On limited memory SQP methods for large scale constrained nonlinear least squares problems

Z. F. Li*

(Received 7 August 2000)

Abstract

This paper describes limited memory Sequential Quadratic Programming methods (LSQP) for a large scale equality constrained nonlinear least squares problem. By introducing additional variables, the original problem is transformed into a general equality constrained

* School of Mathematical Sciences, Australian National University, Canberra, ACT 0200, AUSTRALIA. li@maths.anu.edu.au

[0]See http://anziamj.austms.org.au/V42/CTAC99/Li for this article and ancillary services, Published 27 Nov 2000.

nonlinear programming problem with a simple objective. This is then solved by a limited memory variation of SQP methods. This overcomes one of the major drawbacks of the traditional SQP method, where a large matrix needs to be stored, and combines the best performance of the Gauss-Newton and Quasi-Newton methods by a suitable choice of the Lagrangian Hessian approximation. Our numerical tests indicate that the new method is faster than the reduced Hessian (RSQP) method, and is better able to use additional storage to accelerate convergence. For some problems it approaches the performance of the full Hessian SQP (FSQP) method adapted for least squares problems in Schittkowski [9]. However, his method cannot cope with problems with very many observations.

Contents

1 Introduction

Consider the nonlinear least squares problem

$$\min_{x \in R^p} f(x) = \frac{1}{2} \sum_{i=1}^{n} f_i(x)^2 = \frac{1}{2} F(x)^T F(x), \tag{1}$$

where $F(x) = [f_1(x), \cdots, f_n(x)]^T$. Let $J(x)$ denote the Jacobian of $F(x)$ and $S(x) = \sum_{i=1}^{n} f_i(x) \nabla^2 f_i(x)$. Then we have

$$\begin{aligned} \nabla^2 f(x) &= J(x)^T J(x) + \sum_{i=1}^{n} f_i(x) \nabla^2 f_i(x) \\ &= J(x)^T J(x) + S(x). \end{aligned} \tag{2}$$

A commonly used method for (1) is the Gauss-Newton method. This method is essentially a Newton method where the calculation of the Hessian has been simplified by assuming that $S(x)$ is small at the solution and thus can be neglected. This simplification allows us to calculate the Hessian based only on first-order derivatives, and yet a q-quadratic rate of convergence of Newton method can be retained asymptotically as $n \to \infty$. For this result to hold, it is assumed that the fitting model is correct and a large set of data are available. However, these assumptions are not always valid. When $S(x)$

is not small, a Gauss-Newton method may only have a linear rate of convergence. Thus several modifications of this method have been made in order to deal with unconstrained least squares problems efficiently. Such alternative methods involve the Levenberg-Marquart (LM) method by Osborne [8], NL2SOL by Dennis et al. [3], and a hybrid method by Fletcher et al. [4]. However, these methods cannot directly handle parameter constraints. For the SQP-based method in this study, by adding several variables, the original problem is first transformed into a constrained nonlinear programming problem subject to equality constraints, then an SQP method is used. This idea first appeared in Schittkowski [9]. Like NL2SOL and the hybrid method, this new method can achieve the best performance of the Gauss-Newton and Quasi-Newton methods by a suitable choice of the Lagrangian Hessian approximation. Moreover, this method is attractive when there are additional constraints on the sum of squares. However, this method cannot cope with problems with very many observations. This is a typical case. In this paper, we develop a limited memory SQP method (LSQP). This proposed method overcomes one of the major drawbacks of the traditional SQP method where a large matrix needs to be calculated and stored. For better performance, the special structure of the problem has been fully exploited.

In the next section, a reformulation of the original problem into a nonlinear programming problem is discussed, followed by the development of the limited memory SQP method (LSQP). Finally, the results of a comparison study are discussed in Section 4.

2 The SQP-based method

First we introduce the SQP method, and assume we want to solve the nonlinear minimization problem:

$$\min_{x \in R^p} f(x) \quad \text{s.t.} \quad c(x) = 0, \tag{3}$$

where $f(x) : R^p \to R$ and $c(x) : R^p \to R^m$ are smooth functions, and that the first derivatives of $f(x)$ and $c(x)$ are available. The SQP method is one of the most efficient methods for solving (3). The Lagrangian function associated with problem (3) is

$$l(x, \lambda) = f(x) + \lambda^T c(x), \tag{4}$$

where $\lambda \in R^m$ is called the Lagrange multiplier. Let $A(x)$ denote the Jacobian of the constraints $c(x)$. Then, each iteration of this type of method is characterized by the iterative procedure

2.1 The k-th step of the SQP method

A1 Compute the minimizer d_k of the subproblem

$$\begin{aligned} &\min_{d \in R^p} \nabla f(x_k)^T d + \tfrac{1}{2} d^T B_k d \\ &\text{s.t. } c(x_k) + A(x_k) d = 0, \end{aligned} \tag{5}$$

where B_k is an approximation to the Lagrangian Hessian $\nabla_x^2 l(x_k, \lambda_k)$.

A2 Choose a merit function $\phi_k(\alpha)$ and make a line search along d_k such that $\phi_k(\alpha_k) - \phi_k(0)$ is suitably reduced;

A3 Set $x_{k+1} = x_k + \alpha_k d_k$, and update$\lambda_k$ and B_k to give λ_{k+1} and B_{k+1}, respectively.

The solution d_k of the quadratic subproblem (5) can be written in a simple form if we choose a suitable basis for R^p to represent the search direction d_k. For this purpose, we introduce matrices $Y_k \in R^{p\times m}$ and $Z_k \in R^{p\times(p-m)}$ such that $[Y_k, Z_k]$ is nonsingular and $A(x_k)Z_k = 0$. This can be done by computing a QR decomposition of $A(x_k)^T$. We now express d_k as

$$d_k = Y_k p_k^y + Z_k p_k^z \tag{6}$$

for some vector $p_k^y \in R^m$ and $p_k^z \in R^{p-m}$. As $A(x_k)Z_k = 0$, the linear constraints in (5) become

$$c(x_k) + A(x_k)Y_k p_k^y = 0. \tag{7}$$

If $A(x_k)$ has full rank, then $A(x_k)Y_k$ is nonsingular and

$$p_k^y = -[A(x_k)Y_k]^{-1}c(x_k). \tag{8}$$

Thus

$$d_k = -Y_k[A(x_k)Y_k]^{-1}c(x_k) + Z_k p_k^z. \tag{9}$$

Substituting (9) into (5), considering $Y_k p_k^y$ as constant, and ignoring constant terms, we obtain the unconstrained quadratic problem

$$\min_{p_k^z}[Z_k^T \nabla f(x_k) + Z_k^T B_k Y_k p_k^y)]^T p_k^z + \frac{1}{2} p_k^{zT} (Z_k^T B_k Z_k) p_k^z. \tag{10}$$

If we assume that $Z_k^T B_k Z_k$ is positive definite, the solution of (10) is

$$p_k^z = -(Z_k^T B_k Z_k)^{-1}[Z_k^T \nabla f(x_k) + Z_k^T B_k Y_k p_k^y]. \tag{11}$$

This determines the search direction d_k of the SQP method.

Let $s_k = x_{k+1} - x_k$ and $\hat{q}_k = \nabla_x l(x_{k+1}, \lambda_{k+1}) - \nabla_x l(x_k, \lambda_{k+1})$. Parallel to the treatment of an unconstrained Quasi-Newton method, one could updateB_k starting from a suitable symmetric positive definite matrix B_0 via a BFGS updateformula

$$B_{k+1} = U^{BFGS}(B_k, s_k, \hat{q}_k), \tag{12}$$

where

$$U^{BFGS}(B, s, y) = B + \frac{yy^T}{s^T y} - \frac{Bss^T B}{s^T Bs}. \tag{13}$$

The above formula yields a positive definite B_{k+1} provided that B_k is positive and $s_k^T \hat{q}_k > 0$. However the latter condition is not guaranteed by a standard linear search strategy. A simple modification of Powell [7] gives positive

definite matrices even if the above condition is violated. This is done by replacing $\hat{q}_k$ in (12) with

$$q_k = \theta_k \hat{q}_k + (1 - \theta_k) B_k s_k, \tag{14}$$

where

$$\theta_k = \begin{cases} 1\,, & \text{if } s_k^T \hat{q}_k \geq 0.2 s_k^T B_k s_k\,, \\ \frac{0.8 s_k^T B_k s_k}{s_k^T B_k s_k - s_k^T \hat{q}_k}\,, & \text{otherwise.} \end{cases}$$

After we introduce the SQP method for a general optimization problem (3), we now turn our attention solving problem (1). We follow Schittkowski [9] who suggested transforming problem (1) into a general nonlinear optimization problem to which the SQP method is applied. We call this method the FSQP method because it uses the full Hessian approximation.

Introducing $z = [z^{(1)}, \cdots, z^{(n)}]^T$, we can reformulate (1) as

$$\begin{aligned} &\min_{(z,x)} \tfrac{1}{2} z^T z \\ &\text{s.t. } z - F(x) = 0. \end{aligned} \tag{15}$$

We consider (15) as a general nonlinear programming problem of the form

$$\begin{aligned} &\min_{\bar{x}} \bar{f}(\bar{x}) \\ &\text{s.t. } \bar{c}(\bar{x}) = 0, \end{aligned} \tag{16}$$

with $\bar{x} = [z^T, x^T]^T \in R^{p+n}$, $\bar{f}(\bar{x}) = \frac{1}{2} z^T z$ and $\bar{c}(\bar{x}) = z - F(x)$. It is this problem that Schittkowski solves by an SQP method.

Define the Lagrangian function $l(\bar{x}, \lambda)$ of (16) by

$$\begin{aligned} l(\bar{x}, \lambda) &= \bar{f}(\bar{x}) + \lambda^T \bar{c}(\bar{x}) \\ &= \tfrac{1}{2} z^T z + \lambda^T [z - F(x)] \end{aligned} \tag{17}$$

and let $\nabla_{\bar{x}} l(\bar{x}, \lambda)$ and $\nabla^2_{\bar{x}} l(\bar{x}, \lambda)$ denote the gradients and Hessian matrix of the Lagrangian function $l(\bar{x}, \lambda)$ with respect to $\bar{x}$, respectively. It is easy to check that

$$\nabla_{\bar{x}} l(\bar{x}, \lambda) = \begin{pmatrix} z + \lambda \\ -J(x)^T \lambda \end{pmatrix} \tag{18}$$

and

$$\nabla^2_{\bar{x}} l(\bar{x}, \lambda) = \begin{pmatrix} I & 0 \\ 0 & \tilde{S}(x) \end{pmatrix}, \tag{19}$$

where $\tilde{S}(x) = -\sum_{i=1}^{n} \lambda^{(i)} \nabla^2 f_i(x)$.

It seems to be reasonable to proceed now from a quasi-Newton matrix given by

$$B_k = \begin{pmatrix} I & 0 \\ 0 & B_k^{(2,2)} \end{pmatrix}, \tag{20}$$

where $B_k^{(2,2)} \in R^{p \times p}$ denotes a suitable positive definite approximation of $\tilde{S}(x)$. Let $d = [d_z^T, d_x^T]^T$. Inserting (20) into (5) gives the equivalent quadratic

programming subproblem

$$\begin{aligned} &\min_{d\in R^n} \tfrac{1}{2} {d_x}^T B_k^{(2,2)} d_x + \tfrac{1}{2} {d_z}^T d_z + z_k^T d_z \\ &\text{s.t. } d_z - J(x_k) d_x + z_k - F(x_k) = 0. \end{aligned} \tag{21}$$

Some simple calculations show that for solving (21) for d_x is equivalent to solving the linear system

$$[J(x_k)^T J(x_k) + B_k^{(2,2)}] d_x + J(x_k)^T F(x_k) = 0, \tag{22}$$

which yields the Newton method if $B_k^{(2,2)} = S(x_k)$ and the Gauss-Newton method if $B_k^{(2,2)} \equiv 0$. Thus, this method can be considered as a bridge between the Newton method and the Gauss-Newton method by a suitable choice of $B_k^{(2,2)}$.

However, it is not guaranteed that the matrix B_k generated by our modified Powell strategy has the form (20) even if we start from the identity matrix $B_0 = I$, and updating the projected Hessian in x space only has proved unsatisfactory by our numerical experiments. Therefore, in most cases one has to updateand store a full Hessian approximation matrix B_k. This would be impractical when n is very large.

A method which makes storage demands intermediate between the restricted update (20) and the the full SQP method is the RSQP method suggested in Biegler et al [1]. RSQP, which originally attempts to solve the problem (3) for very large p, updates only the projected Hessians in the null

space $Z_k^T B_k Z_k$ and in the range space $Z_k^T B_k Y_k$ in (11). For more details of this method see Biegler et al. [1].

For the least squares problem (16), we have $Z_k^T \nabla^2_{\bar{x}} l(\bar{x}_k, \lambda_k) Y_k = J(x_k)^T$ due to our choices of Y_k and Z_k by simple elimination of variables. We will use this exact information in our implementation of RSQP. The RSQP method works well for some problems in our experiments. However, its performance is vastly inferior to our LSQP method proposed in the next section.

3 The limited memory SQP (LSQP) method

Like RSQP, LSQP also attempts to solve problem (3) for very large p and does not need to calculate or store the full Hessian approximate B_k in (5). At every iterate x_k the algorithm stores a small number, say t, of correction pairs $\{s_i, q_i\}_{i=k-t}^{k-1}$, where s_k and q_k are defined as in Section 2. These correction vectors are used to define the limited memory matrix B_k by implicitly updating a basic matrix $B_{k,0}$. Depending on the number of vectors to be stored the oldest information is discarded and new information is added.

Assume we have $S_k = [s_{k-t}, \cdots, s_{k-1}]$, $Q_k = [q_{k-t}, \cdots, q_{k-1}]$, a diagonal matrix $D_k = \mathrm{diag}[s_{k-t}^T y_{k-t}, \cdots, s_{k-1}^T y_{k-1}]$ and the $t \times t$ lower triangular matrix L_k whose lower triangular elements are defined as: $(L_k)_{ij} = s_{k-t-1+i}^T q_{k-t-1+j}$ for $i > j$. The following lemma is useful:

Lemma 1 *Let $B_{k,0}$ be symmetric and positive definite and assume that the t pairs $\{s_i, q_i\}_{i=k-t}^{k-1}$ satisfy $s_i^T q_i > 0$. If B_k is obtained by updating $B_{k,0}$ using the standard* BFGS *formula (16) applied t times to the pairs $\{s_i, q_i\}_{i=k-t}^{k-1}$, then*

$$B_k = B_{k,0} - [B_{k,0}S_k, Q_k]H_k^{-1}\begin{bmatrix} S_k^T B_{k,0} \\ Q_k^T \end{bmatrix}, \tag{23}$$

where

$$H_k = \begin{bmatrix} S_k^T B_{k,0} S_k & L_k \\ L_k^T & -D_k \end{bmatrix}. \tag{24}$$

Proof: See Byrd et al. [2]. ♠

Note that the matrix H_k in (24) is indefinite. However its inverse can be computed using the Cholesky factorization of a related matrix. First we re-order the blocks of (24) and note that

$$\begin{bmatrix} -D_k & L_k^T \\ L_k & S_k^T B_{k,0} S_k \end{bmatrix} = \begin{bmatrix} D_k^{1/2} & 0 \\ -L_k D_k^{-1/2} & \bar{L}_k \end{bmatrix}\begin{bmatrix} -D_k^{1/2} & D_k^{-1/2} L_k^T \\ 0 & \bar{L}_k^T \end{bmatrix}, \tag{25}$$

where $\bar{L}_k$ is the lower triangular matrix that satisfies

$$\bar{L}_k \bar{L}_k^T = S_k^T B_{k,0} S_k + L_k D_k^{-1} L_k^T. \tag{26}$$

Note that if $B_{k,0}$ is positive definite and $s_i^T q_i > 0,\ i = k-t, \cdots, k-1$, then the right hand side of (26) is positive definite. Therefore, only the Choleski factorization of the $t \times t$ symmetric positive definite matrix $S_k^T B_{k,0} S_k + L_k D_k^{-1} L_k^T$ needs to be computed, to implement (23). This is preferable to factorizing the indefinite $2t \times 2t$ matrix H_k.

It is interesting to note that in our implementation, we do not need to calculate or store the matrix B_k explicitly, only the product $B_k v$ for an arbitrary vector v.

After the new iterate x_{k+1} is generated, we obtain S_{k+1} by deleting s_{k-t} from S_k and adding the new displacement s_k. The matrix Q_{k+1} is updated in the same fashion. This describes the general step when $k > t$. For the first few iterations, when $k \leq t$, we need only replace t by k in the formulae above. The first t search directions are the same as those given by a standard BFGS updateformula.

It is true that some computation and storage requirement of the above procedure for $B_k v$ and $v^T B_k v$ can be reduced if $B_{k,0}$ is sparse. A simple choice of $B_{k,0}$ is

$$B_{k,0} = \sigma_k I, \tag{27}$$

where σ_k is the Oren-Luenberger scaling factor $\sigma_k = \frac{s_{k-1}^T q_{k-1}}{s_{k-1}^T s_{k-1}}$. For $B_{k,0}$ is derived as follows: Note that at the solution x_* of (16), we have by the

Karush-Kuhn-Tucker condition

$$\lambda_* = -F(x_*). \tag{28}$$

Hence from the structure of the Hessian $\nabla^2_{\bar{x}} l(\bar{x}, \lambda)$ in (19), we choose $B_{k,0}$ by

$$B_{k,0} = \begin{pmatrix} I & 0 \\ 0 & \| F_k \| I \end{pmatrix}. \tag{29}$$

A more professional choice of $B_{k,0}$ seems to be

$$B_{k,0} = \begin{pmatrix} I & 0 \\ 0 & B^{(2,2)}_{k-1} \end{pmatrix}, \tag{30}$$

where $B^{(2,2)}_{k-1}$ is the last quasi-Newton approximation to the right bottom corner of $\tilde{S}(x)$ in (19) at x_{k-1}. However, in this case we have to compute and store $B^{(2,2)}_k$ at each step. In order to save some computation of $B^{(2,2)}_k$, one could update it every l iterations, where l is a preset parameter. By setting $l = \infty$ means that we updateit at each step. Our numerical tests show that $l = p$ performs well.

In our above discussion, the information saved in previous iterations is accumulated to construct the Hessian matrix for the current iteration. However, the information far away from the current point could affect the accuracy of the Hessian approximation if it is dominated by inappropriate terms. Hence a procedure, which is similar to the restart procedure in Powell [6] for unconstrained problems, is included.

3.1 The restart procedure

R1 at the k-th iteration,

$$| \nabla_{\bar{x}} l(\bar{x}_{k+1}, \lambda_{k+1})^T \nabla_{\bar{x}} l(\bar{x}_k, \lambda_k) | > 0.2 \parallel \nabla_{\bar{x}} l(\bar{x}_{k+1}, \lambda_{k+1}) \parallel_2^2 . \qquad (31)$$

R2 the number of consecutive iterations is larger than p;

If either of the above two conditions holds, then restarting is required. This means that we delete all stored pairs $\{s_i, q_i\}_{i=k-t}^{k-1}$, and that d_k should then be set to a scaled steepest-descent direction via the basic matrix $B_{k,0}$ in (29).

It is now easy to take extra nonlinear constraints into account for the above procedure. If nonlinear constraints $h(x) = 0$ are added to problem (1), where $h(x) : R^p \rightarrow R^m$ are continuous functions, then the following transformed problem is to be solved by the LSQP method:

$$\begin{aligned} &\min_{(z,x)} \tfrac{1}{2} z^T z \\ &\text{s.t. } z - F(x) = 0 \\ &\quad h(x) = 0. \end{aligned} \qquad (32)$$

4 Numerical Results

The proposed nonlinear least squares algorithm was implemented and tested on a SUN ULTRA SPARC 5 Workstation, using MATLAB 4.2. We have used

the 21 test problems. In our experiments

$$\max\{\| Z_k^T \nabla_{\bar{x}} \bar{f}(\bar{x}_k) \|_\infty, \| \bar{c}(\bar{x}_k) \|_\infty\} \leq 10^{-8}$$

is used as one of the termination criteria. If an iteration terminates after satisfying this criterion, the result is classified as successful, otherwise as a failure. Failures were also caused by excessive numbers of iterations (≥ 100) and excessive numbers of line search steps (≥ 10).

We found that LSQP performs better if we increase the storage requirement number t. However, if t increases beyond the number of variables p, only marginal improvement was observed. To test the choice of the basic matrix $B_{k,0}$ at each iteration, we have run the four choices (28), (29), (30) and a modification of (30) which updates $B_k^{(2,2)}$ every p iterations. Our numerical results show that the choices (29) and the modification of (30) are best in terms of function and Jacobian calls. It is interesting that both choices are cheap.

Compared with other methods, we found that the FSQP method usually requires fewer function and Jacobian calls. But for some problems the LSQP method approaches the performance of the FSQP method. In order to obtain a fair comparison with the RSQP method we specify $t = p$. Therefore, the amount of storage required by LSQP and RSQP are the same. The numerical tests show that the numbers of function and gradient evaluations required by LSQP are less than those of RSQP for test problems where both algorithms converged, and LSQP performs better than RSQP as far as efficiency is concerned. Moreover, we observe that LSQP has displayed a higher degree of

robustness. For the total number of 21 test problems, RSQP had more failure terminations than LSQP (11 versus 2). More details can be found in Li [5].

References

[1] L. T. Biegler, J. Nocedal, C. Schmid and D. Ternet. Numerical experience with a reduced Hessian method for large scale constrained optimization. *Comput. Optim. Appl.*, 15:45–67, 2000. C909, C910

[2] R. H. Byrd, J. Nocedal and R. B. Schnabel. Representation of Quasi-Newton Matrices and Their use in Limited Memory Methods. *Math. Prog.*, 63:129–156, 1994. C911

[3] J. E. Dennis, D. M. Gay and R. E. Welsch. An adaptive nonlinear least-sqaures algorithm. *Trans. Math. Software*, 7:348–368, 1981. C903

[4] R. Fletcher and C. Xu. Hybrid methods for nonlinear least squares. *IMA J. Numer. Anal.*, 7:371–389, 1987. *Mathematical Programming*, 12:241–254, 1977. C903

[5] Z. F. Li. Numerical Experiments with variations of the SQP method for nonlinear least squares. In preparation. C916

[6] M. J. D. Powell. Restarted procedures for the conjugate gradient method. *Mathematical Programming*, 12:241–254, 1977. C913

[7] M. J. D. Powell. A fast algorithm for nonlinearly constrained optimization calculation. In G.A. Watson, editor, *Numerical Analysis Proceedings*, Dundee, 1978. Springer-Verlag. C906

[8] M. R. Osborne. Nonlinear least sqaures-The Levenberg algorithm revisited. *J. Austral. math. Soc.(Series B)*, 19:343–357, 1976. C903

[9] K. Schittkowski. Solving Constrained Nonlinear Least Squares Problems by a General SQP-Method. In K.-H. Hoffmann, J.-B. Hiriart-Urruty, C. Lemarechal, and J. Zowe, editors, *Trends in Mathematical Optimization*, pages 295–309, 1988. International Series of Numerical Mathematics, Vol. 84, Birkhäuser. C901, C903, C907