[1] *AMO - Advanced Modeling and Optimization, Volume 15, Number 2, 2013.*

Traversing non-convex regions

M.C. Bartholomew-Biggs, S.Beddiaf & S.J. Kane

School of Physics Astronomy & Mathematics
University Of Hertfordshire, Hatfield AL10 9AB, United Kingdom

Abstract

This paper considers a method for dealing with non-convex objective functions in optimization problems. It uses the Hessian matrix and combines features of trust-region techniques and continuous steepest descent trajectory-following in order to construct an algorithm which performs curvilinear searches away from the starting point of each iteration. A prototype implementation yields promising results.

1 Introduction

It is well-known that iterative methods for unconstrained minimization can experience difficulties if the search enters a region where the Hessian matrix of the objective function $F(x)$ is not positive-definite. Popular iterative methods based on minimizing a convex quadratic model function become inappropriate in this situation. In the Newton method, for instance, the search direction $p = -\nabla^2 F^{-1} \nabla F$ may lead towards a saddle point or a local maximum rather than a minimum. Quasi-Newton methods which use positive-definite estimates of $\nabla^2 F$ will be able to obtain descent directions; but the standard updates cannot revise the Hessian in non-convex regions and so progress made by successive iterations may be little better than that given by steepest descent.

One minimization technique whose use can be justified in non-convex regions is the trust-region approach. This uses a step which minimizes a local quadratic model subject to a restriction on the two-norm of the step size. This model problem always has a solution, whether or not the Hessian is positive-definite. Another theoretically justifiable way to make progress through a non-convex region is to follow the continuous steepest descent path (CSDP) obtained by solving the ODE

$$\frac{dx}{dt} = -\nabla F(x). \tag{1}$$

If we follow this path *accurately* from $x = x_1$ to $x = x_2$, where $||x_2 - x_1|| = \Delta$, then we obtain the maximum decrease in F for all steps of size Δ away from x_1. This indicates that there is relationship between the CSDP and trust region approaches. We shall discuss this further in the next section.

[1] **AMO - Advanced Modeling and Optimization. ISSN: 1841-4311**

In this paper we consider a minimization method in which each iteration is based on the solution of (1). Such algorithms have been proposed by many authors (see, for instance, [1], [2], [3], [4], [5], [6], [7], [8]) and our focus is on efficient ways of performing curvilinear searches along an approximate CSDP. The methods suggested by Behrman [1] and Brown [2] are probably the ones to which the ideas in this paper are closest. Our algorithmic suggestions are still of a preliminary nature; but initial numerical experience seems sufficiently encouraging to be worth reporting. We outline the main ideas in sections 2 and 3. We then show their outworking on some sample problems in section 4. Based on observed behaviour on these demonstration problems, we discuss a prototype algorithm in sections 5 and 6 and describe its performance in section 7. A concluding section indicates areas for further work.

2 Solving the continuous gradient equation

We use $g(x)$ and $G(x)$ to denote the gradient and Hessian of an objective function $F(x)$. If x_k is an estimate of a local minimum of F obtained after k iterations of a minimization algorithm then we write $F_k = F(x_k),\ g_k = g(x_k)$ and $G_k = G(x_k)$.

Now consider (1) with initial condition $x = x_k$ when $t = 0$. A step of the Implicit Euler method uses

$$x_{k+1} = x_k - \delta t\, g(x_{k+1}) \tag{2}$$

to give x_{k+1} as an estimate of the solution at $t = \delta t$. The right hand side of (2) can be approximated by

$$x_k - \delta t(g_k + G_k(x_{k+1} - x_k))$$

and if we write $p_k = x_{k+1} - x_k$ then we obtain a linearised form of (2), namely

$$(I + \delta t G_k)p_k = -\delta t\, g_k. \tag{3}$$

A well-known observation is that (3) gives a correction step similar to that produced by a trust-region method which solves

$$(\mu I + G_k)p_k = -g_k. \tag{4}$$

The parameter μ in (4) is effectively the reciprocal of the step length δt in (3). Thus (3) causes p_k to approach the Newton step as $\delta t \to \infty$ while (4) gives the Newton step when $\mu = 0$. Similarly (3) gives p_k parallel to $-g_k$ when $\delta t = 0$ while (4) makes p_k tend to a steepest descent step as $\mu \to \infty$.

An important point about (4) is that, even when G_k is not positive definite, it gives p_k as a descent direction, providing μ is chosen sufficiently large. More precisely, (4) yields p_k as the step which minimizes the quadratic model function

$$\frac{1}{2}p^T G_k p + p^T g_k + F_k \tag{5}$$

subject to a step-size constraint

$$||p||_2 \leq \Delta. \tag{6}$$

In (6), Δ is a function of μ. Trust region methods (see [9] for a detailed survey) work by choosing a step size Δ on each iteration and then finding μ so that the solution of (4) satisfies (6). (This is a non-trivial calculation since the relationship between Δ and μ is typically highly nonlinear.) The value of Δ for the next iteration is increased if the new point $x_{k+1} = x_k + p_k$ gives an acceptable decrease in the objective function. On the other hand, if $F(x_{k+1}) \geq F_k$, then Δ must be reduced.

The correspondence between (3) and (4) allows us to state that, whether G_k is positive definite or not, (3) will produce a step p_k such that $F(x_k + p_k) < F_k$ so long as δt is sufficiently small or μ is sufficiently large.

In the discussion which follows, we assume the exact Hessian G_k is available. (The algorithmic ideas we develop may also be applicable when (3) and (4) involve a quasi-Newton estimate of the Hessian which need not be forced to be positive definite.) We propose an approach in which each iteration traces out a *curved* path away from x_k by solving (4) for various values of μ. In the troublesome case when G_k is non positive-definite we must use positive values of μ such that $\mu > \mu_{min} > 0$, where $-\mu_{min}$ is the most negative eigenvalue of G_k. When G_k is positive definite, however, it is natural to use $\mu = 0$ and obtain p_k as the classical Newton step (giving quadratic convergence in the vicinity of a minimum). But, as we shall see later, it may be beneficial to use negative values of μ to extrapolate beyond a Newton step when x_k is not sufficiently near a minimum for quadratic convergence to occur.

3 Searching along a CSDP

We now consider how to choose value(s) of μ in (4) that will allow the search to follow an approximate CSDP away from x_k. Our aim is to make significant progress along this path in order to reach an acceptable new point x_{k+1} at which the Hessian G will be recomputed. We assume the eigenvalues of G_k are available. This enables us to detect immediately whether or not G_k is positive definite.

3.1 Approximating the CSDP when G_k is not positive definite

When G_k is not positive-definite the Newton step given by setting $\mu = 0$ in (4) is likely to lead towards a maximum or saddle point. In order for the coefficient matrix in (4) to be positive definite (and hence for the solution p_k to satisfy the descent property $p_k^T g_k < 0$) we need to use a positive value for μ. More precisely, if $-\mu_{min}$ is the most negative eigenvalue of G_k then we must choose

$$\mu > \mu_{min}. \tag{7}$$

To trace out an approximate CSDP from x_k we first select an initial value $\mu = \mu_1$ (as discussed later in section 5.1) and we then solve (4) for a sequence of values of $\mu_1, \mu_2, \ldots$ decreasing towards μ_{min}. We denote a typical trial point by

$$x_{k,j} = x_k + p_{k,j} \tag{8}$$

where $p_{k,j}$ is obtained by solving (4) with $\mu = \mu_j$. We can justifiably keep following the path defined by points $x_{k,j}$ so long as $F(x)$ is being sufficiently reduced and the $x_{k,j}$ stay acceptably close to the true CSDP.

If $F_{k,j}$ denotes $F(x_{k,j})$ we define

$$D_1 = \frac{F_{k,j} - F_k}{g_k^T p_{k,j}} \tag{9}$$

D_1 compares the actual change in F with a first-order prediction; in particular, if F is quadratic, then $D_1 = 0.5$ at the one-dimensional minimum along the search direction $p_{k,j}$. Hence, if D_1 is appreciably greater than 0.5 we may deduce that there is still useful progress to be made with values of $\mu < \mu_j$. (This is especially true if $D_1 > 1$ which indicates negative curvature over the step from x_k to $x_{k,j}$.)

We can also define

$$D_2 = \frac{F_{k,j} - F_k}{g_k^T p_{k,j} + \frac{1}{2} p_{k,j}^T G_k p_{k,j}} \tag{10}$$

which measures how the actual change in function compares with a quadratic prediction. Hence if D_2 is close to 1 we can assume that $x_{k,j}$ is near the true CSDP.

The measures D_1 and D_2 relate the behaviour at $x_{k,j}$ to expectations at the initial point x_k, However, when we have used several values of μ to compute trial steps away from x_k we can also consider more local behaviour the function around $x_{k,j}$ by using a quadratic model of F interpolating the points $x_{k,j-2}$, $x_{k,j-1}$ and $x_{k,j}$ to indicate whether a further step using $\mu_{j+1} < \mu_j$ is likely to make a significant further reduction in the objective function. This is discussed in a subsequent section.

Of course, a trial point $x_{k,j}$ may turn out be unacceptable because $F_{k,j} \geq F_k$. If this happens then $D_1 < 0$. Hence, in order to ensure that an iteration produces a non-trivial decrease in the objective function, we regard a step as unsuccessful if it causes D_1 to fall below some small positive tolerance D_1^{min}. In such a case we must retreat along the path by solving (4) with a value of $\mu > \mu_j$. (If $j > 1$ we could simply accept the previous point $x_{k,j-1}$.)

Once we have found an acceptable $x_{k,j}$ beyond which it does not seem beneficial to explore the current CSDP any further, we set $x_{k+1} = x_{k,j}$ and recompute the Hessian G_{k+1} ready to trace out a new CSDP. The ideas outlined in this section will be strengthened into computable tests in the algorithm given in section 6.

3.2 Approximating the CSDP when G_k is positive definite

The use of (4) may be rather more straightforward when G_k is positive definite because the initial choice $\mu_1 = 0$ is reasonable. Indeed we could avoid making any other choice by reverting to a standard line search form of Newton method and obtaining

$$x_{k+1} = x_k + sp_k \quad \text{where } G_k p_k = -g_k$$

with the scalar step size, s, being chosen so that $F(x_{k+1}) < F_k$. However we can also use a strategy which conducts a curvilinear search in terms of μ, as in the preceding subsection.

The coefficient matrix in (4) will be positive definite if $\mu > \mu_{min} = -\lambda_{min}$ where λ_{min} (> 0) is the smallest eigenvalue of G_k. The initial choice μ_1 may either be zero or a value calculated on the basis of a step-size restriction as discussed in section 5.1, below. Suppose, as before, that $p_{k,j}$ is the solution of (4) when $\mu = \mu_j$. Decisions on whether to extrapolate beyond a trial point $x_{k,j} = x_k + p_{k,j}$ by reducing μ can be made on the basis of the ratios D_1 and D_2 defined in (9) and (10) and on quadratic interpolations through the three most recent trial points. Exploration along the approximate CSDP must not go beyond the point where $D_1 < D_1^{min}$.

4 Illustrative examples

It may be helpful to give some examples which show how a curvilinear search algorithm can behave. We consider a sample problem **T1** which involves minimizing

$$F = x_1 x_2 + 0.01(x_1^2 + 2x_2^2 - 10)^2 \tag{11}$$

from the starting point $x = (2.5, 1, 6)^T$ (at which the Hessian is not positive definite). Figure 1 shows the points traced out along an approximate CSDP calculated as outlined in section 3.1. The circled points show the start and finish of the iteration and the plain dots are intermediate trial points. The initial μ_1 is chosen to limit the size of the initial step $||p_{k,1}|| < 0.1$ (using a strategy outlined in section 5.1). The adjustment of μ is done according to

$$\mu_{j+1} = \mu_j - \kappa(\mu_j - \mu_{min}) \tag{12}$$

with $\kappa = 0.5$. The path illustrated is obtained by continuing to decrease μ until $F_{k,j} \geq F_{k,j-1}$.

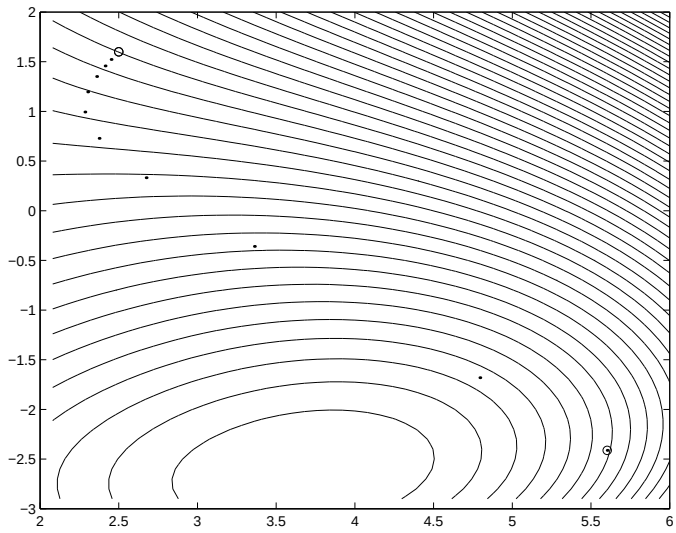

Figure 1: An approximate CSDP iteration for problem T1

At the end point of the approximate CSDP in Figure 1 the Hessian of the function (11) is positive definite and so a single curvilinear search has succeeded in escaping from the non-convex region. From this point, four further simple Newton iterations are sufficient to locate the minimum, as can be seen in Figure 2.

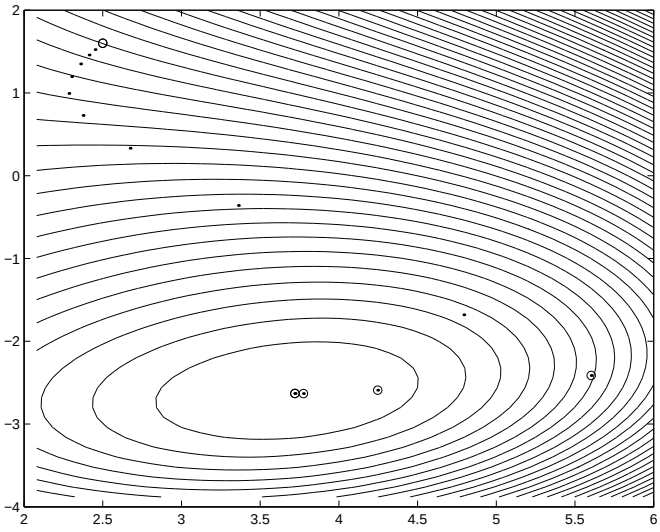

Figure 2: A complete solution of problem **T1**

Figures 1 and 2 suggest that the exploration along the CSDP during the first iteration has gone rather further than necessary. Therefore we consider what happens if we curtail the curvilinear search by allowing extrapolation only as long as the steps are in good agreement with the quadratic model. Specifically, the progress shown in Figure 3 is obtained with the threshold $|1 - D_2| < 0.15$. The iterates now appear to take a rather more direct route to the solution and only fourteen function evaluations are needed (as compared with seventeen evaluations for the path shown in Figure 2).

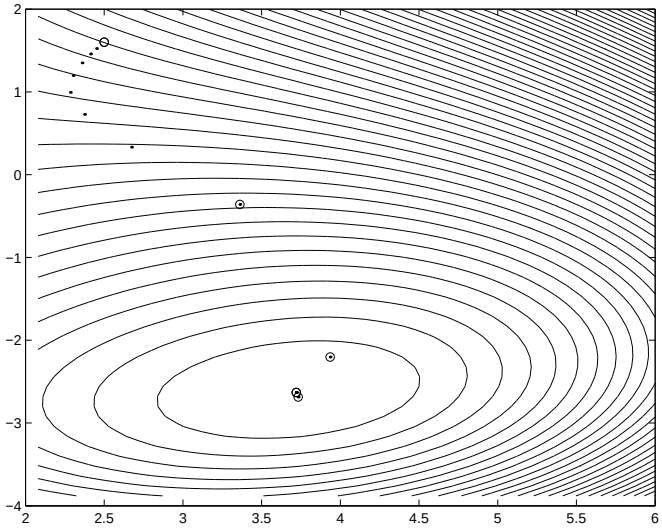

Figure 3: A solution of problem **T1** with CSDP extrapolation governed by D_2

This simple example illustrates the factors which can affect progress made by a curvilinear search. Allowing the CSDP extrapolation to go a little too far, as in Figure 2, involves additional unproductive function evaluations; but, on the other hand, we have found that being overly restrictive about exploring the CSDP can lead to an increase in the number of iterations. We can expect broadly similar principles to apply for larger and more difficult problems and these considerations will inform the development of a practical curvilinear search algorithm.

Before proceeding to a discussion of such an algorithm, we illustrate the use of the system (4) in a case when the Hessian is positive definite. We consider problem **T2** which involves minimizing the function

$$F = x_1 x_2 + 0.001(x_1^2 + 2x_2^2 - 10)^4$$

and we consider the starting guess $x_0 = (4,\ -2)^T$ which is marked as **O** in Figure 4. This figure shows the trial points obtained by solving (4) with $\mu \leq 0$. The point

A is reached by the simple Newton step with $\mu = 0$. The corresponding function value is $F \approx 3.45$. Extrapolation using $\kappa = 0.25$ in (12) gives $\mu \approx -8.4$ and the step produced by (4) gives the trial point **B**. Here the function value is reduced to $F \approx 3.0$. A further extrapolation using $\mu \approx -14.7$ leads to the circled point **C** where $F \approx 2.65$. (This is taken as the end point of the search because the value of D_2 indicates insufficient agreement with the quadratic model.)

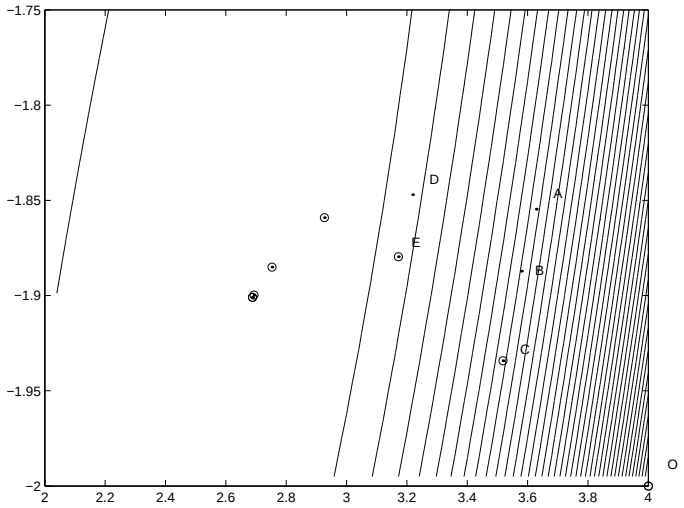

Figure 4: Solution of problem **T2**

Figure 4 also shows the second (and subsequent) iterations based on (4). The dotted point **D** is reached by the Newton step away from **C**; but the circled point **E** is obtained by solving (4) with $\mu \approx -3$ and this gives a bigger function decrease. Hence this example shows that, even when the Hessian is positive definite, it can be beneficial to perform a curvilinear search, based on varying μ in (4).

5 Ingredients of a curvilinear search algorithm

5.1 Initial choice of μ for exploring the CSDP

We now consider the initial choice $\mu = \mu_1$. In the spirit of trust region methods we can relate this to an estimate of $||p_{k,1}||_2$. Suppose Δ_k defines an acceptable step size calculated on the basis of observed function behaviour during the iteration leading to the current point x_k. (A value for Δ_0 at the start of the first iteration must be set arbitrarily). Suppose also that G_k has factors given by

$$G_k = RDR^T \quad \text{where} \quad RR^T = I \quad \text{and } D \text{ is a diagonal matrix} \qquad (13)$$

The elements of D are the eigenvalues of G_k and the system (4) can be written

$$R(\mu I + D)R^T p_k = -g_k.$$

Hence the solution of (4) for each value of μ can be done via

$$\hat{g}_k = R^T g_k; \;\; \hat{p}_{k,i} = \frac{\hat{g}_{k,i}}{\mu + d_{ii}}, \;\; i = 1, ..., n; \;\; p_k = R\hat{p}_k. \tag{14}$$

Because of the orthonormality of R we can deduce that

$$||p_k||_2 \leq \frac{1}{\mu + \lambda_{min}} ||g_k||_2$$

where λ_{min} is the most negative (or least positive) eigenvalue of G_k. Hence in order to *guarantee* that $||p_k||_2 < \Delta_k$ we require

$$\frac{1}{\mu + \lambda_{min}} \leq \frac{\Delta_k}{||g_k||_2} \;\text{ and so }\; \mu \geq \frac{||g_k||_2}{\Delta_k} - \lambda_{min}.$$

Since we must have $\mu > \mu_{min}$, an initial μ_1 for the iteration which moves away from x_k can be obtained from the safeguarded formula

$$\mu_1 = \max(\gamma \mu_{min}, \frac{||g_k||_2}{\Delta_k} - \lambda_{min}) \tag{15}$$

for some $\gamma > 1$. In the case when G_k is positive definite, however, it may be preferable to allow the choice $\mu_1 = 0$ if it does not imply too large an initial step. Hence we could use a formula of the form

$$\mu_1 = \max(0, \frac{||g_k||_2}{c\Delta_k} - \lambda_{min}) \tag{16}$$

which allows the Newton step to be taken using a bound $||p_{k,1}||_2 \leq c\Delta_k$ where c is some chosen constant. (We note that (15) and (16) only require knowledge of λ_{min} and do not require the calculation of the matrices R and D in (13).)

5.2 Choosing the step size Δ_{k+1}

After we have completed a curvilinear search to find a new point x_{k+1} we need to choose Δ_{k+1} as an acceptable size for the first trial step on the iteration which leads away from x_{k+1}. We could simply set $\Delta_{k+1} = ||x_{k+1} - x_k||_2$. However it would probably be better to relate Δ_{k+1} to the length of step over which the local quadratic approximation is accurate. We can base this on the error measure D_2 given by

$$D_2 = \frac{F_{k+1} - F_k}{p^T g_k + \frac{1}{2} p^T G_k p} \quad \text{where} \;\; p = x_{k+1} - x_k.$$

Suppose we regard a local quadratic approximation as being adequate if

$$|D_2 - 1| < D_2^{tol} \tag{17}$$

where D_2^{tol} is a specified tolerance. Clearly, if (17) is satisfied at the final point of a curvilinear search then we can choose $\Delta_{k+1} = ||x_{k+1} - x_k||_2$. However if (17) does not hold we may be able to estimate the largest steplength over which a quadratic model would be adequate. For brevity we write $A = p^T g_k$ and $B = \frac{1}{2} p^T G_k p$. We suppose that

$$F_{k+1} = F_k + p^T g_k + \frac{1}{2} p^T G_k p + C = F_k + A + B + C$$

where C represents the error in the local quadratic model of F over the step p. Then

$$D_2 = 1 + \frac{C}{A+B}$$

and so we can deduce that the non-quadratic error is given by

$$C = (D_2 - 1)(A+B).$$

If q is a positive scalar we can now estimate the value of D_2 that *would have* occurred if x_{k+1} had been obtained as $x_k + qp$. A and B are, respectively, linear and quadratic functions of stepsize and so, if we assume that C varies as the cube of the stepsize, we can write

$$D_2(q) = 1 + \frac{q^3 C}{qA + q^2 B}.$$

In order to estimate the value of q which gives $|D_2 - 1| = D_2^{tol}$ we let

$$D = \begin{cases} D_2^{tol} & \text{if } D_2 > 1 \\ -D_2^{tol} & \text{if } D_2 \leq 1 \end{cases}$$

We then want to solve

$$\frac{q^3 C}{qA + q^2 B} = D$$

which simplifies to

$$Cq^2 - BDq - AD = 0. \tag{18}$$

If (18) has a real positive root q we can choose the initial step size limit for the next iteration as $\Delta_{k+1} = q||x_{k+1} - x_k||_2$. If however (18) has both roots negative or complex then we can revert to a choice like

$$\Delta_{k+1} = ||x_{k+1} - x_k||_2 \quad \text{if } |1 - D_2| \leq D_2^{tol}$$

$$\Delta_{k+1} = \frac{1}{2}||x_{k+1} - x_k||_2 \quad \text{if } |1 - D_2| > D_2^{tol}$$

(In our numerical tests to date we have found that (18) does usually yield a real, positive root.)

5.3 Controlling extrapolation along the CSDP

Each trial point along the CSDP corresponds to some choice of μ in (4). In order to explore the CSDP systematically by choosing suitable values of μ we have found it convenient to work with τ defined by

$$\tau = (\mu - \mu_{min})^{-1} \quad \text{so that} \quad \mu = \mu_{min} + \frac{1}{\tau}. \tag{19}$$

We can think of τ as being related to the stepsize used in the numerical solution of (1). In particular $\tau \to 0$ as $\mu \to \infty$ and hence $\tau \approx 0$ corresponds to the situation when (4) yields a very small step in the steepest descent direction.

Consider the initial step $p_{k,1}$ of an iteration which has been determined using $\mu = \mu_1$ in (4). Suppose that $x_{k,1}$ from (8) gives a function value $F_{k,1}$. Recall that, if F is approximately quadratic, D_1 from (9) will have a value about 0.5 if $x_{k,1}$ is near to the minimum of F along the direction $p_{k,1}$. Hence, if D_1 is greater than some threshold $D_1^{max} (> 0.5)$, there may be scope for extrapolation beyond $x_{k,1}$.

If $\tau_1 = (\mu_1 - \mu_{min})^{-1}$ then we could simply obtain μ_2 from (19) with $\tau = \tau_2 = \alpha\tau_1$ (for some $\alpha > 1$). If $D_1 < 1$, however, a less arbitrary choice of τ_2 could be made by treating D_1 as a linear function of τ and estimating the value which would give $D_1 = 0.5$. This is

$$\tau = \frac{0.5\tau_1}{1 - D_1}$$

In practice we combine these two possibilities and set

$$\tau_2 = \alpha\tau_1 \quad \text{if } D_1 \geq 1; \qquad \tau_2 = \min(\alpha\tau_1, \frac{0.5\tau_1}{1 - D_1}) \quad \text{if } D_1 < 1. \tag{20}$$

Exploration of the CSDP continues with μ_2 being obtained from (19) to yield $p_{k,2}$ from (4) and $x_{k,2}$ from (8). Using $F_{k,2} = F(x_{k,2})$ we can calculate D_1 from (9). The decision to extrapolate beyond $x_{k,2}$ depends on the condition $D_1 > D_1^{max}$ and also on the behaviour of a quadratic model of F

$$F = Q(\tau) = a + b\tau + c\tau^2$$

with coefficients calculated to interpolate

$$Q(0) = F_k, \quad Q(\tau_1) = F_{k,1}, \quad Q(\tau_2) = F_{k,2}.$$

If $c \leq 0$ – i.e. if $Q(\tau)$ is non-convex – then a further exploratory value $\tau_3 = \alpha\tau_2$ can be tried. But if $c > 0$ we choose to keep extrapolating only if two further conditions hold, namely

$$F_c < F_b \quad \text{and} \quad \rho = \frac{Q'(\tau_c)}{Q'(\tau_a)} > \rho^{min}$$

where ρ^{mim} is a small positive constant. These suggest that F is still decreasing significantly along the path. If these conditions do hold then τ^* is found as the minimizer of $Q(\tau)$ and we set

$$\tau_3 = \min(\alpha\tau_3, \tau^*).$$

Extrapolation continues in a similar way using $\tau_j,\ \ j = 3,4,...$ to obtain $\mu_j, p_{k,j}, x_{k,j}, F_{k,j}$ and D_1 from (9). The extrapolation algorithm can be summarised as follows:

Algorithm Q-ext(j)
Given $\tau_{j-2},\ \tau_{j-1},\ \tau_j,\ x_{k,j-2},\ x_{k,j-1},\ x_{k,j},\ F_{k,j-2},\ F_{k,j-1},\ F_{k,j}$.
Set $flag = 0$
Find a,b,c so $Q(\tau) = a + b\tau + c\tau^2$ interpolates $Q(\tau_i) = F_{k,i},\ \ i = j-2,\ j-1, j$
If $c > 0$ and ($F_{k,j} > F_{k,j-1}$ or $Q'(\tau_j) > \rho^{min} Q'(\tau_{j-2})$)
  Set $flag = 1$ and exit
Else
  If $c \leq 0$
    Set $\tau_{j+1} = \alpha\tau_j$
  Else
    Calculate τ^* as the minimizer of $Q(\tau)$ and set $\tau_{j+1} = \min(\alpha\tau_j, \tau^*)$

5.4 Controlling interpolation along the CSDP

If the step taken when $\tau = \tau_j$ does not produce an acceptable function decrease – i.e. if it yields $x_{k,j}$ and $F_{k,j}$ such that $D_1 < D_1^{min}$ – then we need to interpolate a new point using $\tau < \tau_j$.

We consider first the case when this happens when $j = 1$. In this case we can choose a scaling factor $\beta(< 1)$ and set

$$\tau_2 = \max(\beta\tau_1,\ \frac{\bar{D}_1\tau_1}{1 - D_1}) \quad \text{where} \quad \bar{D}_1 = \frac{1}{2}(D_1^{min} + D_1^{max}).$$

This uses linear interpolation to estimate the value of τ which would give $D_1 = \bar{D}_1$ but includes a safeguard to stop τ becoming too small if D_1 is much less than D_1^{min}.

If $D_1 < D_1^{min}$ occurs at $x_{k,j}$ with $j > 1$ and if successful extrapolation has already taken place then we know that there is an acceptable point corresponding to some $\tau_{best} < \tau_j$. Hence we do not want to choose τ_{j+1} smaller than τ_{best} and so we can use a more general interpolation formula

$$\tau_{j+1} = \max((\tau_j - \beta(\tau_j - \tau_{best}),\ \frac{1 - \bar{D}_1}{1 - D_1}\tau_j). \tag{21}$$

6 A prototype algorithm

We now formalise the algorithmic ideas discussed in the previous section, including some extra safeguards – for instance by putting an upper limit on τ_j when an extrapolation phase follows some interpolation steps. The resulting algorithm is called NIMP1 (denoting Newton-based IMPlicit Euler) and it involves parameters α, β, γ which are used in the adjustment of τ and the selection of μ_1 at each iteration.

Algorithm NIMP1 for minimizing $F(x)$

Choose a starting point x_0 and an initial step size Δ_0.
For $k = 0, 1, 2, ...$ (until $||g_k||$ is sufficiently small and G_k is positive definite)
Calculate F_k, g_k, G_k. Compute eigenvalues $\lambda_{max}, ..., \lambda_{min}$ of G_k. Set $\mu_{min} = -\lambda_{min}$
If $\lambda_{min} \leq 0$ obtain μ_1 from (15); otherwise set $\mu_1 = 0$.
Calculate $p_{k,1}$ from (4) with $\mu = \mu_1$. Set $\tau_1 = (\mu_1 - \mu_{min})^{-1}$. Set $\tau_{max} = \infty$.
Calculate $x_{k,1}$ from (8) and set $F_{k,1} = F(x_{k,1})$. Get D_1 from (9).
If $D_1^{max} \geq D_1 \geq D_1^{min}$ set $x_{k+1} = x_{k,1}$ and go to (*)
Else if $D_1 > D_1^{max}$ set $\tau_{best} = \tau_1$ and calculate τ_2 using (20)
Else if $D_1 < D_1^{min}$ set $\tau_{max} = \tau_j$ and calculate τ_2 from (21)
For $j = 2, 3, ...$
Set $\mu_j = \mu_{min} + \tau_j^{-1}$. Get $p_{k,j}$ from (4) with $\mu = \mu_j$ and hence $F_{k,j}$ and D_1 from (9)
If $D_1^{max} \geq D_1 \geq D_1^{min}$ set $x_{k+1} = x_{k,j}$ and go to (*)
Else if $D_1 > D_1^{max}$
Calculate τ_{j+1} using algorithm Q-ext(j)
If $flag = 1$ set $x_{k+1} = x_{k,j}$ and go to (*); otherwise set $\tau_{j+1} = \min(\tau_{j+1}, \beta\tau_{max})$
Else if $D_1 < D_1^{min}$ set $\tau_{max} = \tau_j$ and calculate τ_{j+1} using (21)
End j-loop
(*) Set $\Delta_{k+1} = ||x_{k+1} - x_k||_2$
If $|1 - D_2| > D_2^{tol}$ and $q > 0$ solves (18) then set $\Delta_{k+1} = q||x_{k+1} - x_k||_2$
end k-loop

Typical parameter choices for use in this algorithm are

$$\alpha = 1/(1-\kappa), \;\; \beta = 1/(1+\kappa), \quad \text{with} \quad \kappa = 0.5$$

$$\gamma = 1.01, \quad D_1^{min} = 0.1, \;\; D_1^{max} = 0.6, \;\; D_2^{tol} = 0.2.$$

For an n variable problem the initial step size is usually taken as $\Delta_0 = 0.1\sqrt{n}$. These values are used to obtain the results given in the next section to demonstrate performance of a prototype MATLAB implementation of this algorithm in which the eigenvalues of G_k (but not the eigenvectors) are found using the efficient built-in procedure `eig` [10].

7 Numerical tests with NIMP1

The figures in section 4 for problems **T1** and **T2** show that the NIMP1 strategy can be successful in constructing a curvilinear escape route from a non-convex region and in improving on a Newton step when the function is convex. We now consider the performance of NIMP1 on some n-variable problems whose objective function is of the form

$$F = x^T Dx - b^T x + M[\sum_{k=1}^{n} c_k x_k^2 - 1]^2 \tag{22}$$

where D is a diagonal matrix whose elements are equally spaced between d_{max} and d_{min}; b is an n-vector whose elements are all 0.1; c is an n-vector with $c_k = k/n^2$; and M is a positive constant. We consider four instances in which the matrix D introduces differing amounts of non-convexity.
Problem **P1** has the elements of D lying between $d_{max} = 5$ and $d_{min} = -5$.
Problem **P2** uses $d_{max} = 10$ and $d_{min} = -1$.
Problem **P3** uses $d_{max} = 1$ and $d_{min} = -10$.
Problem **P4** uses $d_{max} = d_{min} = 0$.
The starting guess for all these problems is taken as $x = (0,0,...,0)^T$ so that a large non-convex region around the origin must be traversed by the curvilinear search.

We begin by using these test problems to consider how NIMP1 performs as the parameters κ, D_1^{max} and ρ^{min} vary. These quantities control the stepsize and extent of extrapolation. In particular, as $\kappa \to 1$ the extrapolation steps get bigger; and as $D_1^{max} \to 0.5$ and/or $\rho^{min} \to 0$ the number of permitted extrapolation steps increases. Table 1 shows numbers of iterations and function calls needed for problems **P1** – **P4** with $n = 100$ and $M = 100$.

On the basis of these results we choose the settings $\kappa = 0.7$, $D_1^{min} = 0.7$, $\rho^{min} = 0.2$ for use in a comparison of NIMP1 with a truncated Newton trust-region method implemented `fminunc` in the MATLAB optimization toolbox [10]. The method in `fminunc` is described in [11], [12] and it is comparable with NIMP1 in that it uses the exact Hessian of the objective function and works with a trust-region subproblem (5), (6) on every iteration. The approach differs from NIMP1 however in not obtaining an exact solution to the trust-region subproblem but rather by restricting itself to a two-dimensional subspace. This subspace is defined by the negative gradient $-g_k$ together with *either* an approximate Newton direction, $n \approx -G_k^{-1} g_k$ *or* a direction of negative curvature, s, such that $s^T G_k s < 0$. Obtaining the Newton direction or a direction of negative curvature could involve the solution of (4) with $\mu = 0$ or the calculation of the eigensystem of G_k. However the method in `fminunc` seeks to avoid doing as much work as NIMP1 on each iteration and hence it finds n or s, by applying a preconditioned conjugate gradient (PCG) method (see [13]) to the system $G_k n = -g_k$. When the search is far from the optimum the PCG method may be terminated with quite a low-accuracy approximation to the Newton direction; and, in particular, if G_k is found to be non positive-definite the PCG

	$D_1^{max} = 0.5,\ \rho^{min} = 0.0$		
Problem	$\kappa = 0.5$	$\kappa = 0.7$	$\kappa = 0.9$
P1	6/29	5/22	6/22
P2	4/22	4/18	5/16
P3	8/39	8/36	9/53
P4	11/39	11/37	11/38
	$D_1^{max} = 0.7,\ \rho^{min} = 0.2$		
Problem	$\kappa = 0.5$	$\kappa = 0.7$	$\kappa = 0.9$
P1	6/21	5/16	6/16
P2	4/16	4/13	5/13
P3	8/25	8/22	10/31
P4	12/36	11/26	11/32
	$D_1^{max} = 0.9,\ \rho^{min} = 0.75$		
Problem	$\kappa = 0.5$	$\kappa = 0.7$	$\kappa = 0.9$
P1	8/24	7/19	6/16
P2	4/16	4/13	5/13
P3	8/25	8/22	15/36
P4	11/31	14/42	14/35

Table 1: Iterations/function calls for NIMP1

method returns a direction of negative curvature, rather than an approximation to the Newton direction.

As well as comparing numbers of iterations, function calls needed by NIMP1 and `fminunc` for the solution of various instances of problem **P1** - **P4** we also consider a *time-ratio* which is given by the execution time by NIMP1 divided by the time for `fminunc`.

In Tables 2 - 5 we also quote numbers of iterations and function calls to solve the problems using the trust-region approach NMTR available on the NEOS server [15]. This computes correction steps via an approximate solution to the trust-region problem (5), (6) using an approach described by More and Sorenson [14].

n	M	NIMP1	`fminunc`	time-ratio	NMTR
100	10	6/18	134/134	0.07	18/21
100	100	5/16	54/54	0.14	17/20
100	1000	7/19	40/40	0.24	20/23
100	10000	9/33	54/54	0.45	30/35

Table 2: Comparative results for NIMP1 and `fminunc` on problem **P1**

These results present a fairly consistent picture. NIMP1 typically uses considerably

n	M	NIMP1	fminunc	time-ratio	NMTR
100	10	5/16	51/51	0.15	12/14
100	100	4/13	34/34	0.18	12/14
100	1000	6/17	29/29	0.29	18/20
100	10000	7/20	49/49	0.25	27/30

Table 3: Comparative results for NIMP1 and fminunc on problem **P2**

n	M	NIMP1	fminunc	time-ratio	NMTR
100	10	6/19	216/216	0.04	26/30
100	100	8/22	75/75	0.04	22/25
100	1000	11/29	44/44	0.36	21/24
100	10000	23/62	63/63	0.74	30/33

Table 4: Comparative results for NIMP1 and fminunc on problem **P3**

n	M	NIMP1	fminunc	time-ratio	NMTR
100	10	8/26	85/85	0.15	13/15
100	100	11/26	68/68	0.23	15/17
100	1000	19/59	66/66	0.38	23/27
100	10000	34/118	79/79	0.8	42/46
100	100000	68/238	153/154	1.0	77/90

Table 5: Comparative results for NIMP1 and fminunc on problem **P4**

fewer iterations than fminunc although it sometimes uses more function evaluations. This is consistent with the fact that NIMP1 seeks to extrapolate along a curved path on each iteration while fminunc does not. Hence NIMP1 often makes more progress on the basis of each (possibly costly) Hessian evaluation. In consequence, the time-ratio suggests that NIMP1 is very much more efficient. It is only on problem **P4**, where there is no strictly negative curvature in the first term of (22) that fminunc outperforms NIMP1 at larger values of M. Even then the time-ratio does not give fminunc an advantage.

The performance of NMTR is much more competitive with NIMP1. Generally speaking, NIMP1 uses fewer iterations but requires more function calls in about half the examples. NMTR seems to perform better for the larger values of the weighting parameter M.

As a final comparison we consider some larger-dimensional versions of problems P1 – P4 along with one further test example **Problem T6(n)** which features the

objective function:

$$F = \rho \sum_{i=1}^{n} (1 - x_{i+1}/x_i)^2 + (s_n - s_f)^2 + (u_n - u_f)^2$$

where

$$s_i = s_{i-1} + u_{i-1}\tau + \frac{1}{2} x_i \tau^2; \quad u_i = u_{i-1} + x_i \tau$$

using $\rho = 0.01$, $\tau = \frac{3}{n}$, $s_0 = u_0 = u_f = 0$, $s_f = 1.5$.
The starting point is
$x_i = 0.66$ $(i = 1, .., \frac{n}{2})$; $x_i = -0.66$ $(i = \frac{n}{2} + 1, ..., n)$.
This problem is described in [16] and is interesting for us because the Hessian remains non positive-definite until quite close to the solution.

We do not include NMTR in these comparisons because the NEOS server does not allow the larger problems to be submitted. Instead we quote results for a version of NIMP1 called NIMP1-ls which performs an Armijo-type line search along the Newton direction whenever G_k is positive definite. This means that it will behave like a conventional Newton method in the convex region round a solution.

The results in Table 6 are quite promising. While NIMP1 is not invariably better than `fminunc`, it is seldom significantly worse in terms of either run-time or counts of iterations and function calls. Moreover, it is worth noting that the version NIMP1-ls is never inferior to NIMP1 and is occasionally noticeably better. The curvilinear search, therefore, is principally of benefit in the non-convex regions for which it was originally intended.

8 Conclusions and future work

In this paper we have revisited and extended some familiar ideas for dealing with nonconvex objective functions in optimization problems. These ideas centre on the strategy of growing an approximation to the steepest descent trajectory away from the starting point of every iteration. This can be viewed as a curvilinear search, with the search parameter being the value of μ in the trust-region equation (4). We have shown that this basic idea can work quite well when the Hessian matrix is used as in the proposed algorithm NIMP1. An interesting consequence of our initial ideas is the observation that, for highly non-quadratic functions, it can be beneficial to carry out a curvilinear search by varying μ in (4) even when the Hessian matrix is positive-definite.

Our objective has been to understand the features which are important in the construction of an approximate CSDP through a non-convex region in the most favourable case when the Hessian matrix is available. While we have made quite good progress

P1(n,10000)				
n	NIMP1	`fminunc`	time-ratio	NIMP1-ls
200	11/29	62/63	0.42	12/30
400	10/27	86/87	0.44	11/29
800	13/34	124/125	0.5	13/33
P2(n,10000)				
n	NIMP1	`fminunc`	time-ratio	NIMP1-ls
200	7/20	59/60	0.43	7/20
400	8/22	80/81	0.4	8/22
800	6/18	114/115	0.29	6/18
P3(n,10000)				
n	NIMP1	`fminunc`	time-ratio	NIMP1-ls
200	26/70	66/67	0.74	22/55
400	27/71	74/75	1.1	27/68
800	29/71	105/106	1.1	27/68
P4(n,10000)				
n	NIMP1	`fminunc`	time-ratio	NIMP1-ls
200	28/96	125/126	0.75	26/63
400	21/56	228/229	0.37	21/55
800	20/71	442/443	0.26	18/47
T6				
n	NIMP1	`fminunc`	time-ratio	NIMP1-ls
100	11/26	14/15	0.46	11/26
200	13/37	21/22	1.1	13/37
400	15/52	27/28	0.59	15/52
800	21/74	32/33	0.7	21/74

Table 6: Comparative results for NIMP1 and `fminunc` on larger problems

in gaining such understanding there are a number of issues that need further exploration. One of these concerns the repeated solution of (4).

If each iteration of NIMP1 solves the linear system (4) for several different values of μ then the computational cost will be quite high and may be justifiable only if the iteration makes much more substantial progress along a curved path than could be obtained by a straight-line step along p_k given by solving (4) for a single value of μ. In the reported results the repeated solutions of (4) have been done by using a fresh Cholesky factorization of the coefficient matrix for each μ. However we might consider other schemes outlined below.
(i) We can avoid repeated Cholesky factorization by once-and-for-all determination the eigensystem of G_k via the factorization $G_k = RDR^T$. Since $RR^T = I$, the system

(4) can be written

$$R(\mu I + D)R^T p_k = -g_k$$

and the solution for each value of μ can be done via the steps

$$\hat{g}_k = R^T g_k; \;\; \hat{p}_{k,i} = \frac{\hat{g}_{k,i}}{\mu + d_{ii}}, \;\; i = 1, ..., n; \;\; p_k = R\hat{p}_k. \tag{23}$$

For a single solution of (4), use of the eigenvalue calculation is more expensive than a Cholesky factorization. But if several values of μ are tried then the second and subsequent solutions are cheaper than a re-factorization.
(ii) Another possible approach to repeated solution of (4) would be to seek $p_{k,j}$ iteratively. For this we would use the factors of the coefficient matrix in (4) with $\mu = \mu_1$. If $L_1 L_1^T = (\mu_1 I + G_k)$ then we can consider an iteration of the form

$$\tilde{p}_{i+1} = L_1^{-T}(L_1^{-1}(-g_k - (\mu_j - \mu_0)\tilde{p}_i))$$

as a way of obtaining $\tilde{p}_i \to p_{k,j}$.
(iii) A third possibility is to consider problem (5), (6) which underlies (4). Hamaker [18] has recently proposed an iterative approach to problems of this type which appears to be very efficient, under certain conditions. This approach might be capable of application to the NIMP1 subproblem of choosing search directions p_k.

The relative merits of *(i)* – *(iii)* against the approach implemented in NIMP1 remain to be investigated but must depend, in part, on how far and how accurately we want to pursue a curved path solution of (1).

The differential equation (1) can also be solved in a way given by Behrman [1]. This approach uses the RDR^T factorization together with the explicit formula

$$p_k = -R\Lambda R^T g_k \tag{24}$$

where Λ is a diagonal matrix whose elements are derived from the eigenvalues on the diagonal of D via

$$\Lambda_{ii} = \begin{cases} \frac{exp(\frac{d_{ii}}{\mu} - 1)}{d_{ii}} & \text{if} \;\; d_{ii} \neq 0 \\ \frac{1}{\mu} & \text{if} \;\; d_{ii} = 0 \end{cases} \tag{25}$$

Behrman uses the search direction (24) in a way which is broadly similar to what we have described in previous sections of this paper – namely, by adjusting the μ parameter in order to perform a curvilinear search so long as the trial points reduce the objective function and are acceptable close to the true CSDP. Beddiaf [17] quotes results with an implementation of the Behrman method which show that it does not perform particularly well in comparison with NIMP1.

In revisiting the familiar problem of non-convexity in unconstrained optimization we have so far adopted a pragmatic approach of developing and testing algorithmic ideas via the observation of behaviour on numerical examples. We have not,

however, given any formal analysis of the iterative scheme that we propose. If – as in NIMP1-ls – we revert to a standard Newton method in the convex region round a solution then the ultimate convergence is assured. Global convergence requires us to be able to show that the Wolfe conditions are satisfied by the curvilinear search procedure. Some refinement of NIMP1 may still be needed in order to establish this. We have not, for example, specified what should be done if $g_k = 0$ and G_k is not positive definite.

References

[1] Behrman, W., An Efficient Gradient Flow Method for Unconstrained Optimization, PhD Thesis, Stanford University, 1998.

[2] Brown, A.A., Optimisation Methods Involving the Solution of Ordinary Differential Equations, PhD Thesis, Hatfield Polytechnic, 1986.

[3] Brown, A.A., Bartholomew-Biggs, M.C. Some Effective Methods for Unconstrained Optimization based on the Solution of Systems of Ordinary Differential Equations, J. Optim. Theory Appl. 62(2):211-224,1989.

[4] Botsaris, C.A., Jacobson, D.H. A Newton-type Curvilinear Search Method for Optimization, J. Maths Anal.Appl., 54(1):217-229,1976.

[5] Vial, J.P., Zang, Israel, Unconstrained Optimization by Approximation of the Gradient Path. Maths. Oper. Res., 2(3):253-265, 1977.

[6] Botsaris, C.A., Differential Gradient Methods. J. Maths. Anal. Appl., 63(1):177-198, 1978

[7] Botsaris, C.A., A Curvilinear Optimization Method Based Upon Iterative Estimation of the Eigensystem of the Hessian Matrix. J. Maths. Anal. Appl., 63(2):396-411, 1978

[8] Neculai,A., Gradient Flow Algorithm for Unconstrained Optimization ICI Technical Report, April 2004.

[9] Conn,A.,R., Gould, N.I.M., Toint, P.T., Trust Region Methods MPS-SIAM Series on Optimization, Philadelphia, 2000.

[10] The Mathworks Co, MATLAB documentation, www.mathworks.com

[11] Branch, M.A., T.F. Coleman, Y. Li, A Subspace, Interior, and Conjugate Gradient Method for Large-Scale Bound-Constrained Minimization Problems, SIAM Journal on Scientific Computing, Vol. 21, Number 1, pp. 1-23, 1999.

[12] Byrd, R.H., R.B. Schnabel, and G.A. Shultz, Approximate Solution of the Trust Region Problem by Minimization over Two-Dimensional Subspaces, Mathematical Programming, Vol. 40, pp. 247-263, 1988.

[13] Broyden C.G. and M.T. Vespucci, Krylov Solvers for Linear Algebraic Systems, Studies in Computational Mathematics, 11, Elsevier, 2004.

[14] More, J.J. and Sorenson, D.C., Computing a Trust Region Step, Technical Report ANL-81-83, Argonne National Laboratory, 1981

[15] http://www-neos.mcs.anl.gov/neos/solvers/uco:NMTR/C.html.

[16] Bartholomew-Biggs, M.C., Nonlinear Optimization with Engineering Applications, Springer, 2008.

[17] Beddiaf, S., Continuous steepest descent path for traversing non-convex regions, Advanced Modeling and Optimization, Vol. 1 Number 1, 2009.

[18] Hamaker, C., Optimization of a constrained quadratic function, Studies in Informatics and Control, Vol. 18, Number 1, 2009.