A NEW LOW RANK QUASI-NEWTON UPDATE SCHEME FOR NONLINEAR PROGRAMMING

R. Fletcher,[1]
[1]*Department of Mathematics, University of Dundee, Dundee DD1 4HN, Scotland, UK, fletcher@maths.dundee.ac.uk*

Abstract

A new quasi-Newton scheme for updating a low rank positive semi-definite Hessian approximation is described, primarily for use in sequential quadratic programming methods for nonlinear programming. Where possible the symmetric rank one update formula is used, but when this is not possible a new rank two update is used, which is not in the Broyden family, although invariance under linear transformations of the variables is preserved. The representation provides a limited memory capability, and there is an ordering scheme which enables 'old' information to be deleted when the memory is full. Hereditary and conjugacy properties are preserved to the maximum extent when minimizing a quadratic function subject to linear constraints. Practical experience is described on small (and some larger) CUTE test problems, and is reasonably encouraging, although there is some evidence of slow convergence on large problems with large null spaces.

keywords: nonlinear programming, filter, SQP, quasi-Newton, symmetric rank one, limited memory.

1. Introduction

This work arises as part of a project to provide effective codes for finding a local solution $\mathbf{x}^*$ of a nonlinear programming (NLP) problem, which for convenience we express in the form

$$\begin{array}{ll} \underset{\mathbf{x}\in\mathbf{R}^n}{\text{minimize}} & f(\mathbf{x}) \\ \text{subject to} & c_i(\mathbf{x}) \geq 0 \quad i = 1, 2, \ldots, m, \end{array} \tag{1.1}$$

although in practice a more detailed formulation would be appropriate, admitting also equations, linear constraints and simple bounds. In particular we aim to develop a new trust-region filter SQP (sequential quadratic programming) code which only uses first derivatives of the problem functions $f(\mathbf{x})$ and $c_i(\mathbf{x})$.

Please use the following format when citing this chapter:
Author(s) [insert Last name, First-name initial(s)], 2006, in IFIP International Federation for Information Processing, Volume 199, System Modeling and Optimization, eds. Ceragioli F., Dontchev A., Furuta H., Marti K., Pandolfi L., (Boston: Springer), pp. [insert page numbers].

Filter methods for NLP were first introduced by Fletcher and Leyffer [6], and a production code `filterSQP` has been shown to be reliable and reasonably efficient. This code requires second derivatives of the problem functions to be made available by the user. The code has been hooked up to the AMPL modelling language, which includes a facility for automatically providing second derivatives, and is available for use under NEOS. More recently, convergence proofs for different types of filter method have been developed, and a code `filter2` has been written to implement the method considered in the paper of Fletcher, Leyffer and Toint [7]. This code also requires second derivatives to be made available. The practical performance of `filter2` is similar to that of `filterSQP`. An early version of the new quasi-Newton filter SQP code, referred to as `filterQN`, has already been tried on a range of problems with some success.

In view of the ready availability of second derivatives through the AMPL modelling language, one might question whether there is a need for NLP algorithms that use only first derivatives. To answer this, one should first point to the success of the NLP solver SNOPT (Gill, Murray and Saunders, [8]), based on an augmented Lagrangian formulation, which only requires first derivatives to be available. This is one of the most effective existing codes for NLP. Other reasons include the fact that Hessian matrices are often indefinite, which in an SQP context might render some QP solvers inapplicable. Even if the QP solver can handle indefinite matrices, there is usually no guarantee that a global (or even local) solution is found to the QP subproblems. (Although it has to be said that there is little evidence that this is a serious difficulty in practice.) Another argument is that NLP problems often have small or even empty null spaces, in which case only a small part of the Hessian is in a sense useful.

There are certain types of problem however in which Hessian calculations can be seriously time consuming and hence impracticable. Such an example is the optimal design of a Yagi-Uda antenna, shown to me by Martijn van Beurden (see [1] for details). The antenna is constructed from a number of wires along an axis, and there are two design variables (length and position along the axis) for each wire. Also there are 31 complex variables on each wire to model the current (62 real variables if complex arithmetic is not available). These variables satisfy a complex dense nonsingular system of linear equations. When modelled in AMPL, both the design and current variables appear explicity in the model, and the second derivative calculation is seriously time consuming. The largest problem that could be handled by AMPL via NEOS, with various solvers, had 5 wires, and hence 10 design variables and 310 real current variables. For this problem, filterSQP took about 2 hours to solve the problem whereas SNOPT, which only requires first derivatives, took about 15 minutes. For filterSQP, the memory usage was 610MB.

An much more effective procedure is not to use AMPL at all, and to use the complex linear equations to eliminate the complex variables, leaving a much smaller problem in just the design variables. Factors derived from the complex equations can be used efficiently to compute the gradient of the reduced problem, whereas computing the Hessian of the reduced problem remains very expensive. When posed in this way, various QN-SQP solvers, such as DONLP2, NPSOL and an early version of the filterQN code, were able to solve the 5-wire problem in around one minute. In fact even the 20-wire problem, with 40 design variables and 2480 real current variables could be solved in reasonable time.

This paper describes a new quasi-Newton scheme for updating a low rank positive semi-definite Hessian approximation, primarily for use in SQP methods for NLP. The paper is organised as follows. Section 2 reviews existing quasi-Newton methodology, and gives two results relating to hereditary conditions and quadratic termination for the symmetric rank one update, one of which may not be well known. Section 3 considers the implications for NLP, and describes the form UU^T of the representation. In Section 4, the interpretation as a limited memory approximation is discussed, and an it is shown how to update the representation so that the most recent information is contained in the leftmost columns of U. Section 5 focusses on how the projection part of the BFGS update might be implemented in this context, and Section 6 describes a new scheme which combines this update with the symmetric rank one update, for use when the latter alone is inapplicable. The outcome is a new rank two update which is not in the Broyden family, although invariance under linear transformations of the variables is preserved. The underlying motivation is seen to be the preservation of hereditary properties to the maximum extent. Conjugacy properties of the update in the quadratic case are brought out in Section 7 and a result somewhat akin to a hereditary property is shown to hold. Preliminary practical experience is described in Section 8 on small (and some larger) CUTE test problems, and is reasonably encouraging, although there is some evidence of slow convergence on large problems with large null spaces. Some conclusions are drawn in Section 9.

2. Quasi-Newton methodology

In this section we review existing quasi-Newton (QN) methodology in the context of uncontrained optimization ($m = 0$ in (1.1)). A QN method is based on updating symmetric matrices $B^{(k)}$ that approximate the Hessian matrix $\nabla^2 f$ of the objective function. These matrices are then used on iteration k of a Newton-like line search or trust region method. The initial matrix $B^{(1)}$ is arbitrary and is usually chosen to be positive definite, for example the unit

matrix. At the completion of iteration k of the QN method, difference vectors

$$\delta^{(k)} = \mathbf{x}^{(k+1)} - \mathbf{x}^{(k)} \tag{2.1}$$

in the variables, and

$$\gamma^{(k)} = \nabla f(\mathbf{x}^{(k+1)}) - \nabla f(\mathbf{x}^{(k)}) \tag{2.2}$$

in the gradients are available, and an updated matrix $B^{(k+1)}$ is computed, usually so as to satisfy the *secant condition*

$$B^{(k+1)}\delta^{(k)} = \gamma^{(k)} \tag{2.3}$$

which would be satisfied to first order by the true Hessian $\nabla^2 f(\mathbf{x}^{(k)})$.

There are many ways to satisfy (2.3), but there are two well known QN updating formulae which have featured in many applications. These are the *Symmetric Rank 1 (SR1)* formula

$$B^{(k+1)} = B + \frac{(\gamma - B\delta)(\gamma - B\delta)^T}{(\gamma - B\delta)^T\delta}, \tag{2.4}$$

suggested independently by various authors in 1968-69, and the *BFGS formula*

$$B^{(k+1)} = B - \frac{B\delta\delta^T B}{\delta^T B\delta} + \frac{\gamma\gamma^T}{\delta^T\gamma} \tag{2.5}$$

suggested independently by various authors in 1970. Superscript (k) has been suppressed on all vectors and matrices on the right hand sides of these formulae, and also elsewhere in the subsequent presentation, so as to avoid over-complicating the notation. More details and references may be found in Fletcher [5] for example.

An important property of the BFGS formula is that if $B^{(k)}$ is positive definite and $\delta^T\gamma > 0$, then $B^{(k+1)}$ is positive definite. Since $\nabla^2 f(\mathbf{x}^*)$ is positive semi-definite and usually positive definite, it is desirable that the approximating matrices $B^{(k)}$ also satisfy this property. It then follows that the Newton direction $-(B^{(k)})^{-1}\nabla f(x^{(k)})$ is a descent direction, and a line search along this direction enables $f(\mathbf{x})$ to be reduced. It is also possible to implement the line search in such a way that $\delta^T\gamma > 0$ always holds. Because of these properties, the BFGS formula has been the method of choice in most cases. On the other hand the denominator $(\gamma - B\delta)^T\delta$ in the SR1 formula may be negative, so that $B^{(k+1)}$ is not positive semi-definite, or even zero, in which case the formula breaks down. However the SR1 formula has been used, particularly in the context of trust region methods, with some safeguards. Indeed there is some evidence (Conn, Gould and Toint [4]) that the matrices $B^{(k)}$ converge more rapidly to $\nabla^2 f(\mathbf{x}^*)$ when the SR1 update is used.

Both formulae usually generate dense matrices $B^{(k)}$, even when the true Hessian $\nabla^2 f$ is sparse, and so are only suitable for solving small to medium size problems. Special purpose methods have been developed for solving large systems, for example the limited memory BFGS (L-BFGS) method (Nocedal, [9]), the sparse Hessian update scheme of Powell and Toint [11], and the use of the SR1 update for partially separable functions (Conn, Gould and Toint, [3]).

Another pointer to the effectiveness of a QN update, albeit somewhat indirect, is whether the property of *quadratic termination* can be proved. That is to say, can the associated QN method find the minimizer of a quadratic function in a finite number of steps. This property usually holds for the BFGS method only if exact line searches along the Newton direction are carried out. A stronger termination property holds for the SR1 method in which the differences $\boldsymbol{\delta}^{(k)}$ can be defined in an almost arbitrary manner. This may be a pointer to the effectiveness of the SR1 update in a trust region context. This result is established in the following well known theorem.

THEOREM 1 *Consider n SR1 updates using difference vectors $\boldsymbol{\delta}^{(k)}$ and $\boldsymbol{\gamma}^{(k)}$ for $k = 1, 2, \ldots, n$, where $\boldsymbol{\gamma}^{(k)} = W\boldsymbol{\delta}^{(k)}$ and W is symmetric. If $B^{(1)}$ is symmetric, and if for $k = 1, 2, \ldots, n$ the denominators in (2.4) are non-zero, and the vectors $\boldsymbol{\delta}^{(k)}$ are linearly independent, then $B^{(n+1)} = W$.*

Proof Clearly the SR1 update preserves the symmetry of the matrices $B^{(k)}$. It is shown by induction that

$$B^{(k)}\boldsymbol{\delta}^{(j)} = \boldsymbol{\gamma}^{(j)} \qquad j = 1, 2, \ldots, k-1, \tag{2.6}$$

where $1 \leq k \leq n+1$. For $k = 1$ the condition is vacuous and hence true. Now let it be true for some k such that $1 \leq k \leq n$. The definition of $B^{(k+1)}$ gives

$$B^{(k+1)}\boldsymbol{\delta}^{(j)} = B^{(k)}\boldsymbol{\delta}^{(j)} + \frac{\mathbf{u}^{(k)}\mathbf{u}^{(k)T}\boldsymbol{\delta}^{(j)}}{\mathbf{u}^{(k)T}\boldsymbol{\delta}^{(k)}} \tag{2.7}$$

where $\mathbf{u}^{(k)} = \boldsymbol{\gamma}^{(k)} - B^{(k)}\boldsymbol{\delta}^{(k)}$. For $j = k$ the right hand side is $B^{(k)}\boldsymbol{\delta}^{(k)} + \mathbf{u}^{(k)}$ which is equal to $\boldsymbol{\gamma}^{(k)}$ by definition of $\mathbf{u}^{(k)}$. For $j < k$ it follows from (2.6) that $B^{(k)}\boldsymbol{\delta}^{(j)} = \boldsymbol{\gamma}^{(j)}$, and also using the definition of $\mathbf{u}^{(k)}$ and symmetry of $B^{(k)}$ that

$$\mathbf{u}^{(k)T}\boldsymbol{\delta}^{(j)} = (\boldsymbol{\gamma}^{(k)} - B^{(k)}\boldsymbol{\delta}^{(k)})^T\boldsymbol{\delta}^{(j)} = \boldsymbol{\gamma}^{(k)T}\boldsymbol{\delta}^{(j)} - \boldsymbol{\delta}^{(k)T}\boldsymbol{\gamma}^{(j)}.$$

Because $\boldsymbol{\gamma}^{(j)} = W\boldsymbol{\delta}^{(j)}$ for all $j = 1, 2, \ldots, n$ it follows for $j < k$ that $\mathbf{u}^{(k)T}\boldsymbol{\gamma}^{(j)} = 0$. Thus for both $j = k$ and $j < k$ it has been shown that $B^{(k+1)}\boldsymbol{\delta}^{(j)} = \boldsymbol{\gamma}^{(j)}$ and hence (2.6) has been established with $k+1$ replacing k. Hence by induction, (2.6) is true for all $k = 1, 2, \ldots, n+1$.

For $k = n + 1$, and using $\gamma^{(j)} = W\delta^{(j)}$, (2.6) can be written as

$$B^{(n+1)}\delta^{(j)} = W\delta^{(j)} \qquad j = 1, 2, \ldots, n, \tag{2.8}$$

or as

$$B^{(n+1)}\Delta = W\Delta, \tag{2.9}$$

where Δ is an $n \times n$ matrix with columns $\delta^{(j)}$, $j = 1, 2, \ldots, n$. But Δ is nonsingular by the linear independence assumption, so it follows that $B^{(n+1)} = W$. **QED**

A consequence of the therorem is that if the SR1 method is applied to minimize a quadratic function with positive definite Hessian W, then a Newton iteration on iteration $n + 1$ will locate the minimizer exactly. A key feature of the proof is the establishment of so-called *hereditary conditions* (2.6), in which secant conditions (2.3) from previous iterations remain satisfied by subsequent $B^{(k)}$ matrices. In other words, when the correct behaviour $B^{(k+1)}\delta^{(k)} = \gamma^{(k)} = W\delta^{(k)}$ is introduced, it persists in subsequent $B^{(k)}$ matrices.

A less well known result in the quadratic case is that if W is positive definite, and $B^{(1)} = 0$ is chosen, then the denominators in the SR1 update are all positive, and the matrices $B^{(k)}$ are positive semi-definite.

THEOREM 2 *Consider n SR1 updates using difference vectors $\delta^{(k)}$ and $\gamma^{(k)}$ for $k = 1, 2, \ldots, n$, where $\gamma^{(k)} = W\delta^{(k)}$ and W is symmetric positive definite. If $B^{(1)} = 0$, and the vectors $\delta^{(k)}$ $k = 1, 2, \ldots, n$ are linearly independent, then the SR1 updates are well defined, the matrices $B^{(k)}$ are positive semi-definite of rank $k - 1$, and $B^{(n+1)} = W$.*

Proof Without loss of generality we can take $W = I$ since the SR1 update is independent under linear transformations ([5], Theorem 3.3.1). It is shown by induction that

$$B^{(k)}\delta^{(j)} = \delta^{(j)} \qquad j = 1, 2, \ldots, k - 1, \tag{2.10}$$

and

$$B^{(k)}\mathbf{v} = 0 \quad \forall\, \mathbf{v} \in \{\mathbf{v} \mid \mathbf{v}^T\delta^{(j)} = 0,\ j = 1, 2, \ldots, k - 1\}, \tag{2.11}$$

so that $B^{(k)}$ is an orthogonal projector of rank $k - 1$. This is true for $k = 1$ because $B^{(1)} = 0$. Because $\gamma^{(k)} = \delta^{(k)}$, (2.4) may be written

$$B^{(k+1)} = B + \frac{(I - B)\delta\delta^T(I - B)}{\delta^T(I - B)\delta}. \tag{2.12}$$

Because the $\delta^{(k)}$ are linearly independent, it follows that the denominator in (2.12) is positive. As in Theorem 1 $B^{(k+1)}\delta^{(k)} = \delta^{(k)}$, and for $j < k$ it follows using (2.10) that $(I - B^{(k)})\delta^{(j)} = 0$ and hence $B^{(k+1)}\delta^{(j)} = \delta^{(j)}$. Also the rank one correction in (2.12) is in $\text{span}\{\delta^{(1)}, \delta^{(2)}, \ldots, \delta^{(k)}\}$, so (2.11) follows for $B^{(k+1)}$. Thus the inductive step has been established. The rest of the theorem follows as for Theorem 1. **QED**

In a non-quadratic context, this theorem suggests that if $B^{(1)} = 0$ is chosen, then there is less likelihood that the SR1 update will break down, or give rise to an indefinite matrix.

3. Quasi-Newton updates in NLP

This section looks at the new issues that arise when QN updates are used in the context of an NLP calculation, particularly when n is large. In this case it is impracticable to update a full dense Hessian. However, it is only the reduced Hessian that needs to be positive semi-definite at a solution, and local and superlinear convergence can be achieved without updating a full Hessian. A low rank Hessian approximation is presented which allows rapid local convergence of SQP, whilst requiring much less storage to implement.

When the NLP problem has nonlinear constraints, it is the Hessian $W = \nabla^2\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}^*)$ of a Lagrangian function $\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}^*) = f(\mathbf{x}) - \mathbf{c}(\mathbf{x})^T\boldsymbol{\lambda}^*$ that determines the local convergence properties of an SQP method, where $\boldsymbol{\lambda}^*$ is the vector of KT multipliers at the solution (see, for example [5]). In this case $\gamma^{(k)}$ should be computed from differences in the Lagrangian gradients, and Nocedal and Overton [10] recommend

$$\gamma^{(k)} = \nabla\mathcal{L}(\mathbf{x}^{(k+1)}, \boldsymbol{\lambda}^{(k+1)}) - \nabla\mathcal{L}(\mathbf{x}^{(k)}, \boldsymbol{\lambda}^{(k+1)}) \tag{3.1}$$

as an effective choice (amongst others), where $\boldsymbol{\lambda}^{(k+1)}$ is the most recently available estimate of the KT multipliers.

In general, the Lagrangian Hessian matrix $W^* = \nabla^2\mathcal{L}(\mathbf{x}^*, \boldsymbol{\lambda}^*)$ at the solution may not be positive semi-definite, in contrast to the unconstrained case. Only the $d \times d$ reduced Hessian matrix Z^TW^*Z is positive semi-definite (and usually positive definite), where columns of the matrix Z are a basis for the null space $\mathcal{N}^* = \{\mathbf{z} \mid A^{*T}\mathbf{z} = \mathbf{0}\}$, where A^* denotes the matrix of active constraint gradients at the solution (see [5] for example). Quite often the dimension of $\mathcal{N}^*$ much smaller than n. A related consequence of this is that the denominator $\delta^T\gamma$ that arises in the BFGS method cannot be assumed to be positive, again in contrast to the unconstrained case.

Sufficient conditions for the Q-superlinear convergence of SQP methods under mild assumptions are that

$$B^{(k)}Z \sim W^*Z, \tag{3.2}$$

(see Boggs, Tolle and Wang [2]). That is to say, $B^{(k)}$ should map the null space correctly in the limit, but $B^{(k)} \sim W^*$ is not necessary. In this paper we aim to achieve something akin to (3.2) based on the quadratic termination properties of the SR1 update.

Quadratic termination for an NLP solver relates to how the solver performs when applied to solve the equality constrained QP problem

$$\begin{array}{ll} \underset{\mathbf{x}\in\mathbb{R}^n}{\text{minimize}} & q(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T W\mathbf{x} + \mathbf{c}^T\mathbf{x} \\ \text{subject to} & A^T\mathbf{x} = \mathbf{b}, \end{array} \tag{3.3}$$

where $A \in \mathbb{R}^{n\times m}$, $m < n$, and $\text{rank}(A) = m$. We let $Z \in \mathbb{R}^{n\times d}$ be a matrix whose columns are a basis for the null space $\mathcal{N}(A^T) = \{\mathbf{z} \mid A^T\mathbf{z} = \mathbf{0}\}$ of dimension $d = n - m$. The QP problem (3.3) has a unique solution if and only if the reduced Hessian $Z^T W Z$ is positive definite. In this case, if

$$B^{(k)}Z = WZ, \tag{3.4}$$

if $\mathbf{x}^{(k)}$ is a feasible point, and if iteration k is an SQP iteration

$$Z^T B^{(k)} Z\mathbf{t}^{(k)} = -Z^T\nabla f(x^{(k)}) \tag{3.5}$$

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + Z\mathbf{t}^{(k)}, \tag{3.6}$$

then $\mathbf{x}^{(k+1)}$ solves (3.3). If the SR1 update is used, and if d consecutive and linearly independent steps $\boldsymbol{\delta}^{(k-1)}$, $\boldsymbol{\delta}^{(k-2)}, \ldots, \boldsymbol{\delta}^{(k-d)}$ in $\mathcal{N}(A^T)$ can be completed, then these vectors can form the columns of Z, and (3.4) follows by the hereditary properties of the SR1 update. Thus quadratic termination is obtained under these conditions.

These results do not require W to be positive definite. However, if necessary a related QP problem with the same solution can be defined by adding a squared penalty $\frac{1}{2}\sigma(A^T\mathbf{x}-\mathbf{b})^T(A^T\mathbf{x}-\mathbf{b})$ into $q(\mathbf{x})$. If $Z^T W Z$ is positive definite and σ is sufficiently large, then the Hessian $W + \sigma AA^T$ of the modified objective function is positive definite.

Quadratic termination for an inequality constraint QP problem is less obvious, because it is uncertain what will happen to $B^{(k)}$ before the correct active set is located. However tests with `filterQN` on some inequality QP problems from the CUTE test set did exhibit termination.

In this paper we represent $B^{(k)}$ by the low rank positive semi-definite approximation

$$B^{(k)} = U^{(k)}U^{(k)T}, \tag{3.7}$$

where $U^{(k)}$ is a dense $n \times r$ matrix. Usually $U^{(k)}$ has rank r but the current implementation does not guarantee that this is so. Clearly $B^{(k)}$ is positive semi-definite and has the same rank as $U^{(k)}$. Of course only $U^{(k)}$ is stored, and $B^{(k)}$

is recovered implicitly from (3.7). We shall use the SR1 formula to update $U^{(k)}$ whenever possible. When this is not possible, we shall arrange matters so as to retain as many of the most recent hereditary properties as possible in $B^{(k)}$. By this means we hope that, once the correct active set is located by the NLP solver, we shall then build up hereditary properties in the correct null space, and hence obtain rapid convergence.

In using (3.7), it follows from the remarks in Section 2 that there may be some advantage to be gained by initializing $B^{(1)} = 0$. We do this simply by setting $r = 0$. In general, a trust region constraint $\|\delta\| \leq \rho$ will ensure that the SQP subproblem is bounded, so that no difficulty arises from the rank deficiency of $B^{(k)}$.

In passing, we note that even less storage is needed if an approximating matrix $M^{(k)} \sim Z^T W Z$ is used, and $B^{(k)} = Z^{(k)} M^{(k)} Z^{(k)T}$, where $Z^{(k)}$ is a current approximation to Z^*, obtained from the current QP subproblem. However, the active set in the QP subproblem can change considerably from iteration to iteration, and it is not easy to suggest a robust strategy for updating $M^{(k)}$.

4. Updating the representation $B^{(k)} = U^{(k)} U^{(k)T}$

In this section we consider some issues relating the use of an update formula in conjunction with (3.7). The SR1 update is seen to be most suitable, if it is applicable. It is shown how to implement the update so as order the columns of U in such a way that the most recent information is contained in the leftmost columns. This provides a useful limited memory capability.

The SR1 update (2.4) can only be used to update $U^{(k)}$ if $(\gamma - B\delta)^T \delta > 0$. In this case, one way to update $U^{(k)}$ would be simply to append the column vector

$$\mathbf{u} = \frac{\gamma - B\delta}{((\gamma - B\delta)^T \delta)^{1/2}} \tag{4.1}$$

to $U^{(k)}$. Unless $\mathbf{u}$ is in the range space of $U^{(k)}$, we would obtain a matrix $U^{(k+1)}$ with rank $r^{(k+1)} = r^{(k)} + 1$. Thus the SR1 update provides a way of building up information in U and is used whenever possible. For the BFGS update, the first two terms on the right hand side of (2.5) perform a projection operation which usually reduces the rank of B by one. The final rank one term restores the rank, so that usually $r^{(k+1)} = r^{(k)}$. Thus the BFGS update is unable to build up information in U.

The low rank representation (3.7) gives the method the flavour of a limited memory method, and indeed we shall introduce a *memory limit*

$$r \leq r_{max}. \tag{4.2}$$

Ideally r_{max} should be greater or equal to the dimension d of the null space at the solution. In this case we would hope for local superlinear convergence of our SQP method. When $r_{max} < d$ only linear convergence can be expected, and it is not clear how slow this might be. However some hope might be derived from the fact that methods such as conjugate gradients and some new gradient methods are able to solve instances of very large problems in relatively few iterations.

When the memory is full, the SR1 update would usually cause $r^{(k+1)}$ to be greater than r_{max}, so that we are faced with the need to delete a column of $U^{(k+1)}$ from the memory. We would like to ensure that when this occurs, it is the *oldest* information that is deleted, in a certain sense. We can exploit the fact that the representation (3.7) is not unique, to the extent that

$$B = UU^T = UQQ^TU^T = (UQ)(UQ)^T$$

where Q is any orthogonal matrix. We shall choose Q in such a way that the most recent information is contained in the *leftmost* columns of U, when the SR1 update is being used. Then we delete the rightmost column of U when the memory is overfull. We shall refer to this as the *priority ordering* of U.

We therefore express

$$U^{(k+1)} = [\,\mathbf{u} \quad U\,]\,Q \tag{4.3}$$

(suppressing superscript (k)), in which

$$\mathbf{u} = \frac{\gamma - B\delta}{((\gamma - B\delta)^T\delta)^{1/2}} = \frac{\gamma - U\mathbf{v}}{\alpha}, \tag{4.4}$$

where $\mathbf{v} = U^T\delta$ and $\alpha = (\delta^T\gamma - \mathbf{v}^T\mathbf{v})^{1/2}$. Using (4.4) we may write

$$[\,\mathbf{u} \quad U\,] = [\,\gamma \quad U\,] \begin{bmatrix} 1/\alpha & & & & \\ -v_1/\alpha & 1 & & & \\ -v_2/\alpha & & 1 & & \\ \vdots & & & \ddots & \\ -v_r/\alpha & & & & 1 \end{bmatrix}. \tag{4.5}$$

Next we implicitly transform the spike matrix to an upper triangular matrix, R say, by postmultiplying by a product of plane rotation matrices in columns $(1,\, j)$ for $j = r+1, r, \ldots, 2$, each rotation being chosen so as to eliminate the entry in row j of column 1. These rotations are explicitly applied to the columns of the left hand side matrix in (4.5).

Denoting the product of plane rotations by Q, the resulting matrix may be expressed as

$$U^{(k+1)} = [\,\mathbf{u} \quad U\,]\,Q = [\,\gamma \quad U\,]\,R. \tag{4.6}$$

It follows by a simple induction argument that

- column 1 of $U^{(k+1)}$ depends only on $\gamma^{(k)}$
- column 2 of $U^{(k+1)}$ depends only on $\gamma^{(k)}, \gamma^{(k-1)}$
- column 3 of $U^{(k+1)}$ depends only on $\gamma^{(k)}, \gamma^{(k-1)}, \gamma^{(k-2)}$

etc., over the range of previous iterations on which SR1 updates have been used, by virtue of R being upper triangular.

We refer to this calculation as

$$U^{(k+1)} = \mathrm{sr1}(U, \delta, \gamma). \tag{4.7}$$

Its cost is $O(nr)$ arithmetic operations, which is the same order of magnitude as the cost of a matrix product with U or U^T, and hence is readily affordable.

It is possible that (4.7) may give rise to a matrix $U^{(k+1)}$ whose columns are rank deficient. An example is given by

$$U = \begin{bmatrix} 1 & 1 \\ 1 & 0 \\ 2 & 0 \end{bmatrix} \qquad \delta = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \qquad \gamma = \begin{pmatrix} 3 \\ 2 \\ 4 \end{pmatrix}. \tag{4.8}$$

It is clear that γ is in the range of U, and $\alpha = 1$, so that the SR1 update does not break down. The matrix $U^{(k+1)}$ has 3 non-trivial columns but has rank 2. At present, there is no evidence to suggest that this possibility is causing any practical disadvantages, although if it were so, it would not be difficult to suggest modifications to ensure that $U^{(k)}$ always has full rank. Indeed, it does seem more appropriate in these circumstances to use an update that keeps the same the same number of columns in U, such as is described in Section 6 below.

5. The BFGS projection update

Although we have argued that the SR1 update will often be well defined, this will not always be so. Thus we have to decide how to update $U^{(k)}$ when the SR1 denominator is non-positive.

An extreme case is when $\delta^T \gamma \leq 0$. In the quadratic case (3.3), this suggests that δ is not in the null space spanned by columns of Z, so that γ provides no useful information for updating B. Now the curvature estimate of the current B matrix along δ is $\delta^T B \delta$. Thus, if $\delta^T B \delta > 0$, the curvature estimate is seen to be incorrect, and we use the part of the BFGS update

$$B^{(k+1)} = B - \frac{B\delta\delta^T B}{\delta^T B \delta}. \tag{5.1}$$

which projects out the existing information along δ, and has the correct invariance properties. This update reduces the rank of B by one. If $r = 0$ or $B\delta = \mathbf{0}$

we just set $B^{(k+1)} = B$. We implement (5.1) in such a way as to reduce the number of columns in U by one, which ensures that $U^{(k+1)}$ has full rank if $U^{(k)}$ has. We may express

$$B^{(k+1)} = U\left(I - \frac{\mathbf{v}\mathbf{v}^T}{\mathbf{v}^T\mathbf{v}}\right)U^T = UQ\left(I - \frac{Q^T\mathbf{v}\mathbf{v}^TQ}{\mathbf{v}^T\mathbf{v}}\right)Q^TU^T \qquad (5.2)$$

where $\mathbf{v} = U^T\boldsymbol{\delta}$ and $\mathbf{v}^TQ$ is an orthogonal transformation with a product of plane rotation matrices in columns (j, r), $j = 1, 2, \ldots, r-1$, so as eliminate successive elements v_j of $\mathbf{v}^T$. That is to say, $\mathbf{v}^TQ = \pm\|\mathbf{v}\|_2\mathbf{e}_r^T$ where $\mathbf{e}_r^T = (0, \ldots, 0, 1)$. Then

$$B^{(k+1)} = UQ\left(I - \mathbf{e}_r\mathbf{e}_r^T\right)Q^TU^T. \qquad (5.3)$$

Thus to update U we apply the same rotations to the columns of U, and then delete the last column of the resulting matrix. We may write this as

$$U^{(k+1)} = U\bar{Q} \qquad \text{where} \qquad \bar{Q} = Q\begin{bmatrix} I \\ \mathbf{0}^T \end{bmatrix}. \qquad (5.4)$$

It follows that $\boldsymbol{\delta}^TU^{(k+1)} = \mathbf{v}^T\bar{Q} = \mathbf{0}^T$, reflecting the fact that $B^{(k+1)}\boldsymbol{\delta} = \mathbf{0}$. We refer to the entire projection update as

$$U^{(k+1)} = \text{proj}(U, \boldsymbol{\delta}). \qquad (5.5)$$

As for (4.7), the cost is $O(nr)$ arithmetic operations. We observe that (5.5) destroys any priority ordering properties, and in the quadratic case, hereditiary properties.

6. A new QN update

We have seen that the SR1 update can be used when $\boldsymbol{\delta}^T\boldsymbol{\gamma} > \mathbf{v}^T\mathbf{v}$, and the projection update when $\boldsymbol{\delta}^T\boldsymbol{\gamma} \leq 0$. When $0 < \boldsymbol{\delta}^T\boldsymbol{\gamma} \leq \mathbf{v}^T\mathbf{v}$, there would appear to be useful information in $\boldsymbol{\delta}$ and $\boldsymbol{\gamma}$, but the SR1 update can no longer be used. In this section we present a new update which maximizes the extent to which hereditary properties, built up using the SR1 update on previous iterations, are preserved.

To do this we partition

$$U^{(k)} = [\,U_1 \quad U_2\,], \quad \mathbf{v} = \begin{pmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \end{pmatrix}, \quad B = B_1 + B_2 = U_1U_1^T + U_2U_2^T \qquad (6.1)$$

where U_1 has $r_1 \geq 0$ columns and U_2 has $r_2 \geq 0$ columns. The new QN update applies the SR1 update to U_1 and the projection update to U_2, and $U^{(k+1)}$ is obtained by concatenating the resulting matrices, that is

$$U^{(k+1)} = [\,\text{sr1}(U_1, \boldsymbol{\delta}, \boldsymbol{\gamma}) \quad \text{proj}(U_2, \boldsymbol{\delta})\,]. \qquad (6.2)$$

By choosing r_1 sufficiently small, it is always possible to ensure that the SR1 denominator $\alpha = (\delta^T\gamma - \mathbf{v}_1^T\mathbf{v}_1)^{1/2}$ in (6.2) exists and is positive. We observe that (6.2) usually leaves the rank of U unchanged.

In our current implementation we have chosen r_1 to be the largest integer for which $\delta^T\gamma - \mathbf{v}_1^T\mathbf{v}_1 > 0$. This choice maximizes the extent to which priority ordering and, in the quadratic case, hereditary conditions in U_1 are preserved in $U^{(k+1)}$. In fact it may well be better to require $\delta^T\gamma - \mathbf{v}_1^T\mathbf{v}_1 \geq \tau$ where $\tau > 0$ is some tolerance, for example $\tau = \varepsilon\delta^T\gamma$ with $\varepsilon = 10^{-6}$ say.

The new update may also be expressed in the form

$$B^{(k+1)} = B_1 + \frac{(\gamma - B_1\delta)(\gamma - B_1\delta)^T}{(\gamma - B_1\delta)^T\delta} + B_2 - \frac{B_2\delta\delta^T B_2}{\delta^T B_2\delta}. \tag{6.3}$$

If $r_1 = 0$ we get the BFGS update (2.5), and if $r_1 = r$ then the SR1 update (2.4). Intermediate values give a new rank two correction formula, but not one that is in the Broyden class. We observe the following properties

- Satisfies the secant condition (2.3).
- Preserves positive semi-definite $B^{(k)}$ matrices.
- Is invariant under a linear transformation of variables (see [5], Section 3.3).
- Any priority ordering or hereditary conditions in U_1 are preserved.

To summarize, if $\delta^T\gamma \leq 0$, then the BFGS projection update is used and the rank of U decreases by one (usually). If $0 < \delta^T\gamma \leq \mathbf{v}^T\mathbf{v}$, then the new update is used, choosing r_1 as described, and the rank of U is unchanged. The choice $r_1 = 0$ gives rise to a BFGS update. If $\delta^T\gamma > \mathbf{v}^T\mathbf{v}$ then an SR1 update is used (the rank of U usually increasing by one), except in the case that $r = r_{max}$ and the memory is full. In this case we apply the SR1 update and then delete column $r_{max} + 1$ of the resulting matrix. We note that this procedure still preserves the secant condition (2.3). This follows from (4.6) and (4.4) by virtue of

$$\delta^T U^{(k+1)} = \delta^T [\,\mathbf{u} \quad U\,] Q = [\,\alpha \quad \mathbf{v}^T\,] Q = (\eta, 0, \ldots, 0)$$

where $\eta^2 = \delta^T\gamma$, by virtue of the way in which Q is chosen. Since the last element on the right hand side is zero, the secant condition $\delta^T U^{(k+1)} U^{(k+1)T} = \gamma^T$ is unaffected by deleting the last column of $U^{(k+1)}$. We also observe that any priority ordering in U is preserved, and any hereditary conditions up to a maximum of $r - 1$.

7. Conjugacy conditions

The use of the BFGS projection operation can destroy hereditary properties that have been built up in the U_2 matrix, in the quadratic case. In this section we

show that this is not as unfavourable as it might seem, and that something akin to a hereditary property holds, even when the BFGS projection operation is used to update U_2. Also some conjugacy properties of the new update are shown. The results of this section apply when $B^{(1)} = 0$ and we are investigating the quadratic case in which the relationship $\gamma^{(k)} = W\delta^{(k)}$ holds for all k, where W is a fixed matrix. We shall also assume that W is nonsingular, which is a minor requirement that can always be achieved if necessary with an arbitrarily small perturbation using a quadratic penalty (see the paragraph following (3.6)).

When $B^{(1)} = 0$, it follows easily for both (5.5) and (6.2) by induction that the columns of $U^{(k)}$ are in $\text{span}(\gamma^{(1)}, \gamma^{(2)}, \ldots, \gamma^{(k-1)})$. It follows that U has an image

$$\Delta = W^{-1}U \tag{7.1}$$

whose columns are correspondingly in $\text{span}(\delta^{(1)}, \delta^{(2)}, \ldots, \delta^{(k-1)})$. In Theorem 3 below it is proved that normalised conjugacy coditions

$$U^T W^{-1} U = I \tag{7.2}$$

are satisfied by U (that is, $U^{(k)}$) for all k, a consequence of which is that $U^T\Delta = I$. Likewise, conjugacy conditions

$$\Delta^T W \Delta = I \tag{7.3}$$

are satisfied by Δ. A consequence is that

$$W\Delta = U = UU^T W^{-1} U = UU^T\Delta = B\Delta. \tag{7.4}$$

Hence B maps the subspace $\text{range}(\Delta)$ in the 'correct' way, that is $B\Delta = W\Delta = U$.

The implication of this in the quadratic case is that although the use of some projection operations may destroy hereditary conditions, it does not introduce any 'wrong' information. That is to to say, the image Δ of U has columns in $\text{span}(\delta^{(1)}, \delta^{(2)}, \ldots, \delta^{(k-1)})$, and the current matrix $B = UU^T$ maps Δ correctly into U (which is $W\Delta$). Thus, although hereditary conditions may not be satisfied, the exists a set of directions (columns of Δ) which satisfy a similar condition $B\Delta = U$ to hereditary conditions.

We now prove the theorem on which (7.2) depends.

THEOREM 3 *Let $B^{(1)} = 0$, and let there exist a nonsingular symmetric matrix W such that the difference vectors are related by $\gamma^{(k)} = W\delta^{(k)}$ for all k. Then conjugacy conditions (7.2) are preserved by the updating scheme described in Section 6.*

Proof We prove the result by induction. The result is trivially true when $k = 1$. Now we assume that (7.2) is true for some value of $k \geq 1$ and consider the calculation of $U^{(k+1)}$. If $\delta^T\gamma > 0$, then $U^{(k+1)}$ is defined by (6.2), which may be expressed as

$$U^{(k+1)} = [\,\mathbf{u} \quad U_1 \quad U_2\,] \begin{bmatrix} Q_1 & \\ & \bar{Q}_2 \end{bmatrix}, \tag{7.5}$$

using the notation of (4.6) and (5.4) with subscripts 1 and 2 to indicate matrices derived from the SR1 and projection updates respectively. We note that Q_1 has $r_1 + 1$ rows and columns, and $\bar{Q}_2$ has r_2 rows and $r_2 - 1$ columns. It follows that

$$\begin{aligned} &U^{(k+1)T}W^{-1}U^{(k+1)} \\ &= \begin{bmatrix} Q_1^T & \\ & \bar{Q}_2^T \end{bmatrix} \begin{bmatrix} \mathbf{u}^TW^{-1}\mathbf{u} & \mathbf{u}^TW^{-1}U \\ U^TW^{-1}\mathbf{u} & I \end{bmatrix} \begin{bmatrix} Q_1 & \\ & \bar{Q}_2 \end{bmatrix}, \end{aligned} \tag{7.6}$$

where $U^TW^{-1}U = I$ has been substituted from the inductive hypothesis. It now follows from (4.4) and $W^{-1}\gamma = \delta$ that

$$\begin{aligned} U^TW^{-1}\mathbf{u} &= U^TW^{-1}\left(\frac{\gamma - B_1\delta}{\alpha}\right) \\ &= \begin{pmatrix} U_1^T \\ U_2^T \end{pmatrix} \left(\frac{\delta - W^{-1}U_1U_1^T\delta}{\alpha}\right) = \begin{pmatrix} \mathbf{0} \\ U_2^T\delta/\alpha \end{pmatrix} \end{aligned} \tag{7.7}$$

using $U_1^TW^{-1}U_1 = I$ and $U_2^TW^{-1}U_1 = 0$ from the inductive hypothesis. Also from (4.4)

$$\mathbf{u}^TW^{-1}\mathbf{u} = \frac{(\gamma - B_1\delta)^TW^{-1}(\gamma - B_1\delta)}{\alpha^2} = 1 - \frac{(\gamma - B_1\delta)^TW^{-1}B_1\delta}{\alpha^2}$$

from $\delta = W^{-1}\gamma$ and the definition of α. But

$$\begin{aligned} (\gamma - B_1\delta)^TW^{-1}B_1\delta &= \delta^T((W - U_1U_1^T)W^{-1}U_1U_1^T\delta \\ &= \delta^T(U_1U_1^T - U_1U_1^TW^{-1}U_1U_1^T)\delta = 0 \end{aligned}$$

from $U_1^TW^{-1}U_1 = I$. Hence $\mathbf{u}^TW^{-1}\mathbf{u} = 1$. Finally we substitute this and (7.7) into (7.6). Then using $Q_1^TQ_1 = I$, $\bar{Q}_2^T\bar{Q}_2 = I$ and $\bar{Q}_2^TU_2^T\delta = \mathbf{0}$ (from (5.4)), it follows that $U^{(k+1)T}W^{-1}U^{(k+1)} = I$ and establishes that the inductive hypothesis holds for $k + 1$ in this case.

In the case that $\delta^T\gamma \leq 0$, only the 2,2 partition of the above argument is used and the result follows similarly. In the case that the memory is filled, and a column of $U^{(k+1)}$ is deleted, it is clear that $U^{(k+1)T}W^{-1}U^{(k+1)} = I$ will continue to hold, but for a unit matrix with one fewer row and column. Thus the result is established in all cases used in the update scheme. **QED**

8. Practical experience

In this section, some preliminary practical experience with the new update scheme is described. An experimental code `filterQN` is currently under development, and indeed has been so for some time. It is a trust region filter SQP code, based on the method considered in [7], but using the quasi-Newton update scheme described in this paper, rather than the exact second derivatives as used in the `filter2` code, referred to in Section 1. The delay in finalizing the code is mainly due to uncertainty as to how best to implement feasibility restoration when second derivatives are not available.

The results in this section are sampled from CUTE test problems in which the dimension d of the null space at the solution is a significant proportion of n. For such problems, feasibility restoration often plays a minor role in determining the outcome of the calculation. Thus, although the results cannot yet be taken as definitive, they do give some indication as to what level of performance can be expected from a QN code. The problems are solved to an accuracy of better than 10^{-6} in the KT conditions. The memory limit is $r_{max} = \min(n, 100)$.

Table 1. Performance of `filterQN` on small CUTE problems

	n	m	d	$\#g$	$f2$
HS90	4	1	3	29	6*
HS92	6	1	5	33	6*
HS99	7	2	5	11	6
HS100	7	4	5	14	14
HS100LNP	7	2	5	15	18
HS100MOD	7	4	6	18	13
HS101	7	5	5	230	21
HS102	7	5	4	209	18
HS103	7	5	3	28	25
HS111	10	3	6	45	25
HS111LNP	10	3	6	45	25
HS113	10	8	4	13	6
HS117	10	5	4	19	21
CANTILVR	5	1	4	25	17
DIPIGRI	7	4	5	14	14
ERRINBAR	18	9	3	70	25
MISTAKE	9	13	5	17	14
POLAK3	12	10	9	58	37
ROBOT	14	2	5	19	11
TENBARS1	18	9	5	59	28
TENBARS2	18	8	4	36	28
TENBARS3	18	8	4	76	28
TENBARS4	18	9	5	82	29

* `filter2` finds a locally infeasible point

Table 1 gives results on Hock-Schittkowski test problems in the left hand column, and other small CUTE test problems in the right hand column. Headings give the number of variables n, the number of constraints m (excluding simple bounds), the dimension d of the null space at the solution, the number of gradient calls $\#g$ required by `filterQN`, and the number of gradient calls $f2$ required by `filter2`. One gradient call includes the evaluation of both the gradient of the objective function and the Jacobian of the vector of constraint functions. In the case of `filter2`, it also includes the evaluation of all second derivatives. Both codes require about one QP subproblem to be solved for each gradient evaluation, most often in warm start mode.

Generally the problems are solved reliably and accurately by `filterQN`, and we see rapid local convergence. HS90 and HS92 are successfully solved by `filterQN`, whereas `filter2` can only find a locally infeasible point. For HS101 and HS102, `filterQN` spends about 200 and 180 iterations respectively in feasibility restoration, which accounts for the poor performance. Future work on the feasibility restoration algorithm should resolve this difficulty. Apart from that, we observe that `filterQN` mostly takes more iterations than `filter2`. To some extent this is expected due to the need for `filterQN` to spend something like d extra iterations in building up a matrix $B = UU^T$ with the property that $BZ \sim WZ$.

Next we show some results in Table 2, obtained on some larger CUTE test problems, again chosen to have significant null space dimensions. In fact the exact dimension of the null space is not so easy to determine, because the accuracy required from the QP subproblem can often be obtained without having to build up the full reduced Hessian. In the table an approximate value of d is given based on the size of the QP reduced Hessian on termination of the `filter2` run.

Some of the problems are solved quite efficiently by `filterQN`. The largest problem DTOC1L is the only linearly constrained problem in the set, so feasibility restoration is not an issue in this case. Thus it is very satisfactory that this problem is solved accurately and quickly by `filterQN`, even though the dimension d of the null space is very large. Less satisfactory is the performance on the ORTHREG problems, and also DIXCHLNV, ORTHRGDS and ZAMB2, which mostly have large null spaces with $d \gg r_{max}$. In these problems, slow convergence is observed in the asymptotic phase, and the build up of information in the U matrix is very slow. The restriction on r may also be having an effect. On other problems, the behaviour of `filterQN` is reasonably satisfactory, bearing in mind the need for for extra iterations to build up an effective Hessian approximation.

Table 2. Performance of `filterQN` on some larger CUTE problems

	n	m	$\sim d$	$\#g$	$f2$
AIRPORT	84	42	42	74	12
LAKES	90	78	12	62	421
READING6	102	50	8	40	23
ZAMB2-8	138	48	30	36	7
ZAMB2-9	138	48	10	25	4
ZAMB2-10	270	96	10	19	6
ZAMB2-11	270	96	59	65	6
DIXCHLNV	100	50	35	473	22
DTOC1L	5998	3996	1998	287	8
DTOC5	1999	999	998	24	5
EIGENB2	110	55	52	64	21
OPTCDEG2	1202	800	10	18	6
OPTCTRL6	122	80	39	74	6
ORTHRDM2	203	100	103	49	9
ORTHRDS2	203	100	103	36	32
ORTHREGC	1005	500	501	392	10
ORTHREGD	1005	500	280	216	13
ORTHREGE	36	20	12	111	63
ORTHREGF	1205	400	767	228	22
ORTHRGDS	1003	500	503	458	29
SVANBERG	500	500	19	71	9
TRAINH	808	402	17	39	14
ZAMB2	1326	480	142	349	8

9. Conclusions

We itemize the main features of this paper as follows. A new QN update scheme for low rank Hessian approximation in SQP has been presented. Where possible it uses the SR1 update formula, but when this is not possible a new rank two update is used, which is not in the Broyden family, although invariance under linear transformations of the variables is preserved. The Hessian approximation is a positive semi-definite matrix, which ensures that global solutions of QP subproblems are calculated. It also enables interior point methods to be used to solve the QP subproblems, if required. The representation provides a limited memory capability, and there is a priority ordering scheme which enables 'old' information to be deleted when the memory is full. Hereditary and conjugacy properties are preserved to the maximum extent when minimizing a quadratic function subject to linear constraints. Practical experience is reasonably encouraging on small (and some larger) problems. There is some evidence of slow convergence on some larger problems with large null spaces. It may be that this is to some extent caused by the use of an l_∞ trust region. Future

work will continue to develop the `filterQN` code, especially the feasibility restoration part, and will also investigate the use of an l_2 trust region.

References

[1] M.C. van Beurden. Integro-differential equations for electromagnetic scattering: analysis and computation for objects with electric contrast. Ph.D. Thesis, Eindhoven University of Technology, Eindhoven, The Netherlands, 2003.

[2] P.T. Boggs, J.W. Tolle and P. Wang. On the local convergence of quasi-Newton methods for constrained optimization. *SIAM J. Control and Optimization.* 20:161-171, 1982.

[3] A.R. Conn, N.I.M. Gould and Ph.L. Toint. An introduction to the structure of large scale nonlinear optimization problems and the LANCELOT project. In *Computing Methods in Applied Sciences and Engineering.* R. Glowinski and A. Lichnewsky, eds, SIAM Publications, Philadelphia, 1990.

[4] A.R. Conn, N.I.M. Gould and Ph.L. Toint. Convergence of quasi-Newton matrices generated by the symmetric rank one update. *Mathematical Programming.* 50:177-196, 1991.

[5] R. Fletcher. *Practical Methods of Optimization.* Wiley, Chichester, 1987.

[6] R. Fletcher and S. Leyffer. Nonlinear Programming Without a Penalty Function. *Mathematical Programming.* 91:239-269, 2002.

[7] R. Fletcher, S. Leyffer and Ph.L. Toint. On the Global Convergence of a Filter–SQP Algorithm. *SIAM J. Optimization.*, 13:44-59, 2002.

[8] P.E. Gill, W. Murray, M.A. Saunders. SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization. *SIAM J. Optimization.* 12:979-1006, 2002.

[9] J. Nocedal. Updating quasi-Newton matrices with limited storage. *Mathematics of Computation.* 35:773-782, 1980.

[10] J. Nocedal and M.L. Overton. Projected Hessian updating algorithms for nonlinearly constrained optimization. *SIAM J. Numerical Analysis.* 22:821-850, 1985.

[11] M.J.D. Powell and Ph.L. Toint. On the estimation of sparse Hessian matrices. *SIAM J. Numerical Analysis.* 16:1060-1074, 1979.