Computational solutions of a family of generalized Procrustes problems

Jens Fankhänel, Peter Benner

Preprint 2014-6

Preprintreihe der Fakultät für Mathematik
ISSN 1614-8835

Contents

1 Introduction

Procrustes problems ask for a transformation which matches a given matrix A as close as possible to a target matrix B subject to some constraints on the feasible transformations. The traditional example is the orthogonal Procrustes problem:

$$\begin{aligned} &\text{minimize } ||AQ - B||_F \\ &\text{s. t. } Q^T Q = I, \end{aligned} \tag{1.1}$$

where $||\cdot||_F$ denotes the Frobenius norm of a matrix. This problem was solved by Schönemann in 1966 [40].
Several Procrustes problems differ in at least one of the following two items:

- Which constraints are imposed on the feasible transformations?
- How is the distance between the target matrix and the transformation of the matrix A measured?

In the orthogonal Procrustes problem the feasible transformations are imposed to be rotations and/or reflections. Schönemann extended this problem by additionally admitting translations. A further extension is to allow to multiply the matrix A by a scaling factor, too [41], [7].
Admitting scaling arbitrary but orthogonal axes by different nonnegative factors leads to the real semidefinite Procrustes problem which was considered by Allwright [2]. Kiskiras and Halikias considered the complex case in [27].
The distance between the target matrix and the transformation of the matrix A is often measured by the Frobenius norm. However, other measures are possible. Watson considered the Procrustes problem for a family of orthogonally invariant norms which contains (1.1) as a spezial case [47].
Kintzel explored the Procrustes problem in indefinite inner product spaces [26]. The feasible transformations are imposed to be isometric with respect to an indefinite inner product. The distance between the target matrix and the transformation of the matrix A is also measured by an indefinite inner product.
Procrustes problems have many applications in statistics, robotics and computer graphics.
This paper is organized as follows: In chapter 2 we define the (ℓ_p, ℓ_q) Procrustes problem and show a decomposition into smaller problems for $p \neq 2$. Furthermore, we display some numerical results for the real cases as well as the complex case with $q = 2$. In chapter 3 we discuss some optimization methods for the remaining cases with $p \neq 2$. Chapter 4 treats applications for these cases. The cases $p = 2$ and $q \neq 2$ remain open and are under current investigation. However, there is a paper from Trendafilov [43] which treats the real case for $p = 2$ and $q = 1$. The case $p = 2 = q$ is, of course, the well known orthogonal Procrustes problem (1.1).
We denote the ℓ_p norm of a row or column vector v by $||v||_p$. v^T or v^* is the transposed or the conjugate transposed of the vector v. $\Re z$ or $\Im z$ means the real or the imaginary part of a complex number z, respectively. Let A be a matrix. Then $a_{j\cdot}$, $a_{\cdot k}$ or a_{jk} denotes the j-th row vector, the k-th column vector or the (j,k)-th entry of A, respectively. The symbol i is the imaginary unit which is a square root of -1.

2 The (ℓ_p, ℓ_q) Procrustes problem

Definition 2.1. *Let $\mathbb{F} = \mathbb{R}$ be the field of the real numbers or $\mathbb{F} = \mathbb{C}$ be the field of the complex numbers.*
For a natural number n and a real number $p \in [1, \infty]$, we define the set of ***isometries*** *with respect to the ℓ_p norm:*

$$\mathcal{U}_{n,p} = \left\{ U \in \mathbb{F}^{n \times n} \quad | \quad ||U^* v||_p = ||v||_p \qquad \forall v \in \mathbb{F}^n \right\}.$$

Furthermore we define for natural numbers $n < N$, real numbers $p \in [1, \infty]$, $q \in [1, \infty)$ and an ordered pair (A, B) with $A, B \in \mathbb{F}^{N \times n}$ the (ℓ_p, ℓ_q) Procrustes problem:

$$\begin{aligned} r_{p,q}(A,B) &= \sum_{j=1}^{N} \left|\left| \left(a_{j\cdot} U_{A,B}^{(p,q)} - b_{j\cdot} \right)^* \right|\right|_q^q \\ &= \min_{U \in \mathcal{U}_{n,p}} \sum_{j=1}^{N} \left|\left| (a_{j\cdot} U - b_{j\cdot})^* \right|\right|_q^q \end{aligned} \tag{2.1}$$

as well as the (ℓ_p, ℓ_∞) Procrustes problem:

$$\begin{aligned} r_{p,\infty}(A,B) &= \max_{1 \le j \le N} \left|\left| \left(a_{j\cdot} U_{A,B}^{(p,\infty)} - b_{j\cdot} \right)^* \right|\right|_\infty \\ &= \min_{U \in \mathcal{U}_{n,p}} \max_{1 \le j \le N} \left|\left| (a_{j\cdot} U - b_{j\cdot})^* \right|\right|_\infty . \end{aligned} \tag{2.2}$$

The matrix $U_{A,B}^{(p,q)} \in \mathcal{U}_{n,p}$ is called a solution of the (ℓ_p, ℓ_q) Procrustes problems ($q \in [1, \infty]$) with respect to the pair of matrices (A, B).

If $p = q = 2$, we get exactly the usual orthogonal Procrustes problem, which was investigated in [40] for example. According to [26] and [40] this problem can be solved by the positive semidefinite polar decomposition

$$A^* B = U_{A,B}^{(2,2)} M, \tag{2.3}$$

where $U_{A,B}^{(2,2)}$ is a unitary matrix and M is a positive semidefinite matrix. The factor $U_{A,B}^{(2,2)}$ of (2.3) is a solution of the minimization problem (2.1), if $p = q = 2$.
Trendafilov explored the real (ℓ_2, ℓ_1) Procrustes problem in [43] and together with Watson in [44].

Proposition 2.2.

(i) A matrix $U \in \mathbb{F}^{n \times n}$ is an isometry with respect to the ℓ_2 norm, if and only if U is a unitary matrix.

(ii) *Let $p \in [1,\infty]\backslash\{2\}$. A matrix $U \in \mathbb{F}^{n\times n}$ is an isometry with respect to the ℓ_p norm, if and only if $U = DP$, where P is a permutation matrix and D is a unitary diagonal matrix.*

Item (i) is trivial, and item (ii) is proved in [4] and [31] for example.

Remark 2.3. *In the cases (ii) of Proposition 2.2, it is true that $U = DP = P\tilde{D}$, where $\tilde{D}$ is a unitary diagonal matrix, too, and can be obtained by permutation of the entries of the matrix D.*

2.1 The cases with $p \neq 2$

Proposition 2.2 (ii) gives good reason for the following assumption:

Conjecture 2.4. *Let $p \in [1,\infty]\setminus\{2\}$, $q \in [1,\infty]$ and $\mathbb{F}, n, N, A, B$ as in Definition 2.1. Then we can decompose the solution of the (ℓ_p, ℓ_q) Procrustes problem into three steps. First we compute at most n^2 candidates for the entries of a unitary diagonal matrix D. Then we search an optimal permutation matrix P. Finally we construct D and compute $U_{A,B}^{(p,q)} = DP$.*

For each pair $(j,k) \in \{1,2,...,n\}^2$ there is a permutation matrix $P \in \{0,1\}^{n\times n}$ with $p_{jk} = 1$.[1] Because of this we could compute an optimal factor $\mu_{jk}^{(q)} \in \mathbb{F}$ for the j-th column of A with $\left|\mu_{jk}^{(q)}\right| = 1$. The actual affiliation of this factor to the diagonal matrix D depends apparently only on the j-th row of the optimal permutation matrix which is found in the second step. We can do so for all pairs $(j,k) \in \{1,2,...,n\}^2$. This gives rise to the following definitions of numbers $c_{jk}^{(q)}$, $\mu_{jk}^{(q)}$ and functions $\psi_{jk}^{(q)}$:

$$\begin{aligned} c_{jk}^{(q)} &= \sum_{l=1}^{N} \left|\mu_{jk}^{(q)} \cdot a_{lj} - b_{lk}\right|^q \\ &= \min_{|\mu|=1} \underbrace{\sum_{l=1}^{N} |\mu \cdot a_{lj} - b_{lk}|^q}_{=:\psi_{jk}^{(q)}(\mu)} \qquad \text{for } q \in [1,\infty) \\ \text{or} \quad c_{jk}^{(\infty)} &= \max_{1\le l\le N} \left|\mu_{jk}^{(\infty)} \cdot a_{lj} - b_{lk}\right| \\ &= \min_{|\mu|=1} \underbrace{\max_{1\le l\le N} |\mu \cdot a_{lj} - b_{lk}|}_{=:\psi_{jk}^{(\infty)}(\mu)} \qquad j,k = 1,2,...,n, \end{aligned} \tag{2.4}$$

respectively. The optimal numbers $\mu_{jk}^{(q)}$ are candidates for the diagonal entries of the matrix D in the sense of Conjecture 2.4, and the nonnegative real numbers $c_{jk}^{(q)}$ are the contributions of the k-th coordinate to the residual $r_{p,q}(A,B)$ in (2.1). We want to clarify this issue exactly:

Lemma 2.5. *Let $p \in [1,\infty]\setminus\{2\}$, $q \in [1,\infty]$ and $\mathbb{F}, n, N, A, B, r_{p,q}(A,B)$ as in Definition 2.1. Assume $U_{A,B}^{(p,q)} = DP$ is a solution of the (ℓ_p, ℓ_q) Procrustes problems with respect to the pair of matrices (A,B), where P is a permutation matrix and D a unitary diagonal matrix. Furthermore, let $\mu_{jk}^{(q)}$ and $c_{jk}^{(q)}$ $(j,k = 1,2,...,n)$ be as in (2.4). Then the following two statements hold:*

(i)

$$p_{jk} = 1 \quad \Longrightarrow \quad d_{jj} = \mu_{jk}^{(q)}. \tag{2.5}$$

[1] Indeed there are $(n-1)!$ permutation matrices with $p_{jk} = 1$.

(ii)

$$r_{p,q}(A,B) = \begin{cases} \sum_{j=1}^{n}\sum_{k=1}^{n} c_{jk}^{(q)} \cdot p_{jk}, & q < \infty \\ \max_{1\le j\le n}\max_{1\le k\le n} c_{jk}^{(q)} \cdot p_{jk}, & q = \infty. \end{cases}$$

Proof. Any solution of the (ℓ_p, ℓ_q) Procrustes problems has the form $U_{A,B}^{(p,q)} = DP$ according to Proposition 2.2 (ii), because $p \neq 2$.
(i): The permutation matrix P represents a bijective function

$$\begin{array}{rccc} & \pi : \{1,2,...,n\} & \longrightarrow & \{1,2,...,n\} \\ \text{with} & \pi(k) = j & \Longleftrightarrow & p_{jk} = 1. \end{array}$$

So we obtain for $q < \infty$ from (2.1):

$$\begin{aligned} r_{p,q}(A,B) &= \min_{U\in\mathcal{U}_{n,p}} \sum_{l=1}^{N} \left|\left|(a_{l\cdot}U - b_{l\cdot})^*\right|\right|_q^q \\ &= \sum_{l=1}^{N} \left|\left|\left(a_{l\cdot}U_{A,B}^{(p,q)} - b_{l\cdot}\right)^*\right|\right|_q^q \\ &= \sum_{l=1}^{N} \left|\left|(a_{l\cdot}DP - b_{l\cdot})^*\right|\right|_q^q \\ &= \sum_{l=1}^{N}\sum_{k=1}^{n} \left|a_{l,\pi(k)}d_{\pi(k),\pi(k)} - b_{l,k}\right|^q \\ &= \sum_{k=1}^{n}\sum_{l=1}^{N} \left|a_{l,\pi(k)}d_{\pi(k),\pi(k)} - b_{l,k}\right|^q \end{aligned} \tag{2.6}$$

$$\Longrightarrow \quad d_{\pi(k),\pi(k)} = \mu_{\pi(k),k}^{(q)}.$$

For $q = \infty$ we obtain from (2.2):

$$\begin{aligned} r_{p,q}(A,B) &= \min_{U\in\mathcal{U}_{n,p}} \max_{1\le l\le N} \left|\left|(a_{l\cdot}U - b_{l\cdot})^*\right|\right|_\infty \\ &= \max_{1\le l\le N} \left|\left|\left(a_{l\cdot}U_{A,B}^{(p,q)} - b_{l\cdot}\right)^*\right|\right|_\infty \\ &= \max_{1\le l\le N} \left|\left|(a_{l\cdot}DP - b_{l\cdot})^*\right|\right|_\infty \\ &= \max_{1\le l\le N}\max_{1\le k\le n} \left|a_{l,\pi(k)}d_{\pi(k),\pi(k)} - b_{l,k}\right| \\ &= \max_{1\le k\le n}\max_{1\le l\le N} \left|a_{l,\pi(k)}d_{\pi(k),\pi(k)} - b_{l,k}\right| \end{aligned} \tag{2.7}$$

$$\Longrightarrow \quad d_{\pi(k),\pi(k)} = \mu_{\pi(k),k}^{(\infty)}.$$

(ii) follows for $q < \infty$ from (2.4) and (2.6) by summation over the coordinates, for $q = \infty$ from (2.4) and (2.7) by taking the maximum over the coordinates.

□

Now we look for a permutation matrix P in the sense of Conjecture 2.4. Hereby we can hardly use the standard methods of analysis, because the set of the permutation matrices is discrete. Therefore it makes sense to use appropriate models of Discrete Mathematics.

Lemma 2.6. *Let $q \in [1, \infty]$ and $\mathbb{F}, n, N, p, A, B, r_{p,q}(A, B)$ as well as $U_{A,B}^{(p,q)} = DP$ as in Lemma 2.5. Assume, the n^2 one-dimensional minimization problems (2.4) are already solved. Then the complexity of the computation of the permutation matrix P is in $\mathcal{O}\left(n^3\right)$. In the case $q = \infty$ the complexity of the computation of the permutation matrix P is even in $\mathcal{O}\left(n^{2.5}\right)$.*

Proof. We construct a bipartite graph $G = (V, E)$ of the set of vertices $V = \{v_1, v_2, ..., v_n, w_1, w_2, ..., w_n\}$ and the set of edges $E = \big\{\{v_j, w_k\} \quad | \quad j, k \in \{1, 2, ..., n\}\big\}$. The vertex v_j represents the j-th row and the vertex w_k the k-th column of a matrix $K = K_{\tilde{G}} \in \{0, 1\}^{n \times n}$, which represents a subgraph $\tilde{G} = \left(V, \tilde{E}\right)$ with $\tilde{E} \subseteq E$. The entry (j, k) of the matrix $K_{\tilde{G}}$ is one if and only if the edge $\{v_j, w_k\}$ belongs to the set $\tilde{E}$. Furthermore we let the edge $\{v_j, w_k\} \in E$ possess the weight $c_{jk}^{(q)}$ from (2.4) (for $j, k = 1, 2, ..., n$).
Then $K_{\tilde{G}}$ is a permutation matrix if and only if $\tilde{G}$ is a perfect matching within the bipartite graph G. $U_{A,B}^{(p,q)} = DP$ minimizes the residual $r_{p,q}(A, B)$ in (2.1), so it follows from Lemma 2.5 for $q < \infty$, that the permutation matrix P represents a perfect matching of minimal weight. According to [29, chapter 11] this problem can be solved by the Successive Shortest Path Algorithm. This algorithm can according to [29, chapter 9] be implemented in such a way that its runtime is in $\mathcal{O}\left(|V| \cdot |E| + |V|^2 \cdot \log |V|\right)$. In our case we have $|V| = 2n, \quad |E| = n^2$, and so the rumtime of the algorithm is in $\mathcal{O}\left(n^3\right)$.
According to Lemma 2.5 for $q = \infty$, the permutation matrix P represents a perfect matching for which the largest edge weight is minimal. In other words: P is a solution of the bottleneck assignment problem or the bottleneck bipartite matching problem, respectively. In [38], Punnen and Nair suggest for this problem an algorithm of runtime in $\mathcal{O}\left(|V| \cdot \sqrt{|V| \cdot |E|}\right)$. In our case this yields a runtime in $\mathcal{O}\left(n^{2.5}\right)$.

□

The algorithm of Punnen and Nair consists of two phases:

- First it calls $\mathcal{O}(\log n)$ times an algorithm of Alt, Blum, Mehlhorn and Paul [3]. The algorithm of Alt et al. has a runtime in $\mathcal{O}\left(n^{2.5} / \log n\right)$ and applies methods from [1], [15] and [22]. The first phase of the algorithm of Punnen and Nair yields a matching of cardinality $n - \lceil \sqrt{n} \rceil$.
- The remaining $\lceil \sqrt{n} \rceil$ augmentations of the matching are computed by the Successive Shortest Path Algorithm with a modified cost function and a heuristic from [16].

This combined algorithm speeds up the runtime relatively to the exclusive application of the Successive Shortest Path Algorithm.
If we know the numbers $c_{jk}^{(q)}$ and $\mu_{jk}^{(q)} \quad (j, k = 1, 2, ..., n)$ as in (2.4), then we can find a permutation matrix P. Moreover, the implication (2.5) yields the construction of the unitary diagonal matrix D as in Conjecture 2.4. This shows that Conjecture 2.4 is true, if the one-dimensional minimization problems (2.4) are computable in finite time.

2.1.1 The solution of the one-dimensional minimization problems in the cases $\mathbb{F} = \mathbb{R}$ as well as $\mathbb{F} = \mathbb{C}$ and $q = 2$

Now we want to investigate how to solve the n^2 one-dimensional minimization problems (2.4). We look for the optimal solutions $\mu_{jk}^{(q)}$, for $j, k = 1, 2, ..., n$.
This is easy for $\mathbb{F} = \mathbb{R}$, because $\mu_{jk}^{(q)} \in \{-1, 1\} \quad \forall j, k \in \{1, 2, ..., n\}$. Therefore we must only calculate

the sum for both values and compare these two sums. The complexity of this computation is thus in $\mathcal{O}\left(Nn^2\right)$.

In the case $\mathbb{F} = \mathbb{C}$ and $q = 2$, (2.4) is a usual orthogonal Procrustes Problem which can be solved by a positive semidefinite polar decomposition (2.3) (for all $j, k \in \{1, 2, ..., n\}$.) This is particularly easy in the one-dimensional case. We compute

$$z_{jk} = a_{\cdot j}^* b_{\cdot k}.$$

The complex number z_{jk} possesses the representation

$$z_{jk} = \exp\left(i\xi_{jk}^{(2)}\right) \cdot |z_{jk}|\,; \qquad \xi_{jk}^{(2)} \in \mathbb{R}; \tag{2.8}$$

and this is a positive semidefinite polar decomposition, because $\left|\exp\left(i\xi_{jk}^{(2)}\right)\right| = 1$ and $|z_{jk}| \geq 0$. We can set

$$\mu_{jk}^{(2)} = \exp\left(i\xi_{jk}^{(2)}\right).$$

If $z_{jk} \neq 0$, then the factor $\exp\left(i\xi_{jk}^{(2)}\right)$ in (2.8) is unique. Otherwise, every unitary factor yields a positive semidefinite polar decomposition of the form (2.8). In that special case every complex number of modulus 1 is an optimal solution $\mu_{jk}^{(2)}$ for (2.4).

2.2 Numerical results

We programmed the algorithms as m-files in MATLAB[2] for the cases of subsection 2.1.1 and tested them. We always let $N = 2 \cdot n$ and drew pseudo-random numbers for all entries of A and B.

2.2.1 The case $\mathbb{F} = \mathbb{R}$ and $q \in [1, \infty)\backslash\{2\}$

We tested 50 examples for each entry of the Table 2.1 and Table 2.2.

We compare these results graphically to the graph of a function $\text{time} = c \cdot n^3$, because the complexity of the computation is in $\mathcal{O}\left(n^3\right)$. We determine the unknown factor c so that its cube root is the arithmetic mean of the cube roots of the factors c_{kjl} which would stand in this equation for the tested examples. The index j runs over the q's, the index k over the n's and the index l runs over the number of the examples.

$$\begin{aligned} \text{time}_{kjl} &= c_{kjl} \cdot n_k^3, \qquad\qquad k = 1, 2, ..., m_1, \quad j = 1, 2, ...m_2, \quad l = 1, 2, ..., m_3 \\ \text{and} \qquad \sqrt[3]{c} &= \frac{1}{m_1 m_2 m_3} \cdot \sum_{k=1}^{m_1} \sum_{j=1}^{m_2} \sum_{l=1}^{m_3} \sqrt[3]{c_{kjl}} \\ &= \frac{1}{m_1 m_2 m_3} \cdot \sum_{k=1}^{m_1} \sum_{j=1}^{m_2} \sum_{l=1}^{m_3} \sqrt[3]{\frac{\text{time}_{kjl}}{n_k^3}}. \end{aligned} \tag{2.9}$$

Here, we have $m_1 = 10, m_2 = 3$ and $m_3 = 50$. Figure 2.1 and 2.2 display the measured runtimes together with the graph of the function $c \cdot n^3$.

[2] MATLAB is a registered trademark of The MathWorks Inc.; see www.mathworks.com

2.2.2 The case $\mathbb{F} = \mathbb{C}$ and $q = 2$

We tested 50 examples for each problem size of Table 2.3 and drew pseudo-random numbers for the real and imaginary parts of all entries of A and B. In order to compare these results to the graph of a function time $= c \cdot n^3$, we compute a factor c as in (2.9) with $m_1 = 10, m_2 = 1$ and $m_3 = 50$. The measured runtimes are displayed in Table 2.3 and together with the graph of the function $c \cdot n^3$ in Figure 2.3.

2.2.3 The case $\mathbb{F} = \mathbb{R}$ and $q = \infty$

We tested 50 examples for each problem size of Table 2.4. The algorithm from Punnen and Nair promises a better runtime for the computation of the optimal permutation (see Lemma 2.6). However, the computation of the numbers $\mu_{jk}^{(\infty)}$ as in (2.4) requires cubic runtime. Therefore we use a function time $= c \cdot n^3$ with c as in as in (2.9) with $m_1 = 10, m_2 = 1$ and $m_3 = 50$. for the comparison to the expected runtime. The measured runtimes are displayed in Table 2.4 and together with the graph of the function $c \cdot n^3$ in Figure 2.4.
The runtimes are actually better than those in the previous tests. This is because of the fast algorithm from Punnen and Nair. Nevertheless we cannot give a better upper bound than $\mathcal{O}\left(n^3\right)$.

2.2.4 Conclusions of the numerical tests

The results confirm the stated complexity bounds. But the runtime of real examples can be better than the upper bound.

n	Average runtime in seconds		
	$q = 1.0$	$q = 1.2$	$q = 1.5$
10	0.014748	0.015023	0.015930
20	0.078757	0.084309	0.084140
30	0.234773	0.252455	0.252869
40	0.524623	0.564437	0.566878
50	0.991554	1.071351	1.073917
60	1.669336	1.801428	1.805245
70	2.611202	2.823891	2.826408
80	3.831442	4.124252	4.126859
90	5.367909	5.825500	5.831750
100	7.479148	8.101754	8.106456

Table 2.1: The measured runtimes for $\mathbb{F} = \mathbb{R}$ and $q \in [1, 2)$.

n	Average runtime in seconds		
	$q = 3$	$q = 5$	$q = 10$
10	0.015959	0.015149	0.015800
20	0.084742	0.085035	0.085819
30	0.255369	0.254879	0.256215
40	0.573667	0.573160	0.575470
50	1.087895	1.086968	1.088257
60	1.836464	1.838448	1.842627
70	2.883685	2.888313	2.889532
80	4.269644	4.249226	4.256956
90	6.022608	6.017283	6.028104
100	8.140372	8.135836	8.136754

Table 2.2: The measured runtimes for $\mathbb{F} = \mathbb{R}$ and $q \in (2, \infty)$.

n	Runtime in seconds		
	minimum	average	maximum
10	0.016921	0.017866	0.026170
20	0.086813	0.088923	0.099390
30	0.253735	0.257121	0.264710
40	0.558403	0.565227	0.576827
50	1.038469	1.053656	1.067858
60	1.750335	1.771959	1.788947
70	2.722435	2.761703	2.783121
80	4.024518	4.053560	4.109038
90	5.700010	5.719499	5.776263
100	7.677121	7.751119	7.804609

Table 2.3: The measured runtimes for $\mathbb{F} = \mathbb{C}$ and $q = 2$.

n	Runtime in seconds		
	minimum	average	maximum
10	0.008056	0.009847	0.020849
20	0.026735	0.030201	0.034344
30	0.058850	0.066005	0.076477
40	0.113653	0.126223	0.142218
50	0.191917	0.208091	0.232473
60	0.268390	0.292564	0.324753
70	0.401007	0.436926	0.464889
80	0.522879	0.564658	0.599515
90	0.709464	0.754060	0.807523
100	0.869627	0.932012	0.985356

Table 2.4: The measured runtimes for $\mathbb{F} = \mathbb{R}$ and $q = \infty$.

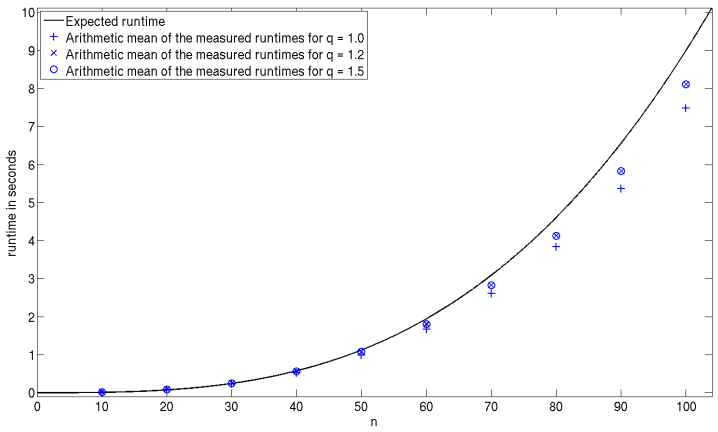

Figure 2.1: The black curve represents the graph of the function time $= c \cdot n^3$ with c as in (2.9). The stars correspond to the measured runtimes for $\mathbb{F} = \mathbb{R}$.

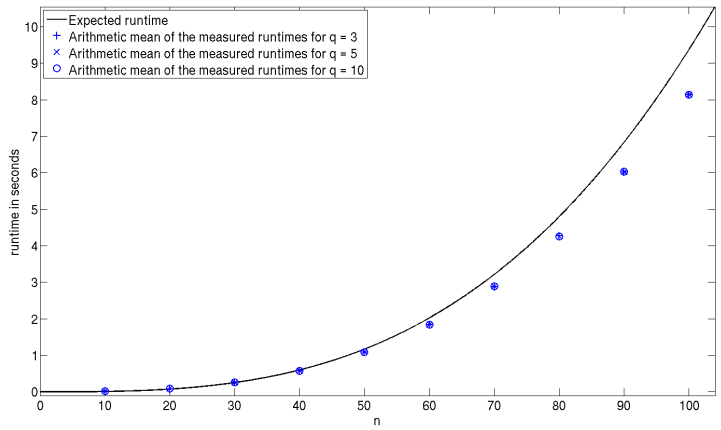

Figure 2.2: The black curve represents the graph of the function $\text{time} = c \cdot n^3$ with c as in (2.9). The stars correspond to the measured runtimes for $\mathbb{F} = \mathbb{R}$.

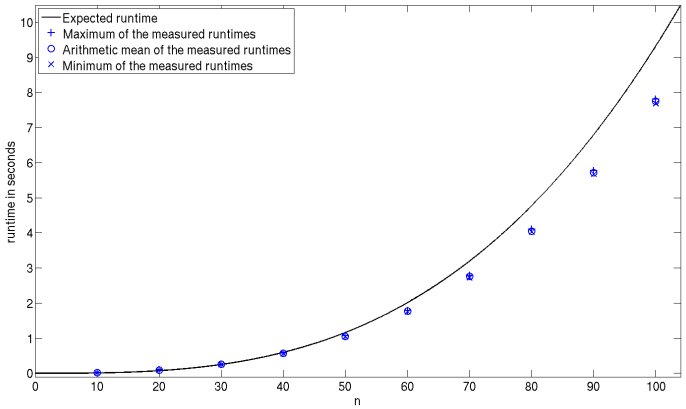

Figure 2.3: The black curve represents the graph of the function $\text{time} = c \cdot n^3$ with c as in (2.9). The stars correspond to the measured runtimes for $\mathbb{F} = \mathbb{C}$ and $q = 2$.

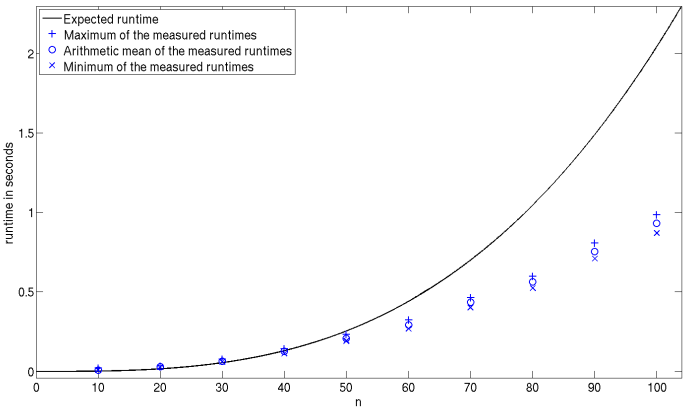

Figure 2.4: The black curve represents the graph of the function $\text{time} = c \cdot n^3$ with c as in (2.9). The stars correspond to the measured runtimes for $\mathbb{F} = \mathbb{R}$ and $q = \infty$.

3 Optimization methods for the remaining cases with $p \neq 2$

In the following chapter we will investigate the solution of the one-dimensional minimization problems (2.4) for $\mathbb{F} = \mathbb{C}$ and $q \in [1, \infty] \backslash \{2\}$. The different cases can be treated with various optimization algorithms which we will briefly review here.

3.1 Standard iteration methods for convex optimization

For convex minimization, interior-point methods and active-set methods are usually applied. There exist already several optimization solvers in which such methods are implemented.
The optimization solver KNITRO consists of three algorithms in one: a direct interior-point algorithm, an interior-point-CG-algorithm and an active-set-algorithm. The objective function as well as the constraints should be smooth. But convexity is not required. KNITRO is even globally convergent, but only to a local minimum.
The solver KNITRO is available on the NEOS[1] server. It accepts tasks in the modelling languages AMPL[2] and GAMS[3]. We used AMPL and sent several examples to the solver KNITRO on the NEOS solver.

3.2 Some statements about concave minimization and the SLA

If we transform our one-dimensional complex problem (2.4) into a two-dimensional real problem, then our feasible set is the unit circle. We can relax this problem to the closed unit disk in order to make our our feasible set convex. If our objective function is also convex, then each local minimum is a global minimum, too. But this minimum does not necessesarily lie on the unit circle.
Minimizing a concave function over a convex set has other advantages and disadvantages. It is well known that a continuous and concave function over a convex set has always a global minimum at an extreme point [5], [32]. This is quite a desired property for our problems. On the other hand, concave minimization is NP-hard in general [5], [34], [37]. The paper [5] provides also a survey about algorithms which yield always the exact solution. But such algorithms are inefficient, of course. It is often necessery to strike a good balance between speed and correctness.
Mangasarian developed the Successive Linearization Algorithm (SLA) for the minimization of a concave function over a polytope [17], [32]. The SLA finds always a stationary point, which is often also a global minimum. The SLA computes in each iteration

$$x^{(k+1)} \in \arg \min_{x \in \mathcal{E}(P)} \nabla f\left(x^{(k)}\right) \cdot \left(x - x^{(k)}\right), \tag{3.1}$$

where $\nabla f(\cdot)$ is the gradient of the objective function, and $\mathcal{E}(P)$ is the set of extrempoints of the feasible polytope P, which are the vertices of P. The algorithm stops if $\nabla f\left(x^{(k)}\right) \cdot \left(x^{(k+1)} - x^{(k)}\right) = 0$.

[1] http://www.neos-server.org/neos/solvers/index.html
[2] http://www.ampl.com
[3] http://www.gams.com

We can consider the closed unit disk to be a polytope with infinitely many vertices. But in our case we cannot expect, that $\nabla f\left(x^{(k)}\right) \cdot \left(x^{(k+1)} - x^{(k)}\right)$ becomes zeros in finite time, because we have infinitely many extreme points on the disk. Therefore we must introduce a tolerance parameter $\varepsilon > 0$, such that the algorithm terminates if $\nabla f\left(x^{(k)}\right) \cdot \left(x^{(k+1)} - x^{(k)}\right)$ is small enough.

3.3 A Branch-and-Bound-Method for concave minimization

Branch-and-Bound is one of the proposed methods for concave minimization in [5]. A Branch-and-Bound-Method decomposes the feasible set into finitely many pieces and computes a lower bound of the values of the objective function for each piece. In each step the algorithm considers a piece of the minimal lower bound and decomposes it into smaller pieces. Then the lower bounds for the new pieces are computed.
Schöning gives a detailed description of the Branch-and-Bound-principle in [42]. He suggests using a heap in order to find a search node of the minimal lower bound efficiently in each step. Nevertheless, the number of steps of a Branch-and-Bound-Method can grow exponentially with regard to the size of the problem. It is only appropriate for small instances of a problem.

3.4 An approach from Location Theory

This approach is concerned with the special case $q = 1$. Let $p_1, p_2, ..., p_N \in \mathbb{R}^2$ be given points in the Euclidean plane, called facilities, and $w_1, w_2, ..., w_N \in \mathbb{R}_+$ nonnegative weights. According to [39], the Euclidean single facility location problem (ESFL) asks for the minimum of the following function

$$\varphi(x) = \sum_{l=1}^{N} w_l \cdot ||x - p_l||_2, \qquad x \in \mathbb{R}^2. \tag{3.2}$$

In the ESFL we search a new facility x in the plane such that the sum of its Euclidean distances to the existing facililies is minimized. The ESFL is also called the general Fermat problem. Obviously, the ESFL is convex. Thus every local minimum is also a global minimum. Weiszfeld gave a simple iteration algorithm solving this problem [48]. Kuhn proved the global convergence of this algorithm [30].

Lemma 3.1. *The one-dimensional minimization problem* (2.4) *for* $q = 1$ *without the constraint* $|\mu| = 1$ *is equivalent to some Euclidean single facility location problem.*

Proof. For a pair of indices (j, k) it suffices to minimize

$$\begin{aligned}
\tilde{\psi}_{jk}^{(1)}(\mu) &= \sum_{l \in I_1(j,k)} |\mu \cdot a_{lj} - b_{lk}| \\
&= \sum_{l \in I_1(j,k)} \left| a_{lj} \cdot \left(\mu - \frac{b_{lk}}{a_{lj}} \right) \right| \\
&= \sum_{l \in I_1(j,k)} |a_{lj}| \cdot \left| \mu - \frac{b_{lk}}{a_{lj}} \right|.
\end{aligned}$$

We can consider the positive numbers $|a_{lj}|$ as weights, the complex numbers b_{lk}/a_{lj} as facilities in the real plane and the complex number μ as a new facility in the real plane.

□

It is also recommendable to apply a hyperbolic approximation of the fuction φ in (3.2) in order to avoid non-differentiability in any points [39]:

$$\tilde{\varphi}(x) = \sum_{l \in I_1} w_l \cdot \sqrt{(x - p_l)^T (x - p_l) + \epsilon}. \tag{3.3}$$

3.5 Addition of a paraboloid

We will transform our one-dimensional complex problem (2.4) into a two-dimensional real problem. Then our feasible set is the unit circle. We can exploit the special structur of our feasible set by adding a paraboloid

$$\lambda \cdot \left(1 - x^T x\right) \qquad \text{with some } \lambda \in \mathbb{R} \tag{3.4}$$

to our objective function. This modification has obviously no effect to the set of the feasible solutions because the paraboloid (3.4) vanishes on the unit circle for all $\lambda \in \mathbb{R}$.

4 The one-dimensional minimization problems in the cases $\mathbb{F} = \mathbb{C}$, $p \neq 2$ and $q \in [1, \infty] \backslash \{2\}$

For simplification we set:

$$\begin{aligned} \alpha_l^{(jk)} &:= 2 \cdot (\Re(a_{lj}) \cdot \Re(b_{lk}) + \Im(a_{lj}) \cdot \Im(b_{lk}))\,; \\ \beta_l^{(jk)} &:= 2 \cdot (\Re(a_{lj}) \cdot \Im(b_{lk}) - \Im(a_{lj}) \cdot \Re(b_{lk}))\,; \\ \gamma_l^{(jk)} &:= |a_{lj}|^2 + |b_{lk}|^2 \qquad\qquad \forall j,k \in \{1,2,...,n\}, \quad l \in \{1,2,...,N\}. \end{aligned} \tag{4.1}$$

We consider the functions $g_{jk}^{(q)}(\xi) := \psi_{jk}^{(q)}(\exp(i \cdot \xi))$ over an angle with $\mu = \exp(i \cdot \xi)$ with $\psi_{jk}^{(q)}$ as in (2.4). Together with the replacements (4.1) we can write these functions as follows:

$$\begin{aligned} g_{jk}^{(q)}(\xi) \;=\; & \begin{cases} \sum\limits_{l=1}^{N} \left(\gamma_l^{(jk)} - \alpha_l^{(jk)} \cdot \cos\xi - \beta_l^{(jk)} \cdot \sin\xi\right)^{q/2}, & q < \infty \\ \max\limits_{1 \le l \le N} \left(\gamma_l^{(jk)} - \alpha_l^{(jk)} \cdot \cos\xi - \beta_l^{(jk)} \cdot \sin\xi\right)^{1/2}, & \text{otherwise} \end{cases} \\ \text{with} \quad \mu \;=\; & \cos\xi + i \cdot \sin\xi \qquad j,k = 1,2,...,n. \end{aligned} \tag{4.2}$$

Vanishing summands can give rise to poles in the first derivative, if $q < 2$, and in the second derivative, if $q \in [1,4) \backslash \{2\}$. Therefore we define two index sets and slightly modified functions $\tilde{g}_{jk}^{(q)}(\xi)$ and $\hat{g}_{jk}^{(q)}(\xi)$ for $q < \infty$:

$$\begin{aligned} & I_1(j,k) := \{l \in \{1,2,...,N\} \;\mid\; a_{lj} \neq 0 \;\wedge\; b_{lk} \neq 0\}, \\ & \tilde{g}_{jk}^{(q)}(\xi) := \sum_{l \in I_1(j,k)} \left(\gamma_l^{(jk)} - \alpha_l^{(jk)} \cdot \cos\xi - \beta_l^{(jk)} \cdot \sin\xi\right)^{q/2} \\ \text{and} \quad & I_2(j,k) := \left\{l \in \{1,2,...,N\} \;\mid\; \alpha_l^{(jk)} \neq 0 \;\vee\; \beta_l^{(jk)} \neq 0\right\}, \\ & \hat{g}_{jk}^{(q)}(\xi) := \sum_{l \in I_2(j,k)} \left(\gamma_l^{(jk)} - \alpha_l^{(jk)} \cdot \cos\xi - \beta_l^{(jk)} \cdot \sin\xi\right)^{q/2}. \end{aligned} \tag{4.3}$$

Obviously the differences $g_{jk}^{(q)}(\xi) - \tilde{g}_{jk}^{(q)}(\xi)$ and $g_{jk}^{(q)}(\xi) - \hat{g}_{jk}^{(q)}(\xi)$ are independent of ξ. Thus the derivative of $\tilde{g}_{jk}^{(q)}(\xi)$ or $\hat{g}_{jk}^{(q)}(\xi)$, respectively, has exactly the same value as the derivative of $g_{jk}^{(q)}(\xi)$ for all $\xi \in \mathbb{R}$. In particular, $\xi \in \mathbb{R}$ minimizes $g_{jk}^{(q)}(\xi)$ if and only if it minimizes $\tilde{g}_{jk}^{(q)}(\xi)$ or $\hat{g}_{jk}^{(q)}(\xi)$, respectively.

4.1 Transformation into two-dimensional real minimization problems

We transform the one-dimensional complex minimization problems (4.2) into two-dimensional real minimization problems. Hereby we replace the functions $g_{jk}^{(q)}(\xi) = \psi_{jk}^{(q)}(\exp(i \cdot \xi)) = \psi(\mu)$ by functions $f_{jk}^{(q)}(x)$ where $x = \begin{bmatrix} \cos\xi & \sin\xi \end{bmatrix}^T = \begin{bmatrix} \Re\mu & \Im\mu \end{bmatrix}^T$.
In the cases $q < \infty$ we can write the problem to minimize (4.2) as follows:

$$\begin{aligned} \min f_{jk}^{(q)}(x) &= \sum_{l \in I_1(j,k)} \left(\gamma_l^{(jk)} - \begin{bmatrix} \alpha_l^{(jk)} & \beta_l^{(jk)} \end{bmatrix} \cdot x \right)^{q/2} \\ \text{s. t. } x^T x &= 1, \quad x \in \mathbb{R}^2 \end{aligned} \tag{4.4}$$

with $\alpha_l^{(jk)}$, $\beta_l^{(jk)}$ and $\gamma_l^{(jk)}$ $l \in \{1, 2, ..., N\}$ as in (4.1). In the case $q = \infty$ we can write it as follows:

$$\begin{aligned} &\min f_{jk}^{(\infty)} \\ \text{s. t. } f_{jk}^{(\infty)}(x) &\geq \sqrt{\gamma_l^{(jk)} - \begin{bmatrix} \alpha_l^{(jk)} & \beta_l^{(jk)} \end{bmatrix} \cdot x}, \qquad l = 1, 2, ..., N \\ x^T x &= 1, \quad x \in \mathbb{R}^2. \end{aligned} \tag{4.5}$$

Obviously, $\tilde{g}_{jk}^{(q)}(\xi) \equiv f_{jk}^{(q)}\left(\begin{bmatrix} \cos\xi \\ \sin\xi \end{bmatrix} \right)$ for $q < \infty$ and $g_{jk}^{(\infty)}(\xi)$ is equal to the minimal possible value of $f_{jk}^{(\infty)}\left(\begin{bmatrix} \cos\xi \\ \sin\xi \end{bmatrix} \right)$ for all $\xi \in \mathbb{R}$.
It is well known, that a local minimum of a convex program is always a global minimum, too. But the circle in the constraint of (4.4) and the $(N+1)$-th constraint of (4.5) is not a convex set. Therefore let us consider the following relaxed program for $q < \infty$:

$$\begin{aligned} \min f_{jk}^{(q)}(x) &= \sum_{l \in I_1(j,k)} \left(\gamma_l^{(jk)} - \begin{bmatrix} \alpha_l^{(jk)} & \beta_l^{(jk)} \end{bmatrix} \cdot x \right)^{q/2} \\ \text{s. t. } x^T x &\leq 1, \quad x \in \mathbb{R}^2 \end{aligned} \tag{4.6}$$

with $\alpha_l^{(jk)}$, $\beta_l^{(jk)}$ and $\gamma_l^{(jk)}$ $l \in \{1, 2, ..., N\}$ as in (4.1).

Lemma 4.1. *If $q \in [2, \infty)$, then the program (4.6) is convex in the closed unit disk.*

Proof. Obviously, the Hessian matrix of the objective function is zero for $q = 2$. For $q \in (2, \infty)$ we consider the l-th summand of the objective function:

$$f_{jkl}^{(q)}(x) := \left(\gamma_l^{(jk)} - \begin{bmatrix} \alpha_l^{(jk)} & \beta_l^{(jk)} \end{bmatrix} \cdot x \right)^{q/2}$$

When this summand is nonzero, then we can calculate the Hessian matrix:

$$\begin{aligned}\frac{d^2 f_{jkl}^{(q)}(x)}{dx^2} &= \frac{q}{2} \cdot \left(\frac{q}{2} - 1\right) \cdot \left(\gamma_l^{(jk)} - \begin{bmatrix}\alpha_l^{(jk)} & \beta_l^{(jk)}\end{bmatrix} \cdot x\right)^{q/2-2} \cdot \begin{bmatrix}\left(\alpha_l^{(jk)}\right)^2 & \alpha_l^{(jk)} \cdot \beta_l^{(jk)} \\ \alpha_l^{(jk)} \cdot \beta_l^{(jk)} & \left(\beta_l^{(jk)}\right)^2\end{bmatrix} \\ &= \begin{bmatrix}\alpha_l^{(jk)} \\ \beta_l^{(jk)}\end{bmatrix} \cdot \begin{bmatrix}\alpha_l^{(jk)} & \beta_l^{(jk)}\end{bmatrix} \cdot \underbrace{\frac{q}{2} \cdot \left(\frac{q}{2} - 1\right)}_{\geq 0} \cdot \underbrace{\left(\gamma_l^{(jk)} - \begin{bmatrix}\alpha_l^{(jk)} & \beta_l^{(jk)}\end{bmatrix} \cdot x\right)^{q/2-2}}_{\geq 0}\end{aligned} \tag{4.7}$$

So the l-th summand is convex, because its Hessian matrix is positive semidefinite.
The l-th summand of the objective can become zero only on the unit circle. When $f_{jkl}^{(q)}(x) = 0$, then we choose a neighborhood $U_\varepsilon(x) := \left\{ \tilde{x} \in \mathbb{R}^2 \quad | \quad ||\tilde{x} - x||_2 < \varepsilon \right\}$ for an arbitrary small $\varepsilon > 0$. Additionally we draw the tangent $L(x)$ on the unit circle through x. $L(x)$ borders an open half plane $P_l^+ \subset \mathbb{R}^2$ where $f_{jkl}(\tilde{x}) > 0$. Now the Hessian matrix of the l-th summand is positive semidefinite at all $\tilde{x} \in U_\varepsilon(x) \cap P_l^+$. Since the l-th summand is continuous in the closed unit disk, it is also convex there.
The objective function is convex, because it is a sum of convex functions.

□

Corollary 4.2. *If $q \in [2, \infty)$, then each local minimum of (4.6) is also a global minimum.*

Lemma 4.3. *If $q = 2$, then a global minimum of (4.6) lies on the unit circle, and with probability one no minimum lies in the interior of the disk.*

Proof. Let A and B be random matizes as in Definition 2.1 with any continuous distribution. The gradient of the objective function is

$$\frac{df_{jk}^{(q)}(x)}{dx} = -\begin{bmatrix}\sum_{l=1}^{N} \alpha_l(jk) \\ \sum_{l=1}^{N} \beta_l^{(jk)}\end{bmatrix}.$$

If a minimum of (4.6) lies in the interior of the unit disk, then this gradient must be zero. It is obvious, that this special case occurs only with probability zero. In such a case the objective function is independent of x, and we can choose an arbitrary point of the unit circle as a global minimum.

□

4.2 The case $q \in [1, 2)$

The objective function of (4.6) is concave, if $q < 2$. Since concave minimization is NP-hard in general, we cannot expect an efficient algorithm which yields the exact solution in this case.
First we want to apply the Branch-and-Bound-principle for our problem. A Branch-and-Bound methods finds either a global minimum or fails by exceeding the assigned space. It can require an exponential runtime with respect to the length of the input. Our Branch-and-Bound algorithm should decompose our feasible set into pieces and compute a lower bound of the values of the objective function for each piece. However, it is (in general) difficult to determine a lower bound for a piece of the feasible region which has a curved border. Therefore we extend the considered region not only to the unit disk, but also beyond. Hereby we must be cautious, because a term in parentheses in (4.6) could become negative outside the unit disk[1].

[1] These terms are exponentiated by $\frac{q}{2}$.

4 The one-dimensional minimization problems in the cases $\mathbb{F} = \mathbb{C}$, $p \neq 2$ and $q \in [1, \infty] \backslash \{2\}$

Definition 4.4. *Given the program* (4.6) *with* $q \in [1, 2)$. *Let* $I_2(j, k)$ *be as in* (4.3). *The low points of the program are*

$$y_m^{(jk)} := \frac{1}{\sqrt{\left(\alpha_m^{(jk)}\right)^2 + \left(\beta_m^{(jk)}\right)^2}} \cdot \begin{bmatrix} \alpha_m^{(jk)} \\ \beta_m^{(jk)} \end{bmatrix}$$

for all $m \in I_2(j, k)$.

We compute the set of low points as in Definition 4.4. If this set is empty, then the problem is trivial. Therefore let us assume that the set of low points is not empty. The tangent to the unit circle at the point $y_m^{(jk)}$ borders a half plane in which the term in parentheses of the m-th summand of $f_{jk}^{(q)}$ in (4.6) cannot become negative.
We can assume that we have at least two different low points[2]. Then the tangents to the unit circle at the low points form a (possibly unbounded) polygon in which no term in parentheses in $f_{jk}^{(q)}$ can become negative.
If the polygon which is formed by the tangents at the low points is not bounded, then we can insert an artificial low point and draw the tangent to the unit circle at this point such that we get a bounded polygon. Then we draw all the rays from the origin to all vertices of the polygon as well as the rays from the origin to all the low points (including the possibly inserted artificial low point). The polygon together with the rays yields our initial search nodes for the Branch-and-Bound-Method (see Figure 4.1). Note that each search node represents a triangle which has exactly one vertex on the unit circle. We compute the value of the objective function on each vertex of such a triangle. The minimum of these three values is obviously a lower bound for the values of the objective function over the whole triangle, because the objective function is concave. We do so for all the search nodes.
Now we insert the initial search nodes into a heap. The heap condition guarantees that a search node of the minimal lower bound is stored in the root of the heap. In each step the algorithm treats such a node and decomposes it into two child nodes by a straight line through the origin. Another area is cut off from exactly one of the two child triangles in order to the recovery of the status that each triangle has one vertex on the unit circle (see Figure 4.2). Then the father node is deleted, lower bounds of the child node are computed, and the heap is updated. The algorithm stops in three cases:

(i) The lower bound of the considered triangle is achieved on its vertex which lies on the unit circle.

(ii) The angle of the considered triangle on the origin is smaller than a predetermined tolerance parameter.

(iii) The number of the search nodes exceeds a predetermined amount.

Table 4.1: The cases for the termination of the Brach-and-Bound algorithm.

The perfect case is (i), because the algorithm gives an exact solution. Stopping in case (ii) yields obviously an approximative solution. Case (iii) should be avoided if possible.

Algorithm 4.5.

Input:	N, $q \in [1, 2)$, $\alpha_l, \beta_l, \gamma_l \quad (l = 1, 2, ..., N)$ *as in* (4.1), *a positive integer* $\hat{m}$ *giving the maximal size of the heap, a tolerance parameter* $\varepsilon > 0$.
Output:	*An error message or a point* x *as in* (4.6) *which is globally optimal up to a tolerance depending on* ε.
Data structures:	*Search nodes which store at least two vertices in the plane as well as a lower bound; a min-heap for*

[2]Otherwise the one-dimensional minimization problem was trivial.

such search nodes which stores additionally the current and the maximal number of search nodes.

1.) *Construct a min-heap for maximal $\hat{m}$ search nodes.*
Set the current number of search nodes to zero.
2.) *Compute the initial search nodes as described above.*
3.) *If the problem has fewer than two proper low points*
solve the problem immediately and stop.
4.) *For each initial search node ν do*
compute the values of the objective function at the three vertices of ν,
store a minimum of these three values as a lower bound of ν,
insert ν into the min-heap,
increment the number of search nodes by one.
5.) *Restore the heap condition.*
6.) *Repeat* {
Read the search node ν from the root of the heap.
If the lower bound of ν is attained on a vertex which lies on the unit circle // *case (i)*
output this vertex point as the solution and stop.
If the angle of ν on the origin is smaller than ε // *case (ii)*
output the vertex point of ν which lies on the unit circle as the solution and stop.
If the current number of search nodes is $\hat{m}$, // *case (iii)*
display an error message and stop.
Decompose ν into two search nodes ν_1 and ν_2 as described above.
For $k := 1$ to 2 do
compute the values of the objective function at the three vertices of ν_k,
store a minimum of these three values as a lower bound of ν_k.
Remove ν from the heap and insert ν_1, ν_2 into the heap.
Increment the current number of search nodes by one.
Restore the heap condition.
}

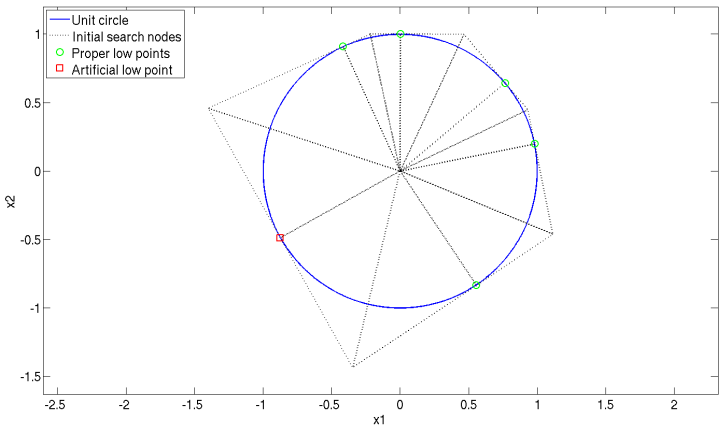

Figure 4.1: The initial search nodes for the Branch-and-Bound Method.

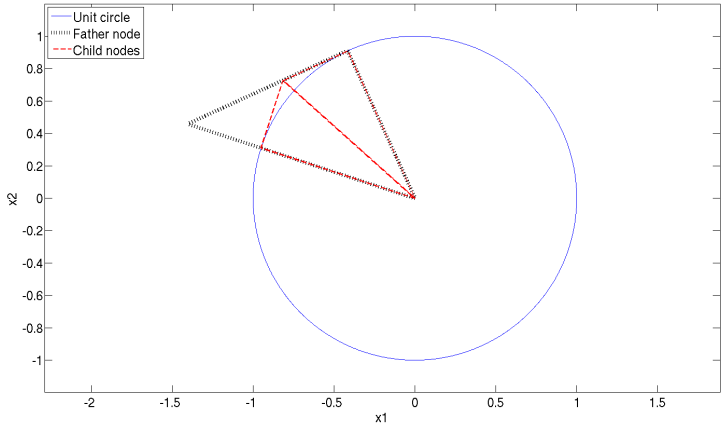

Figure 4.2: The decomposition of a search node.

Lemma 4.6. *The maximal number of sweeps through the loop in step 6 of Algorithm 4.5 is at most* $\frac{4\pi}{\varepsilon}$.

Proof. Obviouly, the maximal number of search nodes which exist during the last sweep is at most $\left\lceil \frac{2\pi}{\varepsilon} \right\rceil$. When the algorithm reaches the loop, then at least three initial search nodes exist, and each sweep in step 6 of Algorithm 4.5, except the last one, increments the number of search nodes by one. Thus the number of executions of this loop can not be greater than $\frac{4\pi}{\varepsilon}$.

□

Theorem 4.7. *The entire runtime of Algorithm 4.5 is in* $\mathcal{O}(N \cdot \log N) + \mathcal{O}\left(\frac{\log(1/\varepsilon)}{\varepsilon}\right)$.

Proof. Sorting the low points with respect to their angles in step 2 and restoring the heap condition in step 5 of Algorithm 4.5 is possible in $\mathcal{O}(N \cdot \log N)$. The rest of step 1 till 4 is done in $\mathcal{O}(N)$. A sweep in step 6 requires $\mathcal{O}(\log(1/\varepsilon))$ because of the restoration of the heap condition, and the algorithm needs only $\mathcal{O}(1/\varepsilon)$ sweeps according to Lemma 4.6. □

Remark 4.8. *The summand* $\mathcal{O}\left(\frac{\log(1/\varepsilon)}{\varepsilon}\right)$ *in Theorem 4.7 can be exponential with respect to the input length, because the input of the tolerance parameter* ε *requires only a logarithmic number of bits.*

Next we apply the SLA and the solver KNITRO in order to solve our program (4.6) faster. At least KNITRO is efficient, but both methods yield only a stationary point. The computation of a step of the SLA as in (3.1) is very easy in our case.

Lemma 4.9. *Given a minimization problem over the disk* P *of radius* $R > 0$ *around the origin. Let the objective function* $f : P \subset \mathbb{R}^2 \to \mathbb{R}$ *be concave and differentiable. Assume* $\nabla f\left(x^{(k)}\right) \neq 0$ *at a point* $x^{(k)} \in P$. *Then*

$$x^{(k+1)} = -R \cdot \frac{\nabla f\left(x^{(k)}\right)}{\left|\left|\nabla f\left(x^{(k)}\right)\right|\right|_2} \tag{4.8}$$

is the unique solution of (3.1).

4 The one-dimensional minimization problems in the cases $\mathbb{F} = \mathbb{C}$, $p \neq 2$ *and* $q \in [1, \infty] \backslash \{2\}$

Proof. It holds for the minimal value in (3.1):

$$\begin{aligned}
\min_{x \in \mathcal{E}(P)} \nabla f\left(x^{(k)}\right) \cdot \left(x - x^{(k)}\right) &= -\nabla f\left(x^{(k)}\right) \cdot x^{(k)} \\
&\quad + \min_{x \in \mathcal{E}(P)} \nabla f\left(x^{(k)}\right) \cdot x \\
&= -\nabla f\left(x^{(k)}\right) \cdot x^{(k)} \\
&\quad + \min_{x^T x = R} \nabla f\left(x^{(k)}\right) \cdot x \\
&= -\nabla f\left(x^{(k)}\right) \cdot x^{(k)} \\
&\quad + \nabla f\left(x^{(k)}\right) \cdot x^{(k+1)}
\end{aligned}$$

Now the statement follows by the Cauchy-Schwarz inequality.

□

Fung and Mangasarian consider the real multidimensional ℓ_q minimization problem for $q < 1$ in [17]. They suggest starting the SLA iteration with the solution of the ℓ_1 minimization problem which is linear in the real case. However, we have not a proper ℓ_q minimization problem in the real plane, but a relaxed optimization problem over the circle. The case $q = 1$ is not linear in (4.6). Therefore we use the solution of the ℓ_2 minimization problem as start point, because the ℓ_2 optimization is linear in our case. Now the SLA can be adapted for (4.6) with $q \in [1, 2)$ in the following way:

Algorithm 4.10.

Input: N, $q \in [1, 2)$, $\alpha_l, \beta_l, \gamma_l \quad (l = 1, 2, ..., N)$ *as in* (4.1),
a positive number $R \leq 1$ *giving the radius of the disk,*
a positive integer $\hat{m}$ *giving the maximal number of iterations,*
a tolerance parameter $\varepsilon > 0$.

Output: *A stationary point* $x \in \mathbb{R}^2$ *with* $\mu = (x_1 + i \cdot x_2) / R$ *as in* (2.4).

1.) $I_2 := \emptyset$.

2.) *For* $l := 1$ *to* N *do*
if $\left(\alpha_l^2 + \beta_l^2 > \varepsilon^2\right)$
$I_2 := I_2 \cup \{l\}$;

3.) $y^{(0)} := \sum_{l \in I_2} \begin{bmatrix} \alpha_l & \beta_l \end{bmatrix}^T, \qquad \hat{n} := \left|\left| y^{(0)} \right|\right|_2$.

4.) *If* $(\hat{n} > \varepsilon)$
then $x^{(0)} := R \cdot \dfrac{y^{(0)}}{\hat{n}}$; // *start with* R *times*
// *the* ℓ_2 *solution*
else $x^{(0)} := \begin{bmatrix} R & 0 \end{bmatrix}^T$. // *an arbitrary point*
// *with radius* R

// *(see the proof of*
// *Lemma 4.3.)*

5.) *For* $k := 1$ *to* $\hat{m}$ *do*

$$y^{(k)} := \sum_{l \in I_2} \frac{\begin{bmatrix} \alpha_l & \beta_l \end{bmatrix}^T}{\left(\gamma_l - \begin{bmatrix} \alpha_l & \beta_l \end{bmatrix} \cdot x^{(k-1)}\right)^{1-q/2}},$$

// $y^{(k)} = -\nabla f\left(x^{(k-1)}\right)$

$\hat{n} := \left|\left| y^{(k)} \right|\right|_2$.
If $(\hat{n} < \varepsilon)$
then $x^{(k)} := x^{(k-1)}$, *break;*

4 The one-dimensional minimization problems in the cases $\mathbb{F} = \mathbb{C}$, $p \neq 2$ *and* $q \in [1, \infty] \backslash \{2\}$

$$\textit{else } x^{(k)} := R \cdot \frac{y^{(k)}}{\hat{n}}; \qquad \textit{// see (4.8)}$$

$$\textit{if } \left(\left(x^{(k)} - x^{(k-1)} \right)^T \cdot \left(x^{(k)} - x^{(k-1)} \right) < \varepsilon^2 \right)$$

then break.

6.) $\mu := \frac{1}{R} \cdot \left(x_1^{(k)} + i \cdot x_2^{(k)} \right)$.

Remark 4.11. *The gradient of the objective function can have poles on the unit circle if* $|a_{lj}| = |b_{lk}|$ *for at least one* $l \in \{1, 2, ..., N\}$. *Thus we can use a slightly smaller disk to avoid this difficulty. The radius* R *could be* $1 - \varepsilon$ *for a small* $\varepsilon > 0$.
On the other hand, poles can only occur with probability zero. Therefore we can first test whether a pair of corresponding coordinates has the same absolute value. Then we decrease the radius R *below one only is necessary.*

Example 4.12. *We set* $N := 1000$, $q := 1$ *and drew 20 pseudo-random numbers for the real and the imaginary parts of* a_l *and* b_l $(l = 1, 2, ..., N)$, *respectively. We solved the instances of the problem by Algorithm 4.10 which we have implemented in MATLAB and compiled. Thereby we set the tolerance parameter* $\varepsilon := 10^{-7}$ *and* $\hat{m} := 2000$. *We sent the instances of the problem also to the solver KNITRO on the NEOS server. We verified by the Branch-and-Bound method which method yielded a global minimum.*

	SLA		*KNITRO*	
no.	*Did it yield a global minimum?*	*required runtime in milliseconds*	*Did it yield a global minimum?*	*required runtime in milliseconds*
1	✓	*15*	✓	*4.39*
2	✓	*15*	✓	*4.79*
3	✓	*31*	✓	*5.58*
4	✓	*16*	✓	*7.42*
5	✓	*29*		*4.79*
6		*140*		*4.15*
7	✓	*16*		*5.19*
8	✓	*16*	✓	*32.53*
9	✓	*31*	✓	*8.14*
10	✓	*11*	✓	*6.84*
11	✓	*31*		*5.05*
12	✓	*31*	✓	*4.53*
13	✓	*15*		*3.16*
14	✓	*16*	✓	*2.27*
15	✓	*31*	✓	*11.05*
16	✓	*31*	✓	*3.47*
17		*31*		*5.67*
18	✓	*16*		*3.92*
19	✓	*32*	✓	*7.45*
20	✓	*15*	✓	*26.24*
sum:		*569*		*156.63*

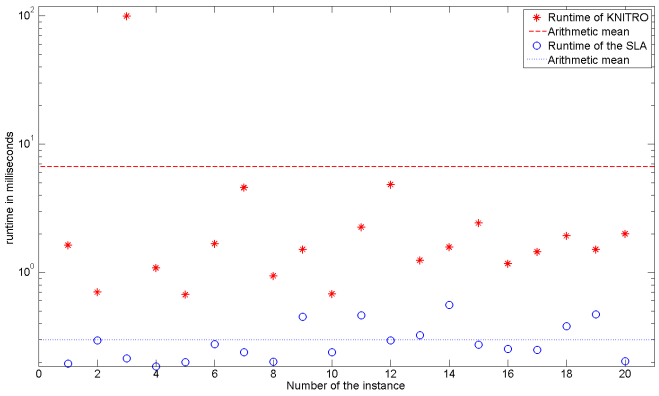

Figure 4.3: The runtimes of KNITRO and the SLA for Example 4.12.

The SLA yielded eighteen times a global minimum, KNITRO found only thirteen times a global minimum.

□

We programmed the Branch-and-Bound method in C++. We coded also the SLA in C++ in order to treat the data with the same program.

Example 4.13. *We set* $q := 1$ *and drew 50 sets of pseudo-random numbers for the real and imaginary parts of* a_l *and* b_l *for* $l = 1, 2, ..., N$ *and* $N \in \{100, 200, ..., 1000\}$. *The tolerance parameter* ε *was* 10^{-7}, *the maximal heap size was* 5000 *and the maximal number of iterations for the SLA was bounded to* 1000. *Never the maximal heap size nor the maximal number of iterations was attained. A solution of the SLA was considered being a global minimum if* $|\xi_{BaB} - \xi_{SLA}| < 10^{-6}$ *or* $|g(\xi_{BaB}) - g(\xi_{SLA})| < 10^{-6} \cdot g(\xi_{BaB})$. *Table 4.2 and Figure 4.4 display the results.*

□

We wanted also to test how the runtime depends on the tolerance parameter. If ε becomes very small, then the number of search nodes for the Branch-and-Bound-Method could explode and therefore the required runtime could also rise.

Example 4.14. *We set* $q := 1$ *and drew 50 sets of pseudo-random numbers for the real and imaginary parts of* a_l *and* b_l *for* $l = 1, 2, ..., N$ *with* $N = 500$. *for* $\varepsilon \in \left\{10^{-12}, 10^{-11}, ..., 10^{-3}\right\}$. *The maximal heap size was* 5000 *and the maximal number of iterations for the SLA was bounded to* 2000. *Never the maximal heap size nor the maximal number of iterations was attained. A solution of the SLA was considered being a global minimum if* $|\xi_{BaB} - \xi_{SLA}| < 10^{-6}$ *or* $|g(\xi_{BaB}) - g(\xi_{SLA})| < 10 \cdot \varepsilon \cdot g(\xi_{BaB})$. *Table 4.3 and Figure 4.5 display the results.*
There was no significant increase of the runtime of the Branch-and-Bound method for decreasing ε *in this example.*

□

N	Average runtime in milliseconds Branch and Bound	SLA	How often did the SLA find a global minimum?
100	6.86	0.32	16
200	18.12	2.78	50
300	35.92	5.56	50
400	63.62	6.26	39
500	95.52	9.62	50
600	135.76	16.80	36
700	192.90	24.26	50
800	248.90	22.56	50
900	365.32	40.28	48
1000	342.28	33.02	47

Table 4.2: The results of Example 4.13.

ε	Average runtime in milliseconds Branch and Bound	SLA	How often found the SLA a global minimum?
10^{-12}	102.14	19.14	50
10^{-11}	98.90	17.36	42
10^{-10}	98.24	16.36	50
10^{-9}	93.98	12.22	50
10^{-8}	95.46	15.10	40
10^{-7}	98.02	11.94	47
10^{-6}	98.44	9.26	43
10^{-5}	102.92	6.98	41
10^{-4}	90.90	4.54	50
10^{-3}	85.68	2.86	50

Table 4.3: The results of Example 4.14.

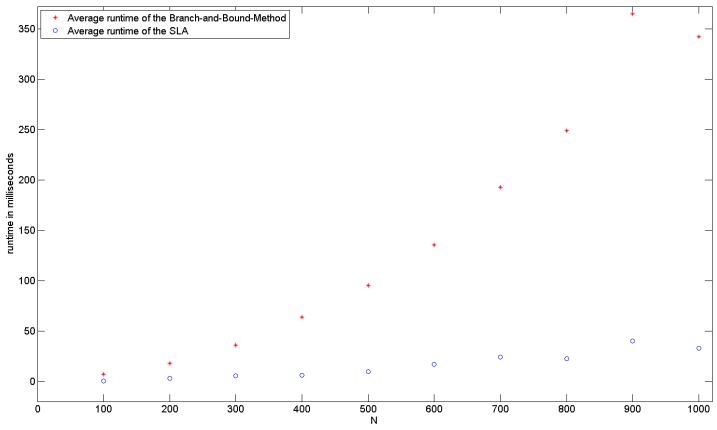

Figure 4.4: The average runtimes of the Branch-and-Bound-Method and the SLA for Example 4.13.

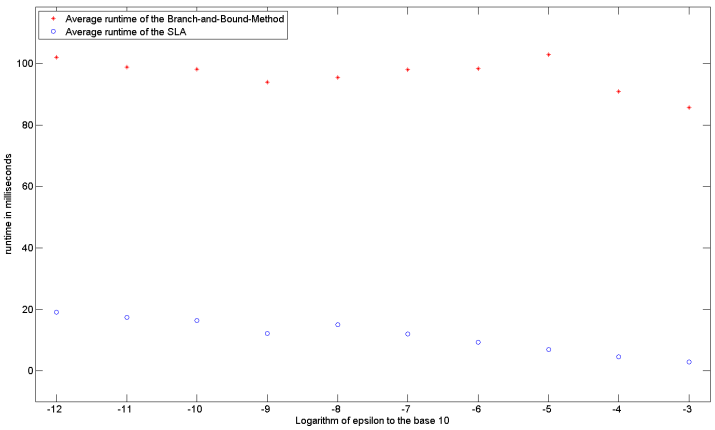

Figure 4.5: The average runtimes of the Branch-and-Bound-Method and the SLA for xample 4.14.

In practical tests the runtime of the Branch-and-Bound method depended hardly on the tolerance parameter ε, although this is the most crucial dependence in the worst case according to Theorem 4.7 and Remark 4.8. In Example 4.13 the runtime rose faster as predicted in Theorem 4.7. This is because a problem with fewer summands will possess fewer stationary points on average. Therefore we can expect that the Branch-and-Bound method will faster find an optimal search node or its ancestors, respectively.

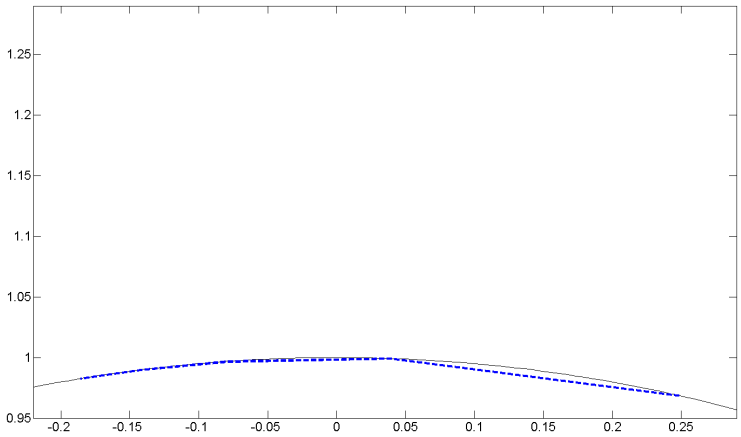

Figure 4.6: The SLA iteration: The solid curve is a section of the unit circle, and an SLA iteration with $\hat{m} = 5$ is dashed.

Finally, we investigate the solution of the problems (2.4) by Weiszfelds algorithm for the ESFL with the hyperbolic approximation from Rosen and Xue (3.3). However, a global minimum need not lie on the unit circle. It is possible to add a penalty function [25], [35] in order to force the solution to the unit circle. But we cannot expect to obtain a global minimum with respect to the unit circle in this way, because we loose the convexity of our problem. Instead of the solution of our problems with several parameters for the penalty function, we can exploit the geometric structure of the circle. Past each iteration in the sense of Weiszfelds algorithm we project the new iteration point to the circle in the following way:

4 The one-dimensional minimization problems in the cases $\mathbb{F} = \mathbb{C}$, $p \neq 2$ and $q \in [1, \infty] \backslash \{2\}$

$$\pi(x) := \frac{x}{||x||_2}, \qquad x \neq 0.$$

Now we can present our algorithm based on [48] and [39]:

Algorithm 4.15.
Input: N, a_l, b_l, $(l = 1, 2, ..., N)$,
a natural number $\hat{m}$ *giving the maximal number of iterations,*
a tolerance parameter $\varepsilon > 0$
an approximation parameter $\epsilon > 0$.
Output: *A locally minimizing* μ *as in* (2.4) *for* $q = 1$.
1.) $I_1 := \emptyset$.
2.) *For* $l := 1$ *to* N *do*
if $\left(\left(|a_l|^2 > \varepsilon^2\right) \text{ and } \left(|b_l|^2 > \varepsilon^2\right)\right)$
$I_1 := I_1 \cup \{l\}$;
$p_{1l} := \Re \frac{b_l}{a_l}$;
$p_{2l} := \Im \frac{b_l}{a_l}$;
3.) $y^{(0)} := a^* b, \qquad \hat{n} := \left|\left|y^{(0)}\right|\right|_2$.
If $(\hat{n} > \varepsilon)$
then $x^{(0)} := \frac{y^{(0)}}{\hat{n}}$; // *start with the* ℓ_2 *solution*
else $x^{(0)} := \begin{bmatrix} 1 & 0 \end{bmatrix}^T$. // *an arbitrary point on the circle.*
4.) *For* $k := 1$ *to* $\hat{m}$ *do*
$y^{(k)} := \sum_{l \in I_1} \frac{|a_l| \cdot p_{\cdot l}}{\sqrt{\left(y^{(k-1)} - p_{\cdot l}\right)^T \left(y^{(k-1)} - p_{\cdot l}\right) + \epsilon}}$;
$\hat{n} := \left|\left|y^{(k)}\right|\right|_2$.
If $(\hat{n} < \varepsilon)$
then $x^{(k)} := x^{(k-1)}$, *break;*
else $x^{(k)} := \frac{y^{(k)}}{\hat{n}}$;
if $\left(\left(x^{(k)} - x^{(k-1)}\right)^T \cdot \left(x^{(k)} - x^{(k-1)}\right) < \varepsilon^2\right)$
then break.
4.) $\mu := x_1^{(k)} + i \cdot x_2^{(k)}$.

We programmed Algorithm 4.15 in order to compare it with our Branch-and-Bound-Method as well as the SLA.

Example 4.16. *We set* $q := 1$ *and drew 50 sets of pseudo-random numbers for the real and imaginary parts of* a_l *and* b_l *for* $l = 1, 2, ..., N$ *and* $N \in \{100, 200, ..., 1000\}$. *The tolerance parameter* ε *was* 10^{-7}, *the approximation parameter* ϵ *was* 10^{-9}, *the maximal heap size was* 5000 *and the maximal number of iterations for Algorithm 4.15 and the SLA was bounded to* 2000. *Never the maximal heap size nor the maximal number of iterations was anytime attained. A solution of the SLA was considered to be a global minimum if* $|\xi_{BaB} - \xi_{SLA}| < 10^{-6}$ *or* $|g(\xi_{BaB}) - g(\xi_{SLA})| < 10^{-6} \cdot g(\xi_{BaB})$, *analogously for Algorithm 4.15. The average runtimes are displayed in Table 4.4 and Figure 4.7.*

□

N	Average runtime of			How often found	
	B. a. B.	Algor. 4.15	SLA	Algor. 4.15	SLA
		in milliseconds		a global minimum?	
100	4.98	0.32	1.26	29	29
200	13.72	0.96	2.82	50	50
300	25.96	0.32	6.54	50	50
400	52.70	3.10	9.50	50	50
500	79.06	2.18	8.14	50	50
600	109.26	3.82	10.04	50	50
700	148.50	7.08	14.72	43	43
800	193.18	5.98	15.54	50	50
900	241.60	6.82	21.62	35	35
1000	292.48	8.16	18.74	46	46

Table 4.4: The results of Example 4.16.

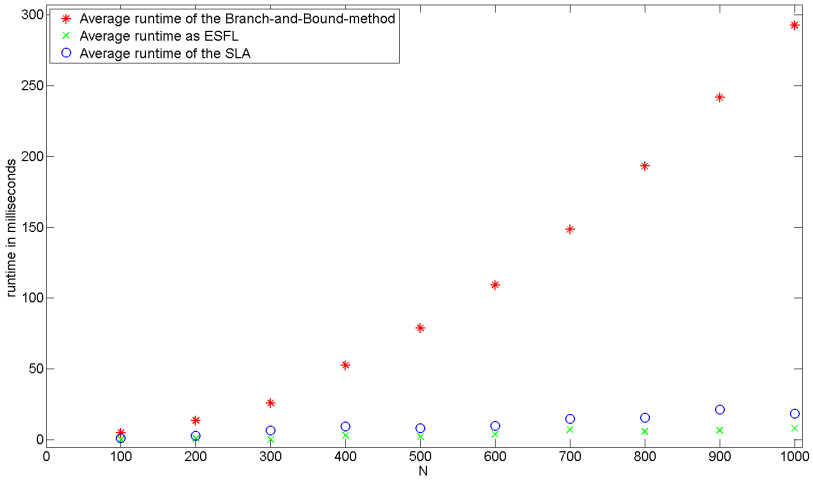

Figure 4.7: The average runtimes of Algorithm 4.15 and the SLA.

4.3 The case $q \in [4, \infty)$

In the next section we will investigate the case $q \in (2, 4)$. The treatment of that case will be a refinement of our approach in this section. Therefore we first present the case $q \in [4, \infty)$. In section 4.5 we will see that there exists a much better approach for the special case $q = 4$. But for $q \in (4, \infty)$, we are not able to give an efficient algorithm which always yields a global minimum.
Let us consider the following modification of the program (4.6):

$$\begin{aligned} \min \hat{f}_{jk}^{(q)}(x) &= \sum_{l \in I_1(j,k)} \left(\gamma_l^{(jk)} - \begin{bmatrix} \alpha_l^{(jk)} & \beta_l^{(jk)} \end{bmatrix} \cdot x \right)^{q/2} + \frac{\lambda^{(jk)}}{2} \cdot \left(1 - x^T x\right) \\ \text{s.t. } & x^T x \leq 1, \quad x \in \mathbb{R}^2 \end{aligned} \tag{4.9}$$

with $\alpha_l^{(jk)}$, $\beta_l^{(jk)}$ and $\gamma_l^{(jk)}$, $l \in \{1, 2, ..., N\}$, as in (4.1) and $\lambda^{(jk)} \in \mathbb{R}$.

Obviously, the additional summand $\frac{\lambda^{(jk)}}{2} \cdot \left(1 - x^T x\right)$ vanishes on the unit circle. Therefore the

objective function of the program (4.9) has the same value as the objective function of the program (4.6) on each point of the unit circle.

Lemma 4.17. *Let $f_{jk}^{(q)}(x)$ be the objective function as in program (4.6) and $U \subseteq \mathbb{R}^2$ a closed bounded convex set. If $q \in (4, \infty)$ and $\lambda^{(jk)}$ is greater than or equal to the largest eigenvalue of the Hessian matrix $\dfrac{d^2 f_{jk}^{(q)}(x)}{dx^2}$ for all $x \in U$, then the objective function of the program (4.9) is concave over U.*

Proof. The Hessian matrix of the objective function is

$$\frac{d^2 \hat{f}_{jk}^{(q)}\left(x, \lambda^{(jk)}\right)}{dx^2} = \frac{d^2 f_{jk}^{(4)}(x)}{dx^2} - \lambda^{(jk)} \cdot I_2,$$

and this matrix is negative semidefinite for all $x \in U$, because $\lambda^{(jk)}$ is greater than or equal to the largest eigenvalue of the first summand if $x \in U$.

□

Lemma 4.18. *Let $f_{jk}^{(q)}(x)$ be the objective function as in program (4.6) as well as $\alpha_l^{(jk)}$, $\beta_l^{(jk)}$ and $\gamma_l^{(jk)}$ for $l \in \{1, 2, ..., N\}$ as in (4.1) and $R \in (0, 1]$. If we set*

$$\begin{aligned} \lambda^{(jk)} &= \lambda^{(jk)}(R) \\ &:= \frac{q}{2} \cdot \left(\frac{q}{2} - 1\right) \cdot \sum_{l \in I_1(j,k)} \left((1+R)\gamma_l^{(jk)}\right)^{q/2-2} \cdot \left(\left(\alpha_l^{(jk)}\right)^2 + \left(\beta_l^{(jk)}\right)^2\right), \end{aligned} \tag{4.10}$$

then $\lambda^{(jk)}$ is greater than or equal to the largest eigenvalue of the Hessian matrix $\dfrac{d^2 f_{jk}^{(q)}(x)}{dx^2}$ for all x within the closed disk with radius R around the origin.

Proof. According to (4.7) the Hessian matrix of $f_{jk}^{(q)}(x)$ is

$$\begin{aligned} \frac{d^2 f_{jk}^{(q)}(x)}{dx^2} &= \frac{q}{2} \cdot \left(\frac{q}{2} - 1\right) \cdot \sum_{l \in I_1(j,k)} \left(\gamma_l^{(jk)} - \begin{bmatrix} \alpha_l^{(jk)} & \beta_l^{(jk)} \end{bmatrix} \cdot x\right)^{q/2-2} \cdot \begin{bmatrix} \alpha_l^{(jk)} \\ \beta_l^{(jk)} \end{bmatrix} \cdot \begin{bmatrix} \alpha_l^{(jk)} & \beta_l^{(jk)} \end{bmatrix} \\ &=: \begin{bmatrix} h_{11}(x) & h_{12}(x) \\ h_{21}(x) & h_{22}(x) \end{bmatrix}. \end{aligned}$$

The largest eigenvalue of this matrix is

$$\begin{aligned} \lambda_1(x) &= \frac{h_{11}(x) + h_{22}(x)}{2} + \sqrt{\frac{\left(h_{11}(x) + h_{22}(x)\right)^2}{4} + h_{12}(x) \cdot h_{21}(x) - h_{11}(x) \cdot h_{22}(x)} \\ &\leq h_{11}(x) + h_{22}(x) \\ &= \frac{q}{2} \cdot \left(\frac{q}{2} - 1\right) \cdot \sum_{l \in I_1(j,k)} \left(\gamma_l^{(jk)} - \begin{bmatrix} \alpha_l^{(jk)} & \beta_l^{(jk)} \end{bmatrix} \cdot x\right)^{q/2-2} \cdot \left(\left(\alpha_l^{(jk)}\right)^2 + \left(\beta_l^{(jk)}\right)^2\right) \\ &\leq \frac{q}{2} \cdot \left(\frac{q}{2} - 1\right) \cdot \sum_{l \in I_1(j,k)} \left((1+R)\gamma_l^{(jk)}\right)^{q/2-2} \cdot \left(\left(\alpha_l^{(jk)}\right)^2 + \left(\beta_l^{(jk)}\right)^2\right) \\ &\qquad \forall x \quad \text{with} \quad x^T x \leq 1. \end{aligned} \tag{4.11}$$

□

If we set $\lambda^{(jk)} = \lambda^{(jk)}(1)$ as in (4.10), then the objective function of (4.9) is concave over the unit disk according to Lemma 4.17 and 4.18. Our modification has the advantage that the global minimum lies on the unit circle and has there the same value as the original program (4.6) according to section 3.5. The modified program is still hard to solve. But the Successive Linearization Algorithm ([17], [32]) as well as the solver KNITRO finds always a stationary point which is also a global minimum in many cases. Here is the slightly modified Successive Linearization Algorithm from Mangasarian for the case $q \in (4, \infty)$.

Algorithm 4.19.

Input: N, $q \in (4, \infty)$, $\alpha_l, \beta_l, \gamma_l \quad (l = 1, 2, ..., N)$ *as in* (4.1), *a nonnegative real number* $\lambda := \lambda^{(jk)}(1)$ *as in* (4.10), *a positive integer* $\hat{m}$ *giving the maximal number of iterations, a tolerance parameter* $\varepsilon > 0$.

Output: *A stationary point* $x \in \mathbb{R}^2$ *with* $\mu = (x_1 + i \cdot x_2)$ *as in* (2.4).

1.) $y^{(0)} := \sum_{l=1}^{N} \begin{bmatrix} \alpha_l & \beta_l \end{bmatrix}^T, \qquad \hat{n} := \left|\left| y^{(0)} \right|\right|_2 .$

2.) *If* $(\hat{n} > \varepsilon)$

then $x^{(0)} := \dfrac{y^{(0)}}{\hat{n}}$; // *start with the* ℓ_2 *solution*

else $x^{(0)} := \begin{bmatrix} 1 & 0 \end{bmatrix}^T$. // *an arbitrary point* // *with radius* 1

3.) *For* $k := 1$ *to* $\hat{m}$ *do*

$$y^{(k)} := \lambda \cdot x^{(k-1)} + \sum_{l=1}^{N} \left(\gamma_l - \begin{bmatrix} \alpha_l & \beta_l \end{bmatrix} \cdot x^{(k-1)}\right)^{q/2-1} \cdot \begin{bmatrix} \alpha_l \\ \beta_l \end{bmatrix},$$

$\hat{n} := \left|\left| y^{(k)} \right|\right|_2 .$

If $(\hat{n} < \varepsilon)$

then $x^{(k)} := x^{(k-1)}$, *break;*

else $x^{(k)} := \dfrac{y^{(k)}}{\hat{n}}$;

if $\left(\left(x^{(k)} - x^{(k-1)}\right)^T \cdot \left(x^{(k)} - x^{(k-1)}\right) < \varepsilon^2\right)$

then break.

4.) $\mu := \left(x_1^{(k)} + i \cdot x_2^{(k)}\right).$

In the third step of the algorithm we used the convention $0^{q/2-1} := 0$ for all $q \in (4, \infty)$.
Furthermore, we can use $\lambda^{(jk)}(R)$ as in Lemma 4.18 in order to make our modified objective function as in (4.9) concave over a disk with an arbitrary radius R around the origin. This modification works also for our Branch-and-Bound-Method which we have introduced in section 4.2. If we set R to the maximal Euclidean norm over the vertices of the triangles which correspond to the initial search nodes of our Branch-and-Bound-Method, then our modified objective function with $\lambda^{(jk)}(R)$ will be concave over all triangles which are represented by the initial search nodes and thus for all the triangles which correspond to the search nodes which are produced by the Branch-and-Bound-Method (see Figure 4.1).

Example 4.20. *We set* $q := 5$ *and drew 50 sets of pseudo-random numbers for the real and imaginary parts of* a_l *and* b_l *for* $l = 1, 2, ..., N$ *and* $N \in \{100, 200, ..., 1000\}$. *The tolerance parameter* ε *was* 10^{-7}, *the maximal heap size was* 5000 *and the maximal number of iterations for the SLA was bounded to* 2000. *Neither the maximal heap size nor the maximal number of iterations was attained. A solution of the SLA was considered being a global minimum if* $|\xi_{BaB} - \xi_{SLA}| < 10^{-6}$ *or* $|g(\xi_{BaB}) - g(\xi_{SLA})| < 10^{-6} \cdot g(\xi_{BaB})$. *Table 4.5 and Figure 4.8 display the results.*

□

Example 4.21. *We set* $q := 10$ *and drew 50 sets of pseudo-random numbers for the real and imaginary parts of* a_l *and* b_l *for* $l = 1, 2, ..., N$ *and* $N \in \{100, 200, ..., 1000\}$. *The tolerance parameter* ε *was*

N	Average runtime in milliseconds Branch and Bound	SLA	How often did the SLA find a global minimum?
100	35.88	3.44	50
200	74.32	6.50	50
300	118.86	6.54	50
400	160.10	11.80	50
500	208.40	19.98	50
600	260.94	32.36	40
700	344.46	36.36	45
800	451.16	47.70	50
900	532.28	43.12	44
1000	585.22	55.88	48

Table 4.5: The results for Example 4.20.

N	Average runtime in milliseconds Branch and Bound	SLA	How often did the SLA find a global minimum?
100	14.34	6.24	50
200	40.60	11.84	50
300	90.48	19.34	33
400	146.02	27.76	39
500	203.90	43.84	50
600	269.64	76.98	34
700	333.48	81.18	39
800	412.44	114.52	45
900	480.08	149.50	40
1000	573.46	195.62	31

Table 4.6: The results for Example 4.21.

10^{-7} *and the maximal heap size was* 5000. *The SLA became slower for* $q = 10$. *Therefore we set the maximal number of iterations for the SLA to* 2000. *Neither the maximal heap size nor the maximal number of iterations was attained. A solution of the SLA was considered being a global minimum if* $|\xi_{BaB} - \xi_{SLA}| < 10^{-6}$ *or* $|g(\xi_{BaB}) - g(\xi_{SLA})| < 10^{-6} \cdot g(\xi_{BaB})$. *Table 4.6 and Figure 4.9 display the results.*

□

4.4 The case $q \in (2, 4)$

This case is quite similar to the case $q \in (4, \infty)$, but there is an essential difference. If $q \in (2, 4)$ and $|a_{lj}| = |b_{lk}|$ for at least one $l \in \{1, 2, ..., N\}$, then the second derivative of the objective function has at least one pole on the unit circle. Thus we can use a slightly smaller disk to avoid this difficulty. On the other hand poles can only occur with probability zero. Therefore we can first test whether a pair of corresponding coordinates has the same absolute value. Then we decrease the radius R below one only if necessary.

Lemma 4.22. *Let* $f_{jk}^{(q)}(x)$ *be the objective function as in the program* (4.6) *as well as* $\alpha_l^{(jk)}$, $\beta_l^{(jk)}$ *and* $\gamma_l^{(jk)}$, $l \in \{1, 2, ..., N\}$, *as in* (4.1). *Let* $R \in (0, 1]$. *If we set*

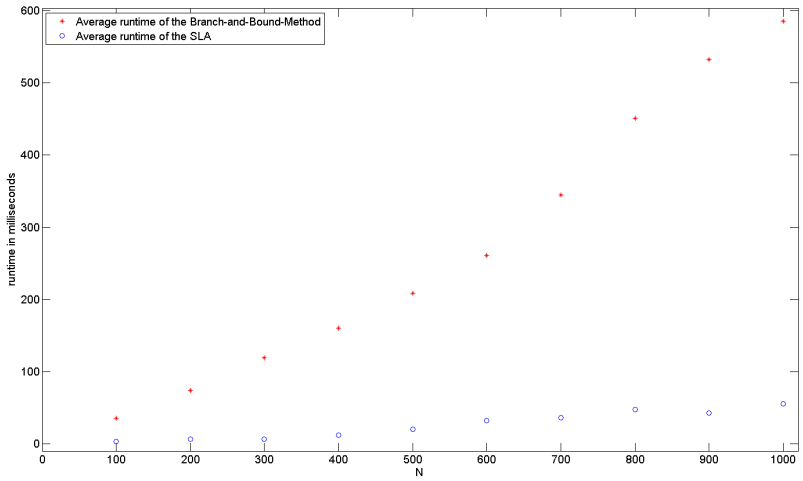

Figure 4.8: The average runtimes of the Branch-and-Bound-Method and the SLA in Example 4.20.

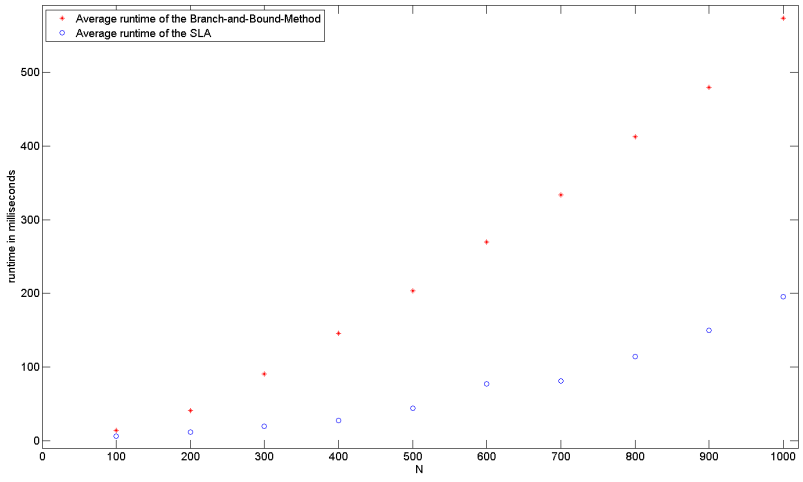

Figure 4.9: The average runtimes of the Branch-and-Bound-Method and the SLA in Example 4.21.

4 The one-dimensional minimization problems in the cases $\mathbb{F} = \mathbb{C}$, $p \neq 2$ and $q \in [1, \infty] \backslash \{2\}$

$$\begin{aligned}\lambda^{(jk)} &= \lambda^{(jk)}(R) \\ &:= \frac{q}{2} \cdot \left(\frac{q}{2} - 1\right) \cdot \begin{cases} \sum\limits_{l \in I_1(j,k)} \left((1-R) \cdot \gamma_l^{(jk)}\right)^{q/2-2} \cdot \left(\left(\alpha_l^{(jk)}\right)^2 + \left(\beta_l^{(jk)}\right)^2\right), & \text{if } R < 1, \\ \sum\limits_{l \in I_1(j,k)} \left||a_{lj}| - |b_{lk}|\right|^{q/2-2} \cdot \left(\left(\alpha_l^{(jk)}\right)^2 + \left(\beta_l^{(jk)}\right)^2\right), & \text{otherwise,} \end{cases}\end{aligned} \tag{4.12}$$

then $\lambda^{(jk)}(R)$ is greater than or equal to the largest eigenvalue of the Hessian matrix $\dfrac{d^2 f_{jk}^{(q)}(x)}{dx^2}$ for all x within the closed disk around the origin with radius R.

Proof. According to (4.7), the Hessian matrix of $f_{jk}^{(q)}(x)$ is

$$\begin{aligned}\frac{d^2 f_{jk}^{(q)}(x)}{dx^2} &= \frac{q}{2} \cdot \left(\frac{q}{2} - 1\right) \cdot \sum_{l \in I_1(j,k)} \left(\gamma_l^{(jk)} - \begin{bmatrix} \alpha_l^{(jk)} & \beta_l^{(jk)} \end{bmatrix} \cdot x\right)^{q/2-2} \cdot \begin{bmatrix} \alpha_l^{(jk)} \\ \beta_l^{(jk)} \end{bmatrix} \cdot \begin{bmatrix} \alpha_l^{(jk)} & \beta_l^{(jk)} \end{bmatrix} \\ &=: \begin{bmatrix} h_{11}(x) & h_{12}(x) \\ h_{21}(x) & h_{22}(x) \end{bmatrix}.\end{aligned}$$

We have the bound $\lambda_1(x) \leq h_{11}(x) + h_{22}(x)$ for the largest eigenvalue of the Hessian matrix from (4.11). This means for $q \in (2,4)$ and $R < 1$

$$\begin{aligned}\lambda_1(x) &\leq \frac{q}{2} \cdot \left(\frac{q}{2} - 1\right) \cdot \sum_{l \in I_1(j,k)} \left(\gamma_l^{(jk)} - \begin{bmatrix} \alpha_l^{(jk)} & \beta_l^{(jk)} \end{bmatrix} \cdot x\right)^{q/2-2} \cdot \left(\left(\alpha_l^{(jk)}\right)^2 + \left(\beta_l^{(jk)}\right)^2\right) \\ &\leq \frac{q}{2} \cdot \left(\frac{q}{2} - 1\right) \cdot \sum_{l \in I_1(j,k)} \left((1-R) \cdot \gamma_l^{(jk)}\right)^{q/2-2} \cdot \left(\left(\alpha_l^{(jk)}\right)^2 + \left(\beta_l^{(jk)}\right)^2\right) \\ &\qquad \forall x \quad \text{with} \quad x^T x \leq R^2.\end{aligned}$$

For the largest eigenvalue of the Hessian matrix in the case $R = 1$, it holds

$$\begin{aligned}\lambda_1(x) &\leq \frac{q}{2} \cdot \left(\frac{q}{2} - 1\right) \cdot \sum_{l \in I_1(j,k)} \left(\gamma_l^{(jk)} - \begin{bmatrix} \alpha_l^{(jk)} & \beta_l^{(jk)} \end{bmatrix} \cdot x\right)^{q/2-2} \cdot \left(\left(\alpha_l^{(jk)}\right)^2 + \left(\beta_l^{(jk)}\right)^2\right) \\ &\leq \frac{q}{2} \cdot \left(\frac{q}{2} - 1\right) \cdot \sum_{l \in I_1(j,k)} \left||a_{lj}| - |b_{lk}|\right|^{q/2-2} \cdot \left(\left(\alpha_l^{(jk)}\right)^2 + \left(\beta_l^{(jk)}\right)^2\right) \\ &\qquad \forall x \quad \text{with} \quad x^T x \leq 1.\end{aligned}$$

□

Now we can solve the program

$$\begin{aligned}\min \hat{f}_{jk}^{(q)}(x, R) &= \sum_{l \in I_1(j,k)} \left(\gamma_l^{(jk)} - \begin{bmatrix} \alpha_l^{(jk)} & \beta_l^{(jk)} \end{bmatrix} \cdot x\right)^{q/2} + \frac{\lambda^{(jk)}(R)}{2} \cdot \left(R^2 - x^T x\right) \\ \text{s.t. } & x^T x \leq R^2, \quad x \in \mathbb{R}^2\end{aligned} \tag{4.13}$$

with $R \in (0, 1]$ and $\lambda^{(jk)}(R)$ as in (4.12). According to section 3.5 the objective function has the same value as the program 4.6 on the circle around the origin with radius R. Acoording to Lemma 4.17 and 4.22 the objective function of the program 4.13 is concave. We adopted the Successive Linearization Algorithm from Mangasarian also for this case.

Algorithm 4.23.

Input: N, $q \in (2, 4)$, $\alpha_l, \beta_l, \gamma_l \quad (l = 1, 2, ..., N)$ *as in* (4.1),
a radius $R \in (0, 1]$ *and a nonnegative number* $\lambda := \lambda^{(jk)}(R)$
as in (4.12),
a positive integer $\hat{m}$ *giving the maximal number of iterations,*
a tolerance parameter $\varepsilon > 0$.

Output: *A stationary point* $x \in \mathbb{R}^2$ *with* $\mu = (x_1 + i \cdot x_2)/R$ *as in* (2.4).

1.) $y^{(0)} := \sum_{l=1}^{N} \begin{bmatrix} \alpha_l & \beta_l \end{bmatrix}^T, \qquad \hat{n} := \left|\left| y^{(0)} \right|\right|_2 .$

2.) *If* $(\hat{n} > \varepsilon)$

then $x^{(0)} := R \cdot \dfrac{y^{(0)}}{\hat{n}}$; // *start with R times*
// *the* ℓ_2 *solution*

else $x^{(0)} := \begin{bmatrix} R & 0 \end{bmatrix}^T$. // *an arbitrary point*
// *with radius R*

3.) *For* $k := 1$ *to* $\hat{m}$ *do*

$$y^{(k)} := \lambda \cdot x^{(k-1)} + \sum_{l=1}^{N} \left(\gamma_l - \begin{bmatrix} \alpha_l & \beta_l \end{bmatrix} \cdot x^{(k-1)}\right)^{q/2-1} \cdot \begin{bmatrix} \alpha_l \\ \beta_l \end{bmatrix},$$

$$\hat{n} := \left|\left| y^{(k)} \right|\right|_2 .$$

If $(\hat{n} < \varepsilon)$
then $x^{(k)} := x^{(k-1)}$, *break;*
else $x^{(k)} := R \cdot \dfrac{y^{(k)}}{\hat{n}}$;
if $\left(\left(x^{(k)} - x^{(k-1)}\right)^T \cdot \left(x^{(k)} - x^{(k-1)}\right) < \varepsilon^2\right)$
then break.

4.) $\mu := \dfrac{1}{R} \cdot \left(x_1^{(k)} + i \cdot x_2^{(k)}\right).$

Obviously, we can use $\hat{f}_{jk}^{(q)}(x, R)$ as in (4.13) with the same radius R as in the SLA also for our Branch-and-Bound-Method. If $\varepsilon_l \approx 0$ for a $l \in I_1(j, k)$, then we must use a radius which is slightly smaller than one (see Figure 4.11).

Example 4.24. *We set* $q := 3$ *and drew 50 sets of pseudo-random numbers for the real and imaginary parts of* a_l *and* b_l *for* $l = 1, 2, ..., N$ *and* $N \in \{100, 200, ..., 1000\}$. *The tolerance parameter* ε *was* 10^{-7}. *According to our expierence, both the SLA and the Branch-and-Bound-Method became very slow for* $q \in (2, 4)$ *Therefore we set the maximal heap size to* 20000 *and the maximal number of iterations for the SLA to* 100000. *After these settings neither the maximal heap size nor the maximal number of iterations was anymore attained. A solution of the SLA was considered being a global minimum if* $|\xi_{BaB} - \xi_{SLA}| < 10^{-6}$ *or* $|g(\xi_{BaB}) - g(\xi_{SLA})| < 10^{-6} \cdot g(\xi_{BaB})$.

N	*Average runtime in milliseconds*		*How often did*
	Branch and Bound	*SLA*	*the SLA find a global minimum?*
100	*48.44*	*15.38*	*50*
200	*101.12*	*80.96*	*50*
300	*145.80*	*122.44*	*48*
400	*151.32*	*170.32*	*50*
500	*220.84*	*301.20*	*49*
600	*296.88*	*395.30*	*48*
700	*411.08*	*632.22*	*50*
800	*560.04*	*843.12*	*50*
900	*657.42*	*1138.10*	*48*
1000	*713.02*	*1020.96*	*50*

Table 4.7: The average runtimes of the Branch-and-Bound-Method and the SLA for Example 4.24.

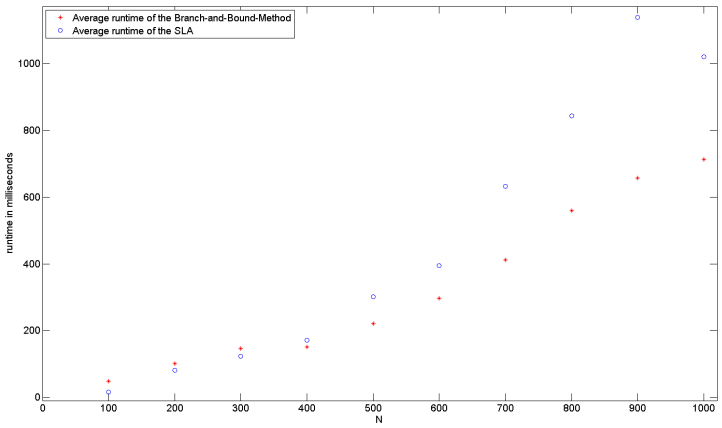

Figure 4.10: The average runtimes for Example 4.24.

□

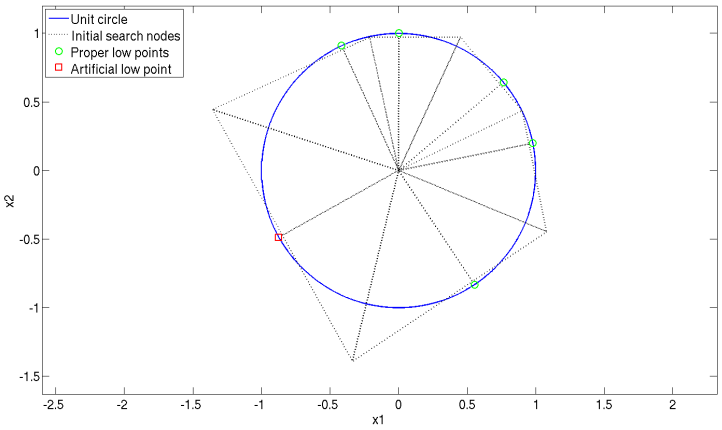

Figure 4.11: The initial search nodes for the Branch-and-Bound-Method if $R < 1$.

4.5 The special case $q = 4$

In section 4.3 ($q \in [4, \infty)$) we modified the objective function in order to make it concave. By that the local mimima lay on the border of the unit disk. But the global minimum could not be found efficiently.
However, we can find the global minimum very efficiently in the special case $q = 4$. The essential approach is the addition of a paraboloid again. But the resulting objective function stays convex this time and has a global minimum on the unit circle. Then the modified program can be solved exactly by a standard solver like KNITRO.
According to the proof of Lemma 4.1, the Hessian matrix is constant in the case $q = 4$. This matrix can be computed in $\mathcal{O}(N)$, and its eigenvalues can be computed in constant time, because this matrix is in $\mathbb{R}^{2\times 2}$. Let $\lambda_1 \geq \lambda_2 \geq 0$ be the eigenvalues of this matrix. Then we can modify the program 4.6 as follows:

$$\begin{aligned} \min \tilde{f}_{jk}^{(4)}(x) \;&=\; \sum_{l \in I_1(j,k)} \left(\gamma_l^{(jk)} - \begin{bmatrix} \alpha_l^{(jk)} & \beta_l^{(jk)} \end{bmatrix} \cdot x \right)^2 + \frac{\lambda_2}{2} \cdot \left(1 - x^T x\right) \\ \text{s.t. } x^T x \;&\leq\; 1, \quad x \in \mathbb{R}^2 \end{aligned} \tag{4.14}$$

with $\alpha_l^{(jk)}$, $\beta_l^{(jk)}$ and $\gamma_l^{(jk)}$, $l \in \{1, 2, ..., N\}$, as in (4.1). Obviously, the additional summand $\frac{\lambda^{(jk)}}{2} \cdot \left(1 - x^T x\right)$ vanishes on the unit circle. Therefore the objective function of the program (4.9) has the same value as the objective function of the program (4.6) on each point of the unit circle.

Lemma 4.25. *If $q = 4$ and λ_2 is the smallest eigenvalue of the matrix $\frac{d^2 f_{jk}^{(4)}(x)}{dx^2}$ with $f_{jkl}^{(4)}(x)$ as in program (4.6), then the program (4.14) is convex.*

Proof. The Hessian matrix of the objective function is

$$\frac{d^2 \tilde{f}_{jk}^{(4)}(x)}{dx^2} \;=\; \frac{d^2 f_{jk}^{(4)}(x)}{dx^2} \;-\; \lambda_2 \cdot I_2,$$

and this matrix is positive semidefinite, because λ_2 is the smallest eigenvalue of the first summand. □

Lemma 4.26. *If $q = 4$ and λ_2 is the smallest eigenvalue of the matrix $\frac{d^2 f_{jkl}^{(4)}(x)}{dx^2}$ with $f_{jk}^{(4)}(x)$ as in program (4.6), then a global minimum of (4.14) lies on the unit circle, and with probability one no minimum lies in the interior of the disk.*

Proof. Let A and B be random with some continuous distribution, $\lambda_1 \geq \lambda_2 \geq 0$ the eigenvalues of the Hessian matrix $\frac{d^2 f_{jk}^{(4)}(x)}{dx^2}$ and v_1 as well as v_2 the corresponding eigenvectors. Moreover, let $y = Qx$ be an orthogonal transformation of x into the v_1-v_2-coordinate system. The objective function of the program (4.14) has the form

$$\tilde{f}_{jk}^{(4)}\left(Q^{-1} y\right) \;=\; \frac{1}{2} \cdot (\lambda_1 - \lambda_2) \cdot y_1^2 + s \cdot y_1 + t \cdot y_2 + u$$

with some constants s, t and u depending only on $a_{\cdot j}$ and $b_{\cdot k}$. The gradient with respect of the v_1-v_2-coordinate system is

4 The one-dimensional minimization problems in the cases $\mathbb{F} = \mathbb{C}$, $p \neq 2$ *and* $q \in [1, \infty] \backslash \{2\}$

$$\frac{d\tilde{f}_{jk}^{(4)}\left(Q^{-1}y\right)}{dy} = \begin{bmatrix} (\lambda_1 - \lambda_2) \cdot y_1 + s \\ t \end{bmatrix}.$$

If a minimum lies in the interior of the disk, then this gradient must be zero. But the second component of the gradient can be zero only with probability zero. Now, let $\tilde{y} = Q\tilde{x}$ be such a global minimum in the interior of the disk. The objective function has the same value on each point of the straight line $L := \{\tilde{x} + dv_2 \quad | \quad d \in \mathbb{R}\}$. The line L intersects the unit circle in exactly two points, and each of them is a global minimum.

□

Example 4.27. *We set* $N := 12$ *and drew pseudo-random numbers for the real and imaginary parts of* a_l *and* b_l $(l = 1, 2, ..., 12)$. *The following table shows these randomly drawn numbers as well as the computed numbers* $\alpha_l, \beta_l, \gamma_l$ $(l = 1, 2, ..., 12)$ *as in* (4.1).

l	a_l	b_l	α_l	β_l	γ_l
1	$2.065 - 11.242i$	$-9.393 + 14.520i$	-365.26	-151.22	429.71
2	$-6.038 - 8.704i$	$6.636 - 12.277i$	133.58	263.78	306.98
3	$-4.042 + 1.931i$	$-9.281 - 13.444i$	23.107	144.52	286.94
4	$-14.745 + 2.481i$	$11.363 + 9.354i$	-288.68	-332.23	440.19
5	$-5.747 - 6.257i$	$-14.826 + 8.653i$	62.126	-284.99	366.86
6	$-6.673 + 7.203i$	$6.251 + 4.829i$	-13.859	-154.50	158.81
7	$12.763 - 10.924i$	$10.047 + 0.879i$	237.26	241.94	383.94
8	$9.274 - 8.142i$	$-10.771 + 10.689i$	-373.84	22.865	382.57
9	$14.695 + 10.880i$	$-3.386 + 5.177i$	13.137	225.83	372.58
10	$6.722 - 0.656i$	$5.819 - 9.655i$	90.898	-122.17	172.70
11	$6.528 - 10.835i$	$11.613 - 11.115i$	392.48	106.54	418.42
12	$6.981 - 0.866i$	$-0.866 + 8.280i$	-26.432	114.11	118.79

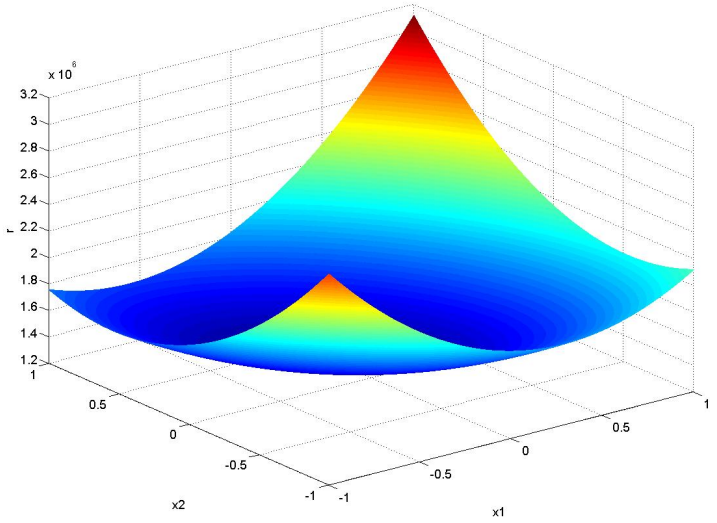

Figure 4.12: The graph of the function $f^{(4)}(x)$ as in (4.6) for Example 4.27. It is curved in each direction in the plane.

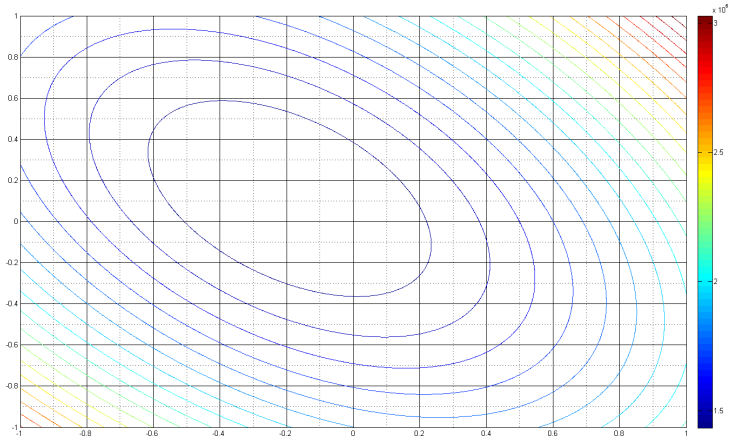

Figure 4.13: The niveau lines of the function $f^{(4)}(x)$ are ellipses.

The Hessian matrix of $f^{(4)}(x)$ *as in* (4.6) *is*

$$\frac{d^2 f^{(4)}(x)}{dx^2} = 2 \cdot \sum_{l=1}^{12} \begin{bmatrix} \alpha_l \\ \beta_l \end{bmatrix} \cdot \begin{bmatrix} \alpha_l & \beta_l \end{bmatrix} = \begin{bmatrix} 1.1968 \cdot 10^6 & 5.0734 \cdot 10^5 \\ 5.0734 \cdot 10^5 & 9.5631 \cdot 10^5 \end{bmatrix}$$

(See (4.7)*.) The spectral decomposition of the Hessian matrix is*

$$\begin{aligned} \frac{d^2 f^{(4)}(x)}{dx^2} &= 1.597957177610454 \cdot 10^6 \cdot v_1 v_1^T + 5.551633660185665 \cdot 10^5 \cdot v_2 v_2^T \\ \textit{with} \quad v_1 &= \begin{bmatrix} -0.78442 & -0.62023 \end{bmatrix}^T \\ \textit{and} \quad v_2 &= \begin{bmatrix} 0.62023 & -0.78442 \end{bmatrix}^T. \end{aligned} \tag{4.15}$$

We sent the following program to the solver KNITRO on the NEOS server.

$$\begin{aligned} \min \tilde{f}^{(4)}(x) &= \sum_{l=1}^{12} \left(\gamma_l - \begin{bmatrix} \alpha_l & \beta_l \end{bmatrix} \cdot x\right)^2 + 277581.68300928325 \cdot \left(1 - x^T x\right) \\ \text{s. t. } x^T x &\leq 1 \\ x &\in \mathbb{R}^2. \end{aligned}$$

with α_l, β_l *and* γ_l $(l = 1, 2, ..., 12)$ *as in the table at the beginning of this example. The solution is*

$$\begin{aligned} x &= \begin{bmatrix} -0.601645 & 0.798764 \end{bmatrix}^T. \\ \Longrightarrow \quad x^T x &= 1 \\ \textit{and} \quad \mu^{(4)} &= -0.601645 + 0.798764i. \\ \Longrightarrow \quad \xi^{(4)} &\approx 2.21635 \\ \textit{with} \quad \mu^{(4)} &= \exp\left(\xi^{(4)} \cdot i\right). \end{aligned}$$

Figure 4.12 shows the graph of the convex function $f^{(4)}(x)$ as in (4.6) and Figure 4.13 its niveau lines. Figure 4.14 shows the modified function $\tilde{f}^{(4)}(x)$ as in (4.14) and Figure 4.15 its niveau lines. For comparison we show the graph of the function $g^{(4)}(\xi)$ as in (2.4) and in (4.2) in the interval $[0, 2\pi]$ in Figure 4.16.

□

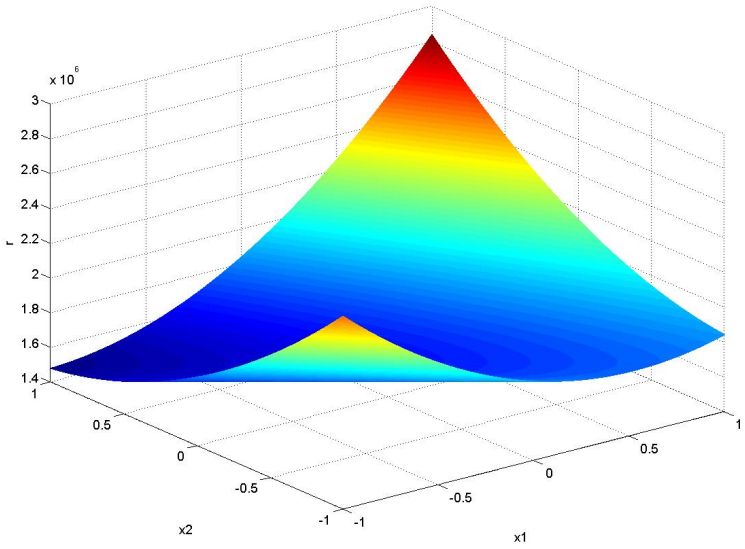

Figure 4.14: The graph of the modified function $\tilde{f}^{(4)}(x)$ as in (4.14) for Example 4.27. The graph of the function $\tilde{f}^{(4)}(x)$ is not curved any more in v_2-direction (with v_2 as in (4.15)).

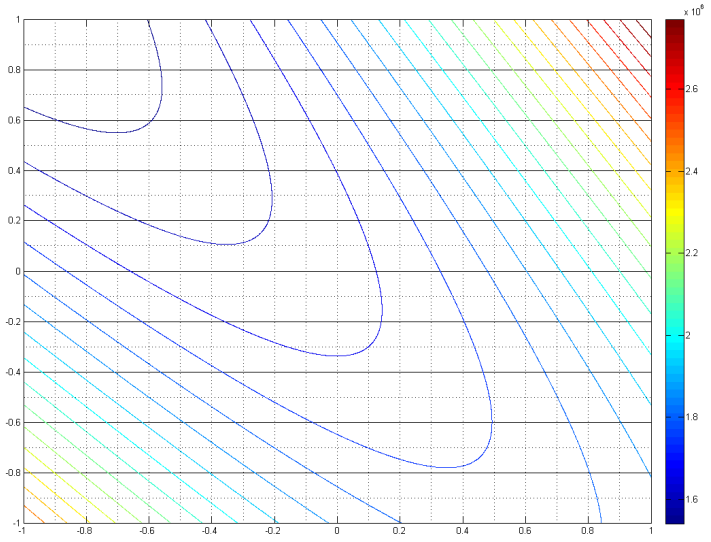

Figure 4.15: The niveau lines of the modified function $\tilde{f}^{(4)}(x)$ are parabolas.

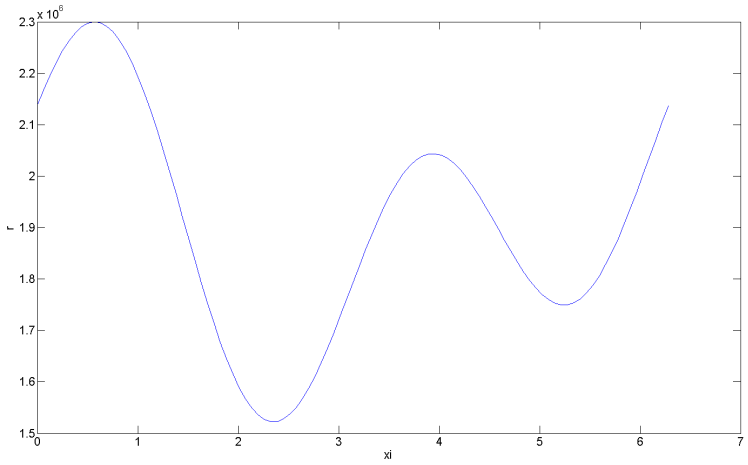

Figure 4.16: The graph of the function $g^{(4)}(\xi)$ as in (4.2) in the interval $[0, 2\pi]$ for Example 4.27.

Example 4.28. *We set $N := 40$, $q := 4$ and drew pseudo-random numbers for the real and imaginary parts of a_l and b_l $(l = 1, 2, ..., 40)$.*

We computed the Hessian matrix of $f^{(4)}(\vec{x})$ as in (4.6). Its eigenvalue decomposition is

$$\begin{aligned} \frac{d^2 f^{(4)}(\vec{x})}{d\vec{x}^2} &= 5.748610620006479 \cdot 10^6 \cdot \vec{v_1}\vec{v_1}^T + 3.064161625161817 \cdot 10^6 \cdot \vec{v_2}\vec{v_2}^T \\ \textit{with} \quad \vec{v_1} &= \begin{bmatrix} -0.501848768960128 & 0.864955382140145 \end{bmatrix}^T \\ \textit{and} \quad \vec{v_2} &= \begin{bmatrix} -0.864955382140145 & -0.501848768960128 \end{bmatrix}^T. \end{aligned} \tag{4.16}$$

We sent the following program to the solver KNITRO on the NEOS server.

$$\begin{aligned} \min \tilde{f}^{(4)}(\vec{x}) &= \sum_{l=1}^{40} \left(\gamma_l - \begin{bmatrix} \alpha_l & \beta_l \end{bmatrix} \cdot \vec{x}\right)^2 + 1532080.812580908 \cdot \left(1 - \vec{x}^T\vec{x}\right) \\ \text{s.t. } \vec{x}^T\vec{x} &\leq 1, \quad \vec{x} \in \mathbb{R}^2. \end{aligned}$$

with α_l, β_l and γ_l $(l = 1, 2, ..., 40)$ as in (4.1). The solution is

$$\begin{aligned} \vec{x} &= \begin{bmatrix} 0.778031 & 0.628226 \end{bmatrix}^T. \\ \Longrightarrow \quad \vec{x}^T\vec{x} &= 1 \\ \textit{and} \quad \mu^{(4)} &= 0.778031 + 0.628226i. \\ \Longrightarrow \quad \xi^{(4)} &\approx 0.67927 \\ \textit{with} \quad \mu^{(4)} &= \exp\left(\xi^{(4)} \cdot i\right). \end{aligned}$$

□

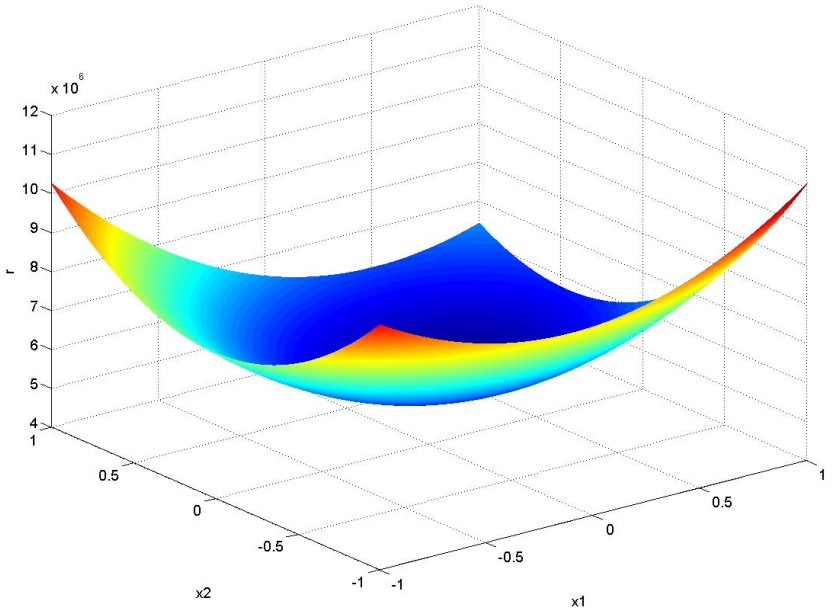

Figure 4.17: The graph of the function $f^{(4)}(\vec{x})$ as in (4.6) for Example 4.28. The graph of the function $f^{(4)}(\vec{x})$ is curved in each direction in the plane.

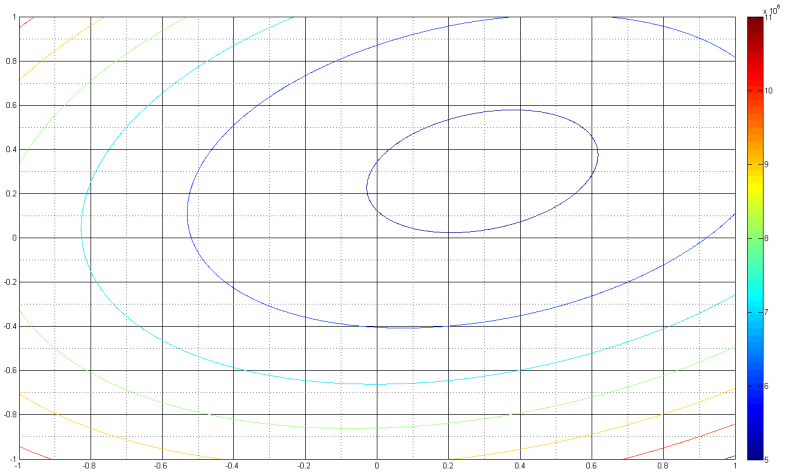

Figure 4.18: The niveau lines of the function $f^{(4)}(\vec{x})$ are ellipses.

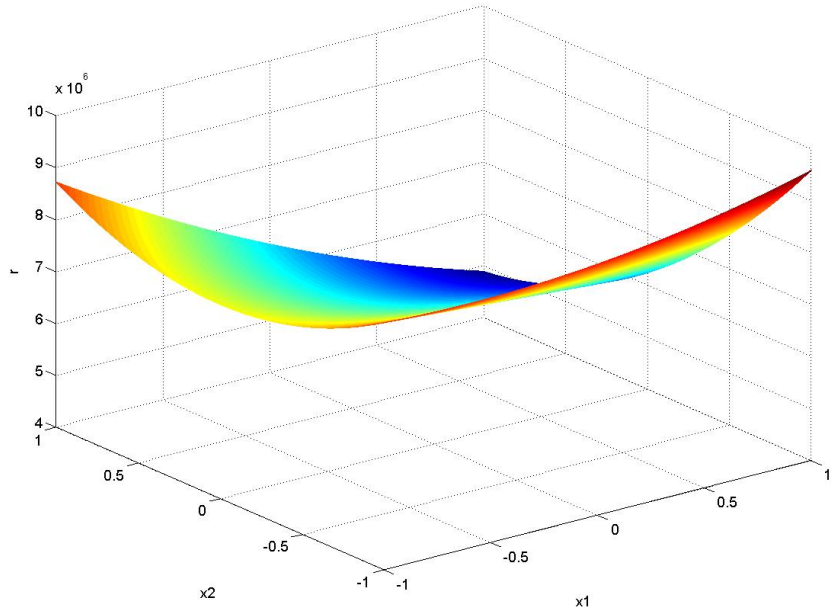

Figure 4.19: The graph of the modified function $\tilde{f}^{(4)}(\vec{x})$ as in (4.14) for Example 4.28. The graph of the function $\tilde{f}^{(4)}(\vec{x})$ is not curved any more in $\vec{v_2}$-direction (with $\vec{v_2}$ as in (4.16)).

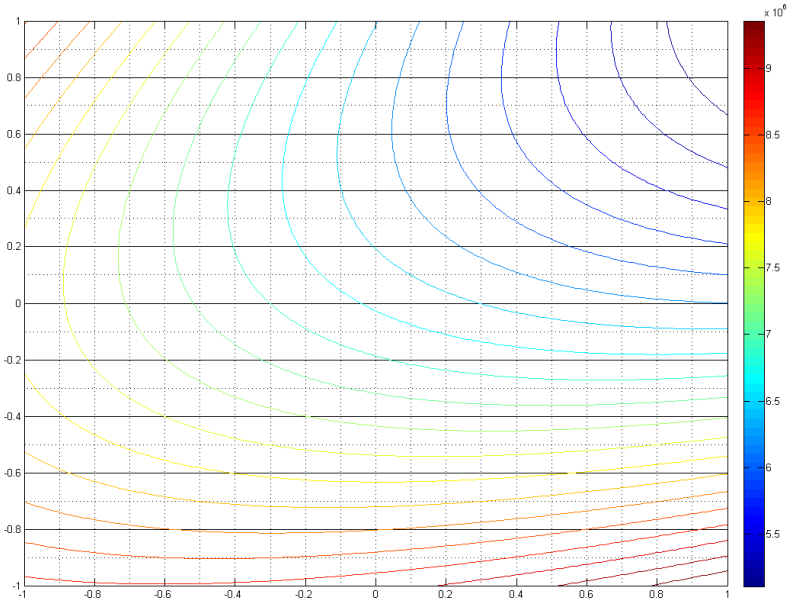

Figure 4.20: The niveau lines of the modified function $\tilde{f}^{(4)}(\vec{x})$ are parabolas.

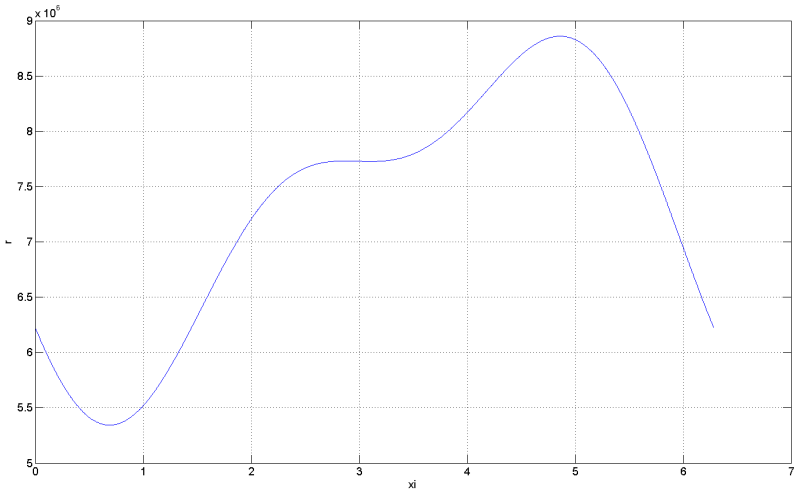

Figure 4.21: The graph of the function $g^{(4)}(\xi)$ as in (4.2) in the interval $[0, 2\pi]$ for Example 4.28.

4.6 The solution in the case $q = \infty$

For this case we developed a special combinatorial method which was not mentioned in the previous chapter.
The constraints of the minimization problem (4.5) are not convex in general. But we can square them and obtain an equivalent minimization problem (2.4) for the case $q = \infty$.

$$\begin{aligned} &\min r_{jk} \\ &\text{s.t. } r_{jk}(x) \geq \gamma_l^{(jk)} - \begin{bmatrix} \alpha_l^{(jk)} & \beta_l^{(jk)} \end{bmatrix} \cdot x, \qquad l = 1, 2, ..., N \\ &\quad x^T x = 1, \quad x \in \mathbb{R}^2 \end{aligned} \tag{4.17}$$

with $\alpha_l^{(jk)}$, $\beta_l^{(jk)}$ and $\gamma_l^{(jk)}$ $(l \in \{1, 2, ..., N\})$ as in (4.1). Obviously, $\left(g_{jk}^{(\infty)}(\xi)\right)^2$ is the minimal possible value of $r_{jk}\left(\begin{bmatrix} \cos\xi \\ \sin\xi \end{bmatrix}\right)$. Moreover, the program (4.17) is linear. However, the global minimum lies in the interior of the disk in general. Finding a global minimum over the circle requires more considerations.
Let

$$\begin{aligned} S &:= \left\{ \begin{bmatrix} x \\ r \end{bmatrix} \;\middle|\; x^T x = 1, \quad r \in \mathbb{R} \right\} \subset \mathbb{R}^3, \\ F_l &:= \left\{ \begin{bmatrix} x \\ r \end{bmatrix} \;\middle|\; r = \gamma_l^{(jk)} - \begin{bmatrix} \alpha_l^{(jk)} & \beta_l^{(jk)} \end{bmatrix} \cdot x \right\} \subset \mathbb{R}^3 \\ \text{and} \quad E_l &:= F_l \cap S \subset \mathbb{R}^3, \qquad l = 1, 2, ..., N, \end{aligned} \tag{4.18}$$

with $\alpha_l^{(jk)}$, $\beta_l^{(jk)}$ and $\gamma_l^{(jk)}$, $l \in \{1, 2, ..., N\}$, as in (4.1). Obviously, the sets F_l are planes, S is the lateral area of a cylinder and the sets E_l are ellipses. We introduce some new terms in order to explain our ideas concisely.

4 The one-dimensional minimization problems in the cases $\mathbb{F} = \mathbb{C}$, $p \neq 2$ and $q \in [1, \infty] \backslash \{2\}$

Definition 4.29. *Given the program* (4.17). *Let* S *and* E_l $(l = 1, 2, ..., N)$ *be as in* (4.18). *A point* $y \in S$ *is called a* ***breakpoint****, if it is an intersection point of two ellipses* E_l, E_m *with* $l, m \in \{1, 2, ..., N\}$ *as in* (4.18) *and* $E_l \cap E_m \neq E_l$.
If $m \in I_2(j,k)$, *then* $y_m \in E_m$ *is called the* ***low point*** *of the ellipse* E_m, *if* y_m *minimizes the third coordinate* r *with respect to* E_m. *We call a point* $y \in S$ *a* ***white point****, if it is a breakpoint or a low point of one of the* N *ellipses.*
We say, that a white point $\tilde{y} = \begin{bmatrix} \tilde{x} & \tilde{r} \end{bmatrix}^T \in \mathbb{R}^3$ *violates a constraint, if* $\tilde{r} < \gamma_l^{(jk)} - \begin{bmatrix} \alpha_l^{(jk)} & \beta_l^{(jk)} \end{bmatrix} \cdot \tilde{x}$ *for some* $l \in \{1, 2, ..., N\}$.

Obviously, a low point $y_m \in E_m$ is unique with respect to the ellipse E_m.

Lemma 4.30. *Each white point as in Definition* (4.29) *can be computed in constant time.*

Proof. If $m \in I_2(j,k)$, then

$$y_m = \begin{bmatrix} 0 \\ 0 \\ \gamma_m^{(jk)} \end{bmatrix} + \frac{1}{\sqrt{\left(\alpha_m^{(jk)}\right)^2 + \left(\beta_m^{(jk)}\right)^2}} \cdot \begin{bmatrix} \alpha_m^{(jk)} \\ \beta_m^{(jk)} \\ -\left(\alpha_m^{(jk)}\right)^2 - \left(\beta_m^{(jk)}\right)^2 \end{bmatrix}$$

is the low point of the ellipse E_m. Existing breakpoints satisfy the system of equations

$$\begin{aligned} \gamma_l^{(jk)} - \begin{bmatrix} \alpha_l^{(jk)} & \beta_l^{(jk)} \end{bmatrix} \cdot x &= \gamma_m^{(jk)} - \begin{bmatrix} \alpha_m^{(jk)} & \beta_m^{(jk)} \end{bmatrix} \cdot x, \\ x^T x &= 1. \end{aligned} \tag{4.19}$$

We convert the first equation and square it.

$$\begin{aligned} \gamma_l^{(jk)} - \gamma_m^{(jk)} + \left(\alpha_m^{(jk)} - \alpha_l^{(jk)}\right) \cdot x_1 &= \left(\beta_l^{(jk)} - \beta_m^{(jk)}\right) \cdot x_2 \\ \left(\gamma_l^{(jk)} - \gamma_m^{(jk)}\right)^2 + 2 \cdot \left(\gamma_l^{(jk)} - \gamma_m^{(jk)}\right) \left(\alpha_m^{(jk)} - \alpha_l^{(jk)}\right) \cdot x_1 + \left(\alpha_m^{(jk)} - \alpha_l^{(jk)}\right)^2 \cdot x_1^2 &= \left(\beta_l^{(jk)} \beta_m^{(jk)}\right)^2 \cdot x_2^2 \end{aligned}$$

Using $x_2^2 = 1 - x_1^2$ we obtain

$$\begin{aligned} &\overbrace{\left(\left(\alpha_m^{(jk)} - \alpha_l^{(jk)}\right)^2 + \left(\beta_l^{(jk)} - \beta_m^{(jk)}\right)^2\right)}^{h:=} \cdot x_1^2 + \\ &2 \cdot \underbrace{\left(\gamma_l^{(jk)} - \gamma_m^{(jk)}\right) \left(\alpha_m^{(jk)} - \alpha_l^{(jk)}\right)}_{=:z_2} \cdot x_1 \quad + \underbrace{\left(\gamma_l^{(jk)} - \gamma_m^{(jk)}\right)^2 - \left(\beta_l^{(jk)} - \beta_m^{(jk)}\right)^2}_{=:t_2} = 0. \end{aligned} \tag{4.20}$$

We can try to solve this quadratic equation and insert the solution in the first equation of (4.19). If the intersection line of F_l and F_m is parallel to x_1, then that linear equation does not have a unique solution. In this case we can use the analogous quadratic equation for x_2.

4 The one-dimensional minimization problems in the cases $\mathbb{F} = \mathbb{C}$, $p \neq 2$ and $q \in [1, \infty] \backslash \{2\}$

$$\left(\left(\alpha_m^{(jk)} - \alpha_l^{(jk)} \right)^2 + \left(\beta_l^{(jk)} - \beta_m^{(jk)} \right)^2 \right) \cdot x_2^2 +$$
$$2 \cdot \underbrace{\left(\gamma_l^{(jk)} - \gamma_m^{(jk)} \right) \left(\beta_m^{(jk)} - \beta_l^{(jk)} \right)}_{=:z_1} \cdot x_2 + \underbrace{\left(\gamma_l^{(jk)} - \gamma_m^{(jk)} \right)^2 - \left(\alpha_l^{(jk)} - \alpha_m^{(jk)} \right)^2}_{=:t_1} = 0. \tag{4.21}$$

Therefore each existing breakpoint or low point can be computed in constant time.

□

Lemma 4.31. *The one-dimensional minimization problem* (2.4) *with* $q = \infty$ *can be solved in* $\mathcal{O}\left(N^3\right)$ *for one pair of coordinates* (j,k) *with* $I_2(j,k) = \{1, 2, ..., N\}$.

Proof. The program (4.17) is equivalent to the problem (2.4) for the case $q = \infty$. The first N constraints describe a polyeder P which is bounded by at most N planes F_l as in (4.18). The $(N+1)$-st constraint of (4.17) describes the lateral surface area S of a cylinder, and the planes F_l intersect the area S in ellipses E_l as in (4.18).
Now, let us consider an arbitrary point $\tilde{x} \in \mathbb{R}^2$ on the unit circle. There is a minimal $r(\tilde{x})$ which satisfies the first N constraints of (4.17) and an index $m \in \{1, 2, ..., N\}$ such that

$$\tilde{y} := \begin{bmatrix} \tilde{x} \\ r(\tilde{x}) \end{bmatrix} \in E_m \cap P.$$

If $\tilde{y} = y_m$ is the low point of the ellipse E_m as in Definition 4.29, then the point $\tilde{y}$ is also a global minimum of the program (4.17) because of the first N constraints.
Otherwise, E_m contains a directed arc L from $\tilde{y}$ to the low point y_m as in Definition 4.29 such that the third coordinate decreases strictly monotonically on L. The arc L can only leave the polyeder P when it intersects a plane $F_{\tilde{m}}$ with $m \neq \tilde{m} \in \{1, 2, ..., N\}$. Since $\tilde{x}$ has been arbitrary, a global minimum of the program (4.17) can only lie on breakpoints or low points as in Definition 4.29. It is easy to see that two different ellipses in $\mathbb{R}^3$ can intersect in at most two points. Thus there are at most $N + 2 \cdot \binom{N}{2} = N^2$ white points. According to Lemma 4.30 all the white points can be computed in $\mathcal{O}\left(N^2\right)$. For each white point we must check, whether one of the first N constraints is violated. This needs a runtime of $\mathcal{O}\left(N^3\right)$. Among the remaining white points, which satisfy all the constraints, we must determine a point with minimal third coordinate r. This is possible in $\mathcal{O}\left(N^2\right)$.

□

Theorem 4.32. *The one-dimensional minimization problem* (2.4) *with* $q = \infty$ *can be solved in* $\mathcal{O}\left(N^3\right)$ *for one pair of coordinates* (j,k).

Proof. If $I_2(j,k) = \{1, 2, ..., N\}$, then Lemma 4.31 yields the statement.
Now let $m \in \{1, 2, ..., N\}$ be an index with $\alpha_m^{(jk)} = \beta_m^{(jk)} = 0$. We use the same method as in the proof of Lemma 4.31 and show that we can choose an arbitrary point $\tilde{y} \in E_m$ instead of the not existing low point of this ellipse. If no global minimum of the program (4.17) lies on the ellipse E_m, the choice of $\tilde{y}$ has obviously no effect on the solution. Thus let $\hat{y} \in E_m$ be a global minimum of the program.
If $\tilde{y} \in E_m$ satisfies all the constraints, then it is also a global minimum, because the third coordinate is constant on E_m. Otherwise $\tilde{y} \in E_m$ violates the $\bar{m}$-th constraint of the program with $\bar{m} \in \{1, 2, ..., N\} \backslash \{m\}$. Thus the ellipse E_m contains at least two breakpoints, because it intersects the ellipse $E_{\bar{m}}$. One of these breakpoints which is the nearest one to $\tilde{y} \in E_m$ satisfies all the constraints and is a global minimum, too. This breakpoint is checked because we use the same method as in the proof of Lemma 4.31 using an arbitrary point instead of the not existing low points of some ellipses.

□

4 The one-dimensional minimization problems in the cases $\mathbb{F} = \mathbb{C}$, $p \neq 2$ and $q \in [1, \infty] \backslash \{2\}$

Now we are able to construct an algorithm applying the proposed method of Lemma 4.31 and Theorem 4.32. However, our algorithm has some features which shall be explained before.

Lemma 4.33. *(i) If a low point y_l of some ellipse E_l $(l \in \{1, 2, ..., N\})$ violates no constraint, then y_l is a global minimum.*

(ii) If $\alpha_m^{(jk)} = \beta_m^{(jk)} = 0$ for some $m \in \{1, 2, ..., N\}$ and $y_m \in E_m$ violates no constraint, then y_m is a global minimum.

(iii) If $\left(\alpha_l^{(jk)} - \alpha_m^{(jk)}\right)^2 + \left(\beta_l^{(jk)} - \beta_m^{(jk)}\right)^2 = 0$, then the planes F_l and F_m are parallel. This means particularly that the ellipses E_l and E_m cannot have an intersection point.

Proof. (i) This follows immediately from the definition of the low point.
(ii) If $\alpha_m^{(jk)} = \beta_m^{(jk)} = 0$, then the third coordinate is constant on E_m. This means that no point of the feasible polyeder can have a lower value of the third coordinate.
(iii) If $\left(\alpha_l^{(jk)} - \alpha_m^{(jk)}\right)^2 + \left(\beta_l^{(jk)} - \beta_m^{(jk)}\right)^2 = 0$, then the gradients of the planes F_l and F_m must be equal.

□

Our algorithm first sets the current optimal value r to infinity. In the second step it verifies whether one of the low points satisfies all the constraints. If it can find such a low point, then it updates the current optimal value. Moreover, in this case we will be ready according to Lemma 4.33 (i). If a plane F_l has no descent up to a tolerance parameter ε, then we can replace the low point by an arbitrary point of the ellipse E_l according to the proof of Theorem 4.32 and Lemma 4.33 (ii).
The third step must only be executed if the current optimal value r is still infinity after the second step. The algorithm computes the intersection points of all pairs of ellipses which have two intersection points. If one of these points satisfies all the constraints and its third coordinate is lower than the current optimal value r, then r is updated.

Algorithm 4.34.
Input: N, $\alpha_l, \beta_l, \gamma_l$ $(l = 1, 2, ..., N)$ *as in* (4.1), *a tolerance parameter* $\varepsilon > 0$.
Output: *An optimal* $\mu^{(\infty)}$ *as in* (2.4).

1.) Set $r := \infty$. // *r is the current optimal value.*

2.) For $l := 1$ *to* N *do*

if $\left(\alpha_l^2 + \beta_l^2 < \varepsilon^2\right)$ *set* $y := \begin{bmatrix} 1 & 0 \end{bmatrix}^T$, // *an arbitrary point on the unit circle*

else set $y := \dfrac{1}{\sqrt{\alpha_l^2 + \beta_l^2}} \cdot \begin{bmatrix} \alpha_l & \beta_l \end{bmatrix}^T$. // *the low point of the l-th ellipse*

Set $\tilde{r} := \gamma_l - \begin{bmatrix} \alpha_l & \beta_l \end{bmatrix} \cdot y$.
if $(\tilde{r} < r)$ *do* // *Better than the current optimal value? Then verify, whether a constraint is violated.*

above := 0.
for $j := 1$ *to* N *do*
if $\left(\gamma_j - \begin{bmatrix} \alpha_j & \beta_j \end{bmatrix} \cdot y > \tilde{r} + \varepsilon\right)$ *do*
above := 1;
break;
if (above == 0) // *If no constraint is violated, then a global minimum is found. We are ready, if a low point satisfies all the constraints.*

set $x := y$ *and* $r := \tilde{r}$.
break;

3.) If $(r == \infty)$ *do* // *Otherwise we*
for $j := 1$ *to* $N - 1$ *do* // *search breakpoints.*
for $k := j + 1$ *to* N *do*
$h := (\alpha_j - \alpha_k)^2 + (\beta_j - \beta_k)^2$;
if $(h < \varepsilon)$
continue; // *(Lemma 4.33 (iii))*
set $z_1 := (\gamma_j - \gamma_k) \cdot (\beta_j - \beta_k)$
and $t_1 := (\gamma_k - \gamma_j)^2 - (\alpha_k - \alpha_j)^2$;
set $z_2 := (\gamma_j - \gamma_k) \cdot ((\alpha_j - \alpha_k)$
and $t_2 := (\gamma_k - \gamma_j)^2 - (\beta_k - \beta_j)^2$;
set $d_1 := \left(\frac{z_1}{h}\right)^2 - \frac{t_1}{h}$
and $d_2 := \left(\frac{z_2}{h}\right)^2 - \frac{t_2}{h}$;
if $(d_1 > \varepsilon)$ // *See Remark 4.35.*
set $s_1 := \frac{z_1}{h} - \sqrt{d_1}$
and $s_2 := \frac{z_1}{h} + \sqrt{d_1}$;
set $c_1 := \frac{\gamma_k - \gamma_j + (\beta_j - \beta_k) * s_1}{\alpha_k - \alpha_j}$
and $c_2 := \frac{\gamma_k - \gamma_j + (\beta_j - \beta_k) * s_2}{\alpha_k - \alpha_j}$;
else if $(d_2 > \varepsilon)$ // *See Remark 4.35.*
set $c_1 := \frac{z_2}{h} - \sqrt{d_2}$
and $c_2 := \frac{z_2}{h} + \sqrt{d_2}$;
set $s_1 := \frac{\gamma_k - \gamma_j + (\alpha_j - \alpha_k) * c_1}{\beta_k - \beta_j}$
and $s_2 := \frac{\gamma_k - \gamma_j + (\alpha_j - \alpha_k) * c_2}{\beta_k - \beta_j}$;
set $r_1 := \gamma_j - \begin{bmatrix}\alpha_j & \beta_j\end{bmatrix} \cdot \begin{bmatrix}c_1 \\ s_1\end{bmatrix}$
and $r_2 := \gamma_j - \begin{bmatrix}\alpha_j & \beta_j\end{bmatrix} \cdot \begin{bmatrix}c_2 \\ s_2\end{bmatrix}$;
if $(r_1 < r)$ *do* // *Better than the current*
// *optimal value?*
above $:= 0$.
for $l := 1$ *to* N *do*
if $\left(\gamma_l - \begin{bmatrix}\alpha_l & \beta_l\end{bmatrix} \cdot \begin{bmatrix}c_1 \\ s_1\end{bmatrix} > r_1 + \varepsilon\right)$ *do*
above $:= 1$;
break;
if (above $== 0$)
set $x := \begin{bmatrix}c_1 \\ s_1\end{bmatrix}$ *and* $r := r_1$. // *found an*
// *improvement*
if $(r_2 < r)$ *do* // *Better than the current*
// *optimal value?*
above $:= 0$.
for $l := 1$ *to* N *do*
if $\left(\gamma_l - \begin{bmatrix}\alpha_l & \beta_l\end{bmatrix} \cdot \begin{bmatrix}c_2 \\ s_2\end{bmatrix} > r_2 + \varepsilon\right)$ *do*
above $:= 1$;
break;
if (above $== 0$)

4 The one-dimensional minimization problems in the cases $\mathbb{F} = \mathbb{C}$, $p \neq 2$ *and* $q \in [1, \infty] \backslash \{2\}$

$$\text{set } x := \begin{bmatrix} c_2 \\ s_2 \end{bmatrix} \text{ and } r := r_2. \qquad \begin{array}{l} // \text{ found an} \\ // \text{ improvement} \end{array}$$

Remark 4.35. *The quadratic equation* (4.21) *has two real solutions if and only if the discriminant satisfies* $d_1 > 0$ *as well as the the quadratic equation* (4.20) *has two real solutions if and only if the discriminant satisfies* $d_2 > 0$.

Example 4.36. *We implemented Algorithm 4.34 in MATLAB and used a and b as in Example 4.27 with computing* α, β *and* γ *as in* (4.1). *We replaced q by* ∞.

The algorithm yields the solution

$$\begin{aligned} & x = \begin{bmatrix} 0.44412544 & -0.89596462 \end{bmatrix}^T. \\ \Longrightarrow \quad & x^T x = 1 \\ \text{and} \quad & \mu^{(\infty)} = 0.44412544 - 0.89596462i. \\ \Longrightarrow \quad & \xi^{(\infty)} \approx 5.1725869 \\ \text{with} \quad & \mu^{(\infty)} = \exp\left(\xi^{(\infty)} \cdot i\right). \end{aligned}$$

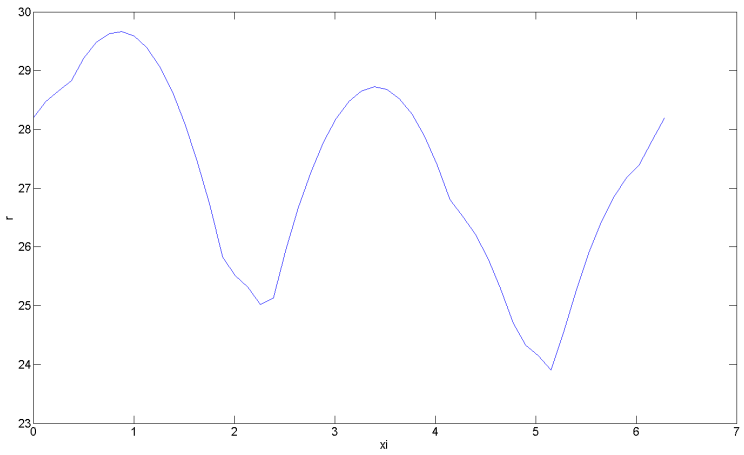

Figure 4.22: The graph of the function $g^{(\infty)}(\xi)$ as in (4.2) in the interval $[0, 2\pi]$ for Example 4.36.

The required runtime was $2.315238 \cdot 10^{-4}$ *seconds for* $N = 12$. *For comparison we show the graph of the function* $g^{(\infty)}(\xi)$ *as in* (2.4) *and in* (4.2) *in the interval* $[0, 2\pi]$ *in Figure 4.22.*

□

Example 4.37. *We used a and b as in Example 4.27 with computing* α, β *and* γ *as in* (4.1). *We replaced q by* ∞.

The algorithm yields the solution

$$
\begin{aligned}
x &= \begin{bmatrix} 0.73848348 & -0.67427156 \end{bmatrix}^T. \\
\Longrightarrow \quad x^T x &= 1 \\
\textit{and} \quad \mu^{(\infty)} &= 0.73848348 - 0.67427156i. \\
\Longrightarrow \quad \xi^{(\infty)} &\approx 5.543207 \\
\textit{with} \quad \mu^{(\infty)} &= \exp\left(\xi^{(\infty)} \cdot i\right).
\end{aligned}
$$

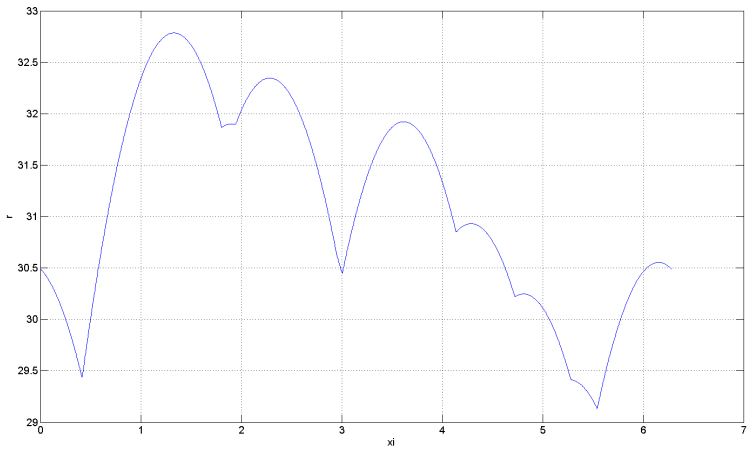

Figure 4.23: The graph of the function $g^{(\infty)}(\xi)$ as in (4.2) in the interval $[0, 2\pi]$ for Example 4.37.

5 Conclusions

We treated here the (ℓ_p, ℓ_q) Procrustes problem with $p \neq 2$. If $\mathbb{F} = \mathbb{R}$, then we can find efficiently a global minimum for each $q \in [1, \infty]$.
If $\mathbb{F} = \mathbb{C}$, then we can find efficiently a global minimum for the cases $q \in \{2, 4, \infty\}$. In the remaining cases we have the choice between speed and precision.
If a fast solution for an (ℓ_p, ℓ_q) Procrustes problem with $\mathbb{F} = \mathbb{C}$, $p \neq 2$ and $q \notin \{2, 4, \infty\}$ is needed, then the SLA is recommendable, because it it very fast and its solution is often a global minimum. If we need a global minimum for such a problem, then we can try to apply the Branch-and-Bound method. Although its runtime can grow exponentially with the input length in the worst case, its measured runtime in our tests was slower than the SLA by a moderate factor only. If the Branch-and-Bound method is requiring too much time for a pair (j, k) of coordinates, then abort it and apply the SLA instead. Remember, we compute n^2 candidates for n entries of a unitary matrix which is our entire solution. The SLA yields often a global minimum. If it fails to find a global minimum, then with high probability its partial solution has no effect to the entire solution.

Bibliography

[1] Aho, A. V.; Hopcroft, J. E. and Ullmann, J. D.: The Design and Analysis of Computer Algorithms, Addison-Wesley, Reading, Mass., (1974).

[2] Allwright, J. C.: Positive semidefinite matrices: Characterization via conical hulls and least-squares solution of a matrix equation, SIAM J. Control and Optimization, Vol. 26, no. 3, 537 - 556 (1988).

[3] Alt, H.; Blum, N.; Mehlhorn, K. and Paul, M.: Computing a maximum cardinality matching in a bipartite graph in time $\mathcal{O}\left(n^{1.5}\sqrt{m/\log n}\right)$, Information Processing Letters 37, 237 - 240 (1991).

[4] Arazy, J.: Isometries of Complex Symmetric Sequence Spaces, Math. Z. 188, 427 - 431 (1985).

[5] Benson, H. P.: Deterministic Algorithms for Constrained Concave Minimization: A Unified Critical Survey, Naval Research Logistics, Vol. 43, pp. 765 - 795 (1996).

[6] Borg, I.; Groenen, P. J. F.: Modern multidimensional scaling (2nd ed.) NewYork: Springer (2005).

[7] Borg, I.; Groenen, P. J. F.; Mair, P.: Applied Multidimensional Scaling, Springer (2013).

[8] Byrd, R. H.; Nocedal, J.; Waltz, R.: KNITRO: An integrated package for nonlinear optimization, (English summary) Large-scale nonlinear optimization, 35 - 39, Nonconvex Optim. Appl. 83, Springer, New York (2006).

[9] Byrd, R. H.; Hribar, M. E.; Nocedal, J.: An interior point algorithm for large-scale nonlinear programming. Dedicted to John E. Dennis Jr. on his 60th birthday. SIAM J. Optim. 9 , no. 4, 877 - 900 (1999).

[10] Byrd, R. H.; Guanghui Liu; Nocedal, J.: On the local behaviour of an interior point method for nonlinear programming. Numerical analysis 1997 (Dundee), 37 - 56, Pitman Res. Notes Math. Ser., 380, Longman, Harlow, (1998).

[11] Byrd, R. H.; Gilbert, J. C.; Nocedal, J.: A trust region method based on interior point techniques for nonlinear programming. (English summary) Math. Program. 89 (2000), no. 1, Ser. A, 149 - 185.

[12] Burkard, R. E.; Hahn, W.; Zimmermann, U.: An algebraic approach to assignment problems, Mathematical Programming 12, 318 - 327, North-Holland Publishing Company (1977).

[13] Calamai, P. H. and Conn, A. R.: A stable algorithm for solving the multifacility location problem involving Euclidean distances, SIAM J. Sci. Stat. Comp., Vol. 1, No. 4, 512 - 526 (1980).

[14] Chang, S.; Li, C. K.: Certain isometries on $\mathbb{R}^n$, Linear Algebra and its Applications, 165, 251 - 265 (1992).

[15] Cheriyan, J.; Mehlhorn, K.: Can a maximum flow be computed in $\mathcal{O}(nm)$ time?; Automata, languages and programming, Proc. 17th Int. Colloq., Warwick/GB 1990; Lect. Notes Comput. Sci. 443, 235 - 248 (1990).

[16] Derigs, U. and Zimmerman, U.: An augmenting path method for solving linear bottleneck assignment problems, Computing 19, 285 - 295 (1978).

[17] Fung, G. M.; Mangasarian, O. L.: Equivalence of Minimal ℓ_0 and ℓ_p Norm Solutions of Linear Equalities, Inequalities and Linear Programs for Sufficiently Small p (English summary), J. Optim. Theory Appl. 151, no. 1, 1 - 10, 1573 - 2878 (2011).

[18] Golub, G. H. and Van Loan, C. F.: Matrix Computations, 2nd ed., John Hopkins University Press, Baltimore, MD (1989).

[19] Gower, J. C.: Generalized Procrustes Analysis, Psychometrika, Vol. 40, no. 1, 33 - 51 (1975).

[20] Gower, J. C.: Multivariate analysis: ordination, multidimensional scaling and allied topics, in Handbook of Applicable Mathematics, Vol. VI: Statistics, Lloyd, E.H., ed., John Wiley, Chichester, 1984, pp. 727 - 781 (1984).

[21] Higham, N.: Matrix Procrustes Problems (1995).
`http://www.maths.manchester.ac.uk/~higham/talks/procrust94.ps.gz`

[22] Hopcroft, J. E. and Karp, R. M.: An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs, SIAM J. Comput., Vol. 2, No. 4, 225 - 231 (1973).

[23] Horn, R.; Johnson, C. R.: Matrix Analysis, Cambrigde University Press (1985).

[24] Jarosz, K.: Any Banach space has an equivalent norm with trivial isometries, Israel Journal of Mathematics, Vol. 64, No. 1, 49 - 56 (1988).

[25] Jarre, F.; Stoer, J.: Optimierung, Springer-Verlag, Berlin - Heidelberg, (2004).

[26] Kintzel, U.: Polar Decompositions and Procrustes Problems in Finite Dimensional Indefinite Scalar Product Spaces, Dissertation, TU Berlin, Fakultät II: Mathematik und Naturwissenschaften, (2004).

[27] Kiskiras, J. and Halikias, G. D.: A note on the complex semi-definite matrix Procrustes problem, Numer. Linear Algebra Appl. 14, 485 - 502 (2007).

[28] Klotzek, B.: Euklidische und nichteuklidische Elementargeometrien, Verlag Harri Deutsch, (2001).

[29] Korte, B.; Vygen, J.: Combinatorial Optimization, Theory and Algorithms, fourth edition, Springer (2008).

[30] Kuhn, H. W.: A Note on Fermat's Problem, Math. Prog. 4, 98 - 107, North-Holland Publishing Company, (1973).

[31] Chi-Kwong Li, Wasin So: Isometries of ℓ_p-norm, Amer. Math. Monthly 101, no. 5, 452 - 453 (1994).

[32] Mangasarian, O. L.: Machine learning via polyhedral concave minimization. In: H. Fischer, B. Riedmueller, S. Schaeffler (eds.): Applied Mathematics and Parallel Computing - Festschrift for Klaus Ritter, pp. 175 - 188. Physica-Verlag, Springer, Heidelberg (1996).
`ftp://ftp.cs.wisc.edu/math-prog/tech-reports/95-20.ps`

[33] Lins, B.; Meade, P.; Mehl, C; Rodman, L.: Normal Matrices and Polar Decompositions in Indefinite Inner Products, Linear and Multilinear Algebra 49, no. 1, 45-89 (2001).

[34] Pardalos, P. M. and Vavasis, S. A.: Quadratic programming with one negative eigenvalue is NP-hard, J. Global Optimization 1, 15 - 22 (1991).

[35] Pietrzykowski, T.: An exact potential method for constrained maxima, SIAM J. Numer. Anal., Vol. 6, No. 2, 299 - 304 (1969).

[36] Plantenga, T.: KNITRO for Nonlinear Optimization; Ziena Optimization, Inc., INFORMS Practices (2006).

[37] Porembski, M.: Cutting Planes for Low-Rank-Like Concave Minimization Problems, Operations Research, Vol. 52, No. 6, 942 - 953 (2004).

[38] Punnen, A. P.; Nair, K. P. K.: Improved complexity bound for the maximum cardinality bottleneck bipartite matching problem, Discrete Applied Mathematics 55, 91 - 93 (1994).

[39] Rosen, J. B.; Xue, G. L.: On the convergence of a hyperboloid approximation procedure for the perturbed Euclidean multifacility location problem, Operations Research, Vol. 41, No. 6, 1164 - 1171 (1993).

[40] Schönemann, P. H.: A Generalized Solution of the Orthogonal Procrustes Problem, Psychometrika, Vol. 31, No. 1, 1 - 10 (1966).

[41] Schönemann, P. H. and Carroll, R. M.: Fitting one matrix to another under choice of a central dilation and a rigid motion, Psychometrika, Vol. 35, no. 2, 245 - 255 (1970).

[42] Schöning, U.: Algorithmik, Spektrum Akademischer Verlag Heidelberg - Berlin, (2001).

[43] Trendafilov, N. T.: On the ℓ_1 Procrustes Problem, Future Generation Computer Systems 19, 1177 - 1186 (2003).

[44] Trendafilov, N. T.; Watson, G. A.: The ℓ_1 oblique procrustes problem, Statistics and Computing 14, 39 - 51, (2004).

[45] Viklands, T.: Algorithms for the weighted orthogonal Procrustes problem and other least squares problems, Ph. D. Thesis, (2006), `http://www8.cs.umu.se/~viklands/PhD.pdf`

[46] Watson, G. A.: Solving generalizations of orthogonal Procrustes problems, World Scientific Series in Applicable Analysis 2, 413-426 (1993).

[47] Watson, G. A.: The solution of orthogonal Procrustes problems for a family of orthogonally invariant norms, Advances in Computational Mathematics 2, 393-405 (1994).

[48] Weiszfeld, E.: Sur le Point par Lequel le Somme des Distances de n Points Donnes est Minimum, Tokohu Math. J. 43, 355 - 386, (1937).

Impressum:
Herausgeber:
Der Dekan der
Fakultät für Mathematik
an der Technischen Universität Chemnitz
Sitz:
Reichenhainer Strae 39
09126 Chemnitz
Postanschrift:
09107 Chemnitz
Telefon: (0371) 531-22000
Telefax: (0371) 531-22009
E-Mail: dekanat@mathematik.tu-chemnitz.de
Internet:
http://www.tu-chemnitz.de/mathematik/
ISSN 1614-8835 (Print)