

---

# On Hessian- and Jacobian-free SQP methods - a total quasi-Newton scheme with compact storage

Torsten Bosse<sup>1</sup>, Volker Schloßhauer<sup>2</sup> and Andreas Griewank<sup>3</sup>

<sup>1</sup> † Humboldt Universität zu Berlin, Institut für Mathematik, Unter den Linden 6,  
10099 Berlin, Germany, [bosse@math.hu-berlin.de](mailto:bosse@math.hu-berlin.de)

<sup>2</sup> † Weierstraß-Institut für Angewandte Analysis und Stochastik, Mohrenstr. 39,  
10117 Berlin, Germany, [schlosshauer@wias-berlin.de](mailto:schlosshauer@wias-berlin.de)

<sup>3</sup> † Humboldt Universität zu Berlin, Institut für Mathematik, Unter den Linden 6,  
10099 Berlin, Germany, [griewank@math.hu-berlin.de](mailto:griewank@math.hu-berlin.de)

## Abstract

*Keywords:* NLP; Total quasi-Newton method; Limited memory; Symmetric rank-one update;  
Compact representation (with damping); ZED is DEAD; Algorithmic differentiation

In this paper we describe several modifications of the total quasi-Newton method proposed in [5] to reduce the memory requirement drastically.

The basic formulae for limited memory versions of the BFGS and symmetric rank-one update were given in [2]. The use in SNOPT was reported in [3]. It was shown in [8] how this idea can be extended to the total quasi-Newton approach using an updated nullspace factorization for the KKT system.

A brief introduction to the limited memory approach for a total quasi-Newton method based on Automatic Differentiation (cf. [4]) is described in the present paper. Also an effective way for the implementation of the nullspace factorization is given in which the nullspace representation is not stored directly. Thus, it can be proven (cf. [1]) that the number of operations per iteration is bounded by a bilinear order  $\mathcal{O}(n \cdot \max(m, l))$  instead of the cubic order  $\mathcal{O}(m \cdot n^2)$  for standard SQP methods. Here,  $n$  is the number of variables,  $m$  the maximal number of active constraints and  $l$  the user selected number of stored update vectors.

---

<sup>†</sup> Supported by the DFG Research Center MATHEON "Mathematics for Key technologies", Straße des 17. Juni 136, 10623 Berlin, Germany, [www.matheon.de](http://www.matheon.de)

## 1 Introduction

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

$$\left. \begin{array}{l} \min_{x \in \mathbb{R}^n} f(x) \\ \text{s.t. } c_{\mathcal{I}}(x) \leq 0 \\ c_{\mathcal{E}}(x) = 0 \end{array} \right\} NLP$$

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  $c_i : \mathbb{R}^n \rightarrow \mathbb{R}, i \in \mathcal{I}$  and *equality constraint* functions  $c_i : \mathbb{R}^n \rightarrow \mathbb{R}, i \in \mathcal{E}$ , where  $f$  and  $c_i, i \in \mathcal{I} \cup \mathcal{E}$  are at least  $C^2$ . Furthermore, the existence of a regular solution 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 the NLP is known and denote by  $c_{\mathcal{A}} : \mathbb{R}^n \rightarrow \mathbb{R}^m$  the correspondingly restricted mapping. Then solving the NLP is equivalent to finding a solution of the *reduced equality constraint problem*:

$$\left. \begin{array}{l} \min_{x \in \mathbb{R}^n} f(x) \\ \text{s.t. } c_{\mathcal{A}}(x) = 0 \end{array} \right\} (R - ECP)$$

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 yields the first order optimality condition for stationary points  $(x^*, \lambda^*)$

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

According to [5] a total quasi-Newton method can be applied to determine these KKT-points. In such a method a *nullspace factorization*

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

defined by an *extended QR factorization*  $A = [L, 0][Y, Z]^{\top}$  is updated in every step. Here  $A \approx c'_{\mathcal{A}}(x)$  and  $B \approx \nabla_{xx}^2 \mathcal{L}(x, \lambda)$  denote the approximation of the constraint Jacobian and the Hessian of the Lagrangian, respectively.

The approximate *projected Hessian*  $Z^{\top} B Z$  is kept positive definite, 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 may prefer to store a tuple  $(s, w, \beta) \in \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}$ , where  $s = x_+ - x$  and  $w = \nabla_x \mathcal{L}(x_+, \lambda) - \nabla_x \mathcal{L}(x, \lambda)$  contain all information to recover the secant condition

$$w = B_+ s.$$

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

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

A similar result for the BFGS formula like the following can be found in [2].

**Theorem 1.** *Denote with  $l$  the number of undamped symmetric rank-one update pairs  $(s_j, w_j, 1)_{j=0}^{l-1}$  applied to the initial matrix  $B^{(0)} = \gamma I$  and assume*

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

where  $B^{(j)}$  denotes the intermediate matrix on applying the first  $j \leq l$  updates. Then  $M := P - \gamma S^\top S \in \mathbb{R}^{l \times l}$  is invertible and  $B = B^{(l)}$  is given by

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

*Remark:* To deal with numerical instabilities of the SR1 update one can analyze the minors of  $M$  which become zero, if any condition in (1) is not fulfilled.

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

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

These 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  $\gamma$  which 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  $W$  and 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. The bound  $\mathcal{O}(l \cdot n) + \mathcal{O}(l^3)$  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 as observed in many numerical experiments, see [7].

## 2.2 Maintaining the Positive Definiteness of the Hessian

Unlike the BFGS update the SR1 update does not necessarily preserve the property of positive definiteness for  $Z^\top B Z$  which implies together with non-singularity of the Jacobian the solvability of the KKT system. A remedy is proposed in [8] for the limited memory approach. It consists of both introducing damping parameters  $\beta_i$  for the updates  $(s_i, w_i)$  and adapting the scaling parameter  $\gamma$ . More specifically we have for  $Q \in \mathbb{R}^{l \times l}$  as defined before:

**Lemma 1 (Lehmann).** *If  $Q$  is positive definite,<sup>4</sup> 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

$$\begin{aligned} T_1 &= \begin{pmatrix} \gamma I & U \\ U^\top & -M \end{pmatrix} \quad \text{with } U = (W - \gamma S), \\ T_2 &= \begin{pmatrix} I & U M^{-1} \\ 0 & I \end{pmatrix} \begin{pmatrix} \gamma I & U \\ U^\top & -M \end{pmatrix} \begin{pmatrix} I & 0 \\ M^{-1} U^\top & I \end{pmatrix} = \begin{pmatrix} B & 0 \\ 0 & -M \end{pmatrix}, \\ T_3 &= \begin{pmatrix} I & 0 \\ -\gamma^{-1} U^\top & I \end{pmatrix} \begin{pmatrix} \gamma I & U \\ U^\top & -M \end{pmatrix} \begin{pmatrix} I & -\gamma^{-1} U \\ 0 & I \end{pmatrix} = \begin{pmatrix} \gamma I & 0 \\ 0 & -M - \gamma^{-1} U^\top U \end{pmatrix}. \end{aligned}$$

Simplifying the last equation one recovers the midterm  $N$  of  $B^{-1}$ :

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

Due to Sylvester's law, the inertia of  $T_1$ ,  $T_2$  and  $T_3$  coincides. So, one can deduce:  $B$  is positive definite (as  $B^{(0)} = \gamma I$ ), if and only if  $-M$  and  $N$  have the same number of positive and negative eigenvalues. If  $Q$  is positive definite there exists  $\Gamma > 0$  such that  $N$  becomes positive definite as well as  $T_3$  and  $B$ .

In a limited memory approach the replacement of an update  $(s_i, w_i)$  by  $(s_+, w_+)$  in  $M$ ,  $N$ ,  $S_+$  and  $W_+$  has to be taken into account. To guarantee the assumptions of the previous lemma a damping on the updates is applied.

<sup>4</sup> The assumption of positive definiteness for  $Q$  is reasonable, as in quadratic programming one retrieves:  $Q = S^\top \nabla_{xx}^2 \mathcal{L}(x, \lambda) S$ .

**Theorem 2 (Compact representation with damping).**

The application of  $l$  damped SR1 updates  $(s_j, w_j, \beta_j)_{j=0, \dots, l-1}$  on  $B^{(0)} = \gamma I$  with  $(w_j - B^{(j)} s_j)^\top s_j \neq 0$  and  $\beta_j \neq 0$  for  $j = 0, \dots, l-1$  yields

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

where  $D$  is given by the diagonal matrix  $D = \text{diag}(D_{jj})_{j=0, \dots, l-1}$  with

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

A suitable choice for  $\beta_+$  can be deduced from the following statement, such that an analogous lemma to the previous one is applicable for  $\bar{Q} = Q + D$ .

**Lemma 2 (Determinant ratio for damping parameters).**

Let  $(s_i, w_i, \beta_i)_{i=0, \dots, l-1}$  be a sequence of  $l$  SR1 updates and  $(s_+, w_+, \beta_+)$  be a single update as in (1) replacing  $(s_h, w_h, \beta_h)$ ,  $h \in \{0, \dots, l-1\}$ . Define  $\bar{Q} = Q + D$  and  $q : \mathbb{R} \rightarrow \mathbb{R}$  as

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

Then, it holds

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

where  $b = \bar{Q}^{-1} e_h$ ,  $c = \bar{Q}^{-1} d$  and the vector

$$(d_j)_{j=0}^{l-1} = \begin{cases} s_j^\top w_+ - \bar{Q}_{ij} & \text{if } (j \neq i) \\ \frac{1}{2}(s_+^\top w_+ - \bar{Q}_{ii}) & \text{otherwise.} \end{cases}$$

In [8] it is shown that determining  $\beta_+$  such that  $q(\beta_+) > 0$  ensures the positive definiteness of  $\bar{Q}_+$ . In the next step one can numerically try ascending values for  $\gamma$  and verify the positive definiteness of  $B$  by means of Lemma 1.

**2.3 Constrained Optimization and Limited Memory**

Consider again the nullspace factorized KKT system of the equality constrained problem for computing a total quasi-Newton step

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

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

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

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

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

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

## 2.4 ZED is DEAD

When using a partial limited memory approach in conjunction with a total quasi-Newton method with nullspace factorization a significant amount of memory is expended on the matrix  $Z$  containing the nullspace basis of the Jacobian. This fact limits the benefits from the limited memory approach, especially if only a small number of constraints is active. In [8] a range-space implementation is suggested to avoid the storage of  $Z$ . In the following we give the basic idea how the newly established partial limited memory nullspace approach can be improved by utilizing the orthonormality relation  $ZZ^\top + YY^\top = I$  for the range- and nullspace 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 needed neither to get a total quasi-Newton step nor for the update of the factorized KKT system itself. Therefore, by eliminating  $Z$  a further reduction of the computational effort of a practical algorithm is possible. Also a remarkably low, upper bound on memory and the operation count per loop step is obtained.

**Theorem 3 (Solving KKT without Z).** *The step direction / solution*

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

for the approximated nullspace factorized KKT system given by

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

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

*Proof.* Consider the computation of the step vector  $s$  which can be written as

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

Here only the factor  $Z(Z^{\top}BZ)^{-1}Z^{\top}$  is interesting, as it depends on  $Z$ . Due to the previous chapter about limited memory,  $(Z^{\top}BZ)^{-1}$  is given by

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

Multiplication from the left and the right side by  $Z$  resp.  $Z^{\top}$  yields

$$Z(Z^{\top}BZ)^{-1}Z^{\top} = \gamma^{-1}ZZ^{\top} + \gamma^{-2}ZZ^{\top}(W - \gamma S)[- \bar{M} \\ - \gamma^{-1}(W - \gamma S)^{\top}ZZ^{\top}(W - \gamma S)]^{-1}(W - \gamma S)^{\top}ZZ^{\top}$$

Applying the identity  $ZZ^{\top} = (I - YY^{\top})$  to the above equation as well as for the computation of the Lagrange multiplier step via

$$\begin{aligned} \sigma &= -L^{-\top} [Y^{\top} \nabla_x \mathcal{L}(x, \lambda) + Y^{\top} B Y Y^{\top} s + Y^{\top} B Z Z^{\top} s] \\ &= -L^{-\top} [Y^{\top} \nabla_x \mathcal{L}(x, \lambda) + Y^{\top} B Y Y^{\top} s + Y^{\top} B (I - Y Y^{\top}) s] \\ &= -L^{-\top} Y^{\top} [\nabla_x \mathcal{L}(x, \lambda) + B s] \end{aligned}$$

concludes the proof.

## 2.5 Improving Computational Effort Efficiently

From a computational 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

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

involves a computation and factorization of the middle matrix  $\bar{N}$  given by

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

As proved in [1], the basic idea to overcome this drawback is to avoid recomputation of  $\bar{N}$  from scratch, but apply updates performed on the Hessian and Jacobian directly to the matrix  $\bar{N}$ .

Because matrix-matrix addition and matrix-vectors products are 'cheap', and since for  $l \ll n$  sufficiently small  $\bar{N}$  can be factorized from scratch without exceeding  $\mathcal{O}(n \cdot l)$  operations, one can show by multiplication from right to left that

$$OPS(Z(Z^{\top}BZ)^{-1}Z^{\top} \cdot v) = \mathcal{O}(n \cdot \max(m, l))$$

holds and therefore, the whole step-computation has bilinear complexity.

The proof is split up into three parts, which examine all possible changes having an influence on the matrix  $\bar{N}$  and show that the effort is bounded by  $\mathcal{O}(n \cdot \max(m, l))$  operations. Since the proofs of the following propositions are quite similar, only one will be given.

**Proposition 1 (Updating  $\bar{N}$  - Hessian updates).** *The midmatrix  $\bar{N}$  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  $(S, W)$
3. or an old update  $(s_i, w_i)$  is exchanged by a new one  $(s_{new}, w_{new})$ .

All these cases modify the matrix  $\bar{N}$ , since 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 saved and updated, but also  $S^\top Y$  and  $W^\top Y$  and all summands of  $\bar{N}$  up to transpositions. All the three cases will be illustrated on  $S^\top W$

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

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

2. Assume the secant pair  $(s_i, w_i)$ , which shall be deleted is in last position in  $(S, W)$  i.e.  $(S, W) = ((s_1, \dots, s_i), (w_1, \dots, w_i))$ , otherwise use the routine described in the next point to overwrite it with the last one. Then by erasing the last row and column of  $S^\top W$  the secant pair can be removed.
3. Exchanging a secant pair  $(s_i, w_i)$  by a new one, can be done by three rank-1 terms on  $(S, W)$  with  $\tilde{s} := (s_{new} - s_i)$  and  $\tilde{w} := (w_{new} - w_i)$ :

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

For the terms including  $Y$  two extra calculations are needed, i.e.  $s_{new}^\top Y$  and  $w_{new}^\top Y$ , which are restricted by  $\mathcal{O}(n \cdot m)$ .

In order to keep the computational effort low the expressions  $e_i(s_i^\top W)^\top$  are evaluated by computing  $s_i^\top W$  first and only storing these vectors, instead of matrices filled with zeros.

Applying these results on  $\bar{N}$ , one gets a sequence of rank-1 updates :

1. Adding  $(s_i, w_i)$  to  $(S, W)$  gives<sup>1</sup> :

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

2. Deletion of  $(s_i, w_i)$  in  $((s_1, \dots, s_k), (w_1, \dots, w_k))$  yields<sup>4</sup>:

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

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

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

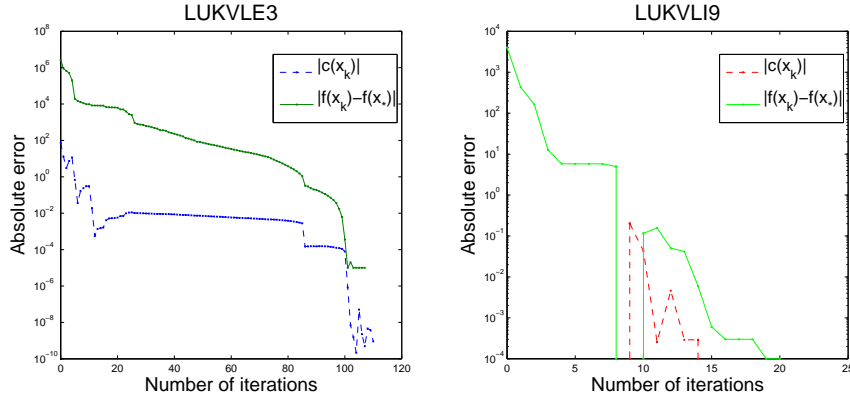
where the vectors  $v_j$  and scalars  $\lambda_j$  are defined by the performed action.

Hence, by a careful implementation of the linear algebra for the updating of the KKT system, the following remarkable result is obtained (cf. [1]):

**Theorem 4 (ZED is DEAD).** *For a partial limited memory approach on a total quasi-Newton-Method with nullspace factorization, the needed memory size and computational effort per iteration are both of order  $\mathcal{O}(nm + nl + l^3)$ .*

### 3 Example

The effectiveness of the presented method has been verified on the two examples LUKVLE3 and LUKVLI9 from the CUTER test set.



Here, the number of constraints is small ( $m = 2, m = 6$ ), whereas the number

<sup>1</sup> using Matlab-like notation

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 nullspace factorization of the KKT system. IPOPT takes 9 and 33 steps, respectively, using full first and second 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  $4 l \cdot n$  operations. A nullspace factorization of the KKT system in the constrained case is reformulated by means of compact representation formulae and efficiently solved using an updated  $QR$  decomposition of the Jacobian.

The “ZED Is DEAD” approach circumvents the necessity of storing the matrix  $Z$  for the solution of the system while reducing the computational effort 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  $m \ll n$ .

First practical runs on the CUTer test set indicate acceptable linear convergence rates even for small  $l < 10$  with drastic reduction in computing time per iteration. A further reduction to order  $\mathcal{O}(m^2/2 + n \cdot l)$  in the storage and operation count is envisioned by a *semi-normal* approach.

## References

1. Bosse T (2009) A derivative-matrix-free NLP solver without explicit nullspace representation. Diplom-Thesis, Humboldt Universität zu Berlin, Berlin
2. Byrd, Richard H, et al. (1994) Representations of quasi-Newton matrices and their use in limited memory methods, *Math. Programming* 63:129–156
3. Gill P, Murray W, Saunders M (2005) SNOPT: An SQP algorithm for large-scale constrained optimization. *SIAM Review* 47(1):99–131
4. Griewank A, Walther A (2008) Evaluating derivatives. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA
5. 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
6. Korzec M (2008) A General Low Rank Update Based Quadratic Programming Solver. Diplom-Thesis, Humboldt Universität zu Berlin, Berlin
7. Nocedal J, Wright S (2006) *Numerical Optimization*, Springer Series in Operations Research, 2nd Edt.
8. Schloßhauer V (2008) Strukturausnutzung und Speicherplatzbegrenzung für hochdimensionale, nichtlineare Optimierung. Diplom-Thesis, Humboldt Universität zu Berlin, Berlin

---

## Index

Limited memory, 3  
ZED is DEAD , 6

Algorithmic differentiation, 1

Compact representation, 3

Compact representation with damping,  
5

NLP, 2

Symmetric rank-one update, 3

Total quasi-Newton method, 2