On Hessian- and Jacobian-Free SQP Methods - a Total Quasi-Newton Scheme with Compact Storage

Torsten Bosse[1], Andreas Griewank[2], Lutz Lehmann[3], and Volker Schloßhauer[4]

[1] † Humboldt-Universität zu Berlin, Institut für Mathematik, Unter den Linden 6, 10099 Berlin, Germany, bosse@math.hu-berlin.de
[2] † Humboldt-Universität zu Berlin, Institut für Mathematik, Unter den Linden 6, 10099 Berlin, Germany, griewank@math.hu-berlin.de
[3] Humboldt-Universität zu Berlin, Institut für Mathematik, Unter den Linden 6, 10099 Berlin, Germany, llehmann@math.hu-berlin.de
[4] † Weierstraß-Institut für Angewandte Analysis und Stochastik, Mohrenstr. 39, 10117 Berlin, Germany, schlosshauer@wias-berlin.de

Keywords: NLP; Total quasi-Newton method; Limited-memory; Symmetric rank-one update; Compact representation (with damping); Algorithmic Differentiation; BFGS

Summary. In this paper we describe several modifications to reduce the memory requirement of the total quasi-Newton method proposed by Andreas Griewank et al..

The idea is based on application of the compact representation formulae for the well-known BFGS and SR1 update for unconstrained optimization. It is shown how these definitions can be extended to a total quasi-Newton approach for the constrained case.

A brief introduction to the limited-memory approach is described in the present paper using an updated null-space factorization for the KKT system as well as an efficient numerical implementation of the null-space method in which the null-space representation is not stored directly. It can be proven that the number of operations per iteration is bounded by a bilinear order $\mathcal{O}(n \cdot \max(m, l))$ instead of a cubic order $\mathcal{O}(m \cdot n^2)$ for standard SQP methods. Here n denotes the number of variables, m the maximal number of active constraints, and l the user-selected number of stored update vectors.

† Supported by the DFG Research Center MATHEON "Mathematics for Key technologies", Straße des 17. Juni 136, 10623 Berlin, Germany, www.matheon.de

M. Diehl et al. (eds.), *Recent Advances in Optimization and its Applications in Engineering*,
DOI 10.1007/978-3-642-12598-0_6,

1 Introduction

The main goal of this work is to sketch an efficient approach to solve *nonlinear programs* of the form:

$$\left.\begin{array}{rl} \min\limits_{x\in\mathbb{R}^n} & f(x) \\ \text{s.t.} & c_{\mathcal{I}}(x) \le 0 \\ & c_{\mathcal{E}}(x) = 0 \end{array}\right\} NLP.$$

Here $c_{\mathcal{I}} = (c_i)_{i\in\mathcal{I}}$ and $c_{\mathcal{E}} = (c_i)_{i\in\mathcal{E}}$ denote the mappings composed of the *inequality constraint* functions $c_i : \mathbb{R}^n \to \mathbb{R}$, $i \in \mathcal{I}$, and *equality constraint* functions $c_i : \mathbb{R}^n \to \mathbb{R}, i \in \mathcal{E}$, where f and c_i, $i \in \mathcal{I} \cup \mathcal{J}$, are at least C^2. Also, the existence of a regular solution $x^* \in \mathbb{R}^n$ where LICQ holds is assumed.

An active-set strategy can be used to handle the inequalities. Assume that the active set $\mathcal{A}(x^*)$ of such a solution x^* is known and denote by $c_{\mathcal{A}} : \mathbb{R}^n \to \mathbb{R}^m$ the corresponding constraint mapping. Then solving the NLP is equivalent to finding the solution of the *reduced equality constraint problem*:

$$\begin{array}{rl} \min\limits_{x\in\mathbb{R}^n} & f(x) \\ \text{s.t.} & c_{\mathcal{A}}(x) = 0. \end{array}$$

Let λ_i be the Lagrange multiplier for the constraint c_i. The *Lagrangian*

$$\mathcal{L}(x,\lambda) := f(x) + \sum_{i\in\mathcal{A}} \lambda_i c_i(x)$$

associated with the reduced equality-constrained problem can be used to state the first-order optimality condition for the stationary point (x^*, λ^*):

$$\nabla_{x,\lambda_{\mathcal{A}}} \mathcal{L}(x^*, \lambda^*) = 0. \tag{1}$$

According to [4], a total quasi-Newton approach can be applied to determine (x^*, λ^*). In such an approach the reduced Hessian of the Lagrangian and the constraint Jacobian are approximated by some matrices $B \approx \nabla^2_{xx}\mathcal{L}(x,\lambda)$ and $A \approx c'_{\mathcal{A}}(x)$. Applying a null-space method based on an *extended QR factorization* $A = [L, 0][Y, Z]^\top$, one obtains the approximating *null-space factorized KKT system*

$$\begin{pmatrix} Y^\top BY & Y^\top BZ & L^\top \\ Z^\top BY & Z^\top BZ & 0 \\ L & 0 & 0 \end{pmatrix} \begin{pmatrix} Y^\top s \\ Z^\top s \\ \sigma \end{pmatrix} = - \begin{pmatrix} Y^\top \nabla_x \mathcal{L}(x,\lambda) \\ Z^\top \nabla_x \mathcal{L}(x,\lambda) \\ c_{\mathcal{A}}(x) \end{pmatrix}$$

for (1) that can be efficiently updated by low-rank formulae. Here, the matrix $Z \in \mathbb{R}^n \times \mathbb{R}^{n-m}$ contains an orthonormal null-space basis of A. The right-hand side of the equation is obtained exactly by use of the backward mode in Algorithmic Differentiation (cf. [3]). The approximate *projected Hessian* $Z^\top BZ$ is kept positive definite throughout the optimization procedure, since the exact one will have this property near local minima where second-order sufficiency conditions hold.

2 A Limited-Memory Approach for the SR1 Method

2.1 Compact Representation Formula

Consider a symmetric rank-one update (SR1) of the Hessian B defined by

$$B_+ = B + \beta \frac{(w - Bs)(w - Bs)^\top}{(w - Bs)^\top s} \quad \text{with} \quad (w - Bs)^\top s \neq 0$$

where $\beta \in (0, 1]$ is a damping parameter. In order to avoid the complete fill-in caused by the addition of low-rank terms, one prefers to store the triples $(s, w, \beta) \in \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}$ where $s := x_+ - x$ and $w := \nabla_x \mathcal{L}(x_+, \lambda) - \nabla_x \mathcal{L}(x, \lambda)$. Unless $w^\top s = 0$ the pairs (s, w) are scaled throughout such that $|w^\top s| = 1$, which leaves the secant condition $w = B_+ s$ unaffected.

In the following a sequence of damped SR1 updates identified with (s_j, w_j, β_j), $j \in \{0, \ldots, l-1\}$, is applied to $B^{(0)} := \gamma I$ using a compact representation formula, which is well-known for many quasi-Newton updates. The scaled update vectors and scalar products are, therefore, arranged in matrices

$$\begin{aligned} S &:= \begin{pmatrix} s_0 \cdots s_{l-1} \end{pmatrix} \in \mathbb{R}^{n \times l}, \quad W := \begin{pmatrix} w_0 \cdots w_{l-1} \end{pmatrix} \in \mathbb{R}^{n \times l}, \\ Q &\in \mathbb{R}^{l \times l} \quad \text{with} \quad Q_{ih} := Q_{hi} = w_{i-1}^\top s_{h-1} (i \geq h), \\ P &\in \mathbb{R}^{l \times l} \quad \text{with} \quad P_{ih} := P_{hi} = s_{i-1}^\top w_{h-1} (i \geq h). \end{aligned}$$

Theorem 1 (SR1 - Compact representation formula).
Let l be the number of damped regular SR1 updates $(s_j, w_j, \beta_j)_{j=0}^{l-1}$, i.e.

$$(w_j - B^{(j)} s_j)^\top s_j \neq 0, \ \beta_j \neq 0 \quad \forall j \in \{0, \ldots, l-1\},$$

applied to the initial matrix $B^{(0)} = \gamma I$ with $B^{(j)}$ defined as the intermediate matrix after applying the first $j \leq l$ updates. Then $M := P - D - \gamma S^\top S \in \mathbb{R}^{l \times l}$ is invertible and $B = B^{(l)}$ is given by

$$B = \gamma I + (W - \gamma S) M^{-1} (W - \gamma S)^\top \tag{2}$$

where D denotes the diagonal matrix $D = \operatorname{diag}(D_{jj})_{j=0}^{l-1}$ with

$$D_{jj} := (1 - \beta_j^{-1})(w_j - B^{(j)} s_j)^\top s_j.$$

A compact representation formula for the BFGS update can be found in [2].

Remark: Equation (2) represents a generalization of the usual SR1 update formula in [2]. In the undamped case, i.e. $(\beta_j)_{j=0}^{l-1} = 1$, D vanishes.

Due to the Sherman Morrison Woodbury formula, one obtains a similar formula for the inverse. Therefore, define $N := Q + D - \gamma^{-1} W^\top W$ and verify

$$B^{-1} = \gamma^{-1} I + (S - \gamma^{-1} W) N^{-1} (S - \gamma^{-1} W)^\top.$$

The compact representation formulae offer a number of advantages over the full-storage implementation. First and foremost the space for storing B is reduced to a pair of low-rank matrices S and W and the scalar γ that ideally represents the average eigenvalue of B. In a limited-memory approach the number l of updates is fixed, so only the most recent update vectors are kept inside S and W. The computational effort for adding (or replacing) update vectors for B is bounded by $\mathcal{O}(l \cdot n)$ compared to $\mathcal{O}(n^2)$ for SR1 updates. The bound $\mathcal{O}(l \cdot n + l^3)$ holds for multiplying vectors by B or its inverse B^{-1}. If $l \ll \sqrt{n}$ is small, the factorization effort for M and N stays negligible.

On the other hand, not storing all updates causes the loss of superlinear convergence (see [6]), which may possibly increase the overall computational effort.

2.2 Maintaining the Positive Definiteness of the Hessian

Positive definiteness of $Z^\top B Z$ and maximal rank of A imply unique solvability of the KKT system. Unlike the BFGS update, the SR1 update does not necessarily preserve the positive definiteness of $Z^\top B Z$. A remedy is proposed in [7] for the limited-memory approach. It consists of both determining suitable values for the damping parameters β_i and adapting the scaling parameter γ. More specifically, one obtains the following statement (cf. [7]) for $\bar{Q} := Q + D \in \mathbb{R}^{l \times l}$ as defined before including a constructive proof for γ:

Lemma 1. *If $\bar{Q}$ is positive definite,*[5] *then there exists $\Gamma > 0$ such that B becomes positive definite for all $\gamma > \Gamma$.*

Proof. Consider auxiliary matrices $T_1, T_2, T_3 \in \mathbb{R}^{(n+l)\times(n+l)}$ defined by

$$T_1 := \begin{pmatrix} \gamma I & U \\ U^\top & -M \end{pmatrix} \quad \text{with} \quad U = (W - \gamma S),$$

$$T_2 := \begin{pmatrix} I & U M^{-1} \\ 0 & I \end{pmatrix} \begin{pmatrix} \gamma I & U \\ U^\top & -M \end{pmatrix} \begin{pmatrix} I & 0 \\ M^{-1} U^\top & I \end{pmatrix} = \begin{pmatrix} B & 0 \\ 0 & -M \end{pmatrix},$$

$$T_3 := \begin{pmatrix} I & 0 \\ -\gamma^{-1} U^\top & I \end{pmatrix} \begin{pmatrix} \gamma I & U \\ U^\top & -M \end{pmatrix} \begin{pmatrix} I & -\gamma^{-1} U \\ 0 & I \end{pmatrix} = \begin{pmatrix} \gamma I & 0 \\ 0 & -M - \gamma^{-1} U^\top U \end{pmatrix}.$$

Simplifying the last equation one recovers the middle term N of B^{-1}:

$$-M - \gamma^{-1} U^\top U = W^\top S + S^\top W - P + D - \gamma^{-1} W^\top W = \bar{Q} - \gamma^{-1} W^\top W.$$

Due to Sylvester's law, the inertias of T_1, T_2 and T_3 coincide. So, one can deduce: B is positive definite (as $B^{(0)} = \gamma I$) if and only if $-M$ and N have the same inertia. Furthermore, if $\bar{Q}$ is positive definite, then there exists $\Gamma > 0$ such that N, T_3 and B become positive definite. □

[5] The assumption is reasonable, as in quadratic programming without damping one retrieves: $\bar{Q} = Q = S^\top \nabla^2_{xx} \mathcal{L}(x, \lambda) S$ is positive definite.

The assumption of the previous lemma can be guaranteed by damping a new secant update pair (s_{new}, w_{new}) to prevent the compact representation (2) of the reduced Hessian losing its positive definiteness property. Therefore, consider the rank-two update formula that describes the replacement of a single update (s_h, w_h, β_h) by $(s_{new}, w_{new}, \beta_{new})$ for $h \in \{0, \ldots, l-1\}$ in $\bar{Q}$:

$$\bar{Q}_{new} := \bar{Q} + \frac{1}{2}(e_h + d)(e_h + d)^\top - \frac{1}{2}(e_h - d)(e_h - d)^\top + \beta_{new} e_h e_h^\top \qquad (3)$$

$$\text{and} \quad (d_j)_{j=0}^{l-1} := \begin{cases} s_j^\top w_{new} - \bar{Q}_{hj} & \text{if } (j \neq h) \\ \frac{1}{2}(s_{new}^\top w_{new} - \bar{Q}_{hh}) & \text{otherwise.} \end{cases}$$

Since the largest eigenvalue of $\bar{Q}_{new}$ cannot grow rapidly but its smallest one could become zero or even negative one can control its conditioning by a Powell-like test on the determinant. A suitable choice for the damping parameter β_{new} can then be derived by investigating the determinant ratio for $\bar{Q}$ and the updated version $\bar{Q}_{new}$:

Lemma 2 (Determinant ratio for damping parameters).
Let $(s_i, w_i, \beta_i)_{i=0}^{l-1}$ be a sequence of l regular SR1 updates and $(s_{new}, w_{new}, \beta_{new})$ be a regular SR1 update replacing (s_h, w_h, β_h), $h \in \{0, \ldots, l-1\}$. Define the determinant ratio function $q : \mathbb{R} \to \mathbb{R}$ as

$$q(\beta_{new}) := \frac{\det \bar{Q}_{new}}{\det \bar{Q}}.$$

Then it holds for $b := \bar{Q}^{-1} e_h$ and $c := \bar{Q}^{-1} d$:

$$q(\beta_{new}) = b_h \beta_{new} + c_h^2 + 2c_h - b_h c^\top d + 1.$$

Choosing β_{new} such that $q(\beta_{new}) \in [1/\mu, \mu]$ maintains the positive definiteness of $\bar{Q}_{new}$ after the update (3) where $1 < \mu$ is a fixed constant. In the next step one can numerically try ascending values for γ and verify the positive definiteness of B by analyzing the inertias of $-M$ and N according to the first Lemma.

2.3 Constrained Optimization and Limited-Memory

Consider again the factorized KKT system of the equality-constrained problem for computing a total quasi-Newton step, as described in Section 1:

$$\begin{pmatrix} Y^\top BY & Y^\top BZ & L^\top \\ Z^\top BY & Z^\top BZ & 0 \\ L & 0 & 0 \end{pmatrix} \begin{pmatrix} Y^\top s \\ Z^\top s \\ \sigma \end{pmatrix} = - \begin{pmatrix} Y^\top \nabla_x \mathcal{L}(x, \lambda) \\ Z^\top \nabla_x \mathcal{L}(x, \lambda) \\ c_{\mathcal{A}}(x) \end{pmatrix}. \qquad (4)$$

Then the limited-memory approach can easily be incorporated by replacing B with the compact representation formula. Hence, instead of storing the factors $Y^\top BY$, $Z^\top BY$, and $Z^\top BZ$, it is sufficient to store and update only

the matrices W, S, and two smaller matrices in $\mathbb{R}^{l\times l}$. In addition, the necessary matrix-vector products can be calculated directly by multiplication from right to left using the reformulation

$$\begin{aligned} Y^\top BY &= \gamma I + (Y^\top W - \gamma Y^\top S)M^{-1}(Y^\top W - \gamma Y^\top S)^\top, \\ Y^\top BZ &= (Y^\top W - \gamma Y^\top S)M^{-1}(Z^\top W - \gamma Z^\top S)^\top, \\ Z^\top BZ &= \gamma I + (Z^\top W - \gamma Z^\top S)M^{-1}(Z^\top W - \gamma Z^\top S)^\top, \text{ and} \\ (Z^\top BZ)^{-1} &= \gamma^{-1} I + \gamma^{-2}(Z^\top W - \gamma Z^\top S)N^{-1}(Z^\top W - \gamma Z^\top S)^\top \end{aligned}$$

where the middle matrices M, $N \in \mathbb{R}^{l\times l}$ are now defined as follows:

$$M := P - D - \gamma S^\top S \text{ and } N := -M - \gamma^{-1}(W - \gamma S)^\top ZZ^\top (W - \gamma S).$$

Since the damping of the update and the choice of γ discussed in section 2.2 ensures the positive definiteness of B, this property will be shared by the reduced Hessian $Z^\top BZ$. A major concern now is to handle the matrices Y and Z of the extended QR-factorization, which also need to be stored. Consequently, one needs at least a complexity of order $\mathcal{O}(n^2)$ to store the Jacobian factorization, even for problems with a few active constraints. The next section gives a possible solution to this drawback.

2.4 Avoidance of the Null-space Factor Z

When using a partial limited-memory method in conjunction with a total quasi-Newton approach and a null-space factorized KKT system, a significant amount of memory is expended on the matrix Z containing the null-space basis of the Jacobian. This fact reduces the benefits of the limited-memory approach, especially, if only a small number of constraints is active. The following summarizes how the partial limited-memory approach can be improved by utilizing the orthonormality relation $ZZ^\top + YY^\top = I$ for the range- and null-space representation $[Y, Z]$.

In this case the storage of the $(n - m) \times n$ matrix Z can be avoided without any loss in theory. According to [1], Z is necessary neither to get a total quasi-Newton step nor for the update of the factorized KKT system (4) itself. Thus, a further reduction of the computational effort in a realization of an algorithm is possible by eliminating Z. Also a bilinear upper bound on memory allocation and the operation count per iteration is obtained.

Theorem 2 (Solving KKT without Z).
The solution of the approximated null-space factorized KKT system (4)

$$\begin{aligned} s &= -YL^{-1}c_{\mathcal{A}}(x) - Z(Z^\top BZ)^{-1}(Z^\top \nabla_x \mathcal{L}(x,\lambda) - Z^\top BYL^{-1}c_{\mathcal{A}}(x)) \\ \sigma &= -L^{-\top}(Y^\top \nabla_x \mathcal{L}(x,\lambda) + Y^\top BYY^\top s + Y^\top BZZ^\top s) \end{aligned}$$

can be computed ***without using Z*** *if the Hessian approximation B is given as a low-rank perturbation of a multiple of the identity matrix.*

Proof. Consider the computation of the vector s, which can be written as

$$s = -YL^{-1}c_{\mathcal{A}}(x) - Z(Z^\top BZ)^{-1}Z^\top \left[\nabla_x \mathcal{L}(x,\lambda) - BYL^{-1}c_{\mathcal{A}}(x)\right].$$

Here only the factor $Z(Z^\top BZ)^{-1}Z^\top$ is interesting, as it depends on Z. With reference to section 2.3, $(Z^\top BZ)^{-1}$ is given by

$$\begin{aligned}(Z^\top BZ)^{-1} = \gamma^{-1}I + \gamma^{-2}(Z^\top W - \gamma Z^\top S)[-M \\ -\gamma^{-1}(W-\gamma S)ZZ^\top(W-\gamma S)]^{-1}(Z^\top W - \gamma Z^\top S)^\top.\end{aligned}$$

Multiplication on left and right by Z and its transpose, respectively, yields

$$\begin{aligned}Z(Z^\top BZ)^{-1}Z^\top = \gamma^{-1}ZZ^\top + \gamma^{-2}ZZ^\top(W-\gamma S)\,[-M \\ -\gamma^{-1}(W-\gamma S)^\top ZZ^\top(W-\gamma S)\big]^{-1}(W-\gamma S)^\top ZZ^\top.\end{aligned}$$

Applying the identity $ZZ^\top = I - YY^\top$ to the equation above as well as to the formula for the Lagrange multiplier step via

$$\begin{aligned}\sigma &= -L^{-\top}\left[Y^\top\nabla_x\mathcal{L}(x,\lambda) + Y^\top BYY^\top s + Y^\top BZZ^\top s\right] \\ &= -L^{-\top}Y^\top\left[\nabla_x\mathcal{L}(x,\lambda) + Bs\right]\end{aligned}$$

concludes the proof. □

2.5 Improving Computational Efficiency

From a numerical point of view the most time-consuming part per iteration is the step computation. Here several matrix-matrix products of order no less than $\mathcal{O}(n\cdot m\cdot l)$ would be necessary, since the reformulation

$$Z(Z^\top BZ)^{-1}Z^\top = (\gamma^{-1}I + \gamma^{-2}ZZ^\top(W-\gamma S)N^{-1}(W-\gamma S)^\top)ZZ^\top$$

involves a computation and factorization of the middle matrix N:

$$N = -M - \gamma^{-1}(W-\gamma S)(I-YY^\top)(W-\gamma S) \in \mathbb{R}^{l\times l}.$$

As proven in [1], the basic idea to overcome this drawback is to avoid recomputation of N from scratch and to apply Hessian and Jacobian updates directly to the matrix N.

Hence, one can show by multiplication from right to left that the whole step-computation has bilinear complexity $\mathcal{O}(n\cdot\max(m,l))$ because the remaining matrix-matrix additions as well as matrix-vector products can be considered as cheap, i.e. of bilinear complexity. Note that N can be factorized from scratch without exceeding $\mathcal{O}(n\cdot l)$ operations for $l \ll n$ sufficiently small.

The update of the matrix N due to changes of the Hessian, the Jacobian, and the scaling parameter γ is examined in three propositions, where it is proven that the effort is bounded by $\mathcal{O}(n\cdot max(m,l))$ operations. Since the proofs are quite analogous, only the one for the Hessian updates is given.

Proposition 1 (Updating N - Hessian updates).
The matrix N can be directly updated with $\mathcal{O}(n \cdot max(m, l))$ operations if the Hessian is subject to a rank-one modification.

Proof. Three different actions can be performed if the Hessian is updated in the limited-memory case:

1. A new secant pair (s_i, w_i) is added to (S, W),
2. an old pair (s_i, w_i) is removed from (S, W), or
3. an old update (s_i, w_i) is exchanged by a new one (s_{new}, w_{new}).

In all these cases the matrix N needs to be modified as it depends on (S, W). The basic idea of the proof is to represent these changes as a constant number of low-rank updates. Therefore, not only the matrices S and W will be stored and updated but also $S^\top Y$, $W^\top Y$, and all summands of N up to transpositions. All the three cases will be illustrated on $S^\top W$.

1. Appending a new update pair (s_{new}, w_{new}) to the set (S, W) by setting $(S, W)_+ = ((s_1, \ldots, s_{i-1}, s_i = s_{new}), (w_1, \ldots, w_{i-1}, w_i = w_{new}))$ results in an extended matrix plus a rank-two update:

$$\begin{aligned}(S^\top W)_+ &= \begin{bmatrix} S^\top W & S^\top w_i \\ s_i^\top W & s_i^\top w_i \end{bmatrix} \\ &= \begin{bmatrix} W^\top S & 0 \\ 0 & 0 \end{bmatrix} + (S^\top w_i) e_i^\top + e_i (s_i^\top W)^\top + w_i^\top s_i (e_i e_i^\top).\end{aligned}$$

2. Assume the secant pair (s_i, w_i) that shall be deleted is in last position in (S, W), i.e. $(S, W) = ((s_1, \ldots, s_i), (w_1, \ldots, w_i))$. Otherwise use the routine described in the next point to exchange it with the last one. Then the secant pair can be removed by erasing the last row and column of $S^\top W$.
3. Exchanging a secant pair (s_i, w_i) by a new one can be realized by a rank-two update on (S, W) with $\tilde{s} := (s_{new} - s_i)$ and $\tilde{w} := (w_{new} - w_i)$:

$$(S^\top W)_+ = S^\top W + e_i \tilde{s}^\top W + S^\top \tilde{w} e_i^\top + \tilde{s}^\top \tilde{w} (e_i e_i^\top).$$

Obviously, the operation count for the updates of all summands is dominated by two extra calculations including the Y-factor, i.e. $s_{new}^\top Y$ and $w_{new}^\top Y$, where the numerical effort is of order $\mathcal{O}(n \cdot m)$. Evaluating the remaining expressions without Y is cheap, e.g. the expression $e_i (s_i^\top W)^\top$ can be computed by first evaluating $s_i^\top W$ and storing this vector. In these cases the complexity bound $\mathcal{O}(n \cdot l)$ is not exceeded. Applying the results on N, one derives that the new middle matrix N_+ is given by a sequence of rank-one updates:

1. Appending a new secant pair (s_i, w_i) to (S, W) gives:

$$N_+ = \begin{bmatrix} N & 0 \\ 0 & 0 \end{bmatrix} + \sum_{j=1}^{8} \lambda_j (e_i v_j^\top + v_j e_i^\top) + \sum_{j=9}^{16} \lambda_j (e_i e_i^\top),$$

2. using MATLAB-like notation, the deletion of (s_i, w_i) in (S, W) yields:

$$N_+ = \left(N + \sum_{j=1}^{8} \lambda_j (e_i v_j^\top + v_j e_i^\top) + \sum_{j=9}^{16} \lambda_j (e_i e_i^\top) \right) [1:k-1; 1:k-1],$$

3. and exchanging (s_i, w_i) with (s_{new}, w_{new}) results in:

$$N_+ = N + \sum_{j=1}^{8} \lambda_j (e_i v_j^\top + v_j e_i^\top) + \sum_{j=9}^{16} \lambda_j (e_i e_i^\top)$$

where the vectors v_j and scalars λ_j are defined by the performed action. □

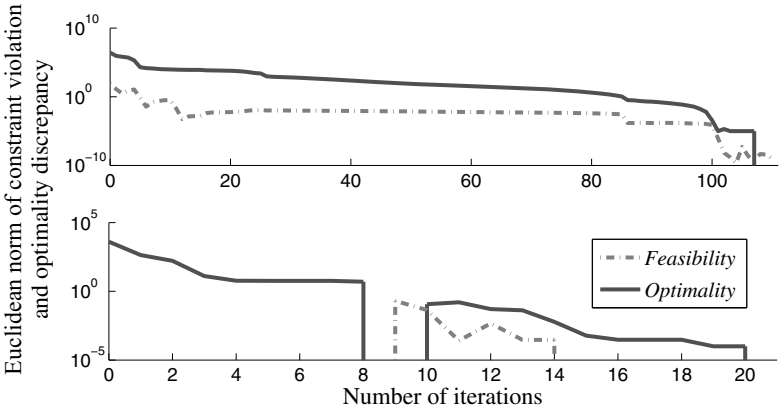
Hence, the following result is obtained by a careful implementation of the linear algebra for the updating of the factorized KKT system.

Theorem 3 (Bosse).
For a partial limited-memory approach on a total quasi-Newton method with an updated null-space factorized KKT system, the needed memory size and computational effort per iteration are both of order $\mathcal{O}(nm + nl + l^3)$.

More details on the proof can be found in [1], Chapter 6: 'Zed is Dead'.

3 Examples

The effectiveness of the presented method has been verified on the two examples LUKVLE3 (top) and LUKVLI9 (bottom) from the CUTEr test set.



Here the number of constraints is small ($m = 2$, $m = 6$) , whereas the number of variables is comparatively large ($n \approx 10000$). For $l = 4$ secant pairs in storage, the two problems were solved within 111 and 20 iterations, respectively. Thus, the overall effort $\sim 100 \cdot 6 \cdot 10^4$ arithmetic operations was less than that for one null-space factorization of the full KKT system. IPOPT takes 9 and 33 steps, respectively, using full first- and second-order derivative information!

4 Conclusion

This article summarizes our recent research on total quasi-Newton methods for nonlinear programming. A practical implementation of the limited-memory SR1 method is presented. It avoids the explicit storage of the Hessian and reduces the computational effort for quasi-Newton updates to about $\mathcal{O}(l \cdot n)$ operations. A null-space factorized KKT system in the constrained case is reformulated by means of compact representation formulae and solved efficiently using an updated QR decomposition of the Jacobian. The new approach circumvents the necessity of storing the matrix Z for the solution of the system while reducing the computational effort per iteration to the bilinear complexity $\mathcal{O}(n \cdot \max(l, m))$. This should be particularly beneficial on dense large-scale problems with a small set of active constraints $m \ll n$.

The quoted results for the large-scale problems LUKVLE3 and LUKVLI9 indicate acceptable linear convergence rates even for a small number of stored secant pairs ($l = 4$) with drastic reduction in computational effort per iteration. More runs on the CUTEr test set have shown as a rule of thumb that the choice of l between ~ 5 and ~ 15 results in a good balance between an acceptable linear convergence rate and an effective step computation.

A further reduction in storage and operations count is envisioned by a *seminormal* approach that is based on a range-space method. In this case also the storage of the range-space basis Y is omitted. The matrix-vector products including Y are replaced by an extra Algorithmic Differentiation operation. A smart updating of the triangular matrix L reduces the effort to the order $\mathcal{O}(m^2/2 + n \cdot l)$.

References

1. Bosse T (2009) A Derivative-matrix-free NLP Solver without Explicit Nullspace Representation. Diploma Thesis, Humboldt Universität zu Berlin, Berlin
2. Byrd R, et al. (1994) Representations of quasi-Newton matrices and their use in limited-memory methods, Math. Programming 63:129–156
3. Griewank A, Walther A (2008) Evaluating derivatives. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA
4. Griewank A, Walther A, Korzec M (2007) Maintaining factorized KKT systems subject to rank-one updates of Hessians and Jacobians, Optimization Methods & Software 22:279–295
5. Korzec M (2006) A General-Low-Rank Update-Based Quadratic Programming Solver. Diploma Thesis, Humboldt Universität zu Berlin, Berlin
6. Nocedal J, Wright S (2006) Numerical Optimization, Springer Series in Operations Research, 2nd Edt.
7. Schloßhauer V (2009) Strukturausnutzung und Speicherplatzbegrenzung für hochdimensionale, nichtlineare Optimierung. Diploma Thesis, Humboldt Universität zu Berlin, Berlin