An NLP Solver with Limited Memory Hessian and Semi-Normal Jacobian Approximation

DIPLOMARBEIT

Humboldt-Universität zu Berlin
Mathematisch-Naturwissenschaftliche Fakultät II
Institut für Mathematik
25. August 2010

Simon Rösel
geb. am 13. April 1985 in Berlin

Betreuer: Prof. Dr. Andreas Griewank
Zweitgutachter: Prof. Dr. Michael Hintermüller

I thank my family, friends and colleagues for their great support and the inspiring and helpful discussions during the course of my studies.

Simon Rösel, August 2010

Contents

Notation

$\mathbf{0}_l$	zero vector with l components
$\mathbb{1}_l$	vector of ones with l components
α	step length parameter
A	approximation of the Jacobian of active constraints
$\mathcal{A}(x)$	index set of active constraints at a point x
B	approximation of the reduced Hessian of the Lagrangian $\nabla_{xx}\mathcal{L}$
$B \succ 0$	B is positive definite
c	constraint mapping $c : \mathbb{R}^n \to \mathbb{R}^M$
$c_{\mathcal{A}}$	constraint mapping consisting only of the active constraints
C_Q	Cholesky factor of a square matrix Q
d	step direction
dim	dimension of a vector space
$\mathcal{E}$	index set of equality constraints
f	objective function $f : \mathbb{R}^n \to \mathbb{R}$
I_n	identity matrix of dimension $n \times n$, $n \in \mathbb{N}$
$\mathcal{I}$	index set of inequality constraints
ι	number of inner loop iterations
l	maximal number of stored update vectors (limited memory)
ker	kernel of a linear mapping
L	semi-normal Cholesky factor
LICQ	Linear independence constraint qualification
$\mathcal{L}$	Lagrangian mapping
m	number of constraints contained in the active set
$m_{\mathcal{E}}$	number of equality constraints
$m_{\mathcal{I}}$	number of inequality constraints
$M = \lvert\mathcal{E}\rvert + \lvert\mathcal{I}\rvert$	total number of constraints
A_+	(low-rank) update on a given matrix A
n	number of variables
$\mathcal{N}(A)$	null space of a matrix A
Ω	feasible set
$OPS(\text{eval}(f))$	number of multiplications to evaluate a function f
$\mathcal{R}(A)$	range space of a matrix A
s	primal step
σ	Lagrange multiplier modification
SOSC	Second order sufficiency criterion
spec(Q)	spectrum of a square matrix Q
Y	matrix with orthonormal columns spanning $\mathcal{R}(A^\top)$
Z	matrix with orthonormal columns spanning $\mathcal{N}(A)$

1 Introduction

This thesis deals with the discipline of Nonlinear Programming as part of the well-established and widely-spread field of Mathematical Optimization. The wish to solve nonlinear programs occurs in various application fields including physics, engineering and economics. Typical application examples comprise the most diverse tasks: portfolio optimization under the condition of a certain risk aversion, optimal control of a large-scale chemical plant or design optimization of an aircraft for best aerodynamic properties. All these applications may share the equivalent mathematical formulation of a nonlinear program but differ in the number of possible variables, constraints or structure. Whereas the first may just give rise to a few variables up to the hundreds, the latter usually requires optimizing over hundreds of thousands of variables, hinging on the parametrization of the given object. Therefore suitable numerical procedures are necessary to find reliable solutions to the underlying real-world problem. Intermediate steps, such as discretizing a partial differential equation (for instance optimization of aerodynamics problems), might be necessary to retrieve the reformulation of the problem as a nonlinear program.

A typical and well-known approach to solve these kinds of problems iteratively, are *Quasi-Newton* (or *Variable-Metric*) methods. Known since the 1950's work[1] by William C. Davidon (Argonne National Laboratories), these methods have been evolving dramatically throughout the last decades leading to different ways of approximating and evaluating the given problem structure depending on closely related mathematical fields such as differentiation techniques and numerical linear algebra. At the same time, research had to be aware of a massive increase in computer performance as well as size and sophistication of the applications. Nowadays problems easily exceed a complexity of more than ten thousand variables. Resulting implementations therefore need to be as economic as possible with the input data they require. This gave rise to the class of large-scale optimization procedures, such as limited memory methods representing a major concern of this thesis. Thereby, the quest for optimal performance on arbitrary problems from within the class of nonlinear programs alongside an efficient organization of the stored data has to serve as a guideline for investigation.

This thesis particularly deals with further extensions to the total Quasi-Newton solver called **L**ow-**R**ank **A**pproximation **M**odification **B**ased **O**ptimizer (LRAMBO) which are especially dedicated to improve its performance on high-dimensional problems.

[1] not published until 1991

2 Background: Constrained Optimization and Algorithmic Differentiation

2.1 Constrained Optimization

The overall goal of this diploma thesis is to solve a Nonlinear Program (NLP), which is defined as follows:

$$\min f(x) \quad \text{s.t.} \quad \begin{cases} c_{\mathcal{E}}(x) = 0 \\ c_{\mathcal{I}}(x) \leq 0, \end{cases} \tag{1}$$

with

$$\begin{aligned} \mathcal{E} &= \{1, \ldots, m_{\mathcal{E}}\}, \\ \mathcal{I} &= \{m_{\mathcal{E}} + 1, \ldots, m_{\mathcal{E}} + m_{\mathcal{I}} = M\}, \end{aligned}$$

where the objective function $f : \mathbb{R}^n \to \mathbb{R}$ and the constraint mapping $c : \mathbb{R}^n \to \mathbb{R}^M$ are required to be twice continuously differentiable functions. In the following, Ω shall denote the feasible set defined by the constraints given in (1):

$$\Omega := \{x \in \mathbb{R}^n : c_{\mathcal{E}}(x) = 0, c_{\mathcal{I}}(x) \leq 0\}.$$

Furthermore, the active set $\mathcal{A}(x)$ of a feasible point x is defined as

$$\mathcal{A}(x) := \mathcal{E} \cup \{i \in \mathcal{I} : c_i(x) = 0\}$$

and throughout this thesis, $m := |\mathcal{A}(x_*)|$ shall denote the number of active constraints at a corresponding solution x_* to the NLP (1).

In order to define the notion of a constraint qualification, it is necessary to introduce certain cones associated with the underlying feasible set Ω and a fixed feasible point $x \in \Omega$.

Definition 2.1 (Important cones)**.**

(i) The tangent cone $\mathcal{T}_\Omega(x)$ on Ω at x is defined by the following relation:

$$u \in \mathcal{T}_\Omega(x) :\Longleftrightarrow \exists\, (x_k)_{k\in\mathbb{N}} : x_k = x + t_k u + o(t_k) : \begin{cases} t_k \downarrow 0 \\ c_i(x_k) = 0 \;\forall\, i \in E \\ c_i(x_k) \leq 0 \;\forall\, i \in \mathcal{I}. \end{cases}$$

In other words, $(x_k)_{k\in\mathbb{N}}$ must be a feasible sequence.

(ii) The cone of linearly feasible directions $\mathcal{F}_\Omega(x)$ is defined by the following relation:

$$u \in \mathcal{F}_\Omega(x) :\Longleftrightarrow \begin{cases} Dc_i(x)u = 0 \;\forall\, i \in E \\ Dc_i(x)u \leq 0 \;\forall\, i \in \mathcal{I} \cap \mathcal{A}(x) \end{cases}$$

We remark that the definition of $\mathcal{F}_\Omega$ does not include inactive inequalities, as a sufficiently small step in an arbitrary direction will always guarantee validity of strict inequalities. Using first order Taylor expansion, it is not hard to show that $\mathcal{T}_\Omega(x) \subset \mathcal{F}_\Omega(x)$. However, the reverse inclusion is not always fulfilled. The subsequent definition aims at filling this gap.

Definition 2.2 (Constraint qualification)**.**
A condition on the constraint mapping c which ensures

$$\mathcal{T}_\Omega(x) = \mathcal{F}_\Omega(x),$$

is called a constraint qualification in $x \in \Omega$.

The most famous examples, among the quite large number of constraint qualifications, will be stated in the following:

Remark 2.3 (Specific constraint qualifications)**.**
The following conditions all represent constraint qualifications in the sense of Definition 2.2:

(i) Affine-linear constraints: all $c_i, i = 1, ..., M$, *are affine:*

$$c_i(x) = a_i^\top x + b_i, \quad \text{with } a_i \in \mathbb{R}^n, b_i \in \mathbb{R}.$$

(ii) Linear independence constraint qualification (LICQ): the gradients of the active constraints

$$(\nabla c_i(x))_{i \in \mathcal{A}(x)}$$

form a linearly independent system.

(iii) Mangasarian-Fromovitz constraint qualification (MFCQ):

$$\begin{aligned} \operatorname{rank}(\nabla c_i(x))_{i\in\mathcal{E}} &= m_\mathcal{E} \\ \exists\, u \in \mathbb{R}^n \quad &: \quad \begin{cases} \nabla c_i(x)^\top u = 0 \ \ \forall\, i \in \mathcal{E} \qquad \wedge \\ \nabla c_i(x)^\top u < 0 \ \ \forall\, i \in \mathcal{A}(x) \cap \mathcal{I}. \end{cases} \end{aligned}$$

It should be remarked, that MFCQ is an often used generalization to LICQ.

The fundamental theoretical basis of a large number of NLP solvers, i.e. algorithms dedicated to find a solution to (1), are the Karush-Kuhn-Tucker (KKT) conditions:

Theorem 2.4 (KKT-conditions)**.**
Let x_* *be a local minimizer to the NLP* (1)*. If* x_* *fulfills a constraint qualification then* $\exists\, \lambda_* \in \mathbb{R}^M$*:*

$$\begin{cases} \nabla f(x_*) + \sum_{i\in\mathcal{E}} \lambda_*^i \nabla c_i(x) + \sum_{i\in\mathcal{I}} \lambda_*^i \nabla c_i(x) = 0, \\ x_* \in \Omega, \\ \lambda_*^i \geq 0 \ \ \forall\, i \in \mathcal{I}, \\ \lambda_*^i c_i(x_*) = 0 \ \ \forall\, i \in \{1, ..., M\}. \end{cases} \tag{2}$$

Proof. A proof of this essential result can for instance be found in [20]. □

A not necessarily uniquely determined solution (x_*, λ_*) to the KKT-conditions (2) is called a *KKT-point*. If additionally LICQ holds at a given local solution $x_* \in \mathbb{R}^n$, then the associated *Lagrange multiplier* vector $\lambda_* \in \mathbb{R}^M$ is uniquely determined.

The second condition is the *feasibility condition* required for a solution to our original problem. With the help of the *Lagrangian function*

$$\begin{aligned} \mathcal{L} : \mathbb{R}^n \times \mathbb{R}^M &\to \mathbb{R} \\ \mathcal{L}(x, \lambda) &:= f(x) + \sum_{i \in \mathcal{E} \cup \mathcal{I}} \lambda^i c_i(x) \end{aligned}$$

and the fourth condition (the so-called *complementarity condition*), the first KKT-condition can be reformulated as

$$\nabla_x \mathcal{L}(x_*, \lambda_*) = \nabla f(x_*) + \sum_{i \in \mathcal{A}(x_*)} \lambda_*^i \nabla c_i(x_*) = 0.$$

The complementarity condition demands that Lagrange multipliers associated with non-active constraints vanish. Another even stronger condition, named *strict complementarity*, requires additionally that Lagrange multipliers associated with active constraints are nonzero. Many NLP solvers presume this strict form of complementarity, in order to correctly identify the active set at the solution.

Note that the KKT-conditions are not sufficient to characterize an optimal point. In order to derive a suitable sufficient optimality criterion, we consider another important cone.

Definition 2.5 (Critical tangent cone)**.**
Let (x_*, λ_*) be a KKT-point. Define the critical tangent cone as follows:

$$\mathcal{C}(x_*) := \{u \in \mathcal{F}(x_*) : \nabla c_i(x_*)^\top u = 0 \ \ \forall\, i \in \mathcal{A}(x_*) \cap \mathcal{I} : \lambda_*^i > 0\}$$

With the help of the Langrangian function, it is easy to see that the critical tangent cone contains those linearly feasible directions for which we cannot decide from first order derivative information alone, whether the objective function decreases along the corresponding direction or not. Notably, this cone enables formulation of a second order sufficiency criterion (SOSC):

Theorem 2.6 (SOSC)**.**
Let (x_*, λ_*) be a KKT-point satisfying

$$\langle u, \nabla_{xx}^2 \mathcal{L}(x_*, \lambda_*) u \rangle > 0 \ \ \forall\, u \in \mathcal{C}(x_*) \setminus \{0\}. \tag{3}$$

Then x_* is a strict local minimizer to the NLP (1)*.*

Proof. See [20]. □

In the case where both, the LICQ and the strict complementarity hold, we have

$$\mathcal{C}(x_*, \lambda_*) = \mathcal{C}(x_*) = \mathcal{N}\left(\left[\nabla c_i(x_*)^\top\right]_{i\in\mathcal{A}(x_*)}\right),$$

and one can conclude that the SOSC simplifies to the verification of the positive definiteness of the *reduced Hessian*

$$Z^\top \nabla^2_{xx}\mathcal{L}(x_*, \lambda_*)Z,$$

where the columns of $Z \in \mathbb{R}^{n\times(n-m)}$ shall constitute a basis for the null space of the Jacobian of active constraints $\left[\nabla c_i(x_*)^\top\right]_{i\in\mathcal{A}(x_*)}$.

A large number of contemporary algorithms employ a *working set strategy* to correctly predict the a priori unknown active set $\mathcal{A} := \mathcal{A}(x_*)$ at a solution x_*. Once $\mathcal{A}$ is correctly identified, the problem of finding a local solution to the general equality- *and* inequality-constrained NLP (1) can locally be reformulated as a corresponding *reduced equality-constrained problem*:

$$\begin{cases} \min f(x) \quad \text{s.t.} \\ c_{\mathcal{A}}(x) = 0, \quad \text{with} \quad c_{\mathcal{A}} := (c_i)_{i\in\mathcal{A}}. \end{cases} \tag{4}$$

Thereby, the inequality constraints and thus the complementarity condition disappear and one exploits the first two KKT-conditions to generate iterates which converge to a zero of $\nabla_{x,\lambda}\mathcal{L}$, where

$$\begin{aligned} \mathcal{L} : \mathbb{R}^n \times \mathbb{R}^m &\to \mathbb{R} \\ \mathcal{L}(x, \lambda) &= f(x) + \lambda^\top c_{\mathcal{A}}(x) \end{aligned}$$

denotes from now on the Lagrangian function associated with the reduced equality-constrained problem. Thus the initial constrained problem (4) can be turned into the numerically tractable problem of finding a zero to the system of $(n + m)$ nonlinear equations given by

$$F(x, \lambda) := \begin{bmatrix} \nabla_x\mathcal{L}(x, \lambda) \\ \nabla_\lambda\mathcal{L}(x, \lambda) \end{bmatrix} = \begin{bmatrix} \nabla f(x) + \nabla c_{\mathcal{A}}(x)\lambda \\ c_{\mathcal{A}}(x) \end{bmatrix} = 0, \tag{5}$$

or, in other words, (4) has been turned into an unconstrained problem via the Lagrangian $\mathcal{L}$.

Naturally, this gives rise to classical root-finding algorithms, such as Newton-type methods. However, it should be mentioned that even if an appropriate algorithm converges to a zero of F, the limit does not necessarily have to be a minimum of f on Ω. Unless additional properties (like convexity) are given, it might as well be a maximum or a saddle point. Hence, verification of the desired minimality property by a suitable suffiency criterion, such as (3), is necessary.

2.2 Algorithmic Differentiation

Computing derivatives is indispensable in almost every optimization procedure. Of course, this also holds true for the total Quasi-Newton framework, which will be discussed in the forthcoming sections. Excluding the analytical approach, there are three major techniques to compute derivatives and associated quantities: *Numerical* (i.e. Divided Differences), *Symbolic* and *Algorithmic Differentiation* (AD). This section shall suggest usage of AD and give an insight into basic complexity results which will influence later considerations on Quasi-Newton methods.

Consider a differentiable scalar function $f : \mathbb{R} \to \mathbb{R}$. Numerical Differentiation proceeds by approximating its derivative $f'(x)$ for a given $x \in \mathbb{R}$, by the difference quotient

$$\frac{f(x+h) - f(x)}{h},$$

by setting $h \approx 0$. Choosing h too small may cause severe problems due too cancellation errors whereas a too big h implies significant truncation errors. The optimal choice of h however can be rather tedious and once having obtained such an optimal value, truncation errors are still present.

Making use of consecutive chain rule application, Symbolic Differentiation merely generates an entirely symbolic algebraic expression of the derivative before evaluating the resulting expression at a desired point. This may lead to an excessive expression swell in addition to a distinct loss in efficiency by ignoring common subexpressions as a well-known drastic example by Speelpenning shows, cf. [23].

Like Symbolic Differentiation, Algorithmic Differentiation is also based on consecutive application of the chain rule. However, symbolic expressions are never generated, the chain rule is rather applied directly to numerical values originating from the intermediate quantities of an evaluation procedure for the underlying function. Applying AD to compute derivative-related expressions thus rather yields a program than a formula. Moreover, in contrast to Divided Differences, Algorithmic Differentiation does not incur truncation errors and returns exact[2] derivative values! To calculate Jacobian-vector and Jacobian-transposed vector products, we use the *forward* and *reverse mode*, respectively. The latter can in turn be split into a *forward sweep* and a *return sweep*. The forward sweep is necessary to compute the *elemental partials* which are needed to compute the *intermediate adjoints* for the return sweep. Moreover, the *vector mode* of AD enables simultaneous tangent and gradient propagation. In terms of accuracy, reliability, efficiency and speed, the superiority of AD is further manifested by the subsequent results on computational complexity bounds. Instead of an elaborate discussion, we merely state major results of AD application and refer to [12] for details.

For simplicity reasons, the number of multiplications (symbolized by OPS) shall serve as a naïve complexity measure for our predominating task of evaluating ('eval') Jacobian-vector products and gradients. It is further assumed that a

[2] disregarding round-off errors

given differentiable function $F : \mathbb{R}^n \to \mathbb{R}^m$ can be safely evaluated at $x \in \mathbb{R}^n$ by an underlying evaluation procedure, which can be represented by a composition of continuously differentiable *elemental functions* φ_i. These can be imagined to be taken from a given library Ψ:

$$\varphi_i \in \Psi := \{c, \pm, *, \log, \exp, \sin, \cos, ...\},$$

which should comprise all intrinsic operators. In the following, we will adhere to the usual notation for tangent and gradient propagation.

Theorem 2.7 (Complexity of derivative calculations using AD)**.**
The computational cost of computing Jacobian-vector or Jacobian-transposed vector products is a task of linearly bounded complexity w.r.t. the number of operations for the function evaluation. More precisely, we obtain the following dimension-invariant bounds for the multiplicative measure OPS:

(i) $OPS(\text{eval}(F'(x)\dot{x})) \leq 3 \cdot OPS(\text{eval}(F(x)))$

(ii) $OPS(\text{eval}(\bar{y}^\top F'(x))) \leq 3 \cdot OPS(\text{eval}(F(x)))$,

with $x \in \mathbb{R}^n, \bar{y} \in \mathbb{R}^m$*. Applying the vector mode for simultaneous propagation of tangents or gradients, we obtain*

(iii) $OPS(\text{eval}(F'(x)\dot{X})) \leq (1 + 2p) \cdot OPS(\text{eval}(F(x)))$

(iii) $OPS(\text{eval}(\bar{Y}^\top F'(x))) \leq (1 + 2q) \cdot OPS(\text{eval}(F(x)))$,

for matrices $\dot{X} \in \mathbb{R}^{n \times p}$ *and* $\bar{Y} \in \mathbb{R}^{m \times q}$.

Proof. By considering the evaluation procedure for F and the corresponding derived evaluation procedure, the bounds can be obtained from the operations count for differentiating and evaluating the elemental functions φ_i. Among those, the multiplication operator ($\varphi_i = *$) requires the highest operations count, i.e. one multiplication for the evaluation of φ_i and two for its derivative, cf. [12]. □

Unfortunately, the task of computing full Jacobians does not have bounded complexity. Instead it holds

$$OPS(\text{eval}(F'(x))) \sim \min(m, n) \cdot OPS(\text{eval}(F(x))).$$

Consequently, exact determination of full derivative matrices will be avoided by the total Quasi-Newton solver presented subsequently.

Similar results hold true for more realistic complexity measures additionally accounting for memory stores/fetches, additions, nonlinear operations and other work associated with the evaluation procedure. By including these criteria, the reverse mode multiplier becomes slightly higher than the multiplier of the forward mode. We refer again to [12] for a detailed complexity analyis of AD calculations.

3 From Quasi-Newton to LRAMBO

In this chapter, we pick up on the issue of iteratively finding a solution $x_* \in \mathbb{R}^n$ to the system of nonlinear equations, given by a function $F : \mathbb{R}^n \to \mathbb{R}^n$ which is assumed to be at least once continuously differentiable, i.e. $F \in \mathcal{C}^1(\mathbb{R}^n, \mathbb{R}^n)$ and $F(x_*) = 0$.

3.1 Theoretical Background

In case of a *nondegenerate* root x_*, i.e. $F'(x_*)$ is nonsingular, it is well known that Newton's method, under the local Lipschitz-continuity of F, converges locally quadratically to x_*. In this context, *locally* refers to the starting point $x_0 \in \mathbb{R}^n$ which has to be chosen sufficiently 'close' to a zero x_*.

In optimization, the overall incentive for Quasi-Newton methods is the aspiration to design a method that combines the behavior of the steepest descent method in view of global convergence and the fast local convergence of Newton's method, without the need to compute exact (and generally costly) Jacobians $F'(x)$.

Quasi–Newton methods embedded in a line search procedure procede by the following iteration rule:

$$x_{k+1} = x_k + \alpha_k d_k, \tag{6}$$

where α_k denotes the step size determined by a properly chosen step size strategy, and d_k is the actual search direction given by the condition

$$M_k d_k = -F(x_k),$$

where M_k denotes the *Quasi-Newton matrix* which is supposed to approximate the real Jacobian at the current iterate x_k, i.e. $M_k \approx F'(x_k)$. As a special case, Newton's method is defined by setting $M_k := F'(x_k)$. From now on, the actual step from x_k to x_{k+1} will be denoted by $s_k := \alpha_k d_k$.

Secant condition and rate of convergence. The question of how to choose the matrices M_k naturally arises. For that reason it should be emphasized that the quality of the approximation is directly related to the so-called *secant condition* or *Quasi-Newton equation*

$$M_{k+1} s_k = y_k := F(x_{k+1}) - F(x_k). \tag{7}$$

This condition is inspired by the first order Taylor expansion of F at x_{k+1} as described for instance in the classical work by Dennis and Moré [9]. In the case, where F originates from an optimization problem, one may demand equivalently that the objective function and its quadratic approximation share the same slope at the current iterate as well as the previous one. The following remarkable result, in particular characterization (iv), established by Dennis and Moré [8], further bolsters imposition of (7):

Theorem 3.1 (Characterization of superlinear convergence)**.**
Let $F \in \mathcal{C}^{1,1}(\mathbb{R}^n, \mathbb{R}^n)$. Suppose there exists an $x_* \in \mathbb{R}^n : F(x_*) = 0$ and $\det F'(x_*) \neq 0$. Assume a sequence $(x_k)_{k\in\mathbb{N}}$ with $x_{k+1} \neq x_k$ and $x_k \neq x_*$ generated by iteration (6) with nonsingular matrices M_k and unit step lengths $a_k \equiv 1$, converges to x_*. Further suppose that (M_k) as well as (M_k^{-1}) are uniformly bounded, i.e. $\sup_k \|M_k\| < \infty$ and $\sup_k \|M_k^{-1}\| < \infty$, respectively. Then the following assertions are equivalent:

(i) x_k *converges superlinearly to* x_**,*

(ii) $\lim_k \frac{\|s_k - s_k^N\|}{\|s_k\|} = 0, \quad$ *with* $\quad s_k^N := -F'(x_k)^{-1}F(x_k)$*,*

(iii) $\lim_k \frac{\|(F'(x_k) - M_k)s_k\|}{\|s_k\|} = 0$*,*

(iv) $\lim_k \frac{\|y_k - M_k s_k\|}{\|s_k\|} = 0.$

Proof. See [8]. □

The key observation is that M_k must 'only' be a good approximation of the exact Jacobian F_k' (alternatively also F_*') *along* the current step s_k, or, equivalently, the Quasi-Newton direction shall approximate the Newton direction s_k^N as accurately as possible. In particular, the matrix sequence $(M_k)_{k\in\mathbb{N}}$ does not need to converge to F_* to obtain superlinear convergence.

In the context of unconstrained optimization, F equals the gradient of an arbitrary twice differentiable function $f : \mathbb{R}^n \to \mathbb{R}$, i.e. $F := \nabla f$. Thus we have

$$d_k = -M_k^{-1}\nabla f(x_k) \text{ and } M_k \approx \nabla^2 f(x_k),$$

such that the basic scheme of any Quasi-Newton procedure can be summarized as follows:

Algorithm 1 (Quasi-Newton)**.**

```
choose  x_0 ∈ ℝ^n, M_0 ∈ ℝ^{n×n}, ε > 0
while   ‖∇f(x_k)‖ > ε
  determine d_k = −M_k^{−1} ∇f(x_k)
  determine α_k
  set x_{k+1} = x_k + a_k d_k
  update M_k → M_{k+1}, k → k + 1
end
```

Thereby, x_0 denotes a 'suitable' starting point (compare remarks on global convergence below), M_0 the initial Jacobian approximation, and $\varepsilon > 0$ determines the preferred accuracy of the computed root. Additionally, one should impose an upper bound on the total number of iterations. M_0 is usually chosen sparse,

for instance a multiple of the identity capturing the scaling of the exact Hessian $\nabla^2_{xx} f(x_0)$.

In contrast to the nonlinear equations case, where nonsymmetric approximations are quite reasonable, we require M_k to be symmetric in order to reflect symmetry of the exact Hessian. Combining the secant condition with the requirement that M_k changes 'as little as possible' (with regard to a certain norm) throughout the iteration leads to the so-called *least-change secant updates*, such as BFGS (see section 3.2.1). Positive definiteness is also desired, in order to obtain a descent direction with respect to f. Moreover, the secant condition implies the *positive curvature condition*

$$s_k^\top y_k > 0,$$

provided that positive definiteness of M_{k+1} is given. Positive curvature can also be ensured by imposing certain line search conditions, for example those proposed by Wolfe, see for instance the textbook by Nocedal and Wright [20].

As far as the choice of the starting point x_0 is concerned, we have to remark that there is no general global convergence result for typical Quasi-Newton methods such as BFGS or SR1, in that convergence to a minimizer x_* from an arbitrary starting point x_0 cannot be guaranteed. In case of convexity, a global convergence result for BFGS (and in a slightly weaker form also for SR1) can be established, cf. [20]. However, one can give a rather theoretical result which is valid for all Quasi-Newton iterations.

Proposition 3.2 (Quasi–Newton: global convergence)**.**
Let $f \in \mathcal{C}^{1,1}(\mathbb{R}^n, \mathbb{R})$. Consider the preceding Algorithm 1 and assume an effective line search (e.g. Wolfe line search) is performed. Suppose the Quasi-Newton matrices M_k are symmetric positive definite and their condition numbers $\kappa(M_k)$ are uniformly bounded. Then Algorithm 1 is globally convergent, in the sense that all cluster points of the sequence $(x_k)_{k\in\mathbb{N}}$ generated by Algorithm 1 are stationary points.

Proof. By virtue of Theorem A.8, it suffices to prove gradient-relatedness of the search directions d_k. Let $\|.\| = \|.\|_2$ denote the Euclidean norm and consider

$$\begin{aligned}
\cos\varphi_k &= -\frac{\nabla f^\top d}{\|\nabla f\|\|d\|} \\
&= \frac{d^\top M_k d}{\|M_k d\|\|d\|} \\
&\geq \frac{\tilde{d}^\top M_k \tilde{d}}{\|M_k\|}, \quad \tilde{d} := d/\|d\| \\
&\geq \frac{\lambda_1(M_k)}{\|M_k\|}, \quad \lambda_1 := \min \operatorname{spec}(M_k) \\
&\overset{M_k \succ 0}{=} \frac{1}{\|M_k\|\|M_k^{-1}\|} = \frac{1}{\kappa(M_k)}.
\end{aligned}$$

Consequently, uniformly bounded condition numbers of the Quasi-Newton matrices M_k imply gradient-relatedness. □

3.2 Total Quasi-Newton

In this section we leave the unconstrained case in favor of the constrained case. Therefore consider the step determination in Newton's method applied to F (5) arising from the equality-constrained NLP (4):

$$\begin{bmatrix} d_k \\ \sigma_k \end{bmatrix} := - \begin{bmatrix} \nabla_{xx}\mathcal{L}(x_k, \lambda_k) & \nabla c_{\mathcal{A}}(x_k) \\ \nabla c_{\mathcal{A}}(x_k)^\top & 0 \end{bmatrix}^{-1} \begin{bmatrix} \nabla_x \mathcal{L}(x_k, \lambda_k) \\ c_{\mathcal{A}}(x_k) \end{bmatrix}.$$

Despite of the quadratic rate of convergence in a vicinity of the solution (x_*, λ_*), this approach is only of theoretical interest, as computation of $(d_k, \sigma_k) \in \mathbb{R}^{n \times m}$ would require determination and factorization of exact Hessians and Jacobians, which can be assumed prohibitively expensive even if Algorithmic Differentiation is used. Especially in large-scale optimization ($n \geq 10000$), it is essential to avoid costly full derivative matrices.

Conventional *mixed Quasi-Newton* methods procede by replacing the reduced Hessian $\nabla_{xx}\mathcal{L}(x_k, \lambda_k) \in \mathbb{R}^{n \times n}$ by a given approximation B, such as BFGS or SR1 (see section 3.2.1), whereas expressions involving $\nabla c_{\mathcal{A}}(x_k)$ are determined by more or less reliable differentiation techniques such as divided differences. Major practical drawback is again the high cost of evaluating and factorizing the (possibly large) Jacobians $c'_{\mathcal{A}} \in \mathbb{R}^{m \times n}$ as well as the cubic effort which is necessary to solve the *KKT-system*

$$\begin{bmatrix} B & c'_{\mathcal{A}}(x)^\top \\ c'_{\mathcal{A}}(x) & 0 \end{bmatrix} \begin{bmatrix} d \\ \sigma \end{bmatrix} = - \begin{bmatrix} g(x, \lambda) \\ c_{\mathcal{A}}(x) \end{bmatrix}.$$

Hereby the mapping $g : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ is defined by

$$g(x, \lambda) := \nabla_x \mathcal{L}(x, \lambda) = \nabla f(x) + \nabla c_{\mathcal{A}}(x)\lambda,$$

and for notational convenience, the index k has been omitted such that x and x_+ denote the current and the subsequent iterate, respectively. NLP solvers which employ a working set strategy, usually solve the KKT-system with varying working set and thus varying constraint number $m \in \{m_{\mathcal{E}}, ..., M\}$ instead of a fixed active set $\mathcal{A}$, which is unknown in the first place. For notational convenience, we will however stick to the current notation.

$g(x, \lambda) \in \mathbb{R}^n$ can be rather inexpensively evaluated by making use of the reverse mode of Algorithmic Differentiation, which essentially means a small multiple of the cost of evaluating the objective and the constraint functions.

In a *full* or, more accurately, *total* Quasi-Newton approach, both $\nabla_{xx}\mathcal{L}(x, \lambda)$ *and* $c'_{\mathcal{A}}(x)$ are replaced by certain approximations B and A, respectively, aiming at a distinct reduction in the per-iteration-cost. To keep the linear algebra cost as low as possible, one usually starts with sparse initial approximations B_0 and A_0, and adds certain *low-rank secant updates* to B and A. Several popular updating formulae will be presented in sections 3.2.1 and 3.2.2.

Finally, the entirely approximative KKT-system has the form

$$\begin{bmatrix} B & A^\top \\ A & 0 \end{bmatrix} \begin{bmatrix} d \\ \sigma \end{bmatrix} = - \begin{bmatrix} g(x,\lambda) \\ c_{\mathcal{A}}(x) \end{bmatrix}, \tag{8}$$

where the right side of the equation can be determined *exactly*, i.e. without incurring truncation errors, by Algorithmic Differentiation.

A system which has the structure of (8) and fulfills the additional conditions

- $\diamond$ B is symmetric positive definite;
- $\diamond$ A has full row rank,

is often called *equilibrium system* (see Golub and Van Loan [11] for a brief summary) and occurs in several physical applications (e.g. electrical networks) as well as in the finite element method, see for instance Vavasis [26] for details.

In the following theorem, the fundamental link between equilibrium systems and quadratic programs of the form

$$\begin{cases} \min\limits_{d} \frac{1}{2} d^\top B d + g^\top d \\ \text{s.t. } Ad + c = 0 \end{cases} \tag{9}$$

will be established. Owing to its importance, a combination of the proofs described in [4] and [20] will be presented, too.

Theorem 3.3 (SQP)**.**
Consider the equilibrium system (8) *with coefficient matrix*

$$M := \begin{bmatrix} B & A^\top \\ A & 0 \end{bmatrix}.$$

Then it holds that:

(i) M is nonsingular if and only if A is surjective and $Z^\top BZ$ is nonsingular for a matrix Z with linearly independent columns spanning $\mathcal{N}(A)$. In this case, (8) *has a unique solution $(d, \sigma)^\top$.*

(ii) If A is surjective and $Z^\top BZ$ is even positive definite, then d is the unique global solution to the QP (9)*.*

Proof. (i) ("$\Longrightarrow$") Surjectivity of A is given by surjectivity of M. Assume $Z^\top BZu = 0$ for $u \in \mathbb{R}^{n-m}$. Define $p := Zu \in \mathcal{N}(A)$. Then $Bp \in \mathcal{N}(Z^\top)$ by assumption. But $\mathcal{N}(Z^\top) = \mathcal{R}(A^\top)$, thus there is a $\mu \in \mathbb{R}^m : M \begin{bmatrix} p \\ \mu \end{bmatrix} = 0$.
Nonsingularity of M yields $p = 0 = u$, since Z is injective by definition.
("$\Longleftarrow$") Let $(p, \mu) \in \mathcal{N}(M)$. It follows

$$Ap = 0 \qquad \text{and} \qquad Bp + A^\top \mu = 0. \tag{10}$$

Thus $p \in \mathcal{N}(A)$ and we have $p = Zu$ for suitable $u \in \mathbb{R}^{n-m}$. (10) also implies $0 = Z^\top Bp = Z^\top BZu$, thus $u = 0$ by assumption and consequently $p = 0$. Now, the right equation in (10) yields $\mu = 0$ by surjectivity of A.

(ii) Let $\bar{d}$ be any other feasible point of the QP (9). Let $q(.)$ denote the quadratic objective function. Define $r := d - \bar{d}$. Now,

$$\begin{aligned} q(\bar{d}) &= q(d-r) = q(d) + \frac{1}{2} r^\top Br - r^\top Bd - r^\top g \\ &= q(d) + \frac{1}{2} r^\top Br \underbrace{-r^\top(-A^\top\sigma - g) - r^\top g}_{=(Ar)^\top\sigma = 0}, \end{aligned}$$

since (d, σ) is a solution to (8) and $Ar = 0$. We further write $r = Zu$ with suitable $u \in \mathbb{R}^{n\times(n-m)}$ and conclude

$$q(\bar{d}) = q(d) + \frac{1}{2} r^\top Br = q(d) + \frac{1}{2} u^\top Z^\top BZu.$$

Positive definiteness of $Z^\top BZ$ yields the assertion. □

As a consequence, computing the step direction d according to (8) is, under reasonable conditions on the approximations, well-defined and equivalent to minimizing the (inexact) quadratic approximation of the Lagrangian considered as a function of x under (inexact) linearized constraints. This is the fundamental link between Newton-type and SQP (*successive quadratic programming*) methods. Fundamental differences might occur in the handling of the inequalities. A typical SQP algorithm might be dedicated to solving inequality constrained quadratic problems at each iterate, incorporating different approaches, such as inner point methods. By contrast, the step calculation of the LRAMBO-solver which is going to be presented here, is just based on repeatingly solving the KKT-system (8) for different choices of the working set, cf. section 3.3.

3.2.1 Hessian Approximations

As suggested above, the approximation of the reduced Hessian of the Lagrangian $\nabla_{xx}\mathcal{L}(x, \lambda)$ shall satisfy the *Hessian secant condition* (HSC) which now reads

$$B_+ s = w \tag{11}$$

where

$$\begin{cases} s = x_+ - x = \alpha d \\ w := \nabla_x \mathcal{L}(x_+, \lambda) - \nabla_x \mathcal{L}(x, \lambda). \end{cases}$$

Of course, symmetry as well as positive definiteness on the null space of the Jacobian of the constraints have to be maintained.

Symmetric rank-one (SR1) updates. The SR1-update formula is given by

$$B_+ = B + \frac{(w - Bs)(w - Bs)^\top}{(w - Bs)^\top s}. \tag{12}$$

It can easily be verified, that the SR1-update is the only symmetric rank-one update fulfilling the secant condition (11).

Problems will occur when the denominator in (12) approaches zero. Therefore, one distinguishes the following cases:

(i) $(w - Bs)^\top s \neq 0$: SR1-update viable.

(ii) $w = Bs$: Set $B_+ = B$, which already satisfies the succeeding secant condition.

(iii) $(w \neq Bs) \ \wedge \ (w - Bs)^\top s = 0$: In this case practice shows that simply skipping the update is a reasonable strategy.

Numerically, the first condition is often verified by checking $|s^\top(w - Bs)| \geq r\|s\|\|w - Bs\|$ for $r \approx 0$. Otherwise one can assume that B already correctly approximates the curvature along s .

In the case where SR1 approximates the Hessian of a quadratic function, the exact Hessian will be obtained after n steps, provided the latter are linearly independent and the SR1-approximations are well-defined. This follows from the fact that SR1 fulfills the secant condition along *all* previous directions (*heredity*):

$$B_k s_j = w_j \quad \forall\, j \leq k - 1.$$

This holds only in the quadratic case.

Although SR1 often produces good approximations to the exact Hessian (see [20]), it does not retain positive definiteness of the full Hessian approximation or its projection onto the nullspace of the constraints. (Thus being able to reflect possible indefiniteness of the exact Hessian.) One of the most common techniques to overcome this drawback is *damping* the update by a suitable $\varepsilon > 0$:

$$B_+ = B + \varepsilon \frac{(w - Bs)(w - Bs)^\top}{(w - Bs)^\top s}. \tag{13}$$

However the choice of ε has to be optimized at each iterate and might be quite difficult. Since this thesis is mainly concerned with the study of the large-scale case, limited memory versions of B will be contemplated. How their structure can be exploited to ensure positive definiteness will be shown in section 7.

BFGS. Another famous Hessian update is given by the *Broyden-Fletcher-Goldfarb-Shanno (BFGS)* formula

$$B_+ = B - \frac{Bss^\top B}{s^\top Bs} + \frac{ww^\top}{w^\top s}, \tag{14}$$

which preserves symmetry of B. The inverse BFGS-update B_+^{-1} can be characterized as the minimizer of $\tilde{B} \mapsto \|\tilde{B} - B^{-1}\|_W$ (where $\|.\|_W$ denotes a special weighted Frobenius norm, see [20]) amongst all symmetric matrices $\tilde{B} \in \mathbb{R}^{n\times n}$ fulfilling the *inverse secant condition* $\tilde{B}w = s$. The BFGS-update thus belongs to the class of *least-change secant updates*. In contrast to the SR1-update, BFGS always ensures positive definiteness of B_+, provided B itself is positive definite and the curvature condition $w^\top s > 0$ is fulfilled. Restrictions on the line search, such as the Wolfe conditions, may enforce positive curvature.

3.2.2 Jacobian Approximations

Unlike the Hessian case, secant updating of generally non-symmetric Jacobians is not that common and does not even occur in recent textbooks such as [20]. Besides reducing the cost of solving the KKT-system (8), the main goal consists of designing a secant update of the constraint Jacobian which is invariant with respect to linear transformations in order to simplify scaling or preconditioning of the given problem. To obtain suitable design criteria for efficient low-rank updating one imposes a 'total' Quasi-Newton secant condition, which means requiring the secant condition to hold for the entire nonlinear system defined by F (cf. Theorem 3.1 and (5)):

$$\begin{bmatrix} B_+ & A_+^\top \\ A_+ & 0 \end{bmatrix} \begin{bmatrix} s \\ \sigma \end{bmatrix} = \begin{bmatrix} \nabla_x \mathcal{L}(x_+, \lambda_+) - \nabla_x \mathcal{L}(x, \lambda) \\ c_{\mathcal{A}}(x_+) - c_{\mathcal{A}}(x) \end{bmatrix}.$$

From the lower equation, one immediately derives the *direct secant condition*

$$A_+ s = y = c_{\mathcal{A}}(x_+) - c_{\mathcal{A}}(x). \tag{15}$$

Considering the first equation and employing the secant condition (for the reduced Hessian) $B_+ s = w$, we obtain the *adjoint secant condition*

$$\sigma^\top A_+ = \mu^\top \tag{16}$$

where $\mu^\top := \sigma^\top c'_{\mathcal{A}}(x_+)$. Note that μ can be calculated rather economically via g, i.e.

$$\mu = g(x_+, \lambda_+) - g(x_+, \lambda).$$

Nevertheless, the two conditions on the Jacobian approximation are not exactly consistent, unless the constraint function c is affine. The idea of taking the adjoint condition into account has been set forth by Griewank and Walther [13] and has led to the so-called 'two-sided-rank-one' update (TR1), which has been proposed at first in [13].

TR1. The TR1-update represents a generalization of the SR1-update and is defined as follows:

$$A_+ = A + \frac{(y - As)(\mu^\top - \sigma^\top A)}{\mu^\top s - \sigma^\top As}. \tag{17}$$

This is the *direct* version of the TR1-update formula, which obviously fulfills the direct secant condition (15) exactly and the adjoint secant equation up to a term of order $\mathcal{O}(\|\sigma_k\|\|s_k\|)$. Replacing $\mu^\top s$ in the denominator by $\sigma^\top y$, one obtains the *adjoint* TR1-update

$$A_+ = A + \frac{(y - As)(\mu^\top - \sigma^\top A)}{\sigma^\top y - \sigma^\top As}.$$

Analogously, the adjoint TR1-update fulfills the adjoint secant condition (16) and the direct secant condition up to order $\mathcal{O}(\|\sigma_k\|\|s_k\|)$. In the case where c is affine, the two updates are equal. In general it can be shown under the Lipschitz-continuity of ∇c, that the discrepancy in the choice of the denominator term is only of order $\mathcal{O}(\|\sigma_k\|\|s_k\|^2)$.

Instabilities. The drawback of the TR1-update lies in the instabilities caused by relatively small δ. In the framework of [15], a determinant control technique is proposed which aims at damping the exact δ from the TR1 formula by imposing

$$\frac{\det(A_+ A_+^\top)}{\det(AA^\top)} \in \left[\nu^2, \frac{1}{\nu^2}\right]$$

for $\nu \in (0,1)$. Alternatively, one might impose a 'not-too-small' condition for δ: $\delta \geq c\|\sigma\|\|y - As\|$ for $c \approx 0$. The development of the TR2 update, which will not be further explained in this thesis, constitutes a possible remedy for the instabilities caused by minuscule d. For further details, see [17].

Properties. As a generalization of SR1, TR1 shares certain properties of SR1 updating, for example there does not seem to exist a suitable norm for which the TR1-update (17) gains a least change characterization. Most notably, invariance with regard to linear transformation on the range and domain space of the problem is one of the major advantages of TR1. Affine invariance is especially useful when appropriate preconditioners or scaling is desired, which is often the case in the course of an optimization process. Another important property which has also been broached in [13] is the heredity property, which basically means that TR1 satisfies the corresponding secant conditions along all previous steps, provided the constraints are affine. An immediate consequence is that after $p := \min(m,n)$ linearly independent steps and well-defined TR1-updates on an initial approximation A_0, one obtains the exact Jacobian $A_p = A_*$.

Adjoint Broyden. Alternatively one may apply a rank-one update of the following type:

$$A_+ = A - \frac{vv^\top D}{\|v\|^2},$$

where D is the residual matrix defined by

$$D := A - c'(x_+).$$

Jacobian updates of this type are referred to as *Adjoint Broyden* updates. Independent of the choice of v, the *adjoint tangent condition*

$$v^\top A_+ = v^\top c'(x_+)$$

always holds. Choosing the *tangent version*, i.e. $v := Ds$, we further obtain an approximate direct secant condition

$$A_+ s = c'(x_+)s \approx y.$$

Singularities of the form $\|v\| \approx 0$ are easy to detect and can be avoided by skipping the update. In contrast to TR1, updates of the Adjoint Broyden class gain a least change characterization with regard to the Frobenius norm. They further display the desirable heredity property for affine-linear constraints, cf. [14].

3.3 Limited Memory Methods

If the number of variables n is comparatively large, the cost of storing and manipulating the full Hessian approximation $B \in \mathbb{R}^{n\times n}$ is prohibitively high. Even if the true Hessian contains a sparsity pattern, the BFGS or SR1 iterates normally are dense. However, both SR1 and BFGS iterates are only depending on their initial approximation and the secant pairs (s, w). For further investigation, consider the following useful compact representation result for SR1, which was proved by Byrd et al. [7]:

Theorem 3.4 (Compact representation of SR1 approximations)**.**
Suppose the symmetric matrix $B_0 := \gamma I, \gamma > 0$, is updated by means of k well-defined SR1-updates (12) *using $(s_j, w_j)_{j=0,...,k-1}$, i.e. $(w_j - B_j s_j)^\top s_j \neq 0 \; \forall\, j \in \{0, ..., k-1\}$, with corresponding intermediate approximations $(B_j)_{j=0..k}$. Then* $\det(P - \gamma S^\top S) \neq 0$ *and $B := B_k$ satisfies*

$$\begin{aligned} B &= \gamma I + \underbrace{(W - \gamma S)}_{U:=} \underbrace{(P - \gamma S^\top S)}_{M:=}^{-1} (W - \gamma S)^\top \\ &= \gamma I + U M^{-1} U^\top, \end{aligned}$$

where

$$\begin{cases} S & := [s_0, ..., s_{k-1}] \in \mathbb{R}^{n\times k}, \\ W & := [w_0, ..., w_{k-1}] \in \mathbb{R}^{n\times k}, \\ P_{ij} & = P_{ji} := s_{i-1}^T w_{j-1} \quad (i \geq j),\ P \in \mathbb{R}^{k\times k}. \end{cases}$$

Proof. See Byrd et al. [7]. □

The basic idea in limited memory Quasi-Newton approaches consists now of keeping only l update pairs (s_i, w_i) in memory, which induces narrower matrices $S \in \mathbb{R}^{n\times l}$ and $W \in \mathbb{R}^{n\times l}$. Usually the most recent pairs are stored, such that $S = [s_{k-l}, ..., s_{k-1}]$ and $W = [w_{k-l}, ..., w_{k-1}]$. For notational convenience, we write $S = [s_0, ..., s_{l-1}]$ and $W = [w_0, ..., w_{l-1}]$. Accordingly, P reduces to an $l \times l$ symmetric matrix. Since we will always assume $\sqrt{n} \gg l$, the overall storage requirement for B reduces considerably from $\mathcal{O}(n^2)$ to $\mathcal{O}(nl)$. Moreover, the cost of factorizing the middle matrix M is negligibly low and the matrix B is invariant to simultaneous scaling of w_j and s_j. We thus have derived the *Limited-Memory-SR1*-method (L-SR1) for constrained optimization.

An application of the Sherman-Morrison-Woodbury formula (A.3) to the limited memory version of B, yields a similar representation for the inverse of B:

$$\begin{aligned} B^{-1} &= \gamma^{-1} I + \underbrace{(S - \gamma^{-1} W)}_{\tilde{U}:=} \underbrace{(Q - \gamma^{-1} W^\top W)}_{N:=}^{-1} (S - \gamma^{-1} W)^\top \qquad (18) \\ &= \gamma^{-1} I + \tilde{U} N^{-1} \tilde{U}^\top, \end{aligned}$$

where $Q \in \mathbb{R}^{l \times l}$ is defined by

$$Q_{ij} = Q_{ji} := s_{j-1}^\top w_{i-1} \ (i \geq j).$$

Matrix-vector products of the form Bv or $B^{-1}v$ can be computed at a cost of order $\mathcal{O}(nl + l^3)$, either by exploiting the compact representation or by 'unrolling' the l updates via scalar products from right to left ([21]). This means a considerable gain in efficiency, in particular if we consider the large-scale case, where $\sqrt{n} \gg l$.

Making further use of the special structure of the compact representation yields a reasonable strategy for the choice of $\gamma > 0$ to enforce nonsingularity of the middle matrices without needing to check the stability conditions $(w_j - B_j s_j)^\top s_j \neq 0 \ \forall j \in \{0, ..., l-1\}$. Moreover, positive definiteness of the full SR1-Hessian approximation can be ensured, see section 7.

Similar results hold true for the BFGS-method and its limited memory version *L-BFGS*:

Theorem 3.5 (Compact representation of BFGS approximations)**.**
Suppose the symmetric matrix $B_0 := \gamma I, \gamma > 0$, is updated by means of k well-defined BFGS-updates (14) *using $(s_j, w_j)_{j=0,...,k-1}$, i.e. $s_j^\top w_j > 0 \ \forall j \in \{0, ..., k-1\}$, with corresponding intermediate approximations $(B_j)_{j=0..k}$. Then $B := B_k$ satisfies*

$$B \quad = \quad \gamma I - \begin{bmatrix} \gamma S & W \end{bmatrix} \begin{bmatrix} \gamma S^\top S & G \\ G^\top & -D \end{bmatrix}^{-1} \begin{bmatrix} \gamma S & W \end{bmatrix}^\top$$

where

$$\begin{cases} D & := \operatorname{diag}(s_0^\top w_0, \ldots, s_{k-1}^\top w_{k-1}), \\ E_{ij} & := \begin{cases} s_{i-1}^\top w_{j-1} & , \text{if} \quad i > j \\ 0 & , \text{otherwise.} \end{cases} \quad (E \in \mathbb{R}^{k \times k}) \end{cases}$$

Proof. See Byrd et al. [7]. □

3.4 The LRAMBO Algorithm

LRAMBO is an acronym for **L**ow-**R**ank **A**pproximation **M**odification **B**ased **O**ptimizer. It is a total Quasi-Newton method whose main goals comprise considerable reduction in the per-iteration-cost of order $\mathcal{O}(mn^2)$ of a standard SQP method to a 'bilinear' operations count, invariance under affine transformations on the variable space as well as invariance to positive diagonal scaling of the constraints. The main idea to achieve these goals is maintaining the factorized representations of Hessian and Jacobian-related approximations subject to low-rank updates without requiring rather costly derivative matrices. Instead, we employ AD whenever exact 'cheap' derivative-related objects are desired. Most importantly, existing factorizations are updated at quadratic cost[3], instead of being recomputed from scratch at each iteration at cubic cost. First details on maintaining factorized representations in the optimization context can be found in [17].

The following table illustrates LRAMBO's essential functioning in pseudocode. Some available options are given in brackets and will be further described in section 8.

Algorithm 2 (LRAMBO - main routine)**.**

```
initialize  k = 0, x_0, λ_0, ε, A_0, B_0
while   stopping criterion not fulfilled
     do
          solve   KKT-system (8) [QR, ZID] → [d_k, σ_k]
          if      descent w.r.t. merit function [AUGMLAGR, L1] break
          update  Jacobian [ADJBR, TR1] if possible
          adjust  working set if necessary
     while   no descent direction found and change in working set
     compute α_k by line search [ShaZ, Backtracking] on merit function
     update  x_{k+1} = x_k + α_k d_k, λ_{k+1} = λ_k + σ_k
     adjust  working set if necessary
     update  Jacobian [ADJBR, TR1]
     update  Hessian [SR1, BFGS, L-SR1]
     set     k → k + 1
end
```

ADJBR: Adjoint Broyden update
QR: QR-factorization of A with full Hessian approximation B
ZID: ZEDISDEAD approach
AUGMLAGR: Augmented Lagrangian function
L1: $\mathcal{L}^1$-merit function

[3] i.e. the operations count is a quadratic function of the corresponding dimension

The linear algebra routine, which governs all objects (or updates) and subroutines involved in solving the KKT-system (8) is basically split into a 'normal' and a limited memory version. The current standard way to solve the resulting KKT-systems contains a Null Space Method with the 'ZEDISDEAD' implementation (see section 4.1) as the currently most efficient limited memory version. Hereby the Jacobian approximation A is represented by its QR-factors. By contrast, this thesis aims at developing another efficient limited memory version by means of a *semi-normal* approach in conjunction with a Limited Memory SR1-Hessian approximation. The resulting options will eventually be added to the current LRAMBO package. So far, the inner loop, enclosed by the '`do ... while`' statement, should be run at most $\min(m, n)$ -times per iteration, as the heredity property of the available Jacobian update options guarantees exactness after $\min(m, n)$ updates. Of course, we aspire to find a descent direction with as less inner loop iterations as possible.

Once a descent direction is found, an appropriate step size with regard to a preferred merit function has to be determined. To this end, LRAMBO additionally features the rather new line search method *Shift and Zoom (ShaZ)*, offering improved robustness in particular on problems with negative curvature, cf. section 8.1.

4 Solving the KKT-System

In the already mentioned paper by Vavasis [26] (or again [20]) one can find a good overview of the conventional methods to solve the KKT-system

$$\begin{bmatrix} B & A^\top \\ A & 0 \end{bmatrix} \begin{bmatrix} d \\ \sigma \end{bmatrix} = - \begin{bmatrix} g(x,\lambda) \\ c_{\mathcal{A}}(x) \end{bmatrix},$$

whose matrix structure occurs in a lot of other mathematical fields as well. Contemporary solution techniques usually incorporate three different alternatives, namely symmetric indefinite factorization, the null space method and the range space method, where the latter is rather seldomly broached in literature. The first alternative means factorizing the entire (in general indefinite) $(n+m) \times (n+m)$ KKT-system, neglecting its special structure, which is inordinately expensive in our large-scale case. The two latter methods usually procede by replacing A by an LU or QR-factorization and further factorizations of the resulting mixed Jacobian-Hessian objects, cf. section 4.1. Using the null space method to solve (8) in combination with low-rank secant updating of Hessians *and* Jacobians has been proposed and developed by Griewank and Walther [13]. Further contributions to this total Quasi-Newton method have been made by Griewank, Walter, and Korzec [15],[17], especially concerning maintenance of factorized representations. The large-scale case, which is of major interest in this thesis, leads to further exploitation of the special structure of B given by the compact representation in Theorem 3.4. In the recent master's theses by Schloßhauer [21] and Bosse [5], the theory of the null space method in combination with limited memory Hessians has been set forth. Our current implementation of the null space method will be briefly summarized in the subsequent section, followed by a description of the range space approach which finally leads to the idea of a *semi-normal* Jacobian approximation.

Assumption. From now on, we assume that the LICQ is fulfilled, i.e.

$$\operatorname{rank}(c'_{\mathcal{A}}(x)) = m,$$

for all x sufficiently close to a stationary point x_*. We further require that the Jacobian approximation $A \in \mathbb{R}^{m \times n}$ mirrors this property in that $\operatorname{rank}(A) = m$. In view of the desirable SOSC, cf. (3), and the assumptions of Theorem 3.3, we further assume that $Z^\top B Z$ is positive definite.

4.1 Null Space Method

The basis of the null space method is an extended QR-factorization of $A^\top$:

$$A^\top = \underbrace{\begin{bmatrix} Y & Z \end{bmatrix}}_{=:Q \in \mathbb{R}^{n \times n}} \begin{bmatrix} L^\top \\ 0 \end{bmatrix} \in \mathbb{R}^{n \times m},$$

where $L \in \mathbb{R}^{m \times m}$ is lower triangular and $Y \in \mathbb{R}^{n \times m}$ and $Z \in \mathbb{R}^{n \times (n-m)}$ are matrices whose columns form orthonormal bases for $\mathcal{R}(A^\top)$ and $\mathcal{N}(A)$, respectively.

Multiplying the KKT-system (8) by the orthogonal matrix

$$\Theta^\top := \begin{bmatrix} Q^\top & 0 \\ 0 & I_m \end{bmatrix}$$

and introducing a 'smart identity' $\Theta\Theta^\top = I_{(n+m)}$[4] yields

$$\begin{bmatrix} Y^\top BY & Y^\top BZ & L^\top \\ Z^\top BY & Z^\top BZ & 0 \\ L & 0 & 0 \end{bmatrix} \begin{bmatrix} Y^\top d \\ Z^\top d \\ \sigma \end{bmatrix} = - \begin{bmatrix} Y^\top g(x,\lambda) \\ Z^\top g(x,\lambda) \\ c_{\mathcal{A}}(x) \end{bmatrix}. \tag{19}$$

Defining

$$\begin{cases} E := Y^\top BY \in \mathbb{R}^{m\times m} \\ C := Y^\top BZ \in \mathbb{R}^{m\times(n-m)} \\ RR^\top := Z^\top BZ \in \mathbb{R}^{(n-m)\times(n-m)}, \\ R \in \mathbb{R}^{(n-m)\times(n-m)} \text{ upper triangular,} \end{cases}$$

the solution $(d,\sigma)^\top$ to (19) finally reads

$$\begin{cases} d = -YL^{-1}c_{\mathcal{A}}(x) - ZR^{-\top}R^{-1}(Z^\top g(x,\lambda) + C^\top L^{-1}c_{\mathcal{A}}(x)) \\ \sigma = -L^{-\top}(Y^\top g(x,\lambda) + EY^\top d + CZ^\top d). \end{cases} \tag{20}$$

As suggested above, $Z^\top BZ$ is forced to stay positive definite. This can be achieved by a determinant control technique; for details see [15]. Consequently, the flipped Cholesky factor R can be determined. Naturally, none of the matrix inverses occurring in (20) are being computed. Instead, we use backward and forward substitution as well as consecutive matrix-vector multiplication from right to left to obtain the desired solution (d,σ).

Limited Memory. In the limited memory version without damping, we make use of the compact representation formula (3.4) for B to find (cf. [21]):

$$\begin{cases} E = \gamma I_m + Y^\top UM^{-1}(Y^\top U)^\top \\ C = Y^\top UM^{-1}(Z^\top U)^\top \\ Z^\top BZ = \gamma I_{(n-m)} + Z^\top UM^{-1}(Z^\top U)^\top. \end{cases}$$

The major drawback of this approach is the need to save the large QR-factors $Z \in \mathbb{R}^{n\times(n-m)}$ and $Y \in \mathbb{R}^{n\times m}$. Thus, the required memory space at this stage is at least of order $\mathcal{O}(n^2)$, which was supposed to be avoided in the first place. Just recently, Bosse [5] has demonstrated how the null space method with a limited memory Hessian can be realized *without* the comparatively large matrix Z, which may appear paradox. The reason is, that while solving the KKT-system with the approach above, Z only appears in the form of $ZZ^\top$, which can be substituted by $I - YY^\top$. This idea has led to the new 'ZEDISDEAD' implementation within the LRAMBO package, bringing a massive relieve in required memory space. The main result is summarized in the following theorem.

[4] please note $QQ^\top = YY^\top + ZZ^\top = I_n$

Theorem 4.1 (ZEDISDEAD)**.**
For a partial limited memory approach on a total Quasi-Newton method with nullspace factorization, the needed memory size and computational complexity per iteration are both of order $\mathcal{O}(n \cdot \max(m, l) + l^3)$.

Proof. See [5]. □

Nevertheless, ZEDISDEAD expects $Y \in \mathbb{R}^{n \times m}$ to be in storage, which may be quite costly in the large-scale case combined with a large number of active constraints, $m \lesssim n$.

However the striving for even more efficiency can be continued. This will be realized by a semi-normal approach, presented in section 4.3, which enables considerable alleviation of the $n \cdot m$ product in the complexity and storage order. As the idea of a semi-normal approach arose from a limited memory range space method, the latter will be broached in the following section.

4.2 Range Space Method

In the following, the basic technique of the range space approach to solve (8) will be outlined. It is based on a *thin QR-factorization* of $A^\top$:

$$A^\top = YL^\top,$$

where

$$\begin{cases} L \in \mathbb{R}^{m \times m} & \text{lower triangular,} \\ Y \in \mathbb{R}^{n \times m} & \text{column-orthonormal.} \end{cases}$$

The upper equality in (8) yields the step direction

$$\begin{aligned} d &= -B^{-1}g(x,\lambda) - B^{-1}A^\top\sigma \\ &= -B^{-1}(g(x,\lambda) + YL^\top\sigma). \end{aligned} \tag{21}$$

Using the expression for d in (21), we obtain

$$(AB^{-1}A^\top)\sigma = c_{\mathcal{A}}(x) - LY^\top B^{-1}g(x,\lambda).$$

At this point, step computation in the case of storing the full Hessian approximation B requires a suitable factorization of the symmetric (and, depending on B, probably also positive definite) matrix $AB^{-1}A^\top \in \mathbb{R}^{m \times m}$. However, we do not further persue this approach in favor of the large-scale case employing a limited memory strategy.

Limited memory. In the limited memory case we are able to improve efficiency in the step calculation by exploiting the SR1 compact representation formula (18) of B^{-1} which results in

$$AB^{-1}A^\top = L\underbrace{(\gamma^{-1}I + Y^\top\tilde{U}N^{-1}\tilde{U}^\top Y)}_{=:\tilde{M}\in\mathbb{R}^{m\times m}}L^\top,$$

as $Y^\top Y = I_m$. Provided existence of the involved inverses is given, the Sherman-Morrison-Woodbury formula (A.3) yields

$$\tilde{M}^{-1} = \gamma I - \gamma^2 Y^\top \tilde{U}(N + \gamma \tilde{U}^\top Y Y^\top \tilde{U})^{-1} \tilde{U}^\top Y$$

Finally, the Lagrange multiplier step writes

$$\sigma = (L^{-\top} \tilde{M}^{-1} L^{-1})(c_{\mathcal{A}}(x) - L Y^\top B^{-1} g(x, \lambda)). \tag{22}$$

Now we could design a method which is similar to the limited memory null space method ZEDISDEAD, in that essentially only the thin QR-factors Y and L as well as related KKT-matrices are used to solve the KKT-system (8). Therefore we make the following remark.

Remark 4.2 (Relation to ZEDISDEAD)**.**
The interest in solving the KKT-system by a limited memory range space approach is rather low, as computational efficiency and required memory space can be expected to be equivalent to the ZEDISDEAD implementation of the null space method.

4.3 Avoiding Explicit Range Space Representation via Semi-Normal Equation

In order to derive the semi-normal method for a limited memory Hessian implementation, we return to the mixed Quasi-Newton step calculation (cf. section 3.2) given by

$$\begin{bmatrix} B & c'_{\mathcal{A}}(x)^\top \\ c'_{\mathcal{A}}(x) & 0 \end{bmatrix} \begin{bmatrix} d \\ \sigma \end{bmatrix} = - \begin{bmatrix} g(x, \lambda) \\ c_{\mathcal{A}}(x) \end{bmatrix}, \tag{23}$$

which incorporates the *exact* Jacobian of active constraints at the current iterate x.

Analogously to (21), the upper equality in (23) yields the primal step

$$d = -B^{-1}(g(x, \lambda) + c'_{\mathcal{A}}(x)^\top \sigma).$$

This time we do not replace $c'_{\mathcal{A}}(x)^\top$ by a certain approximation. Instead, given $\sigma \in \mathbb{R}^m$ and $g(x, \lambda)$, we now maintain the *exact* expressions on the right hand side. Apart from the cost of the Jacobian-transposed vector product $c'_{\mathcal{A}}(x)^\top \sigma$ which is just a small multiple of a function evaluation of c, we can reduce the cost of determining d to order $\mathcal{O}(ln) + \mathcal{O}(l^3)$ by using the inverse formula given in (18). Note again, that cubic effort in l is acceptable in our case, where $l \ll \sqrt{n}$. Consequently, solving the KKT-system is reduced to the problem of solving

$$(c'_{\mathcal{A}}(x) B^{-1} c'_{\mathcal{A}}(x)^\top)\sigma = c_{\mathcal{A}}(x) - c'_{\mathcal{A}}(x) B^{-1} g(x, \lambda)$$

for the adjoint step σ.

Using SR1 compact representation of B^{-1} (cf. (18)), we can reformulate the left hand side according to

$$\left(c'_{\mathcal{A}}(x)B^{-1}c'_{\mathcal{A}}(x)^\top\right)\sigma = \gamma^{-1}c'_{\mathcal{A}}(x)c'_{\mathcal{A}}(x)^\top\sigma + c'_{\mathcal{A}}(x)\tilde{U}N^{-1}\tilde{U}^\top c'_{\mathcal{A}}(x)^\top\sigma. \tag{24}$$

Semi-normal equation[5]. Contemplating the left addend, we now make the assumption that a suitable *semi-normal Jacobian representation* is available, i.e. we want to determine a nonsingular, lower triangular matrix $L \in \mathbb{R}^{m\times m}$ to fulfill the Cholesky decomposition

$$c'_{\mathcal{A}}(x)c'_{\mathcal{A}}(x)^\top \approx LL^\top, \tag{25}$$

as exact as possible *along* the current adjoint step direction σ:

$$\left[c'_{\mathcal{A}}(x)c'_{\mathcal{A}}(x)^\top\right]\sigma \approx LL^\top\sigma. \tag{26}$$

Details on how this goal may be achieved will be given in section 6.

Using (26), we further obtain

$$\begin{aligned} c'_{\mathcal{A}}(x)B^{-1}c'_{\mathcal{A}}(x)^\top\sigma &\approx (\gamma^{-1}LL^\top + c'_{\mathcal{A}}(x)\tilde{U}N^{-1}\tilde{U}^\top c'_{\mathcal{A}}(x)^\top)\sigma \\ &= L\tilde{M}L^\top\sigma \end{aligned}$$

with

$$\tilde{M} := \gamma^{-1}I + L^{-1}c'_{\mathcal{A}}(x)\tilde{U}N^{-1}\tilde{U}^\top c'_{\mathcal{A}}(x)^\top L^{-\top} \in \mathbb{R}^{m\times m}.$$

A second application of the Sherman-Morrison-Woodbury formula (A.3) to $\tilde{M}$ finally yields

$$\tilde{M}^{-1} = \gamma I - \gamma^2 L^{-1}c'_{\mathcal{A}}(x)\tilde{U}\tilde{N}^{-1}\tilde{U}^\top c'_{\mathcal{A}}(x)^\top L^{-\top}$$

with

$$\tilde{N} := N + \gamma(L^{-1}c'_{\mathcal{A}}(x)\tilde{U})^\top L^{-1}c'_{\mathcal{A}}(x)\tilde{U}.$$

However, the inner matrix $\tilde{N}$ requires exact determination of $c'_{\mathcal{A}}(x)\tilde{U}$, which would necessitate computation of l Jacobian-vector products. This considerable computational effort is circumvented by replacing $c'_{\mathcal{A}}(x)\tilde{U} = c'_{\mathcal{A}}(x)S - \gamma^{-1}c'_{\mathcal{A}}(x)W$ in $\tilde{N}$ by a suitable approximation

$$\bar{U} := \bar{S} - \gamma^{-1}\bar{W} \approx c'_{\mathcal{A}}(x)S - \gamma^{-1}c'_{\mathcal{A}}(x)W = c'_{\mathcal{A}}(x)\tilde{U}. \tag{27}$$

This approximation will be refined within the *inner loop* of our final Quasi-Newton algorithm. Corresponding strategies will be discussed in section 5.4. In the 'outer' part of $\tilde{M}^{-1}$ we maintain the exact expressions $c'_{\mathcal{A}}(x)\tilde{U}$ and

[5] the name refers to the semi-normal equation (SNE) method for least-squares problems, which is based on a similar factorization

$\tilde{U}^\top c'_{\mathcal{A}}(x)^\top$, as we perform a sequence of matrix-vector products from right to left, allowing two additional Jacobian-vector products to be computed employing Algorithmic Differentiation. Eventually we simply replace $\tilde{M}$ by its approximation $\bar{M}$ which we define as follows:

$$\bar{M}^{-1} := \gamma I - \gamma^2 L^{-1} c'_{\mathcal{A}}(x) \tilde{U} \bar{N}^{-1} \tilde{U}^\top c'_{\mathcal{A}}(x)^\top L^{-\top}$$

with

$$\bar{N} := N + \gamma (L^{-1}\bar{U})^\top L^{-1}\bar{U} \quad \in \mathbb{R}^{l \times l},$$

assuming $\det(\bar{N}) \neq 0$. Altogether, the step direction (d, σ) is determined according to

$$\begin{cases} \sigma & = (L^{-\top}\bar{M}^{-1}L^{-1})(c_{\mathcal{A}}(x) - c'_{\mathcal{A}}(x)B^{-1}g(x,\lambda)) \\ d & = -B^{-1}(g(x,\lambda) + c'_{\mathcal{A}}(x)^\top \sigma). \end{cases} \tag{28}$$

It has to be remarked, that nonsingularity of the involved matrices will be ensured by a specific choice of γ. For details, see section 7.

Computational complexity. Taking into account that computing the Lagrange multiplier step by (28) requires solving two unfactorized systems with coefficient matrices $N, \bar{N} \in \mathbb{R}^{l \times l}$, the overall effort of computing the KKT-solution (28) is thus given by the following relation:

$$OPS(\text{eval}(d, \sigma)) = \mathcal{O}(ln + m^2 + l^3) + 12 \cdot OPS(\text{eval}(c_{\mathcal{A}})),$$

disregarding the effort of determining the right hand side of (23), which is shared by any Newton-type procedure.

We conclude the section with a proposition about the form of the system actually solved by the semi-normal approach.

Proposition 4.3 (Reverse consideration)**.**
The step direction (d, σ) obtained by (28) satisfies

$$\begin{bmatrix} B & c'_{\mathcal{A}}(x)^\top \\ c'_{\mathcal{A}}(x) & D \end{bmatrix} \begin{bmatrix} d \\ \sigma \end{bmatrix} = - \begin{bmatrix} g(x,\lambda) \\ c_{\mathcal{A}}(x) \end{bmatrix}$$

where D denotes the discrepancy matrix

$$D := c'_{\mathcal{A}}(x) B^{-1} c'_{\mathcal{A}}(x)^\top - L\bar{M}L^\top \ \in \mathbb{R}^{m \times m}.$$

Proof. The first equation in (28) is equivalent to

$$-c'_{\mathcal{A}}(x)B^{-1}g(x,\lambda) = (L\bar{M}L^\top)\sigma - c_{\mathcal{A}}(x).$$

Multiplying the lower equation in (28) by $c'_{\mathcal{A}}(x)$ yields

$$-c'_{\mathcal{A}}(x)B^{-1}g(x,\lambda) = c'_{\mathcal{A}}(x)d + c'_{\mathcal{A}}(x)B^{-1}c'_{\mathcal{A}}(x)^\top \sigma$$

which results in

$$c'_{\mathcal{A}}(x)d + \left(c'_{\mathcal{A}}(x)B^{-1}c'_{\mathcal{A}}(x)^\top - L\bar{M}L^\top\right)\sigma = -c_{\mathcal{A}}(x).$$

Defining $D := c'_{\mathcal{A}}(x)B^{-1}c'_{\mathcal{A}}(x)^\top - L\bar{M}L^\top$ accomplishes the proof. □

Corollary 4.4.
The quality of the step determination with the semi-normal approach in conjunction with L-SR1 is only governed by the size of the Jacobian approximation error D along the current Lagrange multiplier step σ. The size of the discrepancy is in turn characterized by the quality of the semi-normal approximation (26) *and $\bar{U}$. If both are exact, the deviation from the exact SQP step depends only on the quality of the L-SR1 Hessian approximation.*

The above presented approach combining step determination by representing first-order Jacobian information by means of a semi-normal approximation together with an L-SR1 Hessian approximation will from now on be denoted by L-SR1-SN.

5 Updating Hessian-Related Components

5.1 Memory Layout

In this section, we want to give an insight into the data structure which has to be stored in order to retrieve the solution of the KKT-system (28) by means of the semi-normal approach proposed in section 4.3. The effect of typical Hessian modifications from a limited memory strategy will be described as well. It will be shown, how resulting changes in the elementary objects S and W effect their related objects and how the implementation can be organized with maximal efficiency.

Observation. All matrices involved in solving the KKT-system are already contained in the formula for the Lagrange multiplier step σ in (28). Therefore, we need to take into account the basic limited memory Hessian components S and W which are necessary to perform operations with B as well as the mixed objects defining $\bar{N}$ necessary to perform a multiplication with $\bar{M}^{-1}$. We obtain the following classification of the stored data structure which will be proven to ensure minimal computational effort when determining (d, σ):

`Hessian-related objects:`
$S, W \in \mathbb{R}^{n\times l};\ S^\top W,\ S^\top S,\ W^\top W \in \mathbb{R}^{l\times l};\ \gamma \in \mathbb{R}$
`Jacobian-related objects:`
$L \in \mathbb{R}^{m\times m}$
`Mixed objects:`
$L^{-1}\bar{S},\ L^{-1}\bar{W} \in \mathbb{R}^{m\times l};$
$\bar{S}^\top (LL^\top)^{-1}\bar{S},\ \bar{S}^\top (LL^\top)^{-1}\bar{W},\ \bar{W}^\top (LL^\top)^{-1}\bar{W} \in \mathbb{R}^{l\times l}$

Table 1: KKT-related matrices for the semi-normal method L-SR1-SN

In the setting of our total Quasi-Newton algorithm, these matrices are subject to certain low-rank updates, originating from Jacobian updates (including changes in the working set) and Hessian updates. Mixed objects are those who are effected by both. Additionally we may also design suitable 'mixed updates', i.e. separate updates for the mixed objects. For each of these updates, their effect on our data structure has to be set forth.

To summarize, the main goal is to effect stable and efficient updates on the given data structure. The major advantage consists of the avoidance of storing objects of size $m \cdot n$, as it is for instance the case in the ZEDISDEAD approach [5] which requires the large matrix $Y \in \mathbb{R}^{n\times m}$. Instead, the only necessary Jacobian information in memory is contained in the semi-normal factor L and the corresponding mixed objects $L^{-1}\bar{S}$ and $L^{-1}\bar{W}$.

γ-dependend objects. Through changes in γ, the derived objects needed for the computation of σ will be obtained by means of corresponding changes in their constituents. For instance we compute N_+ using the relation

$$N_+ = Q_+ - \gamma_+ (W^\top W)_+,$$

where Q_+ is uniquely determined by the right upper triangular part of $(S^\top W)_+$. The issue of efficient adaptation of the initialization factor γ will be addressed in section 7.

5.2 Limited Memory Hessian Updates

Depending on the preferred limited memory strategy, basic operations on the matrices $S = [s_0, ..., s_{l-1}]$ and $W = [w_0, ..., w_{l-1}]$ may incorporate

- ◇ removing a given update pair $(s_i, w_i),\ i \in \{0, ..., l-1\}$,
- ◇ adding a new pair (s_l, w_l),
- ◇ replacing a given update pair $(s_i, w_i),\ i \in \{0, ..., l-1\}$, by a new pair (s_l, w_l).

Of course, a couple of other elaborate strategies including merging, variation of l, 'backing-up' and restarting are conceivable. For a remarkable variety of several algorithms associated with the management of the limited memory components, see for example the PHD thesis by Kolda [16] which offers interesting testing results in conjunction with L-BFGS. It should be remarked, that it is sometimes recommended to choose the sequence of update vectors (s_i, w_i) forming S and W, such that the correct chronology of the step-taking is preserved, cf. the comments on the L-BFGS algorithm by Liu and Nocedal [19]. Therefore possible replacement operations are usually followed by permutations of the columns of S and W.

5.3 Efficient Adjustment of the KKT-Matrices

Adjustment of the Hessian-related KKT-matrices in the semi-normal approach resembles the procedures investigated for the ZEDISDEAD method, cf. [5]. Therefore, we only emphasize the differences induced by an extended usage of Algorithmic Differentiation which replaces the effort of several matrix vector-products involving Y.

Deletion. Discarding a vector pair (s_i, w_i) from the basic matrices S, W does not cause any difficulty. To obtain $(S^\top W)_+$ we simply need to delete the i-th line and column. Causing negligible effort, the same procedure can be applied to the other Hessian-related objects and the effected mixed objects.

Extension. Appending a given vector pair (s_l, w_l) to S and W requires several matrix-vector products involving $W^\top s_l$, $S^\top w_l$, $S^\top s_l$ and $W^\top w_l$ as well as the scalar products $s_l^\top s_l$, $s_l^\top w_l$ and $w_l^\top w_l$ to obtain the Hessian-related objects. For the mixed objects, consider

$$L^{-1}\bar{S}_+ = L^{-1}\left[\bar{S} \quad \bar{s_l}\right] \approx L^{-1} c'_{\mathcal{A}}(x)\left[S \quad s_l\right].$$

Thus it is plausible to compute

$$\bar{s_l} = c'_{\mathcal{A}}(x) s_l \tag{29}$$

with the forward mode of Algorithmic Differentiation and to determine

$$L^{-1}\bar{S}_+ = [L^{-1}\bar{S} \quad L^{-1}c'_{\mathcal{A}}(x)s_l],$$

using forward substitution. Now consider $\bar{S}^\top(LL^\top)^{-1}\bar{S}$. Updating S yields

$$\begin{aligned} \bar{S_+}^\top(LL^\top)^{-1}\bar{S}_+ &= \begin{bmatrix}\bar{S}^\top \\ \bar{s_l}^\top\end{bmatrix}(LL^\top)^{-1}\begin{bmatrix}\bar{S} & \bar{s_l}\end{bmatrix} \\ &= \begin{bmatrix}\bar{S}^\top(LL^\top)^{-1}\bar{S} & (L^{-1}\bar{S})^\top L^{-1}\bar{s_l} \\ \bar{s_l}^\top L^{-\top}(L^{-1}\bar{S}) & \bar{s_l}^\top(LL^\top)^{-1}\bar{s_l}\end{bmatrix}. \end{aligned}$$

Replacement. Having chosen a vector pair (s_i, w_i) to be surrogated by a new pair (s_l, w_l), we just have to recalculate the i-th line and/or column of the corresponding object. For example, write

$$S_+ = \begin{bmatrix}S_L & s_l & S_R\end{bmatrix}$$

to retrieve

$$\begin{aligned} &\bar{S_+}^\top(LL^\top)^{-1}\bar{S}_+ \\ &= \begin{bmatrix}\bar{S}_L^\top \\ \bar{s_l} \\ \bar{S}_R^\top\end{bmatrix}(LL^\top)^{-1}\begin{bmatrix}\bar{S}_L & \bar{s_l} & \bar{S}_R\end{bmatrix} \\ &= \begin{bmatrix}\bar{S_L}^\top(LL^\top)^{-1}\bar{S}_L & (L^{-1}\bar{S}_L)^\top L^{-1}\bar{s_l} & \bar{S_L}^\top(LL^\top)^{-1}\bar{S}_R \\ \bar{s_l}^\top L^{-\top}L^{-1}\bar{S}_L & \bar{s_l}^\top(LL^\top)^{-1}\bar{s_l} & \bar{s_l}^\top L^{-\top}L^{-1}\bar{S}_R \\ \bar{S_R}^\top(LL^\top)^{-1}\bar{S}_L & (L^{-1}\bar{S_R})^\top L^{-1}\bar{s_l} & S_R^\top(LL^\top)^{-1}\bar{S}_R\end{bmatrix}. \end{aligned}$$

Note that the blocks in the corners are equal to the corresponding blocks in $\bar{S}^\top(LL^\top)\bar{S}$, whereas the i-th line needs to be redetermined by computing $L^{-1}\bar{s_l}$ using Algorithmic Differentiation and forward substitution as well as a matrix-vector product of order $\mathcal{O}(lm)$ with the stored matrix

$$L^{-1}\bar{S}_+ = \begin{bmatrix}L^{-1}\bar{S}_L & L^{-1}\bar{s_l} & L^{-1}\bar{S}_R\end{bmatrix}.$$

$L^{-1}\bar{W}_+$, $\bar{W}_+^\top(LL^\top)^{-1}\bar{W}_+$ and $\bar{S}_+^\top(LL^\top)^{-1}\bar{W}_+$ can be updated analogously, thereby necessitating evaluation of $L^{-1}c'_{\mathcal{A}}(x)w_l$.

We conclude this section with a proposition describing the necessary computational effort due to Hessian updating.

Proposition 5.1 (Hessian update)**.**
The computational effort for any of the proposed updates on the Hessian-related objects in L-SR1-SN is bounded by the order

$$\mathcal{O}(ln + m^2) + 5 \cdot OPS(eval(c_{\mathcal{A}})).$$

Proof. The left addend is directly derived by the proposed updating routines. The product $l \cdot m$ vanishes, for LICQ ensures $\max(m, n) = n$. Moreover, there is a maximal amount of two Jacobian-vector products to be computed. Making use of the vector mode of AD, cf. Theorem 2.7, yields linearly bounded complexity with an upper bound of five constraint evaluations. □

Naturally, the mentioned Hessian updates can be written as low-rank changes. This will be exploited in Section 7 where the Cholesky factors of Q and $S^\top S$ have to be maintained subject to low-rank updates in order to determine suitable shifts in the initialization factor γ.

5.4 Mixed Updates

In this section, we propose two update strategies for the mixed objects, which are part of the definition of $\bar{N}$. More precisely, we want to consider implicit updates on the non-stored objects $\bar{S} \approx c'_{\mathcal{A}}(x)S$ and $\bar{W} \approx c'_{\mathcal{A}}(x)W$ which will be transformed into corresponding updates on the stored mixed objects $L^{-1}\bar{S}, L^{-1}\bar{W}$, $\bar{S}^\top(LL^\top)^{-1}\bar{S}, \bar{S}^\top(LL^\top)^{-1}\bar{W}, \bar{W}^\top(LL^\top)^{-1}\bar{W}$.

To begin with, we make the following remark dedicated to the case of affine-linear constraints.

Remark 5.2 (Exactness in affine case)**.**
In case of affine-linear constraints $c_i, i \in \mathcal{A}(x)$, the approximations $L^{-1}\bar{S}$ and $L^{-1}\bar{W}$ are already exact, i.e. $\bar{S} = c'_{\mathcal{A}}(x)S$ and $\bar{W} = c'_{\mathcal{A}}(x)W$.

Proof. The assertion follows directly from the Hessian extension procedure, cf. (29). □

In the presence of nonlinear constraints, the approximations originating from consecutive Hessian updating are not exact. Therefore we suggest the following implicit mixed updates to improve their accuracy.

Alternative 1: Secant updates. Since the approximation

$$\bar{S} \approx c'_{\mathcal{A}}(x)S$$

implicitly defines a Jacobian approximation $J_S \in \mathbb{R}^{m\times n}$ via

$$\begin{cases} J_S S = \bar{S} \\ J_S := \bar{S}(S^\top S)^{-1}S^\top = L(L^{-1}\bar{S})(S^\top S)^{-1}S^\top, \end{cases} \tag{30}$$

it is conceivable to implicitly define a low-rank secant update on J_S without actually storing the Jacobian approximation $J_S \in \mathbb{R}^{m\times n}$, as J_S is implicitly given by (30). The component matrices are stored and of small dimension, such that the cost of matrix-vector products involving J_S is of order $\mathcal{O}(ln + m^2)$, if a Cholesky factorization of $S^\top S \in \mathbb{R}^{l\times l}$ is available. Therefore we can perform typical Jacobian updates like the direct TR1 or the tangent version of Adjoint Broyden on J_S with an acceptable computational effort. In the latter case, we modify J_S as follows:

$$(J_S)_+ = J_S - \frac{Dss^\top D^\top D}{\|Ds\|^2}, \tag{31}$$

where

$$D := J_S - c'(x_+).$$

By virtue of the heredity property of TR1 or Adjoint Broyden, it suffices to update J_S l times at a fixed iterate x using the stored steps $s_0, \ldots, s_{l-1}$, to obtain the exact matrix $c'_{\mathcal{A}}(x)S = (J_S)_+ S = \bar{S}_+$. This can be exploited in the inner loop of LRAMBO.

The above rank-one update can be rewritten as a rank-one update on $\bar{S}$:

$$\begin{aligned} \bar{S}_+ &:= (J_S)_+ S \\ &= J_S S - \frac{Dss^\top D^\top DS}{\|Ds\|^2} \\ &= \bar{S} + \delta r \rho^\top, \end{aligned}$$

with

$$\begin{cases} \delta := & -\frac{1}{\|Ds\|^2} \\ r := & Ds \\ \rho := & (r^\top D)S. \end{cases}$$

Below, we show how an implicitly defined rank-one update on $\bar{S}$ can be incorporated into our given data structure. Obviously, a similar strategy can be persued if $\bar{W}$ shall be updated.

Alternative 2: Columnwise recomputation. Suppose we do not perform a corresponding secant update on $\bar{S}$ or $\bar{W}$. Owing to the Hessian extension procedure presented in the previous section, we have

$$L^{-1}\bar{S} = L^{-1}\left[c'_{\mathcal{A}}(x_1)s_0, \ldots, c'_{\mathcal{A}}(x_i)s_{i-1}\right]$$

after the i-th (w.l.o.g. $i < l$) Hessian update without replacements, deletions or skips. As we want to approximate

$$L^{-1}c'_{\mathcal{A}}(x)S = L^{-1}\left[c'_{\mathcal{A}}(x_i)s_0, \ldots, c'_{\mathcal{A}}(x_i)s_{i-1}\right],$$

we suggest to simply 'refresh' the columns $(\bar{s}_j)$ of $\bar{S}$ by replacing the least up-to-date column, (normally the first) by substituting $L^{-1}c'_{\mathcal{A}}(x_1)s_0$ by $L^{-1}c'_{\mathcal{A}}(x_i)s_0$. This simple updating strategy can be easily added to the inner loop by memorizing the updating order. Alternatively, one may consider the error measured in terms of the $\|.\|_1$-norm:

$$\begin{aligned} err_1 &= \|L^{-1}c'_{\mathcal{A}}(x)S - L^{-1}\bar{S}\|_1 \\ &\leq \|L^{-1}\|_1 \max_{j\in\{1,\ldots,i-1\}} \|(c'_{\mathcal{A}}(x_i) - c'_{\mathcal{A}}(x_j))s_{j-1}\|_1 \\ &\leq \|L^{-1}\|_1 \max_{j\in\{1,\ldots,i-1\}} \big(\|c'_{\mathcal{A}}(x_i) - c'_{\mathcal{A}}(x_j)\|_1 \|s_{j-1}\|_1\big). \end{aligned}$$

If c' fulfills the Lipschitz property, one may expect the left factor to grow with the distance $p_j := \|x_i - x_j\|$. Thus, a promising heuristic consists of choosing j as a compromise between the largest distance p_j and the largest step s_{j-1},

$j \in \{1, ..., i-1\}$. Again, either of the two choices can be reformulated as a rank-one update on $\bar{S}$, using the update vectors

$$\begin{cases} r := (c'_{\mathcal{A}}(x_i)s_j - \bar{s}_j) \ \in \mathbb{R}^m \\ \rho := e_j \ \in \mathbb{R}^l \\ \delta := 1, \end{cases}$$

for refreshing the j-th column. $\bar{W}$ may be modified in analogous fashion.

Efficient adjustment of the KKT-matrices. As shown above, each of the proposed updates can be reformulated as a low-rank update on $\bar{S}$ (and $\bar{W}$). The resulting changes on the data structure can be easily incorporated, as the following example with $\bar{S}$ shows:

$$\begin{aligned} L^{-1}\bar{S}_+ &= L^{-1}\bar{S} + \delta(L^{-1}r)(\rho^\top) && (32) \\ \bar{S}_+^\top(LL^\top)^{-1}\bar{S}_+ &= (\bar{S}^\top + \delta\rho r^\top)(LL^\top)^{-1}(\bar{S} + \delta r\rho^\top) \\ &= \bar{S}^\top(LL^\top)^{-1}\bar{S} + \delta(L^{-1}\bar{S})^\top(L^{-1}r)\rho^\top \\ &\quad +\delta\rho(L^{-1}r)^\top(L^{-1}\bar{S}) + \delta^2\left((L^{-1}r)^\top(L^{-1}r)\right)\rho\rho^\top && (33) \end{aligned}$$

Bearing in mind that $L^{-1}\bar{S}$ and $\bar{S}^\top(LL^\top)^{-1}\bar{S}$ is kept in storage, mixed updates via columnwise recomputation comprises a couple of matrix-vector products and forward substitutions in addition to two Jacobian-vector products involving an old secant pair $(s_j, w_j), j \in \{0, \ldots, l-2\}$. For the time being, we will prefer the second alternative, since the first alternative additionally requires two reverse mode calls for the update vectors of $\bar{S}$ and $\bar{W}$. The following conclusion can be drawn.

Proposition 5.3 (Computational complexity)**.**
The computational effort for the adjustment of the data structure due to mixed updates on $\bar{S}$ and $\bar{W}$ via Alternative 2 (columnwise recomputation) can be bounded by the order

$$\mathcal{O}(m \cdot \max(m, l)) + 5 \cdot OPS(\text{eval}(c_{\mathcal{A}})).$$

Proof. Analogous to Proposition 5.1. □

6 Updating the Semi-Normal Factor

In the following section, Jacobian modifications in form of a *semi-normal* update will be suggested and analyzed.

6.1 Secant Condition and BFGS-Update

In order to derive an appropriate secant condition for the update of the semi-normal factor L, consider approximation (26)

$$\left[c'_{\mathcal{A}}(x)c'_{\mathcal{A}}(x)^\top\right]\sigma \approx LL^\top\sigma.$$

We thus require the *semi-normal secant condition*

$$L_+L_+^\top\sigma = c'_{\mathcal{A}}(x_+)c'_{\mathcal{A}}(x_+)^\top\sigma, \tag{34}$$

where $L_+L_+^\top$ shall preserve symmetry and positive definiteness owing to the corresponding properties of $c'_{\mathcal{A}}(x_+)c'_{\mathcal{A}}(x_+)^\top$.

Therefore, we propose the following BFGS-update on $LL^\top$:

$$L_+L_+^\top = LL^\top - \frac{LL^\top\sigma\sigma^\top LL^\top}{\|L^\top\sigma\|^2} + \frac{c'_{\mathcal{A}}(x_+)c'_{\mathcal{A}}(x_+)^\top\sigma\sigma^\top c'_{\mathcal{A}}(x_+)c'_{\mathcal{A}}(x_+)^\top}{\|c'_{\mathcal{A}}(x_+)^\top\sigma\|^2} \tag{35}$$

Consequently, $L_+L_+^\top$ stays symmetric and fulfills the semi-normal secant condition (34). Positive definiteness is also preserved, since (cf. section 3.2.1)

$$\sigma^\top c'_{\mathcal{A}}(x_+)c'_{\mathcal{A}}(x_+)^\top\sigma > 0$$

is satisfied in virtue of LICQ and as long as $\sigma \neq 0$. Considering the denominators in (35), stability of the given BFGS-update is theoretically ensured by the properties of L and $c'_{\mathcal{A}}$.

Orthogonality. In order to find an appropriate criterion governing the updating decision, consider the case where L satisfies the semi-normal equation (25) exactly. In this case, the matrix $c'_{\mathcal{A}}(x)^\top L^{-\top}$ has orthonormal columns spanning $\mathcal{R}\left(c'_{\mathcal{A}}(x)^\top\right)$. Furthermore we must have

$$\|\sigma\|_2^2 = \|c'_{\mathcal{A}}(x)^\top L^{-\top}\sigma\|_2^2.$$

Numerically, we suggest to impose updating on $LL^\top$, whenever the approximate *orthogonality condition*

$$\frac{\|c'_{\mathcal{A}}(x)^\top L^{-\top}\sigma\|_2}{\|\sigma\|_2} \in (1-tol, 1+tol), \text{ where } tol \approx 0, \tag{36}$$

is violated.

Computational complexity. The total amount of operations to compute the update vectors defining (35) is bounded by the order $\mathcal{O}(m^2) + 6\cdot\text{eval}(c_{\mathcal{A}})$ incorporating forward and backward substitution for the downdate and one AD-forward

and reverse mode respectively for the update. Hence, determination of the updating quantities defining the semi-normal BFGS-update is even cheaper as a typical Jacobian update such as Adjoint-Broyden or TR1. The latter case comprises one forward and one reverse sweep as well as two matrix-vector products of order $\mathcal{O}(mn)$ to calculate primal and adjoint residuals.

To conclude, we remark, that the proposed BFGS-update is not *hereditary* such that the exact Jacobian cannot be expected to be obtained after a finite number of secant updates as it is the case for example with TR1. However, we can assume an improvement of the semi-normal approximation by consecutive BFGS updating at a fixed iterate within the inner loop. Consequently an adaptation of the break criterion for the inner loop is necessary.

6.2 Changes in the Working Set

In this section, we assume that the active set $\mathcal{A}$ at the solution x_* still has to be determined. Consequently, the working set $\mathcal{W} \subset \{1, ..., M\}, \mathcal{E} \subset \mathcal{W}$, which constitutes our guess for $\mathcal{A}$, might underly various modifications. Resulting modifications on the semi-normal factor L and the mixed objects will be discussed subsequently.

Activation. Suppose, constraint $I \in \mathcal{I} \setminus \mathcal{W}$ needs to be activated. Let $A := c'_{\mathcal{W}}(x)$ denote the Jacobian of constraints contained in the working set $\mathcal{W}$ and assume the gradient of the activation candidate $a := \nabla c_I(x),\ \nabla c_I(x) \neq 0$ is available. In order to be consistent with the semi-normal approach, we write

$$\begin{aligned} A_+ &:= \begin{bmatrix} A \\ a^\top \end{bmatrix}, \\ L_+ &:= \begin{bmatrix} L & 0 \\ l^\top & q \end{bmatrix}, \text{ where } l \in \mathbb{R}^m,\ q \in \mathbb{R}, \end{aligned}$$

and stipulate

$$A_+ A_+^\top = L_+ L_+^\top.$$

Suppose the semi-normal equation, $LL^\top = AA^\top$, holds exactly. Then equating the corresponding expressions gives

$$\begin{aligned} \begin{bmatrix} L & 0 \\ l^\top & q \end{bmatrix} \begin{bmatrix} L^\top & l \\ 0 & q \end{bmatrix} &= \begin{bmatrix} LL^\top & Ll \\ l^\top L^\top & \|l\|_2^2 + q^2 \end{bmatrix} \\ &= \begin{bmatrix} AA^\top & Aa \\ a^\top A^\top & a^\top a \end{bmatrix} \end{aligned}$$

and thus yields the conditions

$$\begin{cases} l = L^{-1} Aa \\ q = \sqrt{a^\top a - \|l\|_2^2}. \end{cases}$$

The term underneath the square root can be reformulated as follows

$$\begin{aligned} a^\top a - \|l\|_2^2 &= a^\top a - a^\top A^\top (LL^\top)^{-1} Aa \\ &= a^\top \left[I - A^\top (AA^\top)^{-1} A \right] a. \end{aligned}$$

Note that the middle matrix $I - A^\top (AA^\top)^{-1} A$ represents the orthogonal projection onto $(\mathcal{R}(A^T))^\perp = \mathcal{N}(A)$. Consequently, the term underneath the square root remains positive, if $a \notin \mathcal{N}(A)^\perp = \mathcal{R}(A^\top)$. This condition should never be violated throughout the Quasi-Newton algorithm, as it must be fulfilled in a vicinity of a potential solution (LICQ).

However, it may well occur that the semi-normal approximation is not sufficiently accurate to ensure positivity of the radicand. In this case, we propose the following intermediate update on L as a remedy.

Space Dilation[6]. Assume the corresponding radicand is negative, i.e.

$$a^\top a < \|l\|_2^2.$$

Define $H := LL^\top$ and $v := Aa$. In order to retrieve a positive radicand, we propose to perform the following rank-one update

$$\begin{aligned} H_0^{-1} &:= H^{-1} + \beta \frac{H^{-1} v v^\top H^{-1}}{v^\top H^{-1} v}, \quad \beta \in \mathbb{R} \\ &= L^{-\top} \underbrace{\left(I + \beta \frac{l l^\top}{l^\top l} \right)}_{\text{space dilation operator}} L^{-1} \end{aligned} \tag{37}$$

with damping factor β, such that

$$v^\top H_0^{-1} v = \alpha \cdot (a^\top a), \quad \alpha \in (0; 1),$$

holds for the intermediate semi-normal approximation $H_0 := L_0 L_0^\top$. This can be achieved by choosing $\beta = \alpha \frac{a^\top a}{v^\top H^{-1} v} - 1 \ \in (0; -1)$. It should be remarked, that this particular choice of β does retain positive definiteness of H_0 since (cf. the Interlacing Eigenvalue Theorem A.4)

$$\det H_0^{-1} = \det \left(L^{-1} \right)^2 (1 + \beta) > 0. \tag{38}$$

From the update formula (37) we obtain

$$\begin{aligned} H_0 &= L \left(I - \frac{\beta}{(1+\beta) l^\top l} l l^\top \right) L^\top \\ &= LL^\top - \frac{\beta}{(1+\beta) l^\top l} v v^\top, \end{aligned} \tag{39}$$

[6] an operator $R_\delta, \delta > 0$ of the form $R_\alpha(v) = I + (\delta - 1) v v^\top$ for $v \in \mathbb{R}^n : \|v\| = 1$, is occasionally called 'space dilation', see Shor [22]

by applying the Sherman-Morrison-Woodbury formula. Again we may find a Cholesky factor L_0 of H_0 using the techniques presented in the subsequent section 6.3.

Now we are able to successfully apply the former procedure to enlarge L_0. To accomplish the activation routine, we still need to consider the necessary changes in the mixed objects. Owing to a constraint activation, L^{-1} gains a new column and line:

$$L_+^{-1} = \begin{bmatrix} L^{-1} & 0 \\ -\frac{1}{q} l^\top L^{-1} & \frac{1}{q} \end{bmatrix},$$

whereas $\bar{S}$ is only enlarged by one additional line defined by $c_I'(x)S$:

$$\bar{S}_+ = \begin{bmatrix} \bar{S} \\ a^\top S \end{bmatrix}.$$

The resulting update on the mixed object is thus given by

$$(L^{-1}\bar{S})_+ = L_+^{-1}\bar{S}_+ = \begin{bmatrix} L^{-1}\bar{S} \\ y^\top \end{bmatrix}, \text{where } y^\top := -\frac{1}{q}(l^\top L^{-1}\bar{S} - a^\top S).$$

Finally, we derive

$$\begin{aligned} (\bar{S}^\top (LL^\top)^{-1}\bar{S})_+ &= (L^{-1}\bar{S})_+^\top (L^{-1}\bar{S})_+ \\ &= \begin{bmatrix} (L^{-1}\bar{S})^\top & y \end{bmatrix} \begin{bmatrix} (L^{-1}\bar{S}) \\ y^\top \end{bmatrix} \\ &= \bar{S}^\top (LL^\top)^{-1}\bar{S} + yy^\top. \end{aligned}$$

Permutations. Suppose we want to exchange two active constraints $\pi(j)$ and $\pi(k)$, with $j, k \in \mathcal{W}, j \neq k$. Let $P = P^\top \in \mathbb{R}^{m \times m}$ denote the corresponding permutation matrix. Consequently the semi-normal approximation is demanded to satisfy

$$L_+ L_+^\top = PLL^\top P \; (\approx PA(A^\top P)),$$

where the approximate equation at the right mirrors the semi-normal property of $LL^\top$. Using the definitions

$$\begin{cases} r := e_j - e_k \; \in \mathbb{R}^m, \\ \rho := l_k - l_j \; \in \mathbb{R}^m, \quad \text{where } l_i^\top := \; i\text{-th line of L} \end{cases}$$

we can reformulate the desired permutation as a rank-two update on the semi-normal approximation:

$$\begin{aligned} L_+ L_+^\top &= PLL^\top P \\ &= (L + r\rho^\top)(L + r\rho^\top)^\top \\ &= LL^\top + (L\rho)r^\top + r(L\rho)^\top + (\rho^\top \rho) rr^\top. \end{aligned} \tag{40}$$

Again, we may apply one of the low-rank Cholesky updating techniques from the forthcoming section 6.3 to obtain the triangular factor L_+.

Yet, implicit modifications on $\bar{S}$ and $\bar{W}$ still need to be discussed. Therefore suppose $L = L_+$ and contemplate necessary changes on $L^{-1}\bar{S}$ through changes in $\bar{S}$. $\bar{S}$ is supposed to approximate $c'_{\mathcal{W}}(x)S$, so it is reasonable to permute lines j and k of $\bar{S}$. Written as a rank-one update on $\bar{S}$, we find

$$\bar{S}_+ = \bar{S} + r\rho^\top,$$

where r and ρ are defined analogously to the permutation update on L:

$$\begin{cases} r := e_j - e_k \ \in \mathbb{R}^m, \\ \rho := \bar{s}_k - \bar{s}_j \ \in \mathbb{R}^l, \quad \text{where } \bar{s}_i^\top := \ i\text{-th line of } \bar{S}. \end{cases}$$

Since we do not want to store $\bar{S}$ itself, we simply exploit the relation:

$$L(L^{-1}\bar{S}) = \bar{S},$$

to obtain the j-th line of $\bar{S}$ according to $\bar{s}_j = l_j^\top L^{-1}\bar{S}$. Finally we apply (32) and (33) to obtain $L^{-1}\bar{S}_+$ and $\bar{S}_+^\top (LL^\top)^{-1}\bar{S}_+$

Deactivation. Suppose we would like to remove constraint $I \in \mathcal{I} \cap \mathcal{W}$ from the current working set $\mathcal{W}$. With the help of the permutation routine, we can assume w.l.o.g. $I = \pi(m)$, i.e. the 'last' constraint $c_{\pi(m)}$ is deactivated. The resulting semi-normal update simply comprises removal of the m-th column and line of L, such that $L_+ \in \mathbb{R}^{(m-1)\times(m-1)}$. In a similar fashion, we obtain $\bar{S}_+$ by discarding the last line (denoted by $y^\top$) from $\bar{S}$ and $\left(\bar{S}^\top (LL^\top)^{-1}\bar{S}\right)_+$ can be easily obtained by a downdate on $\bar{S}^\top (LL^\top)^{-1}\bar{S}$ using

$$\left(\bar{S}^\top (LL^\top)^{-1}\bar{S}\right)_+ = \bar{S}^\top LL^\top \bar{S} - yy^\top.$$

In either case, activation (including space dilation) or deactivation, mixed objects involving the matrix $\bar{W}$ can be updated analogously.

Computational complexity. Excluding the computational effort for the rank-one or rank-two update of the semi-normal approximation resulting from a space dilation or permutation, we can bound the effort for the maintenance of the data structure subject to working set changes by the order $\mathcal{O}(ln + m^2) + 3 \cdot OPS(\text{eval}(c_{\mathcal{W}})))$. If no space dilation is necessary to derive L_+, the latter addend vanishes completely.

6.3 Low-Rank Cholesky Updating

In the previous sections, we have excluded the problem of a low-rank update of the semi-normal approximation $LL^\top$ arising from a BFGS-update (35), a space dilation (39) or a constraint permutation (40). We will now discuss two different strategies for an update of a Cholesky factorization subject to a low-rank modification since we do not want to recompute the decomposition from scratch which would require cubic effort! Moreover, existence of the Cholesky factorization is guaranteed, if the update is a priori known to be positive definite. In the subsequent section, we will directly apply the results in order to arrive at a highly efficient way to adjust our data structure.

6.3.1 Cholesky Update via Rotations

Let $T \in \mathbb{R}^{m\times m}$ be symmetric positive definite with Cholesky factor $L \in \mathbb{R}^{m\times m}$

$$T = LL^\top,$$

which is uniquely determined up by to sign changes in the columns. Further suppose T undergoes the following symmetric rank-two perturbation, i.e.

$$\begin{aligned} T_+ &= T + \delta v r^\top + \delta r v^\top + \gamma\delta^2 r r^\top \quad (\gamma, \delta \in \mathbb{R}) \\ &= LL^\top + \delta v r^\top + \delta r v^\top + \gamma\delta^2 r r^\top \\ &= LL^\top + XCX^\top, \end{aligned} \tag{41}$$

where $\begin{cases} X := \begin{bmatrix} r & v \end{bmatrix} \in \mathbb{R}^{m\times 2}, \quad C := \begin{bmatrix} \gamma\delta^2 & \delta \\ \delta & 0 \end{bmatrix} \in \mathbb{R}^{2\times 2}. \end{cases}$

The idea consists of first updating L in the framework of a Cholesky rank-2 perturbation using the strategy presented in [24]. After some rather technical but straightforward manipulations, we obtain

$$L_+L_+^\top = L[I + \delta\tilde{v}\tilde{r}^\top]\left[I + \frac{\delta^2(\gamma - \|\tilde{v}\|^2)}{(1+\delta\tilde{v}^\top\tilde{r})^2}\tilde{r}\tilde{r}^\top\right][I + \delta\tilde{v}\tilde{r}^\top]^\top L^\top, \tag{42}$$

where the following definitions have been included:

$$\begin{cases} \tilde{v} := L^{-1}v \quad \text{and} \\ \tilde{r} := L^{-1}r. \end{cases}$$

Provided the middle matrix is positive definiteness, there exists a nonsingular square root

$$\left[I + \frac{\delta^2(\gamma - \|\tilde{v}\|^2)}{(1+\delta\tilde{v}^\top\tilde{r})^2}\tilde{r}\tilde{r}^\top\right]^{\frac{1}{2}}.$$

In the general case, we have to find a sufficiently small δ which ensures this property. However, we have already shown that our semi-normal updates originating from the modifications (35), (39) and (40) ensure positive definiteness of $L_+L_+^\top$, such that the middle matrix in (42) is necessarily positive definite. In the latter case, the new Cholesky factor can be written as follows:

$$L_+ = (L + rw^\top)G_1^\top G_2^\top, \tag{43}$$

where w can be determined as a linear combination of $\tilde{r}$ and $\tilde{v}$, and G_1 and G_2 denote two particular sequences of Givens rotations. The first sequence G_1 is supposed to rotate w into a multiple of the first unit vector, i.e. $G_1 w = \alpha e_1, \alpha \in \mathbb{R}$, only employing Givens rotations on adjacent components such that

$$(L + rw^\top)G_1^\top = LG_1^\top + \alpha v e_1^\top$$

is a lower Hessenberg matrix. G_2 is thus determined by the Givens sequence reconverting the factor $(L + rw^\top)G_1^\top$ into lower triangular form

$$L_+ = (L + rw^\top)G_1^\top G_2^\top,$$

for details see [24]. Naturally, neither of the Givens matrices forming G_1 and G_2 are computed explicitly, we rather directly rotate the left side. The computational cost of determining and applying the above sequence of $2 \cdot (m-1)$ Givens rotations to a column vector of dimension m is bounded by order $\mathcal{O}(m)$ such that the total complexity for the update of L is of order $\mathcal{O}(m^2)$. Hence it will be assumed that L_+ is already available and we will now focus on the update of the mixed objects.

Efficient adjustment of the KKT-matrices. Applying (43), we obtain

$$\begin{aligned} L_+^{-1}\bar{S} &= \left[(L + rw^\top)G_1^\top G_2^\top\right]^{-1}\bar{S} \\ &= G_2 G_1 \underbrace{\left(L + rw^\top\right)^{-1}}_{(I+\tilde{r}w^\top)^{-1}L^{-1}}\bar{S} \\ &= G_2 G_1 (I - \tilde{\delta}^{-1}\tilde{r}w^\top)L^{-1}\bar{S}, \quad \text{with} \quad \tilde{\delta} := 1 + \tilde{r}^\top w \\ &= G_2 G_1 L^{-1}\bar{S} - \tilde{\delta}^{-1}\tilde{r}w^\top L^{-1}\bar{S}. \end{aligned}$$

Note that we have stored $L^{-1}\bar{S} \in \mathbb{R}^{m\times l}$ (as well as $w, \tilde{r}$). Hence the cost of computing the left addend is bounded by $\mathcal{O}(lm)$. Again, the update of $L^{-1}\bar{W}$ can be performed likewise.

For the derivation of $\bar{S}^\top(L_+L_+^\top)^{-1}\bar{S}$, we apply the Sherman-Morrison-Woodbury formula (A.3) to obtain

$$\begin{aligned} \bar{S}^\top(L_+L_+^\top)^{-1}\bar{S} &= \bar{S}^\top\left(LL^\top + XCX^\top\right)^{-1}\bar{S} \\ &= \bar{S}^\top L^{-\top}\underbrace{(I + \tilde{X}C\tilde{X}^\top)^{-1}}_{=I-\tilde{X}\tilde{C}^{-1}\tilde{X}^\top}L^{-1}\bar{S} \\ &= \bar{S}^\top(LL^\top)^{-1}\bar{S} - \bar{S}^\top L^{-\top}\tilde{X}\tilde{C}^{-1}\tilde{X}^\top L^{-1}\bar{S}, \end{aligned} \tag{44}$$

with

$$\begin{cases} \tilde{X} := L^{-1}X \in \mathbb{R}^{m\times 2} \\ \tilde{C} := C^{-1} + \tilde{X}^\top\tilde{X} \in \mathbb{R}^{2\times 2}. \end{cases}$$

Note that nonsingularity of $L_+L_+^\top$ ensures existence of the involved matrices. Analogous procedures can be applied to mixed objects involving $\bar{W}$.

The upper approach is applicable if a permutation or a space dilation shall be performed. By contrast, the BFGS update (35) does not admit the form (41). Instead, we obtain two symmetric rank-one modifications[7]. In this case, we may apply a simpler procedure based on a similar application of two Givens sequences, cf. [17].

[7] i.e. the matrix C is diagonal

6.3.2 Bennett's Algorithm

Owing to the continuity argument explained in [15], this paragraph contains an application of the classical method by Bennett [2] to update the Cholesky factor L that does not require rotations. Having obtained the update L_+, we will be able to show that the corresponding mixed objects can be updated with the same minimal effort as in the previous section, but without utilizing any Givens matrices.

Bennett's algorithm. Bennett's algorithm, first introduced in [2], represents a fast (i.e. quadratic complexity in the problem dimension) method to update a general triangular decomposition

$$T = LDM^\top$$

subject to a low-rank modification. Stange et al. [25] have specified this method for the rank-one case.

In the following, Bennett's algorithm will be specified for the symmetric case where a matrix $T = LDL^\top \in \mathbb{R}^{m\times m}$ undergoes a symmetric rank-k modification:

$$T_+ = T + XCX^\top, \tag{45}$$

where $X \in \mathbb{R}^{n\times k}, C = C^\top \in \mathbb{R}^{k\times k}$. The aim is to obtain the new factorization

$$T_+ = L_+ D_+ L_+^\top$$

at minimal cost. The main step consists of demonstrating how the problem can be reduced to dimension $(m-1)\times(m-1)$. Therefore, partition L,X and D as follows:

$$L = \begin{bmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{bmatrix}, \quad X = \begin{bmatrix} X_1^\top \\ X_2^\top \end{bmatrix}, \quad D = \begin{bmatrix} d_1 & 0 \\ 0 & D_{m-1} \end{bmatrix}$$

with $X_1 \in \mathbb{R}^k, X_2^\top \in \mathbb{R}^{(m-1)\times k}, L_{11}, d_1 \in \mathbb{R}$ and $D_{m-1} \in \mathbb{R}^{(m-1)\times(m-1)}$ diagonal. Furthermore, write

$$\begin{aligned} T &= \begin{bmatrix} 1 & 0 \\ L_{21} & L_{22} \end{bmatrix} \begin{bmatrix} d_1 & 0 \\ 0 & D_{m-1} \end{bmatrix} \begin{bmatrix} 1 & L_{21}^\top \\ 0 & L_{22}^\top \end{bmatrix} \\ &= \begin{bmatrix} d_1 & d_1 L_{21}^\top \\ d_1 L_{21} & d_1 L_{21} L_{21}^\top + \tilde{T} \end{bmatrix}. \end{aligned} \tag{46}$$

Hereby, $\tilde{T} := L_{22} D_{n-1} L_{22}^\top$ denotes the Schur complement of T. Now, by making the ansatz

$$T^+ = \begin{bmatrix} 1 & 0 \\ L_{21}^+ & I_{m-1} \end{bmatrix} \begin{bmatrix} d_1^+ & 0 \\ 0 & \tilde{T}^+ \end{bmatrix} \begin{bmatrix} 1 & {L_{21}^+}^\top \\ 0 & I_{m-1} \end{bmatrix}$$

and equating this expression to T^+ from (45), where T has been replaced by (46), one obtains the following conditions:

$$\begin{cases} d_1^+ = d_1 + X_1^\top C X_1 \\ L_{21}^+ = \frac{1}{d_1^+}(d_1 L_{21} + X_2^\top C X_1) \\ \tilde{T}^+ = \tilde{T} + d_1 L_{21} L_{21}^\top + X_2^\top C X_2 - d_1^+ L_{21}^+ {L_{21}^+}^\top. \end{cases} \tag{47}$$

Lemma 6.1.
The Schur complement $\tilde{T}^+ \in \mathbb{R}^{(m-1)\times(m-1)}$ *of* T^+ *can be represented as a symmetric rank-k update on* $\tilde{T}$*, i.e.*

$$\tilde{T}^+ = \tilde{T} + X^+ C^+ {X^+}^\top$$

for certain $C^+ \in \mathbb{R}^{k\times k}$ *and* $X^+ \in \mathbb{R}^{(m-1)\times k}$.

Proof.

$$\begin{aligned}\tilde{T}^+ &= \tilde{T} + d_1 L_{21} L_{21}^\top + X_2^\top C X_2 - d_1^+ L_{21}^+ {L_{21}^+}^\top \\ &= \tilde{T} + \left(d_1 - \tfrac{d_1^2}{d_1^+}\right) L_{21} L_{21}^\top + X_2^\top C X_2 \\ &\quad - \tfrac{d_1}{d_1^+} b L_{21}^\top - \tfrac{d_1}{d_1^+} L_{21} b^\top - \tfrac{1}{d_1^+} b b^\top\end{aligned}$$

with $b := X_2^\top C X_1$. With the definitions

$$X^+ := -L_{21} X_1^\top + X_2^\top \quad \text{and} \quad C^+ := C - \tfrac{1}{d_1^+} C X_1 X_1^\top C \tag{48}$$

we obtain the desired result after some straightforward but rather tedious manipulations. □

Thus we are able to repeat the upper procedure by decomposing

$$\tilde{T}^+ = L_{22} D_{n-1} L_{22}^\top + X^+ C^+ {X^+}^\top \quad \in \mathbb{R}^{(m-1)\times(m-1)},$$

in an analogous way, such that we obtain a recursive procedure to determine the LDL^T-decomposition of T_+ at a computational cost of order $\mathcal{O}(m^2)$, given that k stays comparatively low. (normally, we have $k = 1$ or $k = 2$)

Stability. In order to maintain stability, the situation in which the algorithm encounters a zero pivot d_i needs to be excluded. For this purpose, positive definiteness of T_+ (such is the case in our applications) is sufficient.

Application. In our case ($k = 2$), one might be tempted to define T as $T := LL^\top$. This leads to the problem that the update of $L^{-1}\bar{S}$ will not be performed with minimal effort, albeit L_+ is obtained in $\mathcal{O}(m^2)$ floating point operations. However, computation of $L_+^{-1}\bar{S}$ would require solving l triangular systems leading to a complexity of order $\mathcal{O}(l \cdot m^2)$. In the sequel, it will be demonstrated how one can reduce the cost of updating the mixed object $L^{-1}\bar{S}$ to order $\mathcal{O}(lm)$.

First of all, the application of Bennett's algorithm will be reduced to the case where $T = I$:

$$\begin{aligned}L_+ L_+^\top &= LL^\top + \begin{bmatrix} r & v \end{bmatrix} C \begin{bmatrix} r^\top \\ v^\top \end{bmatrix} \\ &= L\left(I + L^{-1}\begin{bmatrix} r & v \end{bmatrix} C \left(L^{-1}\begin{bmatrix} r & v \end{bmatrix}\right)^\top\right) L^\top \\ &= L(I + XCX^\top)L^\top, \end{aligned} \tag{49}$$

where the new X is given by

$$X := \begin{bmatrix} \tilde{r} & \tilde{v} \end{bmatrix} := L^{-1} \begin{bmatrix} r & v \end{bmatrix} \in \mathbb{R}^{m\times 2}.$$

Then, one attempts to find the $LDL^\top$-factorization of the middle matrix (=: $\bar{L}\bar{D}\bar{L}^\top$), which constitutes a rank-two perturbation of the identity:

$$I + XCX^\top = \bar{L}\bar{D}\bar{L}^\top.$$

By virtue of (49) and nonsingularity of L, the middle matrix is positive definite, provided that $L_+L_+^\top$ is positive definite. Consequently, the occurrence of nonpositive pivots $\bar{d}_i$ is theoretically excluded. If we encounter a nonpositive pivot, we can conclude, that the semi-normal modification has led to a Jacobian approximation which does not fulfill the full rank requirement.

The main benefit in this approach lies in the fact that we can directly derive a low-rank representation for $\bar{L}$, when using Bennett's algorithm. In order to justify this claim, let $\bar{l}_i \in \mathbb{R}^{m-i}$ denote the subdiagonal elements of the i-th column of $\bar{L}$. Evidently, the main diagonal consists of ones:

$$\bar{L} = \begin{bmatrix} 1 & & & & \\ & \ddots & & \mathbf{0} & \\ & & 1 & & \\ \bar{l}_1 & \cdots & \bar{l}_i & \ddots & \\ & & & & 1 \end{bmatrix}$$

Using (47) and (48), one obtains at the i-th step the intermediate quantities for $i = 1, .., m$:

$$\begin{cases} {X_1^i}^\top = \begin{bmatrix} \tilde{r}_i & \tilde{v}_i \end{bmatrix} \in \mathbb{R}^2 \\ {X_2^i}^\top = \begin{bmatrix} \tilde{r}_{i+1} & \tilde{v}_{i+1} \\ \vdots & \vdots \\ \tilde{r}_m & \tilde{v}_m \end{bmatrix} \in \mathbb{R}^{(m-i)\times 2} \\ \bar{d}_i = 1 + {X_1^i}^\top C^i X_1^i \in \mathbb{R}, \end{cases}$$

where $C^i \in \mathbb{R}^{2\times 2}$ is given by the following iterative definition:

$$\begin{cases} C^1 = C, \\ C^{i+1} = C^i - \frac{1}{\bar{d}_i}(C^i X_1^i)(C^i X_1^i)^\top, \ (i = 1, \ldots, m-1). \end{cases} \tag{50}$$

Note that the only changes effected on X^i, are the continuous removal of the first line and the border move down by one line between ${X_1^i}^\top$ and ${X_2^i}^\top$. Now, writing

$$\begin{bmatrix} \alpha_i \\ \beta_i \end{bmatrix} := \frac{1}{\bar{d}_i} C^i X_1^i \quad \in \mathbb{R}^2,$$

for $i = 1, \ldots, m$, we reformulate (50) according to

$$\begin{cases} C^1 = C, \\ C^{i+1} = C^i - \bar{d}_i \begin{bmatrix} \alpha_i^2 & \alpha_i\beta_i \\ \alpha_i\beta_i & \beta_i^2 \end{bmatrix}. \end{cases}$$

and obtain the subdiagonal columns $\bar{l}_i$ of $\bar{L}$ following (47) as

$$\bar{l}_i = \begin{bmatrix} \alpha_i\tilde{r}_{i+1} + \beta_i\tilde{v}_{i+1} \\ \vdots \\ \alpha_i\tilde{r}_m + \beta_i\tilde{v}_m \end{bmatrix} \in \mathbb{R}^{m-i}.$$

Finally, we are to able to derive an important low-rank representation for the intermediate triangular factor:

$$\bar{L} = I + \left[\tilde{r}\alpha^\top + \tilde{v}\beta^\top\right]_{i>j}.$$

It is worth remarking, that the solution of systems involving $\bar{L}$ requires only $\mathcal{O}(m)$ floating point operations using forward substitution. The following practical algorithm can be formulated, to solve systems of the form $\bar{L}x = b$:

Algorithm 3 (Solve $\bar{L}x = b$)**.**

```
set  σ_α, σ_β = 0, x_1 = b_1
for  i = 2, .., m
     σ_α ← σ_α + α_{i-1} x_{i-1}
     σ_β ← σ_β + β_{i-1} x_{i-1}
     x_i = b_i − σ_α r̃_i − σ_β ṽ_i
end
```

We remark that a similar algorithm of the same complexity class can be formulated for the product $\bar{L}x$.

Efficient adjustment of the KKT-matrices. In order to pick up on the issue of updating the mixed objects, we seize on equation (49) to retrieve

$$\begin{aligned} LL^\top + \begin{bmatrix} r & v \end{bmatrix} C \begin{bmatrix} r^\top \\ v^\top \end{bmatrix} &= L(I + XCX^\top)L^\top \\ &= L\bar{L}\bar{D}\bar{L}^\top L^\top \\ &= L_+ L_+^\top, \text{ with } L_+ := L\bar{L}\bar{D}^{\frac{1}{2}}. \end{aligned}$$

Note that we have now determined a triangular factor L_+ at order $\mathcal{O}(m^2)$. Furthermore, we obtain the desired modification on the mixed objects according to

$$L_+^{-1}\bar{S} = \bar{D}^{-\frac{1}{2}}\bar{L}^{-1}\left(L^{-1}\bar{S}\right).$$

This involves solving l linear systems in $\bar{L}$ which does not require more than $\mathcal{O}(lm)$ floating point operations, cf. Algorithm 3, without using any rotations! For the update of $\bar{S}^\top(L_+L_+^\top)^{-1}\bar{S}$, we can simply use the approach (44) suggested in the previous section.

The following summarizing proposition has thus been shown.

Proposition 6.2 (Low-rank semi-normal modification)**.**
The computational complexity for an update of the data structure resulting from a rank-two modification of the semi-normal approximation can be bounded by the order $\mathcal{O}(m \cdot \max(m, l))$. There exists a rotation-free updating procedure, which fulfills this complexity bound.

6.4 Main Result

The subsequent central theorem reflects the main result and concludes the section by giving a total operations count for L-SR1-SN which should comprise all operations necessary to retrieve the matrices involved in solving the KKT-system: step computation, Hessian update, updating the semi-normal approximation, mixed updates and working set changes. Being compulsory for any line search-based Newton-type optimizer, the cost of computing the right side of the KKT-system (23) and the cost of the step size determination is not specific to L-SR1-SN and is thus not accounted for. Otherwise, a couple of objective and constraint function evaluations have to be added to the operations count given below.

Theorem 6.3 (L-SR1-SN - total cost-per-KKT-solution)**.**
The computational cost-per-iteration for solving the KKT-system with L-SR1-SN can be restricted to an operations count of order

$$\mathcal{O}(ln + m \cdot \max(m, l) + l^3) + t \cdot OPS(\text{eval}(c)),$$

where $\mathbb{N} \ni t = \mathcal{O}(1), t \leq 31$, *is a fixed constant. The entire updating procedure can be effected without rotations or reflections.*

Proof. The left addend is an immediate consequence of the complexity results of the corresponding updates, provided $m \leq n$ (LICQ). However, the number of constraint function evaluations can be further diminished by reorganizing derivative calculations in the inner loop as follows:
First, we consecutively compute the two Jacobian-vector and Jacobian-transposed vector products needed for the step calculation (28). W.l.o.g., suppose a descent direction has been found. (Otherwise, the inner loop would be repeated and Hessian updates would not have to be accounted for.) Secondly, after moving to the next iterate x_+, we simultaneously propagate the two directions $c'_{\mathcal{A}}(x_+)^\top \sigma, c'_{\mathcal{A}}(x_+)^\top L^{-\top} \sigma$ from the BFGS-update and the orthogonality condition by employing the vector mode of AD, cf. Theorem 2.7. Thirdly, we may perform the four tangent propagations from the Hessian and mixed updates (Alternative 2) as well as the remaining expression $c'_{\mathcal{A}}(x_+)v, v := c'_{\mathcal{A}}(x_+)^\top \sigma$ from the BFGS-update (35) simultaneously. Adding up the corresponding evaluation multipliers yields $t = 28$. Finally, we have to account for a possible space dilation which comprises computation of another Jacobian-vector product, leading to a maximum of $t = 31$. □

Corollary 6.4 (Total cost-per-iteration)**.**
Suppose $\iota > 1$ *inner loop iterations are necessary to find a descent direction. Then the complexity bound of one outer loop iteration roughly equals* ι *times the upper complexity bound.*

Theorem 6.5 (Semi-normal method - memory space requirement)**.**
L-SR1-SN can be applied with a memory space of maximal dimension

$$m^2 + \mathcal{O}(nl).$$

Proof. The assertion follows directly from the fact that the data structure given in Table 1 suffices to retrieve the KKT-solution according to (28). □

We immediately underscore the distinct reduction in required memory as L-SR1-SN does not require any objects of size $\mathcal{O}(m \cdot n)$ in comparison to conventional Quasi-Newton methods or even the latest ZEDISDEAD implementation! The computational effort of order $\mathcal{O}(m \cdot n)$ from ZEDISDEAD is replaced by an extended usage of algorithmic differentiation incorporating the Jacobian of active constraints, thus necessitating a slight overhead with regard to the number of constraint evaluations per iteration.

7 Maintaining Positive Definiteness of the L-SR1 Hessian

By virtue of the second order sufficiency criterion, Theorem 3, and Theorem 3.3, the Hessian of the Lagrangian should be positive definite on the null space of the Jacobian of active constraints. It is highly desirable to maintain this property for the Hessian approximation B. Since the (L-)SR1-method does not preserve positive definiteness of $Z^\top BZ$ in the first place, this section is dedicated to finding restrictions on the initialization scalar γ which guarantee positive definiteness of the L-SR1 approximation $B = B(\gamma)$.

To this end, we recall the definitions of the middle matrices needed for $B(\gamma)$ and $B(\gamma)^{-1}$, respectively:

$$\begin{cases} M = M(\gamma) = P - \gamma S^\top S \ \in \mathbb{R}^{l\times l} \\ N = N(\gamma) = Q - \gamma^{-1} W^\top W \ \in \mathbb{R}^{l\times l}. \end{cases}$$

We begin with an algebraic lemma describing the influence of γ on the inertia of $B(\gamma)$.

Lemma 7.1 (Index functions)**.**
If Q is positive definite, then the following assertions hold true:

(i) There exists a $\Gamma > 0$, such that $\forall\, \gamma > \Gamma$ it holds: $B(\gamma)$ is positive definite.

(ii) If additionally $S^\top S$ is positive definite, the two index functions defined by

$$\begin{cases} (0,\infty) \ni \gamma \mapsto \operatorname{index}(-M(\gamma)) \in \{0,\dots,l\} \quad \textit{and} \\ (0,\infty) \ni \gamma \mapsto \operatorname{index}(N(\gamma)) \in \{0,\dots,l\}, \end{cases}$$

are monotonically decreasing.

Proof. (i): Positive definiteness of Q implies the existence of a Γ, such that $N = Q - (1/\gamma)W^\top W$ is positive definite $\forall\, \gamma > \Gamma$. Since N is the middle matrix in the compact representation of $B(\gamma)^{-1}$, cf. (18), we also obtain positive definiteness of $B(\gamma)^{-1}$ and thus of $B(\gamma)$.
(ii): Let C_Q denote a Cholesky factor of Q, i.e. $Q = C_Q C_Q^\top$. Sylvester's law of inertia (A.2) ensures that N and $\gamma I - C_Q^{-1} W^\top W C_Q^{-\top}$ share the same inertia. Since $\operatorname{spec}(\gamma I - C_Q^{-1} W^\top W C_Q^{-\top}) = \gamma \cdot (1,\dots,1) - \operatorname{spec}(C_Q^{-1} W^\top W C_Q^{-\top})$, the assertion is proven. The analogous reasoning can be applied to $-M$. □

In any case, we want to ensure that Q, which equals $S^\top \nabla^2_{xx} f(x) S (\succ 0)$[8] in quadratic programming, and $S^\top S$ remain positive definite throughout the algorithm. On one hand, this assumption is related to Lemma 7.1, on the other hand we want to prevent linear dependency among the primal steps stored in S. Iterates which destroy this important property will be discarded or replaced.

[8] provided the steps are linearly independent and the objective function f is strictly convex quadratic

One way to efficiently check positive definiteness is to update an existing $LDL^\top$ factorization (which will be shown to require negligible effort of order $\mathcal{O}(l^2)$) and to verify that all main diagonal entries forming D_+ are strictly positive.

Positive definiteness of Q. For simplicity, write $S = [s_0, \dots, s_{l-1}]$ and $W = [w_0, \dots, w_{l-1}]$. Q, cf. (18), is defined by $Q = D + G + G^\top$ where

$$\begin{cases} D = \operatorname{diag}(s_0^\top w_0, \dots, s_{l-1}^\top w_{l-1}) \\ G_{ij} = \begin{cases} s_{i-1}^\top w_{j-1} & , \text{ if } i < j \\ 0 & , \text{ else.} \end{cases} \end{cases} \quad (G \in \mathbb{R}^{l \times l})$$

To investigate possible changes in the definiteness of Q, suppose that Q is positive definite and a suitable $LDL^\top$-factorization is available, i.e. $Q = L_Q D_Q L_Q^\top$. Further consider the potential Hessian updates mentioned in section 5 and their effects on Q.

Removal. Suppose the oldest secant pair (s_0, w_0) is removed from S and W, respectively. Consequently, Q_+ is obtained from Q by erasing the first column and line. The $LDL^\top$-factorization of Q_+ can be obtained as follows:

$$\begin{aligned} Q &=: \underbrace{\begin{bmatrix} 1 & 0 \\ l & \tilde{L} \end{bmatrix}}_{:= L_Q} \underbrace{\begin{bmatrix} d & 0 \\ 0 & \tilde{D} \end{bmatrix}}_{:= D_Q} \begin{bmatrix} 1 & l^\top \\ 0 & \tilde{L}^\top \end{bmatrix} \\ &= \begin{bmatrix} d & dl^\top \\ dl & dll^\top + \tilde{L}\tilde{D}\tilde{L}^\top \end{bmatrix} \end{aligned}$$

Equating the $(2,2)$-blocks yields a rank-one Cholesky modification:

$$Q_+ = \tilde{L}\tilde{D}\tilde{L}^\top + dll^\top,$$

The new factorization $Q_+ = L_Q^+ D_Q^+ (L_Q^+)^\top$ can be found with complexity $\mathcal{O}(l^2)$ using one of the algorithms introduced in section 6.3. Note that Q_+ always stays positive definite, since

$$Q_+ = XQX^\top \quad \text{for} \quad X := \begin{bmatrix} \mathbf{0}_{l-1} & I_{l-1} \end{bmatrix}.$$

Extension. Suppose a secant pair (s_l, w_l) is added to the limited memory matrices S and W. Consequently, Q gains a new line and column via $q = S^\top w_l$ and $\beta = s_l^\top w_l$.

$$\begin{aligned} Q_+ &= \begin{bmatrix} Q & q \\ q^\top & \beta \end{bmatrix} \\ &= \underbrace{\begin{bmatrix} L_Q & 0 \\ l_+^\top & 1 \end{bmatrix}}_{=: L_Q^+} \underbrace{\begin{bmatrix} D_Q & 0 \\ 0 & d_+ \end{bmatrix}}_{=: D_Q^+} \begin{bmatrix} L_Q^\top & l_+ \\ 0 & 1 \end{bmatrix} \end{aligned}$$

Equating the corresponding entries yields

$$\begin{cases} l_+ = D_Q^{-1} L_Q^{-1} q \\ d_+ = \beta - l_+^\top D_Q l_+ \end{cases}$$

and finally we have to verify that d_+ is strictly positive. Note that this is impossible, whenever the curvature $\beta = s_l^\top w_l$ is not positive!

Replacement. Assume a given secant pair (s_j, w_j) shall be substituted by a new pair (s'_j, w'_j). The corresponding rank-two update on Q is given by

$$Q_+ = Q + ve_j^\top + e_j v^\top - v_j e_j e_j^\top,$$

where e_j denotes the j-th cartesian unit vector and $v \in \mathbb{R}^l$ is defined by

$$v_i := \begin{cases} (S^\top w'_j)_i - Q_{ij} & , \text{ if } i < j \\ s_j'^\top w'_j - Q_{jj} & , \text{ if } i = j \\ (W^\top s'_j)_i - Q_{ij} & , \text{ if } i > j. \end{cases} \qquad (i = 1, ..., l)$$

Again, we can apply the Cholesky low-rank updating methods from section 6.3 to obtain Q_+.

The same procedures can obviously be conducted to ensure that S continues having full column rank. If either $S^\top S$ or Q loses its positive definiteness due to an arbitrary Hessian update, one might consider skipping the update. To summarize, we make the following

Remark 7.2 (Computational Complexity)**.**
The operations count for any of the Cholesky updates on Q and $S^\top S$ resulting from a typical Hessian modification, is bounded by the order $\mathcal{O}(l^2)$. As a consequence, we have to add the rather small matrices $L_{S^\top S}, D_{S^\top S}, L_Q, D_Q \in \mathbb{R}^{l \times l}$ to our data structure in memory.

A different strategy is proposed in [21]: Here a damping parameter, see (13), for the L-SR1-update is introduced and a compact representation formula with damping is derived. Appropriate conditions on the damping parameter ensure positive definiteness of Q.

γ-adjustment. As arbitrary updates on S and W may cause the loss of positive definiteness of $B(\gamma)$, we somehow need to restrict the initialization factor γ to retain this property. In [21], a sufficient augmentation of γ (i.e. $\gamma > \Gamma$) is suggested, thereby enforcing positive definiteness of $B(\gamma)$ according to Lemma 7.1, part (i). In view of the KKT-formulae (28) for (d, σ) however, it is desirable to keep variations in the scaling of $B(\gamma)$ as small as possible. Continuous augmentation of γ may for instance lead to numerical instabilities like blow-up in the Lagrange-multiplier λ. Additionally it is worth remarking, that choosing $\gamma > \Gamma$ leads to positive definiteness of the middle matrix N, which is not indispensable for positive definiteness of $B(\gamma)$. In order to obtain a less volatile adjustment criterion for γ, we state the following lemma.

Lemma 7.3 (Lehmann's criterion)**.**
$B(\gamma)$ is positive definite if and only if $-M(\gamma)$ and $N(\gamma)$ have the same inertia.

Proof. See [21]. □

Conclusion. Assuming positive definiteness of

$$\begin{aligned} Q &= C_Q C_Q^\top, \quad C_Q = L_Q D_Q^{\frac{1}{2}} \quad \text{and} \\ S^\top S &= C_{S^\top S} C_{S^\top S}^\top, \quad C_{S^\top S} = L_{S^\top S} D_{S^\top S}^{\frac{1}{2}}, \end{aligned}$$

we can enforce positive definiteness of $B(\gamma)$ by the following γ-adjustment strategy:

(i) Determine the spectrum of the γ-invariant $l \times l$ matrices $C_{S^\top S}^{-1} P C_{S^\top S}^{-\top}$ and $C_Q^{-1} W^\top W C_Q^{-\top}$ by an appropriate symmetric eigenvalue solver.

(ii) Determine the interval $I = (\gamma_0, \gamma_1) \subset \mathbb{R}^+, \gamma_0 > 0, \gamma_1 \in (\gamma_0, \infty]$, ensuring

$$\begin{cases} \operatorname{index}(-M(\gamma')) = \operatorname{index}(N(\gamma')), \\ \det M(\gamma') \neq 0 \neq \det(N(\gamma')) \end{cases} \qquad \forall\, \gamma' \in (\gamma_0, \gamma_1), \tag{51}$$

which is *closest* to the current initialization scalar γ.

(iii) Choose $\gamma_+ \in (\gamma_0, \gamma_1)$.

One can observe, that index jumps can only occur if the positive eigenvalues of the matrices $C_{S^\top S}^{-1} P C_{S^\top S}^{-\top}$ and $C_Q^{-1} W^\top W C_Q^{-\top}$ are 'exceeded' by continuous augmentation of γ starting from $\gamma = 0$. Consequently, we avoid testing equality of the indices within an interval where both indices remain constant. These intervals can be found easily by concatenating the two sets of eigenvalues in one vector. Afterwards they are sorted by ascending absolute value (cf. definition of 'v' in Algorithm 4). This is the basis for the following γ-adjustment Algorithm which computes an efficient shift in γ inducing positive definiteness of $B(\gamma_+)$. We write in short form for $v \in \mathbb{R}^n : \operatorname{ng}(v) := |\{i \in \{1, ..., n\} : v_i < 0\}|$.

Algorithm 4 (γ-adjustment).

```
set      quot ∈ (0,1), k = 1
compute  T_1 := C_{S^⊤S}^{-1} P C_{S^⊤S}^{-⊤}, T_2 := C_Q^{-1} W^⊤ W C_Q^{-⊤}
compute  spec(T_1), spec(T_2)
set      v_1 = −spec(T_1) + γ1_l, v_2 = −spec(T_2) + γ1_l
sort     v = [|v_1|, |v_2|] ∈ R^{2l} by ascending absolute value
for  (k = 2 ... (2l + 1))
   if  (k = 2l + 1)
       choose suitable △γ > v(2l)
       if (△γ < γ)
           set △γ = −△γ
       break
   set gap = v(k) − v(k − 1)
   set △γ = v(k − 1) + gap ∗ quot
   if  (△γ < γ)
       if  (ng(v_1 − △γ1_l) = ng(v_2 − △γ1_l))
           set △γ = −△γ
           break
   if  (ng(v_1 + △γ1_l) = ng(v_2 + △γ1_l)
           break
end
set γ_+ = γ + △γ
```

Practical application has shown, that choosing $quot = 0.5$ yields the most stable updates, as closeness to singularity should be avoided. Note that we prefer negative shifts $\triangle\gamma$ in order to prevent γ from blowing up. The computational effort for this algorithm is basically governed by the determination of the two spectra by an appropriate symmetric eigenvalue solver, which can be assumed to be bounded by the order $\mathcal{O}(l^3)$.

Example. Figure 7 illustrates a typical application provoked by non-equal indices resulting from a non-suitable γ. The example is taken from the application of L-SR1-SN with $l = 6$ to problem LUKVLI13 from the CUTEr test set [3]. At the seventh iteration, we are confronted with the following situation, which requires adjustment of γ:

$$\begin{aligned}
&\text{spec}(-C_{STS}^{-1} P C_{STS}^{-\top}) \approx \\
&\qquad [-3.81E{+}02, -2.63E{+}01, -2.51E{+}01, -6.07E{+}00, -2.48E{+}00, 8.14E{+}01] \\
&\text{spec}(-C_Q^{-1} W^\top W C_Q^{-\top}) \approx \\
&\qquad [-8.15E{+}02, -2.86E{+}02, -3.03E{+}01, -2.63E{+}01, -1.47E{+}01, -4.96E{+}00] \\
&\gamma = 39.8
\end{aligned}$$

Figure 1: Example: non-equal indices

As one can easily observe, there are two possible intervals for the choice of γ. The γ-augmentation approach from [21] would result in choosing $\gamma_+ (> \Gamma)$ belonging to the interval at the right margin. In this case however, a smaller and thus more efficient modification of γ is absolutely sufficient:

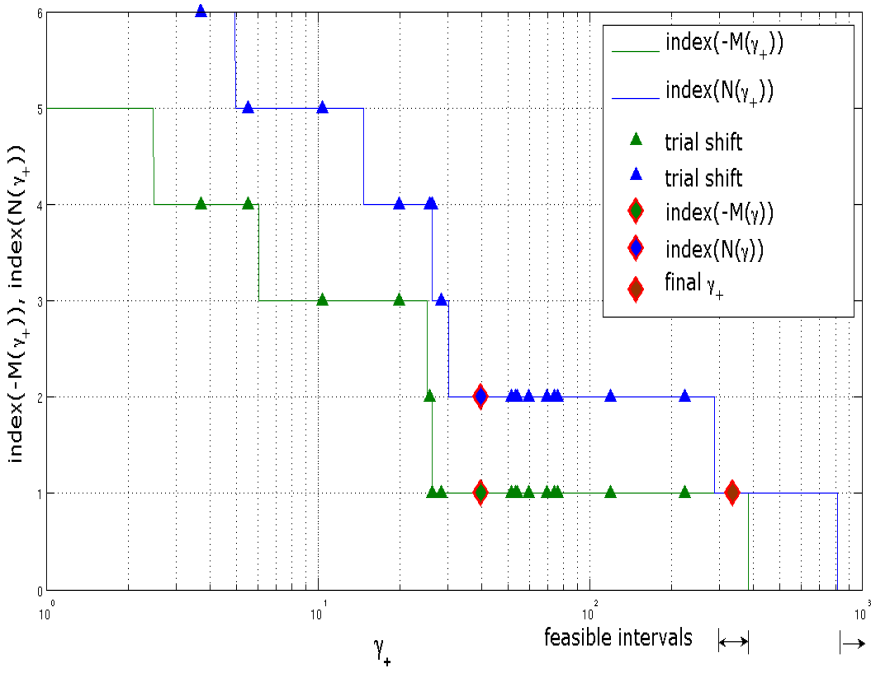

Figure 2: γ-Adjustment for LUKVLI13, $l = 6$, two feasible intervals

8 The LRAMBO Algorithm featuring L-SR1-SN

8.1 Merit Function and Line Search Strategy

The following table roughly sketches how L-SR1-SN can be embedded in the LRAMBO package employing the most successful combination of available options. Note the new features with regard to Algorithm 2.

Algorithm 5 (LRAMBO with L-SR1-SN)**.**

```
initialize  k = 0, x_0, λ_0, ε, L_0 = I_m, B_0 = γI_n
while   stopping criterion not fulfilled
    do
        solve   KKT-system (23) using (28) → [d_k, σ_k]
        if      descent w.r.t. merit function [AUGMLAGR]  break
        update  semi-normal factor L [SNBFGS]
        update  mixed objects [CR]
        adjust  working set if necessary
    while   no descent direction found and change in working set
    compute α_k by line search [ShaZ] on merit function [AUGMLAGR]
    update  x_{k+1} = x_k + α_k d_k, λ_{k+1} = λ_k + σ_k
    adjust  working set if necessary
    update  semi-normal factor L [SNBFGS]
    update  mixed objects [CR]
    update  Hessian [L-SR1], use Algorithm 4 to adjust
            initialization scalar γ, if necessary
    set     k → k + 1
end
```

CR: columnwise recomputation
SNBFGS: BFGS-update of the semi-normal Jacobian approximation

Step size determination. In our total Quasi-Newton framework, merit functions serve as guidelines for efficient step multipliers combining reduction in the objective function with feasibility. The main goal is to globalize the total Quasi-Newton algorithm, in that convergence from arbitrarily remote starting points can be achieved. For the simplified case of a known active set $\mathcal{A}$, we mention the most common representatives:
The classical non-differentiable *L1-penalty* function

$$\mathcal{L}^1(x;\mu) := f(x) + \frac{\mu}{2}\sum_{i\in\mathcal{A}} |c_i(x)|,$$

the differentiable *Augmented Lagrangian* function,

$$\mathcal{L}^A(x,\lambda;\mu) := f(x) + \sum_{i\in\mathcal{A}} \lambda^i c_i(x) + \frac{\mu}{2}\|c_{\mathcal{A}}(x)\|_2^2, \tag{52}$$

and the *Primal-Dual Augmented Lagrangian* function

$$\mathcal{L}^{PD}(x,\lambda;\mu_1,\mu_2) := \mathcal{L}(x,\lambda) + \frac{\mu_1}{2}\|\nabla_x \mathcal{L}(x,\lambda)\|^2 + \frac{\mu_2}{2}\|c_{\mathcal{A}}(x)\|^2,$$

with strictly positive penalty parameters μ,μ_1 and μ_2. As the Augmented Lagrangian function displays the best numerical results in conjuction with L-SR1-SN, we briefly summarize some theoretical properties of this well-known merit function. Provided the set of Lagrange multipliers $\{\lambda_k\}$ is bounded, the iterates x_k can be forced to tend to feasibility by sufficiently increasing μ in $\mathcal{L}^A$. To contrast the dichotomy between exactness and differentiability of ordinary penalty functions of the form

$$\begin{aligned} p_\mu(x) \quad &= \quad f(x) + \mu l(x), \text{ where} \\ l(x) \quad &\text{fulfills} \quad \begin{cases} l(x) = 0 & \text{, if } c_{\mathcal{A}}(x) = 0, \\ l(x) > 0 & \text{, else,} \end{cases} \end{aligned}$$

we state the following proposition dealing with the desirable *exactness* property of the Augmented Lagrangian function:

Proposition 8.1 (Exactness of $\mathcal{L}^A$)**.**
Let x_* be a local solution to (4) *where LICQ and SOSC are fulfilled at the KKT-point $(x_*,\lambda_*) \in \mathbb{R}^{n\times m}$. Then there is a $\bar{\mu} > 0$, such that $\forall\, \mu \geq \bar{\mu}$ it holds:*

x_* is a strict local minimizer of $\mathcal{L}^A(\,.\,,\lambda_*;\mu)$.

Proof. See [20]. □

The major improvement in comparison to the L1-penalty function is thus the possibility to obtain the solution of the constrained problem by solving a differentiable unconstrained problem without necessarily letting the parameter μ tend to infinity, given that the Lagrange multiplier estimate is sufficiently accurate. Indefinite augmentation of μ, which is likely to cause severe numerical instabilities due to a deteriorating condition of the reduced Hessian, can thus be circumvented. Under similar conditions, Proposition 8.1 can be further relaxed to fitting the more common case where we do not dispose of the adjoint solution λ_*, see again the text book by Nocedal and Wright [20]. In the presence of inequalities, the Augmented Lagrangian can locally be restricted to the equality-constrained case by a correct guess of the active set, otherwise we have to work with the current working set. Further extensions like inclusion of slack variables replacing the inequalities are possible. Another issue concerns the adaptation of the penalty parameters. In view of Theorem A.8, one may try to change μ, whenever the angle between the KKT-based direction (d,σ) and the steepest descent direction $(-\nabla_x \mathcal{L}^A(x,\lambda;\mu), -c_{\mathcal{A}}(x))$ tends towards orthogonality. Basic properties as well as further enhancement to the inequality constrained case of the Primal-Dual Augmented Lagrangian can be found in [10] and [5].

To link step computation in L-SR1-SN to the Augmented Lagrangian, consider

Proposition 8.2 (Descent property of L-SR1-SN)**.**
Suppose (d, σ) *is obtained by* (28)*. Then it holds that,* d *is a descent direction of* $\mathcal{L}^A(., \lambda_+; \mu)$ *at* x*, provided the Jacobian approximation is sufficiently accurate.*

Proof. The lower equation in (28) induces

$$\begin{aligned}
\nabla_x \mathcal{L}^A(x, \lambda_+; \mu)^\top d &= \nabla_x \mathcal{L}(x, \lambda_+)^\top d + \mu c_{\mathcal{A}}(x)^\top \nabla c_{\mathcal{A}}(x)^\top d \\
&= g(x, \lambda)^\top d + \sigma^\top \nabla c_{\mathcal{A}}(x)^\top d + \mu c_{\mathcal{A}}(x)^\top \nabla c_{\mathcal{A}}(x)^\top d \\
&= \left[-Bd - c'_{\mathcal{A}}(x)^\top \sigma\right]^\top d + \sigma^\top \nabla c_{\mathcal{A}}(x)^\top d \\
&\quad + \mu c_{\mathcal{A}}(x)^\top \nabla c_{\mathcal{A}}(x)^\top d \\
&= -d^\top B d + \mu c_{\mathcal{A}}(x)^\top \nabla c_{\mathcal{A}}(x)^\top d.
\end{aligned}$$

Taking account of Proposition 4.3, we retrieve

$$c'_{\mathcal{A}}(x) d = -c_{\mathcal{A}}(x) - D\sigma$$

inducing

$$\begin{aligned}
\nabla \mathcal{L}^A(x, \lambda_+; \mu)^\top d &= -d^\top B d - \mu \|c_{\mathcal{A}}(x)\|^2 - \mu \underbrace{c_{\mathcal{A}}(x)^\top D\sigma}_{\approx 0} \\
&\approx \underbrace{-d^\top B d}_{<0} - \mu \|c_{\mathcal{A}}(x)\|^2 < 0,
\end{aligned}$$

for a sufficiently small Jacobian approximation error D. This proves the assertion. □

Line search. Given a suitable merit function Φ (*here:* Augmented Lagrangian) and descent direction $d \in \mathbb{R}^n$, we still need to specify our line-search strategy for the determination of a step size α at a fixed iterate $(x, \lambda) \in \mathbb{R}^n \times \mathbb{R}^m$, i.e.

$$\alpha \approx \arg\min_{\alpha > 0} \Phi(x + \alpha d), \text{ where } \Phi(\,.\,) := \mathcal{L}^A(\,.\,, \lambda; \mu). \tag{53}$$

Our method of choice to tackle problem (53) is the rather new ShaZ ('**Sh**ift **a**nd **Z**oom') line search which is based on a cubic Hermite interpolation of the objective function $\Phi(\,.\,) : \mathbb{R} \to \mathbb{R}$ using function values as well as slopes at the current search interval borders α_l and α_r. Usually, we choose $\alpha_l = 0$ and $\alpha_r = 1$ as our initial search interval. Prior to the line search, we need to fix an upper bound α_{up} on α ($\alpha_{up} > 0$) and on the number of right shifts. In a typical iteration, the actual reduction

$$ared := \Phi(\alpha_l) - \Phi(\alpha_r)$$

is compared to the predicted reduction

$$pred := \Phi(\alpha_l) - \Phi(\alpha_{min}),$$

where α_{min} denotes the minimizer of the cubic interpolation polynomial. For $\Phi \in \mathcal{C}^1(\mathbb{R})$ and a given 'nondegenerate' search interval $[\alpha_l, \alpha_r]$, i.e. $\alpha_l < \alpha_r$ and $\alpha_{up} > \alpha_r$ as well as parameter $q \in (0, 1)$, we summarize the basic functioning of a typical ShaZ iteration in the following table.

Algorithm 6 (A ShaZ iteration)**.**

```
determine  cubic estimate P(α) by Hermite interpolation using
           points α_l, α_r and slopes Φ'(α_l) and Φ'(α_r)
if         P(α) has a minimum
      set  α_min = arg min P(α)
else
      set  α_min = −∞
determine  ared, pred
if         (Φ(α_r) > Φ(α_l))
      set  α_up ← α_r, i.e. reduce upper bound
if         α_min ∉ (α_l; α_up) and Φ'(α_r) < 0
      SHIFT RIGHT
if         (ared < q · pred) and (α_min ∈ (α_l; α_r))
      ZOOM
if         (ared < q · pred) and (α_min > α_r) and (Φ'(α_r) < 0)
      SHIFT RIGHT
if         (ared ≥ q · pred) and (Φ'(α_r) ≤ Φ'(α_l)) and (α_r < α_up)
           and (Φ'(α_r) < 0)
      SHIFT RIGHT
else
      set  α = α_r, i.e. step size found
      break
end
```

For further details such as the exact determination of the shifts, we refer to [18]. In case we do not face a degenerate case, we should have come up with a step size α fulfilling

$$\begin{cases} ared \geq q \cdot pred & \text{and} \\ \Phi'(\alpha_r) \geq \Phi'(\alpha_l). \end{cases}$$

Properties. Under relatively mild conditions on the objective function, the ShaZ algorithm terminates after a finite number of iterations. Similar conditions ensure that ShaZ represents an effective line search in the sense of Definition A.7, which is important in view of global convergence, cf. Theorem A.8. Moreover, its efficiency, particularly on problems with negative curvature, has been verified. For further information, see [18]. However, it has to be pointed out that the SHaZ line search does not guarantee positive curvature for the computed step multiplier α, i.e. $s^\top w > 0$, with $w = w(\alpha) = g(x + \alpha d, \lambda) - g(x, \lambda)$, is not enforced. As the latter is a necessary condition for positive definiteness of Q, this may conflict with the strategy of maintaining positive definiteness of B.

8.2 Numerical Results

Implementation details. A first implementation of the presented semi-normal method within the L-RAMBO package following Algorithm 5 has currently been tested. Despite the rather unconventional character of this new approach, it has already proven quite robust on a large variety of both constrained and unconstrained nonlinear problems. The following table contains a diverse collection of problems from the CUTEr test set [3] solved by L-SR1-SN embedded in LRAMBO.[9] In all tests, $\|c\| < 10^{-7}$ and $\|g\| < 10^{-7}$ has served as a convergence criterion. The corresponding problems have been solved with an upper

AIRCRFTA	AIRCRFTB	AKIVA	ALLINIT	ALLINITC
ALLINITU	ALSOTAME	ARGLINA	ARGLINB	ARGLINC
ARWHEAD	AVGASA	BARD	BDEXP	BEALE
BIGGS3	BIGGS5	BIGGS6	BOOTH	BOX2
BOX3	BROWNAL	BROWNALE	BROWNBS	CLUSTER
CUBENE	DECONVB	DECONVU	DEMYMALO	DIPIGRI
DUALC5	EXPFIT	EXTRASIM	FCCU	GENHS28
GOFFIN	GOTTFR	GULF	HADAMALS	HAIRY
HATFLDC	HIMMELBA	HIMMELBC	HIMMELBE	HIMMELP3
HIMMELP4	HIMMELP6	HS1	HS10	HS100
HS11	HS113	HS119	HS12	HS13
HS16	HS18	HS2	HS21	HS22
HS23	HS24	HS25	HS26	HS27
HS28	HS29	HS3	HS30	HS31
HS32	HS33	HS34	HS35	HS36
HS37	HS39	HS38	HS41	HS42
HS43	HS44	HS46	HS48	HS49
HS5	HS50	HS51	HS52	HS54
HS55	HS6	HS60	HS65	HS7
HS70	HS76	HS8	HS86	HS88
HS89	HS9	HS92	LOTSCHD	LSNNODOC
MANCINO	MARATOS	MIFFLIN1	MIFFLIN2	MINSURF
ODFITS	PORTFL1	PORTFL2	PORTFL3	PORTFL4
PORTFL6	RES	ROSENBR	SIMPLLPA	SIMPLLPB
TFI3	ZECEVIC2	ZECEVIC3	ZECEVIC4	ZY2

Figure 3: Collection of CUTEr problems (mainly small- and medium-sized) solved with L-SR1-SN included in the L-RAMBO package, for employed options see Algorithm 5 (colored names indicate superlinear convergence after at most ten iterations)

bound on the number of stored limited memory update vectors of $l = 4$. In general, we have observed that choosing l reasonably small, i.e. $l \in \{3, ..., 6\}$, has lead to the best performance in view of stability of Hessian-related objects and moderate γ-adjustment. Additionally one can expect a better quality of the mixed objects if l is kept small. The initialization factor was chosen according to

$$\gamma = \gamma_{est} := \begin{cases} \frac{\|\nabla f\|^2}{|f|} & , \ \|f\| \neq 0 \\ 1.0 & , \text{ else.} \end{cases} \quad \text{or} \quad \gamma = 1.0$$

[9] For Algorithmic Differentiation of function code written in Fortran, as produced by decoding SIF files, the HSL_AD02 tool can be used from within CUTEr; for C++ code, use for instance ADOL-C

and the two spectra for the index comparison were determined using the adjoint eigen solver provided by the C++ template library 'Eigen' [1].

The user is only demanded to provide the code which defines the problem as well as an arbitrary starting point. The results were obtained using the options given in the previous section, i.e. we chose Augmented Lagrangian as our preferred merit function and performed a SHaZ line search to a computed descent direction. To illustrate a typical optimization procedure, consider the following two-dimensional example with two constraints.

Example 8.3 (CUTEr test problem HS14 by Bracken and McCormick [6])**.**

$$\min_{x=(x_1,x_2)} f(x) = (x_1-2)^2 + (x_2-1)^2, \ s.t.$$

$$\begin{cases} x_1 - 2x_2 + 1 & = 0 \\ \frac{1}{4}x_1^2 + x_2^2 - 1 & \leq 0 \end{cases}$$

The following picture displays the iteration sequence, which, starting from an infeasible x_0, converges in just a few efficient steps towards the solution x_* where both constraints are active.

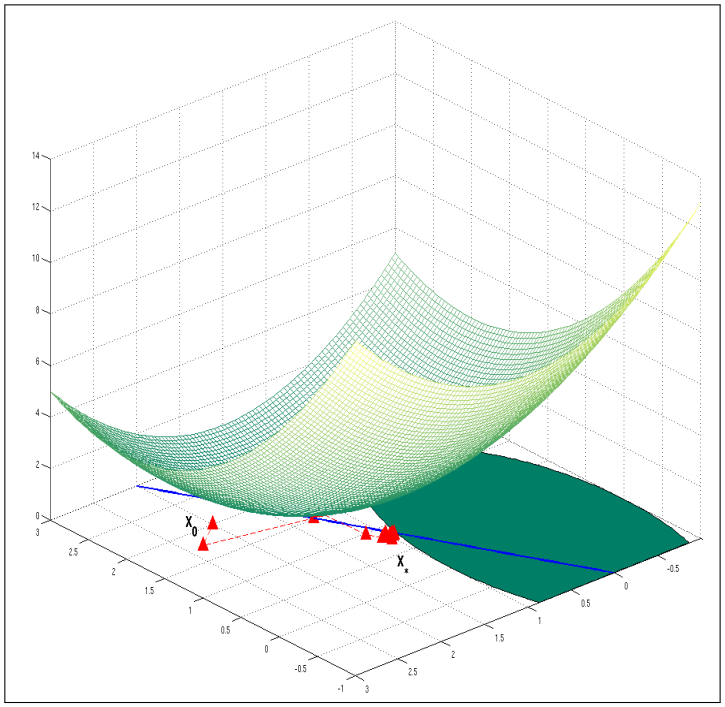

Figure 4: L-SR1-SN method on HS14: iterates, objective function and feasible set

Large-scale optimization. The efficiency of L-SR1-SN as part of LRAMBO has also been verified on a couple of high-dimensional problems with more than 10.000 variables. The following table illustrates the two examples LUKVLI9 ($n = 10000, M = m_{\mathcal{I}} = 6$) and COSINE ($n = 10000, M = 0$) from the CUTEr test set which have been particularly successfully solved by L-SR1-SN. The performance is compared to the IPOPT solver [27] and LRAMBO's ZEDISDEAD (ZID) implementation. The results for L-SR1-SN have been obtained using a notebook with Intel Pentium M processor, 1.4 GHz, 1 GB RAM and a varying amount of stored secant pairs:

NAME	γ	l	iter	c/f-evals	CPU(s): L-SR1-SN (IPOPT / ZID)
LUKVLI9	1.0	3	14	472	2.74 (1.59 / 4.179)
COSINE	γ_{est}	9	53	1672	7.58 (1.25 / 12.86)

Figure 5: L-SR1-SN performance: c/f-evals - total number of objective and constraint function evaluations, CPU(s) - total CPU time in seconds

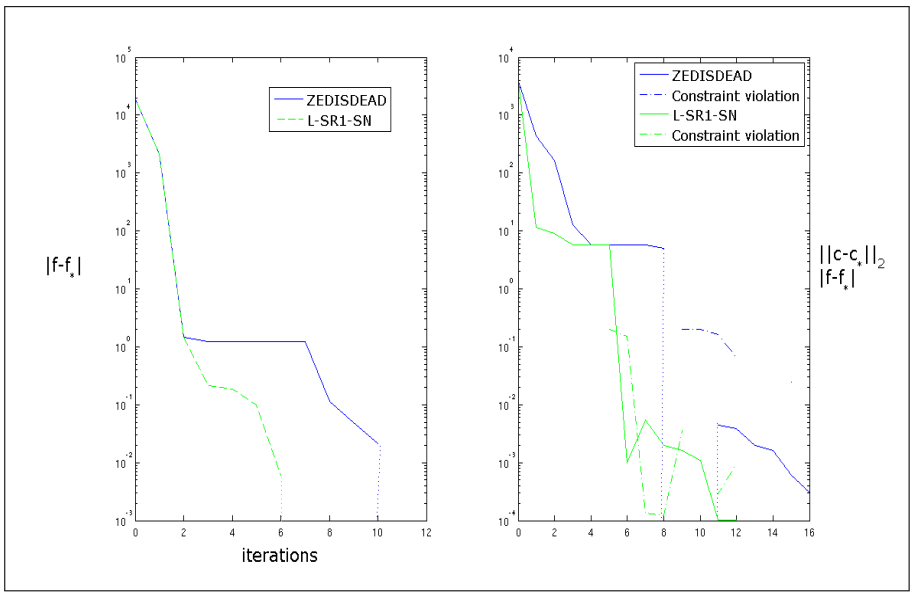

Figure 6: Convergence behavior of L-SR1-SN and ZEDISDEAD on COSINE (left) and LUKVLI9 (right)

Note that the run time of L-SR1-SN may have been slowed down by a lack of parallelization in LRAMBO or other processes occupying the same core. Moreover, we have not yet made use of the vector mode of AD, such that gradients and tangents have been propagated individually, thus causing an overhead of constraint evaluations. In the unconstrained case, the method simplifies to the L-SR1 method: the better performance on COSINE of L-SR1-SN compared to ZEDISDEAD can be attributed to the new γ-adjustment strategy. On LUKVLI9, L-SR1-SN also performs significantly better than ZEDISDEAD. As constraint function evaluation can be assumed comparatively cheap ($M = 6$), the per-iteration-cost of L-SR1-SN is 'only' of order $\mathcal{O}(3n)$ whereas ZEDIS-

DEAD roughly requires the double effort $\mathcal{O}(6n)$. Moreover, the memory requirement is also halved, inducing a drastic reduction in the effort for memory-related operations such as data fetches, pointer moves, etc. . However, L-SR1-SN does not yet reach the robustness of ZEDISDEAD on the entirety of the CUTEr test set.

Issues. Frequent skipping of the Hessian update has been identified as a major source of bad performance[10] of L-SR1-SN. This is mainly due to our skipping rule, i.e. whenever Q loses its positive definiteness, we do not accept the update. This seemingly is a too strong condition for many problems in particular when using SHaZ. A promising remedy may consist of combining the strategy proposed in section 7 with a damping strategy on the SR1-updates. Another issue arising is step acceptance despite a clear violation of the orthogonality condition (36) provoking a corrupted approximation of the SQP step and possible blow-up in $\|\lambda\|$ and $c(x)^\top\lambda$. In case of no appropriate penalty parameter adaptation, this occasionally lead to an unintended increase in $\|c(x)\|$ and $\|g(x,\lambda)\|$ after performing the line search on the Augmented Lagrangian. A promising strategy to circumvent the latter problem, is performing semi-normal updates within the inner loop of Algorithm 5 until the orthogonality condition is fulfilled up to a certain exactness. Hence, an efficient re-organization of the updating procedure in the inner loop is necessary. Moreover, a violation of the LICQ may cause the algorithm to break down at certain points where information from the exact Jacobian is taken to modify L (activation, BFGS-update of L). For these cases, suitable remedies need to be found.

[10] non- or particularly slow convergence

9 Conclusion and Outlook

This work attempts to establish a theoretical basis for the application of semi-normal Jacobian approximations to total Quasi-Newton methods by efficiently extending usage of Algorithmic Differentiation, avoiding computation of full derivative matrices. Since it was demonstrated how explicit storage of large Hessian or Jacobian approximations can be avoided, the method can be implemented at highest efficiency with regard to storage requirement. More precisely, it suffices to allocate memory space with size m^2 plus a small multiple of $l \cdot n$ which means significant improvement in relation to current null space methods. Moreover, the method combines favorable memory requirement with a bilinear operations count per iteration. Accepting a small, fixed number of supplementary AD calculations, computational complexity of order $\mathcal{O}(n \cdot m)$ vanishes completely. Moreover, the γ-adjustment strategy for maintaining positive definiteness of the L-SR1-Hessian has been improved. By smart application of Bennett's algorithm, we have further shown how Cholesky-factorizations as well as the stored data structure can be maintained rotation-free without giving up the optimal operations count. As far as theoretical properties of L-SR1-SN are concerned, it seems to be quite challenging to derive a general global convergence result with regard to the rather intangible nature of the mixed objects.

L-SR1-SN as part of the LRAMBO solver thus represents an attractive alternative to existing total Quasi-Newton solvers for large-scale nonlinear programs. A first implementation within the LRAMBO package has already displayed quite promising results which encourages further research on additional heuristics. For the purpose of improving robustness of the proposed algorithm, more refinements and contributions need to be made. This especially concerns the organization of the inner loop and the combination of different strategies for the maintenance of the positive definiteness of the matrix Q as a key ingredient to ensure the same property for the L-SR1-Hessian.

It is further conceivable to use semi-normal Jacobian approximations in conjunction with a Limited Memory BFGS Hessian in an almost equally efficient way using a similar data structure, cf. Theorem 3.5. This would result in a slightly higher factorization effort as the dimension of the corresponding middle matrices increases from l to $2l$.

A Appendix

A.1 Linear Algebra

The following basic statements from linear algebra as well as their proofs can be found for instance in the classical text book by Golub and Van Loan [11].

Definition A.1 (Inertia, Index)**.**
Let $B \in \mathbb{R}^{n\times n}$ be a symmetric matrix.

(i) The index of B is defined as the number of its negative eigenvalues.

(ii) The triplet

$$(m, z, p) \in \mathbb{N}_0^3,$$

where m, z and p denote the number of negative, zero and positive eigenvalues of B respectively, is called inertia of B.

Theorem A.2 (Sylvester's Law of Inertia)**.**
If $A \in \mathbb{R}^{n\times n}$ is symmetric and $X \in \mathbb{R}^{n\times n}$ is nonsingular, then A and $X^\top AX$ have the same inertia.

Theorem A.3 (Sherman-Morisson-Woodbury formula)**.**
Consider a rank-l modification of $B \in \mathbb{R}^{n\times n}$:

$$B_+ = B + UCV^T, \text{ with } U, V \in \mathbb{R}^{n\times l}, C \in \mathbb{R}^{l\times l}.$$

(i) If $\det(B) \neq 0 \neq \det(C)$ then

$$\det(B_+) = \det(B)\det(C)\det(C^{-1} + V^\top B^{-1}U).$$

(ii) If additionally $\det(C^{-1} + V^\top B^{-1}U) \neq 0$, then

$$B_+^{-1} = B^{-1} - B^{-1}U(C^{-1} + V^\top B^{-1}U)^{-1}V^\top B^{-1}.$$

(iii) If $p = 2$, $B = B^\top \succ 0$, $U = V$ and $C = C^\top \in \mathbb{R}^{2\times 2}$ (symmetric rank-two update) with $det(C) < 0$, it holds

$$B_+ \text{ is positive definite} \iff \det(B_+) > 0.$$

Theorem A.4 (Interlacing Eigenvalue Theorem)**.**
Let $B = B^\top \in \mathbb{R}^{n\times n}$ and $B_+ = B + \delta vv^\top \in \mathbb{R}^{n\times n}, v \in \mathbb{R}^n, \delta \in \mathbb{R}$. Let $\{\lambda_1, ..., \lambda_n\}$ and $\{\lambda_1^+, ..., \lambda_n^+\}$ denote the spectrum of B and B_+, respectively, with

$$\lambda_1 \leq ... \leq \lambda_n \quad \text{and} \quad \lambda_1^+ \leq ... \leq \lambda_n^+.$$

If $\delta \geq 0$, it holds

$$\lambda_1 \leq \lambda_1^+ \leq \lambda_2 \leq ... \leq \lambda_n \leq \lambda_n^+.$$

Otherwise,

$$\lambda_1^+ \leq \lambda_1 \leq \lambda_2^+ \leq ... \leq \lambda_n^+ \leq \lambda_n.$$

A.2 Optimization

Definition A.5 (Superlinear convergence)**.**
A sequence $(x_k)_{k\in\mathbb{N}}$ in $\mathbb{R}^n$ converging to $x_* \in \mathbb{R}^n$, $x_k \neq x_* \ \forall\, k \in \mathbb{N}$ is said to converge (q-)superlinearly, if the quotient

$$q_k := \frac{\|x_{k+1} - x_*\|}{\|x_k - x_*\|}$$

fulfills $\lim\limits_{k\to\infty} q_k = 0$.

Definition A.6 (Gradient-Relatedness)**.**
Define for a given function $f : \mathbb{R}^n \to \mathbb{R}$ the level set $N_0 := \{x \in \mathbb{R}^n : f(x) \leq f(x_0)\}$ to a given start vector $x_0 \in \mathbb{R}^n$. A descent method is called gradient-related if the search directions $d = d(x), x \in N_0$, are selected such that the angle between d and the steepest descent direction does not tend to orthogonality, i.e.

$$\inf_{x\in N_0} \cos\varphi(x) > 0 \quad \textit{with} \quad \cos(\varphi(x)) := -\frac{\nabla f(x)^\top d}{\|\nabla f(x)\|\|d\|}.$$

Definition A.7 (Effective line search)**.**
Let $f \in \mathcal{C}^1(\mathbb{R}^n, \mathbb{R})$. Assume the level set N_0 is compact. A line search strategy $\alpha = \alpha(x; d)$ is called effective, if there exists a constant $\delta > 0$ such that

$$f(x + \alpha d) - f(x) \leq -\delta\|\nabla f(x)\|^2 \cos^2(\varphi(x))$$

holds true for all $x \in N_0$ and corresponding descent directions $d \in \mathbb{R}^n$.

Theorem A.8 (Global convergence)**.**
Let $f \in \mathcal{C}^{1,1}(\mathbb{R}^n, \mathbb{R})$ and suppose the level set N_0 is compact. Then gradient-relatedness of the search directions and effectiveness of the line search imply that all cluster points[11] x_* of the generated sequence $\{x_k\} \subset \mathbb{R}^n$ are stationary, i.e. $\nabla f(x_*) = 0$.

Proof. The definition of an effective line search ensures

$$f(x_{k+1}) - f(x_k) \leq -\delta\|\nabla f(x_k)\|^2 \cos^2(\varphi(x_k)) \quad \forall\, k \in \mathbb{N}.$$

As N_0 is compact, the sequence of negative numbers on the left hand side must converge to zero, otherwise f would not be bounded below. Additionally, $\cos^2(\varphi(x_k))$ is bounded away from zero (gradient-relatedness), which induces $\|\nabla f(x_k)\| \to 0$ and thus $\nabla f(x_k) \to 0$. Any convergent subsequence with limit point x_* must have the same property, such that $\nabla f(x_*) = 0$. □

[11] compactness of N_0 ensures the existence of a cluster point

References

[1] Eigen: A C++ template library for linear algebra: vectors, matrices, and related algorithms. URL http://eigen.tuxfamily.org.

[2] J. M. Bennett. Triangular factors of modified matrices. *Numerische Mathematik*, 7:217–221, 1965.

[3] I. Bongartz, A. R. Conn, N. I. M. Gould, and Ph. L. Toint. CUTE: Constrained and unconstrained testing environment. *ACM Transactions on Mathematical Software*, 21:123–160, 1995.

[4] F. J. Bonnans, J.C. Gilbert, C. Lemaréchal, and C. A. Sagastizábal. *Numerical Optimization*. Springer-Verlag, 2003.

[5] T. Bosse. A derivative-matrix-free NLP solver without explicit null space representation. Master's thesis, Humboldt-Universität zu Berlin, 2009.

[6] J. Bracken and G. P. McCormick. *Selected Applications of Nonlinear Programming*. John Wiley & Sons, 1968.

[7] R.H. Byrd, J. Nocedal, and R.B. Schnabel. Representation of quasi-newton matrices and their use in limited memory methods. *Mathematical Programming*, 63:129–156, 1994.

[8] J.E. Dennis and J.J. Moré. A characterization of superlinear convergence and its application to Quasi-Newton methods. *Mathematics of Computation*, 28:549–560, 1974.

[9] J.E. Dennis and J.J. Moré. Quasi-Newton methods, motivation and theory. *SIAM Review*, 19(1):46–89, 1977.

[10] S. Eichstädt. A total Quasi-Newton method with global convergence for general nonlinear optimization problems. Master's thesis, Humboldt-Universität zu Berlin, 2007.

[11] G.H. Golub and C.F. Van Loan. *Matrix Computations*. The John Hopkins University Press, 3rd edition, 1996.

[12] A. Griewank and A. Walther. *Evaluating Derivatives*. Society for Industrial and Applied Mathematics (SIAM), 2nd edition, 2008.

[13] A. Griewank and A. Walther. On constrained optimization by adjoint based Quasi-Newton methods. *Optimization Methods and Software*, 17(5): 869–889, 2002.

[14] A. Griewank, S. Schlenkrich, and A. Walther. Adjoint Broyden à la GMRES. MATHEON preprint, 2007.

[15] A. Griewank, A. Walter, and M. Korzec. Maintaining factorized KKT systems subject to rank-one updates of Hessians and Jacobians. *Optimization Methods and Software*, 22:279–295, 2007.

[16] T.G. Kolda. *Limited memory matrix methods with applications*. PhD thesis, University of Maryland at College Park, 1997.

[17] M. Korzec. A general low rank update based quadratic programming solver. Master's thesis, 2006.

[18] S. Kreiterling. Effiziente Schrittweitenbestimmung für Optimierungsprobleme mit negativer Krümmung. Master's thesis, Humboldt-Universität zu Berlin, 2007.

[19] D.C. Liu and J. Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45:503–528, 1989.

[20] J. Nocedal and S. Wright. *Numerical Optimization*. Springer Series in Operations Research, 2nd edition, 2006.

[21] V. Schloßhauer. Strukturausnutzung und Speicherplatzbegrenzung für hochdimensionale, nichtlineare Optimierung. Master's thesis, Humboldt-Universität zu Berlin, 2009.

[22] N.Z. Shor. Utilization of the operation of space dilation in the minimization of convex functions. *Kibernetika*, 1:6–12, 1970.

[23] B. Speelpenning. *Compiling fast partial derivatives of functions given by algorithms*. PhD thesis, 1980.

[24] P. Stange. Aufdatierung LU-basierter Faktorisierungen von KKT-Matrizen. Master's thesis, TU Dresden, 2005.

[25] P. Stange, A. Griewank, and M. Bollhöfer. On the efficient update of rectangular LU factorizations subject to low rank modifications. *ETNA*, 26:161–177, 2007.

[26] S. A. Vavasis. Stable numerical algorithms for equilibrium systems. *SIAM J. on Matrix Analysis and Applications*, 15:1108–1131, 1994.

[27] A. Wächter and L. T. Biegler. On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. *Mathematical Programming*, (1):25–57, 2006.

Selbständigkeitserklärung

Ich erkläre, dass ich die vorliegende Arbeit selbständig und nur unter Verwendung der angegebenen Literatur und Hilfsmittel angefertigt habe.

Berlin, den 25. August 2010

Einverständniserklärung

Hiermit erkläre ich mich einverstanden, dass ein Exemplar meiner Diplomarbeit in der Bibliothek des Instituts für Mathematik verbleibt.

Berlin, den 25. August 2010