Mathematik-Verknüpfung von 2D– und 3D–Punktwolken

Dissertation

zur Erlangung des akademischen Grades
"Doktor der Naturwissenschaften"

am Fachbereich
Mathematik und Informatik, Physik, Geographie
der Justus-Liebig-Universität Gießen

vorgelegt von
Thomas Maresch

Gießen 2006

Inhaltsverzeichnis

1 Einleitung **1**

2 Notation **4**

I Theoretische Grundlagen **8**

3 Splines **8**
3.1 Einleitung 8
3.2 Stückweise polynomiale Kurven 9
3.3 Definition der B-Splines 10
3.4 Differenzierbarkeit von B-Splines und Splinekurven 15
3.5 B-Splines als Basis eines Splineraums 17
3.6 Interpolation und Approximation mit Splines 18
3.7 Approximation mit Splines 20

4 Wavelets **23**
4.1 Einleitung 23
4.2 Fouriertransformation 24
4.3 Kontinuierliche Wavelettransformation 28
4.4 Stetige Waveletbasen 34
4.5 Diskrete Wavelettransfomation 36
4.6 Multiresolution Analysis 39
4.7 Schnelle Wavelettransformation 46
4.8 Wavelets mit kompaktem Träger 50
4.9 Biorthogonale Wavelets 54

5 Radiale Basisfunktionen **60**
5.1 Definition und Eigenschaften 60
5.2 Thin-Plate Spline 62
5.3 Multiquadrik und Inverse Multiquadrik 63
5.4 Weitere radiale Basisfunktionen 64
5.5 Anwendungsgebiete für radiale Basisfunktionen 64

II Anwendungen **65**

6 Grundlagen der Koordinatenmesstechnik **65**

7 Konturmathematik **68**
7.1 Problemstellung 68
7.2 Geometrische Eckenbestimmung 72
7.3 Eckenbestimmung mit Wavelets 76
7.4 Kriterium zur Unterscheidung von Singularitäten 85
7.5 Zusammenspiel der Algorithmen zur Eckenerkennung . . . 97
7.6 Interpolation der Konturen 98
7.7 Newton-Verfahren 105
7.8 Startwertberechnung 106
7.9 Das modifizierte Newton-Verfahren 119
7.10 Berechnung der Extremalpunkte 123

8 Regelgeometrien **129**
8.1 Problemstellung Ausgleichselemente 129
8.2 Ausgleichsellipse, -hyperbel und -parabel 131
8.3 Hüllkreis 143
8.4 Pferchkreis 153

9 Filter **173**
9.1 Einleitung 173
9.2 Klassische Rauheitsfilter 173
9.3 Filter auf Splinebasis 177
9.4 Filter auf Wavelet-Basis 186
9.5 Vergleich der Filterverfahren 189
9.6 Filter für Fasertaster-Konturen 200

10 Behandlung von Flächen **208**
10.1 Approximation von Flächen durch radiale Basisfunktionen 208
10.2 Fast-Multipole Methode 214
10.3 Lösung des linearen Gleichungssystems (10.6) 245
10.4 Gesamtalgorithmus zur Flächenberechnung 252
10.5 Ein lokales Glättungsverfahren 256

1 Einleitung

„Miss alles, was sich messen lässt, und mach alles messbar, was sich nicht messen lässt.“ [*Galileo Galilei*]
„Die Natur spricht die Sprache der Mathematik: die Buchstaben dieser Sprache sind Dreiecke, Kreise und andere mathematische Figuren.“ [*Galileo Galilei*]

Die Zitate von Galileo Galilei charakterisieren die Aufgaben der Koordinatenmesstechnik ziemlich genau. Es werden zu verschiedenen Zwecken Werkstücke oder deren Teile vermessen. Sind manche Werkstücke heutzutage noch nicht messbar, weil sie beispielsweise zu klein oder zu groß sind, gehen die Entwicklungen dahin, neue Verfahren zu finden, um auch diese Werkstücke messen zu können. Dabei bedeutet „Messen“, dass wir real existierende Gegenstände in eine einheitliche Sprache übersetzen, nämlich in die Sprache der Mathematik.
Allerdings lassen sich reale Objekte mit nahezu beliebigen Geometrien nicht immer problemlos mit einfachen mathematischen Figuren wie Geraden, Dreiecken, Kreisen im Zweidimensionalen oder Kugeln, Zylindern und Kegeln im Dreidimensionalen ausdrücken. Die Ergebnisse der Messungen solcher komplexer Gegenstände werden in der Koordinatenmesstechnik *Konturen* oder *Punktwolken* genannt.
Wir teilen dabei Konturen oder Punktwolken in zwei Kategorien auf. Die erste Kategorie enthält eindimensionale Wege im $\mathbb{R}^d$, während zu der zweiten Flächen gehören. In der Sprechweise der Koordinatenmesstechnik wird aber grundsätzlich zwischen den beiden Kategorien nicht unterschieden. Wir verständigen uns für diese Arbeit auf einen Kompromiss und nennen eindimensionale Wege *Konturen*, während wir eine *Fläche* auch als solche bezeichnen. Beide Punktmengen fallen unter den Oberbegriff *Punktwolken*.
Das Gebiet der Koordinatenmesstechnik hat seit dem Anfang der neunziger Jahre, ähnlich wie die Computertechnik, äußerst kurze Innovationszyklen. Der Zusammenhang zwischen diesen Gebieten ist offensichtlich, denn die Messergebnisse eines Koordinatenmessgerätes werden an einem Computer aufbereitet und ausgewertet. Die Entwicklungen auf dem Gebiet der Computertechnik ermöglichen eine Verarbeitung von immer größeren Punktwolken, also immer genauere Messungen. Dadurch entsteht die Notwendigkeit, solche Punktmengen effizienter zu verarbeiten, damit ein flüssiger Ablauf der Messsoftware auch bei großen Konturen und Flächen gewährleistet wird.

In der vorliegenden Arbeit beschäftigen wir uns mit Lösungen einiger Fragen aus dem Gebiet der Koordinatenmesstechnik. Wir können die Probleme vier größeren Gebieten zuordnen.

(1) Wir beginnen mit der Frage nach einer effizienten Interpolation einer Kontur. Die Aufgabe wird durch verschiedene Nebenbedingungen erschwert, die an eine solche Kontur gestellt werden. So kann eine Kontur aus mehreren, räumlich getrennten Teilkonturen bestehen. Darüber hinaus kann die Kontur Ecken enthalten, die als solche erkannt und modelliert werden müssen. Und zuletzt kann die Kontur verrauscht sein, was aber weder die Interpolation noch die Eckendetektion beeinflussen darf. Mit einer auf diese Weise interpolierten Kontur berechnen wir Abstände zu so genannten *Regelgeometrien*, das sind wohldefinierte geometrische Objekte, wie Geraden, Kreisen und andere geometrische Figuren.

(2) Anschließend beschäftigen wir uns mit der Anpassung einer Kontur an ein Regelgeometrieelement. Die wohl gebräuchlichsten und bekanntesten Elemente sind die Ausgleichsgerade und der Ausgleichskreis. Wir geben Verfahren zur Berechnung einer Ausgleichshyperbel, -ellipse und -parabel sowie zur Berechnung eines Hüll- und Pferchkreises an.

(3) Das nächste Gebiet, welches wir beschreiben, ist die Filterung von Konturen. Wir stellen die klassischen, nach ISO zertifizierten Filter den modernen Waveletfiltern sowie einem Splinefilter gegenüber. Dabei gehen wir vor allem auf die Vor- und Nachteile der einzelnen Verfahren ein und bewerten die Ergebnisse anhand von Testkonturen.

(4) Zuletzt beschäftigen wir uns mit Flächen. Wir beschreiben zwei Methoden zur Glättung von Flächen. Dabei greifen wir nicht auf die Standardverfahren, wie Tensorprodukt-Splines, NURBS oder Subdivision-Verfahren, sondern auf die radialen Basisfunktionen zurück.

Die Arbeit ist in zwei Teile gegliedert. Der erste Teil umfasst die Kapitel 2 bis 5. Zunächst legen wir unsere Notation fest, dann folgt ein Kapitel über Splinekurven und deren Bestandteile, die Basis- oder B-Splines. Wir formulieren wichtige Sätze über Ableitungen und Interpolation und geben einige Beispiele an.
In Kapitel 4 führen wir Wavelets ein. Dabei erläutern wir neben den schon klassischen Wavelets, wie dem Haar-Wavelet oder den Daubechies-Wavelets,

auch die symmetrischen biorthogonalen Spline-Wavelets, welche für die Detektion von Singularitäten von entscheidender Bedeutung sind.
Ein Kapitel über die Grundlagen der radialen Basisfunktion schließt die Grundlagen ab.
Der Inhalt des zweiten Teils der Arbeit ist ganz den Anwendungen der im ersten Teil eingeführten Techniken gewidmet. In Kapitel 7 stellen wir einen Algorithmus zur Interpolation von Konturen mit Eckenerkennung vor. Dabei beschreiben wir ein Verfahren, mit dem wir in der Lage sind, Singularitäten anhand des Verhaltens der Waveletkoeffizienten zu unterscheiden.
Kapitel 8 beinhaltet ein Verfahren zur Anpassung von Konturen an Regelgeometrieelemente. Die Berechnung einer Ausgleichshyperbel bildet den Anfang des Kapitels. Anschließend besprechen wir Algorithmen zur Berechnung eines Hüll- und eines Pferchkreises.
In Kapitel 9 beschäftigen wir uns mit Rauheitsfiltern und dabei insbesondere mit dem Vergleich aktuell gebräuchlicher Methoden mit den Wavelet- und Splinefiltern.
Ein Kapitel über die Glättung von Flächen bildet den Abschluss dieser Arbeit. Wir stellen hier Verfahren vor, welche die Approximation einer Fläche mit Hilfe radialer Basisfunktionen erheblich beschleunigen.

Ohne die Hilfe vieler Menschen hätte diese Arbeit nicht entstehen können. Ihnen allen, die direkt oder indirekt zum Gelingen der Arbeit beigetragen haben, möchte ich herzlich danken. Herrn Prof. Dr. Tomas Sauer gilt mein besonderer Dank für die intensive Betreuung und die Anregungen zu den vielfältigen Themen dieser Arbeit. Ferner möchte ich mich bei Herrn Dr. Ralph Christoph von *Werth Messtechnik GmbH* für die Finanzierung dieser Arbeit bedanken sowie bei den Herren Dr. Wolfgang Rauh und Dr. Benedikt Feldhaus, ebenfalls von *Werth Messtechnik GmbH*, für die interessanten Fragestellungen und die Betreuung in den vergangenen Jahren. Besonderer Dank gebührt auch meinen Korrekteuren Frau Anika Kurz und Herrn Andreas Kurz. Für die Unterstützung während meines gesamten Studiums danke ich meiner Familie sehr.

2 Notation

Wir legen in diesem Abschnitt unsere Notation fest und stellen gleichzeitig einige der von uns verwendeten Hilfsmittel, wie Funktionenräume und Normen, vor.
Wir arbeiten mit den üblichen Zahlenmengen, der Menge der natürlichen Zahlen $\mathbb{N}$, dem Ring der ganzen Zahlen $\mathbb{Z}$ und den Körpern $\mathbb{R}$ und $\mathbb{C}$.
Wir bezeichnen Zahlen aus $\mathbb{N}, \mathbb{Z}, \mathbb{Q}, \mathbb{R}, \mathbb{C}$ mit kleinen oder großen lateinischen Buchstaben, also zum Beispiel $n \in \mathbb{N}$ oder $a, B \in \mathbb{R}$.
Für ein $x \in \mathbb{R}$ ist

$$\lfloor x \rfloor = \max_{n \in \mathbb{Z}} \{n \leq x\}$$

die *Gaußklammer*.
Ähnlich ist die *Aufrundungsfunktion* (oder *ceil function*) definiert. Hierbei gilt für ein $x \in \mathbb{R}$

$$\lceil x \rceil = \min_{n \in \mathbb{Z}} \{n \geq x\}.$$

Für eine Menge $M \neq \emptyset$ bezeichnen wir mit $|M|$ die *Mächtigkeit von* M und schreiben $|M|_0$ für die Anzahl der Elemente von M, die ungleich Null sind, also

$$|M|_0 = |M \backslash \{0\}| = |S| \text{ mit } S \subseteq M \text{ und } S = \{m \in M \mid m \neq 0\}.$$

Für $M = \{m_j \mid j = 0, \ldots, N-1\}$ ist

$$j_0 = \operatorname{argmin}\{m_j \mid j = 0, \ldots, N-1\} \Leftrightarrow m_{j_0} = \min\{m_j \mid j = 0, \ldots, N-1\}.$$

Analog definieren wir argmax.
Vektoren und Matrizen oder allgemein mehrdimensionale Objekte schreiben wir fett, also $\mathbf{a} \in \mathbb{R}^d$ bzw. $\mathbf{B} \in \mathbb{R}^{m \times n}$ und in Komponentenschreibweise

$$\mathbf{a} = \begin{bmatrix} a^{(1)} \\ \vdots \\ a^{(d)} \end{bmatrix} \in \mathbb{R}^d \text{ bzw. } \mathbf{B} = \begin{bmatrix} b_{11} & \ldots & b_{1n} \\ \vdots & \ddots & \vdots \\ b_{m1} & \ldots & b_{mn} \end{bmatrix} \in \mathbb{R}^{m \times n}.$$

Für eine Menge bzw. Sequenz (zum Beispiel einen Polygonzug)

$$X = \{\mathbf{x}_j \in \mathbb{R}^d \mid j = 0, \ldots, N-1\} \text{ bzw. } \mathbf{X} = (\mathbf{x}_j \in \mathbb{R}^d \mid j = 0, \ldots, N-1)$$

bezeichnen wir mit $X^{(i)}$ bzw. $\mathbf{X}^{(i)}$, $1 \leq i \leq d$, die Menge bzw. Sequenz der i–ten Komponenten. Es gilt also

$$X^{(i)} = \{\mathbf{x}_j^{(i)} \in \mathbb{R}^d \mid j = 0, \ldots, N-1\} \text{ bzw.}$$

$$\mathbf{X}^{(i)} = (\mathbf{x}_j^{(i)} \in \mathbb{R}^d \mid j = 0, \ldots, N-1).$$

Eindimensionale Funktionen und Abbildungen bezeichnen wir entsprechend als

$$f : \mathbb{R}^d \to \mathbb{R} \text{ mit } f(\mathbf{x}) = y \text{ für } \mathbf{x} \in \mathbb{R}^d \text{ und } y \in \mathbb{R},$$

während wir vektor- und matrixwertige Funktionen und Abbildungen als

$$\mathbf{g} : \mathbb{R}^d \to \mathbb{R}^e \text{ mit } \mathbf{g}(\mathbf{x}) = \mathbf{y} \text{ für } \mathbf{x} \in \mathbb{R}^d \text{ und } \mathbf{y} \in \mathbb{R}^e$$

schreiben. Für $\mathbf{a}, \mathbf{b} \in \mathbb{R}^d$ ist

$$\mathbf{a}^t\mathbf{b} = \sum_{j=1}^{d} a^{(j)}b^{(j)} \text{ bzw. } \|\mathbf{a}\|_2 = \sqrt{\mathbf{a}^t\mathbf{a}} = \sqrt{\sum_{j=1}^{d} \left(a^{(j)}\right)^2}$$

das *innere oder skalare Produkt* von $\mathbf{a}$ und $\mathbf{b}$ bzw. die *euklidische oder* 2*–Norm* eines Vektors $\mathbf{a} \in \mathbb{R}^d$.
Für $\mathcal{V} = \{\mathbf{a}_j \mid j = 0, \ldots, N-1\} \subset \mathbb{R}^d$ ist

$$[\mathcal{V}] = [\mathbf{a}_j \mid j = 0, \ldots, N-1] = \left\{ \sum_{j=0}^{N-1} \alpha_j \mathbf{a}_j \mid \alpha_j \geq 0, \sum_{j=0}^{N-1} \alpha_j = 1 \right\}$$

die *konvexe Hülle* von $\mathcal{V}$.
Wir schreiben $\mathbf{I}_N \in \mathbb{R}^{N\times N}$ für die *Einheitsmatrix* der Dimension $N \times N$. Es gilt also $i_{jj} = 1$ für $j = 1, \ldots, N$ und $i_{jk} = 0$ für $j \neq k$. Wir bezeichnen eine Matrix $\mathbf{A} \in \mathbb{R}^{N\times N}$ als (p, q)*–bandiert*, falls gilt

$$a_{jk} = 0 \text{ für } j > k + p \text{ und } k > j + q.$$

Für zwei Matrizen $\mathbf{A} \in \mathbb{R}^{N\times M}$ und $\mathbf{B} \in \mathbb{R}^{K\times L}$ bezeichnen wir mit $\mathbf{A} \otimes \mathbf{B}$ das *Kronecker-Produkt*. Dabei gilt

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{11}\mathbf{B} & \ldots & a_{1M}\mathbf{B} \\ \vdots & \ddots & \vdots \\ a_{N1}\mathbf{B} & \ldots & a_{NM}\mathbf{B} \end{bmatrix} \in \mathbb{R}^{NK\times ML}.$$

Unter einem *Multiindex* verstehen wir einen Vektor $\boldsymbol{\alpha} = [\alpha_0, \ldots, \alpha_d]^t \in \mathbb{N}_0^{d+1}$ aus nichtnegativen ganzen Zahlen. Die Länge von $\boldsymbol{\alpha}$ ist definiert als

$$|\boldsymbol{\alpha}| = \sum_{j=0}^{d} \alpha_j.$$

Die Menge

$$\Gamma_n = \{\boldsymbol{\alpha} \in \mathbb{N}_0^{d+1} \mid |\boldsymbol{\alpha}| = n\}$$

fasst alle Multiindizes der Länge $n \in \mathbb{N}_0$ zusammen.
Für einen Multiindex $\boldsymbol{\alpha} \in \mathbb{N}_0^{d+1}$ definieren wir den *Multinomialkoeffizienten* als

$$m(\boldsymbol{\alpha}) = \frac{|\boldsymbol{\alpha}|!}{\alpha_0! \cdots \alpha_d!}.$$

Häufig verwenden wir die abkürzenden Schreibweise

$$m(\boldsymbol{\alpha}) = \binom{|\boldsymbol{\alpha}|}{\boldsymbol{\alpha}}.$$

Wir setzen den Multinomialkoeffizienten auf $\mathbb{Z}^{d+1}$ fort, indem wir $m(\boldsymbol{\alpha})$ gleich Null setzen, wenn $\boldsymbol{\alpha}$ eine negative Komponente besitzt.
Wir verwenden verschiedene Funktionenräume, die wir samt den zugehörigen Normen, sofern diese für uns von Interesse sind, hier auflisten. Seien im Folgenden $I \subseteq \mathbb{R}$ ein kompaktes Intervall, $k \in \mathbb{N}$, $n \in \mathbb{N}$ und $1 \leq p < \infty$.

$$\begin{aligned}
\mathcal{C}(I) &= \{f : I \to \mathbb{R} \mid f \text{ ist stetig}\},\\
\mathcal{C}^k(I) &= \{f : I \to \mathbb{R} \mid f \text{ ist } k\text{–mal stetig differenzierbar}\},\\
\mathcal{C}_{00}^k(\mathbb{R}) &= \{f \in \mathcal{C}^k(\mathbb{R}) \mid f \text{ hat kompakten Träger}\},\\
\mathcal{C}^\infty(\mathbb{R}) &= \{f : \mathbb{R} \to \mathbb{R} \mid f \text{ ist unendlich oft stetig differenzierbar}\},\\
\mathcal{C}_{00}^\infty(\mathbb{R}) &= \{f \in \mathcal{C}^\infty(\mathbb{R}) \mid f \text{ hat kompakten Träger}\},\\
L_p(\mathbb{R}) &= \{f : \mathbb{R} \to \mathbb{R} \mid \int_{\mathbb{R}} |f(x)|^p \, dx < \infty\},\\
&\qquad \text{mit der Norm } \|f\|_p = \left(\int_{\mathbb{R}} |f(x)|^p \, dx\right)^{1/p},
\end{aligned}$$

$$\begin{aligned}
L_p(I) &= \{f : I \to \mathbb{R} \mid \int_I |f(x)|^p \, dx < \infty\},\\
&\qquad \text{mit der Norm } \|f\|_{I,p} = \left(\int_I |f(x)|^p \, dx\right)^{1/p},\\
L_p^n(\mathbb{R}) &= \{f \in L_p(\mathbb{R}) \mid f^{(n)} \in L_p(\mathbb{R})\},\\
\Pi_m &= \operatorname{span}\{x^j \mid j = 0, \ldots, m\} \subset \mathcal{C}^\infty(I)\\
&\qquad \text{Raum der univariaten Polynome vom Grad höchstens } m,\\
\Pi_{m,d} &= \text{ Raum der multivariaten Polynome vom Grad höchstens } m \text{ in } \mathbb{R}^d.
\end{aligned}$$

Für $f, g \in L_p(\mathbb{R})$ definieren wir das *Skalarprodukt* von f und g als

$$\langle f, g \rangle_{L_p} = \langle f, g \rangle = \int_{\mathbb{R}} f(x)g(x)dx.$$

Meist liegen die zu bearbeitenden Daten nicht kontinuierlich sondern diskret vor. Daher ist die Betrachtung der Funktionen aus $L_p(\mathbb{R})$ eher von theoretischer Bedeutung. Für die Praxis sind die Räume $\ell(\mathbb{Z})$ bzw. $\ell_p(\mathbb{Z})$ der doppelt unendlichen Folgen gefragt, die wir wie folgt definieren

$$\ell(\mathbb{Z}) = \{c : \mathbb{Z} \to \mathbb{R}\} \text{ und } \ell_p(\mathbb{Z}) = \{c \in \ell(\mathbb{Z}) \mid \|c\|_{\ell_p} < \infty\}$$

mit der Norm

$$\|c\|_{\ell_p} = \left(\sum_{j \in \mathbb{Z}} |c(j)|^p \right)^{1/p}.$$

Für die Komplexitäts- und Laufzeitbetrachtungen bezeichnen wir mit $O(\cdot)$ die *Landau-Symbole*. Dabei gilt

$$f(x) = O(g(x)) \text{ genau dann, wenn } 0 \leq \limsup_{x \to \infty} \left| \frac{f(x)}{g(x)} \right| < \infty.$$

$O(\cdot)$ stellt also die asymptotische obere Schranke dar. Wir verwenden die Sprechweise, ein Algorithmus besitzt eine Komplexität oder Laufzeit von $O(\cdot)$ oder auch ein Algorithmus benötigt $O(\cdot)$ Rechenoperationen oder Rechenschritte.

Teil I
Theoretische Grundlagen

3 Splines

3.1 Einleitung

Die abkürzend als B-Splines bezeichneten Basis-Splines wurden 1946 von *I. Schoenberg* eingeführt und definiert. Dabei verwendete *I. Schoenberg* B-Splines mit äquidistanten Knoten. B-Splines mit nicht-äquidistanten Knoten wurden zuerst 1947 von *H. Curry* eingeführt. Anfang der sechziger Jahre begann *C. de Boor*, die B-Splines als Werkzeug für geometrische Darstellungen zu nutzen und entwarf neben dem de Boor-Algorithmus auch die rekursive Formel der B-Splines.
Es war eben diese rekursive Darstellung, welche die B-Splines zu einem wichtigen Werkzeug für das als CAGD (*Computer Aided Geometric Design*) bekannte Gebiet der Mathematik machte. Im CAGD sind parametrisierte Splinekurven von fundamentaler Bedeutung. Sie wurden 1974 von *R. Riesenfeld* und *W. Gordon* eingeführt. Dabei wurde festgestellt, dass die rekursive Form von de Boor die natürliche Verallgemeinerung des Algorithmus von *de Casteljau* darstellt. Danach wurden Kurven, die auf B-Splines basierten, zu einem wichtigen Bestandteil der CAD (*Computer Aided Design*)–Systeme. In den siebziger und achtziger Jahren gab viele Veröffentlichungen zum Thema Splinekurven, die Splines wurden zu einem sehr frequentierten Bereich der Mathematik.
Heutzutage werden B-Splines oder allgemein Splinekurven in vielen Bereichen eingesetzt, nach wie vor als NURBS (*Nonuniform Rational B-Splines*) in der CAD/CAM-Industrie und überall dort, wo Interpolation oder Approximation von Datenmengen benötigt wird.
Wir verwenden Splines auf zwei Arten. Zum einen, um Konturen, also Datenpunkte, zu interpolieren und damit für Minimierungsaufgaben eine geschlossene Form zur Verfügung zu haben. Auf der anderen Seite approximieren wir eine Menge von Daten durch eine Splinekurve, um eine Glättung der Daten herbei zu führen.
Wir beschäftigen uns zunächst mit Splinekurven und deren Basen, den B-Splines. Da Splinekurven und in diesem Zusammenhang vor allem die Interpolationseigenschaft der Splines die Basis für die Berechnung der Konturverknüpfungen bilden werden, befassen wir uns eingehend damit.

3.2 Stückweise polynomiale Kurven

Wir beginnen mit einigen allgemeinenen Aussagen über Splinekurven und definieren dann die für uns wichtigen B-Splines.

3.1 Definition: Sei $I = [a, b] \subset \mathbb{R}$ ein kompaktes Intervall mit einer geordneten Knotenfolge der Form

$$a = t_0 < t_1 < \cdots < t_{N-2} < t_{N-1} = b.$$

Eine Funktion $f : I \to \mathbb{R}$ heißt *polynomialer Spline* der Ordnung m und der Glattheit $m-1$ bezüglich der Knotenfolge $T = \{t_j \mid j = 0, \ldots, N-1\}$, falls folgende Bedingungen erfüllt sind.

(1) $f \in \mathcal{C}^{m-1}(I)$

(2) $f|_{(t_j, t_{j+1})} \in \Pi_m$ für $j = 0, \ldots, N-1$.

Ein einfaches Beispiel einer polynomialen Splinekurve erhalten wir folgendermaßen.

3.2 Beispiel: Seien $I = [a, b] \subset \mathbb{R}$, $T = \{t_j \mid j = 0, \ldots, N-1\}$ eine Knotenfolge im Sinne der Definition 3.1 und sei $f \in \mathcal{C}(I)$. Dann ist

$$S_1 f(x) = \frac{t_{j+1} - x}{t_{j+1} - t_j} f(t_j) + \frac{x - t_j}{t_{j+1} - t_j} f(t_{j+1})$$

für $x \in [x_j, x_{j+1})$ und $j = 0, \ldots, N-1$ ein Spline der Ordnung 1 und Glattheit 0.

Weichen wir von unserer in Definition 3.1 formulierten Forderung ab, dass die Knotenfolge streng monoton steigend sein muss, erhalten wir die Möglichkeit, mehrfache Knoten einzufügen. Damit können wir die Differenzierbarkeit an den Verbundstellen der einzelnen polynomialen Teilkurven steuern.
Wir führen die folgende Sprechweise ein.

3.3 Definition: Seien $I = [a, b] \subset \mathbb{R}$ und $T = \{t_j \mid j = 0, \ldots, N-1\}$.

(1) Die Menge T heißt *Knotenfolge* für I, falls $t_0 \leq t_1 \leq \cdots \leq t_{N-1}$ und $t_0 = a$ sowie $t_{N-1} = b$ gilt.

(2) Wir nennen T *einfache Knotenfolge*, falls die Ordnung der t_j in (1) streng ist.

(3) Wir bezeichnen den *Vektorraum der Splinefunktionen der Ordnung m und Glattheit k bezüglich der Knotenfolge T* mit $\mathbb{S}_{m,k}(T)$ für $k \leq m-1$.

Da wir durch mehrfache Knoten eine geringere Differenzierbarkeit erzielen, assoziieren wir mit einer einfachen Knotenfolge den Raum $\mathbb{S}_{m,m-1}$.
Man kann zeigen, dass die Folge der abgebrochenen Potenzen

$$f_n(x) = (x - x_0)_+^n = \begin{cases} 0 & \text{für } x < x_0 \\ (x - x_0)^n & \text{für } x \geq x_0 \end{cases} \text{ für } n \in \mathbb{N} \text{ und } x_0 \in \mathbb{R}$$

zusammen mit Π_m die Basis von $\mathbb{S}_{m,m-1}$ bildet. Allerdings hat diese Basis des Splineraums einen großen Nachteil, die Basisfunktionen haben einen unendlichen Träger, was sich numerisch als schwierig erweist, insbesondere für ein großes N.
Dies führt uns direkt zu den Basic- oder B-Splines, die ebenfalls eine Splineraumbasis bilden, allerdings über einen minimalen Träger verfügen.

3.3 Definition der B-Splines

Die B-Splines liefern uns eine Basis des Splineraums und wir können mit ihrer Hilfe einige wichtige Eigenschaften der Splinefunktionen angeben.
Im ersten Schritt passen wir die Definition der Knotenfolge an.

3.4 Definition: Seien $m, N > 0$, dann gilt.

(1) Eine *Knotenfolge* $T_{m,N-1} = \{t_0, \ldots, t_{N+m}\}$ ist eine Teilmenge von $\mathbb{R}$, die den Bedingungen

(i) $t_0 \leq t_1 \leq \cdots \leq t_N \leq \cdots \leq t_{N+m}$ und

(ii) $t_j < t_{j+m+1}$ für $j = 0, \ldots, N-1$

genügt.

(2) Wir nennen ein $k \in \mathbb{N}$ mit

$$t_{j-1} < t_j = \cdots = t_{j+k-1} < t_{j+k}$$

Vielfachheit des Knotens $t_j = \cdots = t_{j+k-1}$.

(3) Besitzt eine Knotenfolge $m+1$–fache Endknoten, also

$$t_0 = \cdots = t_m \text{ und } t_N = \cdots = t_{N+m},$$

dann schreiben wir $T^*_{m,N-1}$ für diese Knotenfolge.

Ein weiterer benötigter Begriff sind die baryzentrischen Koordinaten, ein in vielen Bereichen der Geometrie verwendetes Konzept. Da wir das Konzept der baryzentrischen Koordinaten nur univariat verwenden, beschränken wir uns auf diese Definition.

3.5 Definition: Sei $\Delta = [a,b] \subset \mathbb{R}$ ein Intervall. Die *baryzentrischen Koordinaten* eines Punktes $x \in \mathbb{R}$ bezüglich Δ sind definiert als

$$\mathbf{u}(x|\Delta) = (u_0(x|\Delta), u_1(x|\Delta)) \in \mathbb{R}^2 \tag{3.1}$$

mit

$$x = u_0(x|\Delta)a + u_1(x|\Delta)b \text{ und } u_0(x|\Delta) + u_1(x|\Delta) = 1.$$

Mit dem Begriff einer Knotenfolge und der baryzentrischen Koordinaten können wir den Algorithmus von *de Boor* angeben. Bei diesem Algorithmus handelt es sich um eine Verallgemeinerung des Algorithmus von *de Casteljau*.

3.6 Algorithmus: Seien $T_{m,N-1}$ eine Knotenfolge, $\mathbf{D} = \{\mathbf{D}_j \mid j = 0,\ldots,N-1\} \subseteq \mathbb{R}^d$ und $x \in [t_m, t_N]$. Wir setzen $\Delta^k_j = [t_j, t_{j+k}]$.

(1) Wir bestimmen $i \in \{m,\ldots,N-1\}$, so dass gilt $x \in [t_i, t_{i+1})$.

(2) Wir setzen

$$\mathbf{D}^0_j(x) = \mathbf{D}_j \text{ für } j = i-m,\ldots,i.$$

(3) Wir berechnen für $k = 1,\ldots,m$

$$\mathbf{D}^k_j(x) = u_0(x \mid \Delta^{m-k+1}_j)\mathbf{D}^{k-1}_{j-1}(x) + u_1(x \mid \Delta^{m-k+1}_j)\mathbf{D}^{k-1}_j(x),$$

$$j = i-m+k,\ldots,i.$$

(4) Als Ergebnis erhalten wir $\mathbf{D}^m_i$.

Wir bezeichnen die *Splinekurve* $x \mapsto \mathbf{D}^m_j(x)$ für $x \in [t_m, t_N]$ mit $N_{m,T}\mathbf{D}$ und nennen $\mathbf{D} = (\mathbf{D}_0,\ldots,\mathbf{D}_{N-1})$ das *Kontrollpolygon* von $N_{m,T}\mathbf{D}$.

Die folgende Proposition fasst die wesentlichen Eigenschaften der Splinekurve $N_{m,T}\mathbf{D}$ zusammen.

3.7 Proposition: Sei $T_{m,N-1}$ eine Knotenfolge und $\mathbf{D}$ ein Kontrollpolygon, dann gelten folgende Aussagen.

(1) $N_{m,T}\mathbf{D}([t_j, t_{j+1}]) \subset [\mathbf{D}_{j-k} \mid k = 0, \ldots, m]$ für $j = m, \ldots, N-1$.

(2) Gilt $x = t_{j-m+1} = \cdots = t_j$ für ein $j \geq m$, dann ist $N_{m,T}\mathbf{D}(x) = \mathbf{D}_{j-m}$.

(3) Seien $m = N-1$, $t_0 = \cdots = t_m$ sowie $t_{m+1} = \ldots = t_{2m+1}$ und $x \in [t_0, t_{m+1}]$, dann gilt $N_{m,T}\mathbf{D} = B_m\mathbf{D}$. Also ist die Splinekurve gleich der Bezierkurve, die durch Anwendung des Algorithmus von *de Casteljau* (vgl. [Fa02], Seite 115) erzeugt wird.

(4) Es gilt
$$N_{m,T}\mathbf{D}|_{(t_j,t_{j+1})} \in \Pi_m \text{ für } j = m, \ldots, N-1.$$

(5) Ist $T = T^*_{m,N-1}$ eine Knotenfolge mit $m+1$–fachen Endknoten, so besitzt die Splinekurve die Eigenschaft der Endpunktinterpolation, das heißt
$$N_{m,T}\mathbf{D}(t_m) = \mathbf{D}_0 \text{ und } N_{m,T}\mathbf{D}(t_N) = \mathbf{D}_{N-1}.$$

Bei der Einführung der B-Splines orientieren wir uns an [dB78].

3.8 Definition: Sei $T = T_{m,N-1}$ eine Knotenfolge.

(1) Die *B-Splines der Ordnung 0* sind die charakteristischen Funktionen der von T erzeugten Partition
$$N_j^0(x|T) = \chi_{[t_j,t_{j+1})} = \begin{cases} 1 & \text{für } x \in [t_j, t_{j+1}) \\ 0 & \text{für } x \notin [t_j, t_{j+1}) \end{cases}$$
für $j = 0, \ldots, N+m-1$.

(2) Für $k \geq 1$ gilt
$$N_j^k(x|T) = u_1(x|\Delta_j^k)N_j^{k-1}(x|T) + u_0(x|\Delta_{j+1}^k)N_{j+1}^{k-1}(x|T) \tag{3.2}$$
für $j = 0, \ldots, N+m-k-1$.

3.9 Bemerkung:

(1) Wir sehen sofort, dass die auf diese Weise definierten B-Splines Funktionen mit kompakten Trägern sind. Genauer ist $[t_j, t_{j+m+1}]$ der Träger von N_j^m bezüglich der Knotenfolge $T_{m,N-1}$.

(2) Die B-Splines bilden eine nichtnegative Teilung der Eins, es gilt also

$$\sum_{j=0}^{N-1} N_j^m(\cdot|T) = 1.$$

(3) Ein B-Spline $N_j^k(\cdot|T)$ ist nur dann definiert, wenn $T_{m,N-1}$ Knoten der Vielfachheit höchstens k enthält. Ist die Vielfachheit größer als k, so sind die baryzentrischen Koordinaten in (3.2) nicht definiert. Wir setzen in diesem Fall $N_j^k(\cdot|T) \equiv 0$.

Den Zusammenhang zwischen den B-Splines und der Splinekurve, die durch den Algorithmus von *de Boor* 3.6 erzeugt wurde, beschreibt die folgende Aussage. Dabei wenden wir den Algorithmus von *de Boor* 3.6 auf das Kontrollpolygon an.

3.10 Korollar: Sei $T = T^*_{m,N-1}$ eine Knotenfolge mit $m+1$–fachen Endknoten. Dann gilt

$$N_{m,T}\mathbf{D} = \sum_{j=0}^{N-1} \mathbf{D}_j N_j^m(\cdot|T). \tag{3.3}$$

3.11 Beispiel: Wir wählen die Knotenfolge T derart, dass $t_j = j$ gilt. Abbildung 1 zeigt den B-Spline der Ordnung 1 $N_1^1(\cdot|T)$.
In der nächsten Abbildung sehen wir den B-Spline der Ordnung 3 mit vier verschiedenen Knotenfolgen.
Der B-Spline in Abbildung 2 (a) hat eine Knotenfolge mit einfachen Knoten, während die B-Splines in den Abbildungen (b) bis (d) mehrfache Knoten besitzen. Genauer gilt in (b) $t_1 = t_2 = 1$, also ein zweifacher Knoten, in (c) ein dreifacher Knoten $t_1 = t_2 = t_3 = 1$ und schließlich ein vierfacher Knoten $t_1 = t_2 = t_3 = t_4 = 1$ in (d).

Bevor wir uns der Frage nach der Interpolation mit Hilfe von Splinekurven zuwenden, benötigen wir weitere Resultate, so zum Beispiel die Aussagen

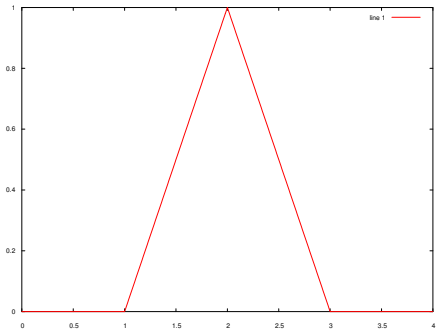

Abbildung 1: *Der B-Spline* $N_1^1(\cdot|T)$.

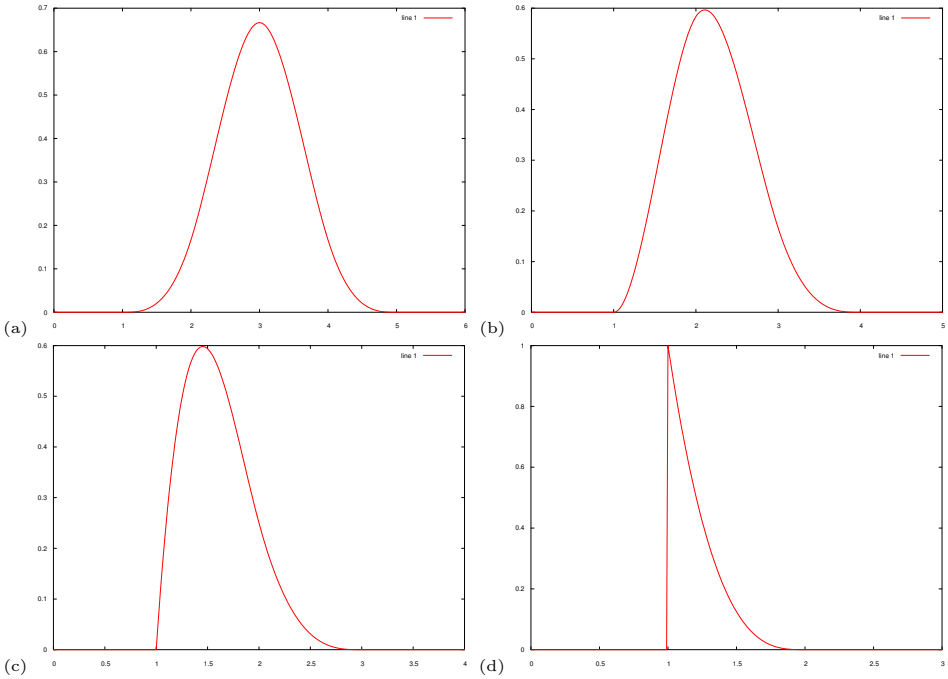

Abbildung 2: *Der B-Spline* $N_1^3(\cdot|T)$ *mit (a) einfachen, (b) einem zweifachen, (c) einem dreifachen und (d) einem vierfachen Knoten.*

zu Differenzierbarkeit und Integrierbarkeit oder nach der Qualität der B-Splines als Basis eines Splineraums.

3.4 Differenzierbarkeit von B-Splines und Splinekurven

Die B-Splines bestehen aus Polynomen vom Grad m. Die Frage ist, ob diese auch m–fach differenzierbar sind. Das folgende Resultat, welches wir in [dB78], Seite 116, finden, beantwortet uns dies.

3.12 Satz: Sei $T = T_{m,N-1}$ eine Knotenfolge. Sei t ein Knoten der Vielfachheit $k \geq 1$, es gilt also $t = t_j$ mit $t_{j-1} < t_j = \cdots = t_{j+k-1} < t_{j+k}$. Dann ist

$$N_l^m(\cdot|T) \in \mathcal{C}^{m-k}(t_{j-1}, t_{j+k}) \text{ für } l = 0, \ldots, N-1.$$

Diese Aussage zeigt eine sehr wichtige Eigenschaft der B-Splines. Wir können durch die Vielfachheit der Knoten die Differenzierbarkeit der Kurve steuern und so zum Beispiel an für uns wichtigen Stellen Ecken oder gar Singularitäten der Kurve modellieren.
Wir können jetzt also den Grad der Differenzierbarkeit eines B-Splines oder einer ganzen Splinekurve angeben. Die Ableitungen können wir wie folgt ausdrücken.

3.13 Satz: Sei $T = T_{m,N-1}$ eine Knotenfolge und t ein Knoten der Vielfachheit $k < m$. Dabei gilt $t_{j-1} < t_j = \cdots = t_{j+k-1} < t_{j+k}$ für $t = t_j$. Dann ist für $x \in (t_{j-1}, t_{j+k})$

$$\left(\frac{d}{dx} N_{m,T}\mathbf{D}\right)(x) = m \sum_{j=0}^{N} \frac{\mathbf{D}_j - \mathbf{D}_{j-1}}{t_{j+m} - t_j} N_j^{m-1}(x|T). \tag{3.4}$$

Dabei setzen wir $\mathbf{D}_{-1} = \mathbf{D} = 0$.

Beweis: Wir können mit Induktion über m zeigen, dass für $x \in (t_{j-1}, t_{j+k})$ und einen Knoten der Vielfachheit $k < m$ gilt

$$\frac{d}{dx} N_j^m(x|T) = \frac{m}{t_{j+m} - t_j} N_j^{m-1}(x|T) - \frac{m}{t_{j+m+1} - t_{j+1}} N_{j+1}^{m-1}(x|T) \tag{3.5}$$

für $j = 0, \ldots, N-1$. Setzen wir (3.5) in die Gleichung einer Splinekurve (3.3) ein, dann ist

$$\begin{aligned}
&\left(\frac{d}{dx} N_{m,T}\mathbf{D}\right)(x) \\
=&\sum_{j=0}^{N-1} \left(\frac{m}{t_{j+m} - t_j} N_j^{m-1}(x|T) - \frac{m}{t_{j+m+1} - t_{j+1}} N_{j+1}^{m-1}(x|T)\right) \mathbf{D}_j \\
=&\sum_{j=0}^{N-1} \frac{m}{t_{j+m} - t_j} \mathbf{D}_j N_j^{m-1}(x|T) - \sum_{j=1}^{N} \frac{m}{t_{j+m} - t_j} \mathbf{D}_{j-1} N_j^{m-1}(x|T) \\
=&m \sum_{j=0}^{N} \frac{\mathbf{D}_j - \mathbf{D}_{j-1}}{t_{j+m} - t_j} N_j^{m-1}(x|T).
\end{aligned}$$

□

Wir wollen Minimierungsverfahren wie das Rang-1-Verfahren von Broyden oder das Newton-Verfahren auf die Splinekurven anwenden, daher benötigen wir noch einen geschlossenen Ausdruck für die zweite Ableitung einer Splinekurve. Wenden wir (3.5) auf $N_j^{m-1}(x|T)$ an, so erhalten wir die folgende Aussage.

3.14 Korollar: Sei $T = T_{m,N-1}$ eine Knotenfolge und t ein Knoten der Vielfachheit $k < m-1$, das heißt $t_{j-1} < t_j = \cdots = t_{j+k-1} < t_{j+k}$ für $t = t_j$. Dann ist für $x \in (t_{j-1}, t_{j+k})$ und $m > 1$

$$\begin{aligned}
\left(\frac{d^2}{dx^2} N_{m,T}\mathbf{D}\right)(x) &= m(m+1) \sum_{j=0}^{N} \frac{\mathbf{D}_j - \mathbf{D}_{j-1}}{t_{j+m} - t_j} \left(\frac{N_j^{m-2}(x|T)}{t_{j+m-1} - t_j} - \frac{N_{j+1}^{m-2}(x|T)}{t_{j+m} - t_{j+1}}\right) \\
&= m(m+1) \sum_{j=0}^{N+1} \left[\frac{(\mathbf{D}_j - \mathbf{D}_{j-1})}{(t_{j+m} - t_j)} - \frac{(\mathbf{D}_{j-1} - \mathbf{D}_{j-2})}{(t_{j+m-1} - t_{j-1})}\right] \frac{N_j^{m-2}(x|T)}{t_{j+m-1} - t_j}
\end{aligned} \tag{3.6}$$

mit $\mathbf{D}_{-2} = \mathbf{D}_{-1} = \mathbf{D}_N = \mathbf{D}_{N+1} = 0$.

Wir wenden uns nun der Fragestellung nach den Eigenschaften der B-Splines als Basis eines Splineraums zu.

3.5 B-Splines als Basis eines Splineraums

Als Erstes definieren wir den Splineraum, dessen Basis unsere B-Splines bilden sollen.

3.15 Definition: Sei $T = T_{m,N-1}$ eine Knotenfolge. Der *Splineraum* $\mathbb{S}_m(T)$ enthält alle Funktionen s, für die gilt

(1)
$$s|_{(t_j,t_{j+1})} \in \Pi_m \text{ für } j = m, \ldots, N-1$$
und

(2) die Funktionen s sind für $j = 0, \ldots, N+m$ an den Knoten t_j, deren Vielfachheit wir mit k_j bezeichnen, $(m-k_j)$–fach differenzierbar.

Der folgende Satz geht auf *Curry* und *Schoenberg* zurück, siehe [dB78], Seite 97

3.16 Satz: (Curry-Schoenberg) Sei $T = T_{m,N-1}$ eine Knotenfolge. Die B-Splines $N_j^m(\cdot|T)$ für $j = 0, \ldots, N-1$ bilden eine Basis des Splineraums $\mathbb{S}_m(T)$.

B-Splines sind nicht nur linear unabhängig, sie sind sogar lokal linear unabhängig.

3.17 Definition: Eine Familie von Funktionen $f_j \in \mathcal{C}(\mathbb{R})$ für ein $j \in J \subset \mathbb{N}$ heißt *lokal linear unabhängig*, wenn für jedes offene Intervall $I = (a,b) \subset \mathbb{R}$ die Funktionen
$$\{f_j|_I \mid f_j|_I \not\equiv 0\}$$
linear unabhängig sind.

Für die B-Splines erhalten wir die folgende Aussage, die wir nicht beweisen werden.

3.18 Lemma: Sei $T = T_{m,N-1}$ eine Knotenfolge. Dann sind die B-Splines $N_j^m(\cdot|T)$ lokal linear unabhängig.

Weitere Eigenschaften, wie das Knoteneinfügen oder Integrale von Splinekurven, spielen für unsere Betrachtungen keine Rolle. Wir beschäftigen uns für den Rest des Kapitels mit der Interpolation und Approximation mit Splines.

3.6 Interpolation und Approximation mit Splines

Wir betrachten in diesem Abschnitt ein allgemeines Interpolationsproblem mit Splinekurven. Die Interpolationspunkte sind dabei nicht notwendig die Knotenpunkte.
Darüber hinaus beschränken wir uns auf den Fall $d = 1$, da wir immer komponentenweise interpolieren und dadurch eine Interpolation in $\mathbb{R}^d$ erreichen können.
Wir stellen als Erstes das Interpolationsproblem auf.

3.19 Definition: Sei $T = T_{m,N-1}$ eine Knotenfolge und die Menge der Abszissen $A = \{a_0, \ldots, a_{N-1}\} \subset [t_m, t_N]$. Wir suchen ein Kontrollpolygon $\mathbf{D} = (D_0, \ldots, D_{N-1})$, so dass für Werte $y_0, \ldots, y_{N-1} \in \mathbb{R}$ gilt

$$N_{m,T}\mathbf{D}(a_j) = y_j \text{ für } j = 0, \ldots, N-1. \tag{3.7}$$

Wir nehmen stets an, dass die Interpolationspunkte, die wir als *Abszissen* bezeichnen, streng monoton steigend sind. Es gilt also

$$a_0 < a_1 < \cdots < a_{N-1}.$$

Schreiben wir das Interpolationsproblem (3.7) aus, so erhalten wir das lineare Gleichungssystem

$$\mathbf{M}_m(A)\mathbf{D} = \mathbf{y}$$

mit der *Kollokationsmatrix* $\mathbf{M}_m(A)$, definiert als

$$\begin{aligned}\mathbf{M}_m(A) &= [N_k^m(a_j) \mid j,k = 0, \ldots, N-1] \\ &= \begin{bmatrix} N_0^m(a_0|T) & \ldots & N_{N-1}^m(a_0|T) \\ \vdots & \ddots & \vdots \\ N_0^m(a_{N-1}|T) & \ldots & N_{N-1}^m(a_{N-1}|T) \end{bmatrix} \in \mathbb{R}^{N\times N},\end{aligned} \tag{3.8}$$

und $\mathbf{y} = [y_0, \cdots, y_{N-1}]^t \in \mathbb{R}^N$.

Offensichtlich ist das Interpolationsproblem (3.7) genau dann lösbar, wenn die Kollokationsmatrix (3.8) invertierbar ist.

Die Frage, wann die Kollokationsmatrix invertierbar ist und wir demzufolge eine eindeutige Lösung des Interpolationsproblems (3.7) haben, beantwortet der folgende Satz, [dB78], Seite 171.

3.20 Satz: (Schoenberg-Whitney) Sei $T = T_{m,N-1}$ eine Knotenfolge mit Vielfachheit höchstens $k < m$. Sei weiter $A = \{a_0, \ldots, a_{N-1}\}$ eine monoton steigende Menge von Abszissen. Die Kollokationsmatrix $\mathbf{M}_m(A)$ ist genau dann invertierbar, wenn gilt

$$t_j < a_j < t_{j+m+1} \text{ für } j = 0, \ldots, N-1. \tag{3.9}$$

Die Schoenberg-Whitney Bedingung zeigt uns, wo die Abszissen liegen müssen, damit wir eine eindeutige Lösung des Interpolationsproblems (3.7) erhalten. Wie wir diese Abszissen praktisch wählen können, erfahren wir aus dem nächsten Lemma.

3.21 Lemma: Sei $T = T^*_{m,N-1}$ eine Knotenfolge mit $(m+1)$–fachen Randknoten. Dann ist das Spline-Interpolationsproblem an den Punkten

$$a_j = \frac{1}{m+2} \sum_{l=0}^{m+1} t_{j+l} \text{ für } j = 0, \ldots, N-1 \tag{3.10}$$

eindeutig lösbar.

Beweis: Wir müssen für die Abszissen a_j die Bedingung (3.9) nachweisen. Alle inneren Knoten haben höchstens die Vielfachheit $m+1$. Sei $t_j = \cdots = t_{j+k}$ ein Knoten der Vielfachheit $k \leq m+1$. Dann gilt

$$a_j = \frac{1}{m+2} \sum_{l=0}^{m+1} t_{j+l} = \frac{1}{m+2}((k+1)t_j + t_{j+k+1} + \cdots + t_{j+m+1}).$$

Wegen $t_j < t_{j+k+1}$ gilt $a_j \in (t_j, t_{j+m+1})$. □

Die nächste Eigenschaft, die es nachzuweisen gilt, betrifft die Kollokationsmatrix $\mathbf{M}_m(A)$ und ist vor allem für das spätere Lösen des Gleichungssystems wichtig. Die Kollokationsmatrix hat eine besonders einfache Form, die uns das Anwenden effizienter Algorithmen ermöglicht.

3.22 Definition: Sei $\mathbf{A} \in \mathbb{R}^{N\times N}$ eine Matrix. Seien $I, J \subset \{1,\ldots,N\}$ mit $|I| = |J| = k$. Mit

$$\mathbf{A}(I,J) = \begin{bmatrix} a_{i_1,j_1} & \ldots & a_{i_1,j_k} \\ \vdots & \ddots & \vdots \\ a_{i_k,j_1} & \ldots & a_{i_k,j_k} \end{bmatrix}$$

bezeichnen wir die Submatrix von $\mathbf{A}$ bezüglich $I = \{i_1,\ldots,i_k\}$ und $J = \{j_1,\ldots,j_k\}$, wobei wir stets $i_1 < \cdots < i_k$ und $j_1 < \cdots < j_k$ fordern.
Wir nennen eine Matrix $\mathbf{A}$ *total positiv*, falls gilt

$$\det \mathbf{A}(I,J) \geq 0 \text{ für } I, J \subset \{1,\ldots,N\} \text{ mit } |I| = |J| \neq 0.$$

Der nachfolgende Satz beschreibt vollständig die Struktur der Kollokationsmatrix $\mathbf{M}_m(A)$.

3.23 Satz: Die Kollokationsmatrix $\mathbf{M}_m(A)$ aus (3.8) ist total positiv und $(m+1, m+1)$–bandiert, [dB78], Seite 174.

Bevor wir den Abschnitt über die Grundlagen der Splines abschließen, diskutieren wir noch die Approximationseigenschaften von Splinekurven.

3.7 Approximation mit Splines

Das Hauptresultat dieses Abschnitts ist die Aussage, dass sich Splines gut dazu eignen, zweimal stetig differenzierbare Funktion zu approximieren.
Wir definieren uns als Erstes ein Werkzeug, mit dessen Hilfe wir Aussagen über die Approximationsgüte der Splines machen können.

3.24 Definition: Seien $f \in \mathcal{C}[a,b]$ und $T = T^*_{m,N-1}$ eine Knotenfolge, so dass $t_0 = \cdots = t_m = a$ und $t_N = \cdots = t_{N+m} = b$ gilt. Der *Schoenbergsche Operator* $S_m f$ zu f und T ist für $m \geq 1$ definiert als

$$S_m f = \sum_{j=0}^{N-1} f\left(\frac{t_{j+1} + \cdots + t_{j+m}}{m}\right) N_j^m(\cdot|T). \tag{3.11}$$

Mit Hilfe des Schoenbergschen Operators sind wir in der Lage, die Approximationsgüte der Splines zu bewerten.

3.25 Satz: Seien $T = T^*_{m,N-1}$ eine Knotenfolge und $f \in \mathcal{C}^2[a,b]$, wobei $a = t_m$ und $b = t_N$ gilt. Dann ist

$$\|f - S_m f\|_{[a,b]} \leq m^2 \|f''\|_{[a,b]} h^2 \tag{3.12}$$

mit

$$h = \max_{j=0,\ldots,N+m-1} |t_{j+1} - t_j| \,.$$

Beweis: Wir betrachten $I_j = [t_j, t_{j+1})$ und die Einschränkung von $S_m f_{I_j}$ auf I_j. Sei $t^* = \frac{1}{2}(t_{j-m+1} + t_{j+m})$, dann hat die Taylorentwicklung von f um t^* die Gestalt

$$f(x) = \underbrace{f(t^*) - (x - t^*) f'(t^*)}_{T_1 f} + \frac{(x-t^*)^2}{2} f''(\xi) \text{ für ein } \xi \in (x, t^*).$$

Wir erhalten

$$|f(x) - T_1 f(x)| \leq \frac{m^2 h^2}{2} \|f''\|_{[a,b]} \tag{3.13}$$

für $x \in I_j$ und damit auch

$$|f - T_1 f| \left(\frac{t_{j+1} + \cdots + t_{j+m}}{m} \right) \leq \frac{m^2 h^2}{2} \|f''\|_{[a,b]}. \tag{3.14}$$

Wir setzen (3.14) in die Gleichung für $S_m f|_{I_j}$ ein und erhalten

$$S_m(f - T_1 f)|_{I_j} \leq \frac{m^2 h^2}{2} \|f''\|_{[a,b]}.$$

Wegen $T_1 f = S_m T_1 f$ gilt $\|f - S_m f\|_{J_j} \leq m^2 h^2 \|f''\|_{[a,b]}$ und damit auch

$$\|f - S_m f\|_{[a,b]} = \max_{j=m,\ldots,N-1} \|f - S_m f\|_{I_j} \leq m^2 h^2 \|f''\|_{[a,b]}.$$

□

Wie messen wir aber die Güte einer Approximation? Zu diesem Zweck definieren wie die Approximationsordnung eines Splineraums $\mathbb{S}_m(T)$ bezüglich einer Knotenfolge $T = T_{m,N-1}$.

3.26 Definition: Sei $T = T_{m,N-1}$ eine Knotenfolge. Wir bezeichnen

$$e_m(T) = \max_{\{f \in \mathcal{C}^k[a,b] \mid \|f^{(k)}\|_{[a,b]} > 0\}} \frac{\min_{s \in \mathbb{S}_m(T)} \|s - f\|_{[a,b]}}{\|f^{(k)}\|_{[a,b]}}$$

als *Approximationsordnung* von $\mathbb{S}_m(T)$ bezüglich des Raumes $\mathcal{C}^k[a,b]$.

Dann haben die Splineräume $\mathbb{S}_m(T)$ in $\mathcal{C}^2[a,b]$ die Approximationsordnung m^2h^2. Da m^2 mit wachsender Ordnung immer größer wird, ist die lineare Approximation optimal - zumindest theoretisch.

4 Wavelets

4.1 Einleitung

Die Theorie der Wavelets und der Wavelettransformation ist ein relativ junger Bereich der Mathematik. Um 1910 verwendete *A. Haar* die charakteristische Funktion für einen multiresolution-ähnlichen Ansatz. Das so genannte *Haar-Wavelet*, das nach ihm benannt wurde, ist das einfachste bekannte Wavelet und findet heute noch häufig Verwendung in der Praxis. Die Grundidee der Wavelettransformation stammt allerdings erst von *A. Calderón* aus dem Jahre 1964. Ihre große Bedeutung in der Signal- und Bildverarbeitung haben die Wavelets aber deutlich später erhalten. So sind sie heute zum Beispiel bei der Datenkompression, beim Entrauschen von Signalen und Bildern oder bei der Regularisierung inverser Probleme nicht mehr wegzudenken. Als allgemeines Beispiel wäre der neue *JPEG200*–Standard der Bildkomprimierung zu nennen, der biorthogonale Wavelets verwenden kann.
Ende der achtziger und Anfang der neunziger Jahre gewannen Wavelets zunehmend an Bedeutung begünstigt durch die neuen Entwicklungen. Damals veröffentlichte *S. Mallat* seinen schnellen Algorithmus für die orthogonale diskrete Wavelettransformation, der auf der Multiresolution Analysis von *Y. Meyer* beruht und der bezüglich der Geschwindigkeit sogar der schnellen Fouriertransformation überlegen ist. Kurze Zeit später bestimmte *I. Daubechies* ihre diskreten orthogonalen Wavelets, welche in Abhängigkeit von der Größe des kompakten Trägers beliebige Regularitätsbedingungen erfüllen. Anfang der neunziger Jahre entwickelten *D. Donoho* und *I. Johnstone* auf Grundlage statistischer Überlegungen Rauschfiltertechniken, die in einem gewissen Sinne optimal sind. Die neuesten Entwicklungen im Bereich der Wavelets und der Wavelettransformationen kann man heute im Internet verfolgen. Beispiele sind Seiten wie
`http://www.math.wustl.edu/wavelet` oder `http://www.wavelet.org`.
In der vorliegenden Arbeit stellen wir zwei Anwendungen der Wavelettransformation in der Koordinatenmesstechnik vor. Zum einen verwenden wir Wavelets zur Detektion von Singularitäten bei einer Kontur und zum anderen stellen wir uns die Frage, ob wir Wavelets zur Rauheitsmessung verwenden können.
Wir beginnen mit einer theoretischen Einführung der Wavelets. Die angesprochenen Anwendungen beschreiben wir in den Abschnitten 7 und 9.

4.2 Fouriertransformation

Wir geben in diesem Abschnitt eine Einführung in die Theorie der Fouriertransformation. Neben der Definition leiten wir wichtige Eigenschaften her und betrachten die gefensterte Fouriertransformation, die ein Grundstein für die Einführung der Wavelets sein wird.

Wir betrachten eine periodische Funktion f mit

$$f(t) = f(t + T_P).$$

Hierbei ist T_P die Länge der Periode. Wir formen diese Funktion mit Hilfe der Transformation

$$x = \frac{2\pi}{T_P} t$$

in eine 2π-periodische Funktion um. Solche Funktionen können wir in einer Fourierreihe entwickeln, deren Definition wir in [Ch92] finden.

4.1 Definition: Sei $f \in L_2([0, 2\pi])$. Wir nennen

$$f(x) = \sum_{j \in \mathbb{Z}} c_j e^{ijx} \tag{4.1}$$

mit

$$c_j = \frac{1}{2\pi} \int_0^{2\pi} f(\nu) e^{-ij\nu} d\nu$$

die *Fourierreihe* von $f(x)$.

Im Folgenden wollen wir uns mit einem der wichtigsten Hilfsmittel bei der Betrachtung der Wavelets beschäftigen, mit der Fouriertransfomierten einer Funktion. Wir orientieren uns dabei stark an [Lo98].

4.2 Definition: Sei $f \in L_1(\mathbb{R})$. Wir definieren die *Fouriertransformierte* $\widehat{f} : \mathbb{R} \to \mathbb{C}$ als

$$\widehat{f}(\xi) = f^\wedge(\xi) = \int_{\mathbb{R}} f(t) e^{-i\xi t} dt \text{ für } \xi \in \mathbb{R}. \tag{4.2}$$

Analog zu (4.2) definieren wir die *Fouriertransformierte* einer Folge $c \in \ell_1(\mathbb{Z})$ als

$$\widehat{c}(\xi) = c^\wedge(\xi) = \sum_{j \in \mathbb{Z}} c(j) e^{-ij\xi} \text{ für } \xi \in \mathbb{R}. \tag{4.3}$$

Bevor wir einige Eigenschaften der Fouriertransformierten einer Funktion angeben, führen wir für die beiden Gruppenoperationen auf $\mathbb{R}$ eine abkürzende Notation ein. Bei den Operationen handelt es sich um die *Translation* und die *Skalierung*, die wir mit Hilfe der Operatoren τ_y und σ_a realisieren

$$\tau_y f = f(\cdot + y) \text{ (Translation) und } \sigma_a f = \sigma(a\cdot) \text{ (Skalierung).} \tag{4.4}$$

Wir verwenden den Begriff der *Faltung* zweier Funktionen oder Folgen wie folgt.

4.3 Definition: Seien $f, g \in L_1(\mathbb{R})$. Dann definiert

$$f * g = \int_{\mathbb{R}} f(\cdot - t)g(t)dt = \int_{\mathbb{R}} \tau_{-t} f g(t)dt \tag{4.5}$$

die *Faltung* von f und g. Benutzen wir an Stelle der Funktion g eine Folge $c \in \ell_p(\mathbb{Z})$, so erhalten wir die Faltung als

$$f * c = \sum_{j\in\mathbb{Z}} \tau_{-j} f c(j). \tag{4.6}$$

Für $c, d \in \ell_p(\mathbb{Z})$ ist

$$c * d = \sum_{j\in\mathbb{Z}} c(\cdot - j)d(j) \tag{4.7}$$

die diskrete Faltung.

Der nachfolgende Satz fasst die für uns wichtigsten und wohlbekannten Eigenschaften einer Fouriertransformierten zusammen.

4.4 Satz: Sei $f \in L_1(\mathbb{R})$, dann gilt:

(1) Für ein festes $y \in \mathbb{R}$ gilt

$$(\tau_y f)^\wedge(\xi) = e^{iy\xi}\widehat{f}(\xi) \text{ für } \xi \in \mathbb{R}.$$

(2) Für ein festes $a \in \mathbb{R}$ ist

$$(\sigma_a f)^\wedge(\xi) = \frac{\widehat{f}(a^{-1}\xi)}{a} \text{ für } \xi \in \mathbb{R}.$$

(3) Für $g \in L_1(\mathbb{R})$ bzw. $c, d \in \ell_1(\mathbb{Z})$ ist $f * g \in L_1(\mathbb{R})$ und $c * d \in \ell_1(\mathbb{Z})$ und für $\xi \in \mathbb{R}$ gilt

$$(f * g)^\wedge(\xi) = \widehat{f}(\xi)\widehat{g}(\xi) \text{ sowie } (c * d)^\wedge(\xi) = \widehat{c}(\xi)\widehat{d}(\xi).$$

(4) Für $c \in \ell_1(\mathbb{R})$ gilt entsprechend $f * c \in L_1(\mathbb{R})$ und

$$(f * c)^\wedge(\xi) = \widehat{f}(\xi)\widehat{c}(\xi) \text{ für } \xi \in \mathbb{R}.$$

(5) Sind $f, f' \in L_1(\mathbb{R})$, dann erhalten wir

$$\left(\frac{d}{d\xi} f\right)^\wedge(\xi) = i\xi\widehat{f}(\xi) \text{ für } \xi \in \mathbb{R}.$$

(6) Sind $f, xf \in L_1(\mathbb{R})$, dann ist $\widehat{f}$ differenzierbar und es gilt

$$\frac{d}{d\xi}\widehat{f}(\xi) = (-ixf)^\wedge(\xi) \text{ für } \xi \in \mathbb{R}.$$

(7) Sind $f, \widehat{f} \in L_1(\mathbb{R})$, so setzen wir

$$f(x) = \left(\widehat{f}\right)^\vee(x) = \frac{1}{2\pi}\int_{\mathbb{R}} \widehat{f}(\vartheta)e^{ix\vartheta}d\vartheta. \tag{4.8}$$

Den obigen Ausdruck bezeichnen wir als *inverse Fouriertransformation.*

Für $f \in L_1(\mathbb{R})$ und $a, b, \widetilde{a} \in \mathbb{R}$ mit $a, \widetilde{a} \neq 0$ bezeichnen wir die nachfolgende Beziehungen als

$$\begin{array}{lcll} f(ax) & \Leftrightarrow & \frac{1}{|a|}\widehat{f}(\frac{\xi}{a}) & \textit{Zeitskalierung,} \\ f(x-b) & \Leftrightarrow & \widehat{f}(\xi)e^{-i\xi b} & \textit{Zeitverschiebung,} \\ \frac{1}{|\widetilde{a}|}f(\frac{x}{\widetilde{a}}) & \Leftrightarrow & \widehat{f}(\widetilde{a}\xi) & \textit{Frequenzskalierung,} \\ f(x)e^{i\vartheta x} & \Leftrightarrow & \widehat{f}(\xi-\vartheta) & \textit{Frequenzverschiebung.} \end{array} \tag{4.9}$$

Im weiteren Verlauf stellen wir noch zwei wichtige Resultate vor, welche uns in den nächsten Abschnitten eine große Hilfe sein werden.

4.5 Satz: (Parseval/Plancherel) Seien $f, g \in L_1(\mathbb{R}) \cap L_2(\mathbb{R})$, dann gilt

$$\int_{\mathbb{R}} f(t)g(t)dt = \frac{1}{2\pi}\int_{\mathbb{R}} \widehat{f}(\xi)\overline{\widehat{g}(\xi)}d\xi. \tag{4.10}$$

Damit ist für $f = g$

$$\|f\|_2 = \frac{1}{2\pi}\|\widehat{f}\|_2. \tag{4.11}$$

Bisher haben wir die Fouriertransformierte auf L_1-Funktionen anwenden können, die Identität (4.11) hilft uns, diese Definition auf $L_2(\mathbb{R})$ zu übertragen.

4.6 Satz: (Poisson-Summenformel) Für $f \in L_1(\mathbb{R}) \cap L_2(\mathbb{R})$ ist

$$\sum_{k\in\mathbb{Z}} f(2\pi k) = \frac{1}{2\pi}\sum_{k\in\mathbb{Z}} \widehat{f}(k) \text{ und } \sum_{k\in\mathbb{Z}} f(k) = \sum_{k\in\mathbb{Z}} \widehat{f}(2\pi k). \tag{4.12}$$

Als Nächstes wenden wir uns der gefensterten Fouriertransformation zu. Im Gegensatz zur Fouriertransformation, bei der eine Frequenzanalyse des gesamten Signals durchgeführt wird, erlaubt die gefensterte Fouriertransformation lokale Aussagen, das heißt Aussagen über die Änderungen von Frequenzanteilen in der Zeit.
In [Ch92], Seite 7, finden wir eine Definition der Fensterfunktion.

4.7 Definition: Eine Funktion $g \in L_2(\mathbb{R})$ heißt *Fensterfunktion*, wenn $xg(x) \in L_2(\mathbb{R})$ ist. Wir bezeichnen den Mittelwert x^* von $|g(x)|^2$ als *Zentrum* und die Standardabweichung Δf von $|g(x)|^2$ als *Radius*. Wir setzen also

$$\begin{aligned} x^* &= \frac{1}{\|g\|_2^2}\int_{\mathbb{R}} x\,|g(x)|^2\,dx \text{ und} \\ \Delta g &= \frac{1}{\|g\|_2}\sqrt{\int_{\mathbb{R}} (x-x^*)^2\,|g(x)|^2\,dx}. \end{aligned}$$

Wir nennen $2\Delta g$ *Breite der Fensterfunktion.*

Mit Hilfe der Fensterfunktion können wir die gefensterte Fouriertransformation wie folgt einführen.

4.8 Definition: Seien $f \in L_2(\mathbb{R})$ und $b \in \mathbb{R}$ sowie g eine Fensterfunktion. Wir bezeichnen

$$\widehat{f_b}(\xi) = \int_{\mathbb{R}} f(x)\overline{g(x-b)}e^{-i\xi x}dx \tag{4.13}$$

als *gefensterte Fouriertransformation*. Eine andere Darstellung sehen wir mit Hilfe des L_2-Skalarprodukts, denn es ist

$$\widehat{f_b}(\xi) = \langle f, h \rangle$$

mit $h = g(\cdot - b)e^{-i\xi\cdot}$.

Ein Beispiel für eine Fensterfunktion ist die Gaußfunktion

$$g_\sigma(x) = \frac{1}{2\sigma\sqrt{\pi}} e^{-(\frac{x}{2\sigma})^2}.$$

Wir erkennen, dass g_σ der Definition (4.7) genügt. Darüber hinaus ist die Fouriertransformierte von g_σ wieder eine Fensterfunktion, da $\widehat{g}_\sigma$ eine Gaußfunktion ist. Im Folgenden interessieren uns stets solche Funktionen, deren Fouriertransformierte wieder eine Fensterfunktion ist. Ebenso wie die Fouriertransformation ist auch die gefensterte Fouriertransformation invertierbar.

4.9 Satz: Sind $f, g \in L_2(\mathbb{R})$, $\|g\|_2 = 1$ und sind g und $\widehat{g}$ Fensterfunktionen, so gilt in jedem Punkt x, in dem f stetig ist

$$f(x) = \frac{1}{2\pi} \int_{\mathbb{R}} \int_{\mathbb{R}} \widehat{f_b}(\xi) g(x-b) e^{i\xi x} d\xi db. \tag{4.14}$$

Die Formel (4.14) stellt die *inverse gefensterte Fouriertransformation* dar. ([Ch92], Seite 52)

4.3 Kontinuierliche Wavelettransformation

Mit Hilfe der gefensterten Fouriertransformation können wir lokale Zeit-Frequenz-Analysen durchführen, aber die verwendete Fenstergröße bleibt immer konstant. Enthält ein Signal sowohl nieder- als auch hochfrequente Anteile, dann werden die niederfreuqenten Teile zu grob und die hochfrequenten zu fein betrachtet. Ein weiterer Nachteil der Fouriertransformation ist die feste, nicht an das Signal angepasste Basis. Beide Probleme können wir mit Hilfe der *Wavelettransformation* lösen. Sie arbeitet einerseits mit flexiblen Zeit-Frequenz-Fenstern und andererseits mit relativ frei wählbaren Basen.
Die Basisfunktionen ψ der Wavelettransformation müssen also neben einer Verschiebung auch einer Skalierung unterworfen werden.
Definieren wir nun ein Wavelet und die zu diesem Wavelet gehörige Wavelettransformation.

4.10 Definition: Eine Funktion $\psi \in L_2(\mathbb{R})$, welche die Bedingung

$$0 < C_\psi = 2\pi \int_{\mathbb{R}} \frac{\left|\widehat{\psi}(\xi)\right|^2}{|\xi|} d\xi < \infty \tag{4.15}$$

erfüllt, heißt *Wavelet*.

C_ψ wird *Calderon-Konstante* genannt.

4.11 Definition: Die *Wavelettransformation* von $f \in L_2(\mathbb{R})$ zum Wavelet ψ ist definiert durch

$$L_\psi f(a,b) = |a|^{-1/2} \int_{\mathbb{R}} f(t) \overline{\psi\left(\frac{t-b}{a}\right)} dt \tag{4.16}$$

mit $a, b \in \mathbb{R}$, $a \neq 0$.
Für die Wavelettransformation ist auch die folgende Schreibweise üblich

$$\psi_{a,b}(x) = |a|^{-1/2} \psi\left(\frac{x-b}{a}\right), \text{ dann ist } L_\psi f(a,b) = \langle f, \psi_{a,b} \rangle. \tag{4.17}$$

Ist $\psi \in L_1(\mathbb{R})$ ein Wavelet, so ist $\widehat{\psi}$ stetig in $\mathbb{R}$. Daher erhalten wir

$$\widehat{\psi}(0) = \frac{1}{\sqrt{2\pi}} \int_{\mathbb{R}} \psi(t) dt = 0$$

als notwendige Bedingung an ein Wavelet.
Als Nächstes wollen wir eine Methode vorstellen, mit der wir eine Vielzahl von Wavelets konstruieren können.

4.12 Lemma: Sei φ eine k–fach stetig differenzierbare Funktion ($k \in \mathbb{N}$) mit $\varphi^{(k)} \in L_2(\mathbb{R})$ und $\varphi^{(k)} \neq 0$. Dann ist

$$\psi(x) = \varphi^{(k)}(x) \tag{4.18}$$

ein Wavelet.

Beweis: Wir wenden die Rechenregeln der Fouriertransformation an und erhalten

$$\left|\widehat{\psi}(\xi)\right| = |\xi|^k \left|\widehat{\varphi}(\xi)\right|.$$

Die Berechnung der Calderon-Konstante C_ψ ergibt

$$\begin{aligned} C_\psi &= 2\pi \int_{\mathbb{R}} \frac{\left|\widehat{\psi}(\xi)\right|^2}{|\xi|} d\xi = 2\pi \int_{\mathbb{R}} \frac{|\xi|^{2k} \left|\widehat{\varphi}(\xi)\right|^2}{|\xi|} d\xi \\ &= 2\pi \int_{-1}^{1} |\xi|^{2k-1} \left|\widehat{\varphi}(\xi)\right|^2 d\xi + 2\pi \int_{|\xi|>1} \frac{|\xi|^{2k} \left|\widehat{\varphi}(\xi)\right|^2}{|\xi|} d\xi \\ &\leq 2\pi \left(\|\varphi\|^2 + \|\varphi^{(k)}\|^2 \right) < \infty. \end{aligned}$$

Nach Definition 4.10 ist ψ ein Wavelet. □

Ähnlich wie bei der Fouriertransformation kann man für die Wavelettransformation Skalierungsgleichungen beweisen. Wegen der Abhängigkeit von Zeit und Frequenz sind aber nicht alle Kombinationen realisierbar.

4.13 Satz: Sei $f \in L_2(\mathbb{R})$ und L_ψ die Wavelettransformierte von f. Dann gilt für $a, b, \widetilde{a} \in \mathbb{R}$ mit $\widetilde{a} \neq 0$

$$\begin{array}{lcll} \sqrt{|\widetilde{a}|} f(\widetilde{a}x) & \Leftrightarrow & L_\psi f(\widetilde{a}a, \widetilde{a}b) & \textit{Frequenzskalierung,} \\ f(\widetilde{a}x) & \Leftrightarrow & \frac{1}{\sqrt{|\widetilde{a}|}} L_\psi f(\widetilde{a}a, \widetilde{a}b) & \textit{Zeitskalierung,} \\ f(x-y) & \Leftrightarrow & L_\psi f(a, b-y) & \textit{Zeitverschiebung.} \end{array} \tag{4.19}$$

Beweis: Wir beweisen hier die Formeln für die Zeitskalierung und -verschiebung. Der Ausdruck für die Frequenzskalierung kann auf ähnliche Weise bewiesen werden.
Zeitskalierung: Wir setzen $g(x) = f(cx)$ und erhalten für die Wavelettransformierte

$$L_\psi g(a,b) = |a|^{-\frac{1}{2}} \int_{\mathbb{R}} f(cx) \overline{\psi\left(\frac{x-b}{a}\right)} dx.$$

Mit der Substitution $cx = y$ und $dx = \frac{1}{|c|} dy$ folgt

$$L_\psi g(a,b) = |ac|^{-\frac{1}{2}} \int_{\mathbb{R}} f(y) \overline{\psi\left(\frac{y-cb}{ca}\right)} dy = \frac{1}{\sqrt{|c|}} L_\psi f(ca, cb),$$

was zu zeigen war.
Zeitverschiebung: Wir setzen $g(x) = f(x - x')$ und berechnen die Wavelettransformierte

$$L_\psi g(a,b) = |a|^{-\frac{1}{2}} \int_{\mathbb{R}} f(x-x') \overline{\psi\left(\frac{x-b}{a}\right)} dx.$$

Mit der Substitution $x - x' = y$ erhalten wir

$$L_\psi g(a,b) = |a|^{-\frac{1}{2}} \int_{\mathbb{R}} f(y)\overline{\psi\left(\frac{y + x' - b}{a}\right)}dy = L_\psi f(a, b - x'),$$

was zu zeigen war. □

In einigen Fällen können wir die Funktion $f(x)$ aus den Werten ihrer Transformierten $L_\psi f(a,b)$ rekonstruieren. Dafür benötigen wir das sogenannte *duale Wavelet* ψ^*. Eine solche inverse Transformation geben wir hier allerdings nur für zwei Parameter a und b an. Die diskrete inverse Transformation behandeln wir später.

4.14 Satz: (Inverse Wavelettransformation für $a, b \in \mathbb{R}$) Sei ψ ein Wavelet, das die Wavelettransformation $L_\psi f(a,b)$ mit $a, b \in \mathbb{R}$ definiert. Dann ist

$$\int_{\mathbb{R}} \int_{\mathbb{R}} L_\psi f(a,b)\overline{L_\psi g(a,b)}\frac{da}{a^2}db = C_\psi \langle f, g\rangle \tag{4.20}$$

für alle $f, g \in L_2(\mathbb{R})$. Für $f \in L_2(\mathbb{R})$ und alle $x \in \mathbb{R}$, in denen f stetig ist, gilt

$$f(x) = \frac{1}{C_\psi} \int_{\mathbb{R}} \int_{\mathbb{R}} L_\psi f(a,b)\psi_{a,b}(x)\frac{da}{a^2}db. \tag{4.21}$$

Die Gleichung (4.21) heißt *inverse Wavelettransformation*, [Da99], Seite 24.

Bevor wir diesen Satz beweisen, bemerken wir, dass das duale Wavelet gleich dem komplex konjugierten Wavelet ist. Es gilt also

$$\psi^*_{a,b} = \overline{\psi}_{a,b}.$$

Beweis: Wir zeigen als Erstes die Gleichung (4.20).
Nach den Formeln für Frequenzskalierung und -verschiebung aus (4.9) gilt für die Fouriertransformierte des Wavelets

$$\widehat{\psi}_{a,b}(\xi) = \sqrt{a}e^{ib\xi}\widehat{\psi}(\xi a).$$

Die inverse Fouriertransformation (4.8) führt zu

$$\psi_{a,b}(x) = \frac{\sqrt{a}}{2\pi} \int_{\mathbb{R}} \widehat{\psi}(\xi a)e^{i\xi x}e^{-ib\xi}d\xi.$$

Wir setzen dies in (4.16) ein und erhalten

$$L_\psi f(a,b) = \frac{\sqrt{a}}{2\pi} \int_{\mathbb{R}} \overline{\widehat{\psi}(\xi a)} e^{i\xi b} \int_{\mathbb{R}} f(x) e^{-i\xi x} dx d\xi.$$

Wir fassen das letzte Integral als Fouriertransformierte von f auf, und bekommen

$$L_\psi f(a,b) = \frac{\sqrt{a}}{2\pi} \int_{\mathbb{R}} \widehat{f}(\xi) \overline{\widehat{\psi}(\xi a)} e^{i\xi b} d\xi.$$

Mit der Notation

$$\widetilde{f}(\xi) = \sqrt{a} \widehat{f}(\xi) \overline{\widehat{\psi}(\xi a)}$$

sowie

$$\widetilde{g}(\xi) = \sqrt{a} \widehat{g}(\xi) \overline{\widehat{\psi}(\xi a)}$$

ergibt sich für die Wavelettransformation

$$L_\psi f(a,b) = \frac{1}{2\pi} \int_{\mathbb{R}} \widetilde{f}(\xi) e^{i\xi b} d\xi.$$

Setzen wir nun diese Gleichung in die entsprechenden Ausdrücke für $L_\psi f$ und $L_\psi g$ ein, so erhalten wir

$$\int_{\mathbb{R}} \int_{\mathbb{R}} L_\psi f(a,b) \overline{L_\psi g(a,b)} \frac{da}{a^2} db$$

$$= \int_{\mathbb{R}} \frac{1}{4\pi^2} \int_{\mathbb{R}} \underbrace{\overline{\left[\int_{\mathbb{R}} \overline{\widetilde{f}(\xi)} e^{-i\xi b} d\xi \right]}}_{=\widehat{\widetilde{f}}} \underbrace{\left[\int_{\mathbb{R}} \overline{\widetilde{g}(\xi)} e^{-i\xi b} d\xi \right]}_{=\widehat{\widetilde{g}}} db \frac{da}{a^2}.$$

Damit ergibt sich

$$\int_{\mathbb{R}} \int_{\mathbb{R}} L_\psi f(a,b) \overline{L_\psi g(a,b)} \frac{da}{a^2} db = \frac{1}{2\pi} \int_{\mathbb{R}} \frac{1}{2\pi} \int_{\mathbb{R}} \widehat{\widetilde{f}}(b) \overline{\widehat{\widetilde{g}}(b)} db \frac{da}{a^2}.$$

Dieser Ausdruck lässt sich mit Hilfe der Parsevalschen Gleichung (4.10) vereinfachen zu

$$\int_{\mathbb{R}} \int_{\mathbb{R}} L_\psi f(a,b) \overline{L_\psi g(a,b)} \frac{da}{a^2} db = \frac{1}{2\pi} \int_{\mathbb{R}} \int_{\mathbb{R}} \widetilde{f}(\xi) \widetilde{g}(\xi) d\xi \frac{da}{a^2}.$$

Wir setzen für $\widetilde{f}$ und $\widetilde{g}$ wieder die entsprechenden Terme ein, dann gilt

$$\int_{\mathbb{R}} \int_{\mathbb{R}} L_\psi f(a,b) \overline{L_\psi g(a,b)} \frac{da}{a^2} db = \frac{1}{2\pi} \int_{\mathbb{R}} \widehat{f}(\xi) \overline{\widehat{g}(\xi)} \int_{\mathbb{R}} \frac{\left| \widehat{\psi}(\xi a) \right|^2}{|a|} da.$$

Das letzte Integral geht durch eine einfache Transformation in die Calderon-Konstante C_ψ über. Durch die Anwendung der Parsevalschen Gleichung (4.10) auf das erste Integral der rechten Seite erhalten wir

$$\int_{\mathbb{R}} \int_{\mathbb{R}} L_\psi f(a,b) \overline{L_\psi g(a,b)} \frac{da}{a^2} db = C_\psi \langle f, g \rangle,$$

womit die erste Aussage bewiesen ist.
Setzen wir in diese Gleichung für g die δ–Distribution $\delta(x - x')$ ein, so erhalten wir unter Berücksichtigung der Beziehung

$$\delta(x - x') = \frac{1}{2\pi} \int_{\mathbb{R}} e^{i(x-x')\xi} d\xi$$

die gewünschte Darstellung (4.21). Dabei wird die δ–Distribution durch

$$f(x') = \int_{\mathbb{R}} f(x) \delta(x - x') dx \text{ für } f \in L_1(\mathbb{R})$$

charakterisiert. □

In der Signalverarbeitung betrachten wir häufig Funktionen mit positiven Frequenzen ξ. Dies bedeutet, dass wir die Wavelettransformation für $a > 0$ durchführen. Um das Signal rücktransformieren zu können, ist eine stärkere Bedingung als (4.15) an das Wavelet notwendig.

4.15 Satz: (Inverse Wavelettransformation für $a \in \mathbb{R}^+$, $b \in \mathbb{R}$)
Sei ψ ein Wavelet, welches der Bedingung

$$\int_0^\infty \frac{\left|\widehat{\psi}(\xi)\right|^2}{\xi} d\xi = \int_0^\infty \frac{\left|\widehat{\psi}(-\xi)\right|^2}{\xi} d\xi = \frac{1}{2} C_\psi < \infty \tag{4.22}$$

genügt. Seien $a \in \mathbb{R}^+$, $b \in \mathbb{R}$. Dann gilt

$$\int_0^\infty \left[\int_{\mathbb{R}} L_\psi f(a,b) \overline{L_\psi g(a,b)} db \right] \frac{da}{a^2} = \frac{1}{2} C_\psi \langle f, g \rangle$$

für alle $f, g \in L_2(\mathbb{R})$. Darüber hinaus gilt für $f \in L_2(\mathbb{R})$ und alle $x \in \mathbb{R}$, in denen f stetig ist,

$$f(x) = \frac{2}{C_\psi} \int_0^\infty \left[\int_{\mathbb{R}} L_\psi f(a,b) \psi_{a,b}(x) db \right] \frac{da}{a^2}. \tag{4.23}$$

([Ch92], Seite 64)

Ähnlich wie in Satz 4.14 entspricht das duale dem komplex konjugierten Wavelet.

4.4 Stetige Waveletbasen

Bevor wir uns mit der für uns wichtigen diskreten Wavelettransformation beschäftigen, gehen wir auf die praxisrelevanten Aspekte der kontinuierlichen Wavelettransformation ein. So sind die Bedingungen, die an ein Wavelet gestellt werden, sehr gering. Dies lässt uns eine relativ freie Wahl für die Waveletfunktion. Wie die beiden folgenden Sätze zeigen, genügen uns bereits zwei Bedingungen, um die Waveletbedingung (4.15) zu erfüllen.

4.16 Satz: Sei $\psi \in L_1(\mathbb{R}) \cap L_2(\mathbb{R})$, $\psi \neq 0$ mit

$$\int_{\mathbb{R}} \psi(x)dx = 0 \text{ und } \int_{\mathbb{R}} |x|^{\beta} |\psi(x)| \, dx < \infty$$

für $\beta > \frac{1}{2}$. Dann ist ψ ein Wavelet.

4.17 Satz: Sei $\psi \in L_2(\mathbb{R})$, $\psi \neq 0$, eine Funktion mit kompaktem Träger. ψ ist genau dann ein Wavelet, wenn der Mittelwert $\int_{\mathbb{R}} \psi(x)dx = 0$ ist.

Beide Sätze samt Beweisen finden wir in [Lo98], Seiten 21-22.
Zur Konstruktion von Wavelets benutzen wir Lemma 4.12 und sind nun in der Lage, eine Vielzahl von Wavelets anzugeben.
Wir wollen eine kleine Auswahl stetiger Waveletbasen vorstellen.

(1) *Haarwavelet*
Die Funktion

$$\psi(x) = \begin{cases} 1 & \text{für } 0 \leq x < \frac{1}{2} \\ -1 & \text{für } \frac{1}{2} \leq x < 1 \\ 0 & \text{sonst} \end{cases}$$

heißt *Haarwavelet* und ist in Abbildung 3 dargestellt.
Für die relevanten Konstanten erhalten wir folgende Werte

$$\begin{aligned} \text{Calderon-Konstante:} \quad & C_\psi = 2 \ln 2, \\ \text{Zentrum:} \quad & x^* = \tfrac{1}{2}, \\ \text{Radius:} \quad & \Delta\psi = \tfrac{1}{2\sqrt{3}}. \end{aligned}$$

Die Fouriertransformierte des Haarwavelets ist

$$\left|\widehat{\psi}(\xi)\right| = \frac{\sqrt{2}}{\xi}\sqrt{3 - 4\cos\frac{\xi}{2} + \cos\xi}.$$

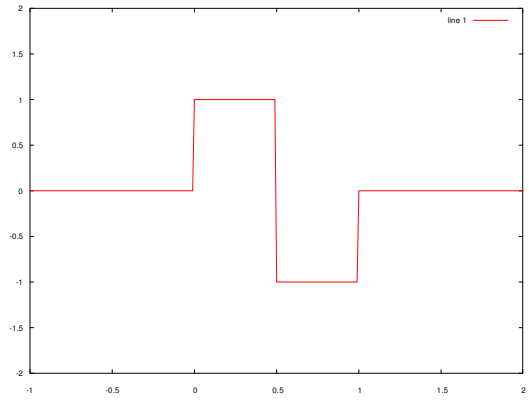

Abbildung 3: *Haarwavelet.*

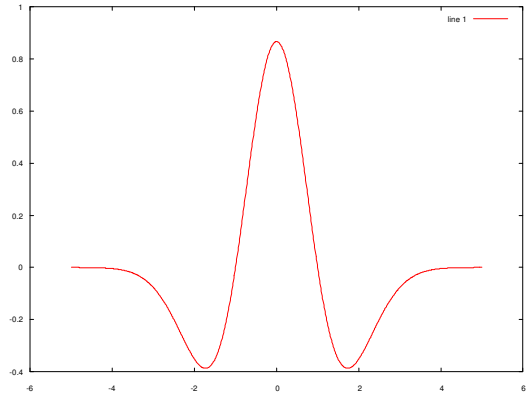

Abbildung 4: *Mexican Hat.*

(2) *Mexican Hat* oder *Sombrero*
Die Funktion

$$\psi(x) = \frac{2}{\sqrt{3}}\pi^{-\frac{1}{4}}(1-x^2)e^{-\frac{x^2}{2}}$$

bezeichnen wir als *Mexican Hat.*
Ihren Verlauf können wir Abbildung 4 entnehmen. Die Konstanten nehmen die folgenden Werte an

$$\begin{aligned} \text{Calderon-Konstante:} \quad & C_\psi = \tfrac{8}{3}\sqrt{\pi}, \\ \text{Zentrum:} \quad & x^* = 0 \text{ und } \psi(x^*) = \tfrac{2}{\sqrt{3}}\pi^{-\frac{1}{4}}, \\ \text{Radius:} \quad & \Delta\psi = \sqrt{\tfrac{7}{6}}. \end{aligned}$$

Die Fouriertransformierte des Mexican Hat ist

$$\widehat{\psi}(\xi) = \frac{2\sqrt{2}}{\sqrt{3}}\pi^{\frac{1}{4}}\xi^2 e^{-\frac{\xi^2}{2}}.$$

(3) *Morletwavelet*
Das *Morletwavelet* ist definiert durch

$$\psi(x) = \pi^{-\frac{1}{4}} \left(e^{-itx} - e^{-\frac{t^2}{2}} \right) e^{-\frac{x^2}{2}}$$

mit

$$t = \pi \left(\frac{2}{\ln 2} \right)^{\frac{1}{2}}.$$

Eine Darstellung und die Werte der Konstanten finden wir in [Da99], Seite 76.

4.5 Diskrete Wavelettransfomation

In der Praxis werden wir nie in der Lage sein, ein Signal kontinuierlich abzutasten, ein Signal wird stets diskret vorliegen. Daher benötigen wir auch eine diskretisierte Wavelettransformation, die der Gegenstand dieses Abschnitts ist.
Bei der diskreten Fouriertransformation müssen Zeit und Ort diskretisiert werden. Bei Wavelets kommen noch die Parameter a und b hinzu. In der Praxis haben sich logarithmische Teilungen bewährt, den Parameter a diskretisieren wir durch $a_j = a_0^j$ für $j \in \mathbb{Z}$. Wie wir sehen werden, ergibt diese Teilung einen numerisch sehr effizienten Algorithmus.
Als Nächstes müssen wir den Parameter b, der die Lage des Waveletfensters im Zeitbereich bestimmt, diskretisieren. Da bei der Wavelettransformation die Zeit- und Frequenzanalyse eng zusammenhängen, ist es naheliegend, die Parameter a und b aneinander zu koppeln. So möchten wir zum Beispiel, dass bei hohen Frequenzen und damit entsprechend kleinen Werten von a der Parameter b feiner diskretisiert wird, was in diesem Fall schmaleren Waveletfenstern entspricht. Eine quantitative Aussage darüber liefert uns das Shannonsche Abtasttheorem.

4.18 Satz: (Shannonsches Abtasttheorem) Sei f ein Signal mit der Bandbreite Ω. Für die interessierenden Kreisfrequenzen $\omega = \frac{2\pi}{T_P}$, wobei $T_P > 0$ die Periode oder Wellenlänge darstellt, gilt $|\omega| \leq \Omega$. Um das Signal so abzutasten, dass das Ursprungssignal ohne Informationsverlust (aber mit unendlich großem Aufwand) rekonstruiert bzw. (mit endlichem Aufwand) beliebig genau approximiert werden kann, muss für die Schrittweite Δx gelten

$$\Delta x < q = \frac{\pi}{\Omega}. \tag{4.24}$$

Dabei ist q die *Nyquist-Rate*.

Als Beispiel betrachten wir ein Sinussignal

$$f(x) = \sin(2x)$$

mit der Periode $T_P = \pi$ und der Kreisfrequenz $\omega = 2$. Gemäß (4.24) können wir die Nyquist-Rate als $q = \frac{\pi}{2}$ bestimmen. Das heißt, um das Signal adäquat abzutasten, muss die Schrittweite kleiner oder gleich $\frac{\pi}{2}$ sein. Da der Skalierungsparameter a der Wavelettransformation proportional zur betrachteten Wellenlänge T_P ist, besagt das Abtasttheorem 4.18, dass der Ortsparameter b proportional zu a, oder feiner, zu diskretisieren ist. Das heißt, es muss

$$\Delta b \leq q \text{ mit } q \sim a$$

gelten.
Wir erhalten für die oben angesprochene logarithmische Diskretisierung von a unter Ausnutzung des Abtasttheorems

$$a_j = a_0^j \text{ sowie } b_k = kb_0a_0^j \text{ für } j,k \in \mathbb{Z} \text{ und } a_0 > 0,\ b_0 > 0$$

und damit

$$L_\psi f(j,k) = L_\psi f(a_j, b_k) = \frac{1}{\sqrt{a_0^j}} \sum_{l \in \mathbb{Z}} f(l) \overline{\psi\left(\frac{l}{a_0^j} - kb_0\right)}. \tag{4.25}$$

Alternativ können wir (4.25) auch als

$$L_\psi f(j,k) = \sum_{l \in \mathbb{Z}} f(l) \overline{\psi_{j,k}(l)} \text{ mit} \tag{4.26}$$

$$\psi_{j,k}(l) = \psi_{a_j,b_k}(l) = a_0^{-\frac{j}{2}} \psi\left(a_0^{-j} l - kb_0\right) \text{ für } j,k \in \mathbb{Z} \tag{4.27}$$

schreiben. Wir bezeichnen (4.25) als *diskrete Wavelettransformation*.
Wird das Signal periodisch durch $f(l) = f(l + \widetilde{m}N)$ für $\widetilde{m} \in \mathbb{Z}$ fortgesetzt, so ergeben sich für j,k und l die folgenden Bereiche

$$j \in \{1, \ldots, j_{max}\},\ k \in \left\{0, \ldots, \frac{N}{b_0 a_0^j} - 1\right\},\ l \in \left\{kb_0a_0^j, \ldots, (kb_0 + K - 1)a_0^j\right\}$$

für $j,k,l \in \mathbb{Z}$, sofern das Wavelet einen kompakten Träger im Bereich $[0, K-1]$ hat und $N \in \mathbb{N}$ die Länge des Signals ist. Ansonsten müssen wir

das Wavelet geeignet abschneiden.
Berechnen wir die Wavelettransformation $L_\psi f(a,b)$ für beschränkte, diskrete Werte von a und b, müssen wir uns die Frage stellen, ob trotzdem das Ausgangssignal noch rekonstruierbar ist. Denn das diskrete Wavelet $\psi_{j,k}$ (4.27) bildet nicht notwendig eine Basis des $L_2(\mathbb{R})$. Wir werden sehen, dass eine Rekonstruktion trotzdem möglich ist, wenn wir stärkere Forderungen als Bedingung (4.10) an das Wavelet stellen. Für die genaue Ausführung benötigen wir einige Vorüberlegungen und Definitionen.
Aus praktischen Gründen machen wir die folgenden Überlegungen für das stetige Signal $f \in L_2(\mathbb{R})$, die Summen über $n \in \mathbb{Z}$ gehen dann wieder in Integrale über. Zur Abschwächung des Basisbegriffs definieren wir zunächst einen Frame.

4.19 Definition: Sei $f \in L_2(\mathbb{R})$. Ein Wavelet $\psi \in L_2(\mathbb{R})$ generiert einen *Wavelet-Frame* $\{\psi_{j,k}\}$ des $L_2(\mathbb{R})$, wie in (4.27) eingeführt, genau dann, wenn Konstanten $A, B \in \mathbb{R}$ mit $0 < A \leq B < \infty$ existieren, die

$$A\|f\|_2^2 \leq \sum_{j,k\in\mathbb{Z}} |L_\psi f(j,k)|^2 \leq B\|f\|_2^2 \tag{4.28}$$

erfüllen. Die Gleichung (4.28) wird häufig *Stabilitätsbedingung* genannt.

Ein Wavelet-Frame ist nicht notwendig eine linear unabhängige Menge. Wählt man zum Beispiel das Haar-Wavelet mit $b_0 = \frac{1}{3}$, so ist die Menge $\{\psi_{j,k}\}$ ein linear abhängiger Wavelet-Frame, wie man in [Ch92] nachlesen kann.
Ein Spezialfall der Wavelettransformation ergibt sich im Falle von orthogonalen Wavelets.

4.20 Definition: Sei $\psi \in L_2(\mathbb{R})$ ein Wavelet, das einen Frame des $L_2(\mathbb{R})$ definiert. ψ heißt *orthogonal*, wenn für die Menge $\{\psi_{j,k}\}$ die Bedingung

$$\langle \psi_{j,k}, \psi_{m,l} \rangle = \delta_{j,m}\delta_{k,l} \text{ für } j,k,m,l \in \mathbb{Z} \tag{4.29}$$

erfüllt ist.

Wir sehen durch Einsetzen in Gleichung (4.28), dass für einen orthogonalen Frame, also einen Frame, der durch ein orthogonales Wavelet generiert wurde, für die Framegrenzen $A = B = 1$ gilt.
Der nächste Satz, aus [Da99], Seite 63, gibt uns Auskunft darüber, in welcher Beziehung Wavelet-Frames zu stetigen Wavelets stehen.

4.21 Satz: Sei

$$\psi_{j,k}(l) = \psi_{a_j,b_k}(l) = a_0^{-\frac{j}{2}} \psi\left(a_0^{-j} l - k b_0\right) \text{ für } j,k \in \mathbb{Z}.$$

Generiert $\psi_{j,k}$ einen Wavelet-Frame des $L_2(\mathbb{R})$ mit den Grenzen $A, B > 0$, so gelten

$$b_0 \ln a_0 A \leq \int_0^\infty \frac{\left|\widehat{\psi}(\xi)\right|^2}{|\xi|} d\xi \leq b_0 \ln a_0 B$$

und

$$b_0 \ln a_0 A \leq \int_{-\infty}^0 \frac{\left|\widehat{\psi}(\xi)\right|^2}{|\xi|} d\xi \leq b_0 \ln a_0 B.$$

Es folgen also aus den Framebedingungen die Zulässigkeitsbedingungen (4.22) und jeder Frame, der wie oben definiert ist, ist gleichzeitig eine Waveletbasis. Die Umkehrung dieser Aussage gilt allerdings nicht, wie wir in [Da99], Seite 67, nachlesen können.
Der nächste Schritt in unseren Betrachtungen auf dem Weg zur schnellen Wavelettransformation wäre die Existenz und Berechnung der inversen diskreten Wavelettransformation. Da diese Aussagen für unsere weiteren Ausführungen nicht benötigt werden, verzichten wir an dieser Stelle darauf und verweisen auf [Ch92], Abschnitt 3.

4.6 Multiresolution Analysis

Eine schnelle und stabile Waveletzerlegung und damit auch diskrete Wavelets führt man am besten über den Begriff der Multiresolution Analysis ein. Der Begriff der Multiresolution Analysis geht auf *Mallat* zurück und der von ihm entwickelte Transformationsalgorithmus ist, was die Geschwindigkeit angeht, sogar der schnellen Fouriertransformation (FFT) überlegen.
Die Idee der Multiresolution Analysis besteht darin, ein Signal $f \in L_2(\mathbb{R})$ schrittweise in Teilsignale mit verschiedenen Wellenlängen zu zerlegen. Dazu wird der Raum $L_2(\mathbb{R})$ in orthogonale Unterräume W_j zerlegt, die von Basen $\{\psi_{j,k}\}_k$ aufgespannt werden. Daher ist auch eine Forderung für die Konstruktion einer Multiresolution Analysis, dass die $\{\psi_{j,k}\}_k$ eine Basis des $L_2(\mathbb{R})$ bilden. Wir erhalten eine eindeutige Zerlegung des Signals $f(x) \in L_2(\mathbb{R})$ als

$$f(x) = \sum_{j \in \mathbb{Z}} g_j(x) \text{ mit } g_j(x) \in W_j.$$

Wir werden im Folgenden die Multiresolution Analysis einführen, die wichtigsten Eigenschaften angeben und die schnelle Wavelettransformation vorstellen.

4.22 Definition: Eine Funktion $\varphi \in V \subseteq L_2(\mathbb{R})$ erzeugt für ein $j \in \mathbb{Z}$ eine *Riesz-Basis*

$$\{\varphi_{j,k} = 2^{-\frac{j}{2}}\varphi(2^{-j}x - k)\}_{k\in\mathbb{Z}}$$

von V, wenn $\operatorname{span}\{\varphi_{j,k}\}_k$ dicht in V liegt und Konstanten $0 < A \leq B < \infty$ existieren, so dass

$$A\|c\|_{\ell_2(\mathbb{Z})} \leq \left\| \sum_{k\in\mathbb{Z}} c_k \varphi_{j,k} \right\|_2^2 \leq B\|c\|_{\ell_2(\mathbb{Z})} \tag{4.30}$$

für alle $c \in \ell_2(\mathbb{Z})$ gilt.

Die Bedingung (4.30) bezeichnet man auch als *Stabilität* von φ. Besonders einfache stabile Funktionen sind diejenigen, welche orthonormale Translate haben, denn dann gilt sogar $A = B = 1$.
Damit können wir die Multiresolution Analysis als eine Zerlegung des $L_2(\mathbb{R})$ definieren.

4.23 Definition: Eine Folge $V_j \subset L_2(\mathbb{R})$, $j \in \mathbb{Z}$, von abgeschlossenen Unterräumen heißt genau dann *Multiresolution Analysis (MRA)*, wenn folgende Bedingungen erfüllt sind.

(1) Es gilt $V_{j+1} \subset V_j$ für alle $j \in \mathbb{Z}$.

(2) $\overline{\bigcup_{j\in\mathbb{Z}} V_j} = L_2(\mathbb{R})$ und $\bigcap_{j\in\mathbb{Z}} V_j = \{0\}$.

(3) Die Räume sind *translationsinvariant*, das heißt es gilt

$$f \in V_j \Leftrightarrow f(\cdot + k) \in V_j \text{ für } j, k \in \mathbb{Z}.$$

(4) Es gilt

$$f \in V_j \Leftrightarrow f(2\cdot) \in V_{j-1} \text{ für } j \in \mathbb{Z}.$$

(5) Der Unterraum V_0 wird von einer *Skalierungsfunktion* $\varphi \in L_2(\mathbb{R})$ erzeugt, wobei die Translate von φ

$$\{\varphi_{0,k} = \varphi(\cdot - k)\}_{k\in\mathbb{Z}}$$

eine *Riesz-Basis* von V_0 bilden.

Eine grundlegende Eigenschaft der Skalierungsfunktion, die eine große Bedeutung für die Theorie und vor allem für die Praxis der MRA hat, ist die so genannte *Verfeinerungs-* oder *Skalierungsgleichung.*

4.24 Lemma: Für jede Skalierungsfunktion φ existiert eine Folge $\{h_k\}_{k\in\mathbb{Z}}$ mit $h_k \in \mathbb{R}$, welche die *Skalierungs-* oder *Verfeinerungsgleichung*

$$\varphi = \sqrt{2}\sum_{k\in\mathbb{Z}} h_k \varphi(2\cdot - k) \tag{4.31}$$

erfüllt. Wir nennen die h_k *Skalierungs-* oder *Verfeinerungskoeffizienten.*

Beweis: Es ist $\varphi \in V_0$ und $V_0 \subset V_{-1}$. Nach Definition 4.23 (4) ist dann $\varphi(2\cdot) \in V_{-1}$ und somit können wir V_{-1} als $V_{-1} = \text{span}\{\varphi(2\cdot - k) \mid k \in \mathbb{Z}\}$ darstellen. □

Wir nennen jede Funktion φ, die eine Verfeinerungsgleichung der Form (4.31) erfüllt, *verfeinerbar.*

4.25 Beispiel: Es gibt einige einfache Beispiele für verfeinerbare Funktionen.

(1) Die *charakteristische Funktion* $\chi_{[0,1]}$ hat die Verfeinerungsgleichung

$$\chi = \sqrt{2}\left(\frac{1}{\sqrt{2}}\chi(2\cdot) + \frac{1}{\sqrt{2}}\chi(2\cdot - 1)\right),$$

also Koeffizienten

$$h_k = \begin{cases} \frac{1}{\sqrt{2}} & \text{für } k = 0, 1 \\ 0 & \text{sonst.} \end{cases}$$

(2) Die *Hutfunktion*

$$\varphi(x) = \begin{cases} x + 1 & \text{für } x \in [-1, 0] \\ 1 - x & \text{für } x \in [0, 1] \\ 0 & \text{für } x \in \mathbb{R} - [-1, 1] \end{cases}$$

erfüllt die Verfeinerungsgleichung

$$\varphi = \sqrt{2}\left(\frac{\sqrt{2}}{4}\varphi(2\cdot + 1) + \frac{1}{\sqrt{2}}\varphi(2\cdot) + \frac{\sqrt{2}}{4}\varphi(2\cdot - 1)\right).$$

Wir erhalten die Koeffizienten

$$h_k = \begin{cases} \frac{\sqrt{2}}{4} & \text{für } k = -1, 1 \\ \frac{1}{\sqrt{2}} & \text{für } k = 0 \\ 0 & \text{sonst.} \end{cases}$$

Für die spätere Konstruktion orthogonaler Wavelets sind die Zusammenhänge zwischen der Skalierungsfunktion und ihren Verfeinerungskoeffizienten wichtig. Für die schnellen Algorithmen sind allein diese Koeffizienten und nicht mehr die Skalierungsfunktion von Interesse.

Wir haben bereits erwähnt, dass die Rekonstruktion eines Signals aus einer Wavelettransformierten dann einfach wird, wenn orthogonale Wavelets vorliegen. Daher wollen wir im Folgenden den Fall orthogonaler Skalierungsfunktionen näher untersuchen, denn wir werden feststellen, dass eben in diesem Fall eine orthogonale Waveletbasis erzeugt wird. In [Lo98], Seiten 116-117, finden wir die folgenden Aussagen samt Beweisen.

4.26 Satz: Sei $\varphi \in L_2(\mathbb{R})$. Genau dann ist $\{\varphi_{j,k}\}_{k\in\mathbb{Z}}$ ein orthonormales System, wenn

$$\sum_{l\in\mathbb{Z}} |\widehat{\varphi}(\xi + 2\pi l)|^2 = \frac{1}{2\pi}$$

fast überall gilt.

Bildet die Skalierungsfunktion φ einer MRA keine orthogonale Basis des $L_2(\mathbb{R})$, so haben wir die Möglichkeit, φ mit Hilfe der folgenden Aussage zu orthogonalisieren.

4.27 Satz: Sei $\varphi \in L_2(\mathbb{R})$. Existieren Konstanten $A, B > 0$ mit

$$A \leq \sum_{l\in\mathbb{Z}} |\widehat{\varphi}(\xi + 2\pi l)|^2 \leq B \tag{4.32}$$

fast überall, so ist $\{\phi_{0,k}\}_{k\in\mathbb{Z}}$ mit

$$\widehat{\phi}(\xi) = \frac{1}{\sqrt{2\pi}} \frac{\widehat{\varphi}(\xi)}{\sqrt{\sum_{l\in\mathbb{Z}} |\widehat{\varphi}(\xi + 2\pi l)|^2}} \tag{4.33}$$

eine Orthonormalbasis von $V_0 = \text{span}\{\varphi_{0,k} \mid k \in \mathbb{Z}\}$.

4.28 Bemerkung: Wir können daher mit Hilfe des Satzes 4.27 für eine stabile Funktion φ stets eine Orthonormalbasis konstruieren. Leider hat die Funktion ϕ keinen kompakten Träger, so dass diese Vorgehensweise für praktische Zwecke unbrauchbar wird.

Der folgende Satz, ebenfalls aus [Lo98], Seiten 120-121, beschäftigt sich mit den Eigenschaften der Verfeinerungskoeffizienten im orthogonalen Fall.

4.29 Satz: Die Verfeinerungskoeffizienten einer orthogonalen Skalierungsfunktion genügen den Beziehungen

$$\sum_{k\in\mathbb{Z}} h_k h_{k+2j} = \delta_{0,j} \tag{4.34}$$

und

$$\sum_{k\in\mathbb{Z}} h_k = \sqrt{2} \text{ sowie } \sum_{k\in\mathbb{Z}} (-1)^k h_k = 0. \tag{4.35}$$

Die nächsten beiden Sätze beschäftigen sich mit der Konstruktion einer Waveletbasis, also damit, wann eine Skalierungsfunktion eine MRA erzeugt und wie wir diese konstruieren können.

4.30 Satz: Genügt eine Funktion $\varphi \in L_2(\mathbb{R})$ den Bedingungen (4.32) sowie

(1)

$$\int_{\mathbb{R}} \varphi(x)dx = \sqrt{2\pi}\widehat{\varphi}(0) > 0$$

und

(2)

$$\widehat{\varphi}(2\pi k) = 0 \text{ für alle } k \in \mathbb{Z}\backslash\{0\},$$

so bilden die Räume V_j eine MRA des $L_2(\mathbb{R})$, [Lo98], Seite 119.

Da wir jetzt wissen, wann eine Skalierungsfunktion eine MRA erzeugt, können wir ausgehend von einer MRA eine Waveletbasis konstruieren. Wir betrachten Unterräume W_j als orthogonale Komplemente von V_j in V_{j-1}, das heißt $V_{j-1} = V_j \perp W_j$. Wir sehen leicht, dass dann

$$L_2(\mathbb{R}) = \bigoplus_{n\in\mathbb{Z}} W_j$$

gilt, also $W_j \perp W_k$, $W_j \cap W_k = \{0\}$ für $j \neq k$. Damit erhalten wir die Zerlegung

$$\begin{aligned} f_n &= g_{n+1} + f_{n+1} \\ &= g_{n+1} + g_{n+2} + \cdots + g_{n+m} + f_{n+m} \text{ mit } f_j \in V_j, g_j \in W_j. \end{aligned}$$

Dabei ist f_j die Darstellung von f auf der Skala j und g_j ist die orthogonale Projektion in den Raum W_j. Die Funktionen g_j enthalten lediglich die Details, die f_j von f_{j-1} unterscheiden.

4.31 Satz: Sei $\{V_j\}_{j\in\mathbb{Z}}$ eine MRA, die von einer orthogonalen Skalierungsfunktion $\varphi \in V_0$ erzeugt wird. Seien $\{h_k\}_{k\in\mathbb{Z}}$ die zugehörigen Verfeinerungskoeffizienten. Wir setzen

$$\psi(x) = \sqrt{2} \sum_{k\in\mathbb{Z}} h_k^{\perp} \varphi(2x-k) \text{ mit } h_k^{\perp} = (-1)^k h_{1-k}. \tag{4.36}$$

Die definierte Funktion $\psi \in V_{-1}$ ist ein Wavelet mit der Calderon-Konstante $C_\psi = 2\ln 2$ und die Funktionen

$$\psi_{j,k} = 2^{-\frac{j}{2}} \psi(2^{-j}x - k) \text{ mit } j,k \in \mathbb{Z}$$

bilden eine Orthonormalbasis des $L_2(\mathbb{R})$. Darüber hinaus gilt

$$W_j = \operatorname{span}\{\psi_{j,k} \mid k \in \mathbb{Z}\},$$

[Lo98], Seite 122.

4.32 Bemerkung:

(1) Die Waveletdarstellung (4.36) ist äquivalent zu der Darstellung

$$\psi_{j,k}(x) = \sum_{l\in\mathbb{Z}} h_l^{\perp} \varphi_{j-1,2k+l}(x).$$

(2) Das von der MRA erzeugte Wavelet ist nicht eindeutig bestimmt. Auch für ein Wavelet, das durch $h_k^{\perp} = (-1)^k h_{1-k+2l}$, $l \in \mathbb{Z}$, erzeugt wurde, gilt Satz 4.31.

(3) Jede MRA erzeugt orthogonale Wavelets. Die Umkehrung der Aussage gilt nicht. Ein Gegenbeispiel für diese Behauptung geht auf *Mallat* zurück und ist zum Beispiel in [Lo98], Seite 125, zu finden.

Satz 4.31 veranschaulichen wir am folgenden Beispiel.

4.33 Beispiel: Wir betrachten die folgende Skalierungsfunktion

$$\varphi(x) = \chi_{[0,1)}(x) = \begin{cases} 1 & \text{für } x \in [0,1) \\ 0 & \text{sonst.} \end{cases}$$

Dann ist $\varphi_{j,k}(x) = \chi_{[2^j k, 2^j(k+1))}(x)$ und wir erhalten

$$\begin{aligned} V_j &= \operatorname{span}\{\varphi_{j,k} \mid k \in \mathbb{Z}\} \\ &= \{f \in L_2(\mathbb{R}) \mid f \text{ konstant auf } [2^j k, 2^j(k+1)),\ k \in \mathbb{Z}\}. \end{aligned}$$

Die Überprüfung der Bedingungen (1) bis (5) der Definition 4.23 ergibt, dass die Menge $\{V_j\}_{j\in\mathbb{Z}}$ eine MRA erzeugt. Dabei ist $\{\varphi_{j,k}\}_{k\in\mathbb{Z}}$ eine Orthonormalbasis von V_j.
φ ist eine verfeinerbare Funktion mit der Verfeinerungsgleichung

$$\varphi = \sqrt{2}\left(\frac{1}{\sqrt{2}}\varphi(2\cdot) + \frac{1}{\sqrt{2}}\varphi(2\cdot -1)\right),$$

also $h_0 = h_1 = \frac{1}{\sqrt{2}}$.
Mit Hilfe der Gleichung (4.36) können wir die orthogonale Waveletbasis ermitteln. Es gilt

$$\psi = \sqrt{2}\left(\frac{1}{\sqrt{2}}\varphi(2\cdot) - \frac{1}{\sqrt{2}}\varphi(2\cdot -1)\right)$$

und damit $h_0^\perp = \frac{1}{\sqrt{2}}$ und $h_1^\perp = -\frac{1}{\sqrt{2}}$. Insgesamt ergibt die Rechnung das Haarwavelet

$$\psi(x) = \begin{cases} 1 & \text{für } 0 \le x < \frac{1}{2} \\ -1 & \text{für } \frac{1}{2} \le x < 1 \\ 0 & \text{sonst.} \end{cases}$$

Abschließend können wir feststellen: Starten wir mit einer beliebigen Funktion, die den Bedingungen des Satzes 4.30 genügt, so können wir direkt eine MRA erzeugen. Die Skalierungsfunktion können wir mit Hilfe des Verfahrens (4.33) orthogonalisieren und dadurch eine Orthonormalbasis erzwingen. In der Regel ist dann aber die Skalierungsfunktion nicht explizit ermittelbar, das heißt sie verfügt über unendliche viele Koeffizienten. Mit Hilfe der Verfeinerungskoeffizienten und der Waveletdarstellung (4.36) erhalten wir eine orthogonale Waveletbasis.
In Abschnitt 4.8 werden wir noch ein praktikables Verfahren zur Konstruktion orthogonaler Waveletbasen kennen lernen.

4.7 Schnelle Wavelettransformation

In diesem Abschnitt diskutieren wir die *schnelle Wavelettransformation* nach *Stephane Mallat*, auch *Mallat-Algorithmus* genannt, zur Berechnung der diskreten Wavelettransformation (DWT) und zur Signalrekonstruktion. Die Grundlage des Algorithmus ist die im letzten Abschnitt eingeführte MRA.
Wir betrachten eine Funktion $f \in L_2(\mathbb{R})$ und deren Projektion $f_n \in V_n$, wobei V_n ein Unterraum der orthogonalen MRA und $n \in \mathbb{Z}$ ist. Die zu der MRA gehörende Skalierungsfunktion bezeichnen wir mit φ. Dann können wir f_n als

$$f_n(x) = \sum_{k \in Z} c_{n,k}(f)\varphi_{n,k}(x) \text{ mit } c_{n,k}(f) = \langle f, \varphi_{n,k}\rangle_{L_2}$$

darstellen. Sei ψ das zur MRA gehörige Wavelet. Dann bildet $\{\psi_{j,k}\}_{j,k\in\mathbb{Z}}$ eine Orthonormalbasis des $L_2(\mathbb{R})$. Zur Berechnung der Wavelettransformation $L_\psi f(j,k) = \langle f, \psi_{j,k}\rangle_{L_2}$ definieren wir

$$\begin{aligned} c_{j,k} &= \langle f, \varphi_{j,k}\rangle_{L_2} \text{ mit } c_j = \{c_{j,k}\}_{k\in\mathbb{Z}} \in \ell_2(\mathbb{Z}), \\ d_{j,k} &= \langle f, \psi_{j,k}\rangle_{L_2} \text{ mit } d_j = \{d_{j,k}\}_{k\in\mathbb{Z}} \in \ell_2(\mathbb{Z}). \end{aligned}$$

Wir erhalten mit Hilfe der Verfeinerungsgleichung (4.31) und der Waveletdarstellung (4.36) die Beziehungen

$$\begin{aligned} c_{j,k} &= \sum_{l\in\mathbb{Z}} h_l \langle f, \varphi_{j-1,2k+l}\rangle_{L_2} = \sum_{l\in\mathbb{Z}} h_l c_{j-1,2k+l} = \sum_{l\in\mathbb{Z}} h_{l-2k} c_{j-1,l} \text{ und} \\ d_{j,k} &= \sum_{l\in\mathbb{Z}} h_l^\perp \langle f, \varphi_{j-1,2k+l}\rangle_{L_2} = \sum_{l\in\mathbb{Z}} h_l^\perp c_{j-1,2k+l} = \sum_{l\in\mathbb{Z}} h_{l-2k}^\perp c_{j-1,l}. \end{aligned} \quad (4.37)$$

Ausgehend von einer Startfolge c_n (wobei im Allgemeinen $n = 0$ ist), welche die Funktion bzw. das Signal repräsentiert, lässt sich die diskrete Wavelettransformation rekursiv berechnen.
Bevor wir uns mit der Rücktransformation beschäftigen, wollen wir den Zerlegungsvorgang mit Hilfe zweier Operatoren formalisieren. Wir definieren

$$\begin{aligned} S: \ \ell_2(\mathbb{Z}) &\to \ell_2(\mathbb{Z}) \\ c &\mapsto Sc \qquad = \left\{ (Sc)_k = \sum_{l\in\mathbb{Z}} h_{l-2k} c_l \right\}_{k\in\mathbb{Z}} \text{ und} \end{aligned} \quad (4.38)$$

$$\begin{array}{ccccccccc} c_n & \xrightarrow{S} & c_{n+1} & \xrightarrow{S} & c_{n+2} & \cdots & c_{n+n'-1} & \xrightarrow{S} & c_{n+n'} \\ & \overset{W}{\searrow} & & \overset{W}{\searrow} & & \cdots & & \overset{W}{\searrow} & \\ & & d_{n+1} & & d_{n+2} & & \cdots & & d_{n+n'} \end{array}$$

Abbildung 5: Schema der schnellen Wavelettransformation.

$$\begin{aligned} W: \ \ell_2(\mathbb{Z}) &\to \ell_2(\mathbb{Z}) \\ c &\mapsto Wc \qquad = \left\{ (Wc)_k = \sum_{l\in\mathbb{Z}} h^{\perp}_{l-2k} c_l \right\}_{k\in\mathbb{Z}} . \end{aligned} \tag{4.39}$$

Wir bezeichnen S als *Skalierungsoperator* und W als *Waveletoperator*. Ein Zerlegungsschritt ist dann durch

$$c_j = Sc_{j-1} \text{ und } d_j = Wd_{j-1}$$

gegeben.
Bei der Rücktransformation wollen wir die Ausgangsfolge c_n aus den Folgen $c_{n+n'}$ und $\{d_j\}_{j=n+1}^{n+n'}$ rekonstruieren. Wir nutzen erneut die Eigenschaften der MRA und betrachten die orthogonale Zerlegung des Raumes $V_j = V_{j+1} \perp W_{j+1}$. Es gilt

$$\sum_{k\in\mathbb{Z}} c_{j,k}\varphi_{j,k} = \sum_{l\in\mathbb{Z}} c_{j+1,l}\varphi_{j+1,l} + \sum_{l\in\mathbb{Z}} d_{j+1,l}\psi_{j+1,l}.$$

Erneut verwenden wir die Verfeinerungsgleichung (4.31) sowie die Waveletdarstellung (4.36) und erhalten

$$\sum_{k\in\mathbb{Z}} c_{j,k}\varphi_{j,k} = \sum_{l\in\mathbb{Z}} c_{j+1,l} \sum_{i\in\mathbb{Z}} h_i\varphi_{j,2l+i} + \sum_{l\in\mathbb{Z}} d_{j+1,l} \sum_{i\in\mathbb{Z}} h^{\perp}_i\varphi_{j,2l+i}.$$

Ein Koeffizientenvergleich ergibt die Rekursionsformel

$$c_{j,k} = \sum_{l\in\mathbb{Z}} \left(h_{k-2l}c_{j+1,l} + h^{\perp}_{k-2l}d_{j+1,l} \right). \tag{4.40}$$

Ähnlich wie bei der Transformation können wir auch die Rücktransformation mit Hilfe zweier Operatoren schreiben. Wir führen zu diesem Zweck die zu S und W adjungierten Operatoren S^* und W^* ein.

$$\begin{array}{ccccccccc} c_{n+n'} & \overset{S^*}{\rightarrow} & c_{n+n'-1} & \overset{S^*}{\rightarrow} & c_{n+n'-2} & \cdots & c_{n-1} & \overset{S^*}{\rightarrow} & c_n \\ & \overset{W^*}{\nearrow} & & \overset{W^*}{\nearrow} & & \cdots & & \overset{W^*}{\nearrow} & \\ d_{n+n'} & & d_{n+n'-1} & & \cdots & & & d_{n-1} & \end{array}$$

Abbildung 6: Schema der schnellen Rekonstruktion.

4.34 Lemma: Die zu den Operatoren S und W, definiert wie in (4.38) und (4.39), adjungierten Operatoren S^* und W^* sind gegeben durch

$$\begin{aligned} S^* :\ & \ell_2(\mathbb{Z}) \to \ell_2(\mathbb{Z}) \\ & c \mapsto S^*c \quad = \left\{ (S^*c)_k = \sum_{l\in\mathbb{Z}} h_{k-2l} c_l \right\}_{k\in\mathbb{Z}} \text{ und} \qquad (4.41) \\ W^* :\ & \ell_2(\mathbb{Z}) \to \ell_2(\mathbb{Z}) \\ & c \mapsto W^*c \quad = \left\{ (W^*c)_k = \sum_{l\in\mathbb{Z}} h^\perp_{k-2l} c_l \right\}_{k\in\mathbb{Z}} . \qquad (4.42) \end{aligned}$$

Beweis: Aus der definierenden Eigenschaft eines adjungierten Operators

$$\langle Sc, b\rangle_{\ell_2} = \langle c, S^*b\rangle_{\ell_2}$$

folgt die Behauptung durch Vertauschung der Reihenfolge der Summanden. □

Ein einzelner Rekonstruktionsschritt ist dann

$$c_{j-1} = S^*c_j + W^*d_j.$$

Mit Hilfe der Rücktransformation sind wir in der Lage, eine Ausgangsfolge rekursiv zu rekonstruieren. Ein Schema finden wir in Abbildung 6. [Lo98], Seite 135, zeigt, dass der Aufwand für die Zerlegung und Rekonstruktion bei $O(N)$ liegt, wobei N die Länge der Ausgangsfolge ist.
Wir verdeutlichen den Vorgang der Zerlegung und der Rekonstruktion an einem Beispiel.

4.35 Beispiel: Sei $\varphi = \chi_{[0,1)}$ die Skalierungsfunktion und ψ das Haarwavelet. Dann erhalten wir nach Beispiel 4.33 die folgenden Verfeinerungs- und Rekonstruktionskoeffizienten

$$h_0 = h_1 = \frac{1}{\sqrt{2}} \text{ sowie } h_0^\perp = \frac{1}{\sqrt{2}},\ h_1^\perp = -\frac{1}{\sqrt{2}}.$$

Wir starten im Raum V_0, das Ausgangssignal sei

$$c_0 = \{c_{0,k}\}_{k=0,\ldots,3} = \{9, 1, 2, 0\}.$$

Wir wenden (4.37) auf c_0 an und erhalten die Koeffizientenfolgen c_1 und d_1. Eine abermalige Anwendung von (4.37) führt zu c_2 und d_2. Eine schematische Darstellung der Zerlegungsschritte sehen wir in Abbildung 7. Wenden wir nun (4.40) auf c_2 und d_2 sowie auf c_1 und d_1 an, so erhalten wir das Ausgangssignal c_0.

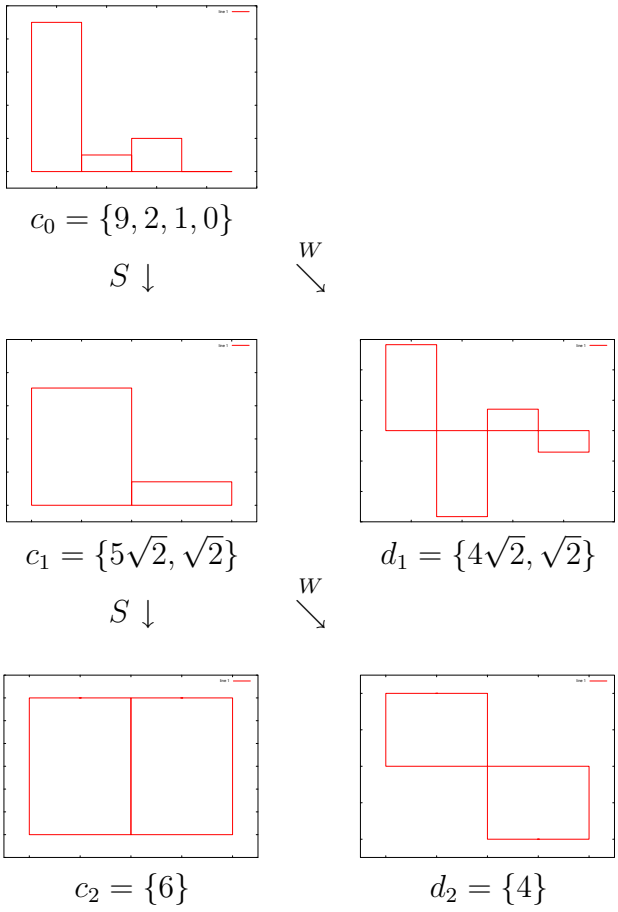

Abbildung 7: Beispiel einer Zerlegung eines aus vier Werten bestehenden Signals c_0 mit Hilfe des Haarwavelets.

4.8 Wavelets mit kompaktem Träger

In diesem Abschnitt diskutieren wir eine weitere Möglichkeit der Konstruktion diskreter Wavelets. In unseren Ausführungen orientieren wir uns hauptsächlich an [Lo98].
Wir stellen zusätzliche Bedingungen an die Skalierungsfunktion, unter denen φ eine Lösung der Verfeinerungsgleichung (4.31) ist. Dann erfüllt φ eine Orthogonalitätsbedingung bezüglich einer Skala, das heißt

$$\int_{\mathbb{R}} \varphi(x)\varphi(x-k)dx = 0.$$

Die Lösung φ ist bis auf Konstanten eindeutig und ihr Träger liegt im Intervall $[0, K-1]$ falls die Folge der Verfeinerungskoeffizienten endlich ist

$$h_k = 0 \text{ für } k < 0 \text{ und } k \geq K.$$

Wir stellen hier eine Methode zur Lösung der Verfeinerungsgleichung und zur Konstruktion von Wavelets mit kompakten Trägern vor. Diese Methode wird in der Literatur manchmal auch *Methode der graphischen Iteration* genannt und geht auf *Ingrid Daubechies* zurück.
Wir betrachten die Verfeinerungsgleichung (4.31) mit endlich vielen Koeffizienten $\{h_k\}_{k=0}^{K-1}$ und der Normierung $\sum_{k=0}^{K-1} h_k = \sqrt{2}$. Wir wollen die Skalierungsfunktion $\varphi \in V_0$ nach der Basis des Raumes V_{-j} entwickeln. Dann gilt

$$\varphi(x) = \sum_{k\in\mathbb{Z}} c_{-j,k} 2^{\frac{j}{2}} \varphi(2^j x - k) \text{ mit } c_{-j,k} \in \mathbb{R},$$

wobei für große j die Basisfunktion $\varphi(2^j x - k)$ die δ-Distribution im Punkt $x = 2^{-j}k$ approximiert. Damit nähert sich $c_{-j,k}$ für große Skalen j dem Funktionswert $\varphi(2^{-j})k$ an. Setzen wir

$$\chi_{[a,b]}(x) = \begin{cases} 1 & \text{für } x \in [a,b] \\ 0 & \text{sonst,} \end{cases}$$

so können wir das *Histogramm der Verfeinerungskoeffizienten* durch

$$\varphi_j(x) = \sum_{k\in\mathbb{Z}} c_{-j,k} 2^{\frac{j}{2}} \chi_{[-\frac{1}{2},\frac{1}{2}]}(2^j x - k)$$

einführen. Nach [Lo98] konvergiert die Folge $\{\varphi_j\}_{j\in\mathbb{Z}}$ für $j \to \infty$ unter gewissen Voraussetzungen gegen eine stetige Lösung φ_∞ der Verfeinerungsgleichung.

Für die Praxis ist es günstig, dass sich die Koeffizienten $c_{j,k}$ rekursiv berechnen lassen. Nach [Lo98], Seite 157, erfüllen die Koeffizienten $c_{-j,k}$ die Rekursion

$$c_{0,k} = \delta_{0,k} \text{ für } k = 0, \ldots, K-1,$$
$$c_{-j,k} = \sum_{l \in \mathbb{Z}} h_{k-2l} c_{-j+1,l}.$$

Also können wir die Koeffizienten von φ_j in entsprechender Weise wie bei der schnellen inversen Wavelettransformation berechnen.
Das diskrete Wavelet können wir nach Satz 4.31 durch

$$\psi(x) = \sqrt{2} \sum_{l=2-K}^{1} (-1)^l h_{1-l} \varphi(2x - l)$$

bestimmen. Setzen wir $x = 2^{-(j+1)}k$, dann folgt

$$\psi(2^{-(j+1)}k) = \sqrt{2} \sum_{l=2-K}^{1} (-1)^l h_{1-l} c_{-j,k-2^j l} \text{ für } k \in \mathbb{Z}.$$

Den so eingeführten Algorithmus wollen wir zur Konstruktion einiger Beispiele für Skalierungsfunktionen und Wavelets mit kompakten Trägern nutzen.

4.36 Beispiel: Wir betrachten einige Beispiele, die sich im Wesentlichen in der Anzahl der von Null verschiedenen Verfeinerungskoeffizienten unterscheiden.

(1) Hat die Skalierungsfunktion genau einen von Null verschiedenen Koeffizienten $h_0 = \sqrt{2}$, so lautet die Verfeinerungsgleichung

$$\varphi = 2\varphi(2\cdot).$$

Die φ_j werden für $j \to \infty$ dünner und höher und die Folge konvergiert gegen

$$\varphi_\infty(x) = \delta(x).$$

(2) Liegen zwei von Null verschiedene Koeffizienten vor, so gilt $h_0 = h_1 = \frac{1}{\sqrt{2}}$ und wir erhalten die Verfeinerungsgleichung

$$\varphi = \varphi(2\cdot) + \varphi(2 \cdot -1).$$

Der Iterationsprozess liefert die charakteristische Funktion auf einem Intervall $[a, b]$ und für ψ das Haarwavelet mit $h_0^\perp = h_0$ und $h_1^\perp = -h_1$.

(3) Für drei Koeffizienten $h_0 = h_2 = \frac{1}{2\sqrt{2}}$ und $h_1 = \frac{1}{\sqrt{2}}$ erhalten wir die so genannte *Hut-Funktion* und das zugehörige Wavelet mit den Koeffizienten $h^\perp_{-1} = h^\perp_1 = \frac{1}{2\sqrt{2}}$, $h^\perp_0 = \frac{1}{\sqrt{2}}$.

(4) Die Koeffizienten

$$\begin{aligned} h_0 &= \tfrac{1+\sqrt{3}}{4\sqrt{2}} h_1 &&= \frac{3+\sqrt{3}}{4\sqrt{2}} \text{ und} \\ h_2 &= \tfrac{3-\sqrt{3}}{4\sqrt{2}} h_3 &&= \frac{1-\sqrt{3}}{4\sqrt{2}} \end{aligned}$$

liefern die Daubechies-Skalierungsfunktion $\varphi = D_2$. Das zugehörige Wavelet, das mit W_2 bezeichnet wird, hat dann die Koeffizienten

$$\begin{aligned} h^\perp_{-2} &= \tfrac{1-\sqrt{3}}{4\sqrt{2}} h^\perp_{-1} &&= \frac{3-\sqrt{3}}{4\sqrt{2}} \text{ sowie} \\ h^\perp_0 &= \tfrac{3+\sqrt{3}}{4\sqrt{2}} h^\perp_1 &&= \frac{1+\sqrt{3}}{4\sqrt{2}}. \end{aligned}$$

Das *Daubechies-2-Wavelet* ist orthogonal und besitzt einen kompakten Träger. D_2 hat vier Verfeinerungs- und vier Waveletkoeffizienten. Die iterative Entstehung sehen wir in Abbildung 8 für verschiedene Iterationstiefen 1 bis 6.

(5) Das *Daubechies-4-Wavelet* hat insgesamt acht von Null verschiedene Verfeinerungskoeffizienten, die wir an dieser Stelle nicht angeben. Dieses Wavelet ist ebenfalls orthogonal. Seine Glattheitseigenschaften sind in [Da99], Kapitel 7 zu finden. Das *Daubechies-5-Wavelet* hat hingegen zehn Verfeinerungskoeffizienten, die Entstehung sehen wir für die ersten sieben Stufen in Abbildung 9.

(6) Den *kubischen B-Spline* erhalten wir für die Parameter $h_0 = h_4 = \frac{1}{8\sqrt{2}}$, $h_1 = h_3 = \frac{1}{2\sqrt{2}}$ und $h_2 = \frac{3}{4\sqrt{2}}$, vergleiche dazu [Lo98], Seite 145. Das zugehörige Wavelet hat die Waveletkoeffizienten $h^\perp_{-3} = h^\perp_1 = -\frac{1}{8\sqrt{2}}$, $h^\perp_{-2} = h^\perp_0 = \frac{1}{2\sqrt{2}}$, $h^\perp_{-1} = -\frac{3}{4\sqrt{2}}$. Eine Abbildung des Wavelets finden wir zum Beispiel in [Da99].

Anwendungen der schnellen Wavelettransformation mit diesen und anderen Skalierungsfunktionen und Wavelets finden wir in Abschnitt 9, wo Filtertechniken auf Grundlage der MRA erläutert werden.

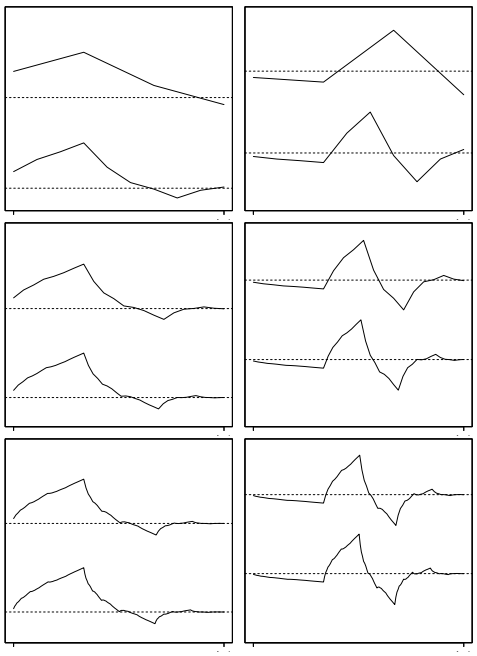

Abbildung 8: Iterative Entstehung der Skalierungsfunktion D_2 (links) und des Wavelets (rechts) W_2

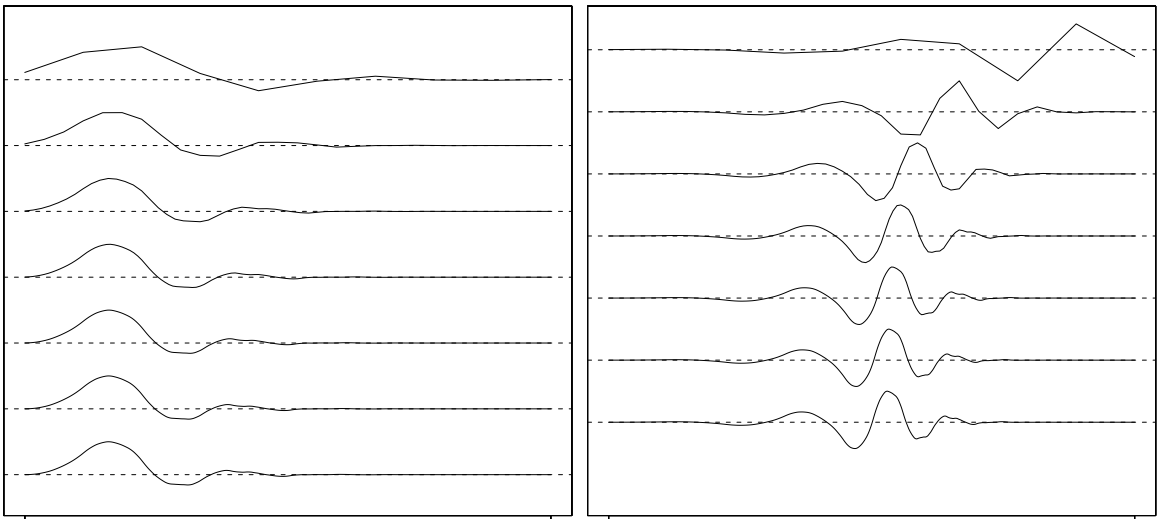

Abbildung 9: Iterative Entstehung der Skalierungsfunktion D_5 (links) und des Wavelets (rechts) W_5

4.9 Biorthogonale Wavelets

Der Gegenstand der letzten Abschnitte war die Konstruktion orthogonaler Skalierungsfunktionen und orthogonaler Wavelets. Diese Funktionen haben für praktische Zwecke viele Vorteile, wie zum Beispiel die hierarchische Anordnung oder die rekursive Berechenbarkeit, sie können mit kompakten Trägern und beliebiger Differenzierbarkeitssordnung konstruiert werden.
Allerdings ist kein Wavelet mit kompaktem Träger symmetrisch und der Träger eines N–fach differenzierbaren Wavelets ist um ein Vielfaches größer als der Träger eines Splines derselben Differenzierbarkeitsordnung.
Leider ist es nicht möglich, orthogonale Wavelets mit allen obigen Eigenschaften zu konstruieren, vgl. dazu [Da99] oder [Ma98].
Einen Ausweg finden wir, indem wir die strikte Orthogonalität der Translate

$$\langle \psi_{j,k}, \psi_{m,l} \rangle = \delta_{j,m}\delta_{k,l}$$

aufgeben und statt dessen Funktionspaare $\{\psi, \widetilde{\psi}\}$ suchen, deren Dilatationen und Translate

$$\begin{aligned} \psi_{j,k} &= 2^{-j/2}\psi(2^{-j}\cdot - k), \\ \widetilde{\psi}_{j,k} &= 2^{-j/2}\widetilde{\psi}(2^{-j}\cdot - k) \end{aligned}$$

biorthogonale Basen des $L_2(\mathbb{R})$ bilden, das heißt

$$\langle \psi_{j,k}, \widetilde{\psi}_{m,l} \rangle = \delta_{j,m}\delta_{k,l} \tag{4.43}$$

und für alle $f \in L_2(\mathbb{R})$ gilt

$$f = \sum_{j\in\mathbb{Z}}\sum_{k\in\mathbb{Z}} \langle f, \widetilde{\psi}_{j,k}\rangle \psi_{j,k} = \sum_{j\in\mathbb{Z}}\sum_{k\in\mathbb{Z}} \langle f, \psi_{j,k}\rangle \widetilde{\psi}_{j,k}. \tag{4.44}$$

Die Konstruktion solcher biorthogonaler Wavelets erfolgt ebenfalls über die Skalierungsfunktionen $\{\varphi, \widetilde{\varphi}\}$ und die zugehörigen Verfeinerungsgleichungen

$$\varphi = \sqrt{2}\sum_{j\in\mathbb{Z}} h_j \varphi(2\cdot - k), \tag{4.45}$$

$$\widetilde{\varphi} = \sqrt{2}\sum_{j\in\mathbb{Z}} \widetilde{h}_j \widetilde{\varphi}(2\cdot - k). \tag{4.46}$$

Unter den Voraussetzungen von [Lo98], Satz 2.4.41, Seite 185, erhalten wir die Orthogonalität der Skalierungsfunktionen φ und $\widetilde{\varphi}$, also

$$\int_{\mathbb{R}} \varphi(x)\widetilde{\varphi}(x-k)dx = 0 \text{ für } k \in \mathbb{Z}\backslash\{0\}.$$

Darüber hinaus gilt für die Funktionen ψ und $\widetilde{\psi}$

$$\psi = \sqrt{2}\sum_{k\in\mathbb{Z}}(-1)^k\widetilde{h}_{1-k}\varphi(2\cdot-k),$$
$$\widetilde{\psi} = \sqrt{2}\sum_{k\in\mathbb{Z}}(-1)^k h_{1-k}\widetilde{\varphi}(2\cdot-k)$$

und

$$\int_{\mathbb{R}} \psi_{j,k}(x)\widetilde{\psi}_{m,l}dx = \delta_{j,m}\delta_{k,l}.$$

Eine Stabilitätsaussage wie bei den orthogonalen Wavelets, die eine stabile Zerlegung und Rekonstruktion haben, erhalten wir nicht sofort. Es gilt

$$C_1\left(\sum_{j\in\mathbb{Z}}\sum_{k\in\mathbb{Z}}|\langle f,\psi_{j,k}\rangle|^2\right)^{1/2} \leq \|f\|_2 \leq C_2\left(\sum_{j\in\mathbb{Z}}\sum_{k\in\mathbb{Z}}|\langle f,\psi_{j,k}\rangle|^2\right)^{1/2}. \tag{4.47}$$

Die Bedingung (4.47) garantiert, dass das Wavelet ψ eine Riesz-Basis erzeugt.

4.37 Bemerkung: In [Lo98], Satz 2.4.41, Seite 185 finden wir einen Satz, der zeigt, wann

$$\{\psi_{j,k} \mid j,k\in\mathbb{Z}\} \text{ und } \{\widetilde{\psi}_{j,k} \mid j,k\in\mathbb{Z}\}$$

Riesz-Basen des $L_2(\mathbb{R})$ bilden.

Wie funktioniert nun die schnelle Wavelettransformation mit biorthogonalen Wavelets? Wir beginnen mit der Folge $c_0 \in \ell(\mathbb{Z})$ und berechnen die einzelnen Skalen mit Hilfe der Operatoren S und W, definiert in (4.38) und (4.39). Das heißt wir berechnen rekursiv

$$c_n = Sc_{n-1} \text{ und } d_n = Wd_{n-1}.$$

Für die Rekonstruktion benutzen wir, wie eingangs angekündigt, nicht die adjungierten Operatoren S^* und W^*. Wir führen neue Rekonstruktionsoperatoren zu den Koeffizienten $\{\widetilde{h}_k\}$ und $\{\widetilde{h}_k^{\perp}\}$ ein, die definiert sind als

$$\begin{aligned}\widetilde{S}^*:\ \ell_2(\mathbb{Z}) &\to \ell_2(\mathbb{Z})\\ c &\mapsto \widetilde{S}^*c \quad = \left\{(\widetilde{S}^*c)_k = \sum_{l\in\mathbb{Z}}\widetilde{h}_{l-2k}c_l\right\}_{k\in\mathbb{Z}}\end{aligned} \tag{4.48}$$

und

$$\begin{aligned}\widetilde{W}^* :\ \ell_2(\mathbb{Z}) &\to \ell_2(\mathbb{Z}) \\ c &\mapsto \widetilde{W}^* c \quad = \left\{ (\widetilde{W}^* c)_k = \sum_{l\in\mathbb{Z}} \widetilde{h}^{\perp}_{l-2k} c_l \right\}_{k\in\mathbb{Z}} . \end{aligned} \tag{4.49}$$

Ähnlich wie bei den orthogonalen Wavelets sind für praktische Zwecke endliche Koeffizientenfolgen geeignet. Wir fordern, dass sich die Operatoren $\widetilde{S}^*, S, \widetilde{W}^*, W$ folgendermaßen zur Identität auf $\ell_2(\mathbb{Z})$ ergänzen

$$\widetilde{S}^* S + \widetilde{W}^* W = I. \tag{4.50}$$

Dann ist mit diesen Operatoren die Zerlegung und Rekonstruktion in $O(N)$ Operationen durchführbar, falls N die Länge des Ausgangssignals ist. Eine Möglichkeit, solche Operatoren zu erzeugen, wäre die Folge $\{h_k\}_{k\in\mathbb{Z}}$ vorzugeben und die Koeffizienten $\{\widetilde{h}_k\}_{k\in\mathbb{Z}}$ so zu bestimmen, dass

$$\begin{aligned} \sum_{j\in\mathbb{Z}} h_j \widetilde{h}_{j-2k} &= \delta_{0,k} \text{ und} \\ h^{\perp}_k = (-1)^k \widetilde{h}_{1-k} \text{ sowie } \widetilde{h}^{\perp}_k &= (-1)^k h_{1-k} \end{aligned} \tag{4.51}$$

erfüllt sind. Dann wird durch die vier Folgen $\{h_k\}_{k\in\mathbb{Z}}$, $\{h^{\perp}_k\}_{k\in\mathbb{Z}}$, $\{\widetilde{h}_k\}_{k\in\mathbb{Z}}$, $\{\widetilde{h}^{\perp}_k\}_{k\in\mathbb{Z}}$ ein Quadrupel von Operatoren mit der Eigenschaft (4.50) erzeugt. Wir kommen jetzt zur Konstruktion von biorthogonalen Wavelets auf Splinebasis. Hierfür betrachten wir die Fourier-Transformation der Verfeinerungsgleichung (4.31), es gilt

$$\widehat{\varphi}(\xi) = H(\xi/2)\widehat{\varphi}(\xi/2) \text{ und } \widehat{\widetilde{\varphi}}(\xi) = \widetilde{H}(\xi/2)\widehat{\widetilde{\varphi}}(\xi/2)$$

mit

$$H(\xi) = \frac{1}{\sqrt{2}} \sum_{k\in\mathbb{Z}} h_k e^{-ik\xi} \text{ und } \widetilde{H}(\xi) = \frac{1}{\sqrt{2}} \sum_{k\in\mathbb{Z}} \widetilde{h}_k e^{-ik\xi}.$$

Grundlegende Eigenschaften der trigonometrischen Polynome H und $\widetilde{H}$ finden wir in [Lo98], Seiten 146-153.
Wir wählen φ als κ–ten Spline, dann gilt

$$\varphi(x) = N^{\kappa}(x) \text{ und } H(\xi) = \begin{cases} \left(\cos\frac{\xi}{2}\right)^{\kappa} & \text{falls } \kappa = 2k, k \in \mathbb{Z} \\ e^{-\frac{i\xi}{2}} \left(\cos\frac{\xi}{2}\right)^{\kappa} & \text{falls } \kappa = 2k+1, k \in \mathbb{Z}. \end{cases}$$

Dabei ist mit den Bezeichnungen des Abschnittes 3

$$T = \mathbb{Z} \text{ also } t_j = j \text{ für } j \in \mathbb{Z}$$

und

$$N^0(x) = N_0^0(x) = \chi_{[0,1]}(x).$$

Entsprechend erhalten wir $N^\kappa(x) = N_0^\kappa(x)$, es handelt sich also um einen B-Spline im Intervall $[0, \kappa + 1)$.
Die Suche nach einer Koeffizientenfolge $\widetilde{h}$ und nach $\widetilde{H}$ führt zu trigonometrischen Polynomen, welche wir als Nächstes vorstellen.
Nach [Da99], Seite 271, hat das gesuchte trigonometrische Polynom $\widetilde{H}$ die Gestalt

$$\widetilde{H}_{\kappa,\widetilde{\kappa}}(\xi) = \begin{cases} \left(\cos\frac{\xi}{2}\right)^{\widetilde{\kappa}} \sum_{l=0}^{k+\widetilde{k}+1} \binom{k+\widetilde{k}-1+l}{l} \left(\sin^2\frac{\xi}{2}\right)^l & \text{falls } \widetilde{\kappa} = 2\widetilde{k} \\ e^{-\frac{i\xi}{2}} \left(\cos\frac{\xi}{2}\right)^{\widetilde{\kappa}} \sum_{l=0}^{k+\widetilde{k}} \binom{k+\widetilde{k}+l}{l} \left(\sin^2\frac{\xi}{2}\right)^l & \text{falls } \widetilde{\kappa} = 2\widetilde{k}+1. \end{cases} \tag{4.52}$$

Dabei gilt $\kappa = 2k$ oder $\kappa = 2k + 1$ für ein $k \in \mathbb{Z}$. Denn für dieses $\widetilde{H}_{\kappa,\widetilde{\kappa}}$ gilt

$$\overline{H(\xi)}\widetilde{H}_{\kappa,\widetilde{\kappa}}(\xi) + \overline{H(\xi+\pi)}\widetilde{H}_{\kappa,\widetilde{\kappa}}(\xi+\pi) = 1 \text{ und}$$
$$1 = H(0) = \widetilde{H}_{\kappa,\widetilde{\kappa}}(0). \tag{4.53}$$

Diese Aussage liefert uns allerdings nur die diskrete Biorthogonalität der Filter H und $\widetilde{H}_{\kappa,\widetilde{\kappa}}$. Die L_2–Konvergenz der Produkte

$$\prod_{j\geq 1} H(2^{-j}\xi) \text{ und } \prod_{j\geq 1} \widetilde{H}_{\kappa,\widetilde{\kappa}}(2^{-j}\xi)$$

kann man nur unter gewissen Bedingungen zeigen und diese sind nicht für alle Paare $\kappa, \widetilde{\kappa}$ erfüllt.
Eine allgemein gültige Gestalt der Koeffizienten der Polynome $H(\xi)$ und $\widetilde{H}_{\kappa,\widetilde{\kappa}}(\xi)$ finden wir in [Pr06]. Es gilt

$$h_k = \sqrt{2}\, 2^{1-\kappa} \binom{\kappa}{k + \lfloor \frac{\kappa}{2} \rfloor} \text{ für } -\left\lfloor \frac{\kappa}{2} \right\rfloor \leq k \leq \left\lceil \frac{\kappa}{2} \right\rceil \tag{4.54}$$

sowie

$$\widetilde{h}_k = \sqrt{2} \sum_{j=0}^{\frac{\kappa+\widetilde{\kappa}}{2}-1} \sum_{l=0}^{2j} 2^{-\widetilde{\kappa}-2j} (-1)^{j+l} \binom{\widetilde{\kappa}}{k + \lfloor \frac{\widetilde{\kappa}}{2} \rfloor - l + j} \binom{\frac{\kappa+\widetilde{\kappa}}{2} - 1 + j}{j} \binom{2j}{l} \tag{4.55}$$

für $-\widetilde{\kappa} - \left\lfloor \frac{\kappa}{2} \right\rfloor - 1 \leq k \leq \widetilde{\kappa} + \left\lceil \frac{\kappa}{2} \right\rceil - 1$.
Im nachfolgenden Beispiel zeigen wir einige Paare $\kappa, \widetilde{\kappa}$ und deren Darstellungen für H und $\widetilde{H}_{\kappa,\widetilde{\kappa}}$, für welche die oben angesprochenen Bedingungen erfüllt sind.

4.38 Beispiel: Sei abkürzend $z = e^{i\xi}$. Wir geben für $\kappa = 1, 2, 3$ jeweils nur die ersten Darstellungen von $\widetilde{H}_{\kappa,\widetilde{\kappa}}$ an. Für weitere Ergebnisse verweisen wir auf [Da99].

(1) Sei

$$H(\xi) = H_1(\xi) = \frac{1}{2} + \frac{z}{2}.$$

Dann erhalten wir folgende Darstellungen für $\widetilde{H}_{1,\widetilde{\kappa}}$.

(i) Für $\widetilde{\kappa} = 1$ ist

$$\widetilde{H}_{1,1}(\xi) = \frac{1}{2} + \frac{z}{2}.$$

(ii) Für $\widetilde{\kappa} = 3$ ist

$$\widetilde{H}_{1,3}(\xi) = -\frac{z^{-2}}{16} + \frac{z^{-1}}{16} + \frac{1}{2} + \frac{z}{2} + \frac{z^2}{16} - \frac{z^3}{16}.$$

(2) Sei

$$H(\xi) = H_2(\xi) = \frac{1}{4}\left(z^{-1} + 2z\right).$$

Dann erhalten wir folgende Darstellungen für $\widetilde{H}_{2,\widetilde{\kappa}}$.

(i) Für $\widetilde{\kappa} = 2$ ist

$$\widetilde{H}_{2,2}(\xi) = \frac{1}{4}\left(-\frac{1}{2}z^{-2} + z^{-1} + 3 + z - \frac{1}{2}z^2\right).$$

(ii) Für $\widetilde{\kappa} = 4$ ist

$$\widetilde{H}_{2,4} = \frac{3}{128}z^{-4} - \frac{3}{64}z^{-3} - \frac{1}{8}z^{-2} + \frac{19}{64}z^{-1} + \frac{45}{64} + \frac{19}{64}z - \frac{1}{8}z^2 - \frac{3}{64}z^3 + \frac{3}{128}z^4.$$

(3) Sei

$$H(\xi) = H_3(\xi) = \frac{1}{8}\left(z^{-1} + 3 + 3z + z^2\right).$$

(i) Für $\widetilde{\kappa} = 1$ ist

$$\widetilde{H}_{3,1}(\xi) = -\frac{1}{4}z^{-1} + \frac{3}{4} + \frac{3}{4}z - \frac{1}{4}z^2.$$

(ii) Für $\widetilde{\kappa} = 3$ ist

$$\widetilde{H}_{3,3}(\xi) = \frac{3}{64}z^{-3} - \frac{9}{64}z^{-2} - \frac{7}{64}z^{-1} + \frac{45}{64} + \frac{45}{64}z - \frac{7}{64}z^2 - \frac{9}{64}z^3 + \frac{3}{64}z^4.$$

Wir bemerken, dass die Koeffizienten der biorthogonalen Spline-Wavelets stets symmetrisch sind. Aus den obigen Darstellungen erhalten wir die Verfeinerungsfolgen h_k bzw. $\widetilde{h}_k$, indem wir die Koeffizienten von z^k mit der Normierungskonstante $\sqrt{2}$ multiplizieren.
Wir verwenden in unseren Algorithmen die biorthogonalen Spline-Wavelets mit $\kappa, \widetilde{\kappa} \in 2\mathbb{N} + 1$.

5 Radiale Basisfunktionen

Wir führen in diesem Abschnitt radiale Basisfunktionen ein, welche wir später zur Approximation von Flächen verwenden. Wir beschäftigen uns mit der Frage, was radiale Basisfunktionen sind und erläutern einige ihrer wichtigen Eigenschaften. Wir machen zunächst einige allgemeine Aussagen über radiale Basisfunktionen und gehen dann kurz auf einige wichtige Arten genauer ein.

5.1 Definition und Eigenschaften

5.1 Definition: Sei $\phi \in \mathcal{C}(\mathbb{R}_+)$ mit $\phi : \mathbb{R}_+ \to \mathbb{R}$. Dann nennen wir $\phi(\|\cdot\|_2)$ *radiale Basisfunktion*.
Sei $X = \{\mathbf{x}_0, \ldots, \mathbf{x}_{N-1}\} \subset \mathbb{R}^d$ eine endliche Menge und ϕ eine radiale Basisfunktion. Wir bezeichnen den Raum

$$S_\phi(X) = \text{span}\{\phi(\|\cdot - \mathbf{x}_j\|_2) \mid j = 0, \ldots, N-1\}$$

als *Raum der radialen Basisfunktion* ϕ. Die Punkte $\mathbf{x}_j \in X$ nennen wir *Zentren* der radialen Basisfunktion ϕ. Für ein $s \in S_\phi(X)$ gilt dann

$$s(\mathbf{x}) = \sum_{j=0}^{N-1} \lambda^{(j)} \phi(\|\mathbf{x} - \mathbf{x}_j\|_2) \text{ für } \mathbf{x} \in \mathbb{R}^d.$$

Dabei sind die Parameter $\lambda^{(j)} \in \mathbb{R}$ für $j = 0, \ldots, N-1$ die *Gewichte* der radialen Basisfunktion ϕ.

Die Funktionen $\phi(\|\cdot\|_2)$ werden als radial bezeichnet, da diese für alle Punkte $\mathbf{x} \in \mathbb{R}^d$ konstant sind, für die $\|\mathbf{x}\|_2 = r$ für ein $r > 0$ gilt.

Die häufigste Anwendung der radialen Basisfunktionen ist die Interpolation oder Approximation an ungeordneten Punkten (*scattered data*) im $\mathbb{R}^d$. Sei hierfür $K = \{\mathbf{v}_j \mid j = 0, \ldots, N-1\} \subset \mathbb{R}^d$ eine Menge von Punkten und seien $f_0, \ldots, f_{N-1} \in \mathbb{R}$, die wir als Funktionswerte bezeichnen. Wir suchen eine Funktion $s \in S_\phi(K)$ für eine beliebige oder vorgegebene radiale Basisfunktion ϕ, für die gilt

$$s(\mathbf{v}_k) = \sum_{j=0}^{N-1} \lambda^{(j)} \phi(\|\mathbf{v}_k - \mathbf{v}_j\|_2) = f_k \text{ für } k = 0, \ldots, N-1. \tag{5.1}$$

Mit anderen Worten, wir suchen die Lösung des linearen Gleichungssystems

$$\mathbf{M}\boldsymbol{\lambda} = \mathbf{f} \tag{5.2}$$

mit

$$\mathbf{M} = \begin{bmatrix} \phi(\|\mathbf{v}_0 - \mathbf{v}_0\|_2) & \dots & \phi(\|\mathbf{v}_0 - \mathbf{v}_{N-1}\|_2) \\ \vdots & \ddots & \vdots \\ \phi(\|\mathbf{v}_{N-1} - \mathbf{v}_0\|_2) & \dots & \phi(\|\mathbf{v}_{N-1} - \mathbf{v}_{N-1}\|_2) \end{bmatrix} \in \mathbb{R}^{N\times N},$$

$$\boldsymbol{\lambda} = \begin{bmatrix} \lambda^{(0)} \\ \vdots \\ \lambda^{(N-1)} \end{bmatrix} \text{ und } \mathbf{f} = \begin{bmatrix} f_0 \\ \vdots \\ f_{N-1} \end{bmatrix}.$$

Welche Eigenschaften hat die Matrix $\mathbf{M}$? Zunächst einmal ist die Matrix symmetrisch, denn es gilt

$$\phi(\|\mathbf{v}_k - \mathbf{v}_j\|_2) = \phi(\|\mathbf{v}_j - \mathbf{v}_k\|_2)$$

für alle $k, j = 0, \dots, N-1$. Viel wichtiger wäre die Frage nach der Singularität von $\mathbf{M}$. Leider lässt sich diese Frage nicht pauschal für alle Typen der radialen Basisfunktionen beantworten. Schränken wir die Auswahl etwas ein, so sind durchaus Aussagen über $\mathbf{M}$ möglich. Wir benötigen allerdings das Konzept der vollständig monotonen Funktionen.

5.2 Definition: Wir nennen eine Funktion $f \in \mathcal{C}^\infty(\mathbb{R}_+)$ *vollständig monoton*, falls für alle $k \in \mathbb{N}_0$ gilt

$$(-1)^k f^{(k)}(t) \geq 0 \text{ für } t \geq 0. \tag{5.3}$$

Nach dieser Definition sind zum Beispiel die Funktionen $f(t) = e^{-\alpha t}$ für $\alpha > 0$ oder $f(t) = (t + c^2)^{-1/2}$ vollständig monoton.
Für vollständig monotone Funktionen können wir die folgenden Aussagen zeigen. Das erste Resultat geht auf Schoenberg zurück, wenngleich es ursprünglich für andere Zwecke bewiesen wurde. Wir entnehmen diese Aussage samt Beweis [Bu03].

5.3 Satz: Sei $f \in \mathcal{C}(\mathbb{R}_+)$ mit $f : \mathbb{R}_+ \to \mathbb{R}$ eine vollständig monotone Funktion. Dann ist die Matrix $\mathbf{M}$ des Interpolationsproblems (5.1) für alle endlichen Teilmengen $K \subset \mathbb{R}^d$ und alle $d > 0$ positiv definit, falls $f(r) = \phi(r^2)$ gilt. Insbesondere ist $\mathbf{M}$ nicht singulär.

Nach Satz 5.3 ist die Matrix $\mathbf{M}$ für

$$\phi(r) = \left(r^2 + c^2\right)^{-1/2} \text{ mit } c \in \mathbb{R}$$

für alle $K \subset \mathbb{R}^d$ und alle $d > 0$ positiv definit, also auch nicht singulär. Analog ist $\mathbf{M}$ nicht singulär für

$$\phi(r) = \left(r^2 + c^2\right)^{1/2} \text{ mit } c \in \mathbb{R}.$$

Das nächste Resultat geht auf Micchelli zurück und erweitert den Kreis der Funktionen ϕ, die in Gleichung (5.1) eine nicht singuläre Matrix $\mathbf{M}$ erzeugen. Die Aussage samt Beweis finden wir in [Bu03].

5.4 Satz: Sei $f \in \mathcal{C}(\mathbb{R}_+)$ mit $f(0) \geq 0$. Sei weiter f' nicht konstant und vollständig monoton. Dann ist die Matrix $\mathbf{M}$ des Interpolationsproblems (5.1) nicht singulär für $f(r) = \phi(r^2)$.

Wir können die obigen Aussagen nicht auf den Thin-Plate Spline $\phi(r) = r^2 \log r$ anwenden. Allerdings ist dies für uns nicht essentiell, da wir eine etwas veränderte Funktion s verwenden, die zusätzlich zu der Linearkombination der radialen Basisfunktionen einen polynomialen Anteil besitzt. Wir betrachten nun einige der gängigsten radialen Basisfunktionen etwas genauer und beginnen mit dem bereits erwähnten Thin-Plate Spline.

5.2 Thin-Plate Spline

Der Thin-Plate Spline wurde 1975 von *Jean Duchon* vorgestellt und hat die Gestalt

$$\phi_{TPS}(r) = r^2 \log r \text{ mit } r = \|\mathbf{x}\|_2 \text{ für } \mathbf{x} \in \mathbb{R}^{d-1}. \tag{5.4}$$

Es gilt per Definition $\phi_{TPS}(0) = 0$. Der Thin-Plate Spline ist die Fundamentallösung der so genannten *Biharmonischen Gleichung*

$$\Delta^2 \phi = 0. \tag{5.5}$$

Dabei ist Δ der Laplace-Operator der folgenden Gestalt

$$\Delta\phi(\mathbf{x}) = \sum_{j=1}^{d} \frac{\partial^2}{\partial \left(x^{(j)}\right)^2} \phi(\mathbf{x})$$

für $\mathbf{x} \in \mathbb{R}^d$. Die biharmonische Gleichung hat, wie fast alle Differentialgleichungen, eine physikalische Bedeutung. Die Lösung der Gleichung

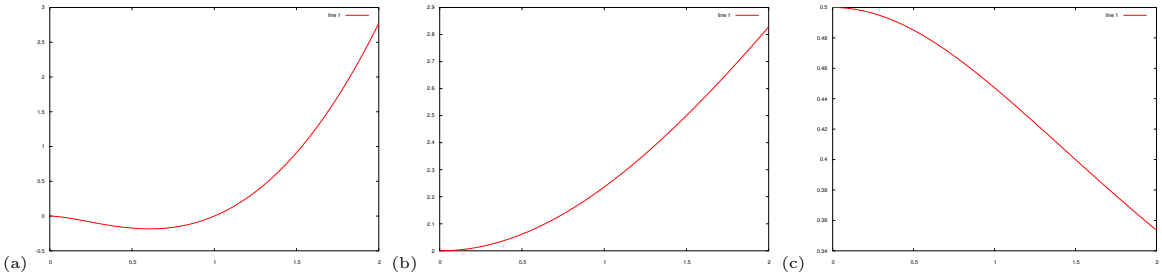

Abbildung 10: (a) Thin-Plate Spline (b) Multiquadrik mit $c = 2$ (c) Inverse Multiquadrik mit $c = 2$

repräsentiert die Näherung der Form, die ein dünnes Metallplättchen einnehmen wird, wenn es durch verschiedene feste Punkte gezwungen wird. Den Verlauf der Funktion sehen wir in Abbildung 10 (a).
Es gibt weitere, allgemeinere Formen der Thin-Plate Splines. Sei hierfür d die Dimension des Raums $\mathbb{R}^d$, dann gilt

$$\phi_k^{(d)}(r) = \begin{cases} r^{2k-d} \log r & \text{für } k \geq d \text{ und } d \text{ ungerade} \\ r^{2k-d} & \text{für } k \geq d \text{ und } d \text{ gerade.} \end{cases}$$

Eine auf diese Weise definierte Funktion ist die Fundamentallösung der Gleichung $\Delta^k \phi = 0$ in d Dimensionen.

5.3 Multiquadrik und Inverse Multiquadrik

Die Funktionen, die in der Literatur *Multiquadrik* genannt werden, gehen auf *R. L. Hardy* zurück und werden häufig zur Lösung partieller Differentialgleichungen verwendet.
Sei $c \in \mathbb{R}$. Die Funktion

$$\phi_{MQ}(r) = \sqrt{r^2 + c^2} \tag{5.6}$$

nennen wir *Multiquadrik*. Entsprechend wird in der Literatur die Umkehrfunktion von ϕ_{MQ}

$$\phi_{IMQ}(r) = \frac{1}{\sqrt{r^2 + c^2}} \tag{5.7}$$

für $c \in \mathbb{R}$ als *Inverse Multiquadrik* bezeichnet. Wir sehen beide Funktionen in den Abbildungen 10 (b) und (c) abgebildet.

5.4 Weitere radiale Basisfunktionen

Als weiterer Typ radialer Basisfunktionen wäre zum Beispiel die Gaußfunktion zu nennen. Diese radiale Basisfunktion ist definiert durch die Gleichung

$$\phi_G(r) = e^{-\frac{r^2}{2\sigma^2}}$$

für $\sigma > 0$. Sie zeichnet sich durch ihre Symmetrie aus und man kann ϕ_G im mehrdimensionalen Fall zerlegen.
Häufig werden im Bereich der Flächeninterpolation und -approximation die Funktionen

$$\phi_{BH}(r) = r \text{ und } \phi_{TH}(r) = r^3$$

benutzt. Dabei nennen wir ϕ_{BH} den *biharmonischen Spline* und ϕ_{TH} den *triharmonischen Spline*.
Weitere Funktionen, die als radiale Basisfunktionen Verwendung finden, sind radiale Basisfunktionen mit kompakten Trägern nach *Wendland* und *Wu*, radiale Basisfunktionen, die auf der Besselfunktion basieren, und viele andere. Für weitere Details verweisen wir auf [Bu03].

5.5 Anwendungsgebiete für radiale Basisfunktionen

Das klassische Anwendungsgebiet für radiale Basisfunktionen ist der Bereich der Flächeninterpolation und -approximation. Mit diesem Thema beschäftigen wir uns ausgiebig in Abschnitt 10.
Weitere Anwendungsgebiete sind zum Beispiel die Lösung partieller Differentialgleichungen mit Hilfe der radialen Basisfunktionen, Konstruktion von Wavelets und Waveletbasen und nicht zuletzt neuronale Netze mit radialen Basisfunktionen.
Eine tiefergehende Beschreibung der Anwendungsgebiete radialer Basisfunktionen finden wir in [Bu03], für eine Einführung in die neuronalen Netze verweisen wir auf [Po89].

Teil II
Anwendungen

Von nun an beschäftigen wir uns mit Anwendungen auf dem Gebiet der Koordinatenmesstechnik. Diesen Teil der Fertigungsmesstechnik stellen wir im ersten Abschnitt vor.

6 Grundlagen der Koordinatenmesstechnik

Wie bereits erwähnt, handelt es sich bei der *Koordinatenmesstechnik* um ein Teilgebiet der *Fertigungsmesstechnik*. Bis vor wenigen Jahren wurde die Koordinatenmesstechnik von Messmikroskopen und Messprojektoren dominiert. In den siebziger Jahren wurden Dreikoordinatenmessgeräte entwickelt, bei denen das Werkstück mit einem *Tastelement* (*Taster*) manuell abgetastet wurde. Eine automatische Messung war aber noch nicht möglich. Erst Ende der achtziger Jahre wurden durch die Entwicklung von Bildverarbeitungsverfahren und Lasersensoren die Voraussetzungen für neue Messverfahren geschaffen. Es entstanden das Gebiet der *Multisensor-Koordinatenmesstechnik* und die entsprechenden Messgeräte, welche sowohl über berührende als auch über optoelektronische Sensoren verfügten.
Ein Koordinatenmessgerät der heutigen Zeit erfasst die Koordinatenwerte eines jeden Ortes innerhalb des üblicherweise quaderförmigen Messbereichs, bezogen auf ein orthogonales dreidimensionales Referenzkoordinatensystem. Allgemein unterscheidet man zwischen schaltenden und messenden Systemen. Während schaltende Systeme keinen automatischen Betrieb zulassen, arbeiten messende Systeme nach einer Kalibrierung weitgehend selbständig.
Beispiele für Koordinatenmessgeräte sehen wir in Abbildung 11.
Grundsätzlich gibt es verschiedene Ansätze, wie ein Messgerät die gemessenen Daten verarbeitet. In den meisten Fällen bildet die Maschine nicht die Wirklichkeit ab, sondern vergleicht die Messdaten mit einem so genannten *Sollelement*. Bei einem *Sollelement* handelt es sich um ein geometrisch ideales Element wie einen Kreis, eine Gerade oder im Dreidimensionalen um eine Kugel, einen Kegel oder einen Zylinder. Dies führt zu den Problemen der Berechnung von Ausgleichselementen oder minimalen und maximalen Elementen. Wir beschäftigen uns mit solchen Berechnungen in Abschnitt 8.

(a)  (b) 

Abbildung 11: (a) Multisensor-Koordinatenmessgerät *Video Check*® IP der Fa. *Werth Messtechnik GmbH* sowohl für optische als auch für taktile Messung (b) Taktiles Koordinatenmessgerät *Probe Check*® der Fa. *Werth Messtechnik GmbH*

In manchen Fällen besteht die Notwendigkeit, die gemessenen Punkte in der ursprünglichen Form weiter zu verarbeiten, weil diese zum Beispiel keinem Sollelement entsprechen. Wir unterscheiden bei den Messdaten zwischen *Konturen*, also Polygonzügen im $\mathbb{R}^3$, und *Flächen*. Mit der Behandlung von Konturen beschäftigen sich die Abschnitte 7 und 9, mit Flächen der Abschnitt 10.
Die Haupteinsatzgebiete der Koordinatenmessgeräte sind die folgenden.

- *Messen*
  Bei Messungen werden Parameter für das Einrichten und Korrigieren von Werkzeugmaschinen vor Beginn der Fertigung an so genannten Einstell- oder Erstschnittmusterwerkstücken ermittelt. Diese Parameter dienen in der heutigen Zeit der Korrektur von *CNC-Programmen* (Computerized Numerical Control).

- *Prüfen*
  Es werden die tatsächlich hergestellten Werkstücke hinsichtlich aller konstruktiv gegebenen Gestaltsmerkmale geprüft. Die geometrisch ideale Gestalt, also die Sollgestalt, mit der man die Istgestalt vergleicht, wird in der Regel durch CAD-Systeme festgelegt.

- *Überwachen*
  Bei der Überwachung des Fertigungsprozesses in der Serienproduktion werden vorzugsweise Werkzeuge, Vorrichtungen und Maschinen oder auch die Werkstücke selbst gemessen, mit dem Ziel, aus den Messergebnissen Prozess- und Korrekturparameter zu gewinnen, mit deren Hilfe der Fertigungsprozess beurteilt und geregelt werden kann. Solche Parameter können zum Beispiel einen Werkzeugwechsel anzeigen

oder Korrekturdaten liefern, die zur Kompensation der auf den Prozess einwirkenden Störgrößen verwendet werden.

Multisensor-Koordinatenmessgeräte werden in vielen Wirtschaftszweigen verwendet. Die Schwerpunkte sind die Kraftfahrzeug-Zuliefererindustrie, der Werkzeugbau und die Konsumgüterindustrie. Nachfolgend listen wir einige Hauptanwendungsfelder auf.

- Kunststoffspritzguss
  Das Spritzgussverfahren ermöglicht es, sehr komplexe Kunstoffteile in guter Qualität preisgünstig herzustellen. Als Beispiele seien hier Funktions- und Gehäuseteile für Fahrzeugkomponenten, elektronische und medizintechnische Geräte und Konsumgüter genannt.
  Ein Schwerpunkt beim Einsatz von Messgeräten in der Kunstoffverarbeitung ist die Endkontrolle der Produkte. Überprüft werden Funktionsmaße mit Toleranzen von wenigen 10 μm und darunter.

- Blechbiegeteile und Stanzwerkzeuge
  Die Herstellung von Blechbiegeteilen ist sowohl in der Elektroindustrie als auch im Fahrzeugbereich weit verbreitet. Ein zweidimensionales Teil mit Ausschnitten ist der klassische Einsatzfall für optische Koordinatenmessgeräte.
  Mit modernen Stanz- und Biegeverfahren und Mehrfachwerkzeugen werden heute auch komplexe dreidimensionale Teile erzeugt, welche nur mit Hilfe der Multisensorik, also einer Kombination aus optischen und taktilen Messverfahren, zufriedenstellend zu prüfen sind.

- Profilwerkstücke
  Profilwerkstücke aus Kunststoff, Gummi oder Aluminium werden beispielsweise im Fahrzeugbau (Aluminium-Spaceframe), im Bauwesen (Fensterprofile) und in der Automatisierungstechnik (Bandprofile) benötigt.
  Die Herstellzeit für eine Charge beträgt in der Regel nur wenige Stunden, in dieser Zeit werden große Stückzahlen produziert. Es ist erforderlich, die Qualität innerhalb kürzester Zeit zu überprüfen, daher kommen manuelle Verfahren nicht in Frage. Die Lösung ist in diesem Fall ein messendes Koordinatenmesssystem, wobei meistens optische Geräte zum Einsatz kommen.

- Weitere Einsatzfelder sind beispielsweise die messtechnische Erfassung von spanabhebenden Werkzeugen, das Vermessen von rotationssym-

metrischen Teilen wie beispielsweise Wellen und Prüfungen von Bauteilen mit Mikrogeometrien.

Da die Koordinatenmesstechnik ein sehr anwendungsspezifisches Gebiet ist, entwickelte sich eine eigene Terminologie. Wir führen im nächsten Abschnitt die für uns relevanten Begriffe ein und verweisen für Einzelheiten auf die weiterführende Literatur. Die Grundlagen der Koordinatenmesstechnik finden wir zum Beispiel in [We99] oder in [ChNe03]. Weitere Anwendungsgebiete der Koordinatenmessgeräte behandelt das Buch [Neu00]. Eine Bibliographie mit mehr als 2200 Literaturverweisen zum Thema Koordinatenmesstechnik stellt [Gal03] vor.
Bevor wir uns den ersten Anwendungen zuwenden, beschäftigen wir uns kurz mit der Genauigkeit und der Zertifizierung der Messgeräte der Fa. *Werth Messtechnik GmbH*. Das Kalibrierlabor des Unternehmens ist das erste nach *ISO 17025* akkreditierte DKD-Kalibrierlabor für optische und Multisensor- Koordinatenmessgeräte in Deutschland.
Diese Akkreditierung berechtigt dazu, im Namen des Deutschen Kalibrierdienstes Annahmetests nach *ISO 10360-2* und nach *VDI/VDE 2617* Blatt 6 durchzuführen. Hierdurch erfüllt die *Werth Messtechnik GmbH* die aktuellen Anforderungen an alle nach dem Qualitätsmanagementsystem der *ISO/TS 16949* arbeitenden Unternehmen, dass sämtliche Messmittel durch Laboratorien überprüft bzw. kalibriert werden müssen, die nach *DIN EN ISO/IEC 17025* akkreditiert sind.
Der 3D-Annahmetest nach *ISO 10360-2* gilt für die Annahmeprüfung von taktilen Koordinatenmessgeräten und der taktilen Funktionen von Multisensorgeräten. Die 2D-/3D-Annahmeprüfung nach *VDI 2617-6* gilt für optische Koordinatenmessgeräte. Die Annahmetests überprüfen, ob das Gerät die vorgegebene Spezifikation einhält. Die entsprechenden Vorgaben werden im Normalfall vom Kunden gemacht.
Allgemein sind Genauigkeiten der für uns relevanten Messgeräte in einem Bereich von 10^{-1} bis 10^{-8} mm einstellbar, wir haben alle unseren Testkonturen mit einer Genauigkeit von 10^{-6} mm abgetastet.

7 Konturmathematik

7.1 Problemstellung

Zu den Standardmessanforderungen, welche die Messsoftware bewältigen muss, gehören neben den Anpassungen auch die Bestimmung von Schnitt-

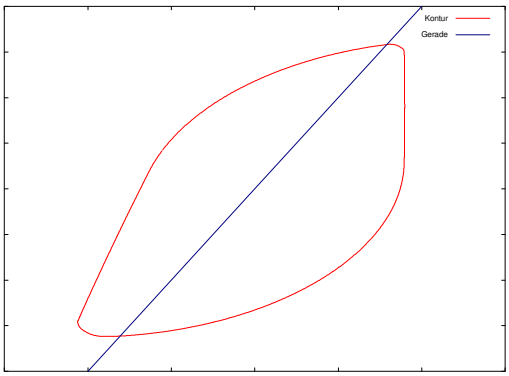

Abbildung 12: Beispiel einer Verknüpfung Kontur-Gerade. Es sollen der minimale Punkt sowie der minimale Abstand der Kontur und der Gerade berechnet werden.

punkten und Distanzen von Konturen und Regelgeometrieobjekten wie Punkt, Gerade, Ebene, Kreis oder anderen Konturen. Unter einer Kontur verstehen wir in diesem Kontext einen Polygonzug im $\mathbb{R}^d$.
Die Aufgabe bestand also darin, einen Algorithmus zu entwickeln, der für verschiedene Regelgeometrieelemente den minimalen bzw. maximalen Punkt samt entsprechender Distanzen zu einer Kontur berechnet. Wegen der Genauigkeit der Bestimmung war die erste Anforderung an den Algorithmus, dass es sich um eine Interpolation der vorhandenen Daten handeln muss. Da allerdings nicht selten Datenmengen von mehr als 10.000 Datenpunkten vorkommen, musste die Berechnung trotzdem schnell sein, damit ein flüssiger Ablauf der Messsoftware *WinWerth*® möglich war. Darüber hinaus musste der Interpolationsalgorithmus die Möglichkeit bieten, Ecken zu modellieren, also war es auch notwendig, solche Ecken im Vorfeld zu erkennen. Diese Bedingungen führten letztlich zu der Entscheidung, Splines als interpolierende Funktionen zu benutzen.
Ein typisches Beispiel für eine Messaufgabe sehen wir in Abbildung 12.
Die vorliegende Kontur können wir nicht ohne große Messabweichungen durch eine Anpassung in ein Regelgeometrieelement überführen, daher muss der Abstand zwischen der Gerade und der Kontur direkt berechnet werden. Bei der Kontur handelt es sich um ein Stanzteil, optisch abgetastet mit 1.714 Punkten.
Zu Beginn erklären wir einige Begriffe aus der Koordinatenmesstechnik und definieren einen minimalen bzw. maximalen Punkt.

7.1 Bezeichnung: Unter einem *Regelgeometrieelement* verstehen wir eines der folgenden Objekte im $\mathbb{R}^d$.

- *Gerade* $\mathbf{R}_G(\mu) = \mathbf{p} + \mu\mathbf{q} \subset \mathbb{R}^d$ mit $\mathbf{p}, \mathbf{q} \in \mathbb{R}^d$, $\mu \in \mathbb{R}$. Den Punkt $\mathbf{p}$ bezeichnen wir als *Schwerpunkt* von $\mathbf{R}_G$, $\mathbf{q}$ wird *Richtungsvektor* genannt. Wir unterscheiden zwischen der *positiven Richtung* ($\mu \geq 0$) und der *negativen* Richtung ($\mu < 0$) der Geraden $\mathbf{R}_G$.

- *Kreis* $\mathbf{R}_C(\mu) = \mathbf{p}_M + r_C\mathbf{P}(\mu) \subset \mathbb{R}^d$ mit $\mu \in [0, 2\pi)$. Dabei ist $\mathbf{p}_M$ der *Mittelpunkt des Kreises*, $r_C > 0$ der *Radius*. $\mathbf{R}_C$ ist die Menge aller Randpunkte des Kreises.

- *Ebene* $\mathbf{R}_E(\boldsymbol{\mu}_E) = \mathbf{p} + \mu^{(1)}\mathbf{q} + \mu^{(2)}\mathbf{r} \subset \mathbb{R}^d$ mit $\mathbf{p}, \mathbf{q}, \mathbf{r} \in \mathbb{R}^d$ und $\boldsymbol{\mu}_E = [\mu^{(1)},\ \mu^{(2)}]^t \in \mathbb{R}^2$. Ähnlich wie bei der Gerade bezeichnen wir $\mathbf{p}$ als *Schwerpunkt der Ebene*, $\mathbf{q}$ und $\mathbf{r}$ nennen wir *Richtungsvektoren*.

- *Kontur* $K = \{\mathbf{v}_j \mid j = 0, \ldots, N-1\} \subset \mathbb{R}^d$. Sie ist ebenfalls ein Regelgeometrieelement, das aber gesondert behandelt wird. Zunächst liegt die Kontur K stets geordnet vor. Da man für eine Kontur $K \subset \mathbb{R}^d$ in der Regel keine geschlossene Formel angeben kann, betrachten wir stattdessen eine Abbildung

  $$\mathbf{K} : \mathbb{R} \to \mathbb{R}^d \text{ mit } \mathbf{K} : \mu \mapsto \mathbf{K}(\mu) \in \mathbb{R}^d,$$

  für die gilt:
  Es existierten $\mu_0, \ldots, \mu_{N-1} \in \mathbb{R}$ mit

  $$\mathbf{K}(\mu_j) = \mathbf{v}_j \text{ für } j = 0, \ldots, N-1.$$

  Wir sagen $\mathbf{K}$ interpoliert die Kontur K.

- *Punkt* $\mathbf{R}_P = \mathbf{p} \in \mathbb{R}^d$.

- *Kugel, Zylinder und Kegel* sind in der Koordinatenmesstechnik ebenfalls Regelgeometrieelemente, werden in unseren Betrachtungen aber nicht behandelt.

Eine Kontur ist für uns eine Teilmenge des $\mathbb{R}^d$, ein Regelgeometrieelement ist eine Abbildung in den Raum $\mathbb{R}^d$. Insbesondere ist der Wertebereich ebenfalls eine Teilmenge des $\mathbb{R}^d$. Was verstehen wir unter dem Begriff minimaler Punkt? Wir verwenden die Sprechweise der Koordinatenmesstechnik, beschrieben zum Beispiel in [We99] oder [WM06] und erhalten die folgende Definition.

7.2 Definition: Sei $K = \{\mathbf{v}_j \mid j = 0, \ldots, N-1\} \subset \mathbb{R}^d$ eine Kontur und $\mathbf{R}(\boldsymbol{\mu}) \subset \mathbb{R}^d$ mit $\boldsymbol{\mu} \in \mathbb{R}^z$, $z \geq 0$ ein Regelgeometrieelement. Sei $\mathbf{K}$ eine Funktion, welche K interpoliert, das heißt es existieren Stützstellen $a_0, \ldots, a_{N-1} \in \mathbb{R}$, für die gilt

$$\mathbf{K}(a_j) = \mathbf{v}_j \text{ für alle } j = 0, \ldots, N-1.$$

Wir nennen $\mathbf{x}_{min} \in \mathbb{R}^d$ *minimalen Punkt* der Kontur K und des Regelgeometrieelements $\mathbf{R}$, wenn gilt:
Es existiert ein $(a_*, \boldsymbol{\mu}_*) \in \mathbb{R}^{z+1}$ mit

$$\|\mathbf{R}(\boldsymbol{\mu}_*) - \mathbf{K}(a_*)\|_2 = \min_{(a,\boldsymbol{\mu}) \in \mathbb{R}^{z+1}} \|\mathbf{R}(\boldsymbol{\mu}) - \mathbf{K}(a)\|_2 \text{ und } \mathbf{x}_{min} = \mathbf{K}(a_*).$$

Wir setzen $d_* = \|\mathbf{R}(\boldsymbol{\mu}_*) - \mathbf{K}(a_*)\|_2$. Im Allgemeinen gilt $\mathbf{K}(a_*) \notin K$.
Einen *minimalen Schnittpunkt* erhalten wir, falls $d_* = 0$ gilt. Es ist offensichtlich, dass sowohl mehrere minimale Punkte als auch mehrere minimale Schnittpunkte existieren können.
Wir bezeichnen $\mathbf{x}_{max} \in \mathbb{R}^d$ als *maximalen Punkt* von K und $\mathbf{R}$, falls die folgende Bedingung erfüllt ist:
Es existiert ein $(a_*, \boldsymbol{\mu}_*) \in \mathbb{R}^{z+1}$ mit

$$\|\mathbf{R}(\boldsymbol{\mu}_*) - \mathbf{K}(a_*)\|_2 = \max_{(a,\boldsymbol{\mu}) \in \mathbb{R}^{z+1}} \|\mathbf{R}(\boldsymbol{\mu}) - \mathbf{K}(a)\|_2 \text{ und } \mathbf{x}_{max} = \mathbf{K}(a_*).$$

Wir wollen uns zunächst mit der Identifizierung der Ecken beschäftigen, danach die Splineinterpolation erläutern, die Berechnung der Startwerte für jede der obigen Verknüpfungen angeben und anschließend die Berechnung der minimalen und maximalen Punkte und Distanzen diskutieren.

Sei

$$K = \{\mathbf{v}_j \mid j = 0, \ldots, N-1\} \subset \mathbb{R}^d$$

eine Kontur. Wir wollen mögliche Ecken, also nicht differenzierbare Stellen, dieser Kontur bestimmen, damit bei einer Interpolation an diesen Stellen Ecken modelliert werden können.
Weshalb wir ein Verfahren zur Bestimmung von Ecken benötigen, können wir in Abbildung 13 beobachten. Ohne die Kenntnis der Lage der Ecken können wir unsere Knotenfolge nicht mit dreifachen Knoten versehen und die Splinekurve mindert die Spannung, indem sie entweder oszilliert oder Schlaufen bildet. Wie die Splinekurve an diesen Stellen aussieht, wenn wir einen Eckendetektionsalgorithmus verwenden, sehen wir in Abbildung 24

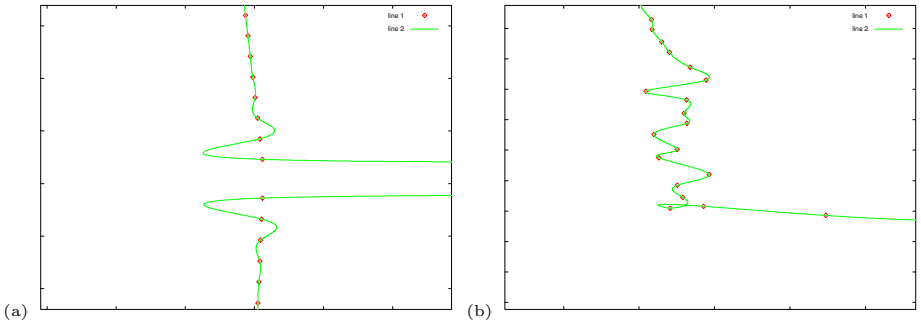

Abbildung 13: (a) Oszillierende Splinekurve an einer Ecke (b) Splinekurve mit einer Schlaufe an einer Ecke

auf Seite 98.
Wir stellen nachfolgend zwei Möglichkeit zur Erkennung der Ecken vor. Die erste benutzen wir zur groben Bestimmung der Ecken und verfeinern diese Suche mit Hilfe des zweiten Verfahrens.

7.2 Geometrische Eckenbestimmung

Für das erste Verfahren bedienen wir uns eines einfachen Hilfsmittels, nämlich des inneren Produktes zweier Vektoren im $\mathbb{R}^d$.

7.3 Definition: Seien $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$. Das *innere Produkt* von $\mathbf{x}$ und $\mathbf{y}$ wird berechnet als

$$\mathbf{x}^t\mathbf{y} = \sum_{j=1}^{d} x^{(j)}y^{(j)} = \|\mathbf{x}\|_2\|\mathbf{y}\|_2 \cos\alpha.$$

Hierbei gilt

$$\mathbf{x} = \begin{bmatrix} x^{(1)} \\ \vdots \\ x^{(d)} \end{bmatrix}, \mathbf{y} = \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(d)} \end{bmatrix}$$

und α ist der von den Vektoren $\mathbf{x}$ und $\mathbf{y}$ eingeschlossene Winkel. Dabei ist stets $\alpha \leq \pi$.

Wir definieren zwei Vektoren $\mathbf{b}$ und $\mathbf{c}$, die den Verlauf der Kontur wiedergeben.

Es gilt für

$$\mathbf{b} = \begin{bmatrix} b^{(1)} \\ \vdots \\ b^{(N-2)} \end{bmatrix} \in \mathbb{R}^{N-2} \text{ und } \mathbf{c} = \begin{bmatrix} c^{(1)} \\ \vdots \\ c^{(N-3)} \end{bmatrix} \in \mathbb{R}^{N-3} \text{ mit}$$

$$b^{(j)} = \cos\alpha_j = \frac{(\mathbf{v}_{j-1} - \mathbf{v}_j)^t(\mathbf{v}_{j+1} - \mathbf{v}_j)}{\|\mathbf{v}_{j-1} - \mathbf{v}_j\|_2 \|\mathbf{v}_{j+1} - \mathbf{v}_j\|_2} \text{ für } j = 1, \ldots, N-2 \qquad (7.1)$$

und

$$c^{(j)} = \cos\beta_j = \frac{(\mathbf{v}_{j-1} - \mathbf{v}_{j+1})^t(\mathbf{v}_{j+2} - \mathbf{v}_j)}{\|\mathbf{v}_{j-1} - \mathbf{v}_{j+1}\|_2 \|\mathbf{v}_{j+2} - \mathbf{v}_j\|_2} \text{ für } j = 1, \ldots, N-3. \qquad (7.2)$$

Die Größen α_j bzw. β_j, die nicht direkt berechnet werden, stellen die Winkel zwischen den Geraden

$$\mathbf{R}_G^{(j-1,1)}(x) = \mathbf{v}_{j-1} + x(\mathbf{v}_j - \mathbf{v}_{j-1}) \text{ und } \mathbf{R}_G^{(j,1)}(x) = \mathbf{v}_j + x(\mathbf{v}_{j+1} - \mathbf{v}_j)$$

bzw.

$$\mathbf{R}_G^{(j-1,2)}(x) = \mathbf{v}_{j-1} + x(\mathbf{v}_{j+1} - \mathbf{v}_{j-1}) \text{ und } \mathbf{R}_G^{(j,2)}(x) = \mathbf{v}_j + x(\mathbf{v}_{j+2} - \mathbf{v}_j)$$

dar. In den Komponenten von $\mathbf{b}$ finden wir die Cosinus-Werte der Winkel α_j wieder, analog sind in $\mathbf{c}$ die Cosinus-Werte der Winkel β_j gespeichert. Die Werte $b^{(j)}$ sind nichts anderes als die *symmetrischen zweiten Differenzen*. Die Einträge $c^{(j)}$ glätten die Kontur gewissermaßen. Falls wir bei $b^{(j)}$ eine Singularität erkennen, so finden wir entweder bei $c^{(j)}$ oder bei $c^{(j+1)}$ ebenfalls eine nicht singuläre Stelle.
Tasten wir mit hinreichender Genauigkeit ein gerades Werkstück ab, so erzeugen wir eine Kontur mit einem linearen Verlauf. In diesem Fall ist dann $b^{(j)} \approx 1$ und $c^{(j)} \approx 1$ für alle j, da $\cos\pi = 1$ gilt.
Für eine Kontur, die das Ergebnis einer Abtastung eines bogenförmigen Werkstückes darstellt, haben wir nicht notwendig $b^{(j)} \approx 1$ oder $c^{(j)} \approx 1$. Statt dessen können wir das Verhalten von $\left|b^{(j)} - b^{(j+1)}\right|$ bzw. $\left|c^{(j)} - c^{(j+1)}\right|$ beobachten. Bei einem ideal kreisförmigen Verlauf mit chordaler Abtastung wären die Beträge konstant, bei einem bogenförmigen Verlauf liegen sie innerhalb bestimmter Grenzen. Mit Hilfe dieser Kriterien finden wir bei der in Abbildung 14(a) dargestellten bogenförmigen Kontur keine Ecken.
Liegt uns eine Kontur wie in Abbildung 14(b) vor, so können wir nicht erwarten, dass wir anhand der Werte $b^{(j)}$ Ecken finden, denn die Daten sind

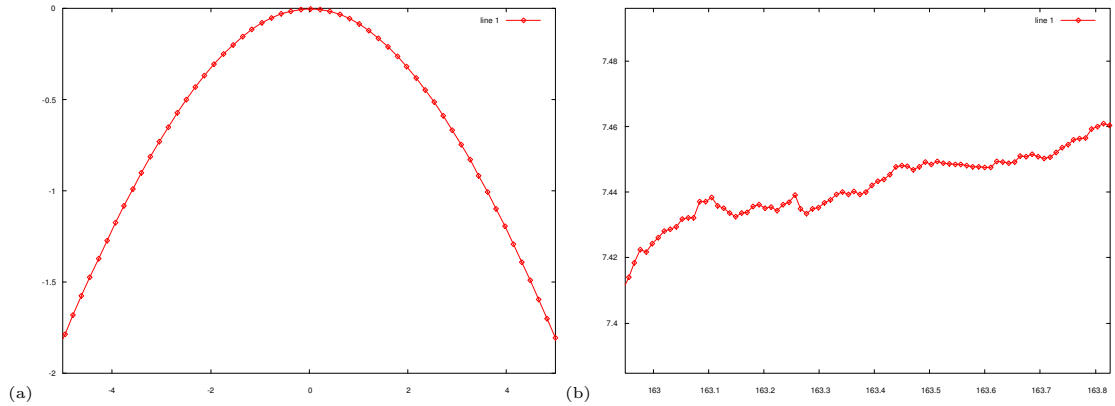

Abbildung 14: (a) Eine glatte Kontur ohne Ecken (b) Eine verrauschte Kontur ohne Ecken

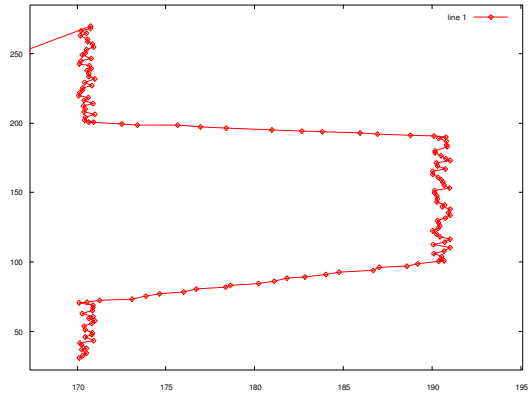

Abbildung 15: Eine verrauschte Kontur mit Ecken.

verrauscht. In diesem Fall bilden wir eine Teilmenge $J \subset \{0, \ldots, N-1\}$, wobei gilt

$$k \in J \;\Leftrightarrow\; b^{(k)} \ll 1.$$

Für die Indizes der Teilmenge J beobachten wir das Verhalten der Einträge in $\mathbf{c}$. Gilt $c^{(k)} \ll 1$ oder $c^{(k+1)} \ll 1$ für ein $k \in J$, so liegt für uns ein Indiz für eine Ecke vor. Auf diese Weise finden wir in der in Abbildung 14(b) dargestellten Kontur ebenfalls keine einzige Ecke. Eine solche Vorgehensweise funktioniert natürlich nur dann, wenn das Rauschen im Vergleich zu den Daten klein ist. Die folgende Abbildung 15 zeigt uns eine Kontur, die vier größere Veränderung des Konturverlaufs hat. In einem solchen Fall möchten wir die vier angesprochenen Verlaufsänderungen als Ecken identifizieren, während die kleineren Störungen ignoriert werden sollen. Wir erstellen den Algorithmus bzw. die Konstanten derart, dass in der Liste der $b^{(j)}$ mehr Ecken erkannt werden, während die Liste $c^{(j)}$ lediglich die vier Ecken identifiziert.

7.4 Algorithmus: Sei $K = \{\mathbf{v}_j \mid j = 0, \ldots, N-1\} \subset \mathbb{R}^d$ die zu untersuchende Kontur.

(1) Wir bestimmen $\mathbf{b} \in \mathbb{R}^{N-2}$ und $\mathbf{c} \in \mathbb{R}^{N-3}$ gemäß (7.1) und (7.2).

(2) Für die Auswertung berechnen wir den Mittelwert und die Standardabweichung der Komponenten von $\mathbf{b}$ und $\mathbf{c}$

$$\overline{b} = \frac{1}{N-2}\sum_{j=1}^{N-2} b^{(j)} \text{ und } \sigma_b = \sqrt{\frac{1}{N-2}\sum_{j=1}^{N-2}(b^{(j)} - \overline{b})^2}$$

sowie

$$\overline{c} = \frac{1}{N-3}\sum_{j=1}^{N-3} c^{(j)} \text{ und } \sigma_c = \sqrt{\frac{1}{N-3}\sum_{j=1}^{N-3}(c^{(j)} - \overline{c})^2}.$$

(3) Jetzt können wir die Menge E_1 bestimmen, die sämtliche Ecken der Kontur als Teilmenge enthält. Wir setzen

$$\begin{aligned} E_1 = \{\mathbf{v}_j \in K \mid \Big[& (1.0 - b^{(j)} \leq d_1\sigma_b) \text{ und} \\ & \Big((1.0 - c^{(j)} \leq d_1\sigma_c) \text{ oder } (1.0 - c^{(j+1)} \leq d_1\sigma_c)\Big)\Big] \text{ oder} \\ & \Big[(\Big|b^{(j)} - b^{(j+1)}\Big| \geq d_2\sigma_b) \text{ und} \\ & \Big((\Big|c^{(j)} - c^{(j+1)}\Big| \geq d_2\sigma_c) \text{ oder } (\Big|c^{(j+1)} - c^{(j+2)}\Big| \geq d_2\sigma_c)\Big)\Big]\}. \end{aligned} \tag{7.3}$$

Dabei sind die Werte $d_1, d_2 \in \mathbb{R}$ Faktoren, die durch statistische Auswertungen gewonnen werden. Wir haben für die Werte $d_1 = 2.5$ und $d_2 = 2.0$ die besten Ergebnisse erzielt.

7.5 Bemerkung: Wir haben die Faktoren d_1 und d_2 in Algorithmus 7.4 so gewählt, dass in der Regel mehr Kandidaten für Ecken gefunden werden als tatsächlich Ecken vorhanden sind. Die Anzahl wird mit Hilfe des zweiten Verfahrens korrigiert.

Bevor wir uns der Bestimmung der Ecken mit Hilfe der Waveletzerlegung zuwenden, gehen wir auf die Laufzeit des obigen Algorithmus ein.

(1) Für die Berechnung von $\mathbf{b}$ bzw. $\mathbf{c}$ benötigen wir $(10d+2)(N-2)$ bzw. $(10d+2)(N-3)$ Operationen.

(2) Die Bestimmung der Mittelwerte und der mittleren Streuung erfordert insgesamt $6N - 3$ Operationen.

Punkt (3) beinhaltet ausschließlich Vergleichsoperationen, wir schätzen den Aufwand mit $O(N)$ nach oben ab.
Daher beträgt der Gesamtaufwand für die Bestimmung der Ecken mit Hilfe des inneren Produktes $N(20d+10)-(50d+13)$ Operationen. Gehen wir von der Annahme aus, dass d unabhängig von N ist, so beträgt die Komplexität von Algorithmus 7.4 $O(N)$.
Ein weiterer, nicht unerheblicher Punkt ist der Speicherplatzbedarf des Verfahrens. Wir benötigen $N-1$ bzw. $N-2$ Speicherplätze für die Einträge von $\mathbf{b}$ bzw. $\mathbf{c}$. Die Mittelwerte und die Werte für die mittlere Streuung können wir vernachlässigen. Für die Eckenmenge wäre der worst case ein Speicherplatzbedarf von N Stellen. Daher beträgt der Speicherplatzbedarf für den Algorithmus 7.4 $O(N)$.

7.3 Eckenbestimmung mit Wavelets

Das zweite Verfahren zur Bestimmung der Ecken einer Kontur fußt auf der schnellen Wavelettransformation, besser gesagt auf der Auswertung der Koeffizienten, die bei der schnellen Wavelettransformation entstehen.
Zunächst benötigen wir etwas Notation.

7.6 Bezeichnung: Sei $K = \{\mathbf{v}_j \mid j = 0, \ldots, N-1\} \subset \mathbb{R}^d$ die zu untersuchende Kontur. Die Konturpunkte $\mathbf{v}_j$ haben für $j = 0, \ldots, N-1$ die Gestalt

$$\mathbf{v}_j = \begin{bmatrix} v_j^{(1)} \\ \vdots \\ v_j^{(d)} \end{bmatrix}.$$

Dann bezeichnen wir mit

$$K^{(i)} = \{v_j^{(i)} \mid j = 0, \ldots, N-1\} \text{ für } i = 1, \ldots, d$$

die Menge der i–ten Komponenten von K.

Wir benutzen die Bezeichnungen und Begriffe des Abschnittes 4. Wir behalten allerdings die Notation der Kontur bei, so dass wir die Koeffizienten $c_{j,k}$ aus Abschnitt 4 als $v_{j,k}$ bezeichnen. Entsprechend schreiben wir $w_{j,k}$ an Stelle von $d_{j,k}$ für die Waveletkoeffizienten. Damit gilt $K_j = \{v_{j,k}\}_{k\in\mathbb{Z}}$

$$\begin{array}{ccccccccc} K & \overset{S}{\rightarrow} & K_1 & \overset{S}{\rightarrow} & K_2 & \cdots & K_{M-1} & \overset{S}{\rightarrow} & K_M \\ & \overset{W}{\searrow} & & \overset{W}{\searrow} & & \cdots & & \overset{W}{\searrow} & \\ & & L_1 & & L_2 & & \cdots & & L_M \end{array}$$

Abbildung 16: Pyramidenschema

sowie $L_j = \{w_{j,k}\}_{k\in\mathbb{Z}}$.
Sei $\varphi \in L_2(\mathbb{R})$ eine Skalierungsfunktion mit der Verfeinerungsfolge $\{h_k\}_{k\in\mathbb{Z}}$ und $\psi \in L_2(\mathbb{R})$ ein zugehöriges Wavelet mit den Koeffizienten $\{h_k^\perp\}_{k\in\mathbb{Z}}$. Wir untersuchen die Kontur komponentenweise, daher steht im Folgenden K für eine beliebige Komponente $K^{(i)}$ für $i \in \{1,\ldots,d\}$. Wir wenden auf K die schnelle Wavelettransformation (4.37) an und erhalten

$$K_1 = \{v_{1,k}\}_{k=0,\ldots,\lfloor\frac{N-1}{2}\rfloor}$$

sowie

$$L_1 = \{w_{1,k}\}_{k=0,\ldots,\lfloor\frac{N-1}{2}\rfloor}.$$

Im Einzelnen gilt

$$v_{1,k} = \sum_{l\in\mathbb{Z}} h_{l-2k} v_l \text{ für } k = 0,\ldots,\left\lfloor\frac{N-1}{2}\right\rfloor,$$
$$w_{1,k} = \sum_{l\in\mathbb{Z}} h^\perp_{l-2k} v_l \text{ für } k = 0,\ldots,\left\lfloor\frac{N-1}{2}\right\rfloor.$$

Dabei setzen wir $v_l = 0$ für $l < 0$ und $l > N-1$. Abhängig von der Länge der Kontur wenden wir die schnelle Wavelettransformation (4.37) erneut an, diesmal auf K_1, und erhalten die Mengen K_2 und L_2. Mit den Bezeichnungen aus Abschnitt 4.7 erhalten wir das in Abbildung 16 dargestellte Pyramidenschema.
Die Frage, inwiefern eine solche Waveletzerlegung beim Suchen von Ecken einer Kontur hilfreich ist, beantworten die folgenden Aussagen. Vorher müssen wir eine zusätzliche Eigenschaft von Funktionen aus $L_1(\mathbb{R})$ definieren.

7.7 Definition: Eine Funktion $f \in L_1(\mathbb{R})$ erfüllt die *Strang-Fix-Bedingung der Ordnung* $n \in \mathbb{N}_0$, wenn gilt

(1) $\widehat{f}(0) \neq 0$ und

(2)
$$\widehat{f}^{(j)}(2\pi k) = 0 \text{ für alle } k \in \mathbb{Z}\backslash\{0\} \text{ und } j = 0, \ldots, n.$$

Mit der Voraussetzung, dass eine Funktion die Strang-Fix-Bedingung erfüllt, können wir den folgenden Satz formulieren.

7.8 Satz: Sei $\psi \in \mathcal{C}_{00}(\mathbb{R})$ ein Wavelet zu einer Skalierungsfunktion $\varphi \in \mathcal{C}_{00}(\mathbb{R})$, welche die Strang-Fix-Bedingung der Ordnung n erfüllt, und sei $f \in L_2^{n+1}(\mathbb{R})$. Dann existiert eine Konstante $C > 0$, so dass mit den Bezeichnungen aus Abschnitt 4.7 gilt

$$|w_{j,k}| \leq 2^{-jn} C \|f^{(n+1)}\|_2. \tag{7.4}$$

Dabei ist $w_{j,k}$ der k–te Waveletkoeffizient der Skala j.

Beweis: Den kompletten Beweis finden wir zum Beispiel in [Str73], wir skizzieren ihn an dieser Stelle lediglich.

(1) Wir bezeichnen mit Ψ_l, $l = 0, \ldots, n$, die l–te Stammfunktion zu ψ. Dann können wir durch vollständige Induktion nach l zeigen, dass gilt
$$\int_{\mathbb{R}} t^j \Psi_l dt = 0 \text{ für } j = 0, \ldots, n-l \text{ und } l = 0, \ldots, n$$
und dass Ψ_l einen kompakten Träger hat. Wir verwenden an dieser Stelle, dass φ der Strang-Fix Bedingung der Ordnung n genügt und ψ daher $n+1$ verschwindende Momente hat.

(2) Mit Hilfe partieller Integration erhalten wir
$$\langle f, \psi(2^j \cdot - k)\rangle = (-1)^{n+1} 2^{-j(n+1)} \int_{\mathbb{R}} f^{(n+1)}(t) \Psi_{n+1}(2^j t - k) dt.$$

(3) Dann folgt
$$\begin{aligned} |w_{j,k}| &\leq \|\Psi_{n+1}\|_\infty 2^{-jn} \int_{\mathbb{R}} \chi_{[a,b]}(t) \left| f^{(n+1)}(t) \right| dt \\ &\leq \|\Psi_{n+1}\|_\infty 2^{-jn} \|\chi_{[a,b]}\|_2 \|f^{(n+1)}\|_2 \\ &= \underbrace{\|\Psi_{n+1}\|_\infty \sqrt{(b-a)}}_{=:C} 2^{-jn} \|f^{(n+1)}\|_2. \end{aligned}$$

Dabei ist $[a, b]$ der Träger von Ψ_{n+1}.

□

7.9 Lemma: Sei $K = \{v_j \mid j = 0, \ldots, N-1\}$ eine Abtastung eines eindimensionalen Signals und sei ψ ein Wavelet mit den in Satz 7.8 beschriebenen Eigenschaften. Ist die dem Signal zu Grunde liegende Funktion $f \in L_2^{n+1}(\mathbb{R})$ für ein $n \in \mathbb{N}$, so gilt $|w_{j,k}| \to 0$ für $j \to \infty$.

Beweis: Sei $f \in L_2^{n+1}(\mathbb{R})$ für ein $n \in \mathbb{N}$ die zu Grunde liegende Funktion. Besitzt ψ den Träger $[a, b]$, so hat die skalierte und verschobene Version $\psi(2^j \cdot - k)$ den Träger $2^{-j}(k + [a, b])$ und berücksichtigt damit nur das lokale Glattheitsverhalten von f um den Punkt $2^{-j}k$.
Ist f an dieser Stelle glatt, also stetig differenzierbar, so gilt nach Satz 7.8

$$|w_{j,k}| \leq 2^{-jn} C \|f^{(n+1)}\|_2 \to 0 \text{ für } j \to \infty,$$

da $\|f^{(n+1)}\|_2 < \infty$. □

7.10 Bemerkung: Die Konsequenz aus Lemma 7.9 ist folgende. Liegt uns ein stetig differenzierbares Signal vor, so werden die Waveletkoeffizienten relativ schnell klein. Liegen in mehreren Skalen Koeffizienten vor, die signifikant groß sind, so haben wir ein Indiz für eine singuläre Stelle. Wir können allerdings nicht pauschal sagen, dass die Koeffizienten signifikant groß werden, falls die dem Signal zu Grunde liegende Funktion nicht differenzierbar oder nicht stetig ist.

Wir verwenden im weiteren Verlauf ausschließlich Wavelets und zugehörige Skalierungsfunktionen, welche die Voraussetzungen des Satzes 7.8 erfüllen. Für den Gesamtalgorithmus betrachten wir die Kontur K als Teilmenge des $\mathbb{R}^d$, berechnen für jede Kontur und jede Komponente der Kontur einige Skalen der schnellen Wavelettransformation und bewerten die entstandenen Koeffizienten $w_{j,k}^{(i)}$.

7.11 Algorithmus: Wir starten mit $K = \{\mathbf{v}_j \mid j = 0, \ldots, N-1\} \subset \mathbb{R}^d$ und schreiben die Kontur als Folge $K = \{\mathbf{v}_j\}_{j=0}^{N-1}$. Wir betrachten die Komponentenfolgen $K^{(1)} = \{v_j^{(1)}\}_{j=0}^{N-1}, \ldots, K^{(d)} = \{v_j^{(d)}\}_{j=0}^{N-1}$. Seien $\{h_k\}_{k\in\mathbb{Z}}$ die Verfeinerungsfolge der Skalierungsfunktion $\varphi \in L_2(\mathbb{R})$ und $\{h_k^{\perp}\}_{k\in\mathbb{Z}}$ die

Koeffizienten des Wavelets ψ. Seien darüber hinaus φ und ψ als Funktionen mit kompakten Trägern gewählt.

(1) Wir bestimmen die Anzahl der berechneten Skalen $M > 0$, so dass $\left|L_M^{(i)}\right|_0 > 10$ gilt. Dann ist

$$M < \left\lfloor \left| \log_2 \frac{10}{N} \right| \right\rfloor .$$

(2) Die Anwendung der schnellen Wavelettransformation (4.37), jeweils auf $K^{(1)}, \ldots, K^{(d)}$, liefert uns die Folgen

$$\begin{array}{cccccc} K_1^{(1)}, & \ldots & K_M^{(1)}, & L_1^{(1)}, & \ldots & L_M^{(1)}, \\ K_1^{(2)}, & \ldots & K_M^{(2)}, & L_1^{(2)}, & \ldots & L_M^{(2)}, \\ \vdots & & \vdots & \vdots & & \vdots \\ K_1^{(d)}, & \ldots, & K_M^{(d)}, & L_1^{(d)}, & \ldots & L_M^{(d)}. \end{array}$$

(3) Für jedes $i = 1, \ldots, d$ und $l = 1, \ldots, M$ bestimmen wir den Mittelwert und die mittlere Streuung der Waveletkoeffizienten

$$\overline{w_l^{(i)}} = \frac{1}{\lfloor \frac{N}{2^l} \rfloor} \sum_{k=0}^{\lfloor \frac{N}{2^l} \rfloor - 1} w_{l,k}^{(i)} \text{ sowie } \sigma_l^{(i)} = \sqrt{\frac{1}{\lfloor \frac{N}{2^l} \rfloor} \sum_{k=0}^{\lfloor \frac{N}{2^l} \rfloor - 1} \left(w_{l,k}^{(i)} - \overline{w_l^{(i)}} \right)^2}.$$

(4) Gilt

$$\left| w_{l,k_l} - w_l^{(i)} \right| > C \sigma_l^{(i)}$$

für eine Konstante $C \in \mathbb{R}$ für mehr als ϱ der Einträge $l \in \{1, \ldots, M\}$ und mindestens ein $i \in \{1, \ldots, d\}$, so liegt eine Ecke vor, also eine nicht differenzierbare oder sogar unstetige Stelle. Dabei bezeichnen wir mit k_l für $l = 1, \ldots, M$ die zugehörigen Waveletkoeffizienten auf der Skala l.

(5) Als Ausgabe erhalten wir eine Menge von Ecken $E_2 = \{e_1, \ldots, e_R\} \subset \{0, \ldots, N-1\}$.

In der Praxis wählen wir die Anzahl der Skalen zur Identifizierung einer Singularität ϱ, so dass $\frac{\varrho}{M} \approx \frac{3}{4}$ gilt.
Nachfolgend wollen wir eine Kontur und die Werte der Waveletkoeffizienten verschiedener Skalen mit unterschiedlichen Wavelets betrachten.

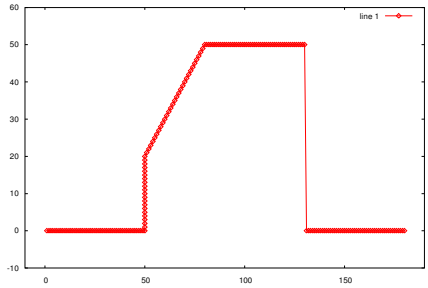

Abbildung 17: Testkontur mit drei Ecken und einer unstetigen Stelle.

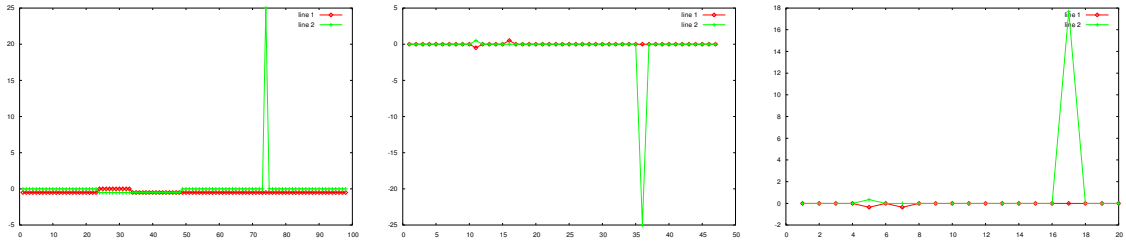

Abbildung 18: Waveletkoeffizienten der ersten drei Skalen des Haarwavelets.

7.12 Beispiel: Wir betrachten die Kontur $K \subset \mathbb{R}^2$, dargestellt in Abbildung 17. Als Wavelets benutzen wir das Haarwavelet, welches die Strang-Fix Bedingung nicht erfüllt, zwei Daubechies-Wavelets, namentlich das D_2– und das D_3–Wavelet sowie das biorthogonale Spline-Wavelet $3,3$. Dabei genügen die Daubechies- und die Spline-Wavelets der Strang-Fix Bedingung.

(1) *Haarwavelet*
Wir sehen in Abbildung 18, dass sich das Haarwavelet nur bedingt zur Eckendetektion eignet, da nicht alle Ecken in allen Skalen erkannt werden. Dies liegt unter anderem an der Tatsache, dass das Haarwavelet die Strang-Fix Bedingungen nicht erüllt.

(2) *Daubechies-Wavelet* D_2
Anhand der Koeffizienten des D_2-Wavelets in Abbildung 19 erkennen wir zuverlässig alle Ecken. Zudem sehen wir, welche Stellen zu den Ecken und welche zu den Unstetigkeiten gehören. Es handelt sich um eine Eigenschaft, die wir später für die symmetrischen Wavelets etwas genauer untersuchen werden.

(3) *Daubechies-Wavelet* D_3
Wir sehen in Abbildung 20, dass wir bei der Betrachtung der Wavelet-

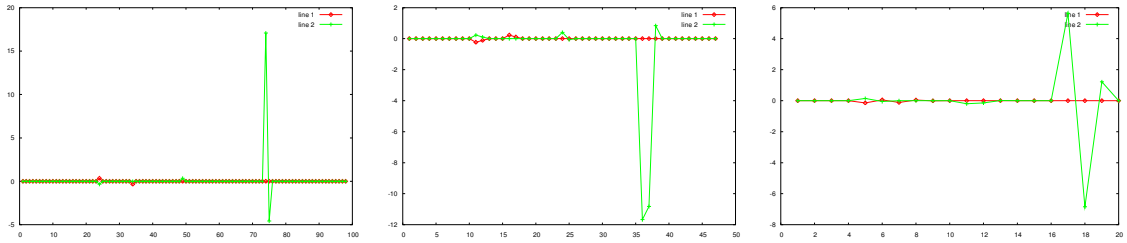

Abbildung 19: Waveletkoeffizienten der ersten drei Skalen des Daubechies-Wavelets D_2.

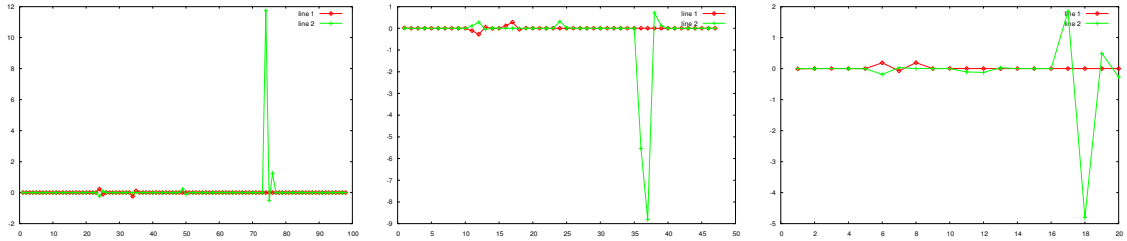

Abbildung 20: Waveletkoeffizienten der ersten drei Skalen des Daubechies-Wavelets D_3.

koeffizienten des Daubechies-Wavelets D_3 ebenfalls alle Singularitäten erkennen können. Allerdings liegt mit Ausnahme der Größe der Waveletkoeffizienten keine erkennbare Struktur vor, um Ecken und nicht stetige Stellen voneinander zu unterscheiden.

(4) *Biorthogonales Spline-Wavelet* $3,3$
Die Koeffizienten des biorthogonalen Spline-Wavelets 3,3 ermöglichen uns ebenfalls die Ermittlung aller singulären Stellen der vorliegenden Kontur, wie wir in Abbildung 21 sehen können. Wir können darüber hinaus eine weitere erfreuliche Verhaltensweise erkennen: Bei nicht stetigen Stellen verhalten sich die Waveletkoeffizienten grundsätzlich anders als bei Ecken. Wir werden diese Eigenschaft untersuchen und in Form eines Satzes zusammenfassen.

Bevor wir uns mit dem Zusammenspiel der beiden vorgestellten Algorithmen zur Detektion von Singularitäten beschäftigen, wollen wir kurz auf zwei weitere Fragestellungen eingehen.
Einerseits stellt sich wie üblich die Frage nach Laufzeit und Speicherbedarf. Andererseits wollen wir wissen, ob mit Wavelets tatsächlich die angedeutete Möglichkeit der Unterscheidung der Singularität besteht. Wir möchten also anhand des Verhaltens der Waveletkoeffizienten entscheiden, ob es sich um eine nicht differenzierbare oder um eine nicht stetige Stelle des Signals

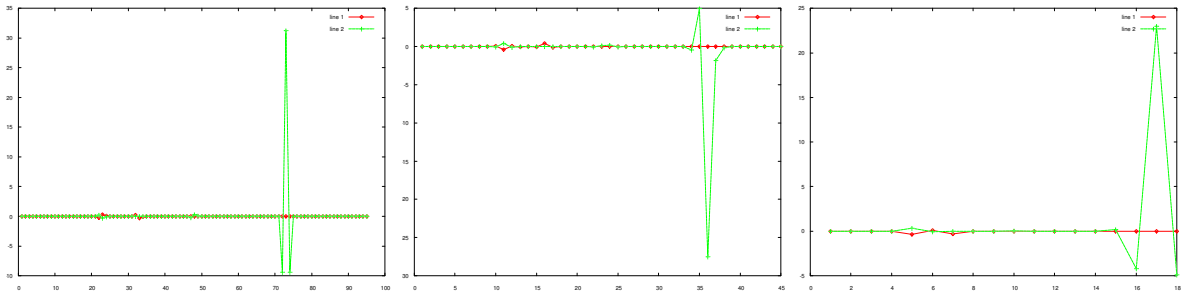

Abbildung 21: Waveletkoeffizienten der ersten drei Skalen des biorthogonalen Spline-Wavelets 3, 3.

handelt.
Wir beschäftigen uns als Erstes mit der Frage nach der Laufzeit und dem Speicherbedarf und geben anschließend ein Kriterium zur Unterscheidung von Singularitäten an.
Der Aufwand der Bestimmung von M ist unabhängig von N und daher können wir Punkt (1) von Algorithmus 7.11 vernachlässigen. Allerdings gilt für die Rekursionstiefe M zumindest in dem Spezialfall, dass N eine Zweierpotenz ist und wir alle Skalen berechnen, $N = 2^M$ und damit $M = \log_2 N$.
Für den ersten Schritt der schnellen Wavelettransformation, der Berechnung von $K_1^{(i)}$ und $L_1^{(i)}$, sind $(\lfloor \frac{N}{2} \rfloor + 1) 2\,|h|$ und $(\lfloor \frac{N}{2} \rfloor + 1) 2\,|h^\perp|$ Rechenoperationen nötig. Diese Berechnung führen wir für alle Koordinaten $i = 1, \ldots, d$ durch und erhalten entsprechend

$$\left(\left\lfloor \frac{N}{2} \right\rfloor + 1 \right) 2\,|h|\,d \text{ sowie } \left(\left\lfloor \frac{N}{2} \right\rfloor + 1 \right) 2\,|h^\perp|\,d$$

Operationen. Bei insgesamt M berechneten Skalen gilt für die Anzahl der Operationen A_{FWT} von Algorithmus 7.11 (2)

$$\begin{aligned} A_{FWT} &= \sum_{s=1}^{M} 4\,|h|\,d \left(\left\lfloor \frac{N}{2^s} \right\rfloor + 1 \right) \leq \sum_{s=1}^{M} 4\,|h|\,d \left(\frac{N}{2^s} + 1 \right) \\ &= 4 \sum_{s=1}^{M} |h|\,d + 4N\,|h|\,d \sum_{s=1}^{M} \frac{1}{2^s} \leq 4\,|h|\,d \left(M + N \underbrace{\sum_{s=1}^{\infty} \frac{1}{2^s}}_{\leq 1} \right) \\ &\leq 4\,|h|\,d(M+N). \end{aligned}$$

Dabei nutzen wir die Tatsache aus, dass für die Daubechies- und das Haarwavelet $|h| = \left|h^{\perp}\right|$ gilt. Für die biorthogonalen Spline-Wavelets verwenden wir $|h| = \max\{|h|, \left|h^{\perp}\right|\}$. Die Berechnung der Mittelwerte und der Werte für die mittlere quadratische Streuung führen wir ebenfalls für jede Skala und jede Koordinate durch. Mit einer ähnlichen Abschätzung wie oben kommen wir auf eine Gesamtanzahl von

$$d(N+M) \text{ bzw. } 2d(N+M)$$

Operationen für die Mittelwertbildung bzw. Bildung der mittleren quadratischen Streuung. Punkt (4) des Algorithmus benötigt dann höchstens $d(M+N)$ Operationen. Insgesamt bekommt man für Algorithmus 7.11 einen Aufwand von

$$4d(M+N)\,(|h|+1)$$

Rechenoperationen, was wegen der Unabhängigkeit von d und $|h|$ von N und der Beziehung $M = \log_2 N$ eine Komplexität von $O(N + \log_2 N) = O(N)$ ergibt.
Bei der Berechnung des Speicherplatzbedarfs schätzen wir ebenfalls etwas genauer ab als bei der Anzahl der Operationen und erhalten folgende Werte.
Wir benötigen für ein festes $i \in \{1, \ldots, d\}$ für $K_1^{(i)}$ insgesamt höchstens $(\lfloor \frac{N}{2} \rfloor + 1)$ Speicherplätze und dieselbe Anzahl für $L_1^{(i)}$. Für den kompletten ersten Schritt des Punktes (2) von Algorithmus 7.11 benötigen wir höchstens $d(N+3)$ Speicherstellen. Da wir keine Rekonstruktion anstreben, brauchen wir die Einträge $K_j^{(i)}$ nur temporär zu speichern und können den entsprechenden Speicherplatz weiter verwenden. Es gilt

$$\left|L_j^{(i)}\right| \leq \left\lfloor \frac{N}{2^j} \right\rfloor + 1 \text{ und } \left|K_j^{(i)}\right| \leq \left\lfloor \frac{N}{2^j} \right\rfloor + 1$$

und damit auch

$$\left|L_j^{(i)}\right| + \left|K_j^{(i)}\right| \leq \left|K_{j-1}^{(i)}\right|$$

für $i = 1, \ldots, d$ und $j = 2, \ldots, M$. Der Gesamtspeicherplatzbedarf Sp_{ges} ergibt sich als

$$\begin{aligned}
Sp_{ges} &= d \underbrace{\left(\sum_{j=1}^{M} \left\lfloor \frac{N}{2^j} \right\rfloor + 1 \right)}_{L_j^{(i)}} + d \underbrace{\left(\left\lfloor \frac{N}{2^M} \right\rfloor + 1 \right)}_{K_M^{(i)}} \\
&= d \left(\sum_{j=1}^{M-1} \left(\left\lfloor \frac{N}{2^j} \right\rfloor + 1 \right) + \left(\underbrace{2 \left\lfloor \frac{N}{2^M} \right\rfloor}_{\leq \left\lfloor \frac{N}{2^{M-1}} \right\rfloor} + 2 \right) \right) \\
&\leq d \left(\sum_{j=1}^{M-1} \left(\left\lfloor \frac{N}{2^j} \right\rfloor + 1 \right) + \left(\left\lfloor \frac{N}{2^{M-1}} \right\rfloor + 2 \right) \right) \\
&\vdots \\
&\leq d \left(2 \left\lfloor \frac{N}{2} \right\rfloor + M \right) \\
&\leq d \, (N + M) \, .
\end{aligned}$$

Wir benötigen also $d(N + M)$ Speicherstellen, was mit den obigen Annahmen einem Speicherplatzbedarf von $O(N + \log_2 N) = O(N)$ entspricht. Wir fassen die Erkenntnisse in einem Korollar zusammen

7.13 Korollar: Sei $K \subset \mathbb{R}^d$ eine Kontur. Dann betragen die Laufzeit und der Speicherbedarf des Algorithmus 7.11 jeweils $O(N)$.

7.4 Kriterium zur Unterscheidung von Singularitäten

Jetzt stellen wir, wie angekündigt, ein Kriterium zur Unterscheidung der Singularitäten vor. Wir beschäftigen uns in diesem Kontext ausschließlich mit biorthogonalen Spline-Wavelets und dabei auch nur mit solchen, deren Verfeinerungsfolge die Länge $2\widetilde{n}$ für ein $\widetilde{n} \in \mathbb{N}$ besitzt. Dabei handelt es sich um die biorthogonalen Spline-Wavelets vom Typ $\kappa, \widetilde{\kappa}$ mit $\kappa, \widetilde{\kappa} \in 2\mathbb{N}+1$ aus Abschnitt 4.9. Wir beschränken uns bei unseren Betrachtungen auf eindimensionale Konturen $v_0, \ldots, v_{N-1} \in \mathbb{R}$. Die Waveletkoeffizienten berechnen wir mit Hilfe der schnellen Wavelettransformation (4.37) und

erhalten für die Waveletkoeffizienten der ersten Skala

$$w_{1,k} = \sum_{l\in\mathbb{Z}} h^{\perp}_{l-2k} v_l = \sum_{l=2k-\widetilde{n}+1}^{2k+\widetilde{n}} h^{\perp}_{l-2k} v_l, \tag{7.5}$$

da gilt $h^{\perp}_l = 0$ für $l \notin \{1-\widetilde{n}, \ldots, \widetilde{n}\}$. Dabei sind die Koeffizienten $\{h_l\}_{l\in\mathbb{Z}}$ und $\{\widetilde{h}_l\}_{l\in\mathbb{Z}}$ symmetrisch, es ist also

$$h_l = h_{1-l} \text{ sowie } \widetilde{h}_l = \widetilde{h}_{1-l} \text{ für } l \geq 1.$$

Nach Konstruktion der Waveletkoeffizienten (4.51) erhalten wir die Beziehung

$$h^{\perp}_l = -h^{\perp}_{1-l} \text{ für } l = 1, \ldots, \widetilde{n},$$

also Antisymmetrie der Koeffizienten.
Für unsere Untersuchungen betrachten wir zwei Datensätze. Im ersten modellieren wir eine stetige, aber nicht differenzierbare Stelle, also eine isolierte Ecke, der zweite Datensatz enthält eine nicht stetige Stelle. Wir fassen die Daten als Funktionswerte einer unbekannten Funktion f mit kompaktem Träger auf, für die gilt

$$f(t_l) = v_l \text{ für } l = 0, \ldots, N-1.$$

Dabei nehmen wir zur Vereinfachung an, dass $t_{l+1} - t_l = \Delta t$ für alle $l = 0, \ldots, N-1$ gilt, dass die Daten also äquidistant abgetastet wurden, und die Funktion f bis auf diese isolierte Singularität glatt ist und kompakten Träger hat. Wir modellieren beide Singularitäten wie folgt.

(1) Eine Ecke v_J für ein $0 < J < N-1$ erhalten wir, falls gilt

$$f(t) = \begin{cases} \alpha_1 t & \text{für } t \leq t_J \\ \alpha_2 t & \text{für } t > t_J \end{cases} \text{ mit } f(t_J) = v_J \text{ sowie } \alpha_1 \neq \alpha_2. \tag{7.6}$$

Dabei verschieben wir das Koordinatensystem, so dass $t_J = 0$ gilt. Durch eine solche Verschiebung können wir die Modellierung einer Ecke durch (7.6) bewerkstelligen.
Des weiteren nehmen wir an, dass wir an dieser Stelle so fein abtasten, dass die Funktion f sowohl vor v_J als auch danach einen praktisch linearen Verlauf hat.

(2) Ein Unstetigkeit zwischen v_J und v_{J+1} für ein $0 < J < N-1$ modellieren wir folgendermaßen

$$f(t) = \begin{cases} \alpha_1 t & \text{für } t \leq t_J \\ \alpha_2 t + \alpha_3 & \text{für } t > t_J \end{cases} \text{ mit } f(t_J) = v_J \text{ sowie } \alpha_3 \neq 0. \tag{7.7}$$

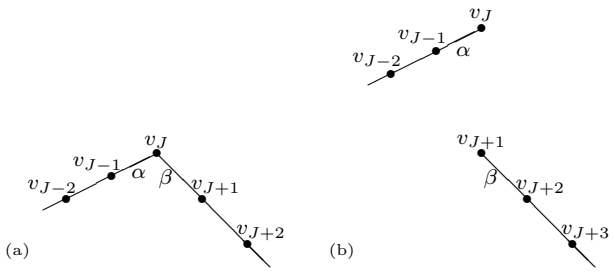

Abbildung 22: Modellierung einer Ecke (a) und einer unstetigen Stelle (b).

In diesem Fall ist $\alpha_1 = \alpha_2$ erlaubt. Wir wählen α_3 signifikant größer als die Abtastung. Hierbei muss nicht mehr $t_{J+1} - t_J = \Delta t$ gelten, diese Tatsache spielt aber für unsere Betrachtungen keine Rolle.
Wie in Fall (1) verschieben wir das Koordinatensystem derart, dass $t_J = 0$ gilt.

In beiden Fällen gehen wir davon aus, dass

$$|f(v_j) - f(v_{j+1})|$$

vor und nach der Singularität ungefähr den gleichen Wert hat. Abbildung 22 zeigt uns beide Fälle.
Aus Abschnitt 4.7 ist bekannt, dass wir die Koeffizienten $w_{r,k}$ als

$$w_{r,k} = \langle f, \widetilde{\psi}_{k,k} \rangle \tag{7.8}$$

berechnen können. In [CoDaFe92] finden wir folgende Eigenschaften von $\widetilde{\psi}$.

7.14 Satz: Sei $\widetilde{\psi}$ das duale Wavelet eines biorthogonalen Spline-Wavelets mit $\kappa, \widetilde{\kappa} \in 2\mathbb{N} + 1$. Dann genügt $\widetilde{\psi}$ folgenden Eigenschaften.

(1) $\widetilde{\psi}$ ist antisymmetrisch um $\frac{1}{2}$ mit $\widetilde{\psi}(x + \frac{1}{2}) = -\widetilde{\psi}(x - \frac{1}{2})$ für alle $x \in \mathbb{R}$. Insbesondere ist $\widetilde{\psi}(\frac{1}{2}) = 0$.

(2) Es gilt $\operatorname{supp} \widetilde{\psi} \subset [-\widetilde{\kappa} - \lfloor \frac{\kappa}{2} \rfloor + 1, \widetilde{\kappa} + \lceil \frac{\kappa}{2} \rceil - 1]$. Wir schreiben $I = [T_1, T_2] = \operatorname{supp} \widetilde{\psi}$.

(3) $\widetilde{\psi}$ hat mindestens $\widetilde{\kappa}$ Nullstellen. Dabei ist die Anzahl der Nullstellen stets ungerade. Seien $n_0 < \cdots < n_{2p}$ diese Nullstellen, so gilt $n_p = \frac{1}{2}$. Dann liegt das globale Maximum von $\widetilde{\psi}$ stets im Intervall (n_{p-1}, n_p) und das globale Minimum entsprechend im Intervall (n_p, n_{p+1}).

(4) Es gilt

$$\int_{\mathbb{R}} \widetilde{\psi}(x)dx = \int_{\mathbb{R}} x^k \widetilde{\psi} dx = 0 \text{ für } k = 0, \ldots, \widetilde{\kappa} - 1. \tag{7.9}$$

Also besitzt $\widetilde{\psi}$ insgesamt $\widetilde{\kappa}$ verschwindende Momente.

In Abbildung 23 sehen wir die Funktionen ψ und $\widetilde{\psi}$ für $\kappa = 3$ und $\widetilde{\kappa} \in \{3, 5, 7\}$.

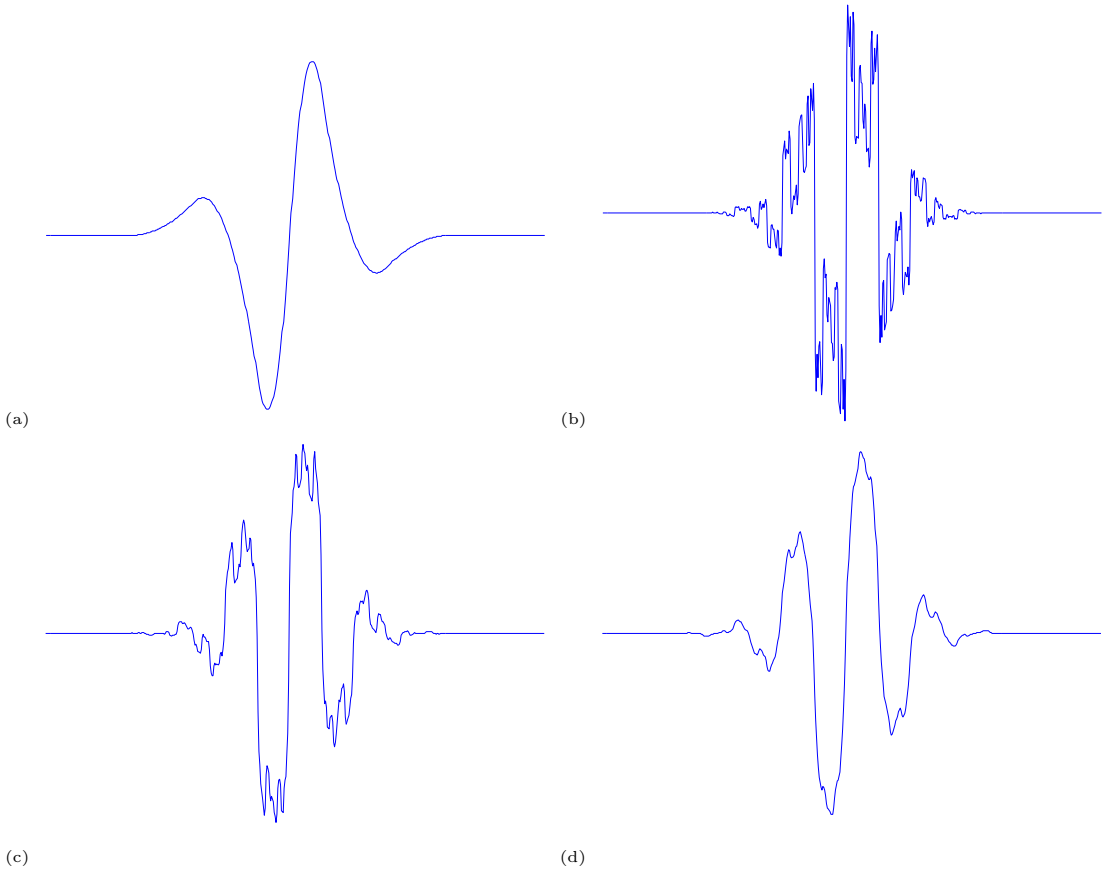

Abbildung 23: (a) Das Wavelet ψ für $\kappa = 3$ (b) Das duale Wavelet $\widetilde{\psi}$ für $\kappa = 3$ und $\widetilde{\kappa} = 3$ (c) Das duale Wavelet $\widetilde{\psi}$ für $\kappa = 3$ und $\widetilde{\kappa} = 5$ (d) Das duale Wavelet $\widetilde{\psi}$ für $\kappa = 3$ und $\widetilde{\kappa} = 7$

Seien $n_0 < \cdots < n_{2p} \in \mathbb{R}$ die Nullstellen von $\widetilde{\psi}$ mit $n_p = \frac{1}{2}$.
Wir beginnen mit einigen Hilfsaussagen und formulieren im Anschluss die Aussagen für die Erkennung der Singularitäten.

7.15 Lemma: Sei $g \in \mathcal{C}(I)$, $I \subseteq \mathbb{R}$, eine antisymmetrische Funktion mit Symmetriepunkt $x_* \in I$. Sei $G \in \mathcal{C}(I)$ die Stammfunktion von g. Dann ist G symmetrisch um $x_* \in I$.

Beweis: Sei $x \in \mathbb{R}$ so gewählt, dass $x_* \pm x \in I$ gilt. Dann erhalten wir

$$G(x_* + x) = \int_{-\infty}^{x_*+x} g(t)dt = \int_{-\infty}^{x_*-x} g(t)dt + \underbrace{\int_{x_*-x}^{x_*+x} g(t)dt}_{=0}$$

$$= G(x_* - x).$$

□

7.16 Bezeichnung: Sei $\widetilde{\psi}$ das duale Wavelet vom Grad $\kappa, \widetilde{\kappa}$ aus Satz 7.14. Wir schreiben

$$\widetilde{\psi}_{r,k} = 2^{\frac{r}{2}} \widetilde{\psi}\,(2^r \cdot -k)$$

für das skalierte und verschobene Wavelet. Dann setzen wir

$$I_{r,k} = 2^{-r}(I + k) = 2^{-r}[T_1, T_2] + 2^{-r}k = \operatorname{supp} \widetilde{\psi}_{r,k}.$$

7.17 Lemma: Sei

$$\Omega = \bigcup_{(r,k)\in\mathcal{J}} I_{r,k} \text{ mit } \mathcal{J} = \{(r,k) \in \mathbb{Z} \times \mathbb{Z} \mid t_J \notin I_{r,k}\}.$$

Sei weiter $f \in \mathcal{C}_{00}^{n+1}(\Omega)$ für $n < \widetilde{\kappa} + 1$ und $\widetilde{\psi}$ das duale Wavelet vom Grad $\kappa, \widetilde{\kappa}$ aus Satz 7.14. Dann gilt

$$|w_{r,k}| = \left|\langle f, \widetilde{\psi}_{r,k}\rangle\right| \leq 2^{-rn} C \|f^{(n+1)}\|_{\Omega,2}. \tag{7.10}$$

Beweis: Da nach Voraussetzung $f \in C_{00}^{n+1}(\Omega)$ gilt, folgt sofort $f \in L_2^{n+1}(\Omega)$. Nun haben die biorthogonalen Spline-Wavelets vom Grad $\kappa, \widetilde{\kappa}$ aus Lemma 7.14 einen kompakten Träger und erfüllen darüber hinaus die Strang-Fix Bedingung der Ordnung $\widetilde{\kappa}$. Damit können wir Satz 7.8 eingeschränkt auf Ω anwenden und es folgt

$$|w_{r,k}| \leq 2^{-rn} C \|f^{(n+1)}\|_{\Omega,2}.$$

□

7.18 Bemerkung: Nach Definition von Ω folgt für die Funktionen (7.6) und (7.7) die Aussage

$$|w_{r,k}| \leq 2^{-rn} C \|f^{(n+1)}\|_{\Omega,2} \to 0 \text{ für } r \to \infty.$$

7.19 Lemma: Sei $\widetilde{\psi}$ das duale Wavelet vom Grad $\kappa, \widetilde{\kappa}$ aus Satz 7.14. Sei $\widetilde{\Psi}$ die Stammfunktion zu $\widetilde{\psi}$. Dann gilt

$$\int_{-\infty}^{n_p} \widetilde{\Psi}(t)dt = \int_{n_p}^{\infty} \widetilde{\Psi}(t)dt = 0.$$

Beweis: Nach Lemma 7.15 ist $\widetilde{\Psi}$ symmetrisch um n_p. Des weiteren hat $\widetilde{\Psi}$ nach dem Beweis von Satz 7.8 $\widetilde{\kappa} - 1$ verschwindende Momente. Dabei ist nach Satz 7.14 $\widetilde{\kappa} \in 2\mathbb{N} + 1$, also $\widetilde{\kappa} > 1$. Damit gilt

$$0 = \int_{\mathbb{R}} \widetilde{\Psi}(t)dt = \int_{-\infty}^{n_p} \widetilde{\Psi}(t)dt + \int_{n_p}^{\infty} \widetilde{\Psi}(t)dt = 2\int_{-\infty}^{n_p} \widetilde{\Psi}(t)dt = 2\int_{n_p}^{\infty} \widetilde{\Psi}(t)dt.$$

□

Jetzt sind wir in der Lage, die erste Aussage über die Erkennung der Singularitäten zu formulieren.

7.20 Satz: Seien f wie in (7.6) und $\widetilde{\psi}$ das duale Wavelet von Grad $\kappa, \widetilde{\kappa}$ aus Satz 7.14. Die Funktion f habe bei t_J eine nicht differenzierbare Stelle. Ist der zugehörige Waveletkoeffizient und einer seiner Nachbar signifikant groß, dann besitzen beide Koeffizienten alternierende Vorzeichen.

Beweis: Wir passen als Erstes die Skalierung r des Wavelets $\widetilde{\psi}_{r,k}$ an. Dabei bestimmen wir $r \in \mathbb{R}$ und $k_1 \in \mathbb{Z}$, so dass gilt

$$-2^{-r}k_1 < 2^{-r}n_p < 2^{-r} - 2^{-r}k_1 \text{ sowie } 0 \in I_{r,k_1}. \tag{7.11}$$

Es ergibt sich

$$r \geq -\log_2 \Delta t \text{ und } k_1 = \lfloor -n_p \rfloor = 0. \tag{7.12}$$

Wir verwenden nachfolgend für die Stammfunktion $\widetilde{\Psi}$ die Notation, welche wir in der Bezeichnung 7.16 für $\widetilde{\psi}$ eingeführt haben. Wir betrachten

$$w_{r,0} = \int_{I_{r,0}} f(t)\widetilde{\psi}_{r,0}(t)dt = \underbrace{\alpha_1 \int_{-\infty}^{0} t \cdot \widetilde{\psi}_{r,0}(t)dt}_{=A} + \underbrace{\alpha_2 \int_{0}^{\infty} t \cdot \widetilde{\psi}_{r,0}(t)dt}_{=B}.$$

Dann erhalten wir mit Hilfe partieller Integration

$$\begin{aligned} A &= 2^{-r} \underbrace{\left[\alpha_1 t \widetilde{\Psi}_{r,0}(t)\right]_{-\infty}^{0}}_{=0} - 2^{-r}\alpha_1 \int_{-\infty}^{0} \widetilde{\Psi}_{r,0}(t)dt \\ &= -2^{-r}\alpha_1 \int_{-\infty}^{0} \widetilde{\Psi}_{r,0}(t)dt \\ &= -2^{-r}\alpha_1 \left[\underbrace{\int_{-\infty}^{2^{-r}n_p} \widetilde{\Psi}_{r,0}(t)dt}_{=0 \text{ nach Lemma 7.19}} - \int_{0}^{2^{-r}n_p} \widetilde{\Psi}_{r,0}(t)dt \right] \\ &= 2^{-r}\alpha_1 \int_{0}^{2^{-r}n_p} \widetilde{\Psi}_{r,0}(t)dt = 2^{-r}\alpha_1 \int_{0}^{2^{-r-1}} \widetilde{\Psi}_{r,0}(t)dt. \end{aligned}$$

In der letzten Zeile erinnern wir uns an $n_p = \frac{1}{2}$. Analog gilt

$$\begin{aligned} B &= 2^{-r} \left[\alpha_2 t \widetilde{\Psi}_{r,0}(t)\right]_{0}^{\infty} - 2^{-r}\alpha_2 \int_{0}^{\infty} \widetilde{\Psi}_{r,0}(t)dt \\ &= -2^{-r}\alpha_2 \int_{0}^{\infty} \widetilde{\Psi}_{r,0}(t)dt \\ &= -2^{-r}\alpha_2 \left[\underbrace{\int_{2^{-r}n_p}^{\infty} \widetilde{\Psi}_{r,0}(t)dt}_{=0 \text{ nach Lemma 7.19}} + \int_{0}^{2^{-r}n_p} \widetilde{\Psi}_{r,0}(t)dt \right] \\ &= -2^{-r}\alpha_2 \int_{0}^{2^{-r}n_p} \widetilde{\Psi}_{r,0}(t)dt = -2^{-r}\alpha_2 \int_{0}^{2^{-r-1}} \widetilde{\Psi}_{r,0}(t)dt. \end{aligned}$$

Es folgt

$$w_{r,0} = A + B = 2^{-r}(\alpha_1 - \alpha_2) \int_{0}^{2^{-r-1}} \widetilde{\Psi}_{r,0}(t)dt. \tag{7.13}$$

Wegen $2^{-r}(n_p - 1) = -2^{-r-1} < 0$ erhalten wir ebenso

$$w_{r,-1} = 2^{-r}(\alpha_2 - \alpha_1) \int_{2^{-r}(n_p-1)}^{0} \widetilde{\Psi}_{r,-1}(t)dt = 2^{-r}(\alpha_2 - \alpha_1) \int_{-2^{-r-1}}^{0} \widetilde{\Psi}_{r,-1}(t)dt.$$

Nun ist $\widetilde{\Psi}_{r,0}$ symmetrisch um $2^{-r}n_p = 2^{-r-1}$, während der Symmetriepunkt von $\widetilde{\Psi}_{r,-1}$ gleich $2^{-r}(n_p - 1) = 2^{-r-1} - 2^{-r} = -2^{-r-1}$ ist. Die Symmetrie-

punkte liegen daher stets symmetrisch um Null. Mit Hilfe einer Variablentransformation erhalten wir

$$w_{r,-1} = 2^{-r}(\alpha_2 - \alpha_1) \int_{-2^{r-1}}^{0} \widetilde{\Psi}_{r,-1}(t)dt = 2^{-r}(\alpha_2 - \alpha_1) \int_{2^{-r-1}}^{2^{-r}} \widetilde{\Psi}_{r,0}(t)dt.$$

Wegen der Symmetrie von $\widetilde{\Psi}_{r,0}$ um 2^{-r-1} und $2^{-r} - 2^{-r-1} = 2^{-r-1}$ gilt

$$\int_{0}^{2^{-r-1}} \widetilde{\Psi}_{r,0}(t)dt = \int_{2^{-r-1}}^{2^{-r}} \widetilde{\Psi}_{r,0}(t)dt.$$

Es ist $(\alpha_1 - \alpha_2) = -(\alpha_2 - \alpha_1)$ und damit folgt $\operatorname{sgn} w_{r,0} \neq \operatorname{sgn} w_{r,-1}$. Damit haben wir das alternierende Verhalten der Koeffizienten gezeigt.
Betrachten wir $w_{r,0}$, so gilt

$$\begin{aligned} w_{r,0} &= (\alpha_1 - \alpha_2) \int_{0}^{2^{-r-1}} 2^{-r} \widetilde{\Psi}_{r,0}(t)dt = (\alpha_1 - \alpha_2) \int_{0}^{2^{-r-1}} 2^{-r} 2^{\frac{r}{2}} \widetilde{\Psi}(2^r t)dt \\ &= 2^{-\frac{r}{2}}(\alpha_1 - \alpha_2) \int_{0}^{2^{-r-1}} \widetilde{\Psi}(2^r t)dt. \end{aligned}$$

Den gleichen Sachverhalt können wir auch für $w_{r,-1}$ zeigen. Es folgt

$$|w_{r,j}| \leq 2^{-\frac{r}{2}} |\alpha_1 - \alpha_2| \, \|\widetilde{\Psi}\|_\infty \text{ für } j \in \{-1, 0\}.$$

Nehmen wir an, für die Funktion f gilt $f \in \mathcal{C}_{00}^{n+1}(\mathbb{R}\backslash\{t_J\})$. Dann erhalten wir nach (7.10) für $k \in \mathbb{Z}$ so gewählt, dass $t_J \notin I_{r,k}$,

$$|w_{r,k}| \leq 2^{-rn} C \|f^{(n+1)}\|_2.$$

Die alternierenden Koeffizienten fallen wie $2^{-\frac{r}{2}}$ ab, während die Koeffizienten außerhalb der Ecke wie 2^{-rn} abfallen, also signifikant schneller. □

7.21 Bemerkung:

(1) Das alternierende Verhalten der Koeffizienten ist essentiell für die Erkennung. In der Theorie könnte man eine nicht differenzierbare Stelle und eine nicht stetige Stelle anhand der Abfallrate der Koeffizienten identifizieren. Da uns aber in der Praxis nur einige wenige Skalen r zur Verfügung stehen, hätte diese Methode zur Folge, dass wir zum Beispiel eine scharfe Ecke und eine kleine Sprungstelle nicht unterscheiden könnten.

(2) Gilt für die Skalierung $r = 1$, so erhalten wir die alternierenden Koeffizienten w_{1,k_J-1} und w_{1,k_J} für $k_J = \frac{J}{2}$ und damit $t_J = t_{2k_J}$. Falls $k_J = \frac{J-1}{2}$ gilt, sind w_{1,k_J} und w_{1,k_J+1} die entsprechenden alternierenden Waveletkoeffizienten.

(3) Für verschiedene Konstellationen von α_1, α_2 und die Skalierung $r = 1$ erhalten wir verschiedene Vorzeichen der Waveletkoeffizienten, welche wir in der nachfolgenden Tabelle aufführen.

	$\alpha_1, \alpha_2 \leq 0$	$\alpha_1, \alpha_2 > 0$	$\alpha_1 \leq 0, \alpha_2 > 0$	$\alpha_1 > 0, \alpha_2 \leq 0$
w_{1,k_J}	< 0	> 0	< 0	> 0
w_{1,k_J-1}	> 0	< 0	> 0	< 0
w_{1,k_J}	> 0	< 0	> 0	< 0
w_{1,k_J+1}	< 0	> 0	< 0	> 0

Wir nehmen stets $\alpha_1 > \alpha_2$ an, falls $\operatorname{sgn}\alpha_1 = \operatorname{sgn}\alpha_2$ gilt.

Wir wenden uns nun der Betrachtung einer nicht stetigen Stelle zu. Um die Vorgehensweise zu motivieren, beginnen wir mit einem Beispiel.

7.22 Beispiel: Wir betrachten zunächst eine Funktion f der folgenden Gestalt

$$f(t) = \begin{cases} -1 & \text{falls } t \leq t_J \\ 1 & \text{sonst.} \end{cases}$$

Wir wählen als Abtastung $\Delta t = 1$. Nach (7.12) gilt $r \geq 0$ und wir wählen $r = 1$. Wir wissen, bei einem Spline-Wavelet vom Typ 3, 3 gilt $n_{p-1} = 0$ und $n_{p+1} = 1$ sowie $T_1 = -1.5$ und $T_2 = 2.5$. Damit suchen wir $k \in \mathbb{Z}$, so dass $t_J \in 2^{-1}(T_1, T_2) + 2^{-1}k$. Wählen wir in diesem Fall $t_J \in \mathbb{Z}$, so finden wir vier $k \in \mathbb{Z}$, so dass die obige Bedingung gilt. Sei zum Beispiel $t_J = 1$, so ist für $k \in \{0, 1, 2, 3\}$ die Bedingung $t_J \in 2^{-1}(T_1, T_2) + 2^{-1}k$ erfüllt. Wir erhalten daher vier Koeffizienten ungleich Null. Dabei gilt $2t_J - k \in \{-1, 0, 1, 2\}$ für $k \in \{0, 1, 2, 3\}$ symmetrisch um $\frac{1}{2}$. Damit erhalten wir für die Koeffizienten $w_{1,1} = w_{1,2}$ und $w_{1,0} = w_{1,3}$.
Wählen wir hingegen $t_J = \frac{1}{4}$, so gilt $t_J \in 2^{-1}(T_1, T_2) + 2^{-1}k$ für $k \in \{-1, 0, 1\}$. Mit demselben Argument wie oben gilt dann $w_{1,-1} = w_{1,1}$. Dabei hat $w_{1,0}$ ein anderes Vorzeichen als $w_{1,-1}$ und $w_{1,1}$ und ist betragsmäßig maximal.

Für den Beweis des nachfolgenden Satzes benötigen wir noch eine Aussage über das Verhalten der Stammfunktion des dualen Wavelets $\widetilde{\psi}$.

7.23 Lemma: Seien $\widetilde{\psi}$ das duale Wavelet von Grad $\kappa, \widetilde{\kappa}$ aus Satz 7.14 und $\widetilde{\Psi}$ die Stammfunktion von $\widetilde{\psi}$. Dann hat $\widetilde{\Psi}$ an der Stelle n_p ein globales Maximum.

Beweis: Wegen der Antisymmetrie von $\widetilde{\psi}$ und der Maximalität von $\int_{T_1}^{n_p} \widetilde{\psi}(t)dt$ folgt

$$\widetilde{\Psi}(n_p) = \int_{-\infty}^{n_p} \widetilde{\psi}(t)dt = \int_{T_1}^{n_p} \widetilde{\psi}(t)dt = \max .$$

□

7.24 Satz: Sei $f : \mathbb{R} \to \mathbb{R}$ modelliert wie in (7.7) und sei $\widetilde{\psi}$ das duale Wavelet vom Grad $\kappa, \widetilde{\kappa}$ aus Satz 7.14. Sei

$$|\alpha_3| > 2^{-r-1} \left| \frac{\widetilde{\Psi}_{r,0}(\xi)}{\widetilde{\Psi}_{r,0}(0)} (\alpha_1 - \alpha_2) \right| \tag{7.14}$$

mit $\xi \in [0, 2^{-r-1}]$, so dass gilt

$$2^{-r-1}\widetilde{\Psi}_{r,0}(\xi) = \int_0^{2^{-r-1}} \widetilde{\Psi}_{r,0}(t)dt.$$

Dann können wir eine nicht stetige Stelle von f durch mindestens zwei aufeinander folgende, signifikant große Waveletkoeffizienten identifizieren. Falls zwei Waveletkoeffizienten vorliegen, haben beide dasselbe Vorzeichen. Drei Koeffizienten können alternierende Vorzeichen haben, bei vier Koeffizienten haben jeweils die beiden mittleren und die beiden äußeren dasselbe Vorzeichen.

Beweis: Seien $f : \mathbb{R} \to \mathbb{R}$ und $t_J = 0$ wie in (7.7). Wir verwenden dieselbe Skalierung r wie im Beweis des Satzes 7.20 sowie den daraus resultierenden Koeffizienten $k_1 = 0$. Dann gilt

$$\begin{aligned} w_{r,0} &= \int_{I_{r,0}} f(t)\widetilde{\psi}_{r,0}(t)dt \\ &= \underbrace{\alpha_1 \int_{-\infty}^{0} t \cdot \widetilde{\psi}_{r,0}(t)dt}_{=A} + \underbrace{\alpha_2 \int_0^{\infty} t \cdot \widetilde{\psi}_{r,0}(t)dt}_{=B} + \underbrace{\alpha_3 \int_0^{\infty} \widetilde{\psi}_{r,0}(t)dt}_{=C} . \end{aligned}$$

Die Ausdrücke A und B erhalten wir auf dieselbe Weise wie im Beweis des Satzes 7.20. Für C folgt

$$C = 2^{-r}\alpha_3 \left(\lim_{t\to\infty} \widetilde{\Psi}_{r,0}(t) - \widetilde{\Psi}_{r,0}(0) \right) = -2^{-r}\alpha_3 \widetilde{\Psi}_{r,0}(0).$$

Damit erhalten wir

$$w_{r,0} = 2^{-r}(\alpha_1 - \alpha_2) \int_0^{2^{-r-1}} \widetilde{\Psi}_{r,0}(t)dt - 2^{-r}\alpha_3 \widetilde{\Psi}_{r,0}(0).$$

Analog können wir $w_{r,-1}$ berechnen. Wegen $2^{-r}(n_p - 1) = -2^{-r-1} < 0$ ergibt sich, mit Hilfe einer Variablentransformation, ähnlich wie im Beweis des Satzes 7.20,

$$w_{r,-1} = 2^{-r}(\alpha_2 - \alpha_1) \int_{2^{-r-1}}^{2^{-r}} \widetilde{\Psi}_{r,0}(t)dt + D$$

mit

$$D = \alpha_3 \int_0^{\infty} \widetilde{\Psi}_{r,-1}(t)dt = -2^{-r}\alpha_3 \widetilde{\Psi}_{r,-1}(0) = -2^{-r}\alpha_3 \widetilde{\Psi}_{r,0}(2^{-r}).$$

Wir erhalten

$$w_{r,-1} = 2^{-r}(\alpha_2 - \alpha_1) \int_{2^{-r-1}}^{2^{-r}} \widetilde{\Psi}_{r,0}(t)dt - 2^{-r}\alpha_3 \widetilde{\Psi}_{r,0}(2^{-r}).$$

Nun ist 2^{-r-1} der Symmetriepunkt von $\widetilde{\Psi}_{r,0}$, also gilt $\widetilde{\Psi}_{r,0}(0) = \widetilde{\Psi}_{r,0}(2^{-r})$. Nach Voraussetzung (7.14) ergibt sich

$$w_{r,0} \begin{cases} > 0 & \text{falls } \alpha_3 < 0 \\ < 0 & \text{falls } \alpha_3 > 0. \end{cases}$$

Analoges Verhalten gilt für $w_{r,-1}$.
Gilt $0 \in I_{r,1}$ und $0 \in I_{r,-2}$, so sind die Koeffizienten $w_{r,1}$ und $w_{r,-2}$ ungleich Null. Mit einer ähnlichen Rechnung erhalten wir die Gleichung $\operatorname{sgn} w_{r,1} = \operatorname{sgn} w_{r,-2}$. Insgesamt liegt uns ein Koeffizientenpaar $(w_{r,0}, w_{r,-1})$ mit gleichen Vorzeichen vor. Abhängig von den äußeren Bedingungen kann zusätzlich ein weiteres Paar $(w_{r,1}, w_{r,-2})$ existieren, das ebenfalls gleiche Vorzeichen hat.
Verändern wir die Voraussetzung der Modellierung (7.7) und setzen $t_J = 2^{-r}n_p = 2^{-r-1}$, so ergibt sich ein anderes Verhalten der Waveletkoeffizienten. Wir sehen sofort, dass wir dieselbe Skalierung r und damit auch $k_1 = 0$

verwenden können. Für $w_{r,0}$ erhalten wir

$$w_{r,0} = \underbrace{\alpha_1 \int_{-\infty}^{2^{-r-1}} t \cdot \widetilde{\psi}_{r,0}(t)dt}_{=A} + \underbrace{\alpha_2 \int_{2^{-r-1}}^{\infty} t \cdot \widetilde{\psi}_{r,0}(t)dt}_{=B} + \underbrace{\alpha_3 \int_{2^{-r-1}}^{\infty} \widetilde{\psi}_{r,0}(t)dt}_{=C}.$$

Im Einzelnen ist

$$\begin{aligned} A &= 2^{-r}\alpha_1 \left(\left[t\widetilde{\Psi}_{r,0}(t) \right]_{-\infty}^{2^{-r-1}} - \underbrace{\int_{-\infty}^{2^{-r-1}} \widetilde{\Psi}_{r,0}(t)dt}_{=0 \text{ nach Lemma 7.19}} \right) \\ &= \alpha_1 2^{-2r-1} \widetilde{\Psi}_{r,0}(2^{-r-1}) \end{aligned}$$

und

$$B = -\alpha_2 2^{-2r-1} \widetilde{\Psi}_{r,0}(2^{-r-1}) \text{ sowie } C = -\alpha_2 2^{-r} \widetilde{\Psi}_{r,0}(2^{-r-1}).$$

Damit folgt

$$\begin{aligned} w_{r,0} &= 2^{-r} \widetilde{\Psi}_{r,0}(2^{-r-1}) \left[2^{-r-1}\alpha_1 - 2^{-r-1}\alpha_2 - \alpha_3 \right] \\ w_{r,-1} &= 2^{-r} \widetilde{\Psi}_{r,-1}(2^{-r-1}) \left[2^{-r-1}\alpha_1 - 2^{-r-1}\alpha_2 - \alpha_3 \right] \\ w_{r,1} &= 2^{-r} \widetilde{\Psi}_{r,1}(2^{-r-1}) \left[2^{-r-1}\alpha_1 - 2^{-r-1}\alpha_2 - \alpha_3 \right]. \end{aligned}$$

Mit $\widetilde{\Psi}_{r,-1}(2^{-r-1}) = \widetilde{\Psi}_{r,0}(0)$ sowie $\widetilde{\Psi}_{r,1}(2^{-r-1}) = \widetilde{\Psi}_{r,0}(2^{-r})$ folgt nach Lemma 7.23, dass $|w_{r,0}|$ maximal ist. Wegen der Symmetrie von $\widetilde{\Psi}$ gilt weiter $\operatorname{sgn} w_{r,-1} = \operatorname{sgn} w_{r,1}$. Über das Vorzeichenverhalten von $w_{r,0}$ und $(w_{r,1}, w_{r,-1})$ können wir allgemein keine näheren Angaben machen.
Mit einer ähnlichen Variablentransformation wie im Beweis des Satzes 7.20 können wir zeigen, dass die Koeffizienten $w_{r,k}$ für $t_J \notin I_{r,k}$ wie 2^{-rn} abfallen, falls $f \in \mathcal{C}_{00}^{n+1}(\mathbb{R}\backslash\{t_J\})$ gilt, während die Koeffizienten $w_{r,j}$, $j \in \{-2,-1,0,1\}$, eine Abfallrate von $2^{-\frac{r}{2}}$ haben. □

7.25 Bemerkung:

(1) Wir erhalten für

$$k_J = \begin{cases} \frac{J}{2} & \text{für gerades } J \\ \frac{J-1}{2} & \text{für ungerades } J \end{cases}$$

an einer unstetigen Stelle bei t_J eine Sequenz von vier bzw. drei Waveletkoeffizienten.

(2) Liegen drei alternierende Waveletkoeffizienten vor, so haben wir für verschiedene Konstellationen von $\alpha_1, \alpha_2, \alpha_3$ und die Skalierung $r = 1$ folgende Vorzeichen der Waveletkoeffizienten.

	$\alpha_1 \leq 0$ $\alpha_2 > 0$ $\alpha_3 > 0$	$\alpha_1 \leq 0$ $\alpha_2 > 0$ $\alpha_3 < 0$	$\alpha_1 \leq 0$ $\alpha_2 < 0$ $\alpha_3 > 0$	$\alpha_1 \leq 0$ $\alpha_2 < 0$ $\alpha_3 < 0$	$\alpha_1 > 0$ $\alpha_2 \geq 0$ $\alpha_3 > 0$	$\alpha_1 > 0$ $\alpha_2 \geq 0$ $\alpha_3 < 0$	$\alpha_1 > 0$ $\alpha_2 \leq 0$ $\alpha_3 > 0$	$\alpha_1 > 0$ $\alpha_2 \leq 0$ $\alpha_3 < 0$
w_{1,k_J-1}	> 0	< 0	> 0	< 0	> 0	< 0	> 0	< 0
w_{1,k_J}	> 0	> 0	< 0	> 0	< 0	> 0	< 0	> 0
w_{1,k_J+1}	> 0	< 0	> 0	< 0	> 0	< 0	> 0	< 0

Mit Hilfe dieser Aussage sind wir in der Lage, Singularitäten zu identifizieren. Wir nutzen in der Implementierung für *WinWerth®* diese Identifizierung zur Überprüfung, da wir die Kontur schon bei der Datenaufbereitung auf Unstetigkeiten testen.
Wie bereits erwähnt, eignen sich für solche Untersuchungen biorthogonale Spline-Wavelets mit einer geringer Anzahl an Nullstellen, wie zum Beispiel solche mit $\kappa = 3$ und $\widetilde{\kappa} = 3$, welche wir für unsere Tests verwendet haben.

7.5 Zusammenspiel der Algorithmen zur Eckenerkennung

Bevor wir uns der Interpolation der Konturen zuwenden, erörtern wir das Zusammenspiel der beiden hier vorgestellten Algorithmen zur Eckenbestimmung. Wir benutzen in den Verfahren für *Werth Messtechnik GmbH* beide Algorithmen und gehen dabei wie folgt vor. Wir wenden zuerst die geometrische Ermittlung der Ecken an. Dabei sind die Parameter so eingestellt, dass tendenziell mehr Ecken gefunden werden als existieren. Es entsteht die Menge E_1. Die Eckenbestimmung mit Hilfe der Wavelets liefert die Eckenmenge E_2. Für die Berechnung der finalen Eckenmenge E unterscheiden wir folgende Fälle.

(1) Es gilt $E_2 \subseteq E_1$, dann setzen wir $E = E_2$.

(2) Es ist $E_1 \subseteq E_2$. Auch in diesem Fall wählen wir $E = E_2$.

(3) In allen anderen Fällen setzen wir $E = E_1 \cup E_2$.

Mit anderen Worten, ist E_2 in E_1 enthalten oder umgekehrt, so ist stets die durch Wavelets erzeugte Menge E_2 die maßgebende Menge. In allen anderen Fällen haben wir einen undefinierten Zustand und wir verwenden alle

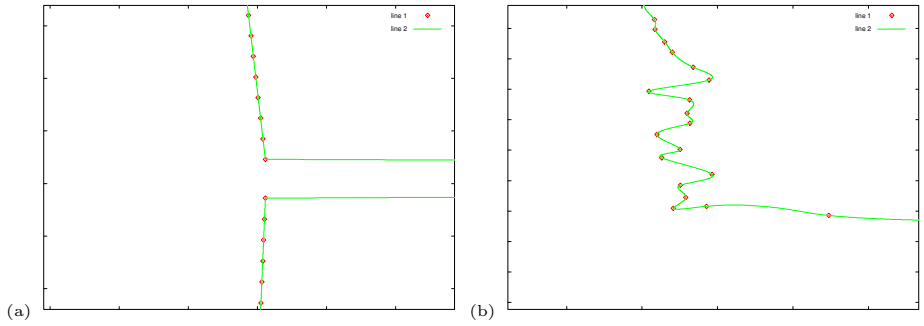

Abbildung 24: (a) Interpolierende Splinekurve mit Eckenerkennung (vgl. Abb. 13 (a)) (b) Interpolierende Splinekurve mit Eckenerkennung (vgl. Abb. 13 (b))

gefundenen Ecken. Dabei müssen wir die finale Menge E einer Überprüfung hinsichtlich der Lage der Ecken unterziehen, da nach Definition keine zwei Ecken innerhalb von drei Konturpunkten liegen dürfen.
Wenden wir den Algorithmus zur Eckenerkennung auf die Konturen aus Abbildung 13 an, so erhalten wir folgende Konturen, abgebildet in Abbildung 24.

7.6 Interpolation der Konturen

Nachdem wir im letzten Abschnitt die Ecken, also die nicht differenzierbaren Stellen, einer Kontur gefunden haben, können wir uns jetzt der Interpolation zuwenden.
Im ersten Schritt, vor der eigentlichen Interpolation, müssen wir die Kontur verschiedenen Untersuchungen unterziehen, welche für die spätere Verarbeitung wichtig sind.

- Wir untersuchen, ob die Kontur K aus mehreren Teilkonturen besteht. Die einzelnen Teilkonturen verbinden wir mit Geraden, die genau 5 Punkte enthalten. Die Festlegung, genau 5 Punkte für die Verbindungsgerade zu benutzen, beruht auf Erfahrungswerten. Wir können wegen der Gestalt der Trennungskonstante (7.15) später die Verbindungsstellen wieder identifizieren.

- Wir überprüfen alle Konturpunkte $\mathbf{v}_j$ auf Redundanz. Dazu eliminieren wir alle $\mathbf{v}_{j+1} \in K$ für die gilt $\mathbf{v}_j = \mathbf{v}_{j+1}$, $j = 0, \ldots, N-2$. Solche Punktepaare würden ungewollte mehrfache Knoten verursachen. Es ist aber $\mathbf{v}_j = \mathbf{v}_k$ für $k \notin \{j-1, j+1\}$ erlaubt.

Wir beschreiben den gesamten Interpolationsprozess in dem nachfolgenden Algorithmus.

7.26 Algorithmus: Sei $K = \{\mathbf{v}_j \mid j = 0, \ldots, N-1\} \subset \mathbb{R}^d$ eine Kontur. Das Ziel des Algorithmus ist die Berechnung eines Kontrollpolygons $\mathbf{D} = \{\mathbf{D}_0, \ldots, \mathbf{D}_{N-1}\}$, einer Knotenmenge $T_{m,N-1}$ und einer Menge von Abszissen $A = \{a_0, \ldots, a_{N-1}\}$, so dass gilt

$$N_{m,T}\mathbf{D}(a_j) = \mathbf{v}_j \text{ für } j = 0, \ldots, N-1.$$

(1) Wir entfernen alle Konturpunkte $\mathbf{v}_j$, die $\mathbf{v}_j = \mathbf{v}_{j+1}$ erfüllen. Wir erzeugen also eine neue Kontur $K' = \{\mathbf{v}_j \mid j = 0, \ldots, N'-1\} \subseteq K$.

(2) Wir überprüfen, ob es sich bei K' um eine zusammenhängende Kontur oder eine aus mehreren Teilkonturen zusammengesetzte Kontur handelt. Hierfür bestimmen wir

$$\varepsilon_{TK} = \theta_1 \left[\frac{1}{\theta_2} \sum_{j=0}^{\theta_2 - 1} \|\mathbf{v}_j - \mathbf{v}_{j+1}\|_2 \right]. \tag{7.15}$$

Gilt für $d_j = \|\mathbf{v}_j - \mathbf{v}_{j+1}\|_2 > \varepsilon_{TK}$, so werden fünf zusätzliche Punkte eingefügt, die äquidistant auf der Strecke $\mathbf{v}_j + t(\mathbf{v}_{j+1} - \mathbf{v}_j)$, $t \in (0,1)$, liegen. Diese Vorgehensweise ist notwendig, da wir im Falle mehrerer Teilkonturen, die zu einer Kontur zusammengesetzt werden, die Teilkonturen mit linearen Teilstücken verbunden haben wollen. Wir erhalten eine neue Gesamtkontur $K'' = \{\mathbf{v}_j \mid j = 0, \ldots, N''-1\} \supseteq K'$.
Wir benutzen für den Rest des Algorithmus die Bezeichnungen K und N für diese Kontur.
Für die Faktoren θ_1 und θ_2 können wir je nach Abstand der Konturpunkte und der Teilkonturen verschiedene Werte verwenden. Bei der Implementierung für *Werth Messtechnik GmbH* wurden die in der Koordinatenmesstechnik üblichen Werte $\theta_1 = 5$ sowie $\theta_2 = 20$ gesetzt, vgl. [Gal03].

(3) Wir bestimmen die Ecken von K. Dazu wenden wir zunächst Algorithmus (7.4) an und erhalten eine Menge E_1 von Ecken. Mit Hilfe von Algorithmus (7.11) bekommen wir die Eckenmenge E_2. Die finale Eckenmenge E bestimmen wir mit Hilfe der Vorgehensweise, die auf Seite 97 beschrieben ist.

(4) Im nächsten Schritt wollen wir die Knotenmenge $T^*_{m,N-1}$ angeben. Wir berechnen die Knotenmenge parametrisiert nach Bogenlänge, da wir kubische Splines verwenden, also nicht davon ausgehen, dass die Konturpunkte linear verbunden sind. Wir setzen $t_0 = \cdots = t_m = 0$ und errechnen

$$t_j = t_{j-1} + \sqrt{\|\mathbf{v}_{j-2} - \mathbf{v}_{j-3}\|_2} \text{ für } j = m, \ldots, N. \tag{7.16}$$

Dann ist $t_N = \cdots = t_{N+m}$. Gilt allerdings $j - 1 \in E$, so bestimmen wir t_j gemäß (7.16) und setzen $t_j = \cdots = t_{j+m-1}$ (Modellierung einer Ecke). Anschließend berechnen wir t_{j+m} als

$$t_{j+m} = t_j + \sqrt{\|\mathbf{v}_{j-1} - \mathbf{v}_{j-2}\|_2}.$$

Die gewünschte Knotenmenge ist

$$T^*_{m,N-1} = \{t_j \mid j = 0, \ldots, N + m\}.$$

(5) Jetzt sind wir in der Lage, die Abszissen $A = \{a_j \mid j = 0, \ldots, N - 1\}$, zu berechnen, die dem Satz von *Schönberg-Whitney* 3.20 genügen und uns damit eine eindeutige Interpolation gewährleisten. Wir erhalten die Menge der Abszissen A gemäß Lemma 3.21 durch

$$a_j = \frac{1}{m+2} \sum_{k=0}^{m+1} t_{j+k} \text{ für } j = 0, \ldots, N - 1. \tag{7.17}$$

(6) Mit diesen Informationen können wir die Kollokationsmatrix $\mathbf{M}$ bestimmen. Nach Satz 3.23 ist $\mathbf{M}$ nicht nur total positiv sondern auch $(m + 1, m + 1)$-bandiert. Das heißt, wir besetzten zwar die komplette Matrix $\mathbf{M}$, außerhalb der $m + 1$ Nebendiagonalen sind aber alle Einträge gleich Null. Allgemein gilt für $T = T^*_{m,N-1}$

$$\mathbf{M} = \begin{bmatrix} N_0^m(a_0 \mid T) & \ldots & N_{N-1}^m(a_0 \mid T) \\ \vdots & \ddots & \vdots \\ N_0^m(a_{N-1} \mid T) & \ldots & N_{N-1}^m(a_{N-1} \mid T) \end{bmatrix} \in \mathbb{R}^{N \times N}. \tag{7.18}$$

In unserem Fall besetzen wir lediglich die Vektoren $\mathbf{m}_0, \ldots, \mathbf{m}_{2m+2}$ der-

art, dass gilt

$$\mathbf{m}_{m+1-k} = \begin{bmatrix} N_0^m(a_k \mid T) \\ \vdots \\ N_{N-k-1}^m(a_{N-1} \mid T) \end{bmatrix} \in \mathbb{R}^{N-k},$$

$$\mathbf{m}_{m+1+k} = \begin{bmatrix} N_k^m(a_0 \mid T) \\ \vdots \\ N_{N-1}^m(a_{N-k-1} \mid T) \end{bmatrix} \in \mathbb{R}^{N-k} \tag{7.19}$$

für $k = m+1, m, \ldots, 1$ und

$$\mathbf{m}_{m+1} = \begin{bmatrix} N_0^m(a_0 \mid T) \\ \vdots \\ N_{N-1}^m(a_{N-1} \mid T) \end{bmatrix} \in \mathbb{R}^N.$$

Ist $m \ll N$, so reduziert sich für Splines vom Grad m der erforderliche Speicherplatz von $O(N^2)$ auf $O((2m+3)N)$. Im Falle der kubischen Splines können wir daher mit einem Speicherplatzbedarf von $O(N)$ rechnen.

(7) Der letzte Schritt auf dem Weg zu der interpolierenden Splinekurve ist die Lösung des linearen Gleichungssystem

$$\mathbf{M}\mathbf{D}^{(i)} = \begin{bmatrix} N_0^m(a_0 \mid T) & \ldots & N_{N-1}^m(a_0 \mid T) \\ \vdots & \ddots & \vdots \\ N_0^m(a_{N-1} \mid T) & \ldots & N_{N-1}^m(a_{N-1} \mid T) \end{bmatrix} \begin{bmatrix} d_0^{(i)} \\ \vdots \\ d_{N-1}^{(i)} \end{bmatrix} = \begin{bmatrix} v_0^{(i)} \\ \vdots \\ v_{N-1}^{(i)} \end{bmatrix} \tag{7.20}$$

für $i = 1, \ldots, d$. Da $\mathbf{M}$ eine bandierte, total positive Matrix ist, können wir zur Lösung des Gleichungssystems (7.20) Gauß-Elimination ohne Pivotsuche anwenden. Wir gehen dabei wie folgt vor.

(a) Wir bestimmen die *LU*–Zerlegung von $\mathbf{M}$ mit Hilfe der Gauß-Elimination ohne Pivotsuche und erhalten

$$\mathbf{M} = \mathbf{L}\mathbf{U} = \begin{bmatrix} 1 & * & \ldots & * & & \\ & 1 & * & & \ddots & \\ & & \ddots & \ddots & & * \\ & & & \ddots & * & \vdots \\ & & & & 1 & * \\ & & & & & 1 \end{bmatrix} \begin{bmatrix} * & & & & & \\ \vdots & * & & & & \\ * & & \ddots & & & \\ & \ddots & & & * & \\ & & & * & \ldots & * \end{bmatrix}$$

mit $\mathbf{L}, \mathbf{U} \in \mathbb{R}^{N \times N}$. Nach [Go96] ist $\mathbf{L}$ eine $(0, m+1)$–bandierte und $\mathbf{U}$ eine $(m+1, 0)$–bandierte Matrix.

(b) Durch Anwendung der Vorwärtselimination auf $\mathbf{U}$ ergeben sich die Gleichungssysteme

$$\mathbf{U}\mathbf{y}^{(i)} = \mathbf{v}^{(i)} \text{ für } i = 1, \ldots, d.$$

Da $\mathbf{U}$ eine untere Dreiecksmatrix ist, können wir die obigen Gleichungssysteme direkt lösen.

(c) Wir verwenden die Lösungen $\mathbf{y}^{(1)}, \ldots, \mathbf{y}^{(d)}$ der Gleichungssysteme aus (b) für die Rückwärtssubstitution und erhalten die gewünschten Koordinaten des Kontrollpolygons

$$\mathbf{L}\mathbf{D}^{(i)} = \mathbf{y}^{(i)} \text{ für } i = 1, \ldots, d.$$

Auch diese Gleichungssysteme lassen sich durch die Gestalt von $\mathbf{L}$ direkt lösen.

(8) Wir speichern das Kontrollpolygon $\mathbf{D} = \{\mathbf{D}_0, \ldots, \mathbf{D}_{N-1}\}$, die Knotenfolge $T^*_{m,N-1} = \{t_0, \ldots, t_{N+m}\}$ und die Menge der Abszissen $A = \{a_0, \ldots, a_{N-1}\}$ für die weitere Verwendung.

Unsere zu untersuchende Kontur K liegt jetzt als interpolierender Spline vor. Der nächste Schritt ist die Berechnung der minimalen und maximalen Abstände bzw. der minimalen und maximalen Punkte eines Regelgeometrieelements und der Kontur. Wir berechnen diese Abstände bzw. Punkte mit Hilfe eines modifizierten Newton-Verfahrens und des Rank-1-Verfahrens von Broyden. Das auf dem Prinzip der konjugierten Gradienten basierte Verfahren von Pollack-Ribière wird auf Grund von Instabilität im Bereich der Ecken nicht angewandt. Alle obigen Verfahren benötigen für eine effiziente und schnelle Durchführung einen geeigneten Startwert. Die Minimierungsverfahren und die Startwertbestimmung beschäftigen uns in den nachfolgenden Abschnitten.
Vorher führen wir eine Laufzeit- und Speicherplatzuntersuchung der einzelnen Schritte des Algorithmus 7.26 durch.

(1) Die Laufzeit der Suche nach doppelten Punkten können wir mit $O(N)$ angeben und der Speicherplatzbedarf liegt bei N, da wir den Konturspeicherplatz weiterhin verwenden können.

(2) Sowohl die Laufzeit als auch der Speicherplatzbedarf liegen bei $O(N)$.

(3) Wie wir in den Abschnitten 7.2 und 7.3 nachgewiesen haben, benötigen wir insgesamt

$$\begin{aligned} &N(20d+10)-(50d+13)+4d(M+N)\,(|h|+1)= \\ &=N\,(10+4d(|h|+6))+4dM(|h|+1)-(50d+13) \end{aligned}$$

Rechenoperationen und der Speicherplatzbedarf liegt bei

$$2N-3+d(N+M)=N(d+2)+(dM-3).$$

(4) Für die Berechnung der Knotenmenge $T_{m,N-1}$ benötigen wir insgesamt $(3d+2)(N-m+1)$ Operationen und $N+m+1$ Speicherplätze.

(5) Der Aufwand für die Berechnung der Abszissen liegt bei $N(m+3)$ Rechenoperationen und wir benötigen N Speicherplätze.

(6) Zunächst bestimmen wir den Aufwand zur Auswertung eines B-Splines $N_k^m(x|T)$. Auf Grund der Rekursionsvorschrift (3.2) sind im worst case Fall, falls alle B-Splines kleinerer Ordnung ausgewertet werden müssen, $9d\frac{m(m+1)}{2}$ Rechenoperationen notwendig.
Wie beschrieben behandeln wir statt der Matrix $\mathbf{M}$ nur die Nebendiagonalen $\mathbf{m}_0,\ldots,\mathbf{m}_{2m+2}$. Damit gilt

Diagonale	Einträge	Operationen
Hauptdiagonale $\mathbf{m}_{m+1}$	N	$9dN(\frac{m(m+1)}{2})$
Nebendiagonalen $\mathbf{m}_m,\mathbf{m}_{m+2}$	$N-1$	$9d(N-1)(\frac{m(m+1)}{2})$
$\vdots$	$\vdots$	$\vdots$
Nebendiagonalen $\mathbf{m}_0,\mathbf{m}_{2m+2}$	$N-m-1$	$9d(N-m-1)(\frac{m(m+1)}{2})$

Zusammen ergeben sich

$$9d\left(\frac{m(m+1)}{2}\right)(N(2m-1)-m(m+1))$$

Rechenoperationen und der Speicherplatzbedarf liegt bei

$$(2N-m)(m+1)+N.$$

(7) Bei der Lösung des linearen Gleichungssystems nutzen wir die Eigenschaften der Matrix $\mathbf{M}$ aus. Uns liegt eine $(m+1,m+1)$–bandierte,

total positive Matrix vor und wir können einen passenden Algorithmus anwenden. Wir benutzen ein Verfahren für bandierte Matrizen aus [Go96] und erhalten folgende Laufzeiten. Die LU–Zerlegung benötigt $2(m+1)^2dN$ Operationen, die Vorwärtselimination und die Rückwärtssubstitution brauchen jeweils $2(m+1)dN$ Rechenschritte. Dies ergibt

$$2dN(m+1)(m+3)$$

Rechenoperationen. Die Matrizen $\mathbf{L}$ und $\mathbf{U}$ können wir zusammen speichern. $\mathbf{L}$ ist $(0, m+1)$–bandiert und $\mathbf{U}$ ist $(m+1, 0)$–bandiert, also benötigen wir insgesamt $(2N-m)(m+1)+N$ Speicherplätze. Hinzu kommt noch der Platz für die Ergebnisse. Da wir $\mathbf{y}$ durch $\mathbf{D}$ überschreiben, liegt der Bedarf an Speicherplatz bei

$$(2N-m)(m+1)+N(d+1).$$

Bezeichnen wir die Anzahl der Rechenoperation für die Interpolation mit Op_{Int} und den Speicherplatzbedarf mit Sp_{int}, so gilt

$$\begin{aligned} Op_{Int} &= N\left(10+4d(|h|+6)\right)+4dM(|h|+1)-(50d+13) \\ &\quad +(3d+2)(N-m+1)+N(m+3)+2dN(m+1)(m+3) \\ &\quad +9d\left(\frac{m(m+1)}{2}\right)\left(N(2m-1)-m(m+1)\right) \\ &= N\left[9dm^3+\frac{13}{2}dm^2+m\left(\frac{7}{2}d+1\right)+d\left(4\,|h|+33\right)+13\right] \\ &\quad +4dM(|h|+1)+C(m,d,|h|). \end{aligned}$$

Mit unseren Annahmen an d, m, $|h|$ sowie M erhalten wir

$$O(N+\log_2 N)=O(N).$$

Der Speicherplatzbedarf können wir wie folgt angeben

$$\begin{aligned} Sp_{Int} &= N(d+2)+(dM-3)+N+m+1+N+(2N-m)(m+1) \\ &\quad +N+(2N-m)(m+1)+N(d+1) \\ &= N(2d+4m+10)+(dM-2m^2-m-2). \end{aligned}$$

Damit ergibt sich ein Speicherbedarf von $O(N+\log_2 N)=O(N)$. Wir können die folgende Aussage formulieren.

7.27 Korollar: Sei $K=\{\mathbf{v}_j \mid j=0,\ldots,N-1\}\subset \mathbb{R}^d$ eine Kontur. Seien m der Grad des interpolierenden Splines, M die Anzahl der berechneten Skalen und $\{h_k\}_{k\in\mathbb{Z}}$ die Verfeinerungsfolge. Die Laufzeit des Algorithmus 7.26 beträgt $O(N)$, der Speicherbedarf liegt ebenfalls bei $O(N)$.

7.7 Newton-Verfahren

In diesem Abschnitt wollen wir uns mit der tatsächlichen Berechnung des minimalen (bzw. maximalen) Punktes einer Kontur und eines Regelgeometrieelements beschäftigen. Beide hier vorgestellten Verfahren finden ihren Ursprung in dem gewöhnlichen Verfahren von Newton, das wir zunächst diskutieren.
Wir betrachten eine Funktion

$$f : \mathbb{R}^k \to \mathbb{R} \text{ mit } f(\mathbf{x}) = \|g(\mathbf{x})\|_2^2$$

und suchen $\mathbf{x}_* \in \mathbb{R}^k$ mit $f(\mathbf{x}_*) = \min$. Die Vorgehensweise für die Suche nach dem Maximum ist analog, in diesem Fall suchen wir $\mathbf{x}_+ \in \mathbb{R}^d$, so dass $-f(\mathbf{x}_+) = \min$ gilt. Daher beschränken wir uns im Folgenden auf die Suche nach einem Minimum.
Wir suchen also, ausgehend von einem Startwert, die Stelle $\mathbf{x}$, an der die Ableitung $\mathbf{F}$ von f Null ist. Es muss demnach

$$\mathbf{F}(\mathbf{x}) = \nabla f(\mathbf{x}) = \mathbf{0} \in \mathbb{R}^k$$

gelten. Für die Suche nach einer Nullstelle von $\mathbf{F}$ können wir das Verfahren von Newton einsetzen. Wir fangen mit einem Startwert $\mathbf{x}_0 \in \mathbb{R}^k$ an und berechnen

$$\mathbf{x}_{j+1} = \mathbf{x}_j - \frac{\mathbf{F}(\mathbf{x}_j)}{\mathbf{F}'(\mathbf{x}_j)} = \mathbf{x}_j - (\mathbf{J}_\mathbf{F}(\mathbf{x}_j))^{-1}\mathbf{F}(\mathbf{x}_j),\ j = 0, 1, 2, \dots . \tag{7.21}$$

Dabei ist $\mathbf{J}_\mathbf{F}(\mathbf{x}_j) = \mathbf{F}'(\mathbf{x}_j)$ die so genannte *Hesse-Matrix von* f bzw., *Jacobi-Matrix von* $\mathbf{F}$, welche die folgende Gestalt hat

$$\mathbf{J}_\mathbf{F}(\mathbf{x}_j) = \begin{bmatrix} \frac{\partial^2}{\partial x_1^2} f & \cdots & \frac{\partial^2}{\partial x_1 \partial x_k} f \\ \vdots & \ddots & \vdots \\ \frac{\partial^2}{\partial x_k \partial x_1} f & \cdots & \frac{\partial^2}{\partial x_k^2} f \end{bmatrix} = \begin{bmatrix} \frac{\partial}{\partial x_1} F_1 & \cdots & \frac{\partial}{\partial x_k} F_1 \\ \vdots & \ddots & \vdots \\ \frac{\partial}{\partial x_1} F_k & \cdots & \frac{\partial}{\partial x_k} F_k \end{bmatrix} \tag{7.22}$$

mit $\mathbf{F} = (F_1, \cdots, F_k)$. Für ein Verfahren, definiert wie in (7.21), erhalten wir die folgende Konvergenzaussage.

7.28 Satz: Sei $\mathbf{F} \in \mathcal{C}^2(\mathbb{R}^k)^k$ und $\mathbf{x}_* \in \mathbb{R}^k$ eine einfache Nullstelle von $\mathbf{F}$, das heißt $\mathbf{F}(\mathbf{x}_*) = \mathbf{0}$ und $\det \mathbf{J}_\mathbf{F}(\mathbf{x}_*) \neq 0$. Sei $(\mathbf{x}_k)_k$ eine Folge, definiert wie in (7.21). Dann existiert eine offene Menge $U \subset \mathbb{R}^k$ mit $\mathbf{x}_* \in U$, so dass gilt

$$\lim_{j \to \infty} \mathbf{x}_j = \mathbf{x}_* \text{ für } \mathbf{x}_0 \in U.$$

Das gewöhnliche Newton-Verfahren hat einige Nachteile. Zunächst einmal ist es stets von einem guten Startwert abhängig, denn nur in diesem Fall ist eine quadratische Konvergenzgeschwindigkeit möglich. Bei einem ungünstigen Startwert kann das Verfahren sogar divergieren. Des weiteren muss in jedem Schritt die Jacobi-Matrix berechnet werden, die Funktion f muss dazu zweimal differenzierbar sein.
Wir stellen zwei auf dem gewöhnlichen Newton-Verfahren basierende Methoden vor, welche die obigen Nachteile zwar nicht komplett beheben, aber zumindest abschwächen. Das modifizierte Newton-Verfahren bietet eine gewisse Globalität, die Berechnung der Jacobi-Matrix bleibt aber bestehen. Das Rang-1-Verfahren von Broyden ersetzt die Berechnung der Jacobi-Matrix durch eine Näherung, das Verfahren wird dadurch in manchen Fällen schneller, liefert aber ungenauere Ergebnisse.
Wir haben bereits erwähnt, dass beide Verfahren sehr stark von einem guten Startwert abhängen. Daher beschäftigen wir uns zunächst mit der Bestimmung der Startwerte für alle unseren Verknüpfungen und stellen anschließend die oben angesprochenen Verfahren vor.

7.8 Startwertberechnung

Die Berechnung der Startwerte für die nachfolgenden Verknüpfung beruht in allen Fällen auf dem Linearsatz. Wir ermitteln für die Bestimmung des minimalen Punktes jeweils den zu dem jeweiligen Regelgeometrieelement nächsten Punkt der Kontur. Analog bestimmen wir für die Berechnung des maximalen Punktes den von dem Regelgeometrieelement am weitesten entfernten Punkt der Kontur.

7.8.1 Startwertberechnung: Kontur-Gerade

Bei der Berechnung des Startwertes für die Verknüpfung Kontur-Gerade bedienen wir uns des Least-Square-Prinzips. Wir betrachten die Kontur $K = \{\mathbf{v}_j \mid j = 0, \ldots, N-1\} \subset \mathbb{R}^d$ und eine Gerade der Form

$$\mathbf{R}_G(\mu) = \mathbf{g}(\mu) = \mathbf{p} + \mu\mathbf{q} \text{ mit } \mathbf{p}, \mathbf{q} \in \mathbb{R}^d \text{ und } \mu \in \mathbb{R}. \tag{7.23}$$

Wir setzen

$$\mathbf{h}_j(\nu_j) = \mathbf{v}_{j+1} + \nu_j(\mathbf{v}_j - \mathbf{v}_{j+1}) \text{ mit } \nu_j \in \mathbb{R} \text{ für } j = 0, \ldots, N-2 \tag{7.24}$$

und wollen die Parameter μ_j, ν_j so bestimmen, dass gilt

$$\mathbf{g}(\mu_j) = \mathbf{p} + \mu_j\mathbf{q} = \mathbf{v}_{j+1} + \nu_j(\mathbf{v}_j - \mathbf{v}_{j+1}) = \mathbf{h}(\nu_j) \text{ für } j = 0, \ldots, N-2.$$

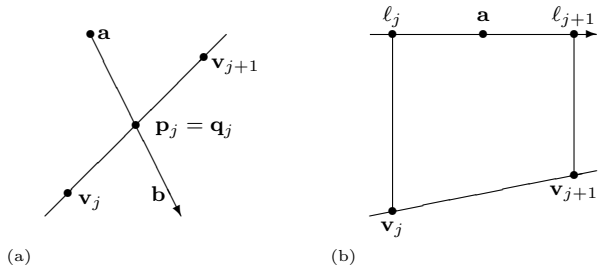

Abbildung 25: (a) Geraden $\mathbf{g}$ und $\mathbf{h}_j$ sind nicht parallel (b) Geraden $\mathbf{g}$ und $\mathbf{h}_j$ sind annähernd parallel, es gilt also $\left|\det \mathbf{C}_j^t \mathbf{C}_j\right| \leq \varepsilon$.

Wir erhalten das überbestimmte Gleichungssystem

$$\mathbf{C}_j \mathbf{z}_j = \left[\begin{array}{cc} \mathbf{q} & \mathbf{v}_{j+1} - \mathbf{v}_j \end{array} \right] \left[\begin{array}{c} \mu_j \\ \nu_j \end{array} \right] = \left[\begin{array}{c} \mathbf{v}_{j+1} - \mathbf{p} \end{array} \right] = \mathbf{c}_j, \tag{7.25}$$

$\mathbf{C}_j \in \mathbb{R}^{d \times 2}$, $\mathbf{z}_j \in \mathbb{R}^2$, $\mathbf{c}_j \in \mathbb{R}^d$ für $j = 0, \ldots, N-2$. Wir lösen (7.25), indem wir die Normalengleichungen

$$\mathbf{C}_j^t \mathbf{C}_j \mathbf{z}_j = \mathbf{C}_j^t \mathbf{c}_j, \text{ für } j = 0, \ldots, N-2 \tag{7.26}$$

lösen. Gilt $\det(\mathbf{C}_j^t \mathbf{C}_j) \approx 0$, so sind die Geraden $\mathbf{g}$ und $\mathbf{h}_j$ annähernd parallel und wir müssen auf eine andere Weise vorgehen. Wir wählen einen Wert $\varepsilon > 0$, abhängig von der Anzahl der Nachkommastellen der Kontur, der die Schwelle darstellt, ab welcher die Least-Square-Berechnung instabil wird. Wir benutzen als minimalen Wert $\varepsilon = 10^{-4}$. Ist die Anzahl der Nachkommastellen $K < 4$, so setzen wir $\varepsilon = 10^{-K+1}$.
Wir betrachten nachfolgend beide Fälle, Beispiele sind in Abbildung 25 dargestellt.

1. $\left|\det(\mathbf{C}_j^t \mathbf{C}_j)\right| > \varepsilon$
   Durch die Lösung von (7.26) erhalten wir die Parameter (μ_j, ν_j) für jedes $j = 0, \ldots, N-2$. Es ergeben sich folgende Möglichkeiten

$$d_j = \begin{cases} \|\mathbf{g}(\mu_j) - \mathbf{h}(\nu_j)\|_2 & \text{für } \nu_j \in [0,1], \mu_j \in \mathbb{R} \\ \|\mathbf{g}(\mu_j) - \mathbf{v}_j\|_2 & \text{für } \nu_j > 1, \mu_j \in \mathbb{R} \\ \|\mathbf{g}(\mu_j) - \mathbf{v}_{j+1}\|_2 & \text{für } \nu_j < 0, \mu_j \in \mathbb{R}. \end{cases} \tag{7.27}$$

   Wir verwenden also in den meisten Fällen die Abstände der Randpunkte $\mathbf{v}_j$ bzw. $\mathbf{v}_{j+1}$ zu der Geraden $\mathbf{g}$.

2. $\left|\det(\mathbf{C}_j^t \mathbf{C}_j)\right| \leq \varepsilon$
   Wir berechnen für die Gerade $\mathbf{h}$, die beiden Abstände $\|\mathbf{v}_j - \mathbf{g}\|_2$ und

$\|\mathbf{v}_{j+1}-\mathbf{g}\|_2$, indem wir die Punkte $\boldsymbol{\ell}_j, \boldsymbol{\ell}_{j+1} \in \mathbf{g}$ bestimmen, so dass die Geraden

$$\mathbf{f}_j : \mathbf{v}_j + \beta_j(\boldsymbol{\ell}_j - \mathbf{v}_j) \text{ mit } \beta_j \in \mathbb{R}$$

$$\mathbf{f}_{j+1} : \mathbf{v}_{j+1} + \beta_{j+1}(\boldsymbol{\ell}_{j+1} - \mathbf{v}_{j+1}) \text{ mit } \beta_{j+1} \in \mathbb{R}$$

senkrecht auf $\mathbf{g}$ stehen und $\mathbf{f}_j$ die Gerade $\mathbf{g}$ in $\boldsymbol{\ell}_j$ und $\mathbf{f}_{j+1}$ die Gerade $\mathbf{g}$ in $\boldsymbol{\ell}_{j+1}$ schneidet. Dann können wir die Schnittpunkte $\boldsymbol{\ell}_j$ und $\boldsymbol{\ell}_{j+1}$ wie folgt darstellen

$$\boldsymbol{\ell}_j = \mathbf{p} + \widetilde{\nu}_j \mathbf{q} \text{ mit } \widetilde{\nu}_j = \frac{\mathbf{q}^t(\mathbf{v}_j - \mathbf{p})}{\mathbf{q}^t\mathbf{q}},$$

$$\boldsymbol{\ell}_{j+1} = \mathbf{p} + \widetilde{\nu}_{j+1} \mathbf{q} \text{ mit } \widetilde{\nu}_{j+1} = \frac{\mathbf{q}^t(\mathbf{v}_{j+1} - \mathbf{p})}{\mathbf{q}^t\mathbf{q}}.$$

Auch wenn die Geraden $\mathbf{g}$ und $\mathbf{h}$ annähernd parallel sind, ist es möglich, dass ein minimaler Punkt vorliegt. Wir überprüfen daher, ob der minimale Abstand am Rand oder im Inneren der Strecke zwischen $\mathbf{v}_j$ und $\mathbf{v}_{j+1}$ liegt. Hierfür verwenden wir den Mittelpunkt $\mathbf{v}_j^* = \mathbf{h}(\frac{1}{2})$ der Strecke $\mathbf{v}_j\mathbf{v}_{j+1}$ und bestimmen

$$\boldsymbol{\ell}_j^* = \mathbf{p} + \widetilde{\nu}_j^* \mathbf{q} \text{ mit } \widetilde{\nu}_j^* = \frac{\mathbf{q}^t(\mathbf{v}_j^* - \mathbf{p})}{\mathbf{q}^t\mathbf{q}},$$

also ist $\boldsymbol{\ell}_j^*$ Fußpunkt des Lotes durch $\mathbf{v}_j^*$ auf die Gerade $\mathbf{g}$. Mit der Bezeichnung

$$d_j^* = \|\mathbf{v}_j^* - \boldsymbol{\ell}_j^*\|_2,\ d_{j,1} = \|\mathbf{v}_j - \boldsymbol{\ell}_j\|_2,\ d_{j,2} = \|\mathbf{v}_{j+1} - \boldsymbol{\ell}_{j+1}\|_2$$

können wir die Lage des minimalen Punktes angeben.
Gilt $d_{j,2} = 2(d_j^* - d_{j,1}) + d_{j,1}$, so existiert kein minimaler Punkt zwischen $\mathbf{v}_j$ und $\mathbf{v}_{j+1}$ und wir erhalten $d_j = \min\{d_{j,1}, d_{j,2}\}$ sowie

$$\mu_j = \begin{cases} \widetilde{\nu}_j & \text{falls } d_j = d_{j,1} \\ \widetilde{\nu}_{j+1} & \text{falls } d_j = d_{j,2} \end{cases} \text{ und } \nu_j = \begin{cases} 0 & \text{falls } d_j = d_{j,1} \\ 1 & \text{falls } d_j = d_{j,2} \end{cases}.$$

Andernfalls gilt

$$\mu_j = \frac{\mathbf{q}^t(\mathbf{h}(\nu_j) - \mathbf{p})}{\mathbf{q}^t\mathbf{q}} \text{ und } \nu_j = -\frac{d_j}{2(d_j^* - d_j)}.$$

In beiden Fällen erhalten wir die Ergebnis-Matrix

$$\mathbf{E} = \begin{bmatrix} d_0 & \mu_0 & \nu_0 \\ \vdots & \vdots & \vdots \\ d_{N-1} & \mu_{N-1} & \nu_{N-1} \end{bmatrix}.$$

Gilt $d_j > 0$ für alle $j = 0, \ldots, N-1$, dann wählen wir

$$j_{min} = \text{argmin}\{d_0, \ldots, d_{N-1}\} \text{ und } j_{max} = \text{argmax}\{d_0, \ldots, d_{N-1}\}.$$

Die minimalen und maximalen Punkte und die minimalen Schnittpunkte müssen nicht eindeutig sein. Falls mehrere minimale oder maximale Punkte existieren, haben wir durch die obige Festlegung den Extremalpunkt mit dem kleinsten Konturindex ausgewählt. Liegen mehrere minimale Schnittpunkte vor, so wird der richtige Schnittpunkt gemäß der folgenden Spezifikation von *Werth Messtechnik GmbH* ausgewählt. Sei $S = \{\mathbf{v}_j \in K \mid d_j = 0\}$ die Menge der Schnittpunkte, dann ist $\mathbf{v}_{min} \in S$ ein minimaler Schnittpunkt falls folgendes gilt.

(1) Es existiert mindestens ein $\mathbf{v}_j \in S$, so dass wir diesen Schnittpunkt als $\mathbf{v}_j = \mathbf{p} + \mu_j \mathbf{q}$ mit $\mu_j > 0$ darstellen können. Dann ist der minimale Schnittpunkt $\mathbf{v}_{min}$ als $\mathbf{v}_{min} = \mathbf{p} + \mu \mathbf{q}$ mit $\mu > 0$ und $\|\mathbf{p} - \mathbf{v}_{min}\|_2 = \min_{\mathbf{v}_j \in S}\{\|\mathbf{p} - \mathbf{v}_j\|_2 \mid \mathbf{x}_j = \mathbf{p} + \mu_j \mathbf{q} \text{ und } \mu_j > 0\}$ definiert.

(2) Existiert kein solcher Schnittpunkt $\mathbf{v}_j \in S$, so ist der minimale Schnittpunkt $\mathbf{v}_{min}$ definiert als $\mathbf{v}_{min} = \mathbf{p} + \mu \mathbf{q}$ mit $\mu < 0$ und $\|\mathbf{p} - \mathbf{v}_{min}\|_2 = \min_{\mathbf{v}_j \in S}\{\|\mathbf{p} - \mathbf{v}_j\|_2 \mid \mathbf{v}_j = \mathbf{p} + \mu_j \mathbf{q} \text{ und } \mu_j < 0\}$.

Mit anderen Worten, da eine Gerade stets eine Richtung hat, wird ein minimaler Schnittpunkt zuerst in der positiven und dann in der negativen Richtung der Geraden gesucht.
In beiden Fällen bestimmen wir auf diese Weise den richtigen Eintrag aus der Ergebnismatrix $\mathbf{E}$. Für die Startwerte $\mathbf{S}_{min}$ und $\mathbf{S}_{max}$ gilt dann

$$\mathbf{S}_{min} = \begin{bmatrix} \mu_{j_{min}} \\ a_{j_{min}} \end{bmatrix} \text{ und } \mathbf{S}_{max} = \begin{bmatrix} \mu_{j_{max}} \\ a_{j_{max}} \end{bmatrix}.$$

Für die Laufzeit- und Speicherplatzberechnung erhalten wir die folgenden Werte. Es sind insgesamt $(N-1)11d$ Operationen durchzuführen und wir benötigen N Speicherplätze. Also betragen die Laufzeit und der Speicherplatzbedarf jeweils $O(N)$.

7.8.2 Startwertberechnung: Kontur-Kreis

Um einen Startwert für die Verknüpfung berechnen zu können, müssen wir zunächst die Darstellung eines Kreises im $\mathbb{R}^d$ festlegen. Unter einem Kreis verstehen wir in diesem Fall den Rand des Kreises.
Seien hierfür $\mathbf{p}_M \in \mathbb{R}^d$ der Mittelpunkt des Kreises, $r_C \in \mathbb{R}$, $r_C > 0$, der Radius und $\mathbf{H}$ die Ebene in $\mathbb{R}^d$, in welcher der Kreis liegt. Dann können wir einen Kreis $\mathbf{R}_C$ im $\mathbb{R}^d$ als Menge darstellen

$$\mathbf{R}_C = B \cap \mathbf{H} \text{ mit } B = \{\mathbf{x} \in \mathbb{R}^d \mid \|\mathbf{p}_M - \mathbf{x}\|_2 = r_C\}.$$

Diese allgemeine Definition des Kreises ist für uns etwas unhandlich. Da wir uns aber auf den Fall $d = 3$ beschränken können, erhalten wir folgende Darstellung für einen Kreis.
Seien $\mathbf{p}_M \in \mathbb{R}^3$, $r_C > 0$ und $\mathbf{z}_1, \mathbf{z}_2 \in \mathbb{R}^3$. Dann können wir einen Kreis $\mathbf{R}_C$ im $\mathbb{R}^3$ durch die Gleichung

$$\mathbf{R}_C(\mu) = \mathbf{p}_M + r_C \left[\sin\mu, \ \cos\mu \right] \left[\mathbf{z}_1, \ \mathbf{z}_2 \right]^t \tag{7.28}$$

mit $\mathbf{z}_1 \perp \mathbf{z}_2$ und $\|\mathbf{z}_1\|_2 = \|\mathbf{z}_2\|_2 = 1$ sowie $\mu \in [0, 2\pi)$ beschreiben.
Die Vektoren $\mathbf{z}_1$ und $\mathbf{z}_2$ definieren die Kreisebene und damit die Lage des Kreises im $\mathbb{R}^3$. $\mathbf{p}_M$ ist der Mittelpunkt von $\mathbf{R}_C$, r_C ist der Radius des Kreises.
Mit der Darstellung (7.28) können wir uns der Startwertbestimmung zuwenden.
Wir bestimmen den Abstand aller Konturpunkte zum Mittelpunkt des Kreises und subtrahieren den Radius von diesem Abstand. Als Startwert für die Bestimmung des minimalen Punkts wählen wir den Konturpunkt, dessen Abstand am nächsten an Null liegt, als Startwert für die Bestimmung des maximalen Punkts wählen wir den Konturpunkt mit dem betragsmäßig größten Abstand. Im Einzelnen gehen wir wie folgt vor.
Sei $K = \{\mathbf{v}_j \mid j = 0, \ldots, N-1\}$ die Kontur und $A = \{a_j \mid j = 0, \ldots, N-1\}$ die Menge der Abszissen zu K. Wir bestimmen für jeden Punkt $\mathbf{v}_j$ den Abstand

$$d_j = \|\mathbf{p}_M - \mathbf{v}_j\|_2$$

und erhalten den Vektor

$$\mathbf{d} = \begin{bmatrix} d_0 \\ \vdots \\ d_{N-1} \end{bmatrix}.$$

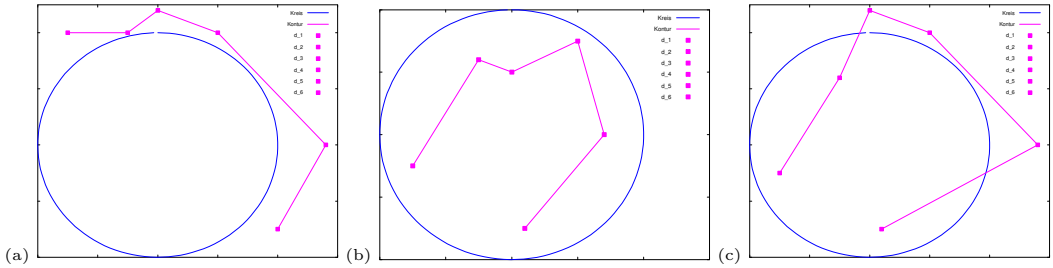

Abbildung 26: (a) Fall (1): Alle Konturpunkte liegen außerhalb von $\mathbf{R}_C$ (b) Fall (2): Alle Konturpunkte befinden sich innerhalb des Kreises $\mathbf{R}_C$ (c) Fall (3): Konturpunkte liegen sowohl innerhalb als auch außerhalb von $\mathbf{R}_C$

Wir unterscheiden folgende drei Fälle, Beispiele sind in Abbildung 26 dargestellt.

(1) Es gilt $d_j > r_C$ für alle $j = 0, \ldots, N-1$. Wir wählen

$$j_{min} = \text{argmin}\{d_0, \ldots, d_{N-1}\}$$

und berechnen zu $\mathbf{v}_{j_{min}}$ den Parameter $\mu_{j_{min}}$, so dass gilt

$$\mathbf{R}_C(\mu_{j_{min}}) \in \mathbf{g}(\nu) \text{ mit } \mathbf{g}(\nu) = \mathbf{p}_M + \nu(\mathbf{v}_{j_{min}} - \mathbf{p}_M) \text{ und } \nu \in [0,1].$$

Damit gilt für den Startwert

$$\mathbf{S}_{min} = \begin{bmatrix} \mu_{j_{min}} \\ a_{j_{min}} \end{bmatrix}.$$

Analog bestimmen wir

$$j_{max} = \text{argmax}\{d_0, \ldots, d_{N-1}\}$$

und berechnen zu $\mathbf{v}_{j_{max}}$ den Parameter $\mu_{j_{max}}$, so dass gilt

$$\mathbf{R}_C(\mu_{j_{max}}) \in \mathbf{g}(\nu) \text{ mit } \mathbf{g}(\nu) = \mathbf{p}_M + \nu(\mathbf{v}_{j_{max}} - \mathbf{p}_M) \text{ und } \nu \in [0,1].$$

Dies ergibt als Startwert für den maximalen Punkt

$$\mathbf{S}_{max} = \begin{bmatrix} \mu_{j_{max}} \\ a_{j_{max}} \end{bmatrix}.$$

(2) Es gilt $d_j < r_C$ für alle $j = 0, \ldots, N-1$. Wir wählen

$$j_{min} = \text{argmax}\{d_0, \ldots, d_{N-1}\}$$

und entsprechend

$$j_{max} = \text{argmin}\{d_0, \ldots, d_{N-1}\}.$$

Dann verfahren wir analog zu Fall (1).

(3) Falls (1) und (2) nicht zutreffen, suchen wir den ersten Wert d_j, so dass $d_{j-1} < r_C$ und $d_j > r_C$ oder umgekehrt $d_{j-1} > r_C$ und $d_j < r_C$ gilt. Dann schneidet die Gerade

$$\mathbf{h}(\nu) = \mathbf{v}_{j-1} + \nu(\mathbf{v}_j - \mathbf{v}_{j-1}) \text{ mit } \nu \in \mathbb{R}$$

den Kreis $\mathbf{R}_C$ in dem Punkt

$$\mathbf{sp} = \mathbf{h}(\nu^*) \text{ mit } \nu^* = \frac{r_C - d_{j-1}}{d_j - d_{j-1}}.$$

Damit können wir den ersten Teil des Startwertes berechnen, denn es gilt

$$a_{j_{min}} = a_{j-1} + \nu^*(a_j - a_{j-1}).$$

Den zweiten Parameter $\mu_{j_{min}}$ des Startwertes bestimmen wir mit der Gleichung $\mathbf{R}_C(\mu_{j_{min}}) = \mathbf{sp}$ und erhalten den Startwert

$$\mathbf{S}_{min} = \begin{bmatrix} \mu_{j_{min}} \\ a_{j_{min}} \end{bmatrix}.$$

Für die Bestimmung des Startwertes des maximalen Punktes ist

$$j_{max} = \text{argmax}\{d_0, \ldots, d_{N-1}\}$$

und wir berechnen wie in (1) und (2) die Parameter $\mu_{j_{max}}$ und $s_{j_{max}}$ und erhalten

$$\mathbf{S}_{max} = \begin{bmatrix} \mu_{j_{max}} \\ a_{j_{max}} \end{bmatrix}.$$

Für die Berechnung der Verknüpfung Kontur-Gerade benötigen wir höchstens $8dN$ Rechenoperationen, können also von einer Laufzeit von $O(N)$ ausgehen. Der Speicherplatzbedarf ist N, also $O(N)$.

7.8.3 Startwertberechnung: Kontur-Ebene

Wir betrachten eine Ebene als einen 2–dimensionalen Unterraum des $\mathbb{R}^d$ und stellen diese als eine Funktion $\mathbf{R}_E : \mathbb{R}^2 \to \mathbb{R}^d$ mit

$$\mathbf{R}_E(\mu^{(1)}, \mu^{(2)}) = \mathbf{p} + \mu^{(1)}\mathbf{q} + \mu^{(2)}\mathbf{r} \text{ mit } \mathbf{p}, \mathbf{q}, \mathbf{r} \in \mathbb{R}^d$$

dar. Wir bezeichnen $\mathbf{p}$ als *Stützvektor* und $\mathbf{q}$, $\mathbf{r}$ als *Richtungsvektoren*. Die Richtungsvektoren $\mathbf{q}$ und $\mathbf{r}$ müssen linear unabhängig sein.
Wir wollen den Abstand der Konturpunkte zu der Ebene $\mathbf{H}$ ermitteln.

Dazu bestimmen wir im Fall $d = 3$ den Normalenvektor $\mathbf{n}_H$ von $\mathbf{H}$ mit Hilfe des Kreuzproduktes als $\mathbf{n}_H = \mathbf{q} \times \mathbf{r}$. Für $d > 3$ müssten wir an dieser Stelle die Erzeugenden des Senkrechtraums $\operatorname{span} \mathbf{H}^{\perp}$ berechnen. Da diese Betrachtung für uns nicht praxisrelevant ist, beschränken wir uns an dieser Stelle auf den Fall $d = 3$.
Mit dem Normalenvektor stellen wir für $j = 0, \ldots, N-1$ die folgenden Gleichungen auf

$$\mathbf{v}_j + \nu_j \mathbf{n}_H = \mathbf{p} + \mu_j^{(1)} \mathbf{q} + \mu_j^{(2)} \mathbf{r},$$

welche die Projektionen von $\mathbf{v}_j$ auf $\mathbf{H}$ darstellen. Diese können wir zu dem linearen Gleichungssystem

$$\mathbf{M}\mathbf{x}_j = \mathbf{f}_j \text{ mit}$$

$$\mathbf{M} = \begin{bmatrix} \mathbf{n}_H & -\mathbf{q} & -\mathbf{r} \end{bmatrix} \in \mathbb{R}^{d \times 3} \text{ und } \mathbf{x}_j = \begin{bmatrix} \nu_j \\ \mu_j^{(1)} \\ \mu_j^{(2)} \end{bmatrix} \in \mathbb{R}^3 \text{ sowie}$$

$$\mathbf{f}_j = \begin{bmatrix} \mathbf{p} - \mathbf{v}_j \end{bmatrix} \in \mathbb{R}^d$$

umformen. Wir erhalten den Vektor

$$\mathbf{x}_j = \begin{bmatrix} \nu_j \\ \mu_j^{(1)} \\ \mu_j^{(2)} \end{bmatrix}$$

als Lösung des Gleichungssystems. Für $d = 3$ kann das obige Gleichungssystem direkt gelöst werden.
Damit sind wir in der Lage, die Abstände d_j für $j = 0, \ldots, N-1$ zu berechnen. Dies geschieht durch

$$d_j = \|\mathbf{v}_j - \mathbf{w}_j\|_2 \text{ mit } \mathbf{w}_j = \mathbf{v}_j + \nu_j \mathbf{n}_H.$$

Die aus dem Berechneten resultierende Ergebnismatrix $\mathbf{E}$ hat die Gestalt

$$\mathbf{E} = \begin{bmatrix} d_0 & \mu_0^{(1)} & \mu_0^{(2)} & a_0 \\ \vdots & \vdots & \vdots & \vdots \\ d_{N-1} & \mu_{N-1}^{(1)} & \mu_{N-1}^{(2)} & a_{N-1} \end{bmatrix}.$$

Wir werten diese aus, indem wir

$$j_{min} = \operatorname{argmin}\{d_0, \ldots, d_{N-1}\} \text{ und } j_{max} = \operatorname{argmax}\{d_0, \ldots, d_{N-1}\}$$

bestimmen. Dann sind

$$\mathbf{S}_{min} = \begin{bmatrix} \mu^{(1)}_{j_{min}} \\ \mu^{(2)}_{j_{min}} \\ a_{j_{min}} \end{bmatrix} \text{ und } \mathbf{S}_{max} = \begin{bmatrix} \mu^{(1)}_{j_{max}} \\ \mu^{(2)}_{j_{max}} \\ a_{j_{max}} \end{bmatrix}.$$

Die Laufzeit beträgt $O(N)$, da wir insgesamt $15dN$ Rechenoperation durchführen müssen. Wie bei den vorherigen Verknüpfungen benötigen wir $O(N)$ Speicherplatz.

7.8.4 Startwertberechnung: Kontur-Kontur

Bei dieser Verknüpfung spielt die zweite Kontur die Rolle des Regelgeometrieelements. Wir starten mit den Konturen

$$K = \{\mathbf{v}_j \mid j = 0, \ldots, N-1\} \subset \mathbb{R}^d,\ K' = \{\mathbf{w}_k \mid k = 0, \ldots, N'-1\} \subset \mathbb{R}^d$$

und den zugehörigen Mengen von Abszissen

$$A = \{a_j \mid j = 0, \ldots, N-1\} \text{ und } B = \{b_k \mid k = 0, \ldots N'-1\}.$$

Als Startwerte für die nachfolgenden Minimierungsverfahren benötigen wir die Abszissen $a_{j_{min}}, b_{k_{min}}$ bzw. $a_{j_{max}}, b_{k_{max}}$, für die gilt

$$\|\mathbf{K}(a_{j_{min}}) - \mathbf{K}'(b_{k_{min}})\|_2 = \min \text{ und } \|\mathbf{K}(a_{j_{max}}) - \mathbf{K}'(a_{k_{max}})\|_2 = \max.$$

Dabei sind $\mathbf{K}, \mathbf{K}' : \mathbb{R} \to \mathbb{R}^d$ die angewendeten Parametrisierungen für die jeweils

$$\mathbf{K}(a_j) = \mathbf{v}_j \text{ und } \mathbf{K}'(b_k) = \mathbf{w}_k$$

ist. Eine Möglichkeit, solche Startwerte zu berechnen, wäre für jedes Paar $\mathbf{v}_j, \mathbf{v}_{j+1} \in K,\ j = 0, \ldots, N-2$ und die Kontur K' die Startwerte für die Verknüpfung Kontur-Gerade zu berechnen. Wir können uns aber leicht überlegen, dass eine solche Vorgehensweise einen Zeitaufwand von $O(NN')$ besitzt und diese Komplexität für zwei große Konturen $(N, N' > 10.000)$ nicht tragbar ist. Daher wenden wir an dieser Stelle ein Divide & Conquer-Verfahren an. Das heißt, wir teilen beide Konturen jeweils in zwei halb so große Teilkonturen, berechnen die Abstände und vereinigen die einzelnen Teilmengen wieder. Dabei werten wir die einzelnen Ergebnisse aus und geben die relevanten Werte zurück. Die Einzelheiten der Vorgehensweise fassen wir in dem nachfolgenden Algorithmus zusammen.

7.29 Algorithmus: Seien $K, K' \subset \mathbb{R}^d$ Konturen wie oben beschrieben und $A, B \subset \mathbb{R}$ die zugehörigen Mengen von Abszissen.

`function` $(abstand, a_{j_{min}}, b_{k_{min}})$ =`KonturKontur`(K, K')
  `if`$(N \leq 10$ `AND` $N' \leq 10)$
    $(abstand, a_{j,min}, b_{k,min})$ = `KonturKonturDirekt`(K, K')
  `else if` $(N \leq 10$ `AND` $N' > 10)$
    $N'_1 = \lfloor N'/2 \rfloor$; $N'_2 = N' - N'_1$;
    $K'_1 \subset K'$ mit $|K'_1| = N'_1$;
    $K'_2 = K' - K'_1$;
    $(abstand_1, a_{j_1,min}, b_{k_1,min})$ =`KonturKontur`(K, K'_1);
    $(abstand_2, a_{j_2,min}, b_{k_2,min})$ =`KonturKontur`(K, K'_2);
    `if`$(abstand_1 < abstand_2)$
      `return`$(abstand_1, a_{j_1,min}, b_{k_1,min})$;
    `else`
      `return`$(abstand_2, a_{j_2,min}, b_{k_2,min})$;
  `else if` $(N > 10$ `AND` $N' \leq 10)$
    $N_1 = \lfloor N/2 \rfloor$; $N_2 = N - N_1$;
    $K_1 \subset K$ mit $|K_1| = N_1$;
    $K_2 = K - K_1$;
    $(abstand_1, a_{j_1,min}, b_{k_1,min})$ =`KonturKontur`(K_1, K');
    $(abstand_2, a_{j_2,min}, b_{k_2,min})$ =`KonturKontur`(K_2, K');
    `if`$(abstand_1 < abstand_2)$
      `return`$(abstand_1, a_{j_1,min}, b_{k_1,min})$;
    `else`
      `return`$(abstand_2, a_{j_2,min}, b_{k_2,min})$;
  `else`
    $N_1 = \lfloor N/2 \rfloor$; $N_2 = N - N_1$;
    $K_1 \subset K$ mit $|K_1| = N_1$;
    $K_2 = K - K_1$;
    $N'_1 = \lfloor N'/2 \rfloor$; $N'_2 = N' - N'_1$;
    $K'_1 \subset K'$ mit $|K'_1| = N'_1$;
    $K'_2 = K' - K'_1$;
    $(abstand_1, a_{j_1,min}, b_{k_1,min})$ =`KonturKontur`(K_1, K'_1);
    $(abstand_2, a_{j_2,min}, b_{k_2,min})$ =`KonturKontur`(K_1, K'_2);
    $(abstand_3, a_{j_3,min}, b_{k_3,min})$ =KonturKontur(K_2, K'_1);
    $(abstand_4, a_{j_4,min}, b_{k_4,min})$ =KonturKontur(K_2, K'_2);
    $j_{min} = min_{0<j<5} abstand_j$;
    `return` $(abstand_{j_{min}}, a_{j_{min}}, b_{k_{min}})$;

```
  endif
endfunction
```

Bilden wir eine Teilmenge $K_1 \subset K$, so verfahren wir stets wie folgt

$$K_1 = \{\mathbf{v}_j \mid j = 0, \ldots, \left\lfloor \frac{N-1}{2} \right\rfloor\} \text{ sowie}$$

$$K_2 = \{\mathbf{v}_j \mid j = \left\lfloor \frac{N-1}{2} \right\rfloor + 1, \ldots, N-1\}$$

für $K = \{\mathbf{v}_j \mid j = 0, \ldots, N-1\}$, da die Konturen geordnet vorliegen.
Die Funktion `KonturKonturDirekt` berechnet direkt den minimalen Abstand der Konturen K, K' mit $N \leq 10$ und $N' \leq 10$. Diese Berechnung haben wir mit Hilfe der bereits vorhandenen Funktion zur Berechnung des Startwertes für die Verknüpfung Kontur-Gerade realisiert.
Den Startwert für den maximalen Punkt erhalten wir ebenfalls mit Hilfe des obigen Algorithmus, indem wir anstelle des Minimums jeweils das Maximum der Abstände suchen und die entsprechenden Abszissen zurückgeben.
Zusammenfassend liefert uns das obige Verfahren die Startwerte

$$\mathbf{S}_{min} = \begin{bmatrix} a_{j_{min}} \\ b_{k_{min}} \end{bmatrix}, \mathbf{S}_{max} = \begin{bmatrix} a_{j_{max}} \\ b_{k_{max}} \end{bmatrix}.$$

Es stellt sich natürlich die Frage, ob der obige Algorithmus schneller als $O(NN')$ ist. Wir nehmen zur Vereinfachung an, dass $N = N'$ gilt. Dann sehen wir mit Hilfe des Master-Theorems zur Analyse von Divide & Conquer Verfahren ([Ste01], Abschnitt 4.2.2), dass wir keine Verbesserung zu erwarten haben, denn es gilt für die Laufzeit

$$T(N) = 4T\left(\frac{N}{2}\right) = \cdots = 4^n T\left(\frac{N}{2^n}\right)$$

und damit $T(N) = N^2$. Also würden wir auf diese Weise weiterhin eine Laufzeit von $O(NN')$ erreichen.
Um den Algorithmus 7.29 zu beschleunigen, bedienen wir uns des *Prinzips der schnellen Ablehnung*. Wir wollen schnellstmöglich eine oder mehrere Vergleiche des Algorithmus eliminieren. Hierfür berechnen wir eine so genannte bounding Box um jede der Teilkonturen K_1, K_2, K_1', K_2'. Sei hierfür $K \subset \mathbb{R}^d$ eine Kontur, dann ist eine *bounding Box* $BBox(K)$ von K definiert als

$$BBox(K) = [v_{min}^{(1)}, v_{max}^{(1)}] \times \cdots \times [v_{min}^{(d)}, v_{max}^{(d)}]$$

mit

$$v_{min}^{(i)} = \min\{v_j^{(i)} \mid j = 0, \ldots, N-1\} \text{ und } v_{max}^{(i)} = \max\{v_j^{(i)} \mid j = 0, \ldots, N-1\}$$

für $i = 1, \ldots, d$. Nun vergleichen wir die Abstände dieser bounding Boxen $BBox(K_1), BBox(K_2), BBox(K_1'), BBox(K_2')$ und starten die Rekursion nur mit den Teilkonturen K_i und K_j' mit den zwei kleinsten Abständen. Ist es uns nicht möglich, durch die Abstände der bounding Boxen zwei solche Teilkonturen auszuwählen, zum Beispiel wenn sich die bounding Boxen von mehr als zwei Teilkonturen schneiden, so setzen wir die Rekursion mit den bounding Boxen fort, die sich schneiden. Unsere Tests haben gezeigt, dass die Verknüpfung Kontur-Kontur fast ausschließlich für separierte Konturen verwendet wird, so dass der Fall, dass sich mehr als zwei Konturen schneiden, in der Praxis so gut wie nie vorkommt. Wir nehmen also im Folgenden an, dass wir stets zwei bounding Boxen mit minimalen Abstand auswählen können. Wir erhalten die folgende Modifikation des Algorithmus 7.29.

`function` $(abstand, a_{j_{min}}, b_{j_{min}})$ =`KonturKontur`(K, K')
  ⋮
  `BBox`(K_1); `BBox`(K_2); `BBox`(K_1'); `BBox`(K_2');
  a_1 =`AbstBBox(` `BBox`(K_1), `BBox`(K_1'));
  a_2 =`AbstBBox(` `BBox`(K_1), `BBox`(K_2'));
  a_3 =`AbstBBox(` `BBox`(K_2), `BBox`(K_1'));
  a_4 =`AbstBBox(` `BBox`(K_2), `BBox`(K_2'));
  `Sort`$(a1, a2, a3, a4)$;
  $a_{max} = \max\{a_1, \ldots, a_4\}$;
  $a_{max,1} = \max\{a_1, \ldots, a_4\} \backslash \{a_{max}\}$;
  `if` $(a_1 < a_{max}$ `AND` $a_1 < a_{max,1})$
    $(abstand_1, a_{j_1,min}, b_{k_1,min})$ =`KonturKontur`(K_1, K_1');
  `else if`$(a_2 < a_{max}$ `AND` $a_2 < a_{max,1})$
    $(abstand_2, a_{j_2,min}, b_{k_2,min})$ =`KonturKontur`(K_1, K_2');
  `else if`$(a_3 < a_{max}$ `AND` $a_3 < a_{max,1})$
    $(abstand_3, a_{j_3,min}, b_{k_3,min})$ =`KonturKontur`(K_2, K_2');
  `else if`$(a_4 < a_{max}$ `AND` $a_4 < a_{max,1})$
    $(abstand_4, a_{j_4,min}, b_{k_4,min})$ =`KonturKontur`(K_2, K_2');
  $j_{min} = \min_{abstand_j > 0}\{abstand_1, \ldots, abstand_4\}$;
  `return` $(abstand_{j_{min}}, a_{j_{min}}, b_{k_{min}})$;

Wie wirkt sich diese Modifikation auf die Laufzeit des Algorithmus aus? Mit Hilfe des Master-Theorems erhalten wir für $T(N)$ unter der Annahme,

dass unsere Sortier- und Vergleichsoperationen in $O(N)$ durchführbar sind,

$$T(N) = 2T\left(\frac{N}{2}\right) + N = 4T(\frac{N}{4}) + 2\frac{N}{2} + N$$
$$= 2^n T\left(\frac{N}{2^n}\right) + nN.$$

Mit $n = \log_2 N$ und $T(1) = 1$ folgt dann

$$T(N) = 2^{\log_2 N} \underbrace{T\left(\frac{N}{2^n}\right)}_{=T(1)=1} + N \log_2 N$$

und damit $T(N) = N \log_2(N)$. Wir fassen die Ergebnisse nun zusammen.

7.30 Lemma: Seien $K, K' \subset \mathbb{R}^d$ zwei Konturen, so dass $N = N'$ gilt. Dann besitzt der modifizierte Algorithmus 7.29 eine Gesamtkomplexität von $O(N \log_2 N)$.

Haben wir zwei Konturen $K, K' \subset \mathbb{R}^d$ mit $N = |K| \neq |K'| = N'$, können wir die Komplexität nach oben durch $O(N_{max} \log_2 N_{max})$ abschätzen, in dem wir $N_{max} = \max\{N, N'\}$ setzen.

7.8.5 Startwertberechnung: Kontur-Punkt

Bei dieser Verknüpfung ist das Regelgeometrieelement ein Punkt $\mathbf{p} \in \mathbb{R}^d$, das heißt wir betrachten ein Geometrieelement ohne Parameter. Daher suchen wir die Abszissen $a_{j_{min}}$ bzw. $a_{j_{max}}$, so dass gilt

$$\|\mathbf{p} - \mathbf{K}(a_{j_{min}})\|_2 = \min \text{ und } \|\mathbf{p} - \mathbf{K}(a_{j_{max}})\|_2 = \max .$$

$\mathbf{K} : \mathbb{R} \to \mathbb{R}^d$ ist wie üblich eine geeignete Parametrisierung von K und es muss nicht zwingend $\mathbf{K}(a_{j_{min}}) \in K$ gelten.
Wir berechnen für jedes Paar $\mathbf{v}_j, \mathbf{v}_{j+1} \in K$ für $j = 0, \ldots, N-2$ den nächsten Punkt

$$\mathbf{u}_j = \mathbf{v}_j + \nu_j(\mathbf{v}_{j+1} - \mathbf{v}_j) \text{ mit } \nu_j = \frac{(\mathbf{v}_{j+1} - \mathbf{v}_j)^t(\mathbf{p} - \mathbf{v}_j)}{(\mathbf{v}_{j+1} - \mathbf{v}_j)^t(\mathbf{v}_{j+1} - \mathbf{v}_j)}.$$

$\mathbf{u}_j$ ist der Fußpunkt des Lotes von $\mathbf{p}$ auf die Gerade durch $\mathbf{v}_{j+1}$ und $\mathbf{v}_j$. Gilt $\nu_j \in [0, 1]$, so setzen wir

$$d_j = \|\mathbf{p} - \mathbf{u}_j\|_2$$

und $a_{j,1} = a_j + \nu_j(a_{j+1} - a_j)$ sowie anschließend $a_j = a_{j,1}$. Ist hingegen $\nu_j \notin [0,1]$, so ist

$$d_j = \min\{\|\mathbf{p} - \mathbf{v}_j\|_2, \|\mathbf{p} - \mathbf{v}_{j+1}\|_2\}.$$

Wir erhalten $j_{min} = \text{argmin}\{d_0, \ldots, d_{N-1}\}$, $j_{max} = \text{argmax}\{d_0, \ldots, d_{N-1}\}$ und damit

$$S_{min} = a_{j_{min}} \text{ und } S_{min} = a_{j_{max}}.$$

Die Laufzeituntersuchung ergibt einen Aufwand von $8dN$ Operationen, also eine Komplexität von $O(N)$. Der Speicherplatzbedarf liegt bei $O(N)$.

7.9 Das modifizierte Newton-Verfahren

Sei für beide Minimierungsverfahren $f : \mathbb{R}^k \to \mathbb{R}$ die zu minimierende Funktion und $\mathbf{F} = \nabla f$ der Gradient von f. Wir bestimmen ein Minimum von f, indem wir eine Nullstelle von $\mathbf{F}$ berechnen. Die folgenden zwei Abschnitte beschäftigen sich daher im Wesentlichen mit der Suche nach einer Nullstelle von $\mathbf{F}$.
Die Idee des modifizierten Newton-Verfahrens ist die folgende. Wir führen in der Iterationsgleichung (7.21) eine Schrittweite ein und können damit gewisse Richtungen sowohl positiv als auch negativ beeinflussen. Wir berechnen also

$$\mathbf{x}_{j+1} = \mathbf{x}_j - \lambda_j \mathbf{d}_j \text{ mit } \mathbf{d}_j = (\mathbf{J}_\mathbf{F}(\mathbf{x}_j))^{-1}\mathbf{F}(\mathbf{x}_j)$$

für $j = 0, 1, 2, \ldots$. Dabei wählen wir $\lambda_j \in \mathbb{R}$ so, dass die Folge

$$(\eta(\mathbf{x}_j))_{j=0}^{\infty} \text{ mit } \eta(\mathbf{x}) = \mathbf{F}(\mathbf{x})^t\mathbf{F}(\mathbf{x})$$

streng monoton fällt und die Folge der $\mathbf{x}_j$ gegen das Minimum von $\eta(\mathbf{x})$ konvergiert. Diese Vorgehensweise führt zu einem Ergebnis, denn es ist $\eta(\mathbf{x}) \geq 0$ für alle $\mathbf{x}$ und $\eta(\mathbf{x}_*) = 0$ genau dann, wenn $\mathbf{F}(\mathbf{x}_*) = 0$. Daher ist jedes lokale Minimum $\mathbf{x}_*$ von η mit $\eta(\mathbf{x}_*) = 0$ auch ein globales Minimum von η und außerdem eine Nullstelle von $\mathbf{F}$. Bevor wir das Verfahren vorstellen, benötigen wir noch einige Begriffe.

7.31 Bezeichnung: Sei $\mathbf{A} \in \mathbb{R}^{k\times k}$. Wir setzen

$$\|\mathbf{A}\|_F = \sqrt{\sum_{j,l=1}^{k} |a_{jl}|^2}$$

die *Frobenius-Norm* von $\mathbf{A}$. Weiter ist

$$\text{cond } \mathbf{A} = \|\mathbf{A}\|_F^{-1}\|\mathbf{A}\|_F$$

die *Konditionszahl* der Matrix $\mathbf{A}$.

Mit den obigen Bezeichnungen können wir das modifizierte Newton-Verfahren skizzieren.

7.32 Algorithmus: Sei $f : \mathbb{R}^k \to \mathbb{R}$ eine zweimal differenzierbare Funktion und $\mathbf{F} : \mathbb{R}^k \to \mathbb{R}^k$ die erste Ableitung von f. Dann können wir ein Minimum von f folgendermaßen berechnen.

(1) Wir wählen einen Startpunkt $\mathbf{x}_0 \in \mathbb{R}^k$.

(2) Wir berechnen für $j = 0, 1, 2, \ldots$ den Punkt $\mathbf{x}_{j+1} \in \mathbb{R}^k$ aus $\mathbf{x}_j$, so dass Folgendes gilt.

(a) Wie in dem klassischen Newton-Verfahren bestimmen wir

$$\mathbf{d}_j = (\mathbf{J}_\mathbf{F}(\mathbf{x}_j))^{-1}\mathbf{F}(\mathbf{x}_j) \text{ und zusätzlich } \gamma_j = \frac{1}{\text{cond } \mathbf{J}_\mathbf{F}(\mathbf{x}_j)}.$$

Für $\eta(\mathbf{x}) = \mathbf{F}(\mathbf{x})^t\mathbf{F}(\mathbf{x})$ definieren wir $\eta_j(\tau) = \eta(\mathbf{x}_j - \tau\mathbf{d}_j)$ Wir suchen dann die kleinste natürliche Zahl $n \in \mathbb{N}_0$ mit

$$\eta_j(2^{-n}) \leq \eta_j(0) - 2^{-n}\frac{\gamma_j}{4}\|\mathbf{d}_j\|_2\|\nabla\eta(\mathbf{x}_j)\|_2. \tag{7.29}$$

(b) Wir bestimmen die Schrittweite λ_j, welche

$$\eta(\mathbf{x}_{j+1}) = \min_{0\leq i\leq n} \eta_j(2^{-i}) \tag{7.30}$$

genügt. Dabei ist $\mathbf{x}_{j+1} = \mathbf{x}_j - \lambda_j\mathbf{d}_j$.

(3) Wir brechen den Algorithmus ab, falls eine der folgenden Bedingungen erfüllt ist.

(i) Es ist $\|F(\mathbf{x}_j)\|_2 < \varepsilon_1$ für ein $j \geq 0$ und einen Schwellenwert $\varepsilon_1 > 0$. In diesem Fall haben wir ein Minimum von f gefunden.

(ii) Es gilt $\|\mathbf{x}_{j+k} - \mathbf{x}_j\|_2 < \varepsilon_2$ für ein $j \geq 0$ sowie ein $k = 1, \ldots, \widetilde{k} \in \mathbb{N}$ und einen Schwellenwert $\varepsilon_2 > 0$. In diesem Fall ist das Verfahren stationär geworden und wir brechen es ab, überprüfen aber vorher, ob eventuell ein Minimum von f, also eine Nullstelle von $\mathbf{F}$, gefunden wurde.

(iii) Es wurde die maximale Anzahl von Iterationen erreicht, also $j \geq j_{maxiter}$. Auch in diesem Fall brechen wir das Verfahren ab und überprüfen, ob ein Minimum gefunden wurde.

(iv) Eine oder mehrere Komponenten von $\mathbf{x}_j$ verletzen eine vorher festgelegte Bedingung. In diesem Fall liegt ein Minimum am Rand der Kontur oder des Regelgeometrieelements und wir beenden das Verfahren an dieser Stelle mit diesem $\mathbf{x}_j$ als Ergebnis.

Für den Algorithmus können wir die folgende Konvergenzaussage formulieren, welche wir in [Sto72] finden.

7.33 Satz: Seien $\mathbf{F} : \mathbb{R}^k \to \mathbb{R}^k$ und $\eta : \mathbb{R}^k \to \mathbb{R}$ mit $\eta(\mathbf{x}) = \mathbf{F}(\mathbf{x})^t\mathbf{F}(\mathbf{x})$. Sei $\mathbf{x}_0 \in \mathbb{R}^k$ so gewählt, dass folgenden Eigenschaften erfüllt sind.

(1) Die Menge $L = \{\mathbf{x} \in \mathbb{R}^k \mid \eta(\mathbf{x}) \leq \eta(\mathbf{x}_0)\}$ ist kompakt.

(2) η ist in einer Umgebung von L stetig differenzierbar.

(3) Für alle $\mathbf{x} \in L$ existiert $(\mathbf{J}(\mathbf{x}))^{-1}$.

Dann ist die Folge $(\mathbf{x}_j)_{j=0}^{\infty}$, erzeugt wie in Algorithmus 7.32, wohldefiniert und es gelten die folgenden Aussagen.

(i) Es ist $\mathbf{x}_j \in L$ für alle $j \geq 0$. Außerdem besitzt $(\mathbf{x}_j)_{j=0}^{\infty}$ mindestens einen Häufungspunkt $\mathbf{x}_* \in L$.

(ii) Jeder Häufungspunkt $\mathbf{x}_*$ von $(\mathbf{x}_j)_{j=0}^{\infty}$ ist auch eine Nullstelle von $\mathbf{F}$.

Wir erhalten durch Anwendung des Algorithmus 7.32 unter den Bedingungen (1) bis (3) stets mindestens eine Nullstelle. Da es sich bei unserer Funktion f um eine Splinekurve vom Grad 3 handelt, sind die Bedingungen (1) bis (3) erfüllt. Eine Ausnahme bilden die Ecken, die wir als nicht differenzierbare Stellen der Splinekurve modelliert haben. Wir brechen das Verfahren daher ab, wenn $\mathbf{x}_j$ eine Ecke erreicht und überprüfen, ob eine Minimal- oder Maximalstelle vorliegt. Ist dies nicht der Fall, starten wir das Verfahren nach der nicht differenzierbaren Stelle neu.
Berechnen wir in jedem Iterationsschritt des Verfahrens γ_j und damit $\|\mathbf{J}_\mathbf{F}(\mathbf{x}_j)\|_2$ direkt, so wird der Algorithmus sehr aufwändig, denn die direkte Berechnung der Konditionszahl ist ein $O(N^3)$-Algorithmus für eine $N \times N$–Matrix.

Verwendet man eine mathematische Bibliothek, wie zum Beispiel *LAPACK* oder eine Software wie *MatLab*® bzw. *Octave*, so wird die Konditionszahl bei der Berechnung der inversen Jacobi-Matrix mit abgeschätzt, siehe dazu [And99] für *LAPACK* und [Mat06] für *MathLab*®.
Da wir keines dieser Pakete bei der Implementierung benutzt haben, gehen wir einen Kompromiss ein und ersetzen γ_j durch eine beliebige untere Schranke $\gamma > 0$. In Gleichung (7.29) bestimmen wir ein $n \in \mathbb{N}_0$, so dass gilt

$$\eta_j(2^{-n}) \leq \eta_j(0). \tag{7.31}$$

Wir erhalten die Schrittweite $\lambda_j = 2^{-n}$. Da die einzige Forderung $\gamma > 0$ lautet, gibt es für das so modifizierte Verfahren keine Konvergenzaussage im Sinne von Satz 7.33.
Sei für die Laufzeitberechnung *maxiter* die maximale Anzahl der Durchläufe des Verfahrens. Bevor wir die Laufzeit für den Algorithmus untersuchen, benötigen wir Laufzeiten für einige Teilergebnisse, wie zum Beispiel die Auswertung der Splinefunktion oder die Berechnung des Gradienten von $\mathbf{F}$.
Für die Auswertung der Splinekurve $N_{m,T*}\mathbf{D} \in \mathbb{R}^d$ an einer Stelle $x \in \mathbb{R}$ müssen wir $m+1$–mal einen B-Spline der Ordnung m auswerten. Es ergibt sich die Gesamtzahl von

$$Op_s = (m+1)\left(9d\frac{m(m+1)}{2} + 3d\right)$$

Operationen. Analog erhalten wir für die 1. Ableitung der Splinekurve an der Stelle x

$$Op_{s1} = m\left(9d\frac{m(m-1)}{2} + 3d\right)$$

Operationen und für die 2. Ableitung folgerichtig

$$Op_{s2} = 2(m-1)\left(9d\frac{(m-1)(m-2)}{2} + 8d\right)$$

Operationen. Sei $k = z+1$ mit der Anzahl z der Parameter des Regelgeometrieelements, dann können wir den Gradienten von $\mathbf{F}$ mit Hilfe von

$$Op_g = Op_s + Op_{s1} + Op_{s2} + kK_1$$

Rechenoperationen berechnen. $K_1 \in \mathbb{N}$ ist hierbei eine Konstante. Eine Iteration des Verfahrens benötigt zweimal die Berechnung des Gradienten

und je einmal die Auswertung der Splinekurve, sowie der 1. und der 2. Ableitung des Splines.
Die Aufstellung der Jacobi-Matrix sowie die Berechnung der Inversen hängen ausschließlich von k ab, wir können diese Rechenschritte mit K_2k^3 für konstantes $K_2 \in \mathbb{N}$ abschätzen.
Zusammenfassend können wir die Anzahl der Rechenschritte mit

$$\begin{aligned} &Op_{Newton} \\ &= maxiter(2Op_g + Op_s + Op_{s1} + Op_{s2} + K_1m^3 + K_2k^3) \\ &= maxiter\left((54d + K_1)m^3 - \frac{189}{2}dm^2 + \frac{429}{2}dm - 93d + kK_1 + k^3K_2\right) \end{aligned}$$

abschätzen. Wir nehmen stets an, dass die maximale Anzahl der Iterationen unabhängig von N ist, daher beträgt die Laufzeit des Verfahrens $O(1)$. Der Speicherplatzbedarf liegt bei $O(1)$, da wir ausschließlich Zwischenergebnisse der Dimension k^2 bzw. m^3 speichern, welche unabhängig von N sind.
Das Rang-1-Verfahren von Broyden, das wir in unseren Berechnungen ebenfalls verwendet haben, ist ein wohlbekanntes Standardverfahren, welches wir an dieser Stelle nicht besprechen wollen. Wir verweisen auf [Bro65].

7.10 Berechnung der Extremalpunkte

In diesem Abschnitt fassen wir die Ergebnisse der vorherigen Abschnitte zusammen und stellen einen Gesamtalgorithmus vor. Wir verwenden die Bezeichnungen aus Definition 7.2 und berechnen ausgehend von einem minimalen und einem maximalen Startwert einen minimalen und einen maximalen Punkt sowie die entsprechenden minimalen und maximalen Abstände.

7.34 Algorithmus: Sei $K = \{\mathbf{v}_j \mid j = 0, \ldots, N-1\} \subset \mathbb{R}^d$ eine Kontur und $\mathbf{R} : \mathbb{R}^z \to \mathbb{R}^d$ für ein $z \geq 0$ ein Regelgeometrieelement. Dann erhalten wir einen minimalen und einen maximalen Punkt und die entsprechenden Abstände d_{min} und d_{max} durch die Anwendung der folgenden Schritte.

(1) Wir konvertieren, wenn notwendig, das Regelgeometrieelement in eine der angegebenen Formen. Wir haben gemäß Bezeichnung 7.1

   (a) Gerade in der Form

$$\mathbf{R}_G(\mu) = \mathbf{p} + \mu\mathbf{q} \subset \mathbb{R}^d \text{ mit } \mathbf{p}, \mathbf{q} \in \mathbb{R}^d \text{ und } \mu \in \mathbb{R}.$$

(b) Kreis als Sonderfall für $d \leq 3$ mit der Gleichung

$$\mathbf{R}_C(\mu) = \mathbf{p}_M + r_C \mathbf{P}(\mu) \subset \mathbb{R}^d \text{ mit}$$

$\mathbf{p}_M \in \mathbb{R}^d$, $\mathbf{P} : \mathbb{R} \to \mathbb{R}^d$, $r_C > 0$ und $\mu \in [0, 2\pi)$.

(c) Ebene definiert durch

$$\mathbf{R}_E(\boldsymbol{\mu}_E) = \mathbf{p} + \mu^{(1)}\mathbf{q} + \mu^{(2)}\mathbf{r} \subset \mathbb{R}^d \text{ mit}$$

$\mathbf{p}, \mathbf{q}, \mathbf{r} \in \mathbb{R}^d$ und $\boldsymbol{\mu}_E = [\mu^{(1)}\ \mu^{(2)}]^t \in \mathbb{R}^2$.

(d) Kontur unverändert als

$$K = \{\mathbf{v}_j \mid j = 0, \ldots, N-1\} \subset \mathbb{R}^d$$

(e) Punkt als

$$\mathbf{R}_P = \mathbf{p} \in \mathbb{R}^d.$$

(f) Kugel, Zylinder und Kegel werden nicht betrachtet.

(2) Die Interpolation wird gestartet. Als Ergebnisse erhalten wir die veränderte Kontur K ohne doppelte Punkte und mit zusätzlichen Punkten, falls es sich um mehrere Teilkonturen gehandelt hat, sowie das Kontrollpolygon $\mathbf{D}$, die Knotenfolge $T^*_{m,N-1}$, die Menge der Abszissen A und die Eckenmenge E, wobei $E = \emptyset$ möglich ist.

(3) Wir bestimmen, ob K und $\mathbf{R}$ in einer Ebene liegen. Dazu berechnen wir eine Least-Square-Ebene durch die Punkte von K und prüfen nach, ob $\mathbf{R}$ auch in dieser Ebene liegt. Die gegenseitige Lage ist wichtig für die spätere Bestimmung der Extremalpunkte.

(4) Wir benutzen einen Algorithmus aus den Abschnitten 7.8.1 bis 7.8.5, um den entsprechenden Startwert zu bestimmen.

(5) Auf Grund der Lage der Ecken und des Startwertes bestimmen wir den Abschnitt der Splinekurve, in dem das Minimum bzw. Maximum liegt. Werden bei der Berechnung die Grenzen dieses Abschnittes verlassen, so wird der Minimierungsalgorithmus gemäß Algorithmus 7.32 (3) (iv) abgebrochen.

(6) Wir unterscheiden zwei verschiedene Zustände.

(i) Kontur und Regelgeometrieelement liegen nicht in derselben Ebene oder das Regelgeometrieelement ist eine Ebene oder ein Punkt. In einem solchen Fall werden stets nur der minimale und maximale Punkt und die entsprechenden Abstände berechnet. Hierfür starten wir zweimal nacheinander Algorithmus 7.32 und werten anschließend die Ergebnisse aus.

(ii) Anderenfalls prüfen wir, ob bei der Startwertberechnung ein zweiter minimaler Punkt ermittelt wurde und starten in diesem Fall den obigen Algorithmus dreimal nacheinander, ansonsten nur zweimal.

Die Auswertung der Ergebnisse geschieht folgendermaßen. Wir vergleichen d_{min}^{Newton} mit dem minimalen Abstand des Startwerts $d_{min}^{Startwert}$. Gilt

$$d_{min}^{Newton} > d_{min}^{Startwert} \text{ sowie } \left|d_{min}^{Newton} - d_{min}^{Startwert}\right| > \varepsilon_{Abst}$$

für einen Schwellenwert $\varepsilon_{Abst} > 0$, so verwenden wir den Startwert als Ergebnis des Verfahrens, ansonsten geben wir d_{min}^{Newton} zurück.
Analog verfahren wir im Fall eines zweiten minimalen Schnittpunktes oder des maximalen Punktes.

(7) Wir geben als Ergebnisse die Punkte $\mathbf{S}_{min}$, eventuell $\mathbf{S}_{min1}$ und $\mathbf{S}_{max}$, sowie die Abstände $d_{min}^{(1)}, \ldots, d_{min}^{(d)}$ und $d_{max}^{(1)}, \ldots, d_{max}^{(d)}$ aus. Dabei gilt

$$d_{min} = \sqrt{\sum_{j=1}^{d} (d_{min}^{(j)})^2} \text{ und } d_{max} = \sqrt{\sum_{j=1}^{d} (d_{max}^{(j)})^2}.$$

Zum Abschluss des Abschnittes beschäftigen wir uns mit der Laufzeit und dem Speicherbedarf von Algorithmus 7.34 und geben eine Fehlerabschätzung für die Ergebnisse an.
Die Laufzeituntersuchung führen wir für die Berechnung eines Punktepaares durch, da die wesentlichen Berechnungen, wie die Interpolation, Startwertberechnung und Minimierungsverfahren, in diesem Verlauf enthalten sind.
Wir nehmen stets an, dass d und $|h|$ unabhängig von N sind und $M = \log_2 N$ gilt. Der Grad der Splinekurve m bleibt beliebig. In der Praxis haben sich allerdings kubische Splines bewährt, die einen guten Kompromiss zwischen Glattheit der Kurve und Geschwindigkeit der Berechnung bieten. Dann erhalten wir folgende Laufzeiten und Speicheranforderungen.

(1) Das Konvertieren der Regelgeometrieelemente und die anderen Aufgaben sind in einer Laufzeit von $O(N)$ zu bewältigen. Der Speicherbedarf liegt bei $O(N)$.

(2) Für die Interpolation haben wir nach Korollar 7.27 eine Laufzeit von $O(N)$ sowie einen Speicherplatzbedarf von $O(N)$.

(3) Die Bestimmung einer Least-Square Ebene der Konturpunkte ist ebenfalls ein $O(N)$–Verfahren und es werden $O(N)$ Speicherplätze belegt.

(4) Für die Startwertbestimmung haben wir folgende Laufzeiten berechnet.

   (a) Kontur-Gerade:
   Den Startwert für die Verknüpfung Kontur-Gerade können wir nach Abschnitt 7.8.1 in $O(N)$ Schritten berechnen.

   (b) Für den Startwert der Verknüpfung Kontur-Kreis benötigen wir $O(N)$ Rechenoperationen.

   (c) Die Berechnung des Startwertes von Kontur-Ebene erfolgt ebenfalls in $O(N)$ Operationen.

   (d) Nehmen wir an, dass zwei Konturen gleicher Länge N vorliegen, so können wir den Startwert für die Verknüpfung Kontur-Kontur nach Lemma 7.30 in $O(N \log_2 N)$ Schritten berechnen.

   (e) Den Startwert der Verknüpfung Kontur-Punkt können wir in einer Zeit von $O(N)$ berechnen.

   Der Speicherplatzbedarf für alle Startwerte beträgt $O(N)$.

(5) Die Bestimmung der relevanten Bereiche der Kontur nimmt höchstens $O(N)$ Zeit in Anspruch.

(6) Für den Minimierungsalgorithmus benötigen wir nach den Ergebnissen aus Abschnitt 7.7 $O(1)$ Speicherplatz und können von einer Laufzeit von ebenfalls $O(1)$ ausgehen.

Punkt (7) ist unabhängig von N und daher vernachlässigbar.
Damit ergibt sich die folgende Aussage.

7.35 Satz: Sei $K = \{\mathbf{v}_j \mid j = 0, \ldots, N-1\} \subset \mathbb{R}^d$, $0 < d \ll N$, eine Kontur. Sei $\mathbf{R} : \mathbb{R}^z \to \mathbb{R}^d$ ein Regelgeometrieelement vom Typ 7.1, entweder ein Punkt, eine Gerade, ein Kreis oder eine Ebene. Sei $M = N \log_2 N$ die Anzahl der Skalen der schnellen Wavelettransformation und $|h| \ll N$

die Länge der Verfeinerungsfolge. Sei weiter $m > 0$ der Grad der interpolierenden Splinekurve. Dann können wir einen minimalen bzw. einen maximalen Punkt von K und $\mathbf{R}$ mit einer Laufzeit von $O(N)$ berechnen. Der Speicherplatzbedarf beträgt insgesamt $O(N)$.

Die Verknüpfung Kontur-Kontur stellt einen Sonderfall dar, da wir eine zusätzliche Annahme an das Regelgeometrieelement Kontur stellen müssen. Daher fassen wir das Ergebnis der Laufzeitberechnung in einer gesonderten Aussage zusammen.

7.36 Satz: Seien $K, K' \subset \mathbb{R}^d$ zwei Konturen, so dass $|K| = |K'| = N$ gilt. Dann können wir einen minimalen bzw. einen maximalen Punkt von K und K' in $O(N \log_2 N)$ Rechenoperationen berechnen. Dabei benötigen wir $O(N)$ Speicherplatz.

7.37 Bemerkung: Liegen uns zwei Konturen unterschiedlicher Länge vor, so können wir die Laufzeit abschätzen, indem wir N als Maximum beider Längen wählen. Wir bleiben auch dann unterhalb der Laufzeit von $O(NN')$.

Wir betrachten nachfolgend die tatsächlichen Laufzeiten der Implementierung der Verknüpfung Kontur-Gerade. Die Tests wurden auf einem PC mit einer Intel Pentium-CPU mit 2,4 GHz, 1024 MB RAM und Windows XP durchgeführt. Der Algorithmus wurde unter Visual C 6.0 implementiert. Bei den Tests handelt es sich um die Debug-Version.
Wir interpolieren jeweils $3D$–Konturen, also $d = 3$, mit einer kubischen Splinekurve, daher gilt $m = 3$. Wir bestimmen Ecken mit Hilfe eines biorthogonalen Spline-Wavelets, das heißt $|h| = 8$, und berechnen dann die ersten 6 Skalen, also ist $M = 6$.

Kontur	20x_1.asc	schlange.asc	test2.asc	d260_2.asc
Anz. Punkte	26017	15182	9015	7265
1000 Punkte	0.325 sec	0.188 sec	0.255 sec	0.312 sec
2000 Punkte	0.687 sec	0.750 sec	0.688 sec	0.469 sec
5000 Punkte	1.297 sec	2.453 sec	1.859 sec	2.078 sec
7265 Punkte	–	–	–	3.172 sec
7500 Punkte	2.109 sec	3.218 sec	2.297 sec	–
9015 Punkte	–	–	2.860 sec	–
10000 Punkte	2.953 sec	5.532 sec	–	–
15000 Punkte	5.188 sec	7.265 sec	–	–
20000 Punkte	8.078 sec	–	–	–
26017 Punkte	14.016 sec	–	–	–

Wir sehen, die gemessene Laufzeit für die Kontur 20x_1.asc und $N = 10000$ beträgt 2.953 sec, für 20000 Punkte haben wir 8.078 sec gemessen. Dabei gilt $2.953^2 = 8.720$. Ähnliches können wir für die Kontur schlange.asc beobachten, welche auch eine Laufzeit von $O(N^2)$ erreicht. Die gemessene Laufzeit für $N = 5000$ beträgt 2.453 sec, für $N = 10000$ sind es dann 5.532 sec. Auch hier bestätigt sich die quadratische Laufzeit, da $2.453^2 = 6.017$ gilt.
Den Grund dafür, dass die tatsächliche Laufzeit in etwa quadratisch ist, während die von uns ermittelte Laufzeit bei $O(N \log_2 N)$ liegt, finden wir in den Konstanten.
Wie wir gesehen haben, liegt die Anzahl der Operationen bei

$$N\left[9dm^3 + \frac{13}{2}dm^2 + m\left(\frac{7}{2}d + 1\right) + d\left(4\left|h\right| + 33\right) + 13\right]$$
$$+4dM(|h| + 1) + C(m, d, |h|).$$

Für verhältnismäßig kleine N ist der Ausdruck in der Klammer recht groß, so dass wir eine schlechtere Laufzeit erhalten. Für die asymptotische Laufzeitberechnung spielt aber nur noch die Rekursionstiefe M eine Rolle, die anderen Einträge sind unabhängig von N.

8 Regelgeometrien

Ein wichtiger Bestandteil der Messungen von *Werth Messtechnik GmbH* ist die Berechnung von *Regelgeometrieelementen*. Wir unterscheiden zwei Arten solcher Elemente. Zum einen die so genannten *Ausgleichselemente*, dabei handelt es sich um Objekte, die durch eine Least-Square Berechnung bestimmt werden, und zum anderen *Hüll-* oder *Pferchelemente*, die einer minimalen oder maximalen Eigenschaft bezüglich einer Kontur genügen.
Die Software *WinWerth* des Unternehmens *Werth Messtechnik GmbH* berechnet die Elemente Ausgleichspunkt, Ausgleichsgerade, Ausgleichskreis, Ausgleichsebene, Augleichskegel, Pferchkreis, Hüllkreis, Gauß-Kreis und minimale Zone Kreis.
Wir stellen in diesem Abschnitt Algorithmen zur Berechnung der Elemente

(1) *Ausgleichsellipse,*

(2) *Ausgleichshyperbel,*

(3) *Ausgleichsparabel,*

(4) *Hüllkreis* (alternative Vorgehensweise für $\mathbb{R}^3$) und

(5) *Pferchkreis* (alternative Vorgehensweise)

vor. Wir beschäftigen uns im ersten Abschnitt allgemein mit der Fragestellung eines Ausgleichselements und wenden uns anschließend der Berechnung der obigen Elemente (1) bis (3) zu. Die letzten beiden Abschnitte dieses Kapitels werden wir den Nicht-Ausgleichselementen Hüll- und Pferchkreis widmen.

8.1 Problemstellung Ausgleichselemente

Sei $K = \{\mathbf{v}_j \mid j =, \ldots, N-1\} \subset \mathbb{R}^d$ eine Kontur, zu der wir ein Ausgleichselement $\mathbf{R}$ bestimmen wollen. Wir betrachten zwei Fälle.

(1) Wir können $\mathbf{R}$ als

$$\mathbf{R}(\mathbf{x}) = \mathbf{y}(\mathbf{x})^t\mathbf{a} + b \text{ mit } \mathbf{y}(\mathbf{x}), \mathbf{a} \in \mathbb{R}^z \text{ und } b \in \mathbb{R} \tag{8.1}$$

darstellen. Dabei ist $\mathbf{y} : \mathbb{R}^d \to \mathbb{R}^z$ zum Beispiel im Falle einer Hyperbel ein mehrdimensionales Polynom. Wie wir gleich sehen werden, ist ein solcher Ansatz zum Beispiel für eine Hyperbel, Parabel, Ellipse oder

Ebene möglich.
Wir berechnen für $j = 0, \ldots, N-1$

$$\mathbf{R}(\mathbf{v}_j) = \mathbf{y}(\mathbf{v}_j)^t \mathbf{a} + b = r^{(j)}$$

und erhalten das lineare Gleichungssystem

$$\mathbf{M}\mathbf{x} = \begin{bmatrix} \mathbf{y}(\mathbf{v}_0)^{(1)} & \ldots & \mathbf{y}(\mathbf{v}_0)^{(z)} & 1 \\ \vdots & \ddots & \vdots & \vdots \\ \mathbf{y}(\mathbf{v}_{N-1})^{(1)} & \ldots & \mathbf{y}(\mathbf{v}_{N-1})^{(z)} & 1 \end{bmatrix} \begin{bmatrix} a^{(1)} \\ \vdots \\ a^{(z)} \\ b \end{bmatrix} = \begin{bmatrix} r^{(0)} \\ \vdots \\ r^{(N-1)} \end{bmatrix} = \mathbf{r}.$$

Wir bezeichnen $\mathbf{r} \in \mathbb{R}^N$ als *Residuenvektor*. In den meisten Fällen gilt $N \gg z$, also ist das lineare Gleichungssystem überbestimmt und wir betrachten die Fehlerquadratsumme

$$\begin{bmatrix} \mathbf{a}^t & b \end{bmatrix} \mathbf{M}^t \mathbf{M} \begin{bmatrix} \mathbf{a} \\ b \end{bmatrix} = \mathbf{r}^t \mathbf{r} \tag{8.2}$$

mit

$$\mathbf{a} = \begin{bmatrix} a^{(1)} \\ \vdots \\ a^{(z)} \end{bmatrix} \in \mathbb{R}^z$$

und $\mathbf{M} \in \mathbb{R}^{N \times z}$ sowie $\mathbf{r} \in \mathbb{R}^N$ wie oben. Dieses quadratische Problem müssen wir mit den Mitteln der numerischen Mathematik oder der Optimierung lösen, um den Parametervektor $\mathbf{a}$ und den Parameter b bestimmen zu können.

(2) Das Regelgeometrieelement $\mathbf{R}$ ist als

$$\mathbf{R}(\mathbf{x}) = \mathbf{y}(\mathbf{x})^t \mathbf{a} + b = x^{(k)} \text{ mit } \mathbf{y}(\mathbf{x}), \mathbf{a} \in \mathbb{R}^z \text{ und } b \in \mathbb{R} \tag{8.3}$$

darstellbar. Dabei ist die Funktion $\mathbf{y}$ ähnlich wie in Fall (1) definiert, $1 \leq k \leq d$ ist eine feste Komponente der Kontur K. Sei im Folgenden $k = d$. Wir berechnen, wie in Fall (1)

$$\mathbf{R}(\mathbf{x}_j) = v_j^{(d)} \text{ mit } \mathbf{x}_j = \begin{bmatrix} v_j^{(1)} \\ \vdots \\ v_j^{(d-1)} \end{bmatrix} \text{ für } j = 0, \ldots, N-1$$

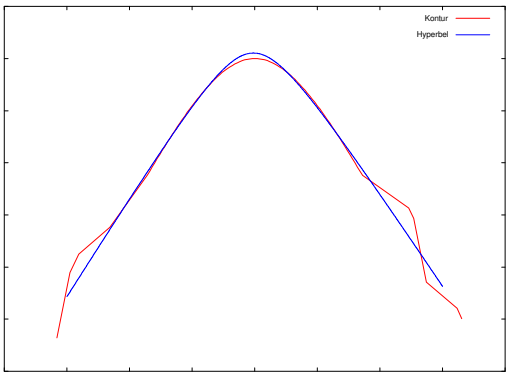

Abbildung 27: Eine hyperbelförmige Kontur mit einer Ausgleichshyperbel

und erhalten die gleiche Designmatrix $\mathbf{M}$ wie in (8.2). Das überbestimmte Gleichungssystem lautet in diesem Fall

$$\mathbf{M} \begin{bmatrix} \mathbf{a} \\ b \end{bmatrix} = \begin{bmatrix} v_0^{(d)} \\ \vdots \\ v_{N-1}^{(d)} \end{bmatrix} = \mathbf{v}^{(d)}. \tag{8.4}$$

Dieses Gleichungssystem lösen wir, indem wir zu der Normalgleichung

$$\mathbf{M}^t\mathbf{M} \begin{bmatrix} \mathbf{a} \\ b \end{bmatrix} = \mathbf{M}^t\mathbf{v}^{(d)}$$

übergehen und auf diese ein geeignetes Verfahren anwenden.

Im Folgenden gehen wir sowohl auf die Darstellung des Regelgeometrieelements als auch auf die Lösung der Gleichungssysteme näher ein.

8.2 Ausgleichsellipse, -hyperbel und -parabel

Die Problemstellung bei *Werth Messtechnik GmbH* war, zu einem hyperbelförmigen Werkstück aus dem Bereich der Vakuumtechnik eine Ausgleichshyperbel zu berechnen. Eine aus einer solchen Messung entstandene Kontur sehen wir in Abbildung 27.
Die Berechnung der Ausgleichselemente Ellipse und Parabel ist ein Nebenprodukt der Bestimmung einer Ausgleichshyperbel und beruht auf keiner konkreten Messaufgabe. Bei allen drei Regelgeometrieelementen handelt es sich um Elemente in einer Ebene, also können wir in diesem Fall $K \subset \mathbb{R}^2$ annehmen.
Das Problem, eine Ausgleichsellipse, -hyperbel und -parabel zu berechnen,

wird in [Ni04] bearbeitet, dabei wird das Ergebnis einer Berechnung hinsichtlich des Typs getestet. Stellt man bei diesem Test fest, dass ein Ausgleichselement berechnet wurde, das nicht erwünscht war, so stellt man eine so genannte geodätische Gleichung auf. Danach wird das nächste Ausgleichselement des richtigen Typs berechnet.
Eine Ausgleichsrechung mit quadratischen Nebenbedingungen wurde zuerst von [Fi99] angewandt, allerdings nur für den Fall der Ausgleichsellipse. Wir wollen hier eine Methode vorstellen, mit der wir alle drei Typen der Ausgleichselemente effizient berechnen können.
Sei im Folgenden

$$K = \{\mathbf{v}_j \mid j = 0, \ldots, N-1\} \subset \mathbb{R}^2$$

die zu untersuchende Kontur.

8.2.1 Anpassung der Daten

Allgemein besteht keine Notwendigkeit, die Daten der Kontur anzupassen. Allerdings zeigte [Ch79], dass sich bei zentrierten Daten, die eine mittlere Streuung von $\sqrt{2}$ besitzen, die numerische Stabilität und die Geschwindigkeit von Algorithmen in der Ebene drastisch verbessern. Von dieser Verbesserung wollen wir profitieren, berechnen daher

$$\overline{\mathbf{v}} = \begin{bmatrix} \overline{v^{(1)}} \\ \overline{v^{(2)}} \end{bmatrix} = \begin{bmatrix} \frac{1}{N}\sum_{j=0}^{N-1} v_j^{(1)} \\ \frac{1}{N}\sum_{j=0}^{N-1} v_j^{(2)} \end{bmatrix}$$

und setzen

$$\widetilde{\mathbf{v}}_j = \begin{bmatrix} \widetilde{v}_j^{(1)} \\ \widetilde{v}_j^{(2)} \end{bmatrix} = \begin{bmatrix} v_j^{(1)} - \overline{v^{(1)}} \\ v_j^{(2)} - \overline{v^{(2)}} \end{bmatrix} \text{ für } j = 0, \ldots, N-1.$$

Mit einem Skalierungsfaktor

$$\sigma = \sqrt{\frac{2N}{\sum_{j=0}^{N-1}\left((v_j^{(1)})^2 + (v_j^{(2)})^2\right)}}$$

können wir die standartisierten Konturdaten $\mathbf{v}_j$ schreiben als

$$\mathbf{v}_j = \begin{bmatrix} v_j^{(1)} \\ v_j^{(2)} \end{bmatrix} = \begin{bmatrix} \sigma\widetilde{v}_j^{(1)} \\ \sigma\widetilde{v}_j^{(2)} \end{bmatrix} \text{ für } j = 0, \ldots, N-1.$$

Damit sind die Konturdaten $\mathbf{v}_j$, $j = 0, \ldots, N-1$ zentriert und haben eine mittlere Streuung von $\sqrt{2}$.

8.2.2 Die Besetzung der Design-Matrix

Die folgende Definition eines Kegelschnittsegments sowie einer Kegelschnittgleichung finden wir in [Kle40].

8.1 Definition: Eine quadratische Form in einer projektiven Ebene, in unserem Fall ein Kegelschnittsegment, hat die Form

$$\mathbf{p}^t\mathbf{K}\mathbf{p} = \begin{bmatrix} x & y & w \end{bmatrix} \begin{bmatrix} a & \frac{b}{2} & \frac{d}{2} \\ \frac{b}{2} & c & \frac{e}{2} \\ \frac{d}{2} & \frac{e}{2} & f \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix} = 0. \tag{8.5}$$

Wir bezeichnen $\mathbf{K}$ als *Conic-Matrix*, $\mathbf{p}$ als *homogenen Punkt* und w als *homogene Komponente*. Eine solche Kegelschnittgleichung können wir auch als

$$\mathbf{d}^t\mathbf{z} = \begin{bmatrix} x^2 & xy & y^2 & x & y & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \end{bmatrix} = 0 \tag{8.6}$$

darstellen. Dabei ist $\mathbf{d}$ der Vektor der dualen Grassman-Koordinaten, $\mathbf{z}$ gibt die Grassman-Koordinaten des Kegelschnittes an.

Setzen wir in $\mathbf{d}$ nacheinander alle Konturpunkte $\mathbf{v}_0, \ldots, \mathbf{v}_{N-1}$ ein, so wird aus dem Vektor $\mathbf{d}^t$ eine Matrix $\mathbf{M}$ und wir erhalten das lineare Gleichungssystem

$$\mathbf{M}\mathbf{z} = \begin{bmatrix} \mathbf{M}_2 & \mathbf{M}_1 & \mathbf{M}_0 \end{bmatrix} \begin{bmatrix} \mathbf{z}_2 \\ \mathbf{z}_1 \\ z_0 \end{bmatrix} = \begin{bmatrix} r^{(0]} \\ \vdots \\ r^{(N-1)} \end{bmatrix} = \mathbf{r} \tag{8.7}$$

mit dem *Residuenvektor* $\mathbf{r}$ der N Punkte der Kontur K. Die Matrix $\mathbf{M}$ teilen wir folgendermaßen in drei Submatrizen

$$\mathbf{M}_2 = \begin{bmatrix} \left(v_0^{(1)}\right)^2 & v_0^{(1)}v_0^{(2)} & \left(v_0^{(2)}\right)^2 \\ \vdots & \vdots & \vdots \\ \left(v_{N-1}^{(1)}\right)^2 & v_{N-1}^{(1)}v_{N-1}^{(2)} & \left(v_{N-1}^{(2)}\right)^2 \end{bmatrix},$$

$$\mathbf{M}_1 = \begin{bmatrix} v_0^{(1)} & v_0^{(2)} \\ \vdots & \vdots \\ v_{N-1}^{(1)} & v_{N-1}^{(2)} \end{bmatrix} \text{ sowie } \mathbf{M}_0 = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}.$$

Entsprechend setzen wir

$$\mathbf{z}_2 = \begin{bmatrix} a \\ b \\ c \end{bmatrix}, \mathbf{z}_1 = \begin{bmatrix} d \\ e \end{bmatrix} \text{ und } z_0 = f.$$

Die Summe der Fehlerquadrate können wir folgendermaßen darstellen

$$\mathbf{z}^t\mathbf{M}^t\mathbf{M}\mathbf{z} = \begin{bmatrix} \mathbf{z}_2^t & \mathbf{z}_1^t & z_0 \end{bmatrix} \begin{bmatrix} \mathbf{S}_{22} & \mathbf{S}_{21} & \mathbf{S}_{20} \\ \mathbf{S}_{12} & \mathbf{S}_{11} & \mathbf{S}_{10} \\ \mathbf{S}_{02} & \mathbf{S}_{01} & \mathbf{S}_{00} \end{bmatrix} \begin{bmatrix} \mathbf{z}_2 \\ \mathbf{z}_1 \\ z_0 \end{bmatrix} = \mathbf{r}^t\mathbf{r}, \tag{8.8}$$

wobei gilt

$$\mathbf{S}_{ij} = \mathbf{M}_i^t\mathbf{M}_j \text{ und } \mathbf{S}_{ij} = \mathbf{S}_{ji}^t \text{ sowie } \mathbf{S}_{00} = N.$$

Wir erhalten die Lösung der Gleichung (8.7), wenn wir die partiellen Ableitungen nach $\mathbf{z}$ von (8.8) bilden und diese gleich Null setzen

$$\mathbf{S}_{22}\mathbf{z}_2 + \mathbf{S}_{21}\mathbf{z}_1 + \mathbf{S}_{20}z_0 = 0 \tag{8.9}$$

$$\mathbf{S}_{21}^t\mathbf{z}_2 + \mathbf{S}_{11}\mathbf{z}_1 + \mathbf{S}_{10}z_0 = 0 \tag{8.10}$$

$$\mathbf{S}_{20}^t\mathbf{z}_2 + \mathbf{S}_{10}^t\mathbf{z}_1 + Nz_0 = 0. \tag{8.11}$$

Wir lösen Gleichung (8.11) nach z_0 auf und erhalten

$$\begin{aligned} z_0 &= -\frac{1}{N} \begin{bmatrix} \mathbf{S}_{20}^t & \mathbf{S}_{10}^t \end{bmatrix} \begin{bmatrix} \mathbf{z}_2 \\ \mathbf{z}_1 \end{bmatrix} \\ &= -\frac{1}{N} \begin{bmatrix} \overline{[v^{(1)}]^2} & \overline{v^{(1)}v^{(2)}} & \overline{[v^{(2)}]^2} & \overline{v^{(1)}} & \overline{v^{(2)}} \end{bmatrix} \begin{bmatrix} \mathbf{z}_2 \\ \mathbf{z}_1 \end{bmatrix}. \end{aligned} \tag{8.12}$$

Haben wir die eingangs besprochene Anpassung der Daten durchgeführt, dann sind die Konturdaten mittelwertfrei und wir erhalten stets

$$z_0 = -\frac{1}{N} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{z}_2 \\ \mathbf{z}_1 \end{bmatrix} = 0.$$

Wenden wir auf die Matrix $\mathbf{M}_2$ die folgende Transformation $T_\mathbf{M}$ an

$$T_\mathbf{M}: \ \mathbf{M}_2 \rightarrow \begin{bmatrix} \left(v_0^{(1)}\right)^2 - \overline{[v^{(1)}]^2} & v_0^{(1)}v_0^{(2)} - \overline{v^{(1)}v^{(2)}} & \left(v_0^{(2)}\right)^2 - \overline{[v^{(2)}]^2} \\ \vdots & \vdots & \vdots \\ \left(v_N^{(1)}\right)^2 - \overline{[v^{(1)}]^2} & v_N^{(1)}v_N^{(2)} - \overline{v^{(1)}v^{(2)}} & \left(v_N^{(2)}\right)^2 - \overline{[v^{(2)}]^2} \end{bmatrix}, \tag{8.13}$$

so ist Gleichung (8.12) für $z_0 = 0$ erfüllt und unser Minimierungsproblem reduziert sich auf

$$\mathbf{S}_{22}\mathbf{z}_2 + \mathbf{S}_{21}\mathbf{z}_1 = 0 \tag{8.14}$$
$$\mathbf{S}_{21}^t\mathbf{z}_2 + \mathbf{S}_{11}\mathbf{z}_1 = 0. \tag{8.15}$$

Lösen wir Gleichung (8.15) nach $\mathbf{z}_1$ auf und setzten diese Lösung, gemeinsam mit der Bedingung $z_0 = 0$, in Gleichung (8.8) ein, so erhalten wir eine Funktion von $\mathbf{z}_2$ der Gestalt

$$\mathbf{z}_2^t \left(\mathbf{S}_{22} - \mathbf{S}_{21}\mathbf{S}_{11}^{-1}\mathbf{S}_{21}^t\right)\mathbf{z}_2 = \mathbf{r}^t\mathbf{r}.$$

Wir bezeichnen die Matrix

$$\mathbf{M}^* = \mathbf{S}_{22} - \mathbf{S}_{21}\mathbf{S}_{11}^{-1}\mathbf{S}_{21}^t \tag{8.16}$$

als *reduzierte Design-Matrix*. $\mathbf{M}^*$ ist das Schur-Komplement von $\mathbf{S}_{11}$ in $\mathbf{M}$.

8.2.3 Ausgleichsrechnung mit quadratischen Nebenbedingungen

Wir können die lineare Minimierungsaufgabe mit quadratischen Nebenbedingungen wie folgt formulieren. Wir suchen

$$\mathbf{z}_2^t\mathbf{M}^*\mathbf{z}_2 = \min_{\mathbf{z}\neq 0} \text{ unter den Nebenbedingungen } \mathbf{z}_2^t\mathbf{C}\mathbf{z}_2 = \alpha,\ \alpha \in \mathbb{R}. \tag{8.17}$$

Dabei ist $\mathbf{C}$ die Matrix der quadratischen Nebenbedingungen in den Variablen a, b und c. Der Arbeit [Bo79] können wir entnehmen, dass das Minimierungsproblem (8.17) als ein Langrange-Multiplikatoren-Problem aufgefasst und somit als verallgemeinertes Eigenwert-Problem gelöst werden kann. Setzen wir die zu minimierende Funktion und die Nebenbedingungen mit den Langrange-Multiplikatoren zusammen, so erhalten wir das System

$$H(\mathbf{z}_2) = \mathbf{z}_2^t\mathbf{M}^*\mathbf{z}_2 + \lambda(\mathbf{z}_2^t\mathbf{C}\mathbf{z}_2 - \alpha),\ \lambda \in \mathbb{R},$$

welches wir über die partiellen Ableitungen

$$\begin{aligned}\mathbf{M}^*\mathbf{z}_2 + \lambda\mathbf{C}\mathbf{z}_2 &= 0 \\ \mathbf{z}_2^t\mathbf{C}\mathbf{z}_2 &= \alpha\end{aligned} \tag{8.18}$$

lösen können. Als Lösung von (8.18) erhalten wir einen Eigenvektor $\mathbf{e}_j$, der die Gleichung $\mathbf{z}_2^t\mathbf{M}^*\mathbf{z}_2$ minimiert. Für die Lösung $\mathbf{e}_j$ von (8.18) und den zugehörigen Eigenwert λ_j gilt

$$\text{sign}(\lambda_j) = \text{sign}(\mathbf{e}_j^t\mathbf{C}\mathbf{e}_j).$$

Ist zudem die Matrix $\mathbf{C}$ nicht singulär, so vereinfacht sich das Minimierungsproblem (8.17) und wir benötigen die Lösung des Eigenwertproblems

$$\mathbf{C}^{-1}\mathbf{M}^{*}\mathbf{z}_2 = \lambda\mathbf{z}_2. \tag{8.19}$$

Wir wenden gemäß [Fi99] die Bedingung für eine Hyperbel bzw. eine Ellipse an

$$b^2 - 4ac = \mathbf{z}_2^t\mathbf{C}\mathbf{z}_2 = \mathbf{z}_2^t \begin{bmatrix} & & -2 \\ & 1 & \\ -2 & & \end{bmatrix} \mathbf{z}_2 = \alpha \tag{8.20}$$

und sehen, dass wir auf Grund der Invertierbarkeit der Matrix $\mathbf{C}$, stets das Minimierungsproblem mit Hilfe von (8.19) lösen können. Wir können zeigen, dass zwei der Eigenvektoren zu der besten elliptischen und hyperbolischen Lösung führen. Wir ermitteln diese Lösungen, indem wir die Komponenten der Eigenvektoren $\mathbf{e}_1$, $\mathbf{e}_2$ und $\mathbf{e}_3$ in die Bedingung $b^2 - 4ac$ einsetzen. Wir definieren hierfür

$$\xi_j = \left(e_j^{(2)}\right)^2 - 4e_j^{(1)}e_j^{(3)} \text{ für } \mathbf{e}_j = \begin{bmatrix} e_j^{(1)} \\ e_j^{(2)} \\ e_j^{(3)} \end{bmatrix} \text{ sowie } j = 1, 2, 3.$$

Damit erhalten wir die Lösung für die Ausgleichsellipse als

$$\mathbf{z}_2^{ellipse} = \mathbf{e}_l \text{ mit } l = \min_{j=1,2,3} \xi_j \tag{8.21}$$

und die Lösung für die Ausgleichshyperbel als

$$\mathbf{z}_2^{hyperbel} = \mathbf{e}_n \text{ mit } n = \min_{j\neq l, \xi_j > 0} |\lambda_j| \,. \tag{8.22}$$

λ_j ist der zum Eigenvektor $\mathbf{e}_j$ korrespondierende Eigenwert. Damit ist die Lösung für die Ellipse der Eigenvektor zum kleinsten Eigenwert. Die restlichen beiden Lösungen erzeugen Hyperbeln, wir wählen die Lösung mit dem kleineren Eigenwert.

8.2.4 Die parabolische Lösung

Wollen wir an Stelle einer Hyperbel oder Ellipse eine Parabel fitten, so können wir nicht die Lösung des Eigenwertproblems (8.19) verwenden, weil die Anwendung der Nullbedingung $b^2 - 4ac = 0$ stets zu der trivialen Lösung führt.

Setzen wir hingegen $\mathbf{C} = \mathbf{I}_3$, so lösen wir das Minimierungsproblem (8.17) mit den Nebenbedingungen

$$a^2 + b^2 + c^2 = 1. \tag{8.23}$$

Die zu diesem System bzw. zu $\mathbf{M}^*$ zugehörigen Eigenvektoren $\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3$ mit den korrespondierenden Eigenwerten $\lambda_1, \lambda_2, \lambda_3$ bilden eine Orthonormalbasis des Raums der quadratischen Segmente aller Kegelschnitte. Wir ordnen die Eigenvektoren, so dass für die entsprechenden Eigenwerte

$$|\lambda_1| \geq |\lambda_2| \geq |\lambda_3| \geq 0$$

gilt. Wegen der Nebenbedingung (8.23) liegen die Lösungen des Minimierungsproblems (8.17) mit $\mathbf{C} = \mathbf{I}_3$ auf der Einheitskugel um den Ursprung. Die für eine Parabel erforderliche Bedingung $b^2 - 4ac = 0$ entspricht einem elliptischen Kegel, der die Einheitskugel schneidet, und eine Grenze zwischen der elliptischen und der hyperbolischen Lösung bildet. Die Schnittkurve der elliptischen und der hyperbolischen Quadrik, die durch eine Kurve der vierten Ordnung dargestellt wird, repräsentiert die parabolische Lösung. Einen quadratischen Koeffizienten einer Parabel erhalten wir daher als Linearkombination der Eigenvektoren von $\mathbf{M}^*$, also

$$\mathbf{z}_2^{parabel} = \mathbf{e}_3 + \mathbf{e}_2 s + \mathbf{e}_1 t. \tag{8.24}$$

Betrachten wir die Singulärwerte von $\mathbf{M}^*$, also die Eigenwerte von $(\mathbf{M}^*)^t\mathbf{M}^*$, so sind diese Werte gerade die Entfernungen der entsprechenden Eigenvektoren zu dem Nullraum von $\mathbf{M}^*$. Daher können wir dem Eigenwert mit dem kleinsten Singulärwert die minimale Lösung zuordnen. Wir nehmen also an, dass $\mathbf{e}_3$ die beste Lösung ist und benutzen eine Linearkombination von $\mathbf{e}_2$ und $\mathbf{e}_1$, um die beste parabolische Lösung zu erhalten. Da die Gleichungen homogen sind, benötigen wir lediglich zwei Parameter, um den gesamten Raum zu erfassen.
Den Fehler, der durch die Linearkombination von $\mathbf{e}_2$ und $\mathbf{e}_1$ entsteht, können wir mit Hilfe des Residuums $\mathbf{r}$ wie folgt erfassen

$$\begin{aligned}
\|\mathbf{r}\|_2 = \delta(s,t) &= (\mathbf{z}_2^{parabel})^t (\mathbf{M}^*)^t \mathbf{M}^* (\mathbf{z}_2^{parabel}) \\
&= (\mathbf{e}_3 + \mathbf{e}_2 s + \mathbf{e}_1 t)^t \, (\mathbf{M}^*)^t \mathbf{M}^* \, (\mathbf{e}_3 + \mathbf{e}_2 s + \mathbf{e}_1 t) \\
&= \mathbf{e}_3^t (\mathbf{M}^*)^t \mathbf{M}^* \mathbf{e}_3 + \mathbf{e}_2^t (\mathbf{M}^*)^t \mathbf{M}^* \mathbf{e}_2 s^2 + \mathbf{e}_1^t (\mathbf{M}^*)^t \mathbf{M}^* \mathbf{e}_1 t^2 \\
&= \lambda_3^2 + \lambda_2^2 s^2 + \lambda_1^2 t^2.
\end{aligned} \tag{8.25}$$

Im nächsten Schritt grenzen wir die Menge der Lösungen ein, indem wir die Parabelbedingung $b^2 - 4ac = 0$ anwenden. Es gilt

$$\mathbf{z}_2^{parabel} = \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \mathbf{e}_3 + \mathbf{e}_2 s + \mathbf{e}_1 t$$

und damit ergibt sich für die Parabelbedingung

$$b^2 - 4ac = \left(e_3^{(2)} + e_2^{(2)} s + e_1^{(2)}\right)^2 - 4\left(e_3^{(1)} + e_2^{(1)} s + e_1^{(1)}\right)\left(e_3^{(3)} + e_2^{(3)} s + e_1^{(3)}\right) = 0.$$

Durch Auflösen und Zusammenfassen erhalten wir

$$\varphi(s,t) = \gamma_1 s^2 + \gamma_2 st + \gamma_3 t^2 + \gamma_4 s\gamma_5 t + \gamma_6 = 0 \tag{8.26}$$

mit

$$\begin{aligned}
\gamma_1 &= \left(e_1^{(2)}\right)^2 - 4e_1^{(1)}e_1^{(3)} \\
\gamma_2 &= 2e_2^{(2)}e_1^{(2)} - 4e_2^{(1)}e_1^{(3)} - 4e_1^{(1)}e_2^{(3)} \\
\gamma_3 &= \left(e_2^{(2)}\right)^2 - 4e_2^{(1)}e_2^{(3)} \\
\gamma_4 &= 2e_3^{(2)}e_1^{(2)} - 4e_3^{(1)}e_3^{(1)} - 4e_1^{(1)}e_3^{(3)} \\
\gamma_5 &= 2e_3^{(2)}e_2^{(2)} - 4e_3^{(1)}e_2^{(3)} - 4e_2^{(1)}e_3^{(3)} \\
\gamma_6 &= \left(e_3^{(2)}\right)^2 - 4e_3^{(1)}e_3^{(3)}.
\end{aligned}$$

Dabei entspricht (8.26) einer Parametergleichung für einen Kegel. Um zu unseren Daten eine Parabel berechnen zu können, müssen wir die Fehlergleichung $\delta(s,t)$ aus (8.25) unter den Nebenbedingungen $\varphi(s,t)$ aus (8.26) minimieren. Dieses Minimierungsproblem können wir unter Verwendung der Lagrange-Multiplikatoren als

$$H(s,t) = \delta(s,t) + \mu\varphi(s,t) \tag{8.27}$$

mit $\mu \in \mathbb{R}$ formulieren. Bestimmen wir die partiellen Ableitungen von $H(s,t)$, so erhalten wir ein Polynom vom Grad 4 in μ

$$K_4\mu^4 + K_3\mu^3 + K_2\mu^2 + K_1\mu + K_0 = 0 \tag{8.28}$$

mit

$$\begin{aligned}
K_0 &= 16\gamma_6\alpha_3^2 & K_1 &= -8\alpha_3(k_1\alpha_2 + k_4\alpha_1) \\
K_2 &= 4\left[(2\gamma_2 k_2 + 4k_8)\,\alpha_3 + \gamma_1 k_4\alpha_1^2 + \gamma_3 k_1\alpha_2^2\right] & K_3 &= 2k_7k_8 \\
K_4 &= k_5k_8
\end{aligned}$$

und

$$\begin{aligned}
&\alpha_1 = \lambda_1^2 && \alpha_2 = \lambda_2^2 && \alpha_3 = \alpha_1\alpha_2 \\
&k_1 = 4\gamma_3\gamma_6 - \gamma_5^2 && k_2 = \gamma_2\gamma_6 - \frac{1}{2}\gamma_4\gamma_5 && k_3 = \frac{1}{2}\gamma_2\gamma_5 - \gamma_3\gamma_4 \\
&k_4 = 4\gamma_6\gamma_1 - \gamma_4^2 && k_5 = 4\gamma_1\gamma_3 - \gamma_2^2 && k_6 = \gamma_2\gamma_4 - 2\gamma_1\gamma_5 \\
&k_7 = -4\left(\gamma_1\alpha_1 + \alpha_2\gamma_3\right) && k_8 = \gamma_1 k_1 - \gamma_2 k_2 + \gamma_4 k_3.
\end{aligned}$$

Wir erhalten vier Lösungen der Gleichung (8.28), die wir mit $\mu_1, \ldots, \mu_4$ bezeichnen. Die Lösung für die Ausgleichsparabel ist der Lagrange-Multiplikator mit dem kleinsten Absolutbetrag

$$\mu_* = \min_{1 \le j \le 4} |\mu_j| \text{ mit } \mu_j \in \mathbb{R}.$$

Wegen (8.27) gilt stets $\nu_j \in \mathbb{R}$ für $1 \le j \le 4$. Die Rücktransformation ergibt für die zugehörigen Werte s_* und t_*

$$s_* = \frac{2\mu_*}{u_*}\left(k_3\mu_* + \alpha_1\gamma_4\right) \text{ und } t_* = \frac{\mu_*}{u_*}\left(k_6\mu_* + \alpha_2\gamma_5\right)$$

mit

$$u_* = k_5\mu_*^2 + k_7\mu_* + 4\alpha_3.$$

Die quadratischen Parameter für eine Ausgleichsparabel erhalten wir als

$$\mathbf{z}_2^{parabel} = \mathbf{e}_3 + \mathbf{e}_2 s_* + \mathbf{e}_1 t_*. \tag{8.29}$$

8.2.5 Rücktransformation

Durch die Rücktransformation berechnen wir die Verschiebung und Skalierung der Asymptoten. Haben wir eine quadratische Lösung $\mathbf{z}_2$ berechnet, also eine Ellipse, Hyperbel oder Parabel, so ist die Transformation stets dieselbe. Denn wenn wir die quadratischen Koeffizienten kennen, so ist uns damit die Richtung der Asymptoten bekannt. Wir können die Rücktransformation in Matrixform angeben als

$$\mathbf{z} = \begin{bmatrix} \mathbf{I}_3 \\ \mathbf{S}_{11}^{-1}\mathbf{S}_{21}^t \\ -\overline{[v^{(1)}]^2} \quad -\overline{v^{(1)}v^{(2)}} \quad -\overline{[v^{(2)}]^2} \end{bmatrix} \mathbf{z}_2 = \mathbf{B}\mathbf{z}_2. \tag{8.30}$$

Dabei gilt $\mathbf{B} \in \mathbb{R}^{6\times 3}$ und $\mathbf{z} \in \mathbb{R}^6$ ist der Vektor aller Kegelschnittkoeffizienten, also der Grassmann-Koordinaten des Kegelschnittsegments.

Als Letztes müssen wir die eingangs angewendete Transformation der Daten rückgängig machen, das heißt die Daten und das Kegelschnittsegment an die ursprüngliche Position verschieben. Dazu führen wir eine Ähnlichkeitstransformation durch und erhalten

$$\mathbf{K}_* = \mathbf{T}^t\mathbf{K}\mathbf{T} \text{ mit } \mathbf{T} = \begin{bmatrix} \sigma & & -\sigma\overline{v^{(1)}} \\ & \sigma & -\sigma\overline{v^{(2)}} \\ & & 1 \end{bmatrix}. \tag{8.31}$$

Zusammenfassend erhalten wir den folgenden Algorithmus zur Berechnung der drei Arten von Kegelschnittsegmenten aus zweidimensionalen Messdaten.

8.2 Algorithmus: Sei $K = \{\mathbf{v}_j \mid j = 0, \ldots, N-1\} \subset \mathbb{R}^2$ eine Kontur, also eine Menge von Messdaten. Dann können wir wie folgt eine Ausgleichsellipse, -hyperbel oder -parabel berechnen.

(1) Wir erzeugen aus K skalierte und zentrierte Daten, also

$$\widetilde{\mathbf{v}}_j = \sigma(\mathbf{v}_j - \overline{\mathbf{v}}) \text{ für } j = 0, \ldots, N-1.$$

(2) Wir führen eine lineare Regression der Daten $\widetilde{v}_j^{(1)}$ und $\widetilde{v}_j^{(2)}$ durch und verändern gegebenenfalls die Skalierung σ, bis das Residuum groß genug ist.

(3) Jetzt erzeugen wir die quadratische Designmatrix $\mathbf{M}$ und subtrahieren die Mittelwerte von den Matrixpalten. Dann können wir die reduzierte Designmatrix $\mathbf{M}^*$ als

$$\mathbf{M}^* = \mathbf{S}_{22} - \mathbf{S}_{21}\mathbf{S}_{11}^{-1}\mathbf{S}_{21}^t$$

bestimmen.

(4) Für die Berechnung einer Ellipse oder Hyperbel bestimmen wir die Eigenvektoren von $\mathbf{C}^{-1}\mathbf{M}^*$, wobei $\mathbf{C}$ die Nebenbedingung $b^2 - 4ac = \alpha$ definiert. Wir wählen die entsprechende Lösung gemäß den Vorschriften (8.21) und (8.22).

(5) Für eine Parabel lösen wir das Eigenvektorproblem ohne Nebenbedingungen. Dann bestimmen wir das Polynom vierter Ordnung

$$K_4\mu^4 + K_3\mu^3 + K_2\mu^2 + K_1\mu + K_0 = 0$$

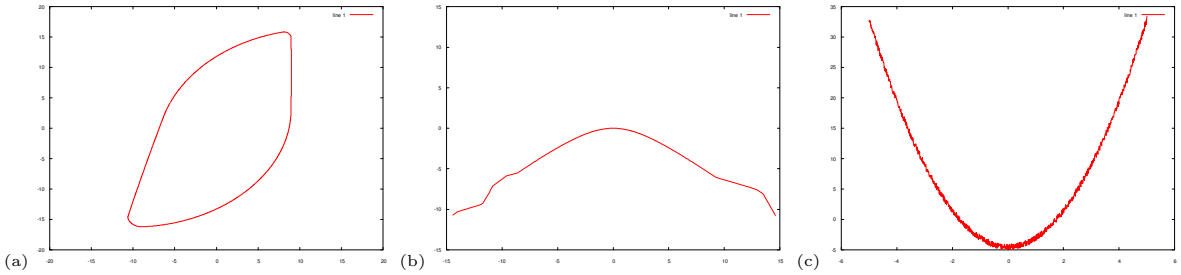

Abbildung 28: Konturen (a) Ellipse, (b) Hyperbel und (c) Parabel.

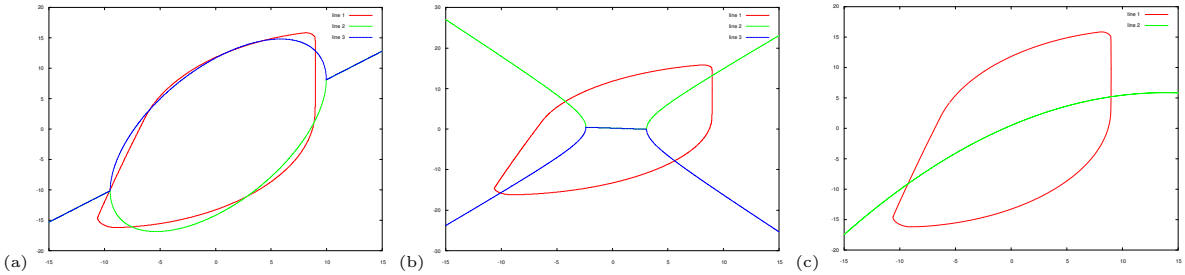

Abbildung 29: Fit einer Ellipse (a), einer Hyperbel (b) und einer Parabel (c) zu der Ellipse-Kontur.

und erhalten die Lösung

$$\mu_* = \min_{1 \le j \le 4} \mu_j \text{ mit } \mu_j \in \mathbb{R}.$$

Mit Hilfe der kleinsten Lösung μ_* können wir die quadratischen Parabelkoeffizienten berechnen.

(6) Wir erhalten jetzt den Koeffizientenvektor $\mathbf{z}$ als Lösung von $\mathbf{z} = \mathbf{B}\mathbf{z}_2$. Dann bestimmen wir die Conic-Matrix $\mathbf{K}$ und wenden die Ähnlichkeitstransformation $\mathbf{K}_* = \mathbf{T}^t\mathbf{K}\mathbf{T}$ an.

8.2.6 Ergebnisse

Wir betrachten drei verschiedene Konturen, abgebildet in Abbildung 28, und berechnen zu jeder der Konturen alle drei Arten von Kegelschnitten. Des weiteren gehen wir der Frage nach der Gesamtkomplexität nach.
Wir starten mit der Ellipse-Kontur, wenden Algorithmus 8.2 dreimal mit den Optionen Ellipse fitten, Hyperbel fitten und Parabel fitten an und erhalten die in Abbildung 29 ersichtlichen drei Ergebnisse.
Im nächsten Schritt wenden wir Algorithmus 8.2 erneut dreimal an, diesmal auf die Daten der Hyperbel. Die Ergebnisse sehen wir in Abbildung 30.

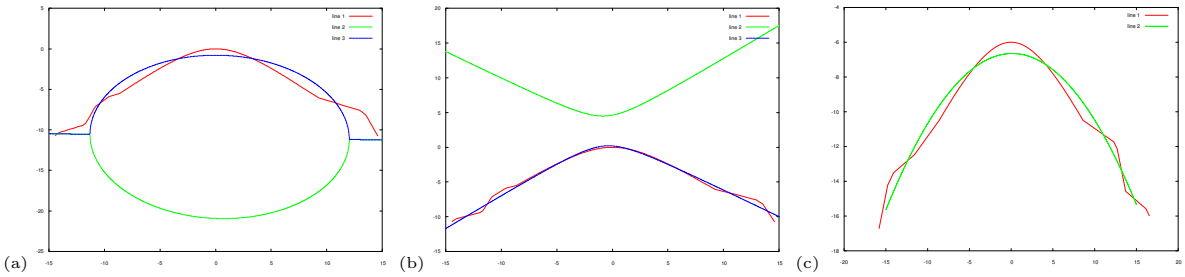

Abbildung 30: Fit einer Ellipse (a), einer Hyperbel (b) und einer Parabel (c) zu der Hyperbel-Kontur.

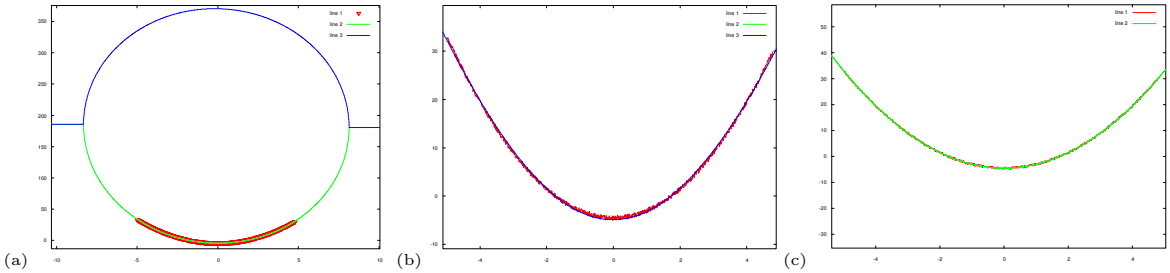

Abbildung 31: Fit einer Ellipse (a), einer Hyperbel (b) und einer Parabel (c) zu der Parabel-Kontur.

Wir wiederholen den obigen Versuch mit den Daten der Parabel und erhalten die Fits aus Abbildung 31.
Um die Gesamtkomplexität des Algorithmus angeben zu können, führen wir eine Laufzeituntersuchung durch.

(1) Für die Anpassung der Daten brauchen wir insgesamt $9N + 2$ Operationen, den Speicherplatz können wir an dieser Stelle vernachlässigen.

(2) Die lineare Regression ist mit einem geeigneten Verfahren in $O(N)$ Zeit durchführbar.

(3) Für das Erzeugen der Designmatrix $\mathbf{M}$ benötigen wir $3N$ Rechenoperationen, für die Transformation $T_{\mathbf{M}}$ $4N$ Rechenoperationen und insgesamt $5N$ Speicherplatz. Weitere Schritte sind die Berechnung der Matrizen $\mathbf{S}_{ij}$. Für diese Operationen brauchen wir insgesamt $19N$ Rechenoperationen, den Speicherbedarf können wir als konstant ansehen.

Die restlichen Punkte (4) bis (6) des Algorithmus 8.2 sind sowohl in Hinblick auf die Laufzeit als auch auf den Speicherbedarf unabhängig von N und wir sehen sie als konstant an. Damit benötigen wir insgesamt $35N + c_1$ Rechenoperationen und $5N + c_2$ Speicherplatz. $c_1, c_2 \in \mathbb{N}$ sind dabei von N unabhängige Konstanten. Wir können die folgende Aussage formulieren.

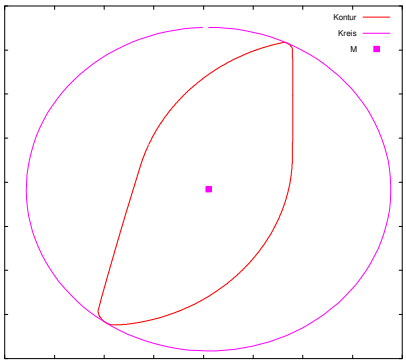

Abbildung 32: Eine Kontur mit einem Hüllkreis

8.3 Korollar: Der Algorithmus 8.2 hat eine Laufzeit und einen Speicherplatzbedarf von jeweils von $O(N)$.

Die tatsächliche Laufzeit des Algorithmus 8.2 lag für alle Testkonturen unter 5 sec. Dabei haben wir Konturen mit einer Größe von bis zu 20.000 Punkten untersucht. Die in Abschnitt 8.2.1 angedeutete Verbesserung der Laufgeschwindigkeit durch die Anpassung der Daten konnten wir nicht beobachten, die tatsächliche Laufzeit war mit und ohne Anpassung in etwa dieselbe. Daher kann man auf die Anpassung und demzufolge auf die Rücktransformation verzichten.

8.3 Hüllkreis

Eine weitere wichtige Anwendung bei der Berechnung von Regelgeometrieelementen sind die maximalen und minimalen Elemente und hier vor allem der *Hüllkreis*, also der minimale Kreis, welcher alle Punkte einer Punktwolke umschließt, sowie der *Pferchkreis*, der maximale Kreis, welcher innerhalb einer Punktwolke liegt.
Die Berechnung eines Hüllkreises kommt zum Beispiel beim Vermessen von Bohrungen und kreisförmigen Objekten oder bei anderen Konturen, die einen Kreis als Bezugsobjekt haben, vor. In Abbildung 32 sehen wir eine Kontur mit 1714 Punkten und einem mit dem nachfolgend vorgestellten Algorithmus berechneten Hüllkreis.
Die Definition eines Kreises aus Abschnitt 7 ist hierfür etwas unhandlich, daher legen wir als Erstes fest, was wir unter einem Kreis im $\mathbb{R}^2$ bzw. einer Kugel im $\mathbb{R}^d$ für $d > 2$ in diesem Abschnitt verstehen.

8.4 Definition: Unter einer *Kugel* $D \in \mathbb{R}^d$ verstehen wir die Menge aller Punkte $\mathbf{b} \in \mathbb{R}^d$, für die gilt

$$D = \{\mathbf{b} \in \mathbb{R}^d \mid \|\mathbf{b} - \mathbf{m}\|_2 = r\}.$$

Wir bezeichnen $\mathbf{m} \in \mathbb{R}^d$ als den *Mittelpunkt* und $r > 0$ als *Radius* der Kugel. Wir schreiben $D = (\mathbf{m}, r)$. Für $d = 2$ heißt D *Kreis*.
Für $\mathbf{v} \in D$ verwenden wir die Sprechweise, $\mathbf{v}$ *liegt am Rand von* D. Wir definieren das *Innere* eines Kreises bzw. einer Kugel $D = (\mathbf{m}, r)$ als $D^\circ = \{\mathbf{v} \in \mathbb{R} \mid \|\mathbf{v} - \mathbf{m}\|_2 < r\}$. Ist $\|\mathbf{v} - \mathbf{m}\|_2 > r$, so bezeichnen wir die Situation mit $\mathbf{v} \notin D^\circ \cup D$ und sagen $\mathbf{v}$ *liegt außerhalb von* D.

Der wesentliche Unterschied zu der Sichtweise aus Abschnitt 7 besteht darin, dass wir in diesem Kontext lediglich für $d = 2$ einen Kreis haben und für $d > 2$ eine Kugel vorliegt, während es sich in Abschnitt 7 für alle Dimensionen $d \geq 2$ stets um einen ebenen Kreis handelt.
Wir beschäftigen uns in diesem Abschnitt zunächst mit der Berechnung des Hüllkreises. Auf den Algorithmus zur Berechnung des Pferchkreises gehen wir im nächsten Abschnitt ein.
Wir beginnen mit einer Definition, der Problemstellung sowie einer einfachen ersten Beobachtung. Wir nehmen für den Rest des Kapitels $|K| > 1$ und $r_D > 0$ an.

8.5 Definition: Sei $K = \{\mathbf{v}_j \mid j = 0, \ldots, N-1\} \subset \mathbb{R}^d$ eine Kontur. Für $d = 2$ verstehen wir unter einem *Hüllkreis* einen Kreis $D_H(K)$ mit Radius $r > 0$ und Mittelpunkt $\mathbf{m} \in \mathbb{R}^d$, so dass gilt

$$D_H(K) = (\mathbf{m}, r) = \min_{r_1 > 0} \{E = (\mathbf{a}, r_1) \mid K \subseteq E^\circ \cup E\}.$$

Analog ist für $d > 2$ die *Hüllkugel* definiert als

$$D_H(K) = (\mathbf{m}, r) = \min_{r_1 > 0} \{E = (\mathbf{a}, r_1) \mid K \subseteq E^\circ \cup E\}.$$

Wir wollen zunächst den Algorithmus zur Berechnung eines Hüllkreises (also für $d = 2$) erläutern und uns anschließend mit der Fragestellung der Berechnung für ein allgemeines $d > 2$ beschäftigen. Bei unseren Ausführungen orientieren wir uns an [Gae96], Seiten 97-101.

8.6 Lemma: Sei $K \subset \mathbb{R}^2$ eine Menge und sei $D_H(K)$ ein Hüllkreis von K. Dann existiert $D_H(K)$ und ist eindeutig.

Beweis: Die Existenz von $D_H(K)$ ist offensichtlich. Seien für die Eindeutigkeit D_1 und D_2 zwei Kreise mit demselben Radius r und Mittelpunkten $\mathbf{z}_1$ und $\mathbf{z}_2$. Gilt $K \subset D_1$ und $K \subset D_2$, dann ist auch $K \subset D_1 \cap D_2$. $D_1 \cap D_2$ ist enthalten in einem Kreis D mit dem Mittelpunkt $\mathbf{z} = \frac{1}{2}(\mathbf{z}_1 + \mathbf{z}_2)$ und Radius $\sqrt{r^2 - r_1^2}$ mit $r_1 = \frac{1}{2}\|\mathbf{z}_1 - \mathbf{z}_2\|$. Dann muss $r_1 = 0$ gelten, sonst hat D einen kleineren Radius als D_1 und D_2. Damit gilt aber $D = D_1 = D_2$. Also ist $D_H(K)$ eindeutig. □

Wir können $D_H(K)$ durch höchstens drei Punkte von K, welche am Rand liegen, bestimmen. Mit anderen Worten existiert eine Teilmenge $S \subset K$ mit $|S| \leq 3$, die am Rand von K liegt und für die gilt $D_H(K) = D_H(S)$ und falls $\mathbf{v} \notin S$, so ist $D_H(K \backslash \{\mathbf{v}\}) = D_H(K)$. Gilt andererseits $D_H(K \backslash \{\mathbf{v}\}) \neq D_H(K)$, dann ist $\mathbf{v} \in S$ und $\mathbf{v}$ liegt am Rand von $D_H(K)$.
Nach der Definition eines Spezialfalls des Kreises $D_H(K)$ skizzieren wir den Verlauf der Berechnung von $D_H(K)$.

8.7 Definition: Seien $K, J \subset \mathbb{R}^2$ zwei Konturen. Falls ein solcher existiert, nennen wir den *kleinsten, K umschließenden Kreis mit der Punktmenge J auf dem Rand* $D_H(K, J)$.
Weiter ist $D_H(\emptyset, J)$ der *kleinste Kreis mit allen Punkten von J auf dem Rand.*
Eine *erzeugende Menge* S von $D_H(K, J)$ ist eine Teilmenge S von K derart, dass $|S| \leq 3$ und $D_H(K, J) = D_H(S, J)$ ist.

Für eine Kontur $K = \{\mathbf{v}_j \mid j = 0, \ldots, N-1\}$ wollen wir die $D_H(K)$ inkrementell berechnen. Wir starten mit der leeren Menge, fügen nach und nach die Punkte von K hinzu und berechnen jeweils den kleinsten Kreis aus den bereits hinzugefügten Punkten. Wir nehmen an, wir haben $D = D_H(\{\mathbf{v}_0, \ldots, \mathbf{v}_j\})$ für ein $0 < j < N-1$ bereits berechnet. Für den Punkt $\mathbf{v}_{j+1}$ gibt es dann zwei Möglichkeiten.

(1) $\mathbf{v}_{j+1} \in D \cup D^\circ$, das heißt $\mathbf{v}_{j+1}$ liegt auf dem Rand oder im Inneren des Kreises, dann ist D der kleinste Kreis für die Menge $\{\mathbf{v}_0, \ldots \mathbf{v}_{j+1}\}$ und wir fahren mit dem Punkt $\mathbf{v}_{j+2}$ fort.

(2) $\|\mathbf{v}_{j+1} - \mathbf{m}\|_2 > r$, also liegt $\mathbf{v}_{j+1}$ außerhalb von D, dann liegt $\mathbf{v}_{j+1}$ auf dem Rand des Kreises $D' = D_H(\{\mathbf{v}_0, \ldots, \mathbf{v}_{j+1}\})$. Wir berechnen D' als $D_H(\{\mathbf{v}_0, \ldots, \mathbf{v}_j\}, \mathbf{v}_{j+1})$.

Wir sehen die Möglichkeiten (1) und (2) in Abbildung 33 dargestellt.

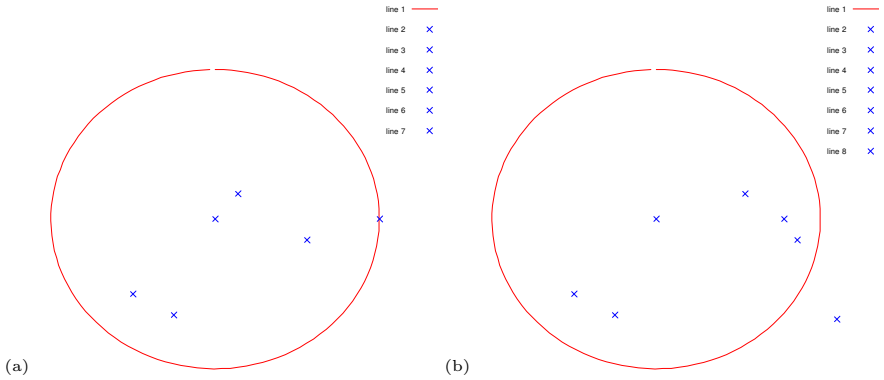

Abbildung 33: (a) Möglichkeit (1), $\mathbf{v}_{j+1}$ auf dem Rand von D. (b) Möglichkeit (2), $\mathbf{v}_{j+1}$ außerhalb von D

Wir stellen den Algorithmus zur Berechnung des kleinsten umschließenden Kreises vor und verwenden dabei bereits die Berechnung von $D_H(K, J)$, die wir in Algorithmus 8.10 erörtern werden.

8.8 Algorithmus:

```
function Kreis= D_H(K)
  if (K = ∅)
    D = ∅;
  else
    waehle v ∈ K;
    D = D_H(K\{v});
    if(v ∉ D)
      D = D_H(K\{v}, v);
  return  D;
```

Bevor wir uns mit der Berechnung von $D_H(K, J)$ beschäftigen, gehen wir kurz auf die Komplexität von Algorithmus 8.8 ein. Wir wählen $\mathbf{v} \in K$ zufällig und gehen davon aus, dass die Wahl mit der Wahrscheinlichkeit $\frac{1}{|K|}$ erfolgt. Sei $t(N)$ die erwartete Anzahl von Schritten des Algorithmus 8.8 für $N = |K|$. Dann gilt

$$t(N) \leq 1 + t(N-1) + \mathbb{P}(\mathbf{v} \notin D_H(K\backslash\{\mathbf{v}\}))c(N-1),$$

wenn wir annehmen, dass die Berechnung von $D_H(K, \mathbf{v})$ in cN Schritten erfolgt. Der zweite und dritte Summand der obigen Ungleichung sind Abschätzungen für die Anzahl der Schritte bei der Berechnung von $D_H(K)$ und $D_H(K, J)$. Mit $\mathbb{P}(\cdot)$ bezeichnen wir die Wahrscheinlichkeit. Wir fin-

den höchstens drei Punkte $\mathbf{v} \in K$, so dass $D_H(K) \neq D_H(K\backslash\{\mathbf{v}\})$ gilt. Für alle übrigen Punkte erhalten wir $\mathbf{v} \in D_H(K\backslash\{\mathbf{v}\})$ und damit $\mathbb{P}(\mathbf{v} \notin D_H(K\backslash\{\mathbf{v}\})) \leq \frac{3}{N}$. Damit ergibt sich die Abschätzung

$$t(N) \leq (1 + 3c)(N - 1).$$

Wir wenden uns jetzt der Berechnung von $D_H(K, J)$ zu. Wir haben erwähnt, dass $D_H(K, J)$ nicht immer existieren muss. Das folgende Lemma beschreibt einige nützliche Eigenschaften von $D_H(K, J)$.

8.9 Lemma: Seien $K, J \subset \mathbb{R}^2$ sowie $K \neq \emptyset$ und $\mathbf{v} \in K$.

(1) Existiert ein Kreis um K mit J auf dem Rand, dann existiert auch $D_H(K, J)$ und ist eindeutig.

(2) Gilt $\mathbf{v} \notin D_H^\circ(K\backslash\{\mathbf{v}\}, J)$, dann liegt $\mathbf{v}$ auf dem Rand von $D_H(K, J)$, falls $D_H(K, J)$ existiert. Das heißt es ist

$$D_H(K, J) = D_H(K\backslash\{\mathbf{v}\}, J \cup \{\mathbf{v}\}).$$

(3) Existiert $D_H(K, J)$, so gibt es eine erzeugende Menge $S \subseteq K$ mit $|S| \leq \max(0, 3 - |J|)$, so dass $D_H(K, J) = D_H(S, J)$.

Beweis: Wir verweisen auf den Beweis der Proposition 8.12, da Lemma 8.9 ein Spezialfall dieser Aussage ist. □

Eine wichtige Beobachtung von 8.9 ist, dass die Menge S nicht eindeutig ist. Weiter folgt aus 8.9 (3), dass höchstens $\max(0, 3 - |J|)$ Punkte in K existieren, für die $\mathbf{v} \notin D_H(K\backslash\{\mathbf{v}\}, J)$ gilt.
Lemma 8.9 (2) deutet an, wie wir $D_H(K, J)$ berechnen können. Für $K = \emptyset$ können wir $D_H(\emptyset, J)$ direkt angeben. Anderenfalls wählen wir einen zufälligen Punkt $\mathbf{v} \in K$ und bestimmen $D = D_H(K\backslash\{\mathbf{v}\}, J)$. Ist $\mathbf{v} \in D$, so ist $D_H(K, J) = D$, sonst ist $D_H(K, J) = D_H(K\backslash\{\mathbf{v}\}, J \cup \{\mathbf{v}\})$.
Wir können den angedeuteten Algorithmus zur Berechnung des Kreises $D_H(K, J)$ formalisieren. $D_H(K, J)$ wird natürlich nur dann berechnet, wenn er existiert. Wir setzen $D_H(\emptyset, \emptyset) = \emptyset$.

8.10 Algorithmus:

```
function Kreis= D_H(K, J)
  if(K = ∅  OR  |J| == 3)
    D = D_H(∅, J);
  else
    waehle v ∈ K;
    D = D_H(K\{v}, J);
    if(D  existiert AND  p ∉ D)
      D = D_H(K\{v}, J ∪ {v});
  return  D;
```

Da Algorithmus 8.10 bereits einen Hüllkreis liefert, vereinfacht sich der Algorithmus 8.8 zu der folgenden Funktion.

```
function Kreis= D_H(K)
  return  D_H(K, ∅);
```

Da $D_H(K, \emptyset)$ stets existiert, liefert während der gesamten Berechnung die Funktion $D_H(K, J)$ niemals undefinierte Ergebnisse.
Als nächstes machen wir eine Aussage über die Gesamtkomplexität des oben beschriebenen Verfahrens.

8.11 Satz: Das in Algorithmus 8.10 beschriebene Verfahren zur Berechnung des kleinsten umschließenden Kreises von N Punkten in der Ebene hat die Komplexität $O(N)$.

Beweis: Die Zeitanalyse bedient sich des Prinzips der Rückwärtsanalyse, beschrieben in [Se93]. Wir wollen als Erstes die Anzahl der Aufrufe des Tests $\mathbf{v} \notin D$ bestimmen. Dann ist die Anzahl der Schritte ein Vielfaches dieses Wertes.
Sei $t_j(N)$ für $0 \leq j \leq 3$ die Anzahl der Aufrufe von $D_H(K, J)$ mit $|K| = N$ und $|J| = 3 - j$. Dann erhalten wir $t_0(N) = 0$ sowie $t_j(0) = 0$.
Für $j > 0$ und $N > 0$ rufen wir

(a) einmal $D_H(K\backslash\{\mathbf{v}\}, J)$ mit einem zufällig gewählten Punkt $\mathbf{v} \in K$,

(b) einmal den Test $\mathbf{v} \notin D^\circ$ und

(c) einmal $D_H(K\backslash\{\mathbf{v}\}, J \cup \{\mathbf{v}\})$ auf, wobei $D_H(K, J) \neq D_H(K\backslash\{\mathbf{v}\}, J)$ gilt .

Lemma 8.9 (3) zeigt uns, dass die Wahrscheinlichkeit, dass Punkt (c) eintritt, höchstens $\frac{j}{N}$ beträgt. Wir erhalten die Rekursion

$$t_j(N) \leq t_j(N-1) + 1 + \frac{j}{N} t_{j-1}(N-1) \tag{8.32}$$

und damit $t_1(N) \leq N$, $t_2(N) \leq 3N$ und $t_3(N) \leq 10N$. So erhalten wir eine Komplexität von $O(N)$. □

Wie angekündigt beschäftigen wir uns jetzt mit der Möglichkeit, eine Hüllkugel im $\mathbb{R}^d$ für $d > 2$ zu berechnen.
Wir können Algorithmus 8.10 für die Berechnung einer Hüllkugel zu einer Kontur K im $\mathbb{R}^d$ benutzen, wir müssen nur die Konstante 3 in der if-Abfrage in 8.10 durch $d+1$ ersetzen. Denn eine Kugel im $\mathbb{R}^d$ wird durch $d+1$ Punkte definiert. Wir erhalten die folgende Aussage.

8.12 Proposition: Seien $K, J \subset \mathbb{R}^d$, $d \geq 2$, $K \neq \emptyset$ und $\mathbf{v} \in K$.

(1) Existiert eine Kugel um K mit J auf dem Rand, dann existiert auch $D_H(K, J)$ und ist eindeutig.

(2) Gilt $\mathbf{v} \notin D_H^\circ(K \backslash \{\mathbf{v}\}, J))$, dann liegt $\mathbf{v}$ auf dem Rand von $D_H(K, J)$, falls $D_H(K, J)$ existiert. Das heißt es ist

$$D_H(K, J) = D_H(K \backslash \{\mathbf{v}\}, J \cup \{\mathbf{v}\}).$$

(3) Existiert $D_H(K, J)$, so gibt es eine erzeugende Menge $S \subseteq K$ mit $|S| \leq \max(0, d + 1 - |J|)$, so dass $D_H(K, J) = D_H(S, J)$ gilt.

Beweis: Wir beweisen die Punkte (1) bis (3) separat. Vor dem eigentlichen Beweis benötigen wir etwas Notation.
Wir sprechen im Beweis pauschal von d–dimensionalen Kugeln, die Aussagen können wir aber genauso auf zweidimensionale Kreise anwenden.
Sei $f : \mathbb{R}^d \to \mathbb{R}$ eine Funktion mit

$$f(\mathbf{x}) = \frac{1}{r_1^2} \|\mathbf{x} - \mathbf{a}\|_2^2.$$

Dabei ist $\mathbf{a} \in \mathbb{R}^d$ und $r_1 > 0$. Dann können wir eine Kugel $E = (\mathbf{a}, r_1)$ als

$$E = \{\mathbf{x} \in \mathbb{R}^d \mid f(\mathbf{x}) = 1\}$$

schreiben. Das Innere der Kugel E können wir entsprechend als

$$E^\circ = \{\mathbf{x} \in \mathbb{R}^d \mid f(\mathbf{x}) < 1\}$$

darstellen. Seien D_0 und D_1 zwei Kugeln, welche die Funktionen f_0 und f_1 definieren. Für jedes λ mit $0 \leq \lambda \leq 1$ definiert

$$f_\lambda(\mathbf{x}) = (1-\lambda)f_0(\mathbf{x}) + \lambda f_1(\mathbf{x}) = 1 \tag{8.33}$$

eine Menge von Punkten $\mathbf{x} \in \mathbb{R}^d$, die wiederum eine Kugel D_λ bilden. Sind D_0 und D_1 verschieden, dann ist für $0 < \lambda < 1$ der Radius von D_λ kleiner als das Maximum der Radien von D_0 und D_1.

(1) Sei $D(K, J)$ eine Kugel mit Radius $r > 0$, die K enthält und die Menge J auf dem Rand hat. Wir betrachten

$$r_0 = \inf\{r > 0 \mid \text{Es existiert ein Kreis } D(K, J) \text{ mit Radius } r\},$$

dann ist r_0 eine stetige Funktion auf der kompakten Menge der Kugelmittelpunkte. Daher wird r_0 von einer Kugel angenommen, diese Kugel zu r_0 ist $D_H(K, J)$. Damit haben wir die Existenz nachgewiesen. Seien für die Eindeutigkeit zwei kleinste Kugeln D_0 und D_1 gegeben. Dann ist D_λ für $\lambda = \frac{1}{2}$, definiert wie in (8.33), eine Kugel um K mit J auf dem Rand, die einen kleineren Radius als D_0 und D_1 hat, Widerspruch.

(2) Wir nehmen an, dass $\mathbf{v} \notin D_0^\circ = D_H^\circ(K \backslash \{\mathbf{v}\}, J)$ und $\mathbf{v}$ nicht auf dem Rand von $D_1 = D_H(K, J)$ liegt. D_λ verändert sich stetig von D_0 zu D_1, wenn λ linear von 0 nach 1 läuft. Dann existiert ein Parameter $0 < \lambda' < 1$, für welchen $\mathbf{v}$ auf dem Rand von $D_{\lambda'}$ liegt. Dann enthält aber $D_{\lambda'}$ die Kontur K, hat $\mathbf{v}$ und J auf dem Rand und besitzt einen kleineren Radius als D_1, Widerspruch.

(3) Für $|J| \geq d+1$ gilt $D_H(K, J) = D_H(\emptyset, J)$, also ist die Behauptung für $|S| = 0$ erfüllt. Wir nehmen an, dass J höchstens d Punkte enthält. Wir unterscheiden drei Fälle für die Mächtigkeit von J.
$|J| = 0$: Es existieren mindestens d Punkte $\mathbf{v}_0, \ldots, \mathbf{v}_{d-1} \in K$ auf dem Rand, sonst wäre die Kugel $D_H(K, J)$ nicht minimal. Ist die durch $\mathbf{v}_0, \ldots, \mathbf{v}_{d-1}$ erzeugte Kugel $D_H(\{\mathbf{v}_0, \ldots, \mathbf{v}_{d-1}\}, J)$ eindeutig und minimal, so ist $S = \{\mathbf{v}_0, \ldots, \mathbf{v}_{d-1}\}$. Anderenfalls benötigen wir einen weiteren Punkt $\mathbf{v}_d \in K$, um einen eindeutigen und minimalen Kreis zu erhalten. Wir bekommen die Aussage $|S| \leq d+1$.
$|J| = k$ für $0 < k < d$: Um einen minimalen und eindeutigen Kreis $D_H(S, J)$ zu erhalten, benötigen wir analog zum ersten Fall zusätzlich zu den k Konturpunkten aus J mindestens $d-k$ und höchstens $d-k+1$ Punkte. Daher gilt $|S| \leq d-k+1$.

$|J| = d$: Definieren die Punkte aus J bereits einen minimalen und eindeutigen Kreis, so gilt $|S| = 0$, andernfalls ist $|S| = 1$.

□

Als letztes beschäftigen wir uns mit der Frage nach der Komplexität der Berechnung im $\mathbb{R}^d$. Die Rekursion (8.32) gilt ebenfalls, wobei die Laufzeit jetzt als $t_\delta(N)$ für $\delta = d+1$ ermittelt wird. Mit Hilfe vollständiger Induktion können wir zeigen, dass für $j \geq 1$

$$t_j(N) \leq \left(\sum_{k=1}^{j} \frac{1}{k!} \right) j!N = \lfloor (e-1)j! \rfloor n, \tag{8.34}$$

gilt. Die Gleichheit in (8.34) erhalten wir für Konturen, deren $d+1$ Punkte einen regulären Simplex bilden und deren restliche Punkte innerhalb dieses Simplexes liegen.
Für die Gesamtlaufzeit können wir die folgenden Aussagen machen.

8.13 Satz: Sei $K = \{\mathbf{v}_j \mid j = 0, \ldots, N-1\} \subset \mathbb{R}^d$. Die Hüllkugel $D_H(K)$ von K im $\mathbb{R}^d$ kann mit Hilfe des entsprechend angepassten Algorithmus 8.8 in $O(\delta\delta!N) = O(N)$, $\delta = d+1$, berechnet werden.

Beweis: Wir berechnen $t_\delta(N)$ aus (8.34). Es gilt

$$t_\delta(N) \leq \lfloor (e-1)\delta! \rfloor N \leq \delta\delta!N.$$

Für eine feste Dimension $d > 2$ ist $\delta\delta!$ konstant und unabhängig von N, also beträgt die Laufzeit $O(\delta\delta!N) = O(N)$. □

Einen Algorithmus zu Berechnung eines Hüllkreises, bzw. einer Hüllkugel haben wir nun angegeben. Jetzt beschäftigen wir uns noch mit einigen Verbesserungen des Algorithmus 8.10. Der aufwändigste Schritt ist die Berechnung von $D_H(K, J)$. Wir konzentrieren uns folgerichtig auf Verbesserungen in diesem Bereich.
Betrachten wir noch einmal Algorithmus 8.10, so ist es offensichtlich, dass die Berechnung von $D = D_H(\emptyset, J)$, vor allem in höheren Dimensionen, die meiste Zeit kostet. Daher untersuchen wir, wie oft wir diese Berechnung durchführen müssen. Angenommen, wir wollen eine Hüllkugel im $\mathbb{R}^d$ berechnen. Wir bezeichnen mit $s_j(N)$ die erwartete Anzahl der Berechnungen von $D = D_H(\emptyset, J)$, wobei wir N Punkte haben und $\delta - j$ davon sich

auf dem Rand befinden. Dabei ist $\delta = d + 1$. Dann gilt $s_0(N) = 1$ sowie $s_j(0) = 1$ und für alle $j > 0$ und $N > 0$ erhalten wir

$$s_j(N) \leq s_j(N-1) + \frac{j}{N} s_{j-1}(N-1). \tag{8.35}$$

8.14 Lemma: Sei $s_j(N)$ definiert wie in (8.35). Dann gilt

$$s_j(N) \leq (1 + H_N)^j \text{ mit } H_N = \sum_{k=1}^{N} \frac{1}{k}. \tag{8.36}$$

Beweis: Wir beweisen die Behauptung mit Induktion nach j und N. Für $j = 0$ und $N = 0$ gilt (8.36), wenn wir $H_0 = 0$ setzen. Für $j > 0$ und $N > 0$ erhalten wir mit Hilfe der Induktionsannahme

$$\begin{aligned} s_j(N) &\leq (1 + H_{N-1})^j + \frac{j}{N}(1 + H_{N-1})^{j-1} \\ &\leq \sum_{k=0}^{j} \binom{j}{k} (1 + H_{N-1})^{j-k} \left(\frac{1}{N}\right)^k \\ &= \left(1 + H_{N-1} + \frac{1}{N}\right)^j \\ &= (1 + H_N)^j. \end{aligned}$$

□

Da $H_N \leq 1 + \ln N$ für $N > 0$ gilt, ist $(2 + \ln N)^\delta$ eine obere Schranke für die erwartete Anzahl der Berechnungen von $D_H(\emptyset, J)$.
Wir wollen im weiteren Verlauf eine Veränderung von 8.10 diskutieren. Wir nehmen an, dass K eine, im Moment zufällig geordnete, Menge von Punkten ist. Wir verändern 8.10 wie folgt.

$\vdots$

`waehlen den letzten Punkt` $\mathbf{v} \in K$;
$D = D_H(K - \{\mathbf{v}\}, J)$;
`if`$(\mathbf{v} \notin K)$
  `waehle eine zufaellige Permutation` π `von` $(1, \ldots, |K| - 1)$
  $D = D_H(\pi(K - \{\mathbf{v}\}), J \cup \{\mathbf{v}\})$

$\vdots$

Die Operation $K \backslash \{\mathbf{v}\}$ entfernt $\mathbf{v}$ aus der Liste, lässt die übrigen Punkte und deren Reihenfolge aber unverändert. Daher können wir $D_H(K)$ die

Permutation π wählen lassen und erhalten für $D_H(K)$ den folgenden Algorithmus.

`function Kreis`$= D_H(K)$
  `waehle eine zufaellige Permutation` π `von` $(1, \ldots, |K|)$
  `return` $D_H(\pi(K), \emptyset)$;

Wir können zeigen, dass eine solche Permutation von K die erwartete Laufzeit nicht verändert. Es ist aber klar, dass durch eine Permutation, die eine Reihenfolge von K erzeugt, welche uns einen Hüllkreis möglichst früh liefert, der Algorithmus schneller mit einem Ergebnis beendet wird.
Da wir aber im Allgemeinen nicht in der Lage sind, eine solche Permutation für alle Konturen anzugeben, verzichten wir auf weitere Untersuchungen und wenden uns den Ergebnissen unserer Tests zu.
Die Testdurchläufe haben wir auf einem PC mit einer Intel Pentium CPU mit 2,4 GHz und 1024 MB Arbeitsspeicher unter Windows XP durchgeführt. Die Implementierung erfolgte unter Visual Studio C/C++ 6.0. Wir haben nur mit Konturen aus $\mathbb{R}^2$ und $\mathbb{R}^3$ getestet und den Algorithmus 8.8 mit den vorgestellten Modifikationen, also mit einer Laufzeit von $O(N)$, implementiert.

Kontur	$d =$	$\lfloor (e-1)(d+1)! \rfloor$	Anzahl Punkte	Zeit
gerd.asc	2	10	1713	0.047 sec
test1.asc	2	10	2448	0.052 sec
16xy.asc	3	41	2834	0.078 sec
d260_2.asc	2	10	7264	0.092 sec
test2.asc	3	41	9015	0.098 sec
schlange.asc	3	41	15181	0.110 sec
20x_1.asc	3	41	26014	0.157 sec

Die Laufzeiten der einzelnen Beispiele sind für eine zuverlässige Analyse des Laufzeitverhaltens zu klein, was sehr für die Effizienz des Verfahrens spricht. Tendenziell erkennen wir, dass sowohl für $d = 2$ als auch für $d = 3$ ein $O(N)$–Algorithmus vorliegt.

8.4 Pferchkreis

Das Thema dieses Abschnittes ist die Berechnung des maximalen eingeschriebenen Kreises einer Kontur. Ein solcher Kreis wird in der Koordinatenmesstechnik als *Pferchkreis* bezeichnet. Ein Pferchkreis kommt in der

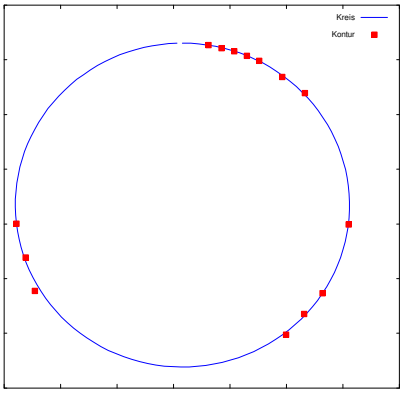

Abbildung 34: Pferchkreis einer Kontur. Alle Punkte der Kontur liegen außerhalb oder am Rand des Kreises.

Regel beim Vermessen von Bohrungen oder beim Abtasten eines Kreises mit wenigen Punkten zum Einsatz.
Während der Hüllkreis auch in höheren Dimensionen als Hüllkugel sinnvoll ist, findet eine Pferchkugel in der Messtechnik keine Verwendung. Daher beschränken wir uns in diesem Abschnitt ausschließlich auf den Pferchkreis in der Ebene.

8.15 Definition: Sei $K = \{\mathbf{v}_j \mid j = 0, \ldots, N-1\} \subset \mathbb{R}^2$. Unter einem *Pferchkreis* der Kontur K verstehen wir einen maximalen eingeschriebenen Kreis $D_P(K) = (\mathbf{m}, r)$, so dass alle Punkte der Kontur auf dem Rand oder außerhalb des Kreises liegen. Das heißt es gilt

$$D_P(K) = (\mathbf{m}, r) = \max_{r_1>0}\{E = (\mathbf{a}, r_1) \mid \|\mathbf{v}_j - \mathbf{a}\|_2 \geq r_1 \text{ für } j = 0, \ldots, N-1\}.$$

Ein Beispiel für einen Pferchkreis sehen wir in Abbildung 34.

Die üblichen Methoden zur Berechnung des Pferchkreises versuchen, das quadratische Ausgangsproblem zu linearisieren. Dadurch erreicht man zwar einerseits eine effiziente und schnelle Berechnung, andererseits brechen diese Algorithmen aber oft ohne Ergebnis ab, zum Beispiel wenn der Mittelpunkt $\mathbf{m}$ nicht in der Nähe des Schwerpunktes der Kontur liegt. Wir wollen hier eine alternative Methode vorstellen, die sich eine Eigenschaft der *Voronoi-Diagramme* zu nutze macht. Bevor wir diese Eigenschaft in Form eines Satzes formulieren, definieren wir *Voronoi-Diagramme* und geben einige ihrer Eigenschaften an.
Wir orientieren uns bei der Einführung an dem Buch [dBe00], Kapitel 7.
Wir nehmen im Folgenden stets an, dass sich die Punkte der Kontur

$K = \{\mathbf{v}_j \mid j = 0, \ldots, N-1\} \subset \mathbb{R}^2$ in allgemeiner Lage befinden, das heißt keine drei Punkte sind kollinear und keine vier Punkte bilden einen Kreis. Wir können diese Annahmen natürlich nicht für die Praxis aufrecht erhalten, werden aber später angeben, an welchen Stellen sich die Laufzeit des Algorithmus verändert.

8.16 Definition: Sei $K = \{\mathbf{v}_j \mid j = 0, \ldots, N-1\} \subset \mathbb{R}^2$ eine Kontur.

(1) Für zwei Punkte $\mathbf{v}, \mathbf{u} \in \mathbb{R}^2$ definieren wir

$$b(\mathbf{v}, \mathbf{u}) = \{\mathbf{x} \in \mathbb{R}^2 \mid \|\mathbf{x} - \mathbf{v}\|_2 = \|\mathbf{x} - \mathbf{u}\|_2\},$$

den *Bisektor* von $\mathbf{v}$ und $\mathbf{u}$.

(2) Wir setzen

$$\mathcal{VR}(\mathbf{v}_j) = \{\mathbf{x} \in \mathbb{R}^2 \mid \|\mathbf{v}_j - \mathbf{x}\|_2 \leq \|\mathbf{v}_k - \mathbf{x}\|_2 \text{ für alle } k = 0, \ldots, N-1\},$$

die *Voronoi-Region* von $\mathbf{v}_j \in K$.

(3) Den Rand einer Voronoi-Region $\mathcal{VR}(\mathbf{v}_j)$ bildet das *Voronoi-Polygon* $\mathcal{VP}(\mathbf{v}_j)$, definiert als

$$\begin{aligned}\mathcal{VP}(\mathbf{v}_j) = \{\mathbf{x} \in \mathbb{R}^2 \mid \|\mathbf{v}_j - \mathbf{x}\|_2 = \|\mathbf{v}_k - \mathbf{x}\|_2,\\ \text{für ein } k \in \{0, \ldots, N-1\}\backslash\{j\}\}.\end{aligned}$$

Wir nennen die einzelnen Segmente eines Voronoi-Polygons *Voronoi-Kanten*. Die Schnittpunkte der Voronoi-Kanten wollen wir als *Voronoi-Ecken* bezeichnen.

(4) Für eine Kontur $K \subset \mathbb{R}^2$ ist ein *Voronoi-Diagramm* $\mathcal{VD}(K)$ definiert als

$$\mathcal{VD}(K) = \bigcup_{j=0}^{N-1} \mathcal{VP}(\mathbf{v}_j). \tag{8.37}$$

Wir können leicht sehen, dass eine Voronoi-Kante stets eine Teilmenge des Bisektors zweier Punkte der Kontur K ist. Daher sind alle Punkte einer Kante, die keine Voronoi-Ecken sind, zu genau zwei Punkten aus K äquidistant.
Nun geben wir einige Eigenschaften von Voronoi-Diagrammen und deren Teilobjekten an. Wir finden die meisten Punkte des Lemmas in [dBe00].

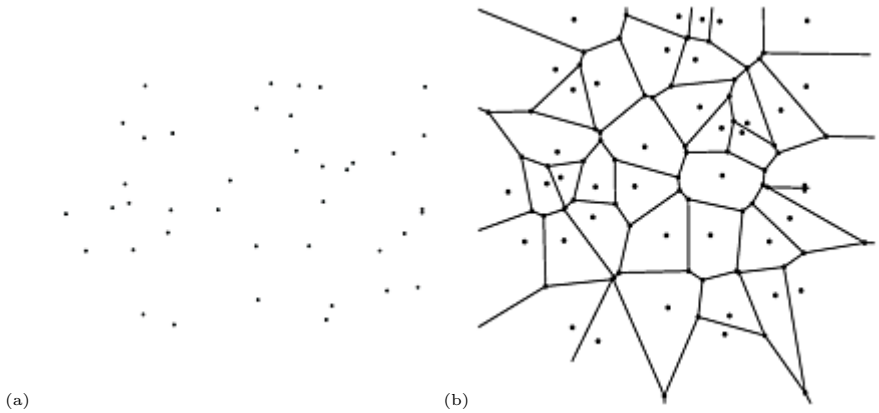

Abbildung 35: (a) Kontur K (b) Voronoi-Diagramm der Kontur K

8.17 Lemma: Sei $K = \{\mathbf{v}_j \mid j = 0, \ldots, N-1\} \subset \mathbb{R}^2$ eine Kontur.

(1) Für $\mathbf{v}, \mathbf{u} \in \mathbb{R}^2$ ist $H(\mathbf{v}, \mathbf{u})$ die Halbebene, die durch den Bisektor $b(\mathbf{v}, \mathbf{u})$ begrenzt wird und den Punkt $\mathbf{v}$ enthält. Dann gilt

$$\mathcal{VR}(\mathbf{v}_j) = \bigcap_{k=0, k\neq j}^{N-1} H(\mathbf{v}_j, \mathbf{v}_k).$$

(2) Sind alle Punkte $\mathbf{v}_j$ von K kollinear, dann besteht $\mathcal{VD}(K)$ aus $N-1$ parallelen Geraden. Anderenfalls ist $\mathcal{VD}(K)$ zusammenhängend und die Voronoi-Kanten sind entweder Strecken oder Halbgeraden.

(3) Für $N \geq 3$ beträgt die Anzahl der Voronoi-Ecken einer Kontur K mit N Punkten höchstens $2N-5$. Die Anzahl der Kanten ist nach oben durch $3N-6$ begrenzt. Da jede Voronoi-Kante zu genau zwei Voronoi-Polygonen gehört, besitzt ein Voronoi-Polygon durchschnittlich sechs Kanten.

(4) Ein Punkt $\mathbf{v} \in \mathbb{R}^2$ ist genau dann eine Voronoi-Ecke, wenn der größte Kreis $C = (\mathbf{v}, r)$, in dessen Innerem kein Konturpunkt liegt, drei oder mehr Punkte aus K auf dem Rand enthält.

(5) Der Bisektor $b(\mathbf{v}_j, \mathbf{v}_k)$ ist genau dann eine Kante, wenn ein Punkt $\mathbf{v}_n$ auf $b(\mathbf{v}_j, \mathbf{v}_k)$ existiert, so dass der Kreis $C = (\mathbf{v}_n, r_n)$ sowohl $\mathbf{v}_j$ als auch $\mathbf{v}_k$ aber keinen anderen Punkt auf dem Rand enthält.

Beweis:

(1) Da $H(\mathbf{v}_j, \mathbf{v}_k) = \{\mathbf{x} \in \mathbb{R}^2 \mid \|\mathbf{v}_j - \mathbf{x}\|_2 \leq \|\mathbf{v}_k - \mathbf{x}\|_2\}$ ist, folgt die Behauptung unmittelbar aus der Definition.

(2) Der erste Teil der Behauptung ist offensichtlich. Wir nehmen daher an, dass nicht alle Punkte von K kollinear sind.
Wir zeigen als Erstes, dass die Voronoi-Kanten von $\mathcal{VD}(K)$ entweder Strecken oder Halbgeraden sind. Wir wissen aus Definition 8.16, dass die Voronoi-Kanten Teilstücke von Geraden, genauer gesagt von Bisektoren zwischen den Punkten, sind. Wir nehmen an, es existiert eine Kante $\mathbf{g}$ des Voronoi-Diagramms $\mathcal{VD}(K)$, die eine vollständige Gerade ist. Sei $\mathbf{g}$ auf dem Rand der Voronoi-Regionen $\mathcal{VR}(\mathbf{v}_j)$ und $\mathcal{VR}(\mathbf{v}_i)$. Sei weiter $\mathbf{v}_l$ ein Punkt, der nicht kollinear zu $\mathbf{v}_j$ und $\mathbf{v}_i$ ist. Dann ist $b(\mathbf{v}_j, \mathbf{v}_l)$ nicht parallel zu $\mathbf{g}$, schneidet also die Gerade $\mathbf{g}$. Dann kann aber der Teil von $\mathbf{g}$, der im Inneren von $H(\mathbf{v}_l, \mathbf{v}_j)$ liegt, nicht den Rand von $\mathcal{VR}(\mathbf{v}_j)$ bilden, weil dieser Teil näher an $\mathbf{v}_l$ liegt als $\mathbf{v}_j$, Widerspruch.
Es bleibt zu zeigen, dass $\mathcal{VD}(K)$ zusammenhängend ist. Wäre dies nicht der Fall, würde eine Voronoi-Region $\mathcal{VR}(\mathbf{v}_j)$ existieren, welche die ganze Ebene teilt. Da $\mathcal{VR}(\mathbf{v}_j)$ nach (1) konvex ist, müsste $\mathcal{VR}(\mathbf{v}_j)$ eine Teilregion enthalten, die durch zwei Geraden begrenzt wäre. Wir haben aber eben gezeigt, dass $\mathcal{VD}(K)$ keine vollständigen Geraden enthalten kann, Widerspruch.

(3) Sind alle Punkte kollinear, so folgt die Behauptung sofort aus (2). Wir nehmen an, dass nicht alle Punkte aus K kollinear sind.
Wir beweisen die Behauptung mit Hilfe der Eulerschen Polyederformel. Diese besagt, dass in einem zusammenhängenden, planaren Graphen mit
a_e = Anzahl der Ecken,
a_k = Anzahl der Kanten und
a_f = Anzahl der Flächen
die Beziehung gilt

$$a_e - a_k + a_f = 2.$$

Wir können die Eulersche Polyederformel nicht sofort auf $\mathcal{VD}(K)$ anwenden, da $\mathcal{VD}(K)$ Halbgeraden enthält, also kein echter Graph ist. Um dieses Problem zu beheben, definieren wir eine zusätzliche Ecke, die so genannte *unendliche Ecke* $\mathbf{v}_\infty$ und verbinden alle Halbgeraden an

dieser Ecke. Jetzt ist $\mathcal{VD}(K)$ ein zusammenhängender, planarer Graph und wir erhalten die Gleichung

$$(a_e + 1) - a_k + N = 2. \tag{8.38}$$

Dabei ist
a_e = Anzahl der Ecken von $\mathcal{VD}(K)$ ohne $\mathbf{v}_\infty$,
a_k = Anzahl der Kanten von $\mathcal{VD}(K)$ und
N = Anzahl der Punkte von K, wobei diese im graphentheoretischen Sinne die Anzahl der Flächen ist. Denn zu jedem Punkt $\mathbf{v} \in K$ gehört eine Voronoi-Region, also eine Fläche des Graphen.
Aus der Definition von $\mathcal{VD}(K)$ geht hervor, dass jede Kante zu genau zwei Ecken gehört. Addieren wir die Ordnungen der Ecken, so erhalten wir die zweifache Anzahl der Kanten. Da jede Ecke, $\mathbf{v}_\infty$ mitgerechnet, mindestens die Ordnung drei hat, gilt die Ungleichung

$$2a_k \geq 3(a_e + 1).$$

Mit Hilfe von Gleichung (8.38) erhalten wir beide zu zeigenden Aussagen.

(4) Wir nehmen an, es existiert ein Punkt $\mathbf{v} \in \mathbb{R}^2$, so dass der Kreis $C = (\mathbf{v}, r)$ drei oder mehr Punkte auf dem Rand enthält. Seien $\mathbf{v}_j, \mathbf{v}_i, \mathbf{v}_k \in K$ drei solche Punkte. Da das Innere von C keine Punkte aus K enthält, muss $\mathbf{v}$ auf dem Rand der Voronoi-Regionen $\mathcal{VR}(\mathbf{v}_j), \mathcal{VR}(\mathbf{v}_i)$ und $\mathcal{VR}(\mathbf{v}_k)$ liegen. Damit ist $\mathbf{v}$ eine Ecke von $\mathcal{VD}(K)$.
Umgekehrt ist jede Ecke $\mathbf{v}$ von $\mathcal{VD}(K)$ inzident zu mindestens drei Kanten und damit zu mindestens drei Voronoi-Regionen $\mathcal{VR}(\mathbf{v}_j), \mathcal{VR}(\mathbf{v}_i)$ und $\mathcal{VR}(\mathbf{v}_k)$. Die Ecke $\mathbf{v}$ ist nach Definition 8.16 äquidistant zu den Punkten $\mathbf{v}_j, \mathbf{v}_i, \mathbf{v}_k$ und es existieren keine weiteren Punkte aus K, die näher an $\mathbf{v}$ liegen als diese drei, sonst könnten die Voronoi-Regionen nicht benachbart sein. Daher enthält der Kreis mit $\mathbf{v}_j, \mathbf{v}_i, \mathbf{v}_k$ auf dem Rand keine Punkte aus K im Inneren.

(5) Wir nehmen an, es existiert ein Punkt $\mathbf{v}$, so dass $\mathbf{v} \in b(\mathbf{v}_j, \mathbf{v}_k)$ ist und der Kreis $C = (\mathbf{v}, r)$ enthält $\mathbf{v}_j$ und $\mathbf{v}_k$ auf dem Rand und sonst keinen anderen Punkt von K. Dann gilt

$$\|\mathbf{v}_j - \mathbf{v}\|_2 = \|\mathbf{v}_k - \mathbf{v}\|_2 \leq \|\mathbf{v}_n - \mathbf{v}\|_2 \text{ für alle } n = 0, \ldots, N-1.$$

$\mathbf{v}$ muss entweder eine Ecke von $\mathcal{VD}(K)$ sein oder auf einer Kante von $\mathcal{VD}(K)$ liegen. Da aber nur zwei Punkte auf dem Rand von C liegen,

folgt aus (4), dass $\mathbf{v}$ keine Ecke sein kann. Damit liegt $\mathbf{v}$ auf einer Kante, die durch $b(\mathbf{v}_j, \mathbf{v}_k)$ definiert wird.
Sei umgekehrt $b(\mathbf{v}_j, \mathbf{v}_k)$ ein Bisektor, der eine Kante von $\mathcal{VD}(K)$ definiert. Der größte Kreis um einen auf der Kante liegenden Punkt $\mathbf{v}$, in dessen Innerem kein Konturpunkt liegt, muss nach der Definition eines Bisektors die Punkte $\mathbf{v}_j$ und $\mathbf{v}_k$ und sonst keine anderen Punkte auf dem Rand enthalten.

□

Als nächstes wollen wir den Bogen zwischen den eben definierten Voronoi-Diagrammen und dem Gegenstand des Abschnittes, dem Pferchkreis, spannen. Es gilt die folgende Beziehung.

8.18 Satz: Sei $K = \{\mathbf{v}_j \mid j = 0, \ldots, N-1\} \subset \mathbb{R}^2$. Dann liegt der Mittelpunkt des Pferchkreises $D_P(K)$ auf einer inneren Ecke des Voronoi-Diagramms $\mathcal{VD}(K)$.

Beweis: Wir nehmen an, der Mittelpunkt des Pferchkreises $D_P(K)$ sei keine innere Ecke des Voronoi-Diagramms $\mathcal{VD}(K)$. Nun wird $D_P(K)$ durch mindestens drei Konturpunkte definiert. Dann ist aber nach Lemma 8.17 (4) der Mittelpunkt des Kreises eine innere Voronoi-Ecke, Widerspruch. Also ist der Mittelpunkt des Pferchkreises eine innere Voronoi-Ecke. □

Demnach berechnen wir zunächst das Voronoi-Diagramm $\mathcal{VD}(K)$ der Kontur K, bestimmen alle inneren Ecken und wählen die richtige Ecke aus.
Ein Nebenprodukt, welches bei der Bestimmung des Mittelpunktes des Pferchkreises berechnet werden könnte, ist die so genannte *Medial Axis* von K, also die Spiegelachse der Kontur.
Wir diskutieren zunächst die Berechnung von $\mathcal{VD}(K)$. Der Bestimmung der richtigen Ecke wenden wir uns später zu.
Intuitiv können wir zur Berechnung eines Voronoi-Diagramms $\mathcal{VD}(K)$ Lemma 8.17 (1) benutzen und bestimmen zu jedem Punkt $\mathbf{v}_j$ die Schnittmenge aller Halbebenen $H(\mathbf{v}_j, \mathbf{v}_k)$ für $k = 0, \ldots, N-1$ mit $k \neq j$. Wenden wir ein Divide & Conquer-Verfahren zur Berechnung der Voronoi-Region $\mathcal{VR}(\mathbf{v}_j)$ an, so ist eine Komplexität von $O(N \log N)$ möglich. Insgesamt erhalten wir bei einer Kontur K mit N Punkten eine Berechnung von $\mathcal{VD}(K)$ in $O(N^2 \log N)$. Eine mehr als quadratische Komplexität ist bei größeren Konturen natürlich etwas zu langsam. Also stellt sich die Frage, ob schnellere Möglichkeit zu Berechnung von $\mathcal{VD}(K)$ existieren. Diese Frage lässt sich

bejahen. Es gibt sogar mehrere solcher Möglichkeiten, wir konzentrieren uns im Folgenden auf den *Sweepline-* oder *Fortunes-Algorithmus.*

8.4.1 Sweepline-Algorithmus

Die Idee des Algorithmus, wahlweise genannt nach seinem Erfinder *Steve Fortune* oder nach der in dem Algorithmus vorkommenden Gerade (*Sweepline*), ist einfach zu beschreiben. Wir bewegen eine horizontale Gerade ℓ_s von oben nach unten über die Ebene. Während der Bewegung erhalten wir Informationen über die Schnittpunkte der Sweepline ℓ_s mit der Kontur. Diese Informationen verarbeiten wir und nutzen sie zur Bildung des Voronoi-Diagrammes. Wir formalisieren die Idee bis hin zum Gesamtalgorithmus.
Sei $K = \{\mathbf{v}_j \mid j = 0, \ldots, N-1\} \subset \mathbb{R}^2$ die zu behandelnde Kontur. Sei $\ell_s : y = t \in \mathbb{R}$ eine zur x–Achse parallele Gerade, die wir als Sweepline bezeichnen. Wir bewegen die Sweepline in der Ebene von oben nach unten, also betrachten wir eine Geradenschar $\ell_s^{(t)} : y = t$ mit $t \in [y_{min}, y_{max}]$, wobei gilt

$$y_{min} < \min_{0 \le j < N} v_j^{(2)} \text{ und } y_{max} > \max_{0 \le j < N} v_j^{(2)}.$$

Sei ℓ^+ die abgeschlossene Halbebene oberhalb von ℓ_s, also

$$\ell^+ = \{\mathbf{x} \in \mathbb{R}^2 \mid x^{(2)} \ge t\}$$

und die offene Halbebene unterhalb von ℓ_s

$$\ell^- = \{\mathbf{x} \in \mathbb{R}^2 \mid x^{(2)} < t\}.$$

Seien $\mathbf{v}_k \in \ell^+$ und $\mathbf{v}_j \in K$, dann gilt

$$\|\mathbf{v}_k - \mathbf{v}_j\|_2 > \|\mathbf{v}_k - \mathbf{b}_s\|_2 \text{ für } \mathbf{v}_j \in \ell^- \text{ und } \mathbf{b}_s \in \ell_s.$$

Dabei ist $\mathbf{b}_s \in \ell_s$ der Punkt auf der Sweepline mit dem kleinsten Abstand zu $\mathbf{v}_k$, also der Fußpunkt des Lotes von $\mathbf{v}_k$ auf ℓ_s. Existieren $\mathbf{b} \in \ell_s$ und $\mathbf{v}_j \in K$, so dass $\|\mathbf{v}_k - \mathbf{b}\|_2 = \|\mathbf{v}_k - \mathbf{v}_j\|_2$, so kann der Konturpunkt, der am nächsten zu $\mathbf{v}_k$ liegt, nicht in der Halbebene ℓ^- liegen. Für ein $\mathbf{v}_j \in K$ wird dann die Menge

$$M_j = \{\mathbf{x} \in \mathbb{R}^2 \mid \|\mathbf{v}_j - \mathbf{x}\|_2 \le \|\mathbf{b} - \mathbf{x}\|_2 \text{ für alle } \mathbf{b} \in \ell_s\}$$

durch eine Parabel begrenzt. Also definiert jedes $\mathbf{v}_j \in K$ eine Parabel β_j. Wir bezeichnen die Funktion, die für jeden x–Wert den minimalen y–Wert

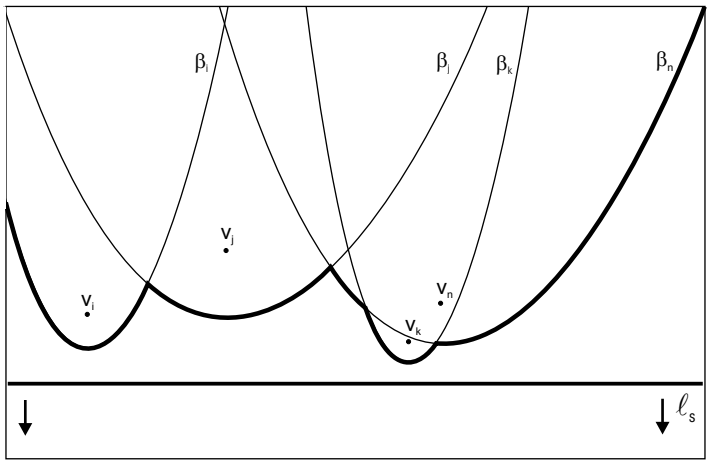

Abbildung 36: Beispiel für eine Beachline.

einer Parabel β_j annimmt, als *Beachline* und machen die nachfolgend beschriebene Beobachtung.
Ein Beispiel für einen möglichen Verlauf der Beachline sehen wir in Abbildung 36.

8.19 Korollar: Die Beachline ist x–*monoton*, das heißt jede Gerade $x = c \in \mathbb{R}$ schneidet die Beachline in genau einem Punkt.

Es ist offensichtlich, dass ein Parabelsegment mehrmals zu einer Beachline gehören kann. Über die Anzahl, wie oft ein Segment Teil der Beachline sein kann, wollen wir uns später Gedanken machen. Vorher beschäftigen wir uns mit den Schnittpunkten zweier Parabelsegmente.

8.20 Lemma: Sei $\mathbf{v}_j \in K$ und sei $\mathbf{v}_k \in K$ der zu $\mathbf{v}_j$ nächste Punkt sowie β_j, β_k die zugehörigen Parabeln. Dann gilt $\beta_j \cap \beta_k \neq \emptyset$ und der Schnittpunkt $\mathbf{a} \in \beta_j \cap \beta_k$ liegt auf der Voronoi-Kante zwischen $\mathbf{v}_j$ und $\mathbf{v}_k$.

Beweis: Es gilt $\beta_j \cap \beta_k \neq \emptyset$, da $\mathbf{v}_k$ der zu $\mathbf{v}_j$ nächste Punkt ist und demzufolge beide Parabeln benachbart und nach Konstruktion nach oben geöffnet sind. Daher existiert mindestens ein Schnittpunkt von β_j und β_k. Sei $\mathbf{a} \in \beta_j \cap \beta_k$. Dann gilt nach Definition von M_j, dass ein $\mathbf{b}_1 \in \ell_s$ existiert mit

$$\|\mathbf{v}_j - \mathbf{a}\|_2 = \|\mathbf{v}_j - \mathbf{b}_1\|_2,$$

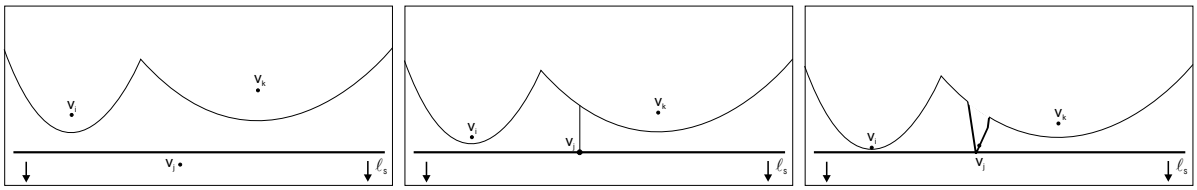

Abbildung 37: Die Entstehung eines Punkt-Events

da $\mathbf{a}$ am Rand von β_j liegt. Da $\mathbf{a}$ ebenfalls am Rand von β_k liegt, erhalten wir

$$\|\mathbf{v}_k - \mathbf{a}\|_2 = \|\mathbf{v}_j - \mathbf{b}_2\|_2$$

für ein $\mathbf{b}_2 \in \ell_s$ und damit $\|\mathbf{v}_j - \mathbf{a}\|_2 = \|\mathbf{v}_k - \mathbf{a}\|_2$. Dann liegt $\mathbf{a}$ auf dem Bisektor $b(\mathbf{v}_j, \mathbf{v}_k)$ und da $\mathbf{v}_k$ der zu $\mathbf{v}_j$ nächste Punkt war, befindet sich $\mathbf{a}$ gleichzeitig auf der Voronoi-Kante zwischen $\mathbf{v}_j$ und $\mathbf{v}_k$. □

Damit bilden die Schnittpunkte der Parabeln, die auf der Beachline liegen, gerade die Kanten des Voronoi-Diagramms. Wir beobachten nachfolgend nicht die Schnittpunkte der Sweepline ℓ_s mit dem Voronoi-Diagramm, sondern konzentrieren uns auf den Verlauf der Beachline. Dabei gibt es zwei Ereignisse, die für uns von Interesse sind. Zum einen das Entstehen eines neuen Parabelsegmentes und zum anderen das Bilden eines einzelnen Punktes aus einem Segment. Fangen wir mit dem Hinzufügen eines neuen Parabelsegmentes an.

(1) Ein neues Segment entsteht, wenn die Sweepline ℓ_s einen Punkt $\mathbf{v}_j \in K$ erreicht. Im ersten Augenblick ist dann $\mathbf{v}_j \in \ell_s$ und wir haben es mit einer degenerierten Parabel der Breite Null zu tun. Bewegt sich ℓ_s weiter in negativer y–Richtung, so haben wir stets $\mathbf{v}_j \in \ell^+$ und die Parabel wird immer breiter. Ein Teil einer solchen Parabel wird stets auch zu einem Teilsegment der Beachline. Wir wollen dieses Ereignis als *Punkt-Event* bezeichnen. Ein Beispiel für die Entstehung eines Punkt-Events sehen wir in Abbildung 37. Was passiert aber mit dem Voronoi-Diagramm an einem solchen Punkt-Event? Da die Schnittpunkte der Segmente gerade die Voronoi-Kanten bilden, erzeugen wir an einem Punkt-Event eine neue Kante. Allerdings ist diese Kante zunächst nicht mit dem restlichen Diagramm verbunden. Die Überlegung, wie eine solche lose Kante an das restliche Diagramm angebunden wird, stellen wir zurück und wollen uns mit der Frage beschäftigen, ob ein Punkt-Event die einzige Möglichkeit darstellt, auf die ein Segment zur Beachline hinzugefügt werden kann.

8.21 Lemma: Die Beachline kann ausschließlich durch ein Punkt-Event um ein Segment erweitert werden.

Beweis: Wir nehmen an, dass eine bereits existierende Parabel β_j, definiert durch einen Punkt $\mathbf{v}_j \in K$, die Beachline schneidet. Es gibt dann zwei Möglichkeiten, auf die eine solche Situation entstehen kann.

(a) Die Parabel β_j schneidet die Beachline in der Mitte eines Segmentes, das zu der Parabel β_k gehört. Also liegen die Scheitelpunkte von β_j und β_k auf einer Geraden g, die senkrecht auf die Sweepline ℓ_s steht. In der Bewegung der Sweepline existiert genau ein Parameter $t_* \in \mathbb{R}$, so dass sich β_j und β_k in genau einem Punkt berühren. Sei $\mathbf{v}_j = [v_j^{(1)},\ v_j^{(2)}]^t$, dann gilt

$$\beta_j = \beta_j(x) = \frac{1}{2(v_j^{(2)} - t_*)} \left[x^2 - 2v_j^{(1)}x + \left(v_j^{(1)}\right)^2 + \left(v_j^{(2)}\right)^2 - t_*^2 \right].$$

Für β_k und den zugehörigen Konturpunkt $\mathbf{v}_k$ erhalten wir die entsprechende Gleichung. Nutzen wir die Tatsache aus, dass $v_j^{(2)} > t_*$ und $v_k^{(2)} > t_*$ gilt, so können wir zeigen, dass die Parabeln β_j und β_j nie genau einen Schnittpunkt haben können. Wir erhalten einen Widerspruch. Also kann die Parabel β_j die Beachline nicht in der Mitte eines Segmentes schneiden.

(b) Die zweite Möglichkeit ist, dass die Parabel β_j die Beachline genau zwischen zwei Segmenten schneidet. Wir nehmen an, diese Segmente gehören zu den Parabeln β_i und β_k. Sei $\mathbf{v} \in \beta_i \cap \beta_k$ der Punkt, an dem β_j die Beachline schneidet. Liege β_i links von $\mathbf{v}$ und β_k rechts von $\mathbf{v}$. Seien $\mathbf{v}_i, \mathbf{v}_k \in K$ die zu den Parabeln β_i und β_k gehörigen Punkte. Es existiert ein Kreis C, der die Punkte $\mathbf{v}_i, \mathbf{v}_j, \mathbf{v}_k$ auf dem Rand und $\mathbf{v}$ als Mittelpunkt hat. Außerdem berührt C wegen $\|\mathbf{v}_j - \mathbf{v}\|_2 = \|\mathbf{v} - \ell_s\|_2$ die Sweepline ℓ_s. Bewegt sich die Sweepline weiter in y–Richtung, so berührt C weiterhin ℓ_s, vgl. Abbildung 38. Der Kreis C kann nicht größer werden und gleichzeitig keinen Konturpunkt im Inneren enthalten, denn dann würden entweder $\mathbf{v}_i$ oder $\mathbf{v}_k$ innerhalb des Kreises liegen. Also kann in einer hinreichend kleinen Umgebung um $\mathbf{v}$ die Beachline nicht um einen Segment erweitert werden, Widerspruch.

□

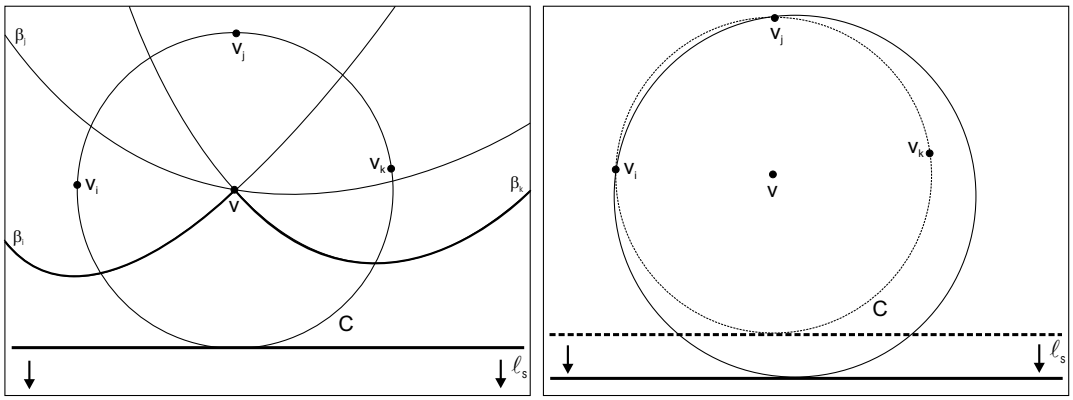

Abbildung 38: Die Parabel β_j soll zwischen zwei weiteren Segmenten der Beachline hinzugefügt werden.

Die Konsequenz aus Lemma 8.21 ist, dass eine Beachline aus höchstens $2N - 1$ Teilsegmenten bestehen kann. Jeder Punkt erzeugt eine Parabel, die zu einem Segment der Beachline werden kann und ein solches Segment kann in höchstens zwei Teilsegmente gesplittet werden.

(2) Das zweite Ereignis, das auftreten kann, ist, dass ein existierendes Segment der Beachline immer kürzer wird, bis es die Länge Null hat, also zu einem Punkt wird. Sei α_j das verschwindende Segment und seien α_i und α_k zwei zu diesem benachbarte Segmente. Aus dem Beweis von 8.21 geht hervor, dass α_i und α_k nicht zu derselben Parabel gehören können, also existieren Punkte $\mathbf{v}_i, \mathbf{v}_j, \mathbf{v}_k \in K$, welche die zugehörigen Parabeln erzeugen. In dem Moment, in dem α_j die Länge Null erreicht, wird ein Punkt $\mathbf{v}$ als Schnittpunkt von α_i und α_k gebildet. Dabei gilt

$$\|\mathbf{v} - \mathbf{v}_j\|_2 = \|\mathbf{v} - \mathbf{v}_i\|_2 = \|\mathbf{v} - \mathbf{v}_k\|_2 = \|\mathbf{v} - \mathbf{b}\|_2,$$

wobei $\mathbf{b} \in \ell_s$ mit $\mathbf{b} = \min_{\mathbf{a}\in\ell_s} \|\mathbf{v} - \mathbf{a}\|_2$ ist. Also existiert ein Kreis

$$C = (\mathbf{v}, r) \text{ mit } r = \|\mathbf{v} - \mathbf{v}_j\|_2 \text{ und } \mathbf{v}_i, \mathbf{v}_j, \mathbf{v}_k, \mathbf{b} \in C.$$

Im Inneren von C liegt kein Punkt der Kontur, sonst wäre der Abstand dieses Punktes zu $\mathbf{v}$ geringer als der Abstand zwischen $\mathbf{v}$ und ℓ_s. Da aber $\mathbf{v}$ auf der Beachline liegt, wäre dies ein Widerspruch. Damit ist $\mathbf{v}$ eine Ecke des Voronoi-Diagramms. Ein solches Ereignis bezeichnen wir als *Kreis-Event* und können eine Folgerung formulieren.

8.22 Korollar: Ein Kreis-Event ist die einzige Möglichkeit, durch die ein Segment aus der Beachline verschwinden kann.

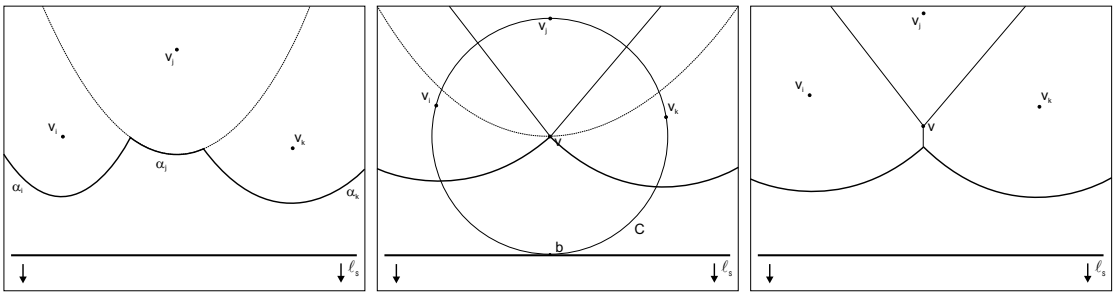

Abbildung 39: Die Entstehung eines Kreis-Events

Alle drei Phasen der Entstehung eines Kreis-Events sind in Abbildung 39 abgebildet.

Also wissen wir jetzt, auf welche Weise das Voronoi-Diagramm mit Hilfe der Sweepline beziehungsweise Beachline gebildet wird. Bei einem Kreis-Event entsteht aus zwei Kanten eine neue Ecke, bei einem Punkt-Event fängt eine Kante an zu wachsen. Insbesondere werden diese Kanten an das Voronoi-Diagramm angebunden.
Um den Sweepline-Algorithmus effizient implementieren zu können, benötigen wir entsprechende Datenstrukturen. Zunächst müssen wir das Voronoi-Diagramm sowohl während der Konstruktion als auch später als fertige Struktur speichern. Zusätzlich brauchen wir zwei weitere Datenstrukturen, eine Ereignis- und eine Statusliste. Diese Strukturen setzen wir wie folgt um.

(1) Wir speichern das Voronoi-Diagramm in einer zweifach verlinkten Liste. Allerdings besteht ein Voronoi-Diagramm auch aus Halbgeraden, die nicht in einer solchen Liste gespeichert werden können. Für die eigentliche Konstruktion stellt dies kein Problem dar, für die endgültige Diagramm-Darstellung bestimmen wir einen, das Diagramm umfassenden, Quader und berechnen die Liste neu.

(2) Die Beachline stellen wir als einen binären Suchbaum $\mathcal{T}$ dar. Ein Blatt des Baumes repräsentiert ein Segment der Beachline. Das Blatt ganz links gehört dann zu dem Segment am weitesten links, wobei jeder Knoten den zu dem Segment α_j zugehörigen Punkt $\mathbf{v}_j$ speichert. Ein Schnittpunkt wird durch einem Knoten dargestellt, wobei das zugehörige Punktepaar $\langle \mathbf{v}_j, \mathbf{v}_k \rangle$ gespeichert wird. Dabei gehört $\mathbf{v}_j$ zu dem linken und $\mathbf{v}_k$ zu dem rechten Segment. In einem auf solche Weise dargestellten binären Baum sind wir in der Lage, das richtige Segment in $O(\log N)$ zu finden.

(3) Die Event- oder Statusliste Q stellen wir als priorisierte Statusliste dar, wobei die Priorität eines Events durch die y–Koordinate repräsentiert wird. Für ein Punkt-Event speichern wir den Punkt selbst, während für ein Kreis-Event der Punkt des Kreises mit der kleinsten y–Koordinate gespeichert wird sowie ein Pointer zu dem Blatt von $\mathcal{T}$, welches das Segment repräsentiert, das durch dieses Event aus der Beachline verschwindet.

Während die Punkt-Events alle bekannt sind, ist das bei den Kreis-Events leider nicht der Fall. Also müssen wir uns, bevor wir den Algorithmus formulieren können, mit der Erkennung eines Kreis-Events beschäftigen.
Während der Bewegung verändert die Beachline ständig ihre Struktur. Einerseits werden neue Tripel von benachbarten Segmenten gebildet und der Beachline hinzugefügt, andererseits verschwinden manche Segmente aus der Beachline.
Wir speichern für jedes Tripel solcher benachbarter Segmente, die ein Kreis-Event definieren könnten, einen Eintrag in der Statusliste Q. Es können zwei Ereignisse eintreten, die nicht zu einem Kreis-Event führen. Zum einen ist es möglich, dass die beiden Schnittpunkte, die zu dem Tripel benachbarter Segmente gehören, nicht zu einem Punkt konvergieren. Damit definieren die Segmente kein Kreis-Event und wir löschen den entsprechenden Eintrag aus Q. Das zweite mögliche Ereignis tritt ein, wenn die beiden Schnittpunkte zwar gegeneinander konvergieren, aber kein Kreis entstehen kann, weil beispielsweise eine neue Kante an der Beachline entsteht. In diesem Fall erhält der zugehörige Eintrag in Q den Status *falscher Alarm*.
Wir wisssen jetzt, wie wir Kreis-Events erkennen können, das nachfolgende Lemma erläutert uns die Funktion solcher Events.

8.23 Lemma: Jede Voronoi-Ecke wird durch ein Kreis-Event gebildet.

Beweis: Sei $\mathbf{v}$ eine Voronoi-Ecke. Seien $\mathbf{v}_i, \mathbf{v}_j, \mathbf{v}_k \in K$ Konturpunkte, die einen Kreis $C = (\mathbf{v}, r)$ mit $r = \|\mathbf{v} - \mathbf{v}_i\|_2$ definieren. Dabei liegen die Punkte $\mathbf{v}_j, \mathbf{v}_k, \mathbf{v}_i$ auf dem Rand von C und im Inneren liegen keine Punkte aus K. Ein solcher Kreis existiert nach Lemma 8.9 (4). Wir nehmen ohne Beschränkung der Allgemeinheit an, dass nur die Punkte $\mathbf{v}_i, \mathbf{v}_j, \mathbf{v}_k$ auf dem Rand von C liegen und der Punkt $\mathbf{u} \in C$ mit der kleinsten y–Koordinate aller Punkte auf C nicht $\mathbf{v}_i, \mathbf{v}_j$ oder $\mathbf{v}_k$ ist. Sei $\mathbf{v}_i, \mathbf{v}_j, \mathbf{v}_k$ die Reihenfolge der Punkte auf dem Kreis, ausgehend von $\mathbf{u}$ im Uhrzeigersinn.

Wir müssen zeigen, dass drei benachbarte Segmente $\alpha_i, \alpha_j, \alpha_k$ auf der Beachline entstehen, bevor die Sweepline ℓ_s den Punkt $\mathbf{u}$ passiert.
Da C keine weiteren Konturpunkte auf dem Rand oder im Inneren enthält, existiert ein Kreis C' durch $\mathbf{v}_i$ und $\mathbf{v}_j$, der die Sweepline ℓ_s berührt, so dass alle anderen Konturpunkte außerhalb von C' liegen. Also existieren benachbarte Segmente α_i und α_j definiert durch $\mathbf{v}_i$ und $\mathbf{v}_j$ auf der Beachline. Analog existiert ein Kreis C'' durch $\mathbf{v}_j$ und $\mathbf{v}_k$ und daher gibt es benachbarte Segmente α'_j und α_k auf der Beachline. Nach Konstruktion ist dann aber $\alpha_j = \alpha'_j$. Wir erhalten ein Tripel benachbarter Segmente $\alpha_i, \alpha_j, \alpha_k$ auf der Beachline. Also wird das zugehörige Event in Q gespeichert und wir erkennen eine Voronoi-Ecke. □

Zusammengefasst erhalten wir den folgenden Algorithmus.

8.24 Algorithmus: Sei $K = \{\mathbf{v}_j \mid j = 0, \ldots, N-1\} \subset \mathbb{R}^2$ eine Kontur.
`function D=Voronoi(K)`

(1) Wir initialisieren die priorisierte Statusliste Q mit allen Punkt-Events. Des weiteren erzeugen wir eine leere Statusliste $\mathcal{T}$ sowie eine zweifach verlinkte Liste $\mathcal{D}$.

(2) `while` $Q \neq \emptyset$

(3) `do` Wir entfernen das Event mit der größten y–Koordinate aus Q.

(4) `if` Event ist ein Punkt-Event auf $\mathbf{v}_j$
`PunktEvent` $(\mathbf{v}_j)$
`else`
`KreisEvent` (γ), wobei wir unter γ das Blatt von $\mathcal{T}$ verstehen, das zu dem Segment gehört, welches entfernt wird.

(5) In der Liste stehen jetzt auch die Einträge zu den Kanten, die durch Halbgeraden dargestellt werden. Wir berechnen ein Rechteck, das alle Ecken des Voronoi-Diagramms umfasst. Dadurch können wir die halb unendlichen Kanten identifizieren und die Liste D entsprechend anpassen.

(6) Wir überprüfen die Einträge von Q auf Halbgeraden und passen die Einträge entsprechend an.

Die Funktionen, welche die einzelnen Events behandeln, sehen dann wie folgt aus.

`PunktEvent`($\mathbf{v}_j$)

(1) Ist $\mathcal{T} = \emptyset$, so fügen wir $\mathbf{v}_j$ zu $\mathcal{T}$ hinzu. Dann besteht $\mathcal{T}$ aus einem einzelnen Blatt und wir beenden `PunktEvent`. Ansonsten fahren wir mit den Punkten (2) bis (5) fort.

(2) Wir durchsuchen $\mathcal{T}$ nach dem Segment α oberhalb von $\mathbf{v}_j$. Hat das Blatt, das α repräsentiert, bereits einen Verweis zu einem Kreis-Event in Q, dann wurde dieses Kreis-Event fälschlicherweise erkannt und wir müssen es aus Q löschen.

(3) Wir tauschen das Blatt von $\mathcal{T}$, das α repräsentiert, gegen einen Teilbaum mit drei Blättern aus. Das mittlere Blatt speichert den neue Konturpunkt $\mathbf{v}_i$ und die restlichen beiden Blätter beziehen sich auf den Punkt $\mathbf{v}_j$, der zu dem Segment α gehört. Wir speichern ebenfalls die Punktepaare $\langle \mathbf{v}_j, \mathbf{v}_i \rangle$ und $\langle \mathbf{v}_i, \mathbf{v}_j \rangle$, welche die Schnittpunkte der Segmente repräsentieren.

(4) Jetzt erzeugen wir eine neue Kante des Voronoi-Diagramms. Diese bildet die Grenze zwischen $\mathcal{VR}(\mathbf{v}_j)$ und $\mathcal{VR}(\mathbf{v}_i)$.

(5) In diesem Schritt überprüfen wir die Tripel von benachbarten Segmenten, in denen $\mathbf{v}_i$ ein äußeres Segment darstellt, hinsichtlich der Konvergenz der Schnittpunkte. Falls Konvergenz gewährleistet ist, fügen wir das Event in die Statusliste Q ein und schreiben die zugehörigen Einträge in die Liste $\mathcal{T}$.

`KreisEvent`(γ)

(1) Wir löschen das Blatt $\gamma \in \mathcal{T}$, welches das zu entfernende Segment α repräsentiert. Danach überprüfen wir die Einträge aus $\mathcal{T}$, die zu den internen Schnittpunkten der Segmente gehören. Anschließend löschen wir alle Kreis-Events aus Q, die das Segment α enthalten.

(2) Den Mittelpunkt des Kreises, der durch dieses Event hervorgerufen wird, fügen wir als Ecke zu $\mathcal{D}$ hinzu. Danach erzeugen wir zwei Kanteneinträge, die zu den neuen Schnittpunkten der Segmente gehören.

(3) Wir überprüfen das Tripel der Segmente, in denen α das linke bzw. das rechte Segment ist, hinsichtlich der Konvergenz der Schnittpunkte. Falls Konvergenz gewährleistet ist, fügen wir das zugehörige Kreis-Event zu Q hinzu.

Jetzt wenden wir uns der Geschwindigkeit und dem erforderlichen Speicherplatz des Algorithmus zu.
Wir haben eingangs erwähnt, dass eine direkte Methode eine Komplexität von $O(N^2 \log N)$ erreichen kann. Kann das Problem auch schneller gelöst werden? Wir können uns leicht überlegen, dass die Berechnung von Voronoi-Diagrammen im $\mathbb{R}^2$ auf das Problem der Sortierung von N reellen Zahlen reduziert werden kann. Da das Sortierproblem in $O(N \log N)$ Zeit gelöst werden kann, beträgt auch die optimale Komplexität für die Berechnung der Voronoi-Diagramme $O(N \log N)$.

8.25 Lemma: Algorithmus 8.24 hat eine Komplexität von $O(N \log N)$ und benötigt $O(N)$ Speicher.

Beweis: Als Erstes überprüfen wir die Komplexität der einfachen Operationen. Solche Vorgänge auf $\mathcal{T}$ und Q, wie das Hinzufügen oder Löschen eines Elementes, benötigen nach [Knu02] $O(\log N)$ Zeit, während dieselben Operationen auf $\mathcal{D}$ in konstanter Zeit zu bewältigen sind. Um ein Event zu behandeln, benötigen wir stets eine konstante Anzahl solcher Operationen, daher bleibt es bei der Behandlung eines Events bei $O(\log N)$.
Wir betrachten N Punkte, also müssen wir auch N verschiedene Events behandeln. Aus jedem Kreis-Event geht eine Ecke des Voronoi-Diagramms $\mathcal{VD}(K)$ hervor. Finden wir allerdings ein Kreis-Event, das sich als fälschlicherweise gefunden herausstellt, so wird dieses Event noch vor der Behandlung aus Q gelöscht. Während der Behandlung eines Events erzeugen und löschen wir weitere Events. Da die Laufzeit stets bei $O(\log N)$ bleibt, können wir diese dem aktuellen Event anrechnen und bleiben stets bei derselben Komplexität. Daher ist die Anzahl der Kreis-Events, die wir behandeln, höchstens $2N - 5$ und wir erhalten die Gesamtkomplexität von $O(N \log N)$ sowie den gesamten benötigten Speicherplatz von $O(N)$. □

Wir haben zu Beginn angenommen, dass die Punkte aus K in allgemeiner Lage vorliegen. Da diese Annahme in der Praxis nicht aufrecht zu erhalten ist, wollen wir uns nun mit den möglichen Degenerationen und deren Auswirkungen auf den Algorithmus 8.24 beschäftigen.

(1) Die erste Möglichkeit der Degeneration der Punkte aus K besteht darin, dass zwei oder mehr Punkte auf einer horizontalen Gerade liegen, also dieselbe y–Koordinate haben. Wir benötigen eine Sonderbehandlung, falls diese Punkte als erste Punkte im Algorithmus vorkommen.

Treten solche Punkte mitten im Algorithmus auf, stellt diese Konstellation kein Problem dar. Dann wird eine vertikale Kante mit den zugehörigen Ecken erstellt.

(2) Die nächste Degeneration kann auftreten, wenn vier oder mehr Punkte auf einem Kreis liegen. Eine Möglichkeit wäre es, eine Ecke der Ordnung vier zu erzeugen. Algorithmus 8.24 geht einen anderen Weg und erzeugt zwei Ecken der Ordnung drei, die durch eine Kante der Länge Null verbunden sind. Auf diese Weise benötigen wir für diesen Fall keine Sonderbehandlung, die Kante der Länge Null bereitet im Algorithmus keine Probleme.

(3) Ein anderer Typ tritt bei der Behandlung eines Events auf, wenn ein Punkt $\mathbf{v}_j$ exakt unterhalb eines Segmentenschnittpunktes liegt. Der Algorithmus erzeugt in diesem Fall zwei neue Segmente, eines davon hat die Länge Null. Das Segment der Länge Null ist dann der mittlere Teil eines Segmentes, das ein Kreis-Event definiert. Wird dieses Event behandelt, so wird eine Ecke in $\mathcal{VD}(K)$ erzeugt. Das Segment der Länge Null kann später gelöscht werden.

(4) Die letzte Art der Degeneration von Punkten kann vorkommen, wenn drei benachbarte Segmente durch drei kollineare, horizontale Punkte erzeugt werden. In diesem Fall wird korrekterweise weder ein Kreis-Event erzeugt, noch durch diese Punkte ein Kreis gebildet.

Wir können also sämtliche möglichen Degenerationen der Punkte behandeln. Dabei benötigt der Algorithmus 8.24 bis auf eine Ausnahme keine zusätzliche Sonderbehandlung. Daher können wir den folgenden Satz formulieren.

8.26 Satz: Das Voronoi-Diagramm einer Menge oder Kontur $K \subset \mathbb{R}^2$ von N Punkten kann mit dem Algorithmus 8.24 in $O(N \log N)$ Zeit unter Benutzung von $O(N)$ Speicherplatz berechnet werden.

8.4.2 Berechnung des Pferchkreises

Mit diesen Erkenntnissen können wir uns jetzt der Berechnung des Pferchkreises zuwenden. Wir wissen, dass nach Satz 8.18 eine der inneren Ecken des Voronoi-Diagramms $\mathcal{VD}(K)$ gleichzeitig der Mittelpunkt des Pferchkreises ist. Die Struktur von Algorithmus 8.24 bietet uns die Möglichkeit,

sämtliche Ecken des Diagramms mitsamt den zugehörigen Punkten der Kontur zu speichern. Wir definieren das Ausgabefeld $\mathbf{V}_j = [V_j^{(1)} \ \cdots \ V_j^{(5)}]$ für $j = 0, \ldots, J-1$ mit
$[V_j^{(1)} \ V_j^{(2)}]$ als Koordinaten der j–ten Voronoi-Ecke und
$V_j^{(n)}$ für $n = 3, 4, 5$ als Index des zugehörigen Konturpunktes.
Zusammengefasst erhalten wir den folgenden Algorithmus.

8.27 Algorithmus: Sei $K = \{\mathbf{v}_j \mid j = 0, \ldots, N-1\} \subset \mathbb{R}^2$ eine Kontur. `PKreis`(K)

(1) Wir entfernen alle doppelt vorhandenen, hintereinander liegenden Punkte aus K. Das heißt wir erzeugen eine Kontur $K' \subseteq K$ mit $K' = \{\mathbf{v}_j \in K \mid \mathbf{v}_j \neq \mathbf{v}_{j+1}\}$ und $|K'| = N'$. Für die nächsten Schritte bezeichnen wir K' mit K und N' mit N.

(2) Dann berechnen wir das Voronoi-Diagramm $\mathcal{VD}(K)$ mit Hilfe des Algorithmus 8.24 und erhalten die Liste $\mathbf{V}_j$ für $j = 0, \ldots, J-1$ als Rückgabe, wobei J die Anzahl der Voronoi-Ecken ist. Dabei gilt nach Lemma 8.17 $J \leq 2N - 5$.

(3) Wir prüfen für alle $j = 0, \ldots, J-1$, ob die Ecke $\mathbf{e}_j = [V_j^{(1)} \ V_j^{(2)}]$ eine innere Ecke der Kontur ist, ob sie sich also innerhalb der Kontur K befindet.

(4) Ist Punkt (3) erfüllt, berechnen wir den Radius des Kreises $C_j = (\mathbf{e}_j, r_j)$ durch $r_j = \|\mathbf{e}_j - \mathbf{v}_{V_j^{(3)}}\|_2$.

(5) Wir setzen

$$D_P(K) = (\mathbf{e}_n, r_n) \text{ mit } n = \text{argmax}_{0 \leq j < J} r_j.$$

Wir stellen nun die Frage nach der Komplexität des gesamten Algorithmus. Wir wissen nach Satz 8.26, dass die Komplexität für (2) $O(N \log N)$ beträgt. Für die Schritte (1), (4) und (5) benötigen wir jeweils $O(N)$ Zeit. Es bleibt Schritt (3), in dem entschieden wird, ob eine Voronoi-Ecke innerhalb oder außerhalb einer Kontur liegt.
Dazu berechnen wir für jede Voronoi-Ecke $\mathbf{e}_j$ für $j = 0, \ldots, J-1$ die Gerade $\mathbf{g}_j(x) = y = V_j^{(2)}$. Anschließend bilden wir die Menge

$$S_j = \{\mathbf{s} \in \mathbb{R}^2 \mid \mathbf{s} = \mathbf{g}_j \cap \mathbf{h}_k \text{ für } k = 0, \ldots, N-2\}.$$

Dabei ist $\mathbf{h}_k$ die Strecke zwischen $\mathbf{v}_k$ und $\mathbf{v}_{k+1}$, also

$$\mathbf{h}_k = \mathbf{h}_k(\mu) = \mathbf{v}_k + \mu(\mathbf{v}_{k+1} - \mathbf{v}_k) \text{ für } \mu \in [0,1].$$

Wir setzen

$$S_j^{(1)} = \{\mathbf{s} \in S_j \mid s^{(1)} < V_j^{(1)}\}$$

und entsprechend

$$S_j^{(2)} = \{\mathbf{s} \in S_j \mid s^{(1)} > V_j^{(1)}\}.$$

Da eine Voronoi-Ecke nicht exakt am Rand des Polygons liegen kann, erhalten wir

$$S_j = S_j^{(1)} \cup S_j^{(2)} \text{ und } S_j^{(1)} \cap S_j^{(2)} = \emptyset.$$

Wir können uns leicht überlegen, dass entweder beide Teilmengen $S_j^{(\cdot)}$ eine gerade oder beide eine ungerade Anzahl an Elementen erhalten. Gilt

$$\left|S_j^{(1)}\right| = 2n_1 + 1 \text{ und } \left|S_j^{(2)}\right| = 2n_2 + 1 \text{ für } n_1, n_2 \in \mathbb{N}_0,$$

so liegt die Voronoi-Ecke $\mathbf{e}_j$ innerhalb der Kontur K. Anderenfalls liegt diese Ecke außerhalb von K.
Wir müssen für jede der Ecken alle Punkte der Kontur überprüfen, daher liegt die Laufzeit bei $O(JN)$.
Wir erhalten die folgende Aussage über die Gesamtlaufzeit des Algorithmus 8.27.

8.28 Satz: Sei $K = \{\mathbf{v}_j \mid j = 0, \ldots, N-1\} \subset \mathbb{R}^2$. Den zu der Kontur K gehörigen Pferchkreis können wir mit Hilfe des Algorithmus 8.27 in $O(NJ + N \log N + N)$ Zeit berechnen. Da nach Lemma 8.17 ein Voronoi-Diagramm einer Menge mit N Punkten höchstens $2N - 5$ Voronoi-Ecken haben kann, erhalten wir eine Gesamtlaufzeit von

$$\begin{aligned} O(NJ + N \log N + N) &= O(N(2N-5) + N \log N + N) \\ &= O\left(2N^2 + N \log N - 4N\right) \\ &= O\left(N^2\right). \end{aligned}$$

9 Filter

9.1 Einleitung

Sowohl bei der optischen als auch bei der taktilen Messung geschieht es häufig, dass die entstehenden Konturen entweder zu fein abgetastet wurden oder dass auf Grund anderer Umstände die Ergebnisse verrauscht sind. Somit hat auch in den Bereichen der Messtechnik das Entrauschen von Daten zunehmend an Bedeutung gewonnen.
Wir wollen in diesem Kapitel verschiedene Rausch- bzw. Rauheitsfiltertechniken diskutieren. Zunächst erläutern wir die klassischen, nach ISO zertifizierten Verfahren, die auf der diskreten Faltung mit der Gauß-Funktion beruhen. Diesen Filterverfahren wollen wir von uns entwickelte Filter auf Spline- und Wavelet-Basis gegenüberstellen.
Im ersten Abschnitt stellen wir die Rauheitsfilter nach ISO 11562, ISO 13565-1 und ISO/TR 16610-10 vor. Anschließend führen wir ein Filterverfahren auf Basis von Smoothing Splines ein. Dann wollen wir uns mit der Möglichkeit beschäftigen, Rauheitsfilter auf Wavelet-Basis zu entwerfen. Im vorletzten Abschnitt folgt eine qualitative Bewertung der Filterverfahren anhand realer Konturen der Firma *Werth Messtechnik GmbH*. Ein Abschnitt über ein Spezialfilter für Fasertasterkonturen schließt dieses Kapitel ab.

9.2 Klassische Rauheitsfilter

Bevor wir mit der Einführung der klassischen, nach ISO zertifizierten Rauheitsfilter beginnen, müssen wir unsere Notation angleichen und darüber hinaus einige zusätzliche Forderungen an die zu bearbeitende Kontur K stellen.
Wir betrachten eine Kontur

$$K = \{\mathbf{v}_j \mid j = 0, \ldots, N-1\} \subset \mathbb{R}^2,$$

die äquidistant abgetastet wurde. Sei $f : \mathbb{R} \to \mathbb{R}$ die unbekannte zu Grunde liegende Funktion, also der Verlauf des idealen Sollelements. Dann können wir $\mathbf{v}_j$ als

$$\mathbf{v}_j = \begin{bmatrix} v_j \\ f(v_j) \end{bmatrix} \text{ für } j = 0, \ldots, N-1$$

schreiben. Es gilt $v_{j+1} > v_j$ für $j = 0, \ldots, N-2$ und wir setzen $\Delta v = v_{j+1} - v_j$ für ein $j = 0, \ldots, N-2$. Abkürzend wollen wir $f_j = f(v_j)$ für alle

$j = 0, \ldots, N-1$ schreiben.
Wir betrachten die klassischen Rauheitsfilter ausschließlich für äquidistant abgetastete $\mathbb{R}^2$–Konturen.
Wir führen folgende Sprechweise ein.

9.1 Bezeichnung: Unter dem *Primärprofil*, auch *P-Profil* genannt, verstehen wir in diesem Kontext die Funktionswerte von f und bezeichnen die Menge mit

$$K^{(2)} = \{f_j \mid j = 0, \ldots, N-1\}.$$

Das *Welligkeitsprofil*, auch *W-Profil* genannt, stellt den langwelligen Anteil des P-Profils dar. Die Berechnung des W-Profils erfolgt durch die Anwendung eines Tiefpassfilters.
Das *Rauheitsprofil*, oder *R-Profil*, zeigt uns den kurzwelligen Anteil des P-Profils und wird entsprechend durch die Anwendung eines Profilfilters mit Hochpasscharakteristik gewonnen.

Was aber verstehen wir unter dem Begriff eines *Profilfilters*? Ein *Profilfilter* in der Rauheitsmessung entspricht einem Hoch- oder Tiefpassfilter, also einer Transformation der Daten, bei der gewisse Anteile aus dem Frequenzband entfernt werden, während andere erhalten bleiben.
In der klassischen Rauheitsmessung wird ein Profilfilter in der Regel durch eine gewichtete diskrete Faltung realisiert. Konkret berechnen wir einen gewichteten Mittelwert der P-Profil-Werte f_j mit einer noch zu definierenden Funktion $s_j = s(j\Delta v, \Lambda)$, welche die Übertragungscharakteristik bestimmt. Der veränderbare Parameter Λ erlaubt uns, den Übertragungsbereich zwischen den lang- und den kurzwelligen Signalanteilen zu kontrollieren.
Wir unterscheiden zwischen Tiefpassprofilfiltern zur Bestimmung des langwelligen Profilverlaufes und Hochpassprofilfiltern, die dementsprechend der Bestimmung des kurzwelligen Profilverlaufes dienen. Mit Hilfe des Gauß-Filters können wir sowohl für das Tiefpass- als auch für das Hochpassfilter als Filterkern dieselbe Gewichtsfunktion s_j benutzen, ohne den Parameter Λ zu verändern.
Allgemein können wir die Elemente x_k eines *W-Profils* W mit Hilfe eines Tiefpassprofilfilters als Faltung

$$x_k = \sum_{j=0}^{N-1} v_j s_{k-j} \text{ mit } \sum_{j=0}^{N-1} s_{k-j} = 1 \text{ für } k = 0, \ldots, N-1 \qquad (9.1)$$

bestimmen. Für hinreichend kleine Abtastraten Δv gilt näherungsweise

$$x_k = \int_0^l f(\xi) s(k\Delta v - \xi, \Lambda) d\xi$$
$$\text{mit } \int_0^l s(k\Delta v - \xi, \Lambda) d\xi = 1 \text{ für } k = 0, \ldots N-1.$$

Die Größe l heißt *Länge der Messstrecke*, auf sie werden wir später genauer eingehen. Das entsprechende Hochpassprofilfilter und das daraus resultierende *R-Profil* erhalten wir als

$$r_k = f_k - x_k \text{ für } k = 0, \ldots, N-1. \tag{9.2}$$

Die *Grenzwellenlänge* λ_{co} eines Profilfilters ist gleich der Wellenlänge eines kosinusförmigen Profils

$$f(v_k) = A_0 \cos\left(\frac{2\pi k \Delta v}{\lambda_{co}}\right), \tag{9.3}$$

das durch das Profilfilter auf 50% seiner Amplitude A_0 gedämpft wird. Formal bestimmen wir die Grenzwellenlänge derart, dass

$$\lim_{l\to\infty}\left(\frac{\int_{-l/2}^{l/2} \cos\left(\frac{2\pi\xi}{\lambda_{co}}\right) s(\xi,\Lambda) d\xi}{\int_{-l/2}^{l/2} s(\xi,\Lambda) d\xi}\right) = \frac{1}{2}$$

gilt.
Als nächstes wollen wir die *Länge der Messstrecke* l festlegen. Wir setzen $ln = v_{N-1} - v_0$ und bezeichnen B_{eff} als *effektiv wirksame Breite*. Im Falle des Gaußfilters setzen wir $B_{eff} = \lambda_{co}$. Für die *Länge der Messstrecke* l gilt dann

$$l = ln + B_{eff}.$$

Wir stellen jetzt die drei wichtigsten nach ISO zertifizierten Profilfilter vor.

9.2.1 Gauß-Filter nach ISO 11562

Für das Gauß-Filter nach ISO 11562 verwenden wir als Filterkern die Gaußsche Funktion

$$s_G(x) = \frac{1}{\sqrt{2\pi\Lambda}} e^{-\frac{x^2}{2\Lambda^2}}. \tag{9.4}$$

Wir erhalten für die Parameter λ_{co} und Λ die Beziehung

$$\Lambda = \lambda_{co}\frac{1}{\pi}\sqrt{\frac{\ln 2}{2}}.$$

Wie bereits erwähnt, wählen wir als Messstreckenlänge

$$l = ln + \lambda_{co}.$$

9.2.2 Sonderfilterverfahren nach ISO 13565-1

Das Sonderfilterverfahren nach ISO 13565-1 ist ein zweistufiges Verfahren, welches das Gauß-Filter nach ISO 11562 als Basis nutzt. Die Vorgehensweise ist die folgende.

(1) Wir bestimmen das W-Profil $W^{[0]}$ nach (9.1) aus dem P-Profil f_k für $k = 0, \ldots, N-1$.

(2) Im nächsten Schritt bilden wir das so genannte temporäre Profil

$$W^{[1]} = \{x_j^{[1]} \mid j = 0, \ldots, N-1\}$$

mit

$$x_k^{[1]} = \begin{cases} f_k, & \text{falls } f_k \geq x_k \\ x_k, & \text{falls } f_k < x_k \end{cases} \text{ für } k = 0, \ldots, N-1.$$

(3) Im letzten Schritt berechnen wir das W-Profil $W^{[2]} = \{x_j^{[2]} \mid j = 0, \ldots, N-1\}$ durch die Anwendung von (9.1) auf das temporäre Profil $W^{[1]}$. Wir setzen $W = W^{[2]}$.

Als Grenzwellenlänge λ_{co} werden durch die ISO die Werte 0.5 bzw 2.5 vorgeschlagen. Erneut wählen wir als Länge der Messstrecke $l = ln + \lambda_{co}$.

9.2.3 Gauß-Filter nach ISO/TR 16610-10

Dieses Verfahren wird häufig in den Anwendungen und der Literatur als robust bezeichnet. Diese Aussage beruht auf seiner Eigenschaft, wenig empfindlich gegenüber Ausreißern, also Profilspitzen und -riefen, zu sein.
Das W-Profil $W = \{x_k \mid k = 0, \ldots, N-1\}$ berechnen wir bei diesem Filter iterativ, indem wir die Gleichung

$$x_k^{[j+1]} = \frac{\sum_{i=0}^{N-1} f_i \omega_i^{[j]} s_{k-i}}{\sum_{i=0}^{N-1} \omega_i^{[j]} s_{k-i}} \text{ für } k = 0, \ldots, N-1$$

anwenden. Dabei gilt für die Gewichtsfunktion

$$\omega_k^{[0]} = 1,\ \omega_k^{[j]} = \begin{cases} 0, & \text{falls } f_k - x_k^{[j]} > c \cdot \text{med}^j \\ \left(1 - \left(\frac{f_k - x_k^{[j]}}{c \cdot \text{med}^j}\right)^2\right)^2, & \text{sonst} \end{cases} \quad \text{für } j > 0.$$

Mit med^j bezeichnen wir den Median der Absolutbeträge der Differenz zwischen der j–ten Iteration und den ursprünglichen Werten, also

$$\text{med}^j = \underset{0 \le k < N}{\text{median}} \left\{ \left| f_k - x_k^{[j]} \right| \right\}.$$

Die Konstante $c \in \mathbb{R}$ legen wir so fest, dass eine Nullgewichtung bei einem mittelwertfreien Profil mit gaußverteilten Werten bei Ns_f^2 einsetzt. Dabei ist s_f^2 die Varianz der f_j. Die ISO schlägt den Wert $c = \frac{N}{0.67448975}$ vor.
Als Filterkern wählen wir erneut die Gaußfunktion (9.4). Wir beenden die Iteration, wenn die Änderung des Medians der Absolutbeträge unterhalb einer Schwelle liegt, das heißt wenn

$$\left(\frac{1 - \text{med}^j}{\text{med}^j}\right) \le tol \text{ für } j \ge 1.$$

Wir setzen dann $W = W^{[j]}$.
Analog zu den vorherigen Filterverfahren verwenden wir die Messstreckenlänge $l = ln + \lambda_{co}$.
Zu diesen nach ISO zertifizierten Filterverfahren gehören spezielle Auswertungen der W– und R–Profile, für die es ebenfalls eine ISO-Zertifizierung gibt. Hierbei werden gewisse Werte ausgewertet und bewertet, welche für die Beurteilung der Oberfläche eine Rolle spielen.
Dazu gehören Werte und Größen wie *Profil-*, *Wellen-* und *Rauheitstiefe*, *arithmetischer Mittelrauhwert*, *gemittelte Rauhtiefe* und viele andere. Diese Werte lassen sich, wie wir später sehen werden, bei den anderen Filterverfahren auf Spline- und Waveletbasis berechnen und spielen daher für unsere Betrachtung keine bedeutende Rolle.

9.3 Filter auf Splinebasis

Es gibt verschiedene Möglichkeiten, Konturen mit Hilfe von Splinekurven zu filtern. Gegenüber den nach ISO zertifizierten Filtern haben Splines den Vorteil, dass es sich hierbei um echte Filter im $\mathbb{R}^d$ handelt. Es wird also keine Projektion oder Umwandlung der Daten benötigt. Die Implementierung für *Werth Messtechnik GmbH* erfolgte im $\mathbb{R}^3$, wir wollen in diesem

Abschnitt aber allgemein Splinefilter im $\mathbb{R}^d$ behandeln.
Wie bisher betrachten wir eine Kontur

$$K = \{\mathbf{v}_j \mid j = 0, \ldots, N-1\} \subset \mathbb{R}^d.$$

Unter einem Filter auf Splinebasis verstehen wir eine die Daten approximierende Splinekurve und zum Berechnen dieser bieten sich folgende Möglichkeiten.

(1) *Algorithmus von de Boor*

(2) *Least-Square Splines*

(3) *Smoothing Splines nach G. Wahba*

Wir wollen im Folgenden alle drei Möglichkeiten diskutieren und auf ihre Tauglichkeit als Filter eingehen.

(1) *Algorithmus von de Boor*
Wir betrachten die Eingangsdaten $K = \{\mathbf{v}_j \mid j = 0, \ldots, N-1\} \subset \mathbb{R}^d$ als Kontrollpolygon. Das heißt wir setzen $\mathbf{D}_j = \mathbf{v}_j$ für $j = 0, \ldots, N-1$ und wenden auf diese den Algorithmus von de Boor 3.6 an.
Das Ergebnis des Algorithmus ist eine Splinekurve $x \mapsto \mathbf{D}_j^m(x)$ für $x \in [t_m, t_N]$, die wir mit $N_{m,T}\mathbf{D}$ bezeichnen. In der Literatur wird diese Splinekurve häufig Quasiinterpolant genannt. Leider neigt die auf diese Weise erzeugte Kurve dazu, bei verrauschten Daten weit von den Datenpunkten abzuweichen. Zudem gib es nur eine Möglichkeit der Korrektur der Kurve. Wir erhalten eine andere Splinekurve $N_{m,T}\mathbf{D}$, indem wir den Grad m verändern.
Daher eignet sich die resultierende Kurve nicht für unsere Zwecke als Filter.

(2) *Least-Square Spline*
Eine zweite Möglichkeit ist die Anpassung der Daten mittels eines Least-Square Splines. Dabei versuchen wir, zu K eine Splinekurve $N(x) = N_{m,T}\mathbf{D}(x)$ so zu bestimmen, dass der Ausdruck

$$\sum_{j=0}^{N-1} \|N(a_j) - \mathbf{v}_j\|_2 \tag{9.5}$$

minimiert wird. Wir berechnen anhand der Daten eine Knotenfolge $T_{m,n}$ und die Menge der Abzissen $A = \{a_j \mid j = 0, \ldots, N-1\}$. Gilt

$n = N$, so erzielen wir keine Approximation der Daten sondern die bereits besprochene Interpolation. Daher wollen wir stets $n < N$ annehmen. Mit der Frage, wie man eine solche Knotenfolge bestimmt, beschäftigen wir uns in diesem Kontext nicht. Da jedoch $T_{m,n}$ und A der Interpolationsbedingung von Schönberg-Whitney (3.9) genügen sollten, muss eine Untermenge $A^* = \{a_{j_k} \mid k = 0, \ldots, n-1\} \subset A$ existieren, so dass

$$t_j < a_j < t_{j+m+1} \text{ für } j = j_0, \ldots, j_{n-1} \tag{9.6}$$

gilt. Wir setzen $T = T_{m,n}$ und

$$\mathbf{v} = \begin{bmatrix} \mathbf{v}_0^t \\ \vdots \\ \mathbf{v}_{N-1}^t \end{bmatrix} \in \mathbb{R}^{N \times d}, \quad \mathbf{D} = \begin{bmatrix} \mathbf{D}_{j_0}^t \\ \vdots \\ \mathbf{D}_{j_{n-1}}^t \end{bmatrix} \in \mathbb{R}^{n \times d}$$

sowie

$$\mathbf{M} = \begin{bmatrix} N_{j_0}^m(a_0|T) & \ldots & N_{j_{n-1}}^m(a_0|T) \\ \vdots & \ddots & \vdots \\ N_{j_0}^m(a_{N-1}|T) & \ldots & N_{j_{n-1}}^m(a_{N-1}|T) \end{bmatrix} \in \mathbb{R}^{N \times n}. \tag{9.7}$$

Dann können wir (9.5) schreiben als

$$\begin{aligned} \sum_{j=0}^{N-1} \|N(a_j) - \mathbf{v}_j\|_2^2 =& (\mathbf{MD} - \mathbf{v})^t (\mathbf{MD} - \mathbf{v}) \\ =& \mathbf{D}^t \mathbf{M}^t \mathbf{MD} - \mathbf{D}^t \mathbf{M}^t \mathbf{v} - \mathbf{v}^t \mathbf{MD} + \mathbf{v}^t \mathbf{v}. \end{aligned} \tag{9.8}$$

Da der Ausdruck (9.8) quadratisch in $\mathbf{D}$ ist, erhalten wir die Lösung des Minimierungsproblems als Lösung des Gleichungssystems

$$\mathbf{M}^t \mathbf{MD}^{(j)} = \mathbf{M}^t \mathbf{v}^{(j)} \text{ für } j = 1, \ldots, d. \tag{9.9}$$

Hierbei bezeichnen wir mit $\mathbf{v}^{(j)}$ bzw. $\mathbf{D}^{(j)}$ die j–te Spalte von $\mathbf{v}$ bzw. $\mathbf{D}$. Die Matrix $\mathbf{M}^t\mathbf{M}$ ist symmetrisch, positiv definit, $(2m+1, 2m+1)$–bandiert und wir können das Gleichungssystem (9.9) effizient lösen, zum Beispiel mit Hilfe der LR-Zerlegung und anschließender Gauß-Elimination ohne Pivotsuche. Als Ergebnis erhalten wir den Least-Square Spline

$$N_{LS}(x) = \sum_{j \in \{j_0, \ldots, j_{n-1}\}} \mathbf{D}_j N_j^m(x|T).$$

Es gibt viele Modifikationen dieses Verfahrens, etwa indem wir

$$\sum_{j=0}^{N-1} \|N(a_j) - \mathbf{v}_j\|_2^2 \sigma_j$$

minimieren. Die Gewichte $\sigma_j > 0$ steuern die Nähe der Splinekurve zu den Daten, je größer σ_j ist, desto näher liegt die Kurve an dem Datenpunkt $\mathbf{v}_j$ und umgekehrt.
Auch diese Kurve ist letzten Endes nicht für unsere Zwecke einsetzbar, da in gewissen Situationen eine Oszillation zu beobachten ist.

(3) *Smoothing Spline*
Die dritte und letzte von uns diskutierte Möglichkeit, Splines als Filter zu nutzen, ist die Verwendung der so genannten *Smoothing Splines* nach *Grace Wahba*. Wir gehen hier einen Kompromiss zwischen der Nähe der resultierenden Kurve zu den Daten und dem Wunsch nach einer möglichst glatten Kurve ein.
Die Einführung wird einfacher, wenn wir uns, ähnlich wie bei der Interpolation, auf eindimensionale Daten beschränken und bei mehrdimensionalen Daten jede Koordinate separat approximieren. Daher steht $v_j = v_j^{(k)}$ im Folgenden für eine beliebige Koordinate, etwa die k–te Koordinate, von $\mathbf{v}_j$. Das Maß für die Abweichung der Kurve und der Daten wird für gewöhnlich wie folgt bestimmt

$$E(f) = \sum_{j=0}^{N-1} \|v_j - f(a_j)\|_2^2.$$

Die Glattheit einer Funktion wird mit Hilfe einer Ableitung gemessen als

$$R(f) = \int_a^b \left(f^{(k)}(t)\right)^2 dt.$$

Hierbei ist $[a, b]$ das für die Daten relevante Intervall. Die beiden Größen $E(f)$ und $R(f)$ wollen wir im Folgenden minimieren. Wir definieren den *Smoothing Spline* f_ζ als

$$f_\zeta = \min_{f \in \mathcal{C}(\mathbb{R})} (\zeta E(f) + (1 - \zeta) R(f)) \tag{9.10}$$

mit $\zeta \in [0, 1]$. ζ ist ein Parameter, durch den der Grad der Glattheit kontrolliert wird. Für ζ können wir zwei Spezialfälle betrachten.

- Im Fall $\zeta \to 0$ erhalten wir eine interpolierende Splinekurve.
- $\zeta = 1$ liefert uns eine Least-Square-Anpassung an die Daten.

Schreiben wir Gleichung (9.10) aus, so erhalten wir

$$f_\zeta = \min_{f \in \mathcal{C}(\mathbb{R})} \left(\zeta \sum_{j=0}^{N-1} (v_j - f(a_j))^2 + (1-\zeta) \int_a^b \left(f^{(k)}(t) \right)^2 dt \right). \quad (9.11)$$

Wollen wir mehr Einfluss auf das Verhalten der Splinekurve nehmen, können wir zusätzliche Parameter $\sigma_j > 0$ hinzufügen. Wir erhalten das allgemeine Smoothing Spline-Problem

$$f_\zeta = \min_{f \in \mathcal{C}(\mathbb{R})} \left(\zeta \sum_{j=0}^{N-1} (v_j - f(a_j))^2 \sigma_j + (1-\zeta) \int_a^b \left(f^{(k)}(t) \right)^2 dt \right). \quad (9.12)$$

Durch die Kombination der Parameter ζ und $\sigma_j > 0$ für $j = 0, \ldots, N-1$ erhalten wir eine Kurve, die sich sehr gut als filternde Kurve eignet. Wie gut die Filtereigenschaft ist, werden wir im Vergleich mit den Standard-Filter-Verfahren und den Wavelet-Filtern in Abschnitt 9.5 sehen. Die für die approximierenden Splines typische und in unserem Fall unerwünschte Oszillation tritt durch das Zusammenspiel der Parameter in der Regel nicht auf.
Daher wollen wir im Folgenden ein Verfahren zur Berechnung der Smoothing Splines vorstellen.

9.3.1 Algorithmus zu Berechnung von Smoothing Splines

Wie wir bereits gesehen haben, sind Smoothing Splines nichts weiter als Least-Square Splines mit zusätzlichen Parametern. Einer dieser Parameter kontrolliert die Glattheit der Splinekurve, während der zweite Parameter eine bessere Anpassung an die Daten gewährleistet.
Ein Smoothing Spline $\mathbf{N}^{Spl} \in \mathbb{S}_m(T)$ für eine Knotenfolge $T = T_{m,N-1}$ ist die Lösung des *penalized criteriums*

$$L_{pen} = \min_{\mathbf{N}^{spl} \in \mathbb{S}} \left(\zeta \sum_{j=0}^{N-1} \|\mathbf{N}^{spl}(a_j) - \mathbf{v}_j\|_2^2 \sigma_j + (1-\zeta) \int_{a_0}^{a_{N-1}} \|(\mathbf{N}^{spl})^{(k)}(x)\|_2^2 dx \right). \quad (9.13)$$

Hierbei sind $\mathbf{v}_0, \ldots, \mathbf{v}_{N-1} \in \mathbb{R}^d$ die zu Grunde liegenden Daten, $a_0 < \ldots < a_{N-1} \in \mathbb{R}$ die an die Daten angepassten Abszissen, deren Berechnung wir

bereits im Lemma 3.21 erörtert haben, und $\mathbf{N}_j^{spl}$, $j = 0, \ldots, N-1$ der j–te Spline vom Grad m.
In [dB78] wurde gezeigt, dass die optimale Lösung des Minimierungsproblems (9.13) ein Spline der Ordnung $k+1$ ist. Daher wollen wir im Folgenden den Fall $k = 2$ betrachten, der als Lösung einen kubischen Spline $\mathbf{N}^{spl}$ hat.
Wir verwenden die Darstellung einer Splinekurve und eines B-Splines aus Abschnitt 3. Sei hierfür $T = T_{m,N-1}^* = \{t_0, \ldots, t_{m+N}\} \subset \mathbb{R}$ eine Knotenfolge mit $(m+1)$–fachen Anfangs- und Endknoten sowie eine Menge von Abszissen $A = \{a_j \mid j = 0, \ldots, N-1\}$, die der Schoenberg-Whitney-Bedingung (3.9) genügen. Dann gilt

$$\mathbf{N}(x) = N_{m,T}\mathbf{D}(x) = \sum_{j=0}^{N-1} \mathbf{D}_j N_j^m(x|T).$$

Dabei ist $N_j^m(x|T)$ der j–te B-Spline, welchen wir rekursiv durch

$$N_j^0(x|T) = \chi_{[t_j, t_{j+1})}(x) \text{ für } j = 0, \ldots, N+m-1$$

und für $1 \leq k \leq m$

$$N_j^k(x|T) = \frac{x - t_j}{t_{j+k} - t_j} N_j^{k-1}(x|T) + \frac{t_{j+k+1} - x}{t_{j+k+1} - t_{j+1}} N_{j+1}^{k-1}(x|T)$$

mit $j = 0, \ldots, N+m-k-1$ definieren. Wir suchen also das Kontrollpolygon $\mathbf{D} = (\mathbf{D}_0, \ldots, \mathbf{D}_{N-1})$, welches die Bedingung (9.13) erfüllt.
Wir beschränken uns im Folgenden auf den Fall $d = 1$, da wir ähnlich wie bei einer Interpolation, jede Komponente separat approximieren. Wir betrachten daher den zu minimierenden Ausdruck folgender Gestalt

$$\underbrace{\zeta \sum_{j=0}^{N-1} |N(a_j) - v_j|^2 \sigma_j}_{=A} + (1-\zeta) \underbrace{\int_{a_0}^{a_{N-1}} (N''(x))^2 dx}_{=B}.$$

Die Summe A können wir durch Matrizen darstellen als

$$A = (\mathbf{MD} - \mathbf{v})^t \mathbf{\Sigma} (\mathbf{MD} - \mathbf{v})$$

mit

$$\mathbf{M} = \begin{bmatrix} N_0^3(a_0|T) & \ldots & N_{N-1}^3(a_0|T) \\ \vdots & \ddots & \vdots \\ N_0^3(a_{N-1}|T) & \ldots & N_{N-1}^3(a_{N-1}|T) \end{bmatrix} \in \mathbb{R}^{N \times N}, \mathbf{D} = \begin{bmatrix} D_0 \\ \vdots \\ D_{N-1} \end{bmatrix} \in \mathbb{R}^N,$$

$$\mathbf{v} = \begin{bmatrix} v_0 \\ \vdots \\ v_{N-1} \end{bmatrix} \in \mathbb{R}^N \text{ und } \mathbf{\Sigma} = \begin{bmatrix} \sigma_0 & & \\ & \ddots & \\ & & \sigma_{N-1} \end{bmatrix} \in \mathbb{R}^{N\times N}.$$

Für das Integral B erhalten wir

$$\begin{aligned} B &= \int_{a_0}^{a_{N-1}} (N''(x))^2 dx = \sum_{j=0}^{N-2} \int_{a_j}^{a_{j+1}} (N''(x))^2 dx \\ &= \sum_{j=0}^{N-2} \int_{a_j}^{a_{j+1}} \left(\sum_{k=0}^{N-1} D_k \frac{d^2}{dx^2} N_k^3(x|T) \right) \left(\sum_{i=0}^{N-1} D_i \frac{d^2}{dx^2} N_i^3(x|T) \right) dx \\ &= \sum_{j=0}^{N-2} \int_{a_j}^{a_{j+1}} (\mathbf{D}^t \mathbf{b}(x))(\mathbf{D}^t \mathbf{b}(x))^t dx \qquad (9.14) \\ &= \mathbf{D}^t \sum_{j=0}^{N-2} \left(\int_{a_j}^{a_{j+1}} \mathbf{b}(x)\mathbf{b}(x)^t dx \right) \mathbf{D}. \end{aligned}$$

Dabei ist

$$\mathbf{b}(x) = \begin{bmatrix} \frac{d^2}{dx^2} N_0^3(x|T) \\ \vdots \\ \frac{d^2}{dx^2} N_{N-1}^3(x|T) \end{bmatrix} \in \mathbb{R}^N.$$

Wir setzen

$$\mathbf{Q}_j = \int_{a_j}^{a_{j+1}} \mathbf{b}(x)\mathbf{b}(x)^t dx \text{ für } j = 0, \ldots, N-2$$

und betrachten die Struktur dieser Matrix. Hierfür müssen wir zunächst die Produkte $\frac{d^2}{dx^2} N_k^3(x|T) \frac{d^2}{dx^2} N_i^3(x|T)$ für $k, i = 0, \ldots, N-1$ in den Intervallen (a_j, a_{j+1}) für $j = 0, \ldots, N-2$ auswerten, denn es gilt

$$q_{ki}^{(j)} = \int_{a_j}^{a_{j+1}} \frac{d^2}{dx^2} N_k^3(x|T) \frac{d^2}{dx^2} N_i^3(x|T) dx$$

mit $\mathbf{Q}_j = (q_{ki}^{(j)})_{k,i=0}^{N-1}$. Damit sind die Matrizen $\mathbf{Q}_j$ für $j = 0, \ldots, N-2$ symmetrisch. Aus Abschnitt 3.3 ist bekannt, dass ein B-Spline $N_k^3(x|T)$ den Träger (t_k, t_{k+4}) besitzt. Wir berechnen die Abszissen a_j so, dass diese der Schoenberg-Whitney-Bedingung genügen. Es gilt also $t_k < a_k < t_{k+4}$ für $k = 0, \ldots, N-1$. Es ergibt sich daher

$$(a_j, a_{j+1}) \subset (t_j, t_{j+5}).$$

Wir können folgern

$$N_k(x|T) \equiv 0 \text{ für } x \in (a_j, a_{j+1}) \text{ und } k \notin \{j-3, \ldots, j+4\}.$$

Also sind die Einträge $q_{ki}^{(j)}$ der Matrix $\mathbf{Q}_j$ für $k, i \in \{j-3, \ldots, j+4\}$ ungleich Null. Hierbei gilt

$$q_{j-3,i}^{(j)} \neq 0 \text{ für } i = j-3, \ldots, j+4$$

sowie

$$q_{j+4,i}^{(j)} \neq 0 \text{ für } i = j-3, \ldots, j+4.$$

Setzen wir $\mathbf{Q} = \sum_{j=0}^{N-2} \mathbf{Q}_j$, so ist $\mathbf{Q}$ eine $(7,7)$–bandierte Matrix. Die Eigenschaften der Matrix $\mathbf{Q}$ fassen wir in dem nachfolgenden Lemma zusammen.

9.2 Lemma: Sei $\mathbf{Q} \in \mathbb{R}^{N \times N}$ wie oben definiert. Dann ist $\mathbf{Q}$ eine symmetrische, $(7,7)$–bandierte, positiv semidefinite Matrix.

Beweis: Da die Matrizen $\mathbf{Q}_j$ symmetrisch sind und die Menge der symmetrischen $N \times N$ Matrizen bezüglich der Addition eine Gruppe bildet, ist auch $\mathbf{Q}$ symmetrisch.
Wir betrachten nun $\mathbf{Q}_j$ für ein $j = 0, \ldots, N-2$. Die Matrix hat die Gestalt

$$\mathbf{Q}_j = \begin{bmatrix} 0 & & & & & & \\ & \ddots & & & & & \\ & & 0 & & & & \\ & & & \mathbf{H}_j & & & \\ & & & & 0 & & \\ & & & & & \ddots & \\ & & & & & & 0 \end{bmatrix}, \mathbf{H}_j = \begin{bmatrix} q_{j-3,j-3}^{(j)} & \cdots & q_{j-3,j+4}^{(j)} \\ \vdots & \ddots & \vdots \\ q_{j+4,j-3}^{(j)} & \cdots & q_{j+4,j+4}^{(j)} \end{bmatrix} \in \mathbb{R}^{8 \times 8}.$$

Seien $\mathbf{e}_0, \ldots, \mathbf{e}_{N-1} \in \mathbb{R}^N$ Einheitsvektoren des $\mathbb{R}^N$. Dann gilt $\mathbf{e}_k^t \mathbf{Q}_j \mathbf{e}_k = 0$ für $k \notin \{j-3, \ldots, j+4\}$. Für $k \in \{j-3, \ldots, j+4\}$ erhalten wir

$$\mathbf{e}_k^t \mathbf{Q}_j \mathbf{e}_k = q_{kk}^{(j)} = \int_{a_j}^{a_{j+1}} \left(\frac{d^2}{dx^2} N_k^3(x|T) \right)^2 \geq 0.$$

Damit gilt $\mathbf{x}^t \mathbf{Q}_j \mathbf{x} \geq 0$ für alle $\mathbf{x} \in \mathbb{R}^N$ sowie $j = 0, \ldots, N-2$ und die Matrizen $\mathbf{Q}_j$ sind positiv semidefinit. Insbesondere ist $\mathbf{Q}$ als Summe positiv semidefiniter Matrizen selbst positiv semidefinit. □

Wir können das penalized criterium in Matrixform als

$$L_{pen} = \min_{N \in \mathbb{S}(T)} \zeta \left(\mathbf{MD} - \mathbf{v}\right)^t \mathbf{\Sigma} (\mathbf{MD} - \mathbf{v}) + (1 - \zeta) \left(\mathbf{D}^t \mathbf{Q} \mathbf{D}\right) \tag{9.15}$$

schreiben. Wir finden ein Minimum, indem wir die Lösung des linearen Gleichungssystems

$$\left(\mathbf{M}^t \mathbf{\Sigma} \mathbf{M} + \mu \mathbf{Q}\right) \mathbf{D} = \mathbf{M}^t \mathbf{\Sigma} \mathbf{v} \tag{9.16}$$

mit

$$\mu = \frac{2(1-\zeta)}{3\zeta}.$$

bestimmen. Wir können das lineare Gleichungssystem (9.16) effizient lösen, denn es gilt die folgende Aussage.

9.3 Lemma: Die Matrix

$$\mathbf{M}^t \mathbf{\Sigma} \mathbf{M} + \mu \mathbf{Q}$$

aus (9.16) ist eine symmetrische, $(7,7)$–bandierte, positiv semidefinite Matrix.

Beweis: Die Matrix $\mathbf{M}$ ist nach Satz 3.23 eine symmetrische, $(4,4)$–bandierte, total positive Matrix. Daher ist die Matrix $\mathbf{M}^t \mathbf{\Sigma} \mathbf{M}$ ebenfalls symmetrisch und $(4,4)$–bandiert.
Wir verwenden nun die folgende Aussage:
Eine symmetrische Matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$ ist genau dann positiv semidefinit, wenn eine Matrix $\mathbf{U} \in \mathbb{R}^{N \times N}$ existiert, so dass $\mathbf{A} = \mathbf{U}^t \mathbf{U}$ gilt.
Da $\sqrt{\mathbf{\Sigma}} \in \mathbb{R}^{N \times N}$ wegen $\sigma_j > 0$ für alle $j = 0, \ldots, N-1$ existiert, gilt

$$\begin{aligned} \mathbf{M}^t \mathbf{\Sigma} \mathbf{M} &= \mathbf{M}^t \sqrt{\mathbf{\Sigma}} \sqrt{\mathbf{\Sigma}} \mathbf{M} = \mathbf{M}^t \sqrt{\mathbf{\Sigma}}^t \sqrt{\mathbf{\Sigma}} \mathbf{M} \\ &= \left(\sqrt{\mathbf{\Sigma}} \mathbf{M}\right)^t \left(\sqrt{\mathbf{\Sigma}} \mathbf{M}\right) = \mathbf{U}^t \mathbf{U}, \end{aligned}$$

also ist $\mathbf{M}^t \mathbf{\Sigma} \mathbf{M}$ positiv semidefinit. Mit der Aussage von Lemma 9.2 ist damit die Matrix $\mathbf{M}^t \mathbf{\Sigma} \mathbf{M} + \mathbf{Q}$ ebenfalls positiv semidefinit. □

Die Laufzeit des Verfahrens betrachten wir gemeinsam mit den anderen Verfahren in einem gesonderten Abschnitt.
Ein Aspekt der smoothing Splines haben wir allerdings unerwähnt gelassen, nämlich die Wahl des Glattheitparameters ζ. Wir haben in der

endgültigen Version für *Werth Messtechnik GmbH* diese Wahl dem Benutzer überlassen. Für die automatische Wahl des Parameters, abhängig von den Daten, verweisen wir auf das Konzept der *Cross Validation* von *Grace Wahba*, beschrieben in [Wah90].

9.4 Filter auf Wavelet-Basis

Die Filtertechniken auf Waveletbasis beziehen sich in der Regel auf Rauschfilter. Dabei gehen wir von der Annahme aus, dass ein diskret abgetastetes Signal g einer unbekannten Funktion f zu Grunde liegt. Diese Funktion f wollen wir mit Hilfe des Filterverfahrens ermitteln. Wir können den Sachverhalt durch eine Gleichung beschreiben

$$g(v_k) = f(v_k) + \Gamma(v_k)\xi_k \text{ für } k = 0, \ldots, N-1. \tag{9.17}$$

$\Gamma(\cdot)$ bezeichnen wir als *Rauschstärke*, die nicht notwendig konstant sein muss. Die Folge $\{\xi_k\}_{k=0}^{N-1}$ ist die Realisierung des *weißen Rauschens*, entsprechend der nachfolgenden Definition. Wie in den meisten Anwendungen üblich, fassen wir die Rauschstärke als Standardabweichung des weißen Rauschens auf.

9.4 Definition: Seien $\xi_0, \ldots, \xi_{N-1}$ unabhängige und identisch normalverteilte Zufallsvariablen mit Erwartungswert Null und Standardabweichung Eins, also $\xi_k \overset{iid}{\backsim} \mathcal{N}(0,1)$. Dann nennen wir $\{\xi_k\}_{k=0}^{N-1}$ *diskretes weißes Rauschen*.

Die Namensgebung hängt mit dem Verhalten des Spektrums von weißem Rauschen zusammen. Ähnlich wie bei weißem Licht enthält das Spektrum der $\{\xi_k\}_{k=0}^{N-1}$ gleichmäßig Anteile aller Frequenzen.
Wir wollen die Rauschfiltertechniken zur Rauheitsfilterung, also zur Profilfilterung, mit Auswertung der Rauheit der Oberfläche und der Rekonstruktion des Profilverlaufs verwenden. Die unbekannte zu Grunde liegende Funktion f aus Gleichung (9.17) ist für uns das Profil oder die Kontur, die wir rekonstruieren wollen. Ebenso wie wir die Funktion f nicht kennen, ist uns der Verlauf der Kontur unbekannt. Die Profilrauheit spielt die Rolle des Rauschens. Damit beschreibt Modell (9.17) die Modellannahme einer Profilfilterung.
Wir diskutieren in diesem Abschnitt die Grundidee der Rauheitsfilter mit Hilfe von Wavelets und geben die Standardverfahren an. Hierzu gehören

die Methode der Frequenztrennung sowie die Hard- und Soft-Treshold-Verfahren nach Donoho und Johnstone. Im nächsten Abschnitt vergleichen wir diese Methoden qualitativ mit den klassischen Methoden nach ISO und dem Filterverfahren auf Spline-Basis.

9.4.1 Die Idee der Filter auf Wavelet-Basis

Wie in Abschnitt 3 bezeichnen wir mit $L_\psi f(j,k)$ die Wavelettransformierte von f. Die zugehörigen Waveletkoeffizienten sind $w_{j,k}$.
Die Idee der Techniken auf Wavelet-Basis beruht im Wesentlichen auf den folgenden beiden Punkten.

(1) Die Wavelettransformierte $L_\psi f(j,k)$ einer glatten Funktion f, bzw. deren Abtastung $f(x_i)$, ist in einem schmalen Frequenzband des Waveletraums lokalisiert. Daher repräsentieren nur einige wenige Waveletkoeffizienten $w_{j,k}$ die gesamte Funktion.

(2) Weißes Rauschen tritt hingegen in allen Waveletskalen gleichmäßig mit derselben Stärke auf.

Wir können ein Filterverfahren realisieren, indem wir das zu Grunde liegende Profil nur aus den Waveletkoeffizienten rekonstruieren, welche die Kontur repräsentieren. Wir bezeichnen die repräsentativen Waveletkoeffizienten mit $w^*_{j,k}$ und das rekonstruierte Profil mit $f^*(j,k)$. Nachfolgend wollen wir die grundlegenden Techniken zur Auswahl und Korrektur der Waveletkoeffizienten vorstellen.

9.4.2 Methode der Frequenztrennung

Liegen uns Informationen über die Frequenzskalen, die das Profil f repräsentieren, vor und ist die Rauheit, also das Rauschen, nicht zu stark, so können wir eine *Frequenztrennung* durchführen.
Seien $J_s = \{j_l \mid l = 0, \ldots, s\}$ die Skalen, die f repräsentieren. Wir behalten die Waveletkoeffizienten der repräsentativen Skalen bei und setzen alle anderen Koeffizienten auf Null. Damit gilt

$$w^*_{j,k} = \begin{cases} w_{j,k} & \text{für } j \in J_s \text{ und } k \in \mathbb{Z} \\ 0 & \text{sonst.} \end{cases} \tag{9.18}$$

Wir wenden die schnelle Wavelettransformation (4.37) auf die Eingangskontur an, führen auf allen Skalen die Frequenztrennung gemäß (9.18)

durch und erhalten durch die Anwendung der schnellen inversen Wavelettransformation (4.40) ein rauschgemindertes oder rauheitsgemindertes Profil. Neben dem notwendigen Vorwissen über die zu Grunde liegende Kontur wäre als weiterer Nachteil auch die Tatsache zu nennen, dass in den Waveletkoeffzienten der Skalen J_s der komplette Rauschanteil enthalten bleibt.
Bei der Rauheitsmessung könnten wir die Methode der Frequenztrennung zum Beispiel dann anwenden, wenn wir zu dem gemessenen Profil Informationen über ein ideales Sollelement und einige Messgrößen haben, wie zum Beispiel die Tastkugelgröße. In diesem Fall sind wir in der Lage, die repräsentativen Skalen J_s zu bestimmen. Allerdings bleibt das Rauschen, also in unserem Fall die Rauheit, in allen nicht repräsentativen Skalen erhalten.

9.4.3 Hard-Treshold-Verfahren

Die Idee des *Hard-Treshold-Verfahrens* ist ähnlich jener der Frequenztrennung. Wir setzen alle Waveletkoeffizienten sämtlicher Skalen auf Null, wenn diese kleiner als ein Schwellenwert τ_N sind. Wir wählen diesen Wert τ_N als

$$\tau_N = \sigma\sqrt{2\ln N}. \tag{9.19}$$

Dabei ist σ die Standardabweichung und N die Länge des ursprünglichen Profils. In der Praxis wird manchmal auch ein von N unabhängiger Schwellenwert benutzt, etwa $\tau_\sigma = 3.0\sigma$. Dieser ist häufig kleiner als der Wert aus Gleichung (9.19), also wird der größte Teil des Rauschens entfernt, das Profil bleibt aber erhalten.
Ausführliche Betrachtungen zur Wahl von τ_N finden wir in den Arbeiten von Donoho [Do92] und Donoho und Johnstone [DoJo92]. Der Schwellenwert τ_σ wird dort als *universeller Schwellenwert* bezeichnet.
Für ein Hard-Treshold-Verfahren erhalten wir die Vorschrift

$$w^*_{j,k} = \begin{cases} 0, & \text{falls } |w_{j,k}| < \tau_N \\ w_{j,k}, & \text{sonst.} \end{cases} \tag{9.20}$$

Eine schematische Darstellung des Verfahrens sehen wir in Abbildung 40. Die Hard-Treshold-Methode hat, ähnlich der Methode der Frequenztrennung, den Nachteil, dass der in den großen Waveletkoeffizienten enthaltene Rauschanteil nicht entfernt wird.

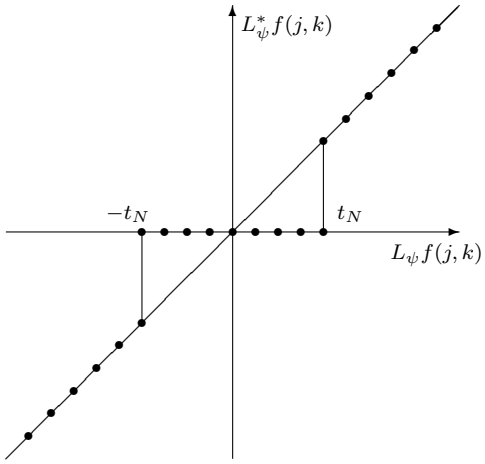

Abbildung 40: Schema des Hard-Treshold-Verfahrens. Die durchgehende Linie bezeichnet die ursprünglichen Waveletkoeffizienten, die Punkte die korrigierten Koeffizienten.

9.4.4 Soft-Treshold-Verfahren nach Donoho und Johnstone

Das hier vorgestellte *Soft-Treshold-Verfahren* behebt die Nachteile der vorherigen Methoden, bei denen die Rauschanteile in den großen Waveletkoeffizienten erhalten blieben. Wir berechnen die neuen Waveletkoeffizienten als

$$w^*_{j,k} = \begin{cases} 0, & \text{falls } |w_{j,k}| < \tau_N \\ \operatorname{sgn}(w_{j,k})\left(|w_{j,k}| - \tau_N\right), & \text{sonst.} \end{cases} \tag{9.21}$$

Den Schwellenwert τ_N wählen wir wie bei dem Hard-Treshold-Verfahren gemäß (9.19). Mit anderen Worten entfernen wir aus allen Waveletkoeffizienten den maximalen möglichen Rauschanteil. Eine schematische Darstellung des Soft-Treshold-Verfahrens nach Donoho und Johnstone zeigt uns Abbildung 41.

9.5 Vergleich der Filterverfahren

In diesem Abschnitt vergleichen wir die in der Koordinatenmesstechnik momentan verwendeten Verfahren zur Rauheitsmessung und -filterung mit den von uns vorgestellten Verfahren auf Spline- und Waveletbasis.
Wir tun dies am Beispiel dreier Konturen aus einer Serie von Profilmessungen der Firma *Werth Messtechnik GmbH*. Dabei handelt es sich um taktile Messungen mit einer Kugel mit jeweils ca. 2000 Punkten und einer Tastergeschwindigkeit von etwa 0.0035 $\frac{mm}{s}$.
Um gleiche Bedingungen für die Vergleiche zu gewährleisten, werden die

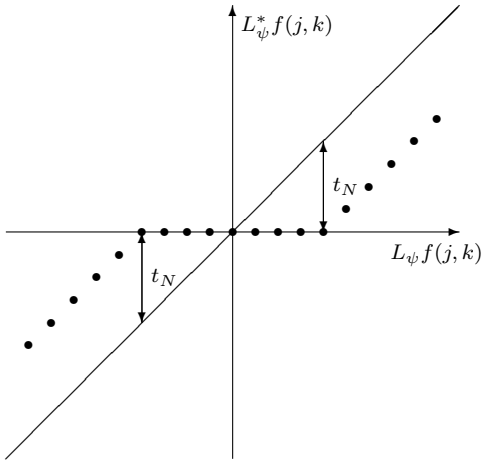

Abbildung 41: Schema des Soft-Treshold-Verfahrens. Die durchgehende Linie bezeichnet die ursprünglichen Waveletkoeffizienten, die Punkte die korrigierten Koeffizienten.

Daten in die xy–Ebene transformiert. Die ISO-Filter sind nur zweidimensional, die Spline- und Wavelet-Filter sind allgemeine Verfahren im $\mathbb{R}^d$, $d \geq 1$.
Wir vergleichen die folgenden Filterverfahren.

- Gauß-Filter nach ISO 11562
- Sonderfilter nach ISO 13565-1
- Robuster Gauß-Filter nach ISO 16610-10
- Filter auf Basis von Smoothing Splines mit den Parameterwerten $\zeta \in \{0.5, 0.75, 0.85, 0.95\}$.
- Filter auf Wavelet-Basis. Als Wavelets wählen wir die biorthogonalen Spline-Wavelets $2,2$ und $3,3$ sowie das Daubechies-Wavelet D_3, jeweils mit der Methode der Frequenztrennung, Hard- und Soft-Tresholding. Wir berechnen jeweils sechs Skalen. Für die Frequenztrennung setzen wir die ersten fünf Skalen auf Null, für Hard- und Soft-Tresholding verwenden wir den Schwellenwert τ_N gemäß (9.19).

Allen drei Konturen liegen plateauartige Profile zu Grunde, das heißt wir können den für die Messung relevanten Bereich mit einer Ausgleichsgerade wiedergeben. Es wird sich allerdings zeigen, dass diese Annahme nicht immer Bestand hat.
Wie wir in Abschnitt 9.2 gesehen haben, erhalten wir durch eine Filterung mit den ISO-Filterverfahren ein W-Profil, das die gefilterte Kontur darstellt, und ein R-Profil, aus dem gewissen Rauheitskennzahlen berechnet

werden. Diese Kennzahlen können wir auch für die Spline- und Waveletfilter berechnen, wenn wir wie folgt vorgehen.

- *Berechnung der Rauheitskennzahlen für das Verfahren aus Abschnitt 9.3*
  Wir haben eine Kontur $K = \{\mathbf{v}_j \mid j = 0, \ldots, N-1\} \subset \mathbb{R}^d$ sowie eine Parametrisierung $A = \{a_j \mid j = 0, \ldots, N-1\}$. Wir wenden auf K das Filterverfahren an und erhalten die gefilterte Kontur $K^* = \{\mathbf{v}_j^* \mid j = 0, \ldots, N-1\} \subset \mathbb{R}^d$ mit derselben Parametrisierung A. Gemäß Abschnitt 9.2 bilden wir für jede Koordinate separat das W- und das R-Profil. Wir setzen für $k = 1, \ldots, d$

  $$K_W^{(k)} = \{\mathbf{w}_j^{(k)} \mid j = 0, \ldots, N-1\} \text{ mit } \mathbf{w}_j^{(k)} = \begin{bmatrix} a_j \\ v_j^{*(k)} \end{bmatrix}$$

  und bezeichnen $K_W^{(k)}$ als *W-Profil für die Koordinate k*. Das *R-Profil für die Koordinate k* ist

  $$K_R^{(k)} = \{\mathbf{r}_j^{(k)} \mid j = 0, \ldots, N-1\} \text{ mit } \mathbf{r}_j^{(k)} = \begin{bmatrix} a_j \\ v_j^{(k)} - v_j^{*(k)} \end{bmatrix}.$$

  Aus $R_K^{(k)}$ erhalten wir für jedes $k = 1, \ldots, d$ die ebenfalls nach ISO zertifizierten Kennzahlen.

- *Berechnung der Rauheitskennzahlen für das Verfahren aus Abschnitt 9.4*
  Wollen wir die Waveletfilter im $\mathbb{R}^d$ anwenden, so benötigen wir ebenfalls eine Parametrisierung. Wir verwenden bei unseren Versuchen eine Parametrisierung nach Bogenlänge, beschrieben in Kapitel 3. Mit Hilfe einer solchen Parametrisierung können wir die W- und R-Profile für jede Koordinate $k = 1, \ldots, d$ genauso wie im Fall der Smoothing Splines bestimmen.

Also sind wir in der Lage, für alle drei Arten der Filterung die relevanten Kennzahlen zu berechnen.
Das erste Profil des Vergleichs wollen wir als *Testkontur*$_1$ bezeichnen. Die Ergebnisse sind in Abbildung 42 dargestellt. Wir sehen sofort, dass die auf Wavelets basierenden Verfahren, die mit Hilfe der Frequenztrennung oder mit Hard-Tresholding arbeiten, zwar zuverlässig das feine Rauschen entfernen, ansonsten aber nah am Profil bleiben. Die Wavelet-Filter, die mit

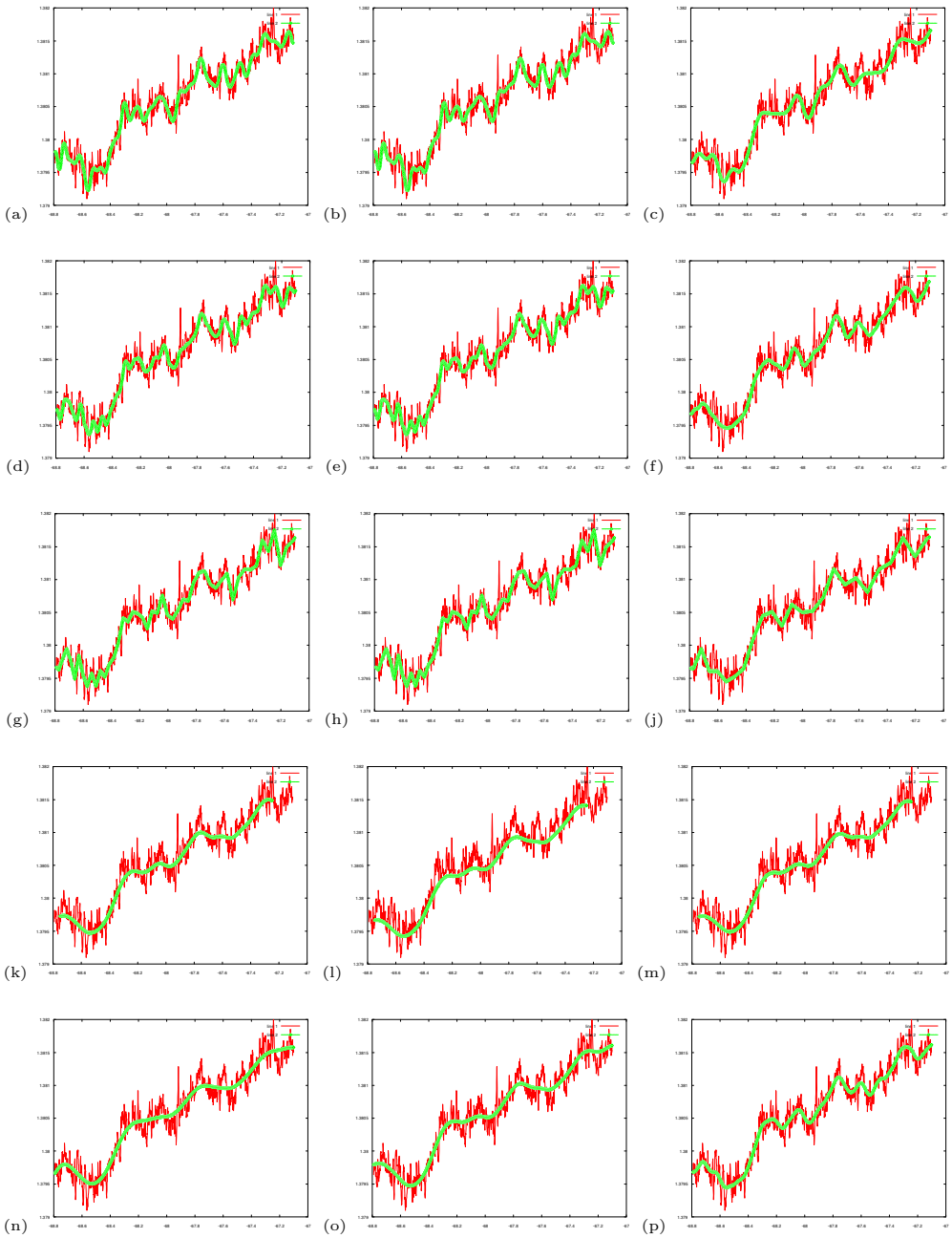

Abbildung 42: Testkontur$_1$ (a) Biorthogonales Spline-Wavelet $2,2$, Methode der Frequenztrennung (b) Biorthogonales Spline-Wavelet $2,2$, Hard-Tresholding (c) Biorthogonales Spline-Wavelet $2,2$, Soft-Tresholding (d) Biorthogonales Spline-Wavelet $3,3$, Methode der Frequenztrennung (e) Biorthogonales Spline-Wavelet $3,3$, Hard-Tresholding (f) Biorthogonales Spline-Wavelet $3,3$, Soft-Tresholding (g) Daubechies Wavelet D_3, Methode der Frequenztrennung (h) Daubechies Wavelet D_3, Hard-Tresholding (j) Daubechies Wavelet D_3, Soft-Tresholding (k) Gauß-Filter nach ISO 11562. (l) Sonderfilter nach ISO 13565-1 (m) Robustes Filter nach ISO 16610-10 (n) Smoothing Spline Filter mit $\zeta = 0.5$ (o) Smoothing Spline Filter mit $\zeta = 0.75$ (p) Smoothing Spline Filter mit $\zeta = 0.85$

Soft-Tresholding arbeiten, glätten die Kontur deutlich besser und kommen, was den Verlauf des gefilterten Profils angeht, nahe an die ISO Filter heran. Die Smoothing Spline Filter mit $\zeta = 0.5$ und $\zeta = 0.75$ erzeugen Profile, die fast exakt den ISO Profilen gleichen.

Das zweite Profil, die *Testkontur*$_2$, zeigt einen deutlich linearen Verlauf, durch einen einzelnen Ausreißer in der Mitte unterbrochen. Wir sehen in Abbildung 43, dass die Ergebnisse die Eindrücke der *Testkontur*$_1$ bestätigen. Während die Methode der Frequenztrennung und Hard-Tresholding im Vergleich nah an der Kontur liegen und das hochfrequente Rauschen eliminieren, liegen die Versuche mit Soft-Tresholding sowie in diesem Fall alle drei durch Smoothing Splines erzeugten Profile sehr nah an den durch die klassischen Filter erzeugten Konturen.

Die dritte Kontur *Testkontur*$_3$ zeigt einen sinusförmigen Verlauf in der rechten Hälfte, wobei die einzelnen Peaks etwa 0.01 mm hoch und ca. 0.2 mm breit sind. Es stellt sich in diesem Kontext natürlich die Frage, ob solche Rillen noch als Rauheit zu bezeichnen oder bereits dem Verlauf des Profils zugehörig sind. Wir sehen in Abbildung 44, dass die klassischen Verfahren nur sehr grob dem Verlauf der Rillen folgen, ähnlich wie die Smoothing Spline Filter mit $\zeta = 0.5$ und $\zeta = 0.75$. Alle anderen Verfahren bilden den Verlauf der Rillen relativ eng nach, wobei erneut die Soft-Tresholding-Verfahren mehr Anteile des niederfrequenten Rauschens entfernen.

Leider liegen uns für die verwendeten Testprofile keine Sollkonturen, also ideale Elemente, vor. Gehen wir allerdings davon aus, dass die Profile einen plateauähnlichen Verlauf haben, können wir das Sollelement durch eine Ausgleichsgerade nachbilden und einen *BestFit*-Vergleich durchführen. Für diesen Vergleich müssen wir nicht mehr alle Verfahren betrachten, wir konzentrieren uns auf das Wavelet-Filter mit dem biorthogonalen Spline-Wavelet $3,3$, das Smoothing Spline-Filter mit $\zeta = 0.75$, sowie das Gauß-Filter nach ISO 11562. Wir verwenden das Profil *Testkontur*$_2$.
Wir bestimmen eine Ausgleichsgerade durch *Testkontur*$_2$ und verwenden diese als ideales Element. Anschließend führen wir eine *BestFit*-Anpassung mit Hilfe eines Gauß-Fits durch. Die von uns verwendeten Grenzen sind 0.0002 mm. Wir erhalten die Anpassungen, die in Abbildung 45 zu sehen sind.
Anhand der *BestFit*-Anpassung können wir beobachten, dass die Profile, die mit den ISO-Filterverfahren und dem Verfahren auf Basis der Smoothing Splines erzeugt wurden, einen ähnlichen Verlauf vorweisen, während

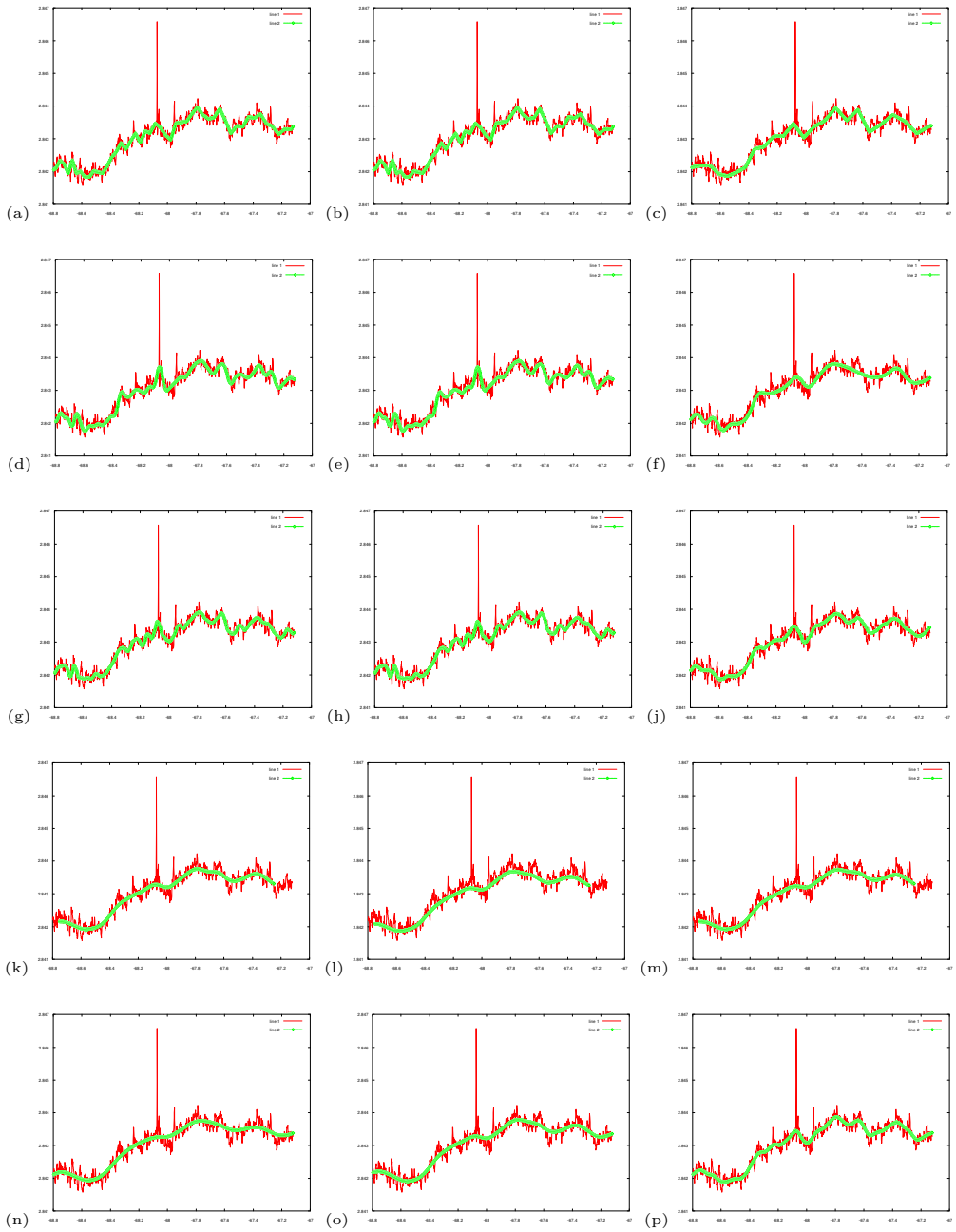

Abbildung 43: Testkontur$_2$ (a) Biorthogonales Spline-Wavelet $2,2$, Methode der Frequenztrennung (b) Biorthogonales Spline-Wavelet $2,2$, Hard-Tresholding (c) Biorthogonales Spline-Wavelet $2,2$, Soft-Tresholding (d) Biorthogonales Spline-Wavelet $3,3$, Methode der Frequenztrennung (e) Biorthogonales Spline-Wavelet $3,3$, Hard-Tresholding (f) Biorthogonales Spline-Wavelet $3,3$, Soft-Tresholding (g) Daubechies Wavelet D_3, Methode der Frequenztrennung (h) Daubechies Wavelet D_3, Hard-Tresholding (j) Daubechies Wavelet D_3, Soft-Tresholding (k) Gauß-Filter nach ISO 11562 (l) Sonderfilter nach ISO 13565-1 (m) Robustes Filter nach ISO 16610-10 (n) Smoothing Spline Filter mit $\zeta = 0.5$ (o) Smoothing Spline Filter mit $\zeta = 0.75$ (p) Smoothing Spline Filter mit $\zeta = 0.85$

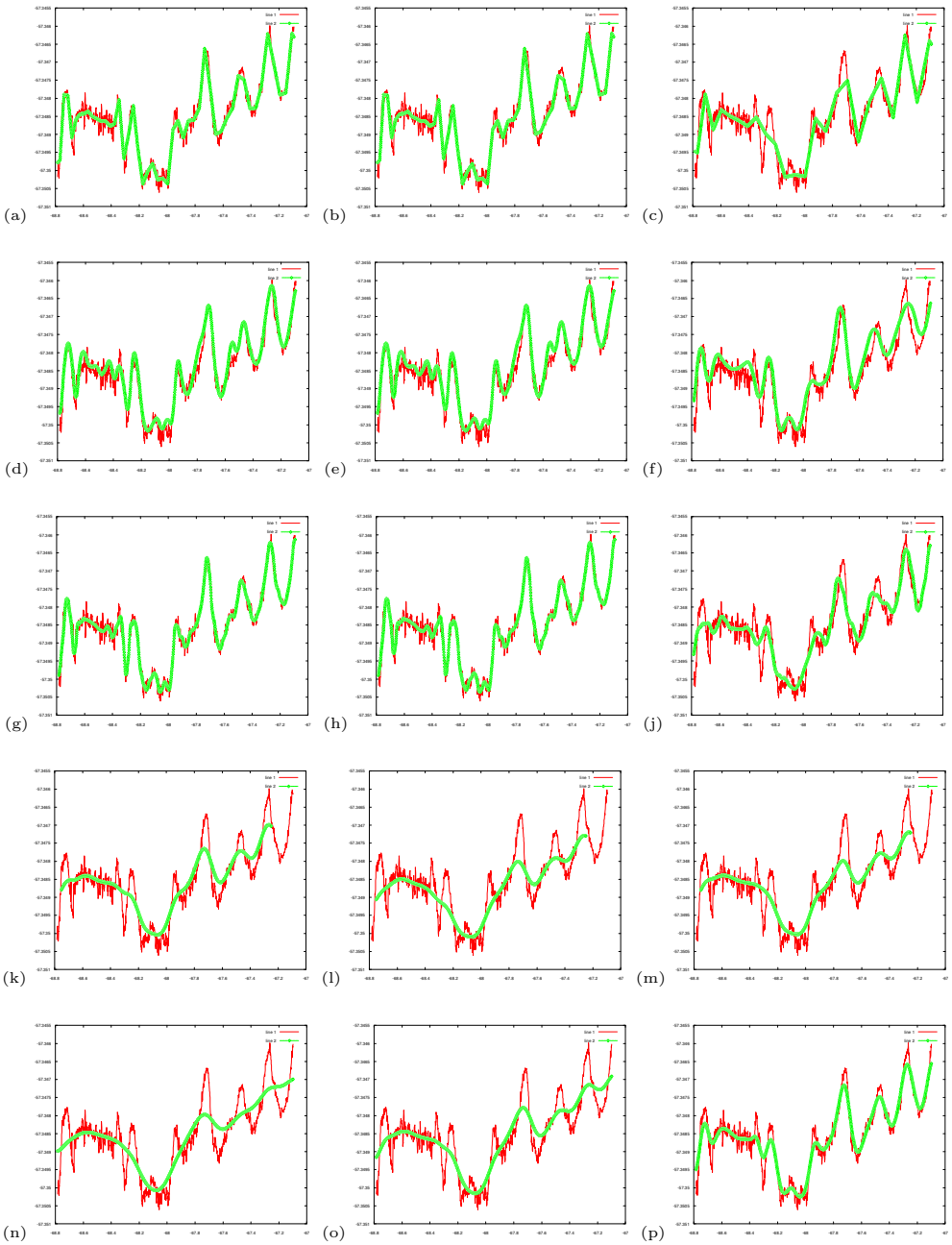

Abbildung 44: Testkontur$_1$ (a) Biorthogonales Spline-Wavelet $2,2$, Methode der Frequenztrennung (b) Biorthogonales Spline-Wavelet $2,2$, Hard-Tresholding (c) Biorthogonales Spline-Wavelet $2,2$, Soft-Tresholding (d) Biorthogonales Spline-Wavelet $3,3$, Methode der Frequenztrennung (e) Biorthogonales Spline-Wavelet $3,3$, Hard-Tresholding (f) Biorthogonales Spline-Wavelet $3,3$, Soft-Tresholding (g) Daubechies Wavelet D_3, Methode der Frequenztrennung (h) Daubechies Wavelet D_3, Hard-Tresholding (j) Daubechies Wavelet D_3, Soft-Tresholding (k) Gauß-Filter nach ISO 11562. (l) Sonderfilter nach ISO 13565-1 (m) Robustes Filter nach ISO 16610-10 (n) Smoothing Spline Filter mit $\zeta = 0.5$ (o) Smoothing Spline Filter mit $\zeta = 0.75$ (p) Smoothing Spline Filter mit $\zeta = 0.85$

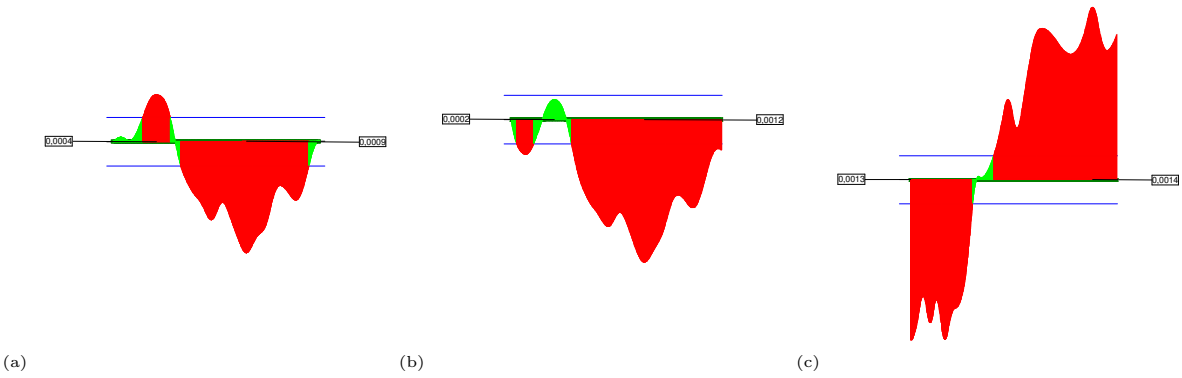

Abbildung 45: Testkontur$_1$ (a) BestFit-Anpassung der nach ISO 11562 gefilterten Kontur (b) BestFit-Anpassung der durch Smoothing Splines gefilterten Kontur (c) BestFit-Anpassung der durch Wavelets gefilterten Kontur

das Profil, das mit dem Waveletverfahren gefiltert wurde, einen deutlich anderen Verlauf zeigt. Es bleibt festzuhalten, dass, insofern wir die nach ISO zertifizierten Filterverfahren als das Maß für andere Verfahren ansehen, lediglich die Filterverfahren auf Basis der Smoothing Splines als Alternative betrachtet werden können. Der Konturverlauf der Waveletverfahren ist ein völlig anderer.

Abschließend beschäftigen wir uns mit einem sinusförmigen Profil, das den geforderten plateauähnlichen Verlauf nicht aufweist. Bei solchen Profilmessungen wurden bisher keine Rauheitsmessungen durchgeführt, weil die vorhandenen ISO-Verfahren, wie wir gleich sehen werden, einen geraden Verlauf als Vorgabe benötigen.
Dem Profil liegt eine ideale sinusförmige Kontur zu Grunde, daher sehen wir in Abbildung 46 sofort, dass die Filter nach ISO zu weit von der Kontur entfernt sind. Die Ergebnisse einer Rauheitsmessung würden mit Hilfe der Filter nach ISO inakzeptable Ergebnisse hervorbringen. Die mit Hard-Thresholding und der Methode der Frequenztrennung erzeugten Ergebnisse der Filter auf Wavelet-Basis sind auch in diesem Fall nur schlecht verwertbar, da diese den hochfrequenten Rauschanteil entfernen und ansonsten nur grob den sinusförmigen Verlauf wiedergeben. Die Ergebnisse der Filterung mit Soft-Tresholding und hier vor allem die auf dem Wavelet D_3 und dem biorthogonalen Spline-Wavelet $3,3$ basierten, erzeugen sehr gute, sinusförmige Verläufe, die in einer *BestFit*-Anpassung nur geringe Toleranzen zu dem idealen Sinusprofil aufweisen. Die auf Smoothing Splines basierenden Filter geben durchweg einen sinusförmigen Verlauf wieder und liegen bei einer *BestFit*-Anpassung in ähnlicher Größenordnung wie

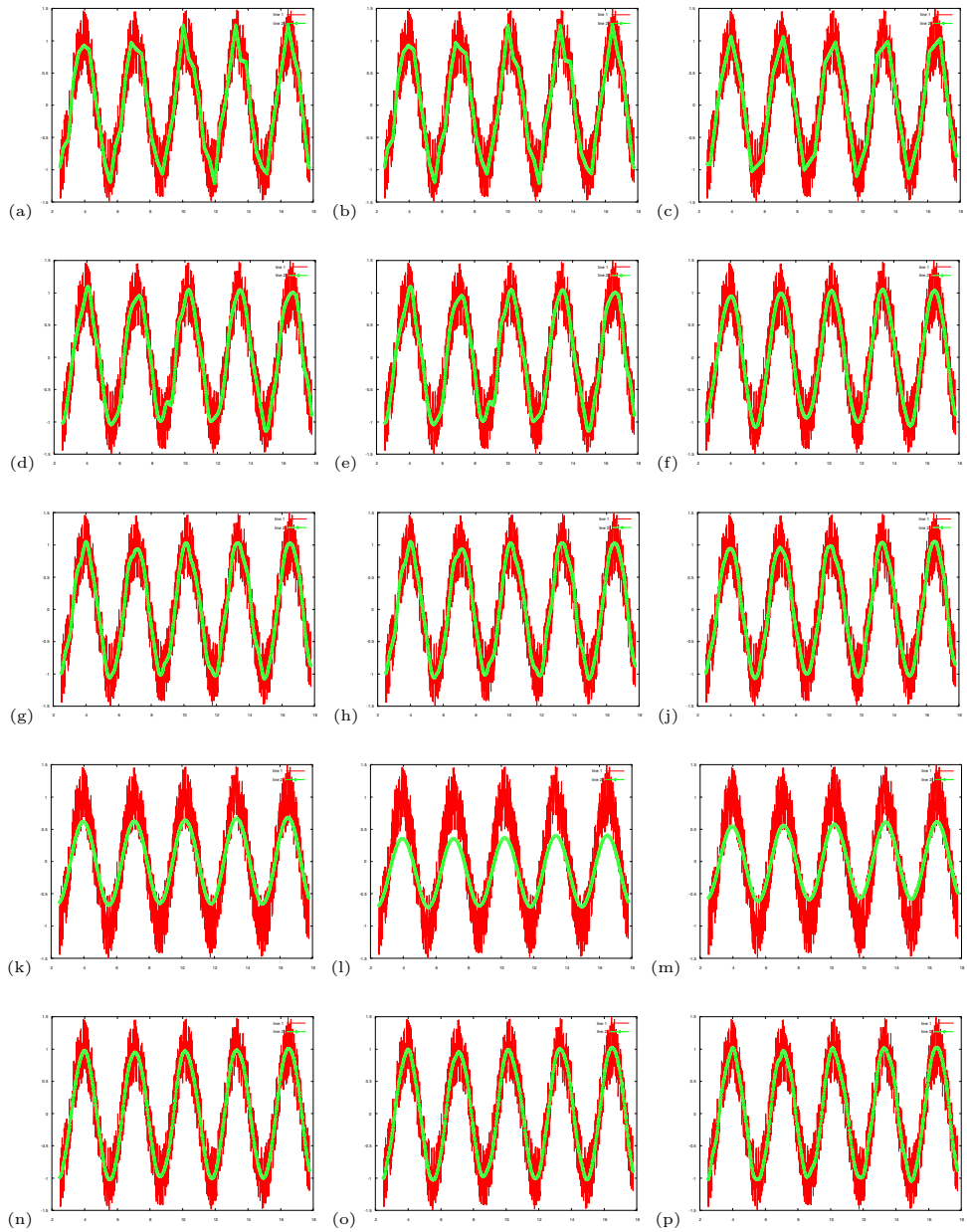

Abbildung 46: Sinusförmige Kontur (a) Biorthogonales Spline-Wavelet $2,2$, Methode der Frequenztrennung (b) Biorthogonales Spline-Wavelet $2,2$, Hard-Tresholding (c) Biorthogonales Spline-Wavelet $2,2$, Soft-Tresholding (d) Biorthogonales Spline-Wavelet $3,3$, Methode der Frequenztrennung (e) Biorthogonales Spline-Wavelet $3,3$, Hard-Tresholding (f) Biorthogonales Spline-Wavelet $3,3$, Soft-Tresholding (g) Daubechies Wavelet D_3, Methode der Frequenztrennung (h) Daubechies Wavelet D_3, Hard-Tresholding (j) Daubechies Wavelet D_3, Soft-Tresholding (k) Gauß-Filter nach ISO 11562 (l) Sonderfilter nach ISO 13565-1 (m) Robustes Filter nach ISO 16610-10 (n) Smoothing Spline Filter mit $\zeta = 0.25$ (o) Smoothing Spline Filter mit $\zeta = 0.5$ (p) Smoothing Spline Filter mit $\zeta = 0.85$

die Wavelet-Filter mit Soft-Tresholding. Auf eine grafische Darstellung der *BestFit*-Anpassungen verzichten wir an dieser Stelle, da auf Grund der Größe der Kontur die Bilddarstellung nicht sehr aussagekräftig ist.

9.5.1 Laufzeit der Verfahren

Die Unterschiede in der Laufzeit können bei der Auswahl eines Filterverfahrens ebenfalls eine Rolle spielen. Daher führen wir eine Laufzeitbetrachtung der drei hier vorgestellten Verfahren durch. Wir verzichten dabei auf die Berechnung der Rauheitskennzahlen und konzentrieren uns ausschließlich auf die Filteralgorithmen.

Filter nach ISO
Alle drei ISO-Filter beruhen auf der diskreten Faltung und diese erfordert N^2 Operationen. Der Einsatz der schnellen Fouriertransformation (FFT) reduziert den Aufwand auf $N \log N$ Operationen. Andere Operationen sind vernachlässigbar, daher haben wir einen Aufwand von $O(N \log N)$ für das Gauß-Filter nach ISO 11562. Das Sonderfilterverfahren nach ISO 13565-1 wird zweimal das Gauß-Filter nach ISO 11562 ausgeführt, also haben wir ebenfalls einen Gesamtaufwand von $O(N \log N)$. Das dritte Verfahren ist iterativ mit einer Anwendung des Gauß-Filters nach ISO 11562 in jedem Durchlauf. Wir erhalten den Gesamtaufwand von $O(maxiter \cdot N \log N) = O(N \log N)$, dabei ist $maxiter$ die maximale Anzahl an Iterationen.

Smoothing Spline Filter
Der aufwändigste Schritt bei der Berechnung der Smoothing Splines ist die Lösung des linearen Gleichungssystems (9.16). Die Matrix des Gleichungssystems ist nach Lemma 9.3 $(7,7)$–bandiert, symmetrisch und positiv semidefinit. Wir benötigen demnach $98N$ Operationen für die Berechnung der LU–Zerlegung sowie jeweils $14N$ Operationen für die Vorwärtelimination und Rückwärtssubstitution. Die Matrix $\mathbf{M}$ ist $(4,4)$–bandiert, die Matrix $\mathbf{Q}$ ist $(7,7)$–bandiert, der Aufwand zur Erstellung dieser Matrizen liegt bei $O(N)$. Dabei verwenden wir bei der Bestimmung der Matrix $\mathbf{Q}$ die Gauß-Quadratur vom Grad 1.
Für die Berechnung der Matrix $\mathbf{M}^t\mathbf{\Sigma}\mathbf{M}$ benötigen wir insgesamt etwa $5N$ Operationen für die erste und etwa $25N$ Operationen für die zweite Multiplikation. Die Berechnung der rechten Seite des Gleichungssystem $\mathbf{M}^t\mathbf{v}$ können wir ebenfalls in $O(N)$ Zeit bewältigen. Daher beträgt der Aufwand $O(N)$. Für das Gesamtverfahren erhalten wir damit ebenfalls $O(N)$.

Filter auf Wavelet-Basis
Wir starten stets mit einem Profil der Länge N und führen $M = \log_2 N$ Zerlegungen und Rekonstruktionen durch. Nach der Bemerkung aus Seite 55 beträgt der Aufwand für eine Zerlegung oder Rekonstruktion $O(N)$, falls die Eingangsfolge die Länge N hat. Nehmen wir an, es gilt $N = 2^n$ für ein $n \in \mathbb{N}$, dann benötigen wir im ersten Zerlegungs- und im letzten Rekonstruktionsschritt $N + |h|$ Operationen, wobei $|h|$ die Länge der Verfeinerungsfolge ist. Im zweiten Zerlegungs- und vorletzten Rekonstruktionsschritt sind es dann $\frac{N}{2} + |h|$ Operationen. Sukzessive erhalten wir für die Anzahl Operationen O_{Wav} die Abschätzung

$$O_{Wav} = 2\sum_{j=0}^{n} \frac{N}{2^j} + |h| \leq 2N + 2\,|h| \log_2 N.$$

Insgesamt liegt also ein $O(N)$–Algorithmus vor.

Fazit

Wir haben gesehen, dass für die klassische Rauheitsmessung die nach ISO zertifizierten Filter nach wie vor sehr gut geeignet sind. Allerdings können wir die Ergebnisse der klassischen Filterverfahren mit den Filtern auf Basis der Smoothing Splines nachbilden. Diese haben gegenüber den ISO-Filtern den Vorteil einer kürzeren Laufzeit, was insbesondere bei großen Profilen eine Rolle spielt, und nicht zuletzt handelt es sich um Filter im $\mathbb{R}^d$. In unseren Versuchen waren Werte für ζ in der Größenordnung zwischen 0.75 und 0.85 optimal. Es bleibt für eine allgemeine Anwendung festzuhalten, dass ein Verfahren benötigt wird, das den Wert für ζ automatisch vor dem Filtervorgang bestimmt.
Die auf dem Soft-Tresholding Verfahren basierenden Wavelet-Filter entfernen die feine Rauheit, größere Profilriefen werden aber als ein Teil des Sollements angesehen. Es stellt sich in diesem Zusammenhang natürlich die Frage, ob Profilriefen, die in der Größenordnung von 0,2 mm liegen, tatsächlich Rauheit darstellen oder bereits ein Teil des Sollements sind.
Verlassen wir das Gebiet der klassischen Rauheitsmessung, das sich ausschließlich mit plateauähnlichen Profilen beschäftigt, so sind neben den Smoothing Spline Filtern, die auch in diesem Fall einsetzbar sind, auch die Wavelet-Filter eine Alternative. Wobei wir bei Filtern auf Wavelet-Basis jeweils eine Einschränkung machen. Wir meinen damit Wavelet-Zerlegung und -Rekonstruktion in Verbindung mit Soft-Tresholding. Bei einem Profil,

das keinen plateauähnlichen Verlauf aufweist und womöglich noch im $\mathbb{R}^d$ für $d > 2$ gemessen wurde, liefern die klassischen Verfahren nur unzureichende Ergebnisse.
Wir können also vor allem mit dem Filterverfahren auf Basis der Smoothing Splines und zum Teil auch mit Filterverfahren auf Wavelet-Basis die Ergebnisse der ISO-Filter nachbilden. Bei einem nicht plateauähnlichen Profil können wir mit Hilfe dieser Verfahren sogar Aufwand sparen. Die übliche Vorgehensweise, eine nicht plateauähnliche Kontur zu filtern, ist sehr aufwändig. Die Kontur muss zuerst auf eine Gerade transformiert und gegebenenfalls in eine Ebene projiziert werden, anschließend wird die Kontur gefiltert und am Ende wieder in die ursprüngliche Form transformiert. Dieser Aufwand ist vor allem bei Konturen, denen kein Regelgeometrieelement zu Grunde liegt, sehr aufwändig oder sogar nur bedingt möglich.

9.6 Filter für Fasertaster-Konturen

Neben den klassischen, nach ISO zertifizierten Filtern und den in den letzten Abschnitten diskutierten Filtern auf Basis von Splines und Wavelets entstand durch die folgende Messanordnung die Notwendigkeit, ein weiteres, spezielles Filterverfahren zu entwerfen. In der Koordinatenmesstechnik unterscheidet man grundsätzlich zwischen taktilen und optischen Messverfahren.
Bei einer optischen Messung werden die Messdaten mittels einer optischen Anordnung in Verbindung mit einem CCD-Chip, ähnlich wie bei einer Digitalkamera, gewonnen. Bei einer taktilen Messung wird hingegen das zu messende Objekt durch einen Taster, in der Regel eine Kugel mit einem bestimmten Radius, abgetastet.
In den letzten Jahren entwickelte *Werth Messtechnik GmbH* ein neues Messverfahren, das sowohl optisch als auch taktil misst. Hierbei wird ein extrem kleiner Tastkopf, ein so genannter Fasertaster, mit einem Radius von $12,5\ \mu$m mittels eines Bildverarbeitungssystems erfasst. Dieses Messsystem hat eine sehr geringe Messabweichung.
Die geringe Größe des Tastkopfes und die große Aufnahmegeschwindigkeit von bis zu 50 Messpunkten pro Sekunde führen zu neuen Problemen. Der Taster bleibt bei manchen Oberflächen und dort bevorzugt am Anfang und am Ende oder bei Geschwindigkeitsveränderungen, wie zum Beispiel an Ecken und Kanten, an dem Messobjekt kleben. Dabei entstehen Anhäufungen von Punkten, die erhebliche Schwierigkeiten bei der Weiter-

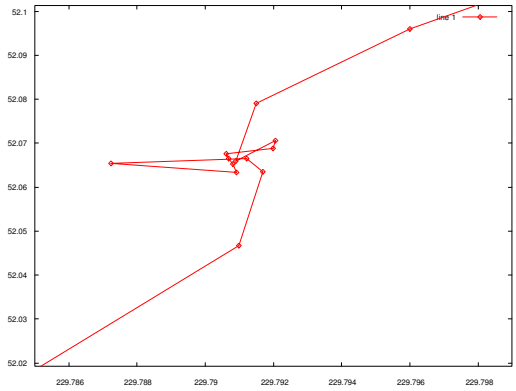

Abbildung 47: Fasertaster-Kontur.

verarbeitung der Messergebnisse zur Folge haben. Eine solche Anhäufung sehen wir in Abbildung 47.
Die üblichen klassischen oder in den vorherigen Abschnitten beschriebenen Verfahren eignen sich nicht für diese Aufgabe, weil wir die korrekt gemessenen Punkte behalten und lediglich die kritischen Stellen, an denen der Taster stehen geblieben ist, eliminieren wollen. Deshalb wird hier ein neues Filterverfahren benötigt.
Wie üblich betrachten wir eine vorliegende Kontur

$$K = \{\mathbf{v}_j \mid j = 0, \ldots, N-1\} \subset \mathbb{R}^d,$$

wobei in diesem Fall nur $d \leq 3$ sinnvoll ist. Als Hilfsmittel bemühen wir erneut das *innere Produkt* zweier Vektoren im $\mathbb{R}^d$, ähnlich wie bei der Eckenerkennung. Wir verwenden die Beziehung

$$\mathbf{v}_j^t \mathbf{v}_k = \|\mathbf{v}_j\|_2 \|\mathbf{v}_k\|_2 \cos \alpha_{jk} = \sum_{i=1}^{d} v_j^{(i)} v_k^{(i)}.$$

Dabei ist α_{jk} der von $\mathbf{v}_j$ und $\mathbf{v}_k$ eingeschlossene Winkel und $v_j^{(i)}$ die i–te Komponente von $\mathbf{v}_j$.
Betrachten wir eine durch einen Fasertaster erzeugte Kontur, die solche unerwünschte Effekte enthält, so können wir leicht erkennen, dass solche Effekte durch mehrere Punkte charakterisiert werden, deren eingeschlossene Winkel signifikant klein sind.
Ein solches Verhalten der Winkel wollen wir nutzen und durchsuchen die gegebene Kontur nach Punkten bzw. Gruppen von Punkten, deren eingeschlossene Winkel klein, das bedeutet kleiner als der lokale Durchschnitt, sind.

Wurde eine solche Gruppe von Punkten $\mathbf{v}_{k+1}, \ldots, \mathbf{v}_{k+n-1}$ identifiziert, so unterscheiden wir drei mögliche Vorgehensweisen.

(1) Wir löschen die Punkte $\mathbf{v}_{k+1}, \ldots, \mathbf{v}_{k+n-1}$ ersatzlos.

(2) Wir betrachten den letzten Punkt $\mathbf{v}_k$ vor dieser Gruppe und den ersten Punkt $\mathbf{v}_{k+n}$ nach dieser Gruppe. Wir bestimmen die Gerade durch $\mathbf{v}_k$ und $\mathbf{v}_{k+n}$ und berechnen den Mittelpunkt der Strecke $\mathbf{v}_k\mathbf{v}_{k+n}$. Wir löschen alle Punkte $\mathbf{v}_{k+1}, \ldots, \mathbf{v}_{k+n-1}$ und ersetzen diese durch den einen Punkt

$$\mathbf{v}_{neu} = \mathbf{v}_k + \frac{1}{2}(\mathbf{v}_{k+n} - \mathbf{v}_k).$$

(3) Wir bestimmen analog zu (1) einen Punkt $\mathbf{v}_{neu}$, diesmal als Schwerpunkt der Punkte $\mathbf{v}_{k+1}, \ldots, \mathbf{v}_{k+n-1}$, das bedeutet

$$\mathbf{v}_{neu} = \frac{1}{n-1} \sum_{j=1}^{n-1} \mathbf{v}_{k+j}.$$

Anschließend ersetzen wir die Punkte $\mathbf{v}_{k+1}, \ldots, \mathbf{v}_{k+n-1}$ durch $\mathbf{v}_{neu}$.

Im Folgenden bezeichnen wir diese Vorgehensweisen mit *Typ 1*, *Typ 2* und *Typ 3*.
Die ersten Versuche haben gezeigt, dass bei allen drei Typen nach einem Durchlauf des Algorithmus immer noch unerwünschte Effekte auftreten können. Dies hängt einerseits mit der Wahl der Gruppenlänge, also der Anzahl der Punkte einer Gruppe, und andererseits mit der Lage der Punkte zusammen. So kann es zum Beispiel passieren, dass die Punkte $\mathbf{v}_k$ und $\mathbf{v}_{k+n}$ nach der Bereinigung einen neuen, unerwünschten Effekt erzeugen, der in Abbildung 48 dargestellt ist. Wir sehen an diesem Beispiel, dass nach obiger Beschreibung genau die Punkte, deren y–Wert größer als 1.5 ist, gelöscht werden. Allerdings erzeugen die Punkte mit den Koordinaten $(5; 1.5)$ und $(4; 1.5)$ einen neuen unerwünschten Effekt und daher ist in diesem Fall ein weiterer Durchlauf des Algorithmus notwendig.

9.5 Algorithmus:

Eingabe:
Als Eingabe für den Algorithmus benötigen wir die zu untersuchende Kontur $K = \{\mathbf{v}_j \mid j = 0, \ldots, N-1\} \subset \mathbb{R}^d$ und den Typ der Bereinigung, gemäß der obigen Beschreibung. Wir wenden die folgenden Konturmanipulationen auf K an, bis eine der Abbruchbedingungen (3) erfüllt ist.

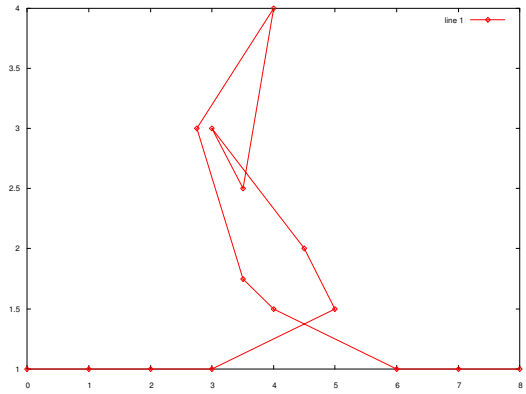

Abbildung 48: Testkontur

(1) Wir bestimmen für je drei Punkte $\mathbf{v}_j, \mathbf{v}_{j+1}, \mathbf{v}_{j+2}$ den Kosinus des eingeschlossenen Winkels. Hierfür setzen wir

$$\mathbf{z}_{j+1} = \mathbf{v}_{j+1} - \mathbf{v}_j \text{ sowie } \mathbf{u}_{j+1} = \mathbf{v}_{j+1} - \mathbf{v}_{j+2}.$$

Dann ist

$$c_{j+1} = \cos(\alpha_{j,j+1}) = \frac{\sum_{k=0}^{N-1} z_{j+1}^{(k)} u_{j+1}^{(k)}}{\|\mathbf{z}_{j+1}\|_2 \|\mathbf{u}_{j+1}\|_2}.$$

Gehört die Sequenz $(\mathbf{v}_j, \mathbf{v}_{j+1}, \mathbf{v}_{j+2})$ zu einem geraden Teilstück der Kontur, so gilt $c_{j+1} \approx -1$.
Wir erhalten eine Liste $\mathbf{C} = (c_1, \ldots, c_{N-2})$, die wir im Folgendem auswerten.

(2) Wir bestimmen den Mittelwert sowie die mittlere Streuung von $\mathbf{C}$. Es ist

$$\overline{c} = \frac{1}{N-2} \sum_{j=1}^{N-2} c_j$$

der Mittelwert und

$$\rho = \sqrt{\frac{1}{N-2} \sum_{j=1}^{N-2} (c_j - \overline{c})^2}$$

die mittlere Streuung.

(3) Wir unterscheiden folgende drei Möglichkeiten.

(i) Es gilt $\overline{c} \leq -0.98$ und $\rho \leq 0.01$. In diesem Fall brechen wir den Algorithmus ab. Dies ist der ideale Fall, denn auf Grund der Werte

von $\overline{c}$ und ρ können wir von einer glatten Kontur mit eventuell (gewollten) Features wie Ecken ausgehen.

(ii) Es treten keine Veränderungen der Kontur auf, das heißt $\overline{c}$ und ρ sind stationär. Falls dieser Zustand zum ersten Mal vorkommt, verkleinern wir die Gruppenlänge und setzen den Algorithmus fort.

(iii) Auch nach Verkleinerung der Gruppenlänge tritt Stationarität von $\overline{c}$ und ρ ein. In diesem Fall wird der Algorithmus abgebrochen. In der Praxis geben wir eine Fehlermeldung oder Warnung aus, denn es wurden eventuell nicht alle unerwünschten Effekte eliminiert.

(4) Wir erzeugen eine neue Liste $\mathbf{L}$, für die gilt

$$c_k \in \mathbf{L} \text{ genau dann, wenn } c_k \geq 3\rho - 1.$$

Das heißt $\mathbf{L}$ besteht aus Einträgen aus $\mathbf{C}$, die entweder eine Ecke sein oder zu einer Gruppe, welche einen unerwünschten Effekt bildet, gehören könnten. $\mathbf{L}$ werten wir im Folgenden aus.

(5) Wir untersuchen $\mathbf{L}$ im Hinblick auf Sequenzen von aufeinander folgenden Elementen. Hierbei gehören zwei Elemente aus $\mathbf{L}$ derselben Sequenz (Gruppe) an, falls sich ihre Indizes höchstens um 3 unterscheiden. So sind zum Beispiel die Elemente $c_{10}, c_{11}, c_{14}, c_{16}, c_{17}$ in einer Gruppe der Länge 8. Hingegen bilden c_{56}, c_{60}, c_{65} keine gemeinsame Gruppe sondern drei Gruppen der Länge 1.

(6) Im Folgenden wollen wir die Punkte eliminieren, die zu den Gruppen der Länge mindestens 3 (und bei Stationarität von $\overline{c}$ und ρ der Länge mindestens 2) gehören. Wir bestimmen den letzten Punkt vor der Gruppe und den ersten nach der Gruppe und löschen bzw. ersetzen je nach Typ die Punkte innerhalb der Gruppe.

Das Ergebnis eines Durchlaufs der Positionen (1) bis (6) ist eine neue Kontur. Mit dieser Kontur starten wir einen neuen Durchlauf, bestimmen wiederum die Liste $\mathbf{C}$ und prüfen anschließend die Abbruchbedingungen.

9.6.1 Testergebnisse

Wir untersuchen in diesem Abschnitt drei Fasertasterkonturen, zählen die Anzahl der Durchläufe mit allen drei Typen und versuchen, eine Auswertung bezüglich der Güte des Algorithmus zu erstellen. Es handelt sich um die Konturen `wfpscan1`, `wfpscan2` und `wfpscan3`.
Wir haben den obigen Algorithmus auf alle drei Konturen mit verschiedenen Typen angewandt und die Ergebnisse in der folgenden Tabelle zusammengefasst.

wfpscan1	Typ 1	Typ 1	Typ 2	Typ 2	Typ 3	Typ 3
Punkte	1208	1208	1208	1208	1208	1208
Restpunkte	551	551	651	651	617	617
Durchläufe	9	7	7	5	9	7
$\overline{c}$	-.99404	-.99404	-.98251	-.98251	-.98237	-.98248
ρ	.00199	.00199	.00604	.00604	.00655	.00652
Stationarität	10^{-12}	10^{-6}	10^{-12}	10^{-6}	10^{-12}	10^{-6}

Tabelle 1: Kontur wfpscan1

wfpscan2	Typ 1	Typ 1	Typ 2	Typ 2	Typ 3	Typ 3
Punkte	908	908	908	908	908	908
Restpunkte	766	766	771	771	795	795
Durchläufe	6	4	6	4	7	5
$\overline{c}$	-.99227	-.99227	-.99288	-.99288	-.99330	-.99330
ρ	.00378	.00378	.00374	.00374	.00363	.00363
Stationarität	10^{-12}	10^{-6}	10^{-12}	10^{-6}	10^{-12}	10^{-6}

Tabelle 2: Kontur wfpscan2

wfpscan3	Typ 1	Typ 1	Typ 2	Typ 2	Typ 3	Typ 3
Punkte	943	943	943	943	943	943
Restpunkte	816	816	818	818	860	860
Durchläufe	6	4	6	4	7	5
$\overline{c}$	-.99091	-.99091	-.99423	-.99423	-.99039	-.99039
ρ	.00319	.00319	.00207	.00207	.00279	.00279
Stationarität	10^{-12}	10^{-6}	10^{-12}	10^{-6}	10^{-12}	10^{-6}

Tabelle 3: Kontur wfpscan3

Bei allen untersuchten Konturen wurde die Gruppenlänge im Verlauf des Algorithmus reduziert, so dass wir durch die Anpassung der Gruppengröße von Anfang an die Anzahl der Durchläufe und damit die Gesamtzeit reduzieren könnten. Die momentane Durchlaufzeit liegt bei 0.015 sec bei jeweils

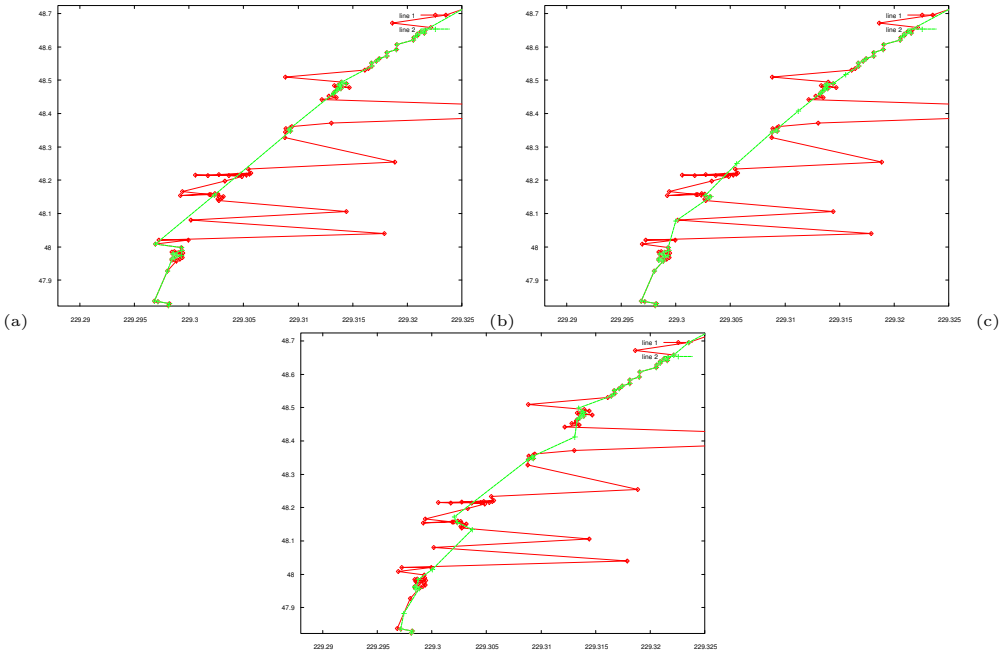

Abbildung 49: wfpscan1 (a) Typ-1-Algorithmus (b) Typ-2-Algorithmus (c) Typ-3-Algorithmus

etwa 1.000 Punkten.
Aus den obigen Tabellen können wir ebenfalls entnehmen, dass bei dem Typ 1-Algorithmus die meisten Punkte gelöscht werden. Dies ist aber nicht weiter verwunderlich, denn bei diesem Typ werden nur Punkte gelöscht und keine hinzugefügt.
Kritisch könnte die Vorgehensweise des Typ-1-Algorithmus werden, wenn eine sehr verrauschte Kontur vorliegt und wir durch das Löschen zu viele, eventuell relevante, Informationen verlieren. Hierfür betrachten wir die Kontur `wfpscan1` und die Anwendung aller drei Typen.
Wir sehen in Abbildung 49, dass während bei den Typen 2 und 3 in der Mitte des Bildes in der x–Richtung zwischen 229.305 und 229.315 Punkte hinzugefügt werden, dieser Bereich beim Typ 1 fast ohne Punkte und demzufolge ohne Informationen ist.
Andererseits benötigen wir bei dem Typ 1-Algorithmus keine Berechnung der Antastvektoren, da wir die vorhandenen Werte verwenden können. Bei den Typ-2- und Typ-3-Algorithmen fügen wir hingegen an Stelle einer gelöschten Gruppe einen neuen Punkt hinzu. In diesem Fall bestimmen wir einen neuen Antastvektor als Mittelwert der Antastvektoren des letzten Punktes vor der Gruppe und des ersten Punktes nach der Gruppe.
Bevor wir den Abschnitt über Filterverfahren abschließen, gehen wir noch

auf die Laufzeit von Algorithmus 9.5 ein. Ein Durchlauf hat offensichtlich die Komplexität $O(N)$, da wir die Liste $\mathbf{C}$ erstellen und anschließend die Werte der Liste verarbeiten. Wie wir in den Tabellen 1 bis 3 gesehen haben, gilt für die Anzahl der Iterationen $maxiter$ stets $maxiter \ll N$, also können wir von einer Gesamtkomplexität von $O(N)$ ausgehen. Die Berechnung des Speicherplatzbedarfs ergibt ebenfalls $O(N)$, da wir lediglich die Einträge von $\mathbf{z}$, $\mathbf{u}$ sowie $\mathbf{C}$ und $\mathbf{L}$ speichern müssen. Für die neu berechnete Kontur können wir den Speicherplatz der ursprünglichen Kontur verwenden.

10 Behandlung von Flächen

10.1 Approximation von Flächen durch radiale Basisfunktionen

Neben den klassischen Methoden, wie Interpolation oder Approximation von Flächen durch multivariate Splines oder diversen Subdivision-Verfahren, sind vor allem durch die Arbeiten von *Rick Beatson* die radialen Basisfunktionen (RBF) als weitere Methode, Flächen zu interpolieren oder zu approximieren, bekannt geworden.
Wir diskutieren in diesem Abschnitt zwei Möglichkeiten der Glättung von Flächen mit Hilfe radialer Basisfunktionen. Wir stellen als Erstes ein globales Verfahren zur Glättung einer Fläche vor, also ein Verfahren, welches die gesamte Fläche auf einmal behandelt und besprechen sodann dessen Vor- und Nachteile. Einige dieser Nachteile, zum Beispiel die hohe Komplexität, können wir mit geeigneten Modifikationen sofort beheben. Im Anschluss diskutieren wir ein lokales Verfahren basierend auf radialen Basisfunktionen für triangulierte Flächen.
Die Grundlagen zu diesem Abschnitt, die radialen Basisfunktionen, haben wir bereits in Kapitel 5 eingeführt.
Sei für den Rest des Abschnittes 10.1

$$K = \{\mathbf{v}_j \mid j = 0, \ldots, N-1\} \subset \mathbb{R}^d$$

die zu glättende Fläche. Die nachfolgenden Aussagen gelten für ein allgemeines $d > 2$, in der Praxis finden wir aber stets die Situation $d = 3$ vor. Wir schreiben

$$\mathbf{v}_j = \begin{bmatrix} v_j^{(1)} \\ \vdots \\ v_j^{(d)} \end{bmatrix} \in \mathbb{R}^d \text{ und } \mathbf{v}'_j = \begin{bmatrix} v_j^{(1)} \\ \vdots \\ v_j^{(d-1)} \end{bmatrix} \in \mathbb{R}^{d-1}.$$

Sei

$$s = P + \sum_{j=0}^{N-1} \lambda^{(j)} \phi(\| \cdot - \mathbf{q}_j \|_2) \tag{10.1}$$

eine radiale Basisfunktion. Wie in Kapitel 5 sind $\mathbf{q}_0, \ldots, \mathbf{q}_{N-1}$ die paarweise verschiedenen *Zentren* und die Parameter $\lambda^{(0)}, \ldots, \lambda^{(N-1)}$ die *Gewichte* der radialen Basisfunktion. Die Funktion ϕ wird häufig als *Basisfunktion* bezeichnet, P ist ein Polynom aus dem Raum $\Pi_{m,d}$.
Dabei ist

$$\phi : \mathbb{R}^{d-1} \to \mathbb{R}$$

nicht notwendig radial.
In unserem Fall suchen wir zu einer Fläche K eine Menge $\{\lambda^{(j)} \mid j = 0, \ldots, N-1\}$ von Gewichten, welche die Gleichung

$$v_k^{(d)} = s(\mathbf{v}'_k) = \mathbf{p}(\mathbf{v}'_k) + \sum_{j=0}^{N-1} \lambda^{(j)} \phi(\|\mathbf{v}'_k - \mathbf{v}'_j\|_2) \text{ für } k = 0, \ldots, N-1 \quad (10.2)$$

erfüllen. In Matrixschreibweise führt (10.2) zu dem linearen Gleichungssystem

$$\left[\begin{array}{c|c} \mathbf{A} & \mathbf{B} \\ \hline \mathbf{B}^t & \mathbf{0} \end{array}\right] \left[\begin{array}{c} \boldsymbol{\lambda} \\ \hline \boldsymbol{\mu} \end{array}\right] = \left[\begin{array}{c} \mathbf{v}^{(d)} \\ \hline \mathbf{0} \end{array}\right]. \quad (10.3)$$

Dabei gilt

$\mathbf{A} = (a_{ij}) \in \mathbb{R}^{N \times N}$ mit $a_{ij} = \phi(\|\mathbf{v}'_i - \mathbf{v}'_j\|_2)$,
$\mathbf{B} = (b_{ij}) \in \mathbb{R}^{N \times K}$ mit den Zeilen

$$\mathbf{b}_i = \left[\begin{array}{cccc} b_{i0} & b_{i1} & \ldots & b_{i,K-1} \end{array}\right] = \left[\begin{array}{ccccc} 1 & v_i^{(1)} & \ldots & v_i^{(d-1)} & \ldots & \left(v_i^{(d-1)}\right)^m \end{array}\right]$$

für $i = 0, \ldots, N-1$,

$$\boldsymbol{\lambda} = \left[\begin{array}{c} \lambda^{(0)} \\ \vdots \\ \lambda^{(N-1)} \end{array}\right] \in \mathbb{R}^N \text{ und}$$

$$\boldsymbol{\mu} = \left[\begin{array}{c} \mu^{(0)} \\ \vdots \\ \mu^{(K-1)} \end{array}\right] \in \mathbb{R}^K,$$

so dass

$$P(\mathbf{x}) = \sum_{\boldsymbol{\beta} \in \Gamma_m} \mu^{(\boldsymbol{\beta})} \mathbf{x}^{\boldsymbol{\beta}} \in \Pi_{m,d}. \quad (10.4)$$

Um die obige Darstellung von $\boldsymbol{\mu}$ als Vektor erhalten zu können, indizieren wir die Einträge von $\boldsymbol{\mu}$ in (10.4) von 0 bis $K-1$ neu.
Weiter ist $\mathbf{0}$ in (10.3) je nach Dimension entweder eine Nullmatrix oder ein Nullvektor.
Bei den hier vorgestellten radialen Basisfunktionen reicht stets $m = 1$ aus, also gilt

$$P(\mathbf{x}) = \mu^{(0)} + \sum_{j=0}^{d-1} \mu^{(j)} x^{(j)}.$$

Wir erhalten daher $K = d$. Es eignen sich natürlich nicht alle radialen Basisfunktionen zur Interpolation oder Approximation von Flächen. Die folgende Tabelle zeigt uns die Funktionen ϕ, die meistens für solche Aufgaben verwendet werden.

RBF	ϕ	P	$\mathbf{A}$	$\mathbf{B}$
Biharmonischer Spline	$\phi(r) = r$	$p \in \Pi_{1,d-1}$	$\mathbb{R}^{N\times N}$	$\mathbb{R}^{N\times d}$
Triharmonischer Spline	$\phi(r) = r^3$	–	$\mathbb{R}^{N\times N}$	–
Thin-Plate Spline	$\phi(r) = r^2\log(r)$	$p \in \Pi_{1,d-1}$	$\mathbb{R}^{N\times N}$	$\mathbb{R}^{N\times d}$
Multiquadrik	$\phi(r) = (r^2 + c^2)^{1/2}$	$p \in \Pi_{1,d-1}$	$\mathbb{R}^{N\times N}$	$\mathbb{R}^{N\times d}$
Inverse Multiquadrik	$\phi(r) = (r^2 + c^2)^{-1/2}$	–	$\mathbb{R}^{N\times N}$	–
Gauß-Funktion	$\phi(r) = e^{-\frac{r^2}{2\sigma^2}}$, $\sigma > 0$	–	$\mathbb{R}^{N\times N}$	–
Wendland-RDF	$\phi(r) = q(r)_+(4r + 1)$	–	$\mathbb{R}^{N\times N}$	–

Wir konzentrieren uns im Folgenden auf die Thin-Plate Splines und dabei insbesondere auf die Berechnung mit einem zusätzlichen polynomialen Anteil. Diese Funktion, die wir mit ϕ_{TPS} bezeichnen, ist in ihrer Form als radiale Basisfunktion zu gegebenen Daten $K \subset \mathbb{R}^d$ die Lösung der Gleichung

$$\phi_{TPS} = \min_{f\in\mathcal{C}(\mathbb{R}^{d-1})} J_\zeta(f) \tag{10.5}$$

$$\text{mit } J_\zeta(f) = \frac{1}{N}\sum_{j=0}^{N-1}\left(f(\mathbf{v}'_j) - v_j^{(d)}\right)^2 + \zeta\int_{\mathbb{R}^{d-1}}(\triangle f(\mathbf{x}))^2\,d\mathbf{x}.$$

Dabei ist $\triangle f$ der Laplace-Operator von f beschrieben in Kapitel 5. Der Least-Square Anteil der Gleichung sorgt für die Nähe der Funktion zu den Daten, der integrale Anteil steuert die Glattheit der Fläche.
Gleichung (10.3) führt allerdings noch nicht zu der gewünschten geglätteten Fläche, diese Gleichung entspricht der Lösung von (10.5) für $\zeta \to 0$, also der Interpolation der Daten. Eine Glättung der Fläche erreichen wir für $\zeta > 0$ und damit als Lösung des nachfolgenden linearen Gleichungssystems

$$\mathbf{Mf} = \left[\begin{array}{c|c} \mathbf{A} + \zeta N\mathbf{I}_N & \mathbf{B} \\ \hline \mathbf{B}^t & \mathbf{0} \end{array}\right]\left[\begin{array}{c} \boldsymbol{\lambda} \\ \hline \boldsymbol{\mu} \end{array}\right] = \left[\begin{array}{c} \mathbf{v}^{(d)} \\ \hline \mathbf{0} \end{array}\right] = \mathbf{b}. \tag{10.6}$$

Da eine Interpolation der Daten offensichtlich für $\zeta \to 0$ erreicht wird, beschäftigen wir uns fortan mit dem Gleichungsystem (10.6). Dabei fassen wir zur Vereinfachung ζN zu einem Parameter ζ zusammen.

Der große Nachteil der Ansätze mit radialen Basisfunktionen ist, dass die Matrix $\mathbf{M}$ sowohl bei radialen Basisfunktionen mit polynomialem Anteil als auch ohne diesen nicht nur recht groß wird (die Dimension beträgt $(N + d + 1) \times (N + d + 1)$), sondern unglücklicherweise auch noch dicht besetzt ist.
Um diese Nachteile schon im Entwurf des Algorithmus zu reduzieren, werden wir nicht alle Punkte der Fläche zur Berechnung zulassen, sondern nur einige wenige Punkte, die auf eine noch festzustellende Art repräsentativ sind. Entspricht die Fläche den festgelegten Kriterien, so können wir diese behalten, ansonsten müssen wir an Stellen, die zu sehr abweichen, mehr Punkte zur Berechnung zulassen und erneut die glättende Fläche bestimmen. Wir erhalten das folgende Verfahren.

10.1 Algorithmus: Sei $K \subset \mathbb{R}^d$, $d \geq 3$ eine Fläche. Seien weiter $\zeta \geq 0$ ein Parameter, den wir als *Glättungsfaktor* bezeichnen wollen, und $\varepsilon > 0$ ein Schwellenwert für die Genauigkeit der Glättung. Wir setzen

$$v_{min}^{(j)} = \min\{v_l^{(j)} \mid l = 0, \ldots, N-1\},$$
$$v_{max}^{(j)} = \max\{v_l^{(j)} \mid l = 0, \ldots, N-1\}$$

für $j = 1, \ldots, d$. Dann ist

$$W = [v_{min}^{(1)}, v_{max}^{(1)}] \times \cdots \times [v_{min}^{(d)}, v_{max}^{(d)}] \subset \mathbb{R}^d$$

ein Quader im $\mathbb{R}^d$. In der Literatur wird W häufig als *Bounding-Box* bezeichnet. Wir teilen jedes Intervall $[v_{min}^{(j)}, v_{max}^{(j)}]$ für $j = 1, \ldots, d$ in t Teilintervalle und erhalten insgesamt $T = t^d$ Teilquader, für die gilt

$$W = \bigcup_{k=1}^{T} W_k.$$

Sei im Folgenden $W_k^\circ = \{\mathbf{v}_l \in K \mid \mathbf{v}_l \in W_k\}$ für $k = 1, \ldots, T$. Es ist offensichtlich, dass in der Regel einige Teilquader W_k existieren werden, für die $W_k^\circ = \emptyset$ gilt.
Für jeden Teilquader W_k mit $W_k^\circ \neq \emptyset$ bestimmen wir einen ausgezeichneten Punkt $\mathbf{v}_{k*}$. Es gibt viele Möglichkeiten, einen solchen Punkt zu bestimmen, wir verwenden den Schwerpunkt der Konturpunkte eines jeden Teilquaders

$$\mathbf{v}_{k*} = \frac{1}{|W_k^\circ|} \sum_{\mathbf{v}_l \in W_k^\circ} \mathbf{v}_l \text{ für } k = 1, \ldots, T.$$

Dann liefert uns das folgende Verfahren eine geglättete Fläche zu den Daten aus K.

(1) Wir ermitteln die Teilquader W_k für $k = 1, \ldots, T_0$ und bestimmen $\mathbf{v}_{k*} \in W_k$ für $k = 1, \ldots, T_0$. Wir bezeichnen mit $\mathbf{v}_k^+ = \mathbf{v}'_{k*}$ den Vektor der ersten $d-1$ Komponenten von $\mathbf{v}_{k*}$. Sei

$$TPS_0(W_k) = \{\mathbf{v}_l \in W_k \mid \mathbf{v}_l \text{ wird zur Berechnung von } s \text{ benutzt}\}$$

für $k = 1, \ldots, T_0$. Damit gilt im ersten Schritt $TPS_0(W_k) = \emptyset$ für W_k mit $|W_k^\circ| > 1$. Wir schreiben

$$V_0^* = \{\mathbf{v}_{k*} \mid k = 1, \ldots, T_0\}.$$

(2) Wir führen die folgenden Schritte $\beta = 1, 2, \ldots$ solange durch, bis eine der Abbruchbedingungen in (c) erfüllt ist.

(a) Gemäß der Gleichung (10.6) bestimmen wir Gewichte $\lambda^{(1)}, \ldots, \lambda^{(T_\beta)}$ und erhalten den Thin-Plate Spline

$$s_\beta(\mathbf{x}) = P(\mathbf{x}) + \sum_{k=1}^{T_\beta} \lambda^{(k)} \phi_{TPS}(\|\mathbf{x} - \mathbf{v}_k^+\|_2).$$

(b) Jetzt sind wir in der Lage, die Fläche an allen Punkten aus K auszuwerten. Hierfür berechnen wir $s_\beta(\mathbf{v}'_j)$ für $j = 0, \ldots, N-1$.

(c) Nun untersuchen wir die Abweichung der gefilterten Flächen zu den tatsächlichen Daten

$$d_j = \left| s_\beta(\mathbf{v}'_j) - v_j^{(d)} \right| \text{ für } j = 0, \ldots, N-1.$$

Gilt $d_j \leq \varepsilon$ für alle $j = 0, \ldots, N-1$, so ist die gefilterte Fläche nah genug an den Daten und wir können den Algorithmus beenden. Anderenfalls unterscheiden wir verschiedene Fälle. Sei hierfür $d_j > \varepsilon$ für ein $j \in \{1, \ldots, N-1\}$.

(i) Ist $\left|W_j^\circ\right| - |TSP_\beta(W_j)| > 0$, wählen wir $\mathbf{v}_k \in W_j \backslash TPS_\beta(W_j)$ mit

$$\|\mathbf{v}_k - \mathbf{v}_{j*}\|_2 = \max_{\mathbf{v}_l \in W_j \backslash TPS_\beta(W_j)} \|\mathbf{v}_l - \mathbf{v}_{j*}\|_2.$$

Dabei ist $\mathbf{v}_j^*$ stets der Schwerpunkt des Teilquaders W_j. Zum Abschluss bilden wir die neue Menge $TPS_{\beta+1}(W_j)$ als

$$TPS_{\beta+1}(W_j) = TSP_\beta(W_j) \cup \{\mathbf{v}_k\}.$$

(ii) Es ist bereits $\left|W_j^\circ\right| - |TSP_\beta(W_j)| = 0$. Also haben wir innerhalb des Teilquaders W_j alle Datenpunkte zur Berechnung hinzugezogen, die gewünschte Genauigkeit aber dennoch nicht erreicht. Wir ermitteln die benachbarten Teilquader $W_{j_1}, \ldots, W_{j_i}$ mit $i \in \mathbb{N}$ und wenden, sofern dies möglich ist, die Vorgehensweise aus (i) auf diese Teilquader an. Die Anzahl und die Bedeutung benachbarter Quader behandeln wir in Abschnitt 10.2.2.
Falls $\left|W_{j_i}^\circ\right| - |TSP_\beta(W_{j_i})| = 0$ für alle $i \in \mathbb{N}$ gilt, wird dieser Wert nicht korrigiert. Eine geringere Abweichung erreichen wir an dieser Stelle durch eine Korrektur des Glättungsfaktors ζ, die zum Beispiel in [Wah90] behandelt wird.

(iii) Werden bereits alle Punkte der Kontur zur Berechnung herangezogen, ohne die nötige Genauigkeit zu erreichen, müssen wir den Glättungsfaktor ζ anpassen.

(d) Falls das Verfahren nicht beendet wurde, bestimmen wir die Menge $V_{\beta+1}^*$ neu. Es gilt

$$V_{\beta+1}^* = \bigcup_{k=1}^{T_\beta} TPS_{\beta+1}(W_k) \text{ sowie } T_{\beta+1} = \left|V_{\beta+1}^*\right|.$$

Wir indizieren die Elemente von $V_{\beta+1}^*$ neu und erhalten

$$V_{\beta+1}^* = \{\mathbf{v}_{1*}, \ldots, \mathbf{v}_{T_{\beta+1}*}\}.$$

(3) Als Ergebnis bekommen wir für ein $\beta > 0$ eine Fläche

$$s = P + \sum_{k=1}^{T_\beta} \lambda^{(k)} \phi_{TPS}(\| \cdot - \mathbf{v}_k^+ \|_2).$$

Verwenden wir Standardverfahren zur Bestimmung der Gewichte $\lambda^{(j)}$, zeigt sich der große Nachteil der Ansätze mit radialen Basisfunktionen. Liegen uns Datenmengen der Größenordnung 20.000 und mehr vor, wird der Algorithmus 10.1 sehr schnell ineffizient.
Es gibt für diese Ineffizienz zwei Gründe. Zum einen müssen wir in jeder Iteration das $(T_\beta + d + 1) \times (T_\beta + d + 1)$ Gleichungssystem (10.6) lösen. Dabei ist die Matrix $\mathbf{M}$ zwar symmetrisch aber dicht besetzt und meist auch schlecht konditioniert. Eine direkte Lösung des Gleichungssystems hätte

daher eine Laufzeit von $O((T_\beta + d + 1)^3)$ in jedem Schritt zur Folge. Zum anderen ist bereits die Auswertung des Thin-Plate Splines sehr aufwändig, da wir in diesem Fall das Matrix-Vektor-Produkt **Mb** berechnen und dabei von einer quadratischen Komplexität ausgehen müssen.
Wir stellen nachfolgend zwei Verfahren vor, welche die Lösung des linearen Gleichungssystems (10.6) und die Berechnung des Matrix-Vektor-Produkts in Algorithmus 10.1 beschleunigen.
Wir führen als Erstes die sogenannte Fast-Multipole Methode ein, die uns eine Beschleunigung der Berechnung eines Matrix-Vektor-Produkts bis zu einem Aufwand von $O(N \log N)$ erlaubt. Danach beschäftigen wir uns mit der Lösung des linearen Gleichungssystems (10.6) und diskutieren ein iteratives Verfahren auf Basis der Methode der konjugierten Gradienten für dicht besetzte Matrizen.

10.2 Fast-Multipole Methode

In vielen Anwendungen benötigt man die Berechnung eines Matrix-Vektor-Produkts, also die Auswertung der Summen

$$h^{(i)} = \sum_{j=0}^{N-1} \phi_{ij} q^{(j)} \text{ für } i = 0, \ldots, N-1 \tag{10.7}$$

für eine Matrix

$$\mathbf{\Phi} = (\phi_{ij})_{i,j=0}^{N-1} \in \mathbb{R}^{N \times N}$$

und Vektoren

$$\mathbf{h} = \begin{bmatrix} h^{(0)} \\ \vdots \\ h^{(N-1)} \end{bmatrix} \in \mathbb{R}^N \text{ sowie } \mathbf{q} = \begin{bmatrix} q^{(0)} \\ \vdots \\ q^{(N-1)} \end{bmatrix} \in \mathbb{R}^N.$$

Existieren $\mathbf{x}_j, \mathbf{x}_i \in \mathbb{R}^N$, so dass für die Einträge ϕ_{ij} gilt

$$\phi_{ij} = \phi(\mathbf{x}_j, \mathbf{x}_i) \text{ für } i, j = 0, \ldots, N-1$$

und damit

$$h^{(i)} = h(\mathbf{x}_i) = \sum_{j=0}^{N-1} \phi(\mathbf{x}_j, \mathbf{x}_i) q^{(j)} \text{ für } i = 0, \ldots, N-1$$

ist, so können wir die sogenannte *Fast-Multipole Methode* (FMM) anwenden und die Berechnung beschleunigen. Im Idealfall ist eine Komplexität

von $O(N \log N)$ erreichbar.
Die direkte Berechnung von (10.7) benötigt $O(N^2)$ Operationen, weil jeder Punkt mit allen anderen Punkten multipliziert wird, ohne dabei ihre Lage zu berücksichtigen. Die Idee der Fast-Multipole Methode ist, die vorliegenden Punkte in nahe und entfernte Punkte einzuteilen. Dann bestimmen wir das Produkt exakt für die Punkte, die nah liegen, und fassen die weit entfernten Punkte zu einem Punkt zusammen. Die Voraussetzung für eine solche Vorgehensweise ist, dass die Punkte entweder getrennt vorliegen oder dass wir eine vernünftige Trennung finden können.
Wir betrachten im Folgenden stets die Matrix

$$\mathbf{\Phi} \in \mathbb{R}^{N \times N}$$

sowie $\mathbf{h}, \mathbf{q} \in \mathbb{R}^N$ und das Produkt

$$\mathbf{h} = \mathbf{\Phi q}.$$

Hierbei gilt $\mathbf{\Phi} = (\phi_{ij})_{i,j=0}^{N-1}$ und $\phi_{ij} : \mathbb{R}^N \times \mathbb{R}^N \to \mathbb{R}$ mit $\phi_{ij} = \phi(\mathbf{y}_j, \mathbf{x}_i)$ für $\mathbf{y}_j, \mathbf{x}_i \in \mathbb{R}^d$, $d \geq 1$. Wir bezeichnen die Punkte $\mathbf{x}_0, \ldots, \mathbf{x}_{N-1} \in \mathbb{R}^d$ als *Quellpunkte*, $\mathbf{y}_0, \ldots, \mathbf{y}_{N-1} \in \mathbb{R}^d$ als *Zielpunkte* und $q^{(0)}, \ldots, q^{(N-1)} \in \mathbb{R}$ als *Quellengewichte*.
Damit bekommen wir eine Verbindung zu den *Potentialen* (Feldern), die aus der Physik bekannt sind. Diese haben die Gestalt

$$h(\mathbf{y}) = \sum_{i=0}^{N-1} q^{(i)} \phi(\mathbf{y}, \mathbf{x}_i) \text{ für } \mathbf{y} \in \mathbb{R}^d.$$

Wir unterscheiden zwischen skalaren und vektorwertigen Feldern. Diese können entweder *regulär* oder *singulär* sein. Einige Beispiele von Potentialen finden wir in der Natur, wie etwa die folgenden.

(a) *Gravitation:* Ein skalares Feld mit

$$\phi(\mathbf{y}, \mathbf{x}_i) = \frac{1}{\|\mathbf{y} - \mathbf{x}_i\|_2}.$$

ϕ ist singulär für $\mathbf{y} = \mathbf{x}_i$.

(b) *Gauß-Feld:* Ein skalares Feld mit

$$\phi(\mathbf{y}, \mathbf{x}_i) = e^{\frac{-\|\mathbf{y} - \mathbf{x}_i\|_2}{\sigma}}.$$

Hier hat ϕ keine Singularitäten, ist also überall regulär.

(c) Bei dem *3D Geschwindigkeitsfeld* handelt es sich um ein vektorwertiges Feld. Es gilt

$$\phi(\mathbf{y}, \mathbf{x}_i) = \nabla_{\mathbf{y}} \frac{1}{\|\mathbf{y} - \mathbf{x}_i\|_2} \in \mathbb{R}^3.$$

ϕ ist singulär für $\mathbf{y} = \mathbf{x}_i$.

Wir verwenden folgende Sprechweise. Besitzt ein Potential ϕ eine oder mehrere Singularitäten, so sprechen wir von einem *singulären Potential*, anderenfalls nennen wir ϕ *regulär*.
Wir wollen uns im Wesentlichen mit skalaren Feldern beschäftigen. Als Nächstes betrachten wir die *Middleman Methode* und erhalten eine Zerlegung des Potentials, zunächst ohne auf die Lage der Punkte einzugehen.

10.2.1 Die Middleman Methode

Seien $\mathbf{x}_i, \mathbf{y}_j \in \Omega \subset \mathbb{R}^d$, dann können wir ϕ als

$$\begin{aligned}\phi(\mathbf{y}_j, \mathbf{x}_i) &= \sum_{k=0}^{\infty} a_k(\mathbf{x}_i, \mathbf{x}_*) f_k(\mathbf{y}_j - \mathbf{x}_*) \\ &= \sum_{k=0}^{p-1} a_k(\mathbf{x}_i, \mathbf{x}_*) f_k(\mathbf{y}_j - \mathbf{x}_*) + Fehler(p, \mathbf{y}_j, \mathbf{x}_i) \qquad (10.8)\end{aligned}$$

darstellen. Wir bezeichnen $a_k(\cdot, \cdot)$ als *Entwicklungskoeffizient*, $f_k(\cdot)$ als *Basisfunktion*, $\mathbf{x}_*$ als *Entwicklungszentrum* und p als *Abbruchzahl*.
Mit Hilfe einer solchen Entwicklung erhalten wir die Zerlegung

$$\begin{aligned}h^{(j)} &= \sum_{i=0}^{N-1} \phi(\mathbf{y}_j, \mathbf{x}_i) q^{(i)} \\ &= \sum_{i=0}^{N-1} \left(\sum_{k=0}^{p-1} a_k(\mathbf{x}_i, \mathbf{x}_*) f_k(\mathbf{y}_j - \mathbf{x}_*) + Fehler(p, \mathbf{y}_j, \mathbf{x}_i) \right) q^{(i)} \qquad (10.9) \\ &= \sum_{k=0}^{p-1} f_k(\mathbf{y}_j - \mathbf{x}_*) \sum_{i=0}^{N-1} a_k(\mathbf{x}_i, \mathbf{x}_*) q^{(i)} + \sum_{i=0}^{N-1} Fehler(p, \mathbf{y}_j, \mathbf{x}_i) q^{(i)} \\ &= \sum_{k=0}^{p-1} c_k f_k(\mathbf{y}_j - \mathbf{x}_*) + Fehler(N, p)\end{aligned}$$

mit

$$c_k = \sum_{i=0}^{N-1} a_k(\mathbf{x}_i, \mathbf{x}_*) q^{(i)}.$$

Damit können wir den folgenden Algorithmus formulieren.

10.2 Algorithmus: Seien $\mathbf{x}_0, \ldots, \mathbf{x}_{N-1} \in \mathbb{R}^d$. Der Algorithmus errechnet näherungsweise das Matrix-Vektor-Produkt $\mathbf{\Phi q} = \mathbf{h}$ mit $\phi_{ij} = \phi(\mathbf{y}_j, \mathbf{x}_i)$.

```
for  k = 0, ..., p − 1
   c_k = 0 ;
  for  i = 0, ..., N − 1
     c_k = c_k + a_k(x_i, x_*) q^(i) ;
  end
end

for  j = 0, ..., N − 1
   h^(j) = 0 ;
  for  k = 0, ..., p − 1
     h^(j) = h^(j) + c_k f_k(y_j − x_*) ;
  end
end
```

Die Komplexität des Algorithmus beträgt $O(pN)$. Ist $p \ll N$, erhalten wir eine deutliche Verbesserung der Gesamtkomplexität.
Welche Möglichkeiten haben wir, die Zerlegung (10.9) zu realisieren? Die ersten Möglichkeiten für die Zerlegung von ϕ bieten die Potenz- und die Taylorreihe. Da wir uns im Wesentlichen mit mehrdimensionalen Problemen beschäftigen, betrachten wir die Taylorreihe.
Liegt ein eindimensionales Problem vor, so erhalten wir sofort für $y, x_i, x_* \in \mathbb{R}$ und eine analytische Funktion ϕ

$$\begin{aligned}\phi(y, x_i) &= \sum_{k=0}^{\infty} \underbrace{\frac{1}{k!}\frac{d^k\phi}{dx}(x_*, x_i)}_{=a_k(x_i,x_*)} \underbrace{(y - x_*)^k}_{=f_k(y-x_*)} \\ &= \sum_{k=0}^{\infty} a_k(x_i, x_*) f_k(y - x_*) \\ &= \sum_{k=0}^{p-1} a_k(x_i, x_*) f_k(y - x_*) + Fehler(p). \end{aligned} \tag{10.10}$$

Konvergenz ist dann für alle y mit $|y - x_*| \leq q$ für $0 \leq q \leq r_*$ gewährleistet, wobei $r_* > 0$ der Konvergenzradius ist.
Für mehrdimensionale Probleme ist die Zerlegung in der Regel nicht sofort

zu sehen. Sei hierfür $f : \mathbb{R}^d \to \mathbb{R}$ eine analytische Funktion, dann gilt

$$f(\mathbf{y}) = f(\mathbf{x}_*) + \sum_{k=1}^{\infty} \frac{1}{k!} \left((\mathbf{y} - \mathbf{x}_*)^t \nabla\right)^k f(\mathbf{x}_*).$$

Dabei ist $\mathbf{a}^t\mathbf{b} \in \mathbb{R}$ das Skalarprodukt der Vektoren $\mathbf{a}, \mathbf{b} \in \mathbb{R}^d$ und mit $\mathbf{a}^k = \mathbf{a} \otimes \cdots \otimes \mathbf{a}$ wird das k–fache Kronecker-Produkt bezeichnet, definiert in Kapitel 2. Damit erhalten wir unter Verwendung der Notation aus Kapitel 2

$$\begin{aligned}(\mathbf{a}^t\mathbf{b})^k = \left(\mathbf{a}^k\right)^t \cdot \mathbf{b}^k &= \sum_{\boldsymbol{\beta} \in \boldsymbol{\Gamma}_k} \binom{|\boldsymbol{\beta}|}{\boldsymbol{\beta}} \mathbf{a}^{\boldsymbol{\beta}} \mathbf{b}^{\boldsymbol{\beta}} \\ &= \sum_{i_1 + \ldots + i_d = k} \binom{k}{i_1 \ldots i_d} a_1^{i_1} \cdots a_d^{i_d} b_1^{i_1} \cdots b_d^{i_d}.\end{aligned} \tag{10.11}$$

Dabei ist $\boldsymbol{\Gamma}_k = \{\boldsymbol{\beta} \in \mathbb{N}_0^d \mid |\boldsymbol{\beta}| = k\}$. In unserem Fall erhält man die abgebrochene Taylorreihenentwicklung

$$\phi(\mathbf{y}, \mathbf{x}_i) = \sum_{k=0}^{p-1} \frac{1}{k!} \left[(\mathbf{y} - \mathbf{x}_*)^t \nabla_{\mathbf{x}_*}\right]^k \phi(\mathbf{x}_*, \mathbf{x}_i) + Fehler(p, N). \tag{10.12}$$

Hierbei bedeutet $\nabla_{\mathbf{x}_*}$, dass nach der Komponente $\mathbf{x}_*$ von ϕ abgeleitet wird, während die Komponente $\mathbf{x}_i$ konstant bleibt.
Wir verdeutlichen den Sachverhalt anhand des folgenden Beispiels.

10.3 Beispiel: Wir betrachten $\phi(\mathbf{y}, \mathbf{x}_i) = e^{\mathbf{y}^t \mathbf{x}_i}$, dann gilt mit Hilfe von (10.12)

$$\phi(\mathbf{y}, \mathbf{x}_i) = e^{\mathbf{x}_*^t \mathbf{x}_i} \sum_{k=0}^{p-1} \frac{1}{k!} \left[(\mathbf{y} - \mathbf{x}_*)^t \mathbf{x}_i\right]^k + Fehler(p).$$

Mit (10.11) erhalten wir

$$\phi(\mathbf{y}, \mathbf{x}_i) = e^{\mathbf{x}_*^t \mathbf{x}_i} \sum_{k=0}^{p-1} \frac{1}{k!} \left((\mathbf{y} - \mathbf{x}_*)^k\right)^t \mathbf{x}_i^k + Fehler(p),$$

und setzen

$$\begin{aligned} a_k(\mathbf{x}_i, \mathbf{x}_*) &= \frac{1}{k!} e^{\mathbf{x}_*^t \mathbf{x}_i} \mathbf{x}_i^k \text{ und} \\ f_k(\mathbf{y} - \mathbf{x}_*) &= (\mathbf{y} - \mathbf{x}_*)^k. \end{aligned}$$

Allerdings sind nicht alle Funktionen so leicht zu handhaben. Dies erkennt man an der Basisfunktion des Thin-Plate Splines

$$\phi_{TPS}(\mathbf{y}_j, \mathbf{x}_i) = \|\mathbf{y}_j - \mathbf{x}_i\|_2^2 \log(\|\mathbf{y}_j - \mathbf{x}_i\|_2).$$

Durch partielle Ableitungen erhalten wir nicht die erforderliche Struktur, mit der wir die Zerlegung (10.9) realisieren können. Daher untersuchen wir weitere Möglichkeiten der Zerlegung.

10.4 Definition: Seien $\mathbf{x}_* \in \mathbb{R}^d$ und $r_* > 0$. Wir nennen die Entwicklung

$$\phi(\mathbf{y}, \mathbf{x}_i) = \sum_{k=0}^{\infty} a_k(\mathbf{x}_i, \mathbf{x}_*) R_k(\mathbf{y} - \mathbf{x}_*) \tag{10.13}$$

regulär oder *lokal* in der Umgebung

$$U_{r_*}(\mathbf{x}_*) = \{\mathbf{y} \in \mathbb{R}^d \mid \|\mathbf{y} - \mathbf{x}_*\|_2 < r_*\},$$

falls die Reihe (10.13) für alle $\mathbf{y} \in U_{r_*}(\mathbf{x}_*)$ konvergiert.
Hierbei sind $a_k(\cdot, \cdot)$ die *Entwicklungskoeffizienten*, $R_k(\cdot)$ die *regulären Basisfunktionen* und $\mathbf{x}_*$ bezeichnen wir als *Entwicklungszentrum*. Die Entwicklung (10.13) heißt *R-Entwicklung*.

Im Falle eines regulären Potentials gibt es für die Konvergenzumgebung $U_{r_*}(\mathbf{x}_*)$ keine Einschränkungen, da das Potential überall existiert. Bei einem singulären Potential darf $U_{r_*}(\mathbf{x}_*)$ den singulären Punkt nicht enthalten.
Wir verdeutlichen den Sachverhalt an einem Beispiel.

10.5 Beispiel:

(1) *R-Entwicklung eines regulären Potentials*
Wir betrachten das eindimensionale reguläre Potential

$$\phi(y, x_i) = e^{-\frac{(y-x_i)^2}{\sigma^2}}.$$

Sei $x_* \in \mathbb{R}$. Unter Verwendung der Taylorreihenentwicklung können

wir die R-Entwicklung für alle $|y - x_*| < r_* < \infty$ wie folgt schreiben

$$\begin{aligned}
\phi(y, x_i) &= e^{-\frac{[(y-x_*)-(x_i-x_*)]^2}{\sigma^2}} = e^{-\frac{(y-x_*)^2}{\sigma^2}} e^{-\frac{(x_i-x_*)^2}{\sigma^2}} e^{\frac{2(y-x_*)(x_i-x_*)}{\sigma^2}} \\
&= e^{-\frac{(y-x_*)^2}{\sigma^2}} e^{-\frac{(x_i-x_*)^2}{\sigma^2}} \sum_{k=0}^{\infty} \frac{2^k}{k!} \left(\frac{y - x_*}{\sigma}\right)^k \left(\frac{x_i - x_*}{\sigma}\right)^k \\
&= \sum_{k=0}^{\infty} \underbrace{\frac{2^k}{k!} e^{-\frac{(x_i-x_*)^2}{\sigma^2}} \left(\frac{x_i - x_*}{\sigma}\right)^k}_{=a_k(x_i, x_*)} \underbrace{e^{-\frac{(y-x_*)^2}{\sigma^2}} \left(\frac{y - x_*}{\sigma}\right)^k}_{=R_k(y - x_k)} \\
&= \sum_{k=0}^{\infty} a_k(x_i, x_*) R_k(y - x_k).
\end{aligned}$$

(2) *R-Entwicklung eines singulären Potentials*
Wir verwenden das eindimensionale singuläre Potential

$$\phi(y, x_i) = \frac{1}{y - x_i}$$

mit einer Singularität bei $y = x_i$. Dann gilt für alle $x_* \in \mathbb{R}$ und alle $y \in \mathbb{R}$ mit $|y - x_i| < r_* \leq |x_i - x_*|$

$$\begin{aligned}
\phi(y, x_i) &= \frac{1}{(y - x_*) - (x_i - x_*)} = -\frac{1}{x_i - x_*} \left(1 - \frac{y - x_*}{x_i - x_*}\right)^{-1} \\
&= -\frac{1}{x_i - x_*} \sum_{k=0}^{\infty} \frac{(y - x_*)^k}{(x_i - x_*)^k}, \text{ da } |y - x_*| < |x_i - x_*| \\
&= \sum_{k=0}^{\infty} \underbrace{\frac{-1}{(x_i - x_*)^{k+1}}}_{=a_k(x_i, x_*)} \underbrace{(y - x_*)^k}_{=R_k(y - x_*)} \\
&= \sum_{k=0}^{\infty} a_k(x_i, x_*) R_k(y - x_*).
\end{aligned}$$

10.6 Definition: Seien $\mathbf{x}_* \in \mathbb{R}^d$ und $R_* > 0$. Wir nennen die Entwicklung

$$\phi(\mathbf{y}, \mathbf{x}_i) = \sum_{k=0}^{\infty} b_k(\mathbf{x}_i, \mathbf{x}_*) S_k(\mathbf{y} - \mathbf{x}_*) \tag{10.14}$$

Entwicklung im Fernfeld oder *singulär* außerhalb der Umgebung

$$A_{R_*}(\mathbf{x}_*) = \{\mathbf{y} \in \mathbb{R}^d \mid \|\mathbf{y} - \mathbf{x}_*\|_2 > R_*\},$$

falls die Reihe (10.14) für alle $\mathbf{y} \in A_{R_*}(\mathbf{x}_*)$ konvergiert. Analog zu Definition 10.4 nennen wir die Funktionen $S_k(\cdot)$ *singuläre Basisfunktionen.* Wir bezeichnen (10.14) als *S-Entwicklung.*

Ähnlich wie bei der R-Entwicklung ist die S-Entwicklung eines regulären Potentials überall gültig, für die Konvergenzregion $A_{R_*}(\mathbf{x}_*)$ gibt es also keine Einschränkungen. Liegt ein singuläres Potential vor, so müssen wir sicherstellen, dass der singuläre Punkt außerhalb von $A_{R_*}(\mathbf{x}_*)$ liegt.

10.7 Beispiel:

(1) *S-Entwicklung eines regulären Potentials*
Wir verwenden wie in Beispiel 10.5(1) die Gauß-Funktion

$$\phi(y, x_i) = e^{-\frac{(y-x_i)^2}{\sigma^2}}$$

und erhalten für $x_* \in \mathbb{R}$ und $y \in A_{R_*}(x_*)$

$$\phi(y, x_i) = \sum_{k=0}^{\infty} b_k(x_i, x_*) S_k(y - x_*)$$

mit

$$b_k(x_i, x_*) = \frac{2^k}{k!} e^{-\frac{(x_i-x_*)^2}{\sigma^2}} \left(\frac{x_i - x_*}{\sigma}\right)^k, S_k(y - x_*) = e^{-\frac{(y-x_*)^2}{\sigma^2}} \left(\frac{y - x_*}{\sigma}\right)^k.$$

(2) *S-Entwicklung eines singulären Potentials*
Sei

$$\phi(y, x_i) = \frac{1}{y - x_i}$$

das singuläre Potential aus Beispiel 10.5(2) mit einer Singularität bei $y = x_i$. Dann gilt für alle $x_* \in \mathbb{R}$ und $y \in \mathbb{R}$ mit $|y - x_*| > R_* \geq |x_i - x_*|$

$$\begin{aligned}
\phi(y, x_i) &= \frac{1}{(y - x_*) - (x_i - x_*)} = \frac{1}{y - x_*} \left(1 - \frac{x_i - x_*}{y - x_*}\right)^{-1} \\
&= \sum_{k=0}^{\infty} \underbrace{(x_i - x_*)^k}_{=b_k(x_i, x_*)} \underbrace{\frac{1}{(y - x_*)^{k+1}}}_{=S_k(y - x_*)} \\
&= \sum_{k=0}^{\infty} b_k(x_i, x_*) S_k(y - x_*). \qquad (10.15)
\end{aligned}$$

Im Allgemeinen ist unser Vorgehen folgendermaßen. Liegt ein singuläres Potential mit einer Singularität bei $\mathbf{x}_i$ vor, dann verwenden wir die R-Entwicklung für alle $\mathbf{y} \in \mathbb{R}^d$ mit $\|\mathbf{y} - \mathbf{x}_*\|_2 < \|\mathbf{x}_i - \mathbf{x}_*\|_2$ und die S-Entwicklung für alle $\mathbf{y} \in \mathbb{R}^d$ mit $\|\mathbf{y} - \mathbf{x}_*\|_2 > \|\mathbf{x}_i - \mathbf{x}_*\|_2$. Dabei liegt der singuläre Punkt $\mathbf{x}_i$ auf der Grenze zwischen der R- und der S-Entwicklung. Bei einem regulären Potential verwenden wir je nach Konvergenzgeschwindigkeit die R- oder S-Entwicklung.
Für reguläre Potentiale können wir direkt den Middleman Algorithmus 10.2 anwenden. Allerdings ist es offensichtlich, dass wir nicht mit einem einzigen Entwicklungszentrum $\mathbf{x}_*$ für die gesamte Berechnung auskommen. Wir benötigen hierfür eine Raumpartitionierung.
Für ein singuläres Potential müssen die Quell- und Zielpunkte getrennt vorliegen. Ist dies nicht der Fall, so brauchen wir ebenfalls eine Raumpartition.
Wir stellen nachfolgend eine Modifikation des Middleman Algorithmus vor, die eine Partition der Zielpunkte verwendet.

10.2.2 Modifizierte Middleman Methode

Seien $\mathbf{x}_j \in \mathbb{R}^d$ für $j = 0, \ldots, N-1$ Quell- und $\mathbf{y}_j \in \mathbb{R}^d$ für $j = 0, \ldots, N-1$ Zielpunkte eines Potentials ϕ. Wir nehmen an, die Quell- und Zielpunkte weisen keine erkennbare Struktur oder Partitionierung auf. Seien $y_{min}^{(i)}, y_{max}^{(i)}$ für $i = 1, \ldots, d$ definiert als

$$y_{min}^{(i)} = \min\{y_j^{(i)} \mid j = 0, \ldots, N-1\},$$
$$y_{max}^{(i)} = \max\{y_j^{(i)} \mid j = 0, \ldots, N-1\}.$$

Wir setzen

$$W = [y_{min}^{(1)}, y_{max}^{(1)}] \times \cdots \times [y_{min}^{(d)}, y_{max}^{(d)}] \subset \mathbb{R}^d,$$

die *bounding Box* der Zielpunkte. Wir wollen zunächst eine gleichmäßige Partition erzeugen, das heißt wir teilen jedes Intervall $[y_{min}^{(i)}, y_{max}^{(i)}]$ in gleich viele, nämlich t, Teile. Wir setzen $T = t^d$, dann gilt

$$W = \bigcup_{k=1}^{T} W_k. \tag{10.16}$$

Betrachten wir unser Problem, die Berechnung von

$$h^{(j)} = \sum_{i=0}^{N-1} \phi(\mathbf{y}_j, \mathbf{x}_i) q^{(i)} \text{ für } j = 0, \ldots, N-1, \tag{10.17}$$

so müssen wir zuerst den Quader $W_{\widetilde{k}}$, $1 \leq \widetilde{k} \leq T$ bestimmen, für den $\mathbf{y}_j \in W_{\widetilde{k}}$ erfüllt ist. Dann gilt für die Anzahl $A(d)$ der Nachbarn von $W_{\widetilde{k}}$ in Abhängigkeit von der Dimension d

$$A(1) \leq 2 \text{ und } A(d) \leq A(d-1) + 2(A(d-1)+1) \text{ für } \quad d \geq 3.$$

Unter den *Nachbarn* des Quaders $W_{\widetilde{k}} \subset \mathbb{R}^d$ verstehen wir solche Quader $W_k \subset \mathbb{R}^d$, die mit $W_{\widetilde{k}}$ mindestens eine Seite der Dimension zwischen 0 und $d-1$ gemeinsam haben. Die *Nachbarschaft* eines Quaders $W_{\widetilde{k}} \subset \mathbb{R}^d$ ist die Menge aller Nachbarn. Als Nächstes stellen wir einen Algorithmus für die Berechnung von $h^{(j)}$ und anschließend die zugehörige Komplexitätsabschätzung vor.

10.8 Algorithmus: Seien

$$W_{\widetilde{k}}^{+} = \{\mathbf{x}_i \in W_k \mid W_k \text{ in Nachbarschaft von } W_{\widetilde{k}}\},$$
$$W_{\widetilde{k}}^{-} = \{\mathbf{x}_i \in W_k \mid W_k \text{ außerhalb der Nachbarschaft von } W_{\widetilde{k}}\}.$$

Dann gilt

$$\begin{aligned} h^{(j)} &= \sum_{i=0}^{N-1} q^{(i)} \phi(\mathbf{y}_j, \mathbf{x}_i) \\ &= \underbrace{\sum_{\mathbf{x}_i \in W_{\widetilde{k}}^{+}} q^{(i)} \phi(\mathbf{y}_j, \mathbf{x}_i)}_{=A} + \underbrace{\sum_{\mathbf{x}_i \in W_{\widetilde{k}}^{-}} q^{(i)} \phi(\mathbf{y}_j, \mathbf{x}_i)}_{=B}. \end{aligned}$$

Dabei stellt A den singulären und B den regulären Teil der Summe dar. Für $\mathbf{x}_i \in W_{\widetilde{k}}^{-}$ gilt

$$\phi(\mathbf{y}_j, \mathbf{x}_i) = \sum_{k=0}^{p-1} a_k(\mathbf{x}_i, \mathbf{x}_{r*}) R_k(\mathbf{y}_j - \mathbf{x}_{\widetilde{k}*}) + Fehler(p) \text{ für } \mathbf{y}_j, \mathbf{x}_{\widetilde{k}*} \in W_{\widetilde{k}}.$$

Vernachlässigen wir den Fehlerterm, so erhalten wir für B

$$\begin{aligned} B &= \sum_{\mathbf{x}_i \in W_{\widetilde{k}}^{-}} q^{(i)} \phi(\mathbf{y}_j, \mathbf{x}_i) \\ &= \sum_{k=0}^{p-1} \left[\sum_{\mathbf{x}_i \in W_{\widetilde{k}}^{-}} q^{(i)} a_k(\mathbf{x}_i, \mathbf{x}_{\widetilde{k}*}) \right] R_k(\mathbf{y}_j - \mathbf{x}_{\widetilde{k}*}). \end{aligned}$$

Hierbei ist $\mathbf{x}_{\widetilde{k}*}$ Entwicklungszentrum für $W_{\widetilde{k}}$. Mit der Annahme der gleichmäßigen Verteilung der Quell- und Zielpunkte folgt, dass in der direkten Nachbarschaft des Zielpunktes $\mathbf{y}_j$ nur verhältnismäßig wenige Quellpunkte liegen. Daher berechnen wir A direkt.

Eine Antwort auf die Frage, ob Algorithmus 10.8 eine Beschleunigung gegenüber der direkten Methode bedeutet, gibt uns das folgende Lemma.

10.9 Lemma: Seien die Punkte von K gleichmäßig verteilt. Dann hat der Algorithmus 10.8 die Komplexität $O(N^{2/3}p^{1/2})$.

Beweis: Wir erhalten für den Algorithmus 10.8 die Komplexität

$$O\left(\sum_{j=1}^{T}(N-V_j)p+|W_j|+|W_j|\,V_j\right),$$

wobei V_j die Anzahl der Punkte in der Nachbarschaft von W_j ist. Mit der Annahme, dass wir eine gleichmäßige Verteilung der Punkte auf die einzelnen Partitionen haben, können wir von $V_j \approx \frac{NA(d)}{T}$ ausgehen. Dann bekommen wir

$$\begin{aligned} F(T) &= \sum_{j=1}^{T}(N-V_j)p+|W_j|+|W_j|\,V_j \\ &= \frac{N^2}{T}A(d)+(T-A(d))Np+Np. \end{aligned}$$

$F(T)$ hat das Minimum

$$T_{opt}=\sqrt{\frac{NA(d)}{p}}$$

und damit eine Gesamtkomplexität von

$$O\left(Np\left(2\sqrt{\frac{NA(d)}{p}}-A(d)\right)+Np\right).$$

Da gilt

$$2\sqrt{\frac{NA(d)}{p}}-A(d)+1\leq\sqrt{\frac{N}{p}},$$

beträgt die Obergrenze für die Komplexität $O(N^{2/3}p^{1/2})$. □

Wir können also, eine effiziente Raumpartitionierung und eine schon vorhandene Zerlegung des Potentials vorausgesetzt, die Berechnung von (10.17) beschleunigen, sofern $p \ll N$ ist.
Leider ist auch dieser Algorithmus nicht immer anwendbar. So sind für reguläre Potentiale die Entwicklungen zwar überall gültig, für große Punktmengen wird aber in der Regel die Abbruchzahl p, welche die Genauigkeit der Berechnung steuert, zu groß und damit die Komplexität der Berechnung zu hoch. Singuläre Potentiale können ausschließlich dann verwendet werden, wenn entweder die Quell- oder die Zielpunkte natürlich gruppiert sind. Von dieser Annahme können wir bei unseren Berechnungen nicht ausgehen.
Daher wenden wir uns im nächsten Abschnitt der Single-Level-Fast-Multipole Methode zu, einer weiteren Modifikation der ursprünglichen Fast-Multipole Methode.

10.2.3 Single-Level-Fast-Multipole Methode

Wir beginnen mit der Idee der Single-Level-Fast-Multipole Methode. Anschließend kümmern wir uns um die Grundlagen des Algorithmus, stellen diesen vor und schätzen die Komplexität ab.
Wir behandeln nach wie vor das Summationsproblem

$$h^{(j)} = \sum_{i=0}^{N-1} \phi(\mathbf{y}_j, \mathbf{x}_i) q^{(i)} \text{ für } j = 0, \ldots, N-1$$

mit $\mathbf{x}_i, \mathbf{y}_j \in \mathbb{R}^d$. Ähnlich wie bei Algorithmus 10.8 erzeugen wir eine Raumpartition und erhalten eine bounding Box

$$W = \bigcup_{k=1}^{T} W_k.$$

Wir betrachten für $1 \leq \widetilde{k} \leq T$ die Mengen

$$\begin{aligned} W_{\widetilde{k}}^{[1]} &= \{W_{\widetilde{k}}\}, \\ W_{\widetilde{k}}^{[2]} &= \{W_k \mid W_k \text{ benachbart zu } W_{\widetilde{k}}\} \cup \{W_{\widetilde{k}}\}, \\ W_{\widetilde{k}}^{[3]} &= \{W_k \mid W_k \notin W_{\widetilde{k}}^{[2]}\} \end{aligned}$$

und setzen

$$\phi_{\widetilde{k}}^{[1]}(\mathbf{y}) = \sum_{\mathbf{x}_i \in W_{\widetilde{k}}^{[1]}} \phi(\mathbf{y}, \mathbf{x}_i) q^{(i)}$$
$$\phi_{\widetilde{k}}^{[2]}(\mathbf{y}) = \sum_{\mathbf{x}_i \in W_{\widetilde{k}}^{[2]}} \phi(\mathbf{y}, \mathbf{x}_i) q^{(i)}$$
$$\phi_{\widetilde{k}}^{[3]}(\mathbf{y}) = \sum_{\mathbf{x}_i \in W_{\widetilde{k}}^{[3]}} \phi(\mathbf{y}, \mathbf{x}_i) q^{(i)}$$

für $\mathbf{y} \in W_{\widetilde{k}}$. Für $\mathbf{y}_j \in W_{\widetilde{k}}$ können wir $h^{(j)}$ durch

$$h^{(j)} = \sum_{i=0}^{N-1} \phi(\mathbf{y}_j, \mathbf{x}_i) q^{(i)} = \phi_{\widetilde{k}}^{[2]}(\mathbf{y}_j) + \phi_{\widetilde{k}}^{[3]}(\mathbf{y}_j)$$

ausdrücken. Wir berechnen nun $\phi_{\widetilde{k}}^{[3]}(\mathbf{y}_j)$ mit der lokalen R-Entwicklung um das Zentrum $\mathbf{x}_{\widetilde{k}*} \in W_{\widetilde{k}}$. Für die Berechnung von $\phi_{\widetilde{k}}^{[2]}(\mathbf{y}_j)$ verwenden wir zunächst die S-Entwicklung, die wir allerdings mittels einer geeigneten Transformation in eine lokale R-Entwicklung überführen. Die Summe aller Teilergebnisse liefert uns eine Näherung von $h^{(j)}$ mit der gewünschten Genauigkeit.
Wir betrachten nachfolgend Reihendarstellungen von Funktionen und verwenden dabei stets einen normierten Unterraum $F(\Omega) \subset \mathcal{C}(\Omega)$ mit kompaktem $\Omega \subset \mathbb{R}^d$ und der Norm

$$\|f\|_{\Omega,\infty} = \max_{\mathbf{x} \in \Omega} |f(\mathbf{x})| \text{ für } f \in F(\Omega).$$

Dabei hat $F(\Omega)$ die Gestalt

$$F(\Omega) = \{f \in \mathcal{C}(\Omega) \mid f = \sum_{n=0}^{\infty} a_n f_n \text{ konvergiert absolut und gleichmäßig in } \Omega \subset \mathbb{R}^d \text{ für } f_n \in F(\Omega) \text{ und } a_n \in \mathbb{R}\}.$$

Die Menge $\{f_n\}_{n \geq 0}$ bildet stets eine vollständige Basis. Eine Reihe konvergiert absolut und gleichmäßig in $\Omega \subset \mathbb{R}^d$ gegen f, falls für alle $\varepsilon > 0$ ein $p(\varepsilon) > 0$ existiert, so dass gilt

$$\|f - f^{p(\varepsilon)}\|_{\Omega,\infty} < \varepsilon.$$

Dabei ist

$$f^{p(\varepsilon)}(\mathbf{y}) = \sum_{n=0}^{p(\varepsilon)-1} a_n f_n(\mathbf{y}).$$

Insbesondere existiert zu jedem $\varepsilon > 0$ ein $p(\varepsilon) > 0$, so dass

$$\sum_{n=p(\varepsilon)}^{\infty} |a_n f_n(\mathbf{y})| < \varepsilon \text{ für } \mathbf{y} \in \Omega$$

gilt.
Wir werden im Folgenden diese Idee formalisieren und beginnen mit der Definition eines Translationsoperators.

10.10 Definition: Seien $F(\Omega_1) \subset F(\Omega) \subset \mathcal{C}(\mathbb{R}^d)$ zwei normierte Unterräume mit $\Omega_1 \subset \Omega \subset \mathbb{R}^d$. Sei weiter $\mathbf{t} \in \mathbb{R}^d$ so gewählt, dass $\Omega_2 = \Omega_1 + \mathbf{t} \subset \Omega$ und $F(\Omega_2) \subset F(\Omega)$ gilt. Wir nennen den Operator $\mathcal{T} : F(\Omega_1) \to F(\Omega_2)$ *Translationsoperator zum Vektor* $\mathbf{t}$, falls gilt

$$\mathcal{T}(\mathbf{t})[f(\mathbf{x})] = f(\mathbf{x} + \mathbf{t})$$

für $f \in F(\Omega)$ und $\mathbf{x} \in \Omega_1$ sowie $\mathbf{x} + \mathbf{t} \in \Omega_2$.

Wir nehmen im Folgenden stets an, dass $F(\Omega)$ einen Oberraum der Räume $F(\Omega_j)$ bildet.
Sei ϕ ein Potential mit einer R-Entwicklung um das Entwicklungszentrum $\mathbf{x}_* \in \mathbb{R}^d$ mit

$$\phi(\mathbf{y}, \mathbf{x}_i) = \sum_{j=0}^{\infty} a_j(\mathbf{x}_i, \mathbf{x}_*) R_j(\mathbf{y} - \mathbf{x}_*)$$

für alle $\mathbf{y} \in U_r(\mathbf{x}_*)$. Die Funktionen $\{R_j\}_{j \geq 0}$ bilden eine Basis des translationsinvarianten Raumes $F_R(\Omega) = \text{span}\{R_j \mid j \geq 0\} \subset F(\Omega)$. Dann ist für ein $\mathbf{t} \in \mathbb{R}^d$ und alle $\mathbf{y} \in \mathbb{R}^d$ derart, dass gilt $\mathbf{y} - \mathbf{x}_* + \mathbf{t} \in U_r(\mathbf{x}_*)$,

$$\mathcal{T}(\mathbf{t})[R_j(\mathbf{y} - \mathbf{x}_*)] = R_j(\mathbf{y} - \mathbf{x}_* + \mathbf{t}) = \sum_{k=0}^{\infty} \alpha_{jk}(\mathbf{t}) R_k(\mathbf{y} - \mathbf{x}_*).$$

Wir bezeichnen die Koeffizienten α_{jk} als $R|R$*–Entwicklungskoeffizienten* und setzen

$$(R|R)_{jk}(\mathbf{t}) = \alpha_{jk}(\mathbf{t}) \text{ für } j, k \geq 0.$$

Ordnen wir die Koeffizienten in einer Matrix an, so erhalten wir die $R|R$–*Entwicklungsmatrix*

$$(\mathbf{R}|\mathbf{R})(\mathbf{t}) = \begin{bmatrix} (R|R)_{0,0}(\mathbf{t}) & (R|R)_{0,1}(\mathbf{t}) & \dots \\ (R|R)_{1,0}(\mathbf{t}) & (R|R)_{1,1}(\mathbf{t}) & \dots \\ \vdots & \vdots & \ddots \end{bmatrix}. \tag{10.18}$$

Den auf der Basis $\{R_j\}_{j\geq 0}$ durch die Translations-Matrix $(\mathbf{R}|\mathbf{R})(\mathbf{t})$ beschriebenen Translationsoperator $\mathcal{T}(\mathbf{t})$ nennen wir $R|R$-*Translationsoperator* und bezeichnen diesen mit $(\mathcal{R}|\mathcal{R})(\mathbf{t})$.
Betrachten wir eine R-Entwicklung von ϕ, dann gilt

$$\begin{aligned}
\phi(\mathbf{y}+\mathbf{t},\mathbf{x}_i) &= (\mathcal{R}|\mathcal{R})(\mathbf{t})[\phi(\mathbf{y},\mathbf{x}_i)] \\
&= \sum_{j=0}^{\infty} a_j(\mathbf{x}_i,\mathbf{x}_*)(\mathcal{R}|\mathcal{R})(\mathbf{t})[R_j(\mathbf{y}-\mathbf{x}_*)] \\
&= \sum_{j=0}^{\infty} a_j(\mathbf{x}_i,\mathbf{x}_*) R_j(\mathbf{y}-\mathbf{x}_*+\mathbf{t}) \\
&= \sum_{j=0}^{\infty} a_j(\mathbf{x}_i,\mathbf{x}_*) \sum_{k=0}^{\infty} (R|R)_{jk}(\mathbf{t}) R_k(\mathbf{y}-\mathbf{x}_*) \\
&= \sum_{k=0}^{\infty} \left[\sum_{j=0}^{\infty} (R|R)_{jk}(\mathbf{t}) a_j(\mathbf{x}_i,\mathbf{x}_*) \right] R_k(\mathbf{y}-\mathbf{x}_*) \\
&= \sum_{k=0}^{\infty} \widetilde{a}_{k,\mathbf{t}}(\mathbf{x}_i,\mathbf{x}_*) R_k(\mathbf{y}-\mathbf{x}_*)
\end{aligned}$$

mit

$$\widetilde{a}_{k,\mathbf{t}}(\mathbf{x}_i,\mathbf{x}_*) = \sum_{j=0}^{\infty} (R|R)_{jk}(\mathbf{t}) a_j(\mathbf{x}_i,\mathbf{x}_*).$$

Damit gilt für die Koeffizientenvektoren

$$\mathbf{A}(\mathbf{x}_i,\mathbf{x}_*) = \begin{bmatrix} a_0(\mathbf{x}_i,\mathbf{x}_*) \\ \vdots \\ a_n(\mathbf{x}_i,\mathbf{x}_*) \\ \vdots \end{bmatrix} \in \mathbb{R}^{\infty} \text{ und } \widetilde{\mathbf{A}}_{\mathbf{t}}(\mathbf{x}_i,\mathbf{x}_*) = \begin{bmatrix} \widetilde{a}_{0,\mathbf{t}}(\mathbf{x}_i,\mathbf{x}_*) \\ \vdots \\ \widetilde{a}_{n,\mathbf{t}}(\mathbf{x}_i,\mathbf{x}_*) \\ \vdots \end{bmatrix} \in \mathbb{R}^{\infty}$$

sowie

$$\widetilde{\mathbf{A}}_{\mathbf{t}}(\mathbf{x}_i,\mathbf{x}_*) = \mathbf{A}(\mathbf{x}_i,\mathbf{x}_*+\mathbf{t}) = (\mathbf{R}|\mathbf{R})(\mathbf{t})\mathbf{A}(\mathbf{x}_i,\mathbf{x}_*). \tag{10.19}$$

Um den Sachverhalt zu verdeutlichen, betrachten wir das folgende Beispiel.

10.11 Beispiel: Sei

$$\phi(y, x_i) = \frac{1}{y - x_i},$$

dann erhalten wir für alle $y \in \mathbb{R}$ mit $|y - x_i| < r_* \leq |x_i - x_*|$ die R-Entwicklung

$$\phi(y, x_i) = \sum_{j=0}^{\infty} \frac{-1}{(x_i - x_*)^{j+1}} (y - x_*)^j = \sum_{j=0}^{\infty} a_j(x_i, x_*) R_j(y - x_*).$$

Wählen wir $t \in \mathbb{R}$ so, dass $x_* + t \in U_{r_*}(x_*)$ gilt, erhalten wir die R-Entwicklung

$$\begin{aligned} \phi(y + t, x_i) &= \sum_{j=0}^{\infty} \frac{-1}{(x_i - (x_* + t))^{j+1}} (y - (x_* + t))^j \\ &= \sum_{j=0}^{\infty} \widetilde{a}_{j,t}(x_i, x_*) R_j(y - (x_* + t)). \end{aligned}$$

Für die Funktionenfolge $\{R_j\}_{j \geq 0}$ gilt $R_j(y) = y^j$ und damit

$$R_j(y + t) = \sum_{k=0}^{j} \binom{j}{k} t^{j-k} y^k = \sum_{k=0}^{j} \binom{j}{k} t^{j-k} R_k(y).$$

Dies liefert uns die $R|R$–Entwicklungskoeffizienten

$$(R|R)_{jk}(t) = \begin{cases} \binom{k}{j} t^{k-j} & \text{für } k \leq j \\ 0 & \text{sonst} \end{cases}$$

und damit die Matrix

$$(\mathbf{R}|\mathbf{R})(t) = \begin{bmatrix} 1 & \binom{1}{0} t & \binom{2}{0} t^2 & \dots \\ 0 & 1 & \binom{2}{1} t & \dots \\ 0 & 0 & 1 & \dots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}.$$

Wir sehen, dass

$$\widetilde{a}_{j,t}(x_i, x_*) = a_j(x_i, x_* + t) = \sum_{k=0}^{\infty} (R|R)_{jk}(t) a_k(x_i, x_*)$$

gilt.

Auf ähnliche Weise können wir einen $S|S$–Translationsoperator einführen. Sei

$$\phi(\mathbf{y}, \mathbf{x}_i) = \sum_{j=0}^{\infty} b_j(\mathbf{x}_i, \mathbf{x}_*) S_j(\mathbf{y} - \mathbf{x}_*) \text{ für } \mathbf{y} \in A_R(\mathbf{x}_*)$$

eine S-Entwicklung des Potentials ϕ um das Entwicklungszentrum $\mathbf{x}_*$. Sei $\mathbf{t} \in \mathbb{R}^d$ derart gewählt, dass $\mathbf{y}-\mathbf{x}_*, \mathbf{y}-\mathbf{x}_*+\mathbf{t} \in A_R(\mathbf{x}_*)$ gilt. Ähnlich wie bei der $R|R$–Entwicklung betrachten wir die Funktionen $\{S_j\}_{j\geq 0}$ als eine Basis des translationsinvarianten Raumes $F_S(\Omega) = \text{span}\{S_j \mid j \geq 0\} \subset F(\Omega)$. Dann können wir $S_j(\mathbf{y} - \mathbf{x}_* + \mathbf{t})$ als

$$S_j(\mathbf{y} - \mathbf{x}_* + \mathbf{t}) = \sum_{k=0}^{\infty} (S|S)_{jk}(\mathbf{t}) S_k(\mathbf{y} - \mathbf{x}_*)$$

darstellen. Wir nennen die Koeffizienten $(S|S)_{jk}(\mathbf{t})$ *$S|S$-Entwicklungskoeffizienten* und die unendliche Matrix

$$(\mathbf{S}|\mathbf{S})(\mathbf{t}) = \begin{bmatrix} (S|S)_{0,0}(\mathbf{t}) & (S|S)_{0,1}(\mathbf{t}) & \dots \\ (S|S)_{1,0}(\mathbf{t}) & (S|S)_{1,1}(\mathbf{t}) & \dots \\ \vdots & \vdots & \ddots \end{bmatrix} \tag{10.20}$$

bezeichnen wir als *$S|S$-Entwicklungsmatrix*.
Den auf der Basis $\{S_j\}_{j\geq 0}$ durch die Matrix $(\mathbf{S}|\mathbf{S})(\mathbf{t})$ beschriebenen Translationsoperator $\mathcal{T}(\mathbf{t})$ nennen wir *$S|S$-Translationsoperator* und bezeichnen diesen mit $(\mathcal{S}|\mathcal{S})(\mathbf{t})$.
Fassen wir die Koeffizienten b_j der S–Entwicklung von ϕ in einem Vektor $\mathbf{B}(\mathbf{x}_i, \mathbf{x}_*) \in \mathbb{R}^\infty$ und die Koeffizienten $\widetilde{b}_{j,\mathbf{t}}$ von $(\mathcal{S}|\mathcal{S})(\mathbf{t})[\phi]$ in einem Vektor $\widetilde{\mathbf{B}}_\mathbf{t}(\mathbf{x}_i, \mathbf{x}_*) \in \mathbb{R}^\infty$ zusammen, so erhalten wir die Beziehung

$$\widetilde{\mathbf{B}}_\mathbf{t}(\mathbf{x}_i, \mathbf{x}_*) = (\mathbf{S}|\mathbf{S})(\mathbf{t}) \mathbf{B}(\mathbf{x}_i, \mathbf{x}_*). \tag{10.21}$$

Der letzte Operator, den wir einführen werden, ist der $S|R$–Translationsoperator. Einen $R|S$–Translationsoperator kann man ebenfalls definieren, da aber das Gebiet der S-Entwicklung in der Regel größer ist als das Gebiet der R-Entwicklung, wird dieser Operator üblicherweise nicht benötigt.
Seien $\{R_j\}_{j\geq 0}$ eine Basis des Raumes $F_R(\Omega) = \text{span}\{R_j \mid j \geq 0\} \subset F(\Omega)$ und $\{S_j\}_{j\geq 0}$ entsprechend eine Basis von $F_S(\Omega) = \text{span}\{S_j \mid j \geq 0\} \subset F(\Omega)$. Sei $\mathbf{t} \in \mathbb{R}^d$ derart gewählt, dass $S_j(\cdot + \mathbf{t}) \in F_R(\Omega)$ und $\|t\|_2 > r$ für ein $r > 0$ sowie $r_1 = \|t\|_2 - r$ gilt. Dann erhalten wir $\text{span}\{S_j(\cdot + \mathbf{t}) \mid j \geq 0\} \subset F_R(\Omega)$ sowie $U_{r_1}(\mathbf{x}_* + \mathbf{t}) \subset A_r(\mathbf{x}_*)$ und wir können $S_j(\mathbf{y} - \mathbf{x}_* + \mathbf{t})$ als

$$S_j(\mathbf{y} - \mathbf{x}_* + \mathbf{t}) = \sum_{k=0}^{\infty} (S|R)_{jk}(\mathbf{t}) R_k(\mathbf{y} - \mathbf{x}_*)$$

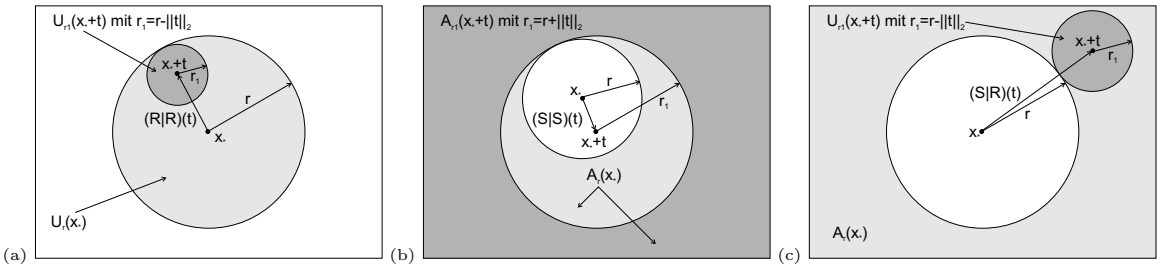

Abbildung 50: (a) $R|R$–Entwicklung (b) $S|S$–Entwicklung (c) $S|R$–Entwicklung

für $\mathbf{y} \in A_r(\mathbf{x}_*)$ darstellen. Wir nennen die Koeffizienten $(S|R)_{jk}(\mathbf{t})$ $S|R$–*Entwicklungskoeffizienten* und die unendliche Matrix

$$(\mathbf{S}|\mathbf{R})(\mathbf{t}) = \begin{bmatrix} (S|R)_{0,0}(\mathbf{t}) & (S|R)_{0,1}(\mathbf{t}) & \dots \\ (S|R)_{1,0}(\mathbf{t}) & (S|R)_{1,1}(\mathbf{t}) & \dots \\ \vdots & \vdots & \ddots \end{bmatrix} \tag{10.22}$$

bezeichnen wir als $S|R$*-Entwicklungsmatrix.*
Den durch die Matrix $(\mathbf{S}|\mathbf{R})(\mathbf{t})$ beschriebenen Translationsoperator $\mathcal{T}(\mathbf{t})$ nennen wir $S|R$*-Translationsoperator* und bezeichnen diesen mit $(\mathcal{S}|\mathcal{R})(\mathbf{t})$. Fassen wir die Koeffizienten b_j der S–Entwicklung von ϕ in einem Vektor $\mathbf{B}(\mathbf{x}_i, \mathbf{x}_*) \in \mathbb{R}^\infty$ und die Koeffizienten $\widetilde{a}_{j,\mathbf{t}}$ von $(\mathcal{S}|\mathcal{R})(\mathbf{t})[\phi]$ in einem Vektor $\widetilde{\mathbf{A}}_\mathbf{t}(\mathbf{x}_i, \mathbf{x}_*) \in \mathbb{R}^\infty$ zusammen, so erhalten wir die Beziehung

$$\widetilde{\mathbf{A}}_\mathbf{t}(\mathbf{x}_i, \mathbf{x}_*) = (\mathbf{S}|\mathbf{R})(\mathbf{t})\mathbf{B}(\mathbf{x}_i, \mathbf{x}_*). \tag{10.23}$$

In Abbildung 50 sehen wir ein Beispiel für die drei Operatoren Wir betrachten die Anwendung des $\mathcal{S}|\mathcal{R}$–Translationsoperators anhand eines Beispiels.

10.12 Beispiel: Sei

$$\phi(y, x_i) = \frac{1}{y - x_i}.$$

Dann erhalten wir eine R-Entwicklung für $y \in U_r(x_i)$ mit $r = |x_i - x_*|$ der folgenden Gestalt

$$\phi(y, x_i) = \sum_{j=0}^{\infty} \frac{-1}{(x_i - x_*)^{j+1}} (y - x_*)^j = \sum_{j=0}^{\infty} a_j(x_i, x_*) R_j(y - x_*).$$

Für $y \in A_r(x_i)$ ist

$$\phi(y, x_i) = \sum_{j=0}^{\infty} (x_i - x_*)^j \frac{1}{(y - x_*)^{j+1}} = \sum_{j=0}^{\infty} b_j(x_i, x_*) S_j(y - x_*)$$

die S-Entwicklung. Betrachten wir $S_j(y-x_*+t)$ für ein $t>0$ mit $|y-x_*|<t$, so liefert uns die Taylorreihenentwicklung

$$S_j(y-x_*+t) = (y-x_*+t)^{-j-1} = \sum_{k=0}^{\infty} \frac{S_j^{(k)}(t)}{k!}(y-x_*)^k$$
$$= \sum_{k=0}^{\infty} \frac{S_j^{(k)}(t)}{k!} R_k(y-x_*) = \sum_{k=0}^{\infty} (S|R)_{jk}(t) R_k(y-x_*).$$

Dabei gilt

$$(S|R)_{jk}(t) = \frac{(-1)^j (j+k)!}{j!k!t^{j+k+1}}.$$

Also ist

$$(\mathbf{S}|\mathbf{R})(t) = \begin{bmatrix} t^{-1} & t^{-2} & t^{-3} & \dots \\ -t^{-2} & -2t^{-3} & -3t^{-4} & \dots \\ t^{-3} & 3t^{-4} & 6t^{-5} & \dots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}.$$

Wir haben somit die gewünschten Operatoren. Allerdings sind sie mit ihrer Darstellung durch unendliche Reihen für praktische Zwecke ungeeignet. Um dieses Problem zu beheben, betrachten wir die eingangs definierte Abbruchzahl $p>0$ und ersetzen die Reihen in der R- und S-Entwicklung sowie bei allen drei Translationsoperatoren durch Summen von 0 bis $p-1$. Wie die Abbruchzahl p im Einzelnen gewählt wird, diskutieren wir bei der Anwendung der Single-Level-Fast-Multipole Methode in Abschnitt 10.2.4. Jetzt wollen wir die obige Theorie auf unser Summationsproblem anwenden.

Seien $K = \{\mathbf{v}_j \mid j = 0, \dots, N-1\} \subset \mathbb{R}^d$ und $\mathbf{v}_*, \mathbf{t} \in \mathbb{R}^d$. Wir betrachten nach wie vor die Funktion $\phi(\mathbf{v}_j, \mathbf{v}_i)$ für $i,j = 0, \dots, N-1$. Sei $\mathbf{v}_* \in \mathbb{R}^d$, dann genügt ϕ einer R-Entwicklung

$$\phi(\mathbf{v}_j, \mathbf{v}_i) = \sum_{k=0}^{\infty} a_k(\mathbf{v}_i, \mathbf{v}_*) R_k(\mathbf{v}_j, \mathbf{v}_*) \tag{10.24}$$

für $\mathbf{v}_j \in U_r(\mathbf{v}_*)$. Analog geben wir die S-Entwicklung

$$\phi(\mathbf{v}_j, \mathbf{v}_i) = \sum_{k=0}^{\infty} b_k(\mathbf{v}_i, \mathbf{v}_*) S_k(\mathbf{v}_j, \mathbf{v}_*) \tag{10.25}$$

für $\mathbf{v}_j \in A_R(\mathbf{v}_*)$ an. Dabei ist $0 < r \leq R$. Dann können wir für ϕ alle drei Translationen definieren.
Betrachten wir den Quader $W = \bigcup_{k=1}^{T} W_k$, definiert in (10.16), so erhalten wir zu jedem $\widetilde{k} \in \{1, \ldots, T\}$ drei Mengen von Quadern.

$$\begin{aligned} W_{\widetilde{k}}^{[1]} &= \{W_{\widetilde{k}}\}, \\ W_{\widetilde{k}}^{[2]} &= \{W_k \mid W_k \text{ benachbart zu } W_{\widetilde{k}}\} \cup \{W_{\widetilde{k}}\}, \\ W_{\widetilde{k}}^{[3]} &= \{W_k \mid W_k \notin W_{\widetilde{k}}^{[2]}\}. \end{aligned}$$

Insbesondere gilt $W_{\widetilde{k}}^{[1]} \subset W_{\widetilde{k}}^{[2]}$.
Wir setzen für $\mathbf{x} \in \mathbb{R}^d$

$$\begin{aligned} \phi_{\widetilde{k}}^{[1]}(\mathbf{x}) &= \sum_{\mathbf{v}_i \in W_{\widetilde{k}}^{[1]}} \phi(\mathbf{x}, \mathbf{v}_i) q^{(i)}, \\ \phi_{\widetilde{k}}^{[2]}(\mathbf{x}) &= \sum_{\mathbf{v}_i \in W_{\widetilde{k}}^{[2]}} \phi(\mathbf{x}, \mathbf{v}_i) q^{(i)}, \\ \phi_{\widetilde{k}}^{[3]}(\mathbf{x}) &= \sum_{\mathbf{v}_i \in W_{\widetilde{k}}^{[3]}} \phi(\mathbf{x}, \mathbf{v}_i) q^{(i)} \end{aligned}$$

und erhalten damit

$$\sum_{i=0}^{N-1} \phi(\mathbf{x}, \mathbf{v}_i) q^{(i)} = \sum_{\mathbf{v}_i \in W_{\widetilde{k}}^{[2]} \cup W_{\widetilde{k}}^{[3]}} \phi(\mathbf{x}, \mathbf{v}_i) q^{(i)} = \phi_{\widetilde{k}}^{[2]}(\mathbf{x}) + \phi_{\widetilde{k}}^{[3]}(\mathbf{x}).$$

Wir fassen jetzt alle Bedingungen zusammen und stellen den Algorithmus vor. Dieser wird in der Literatur meist als *Single-Level-Fast-Multipole Methode* bezeichnet.

Folgende Bedingungen müssen erfüllt sein, damit wir die Single-Level-Fast-Multipole Methode anwenden können.

(1) Wir benötigen eine Menge von Punkten $K = \{\mathbf{v}_j \mid j = 0, \ldots, N-1\} \subset \mathbb{R}^d$.

(2) Wir haben Funktionen oder Potentiale der Form

$$\phi(\mathbf{v}, \mathbf{v}_i) : \mathbb{R}^d \to \mathbb{R} \text{ mit } \mathbf{v} \in \mathbb{R}^d \text{ für } i = 0, \ldots, N-1.$$

(3) ϕ genügt der R–Entwicklung (10.24).

(4) ϕ genügt der S–Entwicklung (10.25).

(5) Für die S–Entwicklungskoeffizienten können wir $S|R$-Translate bilden

$$\widetilde{\mathbf{A}}(\mathbf{v}_i, \mathbf{v}_*) = \mathbf{A}(\mathbf{v}_i, \mathbf{v}_* + \mathbf{t}) = (\mathbf{S}|\mathbf{R})(\mathbf{t})\mathbf{B}(\mathbf{v}_i, \mathbf{v}_*).$$

Damit konvertieren wir S– in R–Entwicklungskoeffizienten.

(6) Wir berechnen die Summe

$$h^{(j)} = \sum_{j=0}^{N-1} \phi(\mathbf{v}_j, \mathbf{v}_i) q^{(i)} \text{ für } j = 0, \ldots, N-1.$$

10.13 Algorithmus: Seien die Bedingungen (1) bis (6) erfüllt. Sei $W = \bigcup_{k=1}^{T} W_k$ eine Raumpartition, definiert wie in (10.16). Dann können wir durch die folgenden Schritte das Matrix-Vektor-Produkt $\mathbf{\Phi q} = \mathbf{h}$ berechnen.

(1) Wir erzeugen S-Entwicklungskoeffizienten für jede Box der Raumpartition.

```
For  W_k̃ ≠ ∅ Durchsuche alle nicht leeren Quader
  Bestimme v_k̃*, das Entwicklungszentrum von W_k̃
   c_k̃ = 0
  For v_i ∈ W_k̃^[1]
     Schleife über alle Punkte des aktuellen Quaders
     Bestimme den S-Koeffizienten b(v_i, v_k̃*)
     c_k̃ = c_k̃ + q^(i) B(v_i, v_k̃*)
  End
End
```

(2) $(S|R)$-Translation der S-Koeffizienten

```
For  W_k̃ ≠ ∅ Durchsuche alle nicht leeren Quader
   Bestimme v_k̃*, das Entwicklungszentrum von W_k̃
   d_k̃ = 0
  For k ∈ ℕ mit W_k ∈ W_k̃^[3]
   Alle Quader außerhalb der Nachbarschaft
     Bestimme v_k*, das Entwicklungszentrum von W_k
```

$\mathbf{d}_{\widetilde{k}} = \mathbf{d}_{\widetilde{k}} + (\mathbf{S}|\mathbf{R})(\mathbf{v}_{\widetilde{k}*} - \mathbf{v}_{k*})\mathbf{c}_k$
End
End

(3) Endsummierung

For $W_{\widetilde{k}} \neq \emptyset$ Durchsuche alle nicht leeren Quader
Bestimme $\mathbf{v}_{\widetilde{k}*}$, das Entwicklungszentrum von $W_{\widetilde{k}}$
For $\mathbf{v}_j \in W_{\widetilde{k}}^{[1]}$
$h^{(j)} = \mathbf{d}_{\widetilde{k}}^t \mathbf{R}(\mathbf{v}_j, \mathbf{v}_{\widetilde{k}*})$
For $\mathbf{v}_i \in W_{\widetilde{k}}^{[2]}$
$h^{(j)} = h^{(j)} + \phi(\mathbf{v}_j, \mathbf{v}_i)$
End
End
End

Hierbei verstehen wir in (3) unter $\mathbf{R}(\mathbf{v}_j, \mathbf{v}_{\widetilde{k}*})$ den Vektor der Funktionen R_j ausgewertet an der Stelle $(\mathbf{v}_j, \mathbf{v}_{\widetilde{k}*})$.

Es stellt sich natürlich die Frage, ob der so durchgeführte Algorithmus eine Verbesserung der Komplexität gegenüber der direkten Methode bietet und falls dies der Fall ist, wie groß die Komplexität ist.

10.14 Lemma: Sei $K = \{\mathbf{v}_j \mid j = 0, \ldots, N-1\} \subset \mathbb{R}^d$. Wir nehmen an, dass die Punkte $\mathbf{v}_j \in K$ gleichmäßig verteilt sind, wir bereits eine Raumpartition, wie in (10.16), haben und die Translationen durch eine $p \times p$–Matrix realisiert wird. Dann hat Algorithmus 10.13 eine Komplexität von $O(N^{4/3}p^{2/3})$.

Beweis: Wir erhalten die folgenden Komplexitäten.

(1) $O(pN)$ für die Erzeugung der S-Koeffizienten.

(2) $O(p^2T^2)$ für die $(S|R)$–Translation.

(3) $O(pN + N\,|W_j|)$ für die Endsummierung.

Dabei ist T, wie in Abschnitt 10.2.2, die Anzahl der Partitionen. Damit ergibt sich die Gesamtkomplexität

$$\begin{aligned} O_{Gesamt} &= O(pN + p^2T^2 + pN + N\,|W_j|) \\ &= O\left(2pN + p^2T^2 + \frac{N^2}{T}\right) \end{aligned}$$

mit $|W_j| = \frac{N}{T}$.
Wollen wir das optimale T bestimmen, so definieren wir die Funktion $F(T)$ mit

$$F(T) = 2pN + p^2T^2 + \frac{N^2}{T}$$

und damit

$$F'(T) = 2p^2T - \frac{N^2}{T^2}.$$

Wir berechnen als Nullstelle von F' und damit als Minimum von F das optimale T

$$T_{opt} = \left(\frac{N^2}{2p^2}\right)^{\frac{1}{3}}$$

Mit T_{opt} gilt dann für die Komplexität

$$\begin{aligned} O_{Gesamt} &= O\left(2pN + p^2T_{opt}^2 + \frac{N^2}{T_{opt}}\right) \\ &= O\left(2pN + p^2\left(\frac{N^2}{2p^2}\right)^{\frac{2}{3}} + \frac{N^2(2p^2)^{\frac{1}{3}}}{N^{\frac{2}{3}}}\right) \\ &= O\left(2pN + p^{\frac{2}{3}}N^{\frac{4}{3}}2^{-\frac{2}{3}} + N^{\frac{4}{3}}(2p^2)^{\frac{1}{3}}\right) \\ &= O\left(N^{\frac{4}{3}}p^{\frac{2}{3}}\right). \end{aligned}$$

□

Unter unserer Standardannahme $p \ll N$ ist dies eine Beschleunigung gegenüber der direkten Methode.
Es gibt noch weitere Modifikationen des Fast-Multipole Algorithmus 10.2, wie zum Beispiel die Multilevel-Fast-Multipole Methode zusammen mit einer hierarchischen Raumpartitionierung. Diese wollen wir hier nicht vorstellen, wir diskutieren statt dessen die Anwendung der Fast-Multipole Methode auf die Thin-Plate Splines.

10.2.4 Anwendung auf Thin-Plate Splines

Wir kommen zu einer Anwendung der Fast-Multipole Methode. Wir untersuchen für den Rest dieses Abschnitts das Matrix-Vektor-Produkt

$$\mathbf{h} = \mathbf{\Phi q} \text{ mit } \phi_{ij} = r_{ij}^2 \log r_{ij} \text{ und } r_{ij} = \|\mathbf{v}'_i - \mathbf{v}'_j\|_2. \tag{10.26}$$

Dabei ist

$$K = \{\mathbf{v}_j \mid j = 0, \ldots, N-1\} \subset \mathbb{R}^d$$

und

$$\mathbf{v}_j = \begin{bmatrix} v_j^{(1)} \\ \vdots \\ v_j^{(d)} \end{bmatrix} \in \mathbb{R}^d \text{ sowie } \mathbf{v}'_j = \begin{bmatrix} v_j^{(1)} \\ \vdots \\ v_j^{(d-1)} \end{bmatrix} \in \mathbb{R}^{d-1} \text{ für } j = 0, \ldots, N-1.$$

Wie wir in den vorhergehenden Abschnitten gesehen haben, wären die gewöhnliche Fast-Multipole Methode, beschrieben in Algorithmus 10.2, oder die Middleman Methode, gemäß Algorithmus 10.8, die richtige Wahl bezüglich der Laufzeit gewesen. Leider benötigen wir für beide Verfahren eine global gültige Entwicklung der Form (10.9). Ein solche Entwicklung, die darüber hinaus numerisch stabil sein sollte, haben wir nicht finden können. Daher diskutieren wir nachfolgend die Anwendung der Single-Level-Fast-Multipole Methode auf Thin-Plate Splines. Wie wir in Abschnitt 10.2.3 gesehen haben, benötigen wir für ein solches Verfahren eine S– und eine R–Entwicklung sowie die im letzten Abschnitt beschriebenen Translationen.

Sei $\varepsilon > 0$ die erforderliche Genauigkeit der Berechnung. Dann benötigen wir insgesamt p Terme einer Zerlegung der Form (10.9). Wir verwenden die Abschätzung aus [Be92] und berechnen p als

$$p = -\left\lfloor \frac{\log \varepsilon}{\log 1.8284} \right\rfloor. \tag{10.27}$$

In unseren Ausführungen orientieren wir uns an der Arbeit [Be92] von *R. Beatson* und *G. Newsam*.

Wir beginnen mit der S–Entwicklung und den $S|S$– sowie $S|R$–Translationen, diese erhalten wir mit Hilfe der nachfolgenden Aussagen. Anschließend führen wir die R–Entwicklung und die $R|R$–Translation ein.

In diesem Abschnitt beschränken wir uns auf die Dimension $d = 3$ und identifizieren Punkte $\mathbf{v}' \in \mathbb{R}^{d-1}$ mit solchen aus $\mathbb{C}$. Wir schreiben für eine Zahl $\mathbf{c} \in \mathbb{C}$

$$\mathbf{c} = \mathcal{R}[\mathbf{c}] + i\mathcal{I}[\mathbf{c}] \text{ sowie } |\mathbf{c}| \text{ statt } \|\mathbf{c}\|_2.$$

Darüber hinaus bezeichnen wir mit $\overline{\mathbf{c}}$ die komplex Konjugierte zu $\mathbf{c} \in \mathbb{C}$. Dann sind auch Operationen wie $\mathbf{cd}, \alpha\mathbf{c}, \mathbf{c}^j \in \mathbb{C}$ für $\mathbf{c}, \mathbf{d} \in \mathbb{C}$, $\alpha \in \mathbb{R}$, $j \in \mathbb{Z}$ wohldefiniert.
Das erste Ergebnis ist eine Entwicklung des Thin-Plate Splines mit einem Zentrum.

10.15 Lemma: Seien $\mathbf{v}, \mathbf{x} \in \mathbb{C}$ und $\widetilde{\lambda} \in \mathbb{R}$ sowie

$$\phi_{\mathbf{x}}(\mathbf{v}) = \widetilde{\lambda} \left|\mathbf{v} - \mathbf{x}\right|^2 \log\left(\left|\mathbf{v} - \mathbf{x}\right|\right).$$

Dann gilt für $|\mathbf{v}| > |\mathbf{x}|$

$$\phi_{\mathbf{x}}(\mathbf{v}) = \mathcal{R}\left[(\overline{\mathbf{v}} - \overline{\mathbf{x}})\widetilde{\lambda}(\mathbf{v} - \mathbf{x})\left(\log \mathbf{v} - \sum_{k=1}^{\infty} \frac{1}{k}\left(\frac{\mathbf{x}}{\mathbf{v}}\right)^k\right)\right] \tag{10.28}$$

$$= \left[\alpha \left|\mathbf{v}\right|^2 - 2\mathcal{R}(\overline{\beta}\mathbf{v}) + \gamma\right] \log|\mathbf{v}| + \mathcal{R}\left[\sum_{k=0}^{\infty}(a_k\overline{\mathbf{v}} + b_k)\mathbf{v}^{-k}\right] \tag{10.29}$$

mit

$$\alpha = \widetilde{\lambda},\ \beta = \widetilde{\lambda}\mathbf{x},\ \gamma = \widetilde{\lambda}\left|\mathbf{x}\right|^2 \tag{10.30}$$

und

$$a_k = \begin{cases} -\beta & \text{für } k = 0 \\ \frac{\widetilde{\lambda}\mathbf{x}^{k+1}}{k(k+1)} & \text{für } k > 0 \end{cases} \text{ sowie } b_k = -\overline{\mathbf{x}}a_k \text{ für } k \geq 0. \tag{10.31}$$

Darüber hinaus gilt für jedes $p \geq 1$ und $0 \neq \mathbf{x} \in \mathbb{C}$

$$\left|\phi_{\mathbf{x}}(\mathbf{v}) - \left[\alpha \left|\mathbf{v}\right|^2 - 2\mathcal{R}(\overline{\beta}\mathbf{v}) + \gamma\right] \log|\mathbf{v}| + \mathcal{R}\left[\sum_{k=0}^{p}(a_k\overline{\mathbf{v}} + b_k)\mathbf{v}^{-k}\right]\right| \leq \frac{\widetilde{\lambda}\left|\mathbf{x}\right|^2}{(p+1)(p+2)}\left(\frac{C+1}{C-1}\right)\left(\frac{1}{C}\right)^p \tag{10.32}$$

mit

$$C = \left|\frac{\mathbf{v}}{\mathbf{x}}\right| > 1.$$

Beweis: Wir nutzen die Eigenschaften der komplexen Zahlen $\mathbf{v}$ und $\mathbf{x}$ und können $\phi_{\mathbf{x}}$ schreiben als

$$\phi_{\mathbf{x}}(\mathbf{v}) = \widetilde{\lambda}(\mathbf{v} - \mathbf{x})(\overline{\mathbf{v}} - \overline{\mathbf{x}})\mathcal{R}\left[\log(\mathbf{v} - \mathbf{x})\right].$$

Für $|\mathbf{v}| > |\mathbf{x}|$ gilt

$$\log(\mathbf{v}-\mathbf{x}) = \log\mathbf{v} + \log\left(1-\frac{\mathbf{x}}{\mathbf{v}}\right) = \log\mathbf{v} - \sum_{k=1}^{\infty}\frac{1}{k}\left(\frac{\mathbf{x}}{\mathbf{v}}\right)^k.$$

Damit erhalten wir Gleichung (10.28). Für die alternative Darstellung (10.29) von $\phi_{\mathbf{x}}$ gilt

$$\begin{aligned}\phi_{\mathbf{x}}(\mathbf{v}) &= \widetilde{\lambda}\mathcal{R}\left[(\overline{\mathbf{v}}-\overline{\mathbf{x}})(\mathbf{v}-\mathbf{x})\left[\log\mathbf{v} - \sum_{k=1}^{\infty}\frac{1}{k}\left(\frac{\mathbf{x}}{\mathbf{v}}\right)^k\right]\right]\\ &= \widetilde{\lambda}\mathcal{R}\left[\left(|\mathbf{v}|^2 - (\mathbf{x}\overline{\mathbf{v}}+\overline{\mathbf{x}}\mathbf{v}) + |\mathbf{x}|^2\right)\log\mathbf{v} - (\overline{\mathbf{v}}-\overline{\mathbf{x}})(\mathbf{v}-\mathbf{x})\sum_{k=1}^{\infty}\frac{1}{k}\left(\frac{\mathbf{x}}{\mathbf{v}}\right)^k\right]\\ &= \left[\alpha|\mathbf{v}|^2 - 2\mathcal{R}(\overline{\beta}\mathbf{v}) + \gamma\right]\log|\mathbf{v}| + \mathcal{R}\left[\sum_{k=0}^{\infty}(a_k\overline{\mathbf{v}}+b_k)\mathbf{v}^{-k}\right]\end{aligned}$$

mit α, β, γ aus (10.30) und den Einträgen a_k, b_k für $k \geq 0$ wie in (10.31). Für die Fehlerabschätzung (10.32) verwenden wir die Abschätzungen

$$\left|\overline{\mathbf{v}}a_k\mathbf{v}^{-k}\right| \leq \frac{C\left|\widetilde{\lambda}\right||\mathbf{x}|^{k+2}}{k(k+1)|C\mathbf{x}|^k} = \frac{\left|\widetilde{\lambda}\right||\mathbf{x}|^2}{k(k+1)C^{k-1}} \text{ für } k \geq 1$$

und

$$\left|b_k\mathbf{v}^{-k}\right| \leq \frac{\left|\widetilde{\lambda}\right||\mathbf{x}|^{k+2}}{k(k+1)|C\mathbf{x}|^k} = \frac{\left|\widetilde{\lambda}\right||\mathbf{x}|^2}{k(k+1)|C|^k} \text{ für } k \geq 1.$$

Damit folgt für die Fehlerabschätzung

$$\begin{aligned}\sum_{k=p+1}^{\infty}\left(|\overline{\mathbf{v}}a_k| + |b_k|\right)\left|\mathbf{v}^{-k}\right| &\leq \frac{\left|\widetilde{\lambda}\right||\mathbf{x}|^2}{(p+1)(p+2)C^p}\left(1+\frac{1}{C}\right)\sum_{k=0}^{\infty}\left(\frac{1}{C}\right)^k\\ &= \frac{\left|\widetilde{\lambda}\right||\mathbf{x}|^2}{(p+1)(p+2)}\frac{C+1}{C-1}\left(\frac{1}{C}\right)^p.\end{aligned}$$

Dies beweist (10.32) und damit sind alle Aussagen von Lemma 10.15 gezeigt. □

Die Fehlerabschätzung (10.32) in Lemma 10.15 ist monoton steigend in $|\mathbf{x}|$ und monoton fallend in $C > 1$.
Wenden wir Lemma 10.15 auf jedes der paarweise verschiedenen Zentren an und summieren die Ergebnisse, so erhalten wir die folgende Approximation des Thin-Plate Splines mit N Zentren.

10.16 Lemma: Seien $\mathbf{v}'_j \in \mathbb{R}^2$ für $j = 0, \ldots, N-1$ Zentren mit $|\mathbf{v}'_j| < r$ und $\widetilde{\lambda}^{(j)}$ für $j = 0, \ldots, N-1$ die zugehörigen Gewichte. Dann können wir für jedes $\mathbf{v} \in \mathbb{R}^2$ mit $|\mathbf{v}| > r$ den Thin-Plate Spline

$$s(\mathbf{v}) = \sum_{j=0}^{N-1} \widetilde{\lambda}^{(j)} \phi_j(\mathbf{v}) = \sum_{j=0}^{N-1} \widetilde{\lambda}^{(j)} \left|\mathbf{v} - \mathbf{v}'_j\right|^2 \log \left|\mathbf{v} - \mathbf{v}'_j\right| \tag{10.33}$$

durch die Entwicklung

$$s(\mathbf{v}) = \left[\alpha \left|\mathbf{v}\right|^2 - 2\mathcal{R}(\overline{\beta}\mathbf{v}) + \gamma\right] \log |\mathbf{v}| + \mathcal{R}\left[\sum_{k=0}^{\infty} (a_k \overline{\mathbf{v}} + b_k)\mathbf{v}^{-k}\right] \tag{10.34}$$

mit

$$\alpha = \sum_{j=0}^{N-1} \widetilde{\lambda}^{(j)},\ \beta = \sum_{j=0}^{N-1} \widetilde{\lambda}^{(j)} \mathbf{v}'_j,\ \gamma = \sum_{j=0}^{N-1} \widetilde{\lambda}^{(j)} \left|\mathbf{v}'_j\right|^2 \tag{10.35}$$

sowie

$$a_k = \begin{cases} -\beta & \text{für } k = 0 \\ \sum_{j=0}^{N-1} \frac{\widetilde{\lambda}^{(j)} \left(\mathbf{v}'_j\right)^{k+1}}{k(k+1)} & \text{für } k \geq 1 \end{cases},\ b_k = \begin{cases} \gamma & \text{für } k = 0 \\ \sum_{j=0}^{N-1} \frac{\widetilde{\lambda}^{(j)} \left(\mathbf{v}'_j\right)^{k+1} \overline{\mathbf{v}'_j}}{k(k+1)} & \text{für } k \geq 1 \end{cases} \tag{10.36}$$

approximieren. Seien

$$M = \sum_{j=0}^{N-1} \left|\widetilde{\lambda}^{(j)}\right| \text{ und } C = \left|\frac{\mathbf{v}}{r}\right|.$$

Dann gilt für alle $p \geq 1$ und $|\mathbf{v}| > r$

$$\left| s(\mathbf{v}) - \left(\left[\alpha \left|\mathbf{v}\right|^2 - 2\mathcal{R}(\overline{\beta}\mathbf{v}) + \gamma\right] \log |\mathbf{v}| + \mathcal{R}\left[\sum_{k=0}^{p} (a_k \overline{\mathbf{v}} + b_k)\mathbf{v}^{-k}\right] \right) \right| \leq \frac{Mr^2}{(p+1)(p+2)} \left(\frac{C+1}{C-1}\right) \left(\frac{1}{C}\right)^p. \tag{10.37}$$

Beweis: Wir wenden die Entwicklung (10.29) für jedes Zentrum, also insgesamt T–mal, an und erhalten sofort die Darstellung (10.38) samt der zugehörigen Darstellungen der Koeffizienten (10.35) und (10.36).

Die Fehlerabschätzung (10.37) erhalten wir durch Aufsummieren der Fehler (10.32) der einzelnen Entwicklungen (10.29). □

Die nachfolgende Gleichung und Abschätzung zeigt uns, wie sich die Verschiebung der Zentren auf eine Thin-Plate Approximation auswirkt.
Sei hierfür

$$s(\mathbf{v}) = \left[\alpha\,|\mathbf{v}-\mathbf{x}| - 2\mathcal{R}(\overline{\beta(\mathbf{v}-\mathbf{x})}) + \gamma\right]\log|\mathbf{v}-\mathbf{x}| + \mathcal{R}\left[\sum_{k=0}^{\infty}\left(a_k(\overline{\mathbf{v}}-\overline{\mathbf{x}})+b_k\right)(\mathbf{v}-\mathbf{x})^{-k}\right]$$

eine Entwicklung des Thin-Plate Splines s um den Punkt $\mathbf{x}$ zu den N Zentren $\mathbf{v}'_0,\ldots,\mathbf{v}'_{N-1}$ und den Gewichten $\widetilde{\lambda}^{(0)},\ldots,\widetilde{\lambda}^{(N-1)}$.
Wir nehmen an, dass alle Zentren innerhalb einer Kugel mit dem Radius $R>0$ um $\mathbf{x}$ liegen. Dann gilt für alle $|\mathbf{v}|>R+|\mathbf{x}|$

$$s(\mathbf{v}) = \left[\alpha\,|\mathbf{v}|^2 - 2\mathcal{R}[\overline{\beta_*}\mathbf{v}] + \gamma_*\right]\log|\mathbf{v}| + \mathcal{R}\left[\sum_{k=0}^{\infty}(A_k\overline{\mathbf{v}}+B_k)\mathbf{v}^{-k}\right] \tag{10.38}$$

mit

$$\beta_* = \alpha\mathbf{x}+\beta \text{ und } \gamma_* = \gamma + 2\mathcal{R}[\overline{\beta}\mathbf{x}] + \alpha\,|\mathbf{x}|^2 \tag{10.39}$$

sowie

$$A_k = \begin{cases} \alpha\mathbf{x}+a_0 = -\beta_* & \text{für } k=0 \\ \frac{\beta_*\mathbf{x}^k}{k} + \frac{\alpha\mathbf{x}^{k+1}}{k+1} + \sum_{j=1}^{k} a_j\binom{k-1}{j-1}\mathbf{x}^{k-j} & \text{für } k\geq 1 \end{cases} \tag{10.40}$$

und

$$B_k = \begin{cases} \overline{\beta_*}\mathbf{x} + (b_0 + a_0\overline{\mathbf{x}}) & \text{für } k=0 \\ \frac{\overline{\beta_*}\mathbf{x}^{k+1}}{k+1} - \frac{\gamma_*\mathbf{x}^k}{k} + \sum_{j=1}^{k}(b_j - a_j\overline{\mathbf{x}})\binom{k-1}{j-1}\mathbf{x}^{k-j} & \text{für } k\geq 1. \end{cases} \tag{10.41}$$

Die Koeffizienten $\alpha,\beta_*,\gamma_*,A_k,B_k$ der verschobenen Entwicklung (10.38) finden wir in der Entwicklung (10.29) wieder, wenn wir um $\mathbf{x}=0$ entwickeln.
Darüber hinaus erhalten wir die Fehlerabschätzung

$$\left|s(\mathbf{v}) - \left(\left[\alpha\,|\mathbf{v}|^2 - 2\mathcal{R}[\overline{\beta_*}\mathbf{v}] + \gamma_*\right]\log|\mathbf{v}| + \mathcal{R}\left[\sum_{k=0}^{\infty}(A_k\overline{\mathbf{v}}+B_k)\mathbf{v}^{-k}\right]\right)\right| \leq \frac{M\mathbf{u}^2}{(p+1)(p+2)}\frac{C+1}{C-1}\left(\frac{1}{C}\right)^p \tag{10.42}$$

mit

$$\mathbf{u} = \max_{0 \le i \le N-1} |\mathbf{v}'_i| \le R + |\mathbf{x}|$$

und

$$C = \left| \frac{\mathbf{v}}{\mathbf{u}} \right|.$$

Mit den eben eingeführten Entwicklungen sind wir nun in der Lage, eine Approximation des Thin-Plate Splines durch eine lokale Taylorreihen-Entwicklung einzuführen.
Sei

$$s(\mathbf{v}) = \sum_{j=0}^{N-1} \widetilde{\lambda}^{(j)} \left| \mathbf{v} - \mathbf{v}'_j \right|^2 \log \left| \mathbf{v} - \mathbf{v}'_j \right| \tag{10.43}$$

ein Thin-Plate Spline mit den Zentren $\mathbf{v}'_0, \ldots, \mathbf{v}'_{N-1} \in \mathbb{R}^2$ und den zugehörigen Gewichten $\widetilde{\lambda}^{(0)}, \ldots, \widetilde{\lambda}^{(N-1)} \in \mathbb{R}$. Wir nehmen an, alle Zentren liegen innerhalb einer Kugel D_1 mit Radius r um $\mathbf{x}$.
Dann approximiert die abgebrochene Reihenentwicklung

$$\begin{aligned} s_1(\mathbf{v}) = &\left[\alpha |\mathbf{v} - \mathbf{x}| - 2\mathcal{R}(\overline{\beta(\mathbf{v} - \mathbf{x})}) + \gamma \right] \log |\mathbf{v} - \mathbf{x}| \\ &+ \mathcal{R} \left[\sum_{k=0}^{p} \left(a_k(\overline{\mathbf{v}} - \overline{\mathbf{x}}) + b_k \right) (\mathbf{v} - \mathbf{x})^{-k} \right] \end{aligned} \tag{10.44}$$

den Thin-Plate Spline (10.43) an der Kugel D_2 mit Radius r um den Ursprung. Dabei gilt für $p \ge 1$

$$\max_{\mathbf{v} \in D_2} |s(\mathbf{v}) - s_1(\mathbf{v})| \le \frac{Mr^2}{(p+1)(p+2)} \frac{C+1}{C-1} \left(\frac{1}{C} \right)^p \tag{10.45}$$

mit

$$M = \sum_{j=0}^{N-1} \left| \widetilde{\lambda}^{(j)} \right|.$$

Darüber hinaus können wir innerhalb von D_2 den Thin-Plate Spline s_1 als

$$s_1(\mathbf{v}) = \mathcal{R} \left[\sum_{l=0}^{\infty} (g_l \overline{\mathbf{v}} + h_l) \mathbf{v}^l \right] \tag{10.46}$$

darstellen. Dabei sind β_* und γ_* definiert wie in (10.39) und

$$g_l = \begin{cases} -\beta_* \log |\mathbf{x}| + \sum_{k=0}^{p} a_k (-1)^k \frac{1}{\mathbf{x}^k} & \text{für } l = 0 \\ \alpha \log |\mathbf{x}| + \frac{\beta_*}{\mathbf{x}} + \sum_{k=0}^{p} a_k (-1)^k \frac{k}{\mathbf{x}^{1+k}} & \text{für } l = 1 \\ \frac{-\alpha}{(l-1)\mathbf{x}^{l-1}} - \frac{\gamma_*}{l\mathbf{x}^l} + \sum_{k=0}^{p} a_k (-1)^k \binom{l+k-1}{k-1} \frac{1}{\mathbf{x}^{l+k}} & \text{für } l \ge 2 \end{cases} \tag{10.47}$$

und

$$h_l = \begin{cases} \gamma_* \log|\mathbf{x}| \sum_{k=0}^{p} (b_k - a_k\overline{\mathbf{x}})(-1)^k \frac{1}{\mathbf{x}^k} & \text{für } l = 0 \\ -\overline{\beta_*} \log|\mathbf{x}| \frac{\gamma_*}{\mathbf{x}} + \sum_{k=0}^{p} (b_k - a_k\overline{\mathbf{x}})(-1)^k \frac{1}{\mathbf{x}^{1+k}} & \text{für } l = 1 \\ \frac{\overline{\beta_*}}{(l-1)\mathbf{x}^{l-1}} - \frac{\gamma_*}{l\mathbf{x}^l} + \sum_{k=0}^{p} (b_k - a_k\overline{\mathbf{x}})(-1)^k \binom{l+k-1}{k-1} \frac{1}{\mathbf{x}^{l+k}} & \text{für } l \geq 2. \end{cases} \tag{10.48}$$

Schließlich gilt für $q \geq 1$

$$\max_{\mathbf{v} \in D_2} \left| s_1(\mathbf{v}) - \mathcal{R}\left[\sum_{l=0}^{q} (g_l \overline{\mathbf{v}} + h_l) \mathbf{v}^l \right] \right|$$

$$\leq Mr^2 \left[\left(\frac{2C^2 + 9C + 9}{Cq} \right) \left(\frac{1}{C+1} \right)^q + 2(C+3) \left(\frac{C}{C-1} \right) \left(\frac{1}{C} \right)^{q+1} \right] \tag{10.49}$$

mit

$$M = \sum_{j=0}^{N-1} \left| \widetilde{\lambda}^{(j)} \right|.$$

Mit diesen Ergebnissen können wir die für die Single-Level-Fast-Multipole Methode wichtigen Entwicklungen und Translationen angeben.
Sei im Folgenden

$$\phi_{\mathbf{x}}(\mathbf{v}) = \widetilde{\lambda} \left| \mathbf{v} - \mathbf{x} \right|^2 \log|\mathbf{v} - \mathbf{x}|. \tag{10.50}$$

Unter der Annahme, dass wir bereits haben eine Raumpartition

$$W = \bigcup_{k=1}^{T} W_k$$

wie in (10.16) haben, bezeichnen wir mit $\mathbf{v}_{\widetilde{k}*} \in W_{\widetilde{k}}$ das Entwicklungszentrum von $W_{\widetilde{k}}$.
Dann beschreibt die Gleichung (10.29) die S-Entwicklung um $\mathbf{v}'_{\widetilde{k}*}$ von $\phi_{\mathbf{x}}$, gemeinsam mit den Gleichungen (10.30) und (10.31) für $\alpha, \beta, \gamma, a_k, b_k$ sowie $\mathbf{x} = \mathbf{v}'_j - \mathbf{v}'_{\widetilde{k}*}$ für ein $j \in \{0, \ldots, N-1\}$.
Analog erhalten wir die R-Entwicklung von $\phi_{\mathbf{x}}$, welche in [Be92] nicht behandelt wird, als

$$s_1(\mathbf{v}) = \mathcal{R}\left[\sum_{l=0}^{\infty} (g_l \overline{\mathbf{v}} + h_l) \mathbf{v}^l \right] \tag{10.51}$$

mit

$$g_l = \begin{cases} -2\widetilde{\lambda}\mathbf{x}\log|\mathbf{x}| & \text{für } l=0 \\ \widetilde{\lambda}(1+\log|\mathbf{v}|) & \text{für } l=1 \\ \frac{-\widetilde{\lambda}\mathbf{x}^{-(l-1)}}{l(l-1)} & \text{für } l\geq 2, \end{cases}$$

$$h_l = \begin{cases} \widetilde{\lambda}\,|\mathbf{x}|^2\log|\mathbf{x}| & \text{für } l=0 \\ \widetilde{\lambda}\mathbf{x} & \text{für } l=1 \\ \frac{\widetilde{\lambda}\overline{\mathbf{x}}\mathbf{x}^{-(l-1)}}{l(l-1)} & \text{für } l\geq 2 \end{cases} \qquad (10.52)$$

$$\text{und } \mathbf{x} = \mathbf{v}'_j - \mathbf{v}'_{\widetilde{k}*}.$$

Für die Translation sei $\mathbf{v}_{\widetilde{k}*}$ das Entwicklungszentrum des Startquaders $W_{\widetilde{k}}$ und $\mathbf{v}_{\widetilde{k}*\mathbf{x}}$ das Entwicklungszentrum des Quaders, in den wir abbilden wollen. Wir schreiben für die Translationen $\mathbf{x} = \mathbf{v}_* - \mathbf{v}_{*\mathbf{x}}$.
Die $S|S$–Translation ist dann gegeben durch die Gleichung (10.38) mit (10.39) sowie (10.40) und (10.41).
Ebenso erhalten wir die $S|R$–Translation durch die Gleichung (10.46) in Kombination mit den Darstellungen (10.47) und (10.48).
Wir müssen jetzt die $R|R$–Translation angeben, welche in [Be92] ebenfalls nicht aufgeführt ist. Dazu betrachten wir (10.46) und entwickeln s_1 an der Stelle $\mathbf{v}-\mathbf{x}$. Dann ist

$$s_1(\mathbf{v}-\mathbf{x}) = \mathcal{R}\left[\sum_{l=0}^{\infty}(g_l(\overline{\mathbf{v}}-\overline{\mathbf{x}})+h_l)(\mathbf{v}-\mathbf{x})^l\right].$$

Wir setzen $h_l^* = h_l - g_l\overline{\mathbf{x}} \in \mathbb{C}$ und erhalten

$$s_1(\mathbf{v}-\mathbf{x}) = \mathcal{R}\left[\sum_{l=0}^{\infty}(g_l\overline{\mathbf{v}}+h_l^*)(\mathbf{v}-\mathbf{x})^l\right].$$

Wenden wir den binomischen Satz auf $(\mathbf{v}-\mathbf{x})^l$ an, so gilt

$$s_1(\mathbf{v}-\mathbf{x}) = \mathcal{R}\left[\sum_{l=0}^{\infty}(g_l\overline{\mathbf{v}}+h_l^*)\sum_{k=0}^{l}\binom{l}{k}(-\mathbf{x})^{l-k}\mathbf{v}^k\right]. \qquad (10.53)$$

Mit Hilfe von (10.53) können wir die $R|R$–Translation als

$$s_1(\mathbf{v}-\mathbf{x}) = \mathcal{R}\left[\sum_{k=0}^{\infty}(G_k\overline{\mathbf{v}}+H_k)\mathbf{v}^k\right] \qquad (10.54)$$

mit

$$G_k = \begin{cases} \sum_{l=0}^{p-1} g_l \binom{l}{k} (-\mathbf{x})^{l-k} & \text{für } l \geq k \\ 0 & \text{für } l < k, \end{cases} \tag{10.55}$$

$$H_k = \begin{cases} \sum_{l=0}^{p-1} h_l^* \binom{l}{k} (-\mathbf{x})^{l-k} & \text{für } l \geq k \\ 0 & \text{für } l < k \end{cases} \tag{10.56}$$

schreiben.
Mit diesen Informationen können wir Algorithmus 10.13 anwenden und so die Berechnung des Matrix-Vektor Produkts $\mathbf{\Phi q} = \mathbf{h}$ abhängig von der Abbruchzahl $p \in \mathbb{N}$ auf $O(N^{4/3}p^{2/3})$ beschleunigen. Die Daten der Firma *Werth Messtechnik GmbH* haben in der Regel eine Genauigkeit von 10^{-6}, gemäß Gleichung (10.27) erhalten wir

$$p = -\left\lfloor \frac{\log 10^{-6}}{\log 1.8284} \right\rfloor \approx -\lfloor -22.89454 \rfloor = 23.$$

Wir erhalten also eine deutliche Beschleunigung bei Einhaltung der gegebenen Genauigkeit, da offensichtlich $p \ll N$ für Flächen der Größe $N > 10000$ gilt.
Auf Grund des vernachlässigbaren Wertes p können wir annehmen, dass sich das Matrix-Vektor Produkt $\mathbf{\Phi q}$ mit einer Komplexität von $O(N^{4/3})$ berechnen lässt.

10.3 Lösung des linearen Gleichungssystems (10.6)

Wie eingangs besprochen hat das lineare Gleichungssystem

$$\mathbf{Mf} = \left[\begin{array}{c|c} \mathbf{A} + \zeta \mathbf{I}_N & \mathbf{B} \\ \hline \mathbf{B}^t & \mathbf{0} \end{array}\right] \left[\frac{\boldsymbol{\lambda}}{\boldsymbol{\mu}}\right] = \left[\frac{\mathbf{v}^{(d)}}{\mathbf{0}}\right] = \mathbf{b}$$

für große N eine $(N+d+1)\times(N+d+1)$–Designmatrix, die dicht besetzt ist. Daher kommen direkte Lösungsverfahren nicht in Frage, denn auf diesem Wege wäre lediglich eine Komplexität in der Größenordnung von $O(N^3)$ zu erreichen. Wir stellen in diesem Abschnitt eine Methode für solche Matrizen vor, die in Verbindung mit der in Abschnitt 10.2.4 besprochenen Fast-Multipole Methode eine Laufzeitverbesserung verspricht. Sei im Folgenden

$$\mathbf{Mf} = \mathbf{b} \tag{10.57}$$

mit

$$\mathbf{M} = \left[\begin{array}{c|c} \mathbf{A} + \zeta \mathbf{I}_N & \mathbf{B} \\ \hline \mathbf{B}^t & \mathbf{0} \end{array}\right] = \left[\begin{array}{c|c} \mathbf{A}_* & \mathbf{B} \\ \hline \mathbf{B}^t & \mathbf{0} \end{array}\right] \tag{10.58}$$

und

$$\mathbf{f} = \left[\frac{\boldsymbol{\lambda}}{\boldsymbol{\mu}}\right] \text{ sowie } \mathbf{b} = \left[\frac{\mathbf{v}^{(d)}}{\mathbf{0}}\right]. \tag{10.59}$$

Bei der Auswahl der für die Lösung des linearen Gleichungssystems (10.6) in Frage kommenden Verfahren wird die Anzahl der möglichen Ansätze durch zwei weitere Eigenschaften von $\mathbf{M}$ verringert. Zum einen ist die Matrix $\mathbf{M}$ in der Regel schlecht konditioniert, zum anderen so gut wie nie positiv definit oder zumindest positiv semidefinit. Als einzige Vorteile versprechende Eigenschaft bleibt die Symmetrie von $\mathbf{M}$. Daher kommen auch die üblichen iterativen Verfahren wie das Jacobi– oder das Gauß-Seidel–Verfahren nicht in Frage. Verfahren für dicht besetzte, große Gleichungssysteme, wie zum Beispiel das SYMMLQ–Verfahren oder das MINRES–Verfahren, beide beschrieben in [Pai75], berücksichtigen nicht die spezielle Gestalt der Matrix $\mathbf{M}$. Wir wollen in dem nachfolgend vorgestellten Verfahren die Struktur von $\mathbf{M}$ ausnutzen und wenden ein modifiziertes Verfahren an, welches auf der Methode der konjugierten Gradienten in Verbindung mit einer geeigneten Projektion basiert.
Wir schreiben das lineare Gleichungssystem (10.57) aus und erhalten für $d = 3$ das System

$$\begin{aligned} \mathbf{A}_* \boldsymbol{\lambda} + \mathbf{B} \boldsymbol{\mu} &= \mathbf{v}^{(3)} \\ \mathbf{B}^t \boldsymbol{\lambda} &= \mathbf{0} \end{aligned} \tag{10.60}$$

mit $\mathbf{A}_* \in \mathbb{R}^{N \times N}$, $\mathbf{B} \in \mathbb{R}^{N \times 3}$, $\boldsymbol{\lambda} \in \mathbb{R}^N$, $\boldsymbol{\mu} \in \mathbb{R}^3$. Wie üblich fassen wir $\mathbf{0}$ je nach Dimension entweder als Nullmatrix oder als Nullvektor auf.
Sei $\mathbf{B} = \mathbf{QR}$ die QR–Zerlegung von $\mathbf{B}$ mit einer orthogonalen Matrix $\mathbf{Q} \in \mathbb{R}^{N \times N}$ und

$$\mathbf{R} = \left[\begin{array}{c} \widehat{\mathbf{R}} \\ \hline \mathbf{0} \end{array}\right] \in \mathbb{R}^{N \times 3}.$$

Dabei ist $\widehat{\mathbf{R}} \in \mathbb{R}^{3 \times 3}$ eine obere Dreiecksmatrix. Seien $\mathbf{e}_0, \ldots, \mathbf{e}_{N-1} \in \mathbb{R}^N$ Einheitsvektoren des $\mathbb{R}^N$. Wir setzen

$$V_{\mathbf{Q}} = \operatorname{span}\{\mathbf{Q}\mathbf{e}_j \mid j = 3, \ldots, N-1\}, \tag{10.61}$$

und formulieren das folgende Lemma.

10.17 Lemma: $V_{\mathbf{Q}}$, definiert wie in (10.61), ist ein $(N-3)$–dimensionaler Unterraum des $\mathbb{R}^N$. Darüber hinaus gilt $\mathbf{B}^t\mathbf{v} = \mathbf{0} \in \mathbb{R}^3$ für alle $\mathbf{v} \in V_{\mathbf{Q}}$.

Beweis: Die Matrix $\mathbf{Q}$ ist orthogonal, hat also insbesondere vollen Rang. Daher sind die Vektoren $\mathbf{Q}\mathbf{e}_j$ für $j = 3, \ldots, N-1$ linear unabhängig. Somit gilt $\dim V_{\mathbf{Q}} = N - 3$.
Weiter ist

$$\mathbf{B}^t\mathbf{Q}\mathbf{e}_j = \mathbf{R}^t \underbrace{\mathbf{Q}^t\mathbf{Q}}_{=\mathbf{I}_N} \mathbf{e}_j = \mathbf{R}^t\mathbf{e}_j = \mathbf{0}$$

für $j \geq 3$, denn für $\mathbf{R} = (r_{ik})_{i,k}$ gilt $r_{ik} = 0$ für alle $i \geq 3$. □

Wir wählen $\boldsymbol{\lambda}_0 \in V_{\mathbf{Q}}$, da in diesem Fall $\mathbf{B}^t\boldsymbol{\lambda}_0 = \mathbf{0}$ gilt. Auch im weiteren Verlauf wählen wir nur solche $\boldsymbol{\lambda}_k$, die in $V_{\mathbf{Q}}$ liegen, damit die zweite Bedingung von (10.60) automatisch erfüllt ist. Den Startwert für $\boldsymbol{\mu}_0$ bestimmen wir so, dass

$$\left\| \left(\mathbf{A}_*\boldsymbol{\lambda}_0 - \mathbf{v}^{(3)} \right) + \mathbf{B}\boldsymbol{\mu}_0 \right\|_2 = \min \tag{10.62}$$

gilt. Wir lösen also das lineare Gleichungsystem

$$\mathbf{B}^t\mathbf{B}\boldsymbol{\mu}_0 = \mathbf{B}^t \left(\mathbf{v}^{(3)} - \mathbf{A}_*\boldsymbol{\lambda}_0 \right). \tag{10.63}$$

Damit haben wir die Startwerte $\boldsymbol{\lambda}_0$ und $\boldsymbol{\mu}_0$ berechnet.
Nun bestimmen wir in jedem Schritt stets zuerst ein neues $\boldsymbol{\lambda}_k \in V_{\mathbf{Q}}$ mit Hilfe eines modifizierten Verfahren der konjugierten Gradienten, das wir im Anschluss diskutieren. Dieses $\boldsymbol{\lambda}_k$ benutzen wir, um $\boldsymbol{\mu}_k$ zu bestimmen. Die Vorgehensweise gleicht der in (10.62) bzw. in (10.63). Gilt für ein $k \in \mathbb{N}$

$$\left\| \mathbf{A}_*\boldsymbol{\lambda}_k + \mathbf{B}\boldsymbol{\mu}_k - \mathbf{v}^{(3)} \right\|_2 < tol_1$$

und

$$\left\| \mathbf{B}^t\boldsymbol{\lambda}_k \right\|_2 < tol_2$$

für vorher festgelegte Schwellenwerte $tol_1, tol_2 \in \mathbb{R}$, so haben wir das Gleichungssystem (10.57) mit für uns ausreichender Genauigkeit gelöst und wir beenden das Verfahren.
Für große Werte N erweist sich die klassische QR–Zerlegung in der Regel als sehr aufwändig, daher haben wir bei der Implementierung eine Modifikation des Verfahrens durchgeführt, die wir jetzt vorstellen werden.

Wir bestimmen mit Hilfe des modifizierten Gram-Schmidt–Verfahrens nach [Go96], Seite 231, die QR–Zerlegung von $\mathbf{B}$

$$\mathbf{B} = \mathbf{Q}_1\widehat{\mathbf{R}} \text{ mit } \mathbf{Q}_1 \in \mathbb{R}^{N\times 3} \text{ und } \widehat{\mathbf{R}} \in \mathbb{R}^{3\times 3}.$$

Dabei besteht $\mathbf{Q}_1$ aus orthonormalen Spalten, nämlich aus den ersten drei Spalten der Matrix $\mathbf{Q}$ der gewöhnlichen QR–Zerlegung. Also gilt $\mathbf{Q}_1^t\mathbf{Q}_1 = \mathbf{I}_3$. Die Matrix $\widehat{\mathbf{R}}$ ist die obere 3×3 Teilmatrix der Matrix $\mathbf{R}$.
Wir setzen $\mathbf{X} = \mathbf{B}^t\mathbf{B}$, dann ist $\mathbf{X}$ eine symmetrische, positiv definite Matrix und wir berechnen die Cholesky-Zerlegung $\mathbf{X} = \mathbf{G}^t\mathbf{G}$. Dabei ist $\mathbf{G} \in \mathbb{R}^{3\times 3}$ eine obere Dreiecksmatrix.
Wegen

$$\mathbf{X} = \mathbf{B}^t\mathbf{B} = \left(\mathbf{Q}_1\widehat{\mathbf{R}}\right)^t\left(\mathbf{Q}_1\widehat{\mathbf{R}}\right) = \widehat{\mathbf{R}}^t\underbrace{\left(\mathbf{Q}_1^t\mathbf{Q}_1\right)}_{=\mathbf{I}_3}\widehat{\mathbf{R}} = \widehat{\mathbf{R}}^t\widehat{\mathbf{R}}$$

und der Eindeutigkeit der Cholesky-Zerlegung erhalten wir $\mathbf{G} = \widehat{\mathbf{R}}$ und damit auch $\mathbf{Q}_1 = \mathbf{B}\mathbf{G}^{-1}$. Wir können die folgende Aussage formulieren.

10.18 Lemma: Wir setzen

$$V_{\mathbf{Q}_1} = \text{span}\{\mathbf{Q}_1^\perp\} \subset \mathbb{R}^N. \tag{10.64}$$

Dann ist $V_{\mathbf{Q}_1}$ ein $(N-3)$–dimensionaler Unterraum des $\mathbb{R}^N$. Darüber hinaus gilt

$$\mathbf{B}^t\mathbf{v} = \mathbf{0} \in \mathbb{R}^3 \text{ für alle } \mathbf{v} \in \mathbb{R}^N.$$

Beweis: Die Spalten von $\mathbf{Q}_1$ spannen einen dreidimensionalen Unterraum des $\mathbb{R}^N$ auf. Damit gilt wegen

$$\mathbb{R}^N = \text{span}\{\mathbf{Q}_1\} \perp \text{span}\{\mathbf{Q}_1^\perp\}$$

$\dim V_{\mathbf{Q}_1} = N-3$. Sei $\mathbf{B} = \mathbf{Q}\mathbf{R}$, dann ist

$$\text{span}\{\mathbf{Q}_1^\perp\} = \text{span}\{\mathbf{Q}\mathbf{e}_j \mid j = 3,\ldots,N-1\}.$$

Dann folgt die zweite Behauptung sofort aus Lemma 10.17. □

Das Ziel ist es also, jedes neu berechnete $\boldsymbol{\lambda}_k \in \mathbb{R}^N$ so zu korrigieren, dass dieses in dem Raum $V_{\mathbf{Q}_1}$ liegt. Damit erfüllen wir die Bedingung $\mathbf{B}^t\boldsymbol{\lambda}_k = \mathbf{0}$.

Die Korrektur können wir mit Hilfe einer geeigneten Projektion bewerkstelligen. Sei $\mathbf{v} \in \mathbb{R}^N$, dann gilt

$$\begin{aligned}\mathbf{Q}_1\mathbf{Q}_1^t\mathbf{v} &= \left(\mathbf{B}\mathbf{G}^{-1}\right)\left(\mathbf{B}\mathbf{G}^{-1}\right)^t\mathbf{v} = \left[\mathbf{B}\mathbf{G}^{-1}\mathbf{G}^{-t}\mathbf{B}^t\right]\mathbf{v} \\ &= \left[\mathbf{B}\left(\mathbf{G}^t\mathbf{G}\right)^{-1}\mathbf{B}^t\right]\mathbf{v} = \left(\mathbf{B}\mathbf{X}^{-1}\mathbf{B}^t\right)\mathbf{v} \\ &= \left(\mathbf{B}\mathbf{X}^{-1}\right)\mathbf{B}^t\mathbf{v} = \mathbf{0}.\end{aligned}$$

Wir erhalten die Projektion als

$$\left[\mathbf{I}_N - \mathbf{Q}_1\mathbf{Q}_1^t\right]\mathbf{v} \in \operatorname{span}\{\mathbf{Q}_1^t\} \text{ für alle } \mathbf{v} \in \mathbb{R}^N. \tag{10.65}$$

Wir setzen $\mathbf{T} = \mathbf{I}_N - \mathbf{Q}_1\mathbf{Q}_1^t$ und bekommen den folgenden Algorithmus.

10.19 Algorithmus: Sei $\mathbf{M}\mathbf{f} = \mathbf{b}$ das lineare Gleichugssystem (10.57) für $d = 3$. Dann liefern die folgenden Schritte einen Lösungsvektor $\mathbf{f}$.

(1) Wir bestimmen die QR–Zerlegung der Matrix $\mathbf{B}$ mit dem modifizierten Gram-Schmidt–Verfahren und erhalten

$$\mathbf{B} = \mathbf{Q}_1\widehat{\mathbf{R}} \text{ mit } \mathbf{Q}_1 \in \mathbb{R}^{N\times 3} \text{ und } \widehat{\mathbf{R}} \in \mathbb{R}^{3\times 3}.$$

(2) Dann berechnen wir

$$\mathbf{T} = \mathbf{I}_N - \mathbf{Q}_1\mathbf{Q}_1^t.$$

(3) Der Startwert für $\boldsymbol{\lambda}$ ergibt sich als

$$\boldsymbol{\lambda}_0 = \mathbf{T}\mathbf{e}_j \text{ für ein } j \in \{3, \ldots, N-1\}.$$

(4) Den Startwert für $\boldsymbol{\mu}$ erhalten wir als Lösung des Minimierungsproblems (10.62), also als Ergebnis des linearen Gleichungssystems

$$\mathbf{B}^t\mathbf{B}\boldsymbol{\mu}_0 = \mathbf{B}^t\left(\mathbf{v}^{(3)} - \mathbf{A}_*\boldsymbol{\lambda}_0\right).$$

(5) Wir führen folgende Schritte so lange durch, bis

$$\left\|\mathbf{A}_*\boldsymbol{\lambda}_k + \mathbf{B}\boldsymbol{\mu}_k - \mathbf{v}^{(3)}\right\|_2 < tol_1 \text{ und } \left\|\mathbf{B}^t\boldsymbol{\lambda}_k\right\|_2 < tol_2$$

für ein $k \geq 0$ und $tol_1, tol_2 \in \mathbb{R}$ gilt.

(i) Durch die Anwendung des modifizierten Verfahrens der konjugierten Gradienten berechnen wir $\boldsymbol{\lambda}_{k+1}$ aus $\boldsymbol{\lambda}_k$.

(ii) Wir verwenden $\boldsymbol{\lambda}_{k+1}$, um $\boldsymbol{\mu}_{k+1}$ zu berechnen. Die Berechnung realisieren wir durch die Lösung der Normalgleichung (10.63).

Bevor wir uns mit der Fehlerabschätzung und der Laufzeit des Algorithmus 10.19 beschäftigen, diskutieren wir noch das modifizierte Verfahren der konjugierten Gradienten. Wir wollen das lineare Gleichungssystem

$$\mathbf{A}_*\boldsymbol{\lambda}_{k+1} = \mathbf{v}^{(3)} - \mathbf{B}\boldsymbol{\mu}_k \tag{10.66}$$

iterativ mit dem Startwert $\boldsymbol{\lambda}_k$ lösen. Im Vergleich zu dem gewöhnlichen Verfahren der konjugierten Gradienten besteht die Modifikation darin, dass die Lösung $\boldsymbol{\lambda}_k$ in jedem Schritt in dem Unterraum $V_{\mathbf{Q}_1}$ liegen soll. Wir fassen die Vorgehensweise nun zusammen.

10.20 Algorithmus: Wir starten mit den Werten $\boldsymbol{\lambda}_k \in \mathbb{R}^N$, $\mathbf{A}_*, \mathbf{T} \in \mathbb{R}^{N\times N}$ und $\mathbf{v}^{(3)} \in \mathbb{R}^N$ aus Schritt $(5)(i)$ des Algorithmus 10.19. Dann liefern die folgenden Schritte eine Lösung des linearen Gleichungssystems (10.66).

(1) Sei $\boldsymbol{\lambda}_{k,0} = \boldsymbol{\lambda}_k$. Wir berechnen

$$\mathbf{r}_0 = \left(\mathbf{v}^{(3)} - \mathbf{B}\boldsymbol{\mu}_k\right) - \mathbf{A}_*\boldsymbol{\lambda}_{k,0} \text{ sowie } \widetilde{\mathbf{r}}_0 = \mathbf{T}\mathbf{r}_0$$

und setzen $\mathbf{t}_0 = \widetilde{\mathbf{r}}_0$.

(2) Solange $\|\widetilde{\mathbf{r}}_k\|_2 > tol$ gilt, führen wir folgende Schritte für $j = 1, 2, \ldots$ durch.

(i)
$$\boldsymbol{\lambda}_{k,j+1} = \boldsymbol{\lambda}_{k,j} + \frac{\widetilde{\mathbf{r}}_j^t\widetilde{\mathbf{r}}_j}{\mathbf{t}_j^t\mathbf{A}_*\mathbf{t}_j}\mathbf{t}_j$$

(ii)
$$\mathbf{r}_{j+1} = \left(\mathbf{v}^{(3)} - \mathbf{B}\mu_k\right) - \mathbf{A}_*\boldsymbol{\lambda}_{k,j+1}$$

(iii)
$$\widetilde{\mathbf{r}}_{j+1} = \mathbf{T}\mathbf{r}_{j+1}$$

(iv)
$$\mathbf{t}_{j+1} = \widetilde{\mathbf{r}}_{j+1} - \frac{\widetilde{\mathbf{r}}_{j+1}^t\widetilde{\mathbf{r}}_{j+1}}{\widetilde{\mathbf{r}}_j^t\widetilde{\mathbf{r}}_j}\mathbf{t}_j$$

(3) Wir setzen $\boldsymbol{\lambda}_{k+1} = \boldsymbol{\lambda}_{k,j}$ für den Index $j \geq 0$, bei dem der Algorithmus im Schritt (2) abgebrochen wird.

Wir starten den Algorithmus 10.20 grundsätzlich mit einem $\boldsymbol{\lambda}_k \in V_{\mathbf{Q}_1}$. Offensichtlich ist durch die Projektion von $\mathbf{r}_j$ stets $\widetilde{\mathbf{r}}_j \in V_{\mathbf{Q}_1}$ und damit liegt auch $\mathbf{t}_{j+1}$ als Linearkombination von $\widetilde{\mathbf{r}}_{j+1}$ und $\mathbf{t}_j$ in $V_{\mathbf{Q}_1}$. Insgesamt ist also jedes neu berechnete $\boldsymbol{\lambda}_{k,j+1}$ in $V_{\mathbf{Q}_1}$.
Wie bereits angekündigt, folgen nun die Laufzeitanalysen der Algorithmen 10.19 und 10.20. Seien $maxiter_1$ die maximale Anzahl der Iterationen des Algorithmus 10.19 und $maxiter_2$ die maximale Anzahl der Iterationen des modifizierten Algorithmus der konjugierten Gradienten 10.20. Wir betrachten die einzelnen Schritte des Algorithmus 10.19.

(1) Die modifizierte QR–Zerlegung benötigt insgesamt $2N3^2 = 18N$ Rechenoperationen. Der erforderliche Speicherplatz liegt bei $3N + 9$.

(2) Die symmetrische Matrix $\mathbf{T}$ bestimmen wir nicht explizit, denn es gilt für ein $\mathbf{v} \in \mathbb{R}^d$

$$\mathbf{T}\mathbf{v} = \left(\mathbf{I}_N - \mathbf{Q}_1\mathbf{Q}_1^t\right)\mathbf{v} = \mathbf{v} - \mathbf{Q}_1\left(\mathbf{Q}_1^t\mathbf{v}\right).$$

Das Produkt $\mathbf{x} = \mathbf{Q}_1^t\mathbf{v} \in \mathbb{R}^3$ können wir in $O(N)$ Zeit berechnen. Die Bestimmung von $\mathbf{Q}_1\mathbf{x} \in \mathbb{R}^N$ ist ebenfalls in $O(N)$ zu bewältigen. Insgesamt benötigen wir für die Berechnung des Matrix-Vektor-Produkts $\mathbf{T}\mathbf{v} \in \mathbb{R}^N$ $O(N)$ Zeit und $O(N)$ Speicherplatz.

(3) Die Startwertbestimmung für $\boldsymbol{\lambda}$ können wir wie in (2) beschrieben realisieren und daher in $O(N)$ Zeit durchführen. Allerdings benötigen wir weitere N Speicherstellen für$\boldsymbol{\lambda}_0$.

(4) Wir berechnen das Produkt $\mathbf{B}^t\mathbf{B}$ in $18N$ Rechenschritten. Für die Bestimmung der rechten Seite wenden wir die Fast-Multipole Methode an und erreichen eine Komplexität von $O(N^{4/3})$. Die Lösung des 3×3–Gleichungssystems lassen wir wegen des vernachlässigbaren Aufwands unberücksichtigt.

(5) Die nachfolgenden Punkte führen wir höchstens $maxiter_1$–mal durch.

(i) Die Lösung des linearen Gleichungssystems (10.66) durch Algorithmus 10.20 benötigt insgesamt $(2maxiter_2 + 1)$ Matrix-Vektor Produkte, die wir mit Hilfe der Fast-Multipole Methode berechnen können sowie $(7maxiter_2 + 2)$ Matrix-Vektor oder Vektor-Vektor-Produkte der Komplexität $O(N)$.

(ii) Genauso wie in Punkt (4) müssen wir in $O(N^{4/3})$ ein Matrix-Vektor Produkt für die Berechnung der linken Seite des Gleichungssystems bestimmen und das 3×3–Gleichungssystem lösen.

Wir erhalten eine Gesamtkomplexität von

$$\begin{aligned} & O_{Gesamt} \\ =& O\left(N^{4/3} + N + maxiter_1\,(2maxiter_2 + 1)\,N^{4/3} + (7maxiter_2 + 2)N\right) \\ =& O\left(N^{4/3}\,(2maxiter_1 maxiter_2 + maxiter_1 + 1)\right) \\ =& O\left(N^{4/3}\right). \end{aligned}$$

Da gilt $maxiter_1, maxiter_2 \ll N$, wobei die Konstanten $maxiter_1, maxiter_2$ fest gewählt und unabhängig von N sind, können wir von einer Komplexität von $O(N^{4/3})$ ausgehen.

10.4 Gesamtalgorithmus zur Flächenberechnung

Sei

$$K = \{\mathbf{v}_j \mid j = 0, \ldots, N-1\} \subset \mathbb{R}^3$$

die zu fittende Fläche. Wir verwenden die Bezeichnungen aus Abschnitt 10.1. Wir wenden Algorithmus 10.1 an und lösen bei jedem Durchlauf das lineare Gleichungssystem $(2)(a)$ mit Algorithmus 10.19, wobei wir für jedes Matrix-Vektor Produkt die Single-Level-Fast-Multipole Methode 10.13 benutzen. Die Einzelheiten fassen wir in dem nachfolgenden Algorithmus zusammen, gehen dabei aber nur auf diejenigen Punkte ein, die sich im Vergleich zu Algorithmus 10.1 verändert haben.

10.21 Algorithmus: Seien $K \subset \mathbb{R}^3$ die Fläche, $\zeta \geq 0$ der Glättungsparameter und $\varepsilon > 0$ die Messgenauigkeit. Wir verwenden als radiale Basisfunktion den Thin-Plate Spline

$$\phi_{TPS}(\mathbf{x}) = \|\mathbf{x} - \mathbf{x}_*\|_2^2 \log \|\mathbf{x} - \mathbf{x}_*\|_2.$$

Sei ferner $W = \bigcup_{k=1}^{T_0} W_k$ mit $T = t^3$ für ein $t \in \mathbb{N}$ die bounding Box von K. Gemäß [Be92] setzen wir

$$t = 2^{\widetilde{m}} \text{ mit } \widetilde{m} \approx \frac{\log_2 N}{4}.$$

Dann liefert uns das folgende Verfahren eine geglättete Fläche zu den Daten aus K.

(1) Wie in Algorithmus 10.1 bestimmen wir für jedes W_k für $k = 1, \ldots, T_0$ das Zentrum $\mathbf{v}_{k*} \in W_k$ und ermitteln die Mengen $TPS_0(W_k)$ für $k = 1, \ldots, T_0$ sowie V_0^*.

(2) Wir führen die Schritte (a) bis (d) für $\beta = 1, 2, \ldots$ solange durch, bis eine der Abbruchbedingungen in (c) erfüllt ist.

(a) Wir lösen das lineare Gleichungssystem (10.57) mit dem Verfahren 10.19 unter Verwendung der Single-Level-Fast-Multipole Methode 10.13. Wir erhalten als Ergebnis die Menge der Gewichte $\lambda^{(1)}, \ldots, \lambda^{(T_\beta)} \in \mathbb{R}$.
Für die Single-Level-Fast-Multipole Methode bestimmen wir eine zweite Raumpartition der bounding Box W mit

$$W = \bigcup_{k=0}^{\widetilde{T}_\beta} \widetilde{W}_k.$$

Wir wählen dabei $\widetilde{t} = t^{m-2}$ und erhalten eine gröbere Partition, wobei zu jedem $\widetilde{W}_k$ Indizes $k_1, \ldots, k_4 \in \{1, \ldots T_\beta\}$ existieren, so dass $W_{k_j} \subset \widetilde{W}_k$ für $k_j \in \{k_1, \ldots, k_4\}$ ist.

(b) Wir werten s_β an den Punkten $\mathbf{v}'_0, \ldots, \mathbf{v}'_{N-1}$ aus. Hierfür ordnen wir die Punkte $\mathbf{v}_j \in K$ der Partition $W = \bigcup_{k=1}^{\widetilde{T}_\beta} \widetilde{W}_k$ zu, das heißt wir bestimmen für jedes $\mathbf{v}_j \in K$ den entsprechenden Teilquader $\widetilde{W}_k$. Dann wenden wir erneut die Single-Level-Fast-Multipole Methode 10.13 an. Als Ergebnis erhalten wir die Menge $\{s_\beta(\mathbf{v}'_j) \mid j = 0, \ldots, N-1\}$.

(c) Dieser Punkt und seine Unterpunkte (i) bis (iii) werden genauso durchgeführt wie in Algorithmus 10.1.

(d) Auch die Bestimmung der neuen Menge $V_{\beta+1}^*$ erfolgt wie in Algorithmus 10.1.

(3) Als Ergebnis erhalten wir für ein $\beta > 0$ eine Fläche

$$s = P + \sum_{k=1}^{T_\beta} \lambda^{(k)} \phi_{TPS}(\| \cdot - \mathbf{v}_j^+ \|_2).$$

Eine Laufzeituntersuchung des Algorithmus 10.21 ist nur unter diversen Annahmen über die Lage und Verteilung der Punkte aus K möglich und

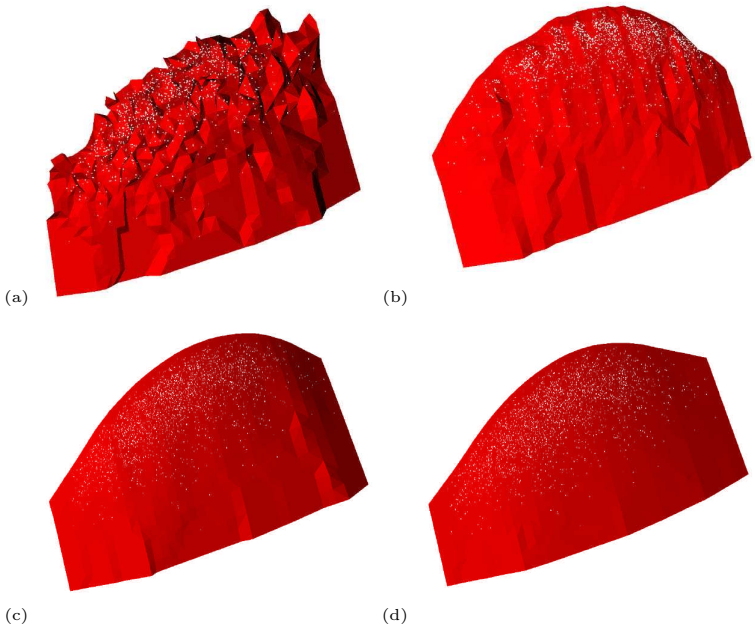

Abbildung 51: (a) Originalfläche (b) Geglättete Fläche mit $\zeta = 0.1$ (c) Geglättete Fläche mit $\zeta = 1$ (d) Geglättete Fläche mit $\zeta = 10$

wir wollen an dieser Stelle darauf verzichten.
Statt dessen betrachten wir einige Ergebnisse der Implementierung mit einer Kontur von *Werth Messtechnik GmbH*.
Wir haben bereits gesehen, dass eine Komplexität von mindestens $O(N^{4/3})$ zu erwarten ist, diese aber durch die Reduktion der Punkte abgemildert werden kann.
Wir arbeiten mit einer Testkontur aus ca. 320.000 Punkten, einer Genauigkeit von 10^{-6} und einem Schwellenwert von 10^{-5}. Das Verfahren wurde in *C* unter Visual Studio 6.0 implementiert, die Tests wurden auf einem Intel Celeron 2,4 GHz PC mit 1024 MB RAM unter Windows XP Professional durchgeführt.
Diese Anwendung wurde bis zum jetzigen Zeitpunkt nicht in die Softwarelandschaft von *Werth Messtechnik* integriert. Aus diesem Grund und wegen Schwierigkeiten mit der Speicherverwaltung für Konturen, welche größer als 64.000 Punkte sind, haben wir mit bis zu 50.000 Punkten getestet. Ein Ausschnitt der gefitteten Fläche und die Ergebnisse der Glättung können wir in Abbildung 51 sehen.
Zusätzlich vergleichen wir die Laufzeit des Algorithmus 10.21 mit der Laufzeit des Algorithmus 10.1 ohne Verwendung der Beschleunigungsmaßnahmen.

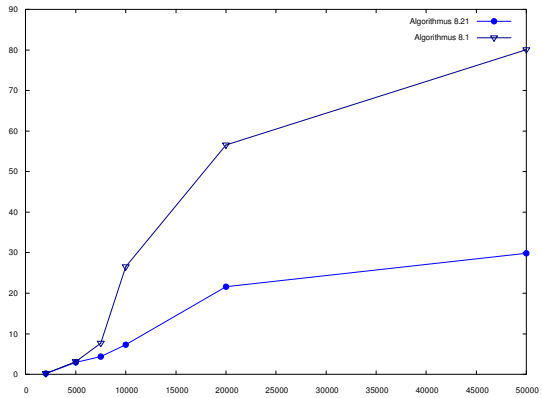

Abbildung 52: Vergleich der Laufzeiten der Algorithmen 10.21 und 10.1

Punkte	ϵ	Iterationen	$\widetilde{m}$	T	Alg. 10.21	Alg. 10.1
2000	10^{-3}	1	direkt	2000	0.176 sec	0.215 sec
5000	10^{-3}	1	direkt	5000	2.96 sec	3.176 sec
7500	10^{-3}	2	4	4096	4.406 sec	7.657 sec
10000	10^{-3}	2	4	4069	7.268 sec	26.578 sec
20000	10^{-3}	4	4	4096	21.587 sec	56.547 sec
50000	10^{-3}	5	4	4096	29.861 sec	80.025 sec

Einen graphischen Vergleich der Laufzeiten gibt uns Abbildung 52.
Wir erkennen, dass die Beschleunigung des Algorithmus 10.21 gegenüber der direkten Methode sehr deutlich ausfällt. Der modifizierte Algorithmus ist etwa doppelt so schnell, wobei sich dieser Faktor bei größeren Punktmengen vergrößern dürfte.
Die Laufzeit des Algorithmus 10.21 können wir in diesem Beispiel auch nicht pauschal als quadratisch, schneller oder langsamer einstufen. So ist es durchaus möglich, eine Punktwolke mit mehr als 100.000 Punkten in etwa derselben Zeit zu fitten wie die Teilkontur mit 10.000 Punkten. Dies geschieht dann, wenn die erste Iteration bereits ein Ergebnis innerhalb der Toleranz liefert. In diesem Fall benutzen wir nur zur Auswertung die tatsächliche Menge von 100.000 Punkten und rechnen ansonsten zum Beispiel mit 4096 Zentren.

10.5 Ein lokales Glättungsverfahren

In manchen Situationen liegen die Daten einer Fläche oder Kontur bereits geordnet vor. Dies ist bei *Werth Messtechnik GmbH* zum Beispiel dann der Fall, wenn eine Fläche oder ein Werkstück mit Hilfe des *Werth TomoScope*® gemessen wurde. Hierbei handelt es sich um einen Röntgen-Computer-Tomographen für Koordinatenmessungen. Dieses Messgerät liefert die $3D$–Daten bereits aufbereitet, diese sind trianguliert.
Für die Glättung bieten sich dann, neben den klassischen Subdivisionverfahren, erneut die radialen Basisfunktionen an. Der Nachteil der Subdivisionverfahren ist, dass diese den nicht unerheblichen Konturumfang zum Teil deutlich erhöhen. Unsere Ansatz bietet zwar ebenfalls die Möglichkeit, die Punktdichte zu erhöhen, für eine Glättung muss dies aber nicht zwingend der Fall sein.
Was verstehen wir unter dem Begriff Triangulierung? Darüber gibt uns die folgende Definition Auskunft.

10.22 Definition: Sei $K = \{\mathbf{v}_j \mid j = 0, \ldots, N-1\} \subset \mathbb{R}^d$ für $d \geq 2$ eine Fläche. Eine *Triangulierung* von K ist eine endliche Menge $\mathcal{T} \subset \{0, \ldots, N-1\}^3$ von Tripeln $\tau = (\tau(1), \tau(2), \tau(3))$, für die gilt.

(1) Für ein $\tau \in \mathcal{T}$ existiert eine Abbildung

$$\tau \mapsto \Delta_\tau \in \mathbb{R}^d.$$

Dabei ist

$$\Delta_\tau = \{\alpha_1 \mathbf{v}_{\tau(1)} + \alpha_2 \mathbf{v}_{\tau(2)} + \alpha_3 \mathbf{v}_{\tau(3)} \mid \alpha_j \geq 0, \sum_{j=1}^{3} \alpha_j = 1\}$$

und die Vektoren $\mathbf{v}_{\tau(2)} - \mathbf{v}_{\tau(1)}$ und $\mathbf{v}_{\tau(3)} - \mathbf{v}_{\tau(1)}$ sind linear unabhängig.

(2) Es gilt für $\tau, \tau' \in \mathcal{T}$ mit $\tau \neq \tau'$

$$\Delta_\tau \cap \Delta_{\tau'} = \begin{cases} \text{Leer} & \emptyset \\ \text{Seite} & \alpha_j \mathbf{v}_{\tau(j)} + \alpha_k \mathbf{v}_{\tau(k)} = \alpha'_l \mathbf{v}_{\tau'(l)} + \alpha'_i \mathbf{v}_{\tau'(i)} \\ \text{Ecke} & \alpha_j \mathbf{v}_{\tau(j)} = \alpha'_k \mathbf{v}_{\tau'(k)} \end{cases}$$

für $j, k, l, i \in \{1, 2, 3\}$. Dabei sind α_j die Koeffizienten von Δ_τ und α'_k die Koeffizienten von $\Delta_{\tau'}$.

(3) Es gilt

$$\bigcup_{\tau\in\mathcal{T}}[\Delta_\tau] \subseteq [K].$$

10.23 Bemerkung:

(1) Bedingung (1) der Definition 10.22 besagt, dass eine Triangulierung aus nicht entarteten Dreiecken besteht. Dabei wird jedes Dreieck durch genau drei Punkte gebildet.

(2) Die zweite Bedingung der obigen Definition stellt sicher, dass sich zwei verschiedene Dreiecke entweder gar nicht, entlang einer Seite oder in einer Ecke schneiden.

Offensichtlich ist eine Triangulierung nach Definition 10.22 nicht eindeutig, es sind bereits bei vier Punkten in allgemeiner Lage zwei verschiedene Triangulierungen möglich. Es gibt verschiedene Kriterien, wie zum Beispiel das Kriterium der kürzeren Diagonalen oder das Min-Max-Winkelkriterium, die sich mit der Güte der einzelnen Dreiecke Δ_j beschäftigen.
In der Literatur wird oft die so genannte *Delaunay-Triangulierung* als optimal genannt. Eine Delaunay-Triangulierung stellt den zu den Voronoi-Diagrammen aus Abschnitt 8.4 dualen Graphen dar. Die wesentliche Eigenschaft einer Delaunay-Triangulierung ist, dass die minimalen Winkel der Dreiecke Δ_j maximiert werden, es werden also möglichst wenige flache Dreiecke erzeugt.
Uns liegt eine Triangulierung als Ergebnis einer Abtastung mit einem *Werth TomoScope*® vor. Die Ausgangsbasis für das Glättungsverfahren umfasst die Kontur

$$K = \{\mathbf{v}_j \mid j = 0, \ldots, N-1\} \subset \mathbb{R}^d$$

und die Menge der Dreiecke bzw. der Verweise

$$\tau \in \mathcal{T} \subset \{0, \ldots, N-1\}^3$$

mit den Informationen über die Triangulierung. Das Ziel des Verfahrens ist das Erzeugen einer geglätteten Kontur

$$\widetilde{K} = \{\widetilde{\mathbf{v}}_j \mid j = 0, \ldots, N-1\} \subset \mathbb{R}^d.$$

Bevor wir die Vorgehensweise erläutern, benötigen wir einen weiteren Begriff.

10.24 Bezeichnung: Sei $K \subset \mathbb{R}^d$ eine Kontur und $\mathfrak{T}$ eine Triangulierung von K. Sei $\mathbf{v}_j \in K$ fest. Seien $\tau_0, \ldots, \tau_L \in \mathfrak{T}$, so dass $\mathbf{v}_j \in \Delta_{\tau_j}$ für alle $j = 0, \ldots, L$ gilt. Wir bezeichnen alle Punkte $\mathbf{v}_l$, für die

$$\mathbf{v}_l \neq \mathbf{v}_j \text{ und } \mathbf{v}_l \in \Delta_{\tau_j}$$

gilt, als *Nachbarn 0–ter Stufe von* $\mathbf{v}_j$.
Entsprechend können wir rekursiv die Menge *der Nachbarn n–ter Stufe* $S_n(\mathbf{v}_j)$ von $\mathbf{v}_j$ angeben. Es gilt

$$\begin{aligned} S_n(\mathbf{v}_j) =& \{\mathbf{v} \in K \mid \mathbf{v} \notin \bigcup_{l=0}^{n-1} S_l(\mathbf{v}_j) \text{ und} \\ & \mathbf{v} \text{ ist Nachbar 0–ter Stufe zu einem } \mathbf{u} \in S_{n-1}(\mathbf{v}_j)\}. \end{aligned}$$

Wollen wir eine Fläche glätten, so gehen wir wie folgt vor. Wir starten mit der ursprünglichen Kontur K und einer Menge $\widetilde{K} = \emptyset$. Wir bestimmen sukzessive zu Punkten $\mathbf{v}_j \in K$ alle Nachbarn der Stufen $0, \ldots, J$, für ein global bestimmtes $J > 0$. Nachbarn der ersten $\widetilde{J} < J$ Stufen werden geglättet und der Menge $\widetilde{K}$ hinzugefügt, während sie aus K gelöscht werden. Dies führen wir solange durch, bis keine Punkte mehr in K enthalten sind. Dabei schreiben wir für die Menge der Nachbarn bis zur J–ten Stufe von $\mathbf{v}_j$

$$M_{j,J} = \{\mathbf{v}_i \in K \mid \mathbf{v}_i \text{ Nachbar } k\text{–ter Stufe von } \mathbf{v}_j \text{ für } 0 \leq k \leq J\}$$

und setzen $m_{j,J} = |M_{j,J}|$. Wir beschränken uns auf die Betrachtung der Nachbarn der ersten J Stufen, da die durch den *Werth TomoScope*® abgetasteten Daten äquidistant sind. Wir verwenden die Bezeichnung aus Abschnitt 10.1 und stellen das lineare Gleichungssystem

$$\mathbf{Mf} = \left[\begin{array}{c|c} \mathbf{A} + \zeta \mathbf{I}_{m_{j,J}} & \mathbf{B} \\ \hline \mathbf{B}^t & \mathbf{0} \end{array}\right] \left[\begin{array}{c} \boldsymbol{\lambda} \\ \hline \boldsymbol{\mu} \end{array}\right] = \left[\begin{array}{c} \mathbf{v}^{(d)} \\ \hline \mathbf{0} \end{array}\right] = \mathbf{b} \tag{10.67}$$

auf. Dabei ist $\zeta > 0$ der Glättungsparameter und wir erhalten eine Interpolation der Daten für $\zeta \to 0$. Darüber hinaus wählen wir J so, dass das Gleichungssystem (10.67) klein genug bleibt, um es effizient direkt lösen zu können.
Als Lösung von (10.67) erhalten wir die Parametervektoren $\boldsymbol{\lambda}, \boldsymbol{\mu}$ und damit den Thin-Plate Spline

$$s = P + \sum_{k=0}^{m_{j,J}-1} \lambda^{(k)} \phi_{TPS}(\| \cdot - \mathbf{v}'_{j_k} \|_2)$$

für $M_{j,J} = \{\mathbf{v}_{j_0}, \ldots, \mathbf{v}_{j_{m_{j,J}-1}}\}$. Wir werten s an den Stellen $\mathbf{v}_{j_i} \in M_{j,J}$ aus und erhalten neue Punkte, die wir mit $\widetilde{\mathbf{v}}_{j_i}$ bezeichnen. Die Konturpunkte der ersten $\widetilde{J} \leq J$ Stufen fügen wir der geglätteten Kontur $\widetilde{K}$ hinzu und löschen diese aus K. Anschließend fahren wir mit dem nächsten, noch nicht geglätteten, Punkt $\mathbf{v}_j \in K$ fort.
Den Algorithmus formalisieren wir wie folgt .

10.25 Algorithmus: Sei $K \subset \mathbb{R}^d$ eine Kontur und $\mathcal{T}$ eine Triangulierung von K. Sei $\zeta > 0$ der Glättungsparameter. Dann liefert uns das folgende Verfahren eine geglättete Fläche $\widetilde{K}$.

(1) Wir sortieren die Einträge von $\mathcal{T}$ aufsteigend. Zunächst einmal können wir die Indizes innerhalb eines Eintrags $\tau \in \mathcal{T}$ aufsteigend sortieren, so dass gilt

$$\tau(1) < \tau(2) < \tau(3).$$

Dann setzen wir

$$\begin{aligned} \tau < \tau' \Leftrightarrow\ & \tau(1) < \tau'(1) \text{ oder} \\ & \tau(1) = \tau'(1) \text{ und } \tau(2) < \tau'(2) \text{ oder} \\ & \tau(1) = \tau'(1) \text{ und } \tau(2) = \tau'(2) \text{ und } \tau(3) < \tau'(3). \end{aligned}$$

Wegen der Eindeutigkeit der Einträge $\tau(j)$ ist eine solche Sortierung möglich. Eine auf diese Weise sortierte Liste $\mathcal{T}$ ermöglicht uns eine effiziente Verarbeitung und eine schnellere Suche nach benachbarten Konturpunkten. In der Literatur wird eine solche Sortierung als lexikographische Ordnung bezeichnet.

(2) Wir bestimmen die Parameter J und $\widetilde{J}$. Hierfür sei ν die höchste Anzahl der Nachbarn 0–ter Stufe eines Konturpunkts. Aus Effizienzgründen wollen wir nicht mehr als t_{eff} Punkte auf einmal verarbeiten, daher bestimmen wir $J \in \mathbb{N}$ so, dass

$$\nu^J \leq t_{eff}$$

ist. Für $\widetilde{J}$ gilt dann

$$\widetilde{J} = \begin{cases} J-1 & \text{für } J \leq 2 \\ J-2 & \text{sonst.} \end{cases}$$

Dabei ist ν^J die größte Anzahl der Nachbarn der Stufen 0 bis J.

(3) Wir führen folgende Schritte solange durch, bis $|K| < t_{eff}$ gilt.

(i) Falls $|K| \geq t_{eff}$ ist, wählen wir einen Punkt $\mathbf{v}_j \in K$, welcher noch nicht als geglättet gekennzeichnet wurde, und bestimmen die Menge $M_{j,J}$. Dabei wird $j \leq N-1$ als minimaler Index der nicht geglätteten Punkte aus K gewählt.
Anderenfalls setzen wir $M_{j,J} = K$.

(ii) Wir lösen das lineare Gleichungssystem (10.67) und erhalten die Parametervektoren $\boldsymbol{\lambda}$ und $\boldsymbol{\mu}$.

(iii) Wir werten den Thin-Plate Spline an allen Stellen $\mathbf{v} \in M_{j,J}$ aus und bekommen die Menge $\widetilde{M}_{j,J}$ der geglätteten Punkte. Wir bezeichnen die geglätteten Punkte mit $\widetilde{\mathbf{v}}$.

(iv) Für alle $\widetilde{\mathbf{v}}_k \in M_{j,J}$ führen wir folgende Fallunterscheidung durch.

(a) $\mathbf{v}_k$ ist ein Nachbar l–ter Stufe von $\mathbf{v}_j$ für $0 \leq l \leq \widetilde{J}$. Dann löschen wir $\mathbf{v}_k$ aus K und fügen es zu $\widetilde{K}$ hinzu.

(b) Ist $\mathbf{v}_k$ ein Nachbar l–ter Stufe von $\mathbf{v}_j$ für $\widetilde{J} < l \leq J$, so wird $\mathbf{v}_k$ korrigiert. Dabei gilt $\mathbf{v}_k^{neu} = \frac{1}{2}(\widetilde{\mathbf{v}}_k + \mathbf{v}_k)$ und wir ersetzen $\mathbf{v}_k$ in K durch $\mathbf{v}_k^{neu}$. Gleichzeitig kennzeichnen wir $\mathbf{v}_k$ als geglättet. Haben wir in (i) $M_{j,J} = K$ gesetzt, so werden alle Punkte $\mathbf{v}_k$ aus K gelöscht und alle geglätteten Punkte $\widetilde{\mathbf{v}}_k$ der Menge $\widetilde{K}$ hinzugefügt.

(4) Das Ergebnis ist die Menge $\widetilde{K} = \{\widetilde{\mathbf{v}}_j \mid j = 0, \ldots, N-1\} \subset \mathbb{R}^d$ der geglätteten Punkte.

Nun führen wir eine Laufzeitanalyse des Algorithmus 10.25 durch.
Um die Menge $\mathcal{T}$ zu sortieren, benutzen wir den Sortieralgorihmus *Mergesort* und erreichen eine Komplexität von $O(|\mathcal{T}| \log |\mathcal{T}|)$. Da für eine Delaunay-Triangulierung $|\mathcal{T}| \leq 2N - 5$ gilt, können wir von einer Komplexität von $O(N \log N)$ ausgehen.
Für die Bestimmung der Parameter J bzw. $\widetilde{J}$ ist es ausreichend, einmal die Triangulierung τ zu durchsuchen. Wir erreichen daher eine Laufzeit von $O(N)$.
Wir verwenden für unsere Betrachtungen die Größe ν als maximale Anzahl der Nachbarn 0–ter Stufe und bestimmen damit einen worst case für die Komplexität.

Es gilt

$$J = \left\lfloor \frac{\log t_{eff}}{\log \nu} \right\rfloor .$$

Sei $t_{eff} = \frac{N}{\beta}$ für ein $\beta > 1$. Wir nehmen im Folgenden an, dass $J > 2$ sowie $\nu^J = t_{eff}$ gilt, dann erhalten wir für die Anzahl der geglätteten Punkte pro Durchlauf

$$\nu^{\widetilde{J}} = \frac{\nu^J}{\nu^2} = \frac{t_{eff}}{\nu^2} = \frac{N}{\beta\nu^2}.$$

Wir können also durchschnittlich $\frac{N}{\beta\nu^2}$ Punkte pro Durchlauf des Schrittes (3) in Algorithmus 10.25 glätten und benötigen daher $\beta\nu^2$ Durchläufe.
Wir wenden uns jetzt Schritt (3) zu.
Für die Bestimmung der Mengen $M_{j,J}$ verwenden wir die sortierte Menge $\mathfrak{T}$. Die Komplexität der Suche nach sämtlicher Nachbarn aller J Stufen beträgt dann $O(N)$.
Die Punkte (ii) und (iii) behandeln wir gemeinsam. Wir lösen das lineare Gleichungssystem (10.67) in $O(t_{eff}^2)$ Zeit, die restlichen Operationen, wie das Auswerten des Thin-Plate Splines, haben ebenfalls eine Komplexität von höchstens $O(t_{eff}^2)$.
Schritt $(3)(iv)$ ist von N weitgehend unabhängig, da wir aber unter Umständen die Mengen K bzw. $\widetilde{K}$ durchsuchen müssen, schätzen wir auch diese Komplexität mit $O(N)$ nach oben ab.
Wir erhalten eine Gesamtlaufzeit von

$$\begin{aligned} O_{ges} &= O\left(N \log N + \beta\nu^2 \left(N + \frac{N^2}{\beta^2}\right)\right) \\ &= O\left(N \log N + \beta N + \frac{N^2}{\beta}\right) \\ &= O\left(N \log N + t_{eff}\beta^2 + t_{eff}N\right) \\ &= O\left(N \log N + t_{eff}\left(N + \beta^2\right)\right) \\ &= O\left(t_{eff}\left(N + \beta^2\right)\right). \end{aligned}$$

Für welche Werte von t_{eff} erreichen wir die optimale Komplexität? Wir verwenden unsere Annahme $t_{eff} = \frac{N}{\beta}$ und erhalten

$$f_N(\beta) = \frac{N}{\beta}\left(N + \beta^2\right).$$

Dann ist

$$0 = f_N'(\beta) = -\frac{N^2}{\beta^2} + N,$$

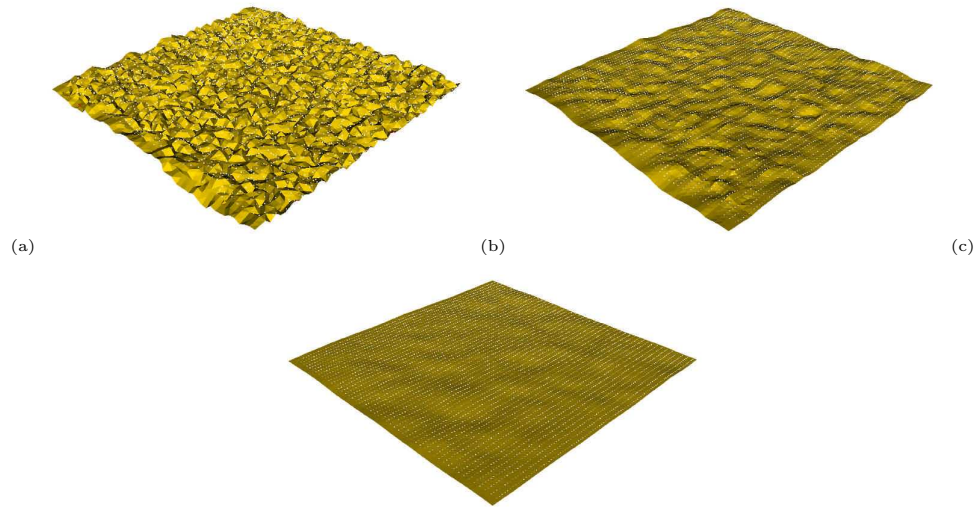

Abbildung 53: (a) Originalfläche (b) Geglättete Fläche mit $\zeta = 0.1$ (c) Geglättete Fläche mit $\zeta = 5$

also ist $\beta_{opt} = \sqrt{N}$ und damit $t_{eff,opt} = \sqrt{N}$. Damit ergibt sich eine Gesamtkomplexität von

$$O_{ges} = O\left(\sqrt{N}\left(N + (\sqrt{N})^2\right)\right) = O\left(N^{3/2}\right).$$

Demnach beträgt die asymptotische Komplexität $O(N^{3/2})$.
Es lagen uns bis zuletzt keine triangulierten Testdaten vor, daher haben wir den Algorithmus an einer Testkontur getestet. Die Testkontur hat eine Größe von 10.000 Punkten, aus diesem Grund haben wir einen Schwellenwert $t_{eff} = 1000$ verwendet. Dies ergab mit $\nu = 8$ den Wert $J = 4$. Die aus der Anwendung des Algorithmus 10.25 resultierende Fläche ist in Abbildung 53 dargestellt.
Wir können mit der lokalen Methode Glättungen von Flächen, zumindest theoretisch, effizienter und schneller durchführen, als mit der globalen Methode. Allerdings ist momentan noch unklar, wie sich die Glättung an den Stellen verhält, die zweimal geglättet wurden. Darüber hinaus fehlen uns Erfahrungen mit tatsächlichen Konturen eines *Werth TomoScope*®.

Literatur

[And99] Anderson, E., et al: **LAPACK Users Guide, Third Edition**, SIAM, 1999

[Be92] Beatson, R. K., Newsam, G. N.: **Fast evaluation of radial basis functions: I**, Computers Math. Applic., Vol. 24, No. 12, 1992

[dBe00] de Berg, M., van Krefeld, M., Overmars, M., Schwarzkopf, O.: **Computational Geometry, Algorithms and Application, 2nd Edition**, Springer, 2000

[Bo79] Bookstein, F.: **Fitting conic sections to scattered data**, Computer Graphics and Image Processing, 1979

[dB78] de Boor, C.: **A Practical Guide to Splines**, Springer-Verlag, 1978

[Bro65] Broyden, C. G.: **A class of methods for solving nonlinear simultaneous equations**, Math. Comp 19, 1965

[Bu03] Buhmann, M. D.: **Radial Basis Functions**, Cambridge University Press, 2003

[Ch79] Chojnacki, W., et al: **Revisiting hartleys normalized eight-point algorithm**, IEEE Trans. on Pattern Analysis and Maschine Intelligence, 1979

[ChNe03] Christoph, R., Neumann, H. J.: **Multisensor-Koordinaten-messtechnik**, Die Bibliothek der Technik, Band 248, 2003

[Ch92] Chui, C. K.: **An Introduction to Wavelets**, Academic Press, 1992

[Co89] Cohen, E., O'Dell, C.: **A data dependent parametrization for spline approximation**, Mathematical Methods in CAGD, 1999

[CoDaFe92] Cohen, A., Daubechies, I., Feauveau, J.-C.: **Biorthogonal bases of compactly supported wavelets**, Comm. Pure and Appl. Math. 45, 1992

[CuSch47] Curry, H. B., Schoenberg, I. J.: **On spline distributions and their limits: The Polya distribution functions**, Bull. Amer. Math. Soc., 53, 1947

[Da99] Daubechies, I.: **Ten Lectures on Wavelets**, SIAM Series in Applied Mathematics, 1999

[Do92] Donoho, D. L.: **De-noising by soft-tresholding**, Technical Report 409, Stanford University, 1992

[DoJo92] Donoho, D. L., Johnstone, I. M.: **Ideal spatial adaption by wavelet shrinkage**, Technical Report 400, Stanford University, 1992

[Ep76] Epstein, M.: **On the influence of parametrization in parametric interpolation**, SIAM J Numer. Analysis, 1976

[Fa02] Farin, G., Hoschek, J., Kim, M.-S.: **Handbook of Computer Aided Geometric Design**, North-Holland, 2002

[Fi99] Fitzgibbon, A., Pilu, M., Fischer, R.: **Direct least square fitting of ellipses**, IEEE Trans. on Pattern Analysis and Maschine Intelligence, 1999

[Gae96] Gärtner, B.: **Algorithmische Geometrie**, Vorlesung FU Berlin, 1996

[Gal03] Gallasch, A.: **Bibliographie zur Längen- und Koordinatenmesstechnik**, BoD GmbH, Norderstedt, 2003

[Go96] Golub, G. H., Van Loan, C. F.: **Matrix Computations**, The John Hopkins University Press, 3rd Edition, 1996

[Kle40] Klein, F.: **Elementary mathematics from an advanced standpoint: geometry**, Dover Publications, 1940

[Knu02] Knuth, D. E.: **Sorting and Searching**, Addison-Wesley, 2002

[Lo98] Louis, A., Maaß, P., Rieder, A.: **Wavelets**, B. G. Teubner, 1998

[Ma98] Mallat, S.: **A Wavelet Tour of Signal Processing**, Academic Press, 1998

[Mat06] The MathWorks Inc.: **Getting startet with MathLab®**, The MathWorks Inc., 2006

[Neu00] Neumann, H. J.: **Koordinatenmesstechnik im industriellen Einsatz**, Die Bibliothek der Technik, Band 203, 2000

[Ni04] Nievergelt, Y.: **Fitting conics of specific types of data**, Linear Algebra and its Applications, 2004

[Pai75] Paige, C., Saunders, M.: **Solution of sparse indefinite systems of linear equations**, SIAM J. Numer. Analysis 12, 1975

[Po89] Poggio, T., Giorosi, F.: **A theory of networks for approximation and learning**, MIT AI Laboratories, 1989

[Pr06] Primbs, M.: **Stabile biorthogonale Spline-Wavelet-Basen auf dem Intervall**, Dissertation, 2006

[Sa01a] Sauer, T.: **Splinekurven- und Flächen in CAGD**, Vorlesung Universität Gießen, 2001

[Sa01b] Sauer, T., **Approximationstheorie**, Vorlesung Universität Gießen, 2001

[Se93] Seidel, R.: **Backward analysis of randomized algorithms**, Algorithms and Combinatorics, Vol. 10, 1993

[Se00] Seul, M., O'Gorman, L., Sammon, M. J.: **Practical Algorithm for Image Analysis**, Cambridge University Press, 2000

[Ste01] Steger, A.: **Diskrete Strukturen 1**, Springer Verlag, 2001

[Sto72] Stoer, J.: **Einführung in die Numerische Mathematik I**, Springer Verlag, 1972

[Str73] Strang, G., Fix, G.: **A Fourier analysis of the finite element variational method**, Constructive aspects of functional Analysis, 1973

[Wah90] Wahba, G.: **Spline models for observational data**, Regional Conference Series in Applied Mathematics, Vol. 59, SIAM, 1990

[We99] Weckemann, A., Gawande, B.: **Koordinatenmesstechnik**, Carl Hanser Verlag, 1999

[WM06] Werth Messtechnik GmbH: **Anwenderhandbuch WinWerth®**, 2006

Erklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbständig und nur mit den angegebenen Hilfsmitteln verfasst habe.

Gießen, den 16.10.2006 _____

Thomas Maresch