Karlsruher Institut für Technologie

Faculty of Economics
Institute for Operations Research (IOR)
Continuous Optimization
Prof. Dr. Oliver Stein

Seminar paper

Algorithmic solution of the Trust-Region auxiliary problem

von

Paul-Niklas Kandora
1977420
Industrial Engineering Bachelor

Date of delivery
06/29/2018

Supervision: Robert Mohr, Ph.D

Declaration

I certify that I have prepared this document on my own initiative, that I have fully acknowledged all aids and sources used, that I have marked the passages taken over verbally or in terms of content as such, and that I have observed the KIT Statutes on Safeguarding Good Scientific Practice.

Date *Name*

Inhaltsverzeichnis

1 Introduction **5**

2 Trust-Region methods **6**

2.1 Trust-Region-methods and auxiliary problems 6

2.2 Optimality condition for trust-region auxiliary problems . . . 7

3 Krylov subspace method **8**

3.1 CG method 8

3.2 Lanczos method 11

4 Approximate solution to the trust region auxiliary problem **13**

4.1 Steihaug-Toint method 13

4.2 Generalized Lanczos-Trust-Region method 17

5 Numerical experiment **22**

5.1 Test data 23

5.2 Results 23

6 Conclusion **27**

1 Introduction

Trust region methods differ from classical search direction methods with respect to step size control, since the search direction in a new iteration is determined before the step size. The calculation of the search direction in general trust-region algorithms is based on the classical search direction methods (e.g., Newton method), where the direction of a new iteration is calculated as the minimum point of a quadratic function. Due to the fact that the quadratic model is bounded by a step-size constraint, computing the search direction results in a constrained optimization problem called the trust-region auxiliary problem. The approximate solution of the trust-region auxiliary problem is a central part of this seminar paper.
The first work that formulated approaches to the trust-region methods as they are known in the literature today (cf. [1, p.116]) was provided by Powell's 1970 publication. In 1983, based on a publication by Toint (1981), the Steihaug-Toint method was formulated, the basis of which is the CG method of Hestens and Stiefel (1952). The Generalized Lanczos Trust Region Method, which was introduced in 1997 by Gould, Luicidi, Roma, and Toint [3], represents a modification of the Steihaug-Toint method.
In this term paper, we will begin by formulating the trust-region auxiliary problem and then clarify how optimality is characterized for the trust-region auxiliary problem. Then, basic ideas and concepts of the CG method and the Lanczos method are formulated to motivate the methods for solving the trust-region auxiliary problem and to understand their approach. Then, the Steihaug-Toint method is introduced based on the CG method. After that, ideas of the Lanczos method are combined with the Steihaug-Toint method, resulting in the Generalized Lanczos Trust Region Method. Finally, this thesis will deal with the question whether the theoretical concepts of the methods, which have been worked out in the chapters before, can be mapped to a practical application.

2 Trust-Region methods

In this chapter, the trust-region auxiliary problem is introduced. Then, an optimality criterion for the trust region auxiliary problem is derived for the convex and non-convex case.

2.1 Trust-Region-methods and auxiliary problems

The trust-region methods are not classical search-direction methods for minimizing a two-state differentiable mapping $f : \mathbb{R}^n \to \mathbb{R}$. In one iteration of the General trust-region method (cf. [1, p.116]), an approximation of the actual function f is minimized on a trust region called *Trust-region*. This optimization problem is hereafter referred to as the *Trust-Region auxiliary problem* and is defined as follows:

$$\min_{s\in\mathbb{R}^n} q(s) = \langle g, s\rangle + \frac{1}{2}\langle s, Hs\rangle \quad \text{s.t} \quad \|s\|_2 \leq \triangle, \tag{2.1}$$

with $g = \nabla f(x^k)$ and $\{x^k\}_{k\in\mathbb{N}}$ as an iterated generated by the general Trust-Region method and $H \in \mathbb{R}^{n\times n}$ is a bounded approximation of the Hessian of f or the actual Hessian of f. $q(s)$ in (2.1) is a second order Taylor approximation of f at the point x^k as a function of $x^k + s$. The constant term $f(x^k)$, which arises from the Taylor approximation, is suppressed since the optimal point of (2.1) is not affected by $f(x^k)$. In (2.1), $q(s)$ is used instead of $q(x^k + s)$. If the global minimum point of (2.1) (s^M) has been generated, or an approximation to s^M, then a new iteration of the General Trust-Region method (x^{k+1}) can be computed as $x^{k+1} = x^k + s^M$. The set $\mathcal{B}_k = \{x \in \mathbb{R}^n \mid \|s\|_2 \leq \triangle\}$, will be called *trust-region* in the following. $\triangle > 0$ will be called *Trust-Region radius*. It is possible to choose a different norm to better scale the variables (cf. [5, p.94]). For simplicity, the ℓ_2 norm is considered in this work.

2.2 Optimality condition for trust-region auxiliary problems

Satz 2.1 *(cf. [1, Corollary 7.2.2]) Each global minimum point s^M of* (2.1) *satisfies the equation:*

$$H(\lambda^M)s^M = -g, \tag{2.2}$$

where $H(\lambda^M) \equiv H + \lambda^M I \succeq 0$, $\lambda^M \geq 0$, and $\lambda^M(\|s^M\|_2 - \triangle) = 0$, where $I \in \mathbb{R}^{n\times n}$ is the unit matrix. If $H(\lambda^M) \succ 0$ holds, then s^M is single global minimal point of (2.1).

It is a first order optimality criterion for (**??**). Furthermore, λ^M is a Lagrange multiplier and $\lambda^M(\|s^M\|_2 - \triangle) = 0$ is a complementarity condition (cf. [1, p.174]). The following theorem shows that provided there exists a solution to (2.1), $q(s)$ is a strictly convex function, and the global minimal point s^M is found within the trust region, then no other solutions exist on the trust region boundary.

Satz 2.2 *(cf. [4, Theorem 3.2.]) Problem* (2.1) *has no solution on the trust-region boundary exactly when $H \succ 0$ and $\|s^M\|_2 < \triangle$, with $s^M = H^{-1}g$.*

In the case that $H \succeq 0$ and is singular, there exist other solutions besides s^M as the unrestricted minimal point of (2.1), which are on the trust-region boundary (cf. [1, p.173]). Now the question arises to what extent optimality from Theorem 2.1 is also characterized for nonconvex functions $q(s)$ from (2.1). From Theorem 2.1 it can be seen that for a nonconvex function $q(s)$ from problem (2.1) $\lambda^M \neq 0$ must hold if a solution s^M exists. In this case, s^M must lie on the trust-region boundary. Such a λ^M can be calculated as a solution of equation[1]:

$$\|s(\lambda)\|_2 = \triangle \quad \text{with } s(\lambda) = H(\lambda)^+ g. \tag{2.3}$$

The solution of (2.3) can be determined with the help of Newton's method, which is discussed again in more detail in chapter 4.2. For the solution of the equation from (2.3) it should be mentioned that the so-called *simple case* (cf. [4, p.19]) can occur. In this scenario, the conditions are satisfied that the resulting matrix $H(\lambda^M)$ is nonsingular and $s(\lambda^M)$ can be computed directly from (2.3). On the other hand, the *severe case* may occur where

[1] $H(\lambda)^+$ is the Moore-Penrose inverse, because it is not known if $H(\lambda)$ is actually singular.

the conditions are satisfied that $H(\lambda^M)$ is singular (cf. [4, p.20]). In this case $s(\lambda^M)$ cannot be determined directly from (2.3). For the use case from chapter 4.2 it becomes obvious that the hard case cannot occur. The computation of a solution of (2.3) in the hard case is treated in more detail in [1, p.181].

3 Krylov subspace method

We now introduce the basics for the methods that will be used to solve the trust-region auxiliary problem inexactly in chapter 4. A Krylov space is defined as follows:

$$\mathcal{K}(H, x, k) = span(x, Hx, H^2x \ldots, H^kx), \tag{3.1}$$

with $H \in \mathbb{R}^{n\times n}$, $x \in \mathbb{R}^n$ and $k \in \mathbb{N}$. CG methods iteratively generate bases of vector spaces of the type (3.1) and minimize in each iteration a quadratic function on Krylov spaces that increase in dimension per iteration. The Lanczos method iteratively generates bases for Krylov spaces, where the Krylov spaces are identical to the Krylov spaces generated by the CG method. However, the bases of the Krylov spaces generated by the two methods differ.

3.1 CG method

Basically, the question is whether CG methods solve problems of the type (2.1). If one disregards the restriction in (2.1) and assumes $H \succ 0$, the CG method solves the problem:

$$\min_{s\in\mathbb{R}^n} q(s) = \langle g, s\rangle + \frac{1}{2}\langle s, Hs\rangle. \tag{3.2}$$

From theorem 2.2 it is evident that the CG method solves problem (2.1) provided $H \succ 0$, s^M lies within the trust region (restriction from (2.1) would be redundant) and a solution s^M exists. In this case, the solution from (3.2) would be equivalent to the solution from (2.1). This will be taken up again in 4.1. The following lemma illustrates a basic idea of the CG procedure:

Lemma 3.1 *(cf. [1, Lemma 5.1.2]) Suppose $p^j \in \mathbb{R}^n$ with $0 < j \leq n-1$ is H-conjugate (cf. [2, p.95]) to its predecessors from $\{p^0, \ldots, p^k\}$, $0 \leq k \leq n-1$ and $p^0 \in \mathbb{R}^n$ is a starting vector. Furthermore, $P_k = (p^0, \ldots, p^k)$, $P_k \in \mathbb{R}^{n\times(k+1)}$. Moreover, let $P_k^T H P_k$ be positive definite. Then s^{k+1} is generated as the minimal point of $q(s)$ from (3.2) on the manifold (cf. [1, p.76]) $s^0+P_k p$, $p \in \mathbb{R}^{k+1}$, by the following iterations:*

$$s^{k+1} = s^k + \sigma_k p^k, \tag{3.3}$$

$$\sigma_k = -\frac{\langle p^k, g^k\rangle}{\langle p^k, Hp^k\rangle} \tag{3.4}$$

und

$$g^{k+1} = g^k + \sigma_k H p^k, \tag{3.5}$$

with $g^k = Hs^k + g$, $s^0 \in \mathbb{R}^n$ is an arbitrary starting vector and $s^k \in \mathbb{R}^n$ is the minimal point of q from (3.2) on the manifold $s^0 + P_{k-1}p$ after applying Lemma 3.1, $p \in \mathbb{R}^k$, provided $k > 0$.

Now suppose that the columns of the matrix P_k form a basis for the subspace $\mathcal{P}_k \subseteq \mathbb{R}^n$, $0 \leq k \leq n-1$ form. From Lemma 3.1 it is seen that if one can find a way to generate H-conjugate bases (with $H \succ 0$) for subspaces $\mathcal{P}_0 \subset \mathcal{P}_1 \subset \cdots \subset \mathcal{P}_{n-1} = \mathbb{R}^n$ and whose dimension increases by 1 per iteration, then at each step k where (3.3), (3.4), and (3.5) are executed, the minimum point of $q(s^0 + P_k p)$ is obtained as an approximation to the global minimum point s^N of (3.2). For $\mathcal{P}_{n-1} = \mathbb{R}^n$, the global minimal point s^N of (3.2) would be found after at most n iterations. The implementation with which H-conjugate bases $\{p^0, \ldots, p^k\}$ are computed for subspaces $\mathcal{P}_k$, $0 \leq k \leq n-1$, which are to iteratively increase in dimension by 1, is provided by the following algorithm.

Algorithmus 3.2 *(vgl. [1, Algorithm 5.1.1]) Given a starting vector $r^0 \in \mathbb{R}^n$. Set $p^0 = -r^0$ and for $k = 0, 1, \ldots,$ perform the following iterations:*

$$\alpha_k = \|r^k\|_2^2 / \langle p^k, Hp^k\rangle,$$

$$r^{k+1} = r^k + \alpha_k H p^k,$$

$$\beta_k = \|r^{k+1}\|_2^2 / \|r^k\|_2^2,$$

$$p^{k+1} = -r^{k+1} + \beta_k p^k.$$

Combining algorithm 3.2 for $r^i = g^i$, $i \geq 0$ with lemma 3.1, the solution s^N of (3.2) can be determined iteratively. The result is the *Method of Conjugate Gradients (CG-method)*:

Algorithmus 3.3 *(vgl. [1, Algorithm 5.1.2]) Given an* $s^0 \in \mathbb{R}^n$*, set* $g^0 = Hs^0 + g$ *and* $p^0 = -g^0$*. For* $k = 0, 1, \ldots$ *to* $\|\nabla q(s^{k+1})\|_2 < \epsilon$*,* $\epsilon > 0$*, perform the following iterations:*

$$\alpha_k = \|g^k\|_2^2 / \langle p^k, Hp^k \rangle,$$
$$s^{k+1} = s^k + \alpha_k p^k,$$
$$g^{k+1} = g^k + \alpha_k Hp^k,$$
$$\beta_k = \|g^{k+1}\|_2^2 / \|g^k\|_2^2,$$
$$p^{k+1} = -g^{k+1} + \beta_k p^k.$$

An advantage of algorithm 3.3 is that the main computational effort consists of matrix-vector products. The choice of the termination criterion $\|\nabla q(s^{k+1})\|_2 < \epsilon$, $\epsilon > 0$ is based on [2, p.101].

The following lemma shows that the conjugate gradients from Algorithm 3.3 span a Krylov space:

Lemma 3.4 *Assuming algorithm 3.3 is in iteration* $0 \leq k \leq n-1$ *and does not terminate in iteration* k*, then holds:*

$$span(p^0, \ldots, p^k) = span(g^0, \ldots, g^k) = \mathcal{K}(H, g^0, k). \tag{3.6}$$

The vectors $\{p^0, \ldots, p^k\}$ form an H-conjugate basis for $\mathcal{K}(H, g^0, k)$, which is evident from Lemma 3.1. The vectors $\{g^0, \ldots, g^k\}$ form an orthogonal basis for $\mathcal{K}(H, g^0, k)$ (cf. [6]).

Dropping the condition $H \succ 0$ in problem (3.2), Algorithm 3.3 may relatively quickly encounter the case where $\langle p^k, Hp^k \rangle = 0$ (which is especially possible if H is not positive definite) or $\langle p^k, Hp^k \rangle$ is vanishingly small. Here, algorithm 3.3 would terminate in one iteration k. This factual situation is also a major problem for solving (2.1) in the course of this work. Nevertheless, we try to solve this problem as elegnatly as possible in chapter 4.

It should also be mentioned that it is possible to modify Algorithm 3.3 with a conditioning matrix M with $M \in \mathbb{R}^{n\times n}$, $M \succ 0$, so that faster convergences are possible than with Algorithm 3.3 without preconditioning. Lastly, it should be mentioned that the preconditioned version of the CG method generates a different Krylov space than algorithm 3.3 (vgl. [1, S.86-S.89]).

3.2 Lanczos method

In the following, we show a procedure that iteratively generates bases for Krylov spaces $\mathcal{K}(H, g^0, 0) \subseteq \mathcal{K}(H, g^0, 1) \subseteq \cdots \subseteq \mathcal{K}(H, g^0, n-1) = \mathbb{R}^n$, which are different from the bases generated by algorithm 3.3. The *Lanczos method* generates an orthonormal basis $\{q^0, \ldots, q^k\}$ for $\mathcal{K}(H, q^0, k)$. The derivation of Lanzcos method is based on the Gram-Schmidt orthogonalization process and a few geometric considerations (cf. [1, pp.93-S.95]):

Algorithmus 3.5 *(vgl. [1, Algorithm 5.2.1]) Given a starting vector $r^0 \in \mathbb{R}^n$, set $y^0 = r^0$, $q^{-1} = 0$, and for $k = 0, 1, \ldots,$ perform the following iterations:*

$$\begin{aligned} \gamma_k &= \|y^k\|_2, \\ q^k &= y^k/\gamma_k, \\ \delta_k &= \langle q^k, Hq^k\rangle, \\ y^{k+1} = \gamma_{k+1}q^{k+1} &= Hq^k - \delta_k q^k - \gamma_k q^{k-1}. \end{aligned} \tag{3.7}$$

An advantage of the Lanczos method, over the Gram-Schmidt method, evident from (3.7), is that to compute the so-called *Lanczos vectors*, $\{q^0, \ldots, q^k\}$, only the vectors q^{k-1}, q^k, and q^{k+1} need to be stored. The following theorem emphasizes the relationship between the Lanczos method and the CG method.

Lemma 3.6 *Assume algorithm 3.5 is in iteration $0 \leq k \leq n-1$ and does not terminate in iteration k. In addition, as starting parameters for algorithm 3.5, $\mathcal{K}(H, g^0, 0) = span(q^0)$, $q^0 = g^0/\|g^0\|_2$ are chosen with $g^0 = Hs^0 + g$. Then the following relation holds:*

$$\mathcal{K}(H, q^0, k) = \mathcal{K}(H, g^0, k) = span(q^0, \ldots, q^k).$$

The Lanczos method terminates in an iteration k if an invariant subspace $\mathcal{K}(H, q^0, k)$ is present under the matrix H.

It should be mentioned that based on vectors $\{q^0, \ldots, q^k\}$ from algorithm 3.5, H-conjugate directions can be generated. To solve (3.2) with algorithm 3.3, instead of $\{p^0, \ldots, p^k\}$, another H-conjugate basis, $\{x^0, \ldots, x^k\}$ with $x^j \in \mathbb{R}^n$, $0 \leq j \leq k$, could be used from $\mathcal{K}(H, g^0, k)$. $\{x^0, \ldots, x^k\}$ would be generated based on the vectors $\{q^0, \ldots, q^k\}$, and the iteration sequences from algorithm 3.3 would be defined differently from the way $\{p^0, \ldots, p^k\}$ is computed. Since the essential result (the matrix T_k[2]) from Algorithm 3.5 is relevant to formulate the procedure from Chapter 4.2, the reader is referred to [1, pp.97-S.98] for the computation of the vectors $\{x^0, \ldots, x^k\}$.

Hq^i can be formulated compactly $\forall i = 0, \ldots, k$ as

$$HQ_k = (Hq^0, \ldots, Hq^k), \tag{3.8}$$

with $Q_k = (q^0, \ldots, q^k)$, $Q_k \in \mathbb{R}^{n \times (k+1)}$ and $H \in \mathbb{R}^{n \times n}$.

The formula $-\delta_i q^i - \gamma_i q^{i-1}$ can be expressed $\forall i = 0, \ldots, k$ by the matrix:

$$T_k = \begin{pmatrix} \delta_0 & \gamma_1 & & & \\ \gamma_1 & \delta_1 & \cdot & & \\ & \cdot & \cdot & \cdot & \\ & & \cdot & \delta_{k-1} & \gamma_k \\ & & & \gamma_k & \delta_k \end{pmatrix}, \tag{3.9}$$

$T_k \in \mathbb{R}^{(k+1)\times(k+1)}$ and Q_k, as

$$Q_k T_k = (\delta_0 q^0 + \gamma_1 q^1, \gamma_1 q^0 + \delta_1 q^1 + \gamma_2 q^2, \ldots, \delta_{k-1} q^{k-1} + \gamma_k q^k + \gamma_{k-1} q^{k-2}, \gamma_k q^{k-1} + \delta_k q^k). \tag{3.10}$$

In addition, the matrix expression:

$$\gamma_{k+1} q^{k+1} (e^{k+1})^T = (0_n, 0_n, \ldots, \gamma_{k+1} q^{k+1}), \tag{3.11}$$

with $(e^{k+1})^T \in \mathbb{R}^{1\times(k+1)}$ as the transposed unit vector with which the $(k+1)$ component is a 1 and 0_n the Zero vector from $\mathbb{R}^n$, exist. From algorithm 3.5 it is seen that using relations (3.8), (3.10), (3.11) and $y^{k+1} = \gamma_{k+1} q^{k+1}$ the following matrix equation:

$$HQ_k - Q_k T_k = \gamma_{k+1} q^{k+1} (e^{k+1})^T \tag{3.12}$$

is feasible. Additionally, since the vectors $\{q^0, \ldots, q^k\}$ generated by algorithm 3.5 are pairwise orthonormal, the relations:

$$Q_k^T Q_k = I_{k+1} \quad \text{und} \quad Q_k^T q^{k+1} = 0 \tag{3.13}$$

[2]The matrix T_k is about to be introduced.

can be derived, with $I_{k+1} \in \mathbb{R}^{(k+1)\times(k+1)}$ as the unit matrix. Multiplying Q_k^T from the left at (3.12), as well as exploiting (3.13), we obtain the essential result of Lanczos' method:

$$Q_k^T H Q_k = T_k. \tag{3.14}$$

In chapter 4.2 a subproblem of the trust-region auxiliary problem is introduced using the matrix T_k, which is easier to solve using Newton's method than the trust-region auxiliary problem from (2.1). Here, the tridiagonal structure of the matrix T_k is predominantly exploited, which is evident from (3.9).

Analogous to the preconditioned CG method mentioned at the end of chapter 3.1, a preconditioned version of algorithm 3.5 can be considered. The Krylov spaces generated by the preconditioned CG method coincide with the Krylov spaces computed by the preconditioned Lanczos method (only the bases of the Krylov spaces differ) (cf. [1, pp.103-S.105]).

4 Approximate solution to the trust region auxiliary problem

In this chapter, two methods are introduced to solve the trust-region auxiliary problem from (2.1) inexactly. Algorithm 3.3 is modified so that an acceptable approximation to s^M can be generated even in the case of non-positive curvature or when the trust-region bound is reached. Subsequently, a subproblem from the trust-region auxiliary problem is introduced. By solving this subproblem, an approximation can be generated that is closer to the solution than the computed approximation of the modified version of the CG method.

4.1 Steihaug-Toint method

In the following, all considerations are made starting from the CG method. In this framework, we consider possible scenarios that may occur when algorithm 3.3 is applied to the trust-region auxiliary problem. No preconditions are imposed on the matrix H. Furthermore, the restriction from (2.1) is considered. Based on these considerations, the *Steihaug-Toint procedure* is formulated.

It is already known from Theorem 2.2 and Chapter 3.1 that a solution s^M of (2.1) can be generated after at most n iterations using the CG method,

provided $H \succ 0$, $\|s^M\|_2 < \triangle$, and a solution s^M exists for (2.1). In this case, the unrestricted minimum point s^N of (3.2) coincides with the solution of problem (2.1). For this particular scenario, it would make sense for the Steihaug-Toint method to keep generating iterations of algorithm 3.3 until the trust-region limit is reached or $|\langle p^k, Hp^k\rangle|$ becomes sufficiently small in one iteration k. The case where CG iterates are generated outside the trust region will be discussed later in case (1.).

In the case $H \succeq 0$ and singular, iterates of algorithm 3.3 can be generated until in one iteration $k > 0$, $\langle p^k, Hp^k\rangle = 0$ occurs, $|\langle p^k, Hp^k\rangle| < \epsilon$ with a sufficiently small $\epsilon > 0$ or the trust region boundary is reached. In this scenario, despite a convex model in (2.1), the algorithm 3.3 would terminate. The solution approach to this scenario is addressed in the course of case (2.). Now we consider the more problematic cases already mentioned.

(1.) Iterates of algorithm 3.3 are generated outside the trust region.
(2.) In algorithm 3.3, $\langle p^j, Hp^j\rangle \leq 0, 0 \leq j \leq n-1$ enters.

In case (1.) it becomes clear that the unrestricted optimal point of (2.1) must lie outside the trust region. One might now assume that subsequent CG iterates are again generated inside the trust region. However, the following theorem shows that iterates generated by algorithm 3.3 outside the trust region no longer enter the trust region.

Satz 4.1 *(vgl. [1, Theorem 7.5.1]) Suppose algorithm 3.3 is applied to minimize $q(s)$. Let $s^0 = 0$ and $\langle p^i, Hp^i\rangle > 0$ for $0 \leq i \leq k$. Then the iterates s^j satisfy the following inequality:*

$$\|s^j\|_2 < \|s^{j+1}\|_2 \tag{4.1}$$

für $0 \leq j \leq k-1$.

In this case, to compute an approximation, consider the CG procedure in an iteration k and a computed iterate s^k which is inside the trust region but generated along the direction p^k with the associated step size α_k outside the trust region. Here, $\sigma > 0$ (for $\sigma \leq 0$, descent would not be guaranteed and the following chain of inequalities (4.3) would not have general validity) is calculated along the half-line starting at s^k in the direction p^k. Computing such a $\sigma > 0$ is equivalent to solving the equation:

$$\|s^k + \sigma p^k\|_2 = \triangle\ . \tag{4.2}$$

Due to the fact that algorithm 3.3 is a descent method, $s^k + \sigma p^k$ generates an approximation for which holds:

$$q(s^k + \alpha_k p^k) < q(s^k + \sigma p^k) < q(s^k). \tag{4.3}$$

The solution of the equation (4.2) is implemented in the Steihaug-Toint method with an if-condition and is oriented to [1, p.206].

In case (2.), if $\langle p^j, Hp^j \rangle \leq 0$ occurs with $0 < j \leq n-1$, $q(s)$ from (2.1) is either nonconvex ($\langle p^j, Hp^j \rangle < 0$) and/or H is singular. Then the one-dimensional constraint $q(s^j + \sigma p^j)$ with $\sigma > 0$ is unbounded downward. In particular, this is true for the case $\langle p^j, Hp^j \rangle = 0$, since with $p^j \in \{H\}$ ($p^j \neq 0$), the model $q(s)$ is unbounded along the direction λp^j, $\forall \lambda \in \mathbb{R}$. The solution approach in this case is to minimize $q(s)$ starting from s^j along the direction p^j until the trust-region limit is reached. The computation of such an approximate iterate is possible with the solution of (4.2) for $s^k = s^j$ and $p^k = p^j$.

Now the question arises which starting vector $p^0 \in \mathbb{R}^n$ and starting point $s^0 \in \mathbb{R}^n$ should be chosen. The choice of $g^0 = g = \nabla f(x^k)$, $p^0 = -g^0$ and $s^0 = 0$ provides global convergence when case (1.) and case (2.) occur in one iteration $j = 0$. Here, using the solution of (4.2), the search direction (including step size) of the Cauchy point (cf. [1, p.124])

$$s_k^C = s^1 = -\sigma \nabla f(x^k) \tag{4.4}$$

is calculated. Furthermore, it can be seen that Algorithm 3.3 for the above choice also in one iteration $k = 0$ is the search direction (including step size (cf. [1, p.128]) of the Cauchy point

$$s_k^C = s^1 = -\alpha_0 \nabla f(x^k). \tag{4.5}$$

berechnet.

Based on these considerations, the Steihaug-Toint method is formulated.

Algorithmus 4.2 *Let* $s^0 = 0$, $g^0 = g$ *and* $p^0 = -g^0$. *For* $k = 0, 1, \ldots$ *perform the following steps until* $k > k_{max}$, $k_{max} \geq 0$*:*
Set $\kappa_k = \langle p^k, Hp^k \rangle$.
If $\kappa_k \leq 0$,

> *calculate* σ_k *as the positive root of* $\|s^k + \sigma p^k\|_2 = \triangle$, *set* $s^{k+1} = s^k + \sigma_k p^k$, *and stop.*

End if
Set $\alpha_k = \|g^k\|_2^2 / \kappa_k$.
If $\|s^k + \alpha_k p^k\|_2 \geq \triangle$,

> *calculate* σ_k *as the positive root of* $\|s^k + \sigma p^k\|_2 = \triangle$, *set* $s^{k+1} = s^k + \sigma_k p^k$, *and stop.*

End if
Setze

$$s^{k+1} = s^k + \alpha_k p^k,$$
$$g^{k+1} = g^k + \alpha_k H p^k,$$
$$\beta_k = \|g^{k+1}\|_2^2 / \|g^k\|_2^2,$$
$$p^{k+1} = -g^{k+1} + \beta_k p^k.$$

From (4.4) and (4.5), it can be seen that Algorithm 4.2 computes the search direction (including step size) s_k^C for $k = 0$. Based on the inequalities (4.3) and (**??**) and the fact that algorithm 3.3 generates descent directions, the following lemma is valid. The descent of $q(s)$ at the point s_k^C forms an upper bound on the descent generated by iterates of Algorithm 4.2 in $q(s)$.

Lemma 4.3 *Each iterate* $\{s^i\}_{i \geq 0}$ *generated by algorithm 4.2 and applied to problem* (2.1) *satisfies the inequality*

$$q(s^i) \leq q(s_k^C). \tag{4.6}$$

s_k^C *is chosen as* (4.4) *in case (1.) or case (2.) for* $j = 0$. *Otherwise,* s_k^C *is chosen to be* (4.5).

Based on Lemma 4.3, global convergence of the general trust-region method can be guaranteed against the first-order critical point, provided f is not downward unbounded (cf. [9]).

The question now arises how to evaluate an approximation s^{ST} of algorithm 4.2 compared to the actual optimal point s^M. This question is answered in the following theorem. (cf. [10, Theorem 3.11.]) Suppose algorithm 4.2 is applied to compute s^{ST} as an approximation to s^M as a solution of (2.1). In addition, H is positive definite. Then the following inequality holds:

$$q(s^{ST}) \leq \frac{1}{2} q(s^M). \tag{4.7}$$

If $q(s)$ is a convex function, then algorithm 4.2 with an approximation s^{ST} yields a point that, for $q(s)$, yields at least half the descent that s^M generates as an exact solution of (2.1). It is worth mentioning that Theorem 4.1 has no validity in the nonconvex case, since for example $g^0 = 0$ and H as an indefinite matrix, Algorithm 4.2 terminates at $s^{ST} = 0$ without having generated a descent for $q(s)$ (cf. [1, p.218]).

Algorithm 4.2 takes over the positive aspect of the CG method, namely that the main computational effort consists of matrix-vector products and is applicable under certain conditions also for restricted optimization problems. Moreover, the algorithm can relatively quickly generate a solution that satisfies the requirements of Theorem 4.1. However, this can be interpreted as a disadvantage in some cases. If case (1.) and case (2.) occur relatively early in algorithm 4.2, at most a solution can be generated that is marginally better than the search direction s_k^C, even in the case of a convex function $q(s)$.

4.2 Generalized Lanczos-Trust-Region method

As mentioned at the end of chapter 4.1, algorithm 4.2 generates an approximation that may not always be satisfactory if the algorithm terminates early. The Generalized Lanczos Trust Region method exploits the fact that Algorithm 3.3 is theoretically satisfactory despite the scenario $\langle p^i, Hp^i \rangle < 0$, $i \geq 0$ and case (2.), can generate iterates. For the case $\langle p^i, Hp^i \rangle = 0$ or $|\langle p^i, Hp^i \rangle| < \epsilon$ (with a sufficiently small $\epsilon > 0$), even the Generalized Lanczos Trust Region Method will not be able to generate a solution that is significantly better than an approximation of the Steihaug-Toint method.

It is known from Chapter 3.1 that an iterate of Algorithm 4.2 in one iteration

k, s^k, is the minimum point of $q(s)$ on the Krylov space $\mathcal{K}(H, g^0, k)$ unless the trust-region boundary is reached or nonpositive curvature has not emerged. Therefore, s^k is the solution of the optimization problem:

$$\min_{s \in \mathcal{K}(H,g^0,k)} q(s) = \langle g, s \rangle + \frac{1}{2}\langle s, Hs \rangle \quad \text{s.t} \quad \|s\|_2 \leq \Delta \ . \tag{4.8}$$

From Lemma 3.6 it is known that the columns of the matrix Q_k are a basis for $\mathcal{K}(H, g^0, k)$. Therefore, each point $s \in \mathcal{K}(H, g^0, k)$ can be represented as an element of the set $\mathcal{S} = \{s \in \mathbb{R}^n \mid s = Q_k h\}$, where $h \in \mathbb{R}^{k+1}$. From chapter 3.2 the relations are:

$$Q_k^T H Q_k = T_k \tag{4.9}$$

$$Q_k^T Q_k = I_{k+1} \tag{4.10}$$

und

$$Q_k^T g = \gamma_0 e_1 \tag{4.11}$$

(vgl. [3, S.6]) are known. Using the relations from (4.9), (4.10), (4.11), as well as the representation of s from $\mathcal{S}$, problem (4.8) can be mapped to the subspace $\mathcal{K}(H, g^0, k)$:

$$\begin{aligned}
&\min_{h \in \mathbb{R}^{k+1}} q(h) = \langle g, Q_k h \rangle + \frac{1}{2}\langle Q_k h, HQ_k h \rangle \quad \text{s.t} \quad \|Q_k h\|_2 \leq \Delta \\
&\iff \\
&\min_{h \in \mathbb{R}^{k+1}} q(h) = h^T Q_k^T g + \frac{1}{2} h^T Q_k^T H Q_k h \quad \text{s.t} \quad \sqrt{h^T Q_k^T Q_k h} \leq \Delta \\
&\iff \\
&\min_{h \in \mathbb{R}^{k+1}} \langle h, \gamma_0 e_1 \rangle + \frac{1}{2}\langle h, T_k h \rangle \quad \text{s.t} \quad \|h\|_2 \leq \Delta \ .
\end{aligned} \tag{4.12}$$

Comparing problem (4.12) with problem (2.1), it can be seen that the matrix T_k, H represents on the subspace $\mathcal{K}(H, g^0, k)$. Instead of solving an optimization problem (2.1), which is defined on the $\mathbb{R}^n$, there is now the possibility to solve a lower dimensional optimization problem (4.12), which is defined on the $\mathbb{R}^{k+1}$. The Generalized Lanczos Trust Region method iteratively solves problems of type (4.12) when the trust region boundary is reached or non-positive curvature appears, but performs CG iterations until then.

The following figure of the matrix T_k (cf. [1, p.96]) reaffirms the connection between the Lanczos method and the CG method:

$$T_k = \begin{pmatrix} \frac{1}{\alpha_0} & \frac{\sqrt{\beta_0}}{|\alpha_0|} & & & & & \\ \frac{\sqrt{\beta_0}}{|\alpha_0|} & \frac{1}{\alpha_1}+\frac{\beta_0}{\alpha_0} & \frac{\sqrt{\beta_1}}{|\alpha_1|} & & & & \\ & \frac{\sqrt{\beta_1}}{|\alpha_1|} & \frac{1}{\alpha_2}+\frac{\beta_1}{\alpha_1} & \cdot & & & \\ & & \cdot & \cdot & \cdot & & \\ & & & \cdot & \frac{1}{\alpha_{k-1}}+\frac{\beta_{k-2}}{\alpha_{k-2}} & \frac{\sqrt{\beta_{k-1}}}{|\alpha_{k-1}|} & \\ & & & & \frac{\sqrt{\beta_{k-1}}}{|\alpha_{k-1}|} & \frac{1}{\alpha_k}+\frac{\beta_{k-1}}{\alpha_{k-1}} \end{pmatrix}. \tag{4.13}$$

The representation of (4.13) provides an advantage for the implementation of the matrix T_k, since the components of the matrix can be generated from byproducts of the CG iterations. The question now arises how an optimal point of problem (4.12) is characterized.

Lemma 4.4 *Let $s^k = Q_k h^k$, with $h^k \in \mathbb{R}^{k+1}$ as a solution of problem* (4.12). *Then h^k is a solution of* (4.12) *exactly if s^k is a solution of problem* (4.8).

From Lemma 4.4 it can be seen that the columns of the matrix Q_k must be stored so that s^k can be generated from $Q_k h^k$.

The question now arises to what extent s^k is an approximation to s^M. Applying Theorem 2.1 to problem (4.12), there exists a solution $h^k \in \mathbb{R}^{k+1}$ of (4.12) with $\lambda_k \geq 0$, so that

$$(T_k + \lambda_k I_{k+1})h^k = -\gamma_0 e^1, \tag{4.14}$$

where $e^1 \in \mathbb{R}^{k+1}$ is the unit vector where there is a 1 in the first component and

$$(T_k + \lambda_k I_{k+1}) \succeq 0 \tag{4.15}$$

$$\lambda_k(\|h^k\|_2 - \triangle) = 0 \tag{4.16}$$

holds. By lemma 4.4, $s^k = Q_k h^k$ exists as a solution of (4.8) and $\|s^k\|_2 \leq \triangle$ holds. The following relation holds:

$$Q_k^T(H + \lambda_k I)s^k = Q_k^T(H + \lambda_k I)Q_k h^k = (T_k + \lambda_k I_{k+1})h^k = -\gamma_0 e^1 = -Q_k^T g. \tag{4.17}$$

According to (4.17), (4.16) and (4.15), s^k represents an approximation to s^M (cf. [1, p.223]). With respect to the solution of problem (4.12), we need to clarify below when an approximation s^k is considered acceptable. The following theorem provides such a criterion, with which the error $(H+\lambda_k M)s_k+g$ can be easily computed.

Satz 4.5

$$(H + \lambda_k M)s^k + g = \gamma_{k+1}\langle e^{k+1}, h^k\rangle q^{k+1} \tag{4.18}$$

and

$$\|(H + \lambda_k M)s^k + g\|_2 = \gamma_{k+1} \mid \langle e^{k+1}, h^k\rangle \mid \tag{4.19}$$

The solution of problem (4.12) is easier to calculate than the solution of (2.1). To solve problem (4.12), the tridiagonal structure of the matrix T_k and therefore also of the matrix $T_k + \lambda I_{k+1}$ can be exploited. If $\lambda > \theta_{min}$ is chosen, where $-\theta_{min}$ is the smallest eigenvalue of the matrix T_k, then $T_k + \lambda I_{k+1}$ is positive definite and there exists a Choletsky decomposition; $T_k + \lambda I_{k+1} = LL^T$. Let $h^k(\lambda) = -(T_k + \lambda I_{k+1})^+\gamma_0 e^1$. For numerical reasons (cf. [1, p.182]), solve instead of $\|h^k(\lambda)\|_2 - \triangle = 0$, the following equation:

$$\phi(\lambda) = \frac{1}{\|h^k(\lambda)\|_2} - \frac{1}{\triangle} = 0. \tag{4.20}$$

To determine the zero, Newton's method is applied to the function $\phi(\lambda)$ and the following algorithm is obtained.

Algorithmus 4.6 *Let $\lambda > \theta_{min}$ and $\triangle > 0$, perform the following steps:*

1.) Factorize $T_k + \lambda I_{k+1} = LL^T$.

2.) Solve $LL^T h = -\gamma_0 e^1$.

3.) Solve $L\omega = h$.

4.) Substitute λ by $\lambda^+ = \lambda + (\frac{\|h\|_2 - \triangle}{\triangle})(\frac{\|h\|_2^2}{\|\omega\|_2^2})$

Step (2.) is equivalent to solving the system of equations from (4.14). Step (3.) is needed to compute a component of $\phi'(\lambda)$ to perform a Newton step in step (4.) (cf. [1, p.185]). Another reason why problem (4.12) is easier to solve than the trust-region auxiliary problem is that the *severe case* for problem (4.12) cannot occur, as long as CG iterations can be generated (i.e., as long as $\alpha_i \neq 0$, $|\alpha_i|$, has not become vanishingly small $\forall i = 0, \ldots, k$,) or as long as the critical point has not yet been generated (i.e., as long as $\beta_i \neq 0$, $\forall i = 0, \ldots, k-1$).

Satz 4.7 *(vgl. [1, Lemma 7.5.11])A symmetric tridiagonal matrix is reducible exactly if one or more entries on the sub- and superdiagonal are* 0*. Otherwise the matrix is not reducible.*

Satz 4.8 *(vgl. [1, Theorem 7.5.12]) Assume that T_k is irreducible. Then the hard case cannot occur for problem* (4.12)*.*

Now the generalized Lanczos trust region method is introduced.

Algorithmus 4.9 *Let* $s^0 = 0$, $g^0 = g$, $\gamma_0 = \|g^0\|_2$, *and* $p^0 = -g^0$. *Set* `INTERIOR` *on true. For* $k = 0, 1, \ldots$ *perform the following steps until convergence:*

Set $\alpha_k = \|g^k\|_2^2 / \langle p^k, Hp^k \rangle$.
Generate T_k *from* T_{k-1}, *with* T_k *from* (4.13).
If `INTERIOR` *is true, but* $\alpha_k < 0$ *or* $\|s^k + \alpha_k p^k\|_M \geq \triangle$,

set `INTERIOR` *on wrong.*

If `INTERIOR` *is true, set*

$s^{k+1} = s^k + \alpha_k p^k$.

Otherwise, solve the Tridiagonal-Lanczos-Problem (4.12) *with algorithm 4.6 to estimate* h^k.

End if

Set $g^{k+1} = g^k + \alpha_k H p^k$.

If `INTERIOR` *is true,*

test for convergence with $\|g^{k+1}\|_2$.
Otherwise, test for convergence with $\gamma_{k+1} \mid \langle e^{k+1}, h^k \rangle \mid$.

End if

Set $\beta_k = \|g^{k+1}\|_2^2 / \|g^k\|_2^2$, *and*
$p^{k+1} = -g^{k+1} + \beta_k p^k$.

If `INTERIOR` *is false, generate* $s^k = Q_k h^k$, *with* Q_k *is generated from storage.*

As long as `INTERIOR` from algorithm 4.9 is set to true, the CG procedure results. The conjugate gradients are used to generate the Lanczos vectors $\{q^0, \ldots, q^k\}$ with (cf. [4, p.38]):

$$q^j = \sigma_k g^j / \|g^j\|_2,\ \sigma_k = -sgn(\alpha_{j-1})\sigma_{j-1},\ j = 0, \ldots, k \quad \text{und} \quad \sigma_0 = 1.$$

$sgn(\cdot)$ describes in this application the *sign function* (cf. [7]). The question now arises how, in the case of $\alpha_k = 0$ or $|\alpha_k| \leq \epsilon$ (with a sufficiently small $\epsilon > 0$), further iterations of algorithm 4.9 can be executed provided $k < n-1$ holds. Here we see that the conditions $\alpha_k = 0$ and $|\alpha_k| < \epsilon$ are implicitly skipped by the algorithm. In the following, we assume that algorithm 4.9 could not execute any more sequences in one iteration k. A possible solution in this case would be to generate all previously generated vectors $\{q^0, \ldots, q^k\}$ from a buffer. Then, starting from the last CG iterated one (here q^k), the Lanczos method (algorithm 3.5) is started. It is known from chapter 3.2 that the Lanczos method stops when an invariant subspace $\mathcal{K}(H, g^0, p),\ p > k$ becomes present. To continue the algorithm, a vector $q \in \mathbb{R}^n$ is needed for which holds:

$$q \perp \mathcal{K}(H, g^0, p). \tag{4.21}$$

This process could be continued until an invariant subspace appears again and a vector $q \in \mathbb{R}^n$ satisfying the condition from (4.21) is needed. Such a q could be generated based on the Gram-Schmidt orthogonalization process. This process could be continued until $\mathcal{K}(H, g^0, n-1) = \mathbb{R}^n,\ p < n-1$, is obtained and s^M can be computed as an exact solution using algorithm 4.9. It should be mentioned that for the described extension of Algorithm 4.9 the *severe case* may well occur (cf. [8, p.11]). The work [3] by Gould, Luicidi, Roma and Toint formulates a possibility to compute an exact solution at s^M also in the *severe case*, which, however, has not been proved in practical terms (cf. [4, p.38]).

5 Numerical experiment

This chapter will examine whether the theoretical concepts from the previous chapters can also be transferred to a practical application. For this purpose, four functions are considered which are minimized in the context of the General Trust Region Procedure (cf. [2, p.104]). The focus of this chapter is the question in how far the choice of the Steihaug-Toint method or the Generalized Lanczos Trust-Region method, influences the calculation of a solution. In this framework the points; accuracy of the computed solution, number of iterations and the runtime behavior are examined more near.

5.1 Test data

In this application, only twice-continuous differentiable functions are considered. The numerical experiments are performed using the following functions.

$$f_1(x) = (x_1 + 3)^2 + x_2^2,$$

$$f_2(x) = 10(x_1 - x_2^2) + (1 - x_1)^2,$$

$$f_3(x) = (-13 + x_1 + ((5 - x_2)x_2 - 2)x_2)^2 + (-29 + x_1((x_2 + 1)x_2 - 14)x_2)^2,$$

$$f_4(x) = f_{4.1}(x) + f_{4.2}(x) + f_{4.3}(x) + f_{4.4}(x) + f_{4.5}(x) + f_{4.6}(x),$$

with $f_{4.1}(x) = 100(x_1^2 - x_2)^2$, $f_{4.2}(x) = (x_1 - 1)^2$, $f_{4.3}(x) = (x_3 - 1)^2$, $f_{4.4}(x) = 90(x_3^2 - x_4)^2$, $f_{4.5}(x) = 10.1((x_2 - 1)^2 + (x_4 - 1)^2)$, $f_{4.6}(x) = 19.8(x_2 - 1)(x_4 - 1)$. $f_1(x)$ is a convex function where the global minimum point is found in $\bar{x} = (-3, 0)^T$. $f_2(x)$ is known as *Rosenbrock function* and a local minimum point of the non-convex function is found in $\bar{x} = (1, 1)^T$. $f_3(x)$ is also a non-convex function whose local minimum point is at $\bar{x} = (11.41..., -0.8968...)^T$. In the case of $f_4(x)$, it is also a nonconvex function whose local minimum point is found in $\bar{x} = (1, 1, 1, 1)^T$. It should also be mentioned that for all $q(s)$ from (2.1), the actual Hessian matrix of the functions has been used.
In the present application, the trust-region method from [2, p.104] is implemented with $\epsilon = 0.0001$, $\triangle \in [0, 1)$ and $\eta = 0.25$. The implemented Steihaug-Toint method converges if $k > k_{max}$, where k_{max} corresponds to the dimension of the function space (n), or

$$\|g^{k+1}\|_2 \leq \|g^0\|_2 \min\{\kappa_{fgr}, \|g^0\|_2^{\theta}\} \tag{5.1}$$

applies. The termination criterion (5.1) and the choice of parameters is based on [1, p.207]. In the Generalized Lanczos Trust-Region Method all termination criteria are implemented which can also be found in the Steihaug-Toint method and additionally the termination conditions

$$\|g^{k+1}\|_2 < \epsilon_{VLTRM} \text{ with } \epsilon_{VLTRM} > 0,$$

$$(H + \lambda_k I)s^k - g < \epsilon_{VLTRM}.$$

5.2 Results

The following table lists the test results of the functions $f_i(x),\ i = 1, 2, 3, 4$. Here, the required iterations of the general trust-region procedure until convergence are documented, the corresponding runtimes and the resulting approximation or exact solution. In addition, it is differentiated which procedure

is used to solve the trust-region auxiliary problem inexactly. In the following, the abbreviation STV for Steihaug-Toint method, VLTRM for Generalized Lanczos Trust Region Method and ATRV for General Trust Region Method will be introduced.

$f_i(x)$	Methods	Iterations	Runtime [Sec.]	Approximation
$i = 1$	STV	9	0.003186	(-3, 0)
$i = 1$	VLTRM	9	0.011030	(-3, 0)
$i = 2$	STV	17	0.004374	(1.00000234, 1.00000117)
$i = 2$	VLTRM	15	0.011057	(1.00000023, 1.00000012)
$i = 3$	STV	16	0.004966	(11.41277904, -0.8968025)
$i = 3$	VLTRM	38	0.054623	(11.41277908, -0.8968025)
$i = 4$	STV	65	0.022117	$\bar{x}^T_{STV}$
$i = 4$	VLTRM	233	0.417214	$\bar{x}^T_{VLTRM}$

Tabelle 1: In this table, the following parameters were used to solve the trust region auxiliary problem: $\epsilon_{VLTRM} = 0.00001$, $\kappa_{fgr} = 0.1$, and $\theta = 0.5$.

$\bar{x}^T_{STV} = (1,\ 1,\ 1,\ 1)$ and $\bar{x}^T_{VLTRM} = (1.00000008,\ 1.00000014,\ 1.00000008,\ 1.00000016)$. For comparison, the test results with tightened termination conditions are presented.

$f_i(x)$	Methods	Iterations	Runtime [Sec.]	Approximation
$i = 1$	STV	9	0.001727	(-3, 0)
$i = 1$	VLTRM	9	0.005571	(-3, 0)
$i = 2$	STV	13	0.003071	(0.99999563, 0.99999781)
$i = 2$	VLTRM	15	0.008977	(1.00000023, 1.00000012)
$i = 3$	STV	14	0.007144	(11.41277932, -0.89680523)
$i = 3$	VLTRM	38	0.047495	(11.41277908, -0.89680525)
$i = 4$	STV	51	0.021609	$\tilde{x}^T_{STV}$
$i = 4$	VLTRM	453	0.889066	$\tilde{x}^T_{VLTRM}$

Tabelle 2: In this table, the following parameters were used to solve the trust region auxiliary problem: $\epsilon_{VLTRM} = 0.0000001$, $\kappa_{fgr} = 0.001$, and $\theta = 0.005$.

$\tilde{x}^T_{STV} = (1.00000002, 1.00000003, 0.99999998, 0.99999997)$ and $\tilde{x}^T_{VLTRM} = (1, 1, 1, 1.00000001)$. Before evaluating the results, it should be noted that the results presented in Table 1 and Table 2 are only for low dimensional problems. When Algorithm 4.2 and Algorithm 4.9 are applied to a high dimensional (in this work $n > 4$) problem and compared, there is a possibility that the runtime behavior, iterations, and approximation accuracy, in particular, may differ from the results presented in Table 1 and Table 2. Moreover, both termination criteria were adjusted to the same factor (10^{-2}).
From table 1 and 2 it is clear that in the case of a convex function (here $f_1(x)$) and a sparse Hessian matrix, the computation of an exact solution is possible in a small running time. Due to the convexity of $f_1(x)$ (and therefore also for $q(s)$), it can be confirmed in practical applications that the STV and the VLTRM exploit the efficiency of the conjugate directions.
The example of the Rosenbrock function ($f_2(x)$) in table 1 shows that the ATRV with the VLTRM can generate a more accurate solution in fewer iterations (but a higher runtime) than the ATRV with the STV. By adjusting the termination criterion, the solution with the VLTRM remains unchanged, while the solution with the STV becomes even more inaccurate.
For $f_3(x)$ both variants calculate an identical approximation, although the ATRV with the VLTRM needs more than twice as many iterations as the ATRV with the STV. By tightening the termination criterion, the computed approximations and the running time remain identical to the results from table 2.
In the case of the function $f_4(x)$, there appears to be a large dependence of the VLTRM and STV on the termination criterion. In Table 1, the ATRV with the STV generates an exact solution in a fraction of the run time and iterations that it takes the ATRV with the VLTRM to compute an approximation. When the termination criteria are tightened as in Table 2, the ATRV with the VLTRM generates a near exact solution. The ATRV with the STV generates only an approximation in this case.

In the following, analogous to table 1 and 2, the results for the trust-region gradient procedure (cf. [2, p.106]) are presented. The trust-region gradient procedure will be abbreviated TRGV in the following. No variation of the truncation criterion is given, since no new findings are obtained.

$f_i(x)$	Methods	Iterations	Runtime [Sec.]	Approximation
$i = 1$	TRGV	23	0.001667	(-2.99998168e+00, 7.85161356e-06)
$i = 2$	TRGV	3277	0.258891	(0.99996675, 0.99998328)
$i = 3$	TRGV	7237	0.933731	(11.41271345, -0.8968092)
$i = 4$	TRGV	20060	2.461517	$\dot{x}_{STV}^T$

Tabelle 3: This table shows the test results with the TRGV.

$\dot{x}_{STV}^T = (1.00002626, 1.00005272, 0.9999737, 0.99994731)$. In the first case, the ATRV with the TRGV generates a significantly less accurate approximation than the results from table 1 and table 2. The runtime needed by the ATRV with the TRGV to fulfill the termination criterion is lower than the runtimes from table 1 and table 2. This could be justified by the fact that the gradient search directions in this example can be computed much more cheaply (inexpensively in terms of computing power) than the conjugate directions from algorithm 4.2 and algorithm 4.9.
In table 3, for $f_j(x)$, $j = 2, 3, 4$ are significantly less accurate approximations than in table 1 and table 2, despite the high number of iterations. An interesting observation can be made here. The search directions generated by STV and VLTRM (in the context of ATRV applied to $f_j(x)$, $j = 2, 3, 4$) have to generate a larger descent in most iterations than the search directions generated by TRGV. This fact is reflected in a significantly higher iteration count and runtime in table 3. From Lemma 4.3 it is known anyway that the STV generates at least the descent generated by the search direction of the Cauchy point.
Lastly, it should be mentioned that Algorithm 4.9 in the case of functions $f_j(x)$, $j = 3, 4$ has significantly lower running times and iterations when the algorithm solves the problem of type (4.12) once and then terminates. The resulting approximations differ from the results in table 2 in the seventh decimal place.

$f_i(x)$	Methods	Iterations	Runtime [Sec.]	Approximation
$i = 3$	VLTRM	22	0.015410	(11.41277933, -0.89680524)
$i = 4$	VLTRM	126	0.147711	x_{STV}^T

Tabelle 4: Test results for the scenario described above.

$x_{STV}^T = (1.00000001, 1.00000004, 1.00000002, 1.00000003)$.

The ATRV was able to compute a more accurate solution than the STV in many cases using the VLTRM. In terms of iteration count, run time, and implementation overhead, STV proved to be more efficient all things considered. Both methods produced significantly better results than TRGV with respect to the application of ATRV to the test functions.

6 Conclusion

In this paper, two methods have been presented to solve the trust-region auxiliary problem inexactly: the Steihaug-Toint method and the Generalized Lanczos Trust-Region method. Both methods involve the basic concept of conjugate gradients. The Steihaug-Toint method is modified with the essential result of the Lanczos method, the matrix T_k, to form the Generalized Lanczos-Trust-Region method. In particular, the last chapter has shown that solutions generated by the General Trust-Region method with the Steihaug-Toint method and the Generalized Lanczos-Trust-Region method were closer to the exact solution than with the Trust-Region gradient method. It should be mentioned, however, that the trust-region gradient method can be used to generate solutions very efficiently (e.g., for convex-square functions).
In this thesis by far not all applications of the Lanczos method could be presented. It was already mentioned at the end of chapter 4 that algorithm 4.9 can be extended with the Lanczos method to compute an exact solution of the trust-region auxiliary problem. In fact, it has been seen in Table 4 that small modifications of the Generalized Lanczos Trust Region Method can lead to a much more efficiently computed solution. The Lanczos method is used in many cases to approximate the actual Hessian matrix of a function (cf. [1, p.101]).
A question which was not answered in the course of the last chapter is how the Steihaug-Toint method and the Generalized Lanczos trust-region method generate a solution when the trust-region auxiliary problem to be solved is of high dimension.

Literatur

[1] A.Conn, N.Gould, P.Toint *Trust Region Methods*

[2] O.Stein *Grundzüge der Nichtlinearen Optimierung*

[3] N.Gould, S.Lucidi, M.Roma, P.Toint *Solving the Trust-Region-Subproblem using Lanczos-Method*

[4] C.Fortin *A Survey of the Trust Region Subproblem within a Semidefinite Framework*

[5] Nocedal and Wright *Numerical Optimization*

[6] R.Mohr *Sonderübungsblatt 2*

[7] Anonym *Graphing Absolute values, Greatest Integer Signum Functions*

[8] F. Lenders, C. Kirches, A. Potschka *A vector-free implementation of the GLTR method for iterative solution of the trust region problem*

[9] R.Hauser *Lecture 6, Continuous Optimisation*

[10] N.Gould *An introduction to algorithms for continuous optimization*