On the Adaptive Polygon Inflation Algorithm and its Implementation

Bachelor Thesis

University of Rostock

Faculty of Mathematics and Natural Sciences

Institute of Mathematics

Name:	Tomass Andersons
Matriculation number:	215206569
Supervisor and reviewer:	Prof. Dr. rer. nat. Klaus Neymeyr
Reviewer:	Dr. rer. nat. Mathias Sawall
Submission date:	20.06.2018

Contents

1 Introduction

Multivariate curve resolution techniques are important tools in analytical chemistry. Usually, the goal of these methods is to extract information about pure component concentration and absorptivity from a spectroscopic data matrix. This is a soft modeling problem in the field of chemometrics; it uses a minimal amount of knowledge about the system to derive useful results while using numerical methods.

Based on the assumption of Lambert-Beer law, the data matrix is decomposed into a product of concentration and absorptivity matrices. Thus, the pure component resolution of spectroscopic data can be interpreted as a nonnegative factorization problem. This means that the solutions of this problem could be transferred to other disciplines with similar research questions.

Generally, the aforementioned nonnegative matrix factorization does not have a unique solution, see the research paper from Abdollahi and Tauer [1]. There are multiple types of ambiguities for this problem; here mainly the so-called rotational ambiguity is examined. It leads to a continuum of possible solutions.

This work focuses on the mixture resolution for three-component chemical systems. In this case, there exists an advantageous low-dimension representation that helps to visualize the rotational ambiguity in two dimensions. It is called the area of feasible solutions (AFS).

Various methods have been suggested to approximate the AFS. Polygon inflation algorithm is one of them and together with its inverse alteration can be successfully used for three-component systems. Based on this method, a MATLAB toolbox FACPACK was introduced in 2014 [15], [18].

Some essential research papers include the solution for two-component systems [8] and geometric approaches [9], [2] and [12]. More recent numerical algorithms are described in [5] and [16]. The latter is the first work on the polygon inflation algorithm and was followed by [17] and [14]. Important theoretical background was discussed earlier in [11] and [13].

The goal of this work is to motivate and to examine the polygon inflation algorithm and its inverse counterpart for the pure component resolution of three-component systems by means of a MATLAB implementation. Chapter 2 derives the AFS from the original mixture resolution problem. The polygon inflation algorithm and the inverse polygon inflation algorithm are introduced in chapter 3 and chapter 4 respectively. Also, some difficulties of practical realization are shown there. An optimal choice of parameters and numerical results are of a particular interest; they are summarized in chapter 5.

2 Nonnegative matrix factorization

This chapter introduces the nonnegative matrix factorization problem for the pure component analysis, based on the Lambert-Beer law. Further, an approach, using the singular value decomposition, and the difficulties, associated with the ambiguity of the nonnegative matrix factorization problem, are presented. Finally, with the help of the Perron-Frobenius theorem, the concept of an AFS for three-component systems is established.

The theory behind this chapter is based on the work by Lawton and Sylvestre [8] and Borgen and Kowalski [2], that is further developed in other publications. This chapter predominantly uses [16], [17] and [14] to lay the theoretical foundations of this work.

2.1 The matrix representation of the problem

2.1.1 The chemical background

Pure component resolution starts with a given nonnegative spectroscopic data matrix $D \in \mathbb{R}^{k \times n}$. It contains mixed experimental data.

The chemical justification of the nonnegative matrix factorization problem is the Lambert-Beer law in matrix notation. It reads $D = CA + E$ with a concentration matrix $C \in \mathbb{R}^{k \times s}$, an absorption matrix $A \in \mathbb{R}^{s \times n}$ and an error matrix $E \in \mathbb{R}^{k \times n}$. The chemical meaning is as follows: D_{ij} constitutes the absorption measurement i at the frequency j, A_{lj} the absorptivity of the pure component l at the frequency j and C_{il} the concentration of the pure component l in the measurement i. The number of chemical components is s and it is assumed that $\operatorname{rank}(D) = s$.

The main goal is to find a nonnegative concentration matrix $C \in \mathbb{R}^{k \times s}$ and a nonnegative absorption matrix $A \in \mathbb{R}^{s \times n}$, so that

$$D = CA.$$

The existence of a solution can be assumed for experimental data matrices, as stated in [13].

2.1.2 The singular value decomposition

The singular value decomposition can be used as a foundation to find the nonnegative matrix factorization of D.

Theorem 1. *For $D \in \mathbb{R}^{k \times n}$ exists a factorization*

$$D = U \Sigma V^T,$$

called ***singular value decomposition****, where* $U \in \mathbb{R}^{k\times k}$ *orthogonal,* $\Sigma = \mathrm{diag}(\sigma_i) \in \mathbb{R}^{k\times n}$ *with* σ_i *called the i-th singular value,* $i = \min(m, n)$*, and* $V \in \mathbb{R}^{n\times n}$ *orthogonal.*

Proof. See theorem 2.4.1. in [6]. □

A rank-s matrix can be represented by the first s singular values and vectors. Let $A_{i,:}$ denote the i-th row of matrix A and $A_{:,i}$, the i-th column.

Theorem 2. *If* $D \in \mathbb{R}^{k\times n}$ *with* $\mathrm{rank}(D) = s$*, then*

$$D = \sum_{i=1}^{s} \Sigma_{ii} U_{:,i} V_{:,i}^T.$$

Proof. See corollary 2.4.7. in [6]. □

Since D is a rank-s matrix, only the first s singular values and singular vectors are needed and the rest can be discarded, reducing the matrices to $\hat{U} \in \mathbb{R}^{k\times s}$, $\hat{\Sigma} = \mathrm{diag}(\sigma_i) \in \mathbb{R}^{s\times s}$ and $\hat{V} \in \mathbb{R}^{n\times s}$. The property $U\Sigma V^T = \hat{U}\hat{\Sigma}\hat{V}^T$ holds true. Let's call this representation the truncated singular value decomposition.

2.1.3 The transformation matrix

In general, the singular value decomposition does not provide a nonnegative factorization; however, it can be used for creating one. To do this, a regular transformation is required.

Definition 1. *A regular matrix* $T \in \mathbb{R}^{s\times s}$ *in the context of this thesis is called a feasible transformation matrix if*

$$D = \hat{U}\hat{\Sigma}\hat{V}^T = \underbrace{\hat{U}\hat{\Sigma}(T^{-1}}_{C}\underbrace{T)\hat{V}^T}_{A}, \tag{2.1}$$

with $C = \hat{U}\hat{\Sigma}T^{-1}$ *and* $A = T\hat{V}^T$ *being component-wise nonnegative matrices.*

The existence of such T is proved in Lemma 2.1. in [11], see the theorem below.

Theorem 3. *For a given* D *with* $D = CA$ *and a truncated singular value decomposition* $D = \hat{U}\hat{\Sigma}\hat{V}^T$ *with* $\mathrm{rank}(D) = s$*, there exists a feasible transformation matrix* T*.*

Proof. Let's consider the matrices C and $\hat{U}\hat{\Sigma}$ as linear transformations

$$C : \mathbb{R}^s \to \mathbb{R}^k \text{ and } \hat{U}\hat{\Sigma} : \mathbb{R}^s \to \mathbb{R}^k.$$

From $D = CA = \hat{U}\hat{\Sigma}\hat{V}^T$ and $\mathrm{rank}(D) = \mathrm{rank}(C) = \mathrm{rank}(\hat{U}\hat{\Sigma}) = s$ follows that the spans of the images of both linear transformations coincide. This allows a change of basis from one to the other. Thus, a regular matrix T with $CT = \hat{U}\hat{\Sigma}$ exists.

From $D = CA = \hat{U}\hat{\Sigma}\hat{V}^T$ and $C = \hat{U}\hat{\Sigma}T^{-1}$ follows

$$\hat{U}\hat{\Sigma}T^{-1}A = \hat{U}\hat{\Sigma}\hat{V}^T$$
$$\hat{U}\hat{\Sigma}(T^{-1}A - \hat{V}^T) = 0,$$

where 0 is a matrix consisting of zeros. The last step follows from the fact that $\text{rank}(\hat{U}\hat{\Sigma}) = s$. Consequently, $(T^{-1}A - \hat{V}^T)$ is a matrix consisting of zeros and

$$A = T\hat{V}^T.$$

□

Therefore, the singular value decomposition together with a feasible transformation matrix can be used to obtain a nonnegative factorization of D. The nonnegative factorization problem has been reduced to determining a feasible regular transformation matrix T with s^2 degrees of freedom.

2.1.4 Problem of ambiguity

A feasible transformation matrix T is not unique and a set of possible nonnegative factorizations exists. There are three types of ambiguities for the transformation matrix, according to [1].

Permutation ambiguity can be described, as expanding the original transformation with a permutation matrix $P \in \mathbb{R}^{s\times s}$ to

$$D = \hat{U}\hat{\Sigma}\hat{V}^T = \underbrace{\hat{U}\hat{\Sigma}(T^{-1}P^T}_{\text{C}}\underbrace{PT)\hat{V}^T}_{\text{A}}.$$

The resulting factorization is nonnegative; therefore, PT is also a feasible transformation. This changes only the order of components in the nonnegative matrices C and A, and that is not problematic in the context of this work.

Intensity ambiguity is caused by similar argumentation to the permutation ambiguity. Instead of a permutation matrix, a diagonal matrix $\text{diag}(a_1, \ldots, a_s)$ and its inverse is used here. This type of ambiguity can be limited by additional information or assumptions.

Rotational ambiguity is the most problematic of the three. It is caused by the fact that a feasible transformation is not unique even when the ambiguity caused by permutations and scaling is excluded. The rotational ambiguity is limited by the constraint of nonnegativity of the concentration and absorption matrices.

2.1.5 Target function

The limitations of the transformation matrix T can be expressed with the help of a target function.

Definition 2. *The target function f for the polygon inflation algorithm is defined as*

$$f(T) = \begin{pmatrix} \tilde{C} \\ \tilde{A} \\ \tilde{R} \end{pmatrix}, \; \tilde{C} = \begin{pmatrix} \min\left(0, \frac{C_{11}}{\|C_{:,1}\|_\infty} + \varepsilon\right) \\ \vdots \\ \min\left(0, \frac{C_{k1}}{\|C_{:,1}\|_\infty} + \varepsilon\right) \\ \vdots \\ \min\left(0, \frac{C_{1s}}{\|C_{:,s}\|_\infty} + \varepsilon\right) \\ \vdots \\ \min\left(0, \frac{C_{ks}}{\|C_{:,s}\|_\infty} + \varepsilon\right) \end{pmatrix}, \; \tilde{A} = \begin{pmatrix} \min\left(0, \frac{A_{11}}{\|A_{1,:}\|_\infty} + \varepsilon\right) \\ \vdots \\ \min\left(0, \frac{A_{1n}}{\|A_{1,:}\|_\infty} + \varepsilon\right) \\ \vdots \\ \min\left(0, \frac{A_{s1}}{\|A_{s,:}\|_\infty} + \varepsilon\right) \\ \vdots \\ \min\left(0, \frac{A_{sn}}{\|A_{s,:}\|_\infty} + \varepsilon\right) \end{pmatrix}, \; \tilde{R} = \begin{pmatrix} R_{11} \\ R_{21} \\ \vdots \\ R_{(s-1)s} \\ R_{ss} \end{pmatrix},$$

where $R = Id(s) - TT^+$ and both A and C are defined as before. The symbol $Id(s)$ stands for an $s \times s$ identity matrix and T^+ for the pseudo-inverse of matrix T.

The target function is to be minimized according to

$$\min_{T \in \mathbb{R}^{s \times s}} \|f(T)\|_2^2 = \min_{T \in \mathbb{R}^{s \times s}} \left(f_1(T)^2 + \cdots + f_{s \cdot k + s \cdot n + s^2}(T)^2\right),$$

and a value equal to zero means that the constraints on the transformation matrix T are satisfied. Some reasoning behind the choice of $f(T)$ is shown below.

The matrix R ensures the regularity of T, owing to the fact that the pseudo-inverse T^+ equals T^{-1} for regular matrices. The pseudo-inverse of T is described in [6] as the solution to

$$\min_{X \in \mathbb{R}^{s \times s}} \|TX - \mathrm{Id}(s)\|_F$$

and that corresponds to $\|\tilde{R}\|_2$. Therefore, a very small value of $\|\tilde{R}\|_2$ means that $T^+ \approx T^{-1}$ because numerical rounding-error can cause $\|\hat{R}\|_2 > 0$. In the case of this value being near the machine epsilon, matrix T can be assumed to be regular.

For the vectors $\tilde{A}$ and $\tilde{C}$ a minimum of the actual value and zero is used to eliminate the influence of positive values on the decision, whether the matrices have negative entries. The parameter ε serves the purpose of stabilizing the algorithm by accepting small negative entries, see [14]. The scaling of the elements with the maximal absorptivity (in A) or concentration (in C) of the pure component allows the parameter ε to be universal for different data sets.

2.2 The area of feasible solutions

To obtain the two-dimensional representation of the AFS, the previously described s-component systems are narrowed down to three-component systems. Thus, $s = 3$ and $\mathrm{rank}(D) = 3$ are used in the following chapters. However, these ideas can be generalized, see e.g. [14].

2.2.1 The scaling of the transformation matrix

A feasible transformation matrix $T \in \mathbb{R}^{3\times 3}$ for a three-component system has nine degrees of freedom. It is possible to reduce that to six degrees of freedom by scaling the first column.

From

$$\begin{pmatrix} A_{11} & A_{12} & \dots & A_{1n} \\ A_{21} & A_{22} & \dots & A_{2n} \\ A_{31} & A_{32} & \dots & A_{3n} \end{pmatrix} = \begin{pmatrix} T_{11} & T_{12} & T_{13} \\ T_{21} & T_{22} & T_{23} \\ T_{31} & T_{32} & T_{33} \end{pmatrix} \begin{pmatrix} \hat{V}_{11} & \hat{V}_{21} & \dots & \hat{V}_{n1} \\ \hat{V}_{12} & \hat{V}_{22} & \dots & \hat{V}_{n2} \\ \hat{V}_{13} & \hat{V}_{23} & \dots & \hat{V}_{n3} \end{pmatrix}$$

follows, that

$$\begin{aligned} A_{1,:} &= T_{11}\hat{V}_{:,1} + T_{12}\hat{V}_{:,2} + T_{13}\hat{V}_{:,3} \\ A_{2,:} &= T_{21}\hat{V}_{:,1} + T_{22}\hat{V}_{:,2} + T_{23}\hat{V}_{:,3} \\ A_{3,:} &= T_{31}\hat{V}_{:,1} + T_{32}\hat{V}_{:,2} + T_{33}\hat{V}_{:,3}. \end{aligned}$$

The scaling of the first row $T_{:,1}$ with the element from the first column T_{11} can be interpreted as multiplying the equation 2.1 with a matrix $L = \mathrm{diag}(T_{11}^{-1}, 1, 1)$ and its inverse

$$D = \hat{U}\hat{\Sigma}T^{-1}L^{-1}LT\hat{V}^T,$$

with LT being the new feasible transformation matrix. This can be done with all three rows.

Clearly, the scaling of the matrix T is allowed if and only if $T_{i1} \neq 0,\ \forall i \in \{1,2,3\}$ because the inverse T_{i1}^{-1} of zero is not defined.

Let's assume that the first column of the matrix T does not contain any zeros. Following the naming convention in [16], the problem now looks like,

$$\begin{aligned} A_{1,:} &= \hat{V}_{:,1} + \alpha\hat{V}_{:,2} + \beta\hat{V}_{:,3} \\ A_{2,:} &= \hat{V}_{:,1} + S_{11}\hat{V}_{:,2} + S_{12}\hat{V}_{:,3} \\ A_{3,:} &= \hat{V}_{:,1} + S_{21}\hat{V}_{:,2} + S_{22}\hat{V}_{:,3}, \end{aligned}$$

with

$$T = \begin{pmatrix} 1 & \alpha & \beta \\ 1 & S_{11} & S_{12} \\ 1 & S_{21} & S_{22} \end{pmatrix}$$

and has six degrees of freedom.

2.2.2 The Perron-Frobenius theorem

The assumption, that the first column of the matrix T does not contain any zeros, was made to carry out the scaling. This is indeed true in the most cases and can be proved with the Perron-Frobenius theorem that discussed in detail in [10].

Firstly, a definition of an irreducible matrix is required.

Definition 3. *A nonnegative matrix $A \in \mathbb{R}^{n\times n},\ n \geq 2$ is called reducible if a permutation matrix P exists, so that*

$$P^T AP = \begin{pmatrix} B & C \\ 0 & D \end{pmatrix},$$

where B and D are square submatrices. If the matrix is not reducible, it is called irreducible.

Since the irreducibility of some matrices needs to be tested, an easily implementable evaluation is useful. Note that here and further in the text the inequality signs (namely, $<, >, \leq, \geq$) of matrices and vectors denote a component-wise inequality.

Theorem 4. *A nonnegative matrix $A \in \mathbb{R}^{n\times n},\ n \geq 2$ is irreducible if and only if $(Id(n) + A)^{n-1} > 0$.*

Proof. See corollary 2.2. in [10]. □

Now a part of the actual Perron-Frobenius theorem follows.

Theorem 5. *An irreducible, nonnegative matrix A has a real positive eigenvalue $\lambda_{\max}$ such that*

$$\lambda_{\max} \geq |\lambda_i|$$

for any eigenvalue λ_i of A. There exists an eigenvector $v_{\max}$ with $v_{\max} > 0$ that corresponds to $\lambda_{\max}$.

Proof. See theorem 4.1 in [10]. □

Returning to the nonnegative matrix factorization problem, a condition for the existence of the scaling matrix L is needed. This is given in the following theorem.

Theorem 6. *If $D^T D$ is irreducible, then $T_{i1} \neq 0,\ \forall i \in \{1, 2, 3\}$ for a feasible transformation matrix T.*

Proof. This proof is based on the theorem 2.2 in [17]. The matrix D is nonnegative per definition. Thus, $D^T D$ is also nonnegative. With the Perron-Frobenius theorem from the irreducibility and nonnegativity of $D^T D$ follows the existence of a positive eigenvector that corresponds to the largest eigenvalue of $D^T D$.

The connection between eigenvalues and singular values is as follows: the eigenvalues of $D^T D$ and DD^T are the squared singular values of D. The eigenvectors of $D^T D$ and DD^T are the right and left singular vectors of D respectively. The largest singular value is located in Σ_{11}, and the corresponding singular vectors, in the first columns of U and V.

Therefore, the singular vector, corresponding to the largest singular value and, in extension, the largest eigenvalue of $D^T D$, can be assumed to be component-wise positive, see also the remark after the proof. This singular vector is $\hat{V}_{:,1}$, as it is located in the first column of $\hat{V}$. Moreover, singular vectors are orthogonal to each other because V is an orthogonal matrix.

This means that no vector $(0, \alpha, \beta) \in \mathbb{R}^3 \backslash (0,0,0)$ exists with $(0, \alpha, \beta)\hat{V}^T = A_{s,:}$ being a nonnegative vector.

Let's assume the opposite. Then $(0, \alpha, \beta)\hat{V}^T \geq 0$ holds true. It is also known that $\hat{V}_{:,2}$ and $\hat{V}_{:,3}$ are orthogonal. Consequently, $(c_1\hat{V}_{:,2} + c_2\hat{V}_{:,3})\hat{V}_{:,2}^T = c_1\hat{V}_{:,2}\hat{V}_{:,2}^T = c_1$ equals 0 only for $c_1 = 0$ and similarly $(c_1\hat{V}_{:,2} + c_2\hat{V}_{:,3})\hat{V}_{:,3}^T$ equals 0 only for $c_2 = 0$; hence, from $c_1\hat{V}_{:,2} + c_2\hat{V}_{:,3} = 0$ follows $c_1 = c_2 = 0$. Thus, $\hat{V}_{:,2}$ and $\hat{V}_{:,3}$ are linearly independent and $(0, \alpha, \beta)\hat{V}^T > 0$. From that follows $(0, \alpha, \beta)\hat{V}^T\hat{V}_{:,1} > 0$ because $\hat{V}_{:,1} > 0$. This is a contradiction to the orthogonality of V, since

$$(0, \alpha, \beta)\hat{V}^T\hat{V}_{:,1} = (0, \alpha, \beta)(1,0,0)^T = 0.$$

The assumption must be wrong, and no vector $(0, \alpha, \beta) \in \mathbb{R}^3 \backslash (0,0,0)$ exists, with $(0, \alpha, \beta)\hat{V}^T = A_{s,:}$ being a nonnegative vector.

Therefore, the first singular vector $\hat{V}_{:,1}$ must influence the nonnegative matrix A, and this means that $T_{i1} \neq 0,\ \forall i \in \{1,2,3\}$. □

Also, a comment on the singular value decomposition is necessary. While the existence of a component-wise positive eigenvector, corresponding to the largest eigenvalue of D^TD, is given by the Perron-Frobenius theorem, left and right singular vectors of D can be calculated component-wise negative. This can be easily fixed by multiplying both of them with -1, as that can be interpreted as scaling.

The irreducibility of DD^T and D^TD can be assumed for spectroscopic applications, see [17]; therefore, it does not restrict practical applications of this work. The irreducibility of DD^T guarantees the positivity of the left singular vector and can be used for a similar proof to the previous theorem. It is also required for the proof of the theorem 2.5 in [17] that is used further in this work.

2.2.3 The two-dimensional representation

In this section the original nonnegative factorization problem is finally reduced to finding the area of feasible solutions (AFS). For a three-component system, it can be projected on a plane.

Definition 4. *A solution $(\alpha, \beta) \in \mathbb{R}^2$ is called feasible if and only if $\exists S \in \mathbb{R}^{2\times 2}$ such that*

$$T = \begin{pmatrix} 1 & \alpha & \beta \\ 1 & S_{11} & S_{12} \\ 1 & S_{21} & S_{22} \end{pmatrix}$$

is regular, $C \geq 0$, and $A \geq 0$ for $C = \hat{U}\hat{\Sigma}T^{-1}$ and $A = T\hat{V}^T$.

This can be continued by defining the set of all feasible solutions.

Definition 5. *The area of feasible solutions is defined as*

$$\mathcal{M} = \left\{(\alpha, \beta) \in \mathbb{R}^2 \ :\ (\alpha, \beta) \text{ is feasible}\right\}$$

in accordance with definition 4.

Thus, the feasible solutions (α, β) can be shown on a two-dimensional plane. The topology of the AFS is discussed in [7].

Theorem 7. *The area of feasible solutions for a nonnegative rank-3 matrix $D \in \mathbb{R}^{k \times n}$ is empty or consists of $3m$, $m \in \mathbb{N}$ segments.*

Proof. See theorem 7.3. in [7] □

A useful constraint to AFS, called FIRPOL, is given by Borgen and Kowalski in [2]. It is based upon the nonnegativity of the first row of A and can be applied as a limitation to the AFS.

Definition 6. *The polygon FIRPOL is defined as*

$$\mathcal{F} = \left\{ (\alpha, \beta) \in \mathbb{R}^2 \ : \ (1, \alpha, \beta)\hat{V}^T \geq 0 \right\}.$$

It follows that $\mathcal{M} \subset \mathcal{F}$ and $\mathcal{M} \neq \mathcal{F}$ because $(0,0) \notin \mathcal{M}$, but $(0,0) \in \mathcal{F}$, see theorem 2.5 in [17].

To fix the orientation of the AFS, the second and the third singular vectors of the singular value decomposition are multiplied by -1 under some circumstances. That influences the orientation regarding the α and β axes, see [16], e.g. multiplying the second singular vector by -1 corresponds to reflection over the β-axis.

If the maximum of the column $\hat{V}_{:,i},\ i = 1, 2, 3$ is smaller than the maximum of the column $-\hat{V}_{:,i}$, then columns $\hat{V}_{:,i}$ and $\hat{U}_{:,i}$ are multiplied by -1. This ensures that the element of a column of $\hat{V}$ with the absolute maximum is positive; thus, it can also be done to the first column. This is referred to as the normal orientation. The changes in orientation for the two-dimensional representations are shown in figure 2.1.

2.2.4 The feasibility of a solution

The focus of this section is to explain how the decision is made whether $(\alpha, \beta) \in \mathcal{M}$.

The first step should always be to check whether $(\alpha, \beta) \in \mathcal{F}$. This check is a simple multiplication and is not computationally intensive. If a solution does not belong to FIRPOL, then it can be labeled as not feasible because $\mathcal{M} \subset \mathcal{F}$.

If the solution belongs to FIRPOL, a much more time-consuming test is required. The difficulty is in finding a suitable $S \in \mathbb{R}^{2\times 2}$ for given values (α, β). The decision can be regarded as a nonlinear least-squares problem of two known data points and four adjustable parameters. The target function f from chapter 2.1.5 with $s = 3$ is used for this decision.

The result of the minimization problem is the basis for deciding whether the solution is feasible. In general, rounding-errors are present. Therefore, all solutions with

$$\min_{S \in \mathbb{R}^{2\times 2}} \|f(\alpha, \beta, S)\|_2^2 \leq \varepsilon_{tol},$$

for a small parameter ε_{tol} are considered to be included in AFS. This is referred to as the standard test.

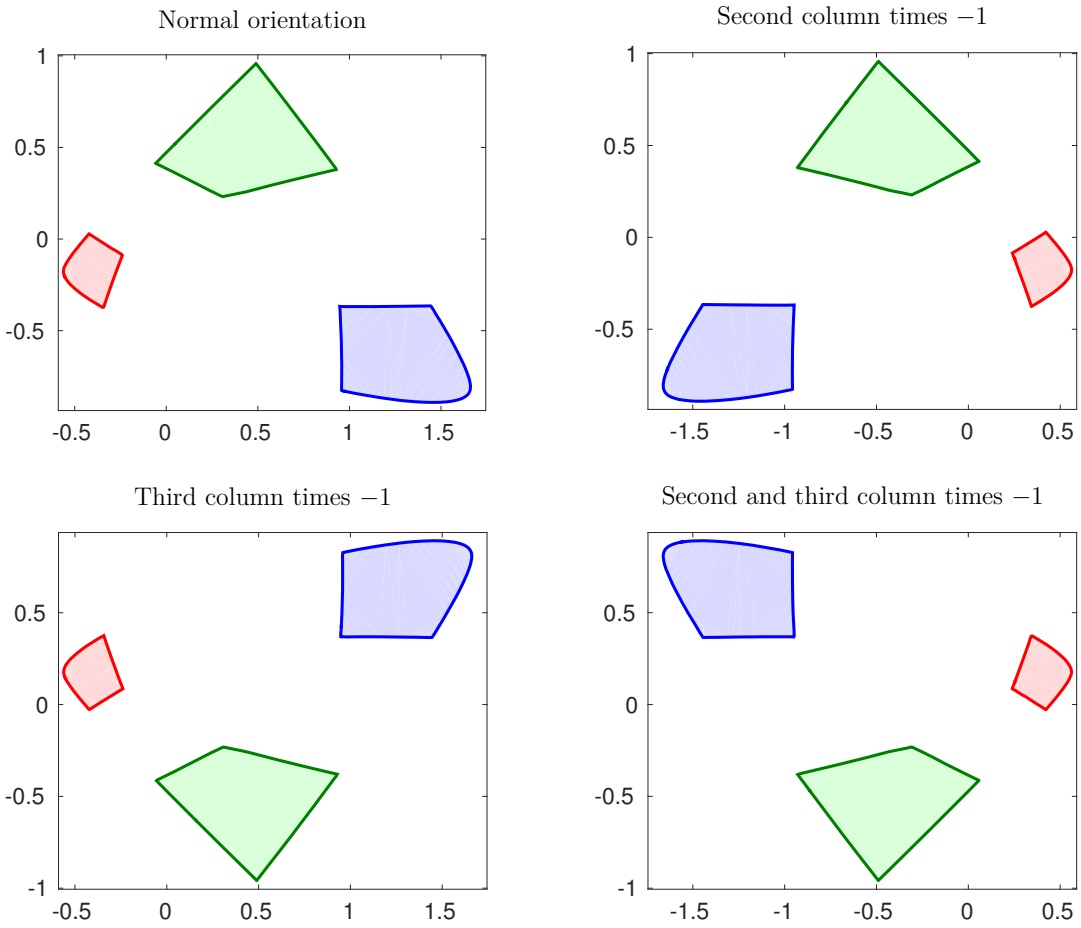

Figure 2.1: The AFS of the FACPACK data set `example2.mat` with different orientations, depending on the columns of $\hat{V}$ and $\hat{U}$.

Note that similar treatment of small negative entries is advised in [14] for the FIRPOL. Thus, the solution belongs to FIRPOL if

$$\min\left(0, \frac{(1,\alpha,\beta)\hat{V}^T}{\|(1,\alpha,\beta)\hat{V}^T\|_\infty} + \varepsilon\right) \geq 0.$$

However, this would mean that the standard test and FIRPOL have different tolerances and a solution, that is acceptable by the standard test, might not be feasible according to FIRPOL. To circumvent this, a slightly modified decision for FIRPOL is used

$$\sum_{i=1}^{n}\left(\min\left(0, \left(\frac{(1,\alpha,\beta)\hat{V}^T}{\|(1,\alpha,\beta)\hat{V}^T\|_\infty} + \varepsilon\right)_{1i}\right)^2\right) \leq \varepsilon_{tol};$$

it provides an equivalent statement to the standard test.

3 Polygon inflation algorithm

In this chapter the polygon inflation algorithm for finding the AFS is introduced. The general idea of the algorithm is explained; mainly [16] and [14] are used as sources. Then some details and difficulties of the implementation are presented.

3.1 General concepts of the algorithm

The general concept of the polygon inflation algorithm is to approximate the area of feasible solutions with a polygon that is expanded by adding new vertices (α, β). The polygon inflation algorithm is intended to be used for the computation of an AFS that consists of exactly three disconnected segments. For other cases alternative methods should be used, e.g., the inverse polygon inflation, see chapter 4.

3.1.1 Adding a new vertex

An essential part of the polygon inflation algorithm is the search for new vertices. A new vertex is added by means of two stages - a pre-bisection stage and a bisection stage.

The pre-bisection stage roughly approximates the boundary of the AFS. The search for a new vertex originates from the midpoint of an edge that is marked for refinement and is realized perpendicularly to the edge. If the midpoint is located in the AFS, then the search for the new vertex is started outwards (away from the polygon) and if the midpoint is not a feasible solution, then the search is started inwards (to the interior of the polygon).

Then some steps are made. After each step, the feasibility of the solution is checked. When two such solutions are found, where one of them lies in the AFS and the other does not, an approximate location of the boundary is detected.

The bisection stage is used to refine the location of the boundary. The use of bisection method is suggested in [16] and it is described in e.g. [3].

For the bisection method, two solutions are required - one of them is feasible (let's call it p_i) and the other one is not (let's call it $\hat{p}_i$). The iteration follows by halving the interval between the two points and determining whether the new point p_{new} is feasible. If it is, then $p_{i+1} = p_{\text{new}}$ and $\hat{p}_{i+1} = \hat{p}_i$, else $\hat{p}_{i+1} = p_{\text{new}}$ and $p_{i+1} = p_i$. This is continued until $\|\hat{p}_i - p_i\| \leq \varepsilon_b$.

After the termination of the bisection method, the new vertex of the polygon is set to be p_i because it has to be feasible and $\hat{p}_i$ is not.

3.1.2 Adaptivity and termination

The subdivision of the edges could simply be done in a set order until a defined number of vertices is reached. However, this approach would require a large amount of vertices because sharp corners of the AFS would be approximated in the same fashion as straight fragments. This is solved by subdividing edges adaptively, see figure 3.1.

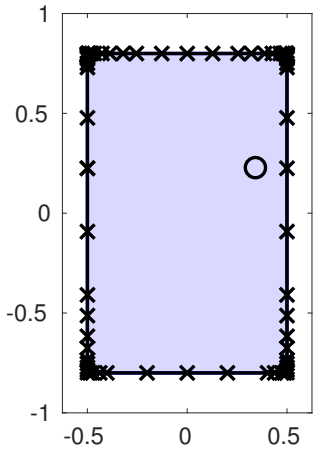

Figure 3.1: A rectangle has been approximated with 73 vertices. The symbol ∘ denotes the initial point (compare with figure 3.4) and ×, the found vertices.

Thus, for each edge i some value Δ_i, that is responsible for the decision which edge is to be subdivided next and when the adaptive iteration should be stopped, is necessary.

In [16] it is suggested to use the gain-of-area for this purpose, defined as

$$\hat{\Delta}_i = \frac{1}{4}\|P_i - P_{i+1}\|_2\|M - P_{new}\|_2,$$

where P_i and P_{i+1} are two successive vertices, M is the midpoint between them and P_{new} is the vertex that was added. Hence, the edge, which previously increased the area of the polygon the most, will be subdivided again. This would, for example, stop straight fragments of the AFS being unnecessary subdivided.

However, using $\hat{\Delta}_i$ failed to provide the necessary adaptivity, see figure 3.2. This can be explained by the dependency of the length of the old edge through $\|P_i - P_{i+1}\|_2$. This halts the subdivision of very short edges that are even significantly different from the previous edge. Using only the height of the isosceles triangle (formed by the old edge as a base and two new edges as legs), as done in FACPACK, circumvents this problem, see figure 3.2. Thus,

$$\Delta_i = \|M - P_{new}\|_2$$

is used.

The edge, chosen to be divided during the polygon inflation, is defined as having the

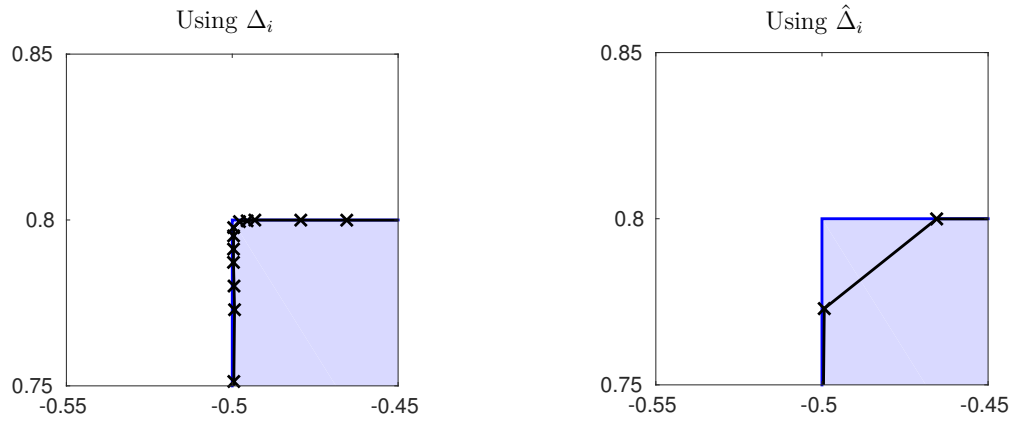

Figure 3.2: A close-up view of the upper left corner of the rectangle in 3.1 using gain-of-height and gain-of-area. For Δ_i 73 steps were taken and for $\hat{\Delta}_i$ only 45 steps. The other parameters were set for both cases as $\varepsilon_b = 10^{-3}$, $\delta = 10^{-10}$ and `par(5)=1e-1`.

maximal Δ_i. Note that the two most recent edges will have identical Δ_i; in this case one of them is to be divided first.

The polygon inflation algorithm is terminated when $\max_i \Delta_i$ drops below a predefined parameter δ. This parameter allows to change the accuracy of the resulting polygon because a lower δ results in more vertices. Hence, a higher refinement follows, see chapter 5.3.

An interesting characteristic is the lack of adaptivity in case of a lucky guess. See figure 3.3 for an example. For this particular triangle, it would happen with any arbitrary initial point, as it is an isosceles triangle. This is connected with the initialization that is explained in the next section. The first two vertices create an edge, that is parallel to the base of the triangle. Thus, the mid-perpendicular of this edge will intersect the apex of the triangle. The following subdivisions result with $\Delta_i = 0$ (or some small numerical error) for both edges adjacent to the apex. The reason is that they coincide with the legs of the triangle. Consequently, the edges near the apex of the triangle will not be subdivided anymore.

3.1.3 The initialization

To start the polygon inflation algorithm, an initial polygon has to be constructed. In [16] an initial triangle is used. However, due to the lack of the gain-of-height values Δ_i for the initial triangle, another three steps are needed before the main iteration can start; it results in a hexagon. To reduce the initiation phase a quadrilateral, with Δ_i calculated for the third and fourth vertex, is used in this work. This change has minimal influence on the resulting polygon.

The construction of the quadrilateral is started with an initial point in the AFS. The standard feasibility test delivers an opportunity to find initial points by changing the

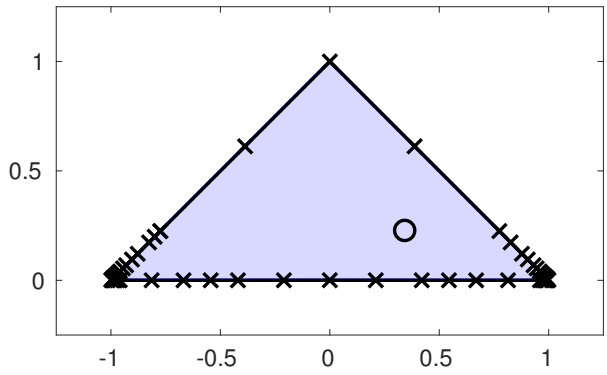

Figure 3.3: The triangle has been approximated with 57 vertices. Note the lack of adaptivity at the apex.

minimization problem to

$$\min_{\alpha\in\mathbb{R},\ \beta\in\mathbb{R},\ S\in\mathbb{R}^{2\times 2}} \|f(\alpha,\beta,S)\|_2^2 \leq \varepsilon_{tol}.$$

A successful minimization provides an initial solution (α, β) and the corresponding matrix S for one segment of the AFS.

The initial points for other segments can be found in remaining rows of the matrix T from the minimization problem, more precisely

$$\begin{pmatrix} 1 & S_{11} & S_{12} \\ 1 & \alpha & \beta \\ 1 & S_{21} & S_{22} \end{pmatrix} \text{ and } \begin{pmatrix} 1 & S_{11} & S_{12} \\ 1 & S_{21} & S_{22} \\ 1 & \alpha & \beta \end{pmatrix},$$

and can be analogously checked for feasibility. This is the case because it is possible to expand the equation 2.1 with a permutation matrix $P \in \mathbb{R}^{3\times 3}$ to

$$D = \hat{U}\hat{\Sigma}\hat{V}^T = \underbrace{\hat{U}\hat{\Sigma}(T^{-1}P^T}_{\text{C}}\underbrace{PT)\hat{V}^T}_{\text{A}}.$$

The original factors A and C as well as the factors A_0 and C_0 that are computed during the initialization are compared in figure 3.5. This also means that the sequence in which the three polygons will be processed depends on the factor A_0 and can vary.

An initial matrix factorization is accepted if all three initial solutions are feasible. Otherwise, it is repeated with different random number generation.

Then a construction of an initial quadrilateral can follow by finding four vertices on the boundary of the AFS, see figure 3.4. The searching directions in the pre-bisection stage for the first two vertices are manually set beforehand. The third and fourth vertex is found by subdividing the first edge in opposite directions.

Then the polygon inflation can proceed normally. The segments are processed consecutively.

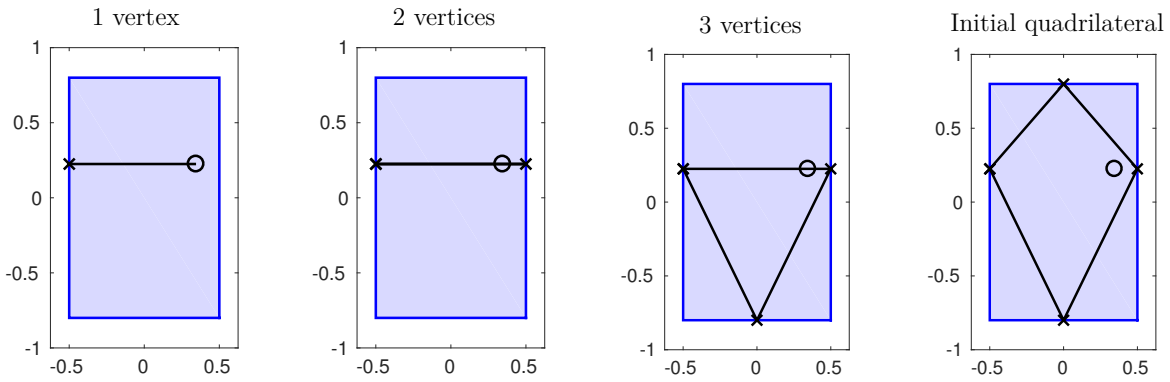

Figure 3.4: The construction of an initial quadrilateral for the approximation of the boundary of a rectangle.

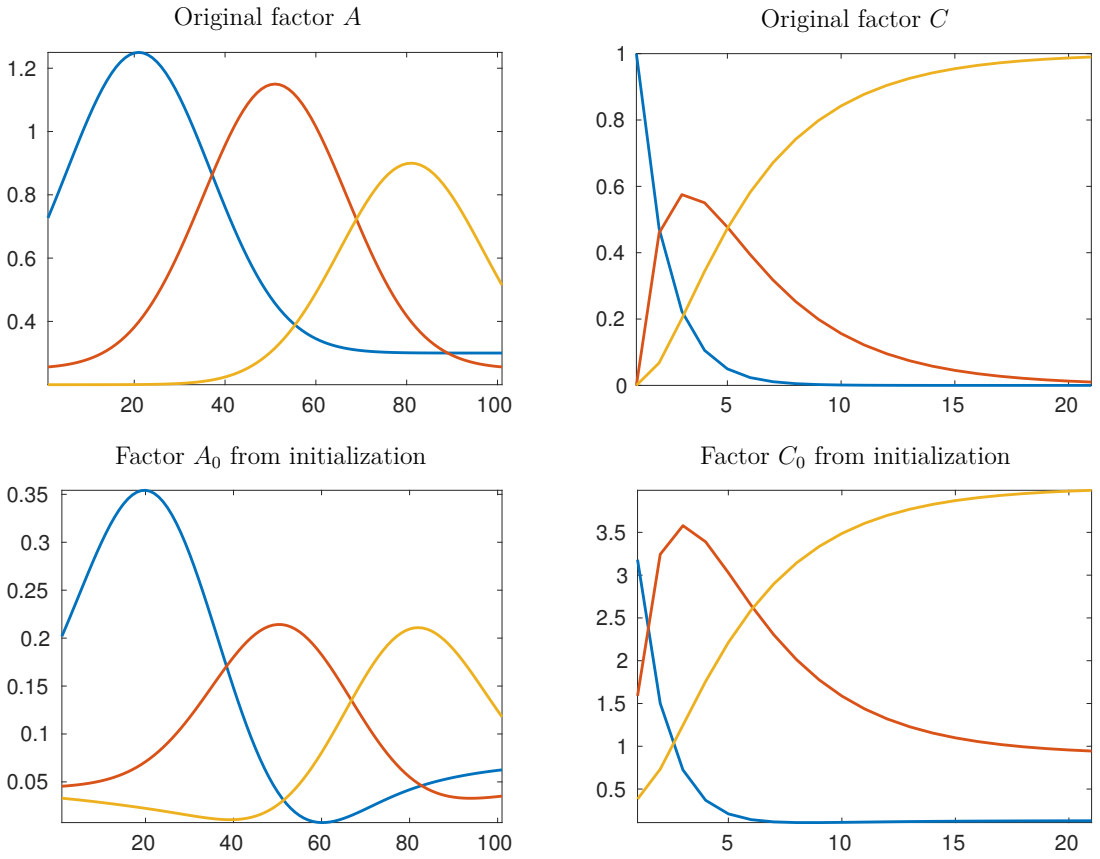

Figure 3.5: Pure component concentration profiles and spectra for the FACPACK data set `example2.mat`, compared with the calculations during the initialization of the algorithm.

3.2 Challenges regarding the numerical optimization

A successful implementation of the adaptive polygon inflation algorithm depends on a diverse spectrum of strategies. Most challenges are associated with flawed results of numerical optimization. Therefore, choosing correct techniques and appropriate parameters is of considerable importance.

3.2.1 The numerical optimization in Matlab

To check the feasibility of a solution, the numerical optimization problem

$$\min_{S \in \mathbb{R}^{2\times 2}} \|f(\alpha, \beta, S)\|_2^2$$

has to be solved. A suitable algorithm for this problem in MATLAB is `lsqnonlin`.

This algorithm requires a starting approximation $S \in \mathbb{R}^{2\times 2}$ and the success of the minimization relies partially on it. Therefore, a poorly chosen matrix S can lead to a solution mistakenly labeled as not feasible. To reduce this possibility, the following strategy is adopted.

For the midpoint of an edge in the pre-bisection stage, the optimization problem should be solved in three steps. Firstly, using `lsqnonlin` with S from one of the both neighboring vertices. If it fails the feasibility test, then it has to be repeated with S from the other neighboring vertex. This ensures that both of the best available starting approximations are used. If it also fails, then an additional test using the genetic algorithm `ga` in MATLAB can be done. The genetic algorithm requires no starting approximation but is slower and delivers less precise solutions. However, it can be used in combination with `lsqnonlin` to obtain a precise feasible solution. If the test with the genetic algorithm also delivers a value that is larger than ε_{tol}, then it is to be assumed, that the solution is not feasible.

If the midpoint was feasible, then each step in the pre-bisection stage can use the matrix S from the previous step. The matrix S from the midpoint can be used for the first step. This is done because the closest possible solution is the most likely to have a similar S to the examined solution and it can be used as a good starting approximation.

The case where the midpoint was not feasible is more complicated. The starting matrix S from the midpoint can be a very poor approximation for the actual matrix. Thus, in each step inwards in pre-bisection stage, a multitude of starting matrices can and should be checked - the matrix from previous approximation, as well as both matrices from neighboring vertices. Since the correct acknowledgment, when a solution is feasible, is critical for this stage, even the genetic algorithm can be used in each step. However, the calculation of it is computationally expensive and, in the case of an infinite loop occurring due to other errors, it means a large computing time leading to no solution. Therefore, the genetic algorithm has not been used for this purpose.

In the bisection stage, both matrices S from endpoints of the halved interval are tested as initial approximations. Only if the new point fails the test with both of them, it is assumed to be not feasible.

Even after such precautions, errors regarding numerical optimization can occur.

3.2.2 Possible errors

Some solutions can be mistakenly labeled as not feasible due to a poor starting matrix S or an unsuccessful minimization. There is no possibility of the opposite because if a minimum, smaller than ε_{tol}, can be found, then the definition of a feasible solution is met. Owing to the fact that ε and ε_{tol} tolerate small negative entries, rounding errors and inaccurate data, solutions near the border of the AFS can also be accepted as feasible; however, this behavior can be influenced by changing these parameters.

Three cases, where the minimum, lower than ε_{tol}, could not be found by an error, are presented.

Case 1. A solution can be mistakenly labeled not feasible while testing the midpoint of two vertices in the pre-bisection stage. Consequently, the following search of the boundary will be started in the incorrect direction - inwards. In the best case, a feasible solution is found while still within the correct AFS segment. However, an infinite loop of steps in the wrong direction can occur when the algorithm fails to find the minimum for the entirety of the AFS segment. In the worst case, a feasible solution in a different AFS segment will be found. This is a rare possibility, as the given S from one of the neighboring vertices is unlikely to be a good approximation for a point in a different segment.

Case 2. A further possible circumstance, when a solution can be mistakenly labeled as not feasible, is during the search for the boundary in the pre-bisection stage. This results in a vertex that is located further in the interior of the AFS segment than it should be. This is not only a poor approximation of the boundary but also can lead to the main iteration never stopping. The erroneous vertex will end up having a large Δ_i and will be chosen for subdivision. Normally the subdivision will run smoothly and end up in a vertex near the boundary. Thus, Δ_i will stay large and the edge with the incorrect vertex will again be subdivided. The outcome is a polygon that wraps around the AFS segment multiple times (disregarding the inaccurate vertices). This can be observed in figure 3.6.

Case 3. Finally, a solution can be mistakenly found not feasible during the bisection stage. This results in a similar problem as in the previous case. Here the greatest challenge is an uneven boundary. The search for new vertices is always done perpendicular to already found edges. If the boundary is too ragged, then the direction in the pre-bisection stage can be shifted and that can lead to a vertex in the wrong position. In this case, the numbering of vertices in the polygon loses its meaning regarding the direction and all following vertices from the wrong edge will be searched in the incorrect direction.

One can infer from these cases alone that a multitude of techniques is required to detect and correct such errors. The results with no error correction can be seen in figure 3.6.

3.2.3 Solutions

The first approach to this problem was to label two vertices as incorrect if they are closer to each other than to their neighbors and to delete both. However, this could result in

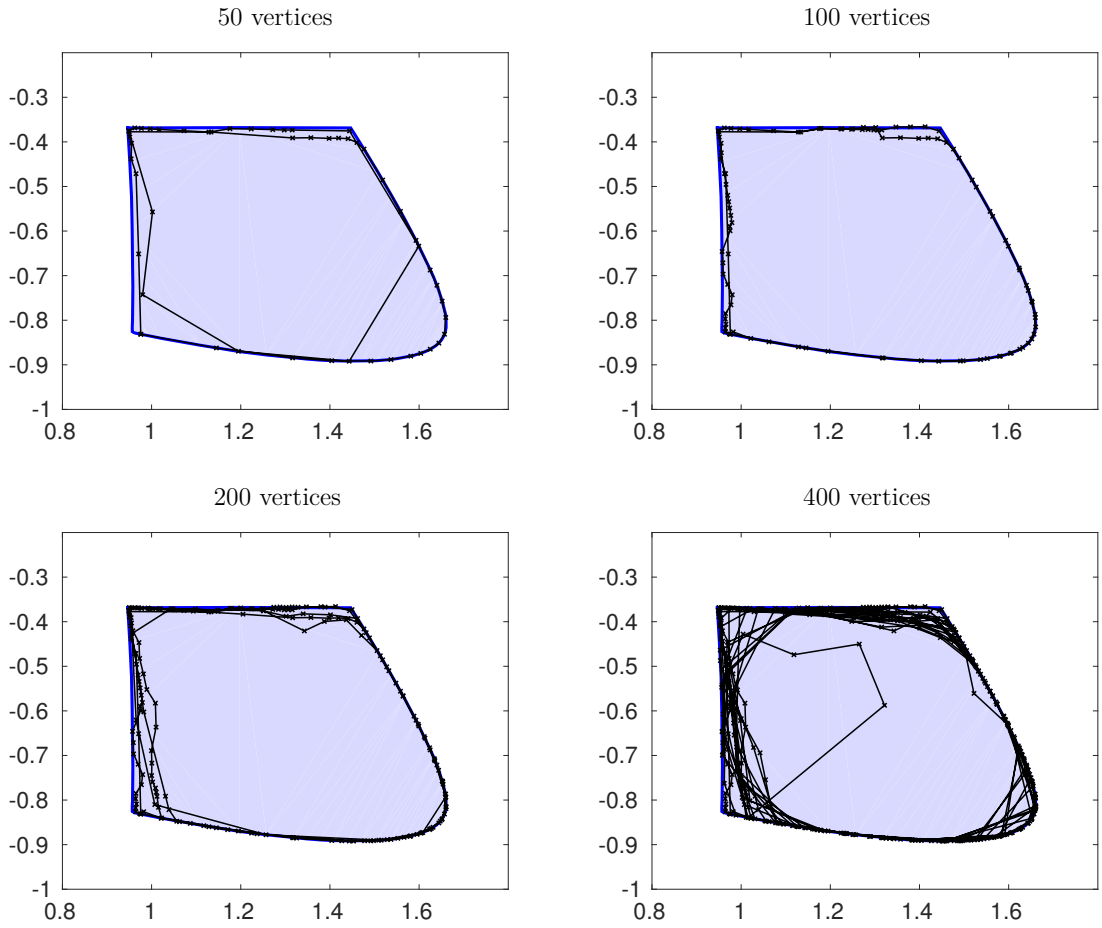

Figure 3.6: The polygon of FACPACK data set `example2.mat` without any removal of incorrect vertices by angle. The stopping criterion was not met after calculation of 400 vertices. The default options of `lsqnonlin` were used to provide a better example. The reference in blue for the AFS segment has been computed using FACPACK.

the removal of all vertices and it can be problematic in sharp edges. This approach is inherently poor because it only detects errors after the vertices have been accepted and only uses distance, instead of direction; it can be improved by using angles.

No unfeasible vertices can be accepted. Therefore, small interior angles of the polygon can be treated as a better approximation of boundary. Large interior angles are to be resolved because they are an indication of a vertex that is accepted far in the interior of the AFS. Which angles should be tolerated can be determined by a maximal parameter α. It can also be done adaptively, depending on the total number of vertices.

The angle test should already be done at the moment when a new vertex has been found. If the interior angle is larger than the parameter α, then the vertex should not be accepted. One can assume that it is too far from the border. A follow-up strategy is to subdivide the the old edge in a different location and try again. Possible subdivisions $\left\{\frac{1}{3}, \frac{2}{3}, \frac{1}{4}, \frac{3}{4}\right\}$, where $\frac{1}{2}$ is the midpoint, are used.

Additionally, an inspection of all vertices should follow each successful introduction of a new vertex, as the newly added vertex could prove to be a much better approximation of the boundary.

An infinite loop in the pre-bisection stage should not be allowed. If a predetermined value of steps is exceeded, then the search is stopped and the algorithm continued similarly to the case, when a vertex is not accepted by the angle.

Finally, a solution to the last problem in the case 1 is presented. If a vertex in a different segment is indeed accepted, then the next subdivision will inevitably fail for all possible subdivisions (either due to an infinite loop during the search for the boundary, or due to the angle parameter). Thus, one of the neighboring vertices must be wrong and should be deleted. If the wrong vertex was not deleted, then the subdivision of the edge will fail again and the wrong vertex will be removed in the next step. This also solves other situations when all defined subdivisions have been tried and still no new vertex has been found.

The problem associated with deleting vertices is how to choose the new Δ_i. It is set as the maximum of both old values Δ_i to avoid stopping the iteration sooner as expected. Furthermore, the deletion of all vertices can be a problem, hence the parameters have to be carefully considered.

3.3 Challenges regarding the initialization

The quality of the initialization is arguably one of the most important parts of the algorithm. There is little information about the AFS, there are not enough vertices to discard any, an unsuccessful search for a vertex cannot be easily mitigated and any erroneous vertex can negatively influence the following subdivisions.

3.3.1 A different approach

It can be challenging to find feasible initial solutions. The first approach that was tested did not provide the necessary stability. Let's call this strategy the nnmf initialization, as it is based on the nonnegative matrix factorization algorithm `nnmf` in MATLAB.

The initial nonnegative factorization of $D = C_0 A_0$ from `nnmf` is used to determine the matrix $T_0 = A_0 \hat{V}$. Scaling is also necessary to ensure that the first column of T_0 consists of 1. Therefore, the transformation matrix T possesses the composition

$$T = \begin{pmatrix} 1 & T_{12} & T_{13} \\ 1 & T_{22} & T_{23} \\ 1 & T_{32} & T_{33} \end{pmatrix},$$

and the solutions (T_{12}, T_{13}), (T_{22}, T_{23}) and (T_{32}, T_{33}) can be tested for feasibility. The elements from the other rows of T are already an optimal starting point for the optimization problem, as they have been obtained directly through the nonnegative factorization. The solutions are not automatically feasible because of the inaccuracy of the factorization $D \approx C_0 A_0$. The nonnegative factorization might need to be repeated until an admissible starting point has been found.

From 1000 tests with `nnmf`, using `'algorithm','mult'` and starting the random number generation with `rng('default')`, the average value of $\text{norm}(D - C_0 A_0)$ was 0.505 and the minimal 0.125. These values indicate that the factorization is poor and that no successful starting matrix was found. The example data set `linesegment.mat` from FACPACK was tested. In comparison, a successful starting matrix for the data set `example2.mat` was found after 8 steps. It was accepted with $\text{norm}(D - C_0 A_0) = 0.140$.

Therefore, this approach was abandoned in favor of a different initialization. It is described in chapter 3.1.3; the minimization problem is solved with `ga` and `lsqnonlin`. Let's call it the ga-lsqnonlin initialization.

The ga-lsqnonlin initialization also might need multiple iterations; for `linesegment.mat` it required three iterations. The values of $\text{norm}(D - C_0 A_0)$ were 0.022, 0.001 and 0.083, with the last corresponding to a feasible initial matrix. This means that the solution could be still improved.

It should be noted that the nnmf initialization is a fitting way to forgo these difficulties for constructed problems. The matrices C and A are known in this case. Therefore, $T_0 = A\hat{V}$ will always provide a feasible solution. This can be successfully used for the examination of the subsequent parts of the algorithm.

3.3.2 The initial quadrilateral

The construction of the initial quadrilateral can also be subject to some complications.

A notable issue occurs when the initial point is near the edge of the AFS and the first two vertices have a small distance between them. Thus, it might not be possible to construct one of the following vertices or the algorithm does not detect the largest part of the AFS due to labeling interior points as not feasible. So first two vertices being closer than the parameter d_{init} are disallowed. This is also important for an AFS consisting of a dot set, see chapter 5.8.

Even when first two vertices are valid, an infinite loop in the pre-bisection stage can occur. A possible explanation is that no neighboring vertex has a good starting approximation S for a new vertex, when they are far apart. A subdivision resulting in an infinite loop is disallowed here since the initial quadrilateral should consist of four as fitting vertices as possible.

If the construction of the initial quadrilateral by some reason fails, then it has to be repeated with a different initial direction vector. The direction vector can be rotated by some parameter θ. Under normal circumstances, this can overcome the major problems of the initialization.

If one of the AFS segments is a dot set, then no initial quadrilateral can be constructed. Therefore, some restriction on rotation should be implemented to recognize this case. For example, if the direction vector is rotated close enough to the original direction vector, then the search for the initial quadrilateral is terminated.

3.4 Other challenges

3.4.1 Segments in close proximity

The adaptive polygon algorithm presents a problem in a case of two segments located close to one another.

If the step size in the pre-bisection stage is chosen too large, then the previous point can be located in one set and the next point in the other. In this case, the correct boundary would not be found.

Most likely the polygon would contain the correct boundary of both segments; however, a polygon, wrapping around the boundary multiple times, is possible. It can happen that near the closest point of the two segments the boundary is not found at all. In the case of a lucky subdivision, the step can lie between the both sets, resulting in only one found set, which is the correct behavior of the algorithm. For an example,

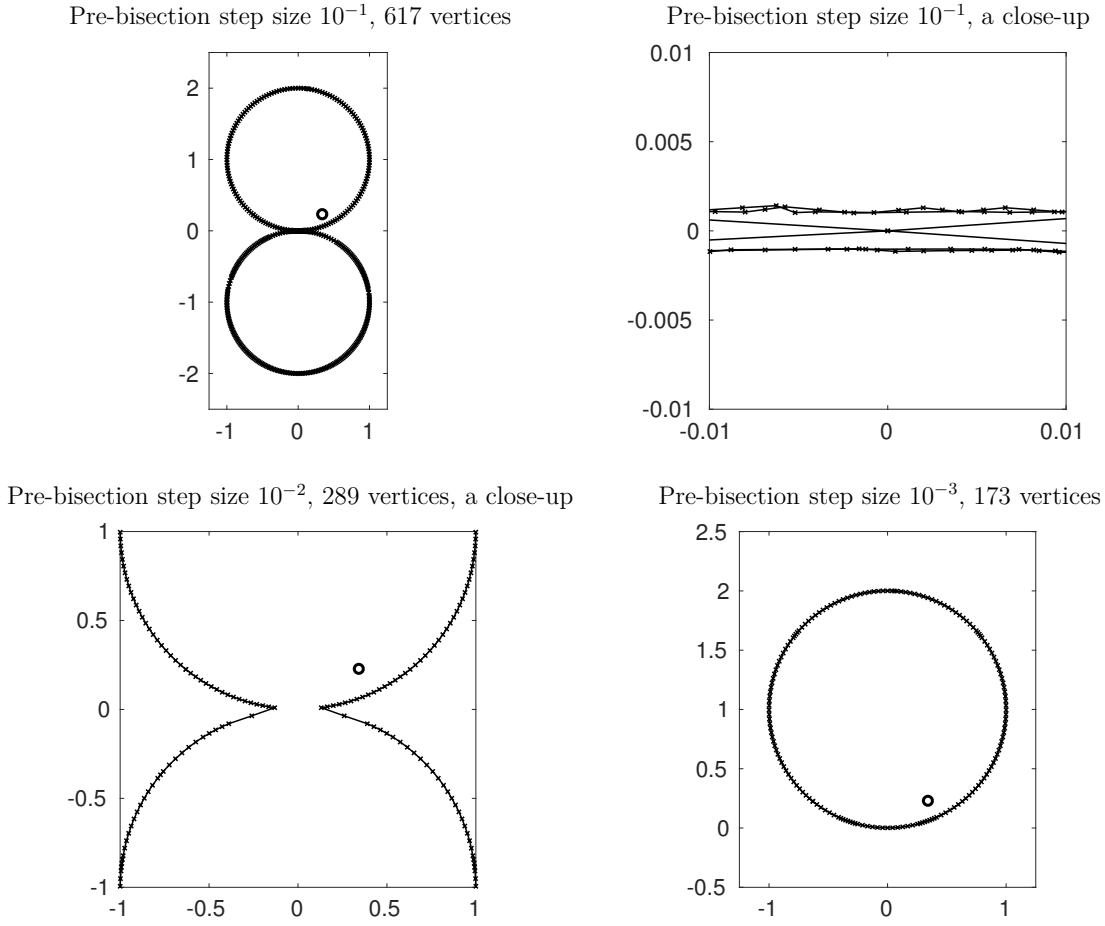

Figure 3.7: The solution of the adaptive polygon algorithm using different pre-bisection step sizes for the problem of two circles; they are located 0.002 from each another at the closest point. Note that the algorithm should not be able to leave the circle with the initial point, marked ∘ .

see figure 3.7.

The solution to this problem would be a reduction of the step size in the pre-bisection stage. However, that leads to an increased number of steps and computing time and should be undertaken only in this particular case. To be absolutely sure that other segments will not be found, the step size in the pre-bisection stage should be less than the smallest distance between the segments. This can be observed in figure 3.7.

Of course, for a real data matrix D it is not known how far apart the sets are located. It is advisable to reduce the pre-bisection step size if the results from the first use are suspicious. It is also possible that the AFS consists of only one segment with a hole; then the inverse polygon inflation algorithm is needed, see chapter 4.

3.4.2 The adaptivity

The adaptivity of the polygon inflation algorithm is one of its strengths. However, a problem can be constructed where this property is troublesome. It can lead to not finding some segments of the AFS.

Let's take the rectangle from the figure 3.1 as an example. No outwards-pointing shapes between the found vertices can be detected with the used precision. This happens because the algorithm has no other way of detecting such characteristics. Thus, it could happen in an unlucky case that some parts of the AFS are not approximated because an assumption of a straight line has been made through adaptivity.

A similar situation occurs when one of the segments is very narrow. If the width of the segment is comparable with the terminating parameter δ, then the iteration could stop prematurely. This leads to a part of AFS that is not approximated, see figure 3.8.

Note, that a different terminating parameter or even a different starting point could

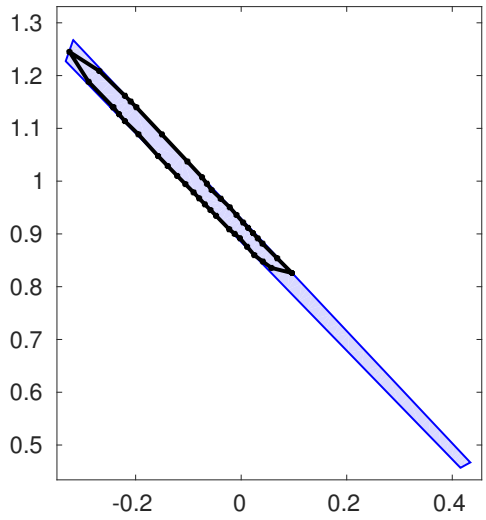

Figure 3.8: The resulting polygon for one segment of the AFS. The iteration with $\delta = \varepsilon_b = 10^{-2}$ (black) has stopped prematurely, when compared to $\delta = \varepsilon_b = 10^{-4}$ (blue). The data set `DotSegment.mat` from FACPACK has been used. Other parameters are set as the default values in chapter 5.2.

solve this problem. This can be used to determine such cases.

The algorithm can be used twice with different initial points and different precisions. If the Hausdorff distance (see chapter 5.1) between the two resulting polygons is larger than some error bound, then it can be assumed that some parts of the AFS are approximated incorrectly. Then a more thorough investigation of the solution can follow.

4 Inverse polygon inflation algorithm

In this chapter the inverse polygon inflation algorithm is outlined. The main differences to the polygon inflation algorithm are explained; mainly [17] and [14] are used as sources. Some challenges of the implementation are also presented here.

4.1 General concepts of the algorithm

The inverse polygon inflation algorithm is mainly used for the cases where the polygon inflation algorithm could not be applied, e.g. if the AFS consists of one segment with a hole in the center, see figure 4.1. The inverse polygon inflation algorithm is, in essence, an adjusted polygon inflation algorithm.

4.1.1 Motivation and initialization

If the polygon inflation algorithm delivers a segment, that lies in at least three quadrants, then it is to assume that there exists only one segment with a hole, as done in FACPACK [17]. In this case, the polygon inflation can only detect the outer boundary.

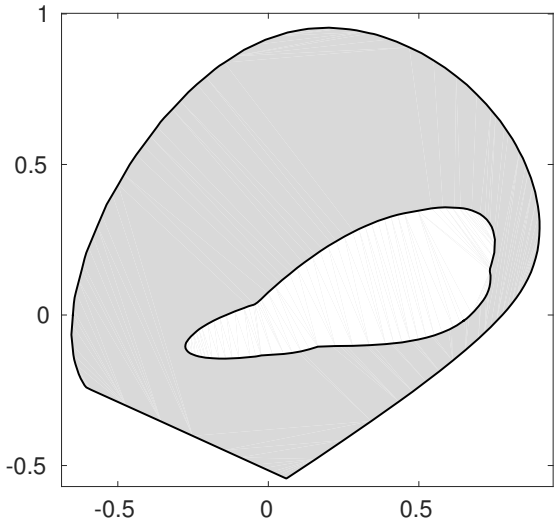

Figure 4.1: AFS segment with a hole. The data set `example3.mat` from FACPACK has been used.

The detection of this case can be a time intensive process because multiple vertices, initially approximating the hole, must be removed. Only then the polygon inflation algorithm can finish approximating the segment and detect the need for the inverse polygon inflation algorithm. Alternatively, this check can be done in each iteration.

The inverse polygon inflation algorithm consists of two steps. First, the outer boundary is approximated with FIRPOL. This is identical to the fast test in the polygon inflation algorithm. Thus, the polygon inflation algorithm can be used normally, with the exception that only the fast test is used and the standard test is omitted. This is computationally more efficient because it does not require numerical optimization. The origin can be used as a starting point since $(0, 0) \in \mathcal{F}$.

Then the approximation of inner boundary follows. It is based on the idea to approximate the hole in the interior of AFS in the same fashion as a feasible segment is approximated with the polygon inflation algorithm and is explained in the following section.

4.1.2 Approximation of the inner boundary

The target function has to be changed to approximate the inner boundary. FIRPOL already takes care of the constraint of nonnegativity of the row $A_{1,:}$. Thus, the remaining constraints have to be enforced and they deliver the inner boundary of the AFS.

Definition 7. *The set, that is used to calculate the inner boundary, is defined as*

$$\mathcal{M}^* = \left\{(\alpha, \beta) \in \mathbb{R}^2 \ : \ \exists S \in \mathbb{R}^{2\times 2} \text{ such that } T \text{ is regular and } A_{2,:} \geq 0,\ A_{3,:} \geq 0,\ C \geq 0\right\},$$

for T, A, and C in accordance with definition 4.

The area of feasible solutions is the intersection of the both constraining sets

$$\mathcal{M} = \mathcal{F} \cap \mathcal{M}^*.$$

The procedure for approximating the inner boundary matches the polygon inflation algorithm, with the exception that no fast test should be used and the target function should be modified to accommodate $\mathcal{M}^*$.

Definition 8. *The target function f for the inverse polygon inflation algorithm is defined as*

$$f(T) = \begin{pmatrix} \tilde{C} \\ \tilde{A} \\ \tilde{R} \end{pmatrix},\ \tilde{C} = \begin{pmatrix} \min\left(0, \frac{C_{11}}{\|C_{:,1}\|_\infty} + \varepsilon\right) \\ \vdots \\ \min\left(0, \frac{C_{k1}}{\|C_{:,1}\|_\infty} + \varepsilon\right) \\ \vdots \\ \min\left(0, \frac{C_{13}}{\|C_{:,3}\|_\infty} + \varepsilon\right) \\ \vdots \\ \min\left(0, \frac{C_{k3}}{\|C_{:,3}\|_\infty} + \varepsilon\right) \end{pmatrix},\ \tilde{A} = \begin{pmatrix} \min\left(0, \frac{A_{21}}{\|A_{2,:}\|_\infty} + \varepsilon\right) \\ \vdots \\ \min\left(0, \frac{A_{2n}}{\|A_{2,:}\|_\infty} + \varepsilon\right) \\ \vdots \\ \min\left(0, \frac{A_{31}}{\|A_{3,:}\|_\infty} + \varepsilon\right) \\ \vdots \\ \min\left(0, \frac{A_{3n}}{\|A_{3,:}\|_\infty} + \varepsilon\right) \end{pmatrix},\ \tilde{R} = \begin{pmatrix} R_{11} \\ R_{21} \\ \vdots \\ R_{23} \\ R_{33} \end{pmatrix},$$

where $R = Id(3) - TT^+$ *and both* A *and* C *are defined as before.*

The only change in the target function is in the vector $\tilde{A}$; the elements, corresponding to $A_{1,:}$, have been removed.

Another notable difference to the polygon inflation algorithm lies in the initialization. The set $\mathcal{M}^*$ has a hole in the middle an the goal is to approximate it. If the initial point would be chosen in the set $\mathcal{M}^*$, then a vertex could be found only in the direction of the hole. The solutions in the directions, pointing away from the hole, would likely result in an infinite loop.

Therefore, the initial quadrilateral should surround the hole from all sides. To reliably do that the inverse polygon inflation should be started at the origin $(0, 0)$. The origin is never included in the AFS and always in the FIRPOL. Consequently, $(0, 0) \notin \mathcal{M}^*$.

A noteworthy property of the approximated polygon is the fact that the vertices are listed in the opposite direction. If the numbering of the vertices for the polygon inflation algorithm was set to be counter-clockwise, then the interior polygon from the inverse polygon inflation algorithm will have a clockwise enumeration.

4.2 Challenges regarding the inverse polygon inflation algorithm

Many difficulties of the polygon inflation algorithm also apply to the inverse version and are not repeated here. The problems with the initialization and the segments in close proximity are not a concern here. The search for the outer boundary is also unproblematic because the fast test does not involve numerical minimization. However, there are some unique issues.

4.2.1 Unique challenges

Firstly, the inverse polygon inflation algorithm is slower than the polygon inflation algorithm. This can be explained by the differences in $\mathcal{M}^*$ and $\mathcal{M}$. The polygon inflation algorithm usually does not require the use of the genetic algorithm because the midpoints of most edges are feasible solutions. In comparison, to approximate the hole, the midpoints of most edges, especially at the earlier stages, are located in the hole and consequently are not feasible. Hence, two tests with `lsqnonlin` and one with `ga` are necessary for many vertices in the pre-bisection stage. This increases the overall computation time.

Secondly, the inverse polygon inflation algorithm approximates the inner boundary from the outside of the hole and not the inside of the segment, as it is with the polygon inflation algorithm. Moreover, the inverse polygon inflation algorithm can only mistakenly accept solutions lying outside of the hole, due to the same argumentation as in 3.2. Thus, for the inverse polygon inflation algorithm, the closing in on the border happens from the outside. In other words, after the first few vertices, the direction of the search for a new vertex will generally be inwards. An improvement in precision means the

reduction of the area of the polygon. This necessitates a separate consideration of the angle parameter α, see chapter 5.7.

4.2.2 A different approach

An alternative approach for the approximation of the inner boundary was tested, but it did not prove very successful.

The idea was to leave the target function similar to the polygon inflation algorithm and only change the condition for accepting the solution. To approximate the hole the condition should be changed to

$$\min_{S \in \mathbb{R}^{2\times 2}} \|f(\alpha, \beta, S)\|_2^2 \geq \varepsilon_{\text{tol}},$$

with the reasoning being that the solutions, accepted for the approximation of the hole, are exactly the solutions found not to be feasible by the polygon inflation algorithm.

This approach does not function well due to the reversing of the condition. As with the the inverse polygon inflation, the erroneous vertices can be accepted outside of the hole in the center. Moreover, it was often observed that the inner boundary was situated in the interior of the AFS, and many neighboring points were erroneous, see figure 4.2. This could be explained with using poor starting approximations from neighboring matrices.

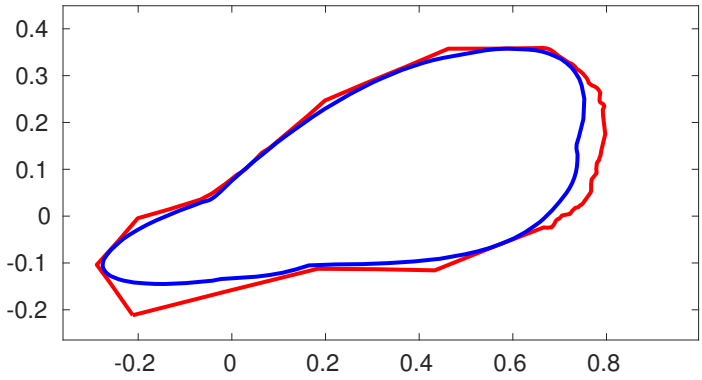

Figure 4.2: The inverse polygon algorithm (blue) compared with the results from the alternative approach (red).

Another problem occurred if no feasible solution was found for the entire AFS and the outer border was reached. Then the algorithm fails to find a new vertex.

In general, this variation failed to provide the necessary stability and precision for the case of one segment. Thus, it was abandoned in favor of the inverse polygon inflation.

4.2.3 The inverse polygon inflation algorithm for an AFS with three separated segments

The inverse polygon inflation algorithm can also be successfully used in scenarios with multiple segments. As previously explained, the calculation will take a longer time. Also the computing power will be wasted on some areas of $\mathcal{M}^*$ that are not relevant for the AFS, see 4.3. The resulting approximation agrees with the results from the polygon inflation algorithm.

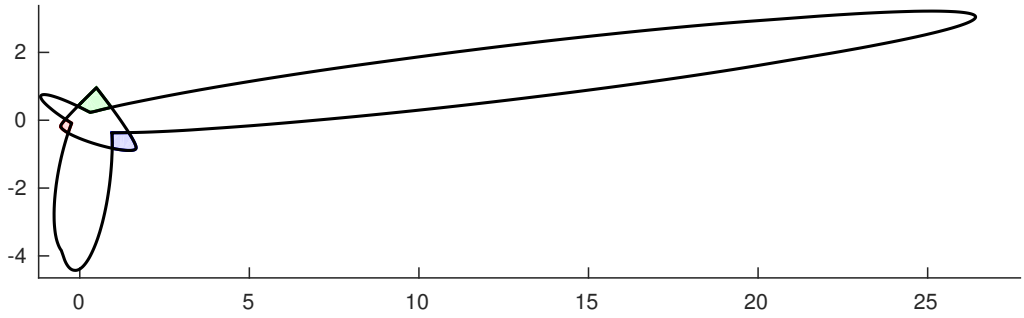

Figure 4.3: The inverse polygon algorithm delivers the inner boundary $\mathcal{M}^*$ and the outer boundary $\mathcal{F}$ of the data set `example2.mat`. The colorful segments of the AFS were generated with the polygon inflation algorithm. Note how they coincide with $\mathcal{F} \cap \mathcal{M}^*$. The parameters are set as the default values in chapter 5.2.

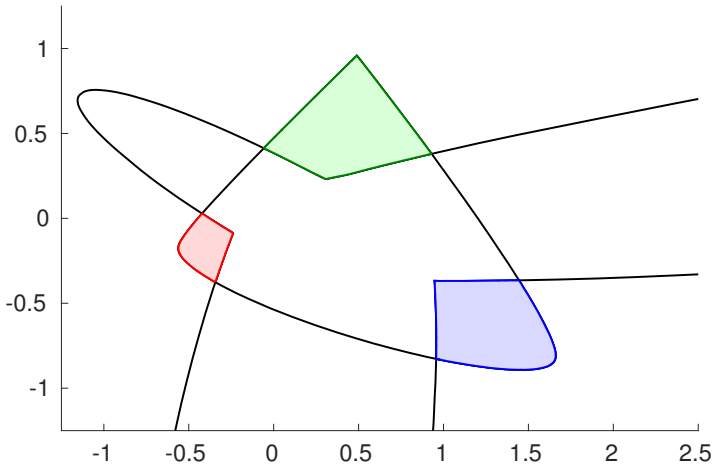

Figure 4.4: A close up on the AFS in the figure 4.3.

5 The numerical results

The polygon inflation algorithm is sensitive to the choice of parameters and an appropriate choice is crucial to its success. Some decisions on their values and their influence on the algorithm are discussed in this chapter. A few examples of varoius AFS are also given.

5.1 The techniques for the analysis of the algorithm

A useful measure to determine the accuracy of the resulting polygon is the Hausdorff distance. It intuitively helps to evaluate how different two sets are.

Definition 9. *The Hausdorff distance between two sets V and U is*

$$d_H(V,U) = \max\left(\sup_{v\in V}\inf_{u\in U}\|v-u\|_2, \sup_{v\in V}\inf_{u\in U}\|v-u\|_2\right).$$

To find the Hausdorff distance between two polygons, it is sufficient to determine the $\inf_{u\in u}\|v-u\|_2$ for all vertices v of the polygon V and vice versa. Thus, the distance between each vertex of a polygon and each edge of the other polygon and the other way round is necessary to determine the Hausdorff distance. The minimal distance is saved for each vertex of both polygons. The maximum of these values is the Hausdorff distance.

Another way to investigate how the algorithm works is to analyze how many steps are required in the pre-bisection and bisection stages for each vertex. To reduce the computing time, the sum of the steps should be held minimal. Also the minimal gain-of-height Δ_i in each iteration is of interest.

A useful statistic regarding numerical approximation is the number of times the additional feasibility test with `ga` was necessary and how many times the initial approximation matrix S_2 from the other neighboring vertex was needed. Also, the number of times, when adding a new vertex failed due to infinite loops or a large interior angle, can be used to describe, how well the algorithm functions. Finally, the number of removed vertices is a good measure for the quality of the numerical approximation.

The algorithm has been mainly tested on the triangle and rectangle from the figures 3.3 and 3.1 and example data sets from FACPACK, such as `example2.mat`.

5.2 Summary of parameters

In this chapter an overview and general properties of most parameters of the polygon inflation algorithm and the inverse polygon inflation algorithm are given. More in-depth

analysis of some parameters is provided in the following chapters.

The main parameters of the implemented MATLAB program are given in the array `par` and this naming convention kept here. A set of appropriate values for the parameters has been chosen for the numerical investigation of the algorithm and in this work is referred to as the default parameters in 5.2. These values are given together with the parameter.

- The parameter `par(1)` $= \varepsilon = 10^{-12}$ changes the degree of nonnegativity that is accepted in the matrices C and A. For a constructed problem with noise-free data it can be set to 0, and [16] suggests values less than 0.05. For further examination see chapter 5.4.

- The parameter `par(2)` $= \varepsilon_b = 10^{-3}$ is the terminating parameter of the bisection method. It is stopped when the distance between the two points is less than ε_b. According to [16] it can be set equal to `par(4)`. For further examination see chapter 5.3.

- The parameter `par(3)` $= \varepsilon_{tol} = 10^{-10}$ is responsible for the decision over the feasibility of a solution. Its main purpose is to negate rounding errors. If it is set too large, an irregular transformation matrix T can be accepted. For further examination see chapter 5.4.

- The parameter `par(4)` $= \delta = 10^{-3}$ is used to terminate the main iteration. If $\Delta_i \leq \delta, \ \forall i$ is true, then the iteration is stopped. For further examination see chapter 5.3.

- The parameter `par(5)` $= 0.1$ defines the length of a step in the pre-bisection stage. The actual step size is a product of `par(5)` and the length of the edge. This is justified with the assumption that already well-refined edges will not have large gains of area. For further examination see chapter 5.5.

- The parameter `par(8)` $= 100$ changes the number of allowed steps in the pre-bisection stage. It depends largely on `par(5)`; however, the only drawback of setting the value too large is the increase in computation time. For further examination see chapter 5.5.

- The parameter `par(9)` $= \alpha = 190$ is responsible for the removal of mistakenly calculated vertices in the interior of the AFS. This value is given in degrees and is always calculated for the interior angle. A simple, non-adaptive value seems to function well. It can be set to 200 or 300 for the inverse polygon inflation algorithm. For further examination see chapter 5.7.

- The parameter `par(10)` $= \theta = 45$ alters how much the initial direction vector is rotated to find the first two vertices in the case of an error in initialization. The angle is given in degrees counterclockwise and a default value of 45 degrees is used. If the angle is chosen too small, then additional rotations might be necessary and it might take longer to detect a dot set. If it is chosen too large, then no initial

quadrilateral might be found after a full 360-degree rotation. It is advisable to choose θ as a divisor of 360, then a dot set would be found only after one rotation.

Other parameters with lesser importance are:

- The parameter $v_{\text{init}} = (-1, 0)$. The influence of the starting direction is negligible because of the rotation of initial direction.
- The stopping criterion for the rotation of the initial direction. It is set as the norm of the new and the initial direction $v_{\text{init}} = (-1, 0)$ being less than 10^{-5} and, as long as θ is appropriately chosen, it serves the purpose of neutralizing any rounding-errors that are accumulated during rotation.
- The parameter $d_{\text{init}} = 10^{-3}$ rules out any two initial vertices closer than its value. Caution is required since it is dependent on the scaling of the system. If the AFS would be scaled down e.g. 1000 times, this parameter would be too large. For further examination see chapter 5.8.
- The parameter `init_step` defines the length of the edge for the pre-bisection stage of the first two vertices (when no such length is available). It is then multiplied with `par(5)` to find the step size in the pre-bisection stage. The default value is set as 1.

There are also some internal parameters for algorithms in MATLAB that can considerably affect the success of the polygon inflation algorithm, see also 5.6.

- For `lsqnonlin` it is possible to change the default algorithm as well as
  - `'FunctionTolerance'=1e-12` (MATLAB default is `1e-6`); this parameter is the termination tolerance and has a strong influence on the quality of the numerical optimization.
  - `'MaxIterations'=400` (MATLAB default is `400`); this parameter seems to have negligible influence on the quality of the result.
  - `'OptimalityTolerance'=1e-12` (MATLAB default is `1e-6`); this parameter changes the tolerance of the first-order optimality. Decreasing this parameter seems to result in more precise solutions but also more failed optimizations.
  - `'StepTolerance'=1e-6` (MATLAB default is `1e-6`); this parameter seems to have negligible influence on the quality of the result.

  and others.
- For `ga` it is possible to change amongst others `'FunctionTolerance'=1e-6` (MATLAB default is `1e-6`); it seems to have negligible influence on the quality of the result.

5.3 The terminating parameters

The parameters ε_b and δ are responsible for terminating the bisection method and the main iteration, respectively. Thus, it is expected that lower value results in a more refined polygon.

5.3.1 The effect on the precision

At first they are set $\varepsilon_b = \delta$, as suggested in [17]. If the algorithm works as intended, then lower termination criterion δ results in a larger amount of vertices added to the polygon and this should ensure a more exact approximation of the AFS. This is indeed the case, see table 5.1.

ε_b, δ	Triangle		Rectangle		`example2.mat`	
	Vertices	d_{H}	Vertices	d_{H}	Vertices	d_{H}
$1 \cdot 10^{-6}$	123	$1.9 \cdot 10^{-6}$	177	$1.9 \cdot 10^{-6}$	2804	–
$1 \cdot 10^{-5}$	113	$1.3 \cdot 10^{-5}$	145	$1.2 \cdot 10^{-5}$	1187	$9.4 \cdot 10^{-5}$
$1 \cdot 10^{-4}$	85	$2.9 \cdot 10^{-4}$	123	$7.8 \cdot 10^{-5}$	531	$2.0 \cdot 10^{-4}$
$1 \cdot 10^{-3}$	57	$1.9 \cdot 10^{-3}$	73	$1.9 \cdot 10^{-3}$	227	$3.8 \cdot 10^{-3}$
$1 \cdot 10^{-2}$	41	$2.5 \cdot 10^{-2}$	57	$2.2 \cdot 10^{-2}$	100	$2.2 \cdot 10^{-2}$
$1 \cdot 10^{-1}$	15	$3.2 \cdot 10^{-1}$	21	$1.4 \cdot 10^{-1}$	27	$1.9 \cdot 10^{-1}$

Table 5.1: The number of vertices and Hausdorff distance for varying ε_b and δ. To calculate the Hausdorff distance the actual polygons have been used for the triangle and the rectangle. For the `example2.mat` the result with $1 \cdot 10^{-6}$ has been used as a reference. Other parameters are set as the default in 5.2.

Note that the solution with FACPACK has not been used as a reference to calculate the Hausdorff distance here, because additional precision is required to evaluate these results. The Hausdorff distance $d_{\mathrm{H}} = 8.2 \cdot 10^{-5}$ between the result with $\varepsilon_b = \delta = 1 \cdot 10^{-6}$ and the solution with FACPACK with both e-bound and d-stopping set to 10^{-6} is a good result; however, FACPACK still is a different implementation. Hence, it has been avoided in this section.

Both parameters ε_b and δ can be adjusted to achieve the desired refinement. The value $1 \cdot 10^{-3}$ is suggested, as it offers a middle ground between precision and computing time. This value has also been used in [17].

5.3.2 The connection between the terminating parameters

Next, the connection between ε_b and δ is analyzed. In the figure 5.1 it is clearly visible where the algorithm would have stopped for a fixed parameter $\varepsilon_b = 10^{-6}$ and different δ, indicated by the black horizontal lines. The results, when δ is held fixed and only ε_b is changed, are shown in table 5.2.

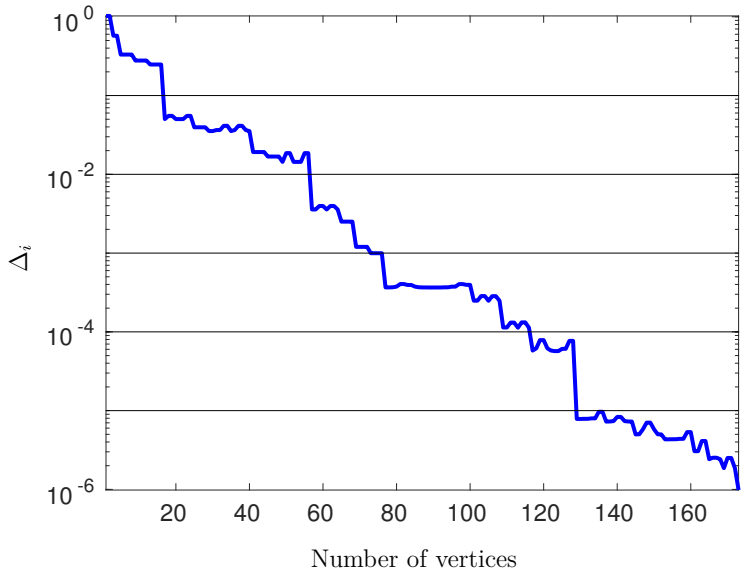

Figure 5.1: Change of Δ_i with $\varepsilon_b = \delta = 1 \cdot 10^{-6}$. Other parameters are set as the default in 5.2.

ε_b	Triangle		`example2.mat`	
	Vertices	d_{H}	Vertices	d_{H}
$1 \cdot 10^{-6}$	65	$1.7 \cdot 10^{-3}$	246	$1.6 \cdot 10^{-3}$
$1 \cdot 10^{-5}$	65	$1.8 \cdot 10^{-3}$	243	$2.0 \cdot 10^{-3}$
$1 \cdot 10^{-4}$	63	$2.4 \cdot 10^{-3}$	239	$1.4 \cdot 10^{-3}$
$1 \cdot 10^{-3}$	57	$1.9 \cdot 10^{-3}$	227	$3.8 \cdot 10^{-3}$
$1 \cdot 10^{-2}$	81	with errors	183	$7.9 \cdot 10^{-3}$
$1 \cdot 10^{-1}$	infinite loop		154	$2.9 \cdot 10^{-2}$

Table 5.2: The number of vertices and Hausdorff distance for varying ε_b, and fixed $\delta = 1 \cdot 10^{-3}$. To calculate the Hausdorff distance the actual sets have been used for the triangle and the rectangle. For the `example2.mat` the result with $1 \cdot 10^{-6}$ from 5.1 has been used as a reference. Other parameters are set as the default in 5.2.

If $\varepsilon_b << \delta$ then the imprecision of bisection can considerably influence the termination of the main loop. Moreover, it can lead to the polygon being self-crossing; thus, the meaning of inside and outside is lost and new vertices can be searched in the wrong direction. This is the reason for the errors and the infinite loop in the triangle case. However a proper removal of incorrect vertices by angle (which is only done for the cases with numerical optimization) circumvents this problem. Consequently, if computation time restricted, a lower value of ε_b can be used. However, it should be noted that more vertices will be removed due to not fulfilling the angle parameter. Six vertices were

removed in the case of $\varepsilon_b = 1 \cdot 10^{-3}$, but 67 vertices in the case of $\varepsilon_b = 1 \cdot 10^{-1}$. The number of steps in bisection dropped considerably. On average 2.74 steps were needed in the case of $\varepsilon_b = 1 \cdot 10^{-3}$, but only 0.0045 steps in the case of $\varepsilon_b = 1 \cdot 10^{-1}$. The resulting polygon can be seen in figure 5.2.

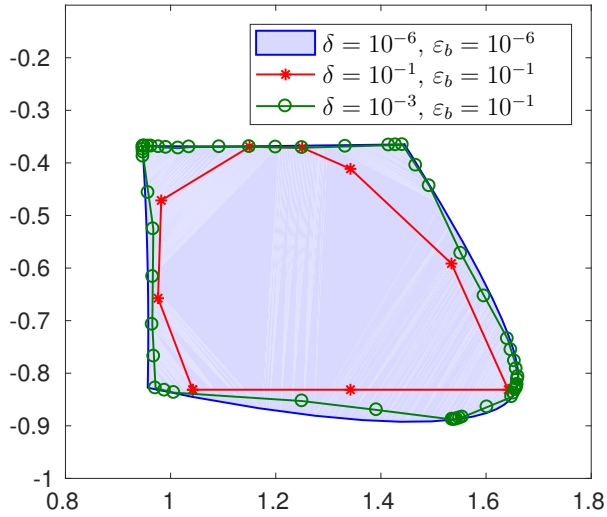

Figure 5.2: One segment of `example2.mat` with different values for ε_b and δ. Other parameters are set as the default in 5.2.

If $\varepsilon_b >> \delta$ then all vertices are calculated more precisely than can be utilized by the main iteration. It provides an improvement of the precision to some extent and can be computationally more efficient. The imprecisions stem from an early termination of the iteration; for the isosceles triangle, they can be observed near both of the base vertices. The number of steps in the pre-bisection stage for `example2.mat` has been considerably increased. On average 12.6 steps were needed in the case of $\varepsilon_b = 1 \cdot 10^{-6}$ (in comparison, 2.74 steps in the case of $\varepsilon_b = 1 \cdot 10^{-3}$).

To conclude, the usage of different ε_b and δ has limited applications if the computing time is of interest. In the general, they can be kept the same.

5.4 The numerical approximation parameters

Parameters ε and ε_{tol} are both used for the decision whether a solution is feasible.

The parameter ε_{tol} negates some small numerical rounding errors during the numerical optimization. If this value is too large, then some non-invertible matrices T can be accepted; this can sabotage the results.

Setting ε_{tol} to zero prevents any numerical errors to be tolerated. Consequently, no AFS can be found. In [16] the value 10^{-10} is used. If the value is reduced to e.g. 10^{-15} then the number of vertices removed by the angle for `example2.mat` increases from 6 to 19. The Hausdorff distance to the reference solution in the table 5.1 drops from

$3.8 \cdot 10^{-3}$ to $2.5 \cdot 10^{-3}$. Therefore, the parameter ε_{tol} can be reduced for a more precise but slightly slower approximation of AFS.

The parameter ε is used to tolerate small negative entries in matrices A and C. In [16] it is revealed, that it especially useful for cases with perturbated data. In [14] it is further recommended to set ε larger than the scaled values of the smallest (negative) entries in A and C if the nonnegative matrix factorization tool is incapable of providing entirely nonnegative matrices. For data set `example2.mat` the Hausdorff distance between solutions with $\varepsilon = 0$ and $\varepsilon = 1 \cdot 10^{-12}$ is $2.0 \cdot 10^{-3}$, indicating minor differences.

An interesting property, demonstrated in [14], is that the area of segments is increasing with rising ε. This behavior is shown in figure 5.3. For $\varepsilon > 0.05$ more noticeable errors, similar to the lower left segment, are beginning to appear, e.g., for $\varepsilon = 0.05$ a neighboring vertex was removed 18 times because the new vertex was not found. This should not happen at all and it indicates problems with numerical optimization for such a degree of accepted negativity. It is important to note that these are not valid representations of the actual AFS.

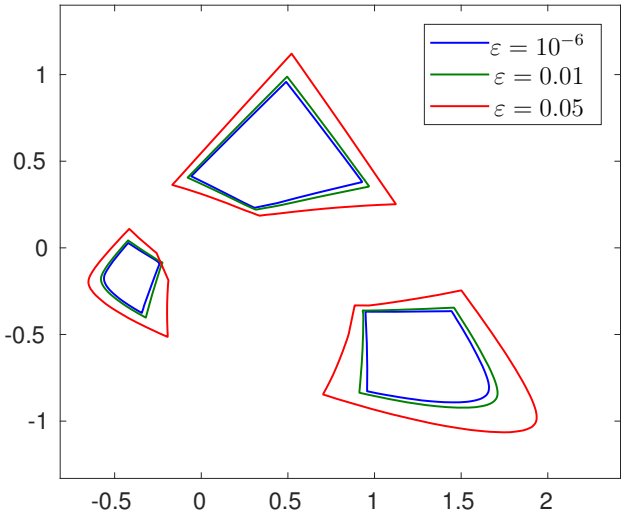

Figure 5.3: The changes of the AFS for different values of ε. Other parameters are set as the default in 5.2.

5.5 The pre-bisection stage parameters

The parameters `par(5)` and `par(8)` influence the pre-bisection stage of the polygon inflation algorithm.

The step size, controlled by `par(5)`, plays a crucial role in the pre-bisection stage. If the size is chosen too large, then bisection stage will need a lot more steps. If the step

size is too small, then the limitation `par(8)` of available steps might be reached before the bisection method could be started.

The actual step size is set adaptively as a product of `par(5)` and the edge that is being subdivided. This ensures that the step sizes are larger at the beginning of the main iteration and smaller at the end when only minor adjustments are required. It also helps to avoid problems of scaling, if one, for example, wants to approximate a triangle, that has a 100 times larger area. Without the adaptivity, `par(5)` and `par(8)` should have been adjusted for each problem.

The goal of this chapter is to find the optimal parameter `par(5)` that produces the least amount of overall steps. Parameter `par(8)` is should be chosen accordingly to the number of steps. The results from the evaluation of the optimal parameter are shown in tables 5.3 and 5.4.

`par(5)`	Avg. pre-bisection steps	Avg. bisection steps	Avg. steps	Max. pre-bisection steps
10	1	9.89	10.89	1
5	1	8.89	9.89	1
1	1.01	6.83	7.84	2
0.5	1.07	5.83	6.89	3
0.1	2.41	3.62	6.03	11
0.05	4.01	2.60	6.61	21
0.01	17.61	1.17	18.78	103
0.005	34.40	0.70	35.10	206
0.001	162.93	0.04	162.97	1027

Table 5.3: The average and maximal number of pre-bisection and bisection stage steps for the rectangle. Other parameters are set as the default in 5.2.

Figure 5.4 illustrates the increase in the number of pre-bisection steps for a decreasing value of `par(5)`. The adaptivity of the pre-bisection step size is visible there. Excluding the first few vertices, the behavior of the number of pre-bisection steps stays about the same, while the number of steps in the bisection stage decreases, indicating that the step size of the pre-bisection stage is nearing the terminating parameter of the bisection method.

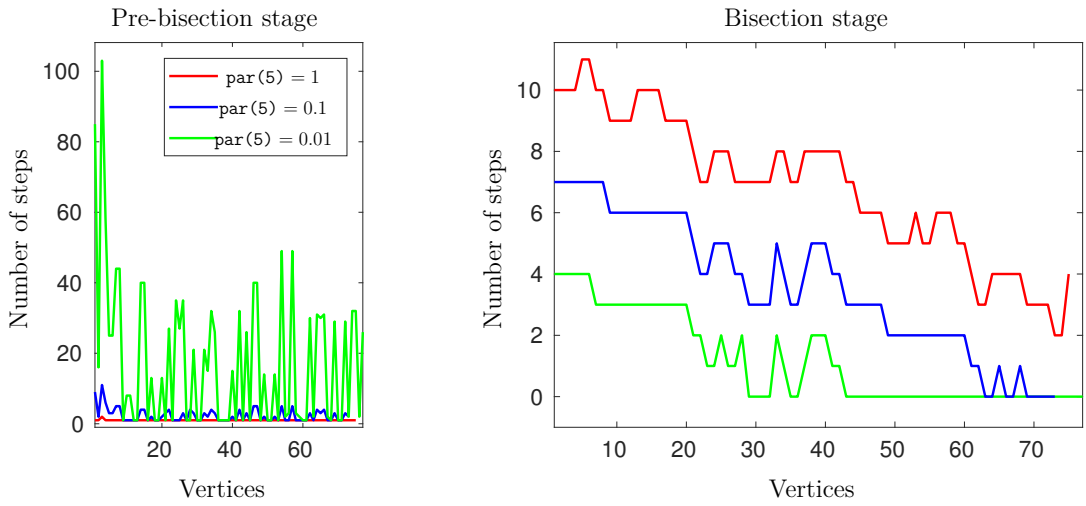

Figure 5.4: The number of steps for pre-bisection and bisection stage for the rectangle with varying `par(5)`. Other parameters are set as the default in 5.2.

`par(5)`	Avg. pre-bisection steps	Avg. bisection steps	Avg. steps	Max. pre-bisection steps
1	1	6.01	7.01	1
0.5	1.04	5.01	6.05	2
0.1	1.94	2.74	4.68	9
0.05	3.54	1.81	5.36	58
0.01	13.23	0.55	13.77	91

Table 5.4: The average and maximal number of pre-bisection and bisection stage steps for `example2.mat`. Other parameters are set as the default in 5.2.

For the case with known boundary as well as for the case with numerical optimization the optimal value for `par(5)` seems to be 0.1, as it provides the smallest number of required steps. Therefore, the total computation effort is reduced. If this value is used, then `par(8)=100` can be safely chosen, ensuring that a successful search for boundary is not stopped prematurely.

5.6 The internal numerical optimization parameters

This chapter is intended to illustrate how strongly some internal parameters of the MATLAB function `lsqnonlin` can affect the quality of the optimization. Table 5.5 shows, how some changes to various parameters affect the resulting approximation.

Settings for `lsqnonlin`	Vertices	S_2 is needed	`ga` is needed	Not added by angle	Cut by angle
Default	251	103	20	5	91
`'MaxIterations',800`	251	103	20	5	91
`'StepTolerance',1e-12`	251	103	20	5	91
`'FunctionTolerance',1e-12`	235	16	3	0	8
`'OptimalityTolerance',1e-12`	237	74	25	10	61
`'FunctionTolerance',1e-9, 'OptimalityTolerance',1e-9`	243	28	2	0	27
`'FunctionTolerance',1e-12, 'OptimalityTolerance',1e-12`	227	17	6	3	6
`'FunctionTolerance',1e-15, 'OptimalityTolerance',1e-15`	228	10	3	0	7

Table 5.5: The number of vertices and various events for `example2.mat` with different parameters for `lsqnonlin` and `ga`. Other parameters are set as the default in 5.2.

Firstly, it is shown that improving the parameters `'MaxIterations'` and `'StepTolerance'` for `lsqnonlin` has no visible effect on the quality of numerical approximation. Changing the `'FunctionTolerance'` for `'ga'` to `1e-12` also did not influence any of the inspected values.

Secondly, changing `'FunctionTolerance'` for `lsqnonlin` has a very strong positive impact on all of the investigated measures, in particular, on the number of mistakenly added vertices and times, when S_2 was needed. This can be explained by the fact that, in general, ε_{tol} is set to $1 \cdot 10^{-10}$. Thus, the tolerance criterion of the polygon inflation algorithm is smaller than the default precision of the minimization routine, leading to misclassified solutions. Improving the precision of the minimization routine to be more sensitive than ε_{tol} greatly reduces the risk of rejecting a valid solution on basis of an unlucky optimization.

Thirdly, decreasing `'OptimalityTolerance'` seems, at first, to have a negative effect on the number of failed `lsqnonlin` approximations. However, this parameter reduces the vertices, that are mistakenly accepted and afterward removed by about a third. Hence, the change is favorable.

Consequently, the combined improvement of the parameters `'FunctionTolerance'` and `'OptimalityTolerance'` has the greatest impact on the quality of the approximation. It can be further increased by lowering these parameters; however, the further improvement is comparably minimal and not required for general use.

5.7 The angle parameter

Arguably the most important parameter of the polygon inflation algorithm is α. Without a way of recognizing and managing poorly approximated vertices, successfully finding the AFS is hard. The importance has already been thoroughly discussed in the chapters 3.2 and 4.2. This chapter deals with finding appropriate values of the parameter α for the polygon inflation and inverse polygon inflation algorithms.

According to chapter 5.6, with appropriate settings only nine problematic vertices were detected from the final number of 227 accepted vertices. Hence, the algorithm would finish without errors, that are depicted in figure 3.6. This means that the importance of α is lower if precision is not very important and the settings for `lsqnonlin` from 5.2 are used. Nonetheless, some inaccurate vertices are still accepted.

Choosing α adaptively has been considered, but it does not seem to be necessary because a simple, fixed parameter functions well. Moreover, an extensive analysis of the adaptive function would be necessary.

5.7.1 The polygon inflation algorithm

First, the choice of the parameter α for the polygon inflation algorithm is outlined. The goal is to find the smallest possible angle, where the algorithm still functions. It should be noted, that it also depends on the problem. If the parameter α is chosen too large, then the border can have a zigzag pattern. However, if a segment has a large reflex inner angle somewhere, then the algorithm might have problems approximating the segment because vertices near this location would fail the angle test.

Let's assume that the AFS segment has a vertex with an inner angle of 200 degrees. Then the algorithm with $\alpha \geq 190$ could approximate this vertex under lucky circumstances (when angles for both vertices nearest the point are equal) and with $\alpha \geq 200$, always.

If the inner angle would be 220 degrees, then under lucky circumstances the algorithm would function with $\alpha \geq 200$ and always, with $\alpha \geq 220$.

Reflex inner angles are also produced by an uneven boundary, caused by bisection method; hence, $\alpha = 180$ does not function.

For `example2.mat` with $\alpha = 200$ and $\alpha = 195$, the total of 5 vertices did not meet the angle criterion. With $\alpha = 190$ the number is 9, with $\alpha = 185$, 15 vertices and with $\alpha = 182$, 26 vertices. The value $\alpha = 190$ was chosen for this data set to provide a middle ground between precision and permitted maximal angle.

It is important to mention, that changing the parameter α is not the primary strategy to improve the precision of the result; this is achieved with the parameter δ because with a reduced δ the problematic vertex will not fulfill the angle test after a few subdivisions and will be removed. The parameter α is primarly meant to avoid problems described in 3.2 and ensure the quality of the resulting polygons. Nevertheless, adapting the α to a particular problem can improve the final precision. For an unknown AFS a higher value of α can be beneficial for a rough first approximation and can be lowered afterward for a more refined boundary of the polygon.

5.7.2 The inverse polygon inflation algorithm

Next, the parameter α for the inverse polygon inflation algorithm is analyzed. For the first stage, in which FIRPOL is found, no removal of vertices is required, since the numerical optimization is not involved. The second stage of the inverse polygon inflation algorithm is different from the polygon inflation algorithm in the respect that after an initial inflation of the polygon an improvement in precision can be achieved by deflation.

A vertex with a large reflex angle can be an evidence of a move inwards which is desired in this case. A vertex with a small acute angle is an indication of the algorithm failing to do that. Hence, two parameters α_{min} and α_{max} were tested for the lower and upper bounds of the acceptable angles. However, practical testing showed no difference for $\alpha_{min} = 0$ and $\alpha_{min} = 40$. An adaptively set value was also tested but it can result in an infinite loop and additional considerations would be needed to prevent that. Therefore, the lower bound was abandoned.

Indeed, the numerical tests showed little to no effect even when the upper bound $\alpha_{max} = \alpha$ was changed. The value 190 was found to be too small for this case. Thus, $\alpha = 200$ and $\alpha = 300$ was compared, see table 5.6.

α	ε_b, δ	Vertices	Removed by angle	Not added by angle	d_{H}
200	10^{-3}	287	1	13	$1.9 \cdot 10^{-3}$
300	10^{-3}	282	0	0	$2.6 \cdot 10^{-3}$
200	10^{-4}	851	1	65	$5.8 \cdot 10^{-4}$
300	10^{-4}	903	0	88	$3.8 \cdot 10^{-4}$

Table 5.6: Number of vertices and various events for `example3.mat` with different parameters ε_b, δ and α. Hausdorff distance is measured to a result from FACPACK with e-bound and d-stopping set to 10^{-5}. Other parameters are set as the default in 5.2.

It is important to remember that the parameter α is not intended to improve precision, but to ensure the functioning of the algorithm. As the results with $\varepsilon_b = \delta = 10^{-4}$ show, even with $\alpha = 300$ it was possible to detect incorrect vertices and avoid adding them. It should also be noted that both values of α delivered polygons with fragments that had a zigzag pattern with still acceptable reflex angles, see figure 5.5. This behavior could be improved by further decreasing the values for $\varepsilon_b = \delta$ or by creating a more complex angle test.

The test did not clearly show a better value for α; both values can be successfully used.

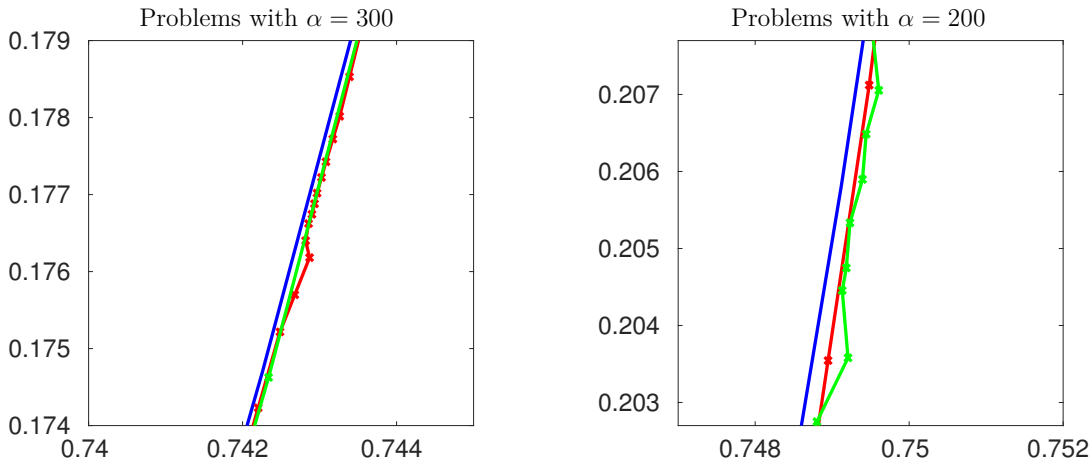

Figure 5.5: Uneven boundaries for the inverse polygon inflation algorithm with $\alpha = 300$ (red) and $\alpha = 200$ (green). The blue line is a solution from FACPACK with e-bound and d-stopping set to 10^{-5}. Here $\varepsilon_b = \delta = 10^{-4}$ and other parameters are set as the default in 5.2.

5.8 An AFS with a dot set

Important parameters for the case, when one of the segments is a dot set, are θ, stopping criterion for the rotation, and $d_{\texttt{init}}$. First two are already thoroughly explained in chapter 5.2. However, the importance of $d_{\texttt{init}}$ can be well illustrated with this example. For a demonstration of this case see figure 5.6.

If $d_{\texttt{init}} = 0$, then it can happen that the initial point is accepted as the first and the second vertex simultaneously. Thus, the direction vector for the search of the third vertex is not defined and no search is possible. It also eliminates the case where two vertices with a distance close to machine epsilon are found; thus, an initial quadrilateral is calculated where it should not exist. These problems originate from numerical approximation mistakes and allowing small negative entries.

It should also be noted that this version of polygon inflation algorithm does not work with an AFS where one of the segments consists of a line because it can not distinguish between dot and line sets. The possibility of finding the direction of the line by rotating the vector is almost non-existent. Thus, another approach should be used to check the solutions that are marked as a dot set.

5.9 Other examples

This chapter explains how model problems can be generated and gives some examples. All of the figures in this chapter are generated with the implementation, described in this thesis.

The generation of the matrix D in `example2.mat` is explained in [18]. It begins with finding matrices C and A and then their product D is calculated.

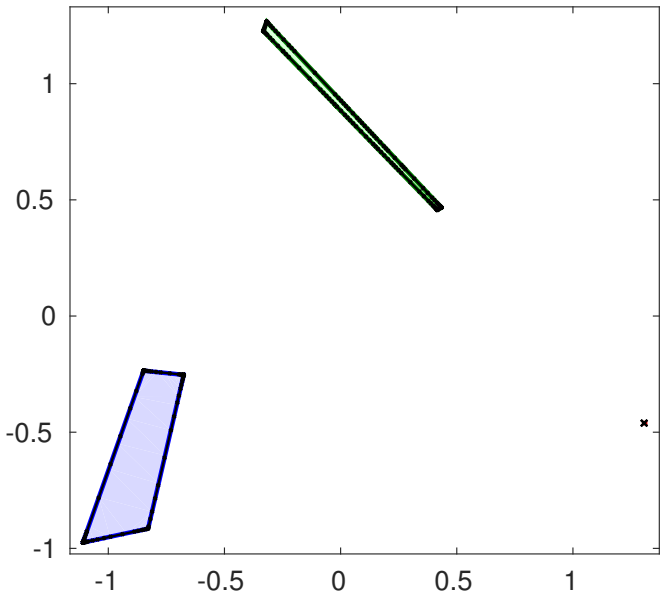

Figure 5.6: The results of the data set `DotSegment.mat` with the polygon inflation algorithm (black) compared with FACPACK (colorful background). $\delta = \varepsilon_b = 1 \cdot 10^{-4}$. Other parameters are set as the default in 5.2.

The vectors t and x are defined as

$$t = (0, 1, \ldots, 20)^T \quad x = (0, 1, \ldots, 100)^T$$

and by performing the following operations component-wise

$$\begin{aligned} C_{:,1} &= e^{-k_1 t} \\ C_{:,2} &= \frac{k_1}{k_2 - k_1}\left(e^{-k_1 t} - e^{-k_2 t}\right) \\ C_{:,3} &= 1 - C_{:,1} - C_{:,2} \end{aligned} \qquad \begin{aligned} A_{1,:} &= a_1 \exp\left(\frac{-(x-b_1)^2}{c_1}\right) + d_1 \\ A_{2,:} &= a_2 \exp\left(\frac{-(x-b_2)^2}{c_2}\right) + d_2 \\ A_{3,:} &= a_3 \exp\left(\frac{-(x-b_3)^2}{c_3}\right) + d_3 \end{aligned}$$

the matrices C and A are calculated with

$$k_1 = 0.75,\ k_2 = 0.25,\ a_1 = 0.95,\ a_2 = 0.9,\ a_3 = 0.7,\ b_1 = 20,\ b_2 = 50,\ b_3 = 80,$$

$$c_1 = c_2 = c_3 = 500,\ d_1 = 0.3,\ d_2 = 0.25,\ d_3 = 0.2.$$

This can be used to create examples for testing purposes by changing some of the parameters. Some alterations in the structure of the AFS are shown in the figures 5.7, 5.8, 5.9, 5.10 and 5.11.

Note that the orientation can change for these examples because the original matrix is altered. This complicates the comparison of the resulting AFS. To avoid that, some of the AFS have been flipped to accommodate the orientation of the original problem from

`example2.mat`, see figure 2.1. Thus, the orientation of the resulting polygons is flipped, when the normal orientation, according to convention in section 2.2.3, is different than in `example2.mat` to ease the comparison between them.

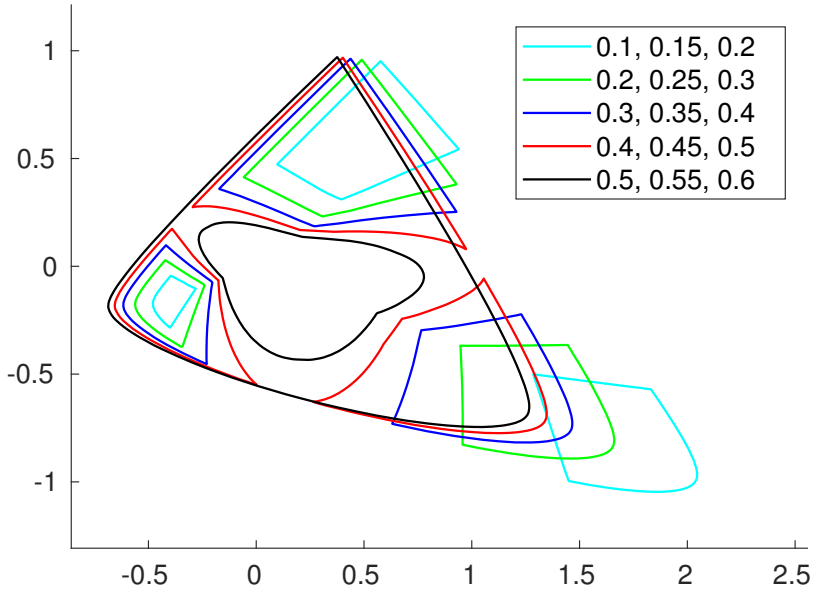

Figure 5.7: The results of the generated problem with varying values for d_1, d_2, d_3 (in this order). The parameters are set as the default in 5.2.

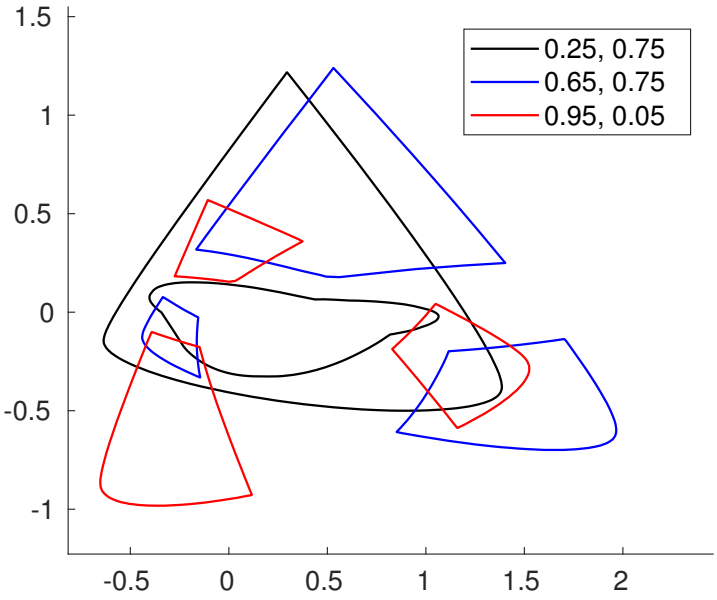

Figure 5.8: The results of the generated problem with varying values for k_1, k_2 (in this order). The red solution has been reflected over the α-axis. The parameters are set as the default in 5.2.

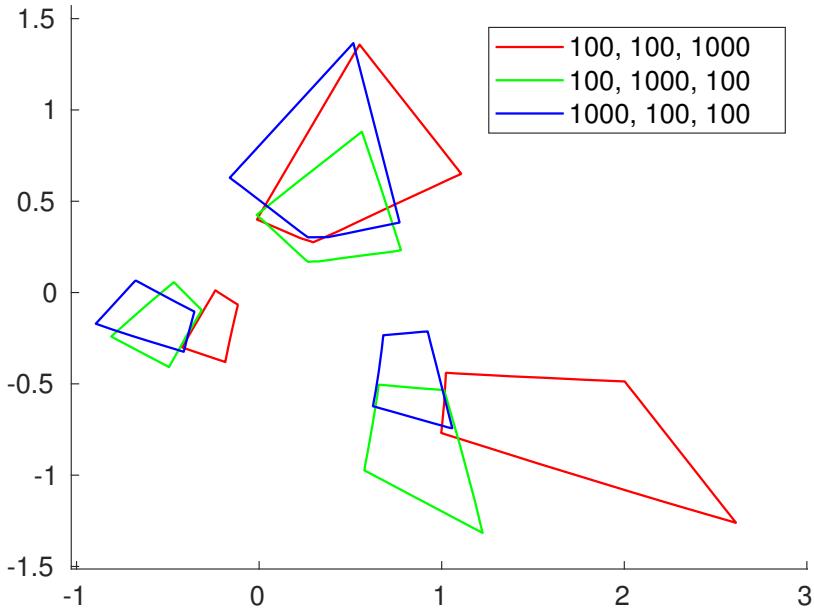

Figure 5.9: The results of the generated problem with varying values for c_1, c_2, c_3 (in this order). The blue solution has been reflected over the β-axis and the green - over both axes. The parameters are set as the default in 5.2.

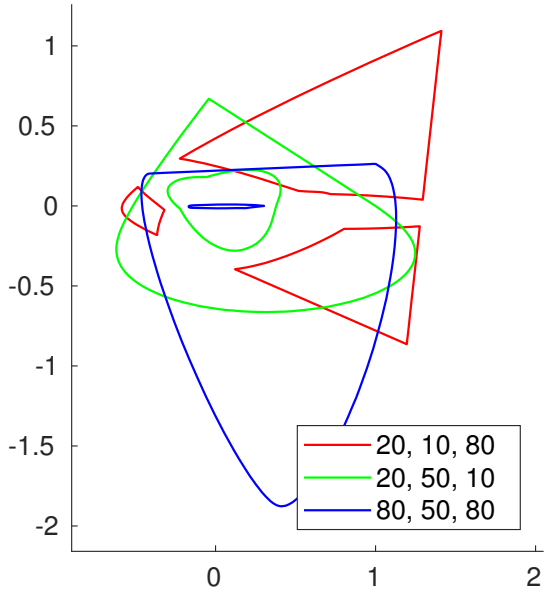

Figure 5.10: The results of the generated problem with varying values for b_1, b_2, b_3 (in this order). The green and blue solution have been reflected over the α-axis. The parameters are set as the default in 5.2.

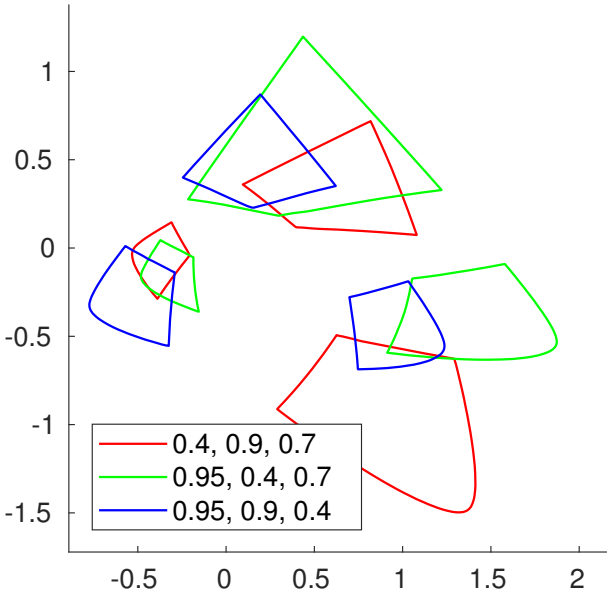

Figure 5.11: The results of the generated problem with varying values for a_1, a_2, a_3 (in this order). The red solution has been reflected over the α-axis. The parameters are set as the default in 5.2.

6 Conclusion

The polygon inflation algorithm and its inverse alteration are robust multivariate curve resolution methods. They are used to describe the AFS that originates from the rotational ambiguity of the nonnegative matrix factorization problem in analytical chemistry. The algorithms were described and analyzed in this thesis. This work focused on a chemometrical application — more specifically, the pure component resolution of three-component systems.

The derivation of the method was studied and an overview of the main challenges of a successful implementation was given. Moreover, appropriate parameters for the algorithm were suggested and tested. The algorithms were implemented in MATLAB and some numerical results from the publications and the FACPACK toolbox could be reproduced.

This work is limited to chemometric applications and three-component systems. The emphasis was set on the polygon inflation algorithm and the inverse polygon inflation algorithm was rather summarized than exhaustively investigated. The extent of the work was limited by the available time for finishing this thesis.

This work could be further extended to four-component systems. Furthermore, the comparison of computing time and stability with other available methods, as well as a more in-depth analysis of the possible variations of the polygon inflation algorithm and its parameters could be performed. The goal would be to reduce the produced errors and the computing time. Future enhancements to the nonlinear least squares algorithms could also improve this. An extension to distinguish between the dot and the line segments of the AFS could be programmed and tested.

In the future, applications of the polygon inflation algorithm could be found in other fields that require the solution of nonnegative factorization problems; a further analysis of this algorithm is essential for that. Also, an extension to n-component systems is desired for a wider practical adoption in chemometry.

This work hopefully makes it easier to create practical implementations of the polygon inflation algorithm for diverse applications in the future.

Bibliography

[1] H. Abdollahi and R. Tauler. Uniqueness and rotation ambiguities in Multivariate Curve Resolution methods. Chemom. Intell. Lab. Syst., 108(2):100–111, 2011.

[2] O.S. Borgen and B.R. Kowalski. An extension of the multivariate component-resolution method to three components. Anal. Chim. Acta, 174(0):1-26, 1985.

[3] J. D. Faires and R. L. Burden. Numerische Methoden. Spektrum Akademischer Verlag, 1994.

[4] A. Golshan, H. Abdollahi, S. Beyramysoltan, M. Maeder, K. Neymeyr, R. Rajko, M. Sawall, and R. Tauler. A review of recent methods for the determination of ranges of feasible solutions resulting from soft modelling analyses of multivariate data. Anal. Chim. Acta, 911:1–13, 2016.

[5] A. Golshan, H. Abdollahi, and M. Maeder. Resolution of rotational ambiguity for three-component systems. Anal. Chem., 83(3):836–841, 2011.

[6] G.H. Golub and C.F. Van Loan. Matrix Computations. Johns Hopkins University Press, 2013.

[7] A. Jürß. Über nichtnegative Matrixfaktorisierungen und geometrische Algorithmen zur Approximation ihrer Lösungsmengen. Doctoral dissertation. University of Rostock, 2017.

[8] W. H. Lawton and E. A. Sylvestre. Self modelling curve resolution. Technometrics, 13:617–633, 1971.

[9] A. Meister, Estimation of component spectra by the principal components method. Anal. Chim. Acta, 161:149–161, 1984.

[10] H. Minc. Nonnegative matrices. John Wiley & Sons, New York, 1988.

[11] K. Neymeyr, M. Sawall, and D. Hess. Pure component spectral recovery and constrained matrix factorizations: Concepts and applications. J. Chemom., 24:67–74, 2010.

[12] R. Rajkó and K. István. Analytical solution for determining feasible regions of self-modeling curve resolution (SMCR) method based on computational geometry. J. Chemom., 19(8):448–463, 2005.

[13] M. Sawall, C. Fischer, D. Heller, and K. Neymeyr. Reduction of the rotational ambiguity of curve resolution techniques under partial knowledge of the factors. Complementarity and coupling theorems. J. Chemom., 26:526–537, 2012.

[14] M. Sawall, A. Jürß, H. Schröder, and K. Neymeyr. On the analysis and computation of the area of feasible solutions for two-, three- and four-component systems, volume 30 of Data Handling in Science and Technology, "Resolving Spectral Mixtures", Ed. C. Ruckebusch, chapter 5, pages 135–184. Elsevier, Cambridge, 2016.

[15] M. Sawall, A. Jürß, and K. Neymeyr. FAC-PACK: A software for the computation of multi-component factorizations and the area of feasible solutions, Revision 1.2. FAC-PACK homepage: http://www.math.uni-rostock.de/facpack/, 2014.

[16] M. Sawall, C. Kubis, D. Selent, A. Börner, and K. Neymeyr. A fast polygon inflation algorithm to compute the area of feasible solutions for three-component systems. I: Concepts and applications. J. Chemom., 27:106–116, 2013.

[17] M. Sawall and K. Neymeyr. A fast polygon inflation algorithm to compute the area of feasible solutions for three-component systems. II: Theoretical foundation, inverse polygon inflation, and FAC-PACK implementation. J. Chemom., 28:633–644, 2014.

[18] M. Sawall and K. Neymeyr. How to compute the area of feasible solutions, a practical study and users' guide to FAC-PACK. In: M. Khanmohammadi, ed. volume in Current Applications of Chemometrics, chapter 6. Nova Science Publishers, New York, 97-134, 2014.

Erklärung

Ich versichere hiermit, dass ich die vorliegende Arbeit selbstständig angefertigt und ohne fremde Hilfe verfasst habe. Dazu habe ich keine außer den von mir angegebenen Hilfsmitteln und Quellen verwendet und die den benutzten Werken inhaltlich und wörtlich entnommenen Stellen habe ich als solche kenntlich gemacht.

Rostock, den 20.06.2018

Tomass Andersons