, \ \ \text{such that} \ \ \forall x, \ \ \text{and} \ \ \forall t \in T,$$

$$\|x - x_c\| < \eta(\epsilon) \ \ \Rightarrow \ \ |g(x_c, t) - g(x, t)| < \tfrac{1}{4} m(\epsilon).$$

Hence, for all these values of x,

$$\forall t \in C - \mathcal{N}_\epsilon(\Omega(x_c)), \quad g(x,t) < g_c - \tfrac{3}{4} m(\epsilon),$$

$$\text{and} \ \ \forall t \in \Omega(x_c), \qquad g(x,t) > g_c - \tfrac{1}{4} m(\epsilon).$$

Hence, $\Omega(x) \subset \mathcal{N}_\epsilon(\Omega(x_c))$. Moreover, as g is continuous, and C compact, $\Omega(x_c)$ is also compact. Whence, for all small positive ϵ, $\mathcal{N}_\epsilon(\Omega(x_c)) \subset \text{int}(C)$, and so $\Omega(x)$ is a subset of the local maximizers of $g(x,t)$ over T. ♣

For any $x_0 \in R^n$, let $\Gamma(x_0) = \{\tau_1, \ldots, \tau_j\}$ be the finite set of strict global maximizers of $g(x_0, t)$. Proposition 2.4 implies each member of $\Gamma(x_0)$ may be considered separately. Let

$$\epsilon_0 = \tfrac{1}{4} \min\{\|\tau_i - \tau_k\| \ : \ i, k \in 1, \ldots, j, \ \ i \neq k\},$$

$$\text{and let} \ \ B_i(\epsilon_0) = \{t \in T \ : \ \|t - \tau_i\| \leq \epsilon_0\}, \ \ \forall i = 1, \ldots, j.$$

The set of global maximizers of $g(x,t)$ on the set $B_i(\epsilon_0)$ is denoted by $\Xi_i(x)$. By choosing $C \equiv T$ and $x_0 \equiv x_c$ in proposition 2.4 it is clear that

$$\Gamma(x) \subseteq \cup_{i=1}^{j} \Xi_i(x) \ \ \forall x \ \ \text{sufficiently near} \ \ x_0.$$

This shows that only the global maximisers of g at x_0 need be considered when forming an approximation of $\theta(x)$ in a sufficiently small neighbourhood of x_0. The behaviour of Γ with respect to changes in x is examined by considering each Ξ_i along each ray of the form $x(\sigma) = x_0 + \sigma u$, where $\sigma \geq 0$, and u is a unit vector in R^n.

Definition 2.5 *A function $t(\sigma)$ is an extension of the global maximizer $\tau_i \in \Gamma(x_0)$ along $x(\sigma) = x_0 + \sigma u$, where $\sigma \geq 0$, iff*

1. $t(0) = \tau_i$.

2. $\exists \sigma_{max} > 0$ *such that* $t(\sigma) \in \Xi_i(x(\sigma)), \quad \forall \sigma \in [0, \sigma_{max}]$. ♣

From proposition 2.4, each τ_i has at least one extension for each u. It may have several, or even an infinite number of extensions. The extensions may be discontinuous functions. Proposition 2.4 implies that, for all x near x_0, the extensions of $\Gamma(x_0)$, evaluated at x, are local maximizers of $g(x,t)$ over T, and contain $\Gamma(x)$. The extensions in the direction u of the members of $\Gamma(x_0)$ yield the following set of values of g along the ray $x(\sigma)$:

$$\{g(x(\sigma), t(\sigma)) \,:\, t(\sigma) \text{ is an extension of some } \tau_i \in \Gamma(x_0)\}.$$

This set is finite; any two extensions of the same τ_i take the global maximal value of $g(x(\sigma), t)$ over $B_i(\epsilon_0)$, for all sufficiently small positive σ. For $i = 1, \ldots, j$ let $t_i(\sigma)$ be an extension of τ_i. Each member of the set $\{g(x(\sigma), t_i(\sigma))\}_{i=1}^{j}$ is locally Lipschitz with respect to σ by the C^1 continuity of g, and the compactness of T. In order to form a set of linear approximations to $\{g(x(\sigma), t_i(\sigma))\}$, the following result is needed.

Proposition 2.6 *Let $t_i(\sigma)$ be any extension of $\tau_i \in \Gamma(x_0)$ along the ray $x(\sigma) = x_0 + \sigma u$, $\sigma \geq 0$. Then*

$$g(x(\sigma), t_i(\sigma)) = g(x_0, \tau_i) + \sigma u^T \nabla_x g(x_0, \tau_i) + o(\sigma).$$

Proof.

$$\begin{aligned} g(x(\sigma), t_i(\sigma)) &\geq g(x(\sigma), \tau_i), \quad \Rightarrow \\ g(x(\sigma), t_i(\sigma)) &\geq g(x_0, \tau_i) + \sigma u^T \nabla_x g(x_0, \tau_i) + o(\sigma). \end{aligned} \tag{2.2}$$

Also,

$$\begin{aligned} g(x(\sigma), t_i(\sigma)) &= g(x_0, t_i(\sigma)) + \sigma u^T \nabla_x g(x_0, t_i(\sigma)) + o(\sigma) \\ &\leq g(x_0, \tau_i) + \sigma u^T \nabla_x g(x_0, t_i(\sigma)) + o(\sigma). \end{aligned}$$

Now, as $\Xi_i(x_0)$ is a singleton set, proposition 2.4 implies every extension of τ_i is right continuous at $\sigma = 0$. Hence

$$g(x(\sigma), t_i(\sigma)) \leq g(x_0, \tau_i) + \sigma u^T \nabla_x g(x_0, \tau_i) + o(\sigma)$$

This, and inequality (2.2) yield the required result. ♣

Proposition 2.6 implies that $g(x_0+s, \Xi_i(x_0+s))$ has a unique linear approximation of order $o(\|s\|)$, for all s such that $\|s\|$ is small.

Define ψ to be a continuous piecewise quadratic approximation to ϕ near x_0, where ψ is based on the finite subset $\mathcal{A}_0$ of T, as follows

$$\psi(x_0, \mathcal{A}_0; \mu, \nu; s) = f(x_0) + s^T\nabla f(x_0) + \tfrac{1}{2}s^T H s + \mu\zeta(s) + \tfrac{1}{2}\nu\zeta^2(s),$$

$$\text{where } \zeta(s) = \max_{t\in\mathcal{A}_0}[g(x_0,t) + s^T\nabla_x g(x_0,t)]_+,$$

and where H is positive definite. Clearly ψ is strictly convex in s.

Let the base set $\mathcal{A}_0$ be $\{t_i\}_{i=1}^r$. For each $i = 1,\ldots,r$ define row i of the matrix B as $B_i = [\nabla_x g(x_0,t_i)]^T$, and define element i of the vector b to be $b_i = g(x_0,t_i)$.

Theorem 2.7 *If $\Gamma(x_0) \subseteq \mathcal{A}_0$ then, for all $s \in R^n$ such that $\|s\|$ is small,*

$$\phi(\mu,\nu; x_0 + s) = \psi(x_0, \mathcal{A}_0; \mu, \nu; s) + o(\|s\|). \tag{2.3}$$

PROOF. For all s sufficiently small, each element of $Bs + b$ arising from some member of $\mathcal{A}_0 - \Gamma(x_0)$ is less than every element of $Bs + b$ arising from some member of $\Gamma(x_0)$; thus $\mathcal{A}_0 - \Gamma(x_0)$ can be disregarded for small s.

The set of extensions of $\Gamma(x_0)$, evaluated at $x_0 + s$, contains $\Gamma(x_0+s)$ for s small, so proposition 2.6 implies

$$\forall\tau \in \Gamma(x_0+s),\ \ \exists t_0 \in \Gamma(x_0) \ \text{ such that}$$

$$g(x_0+s,\tau) = g(x_0,t_0) + s^T\nabla_x g(x_0,t_0) + o(\|s\|).$$

Hence

$$\theta(x_0+s) = \max_{t\in\mathcal{A}_0}[g(x_0,t_0) + s^T\nabla_x g(x_0,t_0)]_+ + o(\|s\|).$$

Using a linear approximation to the objective function, the result follows. ♣

The convergence proof requires that $\|s\|_\infty$ be subject to an upper bound, specifically $S_b \gg 0$. The L_∞QP

$$\min_{s\in R^n}\psi(x_0,\mathcal{A}_0;\mu,\nu;s) \ \text{ subject to } \ \|s\|_\infty \le S_b, \tag{2.4}$$

approximates the PFP near x_0. If $\Gamma(x_0) \subseteq \mathcal{A}_0$, then x_0 is a critical point of ϕ iff $s = 0$ is the global minimizer of $\psi(x_0, \mathcal{A}_0; \mu, \nu; s)$.

2.3 An L_∞-norm algorithm for SIP.

The previous section examined the L_∞QP in detail. In this section the remainder of the algorithm is discussed, and the algorithm is presented.

At each iterate $x^{(k)}$ the global (and other local) maximizers of the constraint function are found, and the approximating L_∞QP is constructed. The solution $s^{(k)}$ to the L_∞QP at $x^{(k)}$ is used to form the line (or arc) search. The algorithm either searches along the line $x^{(k)} + \alpha s^{(k)}$, or along the arc $x^{(k)} + \alpha s^{(k)} + \alpha^2 c^{(k)}$, where $c^{(k)}$ is a correction vector chosen to prevent the Maratos effect [63]. In either case α is chosen to be the first member of the sequence $1, \beta, \beta^2, \ldots$ to satisfy the sufficient descent criterion

$$\phi(x^{(k)}) - \phi\left(x^{(k)} + q^{(k)}(\alpha)\right) \geq \rho\alpha \left[\psi(x^{(k)}; 0) - \psi(x^{(k)}; s^{(k)})\right], \qquad (2.5)$$

where $0 < \rho < \frac{1}{2}$, $0 < \beta < 1$, and $q^{(k)}(\alpha)$ is either the line or arc step as given above. The next iterate is then $x^{(k)} + q^{(k)}(\alpha^{(k)})$. For convenience the line search is treated hereafter as an arc search with $c^{(k)} = 0$.

The penalty parameters are adjusted in order to satisfy (2.1), and (hopefully) to force the sequence of constraint violations $\{\theta^{(k)}\}$ to zero. The first requirement is met by forming lower semi-continuous estimates λ^*_{est} of the optimal Lagrange multipliers at each iterate and adjusting the penalty parameters accordingly. Such estimates may be calculated from the L_∞QP's solution, or by other methods [34].

Algorithm Summary:

1. Coarse approximations to all global maximizers, and as many local maximizers as practicable are found using a grid search or some other method, and then refined using a Quasi-Newton method. Call this set of points $\mathcal{A}^{(k)}$.

2. The approximating L_∞QP is formed, and its solution $s^{(k)}$ is calculated. If $\theta^{(k)}$ exceeds some specified positive value, then the capping constraint $\zeta(s^{(k)}) \leq \zeta(0)$ is imposed on the L_∞QP.

3. If $x^{(k)} + s^{(k)}$ does not satisfy the sufficient descent condition, calculate $c^{(k)}$, and perform the arc search.

4. Estimate the optimal Lagrange multipliers at the new iterate. If θ is less than some positive parameter $\theta_{\text{crossover}}$, and if $\mu \leq \kappa_1 \|\lambda^*_{\text{est}}\|_1$, then μ is increased

to $\kappa_2\|\lambda^*_{\text{est}}\|_1$, where $\kappa_2 > \kappa_1 > 1$ are fixed parameters. Related research [20] suggests that $\kappa_2 < 2$ may be desirable. If $\theta \geq \theta_{\text{crossover}}$, and $\mu + \nu\theta \leq \kappa_3\|\lambda^*_{\text{est}}\|_1$, then ν is adjusted to give $\mu + \nu\theta = \kappa_4\|\lambda^*_{\text{est}}\|_1$, where $\kappa_4 > \kappa_3 > 1$.

5. Update H using a quasi-Newton scheme whilst ensuring,

$$\exists\, \gamma > 0, \text{ such that } \forall x \in R^n - \{0\},\ \forall k,\ 0 < x^T H^{(k)} x \leq \gamma x^T x, \tag{2.6}$$

for example Powell's modified BFGS update [80] could be used.

6. If sufficient accuracy has not been attained, another iteration is begun.

The vector $c^{(k)}$ is essentially that of [63], and is determined as follows. The multi-local optimization subalgorithm is applied to $g(x^{(k)} + s^{(k)}, t)$, yielding the set $\mathcal{A}^{(k)}_{\text{soc}}$. Let $\mathcal{Q}^{(k)}$ denote the set of elements $t \in \mathcal{A}^{(k)}$ satisfying

$$\zeta(s^{(k)}) = g(x^{(k)}, t) + (s^{(k)})^T \nabla_x g(x^{(k)}, t).$$

Define $t_{\text{soc}}(w)$ to be the closest member of $\mathcal{A}^{(k)}_{\text{soc}}$ to w, for each $w \in \mathcal{Q}^{(k)}$. If $t_{\text{soc}}(w)$ is uniquely defined for every w, if t_{soc} is a one to one mapping, and if $\mathcal{Q}^{(k)}$ is non-empty, then $c^{(k)}$ is chosen as the vector of minimum length satisfying

$$[c^{(k)}]^T \nabla_x g(x^{(k)}, w) + g(x^{(k)} + s^{(k)}, t_{\text{soc}}(w)) = 0, \quad \forall w \in \mathcal{Q}^{(k)}. \tag{2.7}$$

Otherwise $c^{(k)} = 0$ is used. If the system (2.7) has no solution, or if $\|c^{(k)}\| \geq \|s^{(k)}\|$, then $c^{(k)}$ is reset to zero.

The vector $c^{(k)}$ is used to avoid the Maratos effect, and thereby ensure superlinear convergence on problems with the required continuity. Mayne and Polak [63] show if f and g are C^3, if x^* is a solution of the SIP at which strict complementarity, second order sufficiency conditions, and an implicit function theorem hold, and if the vectors $\{\nabla_x g(x^*, t) : t \in \Gamma(x^*)\}$ are linearly independent, then $x^{(k)} \to x^*$ implies $x^{(k)} \to x^*$ superlinearly. The vector $c^{(k)}$ is not required for convergence; the algorithm will converge for any choice of $c^{(k)}$ satisfying $\|c^{(k)}\| < \|s^{(k)}\|$, including $c^{(k)} = 0$. However, for problems which are sufficiently continuous, choosing $c^{(k)}$ as above ensures superlinear convergence will be obtained.

Assumption 2.8

(a) At each point x_0 at which the multi-local optimization subalgorithm is used it finds every point in $\Gamma(x_0)$.

(b) Also each $x_1 \in R^n$ has a neighbourhood $\mathcal{N}(x_1)$ such that if $x_0 \in \mathcal{N}(x_1)$ then the multi-local optimisation subalgorithm finds some extension (evaluated at x_0) of each member of $\Gamma(x_1)$. ♣

This assumption is an idealization; in practice only approximations to the points referred to in assumption 2.8 will be available. Assumption 2.8a is needed to ensure the locally approximating L_∞QP at x_0 will be sufficiently accurate.

Assumption 2.8b is needed to ensure the following scenario can not occur. Without assumption 2.8b, it is possible to have x_0 arbitrarily close to x_1 and not detect any extension of some $\tau \in \Gamma(x_1)$. This is possible because every extension of τ, evaluated at x_0, could be a local (non-global) maximiser of $g(x_0, .)$. Without any information about τ appearing in the L_∞QP at x_0, the search direction s_0 could be one in which $g(x, \tau)$ was increasing; effectively s_0 would point directly into the semi-infinite constraint. The step accepted in the arc search would be very much shorter than s_0 because of τ, leading to another point like x_0, and so on.

Actually assumptions 2.3 and 2.8 need only hold on the complement of some closed subset $\mathcal{E}$ of the interior of the feasible region. For any $x \in \mathcal{E}$, $\theta(x) \equiv 0$ on some neighbourhood of x. This renders assumption 2.3, and assumption 2.8a superfluous at x; after all, locally approximating 0 is not a difficult task! For assumption 2.8b, let x_c be any cluster point of the sequence of iterates generated by the algorithm. Then, for the case $x_c \in \mathcal{E}$, the extensions of $\Gamma(x_c)$ are irrelevant because $\theta(x) \equiv 0$ near x_c. Otherwise, x_c lies in the complement $\mathcal{E}^c$ of $\mathcal{E}$, in which case both parts of assumption 2.8 hold at x_c because $\mathcal{E}^c$ is open.

2.4 Convergence.

In this section the convergence properties of the algorithm are examined.

A requirement for convergence is that each arc search be a finite process. This is so if the descent condition (2.5) holds for all small positive α. If $s^{(k)}$ is zero, then $c^{(k)}$ is also zero, and (2.5) holds for all α. If $s^{(k)}$ is non-zero, then from assumption 2.8a and theorem 2.7, the sufficient descent condition (2.5) is equivalent to

$$\psi^{(k)}(0) - \psi^{(k)}(\alpha s^{(k)} + \alpha^2 c^{(k)}) + o(\alpha) \geq \rho\alpha \left[\psi^{(k)}(0) - \psi^{(k)}(s^{(k)})\right],$$

where $\psi(x^{(k)}, \mathcal{A}^{(k)}; s) = \psi^{(k)}(s)$ has been used. Now, because ψ is locally Lipschitz, the $c^{(k)}$ term can be removed from the argument of ψ to yield:

$$\psi^{(k)}(0) - \psi^{(k)}(\alpha s^{(k)}) + o(\alpha) \geq \rho\alpha\left[\psi^{(k)}(0) - \psi^{(k)}(s^{(k)})\right]. \tag{2.8}$$

The strict convexity of ψ ensures (2.8) holds for all small positive α.

Theorem 2.9 *Given:*

1. *All iterates generated by the algorithm lie in a bounded region of R^n.*

2. *Assumptions 2.3, 2.8, and the condition (2.6) hold.*

3. *The parameters μ and ν are only altered a finite number of times.*

Then every cluster point of the sequence of iterates $\{x^{(k)}\}$ generated by the algorithm is a critical point of $\phi(\mu, \nu; x)$, where μ and ν are the final values of these parameters.

PROOF. The proof is by contradiction. This is obtained by assuming some cluster point ($x_*^{(\infty)}$, say) of the sequence of iterates is not a critical point, and so deducing the existence of an iterate satisfying

$$\phi\left(x^{(k)} + q^{(k)}(\alpha^{(k)})\right) < \phi(x_*^{(\infty)}). \tag{2.9}$$

As the sequence $\{\phi^{(k)}\}$ is monotonically decreasing, and ϕ is continuous, a contradiction results. The existence of an iterate satisfying (2.9) is shown by using the following (loosely outlined) argument. If $x_*^{(\infty)}$ is not a critical point then any solution $s_*^{(\infty)}$ of some approximating L_∞QP at $x_*^{(\infty)}$ will be non-zero, and thus will be a direction of strict descent for ϕ. It is shown that the sequence of prospective steps $\{s^{(k)}\}$ converges to $s_*^{(\infty)}$, and the sequence $\{\alpha^{(k)}\}$ is bounded away from zero. Using continuity, arguments, (2.9) is then established.

Let $x_*^{(\infty)}$ be an arbitrary cluster point of $\{x^{(k)}\}$. Select a subsequence $\{x_*^{(k)}\}$ of $\{x^{(k)}\}$, generated after μ and ν assume their final values, and where the subsequences $\{x_*^{(k)}\}$, $\{H_*^{(k)}\}$, and $\{s_*^{(k)}\}$ converge to $x_*^{(\infty)}$, $H_*^{(\infty)}$, and $s_*^{(\infty)}$ respectively. Such a subsequence exists by item 1, requirement (2.6) and the bound on s in (2.4).

First it is shown that $s_*^{(\infty)}$ is a solution of a locally approximating L_∞QP at the point $x_*^{(\infty)}$. Let $\psi_*^{(k)}(s)$ and $\phi_*^{(k)}$ denote $\psi(x_*^{(k)}, \mathcal{A}(x_*^{(k)}); s)$ and $\phi(x_*^{(k)})$ respectively. Also let $\mathcal{A}_*^{(k)}$ denote $\mathcal{A}(x_*^{(k)})$. Define $\mathcal{A}_*^{(\infty)}$ as the set of all cluster points of sequences of the form $\{\xi_i\}_{i=1}^{\infty}$, where $\xi_i \in \mathcal{A}_*^{(i)}$, for all i. Clearly $\mathcal{A}_*^{(\infty)}$ is compact. Also, for all

k sufficiently large, $\mathcal{A}_*^{(\infty)}$ contains approximations to every member of $\mathcal{A}_*^{(k)}$. This follows from the definition of $\mathcal{A}_*^{(\infty)}$, and may be expressed in precise terms as:

$$\forall \epsilon > 0,\ \exists K,\ \forall k > K,\ \max_{t\in\mathcal{A}_*^{(k)}} \left[\min_{\tau\in\mathcal{A}_*^{(\infty)}} \|t-\tau\| \right] < \epsilon. \tag{2.10}$$

Let $\mathcal{S}_{GM}$ be the set of global minimizers of $\psi(x_*^{(\infty)}, \mathcal{A}_*^{(\infty)}; s)$, and let s_∞ be any element of $\mathcal{S}_{GM}$. Then

$$\forall s \notin \mathcal{S}_{GM},\ \exists t(s) \in \mathcal{A}_*^{(\infty)},\ \text{such that } \psi(x_*^{(\infty)}, \{t(s)\}; s) > \psi(x_*^{(\infty)}, \mathcal{A}_*^{(\infty)}; s_\infty). \tag{2.11}$$

Fixing s, let $\{x_\sharp^{(k)}\}$ be a subsequence of $\{x_*^{(k)}\}$, with each $\mathcal{A}_\sharp^{(k)}$ containing an approximation $t_\sharp^{(k)}$ to $t(s)$, such that $t_\sharp^{(k)} \to t(s)$ as $k \to \infty$. Then by (2.10) and (2.11), and also because $\mathcal{A}_\sharp^{(k)}$ contains an approximation to $t(s)$, because $\mathcal{A}_*^{(\infty)}$ contains approximations to each element of $\mathcal{A}_\sharp^{(k)}$, and as ψ is monotonically increasing in $\mathcal{A}$ under inclusion:

$$\lim_{k\to\infty} \psi(x_\sharp^{(k)}, \mathcal{A}_\sharp^{(k)}; s) \geq \psi(x_*^{(\infty)}, \{t(s)\}; s)$$

and

$$\lim_{k\to\infty} \psi(x_\sharp^{(k)}, \mathcal{A}_\sharp^{(k)}; s_\infty) \leq \psi(x_*^{(\infty)}, \mathcal{A}_*^{(\infty)}; s_\infty).$$

Hence $\{s_\sharp^{(k)}\}$ does not converge to s, and so $s_*^{(\infty)} \in \mathcal{S}_{GM}$.

It can be shown that $\psi(x_*^{(\infty)}, \mathcal{A}_*^{(\infty)}; s_*^{(\infty)})$ (hereafter $\psi_*^{(\infty)}(s_*^{(k)})$) is a cluster point of the sequence $\{\psi_*^{(k)}(s_*^{(k)})\}$. Now,

$$\begin{aligned}
\psi(x_*^{(k)}, \mathcal{A}_*^{(k)}; s_*^{(k)}) &= (s_*^{(k)})^T \left[\nabla f_*^{(k)} + \tfrac{1}{2} H_*^{(k)} s_*^{(k)}\right] \\
&\quad + \mu \max_{t\in\mathcal{A}_*^{(k)}} \left[g(x_*^{(k)}, t) + (s_*^{(k)})^T \nabla_x g(x_*^{(k)}, t)\right]_+ \\
&\quad + \tfrac{1}{2}\nu \left\{ \max_{t\in\mathcal{A}_*^{(k)}} \left[g(x_*^{(k)}, t) + (s_*^{(k)})^T \nabla_x g(x_*^{(k)}, t)\right]_+ \right\}^2 .
\end{aligned}$$

The convergence of the sequences $\{x_*^{(k)}\}$, $\{s_*^{(k)}\}$, and $\{H_*^{(k)}\}$ imply the first two terms converge. The definition of $\mathcal{A}_*^{(\infty)}$ implies that

$$\max_{t\in\mathcal{A}_*^{(\infty)}} \left[g(x_*^{(k)}, t) + (s_*^{(k)})^T \nabla_x g(x_*^{(k)}, t)\right]_+ = \limsup_{k\to\infty} \max_{t\in\mathcal{A}_*^{(k)}} \left[g(x_*^{(k)}, t) + (s_*^{(k)})^T \nabla_x g(x_*^{(k)}, t)\right]_+ .$$

Hence $\psi_*^{(\infty)}(s_*^{(\infty)})$ is a cluster point of $\{\psi_*^{(k)}(s_*^{(k)})\}$ as required. By replacing $\{x_*^{(k)}\}$ with a subsequence of itself if necessary, let $\{\psi_*^{(k)}(s_*^{(k)})\}$ converge to $\psi_*^{(\infty)}(s_*^{(\infty)})$.

Now $\alpha_*^{(k)}$ is chosen as the first member of the sequence $1, \beta, \beta^2, \ldots$ which satisfies the sufficient descent condition

$$\phi_*^{(k)} - \phi\left(x_*^{(k)} + q_*^{(k)}(\alpha)\right) \geq \rho\alpha \left[\psi_*^{(k)}(0) - \psi_*^{(k)}(s_*^{(k)})\right]. \tag{2.12}$$

By assumption 2.8, and by the definition of $\mathcal{A}_*^{(\infty)}$, it follows that $\Gamma(x_*^{(\infty)}) \subseteq \mathcal{A}_*^{(\infty)}$. Hence, because $\{x_*^{(k)}\}$ converges to $x_*^{(\infty)}$, $\{\psi_*^{(k)}(0)\}$ converges to $\psi_*^{(\infty)}(0)$. So, denoting terms which tend to zero as $k \to \infty$ by $o(1)$, (2.12) is equivalent to

$$\phi_*^{(k)} - \phi\left(x_*^{(k)} + q_*^{(k)}(\alpha)\right) \geq \rho\alpha\left[\psi_*^{(\infty)}(0) - \psi_*^{(\infty)}(s_*^{(\infty)})\right] + o(1), \qquad (2.13)$$

by the convergence of $\{\psi_*^{(k)}(s_*^{(\infty)})\}$, and $\{\psi_*^{(k)}(0)\}$ to $\psi_*^{(\infty)}(s_*^{(\infty)})$, and $\psi_*^{(\infty)}(0)$. Now, as $q_*^{(k)}(\alpha) = \alpha s_*^{(k)} + \alpha^2 c_*^{(k)}$, and as ϕ is locally Lipschitz, (2.13) is equivalent to

$$\phi_*^{(k)} - \phi\left(x_*^{(k)} + \alpha s_*^{(k)}\right) + O(\|c_*^{(k)}\|).o(\alpha) \geq \rho\alpha\left[\psi_*^{(\infty)}(0) - \psi_*^{(\infty)}(s_*^{(\infty)})\right] + o(1), \quad (2.14)$$

where the $c_*^{(k)}$ part of $q_*^{(k)}$ gives rise to the $o(\alpha)$ term. Now, because $\|c_*^{(k)}\| \leq \|s_*^{(k)}\|$, by the convergence of $\{s_*^{(k)}\}$ to $s_*^{(\infty)}$ and by the convergence of $\{x_*^{(k)}\}$ to $x_*^{(\infty)}$, (2.14) is equivalent to

$$\phi_*^{(\infty)} - \phi\left(x_*^{(\infty)} + \alpha s_*^{(\infty)}\right) + o(\alpha) \geq \rho\alpha\left[\psi_*^{(\infty)}(0) - \psi_*^{(\infty)}(s_*^{(\infty)})\right] + o(1). \qquad (2.15)$$

Now $\Gamma(x_*^{(\infty)}) \subseteq \mathcal{A}_*^{(\infty)}$, and so applying equation (2.3) to the left hand side of (2.15) implies if $\alpha_*^{(k)}$ satisfies

$$\psi_*^{(\infty)}(0) - \psi_*^{(\infty)}(\alpha s_*^{(\infty)}) \geq \rho\alpha\left[\psi_*^{(\infty)}(0) - \psi_*^{(\infty)}(s_*^{(\infty)})\right] + o(\alpha) + o(1), \qquad (2.16)$$

then it also satisfies (2.12). If $x_*^{(\infty)}$ is not a critical point, then for some u in R^n, the directional derivative of ϕ at $x_*^{(\infty)}$ in the direction u is strictly negative. This, and equation (2.3) imply

$$\psi_*^{(\infty)}(s_*^{(\infty)}) - \psi_*^{(\infty)}(0) = \kappa_* < 0.$$

Whence, by the convexity of ψ, and from (2.16) $\{\alpha_*^{(k)}\}$ has a strictly positive lower bound (α_{lower} say). Once again from equation (2.13), $\phi_*^{(k)} \to \phi_*^{(\infty)}$ implies

$$\phi\left(x_*^{(k)} + q_*^{(k)}(\alpha_*^{(k)})\right) \leq \phi_*^{(\infty)} + \rho\alpha_*^{(k)}\kappa_* + o(1).$$

Thus as $\alpha_*^{(k)} \geq \alpha_{\text{lower}}$ for all k, and as $\kappa_* < 0$, the existence of an iterate $x_*^{(k)}$ satisfying equation (2.9) is clear. ♣

2.5 Concluding Remarks.

Under fairly mild assumptions convergence to a set of critical points of the PFP has been shown. Each such critical point, if feasible, is also a solution of the SIP.

In contrast to algorithms based on the exact L_1 penalty function [21, 98], this algorithm does not depend upon the applicability of an implicit function theorem; assumption 2.3 is sufficient. Consequently, the minimum necessary degree of continuity of the semi-infinite constraint function is reduced from C^2 to C^1. This widens the class of problems which may be solved by this type of algorithm. Superlinear convergence is obtainable on problems with the requisite degree of continuity.

The L_∞ penalty function, unlike its L_1 counterpart in [21, 98], is continuous at all points in R^n. Hence one potential method of failure for algorithms based on the L_1 norm does not occur with the algorithm presented herein.

The L_∞ exact penalty function permits greater flexibility in the choice of the sets $\mathcal{A}^{(k)}$ at infeasible points than does the L_1 exact penalty function. With the L_1 penalty function, the only points in $\mathcal{A}^{(k)}$ at which $g(x^{(k)}, .)$ exceeds zero may be the local and global maximisers which take positive values, and these maximisers must feature in $\mathcal{A}^{(k)}$. In practice, a finite number of other points $t \in T$ may be included, but only if each such t appears in $\mathcal{A}^{(k)}$ *every* time $g(x^{(k)}, t)$ exceeds 0. In contrast, for the L_∞ penalty function, the active set $\mathcal{A}^{(k)}$ is only required to be a finite set containing sufficient points to satisfy assumption 2.8. Hence any finite subset of T may be added to $\mathcal{A}^{(k)}$, at whim, without negating the algorithm's convergence properties. This allows points at which local maximisers are expected to appear to be included in the active set. Such points could be detected in, for example, an arc search which rejects one or more points. These points may have been rejected due to the appearance of a (previously unforseen) global maximiser. Including the point at which the new global maximiser appears in the active set for the next iteration would usually enable the algorithm to skirt around that part of the semi-infinite constraint more efficiently.

Chapter 3

MULTI—LOCAL OPTIMISATION.

3.1 Introduction.

A major part of each iteration of the SIP algorithm is the determination of the global (and local) maximizers of the constraint function $g(x,t)$.

More precisely, the global maximizers, and a subset of the local maximizers of $g(x_0,t)$ with respect to t over the set T are sought, where x_0 is fixed. In this chapter, the terms local maximizer, and global maximizer will refer to the local and global maximizers of g as defined in the previous sentence, and the term stationary point will refer to a stationary point of $g(x_0,t)$ with respect to t on T. The number of global maximizers is finite by assumption 2.3, however this is not necessarily true of the local maximizers. This does not present an insurmountable difficulty, as not all the local maximizers are required. Specifically, the set of local maximizers found by the Multi-local Optimisation Subalgorithm (MOS) at the SIP iterate x_0 must contain one extension (evaluated at x_0) of every global maximizer of $g(x_1,t)$ over T, provided x_0 is sufficiently close to x_1. This requirement is formally given as assumption 2.8.

As only first derivatives are available, it is possible that points which the MOS subroutine identifies as local maximisers are not actually local maximisers, but only stationary points. The inclusion of such points in the list of local maximisers does not significantly affect the main SIP algorithm.

This problem is of a similar nature to the standard global optimisation problem of locating a global maximizer. The essential difference is that for any positive ϵ, the MOS must ultimately find a global maximizer (if one exists) over each member of a finite set of open balls of radius ϵ, where each such set of balls forms an open cover of T.

For convenience the term 'prominent local maximizer' will be used to refer to any local maximizer at which $g(x_0, .)$ takes a value close to the global maximal value of $g(x_0, t)$ over T (hereafter g_0).

The two cases $p = 1$ and $p > 1$ are considered separately. This chapter is concerned only with the case when the dimension of T is greater than one. The multi-local optimisation problem in one dimension is discussed in chapter 4. In this chapter T is assumed to be a Cartesian product of closed intervals. For convenience T is taken as the unit hypercube.

There are many approaches to the global optimisation problem. A good review of the subject is given by Törn and Žilinskas [96]. Most global optimisation algorithms consist of two sections: the first being an exploration phase in which the basic shape of the objective function is found, usually by calculating g at a number of test points in T. In the second section coarse estimate(s) of the local maximiser(s) are improved using a local search procedure. In practice these two phases may be divided into smaller parts. Also, the first phase need not be completed before the second is begun.

Other methods include those which make use of a Lipschitz constant [66, 23]. These algorithms typically use the test points and the Lipschitz constant to construct a function which is an upper bound for g on T. Regions of T in which the upper bound is lower than the highest known value of g can not contain a global maximiser, and future test points are not placed in such regions. For the multi-local optimisation problem, it is assumed that a Lipschitz constant L is *not* available. Even if such a constant were available, the dependence of $g(x, t)$ on x would (in most cases) mean that either L was known as a function of x, or the same value of L would be used for all x. In the latter case L would be unnecessarily large for most values of x: when this is the case algorithms of the form sketched above are generally very inefficient. Such methods will not be referred to further.

Several algorithms for global optimisation based on stochastic processes exist [68, 67, 96, 103]. These algorithms often use the stochastic process to model the basic features of the objective function on a global scale. Typically g, or a part of g is treated as the sample path of the stochastic process. Using the stochastic process, the values taken by g at the test points yield information about the values g is likely to take at other points in T. This allows future test points to be chosen based on information generated by past test points. Although they are very efficient in the number of function evaluations, the ancillary computational costs of these algorithms tend to be high; therefore they are usually only of use when the objective function is expensive to compute.

Probabilistic methods for global optimisation which do not model g with a stochastic process also exist. Rinnooy Kan, Timmer, and Boender [89, 90, 88, 91] use a Bayesian approach. Various unknowns are taken to be random variables, and are given *a priori* distributions. These unknowns are the number of strict local maximisers of g, and the Lebesgue measures of their regions of attraction. Here the regions of attraction are with respect to some unspecified local search algorithm. Specifically, for the number of local maximisers, each positive integer is assumed to be equally probable. For a fixed number of local maximisers $|\mathcal{A}|$, the relative sizes of the regions of attraction are assumed to have a uniform distribution on the $|\mathcal{A}|$-dimensional unit simplex.

The algorithm of Rinnooy Kan and Timmer [89, 90] calculates g at each of a set of randomly generated test points. To avoid unnecessary work, the lowest 80 per cent of these test points are rejected. The rest are grouped into clusters. Rinnooy Kan and Timmer give three different sets of criteria for forming the clusters. Only one, multi-level single linkage, will be sketched here. In multi-level single linkage, the clusters of test points are not formed explicitly. Rather each test point in a cluster is linked upwards to some other test point in the same cluster at which g takes a greater value. There is a maximum limit on the length a link can have. For the highest test point in each cluster no such upward link is possible, and so these test points remain unlinked. A local search is performed for each unlinked point, yielding the local maximisers. The number of test points, the number of local maximisers, and the *a priori* distributions on the number and sizes of the regions of attraction give the probability that a local maximiser has been missed.

Unfortunately the algorithm of Rinnooy Kan et al is not directly applicable to the MOS subproblem. Rinnooy Kan et al require the function being maximised to be C^2, whereas g is only guaranteed to be continuously differentiable.

3.2 Outline of the MOS Algorithm.

The basic approach of the multi-dimensional MOS is as follows. Firstly, the topography of g is explored by calculating the value g takes at each member of a set of test points. Using this information the test points are then grouped into clusters, where each cluster contains the test points in the 'region of attraction' of some (specific) local maximiser, and contains no other test points. A local search is then performed for each cluster. The initial point for each local search is chosen as the test point in the relevant cluster at which g takes the highest value. The term 'region of attraction' is used in a somewhat unconventional sense in this paragraph. It denotes a set of points in T for which a path of ascent exists from any such point to a specific local maximiser. These 'regions of attraction' are not unique, however they are required to be disjoint.

The basic structure of the algorithm is very similar to that of Rinnooy Kan et al. The difference is that no *a priori* assumptions are made about the number or relative sizes of the 'regions of attraction.' The second, counterbalancing, difference is in the way the links are constructed. The method of Rinnooy Kan et al places equal faith in each link, irrespective of the length of the link, and of the difference in the values g takes at each end of the link. In the MOS algorithm, for each link these two values are used to assess the reliability of that link. From these assessments, stopping conditions may be formed.

Each links reliability is assessed by modelling g by a stochastic process along the line segment between the endpoints of the link. For convenience, the line segment between the endpoints of the link will be simply referred to as the link. The stochastic process is a generalised Brownian motion process, and its probability distribution is completely determined by the length of the link, the average slope of g along the link, and by one other parameter (the variance parameter c) which contains information on how rapidly slope of g changes along the link. Using this generalised Brownian motion process, an estimate of the probability that g is strictly monotonic along the link is formed. This serves as an assessment of the link's reliability.

The basic form of the algorithm is as follows:

1. Until the required number of test points have been generated, perform each of these tasks in turn:

   (a) Generate two test points.

(b) Update the estimate $\bar{c}$ of c.

2. For each test point y, search through the other test points to find the test point t which satisfies $g(x, t) > g(x, y)$, and has the most reliable link from y to t. Record that y is linked upward to t.

3. For each test point y which is not linked upward to a higher test point by a sufficiently reliable link, do a local search with y as the initial point.

In the next section methods of generating test points are discussed. In section 4 the details of the generalised Brownian motion process are examined, along with the method of assessing each link's reliability.

3.3 Halton sequences.

The sequence of test points $\{y_i\}_{i=1}^{\infty}$ in T at which g is calculated in the first stage of the exploration phase may be generated in many ways. Often the test points are taken as the intersections of a rectangular grid [50, 53], randomly generated [88, 89, 90, 91], or generated by a quasi-random sequence [70].

A grid, whilst covering the set T evenly, can be a very inefficient way of exploring g when g is nearly constant along some directions (and in particular those directions parallel to the axes). When g is of this form the global optimisation subproblem is effectively over the projection of T onto some subspace of R^p. The projection of the grid onto this subspace may map many different grid points into very close proximity with one another. Each of these points then yields the same information about g, and so the efficiency of the grid is reduced markedly [92].

Randomly generated sets of points avoid this effect, however in low dimensions (≤ 6) they cover the space less uniformly than rectangular grids do [3]. The uniformity of the points $\{y_i\}$ is especially important. The MOS must solve a succession of multi-local optimisation problems, where the objective function g (only) is different for each problem. In the k^{th} iteration of the SIP algorithm an approximation to $f(x)$ is minimised, where linear approximations with respect to x of $g(x, t)$ are constrained to be less than zero for all t in the finite set $\mathcal{A}^{(k)}$ consisting of global maximizers, and other local maximizers of $g(x^{(k)}, t)$ found by the MOS. The two aims of minimizing f, and of keeping $g \leq 0$ on T will almost always be in conflict with each other. If any large gaps exist in the pattern of test points used in each iteration of the MOS it is quite possible that a global maximal value exceeding zero will appear in that gap, and remain undetected by the MOS.

Another alternative to randomly generated points is to use a quasi-random sequence, such as the Halton sequences [45], or the LP_τ sequences of Sobol [92]. These sequences cover T more uniformly than those generated randomly (see for example [70]). Here the non-uniformity of the sequence $\{y_i\}$ is measured by its discrepancy, which is defined as follows:

$$D_N = \sup_{J \in \mathcal{J}} \left| \frac{1}{N} \sum_{i=1}^{N} \chi_J(y_i) - \frac{|J|}{|T|} \right|, \tag{3.1}$$

where $\mathcal{J}$ is the class of subsets of T which are Cartesian products of intervals along each of the edges of T, where χ_J is the characteristic function of the set J, and where $|J|$ is the Lebesgue measure of J in R^p.

The expected discrepancy of a random sequence of N points based on a uniform distribution is $O(N^{-\frac{1}{2}}\sqrt{\log\log(N)})$ with probability one, whereas the discrepancies for the Halton sequences, and the LP_τ sequences are both $O(N^{-1}(\log N)^p)$. In contrast, the discrepancy of a $j \times j \times \ldots \times j$ grid in the p dimensional unit cube is at least j^{-1}. Using $N = j^p$, this means the discrepancy of this grid is at least $\sqrt[p]{N^{-1}}$. This compares badly against randomly generated points and Halton sequences. The Halton and LP_τ sequences are also less prone to the projection deficiency than rectangular grids; in particular no two different points from either type of sequence have any co-ordinate values the same. Halton sequences are particularly easy to define for any finite dimension, and extra points may be added to them without difficulty. For these reasons Halton sequences are used to effect the initial exploration of the objective function.

Halton sequences are defined componentwise. First p pairwise coprime numbers $\pi_1, \ldots, \pi_p$ are chosen — frequently the smallest p primes are used. The i^{th} component $y_{k(i)}$ of the k^{th} member of the sequence is generated by writing k in base π_i as

$$k = a_m a_{m-1} \ldots a_1 a_0. \quad \text{base } \pi_i,$$

and then placing the digits $a_0, \ldots, a_m$ in reverse order on the opposite side of the radix:

$$y_{k(i)} = 0.a_0 a_1 \ldots a_m \quad \text{base } \pi_i.$$

This may be generalised slightly, and rewritten as

$$y_{k(i)} = \sum_{r=0}^{\infty} \left\{ \left[\frac{k\delta_i}{\pi_i^r} \right] \bmod \pi_i \right\} . \frac{1}{\pi_i^{r+1}}, \tag{3.2}$$

where $[x]$ denotes the greatest integer not larger than x, and where δ_i is a positive integer coprime with π_i, and satisfying $\delta_i < \pi_i$. Using $\delta_i = 1$ for all i yields the usual form of the Halton sequence as above.

From the definition it is clear that if a Halton sequence in p dimensions is projected onto a hyperplane perpendicular to some axis, then the projected sequence is also a Halton sequence. If T is a hyper-rectangle rather than a hyper-cube, then this may be advantageous even if the regions of attraction of the local maximisers of g are approximately isotropic. If T is much shorter along some axes compared to others, then the exploration of g may, in essence, be a problem over some subspace of R^p. Once again a grid with an equal number of layers of points along each axis may be very inefficient. The efficiency of the grid may be improved by reducing the number of layers along some of the axes, however with a general function it will be far from obvious along which axes (if any) these reductions can be made safely. With Halton sequences this sort of difficulty does not arise.

Proposition 3.1 *The Halton sequence generated as per (3.2) is dense in T.*

PROOF. Without loss of generality, take T to be the unit hypercube in this proof. Let $\mathcal{H}$ denote the set of test points generated by the Halton sequence, and let $\tau \in T$. Use $B_\epsilon(\tau)$ to denote the open ball of radius ϵ centred on τ. There exists a hyper-rectangle

$$\left[\frac{\alpha_1}{\pi_1^{m_1}}, \frac{\alpha_1 + 1}{\pi_1^{m_1}} \right] \times \ldots \times \left[\frac{\alpha_p}{\pi_p^{m_p}}, \frac{\alpha_p + 1}{\pi_p^{m_p}} \right] \subset B_\epsilon(\tau),$$

Now, for each i,

$$\exists \beta_i \in 0, \ldots, \pi_i^{m_i} - 1 \text{ such that } y_{k(i)} \in \left[\frac{\alpha_i}{\pi_i^{m_i}}, \frac{\alpha_i + 1}{\pi_i^{m_i}} \right] \quad \forall k \equiv \beta_i \bmod \pi_i^{m_i}.$$

Hence $y_k \in B_\epsilon(\tau)$ if

$$k \equiv \beta_i \bmod \pi_i^{m_i} \ \forall i = 1, \ldots, p. \tag{3.3}$$

The parameters $\pi_i^{m_i}$ are mutually coprime, and so the existence of a k satisfying (3.3) is guaranteed by the Chinese remainder theorem. Thus $\mathcal{H} \cap B_\epsilon(\tau)$ is non-empty for all positive ϵ, and all $\tau \in T$. Hence $\mathcal{H}$ is dense in T. ♣

In practice only every second test point is chosen from the Halton sequence. This is a consequence of estimating the parameter $\overline{c}$. This requires co-linear triples of test points which have the central point midway between the other two. With each test point generated using the Halton sequence, a second test point is generated to form an equi-spaced co-linear triple with a third, existing test point.

In theory the Halton sequences are very uniform, as judged using the asymptotic discrepancy (3.1). However, as p is increased the number of test points required before (3.1) really becomes valid increases quickly. This imposes a practical upper limit on p of the order of 4 to 6 if reasonable computation times are required.

3.4 The Stochastic Process.

The stochastic process is used to model the behaviour of g along prospective links between test points. Specifically, under suitable assumptions it is shown that between any pair of test points t_0, t_1 sufficiently close together, the component of $\nabla_t g$ parallel to $t_1 - t_0$ evaluated along the line segment between t_0 and t_1 may be modelled as a stochastic process based on a Brownian motion process (BMP). Using this an (under)estimate of the probability that there exists a continuous path of ascent from t_0 to t_1 lying entirely in T is formed. This is done by estimating the probability that the line segment between t_0 and t_1 is such a path based on the stochastic process model of $\nabla_t g$. It is shown that this estimate is a monotonically increasing function of a parameter known hereafter as the linkage parameter. It is not necessary to assume that T is a hyper-rectangle in order to develop the stochastic process model. However, if T is not convex, then the line segment between two test points need not lie totally in T. In this case the stochastic model developed in this section is no longer directly applicable.

Definition 3.2 *For purposes herein, a Brownian Motion Process (BMP) is taken to be a real function $B(\xi; \omega, \sigma)$ defined on the Cartesian product of the interval $[0, \ell]$, and the probability space $(\Omega, \mathcal{B}, P)$. Here $\mathcal{B}$ is a σ–algebra on Ω, and P is a probability measure with respect to $\mathcal{B}$. The BMP has a root $\xi \in [0, \ell]$ at which $B(\xi; \omega, \xi) = m$ for all $\omega \in \Omega$. For each $\sigma_0 \in [0, \ell]$, $B(\xi; \omega, \sigma_0)$ is a Gaussian random variable on $(\Omega, \mathcal{B}, P)$ with mean m, and variance $c|\sigma_0 - \xi|$. The non–negative constant c is referred to as the variance constant. For any $\sigma_1, \sigma_2 \in [0, \ell]$ the covariance function* *of B is defined as follows:*

$$cov_B(\sigma_1,\sigma_2) = \begin{cases} c.\min(\sigma_1-\xi,\sigma_2-\xi) & \textit{if}\, \sigma_1,\sigma_2 \geq \xi, \\ c.\min(\xi-\sigma_1,\xi-\sigma_2) & \textit{if}\, \sigma_1,\sigma_2 \leq \xi, \\ 0 & \textit{otherwise}. \end{cases} \quad \clubsuit$$

It is shown in [102] that any Gaussian stochastic process is completely determined (up to equivalence) by its mean and covariance functions. For a BMP as above, these are in turn determined by the three constants m, c, and ξ.

It is useful to summarize the information on $\nabla_t g$ readily available to the MOS.

1. $\nabla_t g$ is continuous, but not necessarily differentiable.

2. For any two points t_0, t_1 at which $g(x_0,.)$ has been calculated, the average value $(t_1-t_0)^T\nabla_t g$ takes on the line segment between t_0 and t_1 is known.

3. At each known stationary point $\nabla_t g = 0$.

The stochastic process model is only used between pairs of points which are close neighbours. This term is defined as follows.

Definition 3.3 *Let $\mathcal{T}_N$ be the set of the first N test points used. Two test points t_0, t_1 are said to be close neighbours in the set of test points $\mathcal{T}_N$ if and only if*

$$||t_1 - t_0|| \leq \ell_{max}(N). \tag{3.4}$$

The function $\ell_{max}(N)$ on the positive integers is required to be positive, and monotonically decreasing. It is also required to satisfy

$$\lim_{N\to\infty} \ell_{max}(N) = 0. \quad \clubsuit$$

The choice of the function $\ell_{\max}(N)$ is a trade-off between reliability and efficiency. If $\ell_{\max}$ is too large the algorithm may fail to distinguish between different peaks. If it is too small, then many unnecessary line searches may be performed. A good choice of $\ell_{\max}$ will depend on how the test points are generated.

Hence, from this, and the known information about $\nabla_t g$ (or lack thereof) as listed above, the following assumption is made.

Assumption 3.4 *Let t_1, t_0 be close neighbours in $\mathcal{T}_N$, let $\ell = ||t_1 - t_0||$, and let $u = \frac{t_1-t_0}{||t_1-t_0||}$, then*

$$\forall \sigma \in [0,\ell], \quad u^T\nabla_t g(x_0, t_0 + \sigma u) \tag{3.5}$$

is independent of the values g takes at all points in $\mathcal{T}_N - \{t_1, t_0\}$. $\clubsuit$

Similar simplifying assumptions occur in other works on global optimisation which use stochastic processes to model the objective function [89, 68]. Also it should be noted that no restriction is placed on the components of $\nabla_t g$ perpendicular to u, so test points slightly off the line joining t_0 and t_1 may take values wildly different from those values predicted by the BMP along the line segment between t_0 and t_1.

Now, as $\nabla_t g$ is only guaranteed to be continuous, and in the absence of other information on $\nabla_t g$, the following assumption is made.

Assumption 3.5 *For any pair of close neighbours $t_0, t_1 \in \mathcal{T}_N$, neither of which is a known stationary point, and for all positive integers k:*

$$u^T[\nabla_t g(x_0, t_0 + \frac{i+1}{k}\ell u) - \nabla_t g(x_0, t_0 + \frac{i}{k}\ell u)], \qquad i \in 0, \ldots, k-1 \tag{3.6}$$

are independent identically distributed random variables, where $u = \frac{t_1 - t_0}{\|t_1 - t_0\|}$, and $\ell = \|t_1 - t_0\|$ as before. ♣

For convenience, define h by

$$-h = g(x_0, t_1) - g(x_0, t_0) \leq 0. \tag{3.7}$$

Also, let $A_o(\omega, \sigma)$ be the stochastic process which is used to model $u^T \nabla_t g(x_0, t_1 + \sigma u)$ on $[0, \ell]$.

Theorem 3.6 *Any stochastic process $S(\omega, \sigma)$ on $[0, \ell]$ which satisfies assumption (3.5) is equivalent to the sum of a Brownian motion process $B(\omega, \sigma)$ rooted at $\sigma = 0$, and a dependent random process $D_o(\omega)$. All sample paths of D_o are constant with respect to σ.*

PROOF. The existence of a stochastic process satisfying assumption (3.5) is clear: one such process is a Brownian motion process rooted at 0. Let $S(\omega, \sigma)$ be a stochastic process which satisfies assumption (3.5), and let

$$B(\omega, \sigma) = S(\omega, \sigma) - S(\omega, 0).$$

In effect $S(\omega, 0)$ is the random process D_o.

The compactness of T implies $\nabla_t g$ is uniformly continuous on T, and hence $\|\nabla_t g\|$ is bounded on T. Thus the random variables (3.6) have finite variances for all i and k. In the limit $k \to \infty$ the central limit theorem is applicable, and

$B(\sigma_2) - B(\sigma_1)$ is a normally distributed random variable with mean $-\frac{h}{\ell}|\sigma_2 - \sigma_1|$, and variance equal to $c|\sigma_2 - \sigma_1|$, where $c \geq 0$ is some constant.

Let $\mathrm{cov}(\sigma_2, \sigma_1)$ be the covariance between $u^T \nabla_t g(x_0, t_0 + \sigma_2 u)$ and $u^T \nabla_t g(x_0, t_0 + \sigma_1 u)$. Without loss of generality, assume $\sigma_2 \geq \sigma_1 \geq 0$. Clearly

$$B(\sigma_2) = [B(\sigma_2) - B(\sigma_1)] + [B(\sigma_1) - B(0)]$$

$$\text{and} \quad B(\sigma_1) = [B(\sigma_1) - B(0)],$$

where $B(\sigma_2) - B(\sigma_1)$ and $B(\sigma_1) - B(0)$ are independent normally distributed random variables with variances $c|\sigma_2 - \sigma_1|$, and $c|\sigma_1 - 0|$ respectively. Hence

$$\mathrm{cov}(\sigma_2, \sigma_1) = c. \min\{\sigma_2, \sigma_1\} \quad \sigma_2, \sigma_1 \geq 0.$$

By theorem 2.1 in [102] any Gaussian process is completely determined by its mean and covariance functions. Whence $u^T \nabla_t g(x_0, t_0 + \sigma u)$, $\sigma \in [0, \ell]$ is modelled by the sum of the random variable $S(\omega, 0)$, and a Brownian Motion Process with its root at 0. ♣

Any stochastic process $S_o = B + D_B$ which usefully models $\nabla_t g$, must satisfy the following condition:

$$\int_{\sigma=0}^{\ell} B(\omega, \sigma) + D_B(\omega)\, d\sigma = -h = g(x_0, t_1) - g(x_0, t_0) \;\; \forall \omega \in \Omega. \tag{3.8}$$

This is easily achieved by defining D_B as

$$D_B(\omega) = -\frac{h}{\ell} - \frac{1}{\ell} \int_{\sigma=0}^{\ell} B(\omega, \sigma)\, d\sigma, \;\; \forall \omega \in \Omega.$$

Because B is a BMP, it follows that D_B is a Gaussian random variable. Up to equivalence, the process S_o is unique. This expression for S_o is not particularly useful. The dependence of D_B and B makes working with them difficult. A more fruitful approach is to consider $A + D$, where $D(\omega)$ and $A(\omega, \sigma)$ are *independent* Gaussian random processes, and A is required to satisfy

$$\int_{\sigma=0}^{\ell} A(\omega, \sigma)\, d\sigma = 0 \;\; \forall \omega \in \Omega.$$

Because A and D are independent, it is easy to ensure that

$$\int_0^{\ell} A + D\, d\sigma = -h.$$

A process of this type is constructed next.

Definition 3.7 *The process $F(\omega, \sigma)$ is chosen to be a Gaussian process on $\sigma \in [0, \ell]$. The mean of F is zero. Its covariance function is defined as follows:*

$$cov_F(a, b) = \frac{1}{\ell} \int_0^\ell cov_{B(\xi;.)}(a, b)\, d\xi, \quad \forall a, b \in [0, \ell]. \tag{3.9}$$

Here $B(\xi; .)$ is a Brownian motion process with zero mean, variance constant c, and rooted at ξ. By the Kolmogorov extension theorem [102], this defines $F(\omega, \sigma)$ up to equivalence. ♣

The integral (3.9) can be viewed (loosely!) as smearing out a family of Brownian motion processes along $[0, \ell]$. The process F will be referred to as the Integrated Brownian Motion Process (IBMP).

Proposition 3.8 *The IBMP $F(\omega, \sigma)$ as defined above is equivalent to the stochastic process:*

$$\sqrt{\frac{c}{2\ell}} \left[B_0(\omega, \sigma^2) + B_\ell(\omega, (\ell - \sigma)^2)\right], \tag{3.10}$$

where B_0 and B_ℓ are two independent zero mean, unit variance Brownian motion processes rooted at 0.

PROOF. Let $a, b \in [0, \ell]$, and assume for the moment that $a \leq b$. Then

$$\text{cov}_F(a, b) = \frac{1}{\ell} \int_{\xi=0}^{\ell} \text{cov}_{B(\xi;.)}(a, b)\, d\xi,$$

where

$$\text{cov}_{B(\xi;.)}(a, b) = \begin{cases} c.\min(|a - \xi|, |b - \xi|) & \text{if } (a - \xi)(b - \xi) \geq 0 \\ 0 & \text{otherwise.} \end{cases}$$

Hence

$$\begin{aligned} \text{cov}_F(a, b) &= \frac{1}{\ell} \left[\int_0^a c(a - \xi)\, d\xi + \int_a^b 0\, d\xi + \int_b^\ell c(\xi - b)\, d\xi\right] \\ &= \frac{c}{2\ell}[a^2 + \ell^2 - 2b\ell + b^2]. \end{aligned}$$

For a general a, and b,

$$\text{cov}_F(a, b) = \frac{c}{2\ell} \left[\min(a^2, b^2) + \min((\ell - a)^2, (\ell - b)^2)\right].$$

This is identical to the covariance function of

$$\sqrt{\frac{c}{2\ell}} \left[B_0(\sigma^2) + B_\ell((\ell - \sigma)^2)\right].$$

The process (3.10), and the IBMP are both Gaussian, and possess identical means and covariance functions, and thus are equivalent processes (cf. theorem 2.1 of [102]). ♣

Proposition 3.9 *The stochastic process $F(\omega, \sigma)$ satisfies assumption (3.5).*

PROOF. Let $\sigma_1, \sigma_2, \sigma_3, \sigma_4 \in [0, \ell]$, with $\sigma_1 \leq \sigma_2 \leq \sigma_3 \leq \sigma_4$. The independence of the random variables $F(\sigma_4) - F(\sigma_3)$ and $F(\sigma_2) - F(\sigma_1)$ follows immediately from the fact the the increments of both component Brownian motion processes are independent.

Let $a, b \in [0, \ell]$, with $a \leq b$. The variances of the increments in the component Brownian motion processes over the interval $[a, b]$ are respectively

$$\frac{c}{2\ell}(b^2 - a^2) \quad \text{for } B_0, \quad \text{and} \quad \frac{c}{2\ell}\left((\ell - a)^2 - (\ell - b)^2\right) \quad \text{for } B_\ell.$$

Hence the variance of $F(b) - F(a)$ is $c(b - a)$. This depends only on the length of the interval. Hence, provided $\sigma_4 - \sigma_3 = \sigma_2 - \sigma_1$, $F(\omega, \sigma_4) - F(\omega, \sigma_3)$ and $F(\omega, \sigma_2) - F(\omega, \sigma_1)$ have identical variances. As the mean of F is zero, the result follows immediately. ♣

As yet the condition (3.8) remains unaddressed. It is shown next that the IBMP may be split into two *independent* Gaussian processes: the first process has all sample paths constants, and for the second the integral of every sample path from 0 to ℓ is zero. These two processes are referred to as the DC and AC parts of the IBMP respectively.

Proposition 3.10 *Let the IBMP $F(\omega, \sigma)$ be defined on the probability space $(\Omega, \mathcal{B}, P)$, where P is a probability measure with respect to the σ-algebra $\mathcal{B}$ on the set Ω. Then*

$$F(\omega, \sigma) = A(\omega, \sigma) + D(\omega, \sigma),$$

where the sample paths $D(\omega, .)$ of D are constants $\forall \omega \in \Omega$, and

$$\int_0^\ell A(\omega, \sigma)\, d\sigma = 0 \quad \forall \omega \in \Omega.$$

PROOF. Using $w_0(t)$ to denote an arbitrary integrable function, the operator

$$T[w_0(t)](s) = \int_0^\ell \mathrm{cov}_F(s, t) w_0(t)\, dt \tag{3.11}$$

is clearly compact, and positive semi-definite. On noting F is Gaussian, Varberg's theorem (theorem 19.5 in [102]) yields

$$F(\omega, \sigma) = \sum_{i=0}^{\infty} \sqrt{\lambda_i}\, Z_i(\omega) e_i(\sigma),$$

where the Z_i are independent Gaussian random variables with zero means, and unit variances. The λ_i, and $e_i(t)$ are the eigenvalues and corresponding normalized eigenfunctions of T. The eigenfunctions of T form an orthonormal set with respect to the inner product

$$<f,g> = \int_0^\ell f(s)g(s)\,ds.$$

To prove F may be split into independent AC and DC parts it suffices to show that $\frac{1}{\sqrt{\ell}}$ is an eigenfunction of T. Now

$$\int_0^\ell \text{cov}_F(s,t).\frac{1}{\sqrt{\ell}}\,ds$$

$$= \frac{c}{2}\ell^{-\frac{3}{2}}\left[\int_0^\ell \left[\min(s^2,t^2) + \min((\ell-s)^2,(\ell-t)^2)\right]\,ds\right]$$

$$= \frac{c}{2}\ell^{-\frac{3}{2}}\left[\int_0^t s^2\,ds + \int_t^\ell t^2\,ds + \int_0^t (\ell-t)^2\,ds + \int_t^\ell (\ell-s)^2\,ds\right] = \frac{\ell^2 c}{6}\frac{1}{\sqrt{\ell}}.$$

It may appear a little surprising that this integral is independent of t, however it is extremely propitious. Using $e_0(t) = \sqrt{\frac{1}{\ell}}$, it follows that

$$D(\omega,\sigma) = \sqrt{\frac{c\ell}{6}}Z_0(\omega) \quad \text{and} \quad A(\omega,\sigma) = \sum_{i=1}^{\infty}\sqrt{\lambda_i}Z_i(\omega)e_i(\sigma), \tag{3.12}$$

$$\text{where} \quad \int_0^\ell e_i(\sigma)\,d\sigma = 0 \quad \forall i \in \{1,2,3,\ldots\}. \quad \clubsuit$$

The probability that the stochastic process model predicts that g is not strictly decreasing along the line segment from t_0 to t_1 is the probability that F is non-negative at some point in $[0,\ell]$. It follows that this is equal to the probability that the maximum value $A_{\max}$ of $A(\sigma)$ is at least $\frac{h}{\ell}$, because the definition of h (3.7) fixes $D(\omega) = -h\ell^{-1}$, but does not affect $A(\omega,\sigma)$. The next proposition gives an upper bound on this probability.

Proposition 3.11 *Given the IBMP $F(\omega,\sigma)$ satisfies*

$$\int_0^\ell F(\sigma)\,d\sigma = -h, \tag{3.13}$$

the probability that the maximal value A_{max} of $A(\omega,\sigma)$ on the interval $\sigma \in [0,\ell]$ exceeds $\frac{h}{\ell}$ is bounded above as follows:

$$P(F_{max} \geq 0) \leq \frac{4}{\sqrt{2\pi}}\min\left\{\int_{\xi=\sqrt{\frac{h^2}{\ell^3 c}}}^{\infty}\sqrt{\frac{6}{5}}e^{-\frac{3}{5}\xi^2}d\xi,\ \int_{\xi=\sqrt{\frac{h^2}{\ell^3 c}}}^{\infty}2\sqrt{\frac{24}{11}}e^{-\frac{12}{11}\xi^2}\,d\xi\right\}.$$

PROOF. Rather than address $P(F_{\max} \geq 0)$ directly, $P(A_{\max} \geq \frac{h}{\ell})$ is looked at. Proposition 3.10 implies $A_{\max}$ is independent of h. A Gaussian upper bound on $P(F_{\max} \geq \frac{h}{\ell})$ which does not take into account (3.13) is constructed. It is shown that this bound is formed from convoluting two independent Gaussian quantities: $P(D = \gamma)$ and an upper bound on $P(A_{\max} = \frac{h}{\ell} - \gamma)$.

The representation (3.10) of the IBMP is used, where for convenience, the two independent BMPs B_0 and B_ℓ are defined collectively on the probability space $(\Omega, \mathcal{B}, P)$. Two upper bounds, $J(\omega)$ and $K(\omega)$, on

$$F_{\max}(\omega) = \max_{\sigma \in [0,\ell]} F(\omega, \sigma)$$

are constructed. These bounds are pointwise with respect to ω (ie path–by–path). For the first bound, $J(\omega)$ is defined by

$$F_{\max}(\omega) \leq \sqrt{\frac{c}{2\ell}} \left[\max_{\sigma \in [0,\ell]} B_0(\omega, \sigma^2) + \max_{\sigma \in [0,\ell]} B_\ell(\omega, (\ell - \sigma)^2) \right] = J(\omega) \ \ \forall \omega \in \Omega.$$

An upper bound on $P(J(\omega) \geq \gamma)$ as a function of γ is now constructed. By the reflexion principle [102], for a BMP $B(\sigma)$ of zero mean, and rooted at $\sigma = 0$,

$$P(\max_{\sigma \in [0,\ell]} B(\sigma) \geq \gamma) = 2P(B(\ell) \geq \gamma) \quad \forall \gamma \geq 0.$$

Hence, using $\star$ to denote convolution operator

$$(f \star g)(\xi) = \int_{-\infty}^{\infty} f(\gamma) g(\xi - \gamma) \, d\gamma,$$

and letting $U(\gamma)$ denote the Heaviside step function,

$$\begin{aligned} P(J(\omega) = \gamma) &= \frac{4}{2\pi} \frac{2}{c\ell} \left[U(\gamma) e^{-\frac{1}{c\ell}\gamma^2} \star U(\gamma) e^{-\frac{1}{c\ell}\gamma^2} \right] \\ &\leq \frac{4}{2\pi} \frac{2}{c\ell} \left[e^{-\frac{1}{c\ell}\gamma^2} \star e^{-\frac{1}{c\ell}\gamma^2} \right] \\ &\leq \frac{4}{\sqrt{2\pi c\ell}} e^{-\frac{1}{2c\ell}\gamma^2}. \end{aligned}$$

For the second bound $K(\omega)$, the left and right halves of $[0, \ell]$ are considered separately. Using

$$L(\omega) = \max_{\sigma \in [0,\frac{\ell}{2}]} B_0(\omega, \sigma^2) + \max_{\sigma \in [0,\ell]} B_\ell(\omega, (\ell - \sigma)^2),$$

and

$$R(\omega) = \max_{\sigma \in [0,\ell]} B_0(\omega, \sigma^2) + \max_{\sigma \in [0,\frac{\ell}{2}]} B_\ell(\omega, (\ell - \sigma)^2),$$

it follows that

$$F_{\max}(\omega) \leq \sqrt{\frac{c}{2\ell}} \max\{L(\omega), R(\omega)\} = K(\omega),$$

which defines $K(\omega)$. Clearly

$$P(K(\omega) = \gamma) \leq P(L(\omega) = \gamma \text{ and } R(\omega) \leq \gamma) + P(L(\omega) \leq \gamma \text{ and } R(\omega) = \gamma)$$

$$\leq P(L(\omega) = \gamma) + P(R(\omega) = \gamma).$$

Now

$$\begin{aligned} P(L(\omega) = \gamma) &= \frac{4}{2\pi}\sqrt{\frac{2}{c\ell}\cdot\frac{8}{c\ell}}\left[U(\gamma)e^{-\frac{1}{c\ell}\gamma^2} \star U(\gamma)e^{-\frac{4}{c\ell}\gamma^2}\right] \\ &\leq \frac{4}{2\pi}\cdot\frac{4}{c\ell}\left[e^{-\frac{1}{c\ell}\gamma^2} \star e^{-\frac{4}{c\ell}\gamma^2}\right] \\ &= \frac{4}{\sqrt{2\pi}}\sqrt{\frac{8}{5c\ell}}e^{-\frac{4}{5c\ell}\gamma^2} \end{aligned}$$

with an identical estimate for $P(R(\omega) = \gamma)$. Hence

$$P(K(\omega) = \gamma) \leq \frac{8}{\sqrt{2\pi}}\sqrt{\frac{8}{5c\ell}}e^{-\frac{4}{5c\ell}\gamma^2} = \overline{P}(K(\omega) = \gamma). \tag{3.14}$$

It is not necessarily the case that

$$P(F_{\max}(\omega) = \gamma) \leq P(K(\omega) = \gamma),$$

however

$$P(F_{\max}(\omega) \geq \gamma) \leq P(K(\omega) \geq \gamma) \leq \overline{P}(K(\omega) \geq \gamma)$$

is always valid. A similar statement holds for $J(\omega)$.

Consider the two bounds $J(\omega)$ and $K(\omega)$ separately. For $K(\omega)$,

$$K(\omega) \geq F_{\max}(\omega) = D(\omega) + A_{\max}(\omega) \quad \forall \omega \in \Omega.$$

Define $H(\omega)$ as follows

$$H(\omega) = K(\omega) - D(\omega) \geq A_{\max}(\omega) \quad \forall \omega \in \Omega. \tag{3.15}$$

The probability distribution for D, and the upper bound $\overline{P}(K = \gamma)$ for $P(K = \gamma)$ in (3.14) are both Gaussian functions centred on 0, and hence their joint distribution is of the form

$$\kappa_1 e^{-\frac{1}{2}[\kappa_2 K^2 + 2\kappa_3 KD + \kappa_4 D^2]},$$

or equivalently,

$$\kappa_5 e^{-\frac{1}{2}[\kappa_6 H^2 + 2\kappa_7 HD + \kappa_8 D^2]}$$

where $\kappa_1, \ldots, \kappa_8$ are constants. To show $\kappa_7 = 0$, Restrict ω to the subset of Ω on which $A_{\max}(\omega) = \eta$, where η is non-negative, but otherwise arbitrary. A and D are independent, and so D may be chosen freely. Assume $\kappa_7 \neq 0$. On fixing $D = D_0$, an upper bound $\overline{P}(H = H_0)$ on $P(H = H_0)$ is

$$4\sqrt{\frac{\kappa_6}{2\pi}} e^{-\frac{1}{2}\kappa_6 (H_0 + \frac{\kappa_7}{\kappa_6} D_0)^2}.$$

This implies that for any fixed H_0,

$$\overline{P}(H \geq H_0) \to 0, \quad \text{as} \quad D_0 \to \operatorname{sign}(\kappa_7)\infty.$$

However (3.15) implies the overestimate $\overline{P}(H \geq \eta)$ of $P(H \geq \eta)$ exceeds $P(A_{\max} \geq \eta)$ for all positive η — a contradiction. So $\kappa_7 = 0$, D and H are independent, and have a joint distribution of the form

$$\kappa_5 e^{-\frac{1}{2}[\kappa_6 H^2 + \kappa_8 D^2]}.$$

Hence

$$\overline{P}(K = \gamma) = P(D = \gamma) \star \overline{P}(H = \gamma).$$

From (3.12), the variance of D is $\frac{c\ell}{6}$. This, and the equation immediately above, yield

$$\overline{P}(H(\omega) = \gamma) = \frac{8}{\sqrt{2\pi}}\sqrt{\frac{24}{11c\ell}} e^{-\frac{1}{2}\gamma^2 \frac{24}{11c\ell}},$$
$$\text{and} \quad \overline{P}(H(\omega) = \gamma) \geq P(A_{\max}(\omega) = \gamma)$$

follows immediately from (3.15).

By a similar reasoning using $J(\omega)$,

$$P(A_{\max} = \gamma) \leq \frac{4}{\sqrt{2\pi}}\sqrt{\frac{6}{5c\ell}} e^{-\frac{1}{2}\gamma^2 \frac{6}{5c\ell}}.$$

Hence

$$P(A_{\max} \geq \frac{h}{\ell}) \leq \frac{4}{\sqrt{2\pi}} \min\left\{ \int_{\xi=\sqrt{\frac{h^2}{\ell^3 c}}}^{\infty} \sqrt{\frac{6}{5}} e^{-\frac{3}{5}\xi^2} d\xi, \int_{\xi=\sqrt{\frac{h^2}{\ell^3 c}}}^{\infty} 2\sqrt{\frac{24}{11}} e^{-\frac{12}{11}\xi^2} d\xi \right\}, \tag{3.16}$$

as required. ♣

The upper bound (3.16) on $P(A_{\max} \geq \frac{h}{\ell})$ is a function of $\frac{h^2}{\ell^3 c}$ only. Using simple dimensional arguments it can be shown that $P(A_{\max} \geq \frac{h}{\ell})$ itself is also a function of $\frac{h^2}{\ell^3 c}$ only. Clearly $P(A_{\max} \geq \frac{h}{\ell})$ is a function of h, ℓ, and c alone. Now by replacing A with γA, where $\gamma > 0$, it follows that

$$P(\gamma A_{\max} \geq \frac{\gamma h}{\ell}) = P(A_{\max} \geq \frac{h}{\ell}),$$

and also

$$\gamma^2 c_A = c_{\gamma A}.$$

Hence h and c must appear only in the combination $\frac{h^2}{c}$. To include ℓ, stretch A horizontally by replacing σ by $\gamma\sigma$, where $\gamma > 0$. This replaces ℓ by $\gamma\ell$. In addition, replace h by γh. This is needed to keep $P(A_{\max} \geq \frac{h}{\ell})$ unchanged. Finally, as the variance of the random variable $A(\sigma_2) - A(\sigma_1)$ (which equals $c|\sigma_2 - \sigma_1|$) is unchanged by this scaling for any $\sigma_1, \sigma_2 \in [0, \ell]$, it follows that c is replaced by $\frac{c}{\gamma}$. Hence $P(A_{\max} \geq \frac{h}{\ell})$ can contain h, ℓ and c only in the combination $\frac{h^2}{\ell^3 c}$.

Clearly both $P(A_{\max} \geq \frac{h}{\ell})$ and the upper bound (3.16) are monotonically decreasing functions of

$$\wp = c\frac{h^2}{\ell^3 c}. \tag{3.17}$$

Hereafter $\wp$ is referred to as the linkage parameter.

3.4.1 Links and Clusters.

Theoretically the MOS groups the test points into clusters, where every point in some cluster is (expected to) lie in the region of attraction of a specific stationary point of $g(x_0, .)$. Actually, it is not necessary to link a point y into a specific cluster — it is sufficient to observe that a link of the required reliability exists from y to some higher test point. Once this has been established y is, for the moment at least, of no further interest as a starting point for the local search procedure.

There are two situations which need to be considered when making a link, resulting in two different linking criteria. The first is when neither end–point of the link is a known stationary point. In this case the linkage parameter is as described above. The second case is when at least one end-point is a known stationary point. If the lower of the two end–points is a known stationary point then no link is made. Otherwise only the upper end–point is a known stationary point. In this case the the results in proposition (3.11) are no longer applicable. Moreover, as the following example shows, the linkage parameter (3.17) can not be applied meaningfully to this case. Consider $g = -t^2$, where $t \in [0,1]$, and let the end-points of the link be 0 and t. An elementary calculation shows that the linkage parameter $\wp = t.c^{-1}$. As $t \to 0$, the reliability of the link between t and 0 goes to zero. This difficulty is circumvented by automatically linking to a stationary point y_s any test point y which lies in a fixed neighbourhood of y_s, and for which $g(x_0, y) \leq g(x_0, y_s)$.

The cases involving stationary points can arise only if some local searches are performed before the exploration phase is completed, and these stationary points are included in the list of test points.

3.4.2 Estimation of c.

Even without explicit knowledge of c, the linkage parameter provides a means of ordering the links between test points according to the perceived reliability of each link. This assumes that the same value of c is used for each link, irrespective of its orientation, or position. If c varies with the link's orientation or position, then an explicit estimate of it for each link is required if the links are to be ranked in order of their perceived reliabilities. Possession of an estimate $\bar{c}$ of c allows the linkage parameter to be calculated explicitly, thereby giving an absolute measure of perceived reliability to each link. From this a stopping rule for the MOS can be developed. The reliabilities of the solutions generated by the MOS for different iterations of the SIP algorithm can also be compared.

The IBMP models $u^T \nabla g$ along the line segment between two neighbouring test points. As the mean of the IBMP for any particular link is the average value of $u^T \nabla g$ on that link, a third point on the link is needed to provide information about c. When g is calculated at a test point t_0 generated by the Halton sequence, the nearest previous test point t_1 to t_0 is found. The value of g at a third point t_2 is calculated, where t_2 lies on the line through t_0 and t_1. The point t_2 is chosen as the first member of $\{2t_0 - t_1, \frac{1}{2}(t_0 + t_1)\}$ to lie in T. The three points t_0, t_1, t_2 are equally spaced. From now on assume t_1 is the mid-point of the interval $[t_0, t_2]$.

Proposition 3.12 *Let t_0, t_1 and t_2 be three equally spaced co-linear test points, where t_1 is the centrepoint, and where $\|t_2 - t_0\| = 2\ell$. Let the IBMP model of the directional derivative of g along $[t_0, t_2]$ have variance c, and let the random variable $g_e(t_1)$ denote the value of $g(t_1)$ predicted by the IBMP model of $u^T \nabla g$ on $[t_0, t_2]$.*

Then the probability distribution of $g_e(t_1)$ is Gaussian, with

$$\textit{mean} = \frac{g(t_2)+g(t_0)}{2}, \quad \textit{and} \quad \textit{variance} = \frac{1}{6}c\ell^3.$$

PROOF. Knowledge of $g(t_0)$ and $g(t_2)$ excludes all sample paths of F except those which have their mean value on $[0, 2\ell]$ equal to $\frac{1}{2\ell}[g(t_2) - g(t_0)]$. Because

$$\int_0^{2\ell} A(\omega, \sigma)\, d\sigma = 0 \ \ \forall \omega \in \Omega,$$

it follows that

$$D(\omega) = \frac{g(t_2) - g(t_0)}{2\ell}.$$

At each value of σ in $[0, 2\ell]$ the means of F and D are both zero. This implies the mean of A is also zero at each such value of σ. Hence the independence of A and D implies that

$$\text{mean}(g_e) = g(t_0) + \int_0^{\ell} D(\omega)\, d\sigma = \tfrac{1}{2}\left(g(t_2) + g(t_0)\right).$$

As $D(\omega)$ is fixed by $g(t_0)$ and $g(t_2)$, the variance of g_e is entirely due to the AC part of F. The independence of A and D implies that the value at which D is fixed does not affect the variance of g_e. For simplicity, let $g(t_0) = g(t_2)$.

The IBMP $F(\omega, \sigma)$ on $\sigma \in [0, 2\ell]$ has the representation

$$\sqrt{\frac{c}{4\ell}}\left[B_0(\sigma^2) + B_{2\ell}((2\ell - \sigma)^2)\right].$$

The approach taken is to find the joint probability distribution of the two Gaussian random variables

$$G_{10} = g_e(t_1) - g(t_0) = \int_{\sigma=0}^{\ell} F(\sigma)\, d\sigma, \quad \text{and} \quad G_{21} = g(t_2) - g_e(t_1) = \int_{\sigma=\ell}^{2\ell} F(\sigma)\, d\sigma,$$

and then impose the condition $g(t_2) = g(t_0)$ to obtain the variance of $g_e(t_1)$. As B_0 and $B_{2\ell}$ are independent they may be considered separately.

Using E to denote expectation, the variance of G_{10} arising from B_0 is as follows:

$$\frac{c}{4\ell} E\left[\int_{\xi=0}^{\ell} B_0(\xi^2)\, d\xi \,.\, \int_{\gamma=0}^{\ell} B_0(\gamma^2)\, d\gamma\right]$$

$$= \frac{c}{4\ell}\int_{\omega\in\Omega}\left[\int_{\xi=0}^{\ell} B_0(\omega, \xi^2)\, d\xi \,.\, \int_{\gamma=0}^{\ell} B_0(\omega, \gamma^2)\, d\gamma\right] P(d\omega)$$

$$= \frac{c}{4\ell}\int_{\xi=0}^{\ell}\int_{\gamma=0}^{\ell} E\left[B_0(\xi^2)B_0(\gamma^2)\right]\, d\xi\, d\gamma$$

$$= \frac{c}{4\ell} \int_{\xi=0}^{\ell} \int_{\gamma=0}^{\ell} \min(\xi^2, \gamma^2)\, d\xi\, d\gamma = \frac{c\ell^3}{24}.$$

For the variance of G_{21} from B_0:

$$\frac{c}{4\ell} E\left[\int_{\xi=\ell}^{2\ell} B_0(\xi^2)\, d\xi \int_{\gamma=\ell}^{2\ell} B_0(\gamma^2)\, d\gamma\right] = \frac{11c\ell^3}{24}.$$

Finally for the covariance between G_{10} and G_{21} from B_0

$$\frac{c}{4\ell} E\left[\int_{\xi=0}^{\ell} B_0(\xi^2)\, d\xi \int_{\gamma=\ell}^{2\ell} B_0(\gamma^2)\, d\gamma\right] = \frac{c\ell^3}{12}.$$

Hence the covariance matrix for G_{10} and G_{21} is

$$V = \frac{c\ell^3}{6}\begin{pmatrix} 3 & 1 \\ 1 & 3 \end{pmatrix},$$

and $G = [G_{10}, G_{21}]$ has the joint probability density function

$$\frac{1}{2\pi}\sqrt{\det(V)}e^{-\frac{1}{2}GV^{-1}G^T}.$$

Now $g(t_0) = g(t_2)$, which implies $G_{10} = -G_{21}$ $(= \eta$ say), and η has the probability density function

$$\frac{1}{\sqrt{2\pi}}\sqrt{\frac{6}{c\ell^3}}e^{-\frac{1}{2}\eta^2\frac{6}{c\ell^3}},$$

hence $g_e(t_1) - \frac{1}{2}[g(t_0) + g(t_2)]$ has variance $\frac{1}{6}c\ell^3$. ♣

For each test point t_i from the Halton sequence a co-linear triple t_{iL}, t_{iM}, t_{iR} is formed as above, where the random variable

$$\sqrt{\frac{6}{||t_{iM} - t_{iR}||^3}}\left[g(t_{iM}) - \frac{1}{2}[g(t_{iL}) + g(t_{iR})]\right]$$

has a Gaussian distribution with zero mean and variance c. Hence the best estimator $\overline{c}$ of c is

$$\overline{c} = \frac{6}{k}\sum_{i=1}^{k} \frac{\left[g(t_{iM}) - \frac{1}{2}[g(t_{iL}) + g(t_{iR})]\right]^2}{||t_{iL} - t_{iM}||^3} \tag{3.18}$$

To calculate $\overline{c}$, co-linear triples of equally spaced test points are needed. If a Halton sequence is used, then forming these triples means that only every second test point is generated using the Halton sequence. With a grid, estimating $\overline{c}$ would present no difficulty provided the grid had at least 3 points along each edge.

The Algorithm.

1. Calculate $g(x_0, t)$ at N_0 test points generated using the Halton sequence. If the estimate $\bar{c}$ of c is to be improved, then as each test point y is generated do the following:

   (a) Locate the nearest existing test point y_n to y.

   (b) If $2y - y_n \in T$ then calculate $g(x_0, .)$ at $2y - y_n$, otherwise calculate $g(x_0, .)$ at $\frac{1}{2}(y + y_n)$.

   (c) Update $\bar{c}$ using this new co-linear triple of test points.

2. For each test point, if an upward link exists for that point, then determine whether or not that link is still valid. If the link is invalid, then delete it.

3. For each point not linked upward, find the nearest neighbours of that test point. Calculate the linkage parameter for each of the nearest neighbours of the test point. Of the nearest neighbours at which g takes a value greater than that at y, choose the one for which the linkage parameter is greatest. Link y upwards to that point. If no such nearest neighbour exists, then y is not linked upwards.

4. Stopping criteria. If the maximum number of test points have been used, or if the average value of the linkage parameter over the links is sufficiently high and a minimum number of test points have been used, then do the next item. Otherwise increase N_0 and go to 1.

5. For each unlinked point do a local search, using that point as the initial point.

The problem of locating the nearest neighbours of a test point occurs twice in the above algorithm. For a sequence of points such as the Halton sequence this problem is by no means trivial. An efficient solution to this problem when p is not too large ($\leq$ 5 or 6 say) has been given by Bentley et al. [12]. This algorithm is discussed in some detail in chapter 4.

It is desirable that the local search procedure used to refine the estimates of the local maximisers be capable of superlinear convergence on problems which satisfy the pertinent conditions. As g is only guaranteed to be once continuously differentiable, the obvious candidate is a quasi-Newton algorithm. For the purpose of demonstrating convergence, it is assumed that the local search procedure satisfies the following assumption.

Assumption 3.13 *For all $t_0 \in T$, if a local search is performed with t_0 as the initial point, generating the sequence of iterates $t_1, t_2, \ldots$, then for each $n \geq 0$, there exists a path from t_n to t_{n+1}, lying entirely in T, such that $g(x_0, .)$ is strictly increasing along that path.* ♣

This assumption is needed to show that the MOS will always eventually find a finite subset of T satisfying assumption 2.8.

3.5 Convergence.

The purpose of the MOS is to find a set of points S (or more accurately approximations thereto) which satisfies assumption 2.8, which is reproduced here for convenience.

> At each point x_0 at which the global optimisation sub-algorithm is used, it finds every point in $\Gamma(x_0)$. Also, each $x_1 \in R^n$ has a neighbourhood $\mathcal{N}(x_1)$, such that if $x_0 \in \mathcal{N}(x_1)$, then the multi-local optimisation subalgorithm finds some extension (evaluated at x_0) of each member of $\Gamma(x_1)$.

The set $\Gamma(x_0)$ of global maximizers of $g(x_0, t)$ presents no difficulty — assumption 2.3 implies that $\Gamma(x_0)$ is finite, and all members of $\Gamma(x_0)$ are required for the solution set. The extensions of $\Gamma(x_1)$ for x_0 near x_1 present a more complex case. If x_0 is sufficiently near x_1, then by proposition 2.4, the extensions of the members of $\Gamma(x_1)$, evaluated at x_0, lie in the interior of the union of a finite number of disjoint closed balls $\{B_i(\epsilon_1)\}_{i=1}^{j}$, each of radius ϵ_1, where $\epsilon_1 > 0$. Each such ball is centred on a member of $\Gamma(x_1)$, and all of the distances between the centres of these balls are at least $4\epsilon_1$. The set of extensions of any $\tau \in \Gamma(x_1)$ is the full set of global maximizers of $g(x_0, .)$ over the closed ball centred on τ. Also, by setting $C \equiv T$ in proposition 2.4, for x_0 sufficiently close to x_1, any value $g(x_0, .)$ takes at any extension (evaluated at x_0) of any member of $\Gamma(x_1)$ is greater than the supremum of $g(x_0, .)$ on $T - \cup_{i=1}^{j} B_i(\epsilon_1)$.

The problem of finding some extension of each member of $\Gamma(x_1)$, evaluated at x_0, is equivalent to finding a global maximiser (provided one exists) of $g(x_0, .)$ over each member of a set of open balls, where each open ball is of radius ϵ_1, and collectively these balls form a finite covering of T. Unfortunately the precise value of ϵ_1 will not be known. However if this is true for some value δ_k at the k^{th} evaluation of g, and $\delta_k \to 0$ as $k \to \infty$, then the required ϵ_1 will eventually be achieved. This will also ensure that for sufficiently large k, each member of $\Gamma(x_0)$ will be adequately approximated. Under these conditions the algorithm will ultimately provide a sequence of finite sets of points which contain increasingly accurate approximations to a set of points which collectively are a solution set to the multi–local optimisation subproblem.

This situation is no worse than that which occurs with a general global optimisation problem. For that type of problem, although the approximate solution ultimately converges to the global maximal value, on terminating after a finite time the possible existence of high narrow peaks in the objective function means the possibility that the calculated solution is completely wrong always exists. To ensure any such peaks are detected, a global optimisation algorithm must satisfy the same conditions required of the MOS algorithm, as listed in the previous paragraph.

Theorem 3.14 *Given:*

1. *The sequence of test points is dense in T.*
2. *The local search procedure used is one of strict ascent, as described in assumption (3.13).*

Then, for any positive ϵ, in a finite number of steps the MOS will find a finite set of points containing approximations to some finite set satisfying assumption 2.8, where the error in each approximation is at most ϵ.

PROOF. From assumption 2.3 $\Gamma(x_0)$ is finite. Proposition 2.4 implies that the set of extensions of $\Gamma(x_1)$ evaluated at x_0 consist of $|\Gamma(x_1)|$ disjoint sets S_i, each of which is a subset of some $B_i(\epsilon_1)$, where the relation between the S_i's and B_i's is one to one. Again by proposition 2.4, each S_i is closed, and is the full set of global maximizers of $g(x_0, .)$ over B_i. Using

$$\Gamma(x_0) = \{\tau_i\}_{i=1}^{j} \quad \text{and} \quad |\Gamma(x_1)| = k,$$

the class $\mathcal{C}$ of sets $\{S_1, \ldots, S_k, \{\tau_1\}, \ldots, \{\tau_j\}\}$ are all closed and disjoint (assuming possible repetitions of members of the S_i's amongst the $\{\tau_i\}$'s have been removed). Hence $\exists \epsilon_0 > 0$ such that the minimum distance between any two points in different members of $\mathcal{C}$ is at least $5\epsilon_0$ apart. Let $S \in \mathcal{C}$. Define the δ neighbourhood of S as follows:

$$\mathcal{N}_\delta(S) = \{t \in T : \exists s \in S \text{ such that } \|t - s\| < \delta\},$$

and let $\overline{\mathcal{N}_\delta}(S)$ denote the closure of $\mathcal{N}_\delta(S)$. For some sufficiently small positive δ and ϵ, with $\delta < \epsilon < \epsilon_0$,

$$\max_{t \in \overline{\mathcal{N}_{2\epsilon}}(S) - \mathcal{N}_\delta(S)} g(x_0, t) < g(x_0, s) \quad \text{where} \quad s \in S.$$

Now $g(x_0, .)$ is Lipschitz on T, hence for some positive δ, where $\delta < \epsilon$,

$$\forall t \in \mathcal{N}_\delta(S), \;\; \forall s \in \overline{\mathcal{N}_{2\epsilon}}(S) - \mathcal{N}_\epsilon(S), \;\; g(x_0, t) > g(x_0, s).$$

As $\mathcal{N}_\delta(S)$ has a strictly positive Lebesgue measure in R^p, for all sufficiently large N, $\mathcal{T}_N \cap \mathcal{N}_\delta(s)$ will be non-empty, and also $\sqrt{p}.\ell_{\max}(N) < \epsilon$.

Let $t_{\mathrm{m}}(N) \in \mathcal{T}_N \cap \mathcal{N}_\delta(S)$ be a test point at which $g(x_0, .)$ achieves its maximum on $\mathcal{T}_N \cap \mathcal{N}_\delta(S)$. Then no upward link exists from $t_{\mathrm{m}}(N)$ to any point in $\mathcal{T}_N$. Hence a local search is performed, with $t_{\mathrm{m}}(N)$ as the initial point. Given the local search satisfies assumption (3.13), it will converge to a point in $\mathcal{N}_\epsilon(S)$ at which $g(x_0, .)$ takes a value not less than $g(x_0, t_{\mathrm{m}}(N))$.

Because $\mathcal{T}$ is dense in T, $g(x_0, t_{\mathrm{m}}(N)) \to g(x_0, s)$ as $N \to \infty$, where $s \in S$. Hence, for sufficiently large N, a terminal point of some local search will approximate arbitrarily well some member of S. As $\mathcal{C}$ is finite, it is clear that this result holds for all members of $\mathcal{C}$ when N is sufficiently large. ♣

3.6 Other Remarks.

The linking criterion based on the stochastic process is of a purely local character. It is based only on the perceived *ignorance* of the algorithm's knowledge of g in regions between neighbouring test points. Accordingly its use does not imply any tacit assumptions concerning any *a priori* statistics about g as a whole. Also, in contrast to the algorithms of Rinnooy Kan et al.[89, 90], it does not require the test points to be drawn from any particular distribution. Randomly generated sets of points, rectangular, or polar grids, or other sequences of test points could equally well be used in place of a Halton sequence.

The lack of dependence on the distribution of the test points permits some parts of T to be explored more fully than others without great inconvenience. For example, parts of T on which g takes its greatest values could (and perhaps should!) be explored more thoroughly than other parts of T. This could be achieved simply by placing more test points in such regions. If this results in a distribution of test points that is far from even, then the nearest neighbour subalgorithm of Bentley et al [12] will be less efficient. In this event, the loss of efficiency may be reduced in the manner described in Bentley et al. Another tactic is to increase the reliability required of the links between test points at which g takes its greater values.

Alternatively, the algorithm may be used more than once. On the first run the local maximisers of g over T are sought. On each subsequent run the local maximisers of g on some convex subset T_{sub} of T are sought, where T_{sub} is a region on which g takes its greater values. The lack of requirements on the structure of the sequence of test points means that any test points in T_{sub} generated in previous runs of the algorithm may be included with those used to explore T_{sub} without difficulty.

A common implementation of a grid based algorithm is along the following lines. The nearest neighbours of a test point y are usually taken to be other grid points which can be obtained by displacing y along each axis by at most the distance between adjacent layers of points along that axis. If the value g takes at a grid point y is not less than that taken by g at each nearest neighbour of y then a local search is performed with y as the initial point. If there are two adjacent local maximisers at which g takes similar values, then the spacing between adjacent layers of grid points would have to be less than half the spacing between the local maximisers if they are to be reliably resolved. The use of a linking process would improve the algorithms ability to resolve adjacent local maximisers, possibly at the expense of extra local searches elsewhere.

One advantage of a rectangular grid is that the problem of finding the nearest neighbour to a test point becomes trivial. Halton sequences have much less structure than grids. This makes the problem of finding the nearest neighbours of a test point more substantial. A good algorithm for this has been developed by Bentley et al. [12], however this algorithm is exponential in p. This places an upper limit of about 4 to 6 on the number of dimensions of T if the algorithm is to be run in a reasonable length of time. This limitation is not peculiar to the Halton sequences: it also occurs for randomly generated sequences, and indeed for any sequence lacking sufficient structure to support a method for finding the nearest neighbours of a test point which is faster than that of Bentley et al. [12].

The MOS algorithm closely resembles the algorithms of Rinnooy Kan and Timmer [89, 90]. The latter are very efficient in the number of function evaluations required to obtain the global maximum. The statistic Rinnooy Kan et al [89, 90] use for their stopping condition is a function only of the number of test points used, and the number of local maximisers found. Accordingly, the number of test points used does not depend on p. In spite of this fact their algorithm is exponential in p. This is a consequence of using the nearest neighbour algorithm of Bentley et al. [12]. One could attempt to avoid this by, say, attempting links between all pairs of points, but then the work done is then proportional to the square of the number of test points. When the dimension of T is not too large, using the algorithm of Bentley et al is the lesser of two evils. Similar remarks can be made about the MOS algorithm: the stopping conditions do not involve p explicitly, but the same nearest neighbour algorithm is used.

In order to treat each dimension equally, rectangular grid-based algorithms are inherently exponential in p, whereas the MOS algorithm, and the algorithms of Rinnooy Kan et al are not. For a grid based algorithm the work required to find the nearest neighbours of a single test point can vary from being linear in p to exponential in p depending on how the set of nearest neighbours of a test point is defined. In contrast, for the MOS algorithm, and the algorithms of Rinnooy Kan et al, the nearest neighbour subproblem appears to be either exponential in p, or the computational effort required of the MOS algorithm is not linear in the number of test points generated.

Chapter 4

IMPLEMENTATION OF THE ALGORITHM.

The implementation of the algorithm is described in this chapter. The description is in three parts. They are: the main SIP algorithm, the algorithm for solving the multi-dimensional multi-local optimisation problem, and the algorithm for solving the one dimensional multi-local optimisation problem.

4.1 Implementation of the SIP algorithm.

In this section a step-by-step description of the algorithm used to generate the numerical results is given. In this section the MOS algorithms are treated as 'black boxes' — calls to these algorithms are simply listed without qualification. The multi-dimensional, and one dimensional MOS algorithms are respectively detailed in sections 2 and 3 of this chapter.

Before listing the SIP algorithm, some variables and parameters are described. The parameter κ_{minstep} is the minimum length of the difference between consecutive iterates which is interpreted as a non-zero step; any step shorter than this immediately stops the algorithm. The parameter κ_{gradient} is the maximum residual of the first order KKT conditions permissible at a point which is regarded as a solution of the SIP. Finally, κ_{theta} is the largest magnitude a constraint's residual is permitted to take when that constraint is regarded as active, but not violated.

Provision is made for the existence of simple bounds on x. These bounds are represented as follows:

$$l_i \leq x_i \leq u_i.$$

The subscripts denote the i^{th} element of the appropriate vector. If x_i is not bounded above, then u_i is set to $+\infty$. Similarly, if x_i is unbounded below, then l_i is set to $-\infty$.

In some runs of the algorithm, different values than those listed below have been used for some parameters. In particular, $\theta_{\text{crossover}}$ and θ_{cap} were altered in some problems. The details of these changes are listed in chapter 5.

The SIP Algorithm.

1. INITIALIZATION.

$$\mu^{(1)} = 0.1, \quad \nu^{(1)} = 1.0, \quad k = 1,$$
$$\Delta^{(1)} = 2, \quad \beta = 0.5, \quad \rho = 0.33,$$
$$H^{(1)} = I_n,$$
$$\theta_{\text{crossover}} = 1, \quad \theta_{\text{cap}} = 1,$$
$$\kappa_1 = 1.2, \quad \kappa_2 = 1.5, \quad \kappa_3 = 1.2, \quad \kappa_4 = 4,$$
$$\kappa_{\text{minstep}} = 10^{-8},$$
$$\kappa_{\text{gradient}} = 10^{-5},$$
$$\kappa_{\text{theta}} = 10^{-5}.$$

2. SET UP THE FIRST ITERATION.

Using the appropriate MOS algorithm, find the global maximisers of the constraint function at the initial point and also find as many local maximisers as is practical; this yields the set $\mathcal{A}^{(1)}$. Using this $\theta^{(1)}$ is calculated.

3. CALCULATE THE LAGRANGE MULTIPLIERS.

Estimates of the optimal Lagrange multipliers are found by solving the approximating L_∞QP without a trust region. That is by solving

$$\min_{s \in R^n} s^T \nabla f(x^{(k)}) + \tfrac{1}{2} s^T H^{(k)} s + \mu^{(k)} \zeta + \tfrac{1}{2} \nu^{(k)} \zeta^2$$

subject to the constraints:

$$g(x^{(k)}, \tau) + s^T \nabla_x g(x^{(k)}, \tau) \leq \zeta \quad \forall \tau \in \mathcal{A}^{(k)},$$

$$\zeta \geq 0,$$

and also subject to the following simple bounds on s:

$$\min\{l_i - x_i^{(k)}, -\gamma\} \leq s_i \leq \min\{u_i - x_i^{(k)}, \gamma\},$$

where γ is a large positive number.

4. SOLVE THE L_∞QP.

   The search direction $s^{(k)}$ is chosen as the solution to the L_∞QP

$$\min_{s \in R^n} s^T \nabla f(x^{(k)}) + \tfrac{1}{2} s^T H^{(k)} s + \mu^{(k)} \zeta + \tfrac{1}{2} \nu^{(k)} \zeta^2 \tag{4.1}$$

   subject to the constraints:

$$g(x^{(k)}, \tau) + s^T \nabla_x g(x^{(k)}, \tau) \leq \zeta \;\; \forall \tau \in \mathcal{A}^{(k)}, \tag{4.2}$$

$$\zeta \geq 0, \tag{4.3}$$

   and also subject to the following simple bounds on s:

$$\min\{l_i - x_i^{(k)}, -\Delta^{(k)}\} \leq s_i \leq \min\{u_i - x_i^{(k)}, \Delta^{(k)}\}. \tag{4.4}$$

   Here the subscript i denotes the i^{th} component of the relevant vector. If $\theta^{(k)} \geq \theta_{\text{cap}}$ then the following bound (referred to as the capping constraint) is included in the list of L_∞QP constraints:

$$\zeta \leq \theta^{(k)}. \tag{4.5}$$

5. WHEN THE CAPPING CONSTRAINT IS ACTIVE.

   If the capping constraint's Lagrange multiplier ξ indicates that the capping constraint is active, then the penalty parameters are adjusted as described in step 11, except that the quantity $\|\lambda^{(k)}\|_1$ is replaced by $\mu^{(k)} + \nu^{(k)}\theta^{(k)} + |\xi|$. The algorithm then proceeds to step 4.

6. STOPPING CONDITIONS.

   If the following two conditions are satisfied

$$\left\| \nabla f(x^{(k)}) - \sum_{t \in \Gamma(x^{(k)})} \lambda^{(k)} \nabla_x g(x^{(k)}, t) \right\|_2 < \kappa_{\text{gradient}}$$

$$\text{and } \theta^{(k)} \le \kappa_{\text{theta}}$$

then go to step 12. For computational purposes, any element t_0 of $\mathcal{A}^{(k)}$ is regarded as a member of $\Gamma(x^{(k)})$ if

$$g(x^{(k)}, t_0) \ge \theta^{(k)} - \kappa_{\text{theta}}.$$

7. Attempt the proposed step.

Determine if the prospective new iterate $x^{(k)} + s^{(k)}$ satisfies the sufficient descent condition:

$$\phi^{(k)} - \phi\left(x^{(k)} + s^{(k)}\right) \ge \rho\left[\phi^{(k)} - \psi(x^{(k)}; s^{(k)})\right].$$

If the capping constraint was active at the L_∞QP's solution in step 4, then the extra condition

$$\theta\left(x^{(k)} + s^{(k)}\right) \le \theta^{(k)}$$

must also hold for $x^{(k)} + s^{(k)}$ to be accepted as the next iterate. If these conditions hold, then set $x^{(k+1)} = x^{(k)} + s^{(k)}$, and go to step 10.

8. Calculate the Maratos effect correction vector.

(a) Let $\mathcal{Q}^{(k)}$ denote the subset of $\mathcal{A}^{(k)}$ which gives rise to the constraints which are active at the solution of the L_∞QP listed in step 4. Let $\mathcal{A}^{(k)}_{\text{soc}}$ denote the subset of T selected by the multi-local optimization subalgorithm at the point $x^{(k)} + s^{(k)}$. For each member ω of $\mathcal{Q}^{(k)}$, find the closest point in $\mathcal{A}^{(k)}_{\text{soc}}$ to ω. Call this point $t_{\text{soc}}(\omega)$.

(b) If $\mathcal{Q}^{(k)}$ is empty, or if two or more points in $\mathcal{Q}^{(k)}$ have the same closest point in $\mathcal{A}^{(k)}_{\text{soc}}$ then set $c^{(k)} = 0$, and go to step 9.

(c) Solve the following QP for $c^{(k)}$:

$$\min_{c \in R^n} c^T c$$

$$\text{such that } c^T \nabla_x g(x^{(k)}, \omega) + g(x^{(k)} + s^{(k)}, t_{\text{soc}}(\omega)) \le 0, \quad \forall \omega \in \mathcal{Q}^{(k)},$$

and such that:

$$l_i - x_i^{(k)} - s_i^{(k)} \le c_i \le u_i - x_i^{(k)} - s_i^{(k)} \quad \forall i = 1, \ldots, n.$$

(d) If $\|c^{(k)}\|_2 \ge \|s^{(k)}\|_2$ then set $c^{(k)} = 0$.

9. PERFORM THE ARC SEARCH.

   Consider successive values of the sequence $1, \beta, \beta^2, \beta^3, \ldots$ as trial values of α. If $c^{(k)} = 0$ then omit the first member of the sequence, otherwise start with $\alpha = 1$. Accept the first trial value of α which satisfies the following conditions.

   The condition for acceptance of $x^{(k)} + q^{(k)}(\alpha)$ as $x^{(k+1)}$ is that

$$\phi^{(k)} - \phi\left(x^{(k)} + q^{(k)}(\alpha)\right) \geq \rho\alpha\left[\phi^{(k)} - \psi(x^{(k)}; s^{(k)})\right].$$

   If the capping constraint was active at the L_∞QP's solution in step 4, then the extra condition

$$\theta\left(x^{(k)} + q^{(k)}(\alpha)\right) \leq \theta^{(k)}$$

   must also hold for $x^{(k)} + q^{(k)}(\alpha)$ to be accepted as the next iterate.

   After a satisfactory value of α has been found, set

$$x^{(k+1)} = x^{(k)} + q^{(k)}(\alpha).$$

10. ALTER THE TRUST REGION SIZE.

    If the trust region is used, set

$$\Delta^{(k+1)} = 4\|x^{(k+1)} - x^{(k)}\|_\infty, \tag{4.6}$$

    otherwise, use $\Delta^{(k+1)} = \Delta^{(1)}$.

11. UPDATE μ, ν AND H.

    If $\theta^{(k)} < \theta_{\text{crossover}}$ then set $\nu^{(k+1)}$ equal to $\nu^{(k)}$, and update μ according to the following rule:

$$\text{If } \mu^{(k)} \leq \kappa_1\|\lambda^{(k)}\|_1, \text{ then set } \mu^{(k+1)} = \kappa_2\|\lambda^{(k)}\|_1,$$
$$\text{otherwise set } \mu^{(k+1)} = \mu^{(k)}.$$

    If $\theta^{(k)} \geq \theta_{\text{crossover}}$ then set $\mu^{(k+1)}$ equal to $\mu^{(k)}$, and update ν by the following rule:

$$\text{If } \mu^{(k)} + \nu^{(k)}\theta^{(k)} \leq \kappa_3\|\lambda^{(k)}\|_1, \text{ then}$$
$$\text{adjust } \nu^{(k+1)} \text{ to satisfy } \mu^{(k+1)} + \nu^{(k+1)}\theta^{(k)} = \kappa_4\|\lambda^{(k)}\|_1,$$
$$\text{otherwise set } \nu^{(k+1)} = \nu^{(k)}.$$

The matrix $H^{(k)}$ is updated using the BFGS update to yield $H^{(k+1)}$. If this update results in a loss of positive definiteness, then the update is not performed. Actually the Choleski factors of the H matrices are updated and stored, rather than the matrices themselves. The algorithm used to update the Choleski factors is that listed in [35].

If either penalty parameter has been changed, recalculate $\phi^{(k)}$ with the new penalty parameter values.

12. STOPPING CONDITIONS.

If either the maximum number of iterations has been reached, or the maximum number of MOS sub-algorithm calls has been reached, or the stopping conditions listed in step 6 have been satisfied, or

$$||x^{(k+1)} - x^{(k)}||_2 \leq \kappa_{\mathbf{minstep}}$$

then halt. Otherwise increment k by 1, and go to step 3.

The QP subproblems in steps 3, 4, and 8, were solved using the NAG subroutine E04NCF. E04NCF is a two phase primal quadratic programming subroutine for convex quadratic programmes.

The simple bounds on x are not incorporated into $\theta(x)$. They are treated as rigid constraints in the sense that violations of them are not permitted. Consequently the initial point must satisfy the simple bounds, but need not satisfy the semi-infinite constraint. The simple bounds are imposed on the QPs used to generate $s^{(k)}$, $c^{(k)}$, and the Lagrange multipliers. This ensures every iterate generated satisfies the simple bounds. Violations of the simple bounds arising from round-off errors are quashed by resetting each offending element of $x^{(k)}$ to its pertinent bounding value.

If the trust region (4.6) is not used then the second order Lagrange multiplier estimates generated by solving each L_∞QP in step 4 can be carried over into the following iteration, and used there. In this case step 3 is redundant, and is omitted. If the trust region (4.6) is used then this is no longer possible, and Lagrange multiplier estimates must be obtained elsewhere. To furnish these estimates, the L_∞QP is solved twice at each iteration (in steps 3 and 4). First, the L_∞QP is solved with the trust region absent, thereby yielding the Lagrange multiplier estimates. In practice large bounds were used when solving the L_∞QP for the Lagrange multiplier estimates — this avoids any potential problems with overflows when H is nearly singular. The L_∞QP is then solved with the trust region in place, and yields the search direction $s^{(k)}$.

4.2 The Multi-Dimensional MOS Sub-algorithm.

In this section the algorithm used to find the global and local maximisers of $g(x,.)$ when the dimension of T is greater than one is described. The general features of the algorithm are as follows. Firstly the algorithm explores the constraint function by calculating its values at a set of test points. Each of these test points is regarded as a potential starting point for the local search subroutine. To avoid excessive work the algorithm then attempts to find pairs of test points which lie in the region of attraction of the same local maximiser. This is done by finding pairs of points for which a continuous path of ascent from the lower to the higher is deemed likely to exist. The likelihood of such a path existing between a pair of test points is estimated using the IBMP. If this likelihood is sufficiently great, the a path of ascent linking the two points is assumed to exist, otherwise it is assumed that no such path exists. With such pairs it is not necessary to apply the local search algorithm to the lower point. The lower of the pair of points is said to be linked to the higher. This process reduces the number of local searches dramatically.

After performing this linking process, some test points will not have been linked to any higher test point. For each test point t_i not linked to a higher test point, one step of the local search procedure is performed with t_i as the initial point, yielding the point t_+. Without using any information gained in the step from t_i to t_+, one step of the local search procedure is performed with t_+ as the initial point, yielding the point t_{++}. The test point t_i is linked to t_+, and t_+ is linked to t_{++}. The algorithm then tries to make a link from t_{++} to any nearby higher test point. If this is not successful, then a full local search is done with t_{++} as the initial point.

The test points are examined in pairs when forming the links. If every pair of test points is considered then the work performed is proportional to the square of the number of test points. When the number of test points is large this is far too expensive. Fortunately, if two points are far apart then no link between them is permitted. Therefore, when forming links from a test point it is only necessary to consider neighbouring test points. To this end T is divided up into cells. Here the fact that T is the Cartesian product of intervals is used explicitly. If the maximum link length (measured using the infinity norm) is $\ell_{\max}$, then each component interval $[\alpha_i, \beta_i]$ of the Cartesian product yielding T is divided up into subintervals of length $2\ell_{\max}$. These subintervals are chosen so that the midpoint of one of them coincides with the midpoint of the original interval $[\alpha_i, \beta_i]$. The Cartesian products of these subintervals form the cells. These cells are referred to as the storage cells. The test points are 'stored' in these cells by means of a linked list, the first point in each cell being recorded separately.

A second set of cells is also used. These are constructed in an otherwise identical manner to the set of storage cells, except for one detail: for each component interval $[\alpha_i, \beta_i]$ of the Cartesian product forming T, two subintervals have their endpoints equal to the midpoint of $[\alpha_i, \beta_i]$. This difference places the corners of each cell in either set of cells on the centrepoints of the cells in the other set of cells.

To find the close neighbours of a test point t_0, the selection cell C in which the test point lies is found. Every test point within $\ell_{\max}$ of t_0 (in the infinity norm) will lie in one of the 2^p storage cells which has a non-empty intersection with C. It is not necessary to record a list of points in each selection cell; these cells serve merely as a convenient way of choosing which storage cells should be searched.

As the number of test points used increases, the cell structure is periodically updated. As described in Bentley et al. [12], the cell structure is changed if, and only if the number of test points N equals $2^m.N_{\text{recell}}$, for some positive integer m. Here N_{recell} is the smallest number of test points the MOS sub-algorithm is permitted to generate. This strategy ensures the average number of operations performed in placing points in cells is $O(1)$ per test point. It can be seen that the computational effort of placing each test point in the cell structure is independent of the total number of test points used. So the total cost of performing the first m updates of the cell structure is $O([2^m + 2^{m-1} + \cdots + 1].N_{\text{recell}}) = O(2^{m+1}.N_{\text{recell}})$. The cost of placing every test point in the cell structure for the first time, which is not part of updating the cell structure, is also $O(2^m.N_{\text{recell}})$. The sum of these costs, averaged over the number of test points, is clearly $O(1)$ per test point.

Before listing the multi-local optimisation algorithm, some relevant parameters are described. The variable N_{recell} is used to record the number at which the next re-celling is to occur. Also, N denotes the current number of test points, and $N_{\max}$ is the maximum number of test points excluding those generated in step 7.

For a test point t_i, the value the linkage parameter $\wp$ takes for the most reliable upward link from t_i is denoted by $\wp(i)$.

The maximum length a link is permitted to be is measured using the infinity norm, and is denoted by the parameter $\ell_{\max}$. If the length of a link is very short, that is at most $\kappa_{\text{shortlink}}\ell_{\max}$, then the lower of the two endpoints is automatically linked to the higher with linkage parameter value of $\wp_{\max}$. The parameter κ_{statpt} is the closest two approximations to stationary points may be without being regarded as approximations to the same stationary point. The parameter κ_{link} is the minimum value $\wp.\bar{c}^{-1}$ may take for a link if that link is to be regarded as reliable. The value $\kappa_{\text{meanlink}}\bar{c}$ is the minimum for the average of $\wp(i)$ over all test points before the algorithm will stop generating more test points.

An upper limit on each $\wp(i)$ was imposed, for the following reason. On a sufficiently small scale any C^1 function is approximately linear. In such a case, $h \sim \ell$, for a link of length ℓ and change in g of h along that link. Hence, as $\wp \sim h^2\ell^{-3}$, it follows that $\wp \sim \ell^{-1}$. Now some pairs of test points may be very close together, which could lead to extremely high values of the linkage parameter $\wp(i)$ for links between such pairs of points. Also, the average value of $\wp(i)$ over the links is used in the stopping conditions. The presence of extremely high values of $\wp(i)$ for some links would render the average value of $\wp(i)$ meaningless as an indicator of the average reliability of the links. Hence a maximum value (hereafter $\wp_{\max}$) on $\wp(i)$ was imposed for each link.

The Multi-Dimensional MOS sub-algorithm.

1. INITIALIZATION.

$$N = 0 \quad N_{\text{recell}} = 10p^2 \quad N_{\max} = 2400$$

$$\wp_{\max} = 400$$

$$\kappa_{\text{shortlink}} = 0.01 \quad \kappa_{\text{statpt}} = 0.001$$

$$\kappa_{\text{link}} = 2.5 \quad \kappa_{\text{meanlink}} = 6.25$$

$$\mathcal{A} = \emptyset.$$

2. FIND NEW POSITIONS OF KNOWN STATIONARY POINTS.

   For each stationary point t found by the previous multi-local optimisation, a local search is performed, with t as the initial point. Any such stationary points found by these local searches are placed in the set of stationary points $\mathcal{A}$, but are not put in the list of test points.

3. CONSTRUCT THE CELL STRUCTURE.

   Set
   $$\ell_{\max} = \tfrac{1}{2}\sqrt[p]{\frac{\ln(N_{\text{recell}})}{N_{\text{recell}}\ln(2)}}$$
   Set up the linked lists for the storage cells, and re-cell all existing test points.

4. GENERATE THE TEST POINTS.

   (a) Generate a Halton point y, calculate $g(x,y)$, and set $N = N+1$.

   (b) Place this point into the storage cell structure.

   (c) Find the search cell in which the test point y lies, and then find the storage cells which intersect that search cell.

   (d) Search through these storage cells for the closest (in the 2-norm) existing test point t to y.

   (e) If the point $2y - t$ lies in T, then choose it as the third point t_{third}. Otherwise, choose $\frac{1}{2}(y+t)$. Put t_{third} into the cell structure, calculate $g(x, t_{\text{third}})$, and set $N = N+1$.

   (f) Update the estimate $\overline{c}$ of the average covariance using the formula (3.18).

   (g) If $N < N_{\text{recell}}$, then go to step 4(a).

5. LINK THE TEST POINTS TOGETHER.

   For each test point t_i in turn:

   (a) Find the selection cell $\mathcal{C}$ in which t_i lies.

   (b) Find the storage cells which intersect the selection cell $\mathcal{C}$.

   (c) Set $\wp(i) = 0$.

   (d) For each test point t in each intersecting storage cell, do the following:

      i. If $g(x,t_i) > g(x,t)$, move on to the next t.

ii. If

$$\|t_i - t\|_\infty \leq \kappa_{\text{shortlink}} \ell_{\max}$$

and if either $g(x, t_i) < g(x, t)$ or t_i has a smaller index value in the list of test points than t, then link t_i upwards to t with a linkage parameter value $\wp_{\max}$ and proceed on with the next t_i. Otherwise go to step 5(d)iii.

iii. Calculate $\wp$ for the link between t_i and t. If $\wp > \wp(i)$, then set $\wp(i) = \wp$, and link t_i upwards to t.

iv. If $\wp(i) = \wp_{\max}$ move on to the next t_i, otherwise move on to the next t.

6. Stopping conditions for the exploration phase.

If $N \geq N_{\max}$, or if

$$\frac{1}{N} \sum_{i=1}^{N} \wp(i) \geq \kappa_{\text{meanlink}}\, \bar{c}$$

then proceed to the next step. Otherwise, set

$$N_{\text{recell}} = \min\{N_{\max}, 2N_{\text{recell}}\},$$

and go to step 3.

7. Add in extra test points to establish links.

For each test point t_i for which $\wp(i) < \kappa_{\text{link}}\, \bar{c}$ do:

(a) One iteration of the local search procedure is performed using t_i as the initial point, and yielding the point t_+.

(b) t_+ is placed in the cell structure, and t_i is linked upwards to t_+ by setting $\wp(i) = \wp_{\max}$.

(c) Steps 7(a) and 7(b) are repeated with t_+ in place of t_i, yielding the new point t_{++}, where t_+ is linked upwards to t_{++} with linkage parameter value $\wp_{\max}$.

A maximum number of 4000 extra test points were allowed to be generated in this step. On reaching this maximum, the algorithm immediately proceeds to step 8.

8. Re-link the test points.

   The process described in step 5 is used.

9. Do local searches

   Each test point t_i which is not linked upward to another point with a linkage parameter satisfying:

   $$\wp(i) \geq \kappa_{\text{link}}\,\bar{c}$$

   is used as a starting point for the local search algorithm. Each such newly found stationary point t is added to the set of stationary points begun in step 2 unless the new stationary point is less than a distance of κ_{statpt} from some point t_0 already in $\mathcal{A}$. In the latter case, only one of t and t_0 is placed in the set $\mathcal{A}$; whichever gives the larger value of $g(x,.)$ is chosen. A maximum of 25 on the number of points in $\mathcal{A}$ was imposed. If more than 25 stationary points are found, then only the 25 at which $g(x,.)$ takes the largest values are retained.

Each local search in steps 2, 7 and 9 were performed by the NAG (mark 14) subroutine E04UCF, which is essentially NPSOL. In each case the maximum number of feasibility phase iterations, and the maximum number of optimality phase iterations were both set at 250. The default values of both of these were 50. All other parameters were left at their default values. On a few problems different parameter values were used: these changes are detailed with the problem descriptions in chapter 5.

4.2.1 Choosing the maximum link length.

The nearest neighbour structure of a Halton sequence is too complex to permit a direct analysis leading to a formula for the maximum length of a link as function of the number of test points generated using the Halton sequence. Instead, an approximate analysis is performed, yielding an expression for the maximum length of a link which proved to be reasonable in practice.

In the implementation, the 'length' of the link was bounded using the infinity norm, not the 2-norm.

Let T be the unit hypercube, and let N be the number of test points. In the limit $N \to \infty$, $\ell_{\max}(N) \to 0$. Assume for the moment that the number of stationary points in the constraint function is finite. As $N \to \infty$ almost all test points will not be nearest neighbours of any stationary point, and will not be nearest neighbours of any point on the boundary of T. Let t_0 be such a test point. When $\ell_{\max}(N)$ is small, g is approximately linear over the set

$$S_0 = \{t : \|t - t_0\|_\infty \leq \ell_{\max}(N)\}.$$

Because $\nabla_t g(x_0, t_0)$ is non-zero, $g(x_0, .)$ will exceed $g(x_0, t_0)$ on approximately half of S_0. Assume any link from t_0 to any higher point within S_0 is accepted as a valid upward link for t_0. If the test points are randomly distributed in T, then the probability that t_0 is not linked upward is approximately

$$\left(1 - 2^{p-1}\ell^p_{\max}(N)\right)^N.$$

Assuming this formula can be applied to each test point, and that the resulting probabilities are independent of each other, the probability that every point is linked upwards is

$$\left(1 - \left[1 - 2^{p-1}\ell^p_{\max}(N)\right]^N\right)^N = 1 - \epsilon(N) \quad \text{say,}$$

where $\epsilon(N)$ is small and positive. Choosing $\epsilon(N)$ will fix $\ell_{\max}(N)$. Taking logarithms:

$$\begin{aligned} -\epsilon(N) &\approx N \ln\left(1 - \left[1 - 2^{p-1}\ell^p_{\max}(N)\right]^N\right) \\ &\approx -N\left(1 - 2^{p-1}\ell^p_{\max}(N)\right)^N \end{aligned}$$

because $\ell_{\max}(N) \to 0$ as $N \to \infty$. Taking logarithms again,

$$\begin{aligned} \ln(\epsilon(N)) &\approx \ln(N) + N\ln\left(1 - 2^{p-1}\ell_{\max}(N)\right) \\ &\approx \ln(N) - N2^{p-1}\ell^p_{\max}(N). \end{aligned}$$

Using $\epsilon(N) = N^{-1}$ yields

$$\ell_{\max}(N) = \tfrac{1}{2}\sqrt[p]{\frac{4\ln(N)}{N}}.$$

In the implementation the slightly tighter bound

$$\ell_{\max}(N) = \tfrac{1}{2}\sqrt[p]{\frac{\ln(N)}{N\ln(2)}}$$

was used.

4.2.2 Choosing the prime and increment lists for the Halton sequence.

A Halton sequence in p dimensions is determined by p coprime positive integers $\pi_1, \ldots, \pi_p$, and p increments $\delta_1, \ldots, \delta_p$, where $0 < \delta_i < \pi_i$ for each i in $1, \ldots, p$. For any such choice of the π's and δ's the discrepancy (3.1) of the Halton sequence is asymptotically $O(N^{-1}(\log N)^p)$, however some choices are better than others.

Determining exactly which choice of parameter values is the 'best' is not an easy problem. It is compounded by the fact that there is no obvious definition of 'best.' The choices were made in order to avoid obviously bad values for the primes and increments. For example, a bad choice would be $\pi_1 = 17$, $\pi_2 = 19$, $\delta_1 = 8$, and $\delta_2 = 9$. In this case, the ratio of each increment to its associated prime is almost the same. Almost all of the first 100 points of the Halton sequence generated using these values lie in a narrow band along the main diagonal of the unit square.

Various choices of the π's and δ's were examined. The first 100 points for each choice were calculated, along with the covariances between each pair of components of the points in $\mathcal{H}_{100}$. The distance of each $y_n \in \mathcal{H}_{100}$ to the closest point in $\mathcal{H}_{n-1}$ was also calculated. Using this information, for each p from 2 to 6, the following sets of values for the π's and δ's were chosen. For all these values of p, $\pi_1, \ldots, \pi_6$ were chosen as the first six primes, in increasing order. For all p, and for all $i = 1, \ldots, 6$, $\delta_i = 1$ was used, apart from the following exceptions: if $p = 5$, then $\delta_3 = 2$, and $\delta_5 = 2$ were used, and if $p = 6$ then $\delta_4 = 3$ and $\delta_6 = 5$ were used.

4.3 The One Dimensional Multi-Local Optimisation Algorithm.

This problem was solved in a relatively unsophisticated fashion. In essence, a 1-dimensional grid search was performed, followed by local searches where necessary. Without loss of generality assume the interval over which the local maximisers are sought is $[0, 1]$. An increasing sequence $\{t_j\}_{j=0}^{N}$ of equally spaced test points is used. The lowest test point is 0, and the highest is 1. The constraint function is calculated at each test point. Subintervals of $[0, 1]$ which contain local maximisers are then determined, and a search is performed in each such subinterval. Specifically, for each $j \in 1, \ldots, N-1$, if

$$g(x, t_{j-1}) \leq g(x, t_j) \quad \text{and} \quad g(x, t_j) \geq g(x, t_{j+1}),$$

then a local search is performed in the interval $[t_{j-1}, t_{j+1}]$ with t_j as the initial point. The endpoints 0 and 1 are treated somewhat differently. If

$$g(x, t_0) \geq g(x, t_1),$$

then a local search is performed on the interval $[t_0, t_1]$ with $\frac{1}{2}(t_0 + t_1)$ as the initial point. Similarly, if

$$g(x, t_{N-1}) \leq g(x, t_N),$$

then a search starting from $\frac{1}{2}(t_{N-1} + t_N)$ is done on the interval $[t_{N-1}, t_N]$. All stationary points found by these searches are retained, up to a maximum number of 25. If more than 25 stationary points are found, then only the highest 25 are retained.

For problem 4 of the Watson series with $n \geq 4$, $N = 100$ was used. For all other problems with a one dimensional constraint index set, $N = 40$ was used. All local searches were performed using the NAG (mark 14) routine E04UCF.

Chapter 5

NUMERICAL RESULTS.

The algorithm was tested on the 14 test problems of Watson and Coope [99, 21]. This set of problems (hereafter referred to as the Watson series) consists of a variety of problems in 2 to 15 variables, where the constraint index set T is either 1 or 2 dimensional. Many of these problems are one sided approximation problems, or have linear or convex objective functions. This set includes a problem which has an infinite number of global maximisers at its solution (thereby violating assumption 2.3), and also a problem whose solution does not satisfy the first order KKT conditions. Three additional test problems involving higher dimensional constraint index sets were also solved.

The problems in the Watson series, and the results generated by the algorithm presented herein, are given in detail in the following two sections. The problems are divided into two groups according to whether the constraint index set is one or two dimensional. These results are compared with those obtained by other authors in section 5.4. Extended results for some problems, in the Watson series, and other problems which violate any of the assumptions required to show the algorithm converges are presented in section 3.

No test problem with a constraint index set T having dimension greater than 2 appears in the set by Watson. Accordingly, three additional test problems with $3 \leq p \leq 6$ have been created. The performance of the algorithm on these problems is given in section 5.5. The next two sections after that look at the advantages of a two phase algorithm, and the merits of using a penalty function with two penalty parameters. In the final section the overall performance of the algorithm is discussed.

The superscript $\sharp$ denotes values taken by the various quantities in the final iteration of the SIP algorithm; these values are only approximations to those taken at the SIP's solution which the final iterate approximates. All computations were performed on a VAX 3100 workstation in double precision arithmetic. This gives approximately 16 digits accuracy; the machine precision being 1.39E-17.

5.1 One Dimensional Problems.

The results for the problems in the Watson series with $p = 1$ are presented in this section. For consistency the number assigned to each problem is the same as that in [99]. This is why problem 14 appears out of order. Problems not in the Watson set are tagged using letters. For each of the following problems a capping constraint (4.5) at $\theta_{\text{cap}} = 1$ was used. The capping constraint was struck only on problem 6. Also, $\theta_{\text{crossover}} = 1$ was used.

Problem 2.

$$\begin{aligned} f(x) &= \tfrac{1}{3}x_1^2 + x_2^2 + \tfrac{1}{2}x_1 \\ g(x,t) &= (1 - x_1^2 t^2)^2 - x_1 t^2 - x_2^2 + x_2 \\ T &= [0,1] \\ x^{(0)} &= (1,2)^T \end{aligned}$$

This problem actually has more than one solution. Using the initial point $(1,2)^T$ the algorithm found a different solution to the one found from this starting point by the algorithms of Watson, and Coope and Watson. Specifically, it found

$$x^\sharp = (-0.750000, 1.618034)^T; \quad \text{using the initial point} \quad x^{(0)} = (1,2)^T$$

$$f^\sharp = 2.430534; \;\; \theta^\sharp = 0; \;\; \mu^\sharp = 2.664; \;\; \nu^\sharp = 1;$$

$$\Gamma^\sharp = \{0\}; \;\; \mathcal{A}^\sharp - \Gamma^\sharp = \{1\}.$$

Using this starting point, the algorithm of Tanaka et al. found the same solution as the algorithm presented herein. By changing the starting point to the origin, Tanaka et al. were able to obtain the same solution as Coope and Watson. The results for this starting point are as follows:

$$x^\sharp = (-0.749999, -0.618034)^T \;\; \text{using the initial point} \;\; x^{(0)} = 0$$

$$f^\sharp = 0.194466; \quad \theta^\sharp = 0; \quad \mu^\sharp = 0.9791; \quad \nu^\sharp = 1$$

$$\Gamma^\sharp = \{0\}; \quad \mathcal{A}^\sharp - \Gamma^\sharp = \{1\}$$

No difficulties were experienced using either starting point.

Problem 3.

$$\begin{aligned} f(x) &= x_1^2 + x_2^2 + x_3^2 \\ g(x,t) &= x_1 + x_2 e^{x_3 t} + e^{2t} - 2\sin(4t) \\ T &= [0,1] \\ x^{(0)} &= (1,1,1)^T \end{aligned}$$

The solution found is:

$$x^\sharp = (-0.213313, -1.361451, 1.853547)^T$$

$$f^\sharp = 5.334687; \quad \theta^\sharp = 0; \quad \mu^\sharp = 6.365; \quad \nu^\sharp = 1;$$

$$\Gamma^\sharp = \{1\}; \quad \mathcal{A}^\sharp - \Gamma^\sharp = \{0\}.$$

This problem is quite innocuous, and there were no difficulties in solving it.

Problem 4.

$$\begin{aligned} f(x) &= \sum_{i=1}^{n} \frac{x_i}{i} \\ g(x,t) &= \tan(t) - \sum_{i=1}^{n} x_i t^{i-1} \\ T &= [0,1] \\ x^{(0)} &= 0 \end{aligned}$$

This problem can be viewed as a one sided approximation problem. That is, the polynomial which best (as measured using the L_1 norm) approximates $\tan(t)$ on the interval $[0,1]$ is sought, subject to the condition that the polynomial is not less than $\tan(t)$ on $[0,1]$. Using $n = 3$, no difficulties were experienced in obtaining the following results:

$$x^\sharp = (0.089096, 0.423053, 1.045259)^T$$

$$f^\sharp = 0.649042; \quad \theta^\sharp = 1.1E - 11; \quad \mu^\sharp = 1.871; \quad \nu^\sharp = 1$$

$$\Gamma^\sharp = \{0.3333, 1\}; \quad \mathcal{A}^\sharp - \Gamma^\sharp = \emptyset.$$

Problem 5.

$$\begin{aligned} f(x) &= \sum_{i=1}^{n} e^{x_i} \\ g(x,t) &= \frac{1}{1+t^2} - \sum_{i=1}^{n} x_i t^{i-1} \\ T &= [0,1] \\ x^{(0)} &= (1, 0.5, 0)^T \end{aligned}$$

Using $n = 3$, the results obtained were:

$$x^\sharp = (0.100661, -0.126884, -0.379721)^T$$

$$f^\sharp = 4.3012;\ \ \theta^\sharp = 1.0E-9;\ \ \mu^\sharp = 4.187;\ \ \nu^\sharp = 1;$$

$$\Gamma^\sharp = \{0.1061, 1\};\ \ \mathcal{A}^\sharp - \Gamma^\sharp = \emptyset.$$

No difficulties were experienced with this problem.

Problem 6.

$$\begin{aligned} f(x) &= \left[x_1 - 2x_2 + 5x_2^2 - x_2^3 - 13\right]^2 + \left[x_1 - 14x_2 + x_2^2 + x_2^3 - 29\right]^2 \\ g(x,t) &= x_1^2 + 2x_2 t^2 + e^{x_1+x_2} - e^t \\ T &= [0,1] \\ x^{(0)} &= (1,2)^T \end{aligned}$$

The algorithm needed more iterations and more multi-local optimisations to solve this problem than any other test problem in the Watson series, except the extended versions of problems 4 and 8. This is because the objective and constraint functions are so highly non-linear. Nevertheless the algorithm was easily capable of solving problem 6. The results are:

$$x^\sharp = (0.719961, -1.450487)^T$$

$$f^\sharp = 97.158852;\ \ \theta^\sharp = 1.9E-12;\ \ \mu^\sharp = 32.92;\ \ \nu^\sharp = 605.2;$$

$$\Gamma^\sharp = \{0\};\ \ \mathcal{A}^\sharp - \Gamma^\sharp = \emptyset.$$

Problem 14.

$$\begin{aligned} f(x) &= c^2 e^{x_1} + e^{x_2} \\ g(x,t) &= t - e^{x_1+x_2} \\ T &= [0,1] \\ x^{(0)} &= (0.8, 0.9)^T \end{aligned}$$

This problem does not appear in the original set given by Watson [99], but has been included later by Coope and Watson in [21]. It was designed so that the Hessian of the Lagrangian is indefinite at the solution. This presented no difficulty to the algorithm. The results are:

$$x^{\sharp} = (-0.095315, 0.095315)^T$$

$$f^{\sharp} = 2.2000; \quad \theta^{\sharp} = 0; \quad \mu^{\sharp} = 1.403; \quad \nu^{\sharp} = 1;$$

$$\Gamma^{\sharp} = \{1\}; \quad \mathcal{A}^{\sharp} - \Gamma^{\sharp} = \emptyset.$$

Problem K.

$$\begin{aligned} f(x) &= x_2^2 - 4x_2 \\ g(x,t) &= x_1 \cos(t) + x_2 \sin(t) - 1 \\ T &= [0, \pi] \\ x^{(0)} &= (0.9, 0)^T \end{aligned}$$

The *exact* solution of this problem is:

$$x^* = (0,1)^T; \quad f^* = -3;$$

$$\Gamma^* = \left\{\frac{\pi}{2}\right\}; \quad \text{and } \mathcal{A}^* - \Gamma^* = \emptyset.$$

Also $\mu^{\sharp} > 2$ is required for x^* to be a local minimum of the penalty function. This problem is used in section 5.7 to explore the effects of excessive penalty parameter values. The results are presented there.

5.2 Two Dimensional Problems.

For each one dimensional problem, excluding problem 6, the capping constraint, and quadratic penalty term played, in essence, only a passive role. It was decided to reduce the values of θ_{cap} and $\theta_{\text{crossover}}$ from 1 in order to increase their influence on the algorithms behaviour. The problems in this section were run with $\theta_{\text{cap}} = 0.01$ and $\theta_{\text{crossover}} = 0.1$. Results for the problems in this section with $\theta_{\text{cap}} = \theta_{\text{crossover}} = 1$ are listed in the appendix.

Problem 7.

$$\begin{aligned} f(x) &= x_1^2 + x_2^2 + x_3^2 \\ g(x,t) &= x_1(t_1 + t_2^2 + 1) + x_2(t_1 t_2 - t_2^2) + x_3(t_1 t_2 + t_2^2 + t_2) + 1 \\ T &= [0,1] \times [0,1] \\ x^{(0)} &= (2, -1, 1)^T \end{aligned}$$

The algorithm found the following approximation to the solution:

$$x^\sharp = (-1.0, 0.110088E-8, 0.222566E-8)^T$$

$$f^\sharp = 1.0000; \quad \theta^\sharp = 0; \quad \mu^\sharp = 3.162; \quad \nu^\sharp = 1$$

$$\Gamma^\sharp = \{(0,0)\}; \quad \mathcal{A}^\sharp - \Gamma^\sharp = \emptyset$$

No difficulties were experienced with this problem.

Problem 8.

$$\begin{aligned} f(x) &= x_1 + \tfrac{1}{2}x_2 + \tfrac{1}{2}x_3 + \tfrac{1}{3}x_4 + \tfrac{1}{4}x_5 + \tfrac{1}{3}x_6 \\ g(x,t) &= e^{t_1^2+t_2^2} - \left(x_1 + x_2 t_1 + x_3 t_2 + x_4 t_1^2 + x_5 t_1 t_2 + x_6 t_2^2\right) \\ T &= [0,1] \times [0,1] \\ x^{(0)} &= 0 \end{aligned}$$

The algorithm presented herein solved the problem for $n = 6$ without great difficulty, yielding the following solution:

$$x^\sharp = (2.580157, -4.109277, -4.109277, 4.247402, 4.532649, 4.247402)^T$$

$$f^\sharp = 2.4356; \quad \theta^\sharp = 5.7E-10; \quad \mu^\sharp = 1.801; \quad \nu^\sharp = 17.97$$

$$\Gamma^\sharp = \{(0,0)^T, (0,1)^T, (1,0)^T, (0.4000, 0.4000)^T\}; \quad \mathcal{A}^\sharp - \Gamma^\sharp = \emptyset$$

This problem was found to be harder than others in the series. This seems to be largely a consequence of the difficulty of finding the local and global maximisers of the constraint function. These maximisers were calculated using the NAG subroutine E04UCF, which is essentially NPSOL. It was found the limits on the maximum number of iterations of 250 for the feasibility and optimality phases of NPSOL were not enough. These were both increased to 450, which proved to be sufficient. With these limits, no further difficulties were encountered in solving this problem. To illustrate some of the difficulties involved in finding these maximisers, a 3 dimensional plot of $g(x^\sharp, t)$ is given in figure 2. The view is looking up the $t_1 = t_2$ line, with the closest (and lowest) point being $t_1 = t_2 = 0$.

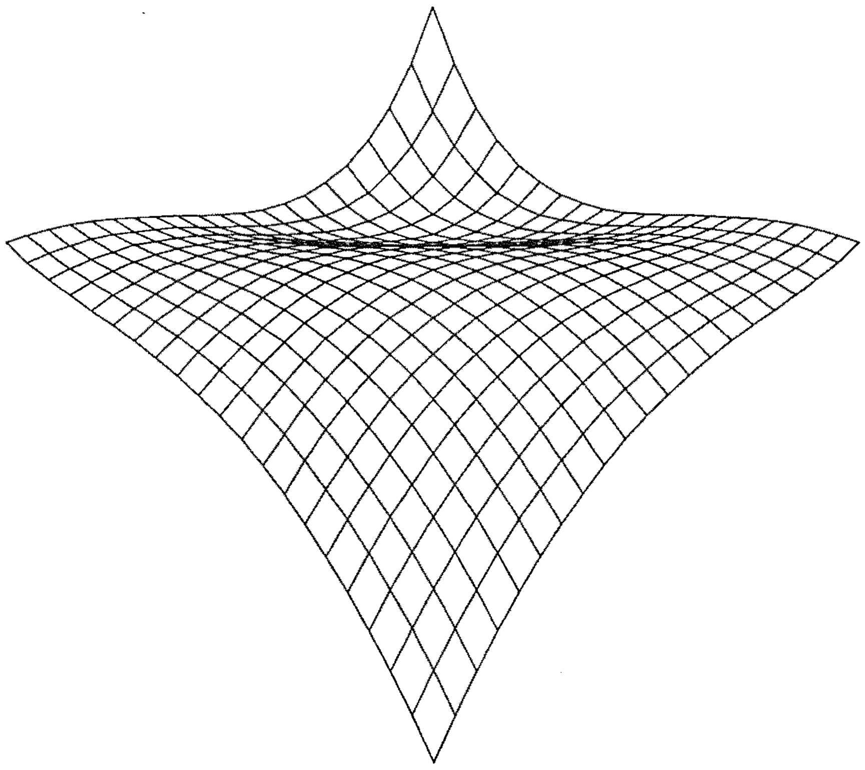

Figure 2. The 3D plot of $g(x,t)$ at $x = x^\sharp$ for problem 8, with $n = 6$.

A contour plot of $g(x^\sharp, t)$ is given in figure 3. From these plots it can be seen that the constraint function is very nearly flat in the region 'between' $(0,1)$, $(0.4,0.4)$ and $(1,0)$. The local maximisers in this region tend to change markedly from iteration to iteration, and at some iterates are difficult to locate.

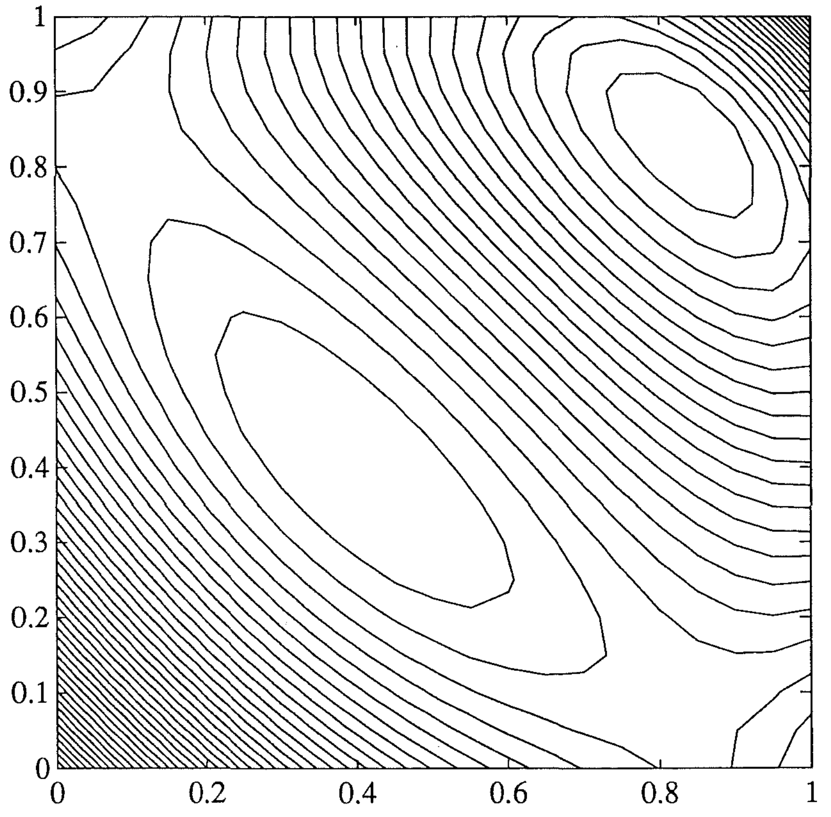

Figure 3. The contour plot of $g(x,t)$ at $x = x^\sharp$ for problem 8, with $n = 6$.

Problem 10.

$$\begin{aligned} f(x) &= 2x_1 + 4x_2 + x_3 \\ g(x,t) &= \sum_{i=1}^{3}(1 - x_i)w_i(t_1,t_2) - \tfrac{1}{2} \\ T &= [-1,4] \times [-1,4] \\ x^{(0)} &= 0 \end{aligned}$$

where

$$\begin{array}{lcll} w_1(t_1,t_2) & = & \frac{1}{t_1}\exp\left[-\frac{1+(t_2-1)^2}{t_1}\right] & \text{for } t_1 > 0, \\ w_2(t_1,t_2) & = & \frac{1}{t_1}\exp\left[-\frac{8+(t_2)^2}{4t_1}\right] & \text{for } t_1 > 0, \\ w_3(t_1,t_2) & = & \frac{1}{t_1-2}\exp\left[-\frac{1+(t_2+1)^2}{t_1-2}\right] & \text{for } t_1 > 2, \\ w_{1,2,3}(t_1,t_2) & = & 0 & \text{elsewhere.} \end{array}$$

This problem also has the following auxillary constraints:

$$0 \le x_i \le 1 \;\text{ for }\; i = 1,2,3.$$

The results obtained are as follows:

$$x^\sharp = (0, 0, 0.275265)^T$$

$$f^\sharp = 0.2753;\;\; \theta^\sharp = 4.296E-7;\;\; \mu^\sharp = 12.10;\;\; \nu^\sharp = 1.0$$

$$\Gamma^\sharp = \{(3.0349, -0.75370)^T\};\;\; \mathcal{A}^\sharp - \Gamma^\sharp = \{(1.3596, 0.9060)^T\}.$$

The auxillary constraints, and the linear objective function make this problem a very quick one for the algorithm to solve.

Problem 11.

This problem is the same as problem 10, except that the auxillary constraints are omitted. The results are:

$$x^\sharp = (1.541997, -2.1011440, 0.934505)^T$$

$$f^\sharp = -4.3861;\;\; \theta^\sharp = 8.806E-11;\;\; \mu^\sharp = 33.99;\;\; \nu^\sharp = 1$$

$$\Gamma^\sharp = \{(2.4610, -0.7327)^T, (1.9467, -0.5487)^T\};\;\; \Gamma^\sharp - \mathcal{A}^\sharp = \emptyset$$

The absence of the auxillary constraints exposes the algorithm more to the non-linearity of the semi-infinite constraint. This problem was easily solved, although more iterations were required than for problem 10.

Problem 12.

$$f(x) = 2x_1 + 4x_2 + x_3 + 30x_3^2(1+x_3)$$

T, g and $x^{(0)}$ are as for problem 10, and the auxillary constraints are included. The results are:

$$x^\sharp = (0.0, 0.355430, 0.111916)^T$$

$$f^\sharp = 1.9511; \quad \theta^\sharp = 3.241E-13; \quad \mu^\sharp = 39.75; \quad \nu^\sharp = 1.0$$

$$\Gamma^\sharp = \{(3.0363, -0.8209)^T\}; \quad \mathcal{A}^\sharp - \Gamma^\sharp = \{(1.2402, 0.9431)^T\}$$

This problem differs from problem 10 in that the objective function is quite non-linear. The solution is not found as quickly, but no real difficulty is experienced.

Problem 13.

This problem is the same as problem 12, except that the auxillary constraints are omitted. The algorithm generated these results:

$$x^\sharp = (-0.065575, 0.389132, 0.111357)^T$$

$$f^\sharp = 1.9502; \quad \theta^\sharp = 1.439E-9; \quad \mu^\sharp = 41.74; \quad \nu^\sharp = 1.0$$

$$\Gamma^\sharp = \{(3.0435, -0.8092)^T\}; \quad \mathcal{A}^\sharp - \Gamma^\sharp = \{(1.2142, 0.9503)^T\}$$

This problem illustrates nicely the need for a capping constraint. Without such a constraint the iterates diverged toward infinity. This occurred because the Lagrange multipliers at each iterate were estimated using the L_∞QP which approximates ϕ at the previous iterate. The cubic term in the objective function ensures that these estimates are ultimately always too low. The penalty parameters are never set to sufficiently large values to prevent the search directions being ones of increasing infeasibility. The sufficient descent criterion (2.5) is always satisfied as the reduction in the objective function is always larger than the increase in infeasibility. With the capping constraint in place no difficulties arose.

5.3 Other Results.

In this section results for the extended versions of problems 4, 5, and 8 are presented, along with results for problems 1 and 9. Some of these extended problems are extremely ill-conditioned, and are perhaps not the sort of problem for which a quasi-Newton algorithm is really intended. Problems 1 and 9 do not satisfy all of the assumptions required to show convergence of the algorithm, and so can not be regarded as valid test problems for the algorithm.

Problem 4: Extended Results.

Problem 4 was extended to higher dimensions simply by increasing n. This extended problem is an excellent test of an algorithms ability to cope with ill-conditioning. Solutions were generated for $n = 4$, 5, 6, and 8. In each case, except $n = 8$, the origin was used as the starting point. Following Coope and Watson, for $n = 8$ the starting point was chosen as the solution to the problem with $n = 6$, rounded to three decimal places, and with the seventh and eighth elements of the starting point set at zero.

For the $n = 6$ case, and especially for the $n = 8$ case, in several iterations the lengths of the steps accepted were very much shorter than the steps predicted by the approximating L_∞QP. The most serious offenders were two steps which occurred in the $n = 8$ case: the length of each accepted step being a mere $5.7E - 14$ times the length of the proposed step. To rectify this a simple trust region was added, and the problem was re-solved for all values of n except $n = 3$. The algorithm was capable of achieving an accuracy of $1.0E - 5$ for each of the above values of n. Coope and Watson also solve the problem with $n = 8$ to an accuracy of $1.0E - 8$. The best that the algorithm herein was able to achieve was an accuracy of $1.01E - 7$. It is perhaps useful to note at this point that Coope and Watson were using a Newton algorithm in 11 digit arithmetic (approximately), whereas double precision on the VAX 3100 is roughly equivalent to 16 digit arithmetic.

As this problem is quite pernicious, the accuracies required of the solutions to the subproblems which define the search direction, and the Maratos effect correction, were tightened from the default values for NPSOL to a maximum residual of $5.0E - 15$ in the first order KKT conditions, and a maximum of $1.0E - 12$ in the violation of the constraints. For $n = 6$, and $n = 8$ the additional bound

$$||H^{(k)}||_\infty < 10^6$$

was imposed on the approximation $H^{(k)}$ to the Hessian at each iteration. With the algorithm using the trust region, this bound was not struck when solving the problem with $n = 6$, and was struck once with $n = 8$.

Whilst this problem is an extremely useful test problem it is not one that is likely to arise in practice. The ill-conditioning is due to a poor choice of basis functions for the approximating polynomial, and could be avoided by a more sensible choice of basis functions.

The results, with the trust region used for $n = 4$, 5, 6, and 8, are:

For $n = 4$:

$$x^\sharp = (0.0, 1.145724, -0.624160, 1.035843)^T$$

$$f^\sharp = 0.623770; \;\; \theta^\sharp = 1.1E-12; \;\; \mu^\sharp = 2.768; \;\; \nu^\sharp = 1$$

$$\Gamma^\sharp = \{0, 0.499999, 1\}; \;\; \mathcal{A}^\sharp - \Gamma^\sharp = \emptyset$$

For $n = 5$:

$$x^\sharp = (0.006662, 0.890764, 0.600543, -0.934173, 0.993611)^T$$

$$f^\sharp = 0.617404; \;\; \theta^\sharp = 2.7E-9; \;\; \mu^\sharp = 2.271; \;\; \nu^\sharp = 1$$

$$\Gamma^\sharp = \{0.155043, 0.644938, 1\}; \;\; \mathcal{A}^\sharp - \Gamma^\sharp = \emptyset$$

For $n = 6$:

$$x^\sharp = (0.0, 1.023268, -0.240684, 1.221954, -1.388625, 0.941495)^T$$

$$f^\sharp = 0.616085; \;\; \theta^\sharp = 3.1E-12; \;\; \mu^\sharp = 4.121; \;\; \nu^\sharp = 1$$

$$\Gamma^\sharp = \{0.0, 0.276392, 0.723606, 1.0\}; \;\; \mathcal{A}^\sharp - \Gamma^\sharp = \emptyset$$

The solution for this case was obtained in 65 iterations and 169 multi-local optimisation calls using the algorithm without the trust region. The accuracy achieved was $2.4E-6$. With the trust region the algorithm found the solution in 57 iterations, and 119 multi-local optimisation calls. The upper bound on $\|H^{(k)}\|_\infty$ was implemented for both of these runs. It was struck on neither.

For $n = 8$:

$$\begin{aligned} x^\sharp &= (0, 1.002913, -0.053486, 0.709800, \\ &\quad -1.299410, 2.499339, -2.205324, 0.903575)^T \end{aligned}$$

$$f^\sharp = 0.615653 \;\; \theta^\sharp = 7.6E-16 \;\; \mu^\sharp = 2.244 \;\; \nu^\sharp = 1$$

$$\Gamma^\sharp = \{0.0, 0.172673, 0.499999, 0.827327, 1.0\} \;\; \mathcal{A}^\sharp - \Gamma^\sharp = \emptyset$$

Without the trust region the problem was solved to an accuracy of $3.9E-7$ in 119 iterations and 622 (!) multi-local optimisation calls. With the trust region, the algorithm took 84 iterations and 164 multi-local optimisation calls to reach a slightly higher accuracy. The upper bound on $\|H^{(k)}\|_\infty$ was struck once on the run with the trust region. It was struck several times on the run without the trust region. When the trust region was not used, in the final few iterations the algorithm generated prospective next iterates at which the estimated value of the penalty function exceeded the value at the current iterate, thereby indicating the search direction is potentially one of ascent.

Problem 5: Extended Results.

$$\begin{aligned} f(x) &= \sum_{i=1}^{n} e^{x_i} \\ g(x,t) &= \frac{1}{1+t^2} - \sum_{i=1}^{n} x_i t^{i-1} \\ T &= [0,1] \\ x^{(0)} &= (1, 0.5, 0)^T \end{aligned}$$

This extended problem was solved for $n = 8$, 10, 12, and 15. In each case $x = e_1$ (the first co-ordinate vector) was used as the initial point. The results for these values of n are summarised in table 5.4. No difficulties were experienced with this problem.

Problem 8: Extended Results.

Problem 8 was modified to yield problems in 10 and 15 variables. The 15 variable problem is listed here:

$$\begin{aligned} f(x) &= x_1 + \tfrac{1}{2}(x_2 + x_3) + \tfrac{1}{3}x_4 + \tfrac{1}{4}x_5 + \tfrac{1}{3}x_6 + \tfrac{1}{2}(x_7 + x_{10}) + \tfrac{1}{6}(x_8 + x_9) \\ &\quad + \tfrac{1}{5}(x_{11} + x_{15}) + \tfrac{1}{8}(x_{12} + x_{14}) + \tfrac{1}{9}x_{13} \\ g(x,t) &= e^{t_1^2+t_2^2} - \Big(x_1 + x_2 t_1 + x_3 t_2 + x_4 t_1^2 + x_5 t_1 t_2 + x_6 t_2^2 + x_7 t_1^3 + x_8 t_1^2 t_2 \\ &\quad + x_9 t_1 t_2^2 + x_{10} t_2^3 + x_{11} t_1^4 + x_{12} t_1^3 t_2 + x_{13} t_1^2 t_2^2 + x_{14} t_1 t_2^3 + x_{15} t_2^4\Big) \\ T &= [0,1] \times [0,1]. \end{aligned}$$

The 10 variable problem is obtained by omitting all terms containing any of the variables $x_{11}, \ldots, x_{15}$ from the 15 variable problem.

The algorithm presented herein solved the problem for $n = 10$ using the solution for $n = 6$ as the starting point. It failed to find a solution for $n = 10$ using the solution for $n = 6$ rounded to one decimal place as the starting point. A solution for $n = 15$ was not attempted.

As with the $n = 6$ case, NPSOL had difficulty in finding the local and global maximisers of the constraint function. For $n = 10$, in the later iterations NPSOL repeatedly halted prematurely, and on each such occasion returned one of these error messages:

- The first derivatives of $g(x,t)$ with respect to t do not tally with the finite difference estimates made by NPSOL.

- The current estimate of the maximiser of g does not satisfy the first order KKT conditions, and no improved point could be found during the final line search.

- The maximum number of optimality phase iterations has been reached, and a solution has not been found.

Using the $n = 6$ solution rounded to one decimal place as a starting point, the maximum iteration limits were increased to 850, 1850, and then 2850 in an attempt to prevent these error messages. As the maximum iteration limits were increased the third error message listed became much rarer, and the second rather more common. For $n = 10$, with the first six elements of the initial point were the the elements of $x^\sharp_{n=6}$, and the last four elements were set at zero. Using this initial point, the results for $n = 10$ are:

$$\begin{aligned} x^\sharp &= (1, 1.262635, 1.260352, -2.706753, -3.359771, \\ &\quad -2.701723, 3.162400, 3.235650, 3.076614, 3.159652) \end{aligned}$$

$$f^\sharp = 2.251282; \quad \theta^\sharp = 6.9E-8; \quad \mu^\sharp = 2.000; \quad \nu^\sharp = 1$$

$$\Gamma^\sharp = \{(0.8309, 0), (0, 0.8304), (1, 1), (0, 0), (0.5, 0.5), (1, 0), (0, 1)\};$$

$$\begin{aligned} \mathcal{A}^\sharp - \Gamma^\sharp &= \{(0.5467, 0.5437), (0.4951, 0.4474), (0.5133, 0.5411), \\ &\quad (0.5412, 0.5059), (0.3420, 0.6554), \\ &\quad (0.5243, 0.5785), (0.6250, 0.3333)\} \end{aligned}$$

An interesting feature of this solution is the number of points in $\Gamma^\sharp$, and in $\mathcal{A}^\sharp$. Coope and Watson list only the last five members of $\Gamma^\sharp$ as global maximisers, and do not mention any of the points in $\mathcal{A}^\sharp - \Gamma^\sharp$. This highlights the difficulty NPSOL had in finding the local maximisers of g in all but the first few iterations.

These last two problems of the Watson series are ones which the algorithm is not designed to solve. For the first of these (problem 1), the KKT conditions do not hold at the solution. Problem 9 does not satisfy assumption 2.3; at the solution the constraint function has an infinite number of global maximisers.

Problem 1.

$$\begin{aligned} f(x) &= \tfrac{1}{3}x_1^2 + x_2^2 + \tfrac{1}{2}x_1 - x_2 \\ g(x,t) &= x_1^2 + 2x_1x_2t - \sin(t) \end{aligned}$$

$$T = [0,2]; \qquad x^{(0)} = (1,2)^T$$

Neither the constraint qualification (1.4) the first order KKT conditions hold at the solution of this problem. This makes it awkward for the algorithm to solve. Using the original values of the parameters κ_1 and κ_2 (which govern the relative magnitudes of the penalty parameters and the infinity norm of the Lagrange multiplier estimates) the algorithm generated a sequence of iterates which converged to the solution. In spite of this, the gradient stopping condition was never satisfied. The condition on the step length eventually halted the algorithm; the length of the final step being $6.4E-8$. For this problem only, the maximum step length was increased from $1.0E-8$ to $1.0E-7$. The results for this are as follows:

$$x^\sharp = (-3.2452590E-7, 0.500000)^T$$

$$f^\sharp = -0.250000; \;\; \theta^\sharp = 1.1E-13; \;\; \mu^\sharp = 7.712E+6; \;\; \nu^\sharp = 1;$$

$$\Gamma^\sharp = \{0\}; \;\; \mathcal{A}^\sharp - \Gamma^\sharp = \{2\}.$$

A second run was performed with increased values of the parameters κ_1, and κ_2. Using $\kappa_1 = 4.2$ and $\kappa_2 = 4.5$ (in place of the original values of $\kappa_1 = 1.2$ and $\kappa_2 = 1.5$) ensured that ultimately $\mu > 4.2\|\lambda^\sharp\|_\infty$. With these values of κ_1 and κ_2, the gradient stopping condition was satisfied. The results are:

$$x^\sharp = (-0.000012, 0.500000)^T$$

$$f^\sharp = -0.250006; \;\; \theta^\sharp = 1.3E-10; \;\; \mu^\sharp = 24250; \;\; \nu^\sharp = 1;$$

$$\Gamma^\sharp = \{0\}; \;\; \mathcal{A}^\sharp - \Gamma^\sharp = \{2\}.$$

The exact solution is $x^* = (0, 0.5)^T$. Although the algorithm did not halt by satisfying the gradient condition with the unadjusted parameters, it did arrive at an iterate considerably more accurate than that required to satisfy the gradient stopping condition with the increased parameters. In many respects this represents a failure of the stopping conditions in particular, rather than of the algorithm as a whole.

Problem 9.

$$\begin{aligned}
f(x) &= -4x_1 - \tfrac{2}{3}(x_4 + x_6) \\
g(x,t) &= x_1 + x_2t_1 + x_3t_2 + x_4t_1^2 + x_5t_1t_2 + x_6t_2^2 - 3 - (t_1^2 - t_2^2)^2 \\
T &= [-1,1] \times [-1,1]
\end{aligned}$$

At the solution the number of global maximisers is infinite, and assumption 2.3 is not satisfied. The set of global maximisers of g at the solution is

$$\{(t_1,t_2) : t_2 = \pm t_1 \ \text{ with } \ t_1 \in [-1,1]\} \tag{5.1}$$

The constraint qualification (1.4) holds at the solution (for example, by choosing $u = e_1$), thus so do the first order KKT conditions. At the solution, ∇f is a linear combination of, for example, $\nabla_x g$ evaluated at the four corners of T and at the origin. Unfortunately, except exactly at the solution, at least one of these points is not a local maximiser of g.

Coope and Watson report that their algorithm made progress towards the solution, eventually stopping prematurely with the distance of the final iterate from the solution being approximately 0.002. In alternate iterations their algorithm found either approximations to the four corners of T, or the origin.

The algorithm of Tanaka et al. was successful on this problem. Unlike the other algorithms discussed in this chapter, at each iteration their algorithm matches up each local maximiser at the proposed next iterate with element of the set $\mathcal{A}^{(k)}$ at the current iterate $x^{(k)}$. If this match-up is not successful then approximations to the positions each unmatched local maximiser would have had at the current iterate are calculated. The set $\mathcal{A}^{(k)}$ is augmented with these approximations, and the proposed step is recalculated. This process was repeated until the matching process was successful. This strategy meant that their algorithm used approximations to a sufficiently large subset of Γ^* that the behaviour of the semi-infinite constraint could be accurately reflected in the finite set of linearised constraints used in the generation of each search direction.

The algorithm presented herein encountered difficulties similar to those found by the algorithm of Coope and Watson. In the earlier iterations $\mathcal{A}^{(k)}$ was alternately an approximation to the origin, or the four corners of T. In later iterations x_i, $i = 2, \ldots, 6$ became small, and g became very nearly flat on the set (5.1). From the point of view of NPSOL's stopping conditions, each such point eventually became regarded as a valid termination point for the local search. From then on each local search simply picked out a point on or near one of the lines joining the opposite corners of T which was close to the starting point of the search. As the iteration number increased $|\mathcal{A}^{(k)}|$ also increased until the maximum number of elements in $\mathcal{A}^{(k)}$ (25 points) was reached. Increasing the maximum size to 125 resulted in $|\mathcal{A}^{(k)}|$ eventually equalling 125. The sets $\mathcal{A}^{(k)}$ grew to this size by a process of inclusion: in each multi-local optimisation each local maximum found in the previous multi-local optimisation is used as a starting point for the local search. Hence, in the later iterations, each point in $\mathcal{A}^{(k)}$ is included in the set of local maximisers found in the following multi-local optimisation call. Additional searches from other points add extra members to this set of local maximisers. Thus $\mathcal{A}^{(k)}$ steadily grows in size.

With the maximum size of $\mathcal{A}^{(k)}$ set at 25 the results were as follows:

$$x^\sharp = (2.999, 1.493E-11, -7.145E-12, -3.865E-3, -9.797E-12, 3.865E-3)^T$$

$$f^\sharp = -12.00; \quad \theta^\sharp = 2.9E-10; \quad \mu^\sharp = 18.18; \quad \nu^\sharp = 68.92$$

The algorithm was stopped on the 41st iteration after taking a step of length less than $1.0E-10$ of the length of the step predicted by the L_∞QP. The error in the solution was about 0.006.

5.4 Comparison of the Various Algorithms.

The results are summarised in tables 5.1, 5.2, and 5.5. Where possible the results of Watson [99], Coope and Watson [21], and Tanaka, Fukushima, and Ibaraki [94] are also listed. Conn and Gould [19] solve those problems of the Watson series which have one dimensional T sets, however detailed results are not given in their paper, and hence can not be listed here. Bell [11] presents results for problem 4 of the Watson series. A comparison with these results is given in table 5.3.

In this chapter k and h denote the number of iterations, and the number of multi-local optimisations required to reach a solution. The subscripts W, CW, TFI, B, and P respectively denote results obtained by Watson [97], by Coope and Watson [21], by Tanaka, Fukushima, and Ibaraki [94], by Bell [11], and by the algorithm presented herein. The symbol Φ' denotes the magnitude of the most negative directional derivative of the L_1 penalty function at $x^\sharp$. Here $|\Gamma^*|$ is the number of global maximisers of g at the solution. It is not necessarily the number of global maximisers

problem	n	p	$\|\Gamma^*\|$	k_P	h_P	Φ'_P	k_{TFI}	h_{TFI}	Φ'_{TFI}
1	2	1	1	17	21	8.2E-6	17	19	4.8E-7
2	2	1	2	8	10	1.4E-8	5	11	2.7E-8
	2	1	2	7	8	4.9E-7	-	-	-
3	3	1	1	11	23	1.3E-6	9	12	5.5E-8
4	3	1	2	10	11	1.9E-6	5	15	2.7E-7
	6	1	4	57	119	7.7E-6	8	27	7.7E-6
	8	1	5	84	164	1.0E-7	3	14	3.4E-6
5	3	1	2	8	14	6.2E-6	4	9	6.8E-7
6	2	1	1	27	87	5.2E-6	16	19	1.3E-18
7	3	2	1	9	14	7.0E-9	2	4	0.0
8	6	2	4	34	40	4.1E-8	11	41	1.1E-7
	10	2	5	21	27	6.7E-7	12	56	3.4E-6
	15	2	?	-	-	-	10	57	3.8E-6
9	6	2	∞	41	192	-	2	6	0.0
10	3	2	1	2	3	1.2E-6	2	3	8.1E-7
11	3	2	2	10	18	9.8E-7	7	18	1.6E-14
12	3	2	1	9	17	3.8E-6	3	5	3.0E-12
13	3	2	1	11	22	7.5E-6	4	6	2.1E-15
14	2	1	1	6	7	8.1E-6	5	8	3.4E-7

Table 5.1: *A comparison of results with those obtained by Tanaka et al.*

problem	n	p	$\|\Gamma^*\|$	k_P	Φ'_P	k_{CW}	Φ'_{CW}	k_W	Φ'_W
1	2	1	1	17	8.2E-6	16	5.7E-6	16	1.1E-5
2	2	1	2	8	1.4E-8	7	2.5E-10	7	3.4E-7
	2	1	2	7	4.9E-7	-	-	-	-
3	3	1	1	11	1.3E-6	10	6.2E-12	14	6.7E-6
4	3	1	2	10	1.9E-6	5	5.4E-8	5	5.3E-8
	6	1	4	57	7.7E-6	20	6.4E-6	25	5.9E-6
	8	1	5	84	1.0E-7	16	7.4E-6	14	9.6E-6
5	3	1	2	8	6.2E-6	4	6.9E-6	5	7.5E-6
6	2	1	1	27	5.2E-6	9	1.1E-8	8	5.3E-6
7	3	2	1	9	7.0E-9	3	0.0	3	0.0
8	6	2	4	34	4.1E-8	9	1.1E-8	19	7.5E-7
9	6	2	∞	41	-	18	4.8E-2	9	3.5E-3
10	3	2	1	2	1.2E-6	3	2.8E-7	3	3.9E-9
11	3	2	2	10	9.8E-7	12	2.2E-7	19	3.0E-8
12	3	2	1	9	3.8E-6	4	1.7E-11	4	2.2E-10
13	3	2	1	11	7.5E-6	4	3.5E-7	4	3.6E-7
14	2	1	1	6	8.1E-6	5	8.2E-7	-	-

Table 5.2: *A comparison of results with those obtained by Watson, and those obtained by Coope and Watson.*

found by any of the algorithms. Where results are not given, or have not been calculated, a '-' appears. The number of global optimisations required by the algorithm of Tanaka, Fukushima, and Ibaraki [94] has been calculated from the results presented in their paper by observing that the number of quadratic programmes solved by their algorithm is one less than the number of global optimisations performed.

The only published results from the solution of SIP problems using a quasi-Newton algorithm known to the author are those by Bell [11]. These comprise problem 4 of the Watson series with $n = 3, \ldots, 6$. The three algorithms with which almost all comparisons are made here are all Newton type algorithms.

On all of the easier problems, (1, 2, 3, 4 with n = 3, 5, 10, 11, and 14) except number 7, the number of iterations taken was at most double that required by any of the Newton type algorithms. On these problems it also took less than twice the number multi-local optimisation calls that the algorithm of Tanaka et al. required.

The results for problem 5 using the higher values of n show a similar ratio of the numbers of iterations taken by the two types of algorithms.

The more non-linear problems (6, 12, and 13) produced greater discrepancies, but the algorithm presented herein had no difficulty in solving them.

The extended version of problem 4 was much more testing: the algorithm was able to solve it for the various values of n, however many more iterations and multi-local optimisation calls were needed than for the Newton type algorithms. In particular, the algorithm of Coope and Watson was able to achieve a higher accuracy on this problem (with $n = 8$) in lower precision arithmetic than the algorithm presented herein. The algorithm presented herein was able to locate all global maximisers, as were those of Tanaka et al. and of Coope and Watson. For $n = 6$, and 8 Watson's algorithm missed 1, and 2 of the global maximisers respectively.

Bell presents solutions to problem 4 for n = 3, 4, 5, and 6. A comparison with these is made in table 5.3. Bell's algorithm takes more iterations to reach a solution than the one presented herein. This is hardly surprising as Bell's algorithm starts by using quite coarse approximations to the global maximisers in the early iterations, and increases the accuracy required of these approximations as the solution process proceeds.

The margin between the Newton type algorithms, and the one presented herein was greatest on problem 8. The algorithm presented herein coped quite well with the $n = 6$ case, requiring one less multi-local optimisation call than the algorithm of

Tanaka et al. although many more iterations were taken. The $n = 10$ case was very different: the local search procedure used in the MOS subalgorithm experienced much difficulty in accurately calculating the local maximisers of the constraint function. Convergence was obtainable, but only by using the $n = 6$ solution as a starting point.

Generally, the algorithm herein found less accurate solutions than the other algorithms (excluding Bell's). This is to be expected, given the different natures of the various algorithms.

The results for the algorithm by Tanaka et al. [94] were generated using a fixed value of the penalty parameter for each problem. Different values were used for different problems. This method of selecting the penalty parameter value presupposes some knowledge of the problem. The values for the penalty parameter r chosen by Tanaka et al. were as follows: $r = 1$ was used on problem 3; $r = 10$ was used on problems 2, 4, 5, 7–10, and 14; $r = 100$ was used on problems 11–13; and $r = 1000$ was used on problems 1 and 6. In contrast the other four algorithms discussed here all start each problem with the penalty parameter(s) set equal to some fixed value, none of these initial values being greater than 1. These algorithms then adjust the penalty parameters accordingly as they proceed.

One would expect that a Newton type algorithm would be superior to the algorithm presented herein. Moreover the margin of superiority of a Newton method over a quasi-Newton method should be greater for SIP problems than for NLP problems. The rationale for this being that the local and global maximisers are also found using a Newton, or quasi-Newton method respectively. The latter will usually find less accurate approximations, thereby introducing larger errors into the linear constraints appearing in the L_∞QP. In light of this the algorithm presented herein fared well.

5.5 Higher Dimensional Problems.

Three problems involving constraint index sets of dimension greater than two were looked at. The first (problem S) was designed to be a non-trivial problem, but one which was not overly treacherous. The second (problem T) was chosen to be quite testing of the algorithms ability to keep track of local maximisers which merge into one another, and then split apart as the iteration number k is increased. Fortuitously, this problem is also a good test of an algorithms ability to cope with a

n	k_B	k_P
3	29	10
4	41	22
5	81	32
6	100	57

Table 5.3: *A comparison of results for the extended version of problem 4 with those obtained by Bell.*

n	$\|\Gamma^*\|$	k_P	Φ'_P	k_{CW}	Φ'_{CW}	k_{TFI}	Φ'_{TFI}
3	2	8	7.6E-6	4	6.9E-6	4	6.8E-7
8	2	7	4.3E-6	4	2.3E-8	2	1.2E-6
10	2	7	2.2E-6	4	1.2E-9	2	7.1E-7
12	2	7	8.7E-6	4	1.8E-8	3	9.2E-8
15	2	8	3.8E-8	4	1.3E-9	3	6.2E-8

Table 5.4: *A comparison of the results for the extended form of problem 5 with those obtained by Tanaka et al., and those obtained by Coope and Watson.*

Problem	n	p	$\lvert\Gamma^*\rvert$	k	h	cpu time
S	4	3	1	24	60	64.05
S	4	4	1	20	37	152.36
S	4	5	1	21	36	331.59
S	4	6	1	23	43	1081.71
T	4	3	4	23	48	125.52
T	4	4	4	20	39	456.29
T	4	5	4	26	68	988.23
T	4	6	4	26	64	5855.90
U	4	6	2	17	18	414.11

Table 5.5: *Results for the higher T dimensional problems.*

constraint function which has an almost flat region taking values close to the global maximum. The final problem was chosen to test an algorithms ability to exploit any lack of curvature of the constraint function along certain directions.

On all runs performed on the higher dimensional problems the trust region (4.6) was used, and θ_{cap} and $\theta_{\text{crossover}}$ were both set at 1. A summary of the results for these three problems is given in table 5.5. The symbol $|\Gamma^*|$ denotes the actual number of global maximisers active at the solution x^*. The cpu time is in seconds, and includes input/output time.

The results for problems S and T show a steady and large increase in computational time as p is increased. This is follows from the increased effort needed to solve the multi-local optimisation subproblems as the dimension of T increases. For each problem, the number of iterations is approximately constant, and the numbers of multi-local optimisation calls are all within a factor of 2 of each other.

Problem U is special: the linearity of g with respect to the last four components of t means that the problem can be solved much more quickly than the other problems with $p = 6$.

Problem S.

Problem S for $p = 6$ is as follows:

$$f(x) = x_1x_2 + x_2x_3 + x_3x_4$$

$$\begin{aligned}
g(x,t) &= 2(x_1^2+x_2^2+x_3^2+x_4^2)-6-2p \\
&\quad + \sin(t_1-x_1-x_4)+\sin(t_2-x_2-x_3) \\
&\quad + \sin(t_3-x_1)+\sin(2t_4-x_2)+\sin(t_5-x_3)+\sin(2t_6-x_4) \\
T &= [0,1]^p \\
x^{(0)} &= (1,1,1,1)^T
\end{aligned}$$

When p is less than 6, all terms in g involving any t_i with $i > p$ are deleted. This problem was solved for $p = 3$, 4, 5, and 6. The results are as follows:

For $p = 3$:

$$x^\sharp = (0.894135, -1.290617, 1.235788, -0.748821)^T$$

$$f^\sharp = -3.674298;\ \ \theta^\sharp = 0;\ \ \mu^\sharp = 1.012;\ \ \nu^\sharp = 1.0$$

$$\Gamma^\sharp = \{(1.7161, 1.5160, 2.0000)^T\};\ \ \mathcal{A}^\sharp - \Gamma^\sharp = \emptyset$$

For $p = 4$:

$$x^\sharp = (0.948247, -1.361576, 1.300981, -0.787553)^T$$

$$f^\sharp = -4.087086;\ \ \theta^\sharp = 2.689491E-8;\ \ \mu^\sharp = 0.932;\ \ \nu^\sharp = 1.0$$

$$\Gamma^\sharp = \{(1.7315, 1.5102, 2.0000, 0.1046)^T\};$$

$$\mathcal{A}^\sharp - \Gamma^\sharp = \{(1.7315, 1.5102, 2.0000, 2.0000)^T\}$$

For $p = 5$:

$$x^\sharp = (0.913759, -1.391873, 1.516069, -0.868445)^T$$

$$f^\sharp = -4.698634;\ \ \theta^\sharp = 0;\ \ \mu^\sharp = 0.7335;\ \ \nu^\sharp = 1.0$$

$$\Gamma^\sharp = \{(1.6161, 1.6950, 2.0000, 0.0895, 2.0000)^T\};$$

$$\mathcal{A}^\sharp - \Gamma^\sharp = \{(1.6161, 1.6950, 2.0000, 2.0000, 2.0000)^T\}$$

For $p = 6$:

$$x^\sharp = (0.960921, -1.456291, 1.581476, -0.905873)^T$$

$$f^\sharp = -5.135086;\ \ \theta^\sharp = 2.254294E-10;\ \ \mu^\sharp = 1.013;\ \ \nu^\sharp = 1.0$$

$$\Gamma^\sharp = \{(1.6258, 1.6960, 2.0000, 0.0573, 2.0000, 0.3325)^T\}$$

$$\begin{aligned}
\mathcal{A}^\sharp - \Gamma^\sharp &= \{(1.6258, 1.6960, 2.0000, 2.0000, 2.0000, 2.0000)^T \\
&= (1.6258, 1.6960, 2.0000, 2.0000, 2.0000, 2.0000)^T \\
&= (1.6258, 1.6960, 2.0000, 0.0573, 2.0000, 0.3325)^T \\
&= (1.6258, 1.6960, 2.0000, 0.0573, 2.0000, 2.0000)^T\}
\end{aligned}$$

Problem T.

$$\begin{aligned}
f(x) &= \sum_{i=1}^{4} x_i^2 - x_i \\
g(x,t) &= -\sum_{i=1}^{4} x_i^2 + \sum_{i=1}^{4} \frac{1}{1+w_i} \\
T &= [-3,3]^p \\
x^{(0)} &= (-2.25, -2.5, -2.75, -3.0)^T
\end{aligned}$$

where

$$\begin{aligned}
w_1 &= \sum_{j=1}^{p} [t_j - x_1]^2 \\
w_2 &= \sum_{j=1}^{p} [t_j - x_2(-1)^j]^2 \\
w_3 &= \sum_{j=1}^{p} [t_j - x_3(-1)^{j \operatorname{div} 2}]^2 \\
w_4 &= \sum_{j=1}^{p} [t_j - x_4(-1)^{(j+1) \operatorname{div} 2}]^2
\end{aligned}$$

This problem was solved for values of p ranging from 3 to 6. Loosely speaking, for $p = 3$ the constraint function consists of the constant $-\sum x_i^2$ plus the sum of four identically shaped humps each of maximum height 1. The peaks of the humps are centred on $t = x_1(1,1,1)^T$, $t = x_2(-1,1,-1)$, $t = x_3(1,-1,-1)^T$, and $t = x_4(-1,-1,1)^T$. At $x = 0$, g has a single peak of height 4 at $t = 0$. As x moves away from the origin the height of this peak decreases, and eventually the peak subdivides into two or more separate ones. For all sufficiently large $\|x\|$, g is non-positive on T. The infeasible region is symmetric under permutations of the elements of x. Roughly, the infeasible region is a roundish shape centred on the origin. The initial point lies on one side of the infeasible region, and the solution on the opposite side.

For each value of p there are four global maximisers active at the solution x^*. Lagrange multiplier estimates indicate that at most two of the four global maximisers are needed to satisfy the first order KKT conditions at x^*.

This problem was first solved for $p = 3$ and $p = 4$ using the algorithm in an unmodified form. From these results, it was observed that the constraint function was very nearly flat in between the global maximisers. In light of this the algorithm was then altered by using $\kappa_{\text{link}} = 0$. The results generated by this modified algorithm are presented below.

For $p = 3$:

$$x^\sharp = (0.659449, 0.659446, 0.659446, 0.659441)^T$$

$$f^\sharp = -0.898308; \quad \theta^\sharp = 0; \quad \mu^\sharp = 2.128; \quad \nu^\sharp = 1.0$$

$$\begin{aligned}\Gamma^\sharp &= \{(0.4502, 0.4502, 0.4502)^T, (0.4502, -0.4502, -0.4502)^T, \\ &\quad (-0.4502, 0.4502, -0.4502)^T\}\end{aligned}$$

$$\mathcal{A}^\sharp - \Gamma^\sharp = \{(0.0000, 0.0000, 0.0000)^T\}$$

In this problem the multi-local optimisation algorithm actually misses the global maximiser at $(-0.4502, -0.4502, 0.4502)^T$. The value taken by $g(x^\sharp, .)$ at the origin is -0.00382. The closeness of this value to zero indicates that $g(x^\sharp, t)$ is very nearly flat in the region 'between' the four global maximisers. This near flatness, and the fact that all the global maximisers lie in a small part of T make them quite difficult to locate.

The solution was found by the unmodified algorithm in 156.98 seconds, using 21 iterations and 43 multi-local optimisation calls. The unmodified and modified algorithms found the same local and global maximisers at the solution.

For $p = 4$:

$$x^\sharp = (0.659442, 0.659450, 0.659448, 0.659443)^T$$

$$f^\sharp = -0.898308; \quad \theta^\sharp = 1.734531E - 6; \quad \mu^\sharp = 2.317; \quad \nu^\sharp = 1.0$$

$$\begin{aligned}\Gamma^\sharp &= \{(0.4502, 0.4502, 0.4502, 0.6594)^T, \\ &\quad (-0.4502, -0.4502, 0.4502, 0.6594)^T, \\ &\quad (0.4502, -0.4502, -0.4502, 0.6594)^T\}\end{aligned}$$

$$\mathcal{A}^\sharp - \Gamma^\sharp = \{(0.0000, 0.0000, 0.0000, 0.6594)^T\}$$

Once again one of the global maximisers has been missed by the algorithm. The unmodified algorithm found all four global maximisers. The unmodified algorithm solved the problem in 868.31 seconds, taking 21 iterations, and 36 multi-local optimisation calls.

For $p = 5$:

$$x^\sharp = (0.636215, 0.636215, 0.636216, 0.636215)^T$$

$$f^\sharp = -0.925782; \quad \theta^\sharp = 0.0; \quad \mu^\sharp = 2.232; \quad \nu^\sharp = 1.0$$

$$\begin{aligned}\Gamma^\sharp &= \{(0.5420, 0.4941, 0.4941, 0.6362, 0.5420)^T,\\ &\quad(-0.5420, 0.4941, -0.4941, 0.6362, -0.5420)^T,\\ &\quad(-0.5420, -0.4941, 0.4941, 0.6362, -0.5420)^T,\\ &\quad(0.5420, -0.4941, -0.4941, 0.6362, 0.5420)^T\}\end{aligned}$$

$$\mathcal{A}^\sharp - \Gamma^\sharp = \emptyset$$

For $p = 6$:

$$x^\sharp = (0.617580, 0.617580, 0.617579, 0.617580)^T$$

$$f^\sharp = -0.944700; \quad \theta^\sharp = 0.0; \quad \mu^\sharp = 2.287; \quad \nu^\sharp = 1.0$$

$$\begin{aligned}\Gamma^\sharp &= \{(-0.5410, -0.5410, 0.5227, 0.6176, -0.5410, -0.5410)^T,\\ &\quad(0.5410, 0.5410, 0.5227, 0.6176, 0.5410, 0.5410)^T,\\ &\quad(-0.5410, 0.5410, -0.5227, 0.6176, -0.5410, 0.5410)^T,\\ &\quad(0.5410, -0.5410, -0.5227, 0.6176, 0.5410, -0.5410)^T\end{aligned}$$

$$\mathcal{A}^\sharp - \Gamma^\sharp = \emptyset$$

Problem U.

$$\begin{aligned}f(x) &= \sum_{i=1}^{4} \frac{1}{10} x_i^2 - x_i\\ g(x,t) &= \frac{x_4}{5}\sin\left(30t_1\sin(x_1) + 30t_2\cos(x_2)\right)\\ &\quad + \frac{x_3}{10}\sin\left(\frac{t_1t_2}{10}\right) + t_3x_1 + t_4x_2 + t_5x_3 + t_6x_4 - 4\\ T &= [-1,1]^6\\ x^{(0)} &= (3,2,1,0)^T\end{aligned}$$

The linearity of g in t_3, t_4, t_5, and t_6 means that finding the maximisers of g over T can be reduced from a search over six dimensions to a search over two dimensions. This feature was put in the problem to make checking the answer found by the algorithm much easier. Any checking procedure must find the global maximum of g at x^*, which can be extra-ordinarily difficult to do by hand. The algorithm made no allowance for the fact that the number of dimensions over which the local, and global maximisers of g are sought can be reduced from six to two.

The results for $p = 6$ are

$$x^\sharp = (1.173288, 1.179673, 1.142275, 0.412150)^T$$

$$f^\sharp = -3.483097; \quad \theta^\sharp = 2.374750E - 11; \quad \mu^\sharp = 1.462; \quad \nu^\sharp = 1.0$$

$$\Gamma^\sharp \;=\; \{(1,1,1,1,1,1)^T, (-0.8928, -1, 1, 1, 1, 1)^T\}$$

$$\begin{aligned} \Gamma^\sharp - \mathcal{A}^\sharp \;=\; & \{(0.1420, 0.3434, 1, 1, 1, 1)^T, \\ & (-0.3124, -0.7556, 1, 1, 1, 1)^T, \\ & (-0.1988, -0.4808, 1, 1, 1, 1)^T, \\ & (-0.0852, -0.2061, 1, 1, 1, 1)^T, \\ & (0.7793, 1, 1, 1, 1, 1)^T, \\ & (-0.6657, -1, 1, 1, 1, 1)^T, \\ & (0.0284, 0.0687, 1, 1, 1, 1)^T, \\ & (0.2556, 0.6182, 1, 1, 1, 1)^T, \\ & (0.3692, 0.8929, 1, 1, 1, 1)^T, \\ & (-0.4385, -1, 1, 1, 1, 1)^T, \\ & (0.5521, 1, 1, 1, 1, 1)^T\} \end{aligned}$$

This problem illustrates very nicely the advantages of using a multi-local optimisation algorithm based on a quasi-random set of points rather than a grid. The constraint function $g(x, .)$ looks approximately like a sheet of corrugated roofing iron in the first two dimensions of t. The period of the oscillations being just over one tenth of the length of one edge of T. If these oscillations are to be detected using a grid, then there needs to be about 20 points along each axis — enough for one point on the peak of each corrugation, and one in each of the intervening troughs. If the points in the troughs are omitted then the algorithm may misinterpret the points lying on the peaks as a set of points on an almost flat surface; most global maximisers could then easily be missed. A six dimensional grid with 20 points per side has 64 million points. If the algorithm takes 18 multi-local optimisation calls to solve the problem (as did the algorithm presented herein), then a total of 1,152 million constraint function evaluations are required. The VAX 3100 takes about 5 minutes and 50 seconds of cpu time to perform one million evaluations of g. The time needed to solve the problem would be over $4\frac{1}{2}$ days!

5.6 Using a NLP First Phase.

The algorithm was modified to incorporate a first phase based on a discrete subset of T. This first phase was similar to that described by Hettich, and others [50, 53]: it approximates the SIP by a Non-Linear Programme (NLP), and solves this non-linear programme to find an estimate of the solution to the semi-infinite programme. In the second phase the SIP algorithm was used to refine the solution of the approximating NLP.

The objective function of the NLP was identical to that of the SIP. The set of constraints of the NLP was

$$\{g(x,t) \leq 0 : t \in \mathcal{H}_m\},$$

where $\mathcal{H}_m$ is the set of the first m points of the Halton sequence.

The algorithm used to solve the NLP was identical to that used to solve the SIP, except that $\mathcal{A}^{(k)} = \mathcal{H}_m$ was used at each iteration instead of choosing $\mathcal{A}^{(k)}$ as the set of global (and other local) maximisers of $g(x^{(k)}, .)$.

At each NLP iteration every NLP constraint arising from replacing T with $\mathcal{H}_m$ was included in the QP subproblem used to determine the search direction. Hettich and Gramlich [50, 53] give a more sophisticated strategy for handling the constraint set arising from the discretization of T; one which does not require all NLP constraints to be included in every subproblem. A sketch of their approach is given in chapter 1. Both approaches solve the same subproblem.

Once the NLP is solved to the required accuracy, the SIP algorithm is applied with the NLP solution as the starting point. No alterations were made to the SIP algorithm. It did, however, use as starting values the first phase final values of the penalty parameters, and the also final estimate of the Hessian calculated in solving the NLP.

Problem S with $p = 4$ was used to test this two phase algorithm. Results were generated for various accuracies required of the NLP solution, and also for various values of m, where m is the number of constraints in the NLP. These are listed in tables 5.6 and 5.7.

In table 5.6, the parameter *Tol* represents the accuracy required of the NLP's solution. More precisely, *Tol* is both the maximum NLP constraint violation permitted, and the maximum (2-norm) residual of the derivative of the NLP's Lagrangian allowed at an acceptable solution to the NLP. The row labelled $Tol = \infty$ in table 5.6

Tol	First Phase					second phase			combined
	k_1	h_1	time	$f^\sharp_{NLP}$	$\|x^\sharp_{NLP} - x^\sharp\|$	k_2	h_2	time	cpu time
∞	0	0	0.00	$+3.000$	2.9775	20	37	152.36	152.36
1.0E-1	24	43	20.80	-4.139	0.2875	9	13	30.29	51.09
1.0E-2	26	45	21.72	-4.132	0.3645	10	14	32.59	54.31
1.0E-5	28	47	22.54	-4.132	0.3641	10	14	32.02	54.56

Table 5.6: *Results for a two phase algorithm on problem S with* $p = 4$. *Here the number of constraints in the first phase has been fixed at 160, and the accuracy to which the NLP was solved has been varied.*

m	First Phase					second phase			combined
	k_1	h_1	time	$f^\sharp_{NLP}$	$\|x^\sharp_{NLP} - x^\sharp\|$	k_2	h_2	time	cpu time
50	19	23	6.46	-4.340	0.7170	13	17	45.11	51.57
160	28	47	22.54	-4.132	0.3641	10	14	32.02	54.56
500	21	30	51.89	-4.128	0.1080	10	15	34.50	86.39
1600	21	25	173.58	-4.128	0.1080	11	14	33.12	206.70

Table 5.7: *Results for a two phase algorithm on problem S with* $p = 4$. *Here the number of constraints in the NLP has been varied, and each NLP was solved to an accuracy of 1.0E-5.*

contains the results obtained by applying the SIP algorithm proper without an NLP first phase. The rest of the legend for tables 5.6 and 5.7 is as follows: k_1 and k_2 are respectively the number of iterations performed in solving the NLP, and the SIP; h_1 is the number times the set of NLP constraints is evaluated, and h_2 is the number of multi-local optimisation calls made in solving the SIP; $f^{\sharp}_{NLP}$ is the value of f at the solution of the NLP; and $\|x^{\sharp}_{NLP} - x^{\sharp}\|$ is the Euclidean distance between the NLP's and SIP's solutions. For the case when $Tol = \infty$, $x^{\sharp}_{NLP} = x^{(0)}$ and $f^{\sharp}_{NLP} = f(x^{(0)})$ have been used. The cpu times required to complete the first, and the second phases are listed in the two columns headed 'time.' The total time required to solve the problem is listed under the heading 'combined cpu time.' Unfortunately it was not possible to separate the input/output times from the cpu times. The input/output times are of the order of 4 to 10 seconds for the test runs listed here, with the input/output time for each run being approximately proportional to $k_1 + k_2$. In spite of this uncertainty in the times, they still provide useful information on the effects of using an NLP first phase.

The results show that the use of a first phase reduces the total time required to solve the problem. The results show that a relatively coarse discretization of the semi-infinite constraint performed better than a finer discretization. As the discretization became finer the time taken to complete the first phase increased accordingly. Curiously, the time taken to complete the second phase was relatively independent of the discretization; the second phase times for $m = 160$, 500, and 1600 being very similar. The $m = 500$ and $m = 1600$ first phases found the same approximation, to four decimal places. The differences in their second phases can be attributed to the fact that the NLP runs had different final penalty parameter values, and different final estimates of the Hessian, both of which were inherited by the second phase. The $m = 160$ first phase found a considerably less accurate approximation, and yet the corresponding second phase was just as fast.

Similarly, these results for $m = 160$ and varying values of Tol show that there is little to be gained by solving the NLP to great accuracy. Discretizing the semi-infinite constraint introduces an error between the solution of the NLP (x^{*}_{NLP}), and x^{*}. There is little point in reducing the error in the calculated value of $x^{\sharp}_{NLP}$ too much below $\|x^{*}_{NLP} - x^{\sharp}\|$.

5.7 Advantages of an Extra Penalty Parameter.

The penalty function ϕ is a hybrid of the standard Single Parameter Exact non-differentiable Penalty Function (SPEPF) and the classical Quadratic Penalty Function (QPF). These are respectively ϕ with $\nu \equiv 0$, and with $\mu \equiv 0$. The characteristics of this hybridization are investigated by varying the threshold parameter $\theta_{\text{crossover}}$. When θ is above this threshold value, any adjustments to the penalty parameters are made to ν, below this threshold the adjustments are made to μ. If $\theta_{\text{crossover}}$ is very large, then the algorithms behaviour imitates that of an algorithm based on a single parameter exact penalty function. If θ is very small, then the algorithm mimics a quadratic penalty function based algorithm.

Problem 6 was chosen as the test problem on which to explore the effects of altering $\theta_{\text{crossover}}$ because it is highly non-linear, and because the standard form of the algorithm (chapter 4, section 1) requires many iterations and multi-local optimisations to solve it. Thus any changes in the algorithm's performance wrought by changing $\theta_{\text{crossover}}$ should be clearly detectable. It was necessary to limit the maximum length of the proposed step: $\|s\|_\infty \leq 2$ was used. Without this bound, very large steps were predicted, and attempted. When the programme attempted to evaluate ϕ at the proposed new iterate, the exponential terms in the constraint function caused an overflow to occur — thereby crashing the programme. The results are presented in table 5.8. The first and last rows of table 5.8 list the results obtained by using a single parameter exact penalty function ($\nu \equiv 0$), and a quadratic penalty function ($\mu \equiv 0$) respectively. For these two rows the initial penalty parameter values were $\mu = 0.1$ and $\nu = 1.0$ respectively. For all other rows, $\mu = 0.1$ and $\nu = 1.0$ were the initial values, with $\theta_{\text{crossover}}$ as listed. Two sets of results were generated: the first set was computed using the algorithm without a capping constraint, and the second set was calculated by the algorithm with a capping constraint set at $\theta_{\text{cap}} = 1$.

The results show that without the capping constraint, the pure non-differentiable penalty function needed over twice as many iterations, and more than four times as many multi-local optimisation calls as the hybrid penalty function with $\theta_{\text{crossover}} = 1$. With $\theta_{\text{crossover}} = 100$, the algorithm did not alter ν, in which case ϕ was effectively the sum of the single parameter non-differentiable penalty function and a $+\frac{1}{2}\theta^2$ term. Even this simple alteration produced a significant improvement in performance. Using lower values of $\theta_{\text{crossover}}$ improved performance further.

The SPEPF performed so poorly without either a non-zero ν or a capping constraint because many iterations are needed before a sufficiently large value of μ is obtained. With $\nu \equiv 0$, and without a capping constraint, $\mu^{(k)}$ can be at most $\kappa_2\mu^{(k-1)}$, where $\kappa_2 = 1.5$ was used. This is a consequence of using the Lagrange multiplier estimates from the L_∞QP, which means that $\|\lambda^{(k-1)}\|_1$ is bounded above by $\mu^{(k-1)}$. The updating scheme for the penalty parameters is designed to ensure that $\mu^{(k)}$ is at most $\kappa_2\|\lambda^{(k-1)}\|_1$. So, if $\mu^{(0)}$ is small, many iterations may be needed before a reasonable value of μ is reached. If $\nu > 0$ then $\mu^{(k)} \leq \kappa_2(\mu^{(k-1)} + \nu^{(k-1)}\zeta^{(k-1)})$ and μ can grow faster than for the SPEPF.

One might expect that the quadratic penalty function performance would be much worse than that of the hybrid penalty function, however the results do not bear this out. All calculations in all test runs were performed in double precision, which is approximately 16 digits. This is enough to cope with the ill-conditioning arising from the high value of ν, whilst still achieving the required accuracy of about five digits. However the deficiencies of the quadratic penalty function are well known.

With the capping constraint in place, the differences between the various penalty functions were not great. The result for $\theta_{\text{crossover}} = 100$ appears to be something of an anomaly. For $\theta_{\text{crossover}} \leq 10$ the uncapped algorithm consistently performed better than the capped algorithm; the difference however was not large.

To investigate the relative merits of the SPEPF and the hybrid penalty function further the algorithm was modified to permit arbitrarily large increases in the penalty parameters. This was accomplished by solving the L_∞QP subproblem with μ reset to a very large number: here $1.0E8$ was used. The Lagrange multiplier estimates calculated whilst solving this L_∞QP were then used to update the penalty parameter values in accordance with the rules given in section 1 of chapter 4. The search direction was then calculated by re-solving the L_∞QP with the new penalty parameter values. The relevant results are presented in table 5.9. In these, the SPEPF does better than the hybrid penalty function with $\theta_{\text{crossover}} = 1$. An examination of the sequences of iterates generated shows that the hybrid penalty function with $\theta_{\text{crossover}} = 1$ allows the sequence of iterates to penetrate deeper into the infeasible region than does the SPEPF. The deeper forays into the infeasible region take longer to correct. The presence or absence of a capping constraint had no effect on the numerical results here.

Allowing arbitrarily large increases in μ and ν does not *quite* make the capping constraint irrelevant. The method used to estimate the Lagrange multipliers when unlimited increases are permitted ensures that the capping constraint will never be active at the solution of the L_∞QP; the capping constraint itself becomes redundant. However, using the capping constraint also imposes the following extra requirement on the line search:

$$\text{if } \theta^{(k)} > \theta_{\text{cap}} \text{ then } \theta^{(k+1)} \leq \theta^{(k)} \text{ is required.}$$

This extra condition is still able to influence how the algorithm selects each iterate.

Additionally, the possibility of allowing reductions in the penalty parameter values as well as unlimited increases was also looked at. To stop the algorithm from endlessly increasing and decreasing the penalty parameters it was necessary to assign μ and ν minimum values $\mu_{\min}$ and $\nu_{\min}$: initially $\mu_{\min} = 0.1$ and $\nu_{\min} = 1.0$ were used. Each time a penalty parameter was decreased, the corresponding minimum value was subsequently doubled. Without these minimum values, the possibility that the algorithm may fail by cycling is admitted.

The necessary changes were implemented as follows. Firstly, $\lambda^{(k)}$ was calculated as described earlier for the case of arbitrarily large increases. Any consequent increases in the penalty parameters were then made. Immediately following this, if $\theta^{(k)} \leq \theta_{\text{crossover}}$ then decreasing either or both of the penalty parameters was considered. If

$$\mu^{(k)} > \max\left(1.8\|\lambda^{(k)}\|_1, \mu_{\min}\right)$$

then the following adjustments were made, in this order:

$$\nu^{(k)} \longleftarrow \nu^{(k)} + \frac{\mu^{(k)} - \max\left(1.5\|\lambda^{(k)}\|_1, \mu_{\min}\right)}{\theta_{\text{crossover}}}$$

$$\text{and} \quad \mu^{(k)} \longleftarrow \max\left(1.5\|\lambda^{(k)}\|_1, \mu_{\min}\right).$$

The first adjustment ensures $\mu + \nu\theta$ is decreased only on the part of the infeasible region where $\theta < \theta_{\text{crossover}}$. For many problems this is the part of the infeasible region which borders on the feasible region. If

$$\nu^{(k)}\theta_{\text{crossover}} > \max\left(50\|\lambda^{(k)}\|_1, \nu_{\min}\theta_{\text{crossover}}\right),$$

then $\nu^{(k)}$ was reset as follows:

$$\nu^{(k)} \longleftarrow \max\left(\frac{4\|\lambda^{(k)}\|_1 - \mu^{(k)}}{\theta_{\text{crossover}}}, \nu_{\min}\right).$$

$\theta_{\text{crossover}}$	Not Capped				Capped			
	k_P	h_P	$\mu^\sharp$	$\nu^\sharp$	k_P	h_P	$\mu^\sharp$	$\nu^\sharp$
SPEPF	42	130	977.1	0	21	40	334.0	0
100	35	99	16970	1.0	16	25	431.0	1.0
10	16	34	210.8	24.64	19	39	429.4	86.97
1	16	31	46.49	234.5	17	34	75.13	86.97
0.1	16	31	22.42	877.0	18	35	7.583	86.97
0.01	17	32	11.79	877.0	20	38	7.341	1182
1.0E-4	19	34	7.398	55760	21	39	7.381	78030
1.0E-6	21	36	7.383	1.5E+7	24	42	7.383	5.2E+6
QPF	22	42	0	1.0E+9	22	39	0	2.1E+6

Table 5.8: *Variations of the algorithms performance on problem 6 with respect to changes in $\theta_{crossover}$. When θ exceeds $\theta_{crossover}$, ν is altered, otherwise μ is altered. The first and last rows are for a single parameter exact non-differentiable penalty function ($\nu \equiv 0$), and for a quadratic penalty function ($\mu \equiv 0$) respectively.*

If $\theta > \theta_{\text{crossover}}$ then the penalty parameters were not reduced.

The results for this are presented in table 5.10. They show that allowing decreases in the penalty parameters led to improvements in the performance of the algorithm in most cases. Once again the SPEPF did better than the hybrid penalty function.

The best result is that of the original algorithm, with $\theta_{\text{cap}} = 1$ and $\theta_{\text{crossover}} = 100$. Other than this apparently rather anomalous result, the best results were obtained using the hybrid penalty function with only restricted increases in the penalty parameters permitted, and without a capping constraint.

Problem K was used to investigate the effects of excessively high values of the penalty parameters. This problem contains a single convex constraint. The initial point lies near this constraint, and the solution lies on it. Between the initial point and the solution the gradient of the objective function points into the constraint. This problem tests an algorithm's ability to generate a sequence of iterates which efficiently skirts around the convex constraint to the solution. As the penalty parameters are increased, the constraint becomes more nearly impenetrable — forcing the algorithm to generate iterates which are either feasible, or only marginally infeasible.

$\theta_{\text{crossover}}$	Not Capped				Capped			
	k_P	h_P	$\mu^\sharp$	$\nu^\sharp$	k_P	h_P	$\mu^\sharp$	$\nu^\sharp$
SPEPF	21	40	334.4	0	21	40	334.4	0
100	21	40	334.4	1.0	21	40	334.4	1.0
1	28	66	1625	3.2E+7	28	66	1625	3.2E+7
0.01	28	68	407.9	3.2E+7	28	68	407.9	3.2E+7

Table 5.9: *The hybrid PF, and the SPEPF with unlimited increases in the penalty parameters permitted.*

$\theta_{\text{crossover}}$	Not Capped				Capped			
	k_P	h_P	$\mu^\sharp$	$\nu^\sharp$	k_P	h_P	$\mu^\sharp$	$\nu^\sharp$
SPEPF	19	34	6.490	0	19	36	7.006	0
100	19	34	6.490	1.142	19	36	7.006	1.237
1	24	37	6.781	4.000	26	41	6.711	10.57
0.01	32	65	7.365	19.64	29	63	7.340	19.57

Table 5.10: *The hybrid PF, and the SPEPF with unlimited increases, and with decreases in the penalty parameters permitted.*

μ	ν	k_P	h_P
3	0	8	15
10	0	9	17
30	0	16	43
100	0	28	82
300	0	35	124
1000	0	44	139
3	1	8	15
3	10	8	15
3	1E+2	9	17
3	1E+3	22	60
3	1E+4	20	54
3	1E+5	38	105
3	1E+6	40	152
3	1E+7	53	152

Table 5.11: *Results for problem K with various values of the penalty parameters. Both penalty parameters were fixed during each run.*

Results were generated for a variety of values of μ and ν. These parameters were kept constant during each run of the algorithm. The results are listed in table 5.11 in two groups. The first is for the single parameter exact penalty function: $\nu \equiv 0$ is used for each of these runs. The second group is for the hybrid penalty function. In the latter group $\mu = 3$ has been used, as this is κ_2 (=1.5) times the minimum value of μ needed to make the solution of problem K a local minimum of ϕ.

The results show that the number of iterations and multi-local optimisation calls required to solve the problem rises with increasing values of either penalty parameter. The degradation in the performance of the SPEPF algorithm brought about by increasing μ by a factor γ is roughly the same as the degradation in performance of the hybrid penalty function algorithm wrought by increasing ν by a factor γ^2.

5.8 Other Comments.

A simple upper bound (of 2) on the infinity norm of the proposed step was used on all problems. On problem 6 it was needed to prevent an overflow occurring. On problem 4 (excepting $n = 3$) and on problems S, T, and U the more sophisticated trust region:

$$\|s^{(k)}\|_\infty \leq 4\|x^{(k)} - x^{(k-1)}\|_\infty$$

was used. On problem 4 this lead to improved performances compared to those obtained when the simple bound was used.

As stated in chapter 2, the $\frac{1}{2}\nu\theta^2$ penalty term was included to provide a mechanism for reducing the risk that μ would be set at an excessively high value. Problem K was designed specifically to test the effects of including the second penalty term. As expected [20], excessively high values of either penalty parameter impair the algorithm's performance. These results also show that an excessively high value of ν degrades the algorithm's performance less than a correspondingly high value of μ. Accordingly, the scheme used to update the penalty parameters should try to avoid selecting unnecessarily large values, particularly for μ. Unfortunately such values may be unavoidable for a variety of reasons, notably:

- Reductions in the penalty parameters are not permitted, and the initial values of the penalty parameters are excessive.
- A highly infeasible iterate is encountered, and one or other penalty parameter must be large if near feasibility is to be subsequently attained.
- The Lagrange multiplier estimates are highly inaccurate.

The inclusion of the second penalty parameter does reduce the susceptibility to the last two causes listed. However if $\mu^{(0)}$ is excessive, then the $\frac{1}{2}\nu\theta^2$ term is of little use. In spite of the results, permitting only restricted increases of the penalty parameters could easily lead to excessive values on some problems, especially as a result of the second reason listed. Moreover, many iterations may be wasted before μ and ν are large enough to achieve feasibility. In addition, the restrictions on the increases in μ and ν are a product of using the Lagrange multiplier estimates from the L_∞QP's solution. If the Lagrange multiplier estimates are calculated in some other way (for example, first order estimates are used) then any restriction of the form $\mu^{(k)} \leq \kappa_2 \mu^{(k-1)}$ becomes essentially ad hoc in nature. Hence permitting both increases and decreases is apparently advantageous. It should be noted that on some problems, permitting decreases in μ and ν may allow the algorithm to cycle until the minimum values of the penalty parameters become high enough to force convergence. In such cases the early iterations are likely to achieve little other than waste time. It appears there is no 'right' strategy: the best scheme depends on the nature of the problem being solved. It is reasonable to expect that, on average, allowing both increases and decreases would be the better strategy on more difficult problems.

The results from problem 6 indicate that the performances obtained using the various strategies are approximately within a factor of two of one another.

The necessity of a capping constraint is closely linked to the method used to estimate the optimal Lagrange multipliers. If the Lagrange multiplier estimates from the L_∞QP are used, then the estimates are bounded in the 1-norm by $\mu^{(k)} + \nu^{(k)}\zeta^{(k)}$. In that case the capping constraint is needed to eliminate the possibility that $\theta^{(k)} \to \infty$ as $k \to \infty$. When this occurs, at each iteration, the increases in the penalty terms are always offset by the reduction in the objective function. If first order estimates of the Lagrange multipliers are used, then each search direction will be one of non-ascent for θ, although the restriction that $\theta^{(k+1)} \leq \theta^{(k)}$ may still be needed if θ is positive and large.

What is a good choice for the the capping constraint value θ_{cap} varies from problem to problem. If θ_{cap} is too small then the sequence of iterates may be forced to follow closely a tightly curving constraint: a task that can require many iterations. In contrast, if θ_{cap} is too large, then it is possible for the sequence of iterates to penetrate deeply into the infeasible region. This risks having to set one or other penalty parameter to a large value in order to regain near feasibility. More seriously, it is possible that $\theta(x)$ has strict local minimisers in the infeasible region. For sufficiently large μ and ν, there will be corresponding infeasible local minimisers in ϕ. Convergence to such a local minimiser is tantamount to failure of the algorithm. An appropriate value of θ_{cap} may lessen the risk of an infeasible local minimiser of ϕ 'trapping' the sequence of iterates.

On problem S, with $n = 4$, the use of a NLP first phase reduced the time required to solve the problem by almost a factor of three. The best results were obtained by using a coarse discretization of the semi-infinite constraint, and then calculating the solution of the resultant NLP to a low accuracy only. The accuracy of the approximation to the solution found by the first phase could be improved by either using a finer discretization of the semi-infinite constraint, or solving the resulting NLP to a higher accuracy, or both. However, the benefits of a more accurate initial value for the second phase were more than offset by the extra effort required to obtain it.

Hettich and Gramlich [50, 53] give a more sophisticated algorithm for discretizing the semi-infinite constraint, and solving the NLP so obtained; the approach taken herein was simply to solve the NLP directly using a quasi-Newton algorithm. The results Hettich and Gramlich [50, 53] list indicate that their algorithm would be significantly faster than the first phase method used herein. It would appear that a good semi-infinite programming algorithm could be constructed using a first phase as described by, say, Hettich, followed by the algorithm presented herein as a second phase.

Chapter 6

DISCUSSION AND CONCLUSION.

In this chapter a discussion of the various aspects of the quasi-Newton algorithm and the multi-local optimisation subalgorithm is undertaken. The comments are grouped loosely into several sections. Finally, a brief summary of what has been achieved is given.

The Role of ϕ.

In SQP methods there are two main ways of using ϕ: either as a penalty function or as a merit function. The role of ϕ determines, to a great extent, the form of the QP which is solved to obtain the search direction. With the standard SQP method, each search direction is generated by approximating the original SIP problem; ϕ merely serves as a merit function. That is to say, in the line search ϕ is used to adjudicate between the two aims of reducing the objective function, and attaining and maintaining feasibility. After calculating the search direction, the penalty parameters are chosen so that the search direction is one of descent for ϕ. In contrast, by generating each search direction using an L_∞QP which approximates ϕ, it is the problem of minimising ϕ that is solved rather than the SIP. The penalty parameters are selected first, and are chosen so that the feasible minimiser(s) of ϕ which are found are also solutions to the SIP.

The generation of each search direction using an L_∞QP has a significant advantage over the use of a QP. At each iteration it is guaranteed that the L_∞QP has a unique global minimiser, even when the linearizations of the constraints are inconsistent. When these linearizations are inconsistent, the resultant QP subproblem for the standard SQP method is infeasible. In such cases, one could minimise the infeasibility of the linearizations of the constraint, and then, subject to keeping this infeasibility to a minimum, minimise the objective function of the original QP. If the infeasibility of the linearizations of the constraint is taken as the maximum violation of these linearizations, then this is equivalent to solving the L_∞QP with μ large enough to force the maximum of the violations of the linearized constraints to its minimum value.

In essence, if ϕ is used as a penalty function, the penalty parameters are selected first, and then a suitable search direction is chosen. In contrast, if ϕ is used as a merit function, the search direction is chosen first, and then suitable values for the penalty parameters are selected so that the chosen search direction is one of descent for the merit function.

Choosing the Search Direction.

At each iteration the search direction is chosen by solving the appropriate quadratic programme. Because ϕ serves as a penalty function rather than as a merit function, an L_∞QP is solved at each iteration. At each iteration the L_∞QP must approximate the exact penalty function in the neighbourhood of the current iterate, but this does not completely determine the L_∞QP. Specifically, there is still some choice in the matrix H, and the set $\mathcal{A}$. Here the L_∞ exact penalty function has a distinct advantage over the L_1 exact penalty function in that the active set is only required to contain a sufficient number of points to satisfy assumption 2.8. Any finite set of points satisfying assumption 2.8 may be augmented by any other set of finite points in T, and still be acceptable as an active set $\mathcal{A}$ under assumption 2.8. This means an L_∞ exact penalty function based method is much less likely to be adversely affected than an L_1 based method if the multi-local optimisation subroutine returns two or more approximations to the same local maximiser.

This property of the L_∞ exact penalty function also permits extra points in T to be included in the active set $\mathcal{A}$. These extra points could include those at which prominent local maximisers are expected to appear, or points regarded as significant for some other reason. An example of this is the matching process of Tanaka et al., which is discussed later in this chapter.

Some other ways of choosing such extra points are as follows: For instance, such points could be prominent local maximisers of g at x values, which were proposed and rejected as iterates during the previous line search. Also, by extrapolating the change in position for each known local maximiser for the last two iterates (or proposed iterates), estimated positions of these local maximisers at the next iterate may be calculated. Finally, elements of $\mathcal{H}_m$ at which the increase in the constraint function value for the last two iterates (or proposed iterates) indicates the possible appearance of a prominent local maximiser could also be included in $\mathcal{A}$.

Inclusion of such points in $\mathcal{A}$ courts the risk of numerical instability in the QP arising from two or more very similar QP constraints. However, any extra points which give rise to near identical constraints in the QP are contributing little or no useful information, and should not be included in $\mathcal{A}$. Accordingly, provided any extra points included in $\mathcal{A}$ are not too close to points already in $\mathcal{A}$ (or each other!), including them in $\mathcal{A}$ appears to convey little risk, and may lead to better search directions. Increasing the number of points in $\mathcal{A}$ will usually lead to increased solution times for the QP subproblems, however any such increases are likely to be insignificant in comparison with the solution times for the multi-local optimisation subproblem. Moreover, if the number of multi-local optimisations performed is reduced, then the inclusion of such points in $\mathcal{A}$ is almost certainly advantageous.

The second order correction was included to ensure superlinear convergence can ultimately be attained on problems which are sufficiently continuous. The second order correction also proved to be useful at iterates far from the solution. Its inclusion allowed the curvature of the constraints along the search direction to be taken into account, in the process replacing the line search with an arc search. This permitted curved constraints to be followed more easily.

Lagrange Multiplier Estimates.

There are several methods of generating estimates of the optimal Lagrange multipliers. They can either be obtained as a by-product of solving the QP subproblem for the search direction, or calculated directly. For the implementation of the algorithm presented herein, the Lagrange multiplier estimates were taken directly from the L_∞QP's solution. Obtaining Lagrange multiplier estimates in this fashion requires no extra computational effort, but is not as robust as some methods. Gill and

Murray [34] describe a method in which both first and second order estimates are calculated: if these differ greatly, then the first order estimates are used, otherwise the second order estimates are used. The second order estimates could either be calculated by solving a QP problem expressly for that purpose, or simply taken as the Lagrange multipliers generated when solving the L_∞QP for the search direction. In the latter case, if the trust region is active at the solution of the L_∞QP, then these estimates should be rejected, and the first order estimates used. Such an approach is more intensive computationally, but the high cost of the multi-local optimisations means that this is likely to be of little significance.

When the Lagrange multiplier estimates are taken from the same L_∞QP used to generate the search directions, these estimates will be affected by the use of a trust region if that trust region is active at the solution of the L_∞QP. This can change the updates to H, leading to a loss of superlinear convergence. There are two basic cases of note: First, if $\zeta = 0$ at the solution of the L_∞QP (4.1,4.2,4.3,4.4,4.5), then any active trust region bounds will alter the Lagrange multipliers corresponding to the semi-infinite constraint, with consequences as indicated in earlier this paragraph. Second, if $\zeta > 0$ at the solution (s, ζ) of the L_∞QP, then the following equations hold:

$$\nabla f + Hs + \sum_{i \in \mathcal{A}} \lambda_i \nabla_x g(x, t_i) + \sum_{i=1}^{n} \eta_i e_i = 0,$$

$$\text{and} \quad \mu + \nu\zeta = \sum_{i \in \mathcal{A}} \lambda_i. \tag{6.1}$$

Here e_i is the i^{th} unit vector, and η_i is the Lagrange multiplier associated with the bounds on the i^{th} element of x. Specifically, if x_i lies on its lower bound, (respectively lies on its upper bound, or is unbounded) then η_i non-positive (respectively non-negative, or zero). Equation (6.1) shows that using the Lagrange multiplier estimates from the L_∞QP means that the penalty parameters will *always* be increased if either the trust region is too small to allow $\zeta = 0$ at the L_∞QP's solution, or if the linear constraints approximating the semi-infinite constraint are inconsistent. This will happen even if the penalty parameters are already high enough to force convergence to a solution, without further alteration. Moreover, as the sum of the Lagrange multipliers is equal to $\mu + \nu\zeta$, the Lagrange multipliers themselves will be wrong, leading to an incorrect update to the approximate Hessian H.

Updating the Penalty Parameters.

The scheme by which the penalty parameters are updated is intimately linked with the method of calculating the Lagrange multiplier estimates. The use of Lagrange multiplier estimates from the L_∞QP places an upper bound on the 1-norm of the Lagrange multipliers, and hence limits the rate of increase of the penalty parameters. Use of first order estimates, or of second order estimates generated from, say, a standard QP allows unlimited increases in μ and ν. Using such estimates, any limits on the increases of the penalty parameters would be entirely arbitrary. The numerical results show that, in such cases, allowing decreases in μ and ν is definitely advantageous.

The use of the hybrid penalty function gives an extra degree of freedom in choosing the penalty parameters. The two penalty parameters penalize infeasibility differently: when the infeasibility θ is small, μ is the dominant penalty parameter, whereas when θ is large ν is the more significant of the two.

The results for problem K, and the work of Coope [20] show that excessively high values of the penalty parameters reduce the rate of convergence. Problem K's results also show that an excessive value of ν is much less damaging. The presence of both penalty parameters means that an excessive value of μ may be corrected by decreasing μ and making a corresponding increase in ν. Moreover, at highly infeasible iterates increasing ν is more effective than increasing μ: increasing only ν at such iterates reduces the risk of generating an excessive value for μ.

Trust Regions and Line Searches.

The quasi-Newton algorithm uses a line search to ensure sufficient descent is obtained. A trust region is also present, although only in the rudimentary form of a bound on the infinity norm of the proposed step. Indeed, the convergence theorem requires that an upper bound of some description be imposed on the length of the proposed step.

Quite apart from the convergence theorem's requirements, using a trust region can be very beneficial. For example, on problem 6, it prevented the algorithm from considering prospective iterates at which the constraint's function values were sufficiently large to cause overflow errors.

Using the somewhat more sophisticated implementation of a trust region (4.6) gave considerable improvements in the quasi-Newton algorithm's performance on the extended version of problem 4 for $n = 6$, and $n = 8$. This suggests that a sophisticated implementation of a trust region could be profitably employed in an algorithm for SIP. Indeed the results of the three Newton type algorithms show that the algorithm of Tanaka et al., which employs a trust region, is definitely competitive with those of Watson, and of Coope and Watson. These algorithms also differ in other respects such as the choice of penalty function, and so they do not yield a direct comparison of the relative merits of a line search and a trust region.

The general features of the differences between line searches and trust regions can be identified. For finite NLPs, one significant difference is that the trust region approach requires a QP to be solved to generate each prospective next iterate, whereas the line search approach only requires one QP to be solved each iteration. In semi-infinite programming, the cost of solving these QPs is unlikely to be significant in comparison to the cost of performing an equal number of multi-local optimisations.

When using a trust region, the approximation to the Hessian of the Lagrangian can be updated each time a proposed iterate is generated, irrespective of whether or not the proposed iterate is accepted. In contrast, with line search algorithms, if the proposed iterate is rejected no updating occurs. In semi-infinite programming, evaluating the penalty function at each proposed iterate is very expensive, and so it is prudent to extract as much useful information as possible from these evaluations. On this point, trust region algorithms have a definite edge. Of course, at QP solutions where the trust region is active, for the purpose of choosing the prospective step the trust region will have interfered with the second order information stored in the approximation of the Hessian.

A trust region based algorithm does not permit a proposed step with a length greatly in excess of the length of the step accepted in the previous iteration. Most line search algorithms impose no such restriction. When large increases in the step length are desired, a trust region approach will hamper this — perhaps requiring more multi-local optimisations in the process. Counterbalancing this is the possibility that the proposed step is far too long. In such cases both types of algorithms will reject several proposed iterates before an acceptable point is found. If this new iterate is little different from the previous one, then a line search algorithm may generate a very similar search direction, and reject a similar number of prospective iterates before a suitable point is found. This sort of behaviour may continue for several iterations. In contrast, after the first such iteration, a trust region algorithm will choose prospective steps of a more suitable length, thus reducing the number of multi-local optimisations performed at each iteration.

In short, the situation for semi-infinite programming is similar to that for finite NLPs when the constraint functions are extremely expensive to evaluate. Both line search and trust region based methods are capable of yielding excellent algorithms for semi-infinite programming.

In the implementation of the arc search, the trial values of α were chosen in a relatively simplistic manner: namely $1, \beta, \beta^2, \ldots$ were tried in that order. For each trial value of α used a multi-local optimisation must be performed. If the number of trial values of α which are rejected is minimised, then, on difficult problems a significant increase in speed would be expected. Accordingly, any scheme for selecting trial α values which reduces the average number of trial α values rejected at every iteration will be very beneficial. One obvious strategy is to construct an approximation to $\phi(x + \alpha s)$ for α values ranging over the interval $[0, \alpha_{\text{previous}}]$ using function and gradient information, where α_{previous} is the most recently rejected α value. The minimiser of this approximation to $\phi(x + \alpha s)$ could then be the next trial α value, subject to it lying within some subinterval of $[0, \alpha_{\text{previous}}]$. Such a line search would be of most use on difficult problems where short steps occur frequently. This avenue has not been explored at all in this thesis.

Tanaka et al.'s Matching Process.

One feature of Tanaka et al.'s algorithm that distinguishes it from the algorithms of Watson, of Coope and Watson, and the algorithm presented herein is that it matches up the prominent local maximisers over successive iterations. If new prominent local maximisers are observed at a proposed iterate, then this prospective new iterate is rejected, estimates of the positions of these new prominent local maximisers at the old iterate are calculated, and a new search direction from the old iterate is chosen with these new prominent local maximisers taken into account. Such a process involves adding extra points to $\mathcal{A}^{(k)}$, and this can only be done if the L_∞ exact penalty function is used. This process is extremely effective on problem 9: the algorithm of Tanaka et al. is the only one to achieve the required accuracy on this problem. Of course problem 9 does not satisfy assumption (2.3), which is used to ensure that the semi-infinite programme in question is tractable.

The method Tanaka et al. use to perform the matching process requires the use of second order derivatives of g — specifically $\nabla^2_{tt}g$, and $\nabla^2_{xt}g$. The quasi-Newton algorithm is designed to work on C^1 functions and so using these second derivatives is not an option.

When close to a solution, a rather simpler matching process could be implemented: this matching process would only need to use the distances between the local maximisers to pair them up. For example, such a matching process could be used whenever the step lengths are sufficiently short.

The method used by the algorithm of Tanaka et al. to estimate the positions of the new global maximisers at the old iterate also uses second derivatives of g. In the absence of these quantities, the estimated positions of the new global maximisers at the old iterate could be taken as the positions of these maximisers at the newly proposed (and rejected) iterate.

The convergence theorem shows that the matching of global maximisers over successive iterations is necessary only for problems which the quasi-Newton algorithm, and indeed almost all other algorithms including those of Tanaka et al., of Watson, and of Coope and Watson, are not designed to solve. Nevertheless, matching can be a useful tactic when little progress is being made due to unforseen local maximisers. In such circumstances, even rudimentary matching and estimating processes will help predict the effects of these nascent local maximisers. In particular, the more progress is inhibited by such nascent local maximisers, the smaller the changes in the constraint function become. Hence, when use of a matching process is likely to be most advantageous, it will be easiest to perform, and yield the most accurate estimates of any nascent local maximisers.

The Multi-Local Optimisation Subproblem.

There are a number of techniques for finding a local optimum of a continuously differentiable function, subject to a finite number of continuously differentiable constraints. The variety of problems in this class is sufficient that no one technique is superior to the others in all cases. The problem of finding the global and prominent local maximisers of a differentiable function subject only to simple bounds is much more difficult than that of simply finding a local maximiser, especially when the number of dimensions is greater than one. It would be unwise to expect that there is *one* basic approach to the multi-local optimisation problem that is the best in all cases. The multi-local optimisation algorithm uses uses a Halton sequence to generate test points, rather than a rectangular grid. The sort of function for which the multi-local optimisation algorithm is intended is typically in two to four dimensions, is multi-modal, but does not possess a large number of prominent local maximisers, and may be either unimodal or nearly constant in some (but not all) dimensions.

One of the reasons for choosing a Halton sequence to explore the topography of g was that the 'projection deficiency' of a rectangular grid was avoided. This deficiency occurs when g is nearly constant with respect to t along some directions, and in particular those parallel to some axis of the grid. In such cases, a drastic loss of efficiency can occur. However, the method of estimating the variance parameter c in the implementation used to generate the results implicitly assumes that g is isotropic. This assumption is in no way essential to the multi-local optimisation algorithm. Perhaps the easiest way of eliminating it is to regard g as being the result of a change of variables on a function which is approximately isotropic. One then has to estimate the metric under which g is an approximately isotropic function as well as c itself.

The assumption that c was approximately independent of direction was made for the implementation of the algorithm for the sake of simplicity. Other 'simplifications' were made in the implementation of the algorithm. For instance, at each iteration the points in the Halton sequence were calculated anew. A more efficient approach would be to store them between iterations. Excluding the isotropy assumption for c, the simplifications made in implementing the multi-local optimisation algorithm did not change the actual algorithm in any way whatsoever. However, these simplifications will have resulted in somewhat increased solution times.

Halton sequences are really only of use up to about 4 to 6 dimensions at most — above that a very large number of test points must be generated before the uniformity theorem becomes valid, and calculating the nearest neighbours of a test point becomes very expensive. In contrast, rectangular grids are subject to similar limitations for different reasons. The nearest neighbour subproblem for a grid is easy to solve. However, the number of nearest neighbours of a test point may be exponential in p, depending on the precise definition of the term 'nearest neighbour.'

Also, if all dimensions are to be treated equally, the number of test points used by a grid is inherently exponential in p.

In most cases, it is reasonable to expect that, for a SIP in n 'x' dimensions and p 't' dimensions, the number of global maximisers of the constraint function at the solution will be approximately n, at most. Rinnooy Kan et al. [89] show that, under certain assumptions, the number of function evaluations required to find the global maximisers of the constraint function is independent of p. In spite of this, the algorithm of Rinnooy Kan et al. is still exponential in p — this is a consequence of using the nearest neighbour algorithm of Bentley et al. [12]. This exponential behaviour can easily be avoided if one is prepared to forfeit the property that the computational costs are proportional to the number of function evaluations. This poses an interesting question: 'given that the expected number of global maximisers is independent of p, is there a viable method of exploring T for which the costs are proportional to the number of function evaluations, yet are not inherently exponential in p?'

Two simple modifications to the multi-local optimisation algorithm which would lead to improvements in performance are described in the following two paragraphs. Neither of these modifications was implemented.

A simple method of reducing the cost of performing the multi-local optimisations is to calculate the largest value of θ for which the sufficient descent conditions are satisfied. Call this value θ_0 say. During each multi-local optimisation, as the constraint function value is calculated at each test point, it is checked against θ_0. If it exceeds θ_0, then the multi-local optimisation is halted immediately, and the proposed iterate is rejected.

Halton sequences are not invariant with respect to the ordering of the co-ordinate vectors. Changing the order of the axes is equivalent to changing the order of the π_i's and δ_i's. Because $\pi_i \neq \pi_j$ unless $i = j$, any two different orderings of the π_i's and δ_i's will yield two different sequences of test points. This fact can be useful in the final iterations when the constraint function remains approximately constant from one iteration to the next. By using a different ordering of the π_i's and δ_i's at each such iteration, a more thorough exploration of the semi-infinite constraint at the final iterate can be made.

Concluding Remarks.

A quasi-Newton algorithm for semi-infinite programming problems has been constructed. The conditions under which the algorithm has been shown to converge correspond closely to those required for a quasi-Newton algorithm for finite nonlinear programmes. The extra conditions are assumptions 2.3 and 2.8, which respectively make the problem tractable, and ensure the multi-local optimisation subalgorithm fulfills its intended purpose. These two extra assumptions are quite mild. These assumptions are considerably weaker than those required for the algorithms of Watson [97], Coope and Watson [21], and Tanaka, Fukushima, and Ibaraki [94]. In particular, it is not necessary that the implicit function theorem be applicable to the local maximisers of the constraint function. The assumptions required by the quasi-Newton algorithm to ensure convergence are weaker, and so the class of problems which can be solved by this algorithm is larger. Neither assumption 2.3, nor assumption 2.8 requires C^2 continuity of either the objective or constraint function, and so the quasi-Newton algorithm is capable of solving C^1 problems. On problems satisfying conditions similar to those required by the three Newton type algorithms, the quasi-Newton algorithm can be shown to exhibit superlinear convergence.

The use of an L_∞ exact penalty function removes the possibility that discontinuities may exist in the penalty function at infeasible points: a flaw from which the L_1 exact penalty function suffers. The use of L_∞QP subproblems to generate search directions ensures that each QP subproblem can always be solved. Together these two facts yield a convergence result similar to that for an unconstrained optimisation problem. Specifically, at least one of the following occurs:

- Convergence to a stationary point of the penalty function occurs.
- The sequences of objective, and penalty function values are unbounded below.
- The sequence of iterates diverges.
- The penalty parameters are increased endlessly.

Of course convergence to a stationary point of the penalty function does not guarantee that the relevant limit point is feasible. Nor, given feasibility, does it guarantee that that point is a local minimum of the SIP, although it must be a stationary point of the SIP. However, this situation is identical to that for quasi-Newton algorithms for finite NLPs. As the latter type of problem is a subclass of semi-infinite programmes, the best that can possibly be achieved for SIPs is equivalence with the situation for NLPs.

For simplicity, this thesis has concerned itself only with semi-infinite programmes with one semi-infinite constraint and no auxillary constraints. The extension to several semi-infinite constraints and auxillary constraints is straightforward. In light of the current preference for the L_1 exact penalty function for finite NLPs, the preferred measure of infeasibility would be the sum of the maximum constraint violations of the semi-infinite constraints, plus the sum of the violations of the auxillary constraints. This would yield a hybrid $L_1 L_\infty$ exact penalty function.

The quasi-Newton algorithm performed well on a wide variety of problems. In general it took more iterations and more multi-local optimisations to solve a problem than any of the three Newton type algorithms did. On the more ill-conditioned problems, the gap between the Newton type and the quasi-Newton algorithms was widest. This is similar to the situation for finite NLPs, and is to be expected. In fact, one would expect the difference between the two types of algorithms to be greater on semi-infinite programmes than on finite non-linear programmes. The reason for this being that the approximations to the local maximisers found by the quasi-Newton algorithm will, in general, be less accurate than those found using a Newton type algorithm; this extra source of error is not present in NLPs.

Although the quasi-Newton algorithm requires more iterations and multi-local optimisations than the Newton type algorithms, this does not automatically imply that the quasi-Newton algorithm is slower. At each iteration the Newton type algorithms must calculate the Hessian of the Lagrangian, which can require a large number of evaluations of the objective and constraint functions' derivatives. The constraint terms in this Hessian are of the form given in equation (1.14), which are messier than those for finite constraints. If the Lagrangian's Hessian is expensive to calculate, then the quasi-Newton algorithm may be faster. Moreover, Newton type algorithms must be provided with not only the first, but also the second derivatives. Unless these can be provided by an automatic differentiation algorithm, this in itself is a major disadvantage.

The penalty function ϕ is a hybrid of an L_∞ based quadratic penalty function, and the L_∞ exact penalty function. The extra penalty parameter allows for considerably more flexibility in updating the penalty parameters, increasing the chance that μ could be kept to a reasonable value. Numerical results indicate that an excessive value for either penalty parameter reduces the rate of convergence. They also show that an excessive μ value is less desirable than an excessive value for ν.

The use of a NLP first phase was shown to be advantageous. The numerical results also showed that there was little point in solving the NLP to a high accuracy. The NLP first phase implementation was simplistic: in view of the work of Hettich and others, it is clear that considerable improvements can be made over what has been done here, and that such improvements are likely to lead to significantly shorter solution times.

The results for the problems with constraint index sets of dimension greater than two shows that although the number of iterations and multi-local optimisations required to solve a problem may not increase with increasing p, the solution times increase rapidly with increasing p. On these problems the great majority of the work is in the multi-local optimisations. To obtain the best possible performance on these problems, the number of multi-local optimisations should be minimised, and they should be done as efficiently as possible. The multi-local optimisation algorithm described herein was designed expressly to avoid the enormous number of test points required by a rectangular grid based algorithm when p exceeds two. As the results for the higher dimensional problems (and especially problem U) show, this approach can lead to significantly faster solution times. Nevertheless, much work remains to be done in the area of multi-local optimisation sub-algorithms for SIPs with constraint index sets T of dimension greater than one.

The results for the quasi-Newton algorithm on Watson's series of problems were generated using the multi-local optimisation algorithm described herein. The quasi-Newton algorithm would still be effective on these problems when used in conjunction with any of the multi-local optimisation routines used in the three Newton type algorithms, as well as many other multi-local optimisation algorithms.

The theoretical and numerical results show that a quasi-Newton algorithm for semi-infinite programming is definitely viable.

Bibliography

[1] Aird, T. J. and J. R. Rice, *Systematic Search in High Dimensional Sets*, SIAM Jnl. Numer. Anal., vol.14, no. 2, pp 296–312 (1977).

[2] Anderson, E. J., and A. B. Philpott, *Infinite Dimensional Linear Programming*, proceedings of an international symposium, Cambridge, 1984, Lecture notes in Economics and Mathematical Systems no. 259, Springer, Berlin ©1985.

[3] Anderssen, R. S., *Global Optimisation*, pp 26–48, in R. S. Anderssen, L. S. Jennings, and D. M. Ryan, *Optimization*, University of Queensland press, St. Lucia, Queensland ©1972.

[4] Archetti, F., *A Probabilistic Method for Global Optimisation Problems with a Dimensionality Reduction Technique*, pp 36–42, in K. Iracki, K. Malanowski, and S. Walukiewicz (Eds.), *Optimisation Techniques (part 2)*, Lecture notes in Control and Information Sciences no. 23, Springer-Verlag, Berlin Heidelberg New York ©1980.

[5] Archetti, F. and M. Cugliani (Eds.), *Numerical Techniques for Stochastic Systems*, North-Holland ©1980.

[6] Ašić. M. D. and V. V. Kovačević-Vujčić., *An Interior Semi-Infinite Programming Method*, Jnl. Opt. Th. and Appl., vol. 59, no. 3, pp 353–367 (1988).

[7] Bachem, A., M. Grötschel, and B. Korte (Eds.), *Mathematical Programming The State of the Art*, Springer-Verlag, Berlin Heidelberg New York Tokyo ©1983.

[8] Bandler, J. W. and C. Charalambous, *Nonlinear Programming Using Minimax Techniques*, J. of Opt. Th. and Appl., vol. 13, pp 607–618 (1974).

[9] Bandler, J. W. and M. R. M. Rizk, *Optimization of Electrical Circuits*, Math. Prog. Study no. 11, pp 1–64 (1979).

[10] Beekman, J. A., *Two Stochastic Processes*, Almqvist & Wiskell international, Stockholm ©1974.

[11] Bell, B. M., *Global Convergence of a Semi-Infinite Optimisation Method*, Appl. Math. Opt., vol. 21, no. 1, pp 89-110 (1990).

[12] Bentley, J. L., B. W. Weide, and A. C. Yao, *Optimal Expected Time Algorithms for Closest Point Problems*, ACM Transactions on Mathematical Software, vol. 6, no. 4, pp 563–580 (1980).

[13] Boggs, P. T., R. H. Byrd, and R. B. Schnabel (Eds.), *Numerical Optimisation 1984*, SIAM Conference on Numerical Optimisation 1984, Boulder, Colorado, SIAM Philadelphia ©1985.

[14] Brodlie, K. W., *Unconstrained Minimisation*, in [57], pp 229–268.

[15] Charalambous, C., *A Lower Bound for the Controlling Parameters of the Exact Penalty Functions*, Math. Prog., vol. 15, pp 278–290 (1978).

[16] Coleman, T. F. and A. R. Conn, *Nonlinear Programming via an Exact Penalty Function: Asymptotic Analysis*, Math. Prog., vol. 24, pp 123–136 (1982).

[17] Coleman, T. F. and A. R. Conn, *Nonlinear Programming via an Exact Penalty Function: Global Analysis*, Math Prog vol. 24, pp 137–161 (1982).

[18] Conn, A. R., *Nonlinear Programming, Exact Penalty Functions, and Projection Techniques for Non-Smooth Functions*, in [13], pp 3–25.

[19] Conn, A. R. and N. I. M. Gould, *An Exact Penalty Function for Semi-Infinite Programming*, Math. Prog., vol. 37, pp 19–40 (1987).

[20] Coope, I. D., *The Maratos effect in Sequential Quadratic Programming Algorithms using the L_1 Exact Penalty Function*, Technical Report CS-85-32, Computer Science Department, University of Waterloo, August 1985.

[21] Coope, I. D. and G. A. Watson, *A Projected Lagrangian Algorithm for Semi-Infinite Programming*, Math. Prog., vol. 32, pp 337–356 (1985).

[22] Dixon, L. C. W., *Quasi-Newton Techniques Generate Identical Points II: the Proofs of Four New Theorems*, Math. Prog., vol. 3, pp 345–358 (1972).

[23] Evtushenko, Y. J., *Numerical Optimisation Techniques*, Optimization Software inc. Publications Division ©1985.

[24] Fiacco, A. V., and K. O. Kortanek (eds.), *Semi-Infinite Programming and Applications*, Lecture notes in Economics and Mathematical Systems no. 215, Springer Verlag, Berlin (1983).

[25] Fletcher, R., *A Model Algorithm for Composite Non-Differentiable Optimisation Problems*, Math. Prog. Study no. 17, pp 67–76 (1982).

[26] Fletcher, R., *An ℓ_1 Penalty Method for Nonlinear Constraints*, in [13], pp 27–40.

[27] Fletcher, R., *A First Derivative Method for Non-Linear Programming $\ell_1 LP$*, pp 43–56, in [73].

[28] Fletcher, R., *Second Order Corrections for Non-Differentiable Optimisation*, pp 85–114, in G. A. Watson (Ed.), *Numerical Analysis*, Lecture notes in Mathematics no. 912, Springer-Verlag (1982).

[29] Fletcher, R., *Practical Methods of Optimisation volume 2: Constrained Optimisation*, John Wiley & sons, Chichester ©1981.

[30] Fletcher, R., *Methods for Solving Non-Linearly Constrained Optimisation Problems*, in [57], pp 365–408.

[31] Fletcher, R., *Recent developments in linear and quadratic programming*, Report NA/94, July 1986, Department of Mathematical Sciences, University of Dundee, Scotland.

[32] Ge, R., *A Filled Function Method for Finding a Global Minimiser of a Function of Several Variables*, Math. Prog., vol. 46, no.2, pp 191–204 (1990).

[33] Gfrerer, H., J. Guddat, Hj. Wacker, and W. Zulehner, *Globalisation of Locally Convergent Algorithms for Non-Linear Optimisation Problems with Constraints*, in [24], pp 128–137 (1983).

[34] Gill, P. E. and W. Murray, *The Computation of Lagrange Multiplier Estimates for Constrained Optimization*, Math. Prog., vol. 17, pp 32–60 (1979).

[35] Gill, P. E. and W. Murray, *Modification of Matrix Factorizations after a Rank One Change*, in [57], pp 55–84.

[36] Gill, P. E., W. Murray, and M. H. Wright, *Practical Optimization*, Academic press ©1981.

[37] Glashoff, and S.-A. Gustafson, *Linear Optimization and Approximation*, Applied Mathematical Sciences series no. 45, Springer-Verlag, New York Heidelberg Berlin ©1983.

[38] Goldfarb, D. and A. Idnani, *A Numerically Stable Dual Method for Solving Strictly Convex Quadratic Programmes*, Math. Prog., vol. 27, pp 1–33 (1983).

[39] Gonzaga, C., E. Polak, and R. Trahan, *An Improved Algorithm for Optimisation Problems with Functional Inequality Constraints*, IEEE trans. on Automatic Control, vol. AC-25, no. 1, pp 49–54 (1980).

[40] Gowers, Sir E., *ABC of Plain Words*, Published under the authority of His Majesty's Stationery Office by Fosh & Cross Ltd., London ©1951.

[41] Gustafson, S.-A., *A Three Phase Algorithm for Semi-Infinite Programmes*, pp 138–157, in [24].

[42] Gustafson, S.-A. and K. O. Kortanek, *Semi-Infinite Programming and Applications*, in [7], pp 132–157.

[43] Gustafson, S.-A. and K. O. Kortanek, *Numerical Treatment of a class of Semi-Infinite Programming Problems*, Nav. Res. Log. Quart. 20, pp 477-504, (1973).

[44] Halmos, P. R., *Measure Theory*, van Nostrand, New York ©1950.

[45] Halton, J. H., *On the Efficiency of Certain Quasi-Random Sequences of Points in Evaluating Multi-Dimensional Integrals*, Numerische Mathematik, vol. 2, pp 84–90 (1960).

[46] Han, S. P., *A Globally Convergent Method for Nonlinear Programming*, Jnl. Opt. Theory Appl., vol. 22, no. 3, pp 297–309 (1977).

[47] Han, S. P., *Superlinearly Convergent Variable Metric Algorithms for General Non-Linear Programming Problems*, Math. Prog., vol. 11, pp 263–281 (1976).

[48] Han S. P. and O. L. Mangasarian, *Exact Penalty Functions in Non-Linear Programming*, Math. Prog., vol. 17, pp 251–269 (1979).

[49] Hettich, R. (Ed.), *Semi-Infinite Programming*, Lecture Notes in Control and Information Sciences no. 15, Springer-Verlag, Berlin Heidelberg New York ©1979.

[50] Hettich, R., *An Implementation of a Discretization Method for Semi-Infinite Programming*, Math. Prog., vol. 34, pp 354–361 (1986).

[51] Hettich, R., *A Review of Numerical Methods for Semi-Infinite Optimisation*, in [24], pp 158–178.

[52] Hettich, R., *A Comparison of some Numerical Methods for Semi-Infinite Programming*, in [49], pp 112–125.

[53] Hettich, R., and G. Gramlich, *A Note on an Implementation of a Method for Semi-Infinite Quadratic Programming*, Math. Prog., vol. 46, no. 2, pp 249–254 (1990).

[54] Hettich, R. and W. van Honstede, *On Quadratically Convergent Methods for Semi-Infinite Programming*, in [49], pp 97–111.

[55] Honstede, W. van, *An Approximation method for Semi-Infinite Problems*, in [49], pp 126–136.

[56] Hu, H., *A One Phase Algorithm for Semi-Infinite Linear Programming*, Math. Prog., vol. 46, pp 85–103 (1990).

[57] Jacobs, D. A. H. (Ed.), *State of the Art in Numerical Analysis 1977*, Academic Press ©1977.

[58] Kiwiel, K. C., *Methods of Descent for Non-Differentiable Optimisation*, Lecture notes in Mathematics no. 1133, Springer-Verlag ©1985.

[59] Kortanek, K. O., and Hoon No, *A Central Cutting Plane Algorithm for Convex Semi-Infinite Programming Problems*, working paper series no. 90–08, revised February 1992.

[60] Krabs, W., *Optimisation and Approximation*, John Wiley & sons, ©1979.

[61] Luenberger, D. G., *Optimisation by Vector Space Methods*, John Wiley & sons, ©1969.

[62] Lundy, M. and A. Mees., *Convergence of an Annealing Algorithm*, Math. Prog., vol. 34, pp 111–124 (1986).

[63] Mayne, D. Q. and E. Polak, *A Superlinearly Convergent Algorithm for Constrained Optimization Problems*, Math. Prog. Study 16, pp 45–61 (1982).

[64] Meewella, C. C. and D. Q. Mayne, *An Algorithm for Global Optimisation of Lipschitz Continuous Functions*, Jnl. Opt. Theory Appl., vol. 57, no. 2, pp 307–322 (1988).

[65] Mine, H., M. Fukushima, and Y. Tanaka, *On the use of ϵ-most Active Constraints in an Exact Penalty Function Method for Non-Linear Optimisation*, IEEE trans. Automatic Control vol. AC-29, no. 11, pp 1040–1042 (1984).

[66] Mladineo, R. H., *An Algorithm for Finding the Global Maximum of a Multimodal, Multivariate Function*, Math. Prog., vol. 34, pp 188–200, (1986).

[67] Mockus, J., *Bayesian Approach to Global Optimisation — Theory and Applications*, Kluwer Academic Publishers ©1989.

[68] Mockus, J., *The Simple Bayesian Algorithm for Multi-Dimensional Global Optimisation*, in [5], pp 369–377.

[69] Moore, R. E. and H. Ratschek, *Inclusion Functions and Global Optimisation II*, Math. Prog., vol. 41, pp 341–356 (1988).

[70] Niederreiter, H., *Quasi Monte Carlo Methods and Pseudo-Random Numbers*, Bulletin of the American Math. Soc., vol. 84, no. 6, pp 957–1041 (November 1978).

[71] Niederreiter, H. and P. Peart, *Localisation of Search in Quasi Monte Carlo Methods for Global Optimisation*, SIAM Jnl. Sci. Stat. Comp., vol. 7, no. 2, pp 660–664 (1986).

[72] Panier, E. R. and A. L. Tits, *A Globally Convergent Algorithm with Adaptively Refined Discretization for Semi-Infinite Optimisation Problems arising in Engineering Design*, IEEE trans. Automatic Control, vol. 34, no. 8, pp 903–908 (1989).

[73] Penot, J.-P. (Ed.), *New methods in Optimisation and their Industrial Uses*, International series on Numerical Mathematics, vol. 87, Birkhäuser-Verlag ©1989.

[74] Phillips, E. M. and D. S. Pugh, *How to get a PhD*, Open University Press ©1987.

[75] Pietrzykowski, T., *An Exact Potential Method for Constrained Maxima*, SIAM Jnl Numer. Anal., vol. 6, no. 2, pp 299–304 (1969).

[76] Pietrzykowski, T., *The Potential Method for Conditional Maxima in the Locally Compact Metric Spaces*, Numerische Mathematik, vol. 14, pp 325–329 (1970).

[77] Polak, E., *Semi-Infinite Optimisation in Engineering Design*, pp 236–248, in [24].

[78] Polak, E., and D. Q. Mayne, *An Algorithm for Optimisation Problems with Functional Inequality Constraints*, IEEE trans. on Automatic Control, vol. AC-21, no. 2, pp 184–193 (1976).

[79] Polak, E., and A. L. Tits, *A Recursive Quadratic Programming Algorithm for Semi-Infinite Optimisation Problems*, Appl. Math. Opt., vol. 8, pp 325–349 (1982).

[80] Powell, M. J. D., *The Convergence of Variable Metric Methods for Non-Linearly Constrained Optimization Calculations*, in *Nonlinear Programming 3*, eds. O. L. Mangasarian, R. R. Meyer, and S. M. Robinson, Academic Press, New York, (1978).

[81] Powell, M. J. D., *On the Quadratic Programming Algorithm of Goldfarb and Idnani*, Math. Prog. Study 25, pp 46–61 (1985).

[82] Powell, M. J. D., *Variable Metric Methods for Constrained Optimisation*, in [7], pp 288–311.

[83] Powell, M. J. D. (Ed.), *Nonlinear Optimization*, NATO conference Series, series II: Systems Science, Academic Press ©1982.

[84] Powell, M. J. D., *A Fast Algorithm for Non-Linearly Constrained Optimisation Calculations*, pp 144–157, in G. A. Watson (Ed.), *Numerical Analysis*, Dundee 1977, Lecture notes in Mathematics no. 630, Springer-Verlag ©1978.

[85] Price, C. J., and I. D. Coope, *An Exact Penalty Function Algorithm for Semi-Infinite Programmes*, BIT, vol. 30, pp 723–734 (1990).

[86] Price, C. J., and I. D. Coope, *The L_∞ Exact Penalty Function in Semi-Infinite Programming*, in Hogarth and Noye (Eds.), *Computational Techniques and Applications 1989*, Hemisphere ©1989.

[87] Ratschek, H., *Inclusion Functions and Global Optimisation*, Math. Prog., vol. 33, pp 300–317 (1985).

[88] Rinnooy Kan, A. H. G., C. G. E. Boender, and G. T. Timmer, *A Stochastic Approach to Global Optimisation*, pp 281–308, in K. Schittkowski (Ed.), *Computational Mathematical Programming*, NATO ASI series vol. F15, Springer-Verlag ©1985.

[89] Rinnooy Kan, A. H. G., and G. T. Timmer, *Stochastic Global Optimisation Methods part I: Clustering Methods*, Math. Prog., vol. 39, pp 27–56 (1987).

[90] Rinnooy Kan, A. H. G., and G. T. Timmer, *Stochastic Global Optimisation Methods part II: Multi-Level Methods*, Math. Prog., vol. 39, pp 57–78 (1987).

[91] Rinnooy Kan, A. H. G. and G. T. Timmer, *Global Optimisation: A Survey*, pp 133–155, in [73].

[92] Sobol, I. M., *On the Systematic Search in a Hypercube*, SIAM Jnl. Numer. Anal., vol. 16, no. 5, pp 790–793 (1979).

[93] Tanaka, Y., M. Fukushima, and T. Hasegawa, *Implementable L_∞ Penalty Function Method for Semi-Infinite Optimisation*, Int. J. Systems Sci., vol. 18, no. 8, pp 1563–1568 (1987).

[94] Tanaka, Y., M. Fukushima, and T. Ibaraki, *A Globally Convergent SQP Method for Semi-Infinite Non-Linear Optimization*, Journal of Computational and Applied Mathematics 23, 141–153 (1988).

[95] van Tiel, J., *Convex Analysis — An Introductory Text*, John Wiley & sons, ©1984.

[96] Törn, A., and A. Žilinskas, *Global Optimisation*, Lecture notes in Computer Science no. 350, Springer Verlag, Berlin-Heidelberg (1989).

[97] Watson, G. A., *Globally Convergent Methods for Semi-Infinite Programming*, BIT 21, 362–373 (1981).

[98] Watson, G. A., *Lagrangian Methods for Semi-Infinite Programming Problems*, in [2], pp 90–107.

[99] Watson, G. A., *Numerical Experiments with Globally Convergent Methods for Semi-Infinite Programming Problems*, in [24], pp 193–205.

[100] Wilson, R. B., *A Simplicial Algorithm for Concave Programming*, Ph.D. Dissertation, Graduate School of Business Administration, Harvard University (1963).

[101] Womersley, R. S., *Optimality Conditions for Piecewise Smooth Functions*, Math. Prog. Study 17, pp 13–27 (1982).

[102] Yeh, J., *Stochastic Processes and the Wiener Integral*, Marcel Dekker inc. (1973).

[103] Žilinskas, A., *Axiomatic Approach to Statistical Models and their use in Multi-Modal Optimisation Theory*, Math. Prog., vol. 22, pp 104–115 (1982).

Appendix: Other Results.

In this appendix a summary of the results for problems 7, 8, and 10–13 of the Watson series is given. The results presented in this appendix were generated using the quasi-Newton algorithm with $\theta_{\text{cap}} = 1$, and $\theta_{\text{crossover}} = 1$, rather than $\theta_{\text{cap}} = 0.01$ and $\theta_{\text{crossover}} = 0.1$, as was used for the results in chapter 5.

problem	n	p	$\lvert\Gamma^*\rvert$	k_P	h_P	Φ'_P
7	3	2	1	12	19	1.4E-6
8	6	2	4	48	77	7.8E-7
10	3	2	1	11	19	2.4E-10
11	3	2	2	25	66	1.2E-6
12	3	2	1	20	34	2.0E-7
13	3	2	1	25	49	6.6E-7