VSP International
Science Publishers
P.O. Box 346, 3700 AH Zeist
The Netherlands

Lecture Series on Computer and Computational Sciences
Volume , , pp. -

A Rectangular Trust-Region Approach for Unconstrained and Box-Constrained Optimization Problems

C. Voglis[1] and I.E. Lagaris[2]

Department of Computer Science,
University of Ioannina,
P.O. BOX 1186 - GR 45110 Ioannina, Greece

Abstract: First a quadratic programming problem with positive definite Hessian and bound constraints is solved, using a Lagrange multiplier approach. Then a trust region method for non-linear optimization with box constraints is developed, where the trust region is a hyperbox, in contrast with the usual hypersphere or hyperellipsoid shapes. The resulting subproblem is solved using our above mentioned QP technique.

Keywords: Quadratic programming,

Mathematics Subject Classification: 90C20, 90C30, 90C53

1 Introduction

Non-linear optimization plays an important role in many fields of science and engineering, in the industry, as well as in a plethora of practical problems. Frequently the optimization parameters are constrained inside a range imposed by the nature of the problem at hand. Developing methods for bound constrained optimization is hence quite useful. The most efficient optimization methods are based on Newton's method where a quadratic model is adopted as a local approximation to the objective function. Two general approaches have been followed. One uses a line–search along a properly selected descent direction, while the other permits steps of restricted size in an effort to maintain the reliability of the quadratic approximation. The approaches in this second class, bear the generic name Trust-Region techniques. In this article we will deal with a method of that type. We develop a method that adopts a hyperbox geometry for the trust region. This has the obvious advantage of the linearity of the trust region subproblem constraints. In addition allows effortless adaptation to bound constrained optimization problems. We analyze the approach for the bound-constrained quadratic programming in section 3, and we present an algorithmic solution in section 3.1. In section 4, we embed this QP technique in the general setting of the trust region approach, for both unconstrained and bound–constrained problems.

2 Trust Region Methods

Trust region methods fall in the category of sequential quadratic programming. The algorithms in this class are iterative procedures in which the objective function $f(x)$ is represented by a quadratic

[1]Corresponding author. E-mail: voglis@cs.uoi.gr
[2]E-mail: lagaris@cs.uoi.gr.gr

model inside a suitable neighborhood (the trust region) of the current iterate, as implied by the Taylor series expansion. This local model of $f(x)$ at the k^{th} iteration can be written as:

$$f(x^k + s) \approx m^k(s) = f(x^k) + s^T g^{(k)} + \frac{1}{2} s^T B^{(k)} s \tag{1}$$

where $g^{(k)} = \nabla f(x^{(k)})$ and $B^{(k)}$ is a symmetric approximation to $\nabla^2 f(x^{(k)})$.

The trust region may be defined by:

$$\mathbf{T}^{(k)} = \{x \in \Re^n \mid ||x - x^{(k)}|| \leq \Delta^{(k)}\} \tag{2}$$

It is obvious that different choices for the norm lead to different trust region shapes. The Euclidean norm $||\cdot||_2$, corresponds to a hypershpere, while the $||\cdot||_\infty$ norm defines a hyperbox.

Given the model and the trust region, we seek a step $s^{(k)}$ with $||s^{(k)}|| \leq \Delta^{(k)}$, such that the model is sufficiently reduced in value. Using this step we compare the reduction in the model to that in the objective function. If they agree to a certain extend, the step is accepted and the trust region is either expanded or remains the same. Otherwise the step is rejected and the trust region is contracted. The basic trust region algorithm is sketched in Alg. 1

Algorithm 1 Basic trust region

```
S0: Pick the initial point and trust region parameter $x_0$ and $\Delta_0$, and set $k = 0$.
S1: Construct a quadratic model:
    $m^{(k)}(s) \approx f(x^{(k)} + s)$
S2: Calculate $s^{(k)}$ with $||s^{(k)}|| \leq \Delta^{(k)}$, so as to sufficiently reduce $m^{(k)}$.
S3: Compute the ratio of actual to expected reduction, $r^{(k)} = \frac{f(x^{(k)}) - f(x^{(k)} + s^{(k)})}{m^{(k)}(0) - m^{(k)}(s^{(k)})}$. This value will
    determine if the step will be accepted or not and the update for $\Delta^{(k)}$.
S4: Increment $k \leftarrow k + 1$ and repeat from S1.
```

3 Bound-constrained QP

Let $x, d \in R^N$ and B a symmetric, positive definite $N \times N$ matrix and $I = \{1, 2, \cdots, N\}$. Consider then the QP problem:

$$\min_x \frac{1}{2} x^T B x + x^T d, \text{ subject to: } a_i \leq x_i \leq b_i, \forall i \in I \tag{3}$$

We follow the Lagrange multipliers line and we construct the Lagrangian:

$$L(x, \lambda, \mu) = \frac{1}{2} x^T B x + x^T d - \lambda^T (x - a) - \mu^T (b - x) \tag{4}$$

The KKT necessary conditions at the minimum x^*, λ^*, μ^* require that:

$$\begin{aligned} Bx^* + d - \lambda^* + \mu^* &= 0 \\ \lambda_i^* \geq 0, \ \ \mu_i^* \geq 0, \ \forall i &\in I \\ \lambda_i^*(x_i^* - a_i) = 0, \ \forall i &\in I \\ \mu_i^*(b_i - x_i^*) = 0, \ \forall i &\in I \\ x_i^* \in [a_i, b_i], \ \forall i &\in I \end{aligned} \tag{5}$$

A solution to the above system of equations (5), can be obtained through an active set strategy described in detail in the following section 3.1.

3.1 The BOXCQP algorithm

Our QP algorithm is sketched in Alg 2:

Algorithm 2 BOXCQP

```
S0: Initially set: $k = 0$, $\lambda^{(0)} = \mu^{(0)} = 0$ and $x^{(0)} = -B^{-1}d$.
    If $x^{(0)}$ is feasible, Stop, the solution is: $x^* = x^{(0)}$.
    At iteration $k$, the quantities $x^{(k)}, \lambda^{(k)}, \mu^{(k)}$ are available.

S1: Define the sets:
        $L^{(k)} = \{i : x_i^{(k)} < a_i, \text{ or } x_i^{(k)} = a_i \text{ and } \lambda_i^{(k)} \geq 0\}$
        $U^{(k)} = \{i : x_i^{(k)} > b_i, \text{ or } x_i^{(k)} = b_i \text{ and } \mu_i^{(k)} \geq 0\}$
        $S^{(k)} = \{i : a_i < x_i^{(k)} < b_i, \text{ or } x_i^{(k)} = a_i \text{ and } \lambda_i^{(k)} < 0,$
              $\text{or } x_i^{(k)} = b_i \text{ and } \mu_i^{(k)} < 0\}$
    Note that $L^{(k)} \cup U^{(k)} \cup S^{(k)} = I$

S2: Set:
        $x_i^{(k+1)} = a_i,\ \mu_i^{(k+1)} = 0,\ \forall i \in L^{(k)}$
        $x_i^{(k+1)} = b_i,\ \lambda_i^{(k+1)} = 0,\ \forall i \in U^{(k)}$
        $\lambda_i^{(k+1)} = 0,\ \mu_i^{(k+1)} = 0,\ \forall i \in S^{(k)}$

S3: Solve:
        $Bx^{(k+1)} + d = \lambda^{(k+1)} - \mu^{(k+1)}$
    for the $N$ unknowns:
        $x_i^{(k+1)},\ \forall i \in S^{(k)}$
        $\mu_i^{(k+1)},\ \forall i \in U^{(k)}$
        $\lambda_i^{(k+1)},\ \forall i \in L^{(k)}$

S4: Check if the new point is a solution and decide to either stop or iterate.
    If ($x_i^{(k+1)} \in [a_i, b_i]\ \forall i \in S^{(k)}$ and $\mu_i^{(k+1)} \geq 0,\ \forall i \in U^{(k)}$
                                          and $\lambda_i^{(k+1)} \geq 0,\ \forall i \in L^{(k)}$) Then
        Stop, the solution is: $x^* = x^{(k+1)}$.
    Else
        set $k \leftarrow k + 1$ and iterate from S1
    Endif
```

The solution of the linear system in step 3 above, needs further consideration. Let us rewrite the system in a componentwise fashion.

$$\sum_{j\in I} B_{ij}x_j^{(k+1)} + d_i = \lambda_i^{(k+1)} - \mu_i^{(k+1)},\ \forall i \in I \tag{6}$$

Since $\forall i \in S^{(k)}$ we have that $\lambda_i^{(k+1)} = \mu_i^{(k+1)} = 0$, we can calculate $x_i^{(k+1)}$, $\forall i \in S^{(k)}$ by spliting the sum in eq. (6) and taking into account step 2 of the algorithm, i.e.:

$$\sum_{j\in S^{(k)}} B_{ij}x_j^{(k+1)} = -\sum_{j\in L^{(k)}} B_{ij}a_j - \sum_{j\in U^{(k)}} B_{ij}b_j - d_i,\ \forall i \in S^{(k)} \tag{7}$$

The submatrix B_{ij}, with $i,j \in S^{(k)}$ is positive definite as can be readily verified, given that the full matrix B is. The calculation of $\lambda_i^{(k+1)}$, $\forall i \in L^{(k)}$ and of $\mu_i^{(k+1)}$, $\forall i \in U^{(k)}$ is straightforward and is given by:

$$\lambda_i^{(k+1)} = \sum_{j\in I} B_{ij}x_j^{(k+1)} + d_i,\ \forall i \in L^{(k)} \tag{8}$$

$$\mu_i^{(k+1)} = -\sum_{j\in I} B_{ij}x_j^{(k+1)} - d_i,\ \forall i \in U^{(k)} \tag{9}$$

3.2 Experiments with QP

We are currently under the process to present an elegant proof for the convergence rate of the BOXCQP algorithm, although we have very strong experimental results of its efficiency. It is our strong belief that if matrix B is sufficiently positive definite our algorithm will converge to the solution.

We have contacted ... types of experiments in order to measure the speed of our BOXCQP algorithm: Random quadratic problems, ...

4 Rectangular trust region approach

The basic motivation under the development of an robust solver for positive definite quadratic problems with simple bounds, was to solve exactly the quadratic subproblem that arises in a trust region framework using infinite norm $||\cdot||_\infty$. In this case the trust region in which we believe that the quadratic model "fits" the objective function, is a hyperbox.

4.1 Model and norm definition

The problem that we try to solve is:

$$\begin{gathered}\min_x f(x)\\ \text{subject to: } l_i \le x_i \le u_i\end{gathered} \tag{10}$$

using an infinite norm trust region method. Let $x^{(k)}$ the estimation of the solution at the k-th step of the algorithm. In each step we construct a *quadratic model* of f, and find a step $s^{(k)}$ that solves:

$$\begin{gathered}\min_s\ m(x^{(k)}+s) = f^{(k)} + {g^{(k)}}^T h + \frac{1}{2}s^T B^{(k)} s\\ \text{subject to: } ||s||_\infty \le \Delta^{(k)} \text{ and } l_i - x_i^{(k)} \le s_i \le u_i + x_i^{(k)}\end{gathered} \tag{11}$$

which is equivalent to:

$$\min_s \; m(s) = g^{(k)^T} h + \frac{1}{2} s^T B^{(k)} s$$

$$\text{subject to: } \max(l_i - x_i^{(k)}, -\Delta^{(k)}) \leq s_i \leq \min(u_i - x_i^{(k)}, \Delta^{(k)}) \tag{12}$$

where $f^{(k)} = f(x^{(k)}), g^{(k)} = \nabla f(x^{(k)}), B^{(k)}$ is a positive definite approximation of the hessian matrix. We use the BFGS formula to update $B^{(k)}$ whenever we move to a new point $x^{(k+1)}$. Thus in **S2** of the basic algorithm we have to solve a bound constrained quadratic problem with positive definite hessian.

4.2 Trust region update

We have incorporated a simple trust radius update, the same that we use in the Merlin [5] implementation for the *double dogleg* method. The update algorithm (incorporated in **S3** of the basic algorithm) is described in ...:

Algorithm 3 Trust region parameter Δ update

```
S1: Calculate the ratio of the actual to the expected reduction $r^{(k)} = (f^{(k)} - f^{(k+1)})/m^{(k)}(x^{(k)}) - q^{(k)}(x^{(k)} + s^{(k)})$ where $f^{(k)}$ stands for $f(x^{(k)})$ and $f^{(k+1)}$ for $f(x^{(k)} + s^{(k)})$

S2: Accept or reject the trial point according to

    If $r^{(k)} \leq 0$ then
        $x^{(k+1)} = x^{(k)}, f^{(k+1)} = f^{(k+1)}$
    Else
        $x^{(k+1)} = x^{(k)} + s^{(k)}$
    Endif

S3: If $r^{(k)} < 0.25$ then
        $\Delta^{(k+1)} = ||s^{(k)}||/4$
    Else if $r^{(k)} > 0.75$ and $||s^{(k)}|| = \Delta^{(k)}$ then
        $\Delta^{(k+1)} = 2\Delta^{(k)}$
    Else
        $\Delta^{(k+1)} = \Delta^{(k)}$
    Endif
```

References

[1] A. Conn, N. Gould and P. Toint, *Trust-Region methods*, MPS-SIAM Series on Optimization (2000)

[2] J. E. Dennis and R. B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, SIAM (1996)

[3] M. J. D. Powell, A new algorithm for unconstrained optimization, *Nonlinear Programming*, pp. 31–65, Academic Press, London (1970)

[4] J. J. Moré, B. S. Garbow and K. E. Hillstrom, Testing unconstrained optimization environment software, *ACM Transactions on Mathematical Software*, 7(1), pp. 17–41 (1983)

[5] D. G. Papageorgiou, I. N. Demetropoulos and I. E. Lagaris , Merlin–3.0 A Multidimensional Optimization Environment, *Comput. Phys. Commun.*, 109, pp. 227–249 (1998)

[6] M.J.D. Powell, TOLMIN: A Fortran Package for Linearly Constrained Optimization Calculation, *DAMTP* (1989)

[7] I. Bongartz, A.R. Conn, Nick Gould, Ph.L. Toint, CUTE: Constrained and unconstrained testing environment *ACM Transactions on Mathematical Software*(1993)

[8] A.R. Conn, Nick Gould and P.L. Toint, Testing a Class of Methods for Solving Minimization Problems with Simple Bounds on the Variables, *Mathematics of Computation*, 50, pp. 399–430 (1988)