universität wien

DIPLOMARBEIT

Titel der Diplomarbeit

Rigorose Padé- und Interpolationsmodelle ein- und mehrdimensionaler Funktionen

angestrebter akademischer Grad

Magister der Naturwissenschaften (Mag.rer.nat.)

Verfasser	Thomas Kammerhuber
Matrikelnummer	0526077
Studienkennzahl	A405
Studienrichtung	Mathematik
Betreuer:	Ao. Univ.-Prof. Dipl.-Ing. Dr. Hermann Schichl

Wien, 3. Mai 2010

Inhaltsverzeichnis

Einleitung

Als Ausgangspunkt dieser Arbeit dient das von Martin Berz und Kyoko Makino in [3] behandelte Taylor Modell. Ursprünglich wurde das Taylor-Modell entwickelt, um praktische Probleme der Physik, insbesondere der nichtlinearen Dynamik zu lösen.

Neben einer Vielzahl von Anwendungsmöglichkeiten werden Taylor-Modelle zur Einschließung des Wertebereichs von Funktionen verwendet. Dargestellt werden diese Einschließungen durch ein Taylorpolynom und ein Restintervall. Mit anderen Worten, es wird versucht, eine Funktion durch ein Polynom zu approximieren, den entstehenden Fehler abzuschätzen und als Fehlerintervall zu schreiben.

Der Grundgedanke dieser Arbeit ist Folgender: Eine Funktion soll nun nicht mehr durch ein Polynom angenähert werden, wie beim Taylor-Modell, sondern durch eine rationale Funktion, also den Quotienten zweier Polynome, um so eine bessere Approximation zu erlangen. Also folgendermaßen:

$$f(x) \approx \frac{p_0 + p_1 x + \cdots + p_n x^n}{1 + q_1 x + \cdots + q_m x^m}.$$

Weiters soll ein zugehöriges Fehlerintervall $\mathbf{I}$ bestimmt werden. Das heißt

$$f(x) \in \frac{p_0 + \mathbf{I} + p_1 x + \cdots + p_n x^n}{1 + q_1 x + \cdots + q_m x^m} \quad \forall x \in \mathbf{x}.$$

Im einführenden Kapitel 1 werden einige Begriffe aus der Intervallarithmetik erklärt. Diese spielt bei der Berechnung von Fehlerintervallen eine wichtige Rolle. Insbesondere bei den Implementationen kann darauf nicht verzichtet werden.
Im darauffolgenden Kapitel werden Slopes definiert und einige Rechenregeln dafür gezeigt. Sowohl Slopes erster Ordnung als auch Slopes zweiter Ordnung werden in den verschiedenen Algorithmen zur Bestimmung von Restintervallen Verwendung finden.

In Kapitel 3 findet man eine Defintion des Taylor-Modells. Der Großteil des Kapitels beschäftigt sich mit der Berechnung eines Taylor-Modells.
Danach wird kurz erklärt, worum es sich bei einer Padé-Approximation handelt. Außerdem wird gezeigt, wie man diese Approximation für ein- und für zweidimensionale Funktionen berechnet.

In den Kapiteln 5, 6 und 7 befinden sich die eigentlichen Resultate dieser Diplomarbeit. Es werden dem Taylor-Modell ähnliche Modelle beschrieben, welche eine Funktion mittels eines Bruches zweier Polynome approximieren. Neben der Bestimmung der Koeffizienten dieser Polynome werden verschiedene Methoden beschrieben, mit denen ein zugehöriges Fehlerintervall berechnet werden kann.

Anschließend folgen eine kurze Beschreibung der benötigten Implementationen, sowie einige Beispiele. Diese werden teilweise vollständig durchgerechnet, bzw. mittels verschiedener Algorithmen gelöst. Die verschiedenen Modelle werden für unterschiedliche Definitionsbereiche angewandt, und deren Ergebnisse verglichen.
Zum Schluss werden noch die Quellcodes der verwendeten Algorithmen angeführt.

1 Intervallarithmetik

In diesem Kapitel werden kurz die wichtigsten Eigenschaften der Intervallarithmetik erläutert, da die Intervallrechnung in den weiteren Kapiteln eine wichtige Rolle spielen wird. Nach der Definition von Intervallen behandelt dieser Abschnitt vor allem die grundlegenden Operationen von Intervallen sowie häufig auftretende Probleme. Genauer behandelt wird das Thema Intervallarithmetik in [9], [10].

Definition 1.1. *Als Intervall* $\mathbf{a} := [\underline{a}, \overline{a}] \subseteq \mathbb{R}$ *bezeichnet man die Menge*

$$\left\{x \in \mathbb{R} : \underline{a} \leq x \leq \overline{a},\ \text{wobei } \underline{a}, \overline{a} \in \overline{\mathbb{R}} = \mathbb{R} \cup \{\pm\infty\}\right\},$$

also die reellen Zahlen zwischen den beiden Endpunkten $\underline{a}$ *und* $\overline{a}$*. Eine reelle Zahl* x *kann auch als Intervall* $[x, x]$ *aufgefasst werden. Solche Intervalle werden als degeneriert bezeichnet.*

Bemerkung 1.2. *Es ist einfacher, anstelle von degenerierten Intervallen die entsprechenden reellen Zahlen zu verwenden.*

Definition 1.3. *Ein Intervall* $\mathbf{a} := [\underline{a}, \overline{a}]$ *heißt positiv falls* $\underline{a} \geq 0$*, strikt positiv falls* $\underline{a} > 0$*, negativ falls* $\overline{a} \leq 0$*, strikt negativ falls* $\overline{a} < 0$*, Nullintervall falls* $\underline{a} \leq 0$ *und* $\overline{a} \geq 0$ *ist.*
Zwei Intervalle $[\underline{a}, \overline{a}]$*,* $[\underline{b}, \overline{b}]$ *sind gleich, genau dann wenn* $\underline{a} = \underline{b}$ *und* $\overline{a} = \overline{b}$*.*

Definition 1.4. *Seien* $\mathbf{a} := [\underline{a}, \overline{a}]$ *und* $\mathbf{b} := [\underline{b}, \overline{b}]$ *zwei Intervalle. Eine Funktion* f *von Intervallen ist definiert als*

$$f(\mathbf{a}, \mathbf{b}) = \square \{f(a, b) | a \in \mathbf{a}, b \in \mathbf{b}\}.$$

Theorem 1.5. *Seien* $\mathbf{I} = [a, b]$ *und* $\mathbf{J} = [c, d]$ *zwei Intervalle. Aus Definition 1.4. folgt für Addition, Subtraktion, Multiplikation, Division von Intervallen*

$$\begin{aligned}
\mathbf{I} + \mathbf{J} &= [a + c, b + d] \\
\mathbf{I} - \mathbf{J} &= [a - d, b - c] \\
\mathbf{I} \cdot \mathbf{J} &= \begin{cases}
[ac, bd] & \text{wenn } a \geq 0, c \geq 0 \\
[bc, bd] & \text{wenn } a \geq 0, c < 0 < d \\
[bc, ad] & \text{wenn } a \geq 0, d \leq 0 \\
[ad, bc] & \text{wenn } a < 0 < b, c \geq 0 \\
[bd, ad] & \text{wenn } a < 0 < b, d \leq 0 \\
[ad, bc] & \text{wenn } b \leq 0, c \geq 0 \\
[ad, ac] & \text{wenn } b \leq 0, c < 0 < d \\
[bd, ac] & \text{wenn } b \leq 0, d \leq 0 \\
[\min(bc, ad), \max(ac, bd)] & \text{wenn } a < 0 < b, c < 0 < d
\end{cases}
\end{aligned}$$

$$\begin{aligned}\frac{1}{\mathbf{I}} &= \left[\frac{1}{b}, \frac{1}{a}\right] && (0 \notin I)\\ \frac{\mathbf{J}}{\mathbf{I}} &= J \cdot \left(\frac{1}{I}\right) && (0 \notin I).\end{aligned}$$

Außerdem gilt für n=0,1,2,...

$$\mathbf{I}^n = \begin{cases} [1,1] & \text{wenn } n = 0 \\ [a^n, b^n] & \text{wenn } a \geq 0, \text{ oder wenn } a \leq 0 \leq b \text{ und } n \text{ ungerade} \\ [b^n, a^n] & \text{wenn } b \leq 0 \\ [0, \max(a^n, b^n)] & \text{wenn } a \leq 0 \leq b \text{ und } n \text{ gerade.} \end{cases}$$

Dieses Theorem kann noch um die Division durch ein Nullintervall erweitert werden. Bezeichnet werden die entstehenden Regeln als erweiterte Intervallarithmetik.

Theorem 1.6. *Seien* $\mathbf{I} = [a, b]$ *und* $\mathbf{J} = [c, d]$ *zwei Intervalle und a, b, c, d endlich. Für* $c \leq 0 \leq d$, $c<d$ *ergibt sich*

$$\frac{\mathbf{I}}{\mathbf{J}} = \begin{cases} [b/c, \infty] & \text{wenn } b \leq 0, d = 0 \\ [-\infty, b/d] & \text{wenn } b \leq 0, c < 0 < d \\ [-\infty, b/d] & \text{wenn } b \leq 0, c = 0 \\ [-\infty, \infty] & \text{wenn } a < 0 < b \\ [-\infty, a/c] & \text{wenn } a \geq 0, d = 0 \\ [-\infty, a/c] \cup [a/d, \infty] & \text{wenn } a \geq 0, c < 0 < d \\ [a/d, \infty] & \text{wenn } a \geq 0, c = 0. \end{cases}$$

Für Addition und Subtraktion von unendlichen oder einseitig unendlichen Intervallen gilt

$$\begin{aligned}[a,b] + [-\infty, d] &= [-\infty, b+d] \\ [a,b] + [c, \infty] &= [a+c, \infty] \\ [a,b] \pm [-\infty, \infty] &= [-\infty, \infty] \\ [a,b] - [-\infty, d] &= [a-d, \infty] \\ [a,b] - [c, \infty] &= [-\infty, b-c].\end{aligned}$$

Definition 1.7. *Als mehrdimensionales Intervall oder Box* $\mathbf{a} = [\underline{a}_1, \overline{a}_1] \times [\underline{a}_2, \overline{a}_2] \times \cdots \times [\underline{a}_n, \overline{a}_n] \subseteq \mathbb{R}^n$ *bezeichnet man die Menge*

$$\left\{ x \in \mathbb{R}^n : \underline{a}_i \leq x_i \leq \overline{a}_i, \text{ wobei } \underline{a}_i, \overline{a}_i \in \overline{\mathbb{R}} = \mathbb{R} \cup \{\pm\infty\} \quad \forall i \in \{1, ..., n\} \right\}.$$

Definition 1.8. *Als Mittelpunkt eines n-dimensionalen, beschränkten Intervalls* $\mathbf{a} = [\underline{a}_1, \overline{a}_1] \times [\underline{a}_2, \overline{a}_2] \times \cdots \times [\underline{a}_n, \overline{a}_n]$ *bezeichnet man den Punkt* $x \in \mathbb{R}^n$ *für den gilt*

$$x_i = \frac{\overline{a}_i + \underline{a}_i}{2} \quad \forall i \in \{1, 2, \ldots, n\}.$$

Bemerkung 1.9. *Im Folgenden wird mit* $\mathrm{mid}(\mathbf{a})$ *der Mittelpunkt eines Intervalls* $\mathbf{a}$ *bezeichnet.*

Probleme, die bei der Intervallrechnung auftreten können:

Häufig tritt bei der Intervallrechnung eine innere Abhängigkeit auf. Das ergibt folgendes Problem: Sei $f(x) = x^2 = x \cdot x$ eine Funktion von den reellen Zahlen in sich selbst. Wird nun der Wertebereich (Range) der Funktion für $x \in \mathbf{I} = [-1, 1]$ abgeschätzt, so errechnet sich $f(\mathbf{I}) = \mathbf{I}^2 = [0, 1]$. Für $f(x) = x \cdot x$ ergibt sich allerdings $f(\mathbf{I}) = \mathbf{I} \cdot \mathbf{I} = [-1, 1]$. Das Ergebnis ist also vom arithmetischen Ausdruck abhängig, der für die Funktion verwendet wird. Das wird als Überschätzung bezeichnet. Ein noch einfacheres Beispiel für das Auftreten von Überschätzung ist die Funktion $f(x) = x - x$. Wird hier $x \in [a, b]$ eingesetzt, um eine Einschließung für $f(x)$ zu berechnen, so ergibt das $[a - b, b - a]$ und nicht das degenerierte Intervall $[0, 0]$.

Bemerkung 1.10.

- *In der Intervallarithmetik gilt die Subdistributivität. Das heißt für Intervalle* $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$ *gilt*

$$\mathbf{a}(\mathbf{b} + \mathbf{c}) \subseteq \mathbf{a} \cdot \mathbf{b} + \mathbf{a} \cdot \mathbf{c}.$$

- *Wenn bei einer Intervallrechnung jede Variable nur einmal vorkommt, so wird der genaue Wertebereich der Funktion bestimmt. Es gibt also keine Überschätzung.*

Ein weiteres Problem stellen Rundungsfehler dar. Zum Beispiel bei der Addition folgender Intervalle: [2.666666,1.000000]+[1.22222,3.00000]=[3.888886,4.000000]. Rundet man dieses Ergebnis nun auf fünf signifikante Nachkommastellen so erhält man das Intervall [3.88889,4.00000], welches das exakte Intervall nicht mehr enthält.

$$[2.666666, 1.000000] + [1.22222, 3.00000] = [3.888886, 4.000000] \not\subseteq [3.88889, 4.00000]$$

In der Intervallarithmetik verwendet man anstelle des gewöhnlichen Rundens das sogenannte Außenrunden, bei dem die untere Grenze des Intervalls nach unten und die obere Grenze nach oben gerundet wird. Angewandt beim vorigen Beispiel sieht das dann folgendermaßen aus:

$$[2.666666, 1.000000] + [1.22222, 3.00000] = [3.888886, 4.000000] \subseteq [3.88888, 4.00000].$$

Bemerkung 1.11. *Für die Intervallrechnung mittels Matlab gibt es eine kostenlose Toolbox namens Intlab, welche von Prof. Dr. Siegfried M. Rump (Technische Universität Hamburg) entwickelt wurde. Für Details siehe [11]. In den verschiedenen Implementation weiter unten wird die Intlab Version 5.5 verwendet.*

2 Slopes

2.1 Eindimensionale Slopes

In diesem Kapitel werden Slopes erster und zweiter Ordnung von eindimensionalen Funktionen beschrieben. Neben der exakten Definition werden die verschiedenen Rechenregeln betrachtet. Zunächst werden Slopes erster Ordnung vorgestellt.

2.1.1 Slopes erster Ordnung

Definition 2.1. *Sei $f(x) : \mathbb{R} \to \mathbb{R}$ eine eindimensionale Funktion und $z \in \mathbb{R}$ beliebig und fest. Sei $f[z,x] \in \mathbb{R}$ und $\mathbf{x}$ ein Intervall. Dann heißt $f[z,x]$ Slope erster Ordnung von f in z über $\mathbf{x}$, falls gilt*

$$f(x) = f(z) + f[z,x](x-z) \quad \forall x \in \mathbf{x}.$$

Theorem 2.2. *Ist $f[z,x]$ ein Slope erster Ordnung von f in z über $\mathbf{x}$, so gilt*

$$f(x) \in f(z) + f[z,\mathbf{x}](x-z) \quad \forall x \in \mathbf{x}.$$

Bemerkung 2.3.

- *Slopes eindimensionaler Funktionen sind eindeutig.*
- *Die Ähnlichkeit von Slopes und Ableitungen ist in der Definition leicht zu sehen. Es gilt $(x \to z) \Rightarrow (f[z,x] \to f'(z))$. Weiter unten in diesem Kapitel wird gezeigt, dass Slopes und Ableitungen gleich strukturierte Kettenregeln haben.*

Rechenregeln

Für einige Operationen von Slopes erster Ordnung gibt es Rechenregeln, welche im Folgenden gezeigt werden.

Addition: Sei $h(x) = f(x) + g(x)$, dann gilt

$$\begin{aligned} h(z) &= f(z) + g(z) \\ h(x) - h(z) &= f(x) + g(x) - f(z) - g(z) = f(x) - f(z) + g(x) - g(z) = \\ &= f[z,x](x-z) + g[z,x](x-z) = \\ &= (f[z,x] + g[z,x])(x-z). \end{aligned}$$

Das heißt

$$h[z,x] = f[z,x] + g[z,x].$$

Subtraktion: Sei $h(x) = f(x) - g(x)$. Es gilt

$$\begin{aligned} h(z) &= f(z) - g(z) \\ h(x) - h(z) &= f(x) - g(x) - (f(z) - g(z)) = f(x) - f(z) - (g(x) - g(z)) = \\ &= f[z,x](x-z) - g[z,x](x-z) = \\ &= (f[z,x] - g[z,x])(x-z). \end{aligned}$$

Daraus folgt

$$h[z,x] = f[z,x] - g[z,x].$$

Multiplikation: Sei $h(x) = f(x) \cdot g(x)$, dann gilt

$$\begin{aligned} h(z) &= f(z)g(z) \\ h(x) - h(z) &= f(x)g(x) - f(z)g(z) = \\ &= f(x)g(x) - f(z)g(x) + f(z)g(x) - f(z)g(z) = \\ &= (f(x) - f(z))\, g(x) + f(z)\, (g(x) - g(z)) = \\ &= f[z,x](x-z)g(x) + f(z)g[z,x](x-z) = \\ &= (f[z,x]g(x) + f(z)g[z,x])\, (x-z). \end{aligned}$$

Es ergibt sich

$$h[z,x] = f[z,x]g(x) + f(z)g[z,x].$$

Diese Darstellung ist nicht eindeutig. Analog kann man

$$h[z,x] = g[z,x]f(x) + g(z)f[z,x]$$

herleiten. Dies ändert jedoch nichts an der Eindeutigkeit von $h[z,x]$.

Division: Sei $h(x) = \frac{f(x)}{g(x)}$. So gilt

$$\begin{aligned} h(z) &= \frac{f(z)}{g(z)} \\ h(x) - h(z) &= \frac{f(x)}{g(x)} - \frac{f(z)}{g(z)} = \\ &= \frac{1}{g(x)} \left(f(x) - h(z)g(x)\right) = \\ &= \frac{1}{g(x)} \left(f(x) + f(z) - f(z) - h(z)g(x)\right) = \\ &= \frac{1}{g(x)} \left(f(x) + h(z)g(z) - f(z) - h(z)g(x)\right) = \\ &= \frac{1}{g(x)} \left(f(x) - f(z) - h(z)\left(g(x) - g(z)\right)\right) = \\ &= \frac{1}{g(x)} \left(f[z,x] - h(z)g[z,x]\right)(x-z). \end{aligned}$$

Also folgt

$$h[z,x] = \frac{f[z,x] - h(z)g[z,x]}{g(x)}.$$

Quadratfunktion: Sei $h(x) = f(x)^2$. Zur Berechnung von $h[z,x]$ betrachtet man wieder $h(x) - h(z)$.

$$\begin{aligned} h(z) &= f(z)^2 \\ h(x) - h(z) &= f(x)^2 - f(z)^2 = \\ &= (f(x) + f(z))\,(f(x) - f(z)) = \\ &= (f(x) + f(z))\,f[z,x](x - z) \end{aligned}$$

Daher folgt

$$h[z,x] = (f(x) + f(z))\,f[z,x].$$

Wurzelfunktion: Sei $h(x) = \sqrt{f(x)}$. Um $h[z,x]$ zu erhalten, berechnet man abermals $h(x) - h(z)$.

$$\begin{aligned} h(z) &= \sqrt{f(z)} \\ h(x) - h(z) &= \sqrt{f(x)} - \sqrt{f(z)} = \\ &= \frac{f(x) - f(z)}{\sqrt{f(x)} + \sqrt{f(z)}} = \\ &= \frac{f[z,x]}{h(x) + h(z)}(x - z) \end{aligned}$$

Man erhält

$$h[z,x] = \frac{f[z,x]}{h(x) + h(z)}.$$

Komposition von Funktionen

Seien $f(x) : \mathbb{R} \to \mathbb{R}$, $g(x) : \mathbb{R} \to \mathbb{R}$ eindimensionale Funktionen und $z \in \mathbb{R}$ beliebig und fest. So gilt

$$\begin{aligned} (f \circ g)\,(x) &= f\,(g(x)) = \\ &= f\,(g(z)) + f\,[g(z), g(x)]\,(g(x) - g(z)) = \\ &= f\,(g(z)) + f\,[g(z), g(x)]\,(g(z) + g[z,x](x - z) - g(z)) = \\ &= (f \circ g)\,(z) + f\,[g(z), g(x)]\,g[z,x](x - z). \end{aligned}$$

Somit kann $(f \circ g)[z,x]$ als $f\,[g(z), g(x)]\,g[z,x]$ gewählt werden und ist ebenfalls wieder eindeutig.

2.1.2 Slopes zweiter Ordnung

Slopes zweiter Ordnung sind analog zu den Slopes erster Ordnung definiert. Ein Slope zweiter Ordnung ist ein Slope zweier Slopes erster Ordnung.

Definition 2.4. *Sei $f[x,z]$ ein Slope erster Ordnung der Funktion $f(x)$ im Intervall $\mathbf{x}$ und sei $f[z',z]$ ein Slope erster Ordnung der Funktion $f(x)$ im dünnen Intervall z' für den Punkt z. Dann heißt $f[z,z',x]$ Slope zweiter Ordnung, falls gilt*

$$f[z,x] = f[z,z'] + (x-z')f[z,z',x] \quad \forall x \in \mathbf{x}.$$

Bemerkung 2.5. *Wird diese Gleichung in die Definition von Slopes erster Ordnung eingesetzt so erhält man*

$$f(x) = f(z) + \big(f[z,z'] + (x-z')f[z,z',x]\big)(x-z) \quad \forall x \in \mathbf{x}.$$

Theorem 2.6. *Seien $f[z,x]$ und $f[z,z',x]$ Slopes erster bzw. zweiter Ordnung einer Funktion $f(x)$. Es folgt*

$$\begin{aligned} f[z,x] &\in f[z,z'] + (x-z')f[z,z',\mathbf{x}] \quad \forall x \in \mathbf{x} \\ f(x) &\in f(z) + \big(f[z,z'] + (x-z')f[z,z',\mathbf{x}]\big)(x-z) \quad \forall x \in \mathbf{x}. \end{aligned}$$

Rechenregeln

Wie schon bei den Slopes erster Ordnung kann man wieder einige Rechenregeln herleiten.

Addition: Sei $h(x) = f(x) + g(x)$, dann gilt

$$\begin{aligned} h[z,x] &= f[z,x] + g[z,x] \\ h[z,z'] &= f[z,z'] + g[z,z']. \end{aligned}$$

Daraus folgt

$$\begin{aligned} h[z,x] - h[z,z'] &= f[z,x] + g[z,x] - (f[z,z'] + g[z,z']) = \\ &= f[z,x] - f[z,z'] + g[z,x] - g[z,z'] = \\ &= (x-z')f[z,z',x] + (x-z')g[z,z',x] = \\ &= (x-z')\big(f[z,z',x] + g[z,z',x]\big). \end{aligned}$$

Man erhält

$$h[z,z',x] = f[z,z',x] + g[z,z',x].$$

Subtraktion: Sei $h(x) = f(x) - g(x)$, dann kann man analog zur Addition

$$h[z,z',x] = f[z,z',x] - g[z,z',x]$$

herleiten.

Multiplikation: Sei $h(x) = f(x) \cdot g(x)$, so gilt laut den Regeln für Slopes erster Ordnung

$$\begin{aligned} h[z,x] &= f[z,x]g(x) + f(z)g[z,x] = \\ &= \big(f[z,z'] + (x-z')f[z,z',x]\big)\, g(x) + \big(g[z,z'] + (x-z')g[z,z',x]\big)\, f(z) \\ h[z,z'] &= f[z,z']g(z') + f(z)g[z,z']. \end{aligned}$$

Weiters gilt

$$\begin{aligned} h[z,x] - h[z,z'] &= \big(f[z,z'] + (x-z')f[z,z',x]\big)\, g(x) + \big(g[z,z'] + (x-z')g[z,z',x]\big)\, f(z) - \\ &\quad - \big(f[z,z']g(z') + f(z)g[z,z']\big) = \\ &= \big(f[z,z']g(x) + (x-z')f[z,z',x]g(x) + g[z,z']f(z) + \\ &\quad + (x-z')g[z,z',x]f(z)\big) - f[z,z']g(z') - g[z,z']f(z) = \\ &= f[z,z'](\underbrace{g(x) - g(z')}_{g[z',x](x-z')}) + (x-z')\left(f[z,z',x]g(x) + g[z,z',x]f(z)\right) = \\ &= (x-z')\left(f[z,z']g[z',x] + f[z,z',x]g(x) + g[z,z',x]f(z)\right). \end{aligned}$$

Insgesamt erhält man also

$$h[z,z',x] = f[z,z']g[z',x] + f[z,z',x]g(x) + g[z,z',x]f(z).$$

Man kann $h[z,z',x]$ noch umformen zu

$$\begin{aligned} h[z,z',x] &= f[z,z']g[z',x] + f[z,z',x]g(x) + g[z,z',x]f(z) = \\ &= f[z,z',x]\left(g(z') + g[z',x](x-z')\right) + g[z,z',x]f(z) + f[z,z']g[z',x] = \\ &= f[z,z',x]g(z') + g[z,z',x]f(z) + f[z,z']g[z',x] + f[z,z',x]g[z',x](x-z') = \\ &= f[z,z',x]g(z') + g[z,z',x]f(z) + g[z',x]\left(f[z,z'] + f[z,z',x](x-z')\right) = \\ &= f[z,z',x]g(z') + g[z,z',x]f(z) + g[z',x]f[z,x]. \end{aligned}$$

Division: Sei $h(x) = \frac{f(x)}{g(x)}$. So gilt

$$\begin{aligned} h[z,x] &= \frac{f[z,x] - h(z)g[z,x]}{g(x)} \\ h[z,z'] &= \frac{f[z,z'] - h(z)g[z,z']}{g(z')}. \end{aligned}$$

Es folgt

$$\begin{aligned} h[z,x] - h[z,z'] &= \frac{f[z,x] - h(z)g[z,x]}{g(x)} - \frac{f[z,z'] - h(z)g[z,z']}{g(z')} = \\ &= \frac{f[z,x]}{g(x)} - \frac{h(z)g[z,x]}{g(x)} - \frac{1}{g(x)g(z')}\left(f[z,z'] - h(z)g[z,z']\right) g(x) = \end{aligned}$$

$$
\begin{aligned}
&= \frac{f[z,z'] + (x-z')f[z,z',x]}{g(x)} - h(z)\frac{g[z,z'] + (x-z')g[z,z',x]}{g(x)} - \\
&- \frac{1}{g(x)g(z')}\left(f[z,z'] - h(z)g[z,z']\right)g(x) = \\
&= \frac{f[z,z',x]}{g(x)}(x-z') - \frac{h(z)g[z,z',x]}{g(x)}(x-z') + \\
&+ \frac{1}{g(x)g(z')}\Bigg(\left(f[z,z']g(z') - g[z,z']g(z')h(z)\right) + \\
&+ \left(-f[z,z'] + h(z)g[z,z']\right)\left(g(z') + g[z',x](x-z')\right)\Bigg) = \\
&= \frac{f[z,z',x]}{g(x)}(x-z') - \frac{h(z)g[z,z',x]}{g(x)}(x-z') + \\
&+ \frac{1}{g(x)g(z')}\Bigg(\left(f[z,z']g(z') - g[z,z']g(z')h(z)\right) + \\
&+ \Bigg(h(z)g[z,z']g(z') + h(z)g[z,z']g[z',x](x-z') - \\
&- f[z,z']g(z') - f[z,z']g[z',x](x-z')\Bigg)\Bigg) = \\
&= \frac{f[z,z',x]}{g(x)}(x-z') - \frac{h(z)g[z,z',x]}{g(x)}(x-z') + \\
&+ \frac{1}{g(x)g(z')}\Bigg(g[z',x]\left(g[z,z']h(z) - f[z,z']\right)\Bigg)(x-z').
\end{aligned}
$$

Daher gilt

$$
\begin{aligned}
h[z,z',x] &= \frac{f[z,z',x]}{g(x)} - \frac{h(z)g[z,z',x]}{g(x)} + \frac{1}{g(x)g(z')}\left(g[z',x]\left(g[z,z']h(z) - f[z,z']\right)\right) = \\
&= \frac{1}{g(x)}\left(f[z,z',x] - h(z)g[z,z',x] + \frac{1}{g(z')}\left(g[z',x]\left(g[z,z']h(z) - f[z,z']\right)\right)\right).
\end{aligned}
$$

Quadratfunktion: Sei $h(x) = f(x)^2$. So gilt

$$
\begin{aligned}
h[z,x] &= (f(x) + f(z))f[z,x] \\
h[z,z'] &= (f(z') + f(z))f[z,z'].
\end{aligned}
$$

Daraus folgt

$$
\begin{aligned}
h[z,x] - h[z,z'] &= (f(x) + f(z))f[z,x] - (f(z') + f(z))f[z,z'] = \\
&= f(x)f[z,x] - f(z')f[z,z'] + f(z)(f[z,x] - f[z,z']) = \\
&= f(x)f[z,x] - f(z')\left(f[z,x] - (x-z')f[z,z',x]\right) + f(z)(x-z')f[z,z',x] = \\
&= f[z,x]f[z',x](x-z') + f(z')(x-z')f[z,z',x] + f(z)(x-z')f[z,z',x] = \\
&= \left(f[z,x]f[z',x] + \left(f(z) + f(z')\right)f[z,z',x]\right)(x-z').
\end{aligned}
$$

Es gilt also

$$h[z,z',x] = f[z,x]f[z',x] + \big(f(z)+f(z')\big)\, f[z,z',x].$$

Wurzelfunktion: Sei $h(x) = \sqrt{f(x)}$. Es gilt

$$h[z,x] = \frac{f[z,x]}{h(x)+h(z)}$$
$$h[z,z'] = \frac{f[z,z']}{h(z')+h(z)}.$$

Somit folgt

$$\begin{aligned}
h[z,x] - h[z,z'] &= \\
&= \frac{f[z,x]}{h(x)+h(z)} - \frac{f[z,z']}{h(z')+h(z)} = \\
&= \frac{\big(f[z,x]-f[z,z']\big)\,h(z) + f[z,x]h(z') - f[z,z']h(x)}{h(x)h(z')+h(x)h(z)+h(z)h(z')+f(z)} = \\
&= \frac{(x-z')f[z,z',x]h(z) + f[z,x]h(z') - \big(f[z,x]-(x-z')f[z,z',x]\big)\,h(x)}{h(x)h(z')+h(x)h(z)+h(z)h(z')+f(z)} = \\
&= \frac{(x-z')f[z,z',x]h(z) + f[z,x]\,(h(z')-h(x)) + (x-z')f[z,z',x]h(x)}{h(x)h(z')+h(x)h(z)+h(z)h(z')+f(z)} = \\
&= \frac{(x-z')f[z,z',x]h(z) - f[z,x]h[z',x](x-z') + (x-z')f[z,z',x]h(x)}{h(x)h(z')+h(x)h(z)+h(z)h(z')+f(z)} = \\
&= \frac{f[z,z',x]h(z) - f[z,x]h[z',x] + f[z,z',x]h(x)}{h(x)h(z')+h(x)h(z)+h(z)h(z')+f(z)}(x-z')
\end{aligned}$$

Das ergibt

$$h[z,z',x] = \frac{f[z,z',x]h(z) - f[z,x]h[z',x] + f[z,z',x]h(x)}{h(x)h(z')+h(x)h(z)+h(z)h(z')+f(z)}.$$

Komposition von Funktionen

Seien $f(x) : \mathbb{R} \to \mathbb{R}$, $g(x) : \mathbb{R} \to \mathbb{R}$ eindimensionale Funktionen und $z, z' \in \mathbb{R}$ beliebig und fest. So gilt

$$\begin{aligned}
(f \circ g)\,[z,x] &= f[g(z),g(x)]g[z,x] = \\
&= \big(f[g(z),g(z')] + \big(g(x)-g(z')\big)\, f[g(z),g(z'),g(x)]\big)\,\big(g[z,z']+ \\
&+ (x-z')g[z,z',x]\,) = \\
&= f[g(z),g(z')]g[z,z'] + \big(g(x)-g(z')\big)\, f[g(z),g(z'),g(x)]g[z,z']+ \\
&+ (x-z')f[g(z),g(z')]g[z,z',x]+ \\
&+ (x-z')\,\big(g(x)-g(z')\big)\, f[g(z),g(z'),g(x)]g[z,z',x] =
\end{aligned}$$

$$
\begin{aligned}
&= f[g(z), g(z')]g[z, z'] + g[z', x](x - z')f[g(z), g(z'), g(x)]g[z, z']+ \\
&+ (x - z')f[g(z), g(z')]g[z, z', x]+ \\
&+ (x - z')g[z', x](x - z')f[g(z), g(z'), g(x)]g[z, z', x] = \\
&= f[g(z), g(z')]g[z, z'] + \big(g[z', x]f[g(z), g(z'), g(x)]g[z, z']+ \\
&+ f[g(z), g(z')]g[z, z', x]+ \\
&+ g[z', x](x - z')f[g(z), g(z'), g(x)]g[z, z', x]\big)(x - z').
\end{aligned}
$$

Somit gilt

$$
\begin{aligned}
(f \circ g)[z, z', x] = {} & g[z', x]f[g(z), g(z'), g(x)]g[z, z'] + f[g(z), g(z')]g[z, z', x]+ \\
& + g[z', x](x - z')f[g(z), g(z'), g(x)]g[z, z', x].
\end{aligned}
$$

Bemerkung 2.7.

- *Für andere elementare Funktionen lassen sich sowohl bei Slopes erster Ordnung als auch bei Slopes zweiter Ordnung, bis auf wenige Funktionen mit bestimmten Voraussetzungen, keine derartigen Formeln finden. Einige Lemmata zu diesen Ausnahmen findet man in [13].*
- *In Kapitel 7 und in den verschiedenen Algorithmen werden Slopes erster und zweiter Ordnung dazu verwendet, um Intervalleinschließungen verschiedener Funktionen zu berechnen.*
- *Weitere Details zum Thema Slopes findet man in [9], [12] und [13].*

2.2 Mehrdimensionale Slopes

Dieser Abschnitt beinhaltet die Berechnung von Slopes erster und zweiter Ordnung von mehrdimensionalen Funktionen. Ähnlich wie im vorigen Abschnitt werden wieder die einfachsten Rechenregeln beschrieben. Als erstes wird eine exakte Definition für Slopes erster Ordnung gegeben.

2.2.1 Slopes erster Ordnung

Definition 2.8. *Sei $f(x) : \mathbb{R}^n \to \mathbb{R}$ eine n-dimensionale Funktion und $z \in \mathbb{R}^n$ beliebig und fest. Sei $f[z, x]$ ein Zeilenvektor mit n Einträgen wobei $f[z, x]_i \in \mathbb{R} \quad \forall i \in 1, \ldots, n$ und $\mathbf{x}$ eine n-dimensionale Box. Dann heißt $f[z, x]$ Slope erster Ordnung von f in z über $\mathbf{x}$ falls gilt*

$$f(x) = f(z) + f[z, x](x - z) \qquad \forall x \in \mathbf{x}.$$

Theorem 2.9. *Sei $f[z, x]$ ein Slope erster Ordnung von f in z über $\mathbf{x}$. So folgt*

$$f(x) \in f(z) + f[z, \mathbf{x}](x - z) \qquad \forall x \in \mathbf{x}.$$

Bemerkung 2.10. *Slopes mehrdimensionaler Funktionen sind im Gegensatz zu den Slopes eindimensionaler Funktionen nicht eindeutig.*

Rechenregeln

Analog zu den Rechenregeln für eindimensionale Funktionen können auch folgende Rechenregeln hergeleitet werden.

Addition: Sei $h(x) = f(x) + g(x)$. Dann gilt

$$h[z, x] = f[z, x] + g[z, x].$$

Subtraktion: Sei $h(x) = f(x) - g(x)$. Es gilt

$$h[z, x] = f[z, x] - g[z, x].$$

Multiplikation: Sei $h(x) = f(x) \cdot g(x)$. Dann gilt

$$\begin{aligned} h[z, x] &= f[z, x]g(x) + f(z)g[z, x] \\ h[z, x] &= g[z, x]f(x) + g(z)f[z, x]. \end{aligned}$$

Division: Sei $h(x) = \frac{f(x)}{g(x)}$. So gilt

$$h[z, x] = \frac{f[z, x] - h(z) \cdot g[z, x]}{g(x)}.$$

Quadratfunktion: Sei $h(x) = f(x)^2$. Es gilt

$$h[z, x] = (f(x) + f(z)) \cdot f[z, x].$$

Wurzelfunktion: Sei $h(x) = \sqrt{f(x)}$. So gilt

$$h[z, x] = \frac{f[z, x]}{h(x) + h(z)}.$$

Bemerkung 2.11. *Man erhält also wieder die gleichen Rechenregeln wie im Eindimensionalen, nur dass im n-dimensionalen Fall die Slopes n-dimensionale Zeilenvektoren sind.*
Für $f(x) : \mathbb{R} \to \mathbb{R}$, $g(x) : \mathbb{R}^n \to \mathbb{R}$ eindimensionale Funktionen und $z \in \mathbb{R}^n$ beliebig und fest kann analog zum eindimensionalen Fall $(f \circ g)[z, x] = f[g(z), g(x)]g[z, x]$ hergeleitet werden.

2.2.2 Slopes zweiter Ordnung

Analog zu den Slopes erster Ordnung werden wieder Slopes zweiter Ordnung definiert.

Definition 2.12. *Sei $f[x,z]$ ein Slope erster Ordnung der Funktion $f(x)$ in der Box $\mathbf{x}$, und sei $f[z',z]$ ein Slope erster Ordnung der Funktion $f(x)$ im dünnen Intervall z' für den Punkt z. Dann heißt eine $n \times n$–Matrix $f[z,z',x]$ Slope zweiter Ordnung, falls gilt*

$$f[z,x] = f[z,z'] + (x-z')^{\mathsf{T}} f[z,z',x] \quad \forall x \in \mathbf{x}.$$

Theorem 2.13. *Seien $f[z,x]$, $f[z,z',x]$ Slopes erster und zweiter Ordnung einer n-dimensionalen Funktion $f(x)$. So gilt*

$$\begin{aligned} f[z,x] &\in f[z,z'] + (x-z')^{\mathsf{T}} f[z,z',\mathbf{x}] \quad \forall x \in \mathbf{x} \\ f(x) &\in f(z) + \Big(f[z,z'] + (x-z')^{\mathsf{T}} f[z,z',\mathbf{x}]\Big)(x-z) \quad \forall x \in \mathbf{x}. \end{aligned}$$

Rechenregeln

Addition: Sei $h(x) = f(x) + g(x)$, dann gilt

$$h[z,z',x] = f[z,z',x] + g[z,z',x].$$

Subtraktion: Sei $h(x) = f(x) - g(x)$, so gilt

$$h[z,z',x] = f[z,z',x] - g[z,z',x].$$

Multiplikation: Sei $h(x) = f(x) \cdot g(x)$, so ergibt sich

$$h[z,z',x] = f[z,z']^{\mathsf{T}} g[z',x] + f[z,z',x]g(x) + g[z,z',x]f(z).$$

Division: Sei $h(x) = \frac{f(x)}{g(x)}$. So gilt

$$\begin{aligned} h[z,z',x] &= \frac{f[z,z',x]}{g(x)} - \frac{h(z)g[z,z',x]}{g(x)} + \frac{1}{g(x)g(z')}\Big(g[z',x]^{\mathsf{T}}\left(g[z,z']h(z) - f[z,z']\right)\Big) = \\ &= \frac{1}{g(x)}\left(f[z,z',x] - h(z)g[z,z',x] + \frac{1}{g(z')}\Big(g[z',x]^{\mathsf{T}}\left(g[z,z']h(z) - f[z,z']\right)\Big)\right). \end{aligned}$$

Quadratfunktion: Sei $h(x) = f(x)^2$. Es gilt

$$h[z,z',x] = f[z',x]f[z,x]^{\mathsf{T}} + \big(f(z) + f(z')\big)\, f[z,z',x].$$

Wurzelfunktion: Sei $h(x) = \sqrt{f(x)}$. Dann ergibt sich

$$h[z,z',x] = \frac{f[z,z',x]h(z) - h[z',x]^{\mathsf{T}} f[z,x] + f[z,z',x]h(x)}{h(x)h(z') + h(x)h(z) + h(z)h(z') + f(z)}.$$

Komposition von Funktionen

Seien $f(x) : \mathbb{R} \to \mathbb{R}$, $g(x) : \mathbb{R}^n \to \mathbb{R}$ eindimensionale Funktionen und $z, z' \in \mathbb{R}^n$ beliebig und fest. So gilt

$$\begin{aligned}(f \circ g)\,[z, z', x] = g[z', x]^{\mathsf{T}} f[g(z), g(z'), g(x)] g[z, z'] + f[g(z), g(z')] g[z, z', x] + \\ + g[z', x]^{\mathsf{T}} f[g(z), g(z'), g(x)] (x - z')^{\mathsf{T}} g[z, z', x].\end{aligned}$$

Bemerkung 2.14. *Die Rechenregeln für mehrdimensionale Slopes zweiter Ordnung weisen wieder die gleiche Form wie die Rechenregeln der eindimensionalen Slopes auf. Beweisen kann man diese Regeln analog zum eindimensionalen Fall, nur dass $f[z, z', x]$ und $g[z, z', x]$ $n \times n$–Matrizen sind und somit in manchen Fällen transponiert werden müssen.*

3 Taylor-Modelle

Beim Taylor-Modell wird versucht, eine Funktion durch ein Taylorpolynom darzustellen und mittels Intervallrechnung eine Schranke für das Taylorrestglied zu bestimmen. Dieses Kapitel gibt einen kurzen Ausblick darüber.
Eine zentrale Rolle dabei spielt das Theorem von Taylor.

Theorem 3.1. *Sei* $f : \mathbf{a} = [\underline{a}, \overline{a}] \subseteq \mathbb{R}^k \to \mathbb{R}$ *eine Funktion die* $(n+1)$*–mal partiell differenzierbar auf* $[\underline{a}, \overline{a}]$ *ist. Sei* $x_0 \in [\underline{a}, \overline{a}]$*. Dann gibt es für jedes* $x \in [\underline{a}, \overline{a}]$
ein $\theta \in \mathbb{R}$ *mit* $0 < \theta < 1$ *sodass*

$$f(x) = \sum_{i=0}^{n} \frac{1}{i!} \left((x - x_0) \cdot \nabla\right)^i f(x_0) + \frac{1}{(n+1)!} \left((x - x_0) \cdot \nabla\right)^{n+1} f(x_0 + (x - x_0)\theta),$$

wobei

$$(h \cdot \nabla)^k = \sum_{0 \le i_1, \cdots i_k \le j, i_1 + \cdots + i_k = j} \frac{j!}{i_1! \cdots i_k!} h_1^{i_1} \cdots h_k^{i_k} \frac{\partial^j}{\partial x_1^{i_1} \cdots \partial x_k^{i_k}}.$$

Das Theorem von Taylor erlaubt also eine Abschätzung für den Fehler, welcher gemacht wird, wenn man eine Funktion durch ihr Taylorpolynom approximiert.
Um die Notation zu erleichtern, definiert man einen Parameter α, der Detailinformationen über eine gegebene Taylorentwicklung enthält.

Definition 3.2.

$\alpha := (n, x_0, [\underline{a}, \overline{a}])$ *wobei*

$n \ldots$ *Ordnung des Taylorpolynoms*
$x_0 \ldots$ *Entwicklungspunkt*
$[\underline{a}, \overline{a}] \ldots$ *Domain auf dem die Funktion betrachtet wird*

Bemerkung 3.3.

- *Mit der vorigen Definition ist also* $P_{\alpha,f}$ *das n-te Taylorpolynom der Funktion f, entwickelt um* x_0*.*
- *Das Theorem von Taylor besagt außerdem, dass* $f(x) = P_{\alpha,f} + O(\|x\|^{n+1})$*.*

Mittels Theorem 3.1. lässt sich jede $(n+1)$–mal partiell differenzierbare Funktion $f : [\underline{a}, \overline{a}] \subseteq \mathbb{R}^k \to \mathbb{R}$ durch das Taylorpolynom n-ter Ordnung und das Restglied $\epsilon_{\alpha,f}$ darstellen. Man schreibt $f(x) = P_{\alpha,f}(x - x_0) + \epsilon_{\alpha,f}(x - x_0)$, wobei $\epsilon_{\alpha,f}(x - x_0)$ stetig am beschränkten Domain ist.
Diese Darstellung führt zu folgender Definition.

Definition 3.4. *Ein Paar $T_{\alpha,f} = (P_{\alpha,f}, I_{\alpha,f})$, aus einem Taylorpolynom und einem Intervallrestglied, wird Taylor-Modell von f genannt, genau dann wenn die Bedingung*

$$f(x) \in P_{\alpha,f}(x - x_0) + I_{\alpha,f} \qquad \forall x \in [\underline{a}, \overline{a}]$$

erfüllt ist.

Im folgenden Teil wird gezeigt, wie ein Taylor-Modell einer Summe, eines Produkts oder eines Quotienten von zwei Funktionen aus den Taylor-Modellen der einzelnen Funktionen berechnet wird.

3.1 Addition von Taylor-Modellen

Seien $f, g : [\underline{a}, \overline{a}] \subseteq \mathbb{R}^k \to \mathbb{R}$ Funktionen mit Taylor Modellen

$$T_{\alpha,f} = (P_{\alpha,f}, I_{\alpha,f})$$
$$T_{\alpha,g} = (P_{\alpha,g}, I_{\alpha,g}).$$

Nun wird das Taylor-Modell für $f + g$ berechnet.
Es gilt für alle $x \in [\underline{a}, \overline{a}]$

$$\begin{aligned} f(x) + g(x) &\in (P_{\alpha,f}(x - x_0) + I_{\alpha,f}) + (P_{\alpha,g}(x - x_0) + I_{\alpha,g}) = \\ &= (P_{\alpha,f}(x - x_0) + P_{\alpha,g}(x - x_0)) + (I_{\alpha,f} + I_{\alpha,g}). \end{aligned}$$

Folgender Zusammenhang ergibt nun ein Taylor-Modell $T_{\alpha,f+g}$ für $f + g$

$$P_{\alpha,f+g} = P_{\alpha,f} + P_{\alpha,g}$$
$$I_{\alpha,f+g} = I_{\alpha,f} + I_{\alpha,g}.$$

Das heißt

$$T_{\alpha,f+g} = T_{\alpha,f} + T_{\alpha,g} = (P_{\alpha,f+g}, I_{\alpha,f+g}).$$

3.2 Multiplikation von Taylor-Modellen

Seien $f, g : [\underline{a}, \overline{a}] \subseteq \mathbb{R}^k \to \mathbb{R}$ wieder Funktionen mit Taylor-Modellen

$$T_{\alpha,f} = (P_{\alpha,f}, I_{\alpha,f})$$
$$T_{\alpha,g} = (P_{\alpha,g}, I_{\alpha,g}).$$

Es gilt für alle $x \in [\underline{a}, \overline{a}]$

$$\begin{aligned} f(x) \cdot g(x) \in (P_{\alpha,f}(x - x_0) + I_{\alpha,f}) \cdot (P_{\alpha,g}(x - x_0) + I_{\alpha,g}) \subseteq \\ \subseteq (P_{\alpha,f}(x - x_0) \cdot P_{\alpha,g}(x - x_0)) + (P_{\alpha,f}(x - x_0) \cdot I_{\alpha,g}) + \\ + (P_{\alpha,g}(x - x_0) \cdot I_{\alpha,f}) + (I_{\alpha,f} \cdot I_{\alpha,g}). \end{aligned}$$

Bemerkung 3.5.

- *$P_{\alpha,f} \cdot P_{\alpha,g}$ ist ein Polynom der Ordnung $2n$. Es wird nun in zwei Teile geteilt, den Teil bis zur Ordnung n welcher mit dem n–ten Taylorpolynom von $P_{\alpha,f\cdot g}$ übereinstimmt und einem Extrapolynom P_e, sodass*

$$P_{\alpha,f}(x - x_0) \cdot P_{\alpha,g}(x - x_0) = P_{\alpha,f\cdot g}(x - x_0) + P_e(x - x_0).$$

- *$B(P)$ bezeichnet eine Schranke des Polynoms P: $[\underline{a}, \overline{a}] \subseteq \mathbb{R}^k \to \mathbb{R}$, sodass*

$$P(x) \in B(P) \quad \forall x \in [\underline{a}, \overline{a}].$$

Zusammengefasst heißt das

$$I_{\alpha,f\cdot g} = B(P_e) + B(P_{\alpha,f}) \cdot I_{\alpha,f} + B(P_{\alpha,g}) \cdot I_{\alpha,f} + I_{\alpha,f} \cdot I_{\alpha,g}.$$

Das ergibt für das Taylor-Modell von $f \cdot g$

$$T_{\alpha,f\cdot g} = T_{\alpha,f} \cdot T_{\alpha,g} = (P_{\alpha,f\cdot g}, I_{\alpha,f\cdot g}).$$

Bemerkung 3.6.

- *Für eine konstante Funktion $f(x) = t$ ergibt sich als Taylor-Modell $T_{\alpha,f} = T_{\alpha,t} = (P_{\alpha,f}, I_{\alpha,f}) = (t, [0, 0])$.*

- *Es kann nun also aus dem Taylor-Modell einer konstanten Funktion der Addition und Multiplikation von Taylor-Modellen, ein Taylor-Modell für jedes Polynom berechnet werden.*

3.3 Division von Taylor-Modellen

Ab nun wird der konstante Teil der Funktion f um x_0 als $c_{\alpha,f}$ und der restlichen Teil von f als $\bar{f}$ geschrieben, sodass

$$f(x) = c_{\alpha,f} + \bar{f}(x).$$

Es gilt

$$\begin{aligned}\frac{1}{f(x)} = \frac{1}{c_{\alpha,f} + \bar{f}(x)} &= \frac{1}{c_{\alpha,f}} \cdot \frac{1}{1 + \bar{f}(x)/c_{\alpha,f}} = \\ &= \frac{1}{c_{\alpha,f}} \cdot \left\{ 1 - \frac{\bar{f}(x)}{c_{\alpha,f}} + \frac{(\bar{f}(x))^2}{c_{\alpha,f}^2} - \cdots + (-1)^k \frac{(\bar{f}(x))^k}{c_{\alpha,f}^k} \right\} + \\ &+ (-1)^{k+1} \frac{(\bar{f}(x))^{k+1}}{c_{\alpha,f}^{k+2}} \frac{1}{(1 + \theta \cdot \bar{f}(x)/c_{\alpha,f})^{k+2}},\end{aligned}$$

wobei $0 < \theta < 1$.

Wird nun k so gewählt, dass der erste Term höchstens von Ordnung n ist, so ergibt sich ein Polynom von $\bar{f}$ und ein Restterm. Der Restterm wird wieder durch eine geeignete Abschätzung zu einem Intervallrestglied gemacht, und wie ein Taylor-Modell eines Polynoms berechnet wird ist bereits bekannt.

Theorem 3.7. *Sei $f : \mathbb{R}^n \to \mathbb{R}$ eine Funktion von der ein Taylor-Modell (P_f, I_f) gegeben ist. Sei $g : \mathbb{R} \to \mathbb{R}$ eine Funktion, gegeben in der Form $g(x) = g_k\big(g_{k-1}\big(\cdots(g_2\,(g_1(x)))\cdots\big)\big)$, wobei die Funktionen g_i, $i = 1, \ldots, k < \infty$ endlich viele Elementaroperationen bzw. intrinsische Funktionen sind. Sei $(P_0, I_0) = (P_f, I_f)$ und (P_j, I_j) ein Taylor-Modell für $g_j(P_{j-1} + I_{j-1})$, $j = 1, \ldots, k$, dann ist (P_k, I_k) ein Taylor-Modell für $g \circ f$.*

Beweis. Siehe [3], Seite 391. □

Bemerkung 3.8.

- *Die Notation in diesem Kapitel ist aus [2] übernommen.*
- *In [1], [2], [3] und [4] gibt es eine detailliertere Beschreibung der behandelten Themen. Insbesondere werden noch weitere Zusammensetzungen von Funktionen behandelt, sodass letztendlich, mittels Theorem 3.7., ein Taylor-Modell für jede Funktion, die aus Elementarfunktionen zusammengesetzt ist, berechnet werden kann.*

4 Padé-Approximation

4.1 Eindimensionale Padé-Approximation

In diesem Abschnitt wird eine kurze Einführung in die eindimensionale Padé-Approximation gegeben und gezeigt, wie man eine solche berechnet.

4.1.1 Problemstellung

Es wird angenommen, dass eine Potenzreihe gegeben ist, welche die zu betrachtende Funktion $f(x)$ repräsentiert

$$f(x) = \sum_{i=0}^{\infty} c_i x^i \qquad x \in \mathbb{R}.$$

Definition 4.1. *Eine Padé-Approximation ist eine rationale Funktion*

$$[L/M] = \frac{a_0 + a_1 x + \cdots + a_L x^L}{b_0 + b_1 x + \cdots + b_M x^M},$$

die eine MacLaurinsche Entwicklung besitzt, welche mit $f(x) = \sum_{i=0}^{\infty} c_i x^i$ so weit als möglich übereinstimmt.

Bemerkung 4.2. *$[L/M]$ besitzt in obiger Gleichung L+1 Zähler-Koeffizienten und M+1 Nenner-Koeffizienten. Setzt man $b_0 = 1$, so ergibt das insgesamt L+M+1 unbekannte Koeffizienten. Das deutet darauf hin, dass man die Padé-Approximation [L/M] so wählen kann, dass sie mit $f(x) = \sum_{i=0}^{\infty} c_i x^i$ bis zur Ordnung L+M übereinstimmt. Anders formuliert bedeutet das*

$$f(x) = \sum_{i=0}^{\infty} c_i x^i = \frac{a_0 + a_1 x + \cdots + a_L x^L}{b_0 + b_1 x + \cdots + b_M x^M} + O(x^{L+M+1}).$$

Im Folgenden werden nun die Koeffizienten $a_0, a_1, \ldots, a_L, b_0, b_1, \ldots, b_M$ berechnet, wobei nur der Fall L=2, M=2 behandelt wird.

4.1.2 Berechnung der Padé-Approximation

Es werden nun a_i, b_i für $i = 0, 1, 2$ berechnet, sodass $\sum_{i=0}^{\infty} c_i x^i$ in möglichst hoher Ordnung mit $\frac{a_0+a_1x+a_2x^2}{b_0+b_1x+b_2x^2}$ übereinstimmt. Führt man einen Fehlerterm $\sum_{i=0}^{\infty} e_i x^i$ ein, so kann man dieses Problem folgendermaßen umformulieren.

Es sollen a_i, b_i für $i = 0, 1, 2$ so berechnet werden, dass die Koeffizienten des Fehlerterms bis in möglichst hohe Ordnung Null sind und

$$\sum_{i=0}^{\infty} c_i x^i = \frac{a_0 + a_1 x + a_2 x^2}{b_0 + b_1 x + b_2 x^2} + \sum_{i=0}^{\infty} e_i x^i.$$

Multipliziert man mit $(b_0 + b_1 x + b_2 x^2)$, so ergibt das

$$(b_0 + b_1 x + b_2 x^2).(c_0 + c_1 x + ...) = (a_0 + a_1 x + a_2 x^2) + \left(\sum_{i=0}^{\infty} \tilde{e}_i x^i \right), \qquad (1)$$

wobei

$$\sum_{i=0}^{\infty} \tilde{e}_i x^i = \sum_{i=0}^{\infty} e_i x^i \cdot (b_0 + b_1 x + b_2 x^2).$$

Ein Koeffizientenvergleich für die Ordnungen x^3 und x^4 liefert

$$\begin{aligned} x^3 : \quad & b_2 c_1 + b_1 c_2 + b_0 c_3 = \tilde{e}_3 \\ x^4 : \quad & b_2 c_2 + b_1 c_3 + b_0 c_4 = \tilde{e}_4. \end{aligned}$$

Wählt man jetzt $b_0 = 1$, $\tilde{e}_3 = 0$, $\tilde{e}_4 = 0$, so ist folgendes Gleichungssystem zu lösen

$$\begin{pmatrix} c_1 & c_2 \\ c_2 & c_3 \end{pmatrix} \begin{pmatrix} b_2 \\ b_1 \end{pmatrix} = \begin{pmatrix} -c_3 \\ -c_4 \end{pmatrix}.$$

Berechnet man die Inverse der Matrix so ergibt das

$$\begin{pmatrix} b_2 \\ b_1 \end{pmatrix} = \frac{1}{c_1 c_3 - c_2^2} \begin{pmatrix} c_3 & -c_2 \\ -c_2 & c_1 \end{pmatrix} \begin{pmatrix} -c_3 \\ -c_4 \end{pmatrix} = \frac{1}{c_1 c_3 - c_2^2} \begin{pmatrix} -c_3^2 + c_2 c_4 \\ c_2 c_3 - c_1 c_4 \end{pmatrix}.$$

Für die Koeffizienten b_1 und b_2 erhält man also

$$\begin{aligned} b_1 &= \frac{c_2 c_3 - c_1 c_4}{c_1 c_3 - c_2^2} \\ b_2 &= \frac{c_2 c_4 - c_3^2}{c_1 c_3 - c_2^2}, \end{aligned}$$

unter der Voraussetzung $c_1 c_3 \neq c_2^2$.

Aus Gleichung (1) folgt mittels Koeffizientenvergleich für die Ordnungen x^0, x^1, x^2

$$\begin{aligned} a_0 + \tilde{e}_0 &= c_0 \\ a_1 + \tilde{e}_1 &= c_1 + b_1 c_0 \\ a_2 + \tilde{e}_2 &= c_2 + b_1 c_1 + b_2 c_0. \end{aligned}$$

Setzt man nun

$$\tilde{e}_0 = \tilde{e}_1 = \tilde{e}_2 = 0$$
$$p(x) = a_0 + a_1 x + a_2 x^2$$
$$q(x) = 1 + b_1 x + b_2 x^2,$$

so gilt

$$f(x) = \frac{p(x) + (c_5 + c_4 b_1 + c_3 b_2)x^5 + (c_6 + c_5 b_1 + c_4 b_2)x^6 + \cdots}{q(x)}.$$

Daraus folgt

$$f(x) = \frac{p(x)}{q(x)} + O(x^5).$$

Bemerkung 4.3.

- *Die Notation wurde aus [6] übernommen.*
- *Genauer behandelt wird dieses Thema in [5] und [6]. Insbesondere werden die Fälle für M und L beliebig betrachtet. Außerdem findet man darin einen Beweis der Annahme in Bemerkung 4.2.. Weiters werden Anwendungen, numerische Methoden und Konvergenzverhalten beschrieben.*

4.2 Zweidimensionale Padé-Approximation

Es wird nun versucht, analog zur eindimensionalen Padé-Approximation, für Funktionen in zwei Variablen die entsprechenden Padé-Approximationen zu berechnen.
Aus Notationsgründen ist folgende Definition sinnvoll.

Definition 4.4. *Ein Multiindex $\alpha = (\alpha_1, \ldots, \alpha_n)$ ist ein n-Tupel von Zahlen $\alpha_i \in \mathbb{N}_0$. Er hat die Ordnung $|\alpha| = \alpha_1 + \alpha_2 + \ldots + \alpha_n$.*
Für $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$ gelte $x^\alpha := x_1^{\alpha_1} \cdot x_2^{\alpha_2} \cdots x_n^{\alpha_n}$.

4.2.1 Problemstellung

Ausgegangen wird wieder von einer bekannten Potenzreihe für die zu behandelnde Funktion $f(x, y)$. Das heißt, die Koeffizienten $c_{i,j}$ sind für alle i, j gegeben, sodass

$$f(x, y) = \sum_{(i,j)=(0,0)}^{\infty} c_{i,j} x^i y^j \qquad (x, y) \in \mathbb{R}^2.$$

Wie schon im Eindimensionalen wird versucht, die Koeffizienten von [L/M] zu bestimmen, sodass $[L/M]$ mit $f(x, y)$ in möglichst hoher Ordnung übereinstimmen. [L/M] wird im Zweidimensionalen folgendermaßen dargestellt:

$$[L/M] = \frac{\sum_{|\alpha|=0}^{L} a_\alpha z^\alpha}{\sum_{|\alpha|=0}^{M} b_\alpha z^\alpha}.$$

Im Folgenden wird abermals nur der Fall L=M=2 behandelt.

4.2.2 Berechnung der Padé-Approximation

Nun werden die Koeffizienten $a_{i,j}$, $b_{i,j}$ für $|(i,j)| = 0,1,2$ berechnet, das heißt für den Fall L=2, M=2.
Die Berechnung startet ausgehend von folgender Gleichung

$$\left(\sum_{|(i,j)|=0}^{\infty} c_{ij}x^iy^j\right)\cdot\left(\sum_{|(i,j)|=0}^{2} b_{ij}x^iy^j\right) = \left(\sum_{|(i,j)|=1}^{2} a_{ij}x^iy^j\right) + \left(\sum_{|(i,j)|=0}^{\infty} e_{ij}x^iy^j\right), \quad (2)$$

wobei $\sum_{|(i,j)|=0}^{\infty} e_{ij}x^iy^j$ wieder einen Fehlerterm darstellt. Man versucht diesen in möglichst hoher Ordnung Null zu setzen.

Bemerkung 4.5. *Im Folgenden werden die Koeffizienten $a_{i,j}$, $b_{i,j}$ für $|(i,j)| = 0,1,2$ bestimmt, indem ein Koeffizientenvergleich der obigen Gleichung für die Ordnungen $|(i,j)| = 0$, $|(i,j)| = 1$, $|(i,j)| = 2$, (i,j)=(4,0) durchgeführt wird.*

Ein Koeffizientenvergleich der Gleichung (2) ergibt

$$\begin{aligned}
(0,0): &\quad c_{00}b_{00} = a_{00} + e_{00}\\
(1,0): &\quad c_{00}b_{10} + b_{00}c_{10} = a_{10} + e_{10}\\
(0,1): &\quad c_{00}b_{01} + b_{00}c_{01} = a_{01} + e_{01}\\
(1,1): &\quad c_{00}b_{11} + b_{00}c_{11} + b_{10}c_{01} + b_{01}c_{10} = a_{11} + e_{11}\\
(2,0): &\quad c_{00}b_{20} + b_{00}c_{20} + b_{10}c_{10} = a_{20} + e_{20}\\
(0,2): &\quad c_{00}b_{02} + b_{00}c_{02} + b_{01}c_{01} = a_{02} + e_{02}
\end{aligned} \quad (3)$$

$$\begin{aligned}
(3,0): &\quad b_{00}c_{30} + b_{10}c_{20} + b_{20}c_{10} = e_{30}\\
(0,3): &\quad b_{00}c_{03} + b_{01}c_{02} + b_{02}c_{01} = e_{03}\\
(2,1): &\quad b_{00}c_{21} + b_{10}c_{11} + b_{01}c_{20} + b_{11}c_{10} + b_{20}c_{01} = e_{21}\\
(1,2): &\quad b_{00}c_{12} + b_{01}c_{11} + b_{10}c_{02} + b_{11}c_{01} + b_{02}c_{10} = e_{12}\\
(4,0): &\quad b_{00}c_{40} + b_{10}c_{30} + b_{20}c_{20} = e_{40}.
\end{aligned} \quad (4)$$

Nun wählt man

$$\begin{aligned}
&b_{0,0} = 1\\
&e_{i,j} = 0 \quad \text{für} \quad (i = 0,\dots,3; j = 3-i), (i,j) = (4,0).
\end{aligned}$$

Somit ergeben die Gleichungen (3) und die Gleichungen (4) zwei lineare Gleichungssysteme aus denen man $b_{i,j}$ bzw. $a_{i,j}$ für alle i,j erhält.

Insgesamt errechnet sich also

$$f(x,y)\underbrace{\left(\sum_{|(i,j)|=0}^{2} b_{ij}x^iy^j\right)}_{=:q(x,y)} - \underbrace{\left(\sum_{|(i,j)|=1}^{2} a_{ij}x^iy^j\right)}_{=:p(x,y)} = \left(\sum_{|(i,j)|=4}^{\infty} e_{ij}x^iy^j\right).$$

Für $z = (x, y)$ folgt daraus

$$f(z) = \frac{p(z)}{q(z)} + O(z^{|\alpha|=4}).$$

Bemerkung 4.6.

- *Analog zu den ein- und zweidimensionalen Padé-Approximationen können auch für Funktionen in mehr als zwei Variablen Padé-Approximationen berechnet werden. Allerdings sollte nicht vergessen werden, dass für n Variablen ein Gleichungssystem für* $\left(1 + n + \frac{n(n+1)}{2}\right) + \left(n + \frac{n(n+1)}{2}\right) = 1 + 3n + n^2$ *Unbekannte gelöst werden muss.*

- *Eine etwas abgeänderte Methode um eine Padé-Approximation für eine Funktion in zwei Variablen zu berechnen ist in [7], Kapitel 1.4 beschrieben.*

5 Eindimensionale Padé-Modelle

Ausgehend von eindimensionalen Funktionen ist es Ziel dieses Kapitels, ein dem Taylor-Modell ähnliches Modell zu entwickeln, das anstatt von Polynomen, mit rationalen Funktionen arbeitet. Wie im vorigen Kapitel zu sehen war, kann man mittels einer Padé-Approximation solche rationale Funktionen berechnen. Allerdings ist nun keine Potenzreihenentwicklung der zu behandelnden Funktion bekannt. Die Idee ist also, wie schon beim Taylor-Modell, die Polynome für einfache Funktionen zu berechnen und dann Methoden zu entwickeln, welche sich mit den Elementaroperationen von Padé-Modellen beschäftigen. Somit kann man eine Funktion in diese Elementarfunktionen zerlegen, und ein Padé-Modell dafür berechnen.
In diesem Abschnitt sind alle diese Polynome vom Grad zwei. Außerdem werden nur die vier Grundoperationen behandelt.

5.1 Padé-Addition

Als erstes wird die Addition von Padé-Modellen betrachtet und folgendes Problem versucht zu lösen.

Problemstellung

Gegeben seien ein Intervall $\mathbf{x} = [\underline{x}, \overline{x}]$ und die Polynome

$$\begin{aligned} p_1(x) &= [\underline{a_0}, \overline{a_0}] + a_1 x + a_2 x^2 \\ q_1(x) &= 1 + b_1 x + b_2 x^2 \\ p_2(x) &= [\underline{c_0}, \overline{c_0}] + c_1 x + c_2 x^2 \\ q_2(x) &= 1 + d_1 x + d_2 x^2. \end{aligned}$$

Es werden zwei Polynome

$$\begin{aligned} p_3(x) &= [\underline{y_0}, \overline{y_0}] + y_1 x + y_2 x^2 \\ q_3(x) &= 1 + z_1 x + z_2 x^2 \end{aligned}$$

vom Grad zwei gesucht, die folgenden Zusammenhang erfüllen

$$\frac{p_1(x)}{q_1(x)} + \frac{p_2(x)}{q_2(x)} \subseteq \frac{p_3(x)}{q_3(x)} \quad \forall x \in \mathbf{x}.$$

Als nächstes definiert man zwei Hilfspolynome. Die Koeffizienten werden so gewählt, dass das Intervall $\mathbf{y_0} - \mathrm{mid}(\mathbf{y_0}) = [\underline{y_0}, \overline{y_0}] - \mathrm{mid}(\mathbf{y_0})$, welches auch als Restintervall von $p_3(x)$ bezeichnet wird, möglichst klein wird. Dazu muss ein lineares Gleichungssystem gelöst werden.

Seien

$$\begin{aligned} r_1(x) &= v_0 + v_1 x + v_2 x^2 + v_3 x^3 + v_4 x^4 \\ r_2(x) &= w_0 + w_1 x + w_2 x^2 + w_3 x^3 + w_4 x^4 \\ \tilde{p}_1 &= \mathrm{mid}(\mathbf{a_0}) + a_1 x + a_2 x^2 \\ \tilde{p}_2 &= \mathrm{mid}(\mathbf{c_0}) + c_1 x + c_2 x^2 \\ \tilde{p}_3 &= \mathrm{mid}(\mathbf{y_0}) + y_1 x + y_2 x^2, \end{aligned}$$

sodass

$$q_3(x)\tilde{p}_1(x) \approx q_1(x) r_1(x) \tag{5}$$
$$q_3(x)\tilde{p}_2(x) \approx q_2(x) r_2(x). \tag{6}$$

Hier bedeutet $\approx$, dass r_1, r_2 so gewählt werden, dass (5), (6) bis in möglichst hohe Ordnung übereinstimmen.

Seien $a_0 = \mathrm{mid}(\mathbf{a_0})$, $c_0 = \mathrm{mid}(\mathbf{c_0})$, $y_0 = \mathrm{mid}(\mathbf{y_0})$. Man kann (5) und (6) umschreiben zu

$$\begin{aligned} \left(1 + \sum_{i=1}^{2} z_i x^i\right) \cdot \left(\sum_{i=0}^{2} a_i x^i\right) &\approx \left(1 + \sum_{i=1}^{2} b_i x^i\right) \cdot \left(\sum_{i=0}^{4} v_i x^i\right) \\ \left(1 + \sum_{i=1}^{2} z_i x^i\right) \cdot \left(\sum_{i=0}^{2} c_i x^i\right) &\approx \left(1 + \sum_{i=1}^{2} d_i x^i\right) \cdot \left(\sum_{i=0}^{4} w_i x^i\right). \end{aligned}$$

Führt man nun einen Koeffizientenvergleich der Gleichungen (5) und (6) durch, so ergibt das

$$\begin{aligned} a_0 &= v_0 \\ a_1 + a_0 z_1 &= v_1 + v_0 b_1 \\ a_2 + a_0 z_2 + a_1 z_1 &= v_2 + v_0 b_2 + v_1 b_1 \\ a_1 z_2 + a_2 z_1 &= v_3 + v_1 b_2 + v_2 b_1 \\ a_2 z_2 &= v_4 + v_3 b_1 + v_2 b_2 \end{aligned} \tag{7}$$

$$\begin{aligned} c_0 &= w_0 \\ c_1 + c_0 z_1 &= w_1 + w_0 d_1 \\ c_2 + c_0 z_2 + c_1 z_1 &= w_2 + w_0 d_2 + w_1 d_1 \\ c_1 z_2 + c_2 z_1 &= w_3 + w_1 d_2 + w_2 d_1 \\ c_2 z_2 &= w_4 + w_3 d_1 + w_2 d_2. \end{aligned} \tag{8}$$

r_1,r_2 und q_3 haben insgesamt also 12 Koeffizienten. Um diese zu bestimmen verwendet man folgende 12 Gleichungen. Die Gleichungen (7) und (8) aus dem Koeffizientenvergleich und die beiden zusätzlichen Gleichungen

$$v_3 + w_3 = 0 \tag{9}$$
$$v_4 + w_4 = 0. \tag{10}$$

Warum gerade diese Gleichungen gewählt werden, wird in Bemerkung 5.1. geklärt.

Bemerkung 5.1. *Man kann* $\left(\frac{p_1(x)}{q_1(x)} + \frac{p_2(x)}{q_2(x)}\right) q_3(x)$ *darstellen als*

$$\left(\frac{p_1(x)}{q_1(x)} + \frac{p_2(x)}{q_2(x)}\right) q_3(x) = r_1(x) + r_2(x) + \\ + \frac{q_3(x)p_1(x) - q_1(x)r_1(x)}{q_1(x)} + \frac{q_3(x)p_2(x) - q_2(x)r_2(x)}{q_2(x)}.$$

Erst in dieser Darstellung wird klar, warum die Gleichungen (9) und (10) verwendet werden, um die Koeffizienten von r_1, r_2 *und* q_3 *zu bestimmen. Denn mit dieser Wahl verschwinden die Koeffizienten von* x^3 *und* x^4 *in der Reihendarstellung von* $\left(\frac{p_1(x)}{q_1(x)} + \frac{p_2(x)}{q_2(x)}\right) q_3(x)$. *Um ein Restintervall zu erhalten, muss also ein Polynom der Form* $c_5x^5 + c_6x^6 + \cdots$ *abgeschätzt werden.*
Würde man anstatt der Gleichungen (9) und (10) etwa die Gleichungen $b_2v_3 + b1v_4 = 0$ *und* $d_2w_3 + d_1w_4 = 0$ *verwenden, welche dem Koeffizientenvergleich der Gleichungen (5) und (6) für* x^5 *entsprechen, so müsste man für das Restintervall ein Polynom der Form* $c_3x^3 + c_4x^4 + c_5x^5 + c_6x^6 + \cdots$ *abschätzen. Lässt man die Intervallbreite von* $\mathbf{x}$ *gegen Null gehen, so würde das Restintervall in diesem Fall wesentlich langsamer gegen Null konvergieren.*

Theorem 5.2. *Löst man das lineare Gleichungssystem der Gleichungen (7)–(10) so erhält man folgendes Ergebnis*

$$\begin{aligned} z_1 =& (a_2b_1 - a_1b_1^2 + a_0b_1^3 + a_1b_2 - 2a_0b_1b_2 + c_2d_1 - c_1d_1^2 + c_0d_1^3 + c_1d_2 - 2c_0d_1d_2 + \\ & ((a_1 - a_0b_1 + c_1 - c_0d_1)(-(b_1(a_2 + b_1(-a_1 + a_0b_1)) + (a_1 - 2a_0b_1)b_2 + \\ & d_1(c_2 + d_1(-c_1 + c_0d_1)) + (c_1 - 2c_0d_1)d_2)^2 + \\ & (a_2 - a_1b_1 + a_0b_1^2 - a_0b_2 + c_2 - c_1d_1 + c_0d_1^2 - c_0d_2)(b_1^2(a_2 + b_1(-a_1 + a_0b_1)) - \\ & (a_2 + b_1(-2a_1 + 3a_0b_1))b_2 + a_0b_2^2 + d_1^2(c_2 + d_1(-c_1 + c_0d_1)) - \\ & (c_2 + d_1(2c_1 + 3c_0d_1))d_2 + c_0d_2^2))) / \\ & ((a_2 - a_1b_1 + a_0b_1^2 - a_0b_2 + c_2 - c_1d_1 + c_0d_1^2 - c_0d_2)^2 - (a_1 - a_0b_1 + c_1 - c_0d_1) \\ & (-a_2b_1 + a_1b_1^2 - a_0b_1^3 - a_1b_2 + 2a_0b_1b_2 - c_2d_1 + c_1d_1^2 - c_0d_1^3 - c_1d_2 + 2c_0d_1d_2))) / \\ & (a_2 - a_1b_1 + a_0b_1^2 - a_0b_2 + c_2 - c_1d_1 + c_0d_1^2 - c_0d_2) \end{aligned}$$

$$
\begin{aligned}
z_2 =& -(-(b_1(a_2+b_1(-a_1+a_0b_1))+(a_1-2a_0b_1)b_2+d_1(c_2+d_1(-c_1+c_0d_1))+\\
&(c_1-2c_0d_1)d_2)^2+(a_2-a_1b_1+a_0b_1^2-a_0b_2+c_2-c_1d_1+c_0d_1^2-c_0d_2)\\
&(b_1^2(a_2+b_1(-a_1+a_0b_1))-(a_2+b_1(-2a_1+3a_0b_1))b_2+\\
&a_0b_2^2+d_1^2(c_2+d_1(-c_1+c_0d_1))-(c_2+d_1(-2c_1+3c_0d_1))d_2+c_0d_2^2))/\\
&((a_2-a_1b_1+a_0b_1^2-a_0b_2+c_2-c_1d_1+c_0d_1^2-c_0d_2)^2-(a_1-a_0b_1+c_1-c_0d_1)\\
&(-a_2b_1+a_1b_1^2-a_0b_1^3-a_1b_2+2a_0b_1b_2-c_2d_1+c_1d_1^2-c_0d_1^3-c_1d_2+c_0d_1d_2))
\end{aligned}
$$

$$
\begin{aligned}
v_4 =& a_2(b_1^2-b_2-b_1z_1+z_2)+a_0(b_1^4-b_1^3z_1+2b_1b_2z_1+b_2(b_2-z_2)+\\
&b_1^2(-3b_2+z_2))-a_1(b_1^3-b_1^2z_1+b_2z_1+b_1(-2b_2+z_2))
\end{aligned}
$$

$$
v_3 = a_2(-b_1+z_1)+a_1(b_1^2-b_2-b_1z_1+z_2)-a_0(b_1^3-2b_1b_2-b_1^2z_1+b_2z_1+b_1z_2)
$$

$$
v_2 = a_2+a_1(-b_1+z_1)+a_0(b_1^2-b_2-b_1z_1+z_2)
$$

$$
v_1 = a_1+a_0(-b_1+z_1)
$$

$$
v_0 = a_0
$$

$$
\begin{aligned}
w_4 =& c_2(d_1^2-d_2-d_1z_1+z_2)+c_0(d_1^4-d_1^3z_1+2d_1d_2z_1+d_2(d_2-z_2)+\\
&d_1^2(-3d_2+z_2))-c_1(d_1^3-d_1^2z_1+d_2z_1+d_1(-2d_2+z_2))
\end{aligned}
$$

$$
w_3 = c_2(-d_1+z_1)+c_1(d_1^2-d_2-d_1z_1+z_2)-c_0(d_1^3-2d_1d_2-d_1^2z_1+d_2z_1+d_1z_2)
$$

$$
w_2 = c_2+c_1(-d_1+z_1)+c_0(d_1^2-d_2-d_1z_1+z_2)
$$

$$
w_1 = c_1+c_0(-d_1+z_1)
$$

$$
w_0 = c_0.
$$

Durch das Lösen des Gleichungssystems ergibt sich

$$
\begin{aligned}
q_3(x)p_1(x)-q_1(x)r_1(x) &= (\mathbf{a}_0-a_0)(1+z_1x+z_2x^2)-(b_2v_3+b_1v_4)x^5-(b_2v_4)x^6\\
q_3(x)p_2(x)-q_2(x)r_2(x) &= (\mathbf{c}_0-c_0)(1+z_1x+z_2x^2)-(d_2w_3+d_1w_4)x^5-(d_2w_4)x^6.
\end{aligned}
$$

Insgesamt erhält man das folgende Theorem.

Theorem 5.3. *Lässt sich das Gleichungssystem (7)-(10) lösen, so ergibt sich für, mittels Theorem 5.2. berechnete Koeffizienten*

$$\frac{p_1(x)}{q_1(x)} + \frac{p_2(x)}{q_2(x)} \subseteq \frac{(v_0 + w_0 + I_1(x) + I_2(x)) + (v_1 + w_1)x + (v_2 + w_2)x^2}{1 + z_1 x + z_2 x^2} \quad \forall x \in \mathbf{x},$$

wobei

$$I_1(x) = \frac{(\mathbf{a_0} - a_0)(1 + z_1 x + z_2 x^2) - (b_2 v_3 + b_1 v_4)x^5 - (b_2 v_4)x^6}{1 + b_1 x + b_2 x^2}$$
$$I_2(x) = \frac{(\mathbf{c_0} - c_0)(1 + z_1 x + z_2 x^2) - (d_2 w_3 + d_1 w_4)x^5 - (d_2 w_4)x^6}{1 + d_1 x + d_2 x^2}.$$

Korollar 5.4. *Unter der Voraussetzung, dass das Gleichungssystems (7)–(10) lösbar ist, kann man $p_3(x)$ und $q_3(x)$ wählen als*

$$p_3(x) = (v_0 + w_0) + (v_1 + w_1)x + (v_2 + w_2)x^2 + I_1(\mathbf{x}) + I_2(\mathbf{x})$$
$$q_3(x) = 1 + z_1 x + z_2 x^2.$$

Bemerkung 5.5. *Kann man z_1 und/oder z_2 nicht berechnen, da im entsprechenden expliziten Lösungsausdruck der Nenner gleich Null ist, so wird zur Lösung des Gleichungssystems (7)–(10) zunächst einer der beiden Koeffizienten z_1 oder z_2 gleich Null gesetzt und versucht unter dieser Annahme den anderen Koeffizienten aus einer der Gleichungen (9) oder (10) zu berechnen. Ist dies wieder nicht möglich, so wird auch der zweite Koeffizient Null gesetzt. Somit ist es kein Problem mehr die restlichen Koeffizienten laut Theorem 5.2. zu berechnen. Es ergibt sich in diesen Fällen allerdings das Problem, dass die Gleichungen, oder zumindest eine der Gleichungen (9), (10), nicht erfüllt ist und somit Theorem 5.3. und Korollar 5.4. nicht verwendet werden können. Können also z_1 und/oder z_2 nicht mittels Theorem 5.2. berechnet werden, so müssen die Terme $v_3 + w_3$ und $v_4 + w_4$ abgeschätzt werden, wodurch sich das Fehlerintervall vergrößert.*

Berechnet man die Koeffizienten für r_1, r_2 und q_3 wie in Bemerkung 5.5. beschrieben, so ergibt sich für $p_3(x)$ folgendes Fehlerintervall

$$(v_3 + w_3)\mathbf{x}^3 + (v_4 + w_4)\mathbf{x}^4 + I_1(\mathbf{x}) + I_2(\mathbf{x}).$$

Es wird nun versucht dieses Fehlerintervall etwas zu verkleinern. Die Idee dabei ist Folgende: Die Funktionen x^3 und x^4 sollen möglichst gut quadratisch approximiert werden. Diese Approximation wird dann von der jeweiligen Funktion subtrahiert. Somit wird nur der verringerte Fehler in das Restintervall miteingerechnet. Um an der Lösung nichts zu verfälschen, muss natürlich die Approximation noch zum Polynom $p_3(x)$ addiert werden.

In eine Gleichung geschrieben sieht das dann folgendermaßen aus

$$\begin{aligned}\left(\frac{p_1(x)}{q_1(x)}+\frac{p_2(x)}{q_2(x)}\right)q_3(x) \subseteq &(v_0+w_0+(v_3+w_3)k_0+(v_4+w_4)j_0)+\\ &+(v_1+w_1+(v_3+w_3)k_1+(v_4+w_4)j_1)x+\\ &+(v_2+w_2+(v_3+w_3)k_2+(v_4+w_4)j_2)x^2+\\ &+(v_3+w_3)(x^3-(k_0+k_1x+k_2x^2))+\\ &+(v_4+w_4)(x^4-(j_0+j_1x+j_2x^2))+I_1(\mathbf{x})+I_2(\mathbf{x}) \quad \forall x\in\mathbf{x},\end{aligned}$$

wobei k_0, k_1, k_2, j_0, j_1, j_2 so gewählt werden, dass

$$\begin{aligned}x^3-(k_0+k_1x+k_2x^2) &\leq x^3 \qquad \forall x\in\mathbf{x}\\ x^4-(j_0+j_1x+j_2x^2) &\leq x^4 \qquad \forall x\in\mathbf{x}.\end{aligned}$$

Sei $\mathbf{x}=[-h,h]$. So gilt $k_0=k_2=j_1=0$, da x^0,x^2,x^4 gerade Funktionen sind und x^1,x^3 ungerade Funktionen, und diese nur durch eine ebenfalls gerade bzw. ungerade Funktion gut approximiert werden können.

Die Wahl für k_1,j_0,j_2 ist nicht eindeutig. Es gilt allerdings Folgendes zu beachten. Erstens sollte man die Bestimmung dieser Koeffizienten abhängig von dem Intervall $\mathbf{x}=[-h,h]$ machen für das man sich interessiert. Und zweitens sollte der maximale Fehler in dem zu betrachtenden Teilintervall minimiert werden.

Approximation von x^3

Sei $\mathbf{x}=[-h,h]$, $0<h\in[\underline{h},\overline{h}[$. Man bestimmt k_1 so, dass folgende heuristische Gleichung erfüllt ist

$$\frac{\text{Maximaler Fehler ohne Korrektur im Intervall } [0,\underline{h}]}{\text{Maximaler Fehler mit Korrektur im Intervall } [0,\underline{h}]}=\frac{\text{Fehler ohne Korrektur bei } \bar{h}}{\text{Fehler mit Korrektur bei } \bar{h}}.$$

Betrachtet man Abbildung 1, so muss also folgende Gleichung erfüllt sein

$$\frac{B}{A} \quad = \quad \frac{D}{C}.$$

Weil der maximale Abstand von x^3 und k_1x im Intervall $[0,\underline{h}]$ bei $\sqrt{\frac{k_1}{3}}$ auftritt, löst man für $h\in[\underline{h},\overline{h}[$ folgende Gleichung nach k_1 auf

$$\frac{\underline{h}^3}{-(k_1/3)^{3/2}+k_1\sqrt{k_1/3}}=\frac{\bar{h}^3}{\bar{h}^3-k_1\bar{h}}. \tag{11}$$

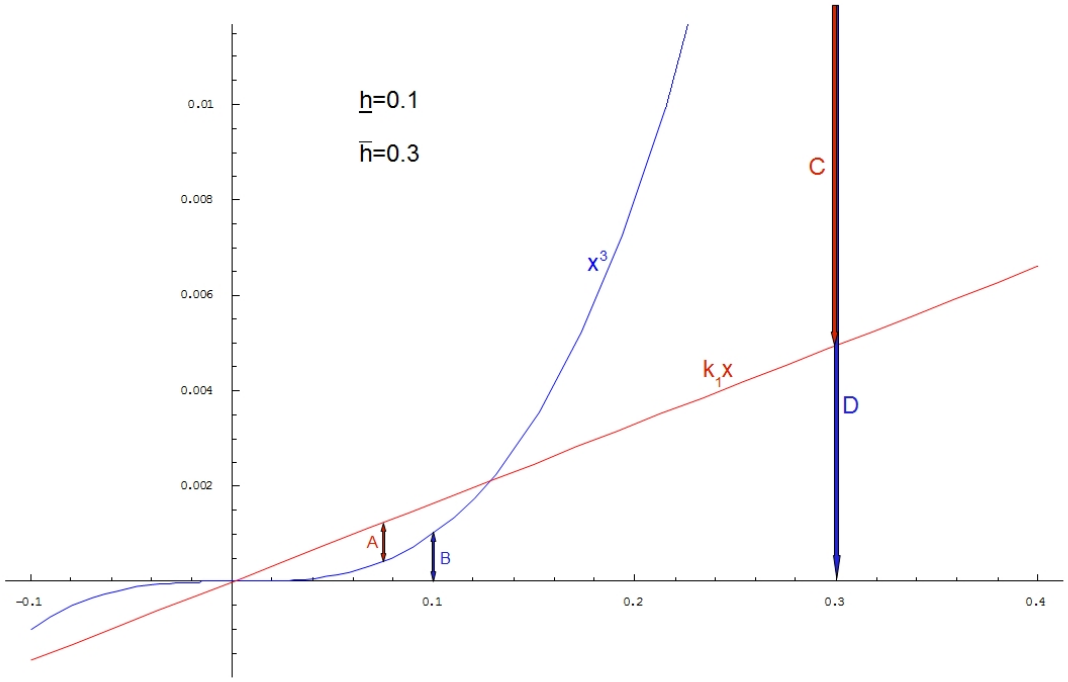

Abb. 1: Quadratische Approximation für x^3

Approximation von x^4

Sei $\mathbf{x} = [-h, h]$, $0 < h \in [\underline{h}, \overline{h}[$. Über folgende zwei heuristische Gleichungen bestimmt man j_0, j_2

Fehler mit Korrektur bei 0 $=$
Maximaler Fehler mit Korrektur zwischen dem kleineren positiven Schnittpunkt und $\underline{h}$

$$\frac{\text{Maximaler Fehler ohne Korrektur im Intervall } [0, \underline{h}]}{\text{Maximaler Fehler mit Korrektur im Intervall } [0, \underline{h}]} = \frac{\text{Fehler ohne Korrektur bei } \overline{h}}{\text{Fehler mit Korrektur bei } \overline{h}}.$$

Betrachtet man Abbildung 2, so müssen also folgende Gleichungen erfüllt sein

$$A = B$$
$$\frac{C}{B} = \frac{D}{E}.$$

Der maximale Abstand von x^4 und $j_0 + j_2 x^2$ zwischen deren kleineren positiven Schnittpunkt und $\underline{h}$ ist bei $\sqrt{\frac{j_2}{2}}$. Somit löst man für $h \in [\underline{h}, \overline{h}[$ folgende Gleichungen nach j_0, j_2 auf

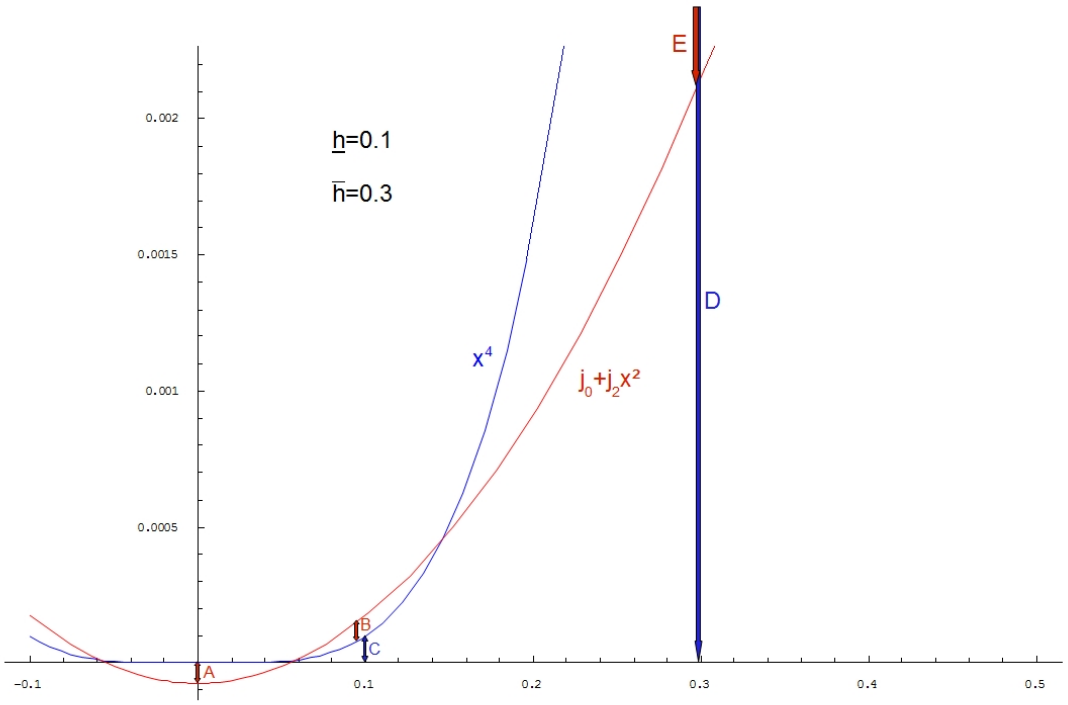

Abb. 2: Quadratische Approximation für x^4

$$
\begin{aligned}
j_0 &= \left(\min\left(\underline{h}, \sqrt{\frac{j_2}{2}}\right)\right)^4 - \left(j_0 + j_2\left(\min\left(\underline{h}, \sqrt{\frac{j_2}{2}}\right)\right)^2\right) \\
\frac{\underline{h}^4}{j_0} &= \frac{\bar{h}^4}{\bar{h}^4 - (j_0 + j_2\bar{h}^2)}.
\end{aligned} \tag{12}
$$

Vergleicht man nun den Fehler ohne Korrektur mit dem korrigierten Fehler, so ist leicht ersichtlich, dass folgender Zusammenhang gilt

$$
\begin{aligned}
&\text{Maximaler korrigierter Fehler für } x^4 = \text{ Maximaler Fehler ohne Korrektur für } x^4 \cdot \gamma_4 \\
&\text{Maximaler korrigierter Fehler für } x^3 = \text{ Maximaler Fehler ohne Korrektur für } x^3 \cdot \gamma_3,
\end{aligned}
$$

wobei

$$0 < \gamma_4 = \frac{\bar{h}^4 - (j_0 + j_2\bar{h}^2)}{\bar{h}^4} \leq 1 \tag{13}$$

$$0 < \gamma_3 = \frac{\bar{h}^3 - (k_1\bar{h})}{\bar{h}^3} \leq 1. \tag{14}$$

Liegt nun $h \in [\underline{h}, \bar{h}]$, so ergibt sich

$$\begin{aligned}\left(\frac{p_1(x)}{q_1(x)} + \frac{p_2(x)}{q_2(x)}\right) q_3(x) \subseteq &(v_0 + w_0 + (v_4 + w_4)j_0) + \\ &+(v_1 + w_1 + (v_3 + w_3)k_1)x + \\ &+(v_2 + w_2 + (v_4 + w_4)j_2)x^2 + \\ &+(v_3 + w_3)\gamma_3 x^3 + (v_4 + w_4)\gamma_4 x^4 + I_1(\mathbf{x}) + I_2(\mathbf{x}) \quad \forall x \in \mathbf{x}.\end{aligned}$$

Nun werden diese Koeffizienten und Faktoren für h aus folgenden Teilintervallen berechnet

$$[0.1, 0.3[, [0.3, 0.5[, \cdots, [1.7, 1.9[, [1.9, 20[.$$

Für $0 < h < 0.1$ setzt man alle Koeffizienten Null bzw. Faktoren Eins, da man für so kleine Werte kaum eine Verbesserung erzielen würde.

Löst man die Gleichungssysteme (11), (12) für $h \in [0.1, 0.3[$, so ergibt sich für die Koeffizienten

$$\begin{aligned}k_1 &= 0.0165104 \\ j_0 &= -0.0000734694 \\ j_2 &= 0.0246939.\end{aligned}$$

Berechnet man die Faktoren γ_3 und γ_4 laut (13), (14) so erhält man

$$\begin{aligned}\gamma_3 &= \frac{\bar{h}^3 - (k_1\bar{h})}{\bar{h}^3} = 0.81656 \\ \gamma_4 &= \frac{\bar{h}^4 - (j_0 + j_2\bar{h}^2)}{\bar{h}^4} = 0.734695.\end{aligned}$$

Analog erhält man die Koeffizienten und Faktoren für die restlichen Teilintervalle, welche in den Tabellen 1 und 2 angeführt sind.

	[0.3,0.5[	**[0.5,0.7[**	**[0.7,0.9[**	**[0.9,1.1[**	**[1.1,1.3[**
k_1	0.11358	0.273816	0.494207	0.774527	1.11478
j_0	-0.00327802	-0.0183771	-0.0585954	-0.141925	-0.291161
j_2	0.161939	0.383428	0.684663	1.06555	1.5262
γ_3	0.470589	0.441193	0.38987	0.359896	0.340368
γ_4	0.404693	0.294034	0.244047	0.216318	0.198868

Tabelle 1

	[1.3,1.5[	**[1.5,1.7[**	**[1.7,1.9[**	**[1.9,20[**
k_1	1.51497	1.97512	2.49525	6.74555
j_0	-0.5339	-0.902543	-1.43429	-12.6865
j_2	2.06669	2.68707	3.38737	10.638
γ_3	0.32669	0.316568	0.308796	0.98314
γ_4	0.186934	0.1782	0.17173	0.97349

Tabelle 2

5.2 Padé-Multiplikation

In diesem Teil wird nun versucht ein analoges Verfahren für die Mulitplikation zweier Padé-Modelle zu finden. Es ist also für folgendes Problem eine Lösung zu finden.

Problemstellung

Für ein Intervall $\mathbf{x} = [\underline{x}, \overline{x}]$ und die gegebenen Polynome $p_1(x), q_1(x), p_2(x), q_2(x)$ der Form

$$\begin{aligned} p_1(x) &= [\underline{a_0}, \overline{a_0}] + a_1 x + a_2 x^2 \\ q_1(x) &= 1 + b_1 x + b_2 x^2 \\ p_2(x) &= [\underline{c_0}, \overline{c_0}] + c_1 x + c_2 x^2 \\ q_2(x) &= 1 + d_1 x + d_2 x^2 \end{aligned}$$

soll eine Methode gefunden werden um zwei Polynome

$$\begin{aligned} p_3(x) &= [\underline{y_0}, \overline{y_0}] + y_1 x + y_2 x^2 \\ q_3(x) &= 1 + z_1 x + z_2 x^2 \end{aligned}$$

zu berechnen, welche folgende Gleichung erfüllen

$$\frac{p_1(x)}{q_1(x)} \cdot \frac{p_2(x)}{q_2(x)} \subseteq \frac{p_3(x)}{q_3(x)} \quad \forall x \in \mathbf{x}.$$

Wie schon zuvor, definiert man zunächst wieder zwei Hilfspolynome, deren Koeffizienten so gewählt werden, dass am Schluss das entstehende Fehlerintervall möglichst klein wird. Seien

$$\begin{aligned} r_1(x) &= v_0 + v_1 x + v_2 x^2 + v_3 x^3 + v_4 x^4 \\ r_2(x) &= w_0 + w_1 x + w_2 x^2 + w_3 x^3 + w_4 x^4 \end{aligned}$$

Polynome, sodass

$$q_3(x)\tilde{p}_1(x) \approx q_1(x) r_1(x) \tag{15}$$

$$\tilde{p}_2(x) \approx q_2(x) r_2(x). \tag{16}$$

Auch hier bedeute $\approx$ wieder, dass die beiden Terme bis in möglichst hohe Ordnung übereinstimmen sollen, d.h.

$$\left(1+\sum_{i=1}^{2} z_i x^i\right)\cdot\left(\sum_{i=0}^{2} a_i x^i\right) \approx \left(1+\sum_{i=1}^{2} b_i x^i\right)\cdot\left(\sum_{i=0}^{4} v_i x^i\right)$$
$$\left(\sum_{i=0}^{2} c_i x^i\right) \approx \left(1+\sum_{i=1}^{2} d_i x^i\right)\cdot\left(\sum_{i=0}^{4} w_i x^i\right).$$

Durch einen Koeffizientenvergleich der Gleichungen (15) und (16) kommt man zu folgendem Gleichungssystem

$$\begin{aligned} a_0 &= v_0 \\ a_1 + a_0 z_1 &= v_1 + v_0 b_1 \\ a_2 + a_0 z_2 + a_1 z_1 &= v_2 + v_0 b_2 + v_1 b_1 \\ a_1 z_2 + a_2 z_1 &= v_3 + v_1 b_2 + v_2 b_1 \\ a_2 z_2 &= v_4 + v_3 b_1 + v_2 b_2 \end{aligned} \tag{17}$$

$$\begin{aligned} c_0 &= w_0 \\ c_1 &= w_1 + w_0 d_1 \\ c_2 &= w_2 + w_0 d_2 + w_1 d_1 \\ 0 &= w_3 + w_1 d_2 + w_2 d_1 \\ 0 &= w_4 + w_3 d_1 + w_2 d_2. \end{aligned} \tag{18}$$

Die Polynome $r_1(x)$, $r_2(x)$ und $q_3(x)$ besitzen in Summe 12 Koeffizienten welche zu bestimmen sind. Um diese zu berechnen, verwendet man folgende 12 Gleichungen.

Aus den Gleichungen (18) des Koeffizientenvergleichs von (16) bis zur 4. Ordnung erhält man die Koeffizienten von r_2. Aus den Gleichungen (17) des Koeffizientenvergleichs von (15) bis zur 4. Ordnung und den beiden Gleichungen

$$v_0 w_3 + v_1 w_2 + v_2 w_1 + v_3 w_0 = 0 \tag{19}$$
$$v_0 w_4 + v_1 w_3 + v_2 w_2 + v_3 w_1 + v_4 w_0 = 0 \tag{20}$$

erhält man die Koeffizienten von r_1 und q_3.

Bemerkung 5.6. *Der Grund für die Wahl der Gleichungen (19) und (20) lässt sich am besten anhand folgender Darstellung von* $\left(\frac{p_1(x)}{q_1(x)}\cdot\frac{p_2(x)}{q_2(x)}\right) q_3(x)$ *erklären*

$$\left(\frac{p_1(x)}{q_1(x)}\cdot\frac{p_2(x)}{q_2(x)}\right) q_3(x) = r_1(x)r_2(x) + \frac{r_1(x)}{q_2(x)}(p_2(x) - q_2(x)r_2(x)) + \\ + \frac{p_2(x)}{q_1(x)q_2(x)}(q_3(x)p_1(x) - q_1(x)r_1(x)).$$

Da

$$\begin{aligned} r_1(x)r_2(x) = (v_0+w_0) &+ (v_0w_1+v_1w_0)x+ \\ &+ (v_0w_2+v_1w_1+v_2w_0)x^2+ \\ &+ (v_0w_3+v_1w_2+v_2w_1+v_3w_0)x^3+ \\ &+ (v_0w_4+v_1w_3+v_2w_2+v_3w_1+v_4w_0)x^4+ \\ &+ (v_1w_4+v_2w_3+v_3w_2+v_4w_1)x^5+ \\ &+ (v_2w_4+v_3w_3+v_4w_2)x^6+ \\ &+ (v_3w_4+v_4w_3)x^7+ \\ &+ (v_4w_4)x^8, \end{aligned}$$

sind mit dieser Wahl der Gleichungen die Koeffizienten von x^3 und x^4 von $\left(\frac{p_1(x)}{q_1(x)}+\frac{p_2(x)}{q_2(x)}\right)q_3(x)$ gleich Null, wodurch das Restintervall mit abnehmender Intervallbreite von $\mathbf{x}$ *schneller gegen Null konvergiert.*

Theorem 5.7. *Das Gleichungssystem (17)-(20) hat die Lösung*

$$\begin{aligned} z_1 =&(a_2^2c_0(-c_1)+c_0(b_1+d_1))+a_1^2(c_1^2d_1+b_1^2c_0(-c_1+c_0d_1)-c_1(c_2+c_0d_1^2)+ \\ &b_1(b_2c_0^2+(c_1-c_0d_1)^2)+c_0^2d_1d_2)-a_0a_1(b_1^3c_0(-c_1+c_0d_1)+b_1(2c_1-c_0d_1) \\ &(-c_2+d_1(c_1-c_0d_1))-(b_2c_0-c_2+c_1d_1)(c_2+d_1(-c_1+c_0d_1))+ \\ &b_1^2(c_1^2+c_0(c_2-3c_1d_1+c_0(b_2+2d_1^2-d_2)))+b_1c_0c_1d_2+c_0(-c_2+c_0(b_2+d_1^2))d_2)+ \\ &a_2(-a_1(c_1-c_0(b_1+d_1))^2+a_0c_0(b_1^3c_0+b_2(c_1-c_0d_1) \\ &b_1^2(-c_1+c_0d_1)+d_1(c_2-c_1d_1+c_0d_1^2)+b_1(c_2-c_1d_1+c_0(-2b_2+d_1^2-d_2))+ \\ &(c_1-2c_0d_1)d_2))+a_0^2(-(b_2c_0-c_2)d_1(c_2+d_1(-c_1+c_0d_1))+ \\ &b_1^2(-c_1+c_0d_1)((c_2-c_1d_1+c_0(b_2+d_1^2-d_2))+(b_2c_0(-c_1+2c_0d_1)+ \\ &d_1(c_1^2-2c_0c_2-c_0c_1d_1))d_2+c_0^2d_1d_2^2+b_1^3c_0(c_2-c_1d_1+c_0d_1^2-c_0d_2)+ \\ &b_1(b_2^2c_0^2+(c_2-c_1d_1)(c_2-c_1d_1+c_0d_1^2)+ \\ &c_0(-c_2+c_0d_1^2)d_2+b_2(c_1^2-c_0(2c_2+c_0d_1^2-2c_0d_2)))))/ \\ &(a_2^2c_0^2+a_2(-a_1c_0(-c_1+c_0(b_1+d_1))+ \\ &a_0(-c_1^2+c_0(2c_2+c_0(b_1^2-2b_2+d_1^2-2d_2))))+a_1^2(b_2c_0^2+c_1^2-c_0c_2-c_0c_1d_1+ \\ &b_1c_0(-c_1+c_0d_1)+c_0^2d_2)-a_0a_1(c_1^2d_1+b_1^2c_0(-c_1+c_0d_1)-c_1(c_2+c_0d_1^2)+ \\ &b_1(b_2c_0^2+(c_1-c_0d_1)^2)+c_0^2d_1d_2)+a_0^2(b_2^2c_0^2+(b_1^2c_0-b_1c_1+c_2)(c_2+ \\ &d_1(-c_1+c_0d_1))-(-c_1^2+c_0(b_1^2c_0+2c_2-b_1c_0d_1+c_1d_1))d_2+c_0^2d_2^2+ \\ &b_2(c_1^2+b_1c_0(-c_1+c_0d_1)- \\ &c_0(2c_2+c_0d_1^2-2c_0d_2)))) \end{aligned}$$

$$
\begin{aligned}
z_2 = &- (-(a_2(c_1 - c_0(b_1 + d_1)) - a_0(b_1^3 c_0 + b_2(c_1 - c_0 d_1) + b_1^2(-c_1 + c_0 d_1) + \\
&d_1(c_2 - c_1 d_1 + c_0 d_1^2) + b_1(c_2 - c_1 d_1 + c_0(-2b_2 + d_1^2 - d_2)) + (c_1 - 2c_0 d_1)d_2) + \\
&a_1(b_1^2 c_0 + c_2 - c_1 d_1 + b_1(-c_1 + c_0 d_1) - c_0(b_2 - d_1^2 + d_2)))^2 + \\
&(a_2 c_0 + a_1(c_1 - c_0(b_1 + d_1)) + a_0(b_1^2 c_0 + c_2 - c_1 d_1 + b_1(-c_1 + c_0 d_1) - \\
&c_0(b_2 - d_1^2 + d_2)))(-a_1 b_1^3 c_0 + b_2(c_1 - c_0 d_1) + b_1^2(-c_1 + c_0 d_1) + \\
&d_1(c_2 - c_1 d_1 + c_0 d_1^2) + b_1(c_2 - c_1 d_1 + c_0(-2b_2 + d_1^2 - d_2)) + (c_1 - 2c_0 d_1)d_2) + \\
&a_2(b_1^2 c_0 + c_2 - c_1 d_1 + b_1(-c_1 + c_0 d_1) - c_0(b_2 - d_1^2 + d_2)) + \\
&a_0(b_1^4 c_0 + b_2^2 c_0 + b_1^3(-c_1 + c_0 d_1) + d_1^2(c_2 - c_1 d_1 + c_0 d_1^2) + \\
&b_1^2(c_2 - c_1 d_1 + c_0(-3b_2 + d_1^2 - d_2)) - (c_2 - 2c_1 d_1 + 3c_0 d_1^2)d_2 + c_0 d_2^2 + \\
&b_2(-c_2 + d_1(c_1 - c_0 d_1) + c_0 d_2) + b_1(2b_2(c_1 - c_0 d_1) + d_1(c_2 - c_1 d_1 + c_0 d_1^2) + \\
&(c_1 - 2c_0 d_1)d_2))))/ \\
&((a_2 c_0 + a_1(c_1 - c_0(b_1 + d_1)) + a_0(b_1^2 c_0 + c_2 - c_1 d_1 + b_1(-c_1 + c_0 d_1) - \\
&c_0(b_2 - d_1^2 + d_2)))^2 - (a_1 c_0 + a_0(c_1 - c_0(b_1 + d_1)))(a_2(c_1 - c_0(b_1 + d_1)) - \\
&a_0(b_1^3 c_0 + b_2(c_1 - c_0 d_1) + b_1^2(-c_1 + c_0 d_1) + d_1(c_2 - c_1 d_1 + c_0 d_1^2) + \\
&b_1(c_2 - c_1 d_1 + c_0(-2b_2 + d_1^2 - d_2)) + (c_1 - 2c_0 d_1)d_2) + \\
&a_1(b_1^2 c_0 + c_2 - c_1 d_1 + b_1(-c_1 + c_0 d_1) - c_0(b_2 - d_1^2 + d_2))))
\end{aligned}
$$

$$
\begin{aligned}
v_4 = &a_2(b_1^2 - b_2 - b_1 z_1 + z_2) + a_0(b_1^4 - b_1^3 z_1 + 2b_1 b_2 z_1 + b_2(b_2 - z_2) + \\
&b_1^2(-3b_2 + z_2)) - a_1(b_1^3 - b_1^2 z_1 + b_2 z_1 + b_1(-2b_2 + z_2))
\end{aligned}
$$

$$
v_3 = a_2(-b_1 + z_1) + a_1(b_1^2 - b_2 - b_1 z_1 + z_2) - a_0(b_1^3 - 2b_1 b_2 - b_1^2 z_1 + b_2 z_1 + b_1 z_2)
$$

$$
\begin{aligned}
v_2 &= a_2 + a_1(-b_1 + z_1) + a_0(b_1^2 - b_2 - b_1 z_1 + z_2) \\
v_1 &= a_1 + a_0(-b_1 + z_1) \\
v_0 &= a_0
\end{aligned}
$$

$$
w_4 = d_1^2(c_2 + d_1(-c_1 + c_0 d_1)) - (c_2 + d_1(-2c_1 + 3c_0 d_1))d_2 + c_0 d_2^2
$$

$$
w_3 = - c_2 d_1 + c_1 d_1^2 - c_0 d_1^3 - c_1 d_2 + 2c_0 d_1 d_2
$$

$$
\begin{aligned}
w_2 &= c_2 - c_1 d_1 + c_0(d_1^2 - d_2) \\
w_1 &= c_1 - c_0 d_1 \\
w_0 &= c_0.
\end{aligned}
$$

Für Lösungen des Gleichungssystems (17)–(20) ergibt sich

$$\begin{aligned}
q_3(x)p_1(x) - q_1(x)r_1(x) &= (\mathbf{a_0} - a_0)(1 + z_1x + z_2x^2) - (b_2v_3 + b_1v_4)x^5 - (b_2v_4)x^6 \\
p_2(x) - q_2(x)r_2(x) &= (\mathbf{c_0} - c_0) - (d_2w_3 + d_1w_4)x^5 - (d_2w_4)x^6 \\
r_1(x)r_2(x) &= (v_0 + w_0) + (v_0w_1 + v_1w_0)x + \\
&\quad + (v_0w_2 + v_1w_1 + v_2w_0)x^2 + \\
&\quad + (v_1w_4 + v_2w_3 + v_3w_2 + v_4w_1)x^5 + \\
&\quad + (v_2w_4 + v_3w_3 + v_4w_2)x^6 + \\
&\quad + (v_3w_4 + v_4w_3)x^7 + \\
&\quad + (v_4w_4)x^8.
\end{aligned}$$

Somit ergibt sich folgendes Theorem.

Theorem 5.8. *Für Lösungen des Gleichungssystems (17)–(20) und alle $x \in \mathbf{x}$ gilt*

$$\frac{p_1(x)}{q_1(x)} \cdot \frac{p_2(x)}{q_2(x)} \subseteq \frac{(y_0 + I_1(x) + I_2(x) + I_5(x) + I_6(x) + I_7(x) + I_8(x)) + y_1x + y_2x^2}{1 + z_1x + z_2x^2},$$

wobei

$$\begin{aligned}
y_0 &= v_0w_0 \\
y_1 &= v_0w_1 + v_1w_0 \\
y_2 &= v_0w_2 + v_1w_1 + v_2w_0 \\
I_1(x) &= \frac{p_2(x)}{q_1(x)q_2(x)} \left((\boldsymbol{a}_0 - a_0)(1 + z_1x + z_2x^2) - (b_2v_3 + b_1v_4)x^5 - (b_2v_4)x^6\right) \\
I_2(x) &= \frac{r_1(x)}{q_2(x)} \left((\boldsymbol{c}_0 - c_0) - (d_2w_3 + d_1w_4)x^5 - (d_2w_4)x^6\right) \\
I_5(x) &= (v_1w_4 + v_2w_3 + v_3w_2 + v_4w_1)x^5 \\
I_6(x) &= (v_2w_4 + v_3w_3 + v_4w_2)x^6 \\
I_7(x) &= (v_3w_4 + v_4w_3)x^7 \\
I_8(x) &= (v_4w_4)x^8.
\end{aligned}$$

Korollar 5.9. *Unter der Voraussetzung, dass das Gleichungssystem (17)–(20) lösbar ist, können $p_3(x)$ und $q_3(x)$ folgendermaßen gewählt werden*

$$\begin{aligned}
p_3(x) &= (y_0 + I_1(\mathbf{x}) + I_2(\mathbf{x}) + I_5(\mathbf{x}) + I_6(\mathbf{x}) + I_7(\mathbf{x}) + I_8(\mathbf{x})) + y_1x + y_2x^2 \\
q_3(x) &= 1 + z_1x + z_2x^2.
\end{aligned}$$

Bemerkung 5.10. *Auch bei der Multiplikation können bei der Berechnung wieder die gleichen Probleme auftreten wie schon bei der Addition, nämlich dass einer der Nenner von z_1 oder z_2 gleich Null ist. Man löst das analog zur Addition, wie in Bemerkung 5.5. beschrieben, und erhält*

$$\begin{aligned}\left(\frac{p_1(x)}{q_1(x)} \cdot \frac{p_2(x)}{q_2(x)}\right) q_3(x) \subseteq\ & y_0 + y_1 x + y_2 x^2 + \\ & + (v_0 w_3 + v_1 w_2 + v_2 w_1 + v_3 w_0)\boldsymbol{x}^3 + \\ & + (v_0 w_4 + v_1 w_3 + v_2 w_2 + v_3 w_1 + v_4 w_0)\boldsymbol{x}^4 + \\ & + I_1(\boldsymbol{x}) + I_2(\boldsymbol{x}) + I_5(\boldsymbol{x}) + I_6(\boldsymbol{x}) + I_7(\boldsymbol{x}) + I_8(\boldsymbol{x}) \quad \forall x \in \mathbf{x}.\end{aligned}$$

Nun werden, genauso wie bei der Addition, x^3 und x^4 wieder quadratisch approximiert um den Fehler zu verringern. Außerdem kann man nun auch x^5, x^6, x^7 und x^8 quadratisch approximieren.
Seien

$$\begin{aligned}\bar{I}_3 &= v_0 w_3 + v_1 w_2 + v_2 w_1 + v_3 w_0 \\ \bar{I}_4 &= v_0 w_4 + v_1 w_3 + v_2 w_2 + v_3 w_1 + v_4 w_0 \\ \bar{I}_5 &= v_1 w_4 + v_2 w_3 + v_3 w_2 + v_4 w_1 \\ \bar{I}_6 &= v_2 w_4 + v_3 w_3 + v_4 w_2 \\ \bar{I}_7 &= v_3 w_4 + v_4 w_3 \\ \bar{I}_8 &= v_4 w_4.\end{aligned}$$

Es gilt

$$\begin{aligned}\left(\frac{p_1(x)}{q_1(x)} \cdot \frac{p_2(x)}{q_2(x)}\right) q_3(x) \subseteq\ & (v_0 w_0 + \bar{I}_4 j_0 + \bar{I}_6 n_0 + \bar{I}_8 s_0) + \\ & + (v_0 w_1 + v_1 w_0 + \bar{I}_3 k_1 + \bar{I}_5 m_1 + \bar{I}_7 r_1) x + \\ & + (v_0 w_2 + v_1 w_1 + v_2 w_0 + \bar{I}_4 j_2 + \bar{I}_6 n_2 + \bar{I}_8 s_2) x^2 + \\ & + \bar{I}_3 (x^3 - k_1 x) + \bar{I}_4 (x^4 - (j_0 + j_2 x^2)) + \\ & + \bar{I}_5 (x^5 - m_1 x) + \bar{I}_6 (x^6 - (n_0 + n_2 x^2)) + \\ & + \bar{I}_7 (x^7 - r_1 x) + \bar{I}_8 (x^8 - (s_0 + s_2 x^2)) + I_1(\mathbf{x}) + I_2(\mathbf{x}) \quad \forall x \in \mathbf{x},\end{aligned}$$

wobei man k_1, j_0, j_2, m_1, n_0, n_2, r_1, s_0, $s_2 \in \mathbb{R}$ so wählt, dass folgende Ungleichungen erfüllt sind

$$\begin{aligned} x^3 - (k_1 x) &\leq x^3 && \forall x \in \mathbf{x} \\ x^4 - (j_0 + j_2 x^2) &\leq x^4 && \forall x \in \mathbf{x} \\ x^5 - (m_1 x) &\leq x^5 && \forall x \in \mathbf{x} \\ x^6 - (n_0 + n_2 x^2) &\leq x^6 && \forall x \in \mathbf{x} \\ x^7 - (r_1 x) &\leq x^7 && \forall x \in \mathbf{x} \\ x^8 - (s_0 + s_2 x^2) &\leq x^8 && \forall x \in \mathbf{x}. \end{aligned}$$

Für k_1, j_0 und j_2 wählt man die schon bei der Addtition berechneten Werte. Die restlichen Koeffizienten bestimmt man auf folgende Art und Weise.

Approximation von x^5

Sei $\mathbf{x} = [-h, h]$, $0 < h \in [\underline{h}, \overline{h}[$. Nun wird m_1 so gewählt, dass folgende heuristische Gleichung erfüllt ist

$$\frac{\text{Maximaler Fehler ohne Korrektur im Intervall } [0, \underline{h}]}{\text{Maximaler Fehler mit Korrektur im Intervall } [0, \underline{h}]} = \frac{\text{Fehler ohne Korrektur bei } \bar{h}}{\text{Fehler mit Korrektur bei } \bar{h}}.$$

Da der maximale Abstand von x^5 und $m_1 \cdot x$ im Intervall $[0, \underline{h}]$ bei $\left(\frac{m_1}{5}\right)^{\frac{1}{5-1}}$ auftritt löst man für $h \in [\underline{h}, \bar{h}[$ folgende Gleichung nach m_1 auf

$$\frac{\underline{h}^5}{-\left(\frac{m_1}{5}\right)^{\frac{5}{5-1}} + m_1 \left(\frac{m_1}{5}\right)^{\frac{1}{5-1}}} = \frac{\bar{h}^5}{\bar{h}^5 - m_1 \bar{h}}. \tag{21}$$

Bemerkung 5.11. *Analog berechnet man eine Approximation von x^7, also den Koeffizienten r_1.*

Approximation von x^6

Sei $\mathbf{x} = [-h, h]$, $0 < h \in [\underline{h}, \overline{h}[$. Es werden die Koeffizienten n_0, n_2 so gewählt, dass folgende heuristische Gleichungen erfüllt sind

$$\begin{aligned}&\text{Fehler mit Korrektur bei } 0 \ = \\ &\text{Maximaler Fehler mit Korrektur zwischen dem kleineren positiven Schnittpunkt und } \underline{h}\end{aligned}$$

$$\frac{\text{Maximaler Fehler ohne Korrektur im Intervall } [0, \underline{h}]}{\text{Maximaler Fehler mit Korrektur im Intervall } [0, \underline{h}]} = \frac{\text{Fehler ohne Korrektur bei } \bar{h}}{\text{Fehler mit Korrektur bei } \bar{h}}.$$

Der maximale Abstand von x^6 und $n_0 + n_2 \cdot x^2$ zwischen dem kleineren positiven Schnittpunkt und $\underline{h}$ liegt bei $\min\left(\left(\frac{2n_2}{6}\right)^{\frac{1}{6-2}}, \underline{h}\right)$. Somit löst man für $h \in [\underline{h}, \bar{h}[$ folgende Gleichungen nach n_0, n_2 auf

$$\begin{aligned} n_0 &= \left(\min\left(\left(\frac{2n_2}{6}\right)^{\frac{1}{6-2}}, \underline{h}\right)\right)^6 - \left(n_0 + n_2 \left(\min\left(\left(\frac{2n_2}{6}\right)^{\frac{1}{6-2}}, \underline{h}\right)\right)^2\right) \\ \frac{\underline{h}^6}{-n_0} &= \frac{\bar{h}^6}{\bar{h}^6 - (n_0 + n_2 \bar{h}^2)}. \end{aligned} \tag{22}$$

Bemerkung 5.12. *Die Koeffizienten s_0 und s_2 der Approximation von x^8 können wieder analog berechnet werden.*

Zwischen korrigiertem Fehler und Fehler ohne Korrektur gilt folgender Zusammenhang

$$\begin{aligned}
&\text{Maximaler korrigierter Fehler für } x^3 = \text{ Maximaler Fehler ohne Korrektur für } x^3 \cdot \gamma_3\\
&\text{Maximaler korrigierter Fehler für } x^4 = \text{ Maximaler Fehler ohne Korrektur für } x^4 \cdot \gamma_4\\
&\text{Maximaler korrigierter Fehler für } x^5 = \text{ Maximaler Fehler ohne Korrektur für } x^5 \cdot \gamma_5\\
&\text{Maximaler korrigierter Fehler für } x^6 = \text{ Maximaler Fehler ohne Korrektur für } x^6 \cdot \gamma_6\\
&\text{Maximaler korrigierter Fehler für } x^7 = \text{ Maximaler Fehler ohne Korrektur für } x^7 \cdot \gamma_7\\
&\text{Maximaler korrigierter Fehler für } x^8 = \text{ Maximaler Fehler ohne Korrektur für } x^8, \cdot \gamma_8
\end{aligned}$$

wobei

$$\begin{aligned}
0 < \gamma_3 &= \frac{\bar{h}^3 - (k_1\bar{h})}{\bar{h}^3} \leq 1\\
0 < \gamma_4 &= \frac{\bar{h}^4 - (j_0 + j_2\bar{h}^2)}{\bar{h}^4} \leq 1\\
0 < \gamma_5 &= \frac{\bar{h}^5 - (m_1\bar{h})}{\bar{h}^5} \leq 1\\
0 < \gamma_6 &= \frac{\bar{h}^6 - (n_0 + n_2\bar{h}^2)}{\bar{h}^6} \leq 1\\
0 < \gamma_7 &= \frac{\bar{h}^7 - (r_1\bar{h})}{\bar{h}^7} \leq 1\\
0 < \gamma_8 &= \frac{\bar{h}^8 - (s_0 + s_2\bar{h}^2)}{\bar{h}^8} \leq 1.
\end{aligned} \tag{23}$$

Liegt nun $h \in [\underline{h}, \bar{h}]$ so ergibt sich

$$\begin{aligned}
\left(\frac{p_1(x)}{q_1(x)} \cdot \frac{p_2(x)}{q_2(x)}\right) q_3(x) \subseteq\ & (v_0w_0 + \bar{I}_4j_0 + \bar{I}_6n_0 + \bar{I}_8s_0) +\\
&+ (v_0w_1 + v_1w_0 + \bar{I}_3k_1 + \bar{I}_5m_1 + \bar{I}_7r_1)x +\\
&+ (v_0w_2 + v_1w_1 + v_2w_0 + \bar{I}_4j_2 + \bar{I}_6n_2 + \bar{I}_8s_2)x^2 +\\
&+ \bar{I}_3\gamma_3x^3 + \bar{I}_4\gamma_4x^4 + \bar{I}_5\gamma_5x^5 + \bar{I}_6\gamma_6x^6 +\\
&+ \bar{I}_7\gamma_7x^7 + \bar{I}_8\gamma_8x^8 + I_1(\mathbf{x}) + I_2(\mathbf{x}) \quad \forall x \in \mathbf{x}.
\end{aligned}$$

Löst man die Gleichungssysteme (21)–(23), für h aus den Intervallen

$$[0.1, 0.3[, [0.3, 0.5[, \cdots, [1.7, 1.9[, [1.9, 20[,$$

so erhält man für die Koeffizienten der Approximationen und für die in (23) definierten Faktoren, die Werte aus Tabelle 3 und 4.
Für $0 < h < 0.1$ werden wieder alle Koeffizienten Null bzw. Faktoren Eins gesetzt.

	[0.1,0.3[	**[0.3,0.5[**	**[0.5,0.7[**	**[0.7,0.9[**	**[0.9,1.1[**
k_1	0.0165104	0.11358	0.273816	0.494207	0.774527
j_0	-0.0000734694	-0.00327802	-0.0183771	-0.0585954	-0.141925
j_2	0.0246939	0.16193	0.383428	0.684663	1.06555
m_1	0.000162289	0.0113763	0.0760206	0.262915	0.666007
n_0	-0.000000965152	-0.00052562	-0.00858149	-0.0532767	-0.20985
n_2	0.000292989	0.019539	0.125747	0.424762	1.0584
r_1	0.00000150433	0.00103587	0.0200604	0.137244	0.572961
s_0	-0.00000000996115	-0.0000583196	-0.00288259	-0.0360082	-0.235398
s_2	0.00000294255	0.00196948	0.0367137	0.243946	0.99734
γ_3	0.81656	0.470589	0.441193	0.38987	0.359896
γ_4	0.734695	0.40469	0.294034	0.244047	0.216318
γ_5	0.979965	0.81798	0.68338	0.599277	0.545109
γ_6	0.96515	0.721017	0.549215	0.452846	0.39488
γ_7	0.998	0.9338	0.82949	0.74176	0.67658
γ_8	0.996116	0.888884	0.737943	0.624623	0.564843

Tabelle 3

	[1.1,1.3[	**[1.3,1.5[**	**[1.5,1.7[**	**[1.7,1.9[**	**[1.9,20[**
k_1	1.11478	1.51497	1.97512	2.49525	6.74555
j_0	-0.291161	-0.5339	-0.902543	-1.43429	-12.6865
j_2	1.5262	2.06669	2.68707	3.38737	10.6385
m_1	1.40492	2.6251	4.49784	7.2203	0
n_0	-0.63253	-1.59653	-3.5477	-7.16397	0
n_2	2.21061	4.09769	6.9785	11.1487	0
r_1	1.79307	4.64341	10.5136	21.5302	0
s_0	-1.05589	-3.69821	-10.8669	-27.9974	0
s_2	3.07401	7.8702	17.6632	35.9194	0
γ_3	0.340368	0.32669	0.316568	0.308796	0.98314
γ_4	0.198868	0.186934	0.17829	0.17173	0.97349
γ_5	0.508099	0.481463	0.461473	0.44597	1
γ_6	0.357052	0.330727	0.226312	0.296797	1
γ_7	0.62852	0.59235	0.56444	0.54236	1
γ_8	0.49258	0.453363	0.42401	0.401354	1

Tabelle 4

5.3 Padé-Division

Analog zu den beiden vorigen Operationen ist es Ziel dieses Abschnittes ein Padé-Modell für den Quotienten zweier Padé-Modelle zu berechnen. Es wird also folgendes Problem behandelt.

Problemstellung

Für ein Intervall $\mathbf{x} = [\underline{x}, \overline{x}]$ und die gegebenen Polynome

$$\begin{aligned} p_1(x) &= [\underline{a_0}, \overline{a_0}] + a_1 x + a_2 x^2 \\ q_1(x) &= 1 + b_1 x + b_2 x^2 \\ p_2(x) &= [\underline{c_0}, \overline{c_0}] + c_1 x + c_2 x^2 \\ q_2(x) &= 1 + d_1 x + d_2 x^2 \end{aligned}$$

sollen Polynome

$$\begin{aligned} p_3(x) &= [\underline{y_0}, \overline{y_0}] + y_1 x + y_2 x^2 \\ q_3(x) &= 1 + z_1 x + z_2 x^2 \end{aligned}$$

berechnet werden, sodass diese folgende Gleichung erfüllen

$$\frac{\frac{p_1(x)}{q_1(x)}}{\frac{p_2(x)}{q_2(x)}} = \frac{p_1(x)}{q_1(x)} \cdot \frac{q_2(x)}{p_2(x)} \subseteq \frac{p_3(x)}{q_3(x)} \quad \forall x \in \mathbf{x}.$$

Diese Problemstellung ergibt natürlich nur dann Sinn, wenn $0 \notin \mathbf{c_0}$. Man kann dies ab jetzt voraussetzen. Da $c_0 = \mathrm{mid}(\mathbf{c_0})$, folgt natürlich dass $c_0 \neq 0$. Ziel ist es, dieses Problem auf die Padé-Multiplikation zurückzuführen. Dafür muss zunächst der konstante Teil des Nenners auf Eins normiert werden.

Es gilt

$$\frac{q_2(x)}{p_2(x)} = \frac{1 + d_1 x + d_2 x^2}{\mathbf{c_0} + c_1 x + c_2 x^2} = \frac{\underbrace{\frac{1}{c_0}}_{=:e_0} + \underbrace{\frac{d_1}{c_0}}_{=:e_1} x + \underbrace{\frac{d_2}{c_0}}_{=:e_2} x^2}{\underbrace{\frac{\mathbf{c_0}}{c_0}}_{=:f_0} + \underbrace{\frac{c_1}{c_0}}_{=:f_1} x + \underbrace{\frac{c_2}{c_0}}_{=:f_2} x^2}.$$

Seien

$$\begin{aligned} \mathbf{I} &= \mathbf{c_0} - c_0 \\ \overline{p_2}(x) &= e_0 + e_1 x + e_2 x^2 \\ \overline{q_2}(x) &= f_0 + f_1 x + f_2 x^2 = 1 + \frac{\mathbf{I}}{c_0} + f_1 x + f_2 x^2. \end{aligned}$$

Als nächstes werden nun wie bei der Multiplikation zwei Hilfspolynome $r_1(x)$, $r_2(x)$ definiert. Werden in die Gleichungen (18) anstatt der Koeffizienten c_0 bis c_2, d_1, d_2 die soeben berechneten Koeffizienten e_0 bis e_2 bzw. f_1, f_2 eingesetzt und die entstehenden Gleichungen mit (18') bezeichnet, so resultiert für dieses Problem ein Gleichungssystem mit den Gleichungen (17), (18'), (19), (20), welches analog zur Multiplikation gelöst werden kann.

Verwendet man nun die zuvor definierten Polynome $\overline{p_2}(x)$, $\overline{q_2}(x)$, $r_1(x)$, $r_2(x)$ so kann man $\left(\frac{p_1(x)}{q_1(x)} \cdot \frac{q_2(x)}{p_2(x)}\right) q_3(x)$ folgendermaßen schreiben

$$\left(\frac{p_1(x)}{q_1(x)} \cdot \frac{q_2(x)}{p_2(x)}\right) q_3(x) = r_1(x)r_2(x) + \frac{r_1(x)}{\overline{q_2}(x)}(\overline{p_2}(x) - \overline{q_2}(x)r_2(x)) + \\ + \frac{\overline{p_2}(x)}{q_1(x)\overline{q_2}(x)}(q_3(x)p_1(x) - q_1(x)r_1(x)).$$

Nachdem man das entstandene Gleichungssystem gelöst hat, ergibt sich mit den berechneten Koeffizienten folgender Zusammenhang

$$q_3(x)p_1(x) - q_1(x)r_1(x) = (\mathbf{a_0} - a_0)(1 + z_1x + z_2x^2) - (b_2v_3 + b_1v_4)x^5 - (b_2v_4)x^6$$

$$\overline{p_2}(x) - \overline{q_2}(x)r_2(x) = -\frac{\mathbf{I}}{c_0}r_2(x) - (f_2w_3 + f_1w_4)x^5 - (f_2w_4)x^6 = \\ = -\frac{\mathbf{c_0} - c_0}{c_0}r_2(x) - (f_2w_3 + f_1w_4)x^5 - (f_2w_4)x^6.$$

Als Produkt der Hilfspolynome $r_1(x)$ und $r_2(x)$ erhält man

$$\begin{aligned} r_1(x)r_2(x) = {} & (v_0 + w_0) + (v_0w_1 + v_1w_0)x + \\ & + (v_0w_2 + v_1w_1 + v_2w_0)x^2 + \\ & + (v_0w_3 + v_1w_2 + v_2w_1 + v_3w_0)x^3 + \\ & + (v_0w_4 + v_1w_3 + v_2w_2 + v_3w_1 + v_4w_0)x^4 + \\ & + (v_1w_4 + v_2w_3 + v_3w_2 + v_4w_1)x^5 + \\ & + (v_2w_4 + v_3w_3 + v_4w_2)x^6 + \\ & + (v_3w_4 + v_4w_3)x^7 + \\ & + (v_4w_4)x^8. \end{aligned}$$

Es gilt also das folgende Theorem.

Theorem 5.13. *Für Lösungen des Gleichungssystems (17), (18'), (19), (20) gilt*

$$\frac{p_1(x)}{q_1(x)} \cdot \frac{q_2(x)}{p_2(x)} = \frac{p_1(x)}{q_1(x)} \cdot \frac{\bar{p_2}(x)}{\bar{q_2}(x)} \subseteq \\ \subseteq \frac{(y_0 + I_1(x) + I_2(x) + I_5(x) + I_6(x) + I_7(x) + I_8(x)) + y_1x + y_2x^2}{1 + z_1x + z_2x^2} \quad \forall x \in \mathbf{x},$$

wobei

$$\begin{aligned} y_0 &= v_0w_0 \\ y_1 &= v_0w_1 + v_1w_0 \\ y_2 &= v_0w_2 + v_1w_1 + v_2w_0 \end{aligned}$$

$$
\begin{aligned}
I_1(x) &= \frac{\overline{p_2}(x)}{q_1(x)\overline{q_2}(x)}\left((\boldsymbol{a}_0 - a_0)(1 + z_1x + z_2x^2) - (b_2v_3 + b_1v_4)x^5 - (b_2v_4)x^6\right)\\
I_2(x) &= \frac{r_1(x)}{\overline{q_2}(x)}\left((\frac{\boldsymbol{c}_0 - c_0}{c_0}r_2(x)) - (d_2w_3 + d_1w_4)x^5 - (d_2w_4)x^6\right)\\
I_5(x) &= (v_1w_4 + v_2w_3 + v_3w_2 + v_4w_1)x^5\\
I_6(x) &= (v_2w_4 + v_3w_3 + v_4w_2)x^6\\
I_7(x) &= (v_3w_4 + v_4w_3)x^7\\
I_8(x) &= (v_4w_4)x^8.
\end{aligned}
$$

Korollar 5.14. *Ist das Gleichungssystem (17), (18'), (19), (20) lösbar, so können* $p_3(x)$ *und* $q_3(x)$ *gewählt werden als*

$$
\begin{aligned}
p_3(x) &= (y_0 + I_1(\mathbf{x}) + I_2(\mathbf{x}) + I_5(\mathbf{x}) + I_6(\mathbf{x}) + I_7(\mathbf{x}) + I_8(\mathbf{x})) + y_1x + y_2x^2\\
q_3(x) &= 1 + z_1x + z_2x^2.
\end{aligned}
$$

Auch bei der Division hat man bei der Berechnung wieder die gleichen Probleme wie schon bei Addition und Multiplikation. Da wir aber die Division auf die Multiplikation zurückgeführt haben, erhält man auch die gleiche Lösung.

Für $h \in [\underline{h}, \bar{h}]$ gilt

$$
\begin{aligned}
\left(\frac{p_1(x)}{q_1(x)} \cdot \frac{q_2(x)}{p_2(x)}\right) q_3(x) \subseteq\ & (v_0w_0 + \bar{I}_4j_0 + \bar{I}_6n_0 + \bar{I}_8s_0) +\\
& + (v_0w_1 + v_1w_0 + \bar{I}_3k_1 + \bar{I}_5m_1 + \bar{I}_7r_1)x +\\
& + (v_0w_2 + v_1w_1 + v_2w_0 + \bar{I}_4j_2 + \bar{I}_6n_2 + \bar{I}_8s_2)x^2 +\\
& + \bar{I}_3\gamma_3x^3 + \bar{I}_4\gamma_4x^4 +\\
& + \bar{I}_5\gamma_5x^5 + \bar{I}_6\gamma_6x^6 +\\
& + \bar{I}_7\gamma_7x^7 + \bar{I}_8\gamma_8x^8 + I_1(x) + I_2(x) \quad \forall x \in \mathbf{x},
\end{aligned}
$$

wobei die Konstanten k_1, j_0, j_2, m_1, n_0, n_2, r_1, s_0, s_2, γ_3, γ_4, γ_5, γ_6, γ_7, γ_8 aus den Tabellen 3 und 4 zu übernehmen sind.

6 Zweidimensionale Padé-Modelle

In diesem Abschnitt wird nun, analog zum Padé-Modell in einer Dimension, für Funktionen in zwei Variablen ein entsprechendes Padé-Modell berechnet. So wie zuvor wird auch in zwei Dimensionen versucht, die gegebene Funktion aufzuspalten und nur die Elementaroperationen zu betrachten, aus denen Funktionen zusammengesetzt werden können. Ebenfalls wie schon im letzten Kapitel, werden hier nur Addition, Multiplikation und Division behandelt.

6.1 Padé-Addition

Als erste Elementaroperation wird die Addition betrachtet. Es folgt also die Berechnung eines Padé-Modells für die Summe von zwei zweidimensionalen Padé-Modellen.

Problemstellung

Gegeben seien eine zweidimensionale Box $\mathbf{x}$ und die Polynome

$$p_1(x) = [\underline{a}, \overline{a}] + \sum_{|(i,j)|=1}^{2} a_{ij} x^i y^j$$

$$q_1(x) = 1 + \sum_{|(i,j)|=1}^{2} b_{ij} x^i y^j$$

$$p_2(x) = [\underline{c}, \overline{c}] + \sum_{|(i,j)|=1}^{2} c_{ij} x^i y^j$$

$$q_2(x) = 1 + \sum_{|(i,j)|=1}^{2} d_{ij} x^i y^j.$$

Gesucht sind nun Polynome $p_3(x)$, $q_3(x)$ der Form

$$p_3(x) = [\underline{y}, \overline{y}] + \sum_{|(i,j)|=1}^{2} y_{ij} x^i y^j$$

$$q_3(x) = 1 + \sum_{|(i,j)|=1}^{2} z_{ij} x^i y^j,$$

sodass folgende Gleichung erfüllt ist

$$\frac{p_1(x,y)}{q_1(x,y)} + \frac{p_2(x,y)}{q_2(x,y)} \subseteq \frac{p_3(x,y)}{q_3(x,y)} \quad \forall (x,y) \in \mathbf{x}.$$

Als erstes werden erneut zwei Hilfspolynome eingeführt.
Seien

$$r_1(x,y) = \sum_{|(i,j)|=0}^{3} v_{ij} x^i y^j$$
$$r_2(x,y) = \sum_{|(i,j)|=0}^{3} w_{ij} x^i y^j,$$

sodass

$$q_3(x,y)\tilde{p}_1(x,y) \approx q_1(x,y) r_1(x,y) \tag{24}$$
$$q_3(x,y)\tilde{p}_2(x,y) \approx q_2(x,y) r_2(x,y). \tag{25}$$

Bemerkung 6.1. *Auch hier bedeute $\approx$ eine Übereinstimmung bis in möglichst hohe Ordnung. Seien* $\tilde{p}_1 = \mathrm{mid}(\mathbf{a_{00}}) + \sum_{|(i,j)|=1}^{2} a_{ij} x^i y^j$, $\tilde{p}_2 = \mathrm{mid}(\mathbf{c_{00}}) + \sum_{|(i,j)|=1}^{2} c_{ij} x^i y^j$, $\tilde{p}_3 = \mathrm{mid}(\mathbf{y_{00}}) + \sum_{|(i,j)|=1}^{2} y_{ij} x^i y^j$.

Werden die entsprechenden Polynome in (24), (25) eingesetzt, so ergibt das

$$\left(1 + \sum_{|(i,j)|=1}^{2} z_{ij} x^i y^j\right) \cdot \left(\sum_{|(i,j)|=0}^{2} a_{ij} x^i y^j\right) \approx \left(1 + \sum_{|(i,j)|=1}^{2} b_{ij} x^i y^j\right) \cdot \left(\sum_{|(i,j)|=0}^{3} v_{ij} x^i y^j\right)$$
$$\left(1 + \sum_{|(i,j)|=1}^{2} z_{ij} x^i y^j\right) \cdot \left(\sum_{|(i,j)|=0}^{2} c_{ij} x^i y^j\right) \approx \left(1 + \sum_{|(i,j)|=1}^{2} d_{ij} x^i y^j\right) \cdot \left(\sum_{|(i,j)|=0}^{3} w_{ij} x^i y^j\right).$$

Ein Koeffizientenvergleich der Gleichungen (24) und (25) ergibt folgende Gleichungen

$$\begin{aligned}
a_{00} &= v_{00} \\
a_{10} + a_{00} z_{10} &= v_{10} + v_{00} b_{10} \\
a_{01} + a_{00} z_{01} &= v_{01} + v_{00} b_{01} \\
a_{11} + a_{00} z_{11} + a_{10} z_{01} + a_{01} z_{10} &= v_{11} + v_{00} b_{11} + v_{10} b_{01} + v_{01} b_{10} \\
a_{20} + a_{00} z_{20} + a_{10} z_{10} &= v_{20} + v_{00} b_{20} + v_{10} b_{10} \\
a_{02} + a_{00} z_{02} + a_{01} z_{01} &= v_{02} + v_{00} b_{02} + v_{01} b_{01} \\
a_{10} z_{20} + a_{20} z_{10} &= v_{30} + v_{10} b_{20} + v_{20} b_{10} \\
a_{01} z_{02} + a_{02} z_{01} &= v_{03} + v_{01} b_{02} + v_{02} b_{01} \\
a_{10} z_{11} + a_{01} z_{20} + a_{11} z_{10} + a_{20} z_{01} &= v_{21} + v_{10} b_{11} + v_{01} b_{20} + v_{11} b_{10} + v_{20} b_{01} \\
a_{01} z_{11} + a_{10} z_{02} + a_{11} z_{01} + a_{02} z_{10} &= v_{12} + v_{01} b_{11} + v_{10} b_{02} + v_{11} b_{01} + v_{02} b_{10}
\end{aligned} \tag{26}$$

$$
\begin{aligned}
c_{00} &= w_{00} \\
c_{10} + c_{00}z_{10} &= w_{10} + w_{00}d_{10} \\
c_{01} + c_{00}z_{01} &= w_{01} + w_{00}d_{01} \\
c_{11} + c_{00}z_{11} + c_{10}z_{01} + c_{01}z_{10} &= w_{11} + w_{00}d_{11} + w_{10}d_{01} + w_{01}d_{10} \\
c_{20} + c_{00}z_{20} + c_{10}z_{10} &= w_{20} + w_{00}d_{20} + w_{10}d_{10} \\
c_{02} + c_{00}z_{02} + c_{01}z_{01} &= w_{02} + w_{00}d_{02} + w_{01}d_{01} \\
c_{10}z_{20} + c_{20}z_{10} &= w_{30} + w_{10}d_{20} + w_{20}d_{10} \\
c_{01}z_{02} + c_{02}z_{01} &= w_{03} + w_{01}d_{02} + w_{02}d_{01} \\
c_{10}z_{11} + c_{01}z_{20} + c_{11}z_{10} + c_{20}z_{01} &= w_{21} + w_{10}d_{11} + w_{01}d_{20} + w_{11}d_{10} + w_{20}d_{01} \\
c_{01}z_{11} + c_{10}z_{02} + c_{11}z_{01} + c_{02}z_{10} &= w_{12} + w_{01}d_{11} + w_{10}d_{02} + w_{11}d_{01} + w_{02}d_{10} \\
a_{20}z_{20} &= v_{30}b_{10} + v_{20}b_{20}.
\end{aligned}
\tag{27}
$$

Um die Koeffizienten der Polynome $r_1(x,y)$, $r_2(x,y)$ und $q_3(x,y)$ zu bestimmen, müssen 25 Koeffizienten berechnet werden. Dazu verwendet man folgende 25 Gleichungen. Die Gleichungen (26) aus dem Koeffizientenvergleich und folgende vier Gleichungen

$$
\begin{aligned}
v_{30} + w_{30} &= 0 \\
v_{21} + w_{21} &= 0 \\
v_{12} + w_{12} &= 0 \\
v_{03} + w_{03} &= 0.
\end{aligned}
\tag{28}
$$

Bemerkung 6.2. *Dass gerade die Gleichungen (28) verwendet werden, hat den gleichen Grund wie im eindimensionalen Fall, beschrieben in Bemerkung 5.1..*

Man kann $\left(\frac{p_1(x,y)}{q_1(x,y)} + \frac{p_2(x,y)}{q_2(x,y)}\right) q_3(x,y)$ wieder umschreiben zu

$$
\begin{aligned}
\left(\frac{p_1(x,y)}{q_1(x,y)} + \frac{p_2(x,y)}{q_2(x,y)}\right) q_3(x,y) = r_1(x,y) + r_2(x,y) + \\
+ \frac{q_3(x,y)p_1(x,y) - q_1(x,y)r_1(x,y)}{q_1(x,y)} + \\
+ \frac{q_3(x,y)p_2(x,y) - q_2(x)r_2(x,y)}{q_2(x,y)}.
\end{aligned}
$$

Nach dem Lösen des Gleichungssystems kann man mit den berechneten Koeffizienten die einzelnen Terme dieser Gleichung bestimmen.

$$q_3(x,y)p_1(x,y) - q_1(x,y)r_1(x,y) = (\mathbf{a_{00}} - a_{00})\left(1 + \sum_{|(i,j)|=1}^{2} z_{ij}x^i y^j\right) + \\ + \sum_{|(i,j)|=4}^{5} (q_3p_1 - q_1r_1)_{ij}x^i y^j,$$

wobei

$$\begin{aligned}\sum_{|(i,j)|=4}^{5} (q_3p_1 - q_1r_1)_{ij}x^i y^j =& (a_{02}z_{02} - b_{01}v_{03} - b_{02}v_{02})y^4 + \\ &+(a_{20}z_{11} + a_{11}z_{20} - b_{01}v_{30} - b_{10}v_{21} - b_{20}v_{11} - b_{11}v_{20})x^3y + \\ &+(a_{02}z_{11} + a_{11}z_{02} - b_{10}v_{03} - b_{01}v_{12} - b_{02}v_{11} - b_{11}v_{02})xy^3 + \\ &+(a_{20}z_{02} + a_{02}z_{20} - b_{20}v_{02} - b_{02}v_{20} - b_{10}v_{12} - b_{01}v_{21})x^2y^2 - \\ &-(b_{20}v_{30})x^5 - (b_{02}v_{03})y^5 - (b_{11}v_{30} + b_{20}v_{21})x^4y - \\ &-(b_{11}v_{03} + b_{02}v_{12})xy^4 - (b_{20}v_{12} + b_{11}v_{21} + b_{02}v_{30})x^3y^2 - \\ &-(b_{02}v_{21} + b_{11}v_{12} + b_{20}v_{03})x^2y^3.\end{aligned}$$

$$q_3(x,y)p_2(x,y) - q_2(x,y)r_2(x,y) = (\mathbf{c}_{00} - c_{00})\left(1 + \sum_{|(i,j)|=1}^{2} z_{ij}x^i y^j\right) + \\ + \sum_{|(i,j)|=4}^{5} (q_3p_2 - q_2r_2)_{ij}x^i y^j,$$

wobei

$$\begin{aligned}\sum_{|(i,j)|=4}^{5} (q_3p_2 - q_2r_2)_{ij}x^i y^j =& (c_{02}z_{02} - d_{01}w_{03} - d_{02}w_{02})y^4 + \\ &+(c_{20}z_{11} + c_{11}z_{20} - d_{01}w_{30} - d_{10}w_{21} - d_{20}w_{11} - d_{11}w_{20})x^3y + \\ &+(c_{02}z_{11} + c_{11}z_{02} - d_{10}w_{03} - d_{01}w_{12} - d_{02}w_{11} - d_{11}w_{02})xy^3 + \\ &+(c_{20}z_{02} + c_{02}z_{20} - d_{20}w_{02} - d_{02}w_{20} - d_{10}w_{12} - d_{01}w_{21})x^2y^2 - \\ &-(d_{20}w_{30})x^5 - (d_{02}w_{03})y^5 - (d_{11}w_{30} + d_{20}w_{21})x^4y - \\ &-(d_{11}w_{03} + d_{02}w_{12})xy^4 - (d_{20}w_{12} + d_{11}w_{21} + d_{02}w_{30})x^3y^2 - \\ &-(d_{02}w_{21} + d_{11}w_{12} + d_{20}w_{03})x^2y^3.\end{aligned}$$

Theorem 6.3. *Für Koeffizienten, welche dem Gleichungssystem (26)–(28) genügen, gilt für alle* $x \in \mathbf{x}$

$$\frac{p_1(x,y)}{q_1(x,y)} + \frac{p_2(x,y)}{q_2(x,y)} \subseteq \frac{y_{00} + I_1(x,y) + I_2(x,y) + y_{10}x + y_{01}y + y_{20}x^2 + y_{11}xy + y_{02}y^2}{1 + z_{10}x + z_{01}y + z_{11}xy + z_{20}x^2 + z_{02}y^2},$$

wobei

$$\begin{aligned}
y_{00} &= v_{00} + w_{00} \\
y_{10} &= v_{10} + w_{10} \\
y_{01} &= v_{01} + w_{01} \\
y_{20} &= v_{20} + w_{20} \\
y_{11} &= v_{11} + w_{11} \\
y_{02} &= v_{02} + w_{02} \\
I_1(x,y) &= \frac{(1 + z_{10}x + z_{01}y + z_{11}xy + z_{20}x^2 + z_{02}y^2)(\mathbf{a_{00}} - a_{00})}{1 + b_{10}x + b_{01}y + b_{11}xy + b_{20}x^2 + b_{02}y^2} + \\
&\quad + \frac{\left(\sum_{|(i,j)|=4}^{5} (q_3p_1 - q_1r_1)_{ij} x^i y^j\right)}{1 + b_{10}x + b_{01}y + b_{11}xy + b_{20}x^2 + b_{02}y^2} \\
I_2(x,y) &= \frac{(\mathbf{c_{00}} - c_{00})(1 + z_{10}x + z_{01}y + z_{11}xy + z_{20}x^2 + z_{02}y^2)}{1 + d_{10}x + d_{01}y + d_{11}xy + d_{20}x^2 + d_{02}y^2} + \\
&\quad + \frac{\left(\sum_{|(i,j)|=4}^{5} (q_3p_2 - q_2r_2)_{ij} x^i y^j\right)}{1 + d_{10}x + d_{01}y + d_{11}xy + d_{20}x^2 + d_{02}y^2}.
\end{aligned}$$

Korollar 6.4. *Für Lösungen des Gleichungssystems (26)–(28) kann man* $p_3(x)$ *und* $q_3(x)$ *wählen als*

$$\begin{aligned}
p_3(x,y) &= y_{00} + I_1(\mathbf{x},\mathbf{y}) + I_2(\mathbf{x},\mathbf{y}) + y_{10}x + y_{01}y + y_{20}x^2 + y_{11}xy + y_{02}y^2 \\
q_3(x,y) &= 1 + z_{10}x + z_{01}y + z_{11}xy + z_{20}x^2 + z_{02}y^2.
\end{aligned}$$

Bemerkung 6.5. *Wie schon im Eindimensionalen kann es natürlich auch im zweidimensionalen Fall vorkommen, dass diverse Nenner gleich Null sind. Dadurch müssen einige Koeffizienten frei gewählt werden, wodurch es dazu kommen kann, dass nicht alle Gleichungen des Gleichungssystems (26)–(28) erfüllt sind. Gelöst wird dieses Problem analog zum eindimensionalen Fall, wie in Bemerkung 5.5. beschrieben.*

Gleiche Probleme werden auch bei der Padé-Multiplikation und der Padé-Division wieder auftreten, welche auf die gleiche Art und Weise gelöst werden.

Ebenfalls wie im eindimensionalen Fall könnte man, wenn das Gleichungssystem nicht erfüllt ist und somit das Theorem 6.3. nicht anwendbar ist, die Fehlerabschätzung noch etwas verbessern, indem man x^3 *und* y^3 *quadratisch approximiert.*
Bei der Padé-Multiplikation weiter unter, kann man in jedem Fall eine Verbesserung durch quadratische Approximation erzielen.

6.2 Padé-Multiplikation

In diesem Abschnitt wird gezeigt, wie man ein Padé-Modell für das Produkt von zweidimensionalen Padé-Modellen berechnet.

Problemstellung

Gegeben seien eine zweidimensionale Box $\mathbf{x}$ und die Polynome

$$p_1(x,y) = [\underline{a}, \overline{a}] + \sum_{|(i,j)|=1}^{2} a_{ij} x^i y^j$$

$$q_1(x,y) = 1 + \sum_{|(i,j)|=1}^{2} b_{ij} x^i y^j$$

$$p_2(x,y) = [\underline{c}, \overline{c}] + \sum_{|(i,j)|=1}^{2} c_{ij} x^i y^j$$

$$q_2(x,y) = 1 + \sum_{|(i,j)|=1}^{2} d_{ij} x^i y^j.$$

Es sollen Polynome

$$p_3(x,y) = [\underline{y}, \overline{y}] + \sum_{|(i,j)|=1}^{2} y_{ij} x^i y^j$$

$$q_3(x,y) = 1 + \sum_{|(i,j)|=1}^{2} z_{ij} x^i y^j$$

berechnet werden, sodass folgende Gleichung erfüllt ist

$$\frac{p_1(x,y)}{q_1(x,y)} \cdot \frac{p_2(x,y)}{q_2(x,y)} \subseteq \frac{p_3(x,y)}{q_3(x,y)} \quad \forall (x,y) \in \mathbf{x}.$$

Nach der Definition von zwei Hilfspolynomen erhält man mittels Koeffizientenvergleich, wie im Eindimensionalen, wieder ein Gleichungssystem das es zu lösen gilt.
Seien

$$r_1(x,y) = \sum_{|(i,j)|=0}^{3} v_{ij} x^i y^j$$

$$r_2(x,y) = \sum_{|(i,j)|=0}^{3} w_{ij} x^i y^j,$$

sodass

$$q_3(x,y)\tilde{p}_1(x,y) \approx q_1(x,y)r_1(x,y) \tag{29}$$
$$\tilde{p}_2(x,y) \approx q_2(x,y)r_2(x,y). \tag{30}$$

Wie schon vorher bedeute $\approx$ eine Übereinstimmung bis in möglichst hohe Ordnung und $\tilde{p}_1(x,y)$, $\tilde{p}_2(x,y)$ seien definiert wie in Bemerkung 6.1..

Somit können (29), (30) geschrieben werden als

$$\left(1+\sum_{|(i,j)|=1}^{2} z_{ij}x^iy^j\right)\cdot\left(\sum_{|(i,j)|=0}^{2} a_{ij}x^iy^j\right) \approx \left(1+\sum_{|(i,j)|=1}^{2} b_{ij}x^iy^j\right)\cdot\left(\sum_{|(i,j)|=0}^{3} v_{ij}x^iy^j\right)$$
$$\left(\sum_{|(i,j)|=0}^{2} c_{ij}x^iy^j\right) \approx \left(1+\sum_{|(i,j)|=1}^{2} d_{ij}x^iy^j\right)\cdot\left(\sum_{|(i,j)|=0}^{3} w_{ij}x^iy^j\right).$$

Ein Koeffizientenvergleich der Gleichungen (29) und (30) ergibt folgendes Gleichungssystem

$$\begin{aligned}
c_{00} &= w_{00}\\
c_{10} &= w_{10} + w_{00}d_{10}\\
c_{01} &= w_{01} + w_{00}d_{01}\\
c_{11} &= w_{11} + w_{00}d_{11} + w_{10}d_{01} + w_{01}d_{10}\\
c_{20} &= w_{20} + w_{00}d_{20} + w_{10}d_{10}\\
c_{02} &= w_{02} + w_{00}d_{02} + w_{01}d_{01}\\
0 &= w_{30} + w_{10}d_{20} + w_{20}d_{10}\\
0 &= w_{03} + w_{01}d_{02} + w_{02}d_{01}\\
0 &= w_{21} + w_{10}d_{11} + w_{01}d_{20} + w_{11}d_{10} + w_{20}d_{01}\\
0 &= w_{12} + w_{01}d_{11} + w_{10}d_{02} + w_{11}d_{01} + w_{02}d_{10}
\end{aligned} \tag{31}$$

$$\begin{aligned}
a_{00} &= v_{00}\\
a_{10} + a_{00}z_{10} &= v_{10} + v_{00}b_{10}\\
a_{01} + a_{00}z_{01} &= v_{01} + v_{00}b_{01}\\
a_{11} + a_{00}z_{11} + a_{10}z_{01} + a_{01}z_{10} &= v_{11} + v_{00}b_{11} + v_{10}b_{01} + v_{01}b_{10}\\
a_{20} + a_{00}z_{20} + a_{10}z_{10} &= v_{20} + v_{00}b_{20} + v_{10}b_{10}\\
a_{02} + a_{00}z_{02} + a_{01}z_{01} &= v_{02} + v_{00}b_{02} + v_{01}b_{01}\\
a_{10}z_{20} + a_{20}z_{10} &= v_{30} + v_{10}b_{20} + v_{20}b_{10}\\
a_{01}z_{02} + a_{02}z_{01} &= v_{03} + v_{01}b_{02} + v_{02}b_{01}\\
a_{10}z_{11} + a_{01}z_{20} + a_{11}z_{10} + a_{20}z_{01} &= v_{21} + v_{10}b_{11} + v_{01}b_{20} + v_{11}b_{10} + v_{20}b_{01}\\
a_{01}z_{11} + a_{10}z_{02} + a_{11}z_{01} + a_{02}z_{10} &= v_{12} + v_{01}b_{11} + v_{10}b_{02} + v_{11}b_{01} + v_{02}b_{10}\\
a_{20}z_{20} &= v_{30}b_{10} + v_{20}b_{20}.
\end{aligned} \tag{32}$$

Zur Berechnung der Koeffizienten von $r_1(x,y)$, $r_2(x,y)$ und $q_3(x,y)$ verwendet man folgende 25 Gleichungen.
Aus den Gleichungen (31) des Koeffizientenvergleichs ergeben sich die Koeffizienten von $r_2(x,y)$. Aus den Gleichungen (32) des Koeffizientenvergleichs und den Gleichungen

$$\begin{aligned} v_{00}w_{30} + v_{10}w_{20} + v_{20}w_{10} + v_{30}w_{00} &= 0 \\ v_{00}w_{03} + v_{01}w_{02} + v_{02}w_{01} + v_{03}w_{00} &= 0 \\ v_{00}w_{21} + v_{10}w_{11} + v_{01}w_{20} + v_{11}w_{10} + v_{20}w_{01} + v_{21}w_{00} &= 0 \\ v_{00}w_{12} + v_{10}w_{02} + v_{01}w_{11} + v_{11}w_{01} + v_{02}w_{10} + v_{12}w_{00} &= 0 \end{aligned} \tag{33}$$

können die Koeffizienten der Polynome $r_1(x,y)$ und $q_3(x,y)$ berechnet werden.

Bemerkung 6.6. *Warum gerade die Gleichungen (33) verwendet werden hat den gleichen Grund wie im Eindimesionalen, und ist in Bemerkung 5.6. angeführt.*

Es muss gelten

$$\begin{aligned} \left(\frac{p_1(x,y)}{q_1(x,y)} \cdot \frac{p_2(x,y)}{q_2(x,y)}\right) q_3(x,y) \subseteq r_1(x)r_2(x) + \frac{r_1(x)}{q_2(x)}(p_2(x) - q_2(x)r_2(x)) + \\ + \frac{p_2(x)}{q_1(x)q_2(x)}(q_3(x)p_1(x) - q_1(x)r_1(x)) \quad \forall x \in \mathbf{x}. \end{aligned}$$

Mit den aus dem Gleichungssystem (31)–(33) erhaltenen Koeffizienten ergibt sich

$$\begin{aligned} q_3(x,y)p_1(x,y) - q_1(x,y)r_1(x,y) = (\mathbf{a_{00}} - a_{00}) \left(1 + \sum_{|(i,j)|=1}^{2} z_{ij}x^i y^j\right) + \\ + \sum_{|(i,j)|=4}^{5} (q_3p_1 - q_1r_1)_{ij}x^i y^j, \end{aligned}$$

wobei

$$\begin{aligned} \sum_{|(i,j)|=4}^{5} (q_3p_1 - q_1r_1)_{ij}x^i y^j =& (a_{02}z_{02} - b_{01}v_{03} - b_{02}v_{02})y^4 + \\ &+ (a_{20}z_{11} + a_{11}z_{20} - b_{01}v_{30} - b_{10}v_{21} - b_{20}v_{11} - b_{11}v_{20})x^3y + \\ &+ (a_{02}z_{11} + a_{11}z_{02} - b_{10}v_{03} - b_{01}v_{12} - b_{02}v_{11} - b_{11}v_{02})xy^3 + \\ &+ (a_{20}z_{02} + a_{02}z_{20} - b_{20}v_{02} - b_{11}v_{11} - \\ &- b_{02}v_{20} - b_{10}v_{12} - b_{01}v_{21})x^2y^2 - \\ &- (b_{20}v_{30})x^5 - (b_{02}v_{03})y^5 - (b_{11}v_{30} + b_{20}v_{21})x^4y - \\ &- (b_{11}v_{03} + b_{02}v_{12})xy^4 - (b_{20}v_{12} + b_{11}v_{21} + b_{02}v_{30})x^3y^2 - \\ &- (b_{02}v_{21} + b_{11}v_{12} + b_{20}v_{03})x^2y^3. \end{aligned}$$

$$p_2(x,y) - q_2(x,y)r_2(x,y) = (\mathbf{c_{00}} - c_{00}) + \sum_{|(i,j)|=4}^{5} (p_2 - q_2r_2)_{ij}x^iy^j,$$

wobei

$$\begin{aligned}\sum_{|(i,j)|=4}^{5} (p_2 - q_2r_2)_{ij}x^iy^j =&(-d_{01}w_{03} - d_{02}w_{02})y^4+\\ &+(-d_{01}w_{30} - d_{10}w_{21} - d_{20}w_{11} - d_{11}w_{20})x^3y+\\ &+(-d_{10}w_{03} - d_{01}w_{12} - d_{02}w_{11} - d_{11}w_{02})xy^3+\\ &+(-d_{20}w_{02} - d_{11}w_{11} - d_{02}w_{20} - d_{10}w_{12} - d_{01}w_{21})x^2y^2-\\ &-(d_{20}w_{30})x^5 - (d_{02}w_{03})y^5 - (d_{11}w_{30} + d_{20}w_{21})x^4y-\\ &-(d_{11}w_{03} + d_{02}w_{12})xy^4 - (d_{20}w_{12} + d_{11}w_{21} + d_{02}w_{30})x^3y^2-\\ &-(d_{02}w_{21} + d_{11}w_{12} + d_{20}w_{03})x^2y^3.\end{aligned}$$

$$\begin{aligned}r_1(x,y)r_2(x,y) =& (v_{00} + w_{00}) + (v_{00}w_{10} + v_{10}w_{00})x + (v_{00}w_{01} + v_{01}w_{00})y+\\ &+ (v_{00}w_{11} + v_{10}w_{01} + v_{01}w_{10} + v_{11}w_{00})xy+\\ &+ (v_{00}w_{20} + v_{10}w_{10} + v_{20}w_{00})x^2 + (v_{00}w_{02} + v_{01}w_{01} + v_{02}w_{00})y^2+\\ &+ (v_{00}w_{30} + v_{10}w_{20} + v_{20}w_{10} + v_{30}w_{00})x^3+\\ &+ (v_{00}w_{03} + v_{01}w_{02} + v_{02}w_{01} + v_{03}w_{00})y^3+\\ &+ (v_{00}w_{21} + v_{10}w_{11} + v_{01}w_{20} + v_{11}w_{10} + v_{20}w_{01} + v_{21}w_{00})x^2y+\\ &+ (v_{00}w_{12} + v_{10}w_{02} + v_{01}w_{11} + v_{11}w_{01} + v_{02}w_{10} + v_{12}w_{00})xy^2+\\ &+ \sum_{|(i,j)|=4}^{6} (r_1r_2)_{ij}x^iy^j,\end{aligned}$$

wobei

$$\begin{aligned}\sum_{|(i,j)|=4}^{6} (r_1r_2)_{ij}x^iy^j =&(v_{10}w_{21} + v_{01}w_{30} + v_{11}w_{20} + v_{20}w_{11} + v_{30}w_{01})x^3y+\\ &+(v_{10}w_{12} + v_{01}w_{21} + v_{11}w_{11} + v_{20}w_{02} + v_{02}w_{20})x^2y^2+\\ &+(v_{01}w_{12} + v_{10}w_{03} + v_{11}w_{02} + v_{02}w_{11} + v_{03}w_{10})xy^3+\\ &+(v_{10}w_{30} + v_{20}w_{20})x^4 + (v_{01}w_{03} + v_{02}w_{02})y^4+\\ &+(v_{20}w_{30} + v_{30}w_{20})x^5 + (v_{02}w_{03} + v_{03}w_{02})y^5+\\ &+(v_{11}w_{30} + v_{20}w_{21} + v_{30}w_{11} + v_{21}w_{20})x^4y+\\ &+(v_{11}w_{03} + v_{02}w_{12} + v_{03}w_{11} + v_{12}w_{02})xy^4+\\ &+(v_{20}w_{12} + v_{11}w_{21} + v_{02}w_{30} + v_{30}w_{02} + v_{21}w_{11} + v_{12}w_{20})x^3y^2+\end{aligned}$$

$$
\begin{aligned}
&+(v_{02}w_{21}+v_{11}w_{12}+v_{20}w_{03}+v_{03}w_{20}+v_{12}w_{11}+v_{21}w_{02})x^2y^3+\\
&+(v_{30}w_{30})x^6+(v_{03}w_{03})y^6+(v_{30}w_{21}+v_{21}w_{30})x^5y+\\
&+(v_{03}w_{12}+v_{12}w_{03})xy^5+(v_{30}w_{12}+v_{21}w_{21}+v_{12}w_{30})x^4y^2+\\
&+(v_{03}w_{21}+v_{12}w_{12}+v_{21}w_{03})x^2y^4+\\
&+(v_{30}w_{03}+v_{21}w_{12}+v_{12}w_{21}+v_{03}w_{30})x^3y^3.
\end{aligned}
$$

Theorem 6.7. *Für Koeffizienten welche dem Gleichungssystem (31)-(33) genügen gilt für alle* $x \in \mathbf{x}$

$$
\begin{aligned}
\frac{p_1(x,y)}{q_1(x,y)} \cdot \frac{p_2(x,y)}{q_2(x,y)} \subseteq & \frac{y_{00}+y_{10}x+y_{01}y+y_{11}xy+y_{20}x^2+y_{02}y^2}{1+z_{10}x+z_{01}y+z_{11}xy+z_{20}x^2+z_{02}y^2}+\\
&+\frac{I_1(x,y)+I_2(x,y)+I_3(x,y)}{1+z_{10}x+z_{01}y+z_{11}xy+z_{20}x^2+z_{02}y^2},
\end{aligned}
$$

wobei

$$
\begin{aligned}
y_{00} &= v_{00}w_{00}\\
y_{10} &= v_{00}w_{10}+v_{10}w_{00}\\
y_{01} &= v_{00}w_{01}+v_{01}w_{00}\\
y_{11} &= v_{00}w_{11}+v_{10}w_{01}+v_{01}w_{10}+v_{11}w_{00}\\
y_{20} &= v_{00}w_{20}+v_{10}w_{10}+v_{20}w_{00}\\
y_{02} &= v_{00}w_{02}+v_{01}w_{01}+v_{02}w_{00}
\end{aligned}
$$

$$
I_1(x,y) = \frac{p_2}{q_1q_2}\left(\left(1+\sum_{|(i,j)|=1}^{2} z_{ij}x^iy^j\right)(\boldsymbol{a}_{00}-a_{00})+\left(\sum_{|(i,j)|=4}^{5}(q_3p_1-q_1r_1)_{ij}x^iy^j\right)\right)
$$

$$
I_2(x,y) = \frac{r_1}{q_2}\left((\boldsymbol{c}_{00}-c_{00})-\left(\sum_{|(i,j)|=4}^{5}(q_2r_2)_{ij}x^iy^j\right)\right)
$$

$$
I_3(x,y) = \sum_{|(i,j)|=4}^{6}(r_1r_2)_{ij}x^iy^j.
$$

Korollar 6.8. *Für Koeffizienten, welche durch Lösen des Gleichungssystems (31)-(33) errechnet wurden, kann man* $p_3(x,y)$ *und* $q_3(x,y)$ *wählen als*

$$
\begin{aligned}
p_3(x,y) &= y_{00}+I_1(\mathbf{x},\mathbf{y})+I_2(\mathbf{x},\mathbf{y})+I_3(\mathbf{x},\mathbf{y})+y_{10}x+y_{01}y+y_{11}xy+y_{20}x^2+y_{02}y^2\\
q_3(x,y) &= 1+z_{10}x+z_{01}y+z_{11}xy+z_{20}x^2+z_{02}y^2.
\end{aligned}
$$

6.3 Padé-Division

Nun wird ein Padé-Modell für den Quotienten zweier zweidimensionaler Padé-Modelle berechnet. Es wird also folgendes Problem behandelt.

Problemstellung

Sei $\mathbf{x}$ eine zweidimensionale Box. Aus den gegebenen zweidimensionalen Polynomen $p_1(x,y)$, $q_1(x,y)$, $p_2(x,y)$, $q_2(x,y)$, der Form

$$p_1(x,y) = [\underline{a}, \overline{a}] + \sum_{|(i,j)|=1}^{2} a_{ij} x^i y^j$$
$$q_1(x,y) = 1 + \sum_{|(i,j)|=1}^{2} b_{ij} x^i y^j$$
$$p_2(x,y) = [\underline{c}, \overline{c}] + \sum_{|(i,j)|=1}^{2} c_{ij} x^i y^j$$
$$q_2(x,y) = 1 + \sum_{|(i,j)|=1}^{2} d_{ij} x^i y^j,$$

sollen die Polynome

$$p_3(x,y) = [\underline{y}, \overline{y}] + \sum_{|(i,j)|=1}^{2} y_{ij} x^i y^j$$
$$q_3(x,y) = 1 + \sum_{|(i,j)|=1}^{2} z_{ij} x^i y^j$$

bestimmt werden, sodass diese folgende Gleichung erfüllen

$$\frac{\frac{p_1(x,y)}{q_1(x,y)}}{\frac{p_2(x,y)}{q_2(x,y)}} = \frac{p_1(x,y)}{q_1(x,y)} \cdot \frac{q_2(x,y)}{p_2(x,y)} \subseteq \frac{p_3(x,y)}{q_3(x,y)} \quad \forall (x,y) \in \mathbf{x}.$$

Analog zum eindimensionalen Fall wird versucht die Padé-Division auf die Padé-Multiplikation rückzuführen. Dazu wird zunächst der konstante Teil des Nenners auf Eins normiert.

Unter der Voraussetzung, dass $c_0 \neq 0$ gilt

$$\frac{q_2(x,y)}{p_2(x,y)} = \frac{1 + \sum_{|(i,j)|=1}^{2} d_{ij} x^i y^j}{\mathbf{c_{00}} + \sum_{|(i,j)|=1}^{2} c_{ij} x^i y^j} = \frac{\underbrace{\frac{1}{c_{00}}}_{=:e_{00}} + \sum_{|(i,j)|=1}^{2} e_{ij} x^i y^j}{\underbrace{\frac{\mathbf{c_{00}}}{c_{00}}}_{=:\mathbf{f_{00}}} + \sum_{|(i,j)|=1}^{2} f_{ij} x^i y^j},$$

wobei

$$e_{ij} = \frac{d_{ij}}{c_{00}} \quad \forall |(i,j)| = 1,2$$
$$f_{ij} = \frac{c_{ij}}{c_{00}} \quad \forall |(i,j)| = 1,2.$$

Seien

$$\begin{aligned}\mathbf{I} &= \mathbf{c_{00}} - c_{00}\\ \overline{p_2}(x,y) &= \sum_{|(i,j)|=0}^{2} e_{ij}x^i y^j\\ \overline{q_2}(x,y) &= \mathbf{f_{00}} + \sum_{|(i,j)|=1}^{2} f_{ij}x^i y^j = 1 + \frac{\mathbf{I}}{c_{00}} + \sum_{|(i,j)|=1}^{2} f_{ij}x^i y^j.\end{aligned}$$

Als nächstes definiert man, genau wie bei der Multiplikation, die Hilfspolynome $r_1(x)$, $r_2(x)$. Verwendet man im Gleichungssystem (31) anstatt $c_{(i,j)}$, $d_{(i,j)}$ die soeben berechneten Koeffizienten $e_{(i,j)}$, $f_{(i,j)}$ und bezeichnet das abgeänderte Gleichungssystem mit (31'), so erhält man mittels Koeffizientenvergleich die Gleichungen (31'), (32), (33), welche schon bei der Multiplikation gelöst wurden.

Es gilt für alle $(x,y) \in \mathbf{x}$

$$\begin{aligned}\left(\frac{p_1(x,y)}{q_1(x,y)} \cdot \frac{q_2(x,y)}{p_2(x,y)}\right) q_3(x,y) \subseteq\ & r_1(x,y)r_2(x,y) + \frac{r_1(x,y)}{\overline{q_2}(x,y)}(\overline{p_2}(x,y) - \overline{q_2}(x,y)r_2(x,y)) +\\ & + \frac{\overline{p_2}(x,y)}{q_1(x,y)\overline{q_2}(x,y)}(q_3(x,y)p_1(x,y) - q_1(x,y)r_1(x,y)).\end{aligned}$$

Für Koeffizienten welche dem Gleichungssystem (31'), (32), (33) genügen, ergibt sich

$$\begin{aligned}q_3(x,y)p_1(x,y) - q_1(x,y)r_1(x,y) =& (\mathbf{a_{00}} - a_{00})\left(1 + \sum_{|(i,j)|=1}^{2} z_{ij}x^i y^j\right) +\\ & + \left(\sum_{|(i,j)|=4}^{5} (q_3p_1 - q_1r_1)_{ij}x^i y^j\right),\end{aligned}$$

wobei

$$\begin{aligned}\sum_{|(i,j)|=4}^{5} (q_3p_1 - q_1r_1)_{ij}x^i y^j =& (a_{02}z_{02} - b_{01}v_{03} - b_{02}v_{02})y^4 +\\ & + (a_{20}z_{11} + a_{11}z_{20} - b_{01}v_{30} - b_{10}v_{21} - b_{20}v_{11} - b_{11}v_{20})x^3y +\\ & + (a_{02}z_{11} + a_{11}z_{02} - b_{10}v_{03} - b_{01}v_{12} - b_{02}v_{11} - b_{11}v_{02})xy^3 +\\ & + (a_{20}z_{02} + a_{02}z_{20} - b_{20}v_{02} - b_{02}v_{20} - b_{10}v_{12} - b_{01}v_{21})x^2y^2 -\\ & - (b_{20}v_{30})x^5 - (b_{02}v_{03})y^5 - (b_{11}v_{30} + b_{20}v_{21})x^4y -\\ & - (b_{11}v_{03} + b_{02}v_{12})xy^4 - (b_{20}v_{12} + b_{11}v_{21} + b_{02}v_{30})x^3y^2 -\\ & - (b_{02}v_{21} + b_{11}v_{12} + b_{20}v_{03})x^2y^3\end{aligned}$$

$$\overline{p_2}(x,y) - \overline{q_2}(x,y)r_2(x,y) = \left(\frac{\mathbf{c}_{00} - c_{00}}{c_{00}} \cdot r_2(x,y)\right) + \left(\sum_{|(i,j)|=4}^{5} (\overline{p_2} - \overline{q_2}r_2)_{ij}x^i y^j\right),$$

wobei

$$\begin{aligned}\left(\sum_{|(i,j)|=4}^{5} (\bar{p_2} - \bar{q_2}r_2)_{ij}x^i y^j\right) =& (-f_{01}w_{03} - f_{02}w_{02})y^4 + \\ &+(-f_{01}w_{30} - f_{10}w_{21} - f_{20}w_{11} - f_{11}w_{20})x^3y + \\ &+(-f_{10}w_{03} - f_{01}w_{12} - f_{02}w_{11} - f_{11}w_{02})xy^3 + \\ &+(-f_{20}w_{02} - f_{02}w_{20} - f_{10}w_{12} - f_{01}w_{21})x^2y^2 - \\ &-(f_{20}w_{30})x^5 - (f_{02}w_{03})y^5 - (f_{11}w_{30} + f_{20}w_{21})x^4y - \\ &-(f_{11}w_{03} + f_{02}w_{12})xy^4 - (f_{20}w_{12} + f_{11}w_{21} + f_{02}w_{30})x^3y^2 - \\ &-(f_{02}w_{21} + f_{11}w_{12} + f_{20}w_{03})x^2y^3.\end{aligned}$$

$$\begin{aligned}r_1(x,y)r_2(x,y) =& (v_{00} + w_{00}) + (v_{00}w_{10} + v_{10}w_{00})x + (v_{00}w_{01} + v_{01}w_{00})y + \\ &+ (v_{00}w_{11} + v_{10}w_{01} + v_{01}w_{10} + v_{11}w_{00})xy + \\ &+ (v_{00}w_{20} + v_{10}w_{10} + v_{20}w_{00})x^2 + (v_{00}w_{02} + v_{01}w_{01} + v_{02}w_{00})y^2 + \\ &+ (v_{00}w_{30} + v_{10}w_{20} + v_{20}w_{10} + v_{30}w_{00})x^3 + \\ &+ (v_{00}w_{03} + v_{01}w_{02} + v_{02}w_{01} + v_{03}w_{00})y^3 + \\ &+ (v_{00}w_{21} + v_{10}w_{11} + v_{01}w_{20} + v_{11}w_{10} + v_{20}w_{01} + v_{21}w_{00})x^2y + \\ &+ (v_{00}w_{12} + v_{10}w_{02} + v_{01}w_{11} + v_{11}w_{01} + v_{02}w_{10} + v_{12}w_{00})xy^2 + \\ &+ \left(\sum_{|(i,j)|=4}^{6} (r_1r_2)_{ij}x^i y^j\right)\end{aligned}$$

wobei

$$\begin{aligned}\left(\sum_{|(i,j)|=4}^{6} (r_1r_2)_{ij}x^i y^j\right) =& (v_{10}w_{21} + v_{01}w_{30} + v_{11}w_{20} + v_{20}w_{11} + v_{30}w_{01})x^3y + \\ &+(v_{10}w_{12} + v_{01}w_{21} + v_{11}w_{11} + v_{20}w_{02} + v_{02}w_{20})x^2y^2 + \\ &+(v_{01}w_{12} + v_{10}w_{03} + v_{11}w_{02} + v_{02}w_{11} + v_{03}w_{10})xy^3 + \\ &+(v_{10}w_{30} + v_{20}w_{20})x^4 + (v_{01}w_{03} + v_{02}w_{02})y^4 + \\ &+(v_{20}w_{30} + v_{30}w_{20})x^5 + (v_{02}w_{03} + v_{03}w_{02})y^5 + \\ &+(v_{11}w_{30} + v_{20}w_{21} + v_{30}w_{11} + v_{21}w_{20})x^4y +\end{aligned}$$

$$
\begin{aligned}
&+(v_{11}w_{03} + v_{02}w_{12} + v_{03}w_{11} + v_{12}w_{02})xy^4 + \\
&+(v_{20}w_{12} + v_{11}w_{21} + v_{02}w_{30} + v_{30}w_{02} + v_{21}w_{11} + v_{12}w_{20})x^3y^2 + \\
&+(v_{02}w_{21} + v_{11}w_{12} + v_{20}w_{03} + v_{03}w_{20} + v_{12}w_{11} + v_{21}w_{02})x^2y^3 + \\
&+(v_{30}w_{30})x^6 + (v_{03}w_{03})y^6 + (v_{30}w_{21} + v_{21}w_{30})x^5y + \\
&+(v_{03}w_{12} + v_{12}w_{03})xy^5 + (v_{30}w_{12} + v_{21}w_{21} + v_{12}w_{30})x^4y^2 + \\
&+(v_{03}w_{21} + v_{12}w_{12} + v_{21}w_{03})x^2y^4 + \\
&+(v_{30}w_{03} + v_{21} + w_{12} + v_{12}w_{21} + v_{03}w_{30})x^3y^3.
\end{aligned}
$$

Theorem 6.9. *Werden als Koeffizienten die Lösungen des Gleichungssystems (31'), (32), (33) verwendet, so gilt für alle* $(x, y) \in \mathbf{x}$

$$
\begin{aligned}
\frac{p_1(x,y)}{q_1(x,y)} \cdot \frac{q_2(x,y)}{p_2(x,y)} &= \frac{p_1(x,y)}{q_1(x,y)} \cdot \frac{\bar{p_2}(x,y)}{\bar{q_2}(x,y)} \subseteq \\
&\subseteq \frac{y_{00} + I_1(x,y) + I_2(x,y) + I_3(x,y) + y_{10}x + y_{01}y + y_{11}xy + y_{20}x^2 + y_{02}y^2}{1 + z_{10}x + z_{01}y + z_{11}xy + z_{20}x^2 + z_{02}y^2},
\end{aligned}
$$

wobei

$$
\begin{aligned}
y_{00} &= v_{00}w_{00} \\
y_{10} &= v_{00}w_{10} + v_{10}w_{00} \\
y_{01} &= v_{00}w_{01} + v_{01}w_{00} \\
y_{11} &= v_{00}w_{11} + v_{10}w_{01} + v_{01}w_{10} + v_{11}w_{00} \\
y_{20} &= v_{00}w_{20} + v_{10}w_{10} + v_{20}w_{00} \\
y_{02} &= v_{00}w_{02} + v_{01}w_{01} + v_{02}w_{00}
\end{aligned}
$$

$$
I_1(x,y) = \frac{\overline{p_2}}{q_1\overline{q_2}} \left(\left(1 + \sum_{|(i,j)|=1}^{2} z_{ij}x^iy^j \right) (\mathbf{a_{00}} - a_{00}) + \left(\sum_{|(i,j)|=4}^{5} (q_3p_1 - q_1r_1)_{ij}x^iy^j \right) \right)
$$

$$
I_2(x,y) = \frac{r_1}{\overline{q_2}} \left(\left(\frac{\boldsymbol{c_{00}} - c_{00}}{c_{00}} r_2 \right) + \left(\sum_{|(i,j)|=4}^{5} (\overline{p_2} - \overline{q_2}r_2)_{ij}x^iy^j \right) \right)
$$

$$
I_3(x,y) = \sum_{|(i,j)|=4}^{6} (r_1r_2)_{ij}x^iy^j.
$$

Korollar 6.10. *Ist das Gleichungssystem (31'), (32), (33) lösbar, so können* $p_3(x, y)$ *und* $q_3(x, y)$ *gewählt werden als*

$$
\begin{aligned}
p_3(x,y) &= y_{00} + I_1(\mathbf{x}) + I_2(\mathbf{x}) + I_3(\mathbf{x}) + y_{10}x + y_{01}y + y_{11}xy + y_{20}x^2 + y_{02}y^2 \\
q_3(x,y) &= 1 + z_{10}x + z_{01}y + z_{11}xy + z_{20}x^2 + z_{02}y^2.
\end{aligned}
$$

7 Interpolationsmodell

In diesem Kapitel geht es darum, an dem in den letzten Abschnitten behandelten Padé-Modell einige Veränderungen vorzunehmen, sodass ein neues Modell entsteht. Dieses Modell soll wieder genau das gleiche Problem behandeln wie das Padé-Modell. Zunächst wird kurz die Grundidee dieses Kapitels vorgestellt, bzw. auf welche Art und Weise man das Padé-Modell verändern könnte. Dann wird gezeigt wie man diese Idee umsetzt und somit die Koeffizienten des neuen Modells berechnet. Zum Schluss des Kapitels werden noch verschiedene Methoden zur Berechnung des Fehlerintervalls vorgestellt.

7.1 Motivation

Es werden nun noch einmal kurz die verschiedenen Fehlerfunktionen des Padé-Modells betrachtet. Im eindimensionalen Fall sind das

Addition:

$$\left(\frac{p_1(x)}{q_1(x)} + \frac{p_2(x)}{q_2(x)}\right) q(x) - p(x) \subseteq (v_3 + w_3)x^3 + (v_4 + w_4)x^4 + I_1(x) + I_2(x) \quad \forall x \in \mathbf{x}.$$

Multiplikation:

$$\begin{aligned}\left(\frac{p_1(x)}{q_1(x)} \cdot \frac{p_2(x)}{q_2(x)}\right) q(x) - p(x) \subseteq\ & (v_0w_3 + v_1w_2 + v_2w_1 + v_3w_0)x^3 + \\ & + (v_0w_4 + v_1w_3 + v_2w_2 + v_3w_1 + v_4w_0)x^4 + \\ & + I_1(x) + I_2(x) + I_5(x) + I_6(x) + I_7(x) + I_8(x) \quad \forall x \in \mathbf{x}.\end{aligned}$$

Division:

$$\begin{aligned}\left(\frac{p_1(x)}{q_1(x)} \cdot \frac{p_2(x)}{q_2(x)}\right) q(x) - p(x) \subseteq\ & (v_0w_3 + v_1w_2 + v_2w_1 + v_3w_0)x^3 + \\ & + (v_0w_4 + v_1w_3 + v_2w_2 + v_3w_1 + v_4w_0)x^4 + \\ & + I_1(x) + I_2(x) + I_5(x) + I_6(x) + I_7(x) + I_8(x) \quad \forall x \in \mathbf{x}.\end{aligned}$$

Bemerkung 7.1. *Mit $I_j(x)$, $j = 1, \ldots, 8$ werden die in den Theoremen 5.3., 5.8., 5.13. definierten Funktionen bezeichnet.*

Durch die Wahl der Gleichungen in den verschiedenen Gleichungssystemen, beschrieben in den Bemerkungen 5.1., 5.6., werden die Koeffizienten von x^3 und x^4 beim Padé-Modell Null gesetzt.

Anders ausgedrückt, man versucht für die Fehlerfunktion bei Null eine Nullstelle möglichst hoher Ordnung zu erhalten.

Nun könnte man natürlich auch versuchen, einfache Nullstellen an verschiedenen Punkten des Definitionsbereichs zu erhalten, um so eventuell den maximalen Fehler zu reduzieren.

Dies wird im folgenden Abschnitt die Aufgabe sein.

7.2 Vorgehensweise

In diesem Abschnitt wird nun gezeigt, wie die Koeffizienten dieses Modells berechnet werden. Im Gegensatz zu den beiden vorigen Modellen, wird hier die zu behandelnde Funktion nicht mehr aufgesplittet, sondern direkt so verwendet, wie sie gegeben ist.

Problemstellung

Gegeben sei eine Funktion $f : \mathbb{R}^n \to \mathbb{R}$. Es sollen ein Intervall $\mathbf{I}$ und zwei Polynome $p(x)$, $q(x)$ zweiten Grades berechnet werden, sodass folgende Gleichung erfüllt ist

$$f(x) \in \frac{p(x) + \mathbf{I}}{q(x)} \quad \forall x \in \mathbf{x} = [-h, h]^n, \quad h \in \mathbb{R}^n_+,$$

wobei

$$p(x) = y_0 + \sum_{|\alpha|=1}^{2} y_\alpha x^\alpha$$

$$q(x) = 1 + \sum_{|\alpha|=1}^{2} z_\alpha x^\alpha.$$

Bemerkung 7.2. *Mit α wird hier ein Multiindex bezeichnet.*

Nun wird versucht die Fehlerfunktion an verschiedenen Stellen Null zu setzen. Ziel ist es also, Folgendes zu erreichen

$$\frac{p(x)}{q(x)} = f(x) \quad \forall x \in M,$$

wobei M jene Menge ist, welche die oben besprochenen Punkte enthält, und somit auf jeden Fall endlich ist.

Zum Beispiel könnte M, wenn man das Intervall [-1,1] betrachtet, folgende Menge sein

$$M = \{-1, -0.5, 0, 0.5, 1\}.$$

Theorem 7.3. *Sei n die Dimension des Problems. Dann ist für α, β, $\gamma \in \mathbb{R}_+$ und*

$$M = \{0, \pm\alpha h_i e_i, \pm\beta h_i e_i, \pm\gamma(h_j e_j \pm h_k e_k) | i, j, k \in \{1, \cdots, n\}, j \neq k\}$$

garantiert, dass die Anzahl der Elemente von M größer gleich der Anzahl der zu bestimmenden Koeffizienten y_α, z_α für $|\alpha| \in \{0, 1, 2\}$ ist.

Beweis. Sei k die Anzahl der zu bestimmenden Parameter, dann gilt

$$k = 2(1 + n + \frac{n(n+1)}{2}) - 1 = n^2 + 3n + 1$$
$$|M| = 1 + 4n + 4\frac{n(n-1)}{2} = 2n^2 + 2n + 1.$$

Zu zeigen bleibt

$$2n^2 + 2n + 1 \geq n^2 + 3n + 1$$
$$\Leftrightarrow \qquad n^2 \geq n$$

Da das für $n \geq 1$ immer erfüllt ist, folgt daraus die Behauptung.

□

Um die Koeffizienten von $p(x)$ und $q(x)$ zu bestimmen muss man also folgendes konvexes, quadratisches Optimierungsproblem lösen

$$\begin{aligned} \min \quad & \sum_{x \in M} (f(x)q(x) - p(x))^2 \\ s.t. \quad & q(x) \geq \epsilon \qquad \forall x \in M \qquad \epsilon > 0. \end{aligned}$$

Bemerkung 7.4. *Im Implementationsteil dieser Arbeit werden diese Optimierungsprobleme mittels einer Matlab Software namens CVX gelöst. CVX ist eine von Michael Grant und Stephen Boyd entwickelte Matlab Bibliothek welche frei zum Download steht ([8]). Mittels dieser Matlab Anwendung lassen sich konvexe Optimierungsprobleme lösen. Außerdem verfügt CVX über einen speziellen GP (geometric programming) mode. Genauere Informationen können in [8] nachgelesen werden.*

7.3 Fehlerabschätzung

In diesem Abschnitt werden nun verschiedene Methoden vorgestellt, mit denen ein Fehlerintervall berechnet werden kann. Dazu muss allerdings die zu behandelnde Funktion wieder aufgesplittet werden. Es werden Addition, Multiplikation, etc. also wieder getrennt betrachtet. Nachdem man die Koeffizienten von $p(x)$ und $q(x)$ bestimmt hat, gilt es in diesem Teil also folgendes Problem zu lösen.

Gesucht ist ein Intervall $\mathbf{I}$ für das gilt

$$\left(\frac{p_1(x)}{q_1(x)} + \frac{p_2(x)}{q_2(x)}\right) q(x) - p(x) \in \mathbf{I} \quad \forall x \in \mathbf{x},$$

wobei die Polynome

$$\begin{aligned} p_1(x) &= [\underline{a_0}, \overline{a_0}] + \sum_{|\alpha|=1}^{2} a_\alpha x^\alpha \\ q_1(x) &= 1 + \sum_{|\alpha|=1}^{2} b_\alpha x^\alpha \\ p_2(x) &= [\underline{c_0}, \overline{c_0}] + \sum_{|\alpha|=1}^{2} c_\alpha x^\alpha \\ q_2(x) &= 1 + \sum_{|\alpha|=1}^{2} d_\alpha x^\alpha \end{aligned}$$

gegeben sind und die Polynome $p(x)$, $q(x)$ schon berechnet wurden.

7.3.1 Methode A

Die naheliegendste und einfachste Methode ist, für jede Variable x das zu betrachtende Intervall $\mathbf{x}$ einzusetzen.

Addition

$$\left(\frac{p_1(x)}{q_1(x)} + \frac{p_2(x)}{q_2(x)}\right) q(x) - p(x) \quad \in \quad \left(\frac{p_1(\mathbf{x})}{q_1(\mathbf{x})} + \frac{p_2(\mathbf{x})}{q_2(\mathbf{x})}\right) q(\mathbf{x}) - p(\mathbf{x}) \quad \forall x \in \mathbf{x}$$

Multiplikation

$$\left(\frac{p_1(x)}{q_1(x)} \cdot \frac{p_2(x)}{q_2(x)}\right) q(x) - p(x) \quad \in \quad \left(\frac{p_1(\mathbf{x})}{q_1(\mathbf{x})} \cdot \frac{p_2(\mathbf{x})}{q_2(\mathbf{x})}\right) q(\mathbf{x}) - p(\mathbf{x}) \quad \forall x \in \mathbf{x}$$

Division

$$\left(\frac{p_1(x)}{q_1(x)} \cdot \frac{q_2(x)}{p_2(x)}\right) q(x) - p(x) \quad \in \quad \left(\frac{p_1(\mathbf{x})}{q_1(\mathbf{x})} \cdot \frac{q_2(\mathbf{x})}{p_2(\mathbf{x})}\right) q(\mathbf{x}) - p(\mathbf{x}) \quad \forall x \in \mathbf{x}$$

7.3.2 Methode A'

Um eine zu große Überschätzung zu vermeiden, wird das zu betrachtende Intervall in mehrere disjunkte Teilintervalle aufgesplittet, deren Vereinigung wieder das ganze Intervall ist.
Sei $\mathbf{I} = [a, b]$. Definiere $t_1, t_2, \cdots, t_m$ sodass $a < t_1 < t_2 < \cdots < t_n < b$ und betrachte ab jetzt die Teilintervalle $I_0 = [a, t_1]$, $I_1 = [t_1, t_2]$, $I_2 = [t_2, t_3]$, $\cdots$, $I_{n-1} = [t_{n-1}, t_n]$, $I_n = [t_n, b]$, deren Vereinigung wieder $\mathbf{I}$ ist.

Bemerkung 7.5.
- *Klarerweise gilt: Geht die Anzahl der Teilintervalle gegen unendlich, so geht die Überschätzung gegen Null.*
- *Im Mehrdimensionalen funktioniert das Ganze vollkommen analog, hat allerdings exponentiellen Aufwand.*

Um weiterzumachen benötigt man zunächst folgendes Theorem.

Theorem 7.6. *Seien zwei n-dimensionale Boxen* $\mathbf{I_1}$, $\mathbf{I_2}$ *und eine Funktion* $f : \mathbb{R}^n \to \mathbb{R}^m$ *gegeben. Dann gilt*

$$f(I_1 \cup I_2) = f(I_1) \cup f(I_2).$$

Beweis.

$$\begin{aligned}
& y \in f(I_1 \cup I_2) && \Leftrightarrow \\
\Leftrightarrow \quad & y \in \{f(x) : x \in I_1 \vee x \in I_2\} && \Leftrightarrow \\
\Leftrightarrow \quad & y \in \{f(x) : x \in I_1\} \cup \{f(x) : x \in I_2\} && \Leftrightarrow \\
\Leftrightarrow \quad & y \in f(I_1) \cup f(I_2)
\end{aligned}$$

□

In Verbindung mit Methode A erhält man für die verschiedenen Operationen also folgende Fehlerabschätzungen.

Theorem 7.7. *Seien* $p_1(x)$, $q_1(x)$, $p_2(x)$, $q_2(x)$, $p(x)$, $q(x)$ *k-dimensionale Polynome vom Grad zwei. Für k-dimensionale Boxen* $\mathbf{x}$ *und* $\mathbf{I_j}$, $j = 0, 1, ..., n$ *mit der Eigenschaft*

$$\mathbf{x} = \bigcup_{j=0}^{n} \mathbf{I_j}$$

gilt

$$\begin{aligned}
\left(\frac{p_1(x)}{q_1(x)} + \frac{p_2(x)}{q_2(x)}\right) q(x) - p(x) \in & \left(\frac{p_1(I_0)}{q_1(I_0)} + \frac{p_2(I_0)}{q_2(I_0)}\right) q(I_0) - p(I_0) \cup \\
& \cup \left(\frac{p_1(I_1)}{q_1(I_1)} + \frac{p_2(I_1)}{q_2(I_1)}\right) q(I_1) - p(I_1) \cup \cdots \cup \\
& \cup \left(\frac{p_1(I_n)}{q_1(I_n)} + \frac{p_2(I_n)}{q_2(I_n)}\right) q(I_n) - p(I_n) \quad \forall x \in \mathbf{x}
\end{aligned}$$

$$\left(\frac{p_1(x)}{q_1(x)} \cdot \frac{p_2(x)}{q_2(x)}\right) q(x) - p(x) \in \left(\frac{p_1(I_0)}{q_1(I_0)} \cdot \frac{p_2(I_0)}{q_2(I_0)}\right) q(I_0) - p(I_0) \cup$$
$$\cup \left(\frac{p_1(I_1)}{q_1(I_1)} \cdot \frac{p_2(I_1)}{q_2(I_1)}\right) q(I_1) - p(I_1) \cup \cdots \cup$$
$$\cup \left(\frac{p_1(I_n)}{q_1(I_n)} \cdot \frac{p_2(I_n)}{q_2(I_n)}\right) q(I_n) - p(I_n) \quad \forall x \in \mathbf{x}$$

$$\left(\frac{p_1(x)}{q_1(x)} \cdot \frac{q_2(x)}{p_2(x)}\right) q(x) - p(x) \in \left(\frac{p_1(I_0)}{q_1(I_0)} \cdot \frac{q_2(I_0)}{p_2(I_0)}\right) q(I_0) - p(I_0) \cup$$
$$\cup \left(\frac{p_1(I_1)}{q_1(I_1)} \cdot \frac{q_2(I_1)}{p_2(I_1)}\right) q(I_1) - p(I_1) \cup \cdots \cup$$
$$\cup \left(\frac{p_1(I_n)}{q_1(I_n)} \cdot \frac{q_2(I_n)}{p_2(I_n)}\right) q(I_n) - p(I_n) \quad \forall x \in \mathbf{x}.$$

Bemerkung 7.8. *Verwendet man die Methode A' und somit das Theorem 7.7. für eine Fehlerabschätzung, so ist es sinnvoll die k-dimensionalen Boxen* $\mathbf{I_j}$, $j = 0, ..., n$ *so zu wählen, dass deren Schnitt leer ist.*

7.3.3 Methode B

Natürlich kann man den abzuschätzenden Term auch vorher ausmultiplizieren. Dann werden bei der Abschätzung einige Koeffizienten sehr klein sein und daher nur mehr geringen Einfluss auf das Ergebnis haben.

Addition

$$\left(\frac{p_1(x)}{q_1(x)} + \frac{p_2(x)}{q_2(x)}\right) q(x) - p(x) =$$

$$= \frac{p_1(x)q_2(x) + p_2(x)q_1(x)}{q_1(x)q_2(x)} q(x) - p(x) =$$

$$= \frac{p_1(x)q_2(x)q(x) + p_2(x)q_1(x)q(x) - p(x)q_1(x)q_2(x)}{q_1(x)q_2(x)}$$

Multipliziert man nun Zähler und Nenner aus so erhält man ein Polynom sechsten und ein Polynom vierten Grades. Daher kann man $c_\alpha, d_\beta \in \mathbb{R}$ so wählen, dass

$$\begin{aligned} \sum_{\alpha=0}^{6} c_\alpha x^\alpha &= p_1(x)q_2(x)q(x) + p_2(x)q_1(x)q(x) - p(x)q_1(x)q_2(x) \\ 1 + \sum_{\beta=1}^{4} d_\beta x^\beta &= q_1(x)q_2(x). \end{aligned} \tag{34}$$

Daraus folgt

$$\left(\frac{p_1(x)}{q_1(x)} + \frac{p_2(x)}{q_2(x)}\right) q(x) - p(x) = \frac{\sum_{\alpha=0}^{6} c_\alpha x^\alpha}{1 + \sum_{\beta=1}^{4} d_\beta x^\beta} \in \frac{\sum_{\alpha=0}^{6} c_\alpha \mathbf{x}^\alpha}{1 + \sum_{\beta=1}^{4} d_\beta \mathbf{x}^\beta} \quad \forall x \in \mathbf{x}.$$

Multiplikation

$$\left(\frac{p_1(x)}{q_1(x)} \cdot \frac{p_2(x)}{q_2(x)}\right) q(x) - p(x) = \frac{p_1(x)p_2(x)q(x) - p(x)q_1(x)q_2(x)}{q_1(x)q_2(x)}$$

Seien $c_\alpha, d_\beta \in \mathbb{R}$ so gewählt, dass

$$\begin{aligned} \sum_{\alpha=0}^{6} c_\alpha x^\alpha &= p_1(x)p_2(x)q(x) - p(x)q_1(x)q_2(x) \\ 1 + \sum_{\beta=1}^{4} d_\beta x^\beta &= q_1(x)q_2(x). \end{aligned} \tag{35}$$

Dann gilt

$$\left(\frac{p_1(x)}{q_1(x)} \cdot \frac{p_2(x)}{q_2(x)}\right) q(x) - p(x) = \frac{\sum_{\alpha=0}^{6} c_\alpha x^\alpha}{1 + \sum_{\beta=1}^{4} d_\beta x^\beta} \in \frac{\sum_{\alpha=0}^{6} c_\alpha \mathbf{x}^\alpha}{1 + \sum_{\beta=1}^{4} d_\beta \mathbf{x}^\beta} \quad \forall x \in \mathbf{x}.$$

Division

$$\left(\frac{p_1(x)}{q_1(x)} \cdot \frac{q_2(x)}{p_2(x)}\right) q(x) - p(x) = \qquad \frac{p_1(x)q_2(x)q(x) - p(x)q_1(x)p_2(x)}{q_1(x)p_2(x)}$$

Wählt man $c_\alpha, d_\beta \in \mathbb{R}$ so, dass

$$\begin{aligned} \sum_{\alpha=0}^{6} c_\alpha x^\alpha &= p_1(x)q_2(x)q(x) - p(x)q_1(x)p_2(x) \\ 1 + \sum_{\beta=1}^{4} d_\beta x^\beta &= q_1(x)p_2(x), \end{aligned} \tag{36}$$

so folgt

$$\left(\frac{p_1(x)}{q_1(x)} \cdot \frac{q_2(x)}{p_2(x)}\right) q(x) - p(x) = \frac{\sum_{\alpha=0}^{6} c_\alpha x^\alpha}{1 + \sum_{\beta=1}^{4} d_\beta x^\beta} \in \frac{\sum_{\alpha=0}^{6} c_\alpha \mathbf{x}^\alpha}{1 + \sum_{\beta=1}^{4} d_\beta \mathbf{x}^\beta} \quad \forall x \in \mathbf{x}.$$

7.3.4 Methode B'

Man kann auch hier wieder das zu betrachtende Intervall aufsplitten und die einzelnen Fehlerintervalle getrennt berechnen, um eine zu große Überschätzung zu vermeiden. Es sollte allerdings bemerkt werden, dass dies eine längere Rechenzeit benötigt.
Es gelten also folgende Fehlerabschätzungen:

Theorem 7.9. *Seien die Voraussetzungen von Theorem 7.7. gegeben. Werden c_α, d_α gewählt wie in (34), dann gilt*

$$\left(\frac{p_1(x)}{q_1(x)} + \frac{p_2(x)}{q_2(x)}\right) q(x) - p(x) \in \frac{\sum_{\alpha=0}^{6} c_\alpha I_0^\alpha}{1 + \sum_{\beta=1}^{4} d_\beta I_0^\beta} \cup \frac{\sum_{\alpha=0}^{6} c_\alpha I_1^\alpha}{1 + \sum_{\beta=1}^{4} d_\beta I_1^\beta} \cup \cdots \cup \frac{\sum_{\alpha=0}^{6} c_\alpha I_n^\alpha}{1 + \sum_{\beta=1}^{4} d_\beta I_n^\beta} \quad \forall x \in \mathbf{x}.$$

Wählt man die Koeffizienten c_α, d_α so, dass sie (35) genügen, so folgt

$$\left(\frac{p_1(x)}{q_1(x)} \cdot \frac{p_2(x)}{q_2(x)}\right) q(x) - p(x) \in \frac{\sum_{\alpha=0}^{6} c_\alpha I_0^\alpha}{1 + \sum_{\beta=1}^{4} d_\beta I_0^\beta} \cup \frac{\sum_{\alpha=0}^{6} c_\alpha I_1^\alpha}{1 + \sum_{\beta=1}^{4} d_\beta I_1^\beta} \cup \cdots \cup \frac{\sum_{\alpha=0}^{6} c_\alpha I_n^\alpha}{1 + \sum_{\beta=1}^{4} d_\beta I_n^\beta} \quad \forall x \in \mathbf{x}.$$

Sei für c_α, d_α die Gleichung (36) gültig, so gilt

$$\left(\frac{p_1(x)}{q_1(x)} \cdot \frac{q_2(x)}{p_2(x)}\right) q(x) - p(x) \in \frac{\sum_{\alpha=0}^{6} c_\alpha I_0^\alpha}{1 + \sum_{\beta=1}^{4} d_\beta I_0^\beta} \cup \frac{\sum_{\alpha=0}^{6} c_\alpha I_1^\alpha}{1 + \sum_{\beta=1}^{4} d_\beta I_1^\beta} \cup \cdots \cup \frac{\sum_{\alpha=0}^{6} c_\alpha I_n^\alpha}{1 + \sum_{\beta=1}^{4} d_\beta I_n^\beta} \quad \forall x \in \mathbf{x}.$$

7.3.5 Methode C

Wie in den anfänglichen Kapiteln vorgestellt, kann man das Fehlerintervall auch mittels Slopes abschätzen. Wie das für ein- und zweidimensionale Slopes genau aussieht, folgt in diesem Abschnitt.

Addition

Seien $z, z' \in \mathbf{x}$, $z \neq z'$.

$$\underbrace{\left(\frac{p_1(x)}{q_1(x)} + \frac{p_2(x)}{q_2(x)}\right) q(x) - p(x)}_{=:f(x)} \in f(z) + f[z, \mathbf{x}](\mathbf{x} - z) \quad \forall x \in \mathbf{x}$$

$$\left(\frac{p_1(x)}{q_1(x)} + \frac{p_2(x)}{q_2(x)}\right) q(x) - p(x) \in f(z) + \left(f[z, z'] + \left(\mathbf{x} - z'\right)^{\mathsf{T}} f[z, z', \mathbf{x}]\right)(\mathbf{x} - z) \quad \forall x \in \mathbf{x}$$

Multiplikation

Seien $z, z' \in \mathbf{x}$, $z \neq z'$.

$$\underbrace{\left(\frac{p_1(x)}{q_1(x)} \cdot \frac{p_2(x)}{q_2(x)}\right) q(x) - p(x)}_{=:f(x)} \in f(z) + f[z, \mathbf{x}](\mathbf{x} - z) \quad \forall x \in \mathbf{x}$$

$$\left(\frac{p_1(x)}{q_1(x)} \cdot \frac{p_2(x)}{q_2(x)}\right) q(x) - p(x) \in f(z) + \left(f[z, z'] + \left(\mathbf{x} - z'\right)^{\mathsf{T}} f[z, z', \mathbf{x}]\right)(\mathbf{x} - z) \quad \forall x \in \mathbf{x}$$

Division

Seien $z, z' \in \mathbf{x}$, $z \neq z'$.

$$\underbrace{\left(\frac{p_1(x)}{q_1(x)} \cdot \frac{q_2(x)}{p_2(x)}\right) q(x) - p(x)}_{=:f(x)} \in f(z) + f[z, \mathbf{x}](\mathbf{x} - z) \quad \forall x \in \mathbf{x}$$

$$\left(\frac{p_1(x)}{q_1(x)} \cdot \frac{q_2(x)}{p_2(x)}\right) q(x) - p(x) \in f(z) + \left(f[z, z'] + \left(\mathbf{x} - z'\right)^{\mathsf{T}} f[z, z', \mathbf{x}]\right)(\mathbf{x} - z) \quad \forall x \in \mathbf{x}$$

Bemerkung 7.10. *Die Slopes können natürlich auf verschiedene Arten berechnet werden. Einerseits direkt, indem man zum Beispiel vorher $f(x)$ ausmultipliziert. Andererseits kann man zuerst Slopes für die verschiedenen Polynome berechnen, und daraus, mittels der in den Kapiteln über Slopes behandelten Rechenregeln, die Slopes für $f(x)$.*

8 Implementationen

Bemerkung 8.1. *In diesem Kapitel werden die in den Beispielen verwendeten Algorithmen stichwortartig beschrieben. Die Quellcodes aller verwendeten Algorithmen stehen im Anhang A dieser Arbeit. Das genaue Unterkapitel wird hier jeweils in Klammer angegeben.*

Algorithmen einDmain(A.1), zweiDmain(A.10), dreiDmain(A.21)

Diese Algorithmen berechnen aus den Punkten der gegebenen Menge M, den Funktionswerten an diesen Punkten und dem Definitionsbereich, abhängig von der Dimension des Problems, mittels der Algorithmen einDKoeffCVX(A.2), zweiDKoeffCVX(A.11), dreiDKoeffCVX(A.22) die Koeffizienten der Polynome, welche die zu betrachtende Funktion approximieren. Gelöst werden diese Probleme, wie der Name vermuten lässt, mit dem CVX-Paket. Außerdem wird ein Fehlerintervall berechnet.

Alle nachfolgenden Algorithmen berechnen für ein Problem der Form $\left(\frac{p_1}{q_1} \circ \frac{p_2}{q_2}\right) q - p$ eine Intervallabschätzung des Wertebereichs. Gegeben seien dafür die Koeffizienten der Polynome, die Rechenoperation, die Fehlerintervalle von p_1 und p_2, die Definitionsbereiche der verschiedenen Variablen.

Algorithmen einDfehler(A.6), zweiDfehler(A.15), dreiDfehler(A.23)

Die Algorithmen einDfehler, zweiDfehler und dreiDfehler bestimmen ein Intervall durch Einsetzen des Definitionsbereichs für jede Variable.

Algorithmen einDfehlerslope(A.3), zweiDfehlerslope(A.12), dreiDfehlerslope(A.24)

Diese Algorithmen berechnen mehrere Intervalle, welche auf folgende Arten bestimmt werden:

- Abschätzung über Slopes erster Ordnung an zwei verschiedenen Punkten, wobei die Slopes mittels der in Kapitel 2 vorgestellten Operationen aus den Slopes der Polynome berechnet werden.

- Abschätzung über Slopes zweiter Ordnung, wobei die Slopes ebenfalls aus den Slopes der Polynome berechnet werden.
- Mittels der Algorithmen einDfehler(A.6), zweiDfehler(A.15), dreiDfehler(A.23).

Ausgegeben wird der Schnitt dieser Intervalle.

Algorithmen einDFehlerzerteilung(A.7), zweiDFehlerzerteilung(A.16)

Der Definitionsbereich wird in beliebig viele Teilboxen unterteilt. Für diese Teilboxen wird mittels geeigneten Algorithmus ein Intervall berechnet. (In den obigen Beispielen wurden dafür die Algorithmen einDfehlerslope(A.3), zweiDfehlerslope(A.12) bzw. einDfehlermult(A.8), zweiDfehlermult(A.17) verwendet.) Danach wird die Vereinigung all dieser Intervalle ausgegeben.

Algorithmen einDfehlermult(A.8), zweiDfehlermult(A.17)

In den Algorithmen einDfehlermult, zweiDfehlermult wird zunächst das zu behandelnde Problem mittels der Algorithmen eindimausmultipl(A.9), zweidimausmultipl(A.18) ausmultipliziert, und zum Beispiel eine Addition von Polynomen auf folgende Form gebracht:

$$\frac{p_1(x)q_2(x)q(x) + p_2(x)q_1(x)q(x) - p(x)q_1(x)q_2(x)}{q_1(x)q_2(x)}.$$

Danach wird ein Intervall durch Einsetzen des Definitionsbereichs für jede Variable bestimmt.

Algorithmus zweiDfehlerslopemult(A.19)

Dieser Algorithmus multipliziert das Problem wie Algorithmus zweiDfehlermult(A.17) aus, und berechnet anschließend mehrere Intervalle:

- Abschätzung über Slopes erster Ordnung an zwei verschiedenen Punkten, wobei die Slopes direkt für die ganze Funktion berechnet werden.
- Abschätzung über Slopes zweiter Ordnung, wobei die Slopes wieder direkt für die ganze Funktion berechnet werden.
- Mittels Algorithmus zweiDfehlermult(A.17).

Ausgegeben wird der Schnitt der vier Intervalle.

Algorithmen zum Berechnen von Slopes

Mittels der Algorithmen slope1(A.4), slope2(A.13), slope3(A.25), slopepoly(A.5), slopepoly2(A.14), slopepoly3(A.26), zweiDmultslope(A.20) werden die Koeffizenten der benötigten Slopes berechnet.

Betrachtet man noch einmal die in Abschnitt 7.3 beschriebenen Methoden, so können diese auf folgende Weise realisiert werden.

Methode A

- In Abhängigkeit der Dimension wird ein Intervall mittels der Algorithmen einDfehler, zweiDfehler, dreiDfehler berechnet.

Methode A'

- Unterteilung des Definitionsbereichs mittels Algorithmus einDFehlerzerteilung bzw. zweiDFehlerzerteilung in Teilboxen.
- Berechnung eines Fehlerintervalls mittels Algorithmus einDfehler bzw. zweiDfehler für jede dieser Teilboxen.
- Ausgabe der Vereinigung dieser Intervalle.

Methode B

- Je nach Dimension wird ein Intervall mittels der Algorithmen einDfehlermult, zweiDfehlermult bestimmt.

Methode B'

- Unterteilung des Definitionsbereichs mittels Algorithmus einDFehlerzerteilung bzw. zweiDFehlerzerteilung in Teilboxen.
- Berechnung eines Fehlerintervalls mittels Algorithmus einDfehlermult bzw. zweiDfehlermult für jede dieser Teilboxen.
- Ausgabe der Vereinigung dieser Intervalle.

Methode C

- Bestimmung eines Intervalls mittels einem der Algorithmen einDfehlerslope, zweiDfehlerslope bzw. dreiDfehlerslope.

9 Beispiele

In diesem Kapitel werden für einige Funktionen verschiedener Dimensionen die bisher behandelten Modelle berechnet. Ein Großteil davon wird mittels der im Anhang A angeführten Algorithmen gelöst. Insbesondere werden die Modelle für verschieden Definitionsbereiche berechnet, und anschließend miteinander verglichen:

9.1 Eindimensionales Beispiel

Aufgabenstellung

Es soll ein Taylor-Modell mittels Taylorarithmetik, ein Padé-Modell mittels Padéarithmetik, und ein Interpolationsmodell für die Funktion

$$f(x) = \frac{1}{x+2} + (x+2)^2$$

berechnet werden. Die Berechnung soll für die Intervalle $x \in [-0.1, 0.1]$, $x \in [-0.01, 0.01]$, $x \in [-10^{-4}, 10^{-4}]$ durchgeführt werden.

Taylor-Modell für $x \in [-0.1, 0.1]$

$$T_{(x+2)} = ((x+2), [0,0])$$
$$T_{(x+2)^2} = ((x+2)^2, [0,0])$$

Laut Formel für $1/f$ in der Taylor Arithmetik gilt für $0 < \theta < 1$

$$1/(x+2) = 1/c_f \left\{ 1 - \frac{\bar{f}(x)}{c_f} + \cdots + (-1)^k \frac{(\bar{f}(x))^k}{c_f^k} \right\} + (-1)^{k+1} \frac{(\bar{f}(x)^{k+1}}{c_f^{k+2}} \frac{1}{(1+\theta \bar{f}(x)/c_f)^{k+2}}.$$

Wählt man $k = 2$ so folgt

$$-\frac{x^3}{c_i^4} \frac{1}{(1+\theta x/c_f)^4} \quad \in \quad -\frac{[-0.001, 0.001]}{2^4([0.95, 1.05])^4} \quad = \quad [-7.67336 \cdot 10^{-5}, 7.67336 \cdot 10^{-5}].$$

Also erhält man für die beiden Terme von $f(x)$ die Taylor-Modelle

$$T_{1/(x+2)} = (1/2 - 1/4x + 1/8x^2, [-7.67336 \cdot 10^{-5}, 7.67336 \cdot 10^{-5}])$$
$$T_{(x+2)^2} = ((x+2)^2, [0,0]) = (4 + 4x + x^2, [0,0]).$$

Daher gilt

$$f(x) = \frac{1}{x+2} + (x+2)^2 = 9/2 + 15/4x + 9/8x^2 + [-7.67336 \cdot 10^{-5}, 7.67336 \cdot 10^{-5}].$$

Padé-Modell für $x \in [-0.1, 0.1]$

Um die bisher kennengelernten Formeln für das Padé-Modell anwenden zu können, muss das Problem vorher umformuliert werden zu

$$\frac{p_1(x)}{q_1(x)} + \frac{p_2(x)}{q_2(x)} = \frac{1}{2+x} + \frac{4+4x+x^2}{1} = \frac{1/2}{1+1/2x} + \frac{4+4x+x^2}{1}.$$

Wird nun das entstehende Gleichungssystem (7)–(10) mittels Theorem 5.2. gelöst, so erhält man die Koeffizienten

$$\begin{array}{lll} z_1 = 1/8 & v_1 + w_1 = -3/16 + 9/2 = 69/16 & v_3 + w_3 = -1/24 + 1/24 = 0 \\ z_2 = -1/48 & v_2 + w_2 = 1/12 + 17/12 = 3/2 & v_4 + w_4 = 1/48 - 1/48 = 0 \\ & v_0 + w_0 = 1/2 + 4 = 9/2. & \end{array}$$

Wird nun die Formel für $p_3(x)$ aus Korollar 5.4. verwendet so erhält man

$$(\frac{9}{2} + \frac{69}{16}x + \frac{3}{2}x^2) + \frac{(1 + z_1x + z_2x^2)\underbrace{(\mathbf{a}_0 - a_0)}_{=0} - \underbrace{b_2}_{=0} v_4x^6 - (b_1v_4 + \underbrace{b_2}_{=0} v_3)x^5}{1+1/2x} +$$

$$+\frac{(1 + z_1x + z_2x^2)\underbrace{(\mathbf{c}_0 - c_0)}_{=0} - \underbrace{d_2}_{=0} w_4x^6 - (\underbrace{d_1}_{=0} w_4 + \underbrace{d_2}_{=0} w_3)x^5}{1} =$$

$$=(\frac{9}{2} + \frac{69}{16}x + \frac{3}{2}x^2) + \frac{-b_1v_4x^5}{1+1/2x} \in (\frac{9}{2} + \frac{69}{16}x + \frac{3}{2}x^2) + \frac{[-1/96 \cdot 10^{-5}, 1/96 \cdot 10^{-5}]}{[0.95, 1.05]} =$$

$$=\frac{9}{2} + \frac{69}{16}x + \frac{3}{2}x^2 + [-1.09649 \cdot 10^{-7}, 1.09649 \cdot 10^{-7}] =: p_3(x).$$

Insgesamt ergibt das

$$f(x) \in \frac{\frac{9}{2} + \frac{69}{16}x + \frac{3}{2}x^2 + [-1.09649 \cdot 10^{-7}, 1.09649 \cdot 10^{-7}]}{1 + \frac{1}{8}x - \frac{1}{48}x^2} \quad \forall x \in [-0.1, 0.1].$$

Wird das Gleichungssystem mittels eines Algorithmus gelöst, so entstehen Rundungsfehler. Es könnten zum Beispiel folgende Werte berechnet werden

$$\begin{array}{ll} v_1 + w_1 = -0.1875 + 4.5 = 4.3125 & v_3 + w_3 = -0.041666 + 0.041666 = 0 \\ v_2 + w_2 = 0.083333 + 1.4166 = 1.499933 & v_4 + w_4 = 0.020833 - 0.020833 = 0 \\ v_0 + w_0 = 0.5 + 4 = 4.5 & z_1 = 0.125 \qquad z_2 = -0.020833. \end{array}$$

Nun können die oben berechneten Formeln für eine Fehlerabschätzung allerdings nicht mehr verwendet werden, da diese Werte das Gleichungssystem nicht erfüllen, und somit noch zusätzliche Fehler entstehen. Es können aber die allgemein gültigen Algorithmen zur Fehlerintervallbestimmung verwendet werden. Diese liefern für $x \in [-0.1, 0.1]$ folgende Ergebnisse.

- Algorithmus einDfehlerslope

$$f(x) \in \frac{4.48864495 + 4.3125x + 1.499933x^2 + [-0.012600525, 0.012600525]}{1 + 0.125x - 0.020833x^2}.$$

- Algorithmus einDFehlerzerteilung, mit Unterteilung in 10 Boxen und Verwendung von einDfehlerslope

$$f(x) \in \frac{4.49988131 + 4.3125x + 1.499933x^2 + [-1.24548938 \cdot 10^{-4}, 1.24548938 \cdot 10^{-4}]}{1 + 0.125x - 0.020833x^2}.$$

- Algorithmus zweiDfehlerslopemult, mit entsprechenden Nullen

$$f(x) \in \frac{4.5 + 4.3125x + 1.499933x^2 + [-1.4701579 \cdot 10^{-7}, 8.681737 \cdot 10^{-7}]}{1 + 0.125x - 0.020833x^2}.$$

- Algorithmus einDFehlerzerteilung, mit Unterteilung in 10 Boxen und Verwendung von einDfehlermult

$$f(x) \in \frac{4.5 + 4.3125x + 1.499933x^2 + [-2.8687 \cdot 10^{-10}, 8.1167264 \cdot 10^{-7}]}{1 + 0.125x - 0.020833x^2}.$$

Interpolationsmodell für $x \in [-0.1, 0.1]$

Als erstes werden die Punkte gewählt an denen die Funktion ausgewertet werden soll. Eine Möglichkeit ist zum Beispiel die Folgende

$$M = \{0, 0.1, -0.1, 0.0707, -0.0707\}.$$

Die Elemente in M entsprechen in etwa den Tschebyschewpunkten.

Wertet man $f(x)$ an diesen Punkten aus so erhält man

$$\begin{aligned} f(0) &= 4.5 \\ f(0.1) &= 4.88619 \\ f(-0.1) &= 4.13632 \\ f(0.0707) &= 4.77073 \\ f(-0.0707) &= 4.24051. \end{aligned}$$

Es muss also das Minimierungsproblem

$$\begin{aligned} \min \quad & \sum_{x\in M} (f(x)q(x) - p(x))^2 \\ s.t. \quad & q(x) > 0 \qquad \forall x \in M \end{aligned}$$

gelöst werden.

Mittels des in Bemerkung 7.4. beschriebenen Matlab Paketes CVX erhält man für $p(x)$ und $q(x)$

$$\begin{aligned} p(x) &= 4.500000004124307 + 4.316597642516412x + 1.502371053503959x^2 \\ q(x) &= 1 + 0.125910468594745x - 0.021064499787566x^2. \end{aligned}$$

Seien $p_1(x) = 1$, $p_2(x) = (x+2)^2$, $q_1(x) = x+2$, $q_2(x) = 1$. Es folgt

$$f(x) = \frac{1}{x+2} + (x+2)^2 = \frac{p_1(x)}{q_1(x)} + \frac{p_2(x)}{q_2(x)}.$$

Fehlerabschätzung:

$$\begin{aligned} f(x)q(x) - p(x) = &\left(\frac{p_1}{q_1} + \frac{p_2}{q_2}\right) q(x) - p(x) = \\ =&\frac{p_1(x)q_2(x)q(x) + p_2(x)q_1(x)q(x) - p(x)q_1(x)q_2(x)}{q_1(x)q_2(x)} = \\ =&(-4.124307118047454 \cdot 10^{-9} - 5.359022133832880 \cdot 10^{-7}x + \\ &+2.687762257910786 \cdot 10^{-6}x^2 + 1.588803068595379 \cdot 10^{-4} \cdot x^3 - \\ &-2.382650653254964 \cdot 10^{-4} \cdot x^4 - 0.010532249893783 \cdot x^5)/\left(1 + \frac{1}{2}x\right) \in \\ &\in [-3.639409 \cdot 10^{-7}, 3.584699 \cdot 10^{-7}] \quad \forall x \in \mathbf{x} \end{aligned}$$

Bemerkung 9.1. *Exakt das gleiche Ergebnis liefert Algorithmus zweiDfehlerslopemult. Das ist deshalb der Fall, weil einerseits die Fehlerabschätzung im Algorithmus genauso gerechnet wird wie soeben, und andererseits weil die Berechnung des Fehlerintervalls mittels Slopes, bei der Intervallbreite von* $\mathbf{x}$ *kein engeres Fehlerintervall liefert.*

Verbessern lässt sich diese Abschätzung noch indem man das Intervall $\mathbf{x}$ in Teilintervalle aufsplittet. Unterteilt man den Definitionsbereich in zehn äquidistante Intervalle, so erhält man

$$\begin{aligned} \mathbf{x} = [-0.1, 0.1] = &[-0.1, -0.08] \cup [-0.08, -0.06] \cup [-0.06, -0.04] \cup [-0.04, -0.02] \cup \\ &\cup [-0.02, 0] \cup [0, 0.02] \cup [0.02, 0.04] \cup [0.04, 0.06] \cup [0.06, 0.08] \cup [0.08, 0.1]. \end{aligned}$$

Daraus ergibt sich das Fehlerintervall

$$\begin{aligned} f(x)q(x) - p(x) = (&-4.124307118047454 \cdot 10^{-9} - 5.359022133832880 \cdot 10^{-7}x + \\ &+ 2.687762257910786 \cdot 10^{-6}x^2 + 1.588803068595379 \cdot 10^{-4} \cdot x^3 - \\ &- 2.382650653254964 \cdot 10^{-4} \cdot x^4 - 0.010532249893783 \cdot x^5)/(1 + \frac{1}{2}x) \in \end{aligned}$$

$$\in \begin{cases} [-9.71002 \cdot 10^{-8}, 9.53263 \cdot 10^{-8}] & \forall x \in [-0.1, -0.08] \\ [-4.70942 \cdot 10^{-8}, 5.52663 \cdot 10^{-8}] & \forall x \in [-0.08, -0.06] \\ [-1.51705 \cdot 10^{-8}, 3.62035 \cdot 10^{-8}] & \forall x \in [-0.06, -0.04] \\ [-0.31386 \cdot 10^{-8}, 2.18179 \cdot 10^{-8}] & \forall x \in [-0.04, -0.02] \\ [-0.548836 \cdot 10^{-8}, -0.548836 \cdot 10^{-8}] & \forall x \in [-0.02, 0] \\ [-1.49142 \cdot 10^{-8}, -0.17606 \cdot 10^{-8}] & \forall x \in [0, 0.02] \\ [-2.46562 \cdot 10^{-8}, -0.04367 \cdot 10^{-8}] & \forall x \in [0.02, 0.04] \\ [-3.24387 \cdot 10^{-8}, 1.64169 \cdot 10^{-8}] & \forall x \in [0.04, 0.06] \\ [-4.58970 \cdot 10^{-8}, 4.95070 \cdot 10^{-8}] & \forall x \in [0.06, 0.08] \\ [-8.49185 \cdot 10^{-8}, 9.08558 \cdot 10^{-8}] & \forall x \in [0.08, 0.1]. \end{cases}$$

Abschließend müssen diese Intervalle noch vereinigt werden, um ein Fehlerintervall für das gegebene Problem zu erhalten.

$$\begin{aligned} f(x)q(x) - p(x) \in &[-9.71002 \cdot 10^{-8}, 9.53263 \cdot 10^{-8}] \cup \\ &\cup [-4.70942 \cdot 10^{-8}, 5.52663 \cdot 10^{-8}] \cup \\ &\cup [-1.51705 \cdot 10^{-8}, 3.62035 \cdot 10^{-8}] \cup \\ &\cup [-0.31386 \cdot 10^{-8}, 2.18179 \cdot 10^{-8}] \cup \\ &\cup [-0.548836 \cdot 10^{-8}, -0.548836 \cdot 10^{-8}] \cup \\ &\cup [-1.49142 \cdot 10^{-8}, -0.17606 \cdot 10^{-8}] \cup \\ &\cup [-2.46562 \cdot 10^{-8}, -0.04367 \cdot 10^{-8}] \cup \\ &\cup [-3.24387 \cdot 10^{-8}, 1.64169 \cdot 10^{-8}] \cup \\ &\cup [-4.58970 \cdot 10^{-8}, 4.95070 \cdot 10^{-8}] \cup \\ &\cup [-8.49185 \cdot 10^{-8}, 9.08558 \cdot 10^{-8}] = \\ &= [-9.71002 \cdot 10^{-8}, 9.53263 \cdot 10^{-8}] \end{aligned}$$

Bemerkung 9.2. *Das gleiche Ergebnis liefert natürlich auch Algorithmus einDFehlerzerteilung, mit Unterteilung in 10 Boxen und Verwendung von einDfehlermult.*

Bisher wurden vor dem eigentlichen Bestimmen des Fehlerintervalls die Koeffizienten von Polynomen vierter und sechster Ordnung berechnet, sodass

$\frac{p_1(x)q_2(x)q(x)+p_2(x)q_1(x)q(x)-p(x)q_1(x)q_2(x)}{q_1(x)q_2(x)}$ einfacher abgeschätzt werden kann. Um diese Berechnungen durchzuführen, sind allerdings unzählige Operationen notwendig, wodurch große Rundungsfehler entstehen können. Im Folgenden wird das Fehlerintervall abgeschätzt, ohne vorher diese Terme auszumultiplizieren. Die Berechnungen werden mit dem Algorithmus einDfehlerslope durchgeführt.

Als Ergebnis erhält man

- Algorithmus einDfehlerslope

$$f(x) \in \frac{4.488581994029055 + 4.316597642516412x + 1.502371053503959x^2 + I}{1 + 0.125910468594745x - 0.021064499787566x^2},$$
wobei $I = [-1.2674208171686 \cdot 10^{-2}, 1.2674208171686 \cdot 10^{-2}]$.

- Algorithmus einDFehlerzerteilung, mit Unterteilung in 10 Boxen und Verwendung von einDfehlerslope

$$f(x) \in \frac{4.499880199653598 + 4.316597642516412x + 1.502371053503959x^2 + I}{1 + 0.125910468594745x - 0.021064499787566x^2},$$
wobei $I = [-1.252810021520047 \cdot 10^{-4}, 1.252810021520047 \cdot 10^{-4}]$.

Taylor-Modell für $x \in [-0.01, 0.01]$

Es gilt

$$T_{(x+2)} = ((x+2), [0,0])$$
$$T_{(x+2)^2} = ((x+2)^2, [0,0]).$$

Laut Formel für $1/f$ in der Taylor Arithmetik gilt für $0 < \theta < 1$

$$1/(x+2) = 1/c_f \left\{ 1 - \frac{\bar{f}(x)}{c_f} + \cdots + (-1)^k \frac{(\bar{f}(x))^k}{c_f^k} \right\} + (-1)^{k+1} \frac{(\bar{f}(x)^{k+1}}{c_f^{k+2}} \frac{1}{(1+\theta \bar{f}(x)/c_f)^{k+2}}.$$

Für $k = 2$ erhält man als Fehlerintervall

$$-\frac{x^3}{c_i^4} \frac{1}{(1+\theta x/c_f)^4} \in -\frac{[-10^{-6}, 10^{-6}]}{2^4([0.995, 1.005])^4} = [-6.3766 \cdot 10^{-8}, 6.3766 \cdot 10^{-8}].$$

Daraus folgt

$$T_{1/(x+2)} = (1/2 - 1/4x + 1/8x^2, [-6.3766 \cdot 10^{-8}, 6.3766 \cdot 10^{-8}])$$
$$T_{(x+2)^2} = ((x+2)^2, [0,0]) = (4 + 4x + x^2, [0,0]).$$

Als Taylor-Modell der gegebenen Funktion $f(x)$ im entsprechenden Intervall erhält man also

$$T_{f(x)} = T_{\frac{1}{x+2}+(x+2)^2} = \left(9/2 + 15/4x + 9/8x^2, [-6.3766 \cdot 10^{-8}, 6.3766 \cdot 10^{-8}]\right).$$

Pade-Modell für $x \in [-0.01, 0.01]$

Auch für diesen Definitionsbereich wird wieder folgende Darstellung des Problems verwendet

$$\frac{p_1}{q_1} + \frac{p_2}{q_2} = \frac{1}{2+x} + \frac{4+4x+x^2}{1} = \frac{1/2}{1+1/2x} + \frac{4+4x+x^2}{1}.$$

Für die Koeffizienten ergeben sich wieder die gleichen Werte wie vorher, da diese unabhängig vom zu betrachtenden Intervall berechnet werden.
Wird das Fehlerintervall mittels Korollar 5.4. berechnet, so erhält man

$$f(x) = \frac{1}{2+x} + (x+2)^2 \in \frac{\frac{9}{2} + \frac{69}{16}x + \frac{3}{2}x^2 + [-1.04691 \cdot 10^{-12}, 1.04691 \cdot 10^{-12}]}{1 + \frac{1}{8}x - \frac{1}{48}x^2}.$$

Mit den rundungsfehlerbehafteten Koeffizienten ergeben sich folgende Ergebnisse.

- Algorithmus einDfehlerslope

$$f(x) \in \frac{4.49988177 + 4.3125x + 1.499933x^2 + [-1.194558 \cdot 10^{-4}, 1.194558 \cdot 10^{-4}]}{1 + 0.125x - 0.020833x^2}.$$

- Algorithmus einDFehlerzerteilung, mit Unterteilung in 10 Boxen und Verwendung von einDfehlerslope

$$f(x) \in \frac{4.49999882 + 4.3125x + 1.499933x^2 + [-1.192836 \cdot 10^{-6}, 1.192836 \cdot 10^{-6}]}{1 + 0.125x - 0.020833x^2}.$$

- Algorithmus zweiDfehlerslopemult, mit entsprechenden Nullen

$$f(x) \in \frac{4.500000003 + 4.3125x + 1.499933x^2 + [-3.6726 \cdot 10^{-11}, 6.92116 \cdot 10^{-9}]}{1 + 0.125x - 0.020833x^2}.$$

- Algorithmus einDFehlerzerteilung, mit Unterteilung in 10 Boxen und Verwendung von einDfehlermult

$$f(x) \in \frac{4.500000003 + 4.3125x + 1.499933x^2 + [-2.84285 \cdot 10^{-13}, 6.8673 \cdot 10^{-9}]}{1 + 0.125x - 0.020833x^2}.$$

Interpolationsmodell für $x \in [-0.01, 0.01]$

Zuerst werden die Punkte, an denen ausgewertet werden soll, wieder gewählt wie vorher.

Löst man dieses Problem mittels des unten beschriebenen Algorithmus einDmain (d.h. mit $x \in [-0.1, 0.1]$) so erhält man für $p(x)$ und $q(x)$ wieder

$$p(x) = 4.500000004124307 + 4.316597642516412x + 1.502371053503959x^2$$
$$q(x) = 1 + 0.125910468594745x - 0.021064499787566x^2.$$

Fehlerabschätzung:

Die verschiedenen Algorithmen ergeben folgende Fehlerintervalle.

- Algorithmus einDfehlerslope

$$f(x) \in \frac{4.499881096307417 + 4.316597642516412x + 1.502371053503959x^2 + I}{1 + 0.125910468594745x - 0.021064499787566x^2},$$
wobei $I = [-1.201557188225468 \cdot 10^{-4}, 1.201557188225468 \cdot 10^{-4}]$.

- Algorithmus einDFehlerzerteilung, mit Unterteilung in 10 Boxen und Verwendung von einDfehlerslope

$$f(x) \in \frac{4.499998809436604 + 4.316597642516412x + 1.502371053503959x^2 + I}{1 + 0.125910468594745x - 0.021064499787566x^2},$$
wobei $I = [-1.200254082981723 \cdot 10^{-6}, 1.200254082981723 \cdot 10^{-6}]$.

- Algorithmus zweiDfehlerslopemult, mit entsprechenden Nullen

$$f(x) \in \frac{4.500000000113141 + 4.316597642516412x + 1.502371053503959x^2 + I}{1 + 0.125910468594745x - 0.021064499787566x^2},$$
wobei $I = [-5.682949840906403 \cdot 10^{-9}, 5.682949840906403 \cdot 10^{-9}]$.

- Algorithmus einDFehlerzerteilung, mit Unterteilung in 10 Boxen und Verwendung von einDfehlermult

$$f(x) \in \frac{4.500000000240684 + 4.316597642516412x + 1.502371053503959x^2 + I}{1 + 0.125910468594745x - 0.021064499787566x^2},$$
wobei $I = [-5.3129918999288 \cdot 10^{-9}, 5.3129918999288 \cdot 10^{-9}]$.

Taylor-Modell für $x \in [-10^{-4}, 10^{-4}]$

Analog zu den vorigen Taylor-Modellen ergibt sich

$$f(x) \in 9/2 + 15/4x + 9/8x^2 + [-6.2513 \cdot 10^{-14}, 6.2513 \cdot 10^{-14}] \quad \forall x \in [-10^{-4}, 10^{-4}].$$

Padé-Modell für $x \in [-10^{-4}, 10^{-4}]$

Berechnet man das Padé-Modell auf die gleiche Art und Weise wie oben, so gilt für alle $x \in [-10^{-4}, 10^{-4}]$

$$f(x) = \frac{1}{2+x} + (x+2)^2 \in \frac{\frac{9}{2} + \frac{69}{16}x + \frac{3}{2}x^2 + [-1.04242 \cdot 10^{-22}, 1.04242 \cdot 10^{-22}]}{1 + \frac{1}{8}x - \frac{1}{48}x^2}.$$

Mit den rundungsfehlerbehafteten Koeffizienten ergeben sich, für $x \in [-10^{-4}, 10^{-4}]$, folgende Ergebnisse.

- Algorithmus einDfehlerslope

  $$f(x) \in \frac{4.499999988 + 4.3125x + 1.499933x^2 + [-1.18751 \cdot 10^{-8}, -1.18751 \cdot 10^{-8}]}{1 + 0.125x - 0.020833x^2}.$$

- Algorithmus einDFehlerzerteilung, mit Unterteilung in 10 Boxen und Verwendung von einDfehlerslope

  $$f(x) \in \frac{4.49999999 + 4.3125x + 1.499933x^2 + [-1.190804 \cdot 10^{-10}, 1.190804 \cdot 10^{-10}]}{1 + 0.125x - 0.020833x^2}.$$

- Algorithmus zweiDfehlerslopemult, mit entsprechenden Nullen

  $$f(x) \in \frac{4.500000000 + 4.3125x + 1.499933x^2 + [-3.55019 \cdot 10^{-17}, 6.850698 \cdot 10^{-13}]}{1 + 0.125x - 0.020833x^2}.$$

- Algorithmus einDFehlerzerteilung, mit Unterteilung in 10 Boxen und Verwendung von einDfehlermult

  $$f(x) \in \frac{4.500000000 + 4.3125x + 1.499933x^2 + [-2.84003 \cdot 10^{-19}, 6.850161 \cdot 10^{-13}]}{1 + 0.125x - 0.020833x^2}.$$

Interpolationsmodell für $x \in [-10^{-4}, 10^{-4}]$

Mittels des Algorithmus einDmain erhält man für $p(x)$ und $q(x)$ wieder

$$p(x) = 4.500000004124307 + 4.316597642516412x + 1.502371053503959x^2$$
$$q(x) = 1 + 0.125910468594745x - 0.021064499787566x^2.$$

Fehlerabschätzung:

Die verschiedenen Algorithmen ergeben für $x \in [-10^{-4}, 10^{-4}]$ folgende Fehlerintervalle.

- Algorithmus einDfehlerslope

$$f(x) \in \frac{4.499999988109970 + 4.316597642516412x + 1.502371053503959x^2 + I}{1 + 0.125910468594745x - 0.021064499787566x^2},$$
$$\text{wobei } I = [-1.194344489988154 \cdot 10^{-8}, 1.194344489988154 \cdot 10^{-8}].$$

- Algorithmus einDFehlerzerteilung, mit Unterteilung in 10 Boxen und Verwendung von einDfehlerslope

$$f(x) \in \frac{4.499999999885931 + 4.316597642516412x + 1.502371053503959x^2 + I}{1 + 0.125910468594745x - 0.021064499787566x^2},$$
$$\text{wobei } I = [-1.674841744146316 \cdot 10^{-10}, 1.674841744146316 \cdot 10^{-10}].$$

- Algorithmus zweiDfehlerslopemult, mit entsprechenden Nullen

$$f(x) \in \frac{4.5 + 4.316597642516412x + 1.502371053503959x^2 + I}{1 + 0.125910468594745x - 0.021064499787566x^2},$$
$$\text{wobei } I = [-5.341379102304318 \cdot 10^{-11}, 5.341379102304318 \cdot 10^{-11}].$$

- Algorithmus einDFehlerzerteilung, mit Unterteilung in 10 Boxen und Verwendung von einDfehlermult

$$f(x) \in \frac{4.500000000000024 + 4.316597642516412x + 1.502371053503959x^2 + I}{1 + 0.125910468594745x - 0.021064499787566x^2},$$
$$\text{wobei } I = [-5.343000653104619 \cdot 10^{-11}, 5.343000653104619 \cdot 10^{-11}].$$

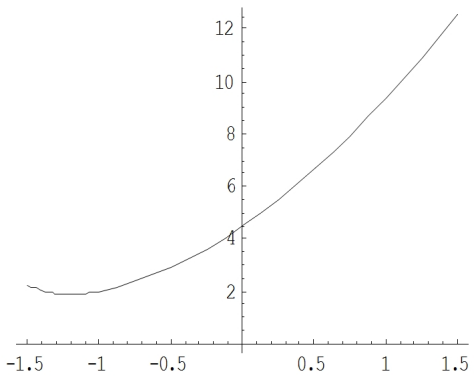

Abb. 3: $f(x) = 1/(2+x) + (x+2)^2$

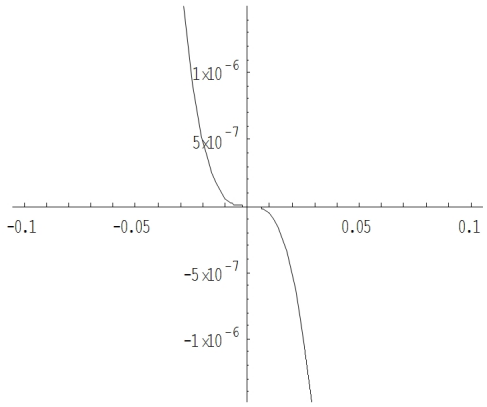

Abb. 4: Fehlerfunktion $f(x)q(x) - p(x)$ für x aus [-0.1,0.1], berechnet mittels Taylor-Modell.

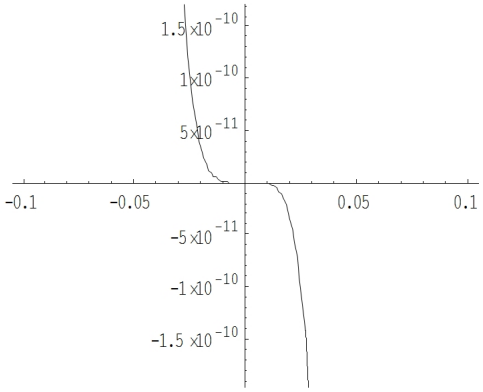

Abb. 5: Fehlerfunktion $f(x)q(x) - p(x)$ für x aus [-0.1,0.1], berechnet mittels Padé-Modell.

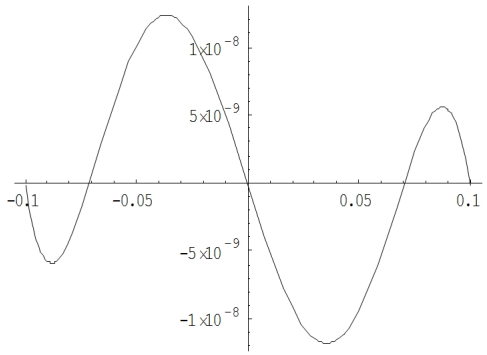

Abb. 6: Fehlerfunktion $f(x)q(x) - p(x)$ für x aus [-0.1,0.1], berechnet mittels Interpolationsmodell.

Vergleich der Lösungen

Stellt man nun die berechneten Lösungen der drei Methoden gegenüber, so bemerkt man in diesem Beispiel einen beachtlichen Unterschied. Plottet man die verschiedenen Ergebnisse zum Beispiel mit Mathematica, so lassen sich, durch Berechnung der Maxima und Minima, leicht Fehlerintervalle mit sehr geringer Überschätzung bestimmen.

Methode	**Fehlerintervallbreite**
Taylor-Modell	$1.25313 \cdot 10^{-4}$
Padé-Modell	$2.08855 \cdot 10^{-7}$
Padé-Modell mit Rundungsfehlern	$7.9344 \cdot 10^{-7}$
Interpolationsmodell	$2.39324 \cdot 10^{-8}$

Tabelle 5: Intervall mit wenig (bzw. keiner) Überschätzung für $x \in [-0.1, 0.1]$

Methode	**Fehlerintervallbreite**
Taylor-Modell	$1.25003 \cdot 10^{-7}$
Padé-Modell	$2.08544 \cdot 10^{-12}$
Padé-Modell mit Rundungsfehlern	$6.8503 \cdot 10^{-9}$
Interpolationsmodell	$1.03641 \cdot 10^{-8}$

Tabelle 6: Intervall mit wenig (bzw. keiner) Überschätzung für $x \in [-0.01, 0.01]$

Methode	**Fehlerintervallbreite**
Taylor-Modell	$1.24345 \cdot 10^{-13}$
Padé-Modell	RF
Padé-Modell mit Rundungsfehlern	$6.84786 \cdot 10^{-13}$
Interpolationsmodell	$1.06768 \cdot 10^{-10}$

Tabelle 7: Intervall mit wenig (bzw. keiner) Überschätzung für $x \in [-10^{-4}, 10^{-4}]$

Bemerkung 9.3. *In den Tabellen steht RF für Rundungsfehler. Denn für diesen Definitionsbereich würde ein berechneter Wert hauptsächlich aus Rundungsfehlern bestehen, und somit nicht mehr interessant für einen Vergleich sein. Die unten angegebene Rechenzeit bezieht sich rein auf die Berechnung des Fehlerintervalls. In den Tabellen auf Seite 95 und in nachfolgender Diskussion werden folgende Bezeichnungen verwendet.*
Alg. a ... Algorithmus einDfehlerslope
Alg. a' ... Algorithmus einDFehlerzerteilung
(mit Unterteilung in 10 Boxen und Verwendung von einDfehlerslope)
Alg. b ... Algorithmus zweiDfehlerslopemult (mit entsprechenden Nullen)
Alg. b' ... Algorithmus einDFehlerzerteilung
(mit Unterteilung in 10 Boxen und Verwendung von einDfehlermult)
Alg. $\tilde{b}$ ... Algorithmus einDfehlermult.

Methode	**Fehlerintervall für** $x \in [-0.1, 0.1]$	**Rechenzeit**
Taylor-Modell	$[-7.67336 \cdot 10^{-5}, 7.67336 \cdot 10^{-5}]$	-
Padé-Modell	$[-1.09649 \cdot 10^{-7}, 1.09649 \cdot 10^{-7}]$	-
Padé-Modell m. RF, Alg. a)	$[-0.0126742081717, 0.0126742081717]$	94ms
Padé-Modell m. RF, Alg. a')	$[-1.24548938 \cdot 10^{-4}, 1.24548938 \cdot 10^{-4}]$	749ms
Padé-Modell m. RF, Alg. b)	$[-1.4701579 \cdot 10^{-7}, 8.681737 \cdot 10^{-7}]$	1950ms
Padé-Modell m. RF, Alg. b')	$[-2.8687 \cdot 10^{-10}, 8.1167264 \cdot 10^{-7}]$	1123ms
Padé-Modell m. RF, Alg. $\tilde{\text{b}}$)	$[-1.4701579 \cdot 10^{-7}, 8.681737 \cdot 10^{-7}]$	109ms
Interpolationsmodell, Alg. a)	$[-0.012600525, 0.012600525]$	94ms
Interpolationsmodell, Alg. a')	$[-1.2528101 \cdot 10^{-4}, 1.2528101 \cdot 10^{-4}]$	733ms
Interpolationsmodell, Alg. b)	$[-3.639409 \cdot 10^{-7}, 3.584699 \cdot 10^{-7}]$	1919ms
Interpolationsmodell, Alg. b')	$[-9.71002 \cdot 10^{-8}, 9.53263 \cdot 10^{-8}]$	1170ms
Interpolationsmodell, Alg. $\tilde{\text{b}}$)	$[-3.639409 \cdot 10^{-7}, 3.584699 \cdot 10^{-7}]$	109ms

Methode	**Fehlerintervall für** $x \in [-0.01, 0.01]$	**Rechenzeit**
Taylor-Modell	$[-6.3766 \cdot 10^{-8}, 6.3766 \cdot 10^{-8}]$	-
Padé-Modell	$[-1.04691 \cdot 10^{-12}, 1.04691 \cdot 10^{-12}]$	-
Padé-Modell m. RF, Alg. a)	$[-1.19455739 \cdot 10^{-4}, 1.19455739 \cdot 10^{-4}]$	94ms
Padé-Modell m. RF, Alg. a')	$[-1.19283535 \cdot 10^{-6}, 1.19283535 \cdot 10^{-6}]$	702ms
Padé-Modell m. RF, Alg. b)	$[-3.672528 \cdot 10^{-11}, 6.9211575 \cdot 10^{-9}]$	1934ms
Padé-Modell m. RF, Alg. b')	$[-2.8428429 \cdot 10^{-13}, 6.867212 \cdot 10^{-9}]$	1139ms
Padé-Modell m. RF, Alg. $\tilde{\text{b}}$)	$[-3.672528 \cdot 10^{-11}, 6.9211575 \cdot 10^{-9}]$	109ms
Interpolationsmodell, Alg. a)	$[-1.2015572 \cdot 10^{-4}, 1.2015572 \cdot 10^{-4}]$	94ms
Interpolationsmodell, Alg. a')	$[-1.2002541 \cdot 10^{-6}, 1.2002541 \cdot 10^{-6}]$	733ms
Interpolationsmodell, Alg. b)	$[-5.6829499 \cdot 10^{-9}, 5.6829499 \cdot 10^{-9}]$	1930ms
Interpolationsmodell, Alg. b')	$[-5.3129919 \cdot 10^{-9}, 5.3129919 \cdot 10^{-9}]$	1108ms
Interpolationsmodell, Alg. $\tilde{\text{b}}$)	$[-5.6829499 \cdot 10^{-9}, 5.6829499 \cdot 10^{-9}]$	109ms

Methode	**Fehlerintervall für** $x \in [-10^{-4}, 10^{-4}]$	**Rechenzeit**
Taylor-Modell	$[-6.2513 \cdot 10^{-14}, 6.2513 \cdot 10^{-14}]$	-
Padé-Modell	$[-1.04242 \cdot 10^{-22}, 1.04242 \cdot 10^{-22}]$	-
Padé-Modell m. RF, Alg. a)	$[-1.1875095 \cdot 10^{-8}, 1.1875095 \cdot 10^{-8}]$	94ms
Padé-Modell m. RF, Alg. a')	$[-1.1908037 \cdot 10^{-10}, 1.1908037 \cdot 10^{-10}]$	718ms
Padé-Modell m. RF, Alg. b)	$[-3.550188 \cdot 10^{-17}, 6.8506976 \cdot 10^{-13}]$	1981ms
Padé-Modell m. RF, Alg. b')	$[-2.840029 \cdot 10^{-19}, 6.8501608 \cdot 10^{-13}]$	1123ms
Padé-Modell m. RF, Alg. $\tilde{\text{b}}$)	$[-3.550188 \cdot 10^{-17}, 6.8506976 \cdot 10^{-13}]$	109ms
Interpolationsmodell, Alg. a)	$[-1.1943445 \cdot 10^{-8}, 1.1943445 \cdot 10^{-8}]$	94ms
Interpolationsmodell, Alg. a')	$[-1.6748418 \cdot 10^{-10}, 1.6748418 \cdot 10^{-10}]$	718ms
Interpolationsmodell, Alg. b)	$[-5.3413792 \cdot 10^{-11}, 5.3413792 \cdot 10^{-11}]$	1950ms
Interpolationsmodell, Alg. b')	$[-5.3430007 \cdot 10^{-11}, 5.3430007 \cdot 10^{-11}]$	1123ms
Interpolationsmodell, Alg. $\tilde{\text{b}}$)	$[-5.38101 \cdot 10^{-11}, 5.38101 \cdot 10^{-11}]$	109ms

Im Intervall $[-0.1, 0.1]$ errechnet sich also, mittels Padé-Modell und dem Interpolationsmodell, eine wesentlich bessere Näherung der Funktion als mit dem Taylor-Modell. Auch die leicht abgeänderten Koeffizienten des Padé-Modells ergeben noch eine wesentlich bessere Approximation als das Taylor-Modell.
Vergleicht man die errechneten Fehlerabschätzungen, so sieht man, dass die Ergebnisse sehr stark von den verwendeten Algorithmen abhängig sind. Klarerweise ergeben die Algorithmen b, $\tilde{\text{b}}$ und b' die besten Ergebnisse bzw. geringste Überschätzung, da bei diesen das Problem vor der eigentlichen Fehlerintervallbestimmung ausmultipliziert wird, und somit einige Koeffizienten sehr klein oder Null sind. Außerdem ist der Definitionsbereich noch zu groß, als dass die Abschätzung mittels Slopes Auswirkungen auf das Ergebnis hätte. Die Fehlerintervalle welche mit Algorithmus b bzw. $\tilde{\text{b}}$ berechnet wurden, sind also kleiner als das des Taylor-Modells.

Vergleicht man allerdings das Padé-Modell mit dem Interpolationsmodell, so stellt man fest, dass das Interpolationsmodell trotz besserer Näherung der Funktion, ein breiteres Fehlerintervall aufweist. Erst durch die zeitlich aufwändige Intervallteilung in Algorithmus b' (verglichen mit Algorithmus $\tilde{\text{b}}$) ergibt sich ein ähnlich breites Fehlerintervall. Der Wert in der Tabelle wurde für zehn Teilintervalle berechnet.

Grund dafür ist, dass die Fehlerfunktion des Padé-Modells eine fünffache Nullstelle hat (im eindimensionalen Fall), und somit die Koeffizienten der niedrigen Potenzen von x wegfallen. Dadurch lässt sich aber das Fehlerintervall besser abschätzen als beim Interpolationsmodell, auch wenn bei diesem der maximale Fehler kleiner ist. Unterteilt man das zu betrachtende Intervall genügend oft, so erhält man beliebig kleine Überschätzungen.

Für die Intervalle $[-0.01, 0.01]$ und $[-10^{-4}, 10^{-4}]$ sieht man beim Taylor-Modell und beim Padé-Modell wieder einen ähnlichen Zusammenhang, wobei das Padé-Modell für kleinere Intervalle im Verhältnis zum Taylor-Modell immer besser wird.

Betrachtet man die Berechnungen des Interpolationsmodells, so sieht man, dass das Taylor-Modell und das Padé-Modell bessere Ergebnisse liefern. Dies ist allerdings wenig verwunderlich, da die mittels CVX berechneten Koeffizienten in den Algorithmen trotz kleinerer Definitionsbereiche weiterhin für das Intervall $[-0.1, 0.1]$ bestimmt werden und nur das Fehlerintervall in Abhängigkeit des Definitionsbereichs berechnet wird. Das liegt daran, dass das CVX-Paket bei sehr kleinen Werten, für das zu lösende Optimierungsproblem nur mehr sehr schlechte Näherungen der Koeffizienten der exakten Lösung ausgibt. Zum Vergleich: Löst man das Optimierungsproblem für $x \in [-10^{-4}, 10^{-4}]$ mittels CVX, so ergibt sich nicht die oben genannte Intervallbreite von $1.06768 \cdot 10^{-10}$ sondern eine Intervallbreite von $5.84403 \cdot 10^{-4}$. Würde man das zu lösende Problem auch für sehr kleine Intervalle exakt lösen, so wäre vermutlich auch weiterhin das Interpolationsmodell die bessere Näherung der zu betrachtenden Funktion.

Bemerkung 9.4. *Auf die angeführten Rechenzeiten wird hier zunächst nicht genauer eingegangen. Erst im Conclusio werden diese miteinander verglichen.*

9.2 Zweidimensionales Beispiel

Aufgabenstellung

Es sollen für folgende Funktion die drei bekannten Modelle berechnet werden.

$$f(x,y) = f(z) = \frac{3+y+2xy}{1+x^2+2xy} - \frac{y^2+xy^2}{1+2xy}$$

Dabei sollen die Definitionsbereiche $[-0.1, 0.1]^2$, $[-0.01, 0.01]^2$, $[-10^{-4}, 10^{-4}]^2$ verwendet werden.

Taylor-Modell für $z \in [-0.1, 0.1]^2$

$$\begin{aligned}
T_{1+x^2+2xy} &= \left(1+x^2+2xy, [0,0]\right) \\
T_{\frac{1}{1+x^2+2xy}} &= \left(1-x^2-2xy, \frac{[0,10^{-4}] + 4[-10^{-4},10^{-4}] + 4[0,10^{-4}]}{(1+[0,1]([0,10^{-2}]+2[-10^{-2},10^{-2}]))^3}\right) = \\
&= \left(1-x^2-2xy, \frac{[-4\cdot 10^{-4}, 6\cdot 10^{-4}]}{(1+[-2\cdot 10^{-2}, 3\cdot 10^{-2}])^3}\right) = \\
&= \left(1-x^2-2xy, [-4.25\cdot 10^{-4}, 6.3749\cdot 10^{-4}]\right)
\end{aligned}$$

$$\begin{aligned}
T_{p_1} &= (3+y+2xy, [0,0]) \\
T_{p_2} &= \left(y^2, [-10^{-3}, 10^{-3}]\right) \\
T_{q_2} &= (1+2xy, [0,0]) \\
T_{\frac{1}{q_2}} &= \left(1-2xy, \frac{[0, 4\cdot 10^{-4}]}{(1+[0,1][-2\cdot 10^{-2}, 2\cdot 10^{-2}])^3}\right) = \\
&= \left(1-2xy, [0, 4.0816\cdot 10^{-4}]\right) \\
T_{\frac{p_1}{q_1}} &= 3+y+2xy-3x^2-6xy+(-x^2y-2x^3y-2xy^2-4x^2y^2)+ \\
&\quad + (1-x^2-2xy)[0,0] + (3+y+2xy)[-4.25\cdot 10^{-4}, 6.3749\cdot 10^{-4}]+ \\
&\quad \in (3+y+2xy-3x^2-6xy) - [-3.2\cdot 10^{-3}, 3.6\cdot 10^{-3}] + [0,0]+ \\
&\quad + [-13.26\cdot 10^{-4}, 19.89\cdot 10^{-4}] = \\
&\quad = \left(3+y+2xy-3x^2-6xy, [-4.926\cdot 10^{-3}, 5.189\cdot 10^{-3}]\right) \\
T_{\frac{p_2}{q_2}} &= (y^2 + [-2\cdot 10^{-4}, 2\cdot 10^{-4}] + [-1.02\cdot 10^{-3}, 1.02\cdot 10^{-3}]+ \\
&\quad + [0, 4.0816\cdot 10^{-6}] + [-4.0816\cdot 10^{-7}, 4.0816\cdot 10^{-7}]) = \\
&\quad = (y^2, [-1.22\cdot 10^{-3}, 1.22\cdot 10^{-3}])
\end{aligned}$$

Als Taylor-Modell für die Funktion $f(z)$ im Intervall $[-0.1, 0.1]$ erhält man

$$T_{f(x,y)} = \left(3+y-4xy-y^2-3x^2, [-6.146\cdot 10^{-3}, 6.409\cdot 10^{-3}]\right).$$

Padé-Modell für $z \in [-0.1, 0.1]^2$

Um für diese Funktion ein Padé-Modell zu berechnen muss $f(z)$ aufgesplittet werden. Als erstes wird folgender Term behandelt

$$\frac{y^2 + xy^2}{1 + 2xy} = \frac{1 + x}{1 + 2xy} y^2 = (*).$$

Wird nun das entstehende Gleichungssystem (31)–(33) laut Theorie gelöst, so erhält man

$$\begin{array}{lll} z_{10} = -1 & w_{02} = 1 & \\ v_{00} = 1 & v_{11} = -2 & v_{20} = -1. \end{array}$$

Alle anderen Koeffizienten sind gleich Null. Aufgrund des Theorems 6.7. folgt daraus

$$(*) = \frac{y^2 - 2xy^3 - x^2y^2 + (y^2)/(1 + 2xy) \cdot (4x^2y^2 + 2x^3y)}{1 - x}.$$

Im zweiten Schritt wird folgendes Problem betrachtet

$$\frac{p_1(x)}{q_1(x)} + \frac{p_2(x)}{q_2(x)} = \frac{3 + y + 2xy + [0, 0]}{1 + x^2 + 2xy} + \frac{-y^2 - [-3.0205 \cdot 10^{-4}, 2.0613 \cdot 10^{-4}]}{1 - x}.$$

Löst man nun das daraus resultierende Gleichungssystem (26)–(28), so ergibt sich

$$\begin{array}{lll} z_{11} = 3 & z_{20} = 1 & \\ v_{00} = 3 & v_{01} = 1 & v_{11} = 5 \\ v_{12} = 1 & w_{12} = -1 & w_{02} = -1. \end{array}$$

Sei $\tilde{p_3} = \left(\frac{p_1}{q_1} + \frac{p_2}{q_2}\right) q_3$, dann gilt

$$\begin{aligned} \tilde{p_3} \subseteq & (v_{00} + w_{00}) + (v_{10} + w_{10})x + (v_{01} + w_{01})y + \\ & + (v_{11} + w_{11})xy + (v_{20} + w_{20})x^2 + (v_{02} + w_{02})y^2 + \\ & + \frac{(1 + z_{10}x + z_{01}y + z_{11}xy + z_{20}x^2 + z_{02}y^2)(\mathbf{a}_{00} - a_{00})}{1 + b_{10}x + b_{01}y + b_{11}xy + b_{20}x^2 + b_{02}y^2} + \\ & + \frac{\left(\sum_{|(i,j)|=4}^{5} (q_3p_1 - q_1r_1)_{ij} x^i y^j\right)}{1 + b_{10}x + b_{01}y + b_{11}xy + b_{20}x^2 + b_{02}y^2} + \\ & + \frac{(\mathbf{c}_{00} - c_{00})(1 + z_{10}x + z_{01}y + z_{11}xy + z_{20}x^2 + z_{02}y^2)}{1 + d_{10}x + d_{01}y + d_{11}xy + d_{20}x^2 + d_{02}y^2} + \\ & + \frac{\left(\sum_{|(i,j)|=4}^{5} (q_3p_2 - q_2r_2)_{ij} x^i y^j\right)}{1 + d_{10}x + d_{01}y + d_{11}xy + d_{20}x^2 + d_{02}y^2}. \end{aligned}$$

Setzt man die berechneten Koeffizienten in diese Formel ein, so folgt

$$\begin{aligned}\tilde{p_3} &\subseteq \left(3+y+5xy-y^2\right)+\frac{-5[-10^{-4},10^{-4}]-10[0,10^{-4}]-3[-10^{-5},10^{-5}]}{1+x^2+2xy}+\\ &+\frac{[0.97,1.04]\cdot[-2.0613\cdot10^{-4},3.0205\cdot10^{-4}]-3[-10^{-4},10^{-4}]-2[0,10^{-4}]}{1-x}\\ &\subseteq \left(3+y+5xy-y^2\right)+\underbrace{[-2.354975\cdot10^{-3},1.2231853\cdot10^{-3}]}_{=:I}.\end{aligned}$$

Laut Korollar 6.4. kann man $p_3(x,y)$ also wählen als

$$p_3(x,y)=3+y+5xy-y^2+I.$$

Man erhält folgendes Padé-Modell für die Funktion $f(x,y)$

$$f(x,y)\subseteq\frac{(3+I)+y+5xy-y^2}{1+3xy+x^2}\quad\forall(x,y)\in[-0.1,0.1]^2.$$

Nun wird noch versucht, den Fehler durch die oben berechneten Faktoren zu verringern. Es gilt

$$\frac{p_1}{q_1}+\frac{p_2}{q_2}=\frac{3+y+2xy+[0,0]}{1+x^2+2xy}-\frac{y^2-2xy^3-x^2y^2+(y^2)/(1+2xy)\cdot(4x^2y^2+2x^3y)}{1-x}.$$

Nun wird folgende Gleichung eingesetzt

$$2xy^3=2x(k_1\cdot y)+2x(y^3-k_1\cdot y)=2x(k_1\cdot y)+2x\cdot\gamma_3\cdot y^3.$$

Es gilt, wie oben berechnet

$$k_1=0.0165104,\quad \gamma_3=0.81656.$$

Es wird also folgende Funktion betrachtet

$$\frac{p_1}{q_1}+\frac{p_2}{q_2}=\frac{3+y+2xy+[0,0]}{1+x^2+2xy}+\frac{-y^2+0.0330208\cdot xy-[-2.65353\cdot10^{-4},1.69435\cdot10^{-4}]}{1-x}.$$

Wird das entstehende Gleichungssystem (26)–(28) gelöst, wie oben beschrieben, so erhält man als Ergebnis die Werte

$z_{01}=0.0110069$	$z_{11}=3.04366$	$z_{20}=1$	$z_{02}=0.0110069$
$v_{01}=1.03302$	$v_{11}=5.13099$	$v_{02}=0.044027$	$v_{12}=0.999637$
$v_{21}=-0.0330208$	$v_{03}=0.0110069$	$v_{00}=3$	$w_{02}=-1$
$w_{11}=0.0330208$	$w_{21}=0.0330208$	$w_{12}=-0.999637$	$w_{03}=-0.0110069.$

Sei wieder $\tilde{p_3} = \left(\frac{p_1}{q_1} + \frac{p_2}{q_2}\right) q_3$.

$$\begin{aligned}\tilde{p_3} \subseteq &(v_{00} + w_{00}) + (v_{10} + w_{10})x + (v_{01} + w_{01})y + \\ &+(v_{11} + w_{11})xy + (v_{20} + w_{20})x^2 + (v_{02} + w_{02})y^2 + \\ &+\frac{\left(1 + \sum_{|(i,j)|=1}^{2} z_{ij}x^i y^j\right)(\mathbf{a}_{00} - a_{00}) + \left(\sum_{|(i,j)|=4}^{5}(q_3p_1 - q_1r_1)_{ij}x^i y^j\right)}{1 + b_{10}x + b_{01}y + b_{11}xy + b_{20}x^2 + b_{02}y^2} + \\ &+\frac{(\mathbf{c}_{00} - c_{00})\left(1 + \sum_{|(i,j)|=1}^{2} z_{ij}x^i y^j\right) + \left(\sum_{|(i,j)|=4}^{5}(q_3p_2 - q_2r_2)_{ij}x^i y^j\right)}{1 + d_{10}x + d_{01}y + d_{11}xy + d_{20}x^2 + d_{02}y^2}.\end{aligned}$$

Durch Einsetzen der berechneten Koeffizienten folgt

$$\begin{aligned}\tilde{p_3} \subseteq &\left(3 + 1.03302y + 5.1640108xy - 0.955973y^2\right) + \frac{[-7.715611 \cdot 10^{-4}, 3.496922 \cdot 10^{-4}]}{1 + x^2 + 2xy} + \\ &+\frac{[0.96846571, 1.04164436] \cdot [-1.69435 \cdot 10^{-4}, 2.65353 \cdot 10^{-4}]}{1 - x} + \\ &+\frac{[-3.0660373 \cdot 10^{-4}, 3.1551715 \cdot 10^{-4}]}{1 - x} \in \\ &\subseteq \left(3 + 1.03302y + 5.1640108xy - 0.9559723y^2\right) + \underbrace{[-1.33608 \cdot 10^{-3}, 1.014519 \cdot 10^{-3}]}_{=:I}.\end{aligned}$$

Mittels Theorem 6.3. erhält man folgendes Padé-Modell

$$f(x,y) \subseteq \frac{(3 + I) + 1.03302y + 5.1640108xy - 0.9559723y^2}{1 + 0.0110069y + 3.04366xy + x^2 + 0.0110069y^2} \quad \forall (x,y) \in [-0.1, 0.1]^2.$$

Interpolationsmodell für $z \in [-0.1, 0.1]^2$

Als erstes muss wieder entschieden werden, an welchen Punkten die zu behandelnde Funktion ausgewertet werden soll. Es werden dazu die folgenden Punkte M gewählt

$$\begin{aligned}M = \Big\{&\begin{pmatrix}0\\0\end{pmatrix}, \begin{pmatrix}0.1\\0\end{pmatrix}, \begin{pmatrix}-0.1\\0\end{pmatrix}, \begin{pmatrix}0\\0.1\end{pmatrix}, \begin{pmatrix}0\\-0.1\end{pmatrix}, \\ &\begin{pmatrix}0.05\\0\end{pmatrix}, \begin{pmatrix}-0.05\\0\end{pmatrix}, \begin{pmatrix}0\\0.05\end{pmatrix}, \begin{pmatrix}0\\-0.05\end{pmatrix}, \\ &\begin{pmatrix}0.1\\0.1\end{pmatrix}, \begin{pmatrix}-0.1\\0.1\end{pmatrix}, \begin{pmatrix}-0.1\\-0.1\end{pmatrix}, \begin{pmatrix}0.1\\-0.1\end{pmatrix}\Big\}.\end{aligned}$$

Da im Zähler des zweiten Terms von f allerdings kein Polynom vom Grad zwei steht, müssen zwei Minimierungsprobleme gelöst werden. Die Funktion wird folgendermaßen berechnet.

Als erstes wird das Problem

$$\frac{\tilde{p}+\tilde{I}}{\tilde{q}} = (-y^2) \cdot \frac{1+x}{1+2xy}$$

gelöst.
Danach wird mittels der Lösung des ersten Minimierungsproblems Folgendes berechnet

$$\frac{p+I}{q} = \frac{3+y+2xy}{1+x^2+2xy} + \frac{\tilde{p}+\tilde{I}}{\tilde{q}}.$$

Die Ergebnisse sind nun abhängig davon, welcher Algorithmus für die Berechnung des Fehlerintervalls verwendet wurde. Hier wurde für die beiden Rechenschritte jeweils derselbe Algorithmus verwendet.

Berechnete Koeffizienten und Fehlerabschätzung:

Sei $(x, y) \in [-0.1, 0.1]^2$. Dann liegt $f(x, y)$ in

- Algorithmus zweiDfehlerslope

  $$\frac{3.000368 + 0.008483x + 0.314507y - 1.938879x^2 + 2.342033xy - 1.930559y^2 + I}{1 + 0.002853x - 0.228481y + 0.346906x^2 + 2.100551xy - 0.235395y^2},$$
  wobei $I = [-4.9615645299733 \cdot 10^{-2}, 4.9615645299733 \cdot 10^{-2}]$.

- Algorithmus zweiDFehlerzerteilung, mit Unterteilung in 5×5 Boxen und Verwendung von zweiDfehlerslope

  $$\frac{2.999874 + 0.008483x + 0.314507y - 1.938879x^2 + 2.342033xy - 1.930559y^2 + I}{1 + 0.002853x - 0.228481y + 0.346906x^2 + 2.100551xy - 0.235395y^2},$$
  wobei $I = [-2.908056797007 \cdot 10^{-3}, 2.908056797007 \cdot 10^{-3}]$.

- Algorithmus zweiDfehlerslopemult

  $$\frac{2.999989 + 0.008483x + 0.314507y - 1.938879x^2 + 2.342033xy - 1.930559y^2 + I}{1 + 0.002853x - 0.228481y + 0.346906x^2 + 2.100551xy - 0.235395y^2},$$
  wobei $I = [-1.510315016387 \cdot 10^{-3}, 1.510315016387 \cdot 10^{-3}]$.

- Algorithmus zweiDFehlerzerteilung, mit Unterteilung in 5×5 Boxen und Verwendung von zweiDfehlermult

  $$\frac{2.999974 + 0.008483x + 0.314507y - 1.938879x^2 + 2.342033xy - 1.930559y^2 + I}{1 + 0.002853x - 0.228481y + 0.346906x^2 + 2.100551xy - 0.235395y^2},$$
  wobei $I = [-7.128674162166237 \cdot 10^{-4}, 7.128674162166237 \cdot 10^{-4}]$.

Taylor-Modell für $z \in [-0.01, 0.01]^2$

$$
\begin{aligned}
T_{1+x^2+2xy} &= \left(1 + x^2 + 2xy, [0,0]\right) \\
T_{\frac{1}{1+x^2+2xy}} &= \left(1 - x^2 - 2xy, \frac{[0,10^{-8}] + 4[-10^{-8},10^{-8}] + 4[0,10^{-8}]}{(1+[0,1]([0,10^{-4}] + 2[-10^{-4},10^{-4}]))^3}\right) = \\
&= \left(1 - x^2 - 2xy, \frac{[-4\cdot 10^{-8}, 6\cdot 10^{-8}]}{(1+[-2\cdot 10^{-4}, 3\cdot 10^{-4}])^3}\right) = \\
&= \left(1 - x^2 - 2xy, [-4.002402\cdot 10^{-8}, 6.003602\cdot 10^{-8}]\right)
\end{aligned}
$$

$$
\begin{aligned}
T_{p_1} &= (3 + y + 2xy, [0,0]) \\
T_{p_2} &= \left(y^2, [-10^{-6}, 10^{-6}]\right) \\
T_{q_2} &= (1 + 2xy, [0,0]) \\
T_{\frac{1}{q_2}} &= \left(1 - 2xy, \frac{[0, 4\cdot 10^{-8}]}{(1+[0,1][-2\cdot 10^{-4}, 2\cdot 10^{-4}])^3}\right) = \\
&= \left(1 - 2xy, [0, 4.002401\cdot 10^{-8}]\right) \\
T_{\frac{p_1}{q_1}} &= 3 + y + 2xy - 3x^2 - 6xy + (-x^2y - 2x^3y - 2xy^2 - 4x^2y^2) + \\
&\quad + (1 - x^2 - 2xy)[0,0] + (3 + y + 2xy)[-4.002402\cdot 10^{-8}, 6.003602\cdot 10^{-8}] + \\
&\quad \in (3 + y + 2xy - 3x^2 - 6xy) - [-3.02\cdot 10^{-6}, 3.06\cdot 10^{-6}] + [0,0] + \\
&\quad + [-1.204803\cdot 10^{-7}, 1.807205\cdot 10^{-7}] = \\
&\quad = \left(3 + y + 2xy - 3x^2 - 6xy, [-3.18049\cdot 10^{-6}, 3.20073\cdot 10^{-6}]\right) \\
T_{\frac{p_2}{q_2}} &= (y^2, [-2\cdot 10^{-8}, 2\cdot 10^{-8}] + [-1.0002\cdot 10^{-6}, 1.0002\cdot 10^{-6}] + \\
&\quad + [0, 4.002401\cdot 10^{-8}] + [-4.002401\cdot 10^{-14}, 4.002401\cdot 10^{-14}]) = \\
&\quad = (y^2, [-1.02021\cdot 10^{-6}, 1.06023\cdot 10^{-6}])
\end{aligned}
$$

Man erhält als Taylor-Modell

$$T_{f(x)} = \left(3 + y - 4xy - y^2 - 3x^2, [-4.2007\cdot 10^{-6}, 4.26096\cdot 10^{-6}]\right).$$

Pade-Modell für $\mathbf{z} \in [-0.01, 0.01]^2$

Analog zu vorher erhält man nach Berechnung des ersten Termes

$$\frac{p_1}{q_1} + \frac{p_2}{q_2} = \frac{3 + y + 2xy + [0,0]}{1 + x^2 + 2xy} + \frac{-y^2 - [-3.0003\cdot 10^{-8}, 2.0007\cdot 10^{-8}]}{1 - x}.$$

Wird nun das entstehende Gleichungssystem (26)–(28) gelöst, so erhält man

$$
\begin{array}{lll}
z_{11} = 3 & z_{20} = 1 & \\
v_{00} = 3 & v_{01} = 1 & v_{11} = 5 \\
v_{12} = 1 & w_{12} = -1 & w_{02} = -1.
\end{array}
$$

Sei wieder $\tilde{p_3} = \left(\frac{p_1}{q_1} + \frac{p_2}{q_2}\right) q_3$.

$$\begin{aligned}
\tilde{p_3} \subseteq & (v_{00} + w_{00}) + (v_{10} + w_{10})x + (v_{01} + w_{01})y \\
& + (v_{11} + w_{11})xy + (v_{20} + w_{20})x^2 + (v_{02} + w_{02})y^2 + \\
& + \frac{\left(1 + \sum_{|(i,j)|=1}^{2} z_{ij} x^i y^j\right)(\mathbf{a}_{00} - a_{00}) + \left(\sum_{|(i,j)|=4}^{5} (q_3 p_1 - q_1 r_1)_{ij} x^i y^j\right)}{1 + b_{10}x + b_{01}y + b_{11}xy + b_{20}x^2 + b_{02}y^2} + \\
& + \frac{(\mathbf{c}_{00} - c_{00})\left(1 + \sum_{|(i,j)|=1}^{2} z_{ij} x^i y^j\right) + \left(\sum_{|(i,j)|=4}^{5} (q_3 p_2 - q_2 r_2)_{ij} x^i y^j\right)}{1 + d_{10}x + d_{01}y + d_{11}xy + d_{20}x^2 + d_{02}y^2}.
\end{aligned}$$

Setzt man die berechneten Koeffizienten in diese Formel ein, so erhält man

$$\begin{aligned}
\tilde{p_3} &\subseteq \left(3 + y + 5xy - y^2\right) + \frac{-5[-10^{-8}, 10^{-8}] - 10[0, 10^{-8}] - 3[-10^{-10}, 10^{-10}]}{1 + x^2 + 2xy} + \\
&+ \frac{[0.97, 1.04] \cdot [-2.0007 \cdot 10^{-8}, 3.0003 \cdot 10^{-8}] - 3[-10^{-8}, 10^{-8}] - 2[0, 10^{-8}]}{1 - x} \\
&\subseteq \left(3 + y + 5xy - y^2\right) + \underbrace{[-2.218526 \cdot 10^{-7}, 1.121314 \cdot 10^{-7}]}_{=:I}.
\end{aligned}$$

Laut Theorem 6.3. ergibt sich folgendes Padé-Modell

$$f(x, y) \subseteq \frac{(3 + I) + y + 5xy - y^2}{1 + 3xy + x^2} \quad \forall (x, y) \in [-0.01, 0.01]^2.$$

Interpolationsmodell für $z \in [-0.01, 0.01]^2$

Zum Auswerten werden wieder folgende Punkte verwendet

$$\begin{aligned}
M = \Big\{ & \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0.1 \\ 0 \end{pmatrix}, \begin{pmatrix} -0.1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 0.1 \end{pmatrix}, \begin{pmatrix} 0 \\ -0.1 \end{pmatrix}, \\
& \begin{pmatrix} 0.05 \\ 0 \end{pmatrix}, \begin{pmatrix} -0.05 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 0.05 \end{pmatrix}, \begin{pmatrix} 0 \\ -0.05 \end{pmatrix}, \\
& \begin{pmatrix} 0.1 \\ 0.1 \end{pmatrix}, \begin{pmatrix} -0.1 \\ 0.1 \end{pmatrix}, \begin{pmatrix} -0.1 \\ -0.1 \end{pmatrix}, \begin{pmatrix} 0.1 \\ -0.1 \end{pmatrix} \Big\}.
\end{aligned}$$

Analog zu vorher wird mittels verschiedener Algorithmen das Problem gelöst. Es ergibt sich folgendes Ergebnis.

Sei $(x, y) \in [-0.01, 0.01]^2$. Dann liegt $f(x, y)$ in

- Algorithmus zweiDfehlerslope

  $$\frac{3.000000 + 0.008483x + 0.314507y - 1.938879x^2 + 2.342033xy - 1.930559y^2 + I}{1 + 0.002853x - 0.228481y + 0.346906x^2 + 2.100551xy - 0.235395y^2},$$

  wobei $I = [-1.350068763984735 \cdot 10^{-4}, 1.350068763984735 \cdot 10^{-4}]$.

- Algorithmus zweiDFehlerzerteilung, mit Unterteilung in 5×5 Boxen und Verwendung von zweiDfehlerslope

$$\frac{2.999998 + 0.008483x + 0.314507y - 1.938879x^2 + 2.342033xy - 1.930559y^2 + I}{1 + 0.002853x - 0.228481y + 0.346906x^2 + 2.100551xy - 0.235395y^2},$$
wobei $I = [-9.702315047545072 \cdot 10^{-6}, 9.702315047545072 \cdot 10^{-6}]$.

- Algorithmus zweiDfehlerslopemult

$$\frac{2.999999 + 0.008483x + 0.314507y - 1.938879x^2 + 2.342033xy - 1.930559y^2 + I}{1 + 0.002853x - 0.228481y + 0.346906x^2 + 2.100551xy - 0.235395y^2},$$
wobei $I = [-6.400157107267510 \cdot 10^{-6}, 6.400157107267510 \cdot 10^{-6}]$.

- Algorithmus zweiDFehlerzerteilung, mit Unterteilung in 5×5 Boxen und Verwendung von zweiDfehlermult

$$\frac{2.999998 + 0.008483x + 0.314507y - 1.938879x^2 + 2.342033xy - 1.930559y^2 + I}{1 + 0.002853x - 0.228481y + 0.346906x^2 + 2.100551xy - 0.235395y^2},$$
wobei $I = [-5.454859516508549 \cdot 10^{-6}, 5.454859516508549 \cdot 10^{-6}]$.

Taylor-Modell für $z \in [-10^{-4}, 10^{-4}]^2$

$$\begin{aligned}
T_{1+x^2+2xy} &= \left(1 + x^2 + 2xy, [0,0]\right) \\
T_{\frac{1}{1+x^2+2xy}} &= \left(1 - x^2 - 2xy, \frac{[0, 10^{-16}] + 4[-10^{-16}, 10^{-16}] + 4[0, 10^{-16}]}{(1 + [0,1]([0, 10^{-8}] + 2[-10^{-8}, 10^{-8}]))^3}\right) = \\
&= \left(1 - x^2 - 2xy, \frac{[-4 \cdot 10^{-16}, 6 \cdot 10^{-16}]}{(1 + [-2 \cdot 10^{-8}, 3 \cdot 10^{-8}])^3}\right) = \\
&= \left(1 - x^2 - 2xy, [-4.00000025 \cdot 10^{-16}, 6.00000037 \cdot 10^{-16}]\right)
\end{aligned}$$

$$\begin{aligned}
T_{p_1} &= (3 + y + 2xy, [0,0]) \\
T_{p_2} &= \left(y^2, [-10^{-12}, 10^{-12}]\right) \\
T_{q_2} &= (1 + 2xy, [0,0]) \\
T_{\frac{1}{q_2}} &= \left(1 - 2xy, \frac{[0, 4 \cdot 10^{-16}]}{(1 + [0,1][-2 \cdot 10^{-8}, 2 \cdot 10^{-8}])^3}\right) = \\
&= \left(1 - 2xy, [0, 4.00000025 \cdot 10^{-16}]\right) \\
T_{\frac{p_1}{q_1}} &= 3 + y + 2xy - 3x^2 - 6xy + (-x^2y - 2x^3y - 2xy^2 - 4x^2y^2) + \\
&\quad + (1 - x^2 - 2xy)[0,0] + (3 + y + 2xy)[-4.00000025, 6.00000037] \cdot 10^{-16} + \\
&\quad \in (3 + y + 2xy - 3x^2 - 6xy) - [-3.0002 \cdot 10^{-12}, 3.0006 \cdot 10^{-12}] + [0,0] + \\
&\quad + [-1.200041 \cdot 10^{-15}, 1.8000601 \cdot 10^{-15}] =
\end{aligned}$$

$$
\begin{aligned}
&= \left(3 + y + 2xy - 3x^2 - 6xy, [-3.0019 \cdot 10^{-12}, 3.0021 \cdot 10^{-12}]\right) \\
T_{\frac{p_2}{q_2}} &= (y^2, [-2 * 10^{-16}, 2 * 10^{-16}] + [-1.00000002 \cdot 10^{-12}, 1.00000002 \cdot 10^{-12}] + \\
&+ [0, 4.00000025 \cdot 10^{-16}] + [-4.00000025 \cdot 10^{-28}, 4.00000025 \cdot 10^{-28}]) = \\
&= (y^2, [-1.0003 \cdot 10^{-12}, 1.0007 \cdot 10^{-12}])
\end{aligned}
$$

Als Taylor-Modell errechnet sich also

$$
T_{f(x)} = \left(3 + y - 4xy - y^2 - 3x^2, [-4.0022 \cdot 10^{-12}, 4.0028 \cdot 10^{-12}]\right).
$$

Pade-Modell für $\mathbf{z} \in [-10^{-4}, 10^{-4}]^2$

Analog zu vorher erhält man nach Berechnung des ersten Termes

$$
\frac{p_1}{q_1} + \frac{p_2}{q_2} = \frac{3 + y + 2xy + [0, 0]}{1 + x^2 + 2xy} + \frac{-y^2 - [-3.00000003 \cdot 10^{-16}, 2.00000007 \cdot 10^{-16}]}{1 - x}.
$$

Löst man nun das entstehende Gleichungssystem so erhält man die Koeffizienten

$$
\begin{array}{lll}
z_{11} = 3 & z_{20} = 1 & \\
v_{00} = 3 & v_{01} = 1 & v_{11} = 5 \\
v_{12} = 1 & w_{12} = -1 & w_{02} = -1.
\end{array}
$$

Für $\tilde{p_3} = \left(\frac{p_1}{q_1} + \frac{p_2}{q_2}\right) q_3$ gilt

$$
\begin{aligned}
\tilde{p_3} \subseteq &(v_{00} + w_{00}) + (v_{10} + w_{10})x + (v_{01} + w_{01})y + \\
&+ (v_{11} + w_{11})xy + (v_{20} + w_{20})x^2 + (v_{02} + w_{02})y^2 + \\
&+ \frac{\left(1 + \sum_{|(i,j)|=1}^{2} z_{ij} x^i y^j\right)(\mathbf{a}_{00} - a_{00}) + \left(\sum_{|(i,j)|=4}^{5} (q_3 p_1 - q_1 r_1)_{ij} x^i y^j\right)}{1 + b_{10}x + b_{01}y + b_{11}xy + b_{20}x^2 + b_{02}y^2} + \\
&+ \frac{(\mathbf{c}_{00} - c_{00})\left(1 + \sum_{|(i,j)|=1}^{2} z_{ij} x^i y^j\right) + \left(\sum_{|(i,j)|=4}^{5} (q_3 p_2 - q_2 r_2)_{ij} x^i y^j\right)}{1 + d_{10}x + d_{01}y + d_{11}xy + d_{20}x^2 + d_{02}y^2}.
\end{aligned}
$$

Setzt man nun die errechneten Koeffizienten ein, so ergibt das

$$
\begin{aligned}
\tilde{p_3} &\subseteq \left(3 + y + 5xy - y^2\right) + \frac{-5[-10^{-16}, 10^{-16}] - 10[0, 10^{-16}] - 3[-10^{-20}, 10^{-20}]}{1 + x^2 + 2xy} + \\
&+ \frac{[0.97, 1.04] \cdot [-2.00000007, 3.00000003] \cdot 10^{-16} - 3[-10^{-16}, 10^{-16}] - 2[0, 10^{-16}]}{1 - x} \\
&\subseteq \left(3 + y + 5xy - y^2\right) + \underbrace{[-2.208101 \cdot 10^{-15}, 1.11209122 \cdot 10^{-15}]}_{=:I}.
\end{aligned}
$$

Für das zu berechnende Modell erhält man also laut Theorem 6.3.

$$
f(x, y) = \frac{(3 + I) + y + 5xy - y^2}{1 + 3xy + x^2} \quad \forall (x, y) \in [-10^{-4}, 10^{-4}]^2.
$$

Interpolationsmodell für $z \in [-10^{-4}, 10^{-4}]^2$

Zum Auswerten werden folgende Punkte M verwendet

$$M = \Big\{ \begin{pmatrix}0\\0\end{pmatrix}, \begin{pmatrix}0.1\\0\end{pmatrix}, \begin{pmatrix}-0.1\\0\end{pmatrix}, \begin{pmatrix}0\\0.1\end{pmatrix}, \begin{pmatrix}0\\-0.1\end{pmatrix}, \\ \begin{pmatrix}0.05\\0\end{pmatrix}, \begin{pmatrix}-0.05\\0\end{pmatrix}, \begin{pmatrix}0\\0.05\end{pmatrix}, \begin{pmatrix}0\\-0.05\end{pmatrix}, \\ \begin{pmatrix}0.1\\0.1\end{pmatrix}, \begin{pmatrix}-0.1\\0.1\end{pmatrix}, \begin{pmatrix}-0.1\\-0.1\end{pmatrix}, \begin{pmatrix}0.1\\-0.1\end{pmatrix} \Big\}.$$

Wie vorher wird mittels verschiedener Algorithmen das Problem gelöst.

Für alle $(x, y) \in [-10^{-4}, 10^{-4}]^2$ liegt $f(x, y)$ in

- Algorithmus zweiDfehlerslope

  $$\frac{3.000000 + 0.008483x + 0.314507y - 1.938879x^2 + 2.342033xy - 1.930559y^2 + I}{1 + 0.002853x - 0.228481y + 0.346906x^2 + 2.100551xy - 0.235395y^2},$$
  wobei $I = [-2.105561482797791 \cdot 10^{-8}, 2.105561482797791 \cdot 10^{-8}]$.

- Algorithmus zweiDFehlerzerteilung, mit Unterteilung in 5×5 Boxen und Verwendung von zweiDfehlerslope

  $$\frac{3.000000 + 0.008483x + 0.314507y - 1.938879x^2 + 2.342033xy - 1.930559y^2 + I}{1 + 0.002853x - 0.228481y + 0.346906x^2 + 2.100551xy - 0.235395y^2},$$
  wobei $I = [-1.321696873455531 \cdot 10^{-8}, 1.321696873455531 \cdot 10^{-8}]$.

- Algorithmus zweiDfehlerslopemult

  $$\frac{3.000000 + 0.008483x + 0.314507y - 1.938879x^2 + 2.342033xy - 1.930559y^2 + I}{1 + 0.002853x - 0.228481y + 0.346906x^2 + 2.100551xy - 0.235395y^2},$$
  wobei $I = [-1.320618 \cdot 10^{-8}, 1.320618 \cdot 10^{-8}]$.

- Algorithmus zweiDFehlerzerteilung, mit Unterteilung in 5×5 Boxen und Verwendung von zweiDfehlermult

  $$\frac{3.000000 + 0.008483x + 0.314507y - 1.938879x^2 + 2.342033xy - 1.930559y^2 + I}{1 + 0.002853x - 0.228481y + 0.346906x^2 + 2.100551xy - 0.235395y^2},$$
  wobei $I = [-1.304236621986802 \cdot 10^{-8}, 1.304236621986802 \cdot 10^{-8}]$.

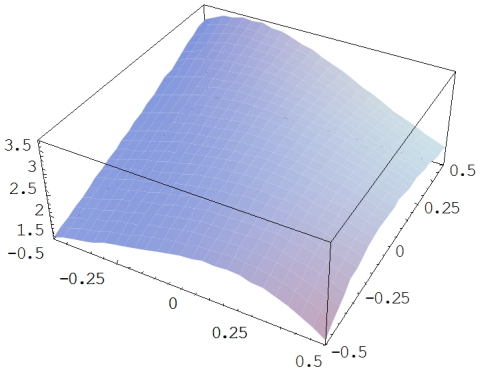

Abb. 7: $f(x,y) = \frac{3+y+2xy}{1+x^2+2xy} - \frac{y^2+xy^2}{1+2xy}$

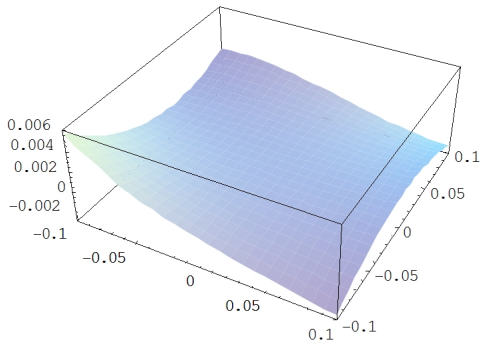

Abb. 8: Fehlerfunktion $f(z)q(z) - p(z)$ für z aus $[-0.1, 0.1]^2$, berechnet mittels Taylor-Modell.

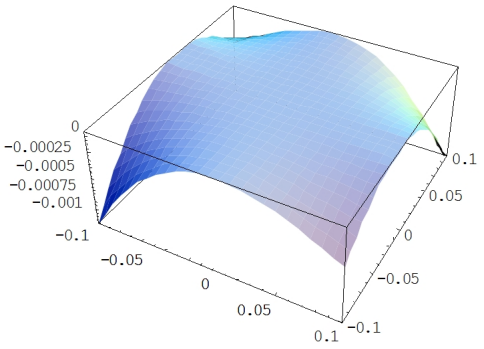

Abb. 9: Fehlerfunktion $f(z)q(z) - p(z)$ für z aus $[-0.1, 0.1]^2$, berechnet mittels Padé-Modell.

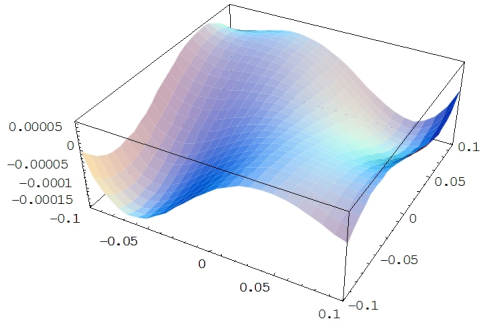

Abb. 10: Fehlerfunktion $f(z)q(z) - p(z)$ für z aus $[-0.1, 0.1]^2$, berechnet mittels Interpolationsmodell.

Vergleich der Lösungen

Untersucht man den maximalen Fehler bzw. die Breite eines Fehlerintervalls ohne Überschätzung (Berechnung zum Beispiel mit Mathematica), so ergeben die verschiedenen Lösungsvarianten Folgendes:

Methode	**Fehlerintervallbreite**
Taylor-Modell	$8.26151 \cdot 10^{-3}$
Padé-Modell	$8.49189 \cdot 10^{-4}$
Padé-Modell mit Faktoren	$1.54282 \cdot 10^{-3}$
Interpolationsmodell	$1.81044 \cdot 10^{-4}$

Tabelle 8: Intervall mit wenig (bzw. keiner) Überschätzung für $z \in [-0.1, 0.1]^2$

Methode	**Fehlerintervallbreite**
Taylor-Modell	$7.9978 \cdot 10^{-6}$
Padé-Modell	$9.16422 \cdot 10^{-8}$
Interpolationsmodell	$8.95463 \cdot 10^{-6}$

Tabelle 9: Intervall mit wenig (bzw. keiner) Überschätzung für $z \in [-0.01, 0.01]^2$

Methode	**Fehlerintervallbreite**
Taylor-Modell	$7.99982 \cdot 10^{-12}$
Padé-Modell	RF
Interpolationsmodell	$2.54148 \cdot 10^{-8}$

Tabelle 10: Intervall mit wenig (bzw. keiner) Überschätzung für $z \in [-10^{-4}, 10^{-4}]^2$

Betrachtet man die Box $[-0.1, 0.1]^2$, so sieht man, dass das Taylor-Modell die schlechteste Näherung an die zu behandelnde Funktion liefert. Wesentlich besser approximieren diese Funktion das Padé-Modell und das Interpolationsmodell, welches noch einmal, ein um den Faktor 4 kleineres Fehlerintervall liefert.

Bemerkung 9.5. *In den nachstehenden Tabellen und in zugehöriger Diskussion werden folgende Bezeichnungen verwendet.*
Alg. a ... Algorithmus zweiDfehlerslope
Alg. a' ... Algorithmus zweiDFehlerzerteilung
(mit Unterteilung in 5×5 *Boxen und Verwendung von zweiDfehlerslope)*
Alg. $\tilde{a}$ *... Algorithmus zweiDfehler*
Alg. b ... Algorithmus zweiDfehlerslopemult
Alg. b' ... Algorithmus zweiDFehlerzerteilung
(mit Unterteilung in 5×5 *Boxen und Verwendung von zweiDfehlermult)*
Alg. $\tilde{b}$ *... Algorithmus zweiDfehlermult*

Untersucht man die berechneten Abschätzungen des Fehlers, so ist auch hier klar ersichtlich, dass die Wahl des Algorithmus großen Einfluss auf das Ergebnis hat. Klarerweise liefern auch hier Algorithmus b und $\tilde{\text{b}}$ ein kleineres Fehlerintervall als Algorithmus a. Vergleicht man die Fehlerabschätzungen, so stellt man fest, dass das Fehlerintervall der mittels Taylor-Modell berechneten Lösung größer ist als die Intervalle des Padé-Modells und des Interpolationsmodells (Algorithmus b). Das ist jedoch nicht verwunderlich, da auch der Maximalfehler bei dieser Methode am größten ist.
Stellt man die Fehlerintervalle des Padé-Modells und des Interpolationsmodells gegenüber, so sieht man, dass die Fehlerintervallbreite des Interpolationsmodells, trotz geringeren maximalen Fehlers, größer ist als die Breite des Fehlerintervalls beim Padé-Modell. Grund dafür ist hier abermals die mehrfache Nullstelle der Fehlerfunktion des Padé-Modells, wodurch die Koeffizienten der niedrigen Potenzen von (x,y) wegfallen, und sich dadurch die Fehlerfunktion besser abschätzen lässt als beim Interpolationsmodell. Erst eine Unterteilung des Definitionsbereichs in kleinere Teilboxen liefert eine bessere Abschätzung.

Methode	**Fehlerintervall für** $z \in [-0.1, 0.1]^2$	**Rechenzeit**
Taylor-Modell	$[-6.146 \cdot 10^{-3}, 6.409 \cdot 10^{-3}]$	-
Padé-Modell	$[-2.354975 \cdot 10^{-3}, 1.2231853 \cdot 10^{-3}]$	-
Padé-Modell mit Koeffizienten	$[-1.33608 \cdot 10^{-3}, 1.014519 \cdot 10^{-3}]$	-
Interpolationsmodell, Alg. a)	$[-0.049616, 0.049616]$	2×0.515s
Interpolationsmodell, Alg. a')	$[-29.080568 \cdot 10^{-4}, 29.080568 \cdot 10^{-4}]$	2×9.890s
Interpolationsmodell, Alg. ã)	$[-0.4279, 0.4279]$	2×0.031s
Interpolationsmodell, Alg. b)	$[-15.103151 \cdot 10^{-4}, 15.103151 \cdot 10^{-4}]$	2×4.976s
Interpolationsmodell, Alg. b')	$[-7.128675 \cdot 10^{-4}, 7.128675 \cdot 10^{-4}]$	2×11.154s
Interpolationsmodell, Alg. $\tilde{\text{b}}$)	$[-15.103151 \cdot 10^{-4}, 15.103151 \cdot 10^{-4}]$	2×0.031s

Methode	**Fehlerintervall für** $z \in [-0.01, 0.01]^2$	**Rechenzeit**
Taylor-Modell	$[-4.2007 \cdot 10^{-6}, 4.26096 \cdot 10^{-6}]$	-
Padé-Modell	$[-2.218526 \cdot 10^{-7}, 1.121314 \cdot 10^{-7}]$	-
Interpolationsmodell, Alg. a)	$[-1.350069 \cdot 10^{-4}, 1.350069 \cdot 10^{-4}]$	2×0.518s
Interpolationsmodell, Alg. a')	$[-9.702316 \cdot 10^{-6}, 9.702316 \cdot 10^{-6}]$	2×9.782s
Interpolationsmodell, Alg. ã)	$[-2.2421 \cdot 10^{-2}, 2.2421 \cdot 10^{-2}]$	2×0.031s
Interpolationsmodell, Alg. b)	$[-6.400158 \cdot 10^{-6}, 6.400158 \cdot 10^{-6}]$	2×4.981s
Interpolationsmodell, Alg. b')	$[-5.4548596 \cdot 10^{-6}, 5.4548596 \cdot 10^{-6}]$	2×11.154s
Interpolationsmodell, Alg. $\tilde{\text{b}}$)	$[-6.400158 \cdot 10^{-6}, 6.400158 \cdot 10^{-6}]$	2×0.031s

Für $z \in [-0.01, 0.01]^2$ bzw. $z \in [-10^{-4}, 10^{-4}]^2$ zeigt sich beim Taylor-Modell und beim Padé-Modell wieder ein ähnliches Bild. Das Interpolationsmodell liefert allerdings, wie schon im eindimensionalen Beispiel, wieder schlechtere Ergebnisse, da die Koeffizienten abermals für $z \in [-0.1, 0.1]^2$ berechnet werden.

Methode	**Fehlerintervall für** $z \in [-10^{-4}, 10^{-4}]^2$	**Rechenzeit**
Taylor-Modell	$[-4.0022 \cdot 10^{-12}, 4.0028 \cdot 10^{-12}]$	-
Padé-Modell	$[-2.208101 \cdot 10^{-15}, 1.11209122 \cdot 10^{-15}]$	-
Interpolationsmodell, Alg. a)	$[-2.1055615 \cdot 10^{-8}, 2.1055615 \cdot 10^{-8}]$	2×0.544s
Interpolationsmodell, Alg. a')	$[-1.321697 \cdot 10^{-8}, 1.321697 \cdot 10^{-8}]$	2×9.902s
Interpolationsmodell, Alg. ã)	$[-2.01925 \cdot 10^{-4}, 2.01925 \cdot 10^{-4}]$	2×0.031s
Interpolationsmodell, Alg. b)	$[-1.320618 \cdot 10^{-8}, 1.320618 \cdot 10^{-8}]$	2×4.983s
Interpolationsmodell, Alg. b')	$[-1.304237 \cdot 10^{-8}, 1.304237 \cdot 10^{-8}]$	2×11.081s
Interpolationsmodell, Alg. b̃)	$[-1.320618 \cdot 10^{-8}, 1.320618 \cdot 10^{-8}]$	2×0.031s

9.3 Dreidimensionales Beispiel

Aufgabenstellung

Es sollen ein Taylor-Modell mittels Taylorarithmetik, und ein Interpolationsmodell für die Funktion f berechnet werden

$$f(p) = f(x, y, z) = 3/(1 - 4z^2) + (4x + 0.5y + 8xy - 2x^2).$$

Die Modelle sollen für die Definitionsbereiche $[-0.1, 0.1]^3$, $[-0.01, 0.01]^3$ und $[-10^{-4}, 10^{-4}]^3$ berechnet werden.

Taylor-Modell für $p \in [-0.1, 0.1]^3$

$$\begin{aligned} T_{3/(1-4z^2)} &= \left(3 \cdot (1 + 4z^2), \left[\frac{16z^4}{(1 + [0,1] \cdot 4z^2)^3}\right]\right) = \\ &= (3 \cdot (1 + 4z^2), [0, 0.0016]) \\ T_{(4x+0.5y+8xy-2x^2)} &= \left(4x + 0.5y + 8xy - 2x^2, [0, 0]\right) \end{aligned}$$

Man erhält also folgendes Taylor-Modell

$$T_{f(p)} = \left(3 + 4x + 0.5y + 8xy - 2x^2 + 12z^2, [0, 0.0016]\right).$$

Interpolationsmodell für $p \in [-0.1, 0.1]^3$

Die Punkte an denen ausgewertet wird, werden wie folgt gewählt

$$\begin{aligned} M = \Big\{ &(0,0,0), (1,0,0), (0,1,0), (0,0,1), (-1,0,0), (0,-1,0), (0,0,-1), \\ &(0.5,0,0), (0,0.5,0), (0,0,0.5), (-0.5,0,0), (0,-0.5,0), (0,0,-0.5), \\ &(1,1,0), (-1,1,0), (1,-1,0), (-1,-1,0), \\ &(1,0,1), (-1,0,1), (1,0,-1), (-1,0,-1), \\ &(0,1,1), (0,-1,1), (0,1,-1), (0,-1,-1) \Big\}. \end{aligned}$$

Mittels Algorithmus dreiDfehlerslope berechnete Koeffizienten und Fehlerabschätzung:

$$\begin{aligned} f(p) \in (&3.0535 + I + 3.5922 \cdot 10^{-11}x + 3.5922 \cdot 10^{-11}y + z - 2.4534 \cdot 10^{-16}xy + \\ &0.0039yz + 0.0039xz + 4.7029 \cdot 10^{-12}x^2 + 4.7029 \cdot 10^{-12}y^2 + 0.0091z^2)/ \\ &(1 + 0.0039x + 0.0039y + 0.0091z - 6.4086 \cdot 10^{-4}xy + \\ &9.4279 \cdot 10^{-9}yz + 9.4279 \cdot 10^{-9}xz - \\ &3.4469 \cdot 10^{-5}x^2 - 3.4469 \cdot 10^{-5}y^2 + 1.1057 \cdot 10^{-4}z^2) \quad \forall p \in [-0.1, 0.1]^3, \\ &\text{wobei } I = [-0.7079, 0.7079]. \end{aligned}$$

Taylor-Modell für $p \in [-0.01, 0.01]^3$

$$\begin{aligned} T_{3/(1-4z^2)} &= \left(3 \cdot (1 + 4z^2), \left[\frac{16z^4}{(1 + [0,1] \cdot 4z^2)^3}\right]\right) = \\ &= (3 \cdot (1 + 4z^2), [0, 1.6 \cdot 10^{-7}]) \end{aligned}$$

Daraus folgt

$$T_{f(p)} = \left(3 + 4x + 0.5y + 8xy - 2x^2 + 12z^2, [0, 1.6 \cdot 10^{-7}]\right).$$

Interpolationsmodell für $p \in [-0.01, 0.01]^3$

Seien die Punkte an denen ausgewertet wird die gleichen wie vorher, so errechnet sich für die Koeffizienten und die Fehlerabschätzung Folgendes.
Algorithmus dreiDfehlerslope:

$$\begin{aligned} f(p) \in (&3.0005 + I + 3.5922 \cdot 10^{-11}x + 3.5922 \cdot 10^{-11}y + z - 2.4534 \cdot 10^{-16}xy + \\ &0.0039yz + 0.0039xz + 4.7029 \cdot 10^{-12}x^2 + 4.7029 \cdot 10^{-12}y^2 + 0.0091z^2)/ \\ &(1 + 0.0039x + 0.0039y + 0.0091z - 6.4086 \cdot 10^{-4}xy + \\ &9.4279 \cdot 10^{-9}yz + 9.4279 \cdot 10^{-9}xz - \\ &3.4469 \cdot 10^{-5}x^2 - 3.4469 \cdot 10^{-5}y^2 + 1.1057 \cdot 10^{-4}z^2) \quad \forall p \in [-0.01, 0.01]^3, \\ &\text{wobei } I = [-0.0571, 0.0571]. \end{aligned}$$

Taylor-Modell für $p \in [-10^{-4}, 10^{-4}]^3$

$$\begin{aligned} T_{3/(1-4z^2)} &= \left(3 \cdot (1 + 4z^2), \left[\frac{16z^4}{(1 + [0,1] \cdot 4z^2)^3}\right]\right) = \\ &= (3 \cdot (1 + 4z^2), [0, 1.6 \cdot 10^{-15}]) \end{aligned}$$

Daher gilt

$$T_{f(p)} = \left(3 + 4x + 0.5y + 8xy - 2x^2 + 12z^2, [0, 1.6 \cdot 10^{-15}]\right).$$

Interpolationsmodell für $p \in [-10^{-4}, 10^{-4}]^3$

Wählt man die auszuwertenden Punkte wie oben, so ergibt sich mittels des Algorithmus dreiDfehlerslope folgendes Ergebnis.

$$\begin{aligned} f(p) \in (&3.0000 + I + 3.5922 \cdot 10^{-11}x + 3.5922 \cdot 10^{-11}y + z - 2.4534 \cdot 10^{-16}xy + \\ &0.0039yz + 0.0039xz + 4.7029 \cdot 10^{-12}x^2 + 4.7029 \cdot 10^{-12}y^2 + 0.0091z^2)/ \\ &(1 + 0.0039x + 0.0039y + 0.0091z - 6.4086 \cdot 10^{-4}xy + \\ &9.4279 \cdot 10^{-9}yz + 9.4279 \cdot 10^{-9}xz - \\ &3.4469 \cdot 10^{-5}x^2 - 3.4469 \cdot 10^{-5}y^2 + 1.1057 \cdot 10^{-4}z^2) \quad \forall p \in [-10^{-4}, 10^{-4}]^3, \\ &\text{wobei } I = [-5.4997 \cdot 10^{-4}, 5.4997 \cdot 10^{-4}]. \end{aligned}$$

Vergleich der Lösungen

In den folgenden Tabellen sind die berechneten Ergebnisse noch einmal zusammengefasst, wobei mit Alg. a) der Algorithmus dreiDfehlerslope bezeichnet wird.

Methode	**Fehlerintervall für** $p \in [-0.1, 0.1]^3$	Rechenzeit
Taylor-Modell	$[0, 0.0016]$	-
Interpolationsmodell, Alg. a)	$[-0.7079, 0.7079]$	258ms

Methode	**Fehlerintervall für** $p \in [-0.01, 0.01]^3$	Rechenzeit
Taylor-Modell	$[0, 1.6 \cdot 10^{-7}]$	-
Interpolationsmodell, Alg. a)	$[-0.0571, 0.0571]$	250ms

Methode	**Fehlerintervall für** $p \in [-10^{-4}, 10^{-4}]^3$	Rechenzeit
Taylor-Modell	$[0, 1.6 \cdot 10^{-15}]$	-
Interpolationsmodell, Alg. a)	$[-5.4997 \cdot 10^{-4}, 5.4997 \cdot 10^{-4}]$	250ms

Man sieht, dass für alle drei Definitionsbereiche, das Taylor-Modell ein wesentlich besseres Fehlerintervall liefert, als der Algorithmus a) beim Interpolationsmodell.

10 Conclusio

In dieser Arbeit werden zwei Modelle entwickelt, welche eine Funktion, ähnlich dem Taylor-Modell, approximieren sollen. Außerdem wird auf verschiedenen Wegen ein zugehöriges Fehlerintervall berechnet.

Vergleicht man die Ergebnisse, welche diese drei Modelle bei den angeführten Beispielen liefern, so ergibt sich Folgendes:
Könnte man die Koeffizienten des Padé-Modells und des Interpolationsmodells sowie die entsprechenden Fehlerintervalle ohne Überschätzung bestimmen, so würden beide Modelle wesentlich bessere Approximationen von Funktionen, und somit engere Fehlerintervalle liefern als das Taylor-Modell.
Dies ist allerdings oft nicht möglich. Beim Padé-Modell ist das größte Problem das zu lösende Gleichungssystem. Denn schon in zwei Dimensionen müssen 25 Koeffizienten bestimmt werden. Vor allem ergeben sich Rundungsfehler, sodass beim Abschätzen des Fehlerintervalls, die Koeffizienten der niedrigen Potenzen von x, nicht mehr wegfallen. Beim Interpolationsmodell treten vor allem bei einem sehr kleinen Definitionsbereich Probleme auf. Denn für sehr kleine Koeffizienten, ist die mittels CVX berechnete Lösung des Optimierungsproblems nicht mehr gut genung.

Untersucht man die erhaltenen Fehlerintervalle mit den zugehörigen Rechenzeiten der verschiedenen Implementation, so lässt sich feststellen, dass die Algorithmen b, $\tilde{b}$ und b' deutlich engere Fehlerintervalle liefern als die Algorithmen a, $\tilde{a}$ und a'. Aufgrund der benötigten Zeit ist allerdings nur der Algorithmus $\tilde{b}$ wirklich verwendbar, welcher von allen Algorithmen über das mit Abstand beste „Preis-Leistungs-Verhältnis“ verfügt. Insbesondere heißt das, dass sich die Abschätzung des Restgliedes mittels Slopes zeitlich nicht rechnet. Doch auch der Algorithmus $\tilde{b}$ ist nicht immer verwendbar, da bei diesem, vor der Berechnung des Fehlerintervalls, das Problem ausmultipliziert werden muss. Für höherdimensionale Probleme sind dafür allerdings eine Unmenge an Operationen notwendig, sodass die errechneten Koeffizienten zu sehr von Rundungsfehlern verfälscht werden, als diese verwenden zu können. So muss für solche Probleme der Algorithmus a verwendet werden. Dieser liefert zwar schnell ein Ergebnis, wie allerdings im dreidimensionalen Beispiel zu sehen ist, ist das keine vernünftige Alternative zum Taylor-Modell.

Abschließend muss man also feststellen, dass die beiden behandelten Modelle, in Verbindung mit den Algorithmen zur Fehlerabschätzung, wohl nur für ein- bzw. zweidimensionale Probleme und nicht allzu kleine Definitionsbereiche bessere Ergebnisse liefern als das Taylor-Modell.

A Programmquellcodes

A.1 einDmain.m

```
function [a0,a1,a2,b1,b2,u,o]=einDmain()

AnzOp=1;         %Anzahl der Operationen
ArtOp=[1,2];     %Welche Operationen werden gemacht
                 %(1..Add., 2..Subtr., 3.. Multipl. , 4.. Div.)

%Definition der Brüche von Polynomen inklusive Fehlerintervall

p0(1)=0.5;
p1(1)=0;
p2(1)=0;
q1(1)=0.5;
q2(1)=0;
u(1)=0;
o(1)=0;

p0(2)=4;
p1(2)=4;
p2(2)=1;
q1(2)=0;
q2(2)=0;
u(2)=0;
o(2)=0;

p0(3)=0;
p1(3)=0;
p2(3)=0;
q1(3)=0;
q2(3)=0;
u(3)=0;
o(3)=0;

I=10^-2;         %Definitionsbereich ist also das Intervall [-I,I].
J=max(0.1,I);    %Für [-J,J] werden mittels CVX Koeffizienten berechnet.

%Punkte an denen ausgewertet wird:
x(1)=-1*J;
x(2)=-0.707*J;
x(3)=0*J;
```

```
x(4)=0.707*J;
x(5)=1*J;


%Berechnung der Funktionswerte an den Punkten
for i=1:5
y(i)=(p0(1)+p1(1)*x(i)+p2(1)*(x(i))^2)/(1+q1(1)*x(i)+q2(1)*(x(i))^2);
end;



for j=2:AnzOp+1

    for i=1:5
        if(ArtOp(j-1)==1)
            y(i)=y(i)+((p0(j)+p1(j)*x(i)+p2(j)*(x(i))^2)/...
                       (1+q1(j)*x(i)+q2(j)*(x(i))^2));
        elseif(ArtOp(j-1)==2)
            y(i)=y(i)-((p0(j)+p1(j)*x(i)+p2(j)*(x(i))^2)/...
                       (1+q1(j)*x(i)+q2(j)*(x(i))^2));
        elseif(ArtOp(j-1)==3)
            y(i)=y(i)*((p0(j)+p1(j)*x(i)+p2(j)*(x(i))^2)/...
                       (1+q1(j)*x(i)+q2(j)*(x(i))^2));
        elseif(ArtOp(j-1)==4)
            y(i)=y(i)/((p0(j)+p1(j)*x(i)+p2(j)*(x(i))^2)/...
                       (1+q1(j)*x(i)+q2(j)*(x(i))^2));
        end;
    end;



    %Berechnung der Koeffizienten mittels CVX
    [p0(2+AnzOp),p1(2+AnzOp),p2(2+AnzOp),q1(2+AnzOp),q2(2+AnzOp)] = ...
        einDKoeffCVX (  x(1),x(2),x(3),x(4),x(5),...
                        y(1),y(2),y(3),y(4),y(5),-I,I);

    %Berechnung des Fehlerintervalls:
    %Anstatt des Algorithmus einDfehlerslope könnte man auch
    %andere Algorithmen zum Bestimmen des Restgliedes verwenden,zb.:
    %einDfehlermult,
    %einDFehlerzerteilung,
    %einDfehler,
    %zweiDfehlerslopemult mit entsprechenden Nullen.
    [u(2+AnzOp),o(2+AnzOp)] = einDfehlerslope ...
        (p0(1),p1(1),p2(1),q1(1),q2(1),u(1),o(1),ArtOp(j-1),...
         p0(j),p1(j),p2(j),q1(j),q2(j),u(j),o(j),...
         p0(2+AnzOp),p1(2+AnzOp),p2(2+AnzOp),q1(2+AnzOp),q2(2+AnzOp),-I,I);

    %Es wird das Fehlerintervall auf die Form [-c,c] gebracht:
    f=(o(2+AnzOp)+u(2+AnzOp))/2;
    o(2+AnzOp)=o(2+AnzOp)-f;
    u(2+AnzOp)=u(2+AnzOp)-f;
    p0(2+AnzOp)=p0(2+AnzOp)+f;

```

```
94
95      p0(1)=p0(2+AnzOp);
96      p1(1)=p1(2+AnzOp);
97      p2(1)=p2(2+AnzOp);
98      q1(1)=q1(2+AnzOp);
99      q2(1)=q2(2+AnzOp);
100     u(1)=u(2+AnzOp);
101     o(1)=o(2+AnzOp);
102
103 end;
104
105 %Ausgabe
106 a0=p0(2+AnzOp);
107 a1=p1(2+AnzOp);
108 a2=p2(2+AnzOp);
109 b1=q1(2+AnzOp);
110 b2=q2(2+AnzOp);
111 u=u(2+AnzOp);
112 o=o(2+AnzOp);
```

A.2 einDKoeffCVX.m

```
1  function [p0,p1,p2,q1,q2] = einDKoeffCVX...
2                              (x1,x2,x3,x4,x5,y1,y2,y3,y4,y5,Iu,Io)
3
4  c=-0.1;             %Zur "Verschärfung" der Nebenbedingungen
5  q=infsup(-1,0);     %Startwert für die Intervallabschätzung des Polynoms q
6  fact=1;
7
8  while(inf_(q)<0)    %Prüft ob die Nebenbedingungen "streng genug" sind
9      c=c+0.1;
10
11     cvx_begin       %Lösen des Optimierungsproblems mittels CVX
12
13         variable a0
14         variable a1
15         variable a2
16         variable b1
17         variable b2
18
19         minimize((a0+a1*x1+a2*x1^2-(y1)*(1+b1*fact*x1+b2*fact^2*x1^2))^2+...
20                  (a0+a1*x2+a2*x2^2-(y2)*(1+b1*fact*x2+b2*fact^2*x2^2))^2+...
21                  (a0+a1*x3+a2*x3^2-(y3)*(1+b1*fact*x3+b2*fact^2*x3^2))^2+...
22                  (a0+a1*x4+a2*x4^2-(y4)*(1+b1*fact*x4+b2*fact^2*x4^2))^2+...
23                  (a0+a1*x5+a2*x5^2-(y5)*(1+b1*fact*x5+b2*fact^2*x5^2))^2)
24
25         subject to
26
27                  (1+b1*fact*x1+b2*fact^2*x1^2)>c
28                  (1+b1*fact*x2+b2*fact^2*x2^2)>c
29                  (1+b1*fact*x3+b2*fact^2*x3^2)>c
```

```
                (1+b1*fact*x4+b2*fact^2*x4^2)>c
                (1+b1*fact*x5+b2*fact^2*x5^2)>c
    cvx_end

    q1=b1;
    q2=b2;

    %Bestimmen des Minimums von q
    if(abs(q2)>0.00001)
        m=-q1/(2*q2);
    else
        m=Iu;
    end

    fm=1+q1*m+q2*m^2;
    fu=1+q1*Iu+q2*Iu^2;
    fo=1+q1*Io+q2*Io^2;
    if(Iu<m<Io)
        qu=min(min(fm,fu),fo);
        qo=max(max(fm,fu),fo);
    else
        qu=min(fu,fo);
        qo=max(fu,fo);
    end

    q=infsup(qu,qo);        %Intervallabschätzung von q


end

p0=a0;
p1=a1;
p2=a2;
```

A.3 einDfehlerslope.m

```
function [u,o] = einDfehlerslope  ( pa0,pa1,pa2,qa1,qa2,ua,oa,operation,...
                                    pb0,pb1,pb2,qb1,qb2,ub,ob,...
                                    p0,p1,p2,q1,q2,Iu,Io)

za=Iu;
zb=Io;

I=infsup(Iu,Io);
Ia=infsup(ua,oa);
Ib=infsup(ub,ob);

%Berechnung der Slopes erster und zweiter Ordnung (f[za,x],f[za,zb,x])
[s0za,s1za,s2]=slope1(pa0,pa1,pa2,qa1,qa2,Ia,pb0,pb1,pb2,qb1,qb2,Ib,...
                      p0,p1,p2,q1,q2,operation,za,zb,Iu,Io);

```

```
%Berechnung der Slopes erster und zweiter Ordnung (f[zb,x],f[zb,za,x])
[s0zb,s1zb,pp]=slope1(pa0,pa1,pa2,qa1,qa2,Ia,pb0,pb1,pb2,qb1,qb2,Ib,...
                      p0,p1,p2,q1,q2,operation,zb,za,Iu,Io);

%Berechnung von f[za,zb]
fzazb=s0za+s1za*zb;


if(operation==1) %Addition

    fza=((pa0+pa1*za+pa2*power(za,2)+Ia)/(1+qa1*za+qa2*power(za,2))+...
         (pb0+pb1*za+pb2*power(za,2)+Ib)/(1+qb1*za+qb2*power(za,2)))*...
        (1+q1*za+q2*power(za,2))-(p0+p1*za+p2*power(za,2));

    fzb=((pa0+pa1*zb+pa2*power(zb,2)+Ia)/(1+qa1*zb+qa2*power(zb,2))+...
         (pb0+pb1*zb+pb2*power(zb,2)+Ib)/(1+qb1*zb+qb2*power(zb,2)))*...
        (1+q1*zb+q2*power(zb,2))-(p0+p1*zb+p2*power(zb,2));

    %Einsetzen des Defintionsbereiches in f(x)
    I1=((pa0+pa1*I+pa2*power(I,2)+Ia)/(1+qa1*I+qa2*power(I,2))+...
        (pb0+pb1*I+pb2*power(I,2)+Ib)/(1+qb1*I+qb2*power(I,2)))*...
       (1+q1*I+q2*power(I,2))-(p0+p1*I+p2*power(I,2));

    %Abschätzen des Fehlerintervalls mittels Slopes erster Ordnung
    %an den Punkte za bzw. zb.
    I2=fza+(s0za+s1za*I)*(I-za);
    I3=fzb+(s0zb+s1zb*I)*(I-zb);
    %Abschätzen des Fehlerintervalls mittels Slopes zweiter Ordnung
    I4=fza+(fzazb+(I-zb)*s2)*(I-za);

    %Schneiden der Intervalle
    I5=intersect(intersect(I1,I2),intersect(I3,I4));


elseif(operation==2) %Subtraktion

    fza=((pa0+pa1*za+pa2*power(za,2)+Ia)/(1+qa1*za+qa2*power(za,2))-...
          (pb0+pb1*za+pb2*power(za,2)+Ib)/(1+qb1*za+qb2*power(za,2)))*...
          (1+q1*za+q2*power(za,2))-(p0+p1*za+p2*power(za,2));
    fzb=((pa0+pa1*zb+pa2*power(zb,2)+Ia)/(1+qa1*zb+qa2*power(zb,2))-...
          (pb0+pb1*zb+pb2*power(zb,2)+Ib)/(1+qb1*zb+qb2*power(zb,2)))*...
          (1+q1*zb+q2*power(zb,2))-(p0+p1*zb+p2*power(zb,2));

    I1=((pa0+pa1*I+pa2*power(I,2)+Ia)/(1+qa1*I+qa2*power(I,2))-...
         (pb0+pb1*I+pb2*power(I,2)+Ib)/(1+qb1*I+qb2*power(I,2)))*...
         (1+q1*I+q2*power(I,2))-(p0+p1*I+p2*power(I,2));
    I2=fza+(s0za+s1za*I)*(I-za);
    I3=fzb+(s0zb+s1zb*I)*(I-zb);
    I4=fza+(fzazb+(I-zb)*s2)*(I-za);

    I5=intersect(intersect(I1,I2),intersect(I3,I4));

elseif(operation==3) %Multiplikation

```

```
70      fza=((pa0+pa1*za+pa2*power(za,2)+Ia)/(1+qa1*za+qa2*power(za,2))*...
71            (pb0+pb1*za+pb2*power(za,2)+Ib)/(1+qb1*za+qb2*power(za,2)))*...
72            (1+q1*za+q2*power(za,2))-(p0+p1*za+p2*power(za,2));
73      fzb=((pa0+pa1*zb+pa2*power(zb,2)+Ia)/(1+qa1*zb+qa2*power(zb,2))*...
74            (pb0+pb1*zb+pb2*power(zb,2)+Ib)/(1+qb1*zb+qb2*power(zb,2)))*...
75            (1+q1*zb+q2*power(zb,2))-(p0+p1*zb+p2*power(zb,2));
76
77      I1=((pa0+pa1*I+pa2*power(I,2)+Ia)/(1+qa1*I+qa2*power(I,2))*...
78           (pb0+pb1*I+pb2*power(I,2)+Ib)/(1+qb1*I+qb2*power(I,2)))*...
79           (1+q1*I+q2*power(I,2))-(p0+p1*I+p2*power(I,2));
80      I2=fza+(s0za+s1za*I)*(I-za);
81      I3=fzb+(s0zb+s1zb*I)*(I-zb);
82      I4=fza+(fzazb+(I-zb)*s2)*(I-za);
83
84      I5=intersect(intersect(I1,I2),intersect(I3,I4));
85
86
87  elseif(operation==4) %Division
88
89      fza=((pa0+pa1*za+pa2*power(za,2)+Ia)/(1+qa1*za+qa2*power(za,2))/...
90            (pb0+pb1*za+pb2*power(za,2)+Ib)/(1+qb1*za+qb2*power(za,2)))*...
91            (1+q1*za+q2*power(za,2))-(p0+p1*za+p2*power(za,2));
92      fzb=((pa0+pa1*zb+pa2*power(zb,2)+Ia)/(1+qa1*zb+qa2*power(zb,2))/...
93            (pb0+pb1*zb+pb2*power(zb,2)+Ib)/(1+qb1*zb+qb2*power(zb,2)))*...
94            (1+q1*zb+q2*power(zb,2))-(p0+p1*zb+p2*power(zb,2));
95
96      I1=((pa0+pa1*I+pa2*power(I,2)+Ia)/(1+qa1*I+qa2*power(I,2))/...
97           (pb0+pb1*I+pb2*power(I,2)+Ib)/(1+qb1*I+qb2*power(I,2)))*...
98           (1+q1*I+q2*power(I,2))-(p0+p1*I+p2*power(I,2));
99      I2=fza+(s0za+s1za*I)*(I-za);
100     I3=fzb+(s0zb+s1zb*I)*(I-zb);
101     I4=fza+(fzazb+(I-zb)*s2)*(I-za);
102
103     I5=intersect(intersect(I1,I2),intersect(I3,I4));
104
105 end;
106
107 u=inf_(I5);
108 o=sup(I5);
```

A.4 slope1.m

```
1  function [s0z,s1z,s0z2] = slope1(   pa0,pa1,pa2,qa1,qa2,Ia,...
2                                      pb0,pb1,pb2,qb1,qb2,Ib,...
3                                      p0,p1,p2,q1,q2,operation,z,zb,Iu,Io)
4
5  %Definitionsbereich
6  I=infsup(Iu,Io);
7
8  %Berechnung von Slopes erster Ordnung der verschiedenen Polynome
9  [pas0z,pas1z]=slopepoly(pa0,pa1,pa2,z);
```

```
[qas0z,qas1z]=slopepoly(1,qa1,qa2,z);
[qas0zb,qas1zb]=slopepoly(1,qa1,qa2,zb);
[pbs0z,pbs1z]=slopepoly(pb0,pb1,pb2,z);
[pbs0zb,pbs1zb]=slopepoly(pb0,pb1,pb2,zb);
[qbs0z,qbs1z]=slopepoly(1,qb1,qb2,z);
[qbs0zb,qbs1zb]=slopepoly(1,qb1,qb2,zb);
[ps0z,ps1z]=slopepoly(p0,p1,p2,z);
[qs0z,qs1z]=slopepoly(1,q1,q2,z);
[qs0zb,qs1zb]=slopepoly(1,q1,q2,zb);


ha=(pa0+pa1*z+pa2*z^2+Ia)/(1+qa1*z+qa2*z^2);
hb=(pb0+pb1*z+pb2*z^2+Ib)/(1+qb1*z+qb2*z^2);

%Berechne die Koeffizienten von x^0,x^1 des Slopes erster Ordnung
%von pa/qa an der Stelle z, anhand bekannter Rechenregeln für Slopes
paqa0=(pas0z-(ha)*qas0z)/(1+qa1*I+qa2*power(I,2));
paqa1=(pas1z-(ha)*qas1z)/(1+qa1*I+qa2*power(I,2));

%Berechne die Koeffizienten von x^0,x^1 des Slopes erster Ordnung
%von pb/qb an der Stelle z
pbqb0=(pbs0z-(hb)*qbs0z)/(1+qb1*I+qb2*power(I,2));
pbqb1=(pbs1z-(hb)*qbs1z)/(1+qb1*I+qb2*power(I,2));

%Berechne die Koeffizienten von x^0,x^1 des Slopes erster Ordnung
%von pb/qb an der Stelle zb
pbqbzb0=(pbs0zb-(hb)*qbs0zb)/(1+qb1*I+qb2*power(I,2));
pbqbzb1=(pbs1zb-(hb)*qbs1zb)/(1+qb1*I+qb2*power(I,2));

hbzb=(pb0+pb1*zb+pb2*zb^2+Ib)/(1+qb1*zb+qb2*zb^2);

%Berechne den Koeffizienten von x^0 des Slopes zweiter Ordnung
%von pa/qa[z,zb,x]
paqa02= (pas1z-(ha)*qas1z+(qas0zb+qas1zb*I)*...
        ((qas0z+qas1z*zb)*ha-(pas0z+pas1z*zb))/...
        (1+qa1*zb+qa2*zb^2))/(1+qa1*I+qa2*power(I,2));

%Berechne den Koeffizienten von x^0 des Slopes zweiter Ordnung
%von pb/qb[z,zb,x]
pbqb02= (pbs1z-(hb)*qbs1z+(qbs0zb+qbs1zb*I)*...
        ((qbs0z+qbs1z*zb)*hb-(pbs0z+pbs1z*zb))/...
        (1+qb1*zb+qb2*zb^2))/(1+qb1*I+qb2*power(I,2));

%Berechnung einer Intervallabschätzung für q
if(abs(q2)>0.00001)
m=-q1/(2*q2);
else
    m=Iu;
end
fm=1+q1*m+q2*m^2;
fu=1+q1*Iu+q2*Iu^2;
fo=1+q1*Io+q2*Io^2;
if(Iu<m<Io)
    qu=min([fm,fu,fo]);
```

```
64      qo=max([fm,fu,fo]);
65  else
66      qu=min(fu,fo);
67      qo=max(fu,fo);
68  end
69  q=infsup(qu,qo);
70
71
72  %Es folgen die Berechnungen der Slopes erster und zweiter Ordnung
73  %für die verschiedenen Operationen anhand den bekannten Rechenregeln
74  %für Slopes:
75
76  if(operation==1)
77      a=paqa0+pbqb0;
78      b=paqa1+pbqb1;
79
80      c=a*q+(ha+hb)*qs0z;
81      d=b*q+(ha+hb)*qs1z;
82
83      s0z=c-ps0z;
84      s1z=d-ps1z;
85
86      e=paqa02+pbqb02;
87      e=e*q+(ha+hb)*qs1z+(a+b*zb)*(qs0zb+qs1zb*I);
88
89      s0z2=e-ps1z;
90
91  elseif(operation==2)
92      a=paqa0-pbqb0;
93      b=paqa1-pbqb1;
94
95      c=a*q+(ha-hb)*qs0z;
96      d=b*q+(ha-hb)*qs1z;
97
98      s0z=c-ps0z;
99      s1z=d-ps1z;
100
101     e=paqa02-pbqb02;
102     e=e*q+(ha-hb)*qs1z+(a+b*zb)*(qs0z+qs1z*I);
103
104     s0z2=e-ps1z;
105
106 elseif(operation==3)
107     a=paqa0*(pb0+pb1*I+pb2*power(I,2)+Ib)/(1+qb1*I+qb2*power(I,2))+ha*pbqb0;
108     b=paqa1*(pb0+pb1*I+pb2*power(I,2)+Ib)/(1+qb1*I+qb2*power(I,2))+ha*pbqb1;
109
110     c=a*q+(ha*hb)*qs0z;
111     d=b*q+(ha*hb)*qs1z;
112
113     s0z=c-ps0z;
114     s1z=d-ps1z;
115
116     e=paqa02*(pb0+pb1*I+pb2*power(I,2)+Ib)/(1+qb1*I+qb2*power(I,2))+...
117         pbqb02*ha+(paqa0+paqa1*zb)*(pbqb0+pbqb1*I);
```

```
118     e=e*q+(ha*hb)*qs1z+(a+b*zb)*(qs0z+qs1z*I);
119
120     s0z2=e-ps1z;
121
122 elseif(operation==4)
123     a=(paqa0-(ha/hb)*pbqb0)/((pb0+pb1*I+pb2*power(I,2)+Ib)/...
124         (1+qb1*I+qb2*power(I,2)));
125     b=(paqa1-(ha/hb)*pbqb1)/((pb0+pb1*I+pb2*power(I,2)+Ib)/...
126         (1+qb1*I+qb2*power(I,2)));
127
128     c=a*q+(ha/hb)*qs0z;
129     d=b*q+(ha/hb)*qs1z;
130
131     s0z=c-ps0z;
132     s1z=d-ps1z;
133
134     e=(paqa02-(ha/hb)*pbqb02+((pbqbzb0+pbqbzb1*I)*((ha/hb)*(pbqb0+pbqb1*zb)-...
135         (paqa0+paqa1*zb))/(hbzb)))/((pb0+pb1*I+pb2*power(I,2)+Ib)/...
136         (1+qb1*I+qb2*power(I,2)));
137     e=e*q+(ha/hb)*qs1z+(a+b*zb)*(qs0z+qs1z*I);
138
139     s0z2=e-ps1z;
140 end;
```

A.5 slopepoly.m

```
1 function [s0,s1]=slopepoly(e0,e1,e2,z)
2
3 s0=e1+e2*z;
4 s1=e2*1;
```

A.6 einDfehler.m

```
1  function [u,o] = einDfehler(pa0,pa1,pa2,qa1,qa2,ua,oa,operation,...
2                              pb0,pb1,pb2,qb1,qb2,ub,ob,p0,p1,p2,q1,q2,Iu,Io)
3
4  I=infsup(Iu,Io);
5  Ia=infsup(ua,oa);
6  Ib=infsup(ub,ob);
7
8  if(operation==1) %Addition
9
10     %Einsetzen des Defintionsbereiches in f(x)
11     I1=((pa0+pa1*I+pa2*power(I,2)+Ia)/(1+qa1*I+qa2*power(I,2))+...
12         (pb0+pb1*I+pb2*power(I,2)+Ib)/(1+qb1*I+qb2*power(I,2)))*...
13        (1+q1*I+q2*power(I,2))-(p0+p1*I+p2*power(I,2));
14
```

```
elseif(operation==2) %Subtraktion

    I1=((pa0+pa1*I+pa2*power(I,2)+Ia)/(1+qa1*I+qa2*power(I,2))-...
         (pb0+pb1*I+pb2*power(I,2)+Ib)/(1+qb1*I+qb2*power(I,2)))*...
         (1+q1*I+q2*power(I,2))-(p0+p1*I+p2*power(I,2));

elseif(operation==3) %Multiplikation

    I1=((pa0+pa1*I+pa2*power(I,2)+Ia)/(1+qa1*I+qa2*power(I,2))*...
         (pb0+pb1*I+pb2*power(I,2)+Ib)/(1+qb1*I+qb2*power(I,2)))*...
         (1+q1*I+q2*power(I,2))-(p0+p1*I+p2*power(I,2));

elseif(operation==4) %Division

    I1=((pa0+pa1*I+pa2*power(I,2)+Ia)/(1+qa1*I+qa2*power(I,2))/...
         (pb0+pb1*I+pb2*power(I,2)+Ib)/(1+qb1*I+qb2*power(I,2)))*...
         (1+q1*I+q2*power(I,2))-(p0+p1*I+p2*power(I,2));

end;

u=inf_(I1);
o=sup(I1);
```

A.7 einDFehlerzerteilung.m

```
function [u,o] = einDFehlerzerteilung...
                         (pa0,pa1,pa2,qa1,qa2,ua,oa,operation,...
                          pb0,pb1,pb2,qb1,qb2,ub,ob,p0,p1,p2,q1,q2,Iu1,Io1)

%Bestimme die Anzahl der Teilintervalle
anzbox_x=10;

m=0;
kx=(Io1-Iu1)/anzbox_x;

%Definiere Ober- und Untergrenzen der Teilintervalle
for k=1:anzbox_x
    m=m+1;
    Io(m)=Io1-(k-1)*kx;
    Iu(m)=Io1-k*kx;
end;

%Berechne die Restglieder der Teilintervalle
%Anstatt des Algorithmus einDfehlerslope könnten auch andere
%alle Algorithmen zur Restintervallbestimmung verwendet werden, zb.:
%einDfehlermult,
%einDfehler,
%zweiDfehlerslopemult mit entsprechenden Nullen.
for i=1:anzbox_x
    [u(i),o(i)]=einDfehlerslope(pa0,pa1,pa2,qa1,qa2,ua,oa,operation,...
                                pb0,pb1,pb2,qb1,qb2,ub,ob,...
```

```
                                    p0,p1,p2,q1,q2,Iu(i),Io(i));
end;

u=min(u);
o=max(o);
```

A.8 einDfehlermult.m

```
function [u,o] = einDfehlermult...
                           (pa0,pa1,pa2,qa1,qa2,ua,oa,operation,...
                            pb0,pb1,pb2,qb1,qb2,ub,ob,p0,p1,p2,q1,q2,Iu,Io)

Ix=infsup(Iu,Io);
Ia=infsup(ua,oa);
Ib=infsup(ub,ob);

%Ausmultiplizieren des Problems, man erhält die Koeffizienten eines
%Polynoms sechster und eines vierter Ordnung.
[e0,e1,e2,e3,e4,e5,e6,n0,n1,n2,n3,n4]= ...
    eindimausmultipl(   pa0+Ia,pa1,pa2,qa1,qa2,operation,...
                        pb0+Ib, pb1 ,pb2,qb1,qb2,p0,p1,p2,q1,q2);

%Abschätzen des Wertebereichs durch Einsetzten der Definitionsbereichs
I=(e0+e1*Ix+e2*power(Ix,2)+e3*power(Ix,3)+e4*power(Ix,4)+e5*power(Ix,5)+ ...
   e6*power(Ix,6))/(n0+n1*Ix+n2*power(Ix,2)+n3*power(Ix,3)+n4*power(Ix,4));

u=inf_(I);
o=sup(I);
```

A.9 eindimausmultipl.m

```
function [e0,e1,e2,e3,e4,e5,e6,n0,n1,n2,n3,n4]= ...
            eindimausmultipl(pa0,pa1,pa2,qa1,qa2,operation,...
                             pb0, pb1 ,pb2,qb1,qb2,p0,p1,p2,q1,q2)

if(operation==1) %Addition

    e0=pa0+pb0-p0;
    e1=pa1+pb1pb0*qa1+pa0*qb1-qa1*p0-qb1*p0-p1+pa0*q1+pb0*q1;
    e2=pa2 + pb2 + pb1 * qa1 + pb0 * qa2 + pa1  * qb1 + pa0 *qb2 - qa2*p0-...
        qa1 *qb1* p0 - qb2*p0 - qa1 *p1 - qb1 *p1 - p2 + pa1 *q1 + pb1*q1+...
        pb0 *qa1 * q1 + pa0 *qb1 * q1 + pa0*q2 + pb0 *q2;
    e3=pb2 *qa1 + pb1 *qa2 + pa2 *qb1 + pa1 *qb2 - ...
        qa2* qb1 *p0-qa1 *qb2*p0 - qa2*p -qa1 *qb1 *p1 -qb2*p1 - qa1*p2- ...
        qb1*p2 + pa2*q1 + pb2*q1 + pb1*qa1*q1 + pb0* qa2 *q1+pa1*qb1*q1+ ...
        pa0 *qb2* q1 + pa1 *q2 + pb1* q2 + pb0 *qa1 *q2 + pa0* qb1* q2;
    e4=pb2 *qa2 + pa2 *qb2 - ...
```

```
        qa2* qb2* p0 - qa2* qb1* p1 - qa1* qb2* p1 - qa2 *p2-qa1*qb1*p2- ...
        qb2* p2 + pb2*qa1* q1 + pb1* qa2*q1 + pa2 *qb1 *q1 + pa1*qb2*q1+ ...
        pa2*q2 +pb2*q2 +pb1*qa1* q2+pb0*qa2*q2+pa1 *qb1* q2 + pa0*qb2*q2;
    e5=-qa2*qb2*p1- qa2* qb1 *p2-qa1 *qb2*p2+ pb2*qa2*q1+pa2* qb2 *q1 + ...
        pb2 *qa1*q2+pb1*qa2*q2 + pa2*qb1* q2 + pa1 *qb2* q2;
    e6=-qa2*qb2*p2 + pb2 *qa2* q2 + pa2* qb2* q2;
    n0=1;
    n1=qa1 + qb1;
    n2=qa2 + qa1 *qb1 + qb2;
    n3=qa2 *qb1 + qa1 *qb2;
    n4=qa2 *qb2;

elseif(operation==2) %Subtraktion
    e0= -p0 + pa0 - pb0;
    e1=-p1+pa1 -pb1 + pa0*q1 -pb0*q1-p0 *qa1 -pb0*qa1 -p0* qb1 + pa0 *qb1;
    e2=-p2+pa2 -pb2 + pa1*q1 -pb1*q1+pa0 *q2 -pb0*q2 - p1 *qa1 - pb1* qa1-...
        pb0*q1*qa1 -p0*qa2 - pb0*qa2 - p1*qb1 + pa1*qb1 + pa0*q1 *qb1 - ...
        p0* qa1* qb1 - p0 *qb2 + pa0 *qb2;
    e3=pa2*q1 - pb2*q1 + pa1*q2 - pb1*q2 - p2*qa1 - pb2*qa1 - pb1*q1*qa1 - ...
        pb0*q2* qa1 - p1 *qa2 - pb1*qa2 - pb0*q1 *qa2 - p2*qb1 +pa2*qb1 + ...
        pa1*q1*qb1 + pa0 *q2 *qb1 - p1 *qa1* qb1 - p0 *qa2* qb1 - p1*qb2+ ...
        pa1* qb2 + pa0 *q1 *qb2 - p0 *qa1* qb2;
    e4=pa2* q2 - pb2* q2 - pb2 *q1*qa1 - pb1*q2*qa1 - p2* qa2 - pb2 *qa2 -...
        pb1 *q1* qa2 - pb0 *q2 *qa2 + pa2* q1 *qb1 + pa1 *...
        q2* qb1 - p2 *qa1 *qb1 - p1* qa2 *qb1 - p2 *qb2 + pa2*qb2 + ...
        pa1* q1* qb2 + pa0* q2 *qb2 - p1* qa1 *qb2 - p0 *qa2 *qb2;
    e5=-pb2* q2* qa1 - pb2* q1* qa2 - pb1* q2* qa2 + pa2 *q2 *qb1 - ...
        p2 *qa2*qb1+pa2 *q1 *qb2 +pa1* q2*qb2 - p2* qa1* qb2 - p1*qa2*qb2;
    e6=-pb2* q2 *qa2 + pa2* q2 *qb2 - p2* qa2 *qb2;
    n0=1;
    n1=qa1 + qb1;
    n2=qa2 + qa1 *qb1 + qb2;
    n3=qa2 *qb1 + qa1 *qb2;
    n4=qa2 *qb2;

elseif(operation==3) %Multiplikation
    e0= -p0 + pa0 *pb0;
    e1=-p1 + pa1* pb0 + pa0* pb1 + pa0* pb0* q1 - p0 *qa1 - p0 *qb1;
    e2=-p2 + pa2 *pb0 + pa1 *pb1 + pa0 *pb2 + pa1*pb0 *q1 + pa0* pb1 *q1 +...
        pa0* pb0 *q2 - p1*qa1 - p0* qa2 - p1*qb1 - p0 *qa1*qb1 - p0 *qb2;
    e3=pa2 *pb1 + pa1 *pb2 +pa2 *pb0* q1 + pa1 *pb1* q1 + pa0* pb2*q1 + ...
        pa1 *pb0*q2 + pa0* pb1 *q2 - p2 *qa1 - p1* qa2 - p2* qb1 - ...
        p1 *qa1* qb1 - p0* qa2 *qb1 - p1* qb2 - p0 *qa1 *qb2;
    e4=pa2*pb2 + pa2 *pb1* q1 + pa1 *pb2* q1 + pa2 *pb0* q2 + pa1*pb1 *q2+...
        pa0*pb2*q2 - p2*qa2 - p2*qa1* qb1 - p1*qa2*qb1-p2*qb2-p1*qa1*qb2 -...
        p0 *qa2* qb2;
    e5=pa2*pb2*q1+pa2*pb1*q2+pa1*pb2*q2-p2*qa2*qb1-p2*qa1*qb2-p1*qa2*qb2;
    e6=pa2* pb2*q2 - p2 *qa2 *qb2;
    n0=1;
    n1=qa1 + qb1;
    n2=qa2 + qa1 *qb1 + qb2;
    n3=qa2 *qb1 + qa1 *qb2;
    n4=qa2 *qb2;

```

```
elseif(operation==4) %Division
    e0= pa0 - p0* pb0;
    e1=pa1 - p1* pb0 - p0 *pb1 + pa0* q1 - p0*pb0 *qa1 + pa0 *qb1;
    e2=pa2 - p2*pb0 - p1*pb1 - p0 *pb2 + pa1*q1 + pa0 *q2 - p1*pb0*qa1- ...
        p0 *pb1* qa1 - p0* pb0 *qa2 + pa1 *qb1 + pa0 *q1* qb1 + pa0 *qb2;
    e3=-p2* pb1 - p1*pb2 + pa2* q1 + pa1* q2 -p2* pb0 *qa1 - p1*pb1*qa1-...
        p0 *pb2 *qa1 - p1 *pb0 *qa2 - p0 *pb1 *qa2 + pa2 *qb1 + ...
        pa1* q1* qb1 + pa0 *q2 *qb1 + pa1*qb2 + pa0 *q1* qb2;
    e4=-p2 *pb2 + pa2 *q2 -p2* pb1*qa1 - p1 *pb2* qa1 - p2* pb0 *qa2 - ...
        p1* pb1*qa2 - p0 *pb2 *qa2 + pa2* q1 *qb1 + pa1* q2 *qb1 + ...
        pa2 *qb2 + pa1* q1* qb2 + pa0* q2* qb2;
    e5=-p2*pb2*qa1 - p2*pb1*qa2-p1*pb2*qa2+pa2*q2*qb1+pa2*q1*qb2+pa1*q2*qb2;
    e6=-p2* pb2*qa2 + pa2* q2 *qb2;
    n0=pb0;
    n1=pb1 + pb0* qa1;
    n2=pb2 + pb1 *qa1 + pb0 *qa2;
    n3=pb2 *qa1 + pb1* qa2;
    n4=pb2 *qa2;

end;
```

A.10 zweiDmain.m

```
function [a00,a10,a01,a20,a11,a02,b10,b01,b20,b11,b02,u,o]=zweiDmain()

AnzOp=2;         %Anzahl der Operationen
ArtOp=[3,1];     %Welche Operationen werden gemacht
                 % (1..Add., 2..Subtr., 3.. Multipl. , 4.. Div.)

%Definition der Brüche von Polynomen inklusive Fehlerintervall
p00(1)=int(0);  %int(0)=infsup(0,0)
p10(1)=int(0);
p01(1)=int(0);
p11(1)=int(0);
p20(1)=int(0);
p02(1)=int(-1);
q10(1)=int(0);
q01(1)=int(0);
q20(1)=int(0);
q11(1)=int(0);
q02(1)=int(0);
u(1)=-0.0;
o(1)=0.0;

p00(2)=int(1);
p10(2)=int(1);
p01(2)=int(0);
p11(2)=int(0);
p20(2)=int(0);
p02(2)=int(0);
q10(2)=int(0);
```

```
29 q01(2)=int(0);
30 q20(2)=int(0);
31 q11(2)=int(2);
32 q02(2)=int(0);
33 u(2)=0;
34 o(2)=0;
35
36 p00(3)=int(3);
37 p10(3)=int(0);
38 p01(3)=int(1);
39 p11(3)=int(2);
40 p20(3)=int(0);
41 p02(3)=int(0);
42 q10(3)=int(0);
43 q01(3)=int(0);
44 q20(3)=int(1);
45 q11(3)=int(2);
46 q02(3)=int(0);
47 u(3)=0;
48 o(3)=0;
49
50
51 coeff=10^-1;
52 Jx=1*coeff;
53 Jy=1*coeff;       %Definitionsbereich ist also die [-Jx,Jx]x[-Jy,Jy].
54 Ix=max(Jx,0.1);   %Für [-Ix,Ix]x[-Iy,Iy] werden mittels CVX Koeffizienten
55 Iy=max(Jy,0.1);   %berechnet.
56 a=1;
57 b=0.5;
58 c=1;
59
60 %Punkte an denen ausgewertet wird:
61 x(1)=0*Ix;
62 y(1)=0*Iy;
63 x(2)=a*Ix;
64 y(2)=0*Iy;
65 x(3)=0*Ix;
66 y(3)=a*Iy;
67 x(4)=-a*Ix;
68 y(4)=0*Iy;
69 x(5)=0*Ix;
70 y(5)=-a*Iy;
71 x(6)=b*Ix;
72 y(6)=0*Iy;
73 x(7)=0*Ix;
74 y(7)=b*Iy;
75 x(8)=-b*Ix;
76 y(8)=0*Iy;
77 x(9)=0*Ix;
78 y(9)=-b*Iy;
79 x(10)=c*Ix;
80 y(10)=c*Iy;
81 x(11)=c*Ix;
82 y(11)=-c*Iy;
```

```
x(12)=-c*Ix;
y(12)=c*Iy;
x(13)=-c*Ix;
y(13)=-c*Iy;

%Berechnung der Funktionswerte an den Punkten
for i=1:13
z(i)=mid((p00(1)+p10(1)*x(i)+p01(1)*y(i)+p11(1)*x(i)*y(i)+p20(1)*x(i)^2+...
    p02(1)*y(i)^2)/(1+q10(1)*x(i)+q01(1)*y(i)+q11(1)*x(i)*y(i)+...
    q20(1)*x(i)^2+q02(1)*y(i)^2));
end;



for j=2:AnzOp+1


    for i=1:13
        if(ArtOp(j-1)==1)
            z(i)=mid(z(i)+((p00(j)+p10(j)*x(i)+p01(j)*y(i)+p11(j)*x(i)*y(i)+...
                 p20(j)*x(i)^2+p02(j)*y(i)^2)/(1+q10(j)*x(i)+q01(j)*y(i)+...
                 q11(j)*x(i)*y(i)+q20(j)*x(i)^2+q02(j)*y(i)^2)));
        elseif(ArtOp(j-1)==2)
            z(i)=mid(z(i)-((p00(j)+p10(j)*x(i)+p01(j)*y(i)+p11(j)*x(i)*y(i)+...
                 p20(j)*x(i)^2+p02(j)*y(i)^2)/(1+q10(j)*x(i)+q01(j)*y(i)+...
                 q11(j)*x(i)*y(i)+q20(j)*x(i)^2+q02(j)*y(i)^2)));
        elseif(ArtOp(j-1)==3)
            z(i)=mid(z(i)*((p00(j)+p10(j)*x(i)+p01(j)*y(i)+p11(j)*x(i)*y(i)+...
                 p20(j)*x(i)^2+p02(j)*y(i)^2)/(1+q10(j)*x(i)+q01(j)*y(i)+...
                 q11(j)*x(i)*y(i)+q20(j)*x(i)^2+q02(j)*y(i)^2)));
        elseif(ArtOp(j-1)==4)
            z(i)=mid(z(i)/((p00(j)+p10(j)*x(i)+p01(j)*y(i)+p11(j)*x(i)*y(i)+...
                 p20(j)*x(i)^2+p02(j)*y(i)^2)/(1+q10(j)*x(i)+q01(j)*y(i)+...
                 q11(j)*x(i)*y(i)+q20(j)*x(i)^2+q02(j)*y(i)^2)));
        end;
    end;



    %Berechnung der Koeffizienten mittels CVX
    [p00(2+AnzOp),p10(2+AnzOp),p01(2+AnzOp),p20(2+AnzOp),p11(2+AnzOp),p02(2+AnzOp),...
        q10(2+AnzOp),q01(2+AnzOp),q20(2+AnzOp),q11(2+AnzOp),q02(2+AnzOp)] = ...
        zweiDKoeffCVX...
        ( x(1),x(2),x(3),x(4),x(5),x(6),x(7),x(8),x(9),x(10),x(11),x(12),x(13),...
          y(1),y(2),y(3),y(4),y(5),y(6),y(7),y(8),y(9),y(10),y(11),y(12),y(13),...
          z(1),z(2),z(3),z(4),z(5),z(6),z(7),z(8),z(9),z(10),z(11),z(12),z(13),...
          -Jx,Jx,-Jy,Jy);

    %Berechnung des Fehlerintervalls:
    %Anstatt des Algorithmus zweiDfehlerslope könnte man auch
    %andere Algorithmen zum Bestimmen des Restgliedes verwenden,zb.:
    %zweiDfehlermult,
    %zweiDFehlermitzerteilung,
    %zweiDfehler,
```

```
137     %zweiDfehlerslopemult.
138
139     [u(2+AnzOp),o(2+AnzOp)] = zweiDfehlerslope ...
140         (p00(1),p10(1),p01(1),p20(1),p11(1),p02(1),...
141          q10(1),q01(1),q20(1),q11(1),q02(1),u(1),o(1),ArtOp(j-1),...
142          p00(j),p10(j),p01(j),p20(j),p11(j),p02(j),...
143          q10(j),q01(j),q20(j),q11(j),q02(j),u(j),o(j),...
144          p00(2+AnzOp),p10(2+AnzOp),p01(2+AnzOp),p20(2+AnzOp),p11(2+AnzOp),...
145          p02(2+AnzOp),q10(2+AnzOp),q01(2+AnzOp),q20(2+AnzOp),q11(2+AnzOp),...
146          q02(2+AnzOp),-Jx,Jx,-Jy,Jy);
147
148
149
150     %Es wird das Fehlerintervall auf die Form [-c,c] gebracht:
151     f=(o(2+AnzOp)+u(2+AnzOp))/2;
152     p00(2+AnzOp)=mid(p00(2+AnzOp)+f);
153     o(2+AnzOp)=o(2+AnzOp)-f;
154     u(2+AnzOp)=u(2+AnzOp)-f;
155
156
157     p00(1)=p00(2+AnzOp);
158     p10(1)=p10(2+AnzOp);
159     p01(1)=p01(2+AnzOp);
160     p11(1)=p11(2+AnzOp);
161     p20(1)=p20(2+AnzOp);
162     p02(1)=p02(2+AnzOp);
163     q10(1)=q10(2+AnzOp);
164     q01(1)=q01(2+AnzOp);
165     q20(1)=q20(2+AnzOp);
166     q11(1)=q11(2+AnzOp);
167     q02(1)=q02(2+AnzOp);
168     u(1)=u(2+AnzOp);
169     o(1)=o(2+AnzOp);
170
171 end;
172
173
174
175 a00=mid(p00(2+AnzOp));
176 a10=mid(p10(2+AnzOp));
177 a01=mid(p01(2+AnzOp));
178 a11=mid(p11(2+AnzOp));
179 a20=mid(p20(2+AnzOp));
180 a02=mid(p02(2+AnzOp));
181 b10=mid(q10(2+AnzOp));
182 b01=mid(q01(2+AnzOp));
183 b20=mid(q20(2+AnzOp));
184 b11=mid(q11(2+AnzOp));
185 b02=mid(q02(2+AnzOp));
186 u=u(2+AnzOp);
187 o=o(2+AnzOp);
```

A.11 zweiDKoeffCVX.m

```
function [p00,p10,p01,p20,p11,p02,q10,q01,q20,q11,q02] = ...
zweiDKoeffCVX(x1,x2,x3,x4,x5,x6,x7,x8,x9,x10,x11,x12,x13,...
                       y1,y2,y3,y4,y5,y6,y7,y8,y9,y10,y11,y12,y13,...
                       z1,z2,z3,z4,z5,z6,z7,z8,z9,z10,z11,z12,z13,...
                       Ixu,Ixo,Iyu,Iyo)

Ix=infsup(Ixu,Ixo);
Iy=infsup(Iyu,Iyo);

c=-0.1;              %Zur "Verschärfung" der Nebenbedingungen
I=infsup(-1,0);      %Startwert für die Intervallabschätzung des Polynoms q

while(inf_(I)<0)     %Prüft ob die Nebenbedingungen "streng genug" sind
    c=c+0.1;

    cvx_begin        %Lösen des Optimierungsproblems mittels CVX
        variable a00
        variable a10
        variable a01
        variable a11
        variable a20
        variable a02

        variable b10
        variable b01
        variable b11
        variable b20
        variable b02

        minimize(...
        (a00+a10*x1+a01*y1+a11*x1*y1+a20*x1^2+a02*y1^2-(  z1  )*...
        (1+b10*x1+b01*y1+b11*x1*y1+b20*x1^2+b02*y1^2))^2 +...
        (a00+a10*x2+a01*y2+a11*x2*y2+a20*x2^2+a02*y2^2-(  z2  )*...
        (1+b10*x2+b01*y2+b11*x2*y2+b20*x2^2+b02*y2^2))^2 +...
        (a00+a10*x3+a01*y3+a11*x3*y3+a20*x3^2+a02*y3^2-( z3  )*...
        (1+b10*x3+b01*y3+b11*x3*y3+b20*x3^2+b02*y3^2))^2 +...
        (a00+a10*x4+a01*y4+a11*x4*y4+a20*x4^2+a02*y4^2-(  z4 )*...
        (1+b10*x4+b01*y4+b11*x4*y4+b20*x4^2+b02*y4^2))^2 +...
        (a00+a10*x5+a01*y5+a11*x5*y5+a20*x5^2+a02*y5^2-(  z5  )*...
        (1+b10*x5+b01*y5+b11*x5*y5+b20*x5^2+b02*y5^2))^2 +...
        (a00+a10*x6+a01*y6+a11*x6*y6+a20*x6^2+a02*y6^2-(  z6  )*...
        (1+b10*x6+b01*y6+b11*x6*y6+b20*x6^2+b02*y6^2))^2 +...
        (a00+a10*x7+a01*y7+a11*x7*y7+a20*x7^2+a02*y7^2-(  z7  )*...
        (1+b10*x7+b01*y7+b11*x7*y7+b20*x7^2+b02*y7^2))^2 +...
        (a00+a10*x8+a01*y8+a11*x8*y8+a20*x8^2+a02*y8^2-(  z8  )*...
        (1+b10*x8+b01*y8+b11*x8*y8+b20*x8^2+b02*y8^2))^2 +...
        (a00+a10*x9+a01*y9+a11*x9*y9+a20*x9^2+a02*y9^2-(  z9  )*...
        (1+b10*x9+b01*y9+b11*x9*y9+b20*x9^2+b02*y9^2))^2 +...
        (a00+a10*x10+a01*y10+a11*x10*y10+a20*x10^2+a02*y10^2-(  z10 )*...
        (1+b10*x10+b01*y10+b11*x10*y10+b20*x10^2+b02*y10^2))^2 +...
```

```
51          (a00+a10*x11+a01*y11+a11*x11*y11+a20*x11^2+a02*y11^2-(  z11  )*...
52          (1+b10*x11+b01*y11+b11*x11*y11+b20*x11^2+b02*y11^2))^2 +...
53          (a00+a10*x12+a01*y12+a11*x12*y12+a20*x12^2+a02*y12^2-(  z12  )*...
54          (1+b10*x12+b01*y12+b11*x12*y12+b20*x12^2+b02*y12^2))^2 +...
55          (a00+a10*x13+a01*y13+a11*x13*y13+a20*x13^2+a02*y13^2-(  z13 )*...
56          (1+b10*x13+b01*y13+b11*x13*y13+b20*x13^2+b02*y13^2))^2 )
57
58          subject to
59
60          (1+b10*x1+b01*y1+b11*x1*y1+b20*x1^2+b02*y1^2)>c
61          (1+b10*x2+b01*y2+b11*x2*y2+b20*x2^2+b02*y2^2)>c
62          (1+b10*x3+b01*y3+b11*x3*y3+b20*x3^2+b02*y3^2)>c
63          (1+b10*x4+b01*y4+b11*x4*y4+b20*x4^2+b02*y4^2)>c
64          (1+b10*x5+b01*y5+b11*x5*y5+b20*x5^2+b02*y5^2)>c
65          (1+b10*x6+b01*y6+b11*x6*y6+b20*x6^2+b02*y6^2)>c
66          (1+b10*x7+b01*y7+b11*x7*y7+b20*x7^2+b02*y7^2)>c
67          (1+b10*x8+b01*y8+b11*x8*y8+b20*x8^2+b02*y8^2)>c
68          (1+b10*x9+b01*y9+b11*x9*y9+b20*x9^2+b02*y9^2)>c
69          (1+b10*x10+b01*y10+b11*x10*y10+b20*x10^2+b02*y10^2)>c
70          (1+b10*x11+b01*y11+b11*x11*y11+b20*x11^2+b02*y11^2)>c
71          (1+b10*x12+b01*y12+b11*x12*y12+b20*x12^2+b02*y12^2)>c
72          (1+b10*x13+b01*y13+b11*x13*y13+b20*x13^2+b02*y13^2)>c
73
74      cvx_end
75
76      q10=infsup(b10,b10);
77      q01=infsup(b01,b01);
78      q20=infsup(b20,b20);
79      q11=infsup(b11,b11);
80      q02=infsup(b02,b02);
81
82      %Intervallabschätzung von q
83      I=(1+q10*Ix+q01*Iy+q20*power(Ix,2)+q11*Ix*Iy+q02*power(Iy,2));
84
85  end
86
87  p00=infsup(a00,a00);
88  p10=infsup(a10,a10);
89  p01=infsup(a01,a01);
90  p20=infsup(a20,a20);
91  p11=infsup(a11,a11);
92  p02=infsup(a02,a02);
```

A.12 zweiDfehlerslope.m

```
1  function [u,o] = zweiDfehlerslope ...
2      (pa00,pa10,pa01,pa20,pa11,pa02,qa10,qa01,qa20,qa11,qa02,ua,oa,operation,...
3       pb00,pb10,pb01,pb20,pb11,pb02,qb10,qb01,qb20,qb11,qb02,ub,ob,...
4       p00,p10,p01,p20,p11,p02,q10,q01,q20,q11,q02,Ixu,Ixo,Iyu,Iyo)
5
6  za1=Ixu;
```

```
7  za2=Iyu;
8  zb1=Ixo;
9  zb2=Iyo;
10
11 Ix=infsup(Ixu,Ixo);
12 Iy=infsup(Iyu,Iyo);
13 Ia=infsup(ua,oa);
14 Ib=infsup(ub,ob);
15
16 %Berechnung der Slopes erster und zweiter Ordnung (f[za,x],f[za,zb,x])
17 [sa00za,sa10za,sa01za,sb00za,sb01za,s11za,s12za,s21za,s22za] = ...
18     slope2(pa00,pa10,pa01,pa20,pa11,pa02,qa10,qa01,qa20,qa11,qa02,Ia,...
19            pb00,pb10,pb01,pb20,pb11,pb02,qb10,qb01,qb20,qb11,qb02,Ia,...
20            p00,p10,p01,p20,p11,p02,q10,q01,q20,q11,q02,operation,...
21            za1,za2,zb1,zb2,Ixu,Ixo,Iyu,Iyo);
22
23 %Berechnung der Slopes erster und zweiter Ordnung (f[zb,x],f[zb,za,x])
24 [sa00zb,sa10zb,sa01zb,sb00zb,sb01zb,s11zb,s12zb,s21zb,s22zb] = ...
25     slope2(pa00,pa10,pa01,pa20,pa11,pa02,qa10,qa01,qa20,qa11,qa02,Ia,...
26            pb00,pb10,pb01,pb20,pb11,pb02,qb10,qb01,qb20,qb11,qb02,Ib,...
27            p00,p10,p01,p20,p11,p02,q10,q01,q20,q11,q02,operation,...
28            zb1,zb2,za1,za2,Ixu,Ixo,Iyu,Iyo);
29
30 %Berechnung von f[za,zb]
31 fzazb=[sa00za+sa10za*zb1+sa01za*zb2,sb00za+sb01za*zb2];
32
33
34 if(operation==1) %Addition
35
36     fza=((pa00+pa10*za1+pa01*za2+pa20*za1^2+pa11*za1*za2+pa02*za2^2+Ia)/...
37         (1+qa10*za1+qa01*za2+qa20*za1^2+qa11*za1*za2+qa02*za2^2)+...
38         (pb00+pb10*za1+pb01*za2+pb20*za1^2+pb11*za1*za2+pb02*za2^2+Ib)/...
39         (1+qb10*za1+qb01*za2+qb20*za1^2+qb11*za1*za2+qb02*za2^2))*...
40         (1+q10*za1+q01*za2+q20*za1^2+q11*za1*za2+q02*za2^2)-...
41         (p00+p10*za1+p01*za2+p20*za1^2+p11*za1*za2+p02*za2^2);
42
43     fzb=((pa00+pa10*zb1+pa01*zb2+pa20*zb1^2+pa11*zb1*zb2+pa02*zb2^2+Ia)/...
44         (1+qa10*zb1+qa01*zb2+qa20*zb1^2+qa11*zb1*zb2+qa02*zb2^2)+...
45         (pb00+pb10*zb1+pb01*zb2+pb20*zb1^2+pb11*zb1*zb2+pb02*zb2^2+Ib)/...
46         (1+qb10*zb1+qb01*zb2+qb20*zb1^2+qb11*zb1*zb2+qb02*zb2^2))*...
47         (1+q10*zb1+q01*zb2+q20*zb1^2+q11*zb1*zb2+q02*zb2^2)-...
48         (p00+p10*zb1+p01*zb2+p20*zb1^2+p11*zb1*zb2+p02*zb2^2);
49
50     %Einsetzen des Defintionsbereiches in f(x)
51     I1=((pa00+pa10*Ix+pa01*Iy+pa20*power(Ix,2)+pa11*Ix*Iy+...
52          pa02*power(Iy,2)+Ia)/...
53         (1+qa10*Ix+qa01*Iy+qa20*power(Ix,2)+qa11*Ix*Iy+qa02*power(Iy,2))+...
54         (pb00+pb10*Ix+pb01*Iy+pb20*power(Ix,2)+pb11*Ix*Iy+...
55          pb02*power(Iy,2)+Ib)/...
56         (1+qb10*Ix+qb01*Iy+qb20*power(Ix,2)+qb11*Ix*Iy+qb02*power(Iy,2)))*...
57         (1+q10*Ix+q01*Iy+q20*power(Ix,2)+q11*Ix*Iy+q02*power(Iy,2))-...
58         (p00+p10*Ix+p01*Iy+p20*power(Ix,2)+p11*Ix*Iy+p02*power(Iy,2));
59
60     %Abschätzen des Fehlerintervalls mittels Slopes erster Ordnung
```

```
61      %an den Punkte za bzw. zb.
62      I2=fza+[sa00za+sa10za*Ix+sa01za*Iy,sb00za+sb01za*Iy]*[Ix-za1;Iy-za2];
63      I3=fzb+[sa00zb+sa10zb*Ix+sa01zb*Iy,sb00zb+sb01zb*Iy]*[Ix-zb1;Iy-zb2];
64      %Abschätzen des Fehlerintervalls mittels Slopes zweiter Ordnung
65      I4=fza+(fzazb+[Ix-zb1;Iy-zb2]'*[s11za,s12za;s21za,s22za])*[Ix-za1;Iy-za2];
66
67      %Schneiden der Intervalle
68      I5=intersect(intersect(I1,I2),intersect(I3,I4));
69
70  elseif(operation==2) %Subtraktion
71
72      fza=((pa00+pa10*za1+pa01*za2+pa20*za1^2+pa11*za1*za2+pa02*za2^2+Ia)/...
73          (1+qa10*za1+qa01*za2+qa20*za1^2+qa11*za1*za2+qa02*za2^2)-...
74          (pb00+pb10*za1+pb01*za2+pb20*za1^2+pb11*za1*za2+pb02*za2^2+Ib)/...
75          (1+qb10*za1+qb01*za2+qb20*za1^2+qb11*za1*za2+qb02*za2^2))*...
76          (1+q10*za1+q01*za2+q20*za1^2+q11*za1*za2+q02*za2^2)-...
77          (p00+p10*za1+p01*za2+p20*za1^2+p11*za1*za2+p02*za2^2);
78
79      fzb=((pa00+pa10*zb1+pa01*zb2+pa20*zb1^2+pa11*zb1*zb2+pa02*zb2^2+Ia)/...
80          (1+qa10*zb1+qa01*zb2+qa20*zb1^2+qa11*zb1*zb2+qa02*zb2^2)-...
81          (pb00+pb10*zb1+pb01*zb2+pb20*zb1^2+pb11*zb1*zb2+pb02*zb2^2+Ib)/...
82          (1+qb10*zb1+qb01*zb2+qb20*zb1^2+qb11*zb1*zb2+qb02*zb2^2))*...
83          (1+q10*zb1+q01*zb2+q20*zb1^2+q11*zb1*zb2+q02*zb2^2)-...
84          (p00+p10*zb1+p01*zb2+p20*zb1^2+p11*zb1*zb2+p02*zb2^2);
85
86      I1=((pa00+pa10*Ix+pa01*Iy+pa20*power(Ix,2)+pa11*Ix*Iy+...
87          pa02*power(Iy,2)+Ia)/...
88          (1+qa10*Ix+qa01*Iy+qa20*power(Ix,2)+qa11*Ix*Iy+qa02*power(Iy,2))-...
89          (pb00+pb10*Ix+pb01*Iy+pb20*power(Ix,2)+pb11*Ix*Iy+...
90          pb02*power(Iy,2)+Ib)/...
91          (1+qb10*Ix+qb01*Iy+qb20*power(Ix,2)+qb11*Ix*Iy+qb02*power(Iy,2)))*...
92          (1+q10*Ix+q01*Iy+q20*power(Ix,2)+q11*Ix*Iy+q02*power(Iy,2))-...
93          (p00+p10*Ix+p01*Iy+p20*power(Ix,2)+p11*Ix*Iy+p02*power(Iy,2));
94
95      I2=fza+[sa00za+sa10za*Ix+sa01za*Iy,sb00za+sb01za*Iy]*[Ix-za1;Iy-za2];
96      I3=fzb+[sa00zb+sa10zb*Ix+sa01zb*Iy,sb00zb+sb01zb*Iy]*[Ix-zb1;Iy-zb2];
97      I4=fza+(fzazb+[Ix-zb1;Iy-zb2]'*[s11za,s12za;s21za,s22za])*[Ix-za1;Iy-za2];
98
99      I5=intersect(intersect(I1,I2),intersect(I3,I4));
100
101 elseif(operation==3) %Multiplikation
102
103     fza=((pa00+pa10*za1+pa01*za2+pa20*za1^2+pa11*za1*za2+pa02*za2^2+Ia)/...
104         (1+qa10*za1+qa01*za2+qa20*za1^2+qa11*za1*za2+qa02*za2^2)*...
105         (pb00+pb10*za1+pb01*za2+pb20*za1^2+pb11*za1*za2+pb02*za2^2+Ib)/...
106         (1+qb10*za1+qb01*za2+qb20*za1^2+qb11*za1*za2+qb02*za2^2))*...
107         (1+q10*za1+q01*za2+q20*za1^2+q11*za1*za2+q02*za2^2)-...
108         (p00+p10*za1+p01*za2+p20*za1^2+p11*za1*za2+p02*za2^2);
109
110     fzb=((pa00+pa10*zb1+pa01*zb2+pa20*zb1^2+pa11*zb1*zb2+pa02*zb2^2+Ia)/...
111         (1+qa10*zb1+qa01*zb2+qa20*zb1^2+qa11*zb1*zb2+qa02*zb2^2)*...
112         (pb00+pb10*zb1+pb01*zb2+pb20*zb1^2+pb11*zb1*zb2+pb02*zb2^2+Ib)/...
113         (1+qb10*zb1+qb01*zb2+qb20*zb1^2+qb11*zb1*zb2+qb02*zb2^2))*...
114         (1+q10*zb1+q01*zb2+q20*zb1^2+q11*zb1*zb2+q02*zb2^2)-...
```

```
115         (p00+p10*zb1+p01*zb2+p20*zb1^2+p11*zb1*zb2+p02*zb2^2);
116
117     I1=((pa00+pa10*Ix+pa01*Iy+pa20*power(Ix,2)+pa11*Ix*Iy+...
118         pa02*power(Iy,2)+Ia)/...
119         (1+qa10*Ix+qa01*Iy+qa20*power(Ix,2)+qa11*Ix*Iy+qa02*power(Iy,2))*...
120         (pb00+pb10*Ix+pb01*Iy+pb20*power(Ix,2)+pb11*Ix*Iy+...
121         pb02*power(Iy,2)+Ib)/...
122         (1+qb10*Ix+qb01*Iy+qb20*power(Ix,2)+qb11*Ix*Iy+qb02*power(Iy,2)))*...
123         (1+q10*Ix+q01*Iy+q20*power(Ix,2)+q11*Ix*Iy+q02*power(Iy,2))-...
124         (p00+p10*Ix+p01*Iy+p20*power(Ix,2)+p11*Ix*Iy+p02*power(Iy,2));
125
126     I2=fza+[sa00za+sa10za*Ix+sa01za*Iy,sb00za+sb01za*Iy]*[Ix-za1;Iy-za2];
127     I3=fzb+[sa00zb+sa10zb*Ix+sa01zb*Iy,sb00zb+sb01zb*Iy]*[Ix-zb1;Iy-zb2];
128     I4=fza+(fzazb+[Ix-zb1;Iy-zb2]'*[s11za,s12za;s21za,s22za])*[Ix-za1;Iy-za2];
129
130     I5=intersect(intersect(I1,I2),intersect(I3,I4));
131
132
133 elseif(operation==4) %Division
134
135     fza=((pa00+pa10*za1+pa01*za2+pa20*za1^2+pa11*za1*za2+pa02*za2^2+Ia)/...
136         (1+qa10*za1+qa01*za2+qa20*za1^2+qa11*za1*za2+qa02*za2^2)/...
137         (pb00+pb10*za1+pb01*za2+pb20*za1^2+pb11*za1*za2+pb02*za2^2+Ib)/...
138         (1+qb10*za1+qb01*za2+qb20*za1^2+qb11*za1*za2+qb02*za2^2))*...
139         (1+q10*za1+q01*za2+q20*za1^2+q11*za1*za2+q02*za2^2)-...
140         (p00+p10*za1+p01*za2+p20*za1^2+p11*za1*za2+p02*za2^2);
141
142     fzb=((pa00+pa10*zb1+pa01*zb2+pa20*zb1^2+pa11*zb1*zb2+pa02*zb2^2+Ia)/...
143         (1+qa10*zb1+qa01*zb2+qa20*zb1^2+qa11*zb1*zb2+qa02*zb2^2)/...
144         (pb00+pb10*zb1+pb01*zb2+pb20*zb1^2+pb11*zb1*zb2+pb02*zb2^2+Ib)/...
145         (1+qb10*zb1+qb01*zb2+qb20*zb1^2+qb11*zb1*zb2+qb02*zb2^2))*...
146         (1+q10*zb1+q01*zb2+q20*zb1^2+q11*zb1*zb2+q02*zb2^2)-...
147         (p00+p10*zb1+p01*zb2+p20*zb1^2+p11*zb1*zb2+p02*zb2^2);
148
149     I1=((pa00+pa10*Ix+pa01*Iy+pa20*power(Ix,2)+pa11*Ix*Iy+...
150         pa02*power(Iy,2)+Ia)/...
151         (1+qa10*Ix+qa01*Iy+qa20*power(Ix,2)+qa11*Ix*Iy+qa02*power(Iy,2))/...
152         (pb00+pb10*Ix+pb01*Iy+pb20*power(Ix,2)+pb11*Ix*Iy+...
153         pb02*power(Iy,2)+Ib)/...
154         (1+qb10*Ix+qb01*Iy+qb20*power(Ix,2)+qb11*Ix*Iy+qb02*power(Iy,2)))*...
155         (1+q10*Ix+q01*Iy+q20*power(Ix,2)+q11*Ix*Iy+q02*power(Iy,2))-...
156         (p00+p10*Ix+p01*Iy+p20*power(Ix,2)+p11*Ix*Iy+p02*power(Iy,2));
157
158     I2=fza+[sa00za+sa10za*Ix+sa01za*Iy,sb00za+sb01za*Iy]*[Ix-za1;Iy-za2];
159     I3=fzb+[sa00zb+sa10zb*Ix+sa01zb*Iy,sb00zb+sb01zb*Iy]*[Ix-zb1;Iy-zb2];
160     I4=fza+(fzazb+[Ix-zb1;Iy-zb2]'*[s11za,s12za;s21za,s22za])*[Ix-za1;Iy-za2];
161
162     I5=intersect(intersect(I1,I2),intersect(I3,I4));
163
164 end;
165
166 u=inf_(I5);
167 o=sup(I5);
```

A.13 slope2.m

```
function [sa00,sa10,sa01,sb00,sb01,s11,s12,s21,s22] = ...
    slope2(pa00,pa10,pa01,pa20,pa11,pa02,qa10,qa01,qa20,qa11,qa02,Ia,...
           pb00,pb10,pb01,pb20,pb11,pb02,qb10,qb01,qb20,qb11,qb02,Ib,...
           p00,p10,p01,p20,p11,p02,q10,q01,q20,q11,q02,operation,...
           za1,za2,zb1,zb2,Ixu,Ixo,Iyu,Iyo)


%Definitionsbereich
Ix=infsup(Ixu,Ixo);
Iy=infsup(Iyu,Iyo);

%Berechnung von Slopes erster Ordnung der verschiedenen Polynome
[pazaa00,pazaa10,pazaa01,pazab00,pazab01,paza11,paza12,paza21,paza22]=...
        slopepoly2(pa00,pa10,pa01,pa20,pa11,pa02,za1,za2);
[qazaa00,qazaa10,qazaa01,qazab00,qazab01,qaza11,qaza12,qaza21,qaza22]=...
        slopepoly2(1,qa10,qa01,qa20,qa11,qa02,za1,za2);
[qazba00,qazba10,qazba01,qazbb00,qazbb01,qazb11,qazb12,qazb21,qazb22]=...
        slopepoly2(1,qa10,qa01,qa20,qa11,qa02,zb1,zb2);
[pbzaa00,pbzaa10,pbzaa01,pbzab00,pbzab01,pbza11,pbza12,pbza21,pbza22]=...
        slopepoly2(pb00,pb10,pb01,pb20,pb11,pb02,za1,za2);
[pbzba00,pbzba10,pbzba01,pbzbb00,pbzbb01,pbzb11,pbzb12,pbzb21,pbzb22]=...
        slopepoly2(pb00,pb10,pb01,pb20,pb11,pb02,zb1,zb2);
[qbzaa00,qbzaa10,qbzaa01,qbzab00,qbzab01,qbza11,qbza12,qbza21,qbza22]=...
        slopepoly2(1,qb10,qb01,qb20,qb11,qb02,za1,za2);
[qbzba00,qbzba10,qbzba01,qbzbb00,qbzbb01,qbzb11,qbzb12,qbzb21,qbzb22]=...
        slopepoly2(1,qb10,qb01,qb20,qb11,qb02,zb1,zb2);
[pzaa00,pzaa10,pzaa01,pzab00,pzab01,pza11,pza12,pza21,pza22]= ...
        slopepoly2(p00,p10,p01,p20,p11,p02,za1,za2);
[qzaa00,qzaa10,qzaa01,qzab00,qzab01,qza11,qza12,qza21,qza22]=...
        slopepoly2(1,q10,q01,q20,q11,q02,za1,za2);
[qzba00,qzba10,qzba01,qzbb00,qzbb01,qzb11,qzb12,qzb21,qzb22]=...
        slopepoly2(1,q10,q01,q20,q11,q02,zb1,zb2);


ha=(pa00+pa10*za1+pa01*za2+pa20*za1^2+pa11*za1*za2+pa02*za2^2+Ia)/...
    (1+qa10*za1+qa01*za2+qa20*za1^2+qa11*za1*za2+qa02*za2^2);
hb=(pb00+pb10*za1+pb01*za2+pb20*za1^2+pb11*za1*za2+pb02*za2^2+Ib)/...
    (1+qb10*za1+qb01*za2+qb20*za1^2+qb11*za1*za2+qb02*za2^2);

%Berechnen der Koeffizienten des Slopes erster Ordnung
%von pa/qa an der Stelle za, anhand bekannter Rechenregeln für Slopes
paqaa00=(pazaa00-(ha)*qazaa00)/(1+qa10*Ix+qa01*Iy+qa20*power(Ix,2)+...
            qa11*Ix*Iy+qa02*power(Iy,2));
paqaa10=(pazaa10-(ha)*qazaa10)/(1+qa10*Ix+qa01*Iy+qa20*power(Ix,2)+...
            qa11*Ix*Iy+qa02*power(Iy,2));
paqaa01=(pazaa01-(ha)*qazaa01)/(1+qa10*Ix+qa01*Iy+qa20*power(Ix,2)+...
            qa11*Ix*Iy+qa02*power(Iy,2));
paqab00=(pazab00-(ha)*qazab00)/(1+qa10*Ix+qa01*Iy+qa20*power(Ix,2)+...
            qa11*Ix*Iy+qa02*power(Iy,2));
paqab01=(pazab01-(ha)*qazab01)/(1+qa10*Ix+qa01*Iy+qa20*power(Ix,2)+...
```

```
            qa11*Ix*Iy+qa02*power(Iy,2));

%Berechnen der Koeffizienten des Slopes erster Ordnung
%von pb/qb an der Stelle za, anhand bekannter Rechenregeln für Slopes
pbqba00=(pbzaa00-(hb)*qbzaa00)/(1+qb10*Ix+qb01*Iy+qb20*power(Ix,2)+...
            qb11*Ix*Iy+qb02*power(Iy,2));
pbqba10=(pbzaa10-(hb)*qbzaa10)/(1+qb10*Ix+qb01*Iy+qb20*power(Ix,2)+...
            qb11*Ix*Iy+qb02*power(Iy,2));
pbqba01=(pbzaa01-(hb)*qbzaa01)/(1+qb10*Ix+qb01*Iy+qb20*power(Ix,2)+...
            qb11*Ix*Iy+qb02*power(Iy,2));
pbqbb00=(pbzab00-(hb)*qbzab00)/(1+qb10*Ix+qb01*Iy+qb20*power(Ix,2)+...
            qb11*Ix*Iy+qb02*power(Iy,2));
pbqbb01=(pbzab01-(hb)*qbzab01)/(1+qb10*Ix+qb01*Iy+qb20*power(Ix,2)+...
            qb11*Ix*Iy+qb02*power(Iy,2));

%Berechnen der Koeffizienten des Slopes erster Ordnung
%von pb/qb an der Stelle zb, anhand bekannter Rechenregeln für Slopes
pbqbzba00=(pbzba00-(hb)*qbzba00)/(1+qb10*Ix+qb01*Iy+qb20*power(Ix,2)+...
            qb11*Ix*Iy+qb02*power(Iy,2));
pbqbzba10=(pbzba10-(hb)*qbzba10)/(1+qb10*Ix+qb01*Iy+qb20*power(Ix,2)+...
            qb11*Ix*Iy+qb02*power(Iy,2));
pbqbzba01=(pbzba01-(hb)*qbzba01)/(1+qb10*Ix+qb01*Iy+qb20*power(Ix,2)+...
            qb11*Ix*Iy+qb02*power(Iy,2));
pbqbzbb00=(pbzbb00-(hb)*qbzbb00)/(1+qb10*Ix+qb01*Iy+qb20*power(Ix,2)+...
            qb11*Ix*Iy+qb02*power(Iy,2));
pbqbzbb01=(pbzbb01-(hb)*qbzbb01)/(1+qb10*Ix+qb01*Iy+qb20*power(Ix,2)+...
            qb11*Ix*Iy+qb02*power(Iy,2));


hbzb=(pb00+pb10*zb1+pb01*zb2+pb20*zb1^2+pb11*zb1*zb2+pb02*zb2^2+Ib)/...
        (1+qb10*zb1+qb01*zb2+qb20*zb1^2+qb11*zb1*zb2+qb02*zb2^2);

%Berechne die Koeffizienten des Slopes zweiter Ordnung
%von pa/qa[za,zb,x]
paqa2s11=(paza11-(ha)*qaza11+(qazba00+qazba10*Ix+qazba01*Iy)*...
  ((qazaa00+qazaa10*zb1+qazaa01*zb2)*ha-(pazaa00+pazaa10*zb1+pazaa01*zb2))/...
  (1+qa10*zb1+qa01*zb2+qa20*zb1^2+qa11*zb1*zb2+qa02*zb2^2))/...
  (1+qa10*Ix+qa01*Iy+qa20*power(Ix,2)+qa11*Ix*Iy+qa02*power(Iy,2));

paqa2s12=(paza12-(ha)*qaza12+(qazba00+qazba10*Ix+qazba01*Iy)*...
    ((qazab00+qazab01*zb2)*ha-(pazab00+pazab01*zb2))/...
    (1+qa10*zb1+qa01*zb2+qa20*zb1^2+qa11*zb1*zb2+qa02*zb2^2))/...
    (1+qa10*Ix+qa01*Iy+qa20*power(Ix,2)+qa11*Ix*Iy+qa02*power(Iy,2));

paqa2s21=(paza21-(ha)*qaza21+(qazbb00+qazbb01*Iy)*...
    ((qazaa00+qazaa10*zb1+qazaa01*zb2)*ha-...
    (pazaa00+pazaa10*zb1+pazaa01*zb2))/...
    (1+qa10*zb1+qa01*zb2+qa20*zb1^2+qa11*zb1*zb2+qa02*zb2^2))/...
    (1+qa10*Ix+qa01*Iy+qa20*power(Ix,2)+qa11*Ix*Iy+qa02*power(Iy,2));

paqa2s22=(paza22-(ha)*qaza22+(qazbb00+qazbb01*Iy)*...
    ((qazab00+qazab01*zb2)*ha-(pazab00+pazab01*zb2))/...
    (1+qa10*zb1+qa01*zb2+qa20*zb1^2+qa11*zb1*zb2+qa02*zb2^2))/...
    (1+qa10*Ix+qa01*Iy+qa20*power(Ix,2)+qa11*Ix*Iy+qa02*power(Iy,2));
```

```
105
106
107 %Berechne die Koeffizienten des Slopes zweiter Ordnung
108 %von pb/qb[za,zb,x]
109 pbqb2s11=(pbza11-(hb)*qbza11+(qbzba00+qbzba10*Ix+qbzba01*Iy)*...
110     ((qbzaa00+qbzaa10*zb1+qbzaa01*zb2)*hb-...
111     (pbzaa00+pbzaa10*zb1+pbzaa01*zb2))/...
112     (1+qb10*zb1+qb01*zb2+qb20*zb1^2+qb11*zb1*zb2+qb02*zb2^2))/...
113     (1+qb10*Ix+qb01*Iy+qb20*power(Ix,2)+qb11*Ix*Iy+qb02*power(Iy,2));
114
115 pbqb2s12=(pbza12-(hb)*qbza12+(qbzba00+qbzba10*Ix+qbzba01*Iy)*...
116     ((qbzab00+qbzab01*zb2)*hb-(pbzab00+pbzab01*zb2))/...
117     (1+qb10*zb1+qb01*zb2+qb20*zb1^2+qb11*zb1*zb2+qb02*zb2^2))/...
118     (1+qb10*Ix+qb01*Iy+qb20*power(Ix,2)+qb11*Ix*Iy+qb02*power(Iy,2));
119
120 pbqb2s21=(pbza21-(hb)*qbza21+(qbzbb00+qbzbb01*Iy)*...
121     ((qbzaa00+qbzaa10*zb1+qbzaa01*zb2)*hb-...
122     (pbzaa00+pbzaa10*zb1+pbzaa01*zb2))/...
123     (1+qb10*zb1+qb01*zb2+qb20*zb1^2+qb11*zb1*zb2+qb02*zb2^2))/...
124     (1+qb10*Ix+qb01*Iy+qb20*power(Ix,2)+qb11*Ix*Iy+qb02*power(Iy,2));
125
126 pbqb2s22=(pbza22-(hb)*qbza22+(qbzba00+qbzba10*Ix+qbzba01*Iy)*...
127     ((qbzab00+qbzab01*zb2)*hb-(pbzab00+pbzab01*zb2))/...
128     (1+qb10*zb1+qb01*zb2+qb20*zb1^2+qb11*zb1*zb2+qb02*zb2^2))/...
129     (1+qb10*Ix+qb01*Iy+qb20*power(Ix,2)+qb11*Ix*Iy+qb02*power(Iy,2));
130
131
132 %Es folgen die Berechnungen der Slopes erster und zweiter Ordnung
133 %für die verschiedenen Operationen anhand den bekannten Rechenregeln
134 %für Slopes:
135
136 if(operation==1)
137     a00=paqaa00+pbqba00;
138     a10=paqaa10+pbqba10;
139     a01=paqaa01+pbqba01;
140     b00=paqab00+pbqbb00;
141     b01=paqab01+pbqbb01;
142
143
144     c00=a00*(1+q10*Ix+q01*Iy+q20*power(Ix,2)+q11*Ix*Iy+q02*power(Iy,2))+...
145         (ha+hb)*qzaa00;
146     c10=a10*(1+q10*Ix+q01*Iy+q20*power(Ix,2)+q11*Ix*Iy+q02*power(Iy,2))+...
147         (ha+hb)*qzaa10;
148     c01=a01*(1+q10*Ix+q01*Iy+q20*power(Ix,2)+q11*Ix*Iy+q02*power(Iy,2))+...
149         (ha+hb)*qzaa01;
150     d00=b00*(1+q10*Ix+q01*Iy+q20*power(Ix,2)+q11*Ix*Iy+q02*power(Iy,2))+...
151         (ha+hb)*qzab00;
152     d01=b01*(1+q10*Ix+q01*Iy+q20*power(Ix,2)+q11*Ix*Iy+q02*power(Iy,2))+...
153         (ha+hb)*qzab01;
154
155
156     sa00=c00-pzaa00;
157     sa10=c10-pzaa10;
158     sa01=c01-pzaa01;
```

```
159     sb00=d00-pzab00;
160     sb01=d01-pzab01;
161
162
163     E(1,1)=paqa2s11+pbqb2s11;
164     E(1,2)=paqa2s12+pbqb2s12;
165     E(2,1)=paqa2s21+pbqb2s21;
166     E(2,2)=paqa2s22+pbqb2s22;
167
168
169
170     E=E*(1+q10*Ix+q01*Iy+q20*power(Ix,2)+q11*Ix*Iy+q02*power(Iy,2))+...
171         (ha+hb)*[qza11,qza12;qza21,qza22]+...
172         [a00+a10*zb1+a01*zb2,b00+b01*zb2]'*...
173         [qzba00+qzba10*Ix+qzba01*Iy,qzbb00+qzbb01*Iy];
174
175     E=E-[pza11,pza12;pza21,pza22];
176
177     s11=E(1,1);
178     s12=E(1,2);
179     s21=E(2,1);
180     s22=E(2,2);
181
182 elseif(operation==2)
183
184
185     a00=paqaa00-pbqba00;
186     a10=paqaa10-pbqba10;
187     a01=paqaa01-pbqba01;
188     b00=paqab00-pbqbb00;
189     b01=paqab01-pbqbb01;
190
191
192     c00=a00*(1+q10*Ix+q01*Iy+q20*power(Ix,2)+q11*Ix*Iy+q02*power(Iy,2))+...
193         (ha-hb)*qzaa00;
194     c10=a10*(1+q10*Ix+q01*Iy+q20*power(Ix,2)+q11*Ix*Iy+q02*power(Iy,2))+...
195         (ha-hb)*qzaa10;
196     c01=a01*(1+q10*Ix+q01*Iy+q20*power(Ix,2)+q11*Ix*Iy+q02*power(Iy,2))+...
197         (ha-hb)*qzaa01;
198     d00=b00*(1+q10*Ix+q01*Iy+q20*power(Ix,2)+q11*Ix*Iy+q02*power(Iy,2))+...
199         (ha-hb)*qzab00;
200     d01=b01*(1+q10*Ix+q01*Iy+q20*power(Ix,2)+q11*Ix*Iy+q02*power(Iy,2))+...
201         (ha-hb)*qzab01;
202
203
204     sa00=c00-pzaa00;
205     sa10=c10-pzaa10;
206     sa01=c01-pzaa01;
207     sb00=d00-pzab00;
208     sb01=d01-pzab01;
209
210     E=zeros(2);
211     E(1,1)=paqa2s11-pbqb2s11;
212     E(1,2)=paqa2s12-pbqb2s12;
```

```
213     E(2,1)=paqa2s21-pbqb2s21;
214     E(2,2)=paqa2s22-pbqb2s22;
215
216
217
218     E=E*(1+q10*Ix+q01*Iy+q20*power(Ix,2)+q11*Ix*Iy+q02*power(Iy,2))+...
219         (ha-hb)*[qza11,qza12;qza21,qza22]+...
220         [a00+a10*zb1+a01*zb2,b00+b01*zb2]'*...
221         [qzba00+qzba10*Ix+qzba01*Iy,qzbb00+qzbb01*Iy];
222
223     E=E-[pza11,pza12;pza21,pza22];
224
225     s11=E(1,1);
226     s12=E(1,2);
227     s21=E(2,1);
228     s22=E(2,2);
229
230
231 elseif(operation==3)
232
233
234     a00=paqaa00*(pb00+pb10*Ix+pb01*Iy+pb20*power(Ix,2)+pb11*Ix*Iy+...
235         pb02*power(Iy,2)+Ib)/(1+qb10*Ix+qb01*Iy+qb20*power(Ix,2)+...
236         qb11*Ix*Iy+qb02*power(Iy,2))+ha*pbqba00;
237     a10=paqaa10*(pb00+pb10*Ix+pb01*Iy+pb20*power(Ix,2)+pb11*Ix*Iy+...
238         pb02*power(Iy,2)+Ib)/(1+qb10*Ix+qb01*Iy+qb20*power(Ix,2)+...
239         qb11*Ix*Iy+qb02*power(Iy,2))+ha*pbqba10;
240     a01=paqaa01*(pb00+pb10*Ix+pb01*Iy+pb20*power(Ix,2)+pb11*Ix*Iy+...
241         pb02*power(Iy,2)+Ib)/(1+qb10*Ix+qb01*Iy+qb20*power(Ix,2)+...
242         qb11*Ix*Iy+qb02*power(Iy,2))+ha*pbqba01;
243     b00=paqab00*(pb00+pb10*Ix+pb01*Iy+pb20*power(Ix,2)+pb11*Ix*Iy+...
244         pb02*power(Iy,2)+Ib)/(1+qb10*Ix+qb01*Iy+qb20*power(Ix,2)+...
245         qb11*Ix*Iy+qb02*power(Iy,2))+ha*pbqbb00;
246     b01=paqab01*(pb00+pb10*Ix+pb01*Iy+pb20*power(Ix,2)+pb11*Ix*Iy+...
247         pb02*power(Iy,2)+Ib)/ (1+qb10*Ix+qb01*Iy+qb20*power(Ix,2)+...
248         qb11*Ix*Iy+qb02*power(Iy,2))+ha*pbqbb01;
249
250
251     c00=a00*(1+q10*Ix+q01*Iy+q20*power(Ix,2)+q11*Ix*Iy+q02*power(Iy,2))+...
252         (ha*hb)*qzaa00;
253     c10=a10*(1+q10*Ix+q01*Iy+q20*power(Ix,2)+q11*Ix*Iy+q02*power(Iy,2))+...
254         (ha*hb)*qzaa10;
255     c01=a01*(1+q10*Ix+q01*Iy+q20*power(Ix,2)+q11*Ix*Iy+q02*power(Iy,2))+...
256         (ha*hb)*qzaa01;
257     d00=b00*(1+q10*Ix+q01*Iy+q20*power(Ix,2)+q11*Ix*Iy+q02*power(Iy,2))+...
258         (ha*hb)*qzab00;
259     d01=b01*(1+q10*Ix+q01*Iy+q20*power(Ix,2)+q11*Ix*Iy+q02*power(Iy,2))+...
260         (ha*hb)*qzab01;
261
262
263     sa00=c00-pzaa00;
264     sa10=c10-pzaa10;
265     sa01=c01-pzaa01;
266     sb00=d00-pzab00;
```

```
267     sb01=d01-pzab01;
268
269     E=zeros(2);
270     E=E+[paqa2s11,paqa2s12;paqa2s21,paqa2s22]*(pb00+pb10*Ix+pb01*Iy+...
271         pb20*power(Ix,2)+pb11*Ix*Iy+pb02*power(Iy,2)+Ib)/...
272         (1+qb10*Ix+qb01*Iy+qb20*power(Ix,2)+qb11*Ix*Iy+qb02*power(Iy,2))+...
273         [pbqb2s11,pbqb2s12;pbqb2s21,pbqb2s22]*ha+...
274         [paqaa00+paqaa10*zb1+paqaa01*zb2,paqab00+paqab01*zb2]'*...
275         [pbqba00+pbqba10*Ix+pbqba01*Iy,pbqbb00+pbqbb01*Iy];
276
277
278
279     E=E*(1+q10*Ix+q01*Iy+q20*power(Ix,2)+q11*Ix*Iy+q02*power(Iy,2))+...
280         (ha-hb)*[qza11,qza12;qza21,qza22]+...
281         [a00+a10*zb1+a01*zb2,b00+b01*zb2]'*...
282         [qzba00+qzba10*Ix+qzba01*Iy,qzbb00+qzbb01*Iy];
283
284     E=E-[pza11,pza12;pza21,pza22];
285
286     s11=E(1,1);
287     s12=E(1,2);
288     s21=E(2,1);
289     s22=E(2,2);
290
291
292 elseif(operation==4)
293
294
295
296
297     a00=(paqaa00-(ha/hb)*pbqba00)/...
298         ((pb00+pb10*Ix+pb01*Iy+pb20*power(Ix,2)+...
299         pb11*Ix*Iy+pb02*power(Iy,2)+Ib)/...
300         (1+qb10*Ix+qb01*Iy+qb20*power(Ix,2)+qb11*Ix*Iy+qb02*power(Iy,2)));
301     a10=(paqaa10-(ha/hb)*pbqba10)/...
302         ((pb00+pb10*Ix+pb01*Iy+pb20*power(Ix,2)+...
303         pb11*Ix*Iy+pb02*power(Iy,2)+Ib)/...
304         (1+qb10*Ix+qb01*Iy+qb20*power(Ix,2)+qb11*Ix*Iy+qb02*power(Iy,2)));
305     a01=(paqaa01-(ha/hb)*pbqba01)/...
306         ((pb00+pb10*Ix+pb01*Iy+pb20*power(Ix,2)+...
307         pb11*Ix*Iy+pb02*power(Iy,2)+Ib)/...
308         (1+qb10*Ix+qb01*Iy+qb20*power(Ix,2)+qb11*Ix*Iy+qb02*power(Iy,2)));
309     b00=(paqab00-(ha/hb)*pbqbb00)/...
310         ((pb00+pb10*Ix+pb01*Iy+pb20*power(Ix,2)+...
311         pb11*Ix*Iy+pb02*power(Iy,2)+Ib)/...
312         (1+qb10*Ix+qb01*Iy+qb20*power(Ix,2)+qb11*Ix*Iy+qb02*power(Iy,2)));
313     b01=(paqab01-(ha/hb)*pbqbb01)/...
314         ((pb00+pb10*Ix+pb01*Iy+pb20*power(Ix,2)+...
315         pb11*Ix*Iy+pb02*power(Iy,2)+Ib)/...
316         (1+qb10*Ix+qb01*Iy+qb20*power(Ix,2)+qb11*Ix*Iy+qb02*power(Iy,2)));
317
318
319
320
```

```
321
322     c00=a00*(1+q10*Ix+q01*Iy+q20*power(Ix,2)+q11*Ix*Iy+q02*power(Iy,2))+...
323         (ha*hb)*qzaa00;
324     c10=a10*(1+q10*Ix+q01*Iy+q20*power(Ix,2)+q11*Ix*Iy+q02*power(Iy,2))+...
325         (ha*hb)*qzaa10;
326     c01=a01*(1+q10*Ix+q01*Iy+q20*power(Ix,2)+q11*Ix*Iy+q02*power(Iy,2))+...
327         (ha*hb)*qzaa01;
328     d00=b00*(1+q10*Ix+q01*Iy+q20*power(Ix,2)+q11*Ix*Iy+q02*power(Iy,2))+...
329         (ha*hb)*qzab00;
330     d01=b01*(1+q10*Ix+q01*Iy+q20*power(Ix,2)+q11*Ix*Iy+q02*power(Iy,2))+...
331         (ha*hb)*qzab01;
332
333
334     sa00=c00-pzaa00;
335     sa10=c10-pzaa10;
336     sa01=c01-pzaa01;
337     sb00=d00-pzab00;
338     sb01=d01-pzab01;
339
340     E=zeros(2);
341     E=E+([paqa2s11,paqa2s12;paqa2s21,paqa2s22]-(ha/hb)*...
342         [pbqb2s11,pbqb2s12;pbqb2s21,pbqb2s22]+...
343         [pbqbzba00+pbqbzba10*Ix+pbqbzba01*Iy,pbqbzbb00+pbqbzbb01*Iy]'*...
344         ([pbqba00+pbqba10*zb1+pbqba01*zb2,pbqbb00+pbqbb01*zb2]*(ha/hb)-...
345         [paqaa00+paqaa10*zb1+paqaa01*zb2,paqab00+paqab01*zb2])/...
346         ( hbzb))/...
347         ((pb00+pb10*Ix+pb01*Iy+pb20*power(Ix,2)+...
348         pb11*Ix*Iy+pb02*power(Iy,2)+Ib)/...
349         (1+qb10*Ix+qb01*Iy+qb20*power(Ix,2)+qb11*Ix*Iy+qb02*power(Iy,2)));
350
351
352     E=E*(1+q10*Ix+q01*Iy+q20*power(Ix,2)+q11*Ix*Iy+q02*power(Iy,2))+...
353         (ha-hb)*[qza11,qza12;qza21,qza22]+...
354         [a00+a10*zb1+a01*zb2,b00+b01*zb2]'*...
355         [qzba00+qzba10*Ix+qzba01*Iy,qzbb00+qzbb01*Iy];
356
357     E=E-[pza11,pza12;pza21,pza22];
358
359     s11=E(1,1);
360     s12=E(1,2);
361     s21=E(2,1);
362     s22=E(2,2);
363
364
365 end;
```

A.14 slopepoly2.m

```
1 function [sa00,sa10,sa01,sb00,sb01,s11,s12,s21,s22]=...
2                     slopepoly2(e00,e10,e01,e20,e11,e02,z1,z2)
3
```

```
sa00=e10+e20*z1;
sa10=e20;
sa01=e11;
sb00=e01+e02*z2+e11*z1;
sb01=e02;

s11=e20;
s12=0;
s21=e11;
s22=e02;
```

A.15 zweiDfehler.m

```
function [u,o] = zweiDfehler ...
    (pa00,pa10,pa01,pa20,pa11,pa02,qa10,qa01,qa20,qa11,qa02,ua,oa,operation,...
     pb00,pb10,pb01,pb20,pb11,pb02,qb10,qb01,qb20,qb11,qb02,ub,ob,...
     p00,p10,p01,p20,p11,p02,q10,q01,q20,q11,q02,Ixu,Ixo,Iyu,Iyo)

Ix=infsup(Ixu,Ixo);
Iy=infsup(Iyu,Iyo);
Ia=infsup(ua,oa);
Ib=infsup(ub,ob);

if(operation==1) %Addition

    %Einsetzen des Defintionsbereiches in f(x)
    I1=((pa00+pa10*Ix+pa01*Iy+pa20*power(Ix,2)+...
        pa11*Ix*Iy+pa02*power(Iy,2)+Ia)/...
        (1+qa10*Ix+qa01*Iy+qa20*power(Ix,2)+qa11*Ix*Iy+qa02*power(Iy,2))+...
        (pb00+pb10*Ix+pb01*Iy+pb20*power(Ix,2)+...
        pb11*Ix*Iy+pb02*power(Iy,2)+Ib)/...
        (1+qb10*Ix+qb01*Iy+qb20*power(Ix,2)+qb11*Ix*Iy+qb02*power(Iy,2)))*...
        (1+q10*Ix+q01*Iy+q20*power(Ix,2)+q11*Ix*Iy+q02*power(Iy,2))-...
        (p00+p10*Ix+p01*Iy+p20*power(Ix,2)+p11*Ix*Iy+p02*power(Iy,2));

elseif(operation==2) %Subtraktion

    I1=((pa00+pa10*Ix+pa01*Iy+pa20*power(Ix,2)+...
        pa11*Ix*Iy+pa02*power(Iy,2)+Ia)/...
        (1+qa10*Ix+qa01*Iy+qa20*power(Ix,2)+qa11*Ix*Iy+qa02*power(Iy,2))-...
        (pb00+pb10*Ix+pb01*Iy+pb20*power(Ix,2)+...
        pb11*Ix*Iy+pb02*power(Iy,2)+Ib)/...
        (1+qb10*Ix+qb01*Iy+qb20*power(Ix,2)+qb11*Ix*Iy+qb02*power(Iy,2)))*...
        (1+q10*Ix+q01*Iy+q20*power(Ix,2)+q11*Ix*Iy+q02*power(Iy,2))-...
        (p00+p10*Ix+p01*Iy+p20*power(Ix,2)+p11*Ix*Iy+p02*power(Iy,2));

elseif(operation==3) %Multiplikation

    I1=((pa00+pa10*Ix+pa01*Iy+pa20*power(Ix,2)+...
        pa11*Ix*Iy+pa02*power(Iy,2)+Ia)/...
        (1+qa10*Ix+qa01*Iy+qa20*power(Ix,2)+qa11*Ix*Iy+qa02*power(Iy,2))*...
```

```
39          (pb00+pb10*Ix+pb01*Iy+pb20*power(Ix,2)+...
40          pb11*Ix*Iy+pb02*power(Iy,2)+Ib)/...
41          (1+qb10*Ix+qb01*Iy+qb20*power(Ix,2)+qb11*Ix*Iy+qb02*power(Iy,2)))*...
42          (1+q10*Ix+q01*Iy+q20*power(Ix,2)+q11*Ix*Iy+q02*power(Iy,2))-...
43          (p00+p10*Ix+p01*Iy+p20*power(Ix,2)+p11*Ix*Iy+p02*power(Iy,2));
44
45  elseif(operation==4) %Division
46
47      I1=((pa00+pa10*Ix+pa01*Iy+pa20*power(Ix,2)+...
48          pa11*Ix*Iy+pa02*power(Iy,2)+Ia)/...
49          (1+qa10*Ix+qa01*Iy+qa20*power(Ix,2)+qa11*Ix*Iy+qa02*power(Iy,2))/...
50          (pb00+pb10*Ix+pb01*Iy+pb20*power(Ix,2)+...
51          pb11*Ix*Iy+pb02*power(Iy,2)+Ib)/...
52          (1+qb10*Ix+qb01*Iy+qb20*power(Ix,2)+qb11*Ix*Iy+qb02*power(Iy,2)))*...
53          (1+q10*Ix+q01*Iy+q20*power(Ix,2)+q11*Ix*Iy+q02*power(Iy,2))-...
54          (p00+p10*Ix+p01*Iy+p20*power(Ix,2)+p11*Ix*Iy+p02*power(Iy,2));
55
56  end;
57
58  u=inf_(I1);
59  o=sup(I1);
```

A.16 zweiDFehlerzerteilung.m

```
1   function [u,o] = zweiDFehlerzerteilung ...
2       (pa00,pa10,pa01,pa20,pa11,pa02,qa10,qa01,qa20,qa11,qa02,ua,oa,operation,...
3        pb00,pb10,pb01,pb20,pb11,pb02,qb10,qb01,qb20,qb11,qb02,ub,ob,...
4        p00,p10,p01,p20,p11,p02,q10,q01,q20,q11,q02,Ixu1,Ixo1,Iyu1,Iyo1)
5
6   %Bestimme die Anzahl der Teilboxen
7   anzbox_x=5;
8   anzbox_y=5;
9   m=0;
10  kx=(Ixo1-Ixu1)/anzbox_x;
11  ky=(Iyo1-Iyu1)/anzbox_y;
12
13  %Definiere Ober- und Untergrenzen der Teilintervalle
14  for j=1:anzbox_y
15      for k=1:anzbox_x
16          m=m+1;
17          Ixo(m)=Ixo1-(k-1)*kx;
18          Iyo(m)=Iyo1-(j-1)*ky;
19          Ixu(m)=Ixo1-k*kx;
20          Iyu(m)=Iyo1-j*ky;
21      end;
22  end;
23
24  %Berechne die Restglieder der Teilintervalle
25  %Anstatt des Algorithmus zweiDfehlerslope könnten auch andere
26  %alle Algorithmen zur Restintervallbestimmung verwendet werden, zb.:
27  %zweiDfehlermult,
```

```
%zweiDfehler,
%zweiDfehlerslopemult
for (i=1:anzbox_y*anzbox_y)
    [u(i),o(i)]=zweiDfehlerslope...
                      (pa00,pa10,pa01,pa20,pa11,pa02,qa10,qa01,qa20,qa11,qa02,...
                       ua,oa,operation,pb00,pb10,pb01,pb20,pb11,pb02,qb10,...
                       qb01,qb20,qb11,qb02,ub,ob,p00,p10,p01,p20,p11,p02,...
                       q10,q01,q20,q11,q02,Ixu(i),Ixo(i),Iyu(i),Iyo(i));

end;

 u=min(u);
 o=max(o);
```

A.17 zweiDfehlermult.m

```
function [u,o] = zweiDfehlermult ...
    (pa00,pa10,pa01,pa20,pa11,pa02,qa10,qa01,qa20,qa11,qa02,ua,oa,operation,...
     pb00,pb10,pb01,pb20,pb11,pb02,qb10,qb01,qb20,qb11,qb02,ub,ob,...
     p00,p10,p01,p20,p11,p02,q10,q01,q20,q11,q02,Ixu,Ixo,Iyu,Iyo)


Ix=infsup(Ixu,Ixo);
Iy=infsup(Iyu,Iyo);
Ia=infsup(ua,oa);
Ib=infsup(ub,ob);

%Ausmultiplizieren des Problems, man erhält die Koeffizienten eines
%Polynoms sechster und eines vierter Ordnung.
[e00,e10,e01,e20,e11,e02,e30,e21,e12,e03,e40,e31,e22,e13,e04,...
 e50,e41,e32,e23,e14,e05,e60,e51,e42,e33,e24,e15,e06,...
 n00,n10,n01,n20,n11,n02,n30,n21,n12,n03,n40,n31,n22,n13,n04]= ...
     zweidimausmultipl...
       (pa00+Ia,pa10,pa01,pa20,pa11,pa02,qa10,qa01,qa20,qa11,qa02,operation,...
        pb00+Ib,pb10,pb01,pb20,pb11,pb02,qb10,qb01,qb20,qb11,qb02,...
        p00,p10,p01,p20,p11,p02,q10,q01,q20,q11,q02) ;

%Abschätzen des Wertebereichs durch Einsetzten der Definitionsbereichs
I=(e00+e10*Ix+e01*Iy+e20*power(Ix,2)+e11*Ix*Iy+e02*power(Iy,2)+...
   e30*power(Ix,3)+e21*power(Ix,2)*Iy+e12*Ix*power(Iy,2)+...
   e03*power(Iy,3)+e40*power(Ix,4)+e31*power(Ix,3)*Iy+...
   e22*power(Ix,2)*power(Iy,2)+e13*Ix*power(Iy,3)+e04*power(Iy,4)+...
   e50*power(Ix,5)+e41*power(Ix,4)*Iy+e32*power(Ix,3)*power(Iy,2)+...
   e23*power(Ix,2)*power(Iy,3)+e14*Ix*power(Iy,4)+e05*power(Ix,5)+...
   e60*power(Ix,6)+e51*power(Ix,5)*Iy+e42*power(Ix,4)*power(Iy,2)+...
   e33*power(Ix,3)*power(Iy,3)+e24*power(Ix,2)*power(Iy,4)+...
   e15*Ix*power(Iy,5)+e06*power(Iy,6))/...
   (n00+n10*Ix+n01*Iy+n20*power(Ix,2)+n11*Ix*Iy+n02*power(Iy,2)+...
   n30*power(Ix,3)+n21*power(Ix,2)*Iy+n12*Ix*power(Iy,2)+n03*power(Iy,3)+...
   n40*power(Ix,4)+n31*power(Ix,3)*Iy+n22*power(Ix,2)*power(Iy,2)+...
   n13*Ix*power(Iy,3)+n04*power(Iy,4));
```

```
36
37 u=inf_(I);
38 o=sup(I);
```

A.18 zweidimausmultipl.m

```
1  function [e00,e10,e01,e20,e11,e02,e30,e21,e12,e03,e40,e31,e22,e13,e04,...
2            e50,e41,e32,e23,e14,e05,e60,e51,e42,e33,e24,e15,e06,...
3            n00,n10,n01,n20,n11,n02,n30,n21,n12,n03,n40,n31,n22,n13,n04]= ...
4            zweidimausmultipl(a00,a10,a01,a20,a11,a02,b10,b01,b20,b11,b02,...
5                              operation,c00,c10,c01,c20,c11,c02,d10,d01,d20,...
6                              d11,d02,y00,y10,y01,y20,y11,y02,...
7                              z10,z01,z20,z11,z02)
8
9  if(operation==1) %Addition
10
11     e00=a00 + c00 - y00;
12     e10=a10 + b10 *c00 + c10 + a00 *d10 - b10* y00 - d10* y00 - y10 + ...
13         a00* z10 + c00 *z10;
14     e01=a01 + b01* c00 + c01 + a00 *d01 - b01 *y00 - d01* y00 - y01 + ...
15         a00* z01 + c00 *z01;
16     e20=a20 + b20 *c00 + b10* c10 + c20 + a10 *d10 + a00 *d20 - b20* y00 - ...
17         b10* d10*y00 - d20 *y00 - b10* y10 - d10 *y10 - y20 + a10* z10 + ...
18         b10 *c00* z10 + c10*z10 + a00*d10 *z10 + a00* z20 + c00*z20;
19     e11=a11 + b11 *c00 + b10 *c01 + b01*c10 + c11 + a10 *d01 + ...
20         a01 *d10 + a00* d11 - b11* y00 - b10 *d01* y00 - b01* d10* y00 -  ...
21         d11*y00-  b10*y01 - d10* y01 - b01* y10 - d01 *y10 - y11+a10*z01+ ...
22         b10* c00 *z01 + c10 *z01 + a00 *d10 *z01 + a01* ...
23         z10 + b01 *c00* z10 + c01 *z10 + a00* d01* z10 + a00 *z11 + c00* z11;
24     e02=a02 + b02*c00 + b01* c01 + c02 + a01* d01 + a00*d02 - b02*y00 -  ...
25         b01* d01 *y00 - d02 *y00 - b01 *y01 - d01* y01 -y02 + a01 *z01 +  ...
26         b01* c00 *z01 + c01 *z01 + a00 *d01 *z01 + a00 *z02 + c00 *z02;
27     e30=b20* c10 + b10 *c20 + a20 *d10 + a10* d20 - b20* d10* y00 -  ...
28         b10* d20 *y00 - b20* y10 - b10* d10 *y10 - d20*y10 - b10*y20 -d10* ...
29         y20 + a20* z10 + b20*c00* z10 + b10 *c10*z10 + c20* z10 +  ...
30         a10* d10 *z10 + a00*d20 *z10 + a10 *z20 + b10* c00 *z20 +  ...
31         c10 *z20 + a00* d10* z20;
32     e21=b20* c01 + b11* c10 + b10 *c11 + b01*c20 + a20 *d01 + a11* d10 + ...
33         a10* d11 + a01 *d20 - b20* d01* y00 - b11 *d10 *y00 - b10 *d11*y00- ...
34         b01 *d20 *y00 - b20*y01 - b10* d10* y01 - d20*y01 - b11 *y10 -  ...
35         b10* d01 *y10 - b01* d10* y10 - d11*y10 - b10* y11 - d10* y11 -  ...
36         b01 *y20 - d01*y20 + a20* z01 + b20*c00*z01 + b10 *c10* z01 +  ...
37         c20 *z01 + a10 *d10*z01 +a00* d20* z01 + a11* z10 + b11* c00* z10 + ...
38         b10 *c01 *z10 + b01 *c10* z10 + c11* z10 + a10 *d01* z10 +  ...
39         a01 *d10* z10 + a00 *d11 *z10 + a10* z11 + b10 *c00* ...
40         z11 + c10* z11 + a00 *d10 *z11 + a01 *z20 + b01* c00 *z20 +  ...
41         c01 *z20 + a00 *d01 *z20;
42     e12=b11 *c01 + b10 *c02 + b02*c10 + b01 *c11 + a11* d01 + a10* d02 + ...
43         a02* d10 + a01 *d11 - b11 *d01*y00 - b10*d02*y00 - b02*d10*y00 - ...
44         b01* d11 *y00 - b11* y01 - b10 *d01 *y01 - b01* d10*y01 -  ...
45         d11* y01 - b10* y02 - d10* y02 - b02 *y10 - b01 *d01 *y10 -  ...
```

```
46          d02* y10 - b01 *y11 - d01*y11 + a11*z01 + b11 *c00* z01 +  ...
47          b10*c01*z01 + b01* c10 *z01 + c11* z01 + a10 *d01 *z01 +  ...
48          a01 *d10 *z01 + a00 *d11*z01 + a10 *z02 + b10 *c00 *z02 +  ...
49          c10 *z02 + a00 *d10*z02 + a02*z10 + b02*c00*z10 + b01*c01*z10 +  ...
50          c02 *z10 + a01 *d01* z10 + a00 *d02 *z10 + a01* z11 + ...
51          b01* c00* z11 + c01* z11 + a00 *d01 *z11;
52      e03=b02 *c01 + b01 *c02 + a02 *d01 + a01 *d02 -b02 *d01* y00 -  ...
53          b01* d02 *y00 - b02 *y01 - b01 *d01 *y01 - d02 *y01 - b01 *y02 -  ...
54          d01 *y02 + a02* z01 + b02 *c00* z01 + b01* c01 *z01 + c02 *z01 +  ...
55          a01* d01 *z01 +a00* d02* z01 + a01 *z02 + b01*c00*z02 + c01*z02+ ...
56          a00 *d01 *z02;
57      e40=b20* c20 + a20* d20 - b20 *d20* y00 - b20* d10 *y10 -b10*d20*y10- ...
58          b20 *y20 - b10*d10*y20 - d20*y20 + b20*c10*z10 + b10*c20* z10 + ...
59          a20* d10* z10 + a10* d20* z10 + a20 *z20 + b20 *c00 *z20 +  ...
60          b10 *c10* z20 + c20 *z20 + a10 *d10* z20 + a00* d20* z20;
61      e31=b20* c11 + b11 *c20 + a20 *d11 + a11 *d20 - ...
62          b20* d11* y00 - b11* d20* y00 - b20 *d10 *y01 - b10 *d20 *y01 -  ...
63          b20*d01 *y10 - b11 *d10* y10 - b10 *d11 *y10 - b01 *d20* y10 -  ...
64          b20 *y11 - b10*d10* y11 - d20 *y11 - b11* y20 - b10 *d01 *y20 -  ...
65          b01* d10 *y20 - d11 *y20 + b20* c10* z01 + b10* c20 *z01 +  ...
66          a20* d10* z01 + a10 *d20* z01 + b20 *c01 *z10 + b11 *c10* z10 +  ...
67          b10 *c11* z10 + b01 *c20*z10 + a20*d01* z10 + a11*d10 *z10 + a10* ...
68          d11* z10 + a01* d20 *z10 + a20* z11 + b20 *c00 *z11 + b10* c10*z11+...
69          c20* z11 + a10* d10 *z11 + a00 *d20 *z11 + a11 *z20 + b11*c00*z20+...
70          b10* c01 *z20 + b01 *c10* z20 + c11*z20 + a10 *d01 *z20 +  ...
71          a01 *d10* z20 + a00* d11 *z20;
72      e22=b20*c02 + b11 *c11 + b02 *c20 + a20* d02 + a11* d11 + a02 *d20 - ...
73          b20* d02 *y00 - b11* d11 *y00 - b02 *d20 *y00 - b20 *d01 *y01 -  ...
74          b11*d10 *y01 - b10* d11*y01 - b01 *d20 *y01 - b20* y02 -  ...
75          b10 *d10*y02 - d20* y02 - b11 *d01 *y10 - b10 *d02* y10 -  ...
76          b02 *d10* y10 - b01 *d11* y10 - b11*y11 - b10 *d01 *y11 -  ...
77          b01*d10 *y11 - d11* y11 - b02* y20 - b01* d01* y20 - d02 *y20 + ...
78          b20 *c01* z01 + b11 *c10* z01 + b10* c11 *z01 + b01 *c20 *z01 +  ...
79          a20 *d01 *z01 + a11* d10*z01 + a10 *d11* z01 + a01 *d20 *z01 +  ...
80          a20 *z02 + b20* c00*z02 + b10* c10* z02 + c20*z02 + a10 *d10* z02 +...
81          a00*d20 *z02 + b11 *c01* z10 + b10 *c02 *z10 + b02* c10* z10 + ...
82          b01* c11 *z10 + a11* d01* z10 + a10* d02 *z10 + a02* d10* z10 + ...
83          a01 *d11 *z10 + a11* z11 +b11 *c00 *z11 + b10* c01 *z11 +  ...
84          b01*c10* z11 + c11 *z11 + a10* d01 *z11 + a01 *d10* z11 +  ...
85          a00 *d11* z11 + a02* z20 + b02 *c00 *z20 + b01* c01 *z20 + ...
86          c02* z20 + a01 *d01* z20 + a00 *d02 *z20;
87      e13=b11 *c02 + b02*c11 + a11 *d02 + a02 *d11 -b11* d02* y00 -  ...
88          b02 *d11 *y00 - b11 *d01*y01 - b10 *d02* y01 - b02 *d10 *y01 -  ...
89          b01*d11* y01 - b11* y02 - b10 *d01 *y02 - b01 *d10 *y02 -  ...
90          d11 *y02 - b02 *d01 *y10 - b01*d02 *y10 - b02* y11 - b01*d01*y11 - ...
91          d02 *y11 + b11 *c01 *z01 +b10*c02 *z01 + b02*c10 *z01 +  ...
92          b01* c11* z01 + a11 *d01 *z01 + a10 *d02 *z01 + a02 *d10* z01 +  ...
93          a01 *d11 *z01 + a11 *z02 + b11 *c00*z02 + b10 *c01 *z02 +  ...
94          b01 *c10 *z02 + c11* z02 + a10 *d01* z02 + a01* d10 *z02 +  ...
95          a00* d11 *z02 + b02* c01 *z10 + b01*c02 *z10 + a02* d01*z10 +  ...
96          a01 *d02 *z10 + a02 *z11 + b02* c00*z11 + b01 *c01* z11 +  ...
97          c02 *z11 + a01 *d01 *z11 + a00 *d02 *z11;
98      e04=b02* c02 + a02* d02 - b02* d02* y00 - b02*d01 *y01 - b01*d02*y01 -...
99          b02 *y02 - b01*d01 *y02 - d02 *y02 + b02 *c01*z01 + b01*c02* z01+ ...
```

```
100         a02 *d01 *z01 + a01 *d02 *z01 + a02*z02 + b02* c00 *z02 +  ...
101         b01 *c01*z02 + c02 *z02 + a01* d01* z02 + a00 *d02 *z02;
102     e50=-b20* d20* y10 - b20* d10* y20 - b10 *d20* y20 + b20 *c20 *z10 +...
103         a20* d20* z10 + b20 *c10 *z20 + b10 *c20 *z20 + a20 *d10 *z20 +  ...
104         a10 *d20* z20;
105     e41=-b20 *d20* y01 - b20 *d11 *y10 - b11*d20 *y10 - b20 *d10 *y11 -...
106         b10 *d20* y11 - b20*d01*y20 - b11 *d10* y20 - b10 *d11* y20 -...
107         b01 *d20* y20 + b20* c20* z01 + a20 *d20*z01 + b20 *c11 *z10 +...
108         b11 *c20 *z10 + a20* d11* z10 + a11* d20*z10 + b20*c10 *z11 +...
109         b10 *c20 *z11 + a20* d10* z11 + a10 *d20 *z11 + b20 *c01 *z20 +...
110         b11 *c10 *z20 + b10 *c11 *z20 + b01 *c20* z20 + a20 *d01 *z20 +...
111         a11* d10* z20 + a10 *d11 *z20 + a01* d20* z20;
112     e32=-b20*d11* y01 - b11 *d20 *y01 - b20* d10 *y02 - b10* d20* y02 -...
113         b20 *d02* y10 - b11* d11* y10 - b02 *d20 *y10 - b20 *d01* y11 -...
114         b11 *d10*y11 - b10 *d11 *y11 - b01* d20 *y11 - b11*d01* y20 -...
115         b10 *d02 *y20 - b02 *d10 *y20 - b01* d11 *y20 + b20*c11 *z01 +...
116         b11 *c20 *z01 + a20* d11* z01 + a11 *d20 *z01 + b20*c10 *z02 +...
117         b10 *c20* z02 + a20 *d10* z02 + a10 *d20 *z02 + b20* c02 *z10 +...
118         b11* c11 *z10 + b02 *c20* z10 + a20*d02 *z10 + a11* d11* z10 +...
119         a02 *d20 *z10 + b20 *c01*z11 + b11 *c10 *z11 + b10 *c11 *z11 +...
120         b01 *c20*z11 + a20* d01* z11 + a11 *d10 *z11 + a10 *d11 *z11 +...
121         a01 *d20* z11 + b11*c01 *z20 + b10 *c02 *z20 + b02* c10 *z20 +...
122         b01 *c11* z20 + a11 *d01* z20 + a10 *d02* z20 + a02 *d10* z20 +  ...
123         a01* d11 *z20;
124      e23=-b20* d02 *y01 - b11 *d11* y01 - b02 *d20* y01 - b20 *d01 *y02 -...
125         b11*d10 *y02 - b10* d11* y02 - b01* d20 *y02 - b11* d02* y10 -...
126         b02 *d11* y10 - b11* d01 *y11 - b10*d02 *y11 - b02 *d10 *y11 -...
127         b01 *d11 *y11 - b02* d01 *y20 - b01 *d02* y20 + b20* c02 *z01 +...
128         b11* c11 *z01 + b02* c20 *z01 + a20 *d02 *z01 + a11* d11 *z01 +...
129         a02* d20 *z01 + b20 *c01 *z02 + b11 *c10 *z02 + b10* c11*z02 +...
130         b01 *c20* z02 + a20 *d01 *z02 + a11 *d10 *z02 + a10* d11 *z02 +...
131         a01* d20* z02 + b11 *c02* z10 + b02 *c11* z10 + a11* d02* z10 +...
132         a02 *d11 *z10 + b11*c01* z11 + b10*c02* z11 + b02* c10 *z11 +...
133         b01*c11 *z11 + a11 *d01 *z11 + a10 *d02 *z11 + a02 *d10 *z11 +...
134         a01*d11 *z11 + b02 *c01* z20 + b01* c02 *z20 + a02*d01* z20 +  ...
135         a01*d02* z20;
136     e14=-b11*d02 *y01 - b02 *d11* y01 - b11 *d01* y02 - b10 *d02 *y02 -...
137         b02* d10 *y02 - b01* d11 *y02 - b02 *d02 *y10 - b02 *d01* y11 -...
138         b01 *d02* y11 + b11 *c02 *z01 + b02 *c11* z01 + a11* d02* z01 +...
139         a02 *d11 *z01 + b11* c01* z02 + b10 *c02* z02 + b02* c10* z02 +...
140         b01 *c11* z02 + a11* d01* z02 + a10 *d02 *z02 + a02* d10* z02 +...
141         a01* d11* z02 + b02 *c02*z10 + a02 *d02 *z10 + b02 *c01* z11 +...
142         b01* c02* z11 + a02 *d01* z11 + a01 *d02* z11;
143     e05=-b02* d02* y01 - b02 *d01 *y02 - b01*d02 *y02 + b02 *c02 *z01 +...
144         a02* d02* z01 + b02*c01 *z02 + b01 *c02* z02 + a02 *d01 *z02 +  ...
145         a01* d02 *z02;
146     e60=-b20*d20 *y20 + b20 *c20* z20 + a20* d20 *z20;
147     e51=-b20* d20* y11 - b20* d11 *y20 - b11 *d20 *y20 + b20 *c20* z11 +...
148         a20 *d20* z11 + b20 *c11 *z20 + b11 *c20* z20 + a20 *d11 *z20 +  ...
149         a11*d20 *z20;
150     e42=-b20 *d20 *y02 - b20 *d11 *y11 - b11 *d20 *y11 - b20 *d02*y20 -...
151         b11 *d11* y20 - b02 *d20* y20 + b20 *c20*z02 + a20 *d20 *z02 +...
152         b20 *c11 *z11 + b11* c20 *z11 + a20*d11 *z11 + a11*d20* z11 +...
153         b20 *c02 *z20 + b11* c11 *z20 + b02 *c20* z20 + a20 *d02 *z20 +...
```

```
154         a11 *d11*z20 + a02 *d20 *z20;
155     e33=-b20 *d11 *y02 - b11* d20* y02 - b20 *d02 *y11 - b11* d11 *y11 -...
156         b02* d20* y11 - b11 *d02 *y20 - b02* d11 *y20 + b20*c11 *z02 +...
157         b11 *c20 *z02 + a20 *d11 *z02 + a11* d20* z02 + b20* c02 *z11 +...
158         b11 *c11 *z11 + b02* c20* z11 + a20*d02*z11 + a11* d11* z11 +...
159         a02*d20 *z11 + b11 *c02* z20 + b02* c11 *z20 + a11 *d02* z20 +  ...
160         a02 *d11* z20;
161     e24=-b20 *d02* y02 - b11 *d11*y02 - b02 *d20 *y02 - b11 *d02* y11 -...
162         b02* d11* y11 - b02* d02 *y20 + b20* c02* z02 + b11* c11* z02 +...
163         b02 *c20 *z02 + a20 *d02 *z02 + a11 *d11 *z02 + a02* d20 *z02 +...
164         b11 *c02* z11 + b02 *c11* z11 + a11 *d02* z11 + a02 *d11 *z11 +...
165         b02* c02 *z20 + a02 *d02* z20;
166     e15=-b11* d02* y02 - b02* d11* y02 - b02 *d02 *y11 + b11 *c02 *z02 +...
167         b02*c11 *z02 + a11 *d02 *z02 + a02 *d11 *z02 + b02 *c02 *z11 +  ...
168         a02 *d02* z11;
169     e06=-b02* d02 *y02 + b02* c02 *z02 + a02 *d02 *z02;
170     n00=1;
171     n10=b10 + d10;
172     n01=b01 + d01;
173     n20=b20 + b10 *d10 + d20;
174     n11=b11 + b10 *d01 + b01* d10 + d11;
175     n02=b02 + b01 *d01 + d02;
176     n30=b20* d10 + b10 *d20;
177     n21=b20* d01 + b11 *d10 + b10* d11 + b01 *d20;
178     n12=b11* d01 + b10 *d02 + b02* d10 + b01 *d11;
179     n03=b02* d01 + b01 *d02;
180     n40=b20* d20;
181     n31=b20* d11 + b11 *d20;
182     n22=b20* d02 + b11 *d11 + b02* d20;
183     n13=b11* d02 + b02 *d11;
184     n04=b02* d02;
185
186 elseif(operation==2) %Subtraktion
187
188     e00=a00 - c00 - y00;
189     e10=a10 - b10* c00 - c10 + a00* d10 - b10 *y00 - d10 *y00 - y10 +  ...
190         a00* z10 - c00 *z10;
191     e01=a01 - b01* c00 - c01 + a00 *d01 - b01* y00 - d01*y00 - y01 +  ...
192         a00 *z01 - c00 *z01;
193     e20=a20 - b20* c00 - b10 *c10 - c20 + a10 *d10 + a00 *d20 - b20* y00 -...
194         b10 *d10*y00 - d20 *y00 - b10 *y10 - d10* y10 -y20 + a10* z10 - ...
195         b10* c00 *z10 - c10*z10 + a00 *d10 *z10 + a00 *z20 - c00 *z20;
196     e11=a11 - b11* c00 - b10 *c01 - b01 *c10 - c11 + a10 *d01 + ...
197         a01 *d10 + a00 *d11 - b11* y00 - b10*d01 *y00 - b01 *d10* y00 -...
198         d11 *y00 - b10 *y01 - d10* y01 - b01 *y10 - d01 *y10 - y11 +  ...
199         a10 *z01 - b10 *c00 *z01 - c10 *z01 + a00*d10 *z01 + a01*z10 -  ...
200         b01* c00 *z10 - c01 *z10 + a00 *d01* z10 + a00 *z11 - c00* z11;
201     e02=a02 - b02 *c00 - b01* c01 - c02 + a01* d01 + a00* d02 - b02 *y00 -...
202         b01 *d01 *y00 - d02 *y00 - b01 *y01 - d01 *y01 - y02 + a01*z01 -  ...
203         b01* c00 *z01 - c01* z01 + a00 *d01 *01 + a00 *z02 - c00 *z02;
204     e30=-b20* c10 - b10* c20 + a20 *d10 + a10 *d20 -b20* d10 *y00 -  ...
205         b10 *d20*y00 - b20*y10 - b10* d10* y10 - d20* y10 - b10 *y20 - ...
206         d10* y20 + a20*z10 - b20* c00 *z10 - b10 *c10*z10 - c20 *z10 +  ...
207         a10 *d10 *z10 +a00* d20 *z10 + a10*z20 - b10 *c00 *z20 - c10*z20+...
```

```
208         a00 *d10 *z20;
209     e21=-b20* c01 - b11* c10 - b10 *c11 - b01 *c20 + a20 *d01 + a11 *d10 +...
210         a10* d11 + a01* d20 - b20* d01* y00 - b11* d10 *y00 - b10*d11*y00-...
211         b01*d20* y00 - b20 *y01 - b10* d10 *y01 -d20 *y01 - b11* y10 -  ...
212         b10* d01* y10 - b01*d10 *y10 - d11 *y10 -b10 *y11 - d10* y11 -  ...
213         b01 *y20 - d01 *y20 + a20 *z01 - b20*c00 *z01 - b10 *c10 *z01 -  ...
214         c20 *z01 + a10* d10 *z01 +a00*d20*z01 + a11*z10 - b11*c00*z10 -...
215         b10* c01 *z10 - b01* c10* z10 - c11* z10+ a10 *d01 *z10 +  ...
216         a01*d10 *z10 + a00 *d11 *z10 +a10 *z11 - b10* c00 *z11 -  ...
217         c10 *z11 + a00 *d10 *z11 + a01 *z20 -b01 *c00* z20 - c01 *z20 +  ...
218         a00 *d01* z20;
219     e12=-b11 *c01 - b10 *c02 - b02 *c10 - b01 *c11 + a11* d01 + a10 *d02 +...
220         a02 *d10 + a01* d11 - b11* d01 *y00 - b10 *d02 *y00 - b02*d10*y00-...
221         b01 *d11* y00 - b11 *y01 - b10 *d01 *y01 -b01 *d10 *y01 - d11*y01-...
222         b10* y02 - d10* y02 - b02 *y10 - b01* d01 *y10 - d02*y10 -  ...
223         b01* y11 - d01* y11 + a11 *z01 - b11 *c00* z01 - b10 *c01 *z01 -  ...
224         b01* c10 *z01 -c11* z01 + a10* d01 *z01 + a01*d10* z01 +...
225         a00*d11 *z01 + a10* z02 - b10* c00* z02 - c10 *z02 + a00* d10 *z02 +...
226         a02 *z10 - b02* c00* z10 - b01 *c01* z10 - c02* z10 +a01*d01*z10+...
227         a00* d02* z10 + a01* z11 - b01 *c00 *z11 - c01* z11 + a00 *d01 *z11;
228     e03=-b02* c01 - b01* c02 + a02* d01 + a01* d02 -b02*d01*y00 - ...
229         b01* d02* y00 - b02 *y01 - b01 *d01 *y01 - d02* y01 - b01 *y02 - ...
230         d01* y02 + a02 *z01 - b02 *c00* z01 - b01 *c01 *z01 - c02* z01 +  ...
231         a01 *d01* z01 + a00* d02* z01 + a01* z02 - b01*c00 *z02 - c01*z02 +...
232         a00 *d01 *z02;
233     e40=-b20 *c20 + a20*d20 -b20*d20*y00 - b20*d10*y10 - b10*d20*y10 - ...
234         b20* y20 - b10*d10*y20 - d20 *y20 -b20 *c10* z10 - b10* c20 *z10 +...
235         a20 *d10* z10 + a10 *d20 *z10 + a20* z20 - b20 *c00* z20 -  ...
236         b10 *c10 *z20 - c20*z20+ a10* d10* z20 + a00 *d20* z20;
237     e31=-b20* c11 - b11* c20 + a20 *d11 + a11* d20 -b20*d11 *y00 -  ...
238         b11* d20* y00 - b20* d10* y01 - b10 *d20* y01 - b20* d01 *y10 -...
239         b11* d10* y10 - b10 *d11 *y10 -b01 *d20 *y10 - b20* y11 -  ...
240         b10* d10 *y11 - d20 *y11 - b11 *y20 - b10*d01*y20 - b01*d10*y20 -  ...
241         d11* y20 - b20 *c10 *z01 - b10 *c20 *z01 + a20* d10* z01 +...
242         a10* d20* z01 - b20* c01* z10 -b11 *c10 *z10 - b10 *c11* z10 -  ...
243         b01 *c20 *z10 + a20 *d01 *z10 + a11*d10 *z10 +a10* d11* z10 +  ...
244         a01* d20 *z10 + a20 *z11 - b20* c00 *z11 - b10*c10*z11 - c20*z11 +...
245         a10* d10 *z11 + a00 *d20 *z11 + a11*z20 - b11 *c00* z20 -  ...
246         b10 *c01 *z20 - b01 *c10*z20 - c11* z20 + a10* d01 *z20 +  ...
247         a01 *d10* z20 + a00* d11* z20;
248     e22=-b20 *c02 - b11* c11 - b02* c20 + a20 *d02 + a11* d11 + a02 *d20 -...
249         b20 *d02 *y00 - b11* d11*y00 - b02* d20 *y00 -b20 *d01*y01 -  ...
250         b11 *d10 *y01 - b10 *d11 *y01 - b01* d20 *y01 - b20 *y02 - ...
251         b10 *d10*y02 - d20 *y02 - b11 *d01* y10 - b10 *d02 *y10 -  ...
252         b02* d10 *y10 -b01 *d11 *y10 - b11* y11 - b10 *d01* y11 -  ...
253         b01* d10* y11 - d11 *y11 - b02 *y20 - b01* d01 *y20 - d02* y20 -  ...
254         b20* c01 *z01 - b11*c10*z01 - b10 *c11* z01 - b01* c20* z01+  ...
255         a20* d01 *z01 + a11 *d10 *z01 + a10 *d11 *z01 + a01* d20* z01 +...
256         a20*z02 - b20 *c00 *z02 - b10 *c10* z02 - c20 *z02 + a10*d10*z02 +...
257         a00 *d20 *z02 -b11* c01* z10 - b10 *c02 *z10 - b02 *c10* z10 -  ...
258         b01 *c11 *z10 +a11* d01* z10 + a10 *d02 *z10 + a02 *d10 *z10 +  ...
259         a01 *d11 *z10 + a11 *z11 -b11* c00 *z11 - b10 *c01* z11 -  ...
260         b01 *c10 *z11 - c11 *z11 + a10 *d01 *z11 + a01 *d10 *z11 +  ...
261         a00*d11 *z11 + a02* z20 - b02* c00 *z20 - b01 *c01 *z20 -...
```

```
262         c02 *z20 + a01* d01 *z20 + a00 *d02 *z20;
263     e13=-b11 *c02 - b02 *c11 + a11*d02 + a02* d11 -...
264         b11* d02 *y00 - b02 *d11 *y00 - b11 *d01 *y01 - b10* d02 *y01 -  ...
265         b02* d10 *y01 -b01 *d11* y01 - b11 *y02 - b10 *d01* y02 -  ...
266         b01* d10 *y02 - d11* y02 - b02* d01* y10- b01 *d02*y10 -  ...
267         b02* y11 - b01*d01 *y11 - d02*y11 - b11* c01 *z01 -b10* c02*z01 - ...
268         b02* c10 *z01 - b01*c11 *z01 + a11 *d01* z01 + a10 *d02* z01 +  ...
269         a02*d10* z01 + a01* d11* z01 +a11 *z02 - b11* c00* z02 -  ...
270         b10* c01 *z02 - b01 *c10 *z02 - c11* z02 + a10 *d01 *z02 +...
271         a01* d10 *z02 + a00* d11 *z02 -b02 *c01 *z10 - b01* c02 *z10 +  ...
272         a02* d01 *z10 + a01 *d02 *z10 + a02* z11 - b02 *c00* ...\
273         z11 - b01* c01 *z11 - c02* z11 + a01* d01* z11 + a00 *d02 *z11;
274     e04=-b02* c02 + a02 *d02 -b02 *d02*y00 - b02 *d01* y01 - b01*d02 *y01 -...
275         b02* y02 - b01* d01 *y02 - d02*y02 -b02 *c01* z01 - b01 *c02 *z01 +...
276         a02*d01 *z01 + a01* d02 *z01 + a02*z02 - b02 *c00 *z02 -  ...
277         b01* c01 *z02 - c02 *z02+ a01 *d01* z02 + a00 *d02* z02;
278     e50=-b20 *d20* y10 - b20* d10*y20 - b10 *d20 *y20 - b20 *c20 *z10 +...
279         a20* d20* z10 - b20 *c10* z20 - b10 *c20 *z20 + a20 *d10 *z20 +...
280         a10*d20 *z20;
281     e41=-b20* d20 *y01 - b20*d11 *y10 - b11 *d20 *y10 - b20* d10* y11 -...
282         b10* d20* y11 - b20* d01 *y20 - b11* d10 *y20 - b10* d11* y20 -...
283         b01 *d20 *y20 - b20* c20* z01 + a20 *d20 *z01 - b20* c11* z10 -...
284         b11 *c20 *z10 + a20* d11* z10 + a11 *d20 *z10 - b20* c10* z11 -...
285         b10 *c20 *z11 + a20* d10* z11 + a10 *d20 *z11 - b20* c01* z20 -...
286         b11 *c10 *z20 - b10* c11* z20 - b01 *c20 *z20 + a20* d01* z20 +...
287         a11 *d10 *z20 + a10* d11* z20 + a01 *d20 *z20;
288     e32=-b20* d11*y01 - b11* d20* y01 - b20 *d10* y02 - b10 *d20* y02 -...
289         b20* d02* y10 - b11* d11* y10 - b02 *d20 *y10 - b20 *d01 *y11 -...
290         b11* d10* y11 - b10* d11* y11 - b01 *d20 *y11 - b11 *d01 *y20 -...
291         b10* d02* y20 - b02* d10* y20 - b01 *d11 *y20 - b20 *c11 *z01 -...
292         b11* c20 *z01 + a20* d11* z01 + a11 *d20 *z01 - b20 *c10 *z02 -...
293         b10* c20 *z02 + a20* d10* z02 + a10 *d20 *z02 - b20 *c02 *z10 -...
294         b11* c11* z10 - b02* c20* z10 + a20 *d02 *z10 + a11 *d11 *z10 +...
295         a02* d20* z10 - b20* c01* z11 - b11 *c10 *z11 - b10 *c11 *z11 -...
296         b01* c20* z11 + a20* d01* z11 + a11 *d10 *z11 + a10 *d11 *z11 +...
297         a01* d20* z11 - b11* c01* z20 - b10 *c02 *z20 - b02 *c10 *z20 -...
298         b01* c11* z20 + a11* d01* z20 + a10 *d02 *z20 + a02 *d10 *z20 + ...
299         a01* d11* z20;
300     e23=-b20 *d02 *y01 - b11 *d11 *y01 - b02 *d20 *y01 - b20 *d01 *y02 -...
301         b11 *d10* y02 - b10 *d11 *y02 - b01 *d20 *y02 - b11 *d02 *y10 -...
302         b02 *d11* y10 - b11 *d01 *y11 - b10 *d02 *y11 - b02 *d10 *y11 -...
303         b01 *d11* y11 - b02 *d01 *y20 - b01 *d02 *y20 - b20 *c02 *z01 -...
304         b11 *c11 *z01 - b02 *c20 *z01 + a20 *d02 *z01 + a11 *d11 *z01 +...
305         a02 *d20 *z01 - b20 *c01 *z02 - b11 *c10 *z02 - b10 *c11 *z02 -...
306         b01 *c20 *z02 + a20 *d01 *z02 + a11 *d10 *z02 + a10 *d11 *z02 +...
307         a01 *d20 *z02 - b11 *c02 *z10 - b02 *c11 *z10 + a11 *d02 *z10 +...
308         a02 *d11 *z10 - b11 *c01 *z11 - b10 *c02 *z11 - b02 *c10 *z11 -...
309         b01 *c11 *z11 + a11 *d01 *z11 + a10 *d02 *z11 + a02 *d10 *z11 +...
310         a01 *d11 *z11 - b02 *c01 *z20 - b01 *c02 *z20 + a02 *d01 *z20 + ...
311         a01 *d02* z20;
312     e14=-b11* d02* y01 - b02* d11 *y01 - b11* d01* y02 - b10 *d02* y02 -....
313         b02 *d10 *y02 - b01* d11* y02 - b02 *d02 *y10 - b02 *d01* y11 -...
314         b01 *d02 *y11 - b11* c02* z01 - b02 *c11 *z01 + a11 *d02* z01 +...
315         a02 *d11 *z01 - b11* c01* z02 - b10 *c02 *z02 - b02 *c10* z02 -...
```

```
316         b01 *c11 *z02 + a11* d01* z02 + a10 *d02 *z02 + a02 *d10* z02 +...
317         a01 *d11 *z02 - b02* c02* z10 + a02 *d02 *z10 - b02 *c01* z11 -...
318         b01 *c02 *z11 + a02* d01* z11 + a01 *d02 *z11;
319     e05=-b02* d02* y01 - b02 *d01* y02 - b01* d02 *y02 - b02 *c02 *z01 +...
320         a02 *02 *z01 - b02 *c01 *z02 - b01* c02 *z02 + a02 *d01* z02 +  ...
321         a01* d02 *z02;
322     e60=-b20* d20* y20 - b20*c20 *z20 + a20 *d20 *z20;
323     e51=-b20* d20 *y11 - b20*d11 *y20 - b11 *d20 *y20 - b20 *c20 *z11 +...
324         a20 *d20*z11 - b20*c11* z20 - b11*c20 *z20 + a20 *d11* z20 +  ...
325         a11* d20 *z20;
326     e42=-b20* d20* y02 - b20 *d11 *y11 - b11*d20 *y11 - b20* d02 *y20 -...
327         b11 *d11* y20 - b02* d20 *y20 - b20* c20 *z02 + a20* d20 *z02 -...
328         b20 *c11* z11 - b11* c20 *z11 + a20* d11 *z11 + a11* d20 *z11 -...
329         b20 *c02* z20 - b11* c11 *z20 - b02* c20 *z20 + a20* d02 *z20 +...
330         a11 *d11* z20 + a02* d20 *z20;
331     e33=-b20* d11* y02 - b11* d20 *y02 - b20 *d02 *y11 - b11 *d11 *y11 -...
332         b02 *d20* y11 - b11 *d02 *y20 - b02* d11 *y20 - b20* c11 *z02 -...
333         b11 *c20* z02 + a20 *d11 *z02 + a11* d20 *z02 - b20* c02 *z11 -...
334         b11 *c11* z11 - b02 *c20 *z11 + a20* d02 *z11 + a11* d11 *z11 +...
335         a02 *d20* z11 - b11 *c02 *z20 - b02* c11 *z20 + a11* d02 *z20 +  ...
336         a02*d11 *z20;
337     e24=-b20* d02 *y02 - b11* d11* y02 - b02 *d20* y02 - b11* d02 *y11 -...
338         b02 *d11* y11 - b02 *d02 *y20 - b20 *c02 *z02 - b11 *c11 *z02 -...
339         b02 *c20* z02 + a20 *d02 *z02 + a11 *d11 *z02 + a02 *d20 *z02 -...
340         b11 *c02* z11 - b02 *c11 *z11 + a11 *d02 *z11 + a02 *d11 *z11 -...
341         b02 *c02* z20 + a02 *d02 *z20;
342     e15=-b11* d02 *y02 - b02* d11 *y02 - b02 *d02 *y11 - b11* c02* z02 -...
343         b02 *c11 *z02 + a11*d02 *z02 + a02* d11 *z02 - b02* c02 *z11 +  ...
344         a02* d02 *z11;
345     e06=-b02* d02*y02 - b02 *c02 *z02 + a02* d02 *z02;
346     n00=1;
347     n10=b10 + d10;
348     n01=b01 + d01;
349     n20=b20 + b10 *d10 + d20;
350     n11=b11 + b10 *d01 + b01* d10 + d11;
351     n02=b02 + b01 *d01 + d02;
352     n30=b20* d10 + b10 *d20;
353     n21=b20* d01 + b11 *d10 + b10* d11 + b01 *d20;
354     n12=b11* d01 + b10 *d02 + b02* d10 + b01 *d11;
355     n03=b02* d01 + b01 *d02;
356     n40=b20* d20;
357     n31=b20* d11 + b11 *d20;
358     n22=b20* d02 + b11 *d11 + b02* d20;
359     n13=b11* d02 + b02 *d11;
360     n04=b02* d02;
361
362 elseif(operation==3) %Multiplikation
363
364     e00=a00 *c00 - y00;
365     e10=a10 *c00 + a00 *c10 - b10* y00 - d10 *y00 - y10 + a00* c00 *z10;
366     e01=a01 *c00 + a00 *c01 - b01* y00 - d01 *y00 - y01 + a00* c00 *z01;
367     e20=a20 *c00 + a10 *c10 + a00* c20 - b20 *y00 - ...
368         b10* d10*y00 - d20 *y00 - b10 *y10 - d10 *y10 - y20 + a10*c00*z10+...
369         a00* c10 *z10 + a00*c00 *z20;
```

```
e11=a11* c00 + a10*c01 + a01 *c10 + a00 *c11 -b11 *y00 - b10 *d01*y00-...
    b01 *d10 *y00 - d11 *y00 - b10 *y01 - d10*y01 - b01* y10 -  ...
    d01* y10 - y11 + a10 *c00* z01 + a00* c10* z01 + a01* c00 *z10 + ...
    a00* c01* z10 + a00* c00 *z11;
e02=a02* c00 + a01* c01 + a00* c02 - b02* y00 - b01* d01 *y00 -  ...
    d02 *y00 - b01 *y01 - d01 *y01 - y02 + a01* c00* z01 + a00*...
    c01* z01 + a00*c00 *z02;
e30=a20* c10 + a10* c20 - b20* d10 *y00 - b10 *d20 *y00 - b20* y10 - ...
    b10 *d10* y10 - d20 *y10 - b10 *y20 - d10 *y20 + a20* c00* z10 + ...
    a10* c10*z10 + a00 *c20 *z10 + a10* c00 *z20 + a00*c10* z20;
e21=a20*c01 + a11* c10 + a10 *c11 + a01* c20 -b20* d01* y00 -  ...
    b11 *d10 *y00 - b10* d11 *y00 - b01 *d20 *y00 - b20* y01 -  ...
    b10* d10*y01 - d20 *y01 - b11*y10 - b10*d01*y10 - b01*d10*y10 -...
    d11 *y10 - b10* y11 - d10*y11 - b01* y20 - d01*y20 + a20*c00*z01+...
    a10 *c10 *z01 + a00 *c20 *z01 +a11* c00* z10 + a10* c01* z10 +  ...
    a01 *c10 *z10 + a00* c11* z10 + a10 *c00 *z11 + a00*c10* z11 +  ...
    a01* c00* z20 + a00* c01 *z20;
e12=a11 *c01 + a10 *c02 + a02 *c10 + a01* c11 -b11* d01* y00 -  ...
    b10 *d02* y00 - b02* d10 *y00 - b01* d11 *y00 - b11* y01 -  ...
    b10*d01* y01 - b01 *d10 *y01 - d11 *y01 - b10* y02 - d10* y02 -  ...
    b02*y10 - b01 *d01*y10 - d02*y10 - b01* y11 - d01 *y11 +  ...
    a11* c00* z01 + a10* c01 *z01 + a01*c10 *z01 + a00* c11*z01 +  ...
    a10* c00* z02 + a00*c10*z02 + a02* c00* z10 + a01 *c01 *z10 +  ...
    a00* c02 *z10 + a01*c00*z11 + a00 *c01 *z11;
e03=a02* c01 + a01 *c02 -b02 *d01*y00 - b01* d02 *y00 - b02* y01 -  ...
    b01 *d01 *y01 - d02* y01 - b01* y02 - d01* y02 + a02* c00* z01 +  ...
    a01* c01 *z01 + a00 *c02 *z01 + a01 *c00* z02 + a00 *c01 *z02;
e40=a20* c20 - b20 *d20*y00 - b20*d10*y10 - b10*d20*y10 - b20*y20 -...
    b10* d10 *y20 - d20* y20 + a20* c10 *z10 + a10 *c20*z10 +  ...
    a20 *c00* z20 + a10 *c10* z20 + a00* c20 *z20;
e31=a20*c11 + a11* c20 -b20*d11* y00 - b11 *d20 *y00 - b20 *d10 *y01- ...
    b10 *d20* y01 - b20 *d01 *y10 - b11*d10 *y10 - b10* d11 *y10 -  ...
    b01* d20 *y10 - b20 *y11 - b10 *d10* y11 - d20 *y11 - b11*y20 -  ...
    b10 *d01* y20 - b01 *d10 *y20 - d11*y20 + a20*c10 *z01 +  ...
    a10 *c20* z01 + a20* c01* z10 + a11 *c10 *z10 + a10 *c11 *z10 +  ...
    a01 *c20 *z10 + a20 *c00 *z11 +a10 *c10* z11 + a00* c20* z11 +  ...
    a11* c00* z20 + a10 *c01* z20 + a01 *c10 *z20 + a00*c11* z20;
e22=a20 *c02 + a11* c11 + a02* c20 - b20 *d02* y00 - b11* d11* y00 - ...
    b02* d20 *y00 - b20* d01*y01 - b11 *d10*y01 - b10 *d11 *y01 -y10  ...
    - b10 *d02*y10 - b02 *d10 *y10 - b01 *d11*y10 - b11 *y11 - ...
    b10*d01*y11 - b01* d10 *y11 - d11* y11 - b02 *y20 - b01 *d01 *y20-...
    d02* y20 + a20*c01* z01 + a11 *c10*z01 + a10 *c11 *z01 +  ...
    a01* c20 *z01 + a20 *c00 *z02 + a10* c10*z02 + a00*c20 *z02 +  ...
    a11 *c01* z10 + a10 *c02 *z10 + a02 *c10 *z10 + a01 *c11 *z10 + ...
    a11 *c00* z11 + a10 *c01 *z11 + a01 *c10 *z11 + a00* c11 *z11 +  ...
    a02 *c00 *z20 + a01 *c01*z20 + a00* c02* z20;
e13=a11* c02 + a02 *c11 - b11*d02*y00 - b02*d11*y00 - b11*d01*y01 -  ...
    b10 *d02 *y01 - b02* d10 *y01 - b01 *d11* y01 - b11 *y02 -  ...
    b10 *d01*y02 - b01 *d10* y02 - d11 *y02 - b02 *d01 *y10 -  ...
    b01* d02* y10 - b02* y11 - b01* d01 *y11 - d02 *y11 +  ...
    a11* c01* z01 +a10*c02* z01 + a02 *c10*z01 + a01 *c11 *z01 +  ...
    a11*c00 *z02 + a10 *c01 *02 + a01 *c10* z02 + a00* c11* z02 +  ...
    a02 *c01* z10 + a01 *c02 *z10 + a02 *c00*z11 + a01 *c01* z11 +  ...
    a00 *c02 *z11;
```

```
e04=a02 *c02 - b02* d02* y00 - b02 *d01* y01 - b01 *d02 *y01 -  ...
    b02 *y02 - b01* d01* y02 -d02*y02 + a02 *c01 *z01 + a01* ...
    c02*z01 + a02*c00 *z02 + a01* c01 *z02 + a00 *c02 *z02;
e50=-b20* d20 *y10 - b20*d10*y20 - b10 *d20* y20 + a20 *c20 *z10 + ...
    a20 *c10* z20 + a10 *c20* z20;
e41=-b20*d20*y01 - b20*d11*y10 - b11 *d20* y10 - b20 *d10 *y11 -...
    b10* d20 *y11 - b20 *d01 *y20 - b11 *d10 *y20 - b10 *d11 *y20 -...
    b01* d20 *y20 + a20 *c20 *z01 + a20 *c11 *z10 + a11 *c20 *z10 +...
    a20* c10 *z11 + a10 *c20 *z11 + a20 *c01 *z20 + a11 *c10 *z20 +...
    a10* c11 *z20 + a01 *c20 *z20;
e32=-b20* d11*y01 - b11 *d20*y01 - b20 *d10 *y02 - b10 *d20 *y02 -...
    b20 *d02* y10 - b11* d11 *y10 - b02 *d20* y10 - b20 *d01* y11 -...
    b11 *d10* y11 - b10* d11 *y11 - b01 *d20* y11 - b11 *d01* y20 -...
    b10 *d02* y20 - b02* d10 *y20 - b01 *d11* y20 + a20 *c11* z01 +...
    a11 *c20* z01 + a20* c10 *z02 + a10 *c20* z02 + a20 *c02* z10 +...
    a11 *c11* z10 + a02* c20 *z10 + a20 *c01* z11 + a11 *c10* z11 +...
    a10 *c11 *z11 + a01* c20 *z11 + a11 *c01* z20 + a10 *c02* z20 +...
    a02 *c10 *z20 + a01* c11 *z20;
e23=-b20* d02*y01 - b11 *d11 *y01 - b02 *d20 *y01 - b20* d01* y02 -...
    b11 *d10* y02 - b10 *d11 *y02 - b01*d20 *y02 - b11* d02* y10 -...
    b02 *d11* y10 - b11 *d01 *y11 - b10* d02 *y11 - b02 *d10* y11 -...
    b01 *d11* y11 - b02 *d01 *y20 - b01* d02* y20 + a20 *c02* z01 +...
    a11 *c11* z01 + a02 *c20 *z01 + a20* c01* z02 + a11 *c10* z02 +...
    a10 *c11* z02 + a01 *c20 *z02 + a11* c02* z10 + a02 *c11* z10 +...
    a11 *c01 *z11 + a10 *c02 *z11 + a02* c10* z11 + a01 *c11* z11 +...
    a02 *c01 *z20 + a01 *c02 *z20;
e14=-b11 *d02* y01 - b02* d11 *y01 - b11* d01* y02 - b10* d02 *y02 -...
    b02 *d10* y02 - b01* d11 *y02 - b02 *d02 *y10 - b02* d01 *y11 -...
    b01 *d02* y11 + a11* c02 *z01 + a02 *c11 *z01 + a11* c01 *z02 +...
    a10 *c02* z02 + a02* c10 *z02 + a01 *c11 *z02 + a02* c02 *z10 +...
    a02 *c01* z11 + a01* c02 *z11;
e05=-b02* d02* y01 - b02 *d01* y02 - b01* d02 *y02 + a02 *c02 *z01 +...
    a02 *c01 *z02 + a01 *c02* z02;
e60=-b20 *d20 *y20 + a20 *c20 *z20;
e51=-b20 *d20*y11 - b20 *d11*y20 - b11 *d20 *y20 + a20 *c20* z11 +...
    a20 *c11*z20 + a11 *c20 *z20;
e42=-b20 *d20 *y02 - b20 *d11* y11 - b11 *d20*y11 - b20* d02* y20 -...
    b11* d11 *y20 - b02*d20 *y20 + a20 *c20 *z02 + a20*c11 *z11 +...
    a11 *c20 *z11 + a20* c02 *z20 + a11 *c11 *z20 + a02* c20 *z20;
e33=-b20* d11*y02 - b11 *d20* y02 - b20 *d02 *y11 - b11 *d11 *y11 -...
    b02 *d20* y11 - b11 *d02 *y20 - b02 *d11* y20 + a20* c11* z02 +...
    a11 *c20* z02 + a20 *c02 *z11 + a11 *c11* z11 + a02* c20 *z11 +...
    a11 *c02* z20 + a02 *c11 *z20;
e24=-b20 *d02 *y02 - b11 *d11 *y02 - b02 *d20 *y02 - b11 *d02 +y11 -...
    b02 *d11 *y11 - b02* d02 *y20 + a20* c02*z02 + a11 *c11* z02 +...
    a02* c20*z02 + a11* c02 *z11 + a02 *c11 *z11 + a02 *c02 *z20;
e15=-b11 *d02* y02 - b02 *d11 *y02 - b02* d02* y11 + a11 *c02 *z02 +...
    a02 *c11* z02 + a02 *c02 *z11;
e06=-b02 *d02 *y02 + a02* c02* z02;
n00=1;
n10=b10 + d10;
n01=b01 + d01;
n20=b20 + b10 *d10 + d20;
n11=b11 + b10 *d01 + b01* d10 + d11;
```

```
478     n02=b02 + b01 *d01 + d02;
479     n30=b20* d10 + b10 *d20;
480     n21=b20* d01 + b11 *d10 + b10* d11 + b01 *d20;
481     n12=b11* d01 + b10 *d02 + b02* d10 + b01 *d11;
482     n03=b02* d01 + b01 *d02;
483     n40=b20* d20;
484     n31=b20* d11 + b11 *d20;
485     n22=b20* d02 + b11 *d11 + b02* d20;
486     n13=b11* d02 + b02 *d11;
487     n04=b02* d02;
488
489     elseif(operation==4) %Division
490
491     e00=a00 - c00* y00;
492     e10=a10 + a00 *d10 - b10 *c00 *y00 - c10 *y00 - c00 *y10 + a00 *z10;
493     e01=a01 + a00* d01 - b01 *c00 *y00 - c01* y00 - c00 *y01 + a00 *z01;
494     e20=a20 + a10* d10 + a00*d20 - b20* c00 *y00 - b10* c10*y00 - c20*y00-...
495         b10 *c00*y10 - c10 *y10 - c00* y20 + a10 *z10 + a00 *d10*z10 +  ...
496         a00* z20;
497     e11=a11 + a10*d01 + a01*d10 + a00*d11 - b11*c00*y00 - b10*c01*y00 - ...
498         b01* c10 *y00 - c11*y00 - b10*c00*y01 - c10*y01 - b01*c00*y10 -  ...
499         c01* y10 - c00 *y11 + a10* z01 + a00* d10* z01 + a01 *z10 +  ...
500         a00 *d01 *z10 + a00 *z11;
501     e02=a02 + a01* d01 + a00 *d02 - b02 *c00 *y00 - b01* c01 *y00 -  ...
502         c02*y00 - b01* c00 *y01 - c01* y01 - c00* y02 + a01 *z01 +  ...
503         a00 *d01 *z01 + a00* z02;
504     e30=a20* d10 + a10 *d20 - b20* c10* y00 - b10 *c20 *y00 -  ...
505         b20 *c00 *y10 - b10 *c10 *y10 - c20* y10 - b10* c00* y20 -  ...
506         c10* y20 + a20* z10 + a10* d10*z10 + a00 *d20*z10 + a10 *z20 +  ...
507         a00*d10* z20;
508     e21=a20* d01 + a11* d10 + a10* d11 + a01 *d20 -b20*c01 *y00 -  ...
509         b11 *c10* y00 - b10* c11*y00 - b01 *c20*y00 - b20* c00 *y01 -  ...
510         b10*c10 *y01 - c20* y01 - b11* c00* y10 - b10* c01* y10 -  ...
511         b01 *c10 *y10 - c11*y10 - b10*c00* y11 - c10* y11 - b01* c00*y20-...
512         c01 *y20 + a20* z01 + a10*d10*z01 + a00 *d20*z01 + a11* z10 +  ...
513         a10* d01 *z10 + a01 *d10 *z10 + a00* d11 *z10 + a10* z11 +  ...
514         a00 *d10* z11 + a01* z20 + a00 *d01 *z20;
515     e12=a11* d01 + a10 *d02 + a02* d10 + a01 *d11 - b11* c01 *y00 -  ...
516         b10 *c02* y00 - b02*c10 *y00 - b01 *c11* y00 - b11* c00 *01 -  ...
517         b10 *c01* y01 - b01* c10* y01 - c11* y01 - b10 *c00* y02 -  ...
518         c10* y02 - b02 *c00 *y10 - b01* c01 *y10 - c02* y10 -  ...
519         b01 *c00 *y11 - c01 *y11 + a11* z01 + a10* d01* z01 +  ...
520         a01 *d10 *z01 + a00 *d11* z01 + a10* z02 + a00 *d10 *z02 +  ...
521         a02* z10 + a01*  d01* z10 + a00*d02 *z10 + a01 *z11 + a00 *d01* z11;
522     e03=a02* d01 + a01* d02 - ...
523         b02 *c01 *y00 - b01* c02* y00 - b02* c00 *y01 - b01 *c01 *y01 -  ...
524         c02*y01 - b01 *c00*y02 - c01* y02 + a02 *z01 + a01*d01 *z01 +  ...
525         a00 *d02* z01 + a01* z02 + a00 *d01 *z02;
526     e40=a20 *d20 - b20 *c20* y00 - b20 *c10 *y10 - b10 *c20 *y10 - ...
527         b20* c00 *y20 - b10 *c10 *y20 - c20* y20 + a20 *d10 *z10 +  ...
528         a10* d20 *z10 + a20 *z20 + a10* d10 *z20 + a00* d20* z20;
529     e31=a20* d11 + a11* d20 - b20*c11*y00 - b11*c20*y00 - b20*c10*y01 -  ...
530         b10 *c20*y01 - b20 *c01*y10 - b11 *c10* y10 - b10*c11 *y10 -  ...
531         b01*c20*y10 - b20*c00*y11 - b10*c10*y11 - c20*y11 - b11*c00*y20 -...
```

```
532         b10 *c01* y20 - b01* c10 *y20 - c11 *y20 + a20* d10* z01 + ...
533         a10* d20* z01 + a20* d01* z10 + a11 *d10 *z10 + a10 *d11 *z10 + ...
534         a01* d20* z10 + a20* z11* + a10* d10 *z11 + a00*d20 *z11 +  ...
535         a11 *z20 + a10 *d01* z20 + a01* d10* z20 + a00* d11* z20;
536     e22=a20* d02 + a11* d11 + ...
537         a02* d20 - b20* c02* y00 - b11*c11 *y00 - b02* c20* y00 -  ...
538         b20 *c01 *y01 - b11 *c10 *y01 - b10 *c11*y01 - b01 *c20 *y01 -  ...
539         b20 *c00 *y02 - b10 *c10 *y02 - c20 *y02 - b11*c01 *y10 -  ...
540         b10* c02 *y10 - b02* c10 *y10 - b01 *c11 *y10 - b11* c00* y11 -  ...
541         b10 *c01* y11- b01* c10* y11 - c11* y11 - b02* c00 *y20 -  ...
542         b01*c01 *y20 - c02*y20 + a20 *d01* z01 + a11* d10 *z01 +  ...
543         a10* d11* z01 + a01* d20* z01 + a20 *z02 + a10 *d10*z02 +  ...
544         a00* d20* z02 + a11*d01 *z10 + a10* d02* z10 + a02 *d10*z10 +  ...
545         a01 *d11 *z10 + a11 *z11 + a10 *d01* z11 + a01 *d10* z11 +  ...
546         a00* d11 *z11 + a02* z20 + a01 *d01*z20 + a00* d02 *z20;
547     e13=a11 *d02 + a02 *d11 -b11 *c02 *y00 - b02 *c11*y00 - b11*c01*y01 -...
548         b10 *c02 *y01 - b02* c10* y01 - b01 *c11 *y01 - b11 *c00 *y02 -  ...
549         b10* c01* y02 - b01 *c10 *y02 - c11 *y02 - b02 *c01* y10 - ...
550         b01* c02 *y10 - b02*c00 *y11 - b01*c01*y11 - c02 *y11 +  ...
551         a11 *d01 *z01 + a10 *d02 *z01 + a02* d10* z01 + a01 *d11* z01 +  ...
552         a11* z02 + a10*d01* z02 + a01 *d10 *z02 + a00 *d11 *z02 +  ...
553         a02* d01* z10 + a01 *d02 *z10 + a02 *z11 + a01 *d01 *z11 +  ...
554         a00 *d02 *z11;
555     e04=a02* d02 - b02* c02* y00 - b02 *c01*y01 - b01 *c02*y01 -  ...
556         b02* c00* y02 - b01 *c01 *y02 - c02* y02 + a02 *d01* z01 +  ...
557         a01* d02 *z01 + a02 *z02 + a01* d01*z02 + a00 *d02 *z02;
558     e50=-b20* c20 *y10 - b20* c10* y20 - b10 *c20 *y20 + a20* d20 *z10 +...
559         a20* d10* z20 + a10 *d20* z20;
560     e41=-b20 *c20*y01 - b20 *c11 *y10 - b11 *c20 *y10 - b20* c10* y11 -...
561         b10* c20* y11 - b20 *c01* y20 - b11 *c10* y20 - b10 *c11 *y20 -...
562         b01* c20* y20 + a20 *d20* z01 + a20 *d11* z10 + a11 *d20 *z10 +...
563         a20* d10* z11 + a10 *d20* z11 + a20 *d01* z20 + a11 *d10 *z20 +...
564         a10* d11* z20 + a01 *d20* z20;
565     e32=-b20* c11* y01 - b11 *c20* y01 - b20 *c10* y02 - b10 *c20 *y02 -...
566         b20 *c02* y10 - b11 *c11 *y10 - b02 *c20* y10 - b20 *c01 *y11 -...
567         b11 *c10* y11 - b10 *c11 *y11 - b01 *c20* y11 - b11 *c01 *y20 -...
568         b10 *c02* y20 - b02 *c10 *y20 - b01 *c11* y20 + a20 *d11 *z01 +...
569         a11 *d20* z01 + a20 *d10 *z02 + a10 *d20* z02 + a20 *d02 *z10 +...
570         a11 *d11* z10 + a02 *d20 *z10 + a20 *d01* z11 + a11 *d10 *z11 +...
571         a10 *d11* z11 + a01 *d20 *z11 + a11 *d01* z20 + a10 *d02 *z20 +...
572         a02 *d10* z20 + a01 *d11 *z20;
573     e23=-b20 *c02 *y01 - b11* c11 *y01 - b02* c20* y01 - b20* c01 *y02 -...
574         b11 *c10 *y02 - b10* c11* y02 - b01* c20 *y02 - b11 *c02 *y10 -...
575         b02 *c11 *y10 - b11* c01* y11 - b10* c02 *y11 - b02 *c10 *y11 -...
576         b01 *c11 *y11 - b02* c01* y20 - b01* c02 *y20 + a20 *d02 *z01 +...
577         a11 *d11 *z01 + a02* d20* z01 + a20* d01 *z02 + a11 *d10 *z02 +...
578         a10 *d11 *z02 + a01* d20* z02 + a11* d02 *z10 + a02 *d11 *z10 +...
579         a11 *d01 *z11 + a10* d02* z11 + a02* d10 *z11 + a01 *d11 *z11 +...
580         a02 *d01 *z20 + a01* d02* z20;
581     e14=-b11* c02* y01 - b02* c11 *y01 - b11* c01 *y02 - b10* c02*y02 -...
582         b02 *c10 *y02 - b01* c11 *y02 - b02* c02 *y10 - b02 *c01 *y11 -...
583         b01 *c02 *y11 + a11* d02 *z01 + a02* d11 *z01 + a11 *d01 *z02 +...
584         a10 *d02 *z02 + a02* d10 *z02 + a01* d11 *z02 + a02 *d02 *z10 +...
585         a02 *d01 *z11 + a01* d02 *z11;
```

```
    e05=-b02 *c02 *y01 - b02 *c01 *y02 - b01* c02* y02 + a02 *d02 *z01 +...
        a02*d01 *z02 + a01*d02* z02;
    e60=-b20* c20*y20 + a20* d20 *z20;
    e51=-b20* c20* y11 - b20* c11 *y20 - b11* c20 *y20 + a20* d20 *z11 +...
        a20*d11* z20 + a11 *d20* z20;
    e42=-b20* c20* y02 - b20 *c11* y11 - b11* c20 *y11 - b20 *c02 *y20 -...
        b11 *c11*y20 - b02* c20* y20 + a20* d20 *z02 + a20* d11 *z11 +...
        a11*d20 *z11 + a20* d02* z20 + a11* d11 *z20 + a02 *d20 *z20;
    e33=-b20 *c11* y02 - b11* c20 *y02 - b20 *c02* y11 - b11 *c11 *y11 -...
        b02* c20* y11 - b11 *c02 *y20 - b02 *c11 *y20 + a20 *d11*z02 +...
        a11 *d20 *z02 + a20* d02* z11 + a11 *d11* z11 + a02 *d20 *z11 +...
        a11 *d02 *z20 + a02*d11*z20;
    e24=-b20* c02*y02 - b11* c11 *y02 - b02 *c20* y02 - b11 *c02* y11 -...
        b02 *c11 *y11 - b02* c02* y20 + a20* d02 *z02 + a11* d11* z02 +...
        a02*d20* z02 + a11 *d02*z11 + a02*d11 *z11 + a02 *d02 *z20;
    e15=-b11*c02 *y02 - b02 *c11* y02 - b02* c02 *y11 + a11 *d02 *z02 +...
        a02* d11* z02 + a02 *d02* z11;
    e06=b02* c02 *y02 + a02 *d02 *z02;
    n00=c00;
    n10=b10 *c00 + c10;
    n01=b01 *c00 + c01;
    n20=b20 *c00 + b10 *c10 + c20;
    n11=b11 *c00 + b10 *c01 + b01 *c10 + c11;
    n02=b02 *c00 + b01 *c01 + c02;
    n30=b20 *c10 + b10 *c20;
    n21=b20 *c01 + b11 *c10 + b10 *c11 + b01 *c20;
    n12=b11 *c01 + b10 *c02 + b02 *c10 + b01 *c11;
    n03=b02 *c01 + b01 *c02;
    n40=b20 *c20 ;
    n31=b20 *c11 + b11 *c20;
    n22=b20 *c02 + b11 *c11 + b02 *c20;
    n13=b11 *c02 + b02 *c11;
    n04=b02 *c02 ;

end;
```

A.19 zweiDfehlerslopemult.m

```
function [u,o] = zweiDfehlerslopemult ...
                    (pa00,pa10,pa01,pa20,pa11,pa02,qa10,qa01,qa20,qa11,...
                    qa02,ua,oa,operation,pb00,pb10,pb01,pb20,pb11,pb02,...
                    qb10,qb01,qb20,qb11,qb02,ub,ob,p00,p10,p01,p20,p11,...
                    p02,q10,q01,q20,q11,q02,Ixu,Ixo,Iyu,Iyo)

Ix=infsup(Ixu,Ixo);
Iy=infsup(Iyu,Iyo);
I1=infsup(0,0);
I2=infsup(0,0);
I3=infsup(0,0);
I4=infsup(0,0);
Ia=infsup(ua,oa);
```

```
Ib=infsup(ub,ob);

%Ausmultiplizieren des Problems, man erhält die Koeffizienten eines
%Polynoms sechster und eines vierter Ordnung.
[e00,e10,e01,e20,e11,e02,e30,e21,e12,e03,e40,e31,e22,e13,e04,...
 e50,e41,e32,e23,e14,e05,e60,e51,e42,e33,e24,e15,e06,...
 n00,n10,n01,n20,n11,n02,n30,n21,n12,n03,n40,n31,n22,n13,n04]= ...
     zweidimausmultipl...
      (pa00+Ia,pa10,pa01,pa20,pa11,pa02,qa10,qa01,qa20,qa11,qa02,operation,...
       pb00+Ib,pb10,pb01,pb20,pb11,pb02,qb10,qb01,qb20,qb11,qb02,...
       p00,p10,p01,p20,p11,p02,q10,q01,q20,q11,q02);

za1=Ixu;
za2=Iyu;
zb1=Ixo;
zb2=Iyo;

%Berechnung von  f(za)
fza=(e00+e10*za1+e01*za2+e20*za1^2+e11*za1*za2+e02*za2^2+e30*za1^3+...
     e21*za1^2*za2+e12*za1*za2^2+e03*za2^3+e40*za1^4+e31*za1^3*za2+...
     e22*za1^2*za2^2+e13*za1*za2^3+e04*za2^4+e50*za1^5+e41*za1^4*za2+...
     e32*za1^3*za2^2+e23*za1^2*za2^3+e14*za1*za2^4+e05*za2^5+e60*za1^6+...
     e51*za1^5*za2+e42*za1^4*za2^2+e33*za1^3*za2^3+e24*za1^2*za2^4+...
     e15*za1*za2^5+e06*za2^6)/...
   (n00+n10*za1+n01*za2+n20*za1^2+n11*za1*za2+n02*za2^2+n30*za1^3+...
    n21*za1^2*za2+n12*za1*za2^2+n03*za2^3+n40*za1^4+n31*za1^3*za2+...
    n22*za1^2*za2^2+n13*za1*za2^3+n04*za2^4);

%Berechnung von  f(zb)
fzb=(e00+e10*zb1+e01*zb2+e20*zb1^2+e11*zb1*zb2+e02*zb2^2+e30*zb1^3+...
     e21*zb1^2*zb2+e12*zb1*zb2^2+e03*zb2^3+e40*zb1^4+e31*zb1^3*zb2+...
     e22*zb1^2*zb2^2+e13*zb1*zb2^3+e04*zb2^4+e50*zb1^5+e41*zb1^4*zb2+...
     e32*zb1^3*zb2^2+e23*zb1^2*zb2^3+e14*zb1*zb2^4+e05*zb2^5+e60*zb1^6+...
     e51*zb1^5*zb2+e42*zb1^4*zb2^2+e33*zb1^3*zb2^3+e24*zb1^2*zb2^4+...
     e15*zb1*zb2^5+e06*zb2^6)/...
    (n00+n10*zb1+n01*zb2+n20*zb1^2+n11*zb1*zb2+n02*zb2^2+n30*zb1^3+...
     n21*zb1^2*zb2+n12*zb1*zb2^2+n03*zb2^3+n40*zb1^4+n31*zb1^3*zb2+...
     n22*zb1^2*zb2^2+n13*zb1*zb2^3+n04*zb2^4);

%Berechnung des Slopes erster Ordnung (f[za,x])
[s1a00,s1a10,s1a01,s1a20,s1a11,s1a02,s1a30,s1a21,s1a12,s1a03,s1a40,...
 s1a31,s1a22,s1a13,s1a04,s1a50,s1a41,s1a32,s1a23,s1a14,s1a05,z1a00,...
 z1a10,z1a01,z1a20,z1a11,z1a02,z1a30,z1a21,z1a12,z1a03,z1a40,z1a31,...
 z1a22,z1a13,z1a04,s1b00,s1b10,s1b01,s1b20,s1b11,s1b02,s1b30,s1b21,...
 s1b12,s1b03,s1b40,s1b31,s1b22,s1b13,s1b04,s1b50,s1b41,s1b32,s1b23,...
 s1b14,s1b05,z1b00,z1b10,z1b01,z1b20,z1b11,z1b02,z1b30,z1b21,z1b12,...
 z1b03,z1b40,z1b31,z1b22,z1b13,z1b04...
]=...
zweiDmultslope(e00,e10,e01,e20,e11,e02,e30,e21,e12,e03,e40,e31,e22,e13,e04,...
               e50,e41,e32,e23,e14,e05,e60,e51,e42,e33,e24,e15,e06,...
               n00,n10,n01,n20,n11,n02,n30,n21,n12,n03,n40,n31,n22,n13,n04,...
               za1,za2);

s1a60=0;
```

```
s1a51=0;
s1a42=0;
s1a33=0;
s1a24=0;
s1a15=0;
s1a06=0;
s1b60=0;
s1b51=0;
s1b42=0;
s1b33=0;
s1b24=0;
s1b15=0;
s1b06=0;

%Berechnung des Slopes erster Ordnung (f[zb,x])
[s2a00,s2a10,s2a01,s2a20,s2a11,s2a02,s2a30,s2a21,s2a12,s2a03,s2a40,...
 s2a31,s2a22,s2a13,s2a04,s2a50,s2a41,s2a32,s2a23,s2a14,s2a05,z2a00,...
 z2a10,z2a01,z2a20,z2a11,z2a02,z2a30,z2a21,z2a12,z2a03,z2a40,z2a31,...
 z2a22,z2a13,z2a04,s2b00,s2b10,s2b01,s2b20,s2b11,s2b02,s2b30,s2b21,...
 s2b12,s2b03,s2b40,s2b31,s2b22,s2b13,s2b04,s2b50,s2b41,s2b32,s2b23,...
 s2b14,s2b05,z2b00,z2b10,z2b01,z2b20,z2b11,z2b02,z2b30,z2b21,z2b12,...
 z2b03,z2b40,z2b31,z2b22,z2b13,z2b04...
 ]=...
 zweiDmultslope(e00,e10,e01,e20,e11,e02,e30,e21,e12,e03,e40,e31,e22,e13,e04,...
                e50,e41,e32,e23,e14,e05,e60,e51,e42,e33,e24,e15,e06,...
                n00,n10,n01,n20,n11,n02,n30,n21,n12,n03,n40,n31,n22,n13,n04,...
                zb1,zb2);

%Berechnung von f[za,zb]
fzazb=[(s1a00+s1a10*zb1+s1a01*zb2+s1a20*zb1^2+s1a11*zb1*zb2+s1a02*zb2^2+...
        s1a30*zb1^3+s1a21*zb1^2*zb2+s1a12*zb1*zb2^2+s1a03*zb2^3+...
        s1a40*zb1^4+s1a31*zb1^3*zb2+s1a22*zb1^2*zb2^2+s1a13*zb1*zb2^3+...
        s1a04*zb2^4+s1a50*zb1^5+s1a41*zb1^4*zb2+s1a32*zb1^3*zb2^2+...
        s1a23*zb1^2*zb2^3+s1a14*zb1*zb2^4+s1a05*zb2^5+s1a60*zb1^6+...
        s1a51*zb1^5*zb2+s1a42*zb1^4*zb2^2+s1a33*zb1^3*zb2^3+...
        s1a24*zb1^2*zb2^4+s1a15*zb1*zb2^5+s1a06*zb2^6)/...
       (z1a00+z1a10*zb1+z1a01*zb2+z1a20*zb1^2+z1a11*zb1*zb2+z1a02*zb2^2+...
        z1a30*zb1^3+z1a21*zb1^2*zb2+z1a12*zb1*zb2^2+z1a03*zb2^3+...
        z1a40*zb1^4+z1a31*zb1^3*zb2+z1a22*zb1^2*zb2^2+z1a13*zb1*zb2^3+...
        z1a04*zb2^4),(s1b00+s1b10*zb1+s1b01*zb2+s1b20*zb1^2+s1b11*zb1*zb2+...
        s1b02*zb2^2+s1b30*zb1^3+s1b21*zb1^2*zb2+s1b12*zb1*zb2^2+s1b03*zb2^3+...
        s1b40*zb1^4+s1b31*zb1^3*zb2+s1b22*zb1^2*zb2^2+s1b13*zb1*zb2^3+...
        s1b04*zb2^4+s1b50*zb1^5+s1b41*zb1^4*zb2+s1b32*zb1^3*zb2^2+...
        s1b23*zb1^2*zb2^3+s1b14*zb1*zb2^4+s1b05*zb2^5+s1b60*zb1^6+...
        s1b51*zb1^5*zb2+s1b42*zb1^4*zb2^2+s1b33*zb1^3*zb2^3+...
        s1b24*zb1^2*zb2^4+s1b15*zb1*zb2^5+s1b06*zb2^6)/...
       (z1b00+z1b10*zb1+z1b01*zb2+z1b20*zb1^2+z1b11*zb1*zb2+z1b02*zb2^2+...
        z1b30*zb1^3+z1b21*zb1^2*zb2+z1b12*zb1*zb2^2+z1b03*zb2^3+...
        z1b40*zb1^4+z1b31*zb1^3*zb2+z1b22*zb1^2*zb2^2+z1b13*zb1*zb2^3+...
        z1b04*zb2^4)];

%Berechnung der Koeffizienten der ersten Spalte
%des Slopes zweiter Ordnung (f[z,z',x])
[s3a00,s3a10,s3a01,s3a20,s3a11,s3a02,s3a30,s3a21,s3a12,s3a03,s3a40,...
```

```
 s3a31,s3a22,s3a13,s3a04,s3a50,s3a41,s3a32,s3a23,s3a14,s3a05,z3a00,...
 z3a10,z3a01,z3a20,z3a11,z3a02,z3a30,z3a21,z3a12,z3a03,z3a40,z3a31,...
 z3a22,z3a13,z3a04,s3b00,s3b10,s3b01,s3b20,s3b11,s3b02,s3b30,s3b21,...
 s3b12,s3b03,s3b40,s3b31,s3b22,s3b13,s3b04,s3b50,s3b41,s3b32,s3b23,...
 s3b14,s3b05,z3b00,z3b10,z3b01,z3b20,z3b11,z3b02,z3b30,z3b21,z3b12,...
 z3b03,z3b40,z3b31,z3b22,z3b13,z3b04...
]=...
 zweiDmultslope(s1a00,s1a10,s1a01,s1a20,s1a11,s1a02,s1a30,s1a21,s1a12,...
                s1a03,s1a40,s1a31,s1a22,s1a13,s1a04,s1a50,s1a41,s1a32,...
                s1a23,s1a14,s1a05,0,0,0,0,0,0,0,z1a00,z1a10,z1a01,z1a20,...
                z1a11,z1a02,z1a30,z1a21,z1a12,z1a03,z1a40,z1a31,z1a22,...
                z1a13,z1a04,zb1,zb2);



%Berechnung der Koeffizienten der zweiten Spalte
%des Slopes zweiter Ordnung (f[z,z',x])
[s3c00,s3c10,s3c01,s3c20,s3c11,s3c02,s3c30,s3c21,s3c12,s3c03,s3c40,...
 s3c31,s3c22,s3c13,s3c04,s3c50,s3c41,s3c32,s3c23,s3c14,s3c05,z3c00,...
 z3c10,z3c01,z3c20,z3c11,z3c02,z3c30,z3c21,z3c12,z3c03,z3c40,z3c31,...
 z3c22,z3c13,z3c04,s3d00,s3d10,s3d01,s3d20,s3d11,s3d02,s3d30,s3d21,...
 s3d12,s3d03,s3d40,s3d31,s3d22,s3d13,s3d04,s3d50,s3d41,s3d32,s3d23,...
 s3d14,s3d05,z3d00,z3d10,z3d01,z3d20,z3d11,z3d02,z3d30,z3d21,z3d12,...
 z3d03,z3d40,z3d31,z3d22,z3d13,z3d04...
]=...
zweiDmultslope(s1b00,s1b10,s1b01,s1b20,s1b11,s1b02,s1b30,s1b21,s1b12,...
               s1b03,s1b40,s1b31,s1b22,s1b13,s1b04,s1b50,s1b41,s1b32,...
               s1b23,s1b14,s1b05,0,0,0,0,0,0,0,z1b00,z1b10,z1b01,z1b20,...
               z1b11,z1b02,z1b30,z1b21,z1b12,z1b03,z1b40,z1b31,z1b22,...
               z1b13,z1b04,zb1,zb2);

%Einsetzen des Defintionsbereiches in f(x)
I1=I1+(e00+e10*Ix+e01*Iy+e20*power(Ix,2)+e11*Ix*Iy+e02*power(Iy,2)+...
   e30*power(Ix,3)+e21*power(Ix,2)*Iy+e12*Ix*power(Iy,2)+e03*power(Iy,3)+...
   e40*power(Ix,4)+e31*power(Ix,3)*Iy+e22*power(Ix,2)*power(Iy,2)+...
   e13*Ix*power(Iy,3)+e04*power(Iy,4)+e50*power(Ix,5)+e41*power(Ix,4)*Iy+...
   e32*power(Ix,3)*power(Iy,2)+e23*power(Ix,2)*power(Iy,3)+...
   e14*Ix*power(Iy,4)+e05*power(Ix,5)+e60*power(Ix,6)+ e51*power(Ix,5)*Iy+...
   e42*power(Ix,4)*power(Iy,2)+e33*power(Ix,3)*power(Iy,3)+...
   e24*power(Ix,2)*power(Iy,4)+e15*Ix*power(Iy,5)+e06*power(Iy,6))/...
  (n00+n10*Ix+n01*Iy+n20*power(Ix,2)+n11*Ix*Iy+n02*power(Iy,2)+...
   n30*power(Ix,3)+n21*power(Ix,2)*Iy+n12*Ix*power(Iy,2)+n03*power(Iy,3)+...
   n40*power(Ix,4)+n31*power(Ix,3)*Iy+n22*power(Ix,2)*power(Iy,2)+...
   n13*Ix*power(Iy,3)+n04*power(Iy,4)) ;

%Abschätzen des Fehlerintervalls mittels Slopes erster Ordnung
%an den Punkte za (bei I2) bzw. zb (bei I3).
I2=I2+(fza+[(s1a00+s1a10*Ix+s1a01*Iy+s1a20*power(Ix,2)+s1a11*Ix*Iy+...
   s1a02*power(Iy,2)+s1a30*power(Ix,3)+s1a21*power(Ix,2)*Iy+...
   s1a12*Ix*power(Iy,2)+s1a03*power(Iy,3)+s1a40*power(Ix,4)+...
   s1a31*power(Ix,3)*Iy+s1a22*power(Ix,2)*power(Iy,2)+s1a13*Ix*power(Iy,3)+...
   s1a04*power(Iy,4)+s1a50*power(Ix,5)+s1a41*power(Ix,4)*Iy+...
   s1a32*power(Ix,3)*power(Iy,2)+s1a23*power(Ix,2)*power(Iy,3)+...
   s1a14*Ix*power(Iy,4)+s1a05*power(Ix,5))/...
```

```
  (z1a00+z1a10*Ix+n01*Iy+z1a20*power(Ix,2)+z1a11*Ix*Iy+z1a02*power(Iy,2)+...
   z1a30*power(Ix,3)+z1a21*power(Ix,2)*Iy+z1a12*Ix*power(Iy,2)+ ...
   z1a03*power(Iy,3)+z1a40*power(Ix,4)+z1a31*power(Ix,3)*Iy+...
   z1a22*power(Ix,2)*power(Iy,2)+z1a13*Ix*power(Iy,3)+z1a04*power(Iy,4)),...
  (s1b00+s1b10*Ix+s1b01*Iy+s1b20*power(Ix,2)+s1b11*Ix*Iy+s1b02*power(Iy,2)+...
   s1b30*power(Ix,3)+s1b21*power(Ix,2)*Iy+s1b12*Ix*power(Iy,2)+...
   s1b03*power(Iy,3)+s1b40*power(Ix,4)+s1b31*power(Ix,3)*Iy+...
   s1b22*power(Ix,2)*power(Iy,2)+s1b13*Ix*power(Iy,3)+s1b04*power(Iy,4)+...
   s1b50*power(Ix,5)+s1b41*power(Ix,4)*Iy+s1b32*power(Ix,3)*power(Iy,2)+...
   s1b23*power(Ix,2)*power(Iy,3)+s1b14*Ix*power(Iy,4)+s1b05*power(Iy,5))/...
  (z1b00+z1b10*Ix+z1b01*Iy+z1b20*power(Ix,2)+z1b11*Ix*Iy+z1b02*power(Iy,2)+...
   z1b30*power(Ix,3)+z1b21*power(Ix,2)*Iy+z1b12*Ix*power(Iy,2)+ ...
   z1b03*power(Iy,3)+z1b40*power(Ix,4)+z1b31*power(Ix,3)*Iy+...
   z1b22*power(Ix,2)*power(Iy,2)+z1b13*Ix*power(Iy,3)+z1b04*power(Iy,4))]*...
  [Ix-za1;Iy-za2]);

I3=I3+(fzb+[(s2a00+s2a10*Ix+s2a01*Iy+s2a20*power(Ix,2)+s2a11*Ix*Iy+...
   s2a02*power(Iy,2)+s2a30*power(Ix,3)+s2a21*power(Ix,2)*Iy+...
   s2a12*Ix*power(Iy,2)+s2a03*power(Iy,3)+s2a40*power(Ix,4)+...
   s2a41*power(Ix,4)*Iy+s2a32*power(Ix,3)*power(Iy,2)+...
   s2a23*power(Ix,2)*power(Iy,3)+s2a14*Ix*power(Iy,4)+s2a05*power(Ix,5))/...
  (z2a00+z2a10*Ix+n01*Iy+z2a20*power(Ix,2)+z2a11*Ix*Iy+z2a02*power(Iy,2)+...
   z2a30*power(Ix,3)+z2a21*power(Ix,2)*Iy+z2a12*Ix*power(Iy,2)+ ...
   z2a03*power(Iy,3)+z2a40*power(Ix,4)+z2a31*power(Ix,3)*Iy+...
   z2a22*power(Ix,2)*power(Iy,2)+z2a13*Ix*power(Iy,3)+z2a04*power(Iy,4)),...
  (s2b00+s2b10*Ix+s2b01*Iy+s2b20*power(Ix,2)+s2b11*Ix*Iy+s2b02*power(Iy,2)+...
   s2b30*power(Ix,3)+s2b21*power(Ix,2)*Iy+s2b12*Ix*power(Iy,2)+...
   s2b03*power(Iy,3)+s2b40*power(Ix,4)+s2b31*power(Ix,3)*Iy+...
   s2b22*power(Ix,2)*power(Iy,2)+s2b13*Ix*power(Iy,3)+s2b04*power(Iy,4)+...
   s2b50*power(Ix,5)+s2b41*power(Ix,4)*Iy+s2b32*power(Ix,3)*power(Iy,2)+...
   s2b23*power(Ix,2)*power(Iy,3)+s2b14*Ix*power(Iy,4)+s2b05*power(Iy,5))/...
  (z2b00+z2b10*Ix+z2b01*Iy+z2b20*power(Ix,2)+z2b11*Ix*Iy+z2b02*power(Iy,2)+...
   z2b30*power(Ix,3)+z2b21*power(Ix,2)*Iy+z2b12*Ix*power(Iy,2)+ ...
   z2b03*power(Iy,3)+z2b40*power(Ix,4)+z2b31*power(Ix,3)*Iy+...
   z2b22*power(Ix,2)*power(Iy,2)+z2b13*Ix*power(Iy,3)+z2b04*power(Iy,4))]*...
  [Ix-zb1;Iy-zb2]);

 %Abschätzen des Fehlerintervalls mittels Slopes zweiter Ordnung
 I4=  I4+(fza+((fzazb+[Ix-zb1;Iy-zb2]'*...
    [(s3a00+s3a10*Ix+s3a01*Iy+s3a20*power(Ix,2)+s3a11*Ix*Iy+...
      s3a02*power(Iy,2)+s3a30*power(Ix,3)+s3a21*power(Ix,2)*Iy+...
      s3a12*Ix*power(Iy,2)+s3a03*power(Iy,3)+s3a40*power(Ix,4)+...
      s3a31*power(Ix,3)*Iy+s3a22*power(Ix,2)*power(Iy,2)+...
      s3a13*Ix*power(Iy,3)+s3a04*power(Iy,4)+s3a50*power(Ix,5)+ ...
      s3a41*power(Ix,4)*Iy+s3a32*power(Ix,3)*power(Iy,2)+...
      s3a23*power(Ix,2)*power(Iy,3)+s3a14*Ix*power(Iy,4)+s3a05*power(Iy,5))/...
     (z3a00+z3a10*Ix+z3a01*Iy+z3a20*power(Ix,2)+z3a11*Ix*Iy+...
      z3a02*power(Iy,2)+z3a30*power(Ix,3)+z3a21*power(Ix,2)*Iy+...
      z3a12*Ix*power(Iy,2)+z3a03*power(Iy,3)+z3a40*power(Ix,4)+...
      z3a31*power(Ix,3)*Iy+z3a22*power(Ix,2)*power(Iy,2)+...
      z3a13*Ix*power(Iy,3)+z3a04*power(Iy,4)),...
     (s3c00+s3c10*Ix+s3c01*Iy+s3c20*power(Ix,2)+s3c11*Ix*Iy+...
      s3c02*power(Iy,2)+s3c30*power(Ix,3)+s3c21*power(Ix,2)*Iy+...
      s3c12*Ix*power(Iy,2)+s3c03*power(Iy,3)+s3c40*power(Ix,4)+...
```

```
      s3c31*power(Ix,3)*Iy+s3c22*power(Ix,2)*power(Iy,2)+...
      s3c13*Ix*power(Iy,3)+s3c04*power(Iy,4)+s3c50*power(Ix,5)+ ...
      s3c41*power(Ix,4)*Iy+s3c32*power(Ix,3)*power(Iy,2)+...
      s3c23*power(Ix,2)*power(Iy,3)+s3c14*Ix*power(Iy,4)+s3c05*power(Iy,5))/...
     (z3c00+z3c10*Ix+z3c01*Iy+z3c20*power(Ix,2)+z3c11*Ix*Iy+...
     z3c02*power(Iy,2)+z3c30*power(Ix,3)+z3c21*power(Ix,2)*Iy+...
     z3c12*Ix*power(Iy,2)+z3c03*power(Iy,3)+z3c40*power(Ix,4)+...
     z3c31*power(Ix,3)*Iy+z3c22*power(Ix,2)*power(Iy,2)+...
     z3c13*Ix*power(Iy,3)+z3c04*power(Iy,4));...
    (s3b00+s3a10*Ix+s3b01*Iy+s3b20*power(Ix,2)+s3b11*Ix*Iy+...
     s3b02*power(Iy,2)+s3b30*power(Ix,3)+s3b21*power(Ix,2)*Iy+...
     s3b12*Ix*power(Iy,2)+s3b03*power(Iy,3)+s3b40*power(Ix,4)+...
     s3b31*power(Ix,3)*Iy+s3b22*power(Ix,2)*power(Iy,2)+...
     s3b13*Ix*power(Iy,3)+s3b04*power(Iy,4)+s3b50*power(Ix,5)+ ...
     s3b41*power(Ix,4)*Iy+s3b32*power(Ix,3)*power(Iy,2)+...
     s3b23*power(Ix,2)*power(Iy,3)+s3b14*Ix*power(Iy,4)+s3b05*power(Iy,5))/...
    (z3b00+z3b10*Ix+z3b01*Iy+z3b20*power(Ix,2)+z3b11*Ix*Iy+...
     z3b02*power(Iy,2)+z3b30*power(Ix,3)+z3b21*power(Ix,2)*Iy+...
     z3b12*Ix*power(Iy,2)+z3b03*power(Iy,3)+z3b40*power(Ix,4)+...
     z3b31*power(Ix,3)*Iy+z3b22*power(Ix,2)*power(Iy,2)+...
     z3b13*Ix*power(Iy,3)+z3b04*power(Iy,4)),...
    (s3d00+s3d10*Ix+s3d01*Iy+s3d20*power(Ix,2)+s3d11*Ix*Iy+...
     s3d02*power(Iy,2)+s3d30*power(Ix,3)+s3d21*power(Ix,2)*Iy+...
     s3d12*Ix*power(Iy,2)+s3d03*power(Iy,3)+s3d40*power(Ix,4)+...
     s3d31*power(Ix,3)*Iy+s3d22*power(Ix,2)*power(Iy,2)+...
     s3d13*Ix*power(Iy,3)+s3d04*power(Iy,4)+s3d50*power(Ix,5)+ ...
     s3d41*power(Ix,4)*Iy+s3d32*power(Ix,3)*power(Iy,2)+...
     s3d23*power(Ix,2)*power(Iy,3)+s3d14*Ix*power(Iy,4)+s3d05*power(Iy,5))/...
    (z3d00+z3d10*Ix+z3d01*Iy+z3d20*power(Ix,2)+z3d11*Ix*Iy+...
     z3d02*power(Iy,2)+z3d30*power(Ix,3)+z3d21*power(Ix,2)*Iy+...
     z3d12*Ix*power(Iy,2)+z3d03*power(Iy,3)+z3d40*power(Ix,4)+...
     z3d31*power(Ix,3)*Iy+z3d22*power(Ix,2)*power(Iy,2)+...
     z3d13*Ix*power(Iy,3)+z3d04*power(Iy,4))]) *...
     [Ix-za1;Iy-za2]));

%Schneiden der Intervalle
I5=intersect(intersect(I1,I2),intersect(I3,I4));

u=inf_(I5);
o=sup(I5);
```

A.20 zweiDmultslope.m

```
function [sa00,sa10,sa01,sa20,sa11,sa02,sa30,sa21,sa12,sa03,sa40,sa31,...
          sa22,sa13,sa04,sa50,sa41,sa32,sa23,sa14,sa05,za00,za10,za01,...
          za20,za11,za02,za30,za21,za12,za03,za40,za31,za22,za13,za04...
          sb00,sb10,sb01,sb20,sb11,sb02,sb30,sb21,sb12,sb03,sb40,sb31,...
          sb22,sb13,sb04,sb50,sb41,sb32,sb23,sb14,sb05,zb00,zb10,zb01,...
          zb20,zb11,zb02,zb30,zb21,zb12,zb03,zb40,zb31,zb22,zb13,zb04]=...
zweiDmultslope(e00,e10,e01,e20,e11,e02,e30,e21,e12,e03,e40,e31,e22,e13,...
               e04,e50,e41,e32,e23,e14,e05,e60,e51,e42,e33,e24,e15,e06,...
```

```
9                n00,n10,n01,n20,n11,n02,n30,n21,n12,n03,n40,n31,n22,n13,...
10               n04,z1,z2)
11
12 %Berechnung der Koeffizienten eines Slopes
13
14 sa00=(e10 *n00 - e00 *n10 + e20 *n00 *z1 - e00*n20*z1 + e30*n00*z1^2 - ...
15       e00*n30*z1^2 + e40* n00* z1^3 - e00* n40 *z1^3 + e50 *n00* z1^4 + ...
16       e60 *n00* z1^5 + e10* n01 *z2 - e01* ...
17       n10 *z2 + e20 *n01*z1 *z2 - e01* n20* z1*z2 + e30*n01*z1^2*z2-e01*...
18       n30* z1^2* z2 + e40 *n01*z1^3 *z2 - e01 *n40* z1^3*z2 + e50* ...
19       n01 *z1^4* z2 + e12*n00 *z2^2 + e10 *n02* z2^2 - e02* n10 *...
20       z2^2 - e00 *n12*z2^2 + e20 *n02* z1*z2^2 - e02*n20*z1 *z2^2 +e30*...
21       n02 *z1^2 *z2^2 - e02* n30*z1^2 *z2^2 + ...
22       e40*n02*z1^3* z2^2 - e02* n40 *z1^3* z2^2 + e13 *n00 *z2^3 + e10 *...
23       n03*z2^3 - e03* n10* z2^3 - e00* n13* z2^3 + e23 *n00 *...
24       z1* z2^3 + e20* n03 *z1 *z2^3 - e03 *n20* ...
25       z1 *z2^3 + e30 *n03 *z1^2 *2^3 - e03* n30* z1^2* z2^3 + e14* n00* ...
26       z2^4 + e10* n04* z2^4 - e04* n10* z2^4 +...
27       e24* n00*z1* z2^4 + e20* n04* z1* z2^4 -...
28       e04* n20* z1* z2^4 + e15* n00* z2^5 - e05* n10* z2^5);
29
30 sa10=(e20* n00 - e00* n20 + e30* n00* z1 +  ...
31       e20* n10* z1 - e10* n20* z1 - e00* n30* z1 + e40*n00* z1^2 + ...
32       e30* n10*z1^2 - e10* n30* z1^2 - e00* n40*  ...
33       z1^2 + e50*n00* z1^3 + e40* n10* z1^3 - e10*n40* z1^3 + e60* n00* ...
34       z1^4 + e50* n10*z1^4 + e60* n10* z1^5 + e20* n01* z2 - e01* n20*  ...
35       z2 + e30* n01* z1* z2 + e20* n11*z1* z2 + e21* n11* z1* z2 - ...
36       e11* n20*z1* z2 - e01* n30* z1* z2 + e40* n01* z1^2* z2 + ...
37       e30* n11* z1^2* z2 -e11* n30* z1^2* z2 - e01* n40*  ...
38       z1^2* z2 + e50* n01* z1^3* z2 + e40*n11*z1^3*z2 - e11* n40* z1^3* ...
39       z2 + e50* n11* z1^4 *z2 + e60* n11* z1^5* z2 + e20* n02*  ...
40       z2^2 - e02* n20* z2^2 + e30* n02* z1* z2^2 + e31* n10*z1*z2^2 +  ...
41       e20* n12* z1* z2^2 - e12* n20* z1* z2^2 - e10*  ...
42       n22 *z1* z2^2 - e02* n30* z1* z2^2 + e40* n02* z1^2* z2^2 + e30* ...
43       n12* z1^2* z2^2 - e12* n30*z1^2* z2^2 - e02*  ...
44       n40* z1^2* z2^2 + e40* n12* z1^3* z2^2 - ...
45       e12* n40* z1^3* z2^2 + e50* n12*z1^4* z2^2 + e23* n00* z2^3 +  ...
46       e20* n03* z2^3 - e03* n20*z2^3 + e30* n03* z1* z2^3 + e23*  ...
47       n10* z1* z2^3 + e20* n13* z1* z2^3 - e13* n20* z1* z2^3 - ...
48       e03* n30* z1*z2^3 + e33* n10* z1^2*z2^3 + e30* n13*  ...
49       z1^2* z2^3 - e13* n30* z1^2* z2^3 + e40*  ...
50       n13 *z1^3* z2^3 - e13* n40* z1^3* z2^3 + e24*  ...
51       n00 *z2^4 + e20* n04* z2^4 - e04* n20* z2^4 + e24*  ...
52       n10* z1* z2^4 - e14* n20* z1* z2^4 - e14* n30*  ...
53       z1^2* z2^4 - e15* n20* z1* z2^5);
54
55 sa01=(e11*n00 - e00* n11 + e21* n00* z1 - e00*n21*z1 + e31* n00* z1^2 - ...
56       e00* n31* z1^2 + e41* n00* z1^3 + e51*  ...
57       n00* z1^4 + e11* n01* z2 - e01* n11* z2 + e21* n01* z1* z2 - ...
58       e01* n21* z1* z2 + e31* n01* z1^2* z2 - e01*  ...
59       n31*z1^2* z2 + e41* n01* z1^3* z2 + e51* n01* z1^4* z2 + e11*  ...
60       n02* z2^2 - e02* n11* z2^2 + ...
61       e21* n02* z1* z2^2 - e02* n21* z1* z2^2 + e31* n02* z1^2* z2^2 -  ...
62       e02* n31* z1^2* z2^2 + e41* n02* z1^3*z2^2+e51* n02* z1^4* z2^2 +...
```

```
      e13* n01* z2^3 + e11* n03* z2^3 - e03* n11* z2^3 - e01* n13*  ...
      z2^3 + e21* n03* z1* z2^3 - e03* n21* z1*  ...
      z2^3 + e31* n03*z1^2* z2^3 - e23* n30*z1^2* z2^3 - e03* n31*  ...
      z1^2* z2^3 + e41* n03* z1^3* z2^3 + e14* n01* z2^4 + e11* n04* ...
      z2^4 - e04*n11* z2^4 + e24* n01* z1* z2^4 + e21* n04* z1 *z2^4 - ...
      e04* n21* z1* z2^4 + e31* n04* z1^2* z2^4 - e04*n31*z1^2* z2^4 + ...
      e15* n01* z2^5 - e05* n11*z2^5-e05* n21* z1* z2^5 - e06* n11* z2^6);

sa20=(e30*n00 - e00* n30 + e40* n00* z1 + e30* n10* z1 - e10* n30* z1 - ...
      e00* n40*z1 + e50*n00* z1^2 + e40* n10* z1^2 + e30* n20* z1^2-e20*...
      n30* z1^2 - e10* n40* z1^2 + e60*n00* z1^3 + e50* n10* z1^3 + ...
      e40* n20* z1^3 - e20* n40* z1^3 + e60*n10* z1^4 + e50*n20* z1^4 + ...
      e60* n20*z1^5 + e30* n01* z2 - e01* n30* z2 + e40* n01* z1* z2 +  ...
      e30* n11* z1* z2 - e11* n30* z1* z2 - e01* n40* z1* z2 + e50*n01* ...
      z1^2* z2 + e40* n11* z1^2* z2 + e30* n21* z1^2* z2 - ...
      e21* n30* z1^2* z2 -e11* n40* z1^2* z2 + e50* n11* z1^3* z2 + ...
      e40* n21* z1^3* z2 - e21*n40* z1^3* z2 + e60* n11* z1^4* z2 + ...
      e50* n21* z1^4* z2 + e60* n21* z1^5* z2 + e30* n02* z2^2 - ...
      e02* n30* z2^2 + e40* n02* z1*z2^2 + e30*n12*z1* z2^2 - e12*n30* ...
      z1* z2^2 - e02* n40* z1*z2^2 + e40*n12*z1^2*z2^2 + e32*n20*z1^2*  ...
      z2^2 + e30* n22* z1^2* z2^2 - e22* n30*  ...
      z1^2* z2^2 - e12* n40* z1^2* z2^2 + e50*  ...
      n12* z1^3* z2^2 + e40* n22* z1^3* z2^2 - e22* n40* z1^3* z2^2 +  ...
      e50* n22* z1^4* z2^2 + e60* n22* z1^5* z2^2 + ...
      e30* n03*z2^3 - e03* n30* z2^3 + ...
      e33*n10* z1* z2^3 + e30* n13* z1*  ...
      z2^3 - e13* n30* z1* z2^3 + e40* n13* z1^2*  ...
      z2^3 + e33* n20* z1^2* z2^3 - e13* n40* z1^2* z2^3 - e23* n40* ...
      z1^3* z2^3 - e14* n30* z1* z2^4 - e24* n30* z1^2* z2^4 - e24*  ...
      n40* z1^3* z2^4);

sa11=(e21* n00 -e00* n21 + e31* n00* z1 + e21* n10* z1 - e10* n21* z1 -...
      e00* n31* z1 + e41* n00* z1^2 + e31* n10* ...
      z1^2 - e10* n31*z1^2 + e51* n00*  ...
      z1^3 + e41* n10* z1^3 + e51*n10* z1^4 + e21* n01* z2 - e01* n21*  ...
      z2 + e31* n01*z1* z2 + e21* n11* z1* z2 - e11* n21* z1* z2 - ...
      e01* n31*z1* z2 + e41* n01* z1^2* z2 + e31*  ...
      n11* z1^2* z2 - e11*n31*z1^2* z2 + e51* n01* z1^3* z2 + e41* n11* ...
      z1^3* z2 + e51* n11*z1^4* z2 + e21*n02* z2^2 - e02* n21*  ...
      z2^2 + e31* n02* z1* z2^2 - e12* n21*z1* z2^2 - e02* n31* z1* ...
      z2^2 + e41* n02* z1^2* z2^2 + e31* n12*  ...
      z1^2* z2^2 - e12* n31* z1^2* z2^2 + e51* n02* z1^3* z2^2 + e41* ...
      n12* z1^3*z2^2 + e51* n12* z1^4* z2^2 + e21* n03* z2^3 - ...
      e03* n21* z2^3 +e31* n03* z1*z2^3 + e23* n11* z1* z2^3 + ...
      e21* n13* z1* z2^3 -e13* n21*z1* z2^3 - e03*n31*z1* z2^3 +  ...
      e41* n03*z1^2* z2^3 + e31* n13*z1^2*  ...
      z2^3 - e13* n31* z1^2* z2^3 + e41* n13* z1^3* z2^3 + e51* n13* ...
      z1^4* z2^3 + e24* n01* z2^4 + e21* n04* z2^4 -  ...
      e04* n21* z2^4 + e31* n04* z1* z2^4 + e24* n11* z1*  ...
      z2^4 - e14* n21* z1* z2^4 - e04* n31* z1*  ...
      z2^4 - e14*n13* z1^2* z2^4-e05* n21* z2^5 - e15* n21* z1* z2^5 * ...
      - e15* n31* z1^2* z2^5);

sa02=(e22* n00*z1 - e00* n22* z1 + e32* n00* z1^2 + e42* n00* z1^3 + ...
```

```
      e12* n01*z2 - e01* n12* z2 + e22* n01* z1* z2 - e01* n22* z1* z2 +...
      e32* n01* z1^2* z2 + e42* n01* z1^3* z2 + ...
      e12* n02* z2^2 - e02* n12* z2^2 + e22* n02* z1* z2^2 - e02* n22*  ...
      z1* z2^2 + e32* n02* z1^2* z2^2 + e42* n02* z1^3* z2^2 + ...
      e12* n03* z2^3 -e03* n12* z2^3 + e22* n03* z1* z2^3 - e03*  ...
      n22* z1* z2^3 +e32* n03* z1^2* z2^3 + e42* n03*z1^3* z2^3 + e14*  ...
      n02* z2^4 + e12* n04* z2^4 - e04* n12*  ...
      z2^4 + e22* n04* z1* z2^4 - e04* n22* z1*  ...
      z2^4 + e32* n04* z1^2* z2^4 + e42* n04* z1^3*  ...
      z2^4 + e15* n02* z2^5 - e05* n12* z2^5 - e05*n22*  ...
      z1* z2^5 - e06* n12* z2^6 - e06* n22* z1* z2^6);

sa30=(e40*n00 - e00* n40 + e50* n00* z1 + e40* n10* z1 - e10* n40* z1 +...
      e60* n00*z1^2 + e50* n10* z1^2 + e40* n20* z1^2 - e20* n40* z1^2 +...
      e60* n10* z1^3 + e50* n20* z1^3 + e40*n30* z1^3 - e30* n40* z1^3 +...
      e60* n20* z1^4 + e50* n30* z1^4 + e60* n30*z1^5 + e40*n01* z2 -...
      e01* n40* z2 + e50* n01* z1* z2 + e40* n11* z1* z2 - ...
      e11*n40*z1*z2+e50*n11*z1^2*z2+e40*n21*z1^2*z2-e21*n40*z1^2*z2+...
      e60*n11*z1^3*z2 + e50*n21*z1^3* 2 + e40*n31*z1^3*z2 -e31*n40*z1^3*...
      z2 + e60* n21* z1^4* z2 + e50* n31* z1^4* z2 + e60* n31* z1^5*  ...
      z2 + e40* n02* z2^2 - e02*n40* z2^2 + e40* n12*z1* z2^2 - e12*  ...
      n40* z1* z2^2 + e50*n12* z1^2* z2^2 + e40*n22* z1^2* z2^2 - e22* ...
      n40*z1^2* z2^2 + e50* n22* z1^3* z2^2 + ...
      e42* n30* z1^3* z2^2 - e32* n40* z1^3* z2^2 + e60* n22* z1^4* ...
      z2^2 + e40* n13* z1* z2^3 - e13* n40* z1* z2^3 - e23* n40* z1^2*  ...
      z2^3 - e33* n40* z1^3* z2^3 - e24* n40* z1^2* z2^4);

sa21=(e31*n00 - e00* n31 + e41* n00* z1 + e31* n10* z1 - e10* n31* z1 + ...
      e51* n00*z1^2 + e41* n10*z1^2 + e31*n20* z1^2 - e20* n31* z1^2 +...
      e51* n10* z1^3 + e41* n20* z1^3 + e51* n20* z1^4 + e31* n01* z2 - ...
      e01*n31*z2 + e41*n01*z1*z2 + e31*n11*z1*z2 - e11*n31*z1*z2 +e51*  ...
      n01* z1^2* z2 + e41* n11*z1^2* z2 + e31* n21* z1^2* z2 - e21*n31* ...
      z1^2* z2 + e51* n11* z1^3* z2 +e41* n21* z1^3* z2 + ...
      e51* n21* z1^4* z2 + e31* n02* z2^2 - e02* n31* z2^2 + ...
      e41* n02* z1* z2^2 +e31* n12* z1* z2^2 - e12* n31* z1* z2^2 + ...
      e51* n02*z1^2* z2^2 +  e41* n12* z1^2* z2^2 + e31*n22*z1^2*z2^2 -...
      e22* n31* z1^2* z2^2 + e51* n12* z1^3*z2^2 + e41* n22* z1^3* ...
      z2^2 + e51* n22* z1^4* z2^2 + e31* n03* z2^3 - e03* n31*  ...
      z2^3 + e41* n03* z1* z2^3 + e31* n13* z1*  ...
      z2^3 - e13* n31* z1* z2^3 + e41* n13*z1^2*z2^3 + e33* n21* z1^2* ...
      z2^3 - e23* n31* z1^2* z2^3 + e51* n13*z1^3*z2^3 + e31*n04*z2^4 -...
      e04* n31* z2^4 - e14* n13* z1* z2^4 - e24* n31* z1^2*  ...
      z2^4 - e15* n31* z1* z2^5);

sa12=(e22* n00 - e00* n22 + e32* n00* z1 + e42* n00* z1^2 + ...
      e32* n10* z1^2 + e42*n10* z1^3 + e22* n01* z2 - e01*n22* z2 +...
      e32*n01* z1* z2 + e22* n11*z1* z2 - e11* n22* z1* z2 + ...
      e42*n01* z1^2* z2 + e32*n11* z1^2* z2 +e42*n11* z1^3* z2 + e22*  ...
      n02* z2^2 - e02* n22* z2^2 + e32*n02* z1* z2^2 + e22* n12* z1* ...
      z2^2 - e12* n22* z1* z2^2 + e42* n02* z1^2*  ...
      z2^2 + e32* n12* z1^2* z2^2 + e42* n12* z1^3* z2^2 + e22* n03* ...
      z2^3 - e03* n22* z2^3 + e32* n03* z1* z2^3 + e22* n13* z1* z2^3 - ...
      e13*n22*z1* z2^3 + e42* n03* z1^2* z2^3 + e32*n13* z1^2* z2^3 +...
      e42*n13* z1^3* z2^3 +e22*n04* z2^4 - e04* n22* z2^4 + ...
```

```
171       e32* n04*z1* z2^4 + e24* n12* z1* z2^4 - e14* n22* z1* z2^4 + ...
172       e42* n04* z1^2* z2^4 - e05* n22* z2^5 - e15* n22* z1* z2^5 - e06* ...
173       n22* z2^6);
174
175  sa03=(e33*n00* z1^2 + e23* n01* z1* z2 + e33* n01* z1^2* z2 + ...
176       e13* n02* z2^2 - e02* n13* z2^2 + e23* n02* z1* z2^2 + e33*  ...
177       n02* z1^2* z2^2 + e13* n03* z2^3 -e03* n13* z2^3 + ...
178       e23* n03* z1* z2^3 + e33* n03* z1^2* z2^3 + e13* ...
179       n04* z2^4 - e04* n13* z2^4 + e23* n04* z1* z2^4 + e33*  ...
180       n04* z1^2* z2^4 + e15* n03* z2^5 -e05* n13* z2^5 - e06* n13* z2^6);
181
182  sa40=(e50* n00 + e60* n00* z1 + e50* n10* z1 + e60* n10* z1^2 + ...
183       e50* n20*z1^2 + e60* n20* z1^3 + e50* n30* z1^3 + e60* n30* z1^4+ ...
184       e50*n40* z1^4 + e60* n40* z1^5 + e50* n01*z2 + e50* n11* z1* z2 + ...
185       e60*n11* z1^2* z2 + e50*n21* z1^2* z2 + e60* n21* z1^3* z2 + ...
186       e50* n31*z1^3* z2 + e60* n31* z1^4* z2 + e50*n12* z1* z2^2 + ...
187       e50* n22* z1^2* z2^2 + e60* n22* z1^3* z2^2);
188
189  sa31=(e41*n00 + e51* n00* z1 + e41* n10* z1 + e51* n10*z1^2 +  ...
190       e41* n20* z1^2 + e51* n20* z1^3 + e41*n30*z1^3 + e51*n30* z1^4 +  ...
191       e41* n01* z2 + e51* n01* z1* z2 + e41*n11* z1* z2 + e51* n11*  ...
192       z1^2* z2 + e41* n21* z1^2* z2 + e51*n21* z1^3* z2 + ...
193       e41* n31*z1^3* z2 + e51* n31* z1^4* z2 + e41*n02* z2^2 + ...
194       e51* n02* z1* z2^2 + e41* n12* z1* z2^2 + e51* n12* z1^2* z2^2 + ...
195       e41* n22* z1^2* z2^2 *+ e51* n22* z1^3* z2^2 + e41*n03* z2^3 +...
196       e41* n13* z1*z2^3 + e51* n13* z1^2* z2^3);
197
198  sa22=(e32* n00 + e42* n00* z1 + e32* n10* z1 + e42*n10* z1^2 + e42*  ...
199       n20* z1^3 + e32*n01* z2 + e42* n01* z1* z2 + e32* n11* z1* z2 + ...
200       e42* n11* z1^2* z2 + e32* n21* z1^2* z2 + e42*n21* z1^3* z2 + ...
201       e32*n02* z2^2 + e42* n02* z1* z2^2 +  ...
202       e32* n12* z1* z2^2 + e42* n12* z1^2*  ...
203       z2^2 + e32* n22* z1^2* z2^2 + e42* n22*z1^3* z2^2 + e32* n03* ...
204       z2^3 + e42* n03* z1* z2^3 + e32* n13* z1* z2^3 + ...
205       e42* n13* z1^2* z2^3 +e32* n04* z2^4 + e42* n04* z1* z2^4);
206
207  sa13=(e33*n00* z1 + e23* n01* z2 + e33* n01*z1* z2 + e33* n11* z1^2* z2+...
208       e23*n02* z2^2 + e33* n02* z1* z2^2 + e23*n12* z1* z2^2 + ...
209       e33* n12* z1^2*z2^2 + e23*n03* z2^3 + e33* n03*  ...
210       z1* z2^3 + e23* n13*z1* z2^3 + e33* n13*  ...
211       z1^2* z2^3 + e23* n04* z2^4 + e33*n04* z1* z2^4);
212
213  sa04=(e24*n02* z1* z2^2 + e14* n03* z2^3 + e24* n03* z1* z2^3 +...
214       e14* n04* z2^4 *+ e24* n04* z1* z2^4);
215
216  sa50=(e60* n00 +e60* n10* z1 + e60* n20* z1^2 + e60* n30* z1^3 + e60*  ...
217       n40* z1^4 + e60* n11* z1* z2 + e60*n21* z1^2* z2 + ...
218       e60* n31* z1^3* z2 + e60* n22* z1^2* z2^2);
219
220  sa41=(e51*n00 + e51* n10* z1 + e51* n20* z1^2 + e51* n30* z1^3 + ...
221       e51* n40* z1^4 + e51* n01*z2 + e51* 11*z1*z2 + e51*n21*z1^2*z2 + ...
222       e51*n31* z1^3* z2 + e51*n02* z2^2 + e51* n12* z1* z2^2 + ...
223       e51* n22* z1^2* z2^2 + e51* n13* z1* z2^3);
224
```

```
sa32=(e42*n00 + e42* n10* z1 + e42* n20* z1^2 + e42* n01* z2 +  ...
      e42* n11* z1* z2 + e42* n21* z1^2* z2 + e42* n31* z1^3* z2 + ...
      e42* n02* z2^2 + e42* n12* z1* z2^2 +  ...
      e42* n22* z1^2* z2^2 + e42*n03* z2^3 + e42* n13* z1* z2^3 + e42* ...
      n04* z2^4);

sa23=(e33*n00 + e33* n01* z2 + e33*n11* z1* z2 + e33* n02*  ...
      z2^2 + e33* n12* z1* z2^2 + e33* n22* z1^2* z2^2 + ...
      e33* n03* z2^3+ e33* n13* z1* z2^3 + e33* n04* z2^4);

sa14=(e24* n02* z2^2 + e24* n03* z2^3 + e24*n13* z1* z2^3 + ...
      e24* n04* z2^4);

sa05=(e15* n04*z2^4);

za00=n00*(n00+n10*z1+n01*z2+n20*z1^2+n11*z1*z2+n02*z2^2+n30*z1^3+...
     n21*z1^2*z2+n12*z1*z2^2+n03*z2^3+...
     n40*z1^4+n31*z1^3*z2+n22*z1^2*z2^2+n13*z1*z2^3+n04*z2^4);
za10=n10*(n00+n10*z1+n01*z2+n20*z1^2+n11*z1*z2+n02*z2^2+n30*z1^3+...
     n21*z1^2*z2+n12*z1*z2^2+n03*z2^3+...
     n40*z1^4+n31*z1^3*z2+n22*z1^2*z2^2+n13*z1*z2^3+n04*z2^4);
za01=n01*(n00+n10*z1+n01*z2+n20*z1^2+n11*z1*z2+n02*z2^2+n30*z1^3+...
     n21*z1^2*z2+n12*z1*z2^2+n03*z2^3+...
     n40*z1^4+n31*z1^3*z2+n22*z1^2*z2^2+n13*z1*z2^3+n04*z2^4);
za20=n20*(n00+n10*z1+n01*z2+n20*z1^2+n11*z1*z2+n02*z2^2+n30*z1^3+...
     n21*z1^2*z2+n12*z1*z2^2+n03*z2^3+...
     n40*z1^4+n31*z1^3*z2+n22*z1^2*z2^2+n13*z1*z2^3+n04*z2^4);
za11=n11*(n00+n10*z1+n01*z2+n20*z1^2+n11*z1*z2+n02*z2^2+n30*z1^3+...
     n21*z1^2*z2+n12*z1*z2^2+n03*z2^3+...
     n40*z1^4+n31*z1^3*z2+n22*z1^2*z2^2+n13*z1*z2^3+n04*z2^4);
za02=n02*(n00+n10*z1+n01*z2+n20*z1^2+n11*z1*z2+n02*z2^2+n30*z1^3+...
     n21*z1^2*z2+n12*z1*z2^2+n03*z2^3+...
     n40*z1^4+n31*z1^3*z2+n22*z1^2*z2^2+n13*z1*z2^3+n04*z2^4);
za30=n30*(n00+n10*z1+n01*z2+n20*z1^2+n11*z1*z2+n02*z2^2+n30*z1^3+...
     n21*z1^2*z2+n12*z1*z2^2+n03*z2^3+...
     n40*z1^4+n31*z1^3*z2+n22*z1^2*z2^2+n13*z1*z2^3+n04*z2^4);
za21=n21*(n00+n10*z1+n01*z2+n20*z1^2+n11*z1*z2+n02*z2^2+n30*z1^3+...
     n21*z1^2*z2+n12*z1*z2^2+n03*z2^3+...
     n40*z1^4+n31*z1^3*z2+n22*z1^2*z2^2+n13*z1*z2^3+n04*z2^4);
za12=n12*(n00+n10*z1+n01*z2+n20*z1^2+n11*z1*z2+n02*z2^2+n30*z1^3+...
     n21*z1^2*z2+n12*z1*z2^2+n03*z2^3+...
     n40*z1^4+n31*z1^3*z2+n22*z1^2*z2^2+n13*z1*z2^3+n04*z2^4);
za03=n03*(n00+n10*z1+n01*z2+n20*z1^2+n11*z1*z2+n02*z2^2+n30*z1^3+...
     n21*z1^2*z2+n12*z1*z2^2+n03*z2^3+...
     n40*z1^4+n31*z1^3*z2+n22*z1^2*z2^2+n13*z1*z2^3+n04*z2^4);
za40=n40*(n00+n10*z1+n01*z2+n20*z1^2+n11*z1*z2+n02*z2^2+n30*z1^3+...
     n21*z1^2*z2+n12*z1*z2^2+n03*z2^3+...
     n40*z1^4+n31*z1^3*z2+n22*z1^2*z2^2+n13*z1*z2^3+n04*z2^4);
za31=n31*(n00+n10*z1+n01*z2+n20*z1^2+n11*z1*z2+n02*z2^2+n30*z1^3+...
     n21*z1^2*z2+n12*z1*z2^2+n03*z2^3+...
     n40*z1^4+n31*z1^3*z2+n22*z1^2*z2^2+n13*z1*z2^3+n04*z2^4);
za22=n22*(n00+n10*z1+n01*z2+n20*z1^2+n11*z1*z2+n02*z2^2+n30*z1^3+...
     n21*z1^2*z2+n12*z1*z2^2+n03*z2^3+...
     n40*z1^4+n31*z1^3*z2+n22*z1^2*z2^2+n13*z1*z2^3+n04*z2^4);
```

```
za13=n13*(n00+n10*z1+n01*z2+n20*z1^2+n11*z1*z2+n02*z2^2+n30*z1^3+...
     n21*z1^2*z2+n12*z1*z2^2+n03*z2^3+...
     n40*z1^4+n31*z1^3*z2+n22*z1^2*z2^2+n13*z1*z2^3+n04*z2^4);
za04=n04*(n00+n10*z1+n01*z2+n20*z1^2+n11*z1*z2+n02*z2^2+n30*z1^3+...
     n21*z1^2*z2+n12*z1*z2^2+n03*z2^3+...
     n40*z1^4+n31*z1^3*z2+n22*z1^2*z2^2+n13*z1*z2^3+n04*z2^4);


sb00=(e01* n00 - e00* n01 + e11*n00* z1 - e10* n01* z1 + e01* n10* z1 -...
      e00* n11* z1 + e21* n00* z1^2- e20* n01* z1^2 + e01* n20* z1^2 - ...
      e00* n21* z1^2 + e31* n00*z1^3 - e30* n01* z1^3 + e01* n30* z1^3- ...
      e00* n31* z1^3 + e41* n00*z1^4 - e40* n01* z1^4 + e01* n40* z1^4+...
      e51* n00* z1^5 - e50* n01*z1^5 +  e02* n00* z2 - ...
      e00* n02* z2 - e10* n02* z1* z2 + e02* n10* z1* z2 + ...
      e22* n00* z1^2* z2 - e20* n02* z1^2* z2 + e02* n20* z1^2* z2 - ...
      e00* n22* z1^2* z2 +e32* n00* z1^3* z2 - e30* n02* z1^3* z2 + ...
      e02* n30* z1^3* z2 + e42*n00* z1^4* z2 - e40* n02* z1^4* z2 + ...
      e02* n40* z1^4* z2 + e03* ...
      n00* z2^2 - e00* n03*z2^2 - e10* n03* z1* z2^2 + ...
      e03* n10* z1* z2^2- e20* n03* z1^2* z2^2 + e03* n20* ...
      z1^2* z2^2 + e33* n00* z1^3* z2^2 - e30* n03* z1^3* z2^2 +...
      e03* n30* z1^3* z2^2 + e04*n00* z2^3 - e00* n04* z2^3 - e10* n04*...
      z1* z2^3 + e04* n10* z1* z2^3 - e20* n04* z1^2* z2^3 +...
      e04* n20* z1^2* z2^3 + e05* n00* z2^4 + ...
      e05* n10*z1* z2^4 + e06* n00* z2^5);

sb10=(e11*n10* z1 - e10* n11* z1 + e21* n10* z1^2 - e20* n11* z1^2 -  ...
      e21* n11* z1^2 + e11* n20* z1^2 - e10* n21* z1^2 + e31* n10*z1^3- ...
      e30* n11* z1^3 + e11* n30* z1^3 - e10* n31* z1^3 + e41* n10*z1^4- ...
      e40* n11* z1^4 + e11* n40* z1^4 + e51* n10* z1^5 - e50* n11*z1^5- ...
      e60* n11* z1^6 + e12* n00* z2 - e00*  ...
      n12* z2 + e12* n10* z1* z2 - e10* n12*z1*z2 - e20*n12*z1^2* z2 +  ...
      e12* n20* z1^2* z2 + e32* n10* z1^3* z2 - e30*n12*z1^3*z2 + e12* ...
      n30* z1^3* z2 + e42* n10*z1^4*z2 - e40* n12*z1^4*z2 + e12* n40*  ...
      z1^4* z2 - e50*n12* z1^5*z2 + e13* n00* z2^2 - e00* n13 *z2^2 +  ...
      e13* n10* z1* z2^2 - e10* n13* z1* z2^2 - e20*n13* z1^2*  ...
      z2^2 + e13* n20* z1^2* z2^2 - e30* n13*  ...
      z1^3* z2^2 + e13* n30* z1^3* z2^2 - e40* n13* z1^4* z2^2 + e13* ...
      n40* z1^4* z2^2 + e14* n00* z2^3 + e41* n10* z1*  ...
      z2^3 + e14* n20* z1^2* z2^3 + e14* n30*  ...
      z1^3* z2^3 + e15* n00* z2^4 + e15*n10*z1*z2^4 + e15* n20* z1^2* ...
      z2^4);

sb01=(e02* n00 -e00* n02 - e10* n02* z1 +  ...
      e02* n10* z1 + e22* n00* z1^2 - e20* n02* z1^2 +  ...
      e02* n20* z1^2 - e00* n22* z1^2 + e32* n00* z1^3 - e30* n02*  ...
      z1^3 + e02* n30* z1^3 + e42* n00* z1^4 - e40* n02*z1^4 +e02*n40*  ...
      z1^4 + e03* n00* z2 + e02* n01* z2 -  ...
      e01* n02* z2 - e00* n03* z2 + e12* n01*z1*z2 - e11*n02* z1* z2 -  ...
      e10* n03* z1* z2 + e03* n10* z1* z2 + ...
       e02* n11* z1* z2 - e01* n12* z1* z2 +  ...
      e22* n01* z1^2* z2 - e21* n02* z1^2* z2 - e20*n03*z1^2* z2 + e03* ...
      n20* z1^2* z2 + e02* n21* z1^2* z2 - e01* n22* z1^2* z2 +  ...
      e33* n00* z1^3* z2 + e32*n01* z1^3* z2 - e31*n02*z1^3*  ...
```

```
333       z2 - e30*n03* z1^3* z2 + e03* n30* z1^3* z2 + e02*n31* z1^3*z2 +  ...
334       e42* n01* z1^4* z2 - e41* n02* z1^4* z2 - ...
335       e51* n02* z1^5* z2 + e04* n00* z2^2 + e03* n01* z2^2 - ...
336       e01* n03* z2^2  - e00* n04* z2^2 - e11* n03* z1* z2^2 - ...
337       e10*n04* z1* z2^2 + e04* n10* z1* z2^2 + e03* n11* z1* z2^2 + ...
338       e23* n01* z1^2* z2^2 - e21* n03* z1^2*z2^2 - e20* n04*  ...
339       z1^2* z2^2 + e04* n20* z1^2* z2^2 + e03* n21* z1^2* z2^2 + e33* ...
340       n01* z1^3* z2^2 - e31* n03*z1^3* z2^2 + e23*n30*z1^3* z2^2 + e03* ...
341       n31* z1^3* z2^2 - e41* n03*z1^4*z2^2 + e05* n00* z2^3 + e04* n01* ...
342       z2^3 - e01* n04* z2^3 - e11*n04*z1*z2^3 + e05*n10*z1* z2^3 +e04* ...
343       n11* z1* z2^3 - e21* n04* z1^2* z2^3 + e04*n21* z1^2* z2^3 - ...
344       e31* n04* z1^3* z2^3 + e04* n31* z1^3* z2^3 + e06* n00* z2^4 + ...
345       e05*n01*z2^4 + e05* n11* z1* z2^4+ e05* n21* z1^2* z2^4 + ...
346       e06* n01* z2^5 + e06* n11* z1* z2^5);
347
348 sb20=(e21*n20*z1^2 - e20*n21*z1^2 + e31*n20*z1^3 - e30*n21* z1^3 + e21* ...
349       n30* z1^3 - e20* n31* z1^3 + e41* n20* z1^4 - e40* n21* z1^4 + ...
350       e21* n40*z1^4 + e51* n20* z1^5 - e50* n21* z1^5 - e60* n21*z1^6 + ...
351       e31* n10* z1* z2 - e10* n22* z1* z2 + e22*  ...
352       n20* z1^2* z2 - e20* n22* z1^2* z2 - e30*n22* z1^3*z2 + e22* n30* ...
353       z1^3* z2 + e42* n20* z1^4* z2 - e40*n22*z1^4*z2 + e22*n40*z1^4*  ...
354       z2 - e50* n22* z1^5* z2 - e60* n22*  ...
355       z1^6* z2 + e23* n00* z2^2 + e23* n10* z1*z2^2 + e23* n20* z1^2* ...
356       z2^2 + e23* n40* z1^4* z2^2 + e24* n00* z2^3 + ...
357       e24* n10* z1* z2^3 + e24* n20* z1^2*  ...
358       z2^3 + e24* n30* z1^3* z2^3 + e24* n40* z1^4* z2^3);
359
360 sb11=(e12*n00 - e00* n12 + e12* n10* z1 - e10* n12* z1 - ...
361       e20* n12*z1^2 + e12* n20* z1^2 + e32* n10* z1^3 - e30*  ...
362       n12* z1^3 + e12* n30*z1^3 + e42*n10*z1^4 - e40* n12* z1^4 + e12*  ...
363       n40* z1^4 - e50* n12* z1^5 + e13*  ...
364       n00* z2 - e00* n13* z2 + e13* n10* z1* z2 + e12*n11*z1*z2 - e11*  ...
365       n12* z1* z2 - e10* n13*z1*z2 + e22*n11*z1^2*z2 - e20*n13*z1^2*  ...
366       z2 + e13* n20* z1^2* z2 + e12* n21* z1^2*z2 - e11*n22*z1^2*z2 +  ...
367       e32* n11* z1^3* z2 - e31* n12* z1^3*z2 - e30* n13* z1^3* z2 + ...
368       e13* n30* z1^3* z2 + e12* n31* z1^3*z2 + e42* n11* z1^4*z2 -e41*  ...
369       n12* z1^4* z2 - e40* n13* z1^4* z2 + ...
370       e13* n40* z1^4* z2 - e51* n12* z1^5* z2 + e14*n00*z2^2 + e13*n01* ...
371       z2^2 - e01* n13* z2^2 + e41* n10* z1* z2^2 + e13* n11*  ...
372       z1* z2^2 - e11* n13* z1* z2^2 - e21* n13*  ...
373       z1^2* z2^2 + e14* n20* z1^2* z2^2 + ...
374       e13* n21* z1^2* z2^2 + e33* n11* z1^3*z2^2 - e31* n13* z1^3* ...
375       z2^2 + e14* n30* z1^3* z2^2 + e13*n31* z1^3* z2^2 - e41* n13*  ...
376       z1^4* z2^2 - e51* n13* z1^5* z2^2 + e15*  ...
377       n00* z2^3 + e14* n01* z2^3 + e15* n10* z1*  ...
378       z2^3 + e14* n11* z1* z2^3 + e15* n20*  ...
379       z1^2* z2^3 + e14* n21* z1^2* z2^3 + e14*  ...
380       n13* z1^3* z2^3 + e15*n01* z2^4 + e15* n11* z1* z2^4 + e15* n21* ...
381       z1^2* z2^4 + e15* n31* z1^3* z2^4);
382
383 sb02=(e03* n00 -e00* n03 - e10* n03* z1 +e03*n10* z1 - e20* n03* z1^2 + ...
384       e03* n20* z1^2 +  e33* n00* z1^3 - e30* n03* z1^3 + ...
385       e03* n30* z1^3 + e04*n00*z2 + e03* n01* z2 - e01* n03* z2 - e00*  ...
386       n04*z2 - e11* n03* z1* z2 - e10* n04*z1*z2 + e04*n10*z1*z2 +  ...
```

```
387       e03* n11* z1* z2 + e23* n01* z1^2* z2 - ...
388       e21* n03* z1^2* z2 - e20* n04* z1^2* z2 + e04*n20* z1^2*z2 + e03* ...
389       n21* z1^2* z2 + e33*n01* z1^3* z2 - e31* n03*  ...
390       z1^3* z2 + e23* n30* z1^3* z2 +e03* n31* z1^3* z2 -...
391       e41* n03*z1^4* z2 + e05* n00* z2^2 + e04* n01*z2^2 + ...
392       e03* n02* z2^2 - e02* n03* z2^2 -  e01* n04* z2^2 + ...
393       e13* n02* z1* z2^2 - e12* n03* z1* z2^2 - e11* n04* z1* z2^2 + ...
394       e05* n10* z1* z2^2 + e04* n11* z1* z2^2 + e03*  ...
395       n12* z1* z2^2 - e02* n13* z1* z2^2 + e23*  ... ...
396       n02*z1^2* z2^2 - e22* n03* z1^2* z2^2 - ...
397       e21* n04* z1^2* z2^2 + e04* n21* z1^2* z2^2 + e03* n22* z1^2* ...
398       z2^2 + e33* n02*z1^3*z2^2 - e32*n03*z1^3*z2^2 - e31*n04*z1^3*z2^2 ...
399       + e04* n31* z1^3* z2^2 - e42* n03* z1^4* z2^2 + e06* n00* z2^3 +  ...
400       e05* n01* z2^3 + e04* n02* z2^3 -  ...
401       e02* n04* z2^3 - e12*n04*z1* z2^3 + e05*n11* z1* z2^3 + e04* n12* ...
402       z1* z2^3 + e24* n02* z1^2* z2^3 - e22*  ...
403       n04* z1^2* z2^3 + e05* n21* z1^2* z2^3 + e04* n22* z1^2* z2^3 -  ...
404       e32* n04* z1^3* z2^3 - e42* n04* z1^4* z2^3 + e06* n01* z2^4 + ...
405       e05* n02* z2^4 + e06* n11* z1* z2^4 + e05* n12* z1*  ...
406       z2^4 + e05* n22* z1^2* z2^4 + e06* n02* z2^5 + e06* n12* z1* ...
407       z2^5 + e06* n22* z1^2* z2^5);
408
409  sb30=(e31*n30* z1^3 - e30* n31* z1^3 + e41*n30* z1^4 - e40* n31* z1^4 + ...
410       e31*n40* z1^4 + e51* n30* z1^5 - e50* n31*z1^5 - e60* n31* z1^6 + ...
411       e32*n20* z1^2* z2 + e32* n30* z1^3* z2 + e32*n40* z1^4* z2 + ...
412       e33* n10* z1*z2^2 + e33* n20* z1^2* z2^2 + e33* n30* z1^3* z2^2 +...
413       e33* n40*z1^4* z2^2);
414
415  sb21=(e31* n10* z1 - e10* n22* z1 + e22* n20* z1^2 - e20* n22* z1^2 -  ...
416       e30* n22* z1^3 + e22* n30*z1^3 + e42*n20*z1^4 - e40*n22* z1^4 +  ...
417       e22* n40* z1^4 - e50* n22*z1^5 - e60* n22* z1^6 + e23* n00* z2 +  ...
418       e23* n10*z1* z2 + e23* n20* z1^2* z2 + ...
419       e22*n21* z1^2* z2 - e21* n22*z1^2* z2 + e32*n21*z1^3* z2 - e31*  ...
420       n22* z1^3* z2 + e22* n31* z1^3* z2 + e42*n21*z1^4*z2 - e41* n22* ...
421       z1^4* z2 + e23* n40* z1^4*z2 - e51*n22* z1^5* z2 + e24* n00*  ...
422       z2^2 + e24* n10* z1* z2^2 + e23* n11* z1* z2^2 + ...
423       e24* n20* z1^2* z2^2 + e23*n21*z1^2* z2^2 + e24* n30* z1^3* z2^2 +...
424       e23*n31*z1^3*z2^2 + e24*n40*z1^4*z2^2 + e24*n01* z2^3 + e24* n11* ...
425       z1* z2^3 + e24* n21* z1^2* z2^3 + e24* n31* z1^3* z2^3);
426
427  sb12=(e13*n00 - e00* n13 + e13* n10* z1 - e10* n13* z1 - ...
428       e20* n13* z1^2 + e13* n20* z1^2 - e30* n13* z1^3 + e13*  ...
429       n30* z1^3 - e40*n13* z1^4 + e13* n40* z1^4 + e14* n00 *z2 + e13*  ...
430       n01* z2 - e01* n13* z2 + e41* n10* z1*  ...
431       z2 + e13* n11* z1* z2 - e11* n13*z1*z2 - e21*n13* z1^2* z2 + e14* ...
432       n20*z1^2* z2 + e13* n21* z1^2* z2 + ...
433       e33* n11* z1^3* z2 - e31* n13* z1^3* z2 + e14*n30*z1^3* z2 + e13* ...
434       n31 *z1^3* z2 - e41* n13* z1^4* z2 - e51* n13* z1^5*  ...
435       z2 + e15* n00* z2^2 + e14* n01* z2^2 + e15*n10*z1*z2^2 +e14* n11* ...
436       z1* z2^2 + e13* n12* z1*z2^2 - e12*n13* z1*  ...
437       z2^2 + e23* n12* z1^2* z2^2 - e22*n13*  ...
438       z1^2* z2^2 + e15* n20* z1^2* z2^2 + e14*  ...
439       n21* z1^2* z2^2 + e13* n22* z1^2* z2^2 + e33*n12* z1^3* z2^2 +  ...
440       e14* n13* z1^3*  ...
```

```
441       z2^2 - e32* n13* z1^3* z2^2 - e42* n13* z1^4* z2^2 + e15* n01* ...
442       z2^3 + e14* n02*z2^3 + e15*n11*z1*z2^3 +e14* n12* z1* z2^3 + e15* ...
443       n21* z1^2* z2^3 + e14* n22* z1^2* z2^3 + e15* n31* z1^3* z2^3 + ...
444       e15* n02* z2^4 + e15* n12* z1* z2^4 + e15* n22* z1^2* z2^4);
445
446  sb03=(e04*n00 - e00* n04 - e10* n04* z1 + e04* n10* z1 - ...
447       e20* n04* z1^2 + e04* n20* z1^2 + e23* n30* z1^3 + e05*  ...
448       n00* z2 + e04*n01* z2 - e01* n04*z2 - e11 *n04* z1*z2 + e05*n10*  ...
449       z1*z2 + e04* n11* z1*z2 - e21* n04* z1^2*z2 + e04*n21* z1^2*z2 -  ...
450       e31* n04* z1^3* z2 + e04* n31* z1^3* z2 + e06* n00*  ...
451       z2^2 + e05* n01* z2^2 + e04* n02* z2^2 - e02*n04*z2^2 - e12*n04*  ...
452       z1* z2^2 + e05* n11* z1* z2^2 + e04* n12* z1*z2^2 + e24*n02*z1^2* ...
453       z2^2 - e22*n04 *z1^2* z2^2 + e05* n21* z1^2* z2^2 +...
454       e04* n22* z1^2* z2^2 - e32* n04* z1^3*  ...
455       z2^2 - e42*n04* z1^4* z2^2 + e06* n01* z2^3 + e05* n02* z2^3 +  ...
456       e04*n03*z2^3 - e03*n04*z2^3 + e14*n03*z1* z2^3 - e13*n04*z1*z2^3  ...
457       + e06* n11*z1* z2^3 + e05* n12* z1* z2^3 + ...
458       e04* n13* z1* z2^3 + e24* n03* z1^2*  ...
459       z2^3 - e23* n04* z1^2* z2^3 + e05* n22* z1^2* z2^3 - e33* n04* ...
460       z1^3* z2^3 + e06* n02* z2^4 + e05*n03*z2^4 + e06* n12*z1*z2^4 +  ...
461       e05* n13* z1* z2^4 + e06* n22* z1^2* z2^4 + e06*  ...
462       n03 *z2^5 + e06* n13* z1* z2^5);
463
464  sb40=(e41*n40* z1^4 + e51* n40* z1^5 + e42* n30* z1^3* z2 + ...
465       e42* n40* z1^4* z2);
466
467  sb31=(e32*n20* z1^2 + e32* n30* z1^3 + e32* n40* z1^4 + ...
468       e33* n10* z1* z2 + e33*n20*z1^2* z2 + e33* n30* z1^3* z2 + ...
469       e32*n31* z1^3* z2 + e42* n31*z1^4* z2 + e33*n40* z1^4* z2 + ...
470       e33*n21* z1^2* z2^2 + e33* n31* z1^3* z2^2);
471
472  sb22=(e23* n00 +e23* n10* z1 + e23* n20*z1^2 + e23* n40* z1^4 + e24*  ...
473       n00* z2 + e24*n10* z1* z2 + e23* n11*z1*z2 + e24*n20*z1^2* z2 + ...
474       e23* n21* z1^2* z2 + e24*n30*z1^3*z2 + e23*n31* z1^3* z2 + e24*  ...
475       n40* z1^4* z2 + e24* n01* z2^2 + e24* n11* z1* z2^2 + ...
476       e24* n21* z1^2* z2^2 + e23* n22* z1^2* z2^2 + ...
477       e33* n22* z1^3* z2^2 + e24* n31* z1^3* ...
478       z2^2 + e24* n12* z1* z2^3 + e24* n22* z1^2* z2^3);
479
480  sb13=(e14*n00 + e41* n10* z1 + e14* n20* z1^2 + e14* n30* z1^3 + ...
481       e15* n00* z2 + e14* n01* z2 + e15*  ...
482       n10* z1* z2 + e14* n11* z1* z2 + e15* n20* z1^2* z2 + e14* n21*  ...
483       z1^2* z2 + e14* n13* z1^3* z2 + e15* n01* z2^2 + e14*n02*z2^2 +  ...
484       e15* n11* z1* z2^2 + e14* n12* z1*  ...
485       z2^2 + e15* n21* z1^2* z2^2 + e14* n22* z1^2* z2^2 + e15* n31* ...
486       z1^3* z2^2 + e15* n02* z2^3 + e15* n12* z1* z2^3 + e14*n13*  ...
487       z1* z2^3 + e24* n13* z1^2* z2^3 + e15* n22* z1^2* z2^3 + e15*  ...
488       n03*z2^4 + e15* n13* z1* z2^4);
489
490  sb04=(e05*n00 + e05* n10* z1 + e06* n00* z2 + e05* n01* z2 + ...
491       e05* n11*z1* z2 + e05* n21* z1^2* z2 + e06* n01* z2^2 + e05*  ...
492       n02* z2^2 + e06* n11* z1*z2^2 + e05*n12*z1* z2^2 + e05*n22*z1^2* ...
493       z2^2 + e06* n02* z2^3 + e05* n03* z2^3 +e06* n12*z1* z2^3 + ...
494       e05* n13* z1* z2^3 + e06* n22* z1^2*z2^3 + e06* n03*  ...
```

```
       z2^4 + e05* n04* z2^4 + e15* n04*z1*  ...
       z2^4 + e06* n13* z1* z2^4 + e06* n04* z2^5);

sb50=0;
sb41=(e42* n30* z1^3 + e42* n40* z1^4);

sb32=(e33*n10* z1 + e33* n20* z1^2 + e33* n30* z1^3 + ...
       e33* n40* z1^4 + e33*n21* z1^2* z2 + e33* n31* z1^3* z2);

sb23=(e24*n00 + e24* n10* z1 + e24* n20* z1^2 + e24*n30* z1^3 + ...
       e24* n40* z1^4 + e24* n01* z2 + e24* n11* z1* z2 + ...
       e24* n21*z1^2* z2 + e24* n31* z1^3* z2 + e24*  ...
       n12* z1* z2^2 + e24* n22* z1^2* z2^2);

sb14=(e15* n00 + e15* n10*z1 + e15* n20* z1^2 + e15* n01* z2 +  ...
       e15* n11* z1* z2 + e15* n21* z1^2* z2 + e15* n31*z1^3* z2 + ...
       e15* n02* z2^2 + e15* n12* z1* z2^2 +  ...
       e15* n22* z1^2* z2^2 + e15* n03* z2^3 + e15* n13* z1* z2^3);

sb05=(e06*n00 + e06* n01* z2 + e06*n11* z1* z2 + e06* n02*z2^2 + ...
       e06* n12* z1* z2^2 + e06* n22* z1^2* z2^2 + e06* n03* z2^3 +  ...
       e06* n13* z1*z2^3 + e06* n04* z2^4);

zb00=n00*(n00+n10*z1+n01*z2+n20*z1^2+n11*z1*z2+n02*z2^2+n30*z1^3+...
      n21*z1^2*z2+n12*z1*z2^2+n03*z2^3+...
      n40*z1^4+n31*z1^3*z2+n22*z1^2*z2^2+n13*z1*z2^3+n04*z2^4);
zb10=n10*(n00+n10*z1+n01*z2+n20*z1^2+n11*z1*z2+n02*z2^2+n30*z1^3+...
      n21*z1^2*z2+n12*z1*z2^2+n03*z2^3+...
      n40*z1^4+n31*z1^3*z2+n22*z1^2*z2^2+n13*z1*z2^3+n04*z2^4);
zb01=n01*(n00+n10*z1+n01*z2+n20*z1^2+n11*z1*z2+n02*z2^2+n30*z1^3+...
      n21*z1^2*z2+n12*z1*z2^2+n03*z2^3+...
      n40*z1^4+n31*z1^3*z2+n22*z1^2*z2^2+n13*z1*z2^3+n04*z2^4);
zb20=n20*(n00+n10*z1+n01*z2+n20*z1^2+n11*z1*z2+n02*z2^2+n30*z1^3+...
      n21*z1^2*z2+n12*z1*z2^2+n03*z2^3+...
      n40*z1^4+n31*z1^3*z2+n22*z1^2*z2^2+n13*z1*z2^3+n04*z2^4);
zb11=n11*(n00+n10*z1+n01*z2+n20*z1^2+n11*z1*z2+n02*z2^2+n30*z1^3+...
      n21*z1^2*z2+n12*z1*z2^2+n03*z2^3+...
      n40*z1^4+n31*z1^3*z2+n22*z1^2*z2^2+n13*z1*z2^3+n04*z2^4);
zb02=n02*(n00+n10*z1+n01*z2+n20*z1^2+n11*z1*z2+n02*z2^2+n30*z1^3+...
      n21*z1^2*z2+n12*z1*z2^2+n03*z2^3+...
      n40*z1^4+n31*z1^3*z2+n22*z1^2*z2^2+n13*z1*z2^3+n04*z2^4);
zb30=n30*(n00+n10*z1+n01*z2+n20*z1^2+n11*z1*z2+n02*z2^2+n30*z1^3+...
      n21*z1^2*z2+n12*z1*z2^2+n03*z2^3+...
      n40*z1^4+n31*z1^3*z2+n22*z1^2*z2^2+n13*z1*z2^3+n04*z2^4);
zb21=n21*(n00+n10*z1+n01*z2+n20*z1^2+n11*z1*z2+n02*z2^2+n30*z1^3+...
      n21*z1^2*z2+n12*z1*z2^2+n03*z2^3+...
      n40*z1^4+n31*z1^3*z2+n22*z1^2*z2^2+n13*z1*z2^3+n04*z2^4);
zb12=n12*(n00+n10*z1+n01*z2+n20*z1^2+n11*z1*z2+n02*z2^2+n30*z1^3+...
      n21*z1^2*z2+n12*z1*z2^2+n03*z2^3+...
      n40*z1^4+n31*z1^3*z2+n22*z1^2*z2^2+n13*z1*z2^3+n04*z2^4);
zb03=n03*(n00+n10*z1+n01*z2+n20*z1^2+n11*z1*z2+n02*z2^2+n30*z1^3+...
      n21*z1^2*z2+n12*z1*z2^2+n03*z2^3+...
      n40*z1^4+n31*z1^3*z2+n22*z1^2*z2^2+n13*z1*z2^3+n04*z2^4);
zb40=n40*(n00+n10*z1+n01*z2+n20*z1^2+n11*z1*z2+n02*z2^2+n30*z1^3+...
```

```
      n21*z1^2*z2+n12*z1*z2^2+n03*z2^3+...
      n40*z1^4+n31*z1^3*z2+n22*z1^2*z2^2+n13*z1*z2^3+n04*z2^4);
zb31=n31*(n00+n10*z1+n01*z2+n20*z1^2+n11*z1*z2+n02*z2^2+n30*z1^3+...
      n21*z1^2*z2+n12*z1*z2^2+n03*z2^3+...
      n40*z1^4+n31*z1^3*z2+n22*z1^2*z2^2+n13*z1*z2^3+n04*z2^4);
zb22=n22*(n00+n10*z1+n01*z2+n20*z1^2+n11*z1*z2+n02*z2^2+n30*z1^3+...
      n21*z1^2*z2+n12*z1*z2^2+n03*z2^3+...
      n40*z1^4+n31*z1^3*z2+n22*z1^2*z2^2+n13*z1*z2^3+n04*z2^4);
zb13=n13*(n00+n10*z1+n01*z2+n20*z1^2+n11*z1*z2+n02*z2^2+n30*z1^3+...
      n21*z1^2*z2+n12*z1*z2^2+n03*z2^3+...
      n40*z1^4+n31*z1^3*z2+n22*z1^2*z2^2+n13*z1*z2^3+n04*z2^4);
zb04=n04*(n00+n10*z1+n01*z2+n20*z1^2+n11*z1*z2+n02*z2^2+n30*z1^3+...
      n21*z1^2*z2+n12*z1*z2^2+n03*z2^3+...
      n40*z1^4+n31*z1^3*z2+n22*z1^2*z2^2+n13*z1*z2^3+n04*z2^4);
```

A.21 dreiDmain.m

```
function [a000,a100,a010,a001,a110,a011,a101,a200,a020,a002,...
          b100,b010,b001,b110,b011,b101,b200,b020,b002,u,o]=dreiDmain()


AnzOp=1;         %Anzahl der Operationen
ArtOp=[1];       %Welche Operationen werden gemacht
                 %(1..Add., 2..Subtr., 3.. Multipl. , 4.. Div.)

%Definition der Brüche von Polynomen inklusive Fehlerintervall
p000(1)=3;
p100(1)=0;
p010(1)=0;
p001(1)=0;
p110(1)=0;
p101(1)=0;
p011(1)=0;
p200(1)=0;
p020(1)=0;
p002(1)=0;
q100(1)=0;
q010(1)=0;
q001(1)=0;
q110(1)=0;
q101(1)=0;
q011(1)=0;
q200(1)=0;
q020(1)=0;
q002(1)=-4;
u(1)=-0.0;
o(1)=0.0;

p000(2)=0;
p100(2)=4;
p010(2)=0.5;
```

```
35 p001(2)=0;
36 p110(2)=8;
37 p101(2)=0;
38 p011(2)=0;
39 p200(2)=-2;
40 p020(2)=0;
41 p002(2)=0;
42 q100(2)=0;
43 q010(2)=0;
44 q001(2)=0;
45 q110(2)=0;
46 q101(2)=0;
47 q011(2)=0;
48 q200(2)=0;
49 q020(2)=0;
50 q002(2)=0;
51 u(2)=0;
52 o(2)=0;
53
54
55
56 mult=10^-2;
57 Jx=1*mult;
58 Jy=1*mult;              %Definitionsbereich ist also die
59 Jz=1*mult;              %[-Jx,Jx]x[-Jy,Jy]x[-Jz,Jz].
60 Ix=max(Jx,0.1);
61 Iy=max(Jy,0.1);         %Für [-Ix,Ix]x[-Iy,Iy]x[-Iz,Iz]
62 Iz=max(Jz,0.1);         %werden mittels CVX Koeffizienten berechnet
63 a=1;
64 b=0.5;
65 c=1;
66
67 %Punkte an denen ausgewertet wird:
68 x(1)=0*Ix;
69 y(1)=0*Iy;
70 z(1)=0*Iz;
71 x(2)=a*Ix;
72 y(2)=0*Iy;
73 z(2)=0*Iz;
74 x(3)=-a*Ix;
75 y(3)=0*Iy;
76 z(3)=0*Iz;
77 x(4)=0*Ix;
78 y(4)=a*Iy;
79 z(4)=0*Iz;
80 x(5)=0*Ix;
81 y(5)=-a*Iy;
82 z(5)=0*Iz;
83 x(6)=0*Ix;
84 y(6)=0*Iy;
85 z(6)=a*Iz;
86 x(7)=0*Ix;
87 y(7)=0*Iy;
88 z(7)=-a*Iz;
```

```
x(8)=b*Ix;
y(8)=0*Iy;
z(8)=0*Iz;
x(9)=-b*Ix;
y(9)=0*Iy;
z(9)=0*Iz;
x(10)=0*Ix;
y(10)=b*Iy;
z(10)=0*Iz;
x(11)=0*Ix;
y(11)=-b*Iy;
z(11)=0*Iz;
x(12)=0*Ix;
y(12)=0*Iy;
z(12)=b*Iz;
x(13)=0*Ix;
y(13)=0*Iy;
z(13)=-b*Iz;
x(14)=c*Ix;
y(14)=c*Iy;
z(14)=0*Iz;
x(15)=c*Ix;
y(15)=-c*Iy;
z(15)=0*Iz;
x(16)=-c*Ix;
y(16)=c*Iy;
z(16)=0*Iz;
x(17)=-c*Ix;
y(17)=-c*Iy;
z(17)=0*Iz;
x(18)=0*Ix;
y(18)=c*Iy;
z(18)=c*Iz;
x(19)=0*Ix;
y(19)=-c*Iy;
z(19)=c*Iz;
x(20)=0*Ix;
y(20)=c*Iy;
z(20)=-c*Iz;
x(21)=0*Ix;
y(21)=-c*Iy;
z(21)=-c*Iz;
x(22)=c*Ix;
y(22)=0*Iy;
z(22)=c*Iz;
x(23)=-c*Ix;
y(23)=0*Iy;
z(23)=c*Iz;
x(24)=c*Ix;
y(24)=0*Iy;
z(24)=-c*Iz;
x(25)=-c*Ix;
y(25)=0*Iy;
z(25)=-c*Iz;
```

```
143
144
145 %Berechnung der Funktionswerte an den Punkten
146 for i=1:25
147     f(i)=(p000(1)+p100(1)*x(i)+p010(1)*y(i)+...
148           p001(1)*z(i)+p110(1)*x(i)*y(i)+...
149           p101(1)*x(i)*z(i)+p011(1)*y(i)*z(i)+...
150           p200(1)*x(i)^2+p020(1)*y(i)^2+...
151           p002(1)*z(i)^2)/...
152          (1+q100(1)*x(i)+q010(1)*y(i)+q001(1)*z(i)+...
153           q110(1)*x(i)*y(i)+...
154           p101(1)*x(i)*z(i)+q011(1)*y(i)*z(i)+...
155           q200(1)*x(i)^2+q020(1)*y(i)^2+...
156           q002(1)*z(i)^2);
157 end;
158
159
160
161 for j=2:AnzOp+1
162
163
164     for i=1:25
165         if(ArtOp(j-1)==1)
166             f(i)=f(i)+((p000(j)+p100(j)*x(i)+p010(j)*y(i)+p001(j)*z(i)+...
167                  p110(j)*x(i)*y(i)+p101(j)*x(i)*z(i)+p011(j)*y(i)*z(i)+...
168                  p200(j)*x(i)^2+p020(j)*y(i)^2+p002(j)*z(i)^2)/...
169                  (1+q100(j)*x(i)+q010(j)*y(i)+q001(j)*z(i)+...
170                  q110(j)*x(i)*y(i)+...
171                  p101(j)*x(i)*z(i)+q011(j)*y(i)*z(i)+q200(j)*x(i)^2+...
172                  q020(j)*y(i)^2+q002(j)*z(i)^2));
173         elseif(ArtOp(j-1)==2)
174             f(i)=f(i)-((p000(j)+p100(j)*x(i)+p010(j)*y(i)+p001(j)*z(i)+...
175                  p110(j)*x(i)*y(i)+p101(j)*x(i)*z(i)+p011(j)*y(i)*z(i)+...
176                  p200(j)*x(i)^2+p020(j)*y(i)^2+p002(j)*z(i)^2)/...
177                  (1+q100(j)*x(i)+q010(j)*y(i)+q001(j)*z(i)+...
178                  q110(j)*x(i)*y(i)+...
179                  p101(j)*x(i)*z(i)+q011(j)*y(i)*z(i)+q200(j)*x(i)^2+...
180                  q020(j)*y(i)^2+q002(j)*z(i)^2));
181         elseif(ArtOp(j-1)==3)
182             f(i)=f(i)*((p000(j)+p100(j)*x(i)+p010(j)*y(i)+p001(j)*z(i)+...
183                  p110(j)*x(i)*y(i)+p101(j)*x(i)*z(i)+p011(j)*y(i)*z(i)+...
184                  p200(j)*x(i)^2+p020(j)*y(i)^2+p002(j)*z(i)^2)/...
185                  (1+q100(j)*x(i)+q010(j)*y(i)+q001(j)*z(i)+...
186                  q110(j)*x(i)*y(i)+...
187                  p101(j)*x(i)*z(i)+q011(j)*y(i)*z(i)+q200(j)*x(i)^2+...
188                  q020(j)*y(i)^2+q002(j)*z(i)^2));
189         elseif(ArtOp(j-1)==4)
190             f(i)=f(i)/((p000(j)+p100(j)*x(i)+p010(j)*y(i)+p001(j)*z(i)+...
191                  p110(j)*x(i)*y(i)+p101(j)*x(i)*z(i)+p011(j)*y(i)*z(i)+...
192                  p200(j)*x(i)^2+p020(j)*y(i)^2+p002(j)*z(i)^2)/...
193                  (1+q100(j)*x(i)+q010(j)*y(i)+q001(j)*z(i)+...
194                  q110(j)*x(i)*y(i)+...
195                  p101(j)*x(i)*z(i)+q011(j)*y(i)*z(i)+q200(j)*x(i)^2+...
196                  q020(j)*y(i)^2+q002(j)*z(i)^2));
```

```
197         end;
198     end;
199
200
201
202     %Berechnung der Koeffizienten mittels CVX
203     [p000(2+AnzOp),p100(2+AnzOp),p010(2+AnzOp),p001(2+AnzOp),p110(2+AnzOp),...
204      p101(2+AnzOp),p011(2+AnzOp),p200(2+AnzOp),p020(2+AnzOp),p002(2+AnzOp),...
205      q100(2+AnzOp),q010(2+AnzOp),q001(2+AnzOp),q110(2+AnzOp),q101(2+AnzOp),...
206      q011(2+AnzOp),q200(2+AnzOp),q020(2+AnzOp),q002(2+AnzOp)] = ...
207       dreiDKoeffCVX...
208         (x(1),x(2),x(3),x(4),x(5),x(6),x(7),x(8),x(9),x(10),x(11),x(12),...
209          x(13),x(14),x(15),x(16),x(17),x(18),x(19),x(20),x(21),x(22),x(23),...
210          x(24),x(25),y(1),y(2),y(3),y(4),y(5),y(6),y(7),y(8),y(9),y(10),...
211          y(11),y(12),y(13),y(14),y(15),y(16),y(17),y(18),y(19),y(20),y(21),...
212          y(22),y(23),y(24),y(25),z(1),z(2),z(3),z(4),z(5),z(6),z(7),z(8),...
213          z(9),z(10),z(11),z(12),z(13),z(14),z(15),z(16),z(17),z(18),z(19),...
214          z(20),z(21),z(22),z(23),z(24),z(25),-Jx,Jx,-Jy,Jy,-Jz,Jz);
215
216     %Berechnung des Fehlerintervalls:
217     %Anstatt dreiDfehlerslope könnte man auch den Algorithmus
218     %dreiDfehler verwenden.
219     [u(2+AnzOp),o(2+AnzOp)] = dreiDfehlerslope ...
220         (p000(1),p100(1),p010(1),p001(1),p110(1),p101(1),p011(1),p200(1),...
221         p020(1),p002(1),q100(1),q010(1),q001(1),q110(1),q101(1),q011(1),...
222         q200(1),q020(1),q002(1),u(1),o(1),ArtOp(j-1),p000(j),p100(j),...
223         p010(j),p001(j),p110(j),p101(j),p011(j),p200(j),p020(j),p002(j),...
224         q100(j),q010(j),q001(j),q110(j),q101(j),q011(j),q200(j),q020(j),...
225         q002(j),u(j),o(j),p000(2+AnzOp),p100(2+AnzOp),p010(2+AnzOp),...
226         p001(2+AnzOp),p110(2+AnzOp),p101(2+AnzOp),p011(2+AnzOp),...
227         p200(2+AnzOp),p020(2+AnzOp),p002(2+AnzOp),q100(2+AnzOp),...
228         q010(2+AnzOp),q001(2+AnzOp),q110(2+AnzOp),q101(2+AnzOp),...
229         q011(2+AnzOp),q200(2+AnzOp),q020(2+AnzOp),q002(2+AnzOp),...
230         -Jx,Jx,-Jy,Jy,-Jz,Jz);
231
232
233
234     %Es wird das Fehlerintervall auf die Form [-c,c] gebracht:
235     f=(o(2+AnzOp)+u(2+AnzOp))/2;
236     p000(2+AnzOp)=p000(2+AnzOp)+f;
237     o(2+AnzOp)=o(2+AnzOp)-f;
238     u(2+AnzOp)=u(2+AnzOp)-f;
239
240
241
242
243     p000(1)=p000(2+AnzOp);
244     p100(1)=p100(2+AnzOp);
245     p010(1)=p010(2+AnzOp);
246     p001(1)=p001(2+AnzOp);
247     p110(1)=p110(2+AnzOp);
248     p011(1)=p011(2+AnzOp);
249     p101(1)=p101(2+AnzOp);
250     p200(1)=p200(2+AnzOp);
```

```
    p020(1)=p020(2+AnzOp);
    p002(1)=p002(2+AnzOp);
    q100(1)=q100(2+AnzOp);
    q010(1)=q010(2+AnzOp);
    q001(1)=q001(2+AnzOp);
    q110(1)=q110(2+AnzOp);
    q011(1)=q011(2+AnzOp);
    q101(1)=q101(2+AnzOp);
    q200(1)=q200(2+AnzOp);
    q020(1)=q020(2+AnzOp);
    q002(1)=q002(2+AnzOp);
    u(1)=u(2+AnzOp);
    o(1)=o(2+AnzOp);

end;


a000=p000(2+AnzOp);
a100=p100(2+AnzOp);
a010=p010(2+AnzOp);
a001=p001(2+AnzOp);
a110=p110(2+AnzOp);
a011=p011(2+AnzOp);
a101=p101(2+AnzOp);
a200=p200(2+AnzOp);
a020=p020(2+AnzOp);
a002=p002(2+AnzOp);
b100=q100(2+AnzOp);
b010=q010(2+AnzOp);
b001=q001(2+AnzOp);
b110=q110(2+AnzOp);
b011=q011(2+AnzOp);
b101=q101(2+AnzOp);
b200=q200(2+AnzOp);
b020=q020(2+AnzOp);
b002=q002(2+AnzOp);
u=u(2+AnzOp);
o=o(2+AnzOp);
```

A.22 dreiDKoeffCVX.m

```
function [p000,p100,p010,p001,p110,p011,p101,p200,p020,p002,...
          q100,q010,q001,q110,q011,q101,q200,q020,q002] = ...
dreiDKoeffCVX...
(x1,x2,x3,x4,x5,x6,x7,x8,x9,x10,x11,x12,x13,x14,x15,x16,x17,x18,x19,x20,...
 x21,x22,x23,x24,x25,y1,y2,y3,y4,y5,y6,y7,y8,y9,y10,y11,y12,y13,y14,y15,...
 y16,y17,y18,y19,y20,y21,y22,y23,y24,y25,z1,z2,z3,z4,z5,z6,z7,z8,z9,z10,...
 z11,z12,z13,z14,z15,z16,z17,z18,z19,z20,z21,z22,z23,z24,z25,...
 Ixu,Ixo,Iyu,Iyo,Izu,Izo)

Ix=infsup(Ixu,Ixo);
```

```
11 Iy=infsup(Iyu,Iyo);
12 Iz=infsup(Izu,Izo);
13
14 c=-0.1;              %Zur "Verschärfung" der Nebenbedingungen
15 I=infsup(-1,0);      %Startwert für die Intervallabschätzung des Polynoms q
16
17 while(inf_(I)<0)     %Prüft ob die Nebenbedingungen "streng genug" sind
18     c=c+0.1;
19
20     cvx_begin        %Lösen des Optimierungsproblems mittels CVX
21         variable a000
22         variable a100
23         variable a010
24         variable a001
25         variable a110
26         variable a011
27         variable a101
28         variable a200
29         variable a020
30         variable a002
31
32         variable b100
33         variable b010
34         variable b001
35         variable b110
36         variable b011
37         variable b101
38         variable b200
39         variable b020
40         variable b002
41
42
43   minimize( (a000+a100*x1+a010*y1+a001*z1+a110*x1*y1+a011*y1*z1+...
44              a101*x1*z1+a200*x1^2+a020*y1^2+a002*z1^2-(  z1  )*...
45             (1+b100*x1+b010*y1+b001*z1+b110*x1*y1+b011*y1*z1+...
46              b101*x1*z1+b200*x1^2+b020*y1^2+b002*z1^2))^2 +...
47             (a000+a100*x2+a010*y2+a001*z2+a110*x2*y2+a011*y2*z2+...
48              a101*x2*z2+a200*x2^2+a020*y2^2+a002*z2^2-(  z2  )*...
49             (1+b100*x2+b010*y2+b001*z2+b110*x2*y2+b011*y2*z2+...
50              b101*x2*z2+b200*x2^2+b020*y2^2+b002*z2^2))^2 +...
51             (a000+a100*x3+a010*y3+a001*z3+a110*x3*y3+a011*y3*z3+...
52              a101*x3*z3+a200*x3^2+a020*y3^2+a002*z3^2-(  z3  )*...
53             (1+b100*x3+b010*y3+b001*z3+b110*x3*y3+b011*y3*z3+...
54              b101*x3*z3+b200*x3^2+b020*y3^2+b002*z3^2))^2 +...
55             (a000+a100*x4+a010*y4+a001*z4+a110*x4*y4+a011*y4*z4+...
56              a101*x4*z4+a200*x4^2+a020*y4^2+a002*z4^2-(  z4  )*...
57             (1+b100*x4+b010*y4+b001*z4+b110*x4*y4+b011*y4*z4+...
58              b101*x4*z4+b200*x4^2+b020*y4^2+b002*z4^2))^2 +...
59             (a000+a100*x5+a010*y5+a001*z5+a110*x5*y5+a011*y5*z5+...
60              a101*x5*z5+a200*x5^2+a020*y5^2+a002*z5^2-(  z5  )*...
61             (1+b100*x5+b010*y5+b001*z5+b110*x5*y5+b011*y5*z5+...
62              b101*x5*z5+b200*x5^2+b020*y5^2+b002*z5^2))^2 +...
63             (a000+a100*x6+a010*y6+a001*z6+a110*x6*y6+a011*y6*z6+...
64              a101*x6*z6+a200*x6^2+a020*y6^2+a002*z6^2-(  z6  )*...
```

```
(1+b100*x6+b010*y6+b001*z6+b110*x6*y6+b011*y6*z6+...
 b101*x6*z6+b200*x6^2+b020*y6^2+b002*z6^2))^2 +...
(a000+a100*x7+a010*y7+a001*z7+a110*x7*y7+a011*y7*z7+...
 a101*x7*z7+a200*x7^2+a020*y7^2+a002*z7^2-(  z7  )*...
(1+b100*x7+b010*y7+b001*z7+b110*x7*y7+b011*y7*z7+...
 b101*x7*z7+b200*x7^2+b020*y7^2+b002*z7^2))^2 +...
(a000+a100*x8+a010*y8+a001*z8+a110*x8*y8+a011*y8*z8+...
 a101*x8*z8+a200*x8^2+a020*y8^2+a002*z8^2-(  z8  )*...
(1+b100*x8+b010*y8+b001*z8+b110*x8*y8+b011*y8*z8+...
 b101*x8*z8+b200*x8^2+b020*y8^2+b002*z8^2))^2 +...
(a000+a100*x9+a010*y9+a001*z9+a110*x9*y9+a011*y9*z9+...
 a101*x9*z9+a200*x9^2+a020*y9^2+a002*z9^2-(  z9  )*...
(1+b100*x9+b010*y9+b001*z9+b110*x9*y9+b011*y9*z9+...
 b101*x9*z9+b200*x9^2+b020*y9^2+b002*z9^2))^2 +...
(a000+a100*x10+a010*y10+a001*z10+a110*x10*y10+a011*y10*z10+...
 a101*x10*z10+a200*x10^2+a020*y10^2+a002*z10^2-(  z10  )*...
(1+b100*x10+b010*y10+b001*z10+b110*x10*y10+b011*y10*z10+...
 b101*x10*z10+b200*x10^2+b020*y10^2+b002*z10^2))^2 +...
(a000+a100*x11+a010*y11+a001*z11+a110*x11*y11+a011*y11*z11+...
 a101*x11*z11+a200*x11^2+a020*y11^2+a002*z11^2-(  z11  )*...
(1+b100*x11+b010*y11+b001*z11+b110*x11*y11+b011*y11*z11+...
 b101*x11*z11+b200*x11^2+b020*y11^2+b002*z11^2))^2 +...
(a000+a100*x12+a010*y12+a001*z12+a110*x12*y12+a011*y12*z12+...
 a101*x12*z12+a200*x12^2+a020*y12^2+a002*z12^2-(  z12  )*...
(1+b100*x12+b010*y12+b001*z12+b110*x12*y12+b011*y12*z12+...
 b101*x12*z12+b200*x12^2+b020*y12^2+b002*z12^2))^2 +...
(a000+a100*x13+a010*y13+a001*z13+a110*x13*y13+a011*y13*z13+...
 a101*x13*z13+a200*x13^2+a020*y13^2+a002*z13^2-(  z13  )*...
(1+b100*x13+b010*y13+b001*z13+b110*x13*y13+b011*y13*z13+...
 b101*x13*z13+b200*x13^2+b020*y13^2+b002*z13^2))^2 +...
(a000+a100*x14+a010*y14+a001*z14+a110*x14*y14+a011*y14*z14+...
 a101*x14*z14+a200*x14^2+a020*y14^2+a002*z14^2-(  z14  )*...
(1+b100*x14+b010*y14+b001*z14+b110*x14*y14+b011*y14*z14+...
 b101*x14*z14+b200*x14^2+b020*y14^2+b002*z14^2))^2 +...
(a000+a100*x15+a010*y15+a001*z15+a110*x15*y15+a011*y15*z15+...
 a101*x15*z15+a200*x15^2+a020*y15^2+a002*z15^2-(  z15  )*...
(1+b100*x15+b010*y15+b001*z15+b110*x15*y15+b011*y15*z15+...
 b101*x15*z15+b200*x15^2+b020*y15^2+b002*z15^2))^2 +...
(a000+a100*x16+a010*y16+a001*z16+a110*x16*y16+a011*y16*z16+...
 a101*x16*z16+a200*x16^2+a020*y16^2+a002*z16^2-(  z16  )*...
(1+b100*x16+b010*y16+b001*z16+b110*x16*y16+b011*y16*z16+...
 b101*x16*z16+b200*x16^2+b020*y16^2+b002*z16^2))^2 +...
(a000+a100*x17+a010*y17+a001*z17+a110*x17*y17+a011*y17*z17+...
 a101*x17*z17+a200*x17^2+a020*y17^2+a002*z17^2-(  z17  )*...
(1+b100*x17+b010*y17+b001*z17+b110*x17*y17+b011*y17*z17+...
 b101*x17*z17+b200*x17^2+b020*y17^2+b002*z17^2))^2 +...
(a000+a100*x18+a010*y18+a001*z18+a110*x18*y18+a011*y18*z18+...
 a101*x18*z18+a200*x18^2+a020*y18^2+a002*z18^2-(  z18  )*...
(1+b100*x18+b010*y18+b001*z18+b110*x18*y18+b011*y18*z18+...
 b101*x18*z18+b200*x18^2+b020*y18^2+b002*z18^2))^2 +...
(a000+a100*x19+a010*y19+a001*z19+a110*x19*y19+a011*y19*z19+...
 a101*x19*z19+a200*x19^2+a020*y19^2+a002*z19^2-(  z19  )*...
(1+b100*x19+b010*y19+b001*z19+b110*x19*y19+b011*y19*z19+...
 b101*x19*z19+b200*x19^2+b020*y19^2+b002*z19^2))^2 +...
```

```
119             (a000+a100*x20+a010*y20+a001*z20+a110*x20*y20+a011*y20*z20+...
120              a101*x20*z20+a200*x20^2+a020*y20^2+a002*z20^2-(  z20  )*...
121             (1+b100*x20+b010*y20+b001*z20+b110*x20*y20+b011*y20*z20+...
122              b101*x20*z20+b200*x20^2+b020*y20^2+b002*z20^2))^2 +...
123             (a000+a100*x21+a010*y21+a001*z21+a110*x21*y21+a011*y21*z21+...
124              a101*x21*z21+a200*x21^2+a020*y21^2+a002*z21^2-(  z21  )*...
125             (1+b100*x21+b010*y21+b001*z21+b110*x21*y21+b011*y21*z21+...
126              b101*x21*z21+b200*x21^2+b020*y21^2+b002*z21^2))^2 +...
127             (a000+a100*x22+a010*y22+a001*z22+a110*x22*y22+a011*y22*z22+...
128              a101*x22*z22+a200*x22^2+a020*y22^2+a002*z22^2-(  z22  )*...
129             (1+b100*x22+b010*y22+b001*z22+b110*x22*y22+b011*y22*z22+...
130              b101*x22*z22+b200*x22^2+b020*y22^2+b002*z22^2))^2 +...
131             (a000+a100*x23+a010*y23+a001*z23+a110*x23*y23+a011*y23*z23+...
132              a101*x23*z23+a200*x23^2+a020*y23^2+a002*z23^2-(  z23  )*...
133             (1+b100*x23+b010*y23+b001*z23+b110*x23*y23+b011*y23*z23+...
134              b101*x23*z23+b200*x23^2+b020*y23^2+b002*z23^2))^2 +...
135             (a000+a100*x24+a010*y24+a001*z24+a110*x24*y24+a011*y24*z24+...
136              a101*x24*z24+a200*x24^2+a020*y24^2+a002*z24^2-(  z24  )*...
137             (1+b100*x24+b010*y24+b001*z24+b110*x24*y24+b011*y24*z24+...
138              b101*x24*z24+b200*x24^2+b020*y24^2+b002*z24^2))^2 +...
139             (a000+a100*x25+a010*y25+a001*z25+a110*x25*y25+a011*y25*z25+...
140              a101*x25*z25+a200*x25^2+a020*y25^2+a002*z25^2-(  z25  )*...
141             (1+b100*x25+b010*y25+b001*z25+b110*x25*y25+b011*y25*z25+...
142              b101*x25*z25+b200*x25^2+b020*y25^2+b002*z25^2))^2 )
143
144
145         subject to
146
147             (1+b100*x1+b010*y1+b001*z1+b110*x1*y1+b011*y1*z1+b101*x1*z1+...
148                 b200*x1^2+b020*y1^2+b002*z1^2)>c
149             (1+b100*x2+b010*y2+b001*z2+b110*x2*y2+b011*y2*z2+b101*x2*z2+...
150                 b200*x2^2+b020*y2^2+b002*z2^2)>c
151             (1+b100*x3+b010*y3+b001*z3+b110*x3*y3+b011*y3*z3+b101*x3*z3+...
152                 b200*x3^2+b020*y3^2+b002*z3^2)>c
153             (1+b100*x4+b010*y4+b001*z4+b110*x4*y4+b011*y4*z4+b101*x4*z4+...
154                 b200*x4^2+b020*y4^2+b002*z4^2)>c
155             (1+b100*x5+b010*y5+b001*z5+b110*x5*y5+b011*y5*z5+b101*x5*z5+...
156                 b200*x5^2+b020*y5^2+b002*z5^2)>c
157             (1+b100*x6+b010*y6+b001*z6+b110*x6*y6+b011*y6*z6+b101*x6*z6+...
158                 b200*x6^2+b020*y6^2+b002*z6^2)>c
159             (1+b100*x7+b010*y7+b001*z7+b110*x7*y7+b011*y7*z7+b101*x7*z7+...
160                 b200*x7^2+b020*y7^2+b002*z7^2)>c
161             (1+b100*x8+b010*y8+b001*z8+b110*x8*y8+b011*y8*z8+b101*x8*z8+...
162                 b200*x8^2+b020*y8^2+b002*z8^2)>c
163             (1+b100*x9+b010*y9+b001*z9+b110*x9*y9+b011*y9*z9+b101*x9*z9+...
164                 b200*x9^2+b020*y9^2+b002*z9^2)>c
165             (1+b100*x10+b010*y10+b001*z10+b110*x10*y10+b011*y10*z10+...
166                 b101*x10*z10+b200*x10^2+b020*y10^2+b002*z10^2)>c
167             (1+b100*x11+b010*y11+b001*z11+b110*x11*y11+b011*y11*z11+...
168                 b101*x11*z11+b200*x11^2+b020*y11^2+b002*z11^2)>c
169             (1+b100*x12+b010*y12+b001*z12+b110*x12*y12+b011*y12*z12+...
170                 b101*x12*z12+b200*x12^2+b020*y12^2+b002*z12^2)>c
171             (1+b100*x13+b010*y13+b001*z13+b110*x13*y13+b011*y13*z13+...
172                 b101*x13*z13+b200*x13^2+b020*y13^2+b002*z13^2)>c
```

```
173             (1+b100*x14+b010*y14+b001*z14+b110*x14*y14+b011*y14*z14+...
174                 b101*x14*z14+b200*x14^2+b020*y14^2+b002*z14^2)>c
175             (1+b100*x15+b010*y15+b001*z15+b110*x15*y15+b011*y15*z15+...
176                 b101*x15*z15+b200*x15^2+b020*y15^2+b002*z15^2)>c
177             (1+b100*x16+b010*y16+b001*z16+b110*x16*y16+b011*y16*z16+...
178                 b101*x16*z16+b200*x16^2+b020*y16^2+b002*z16^2)>c
179             (1+b100*x17+b010*y17+b001*z17+b110*x17*y17+b011*y17*z17+...
180                 b101*x17*z17+b200*x17^2+b020*y17^2+b002*z17^2)>c
181             (1+b100*x18+b010*y18+b001*z18+b110*x18*y18+b011*y18*z18+...
182                 b101*x18*z18+b200*x18^2+b020*y18^2+b002*z18^2)>c
183             (1+b100*x19+b010*y19+b001*z19+b110*x19*y19+b011*y19*z19+...
184                 b101*x19*z19+b200*x19^2+b020*y19^2+b002*z19^2)>c
185             (1+b100*x20+b010*y20+b001*z20+b110*x20*y20+b011*y20*z20+...
186                 b101*x20*z20+b200*x20^2+b020*y20^2+b002*z20^2)>c
187             (1+b100*x21+b010*y21+b001*z21+b110*x21*y21+b011*y21*z21+...
188                 b101*x21*z21+b200*x21^2+b020*y21^2+b002*z21^2)>c
189             (1+b100*x22+b010*y22+b001*z22+b110*x22*y22+b011*y22*z22+...
190                 b101*x22*z22+b200*x22^2+b020*y22^2+b002*z22^2)>c
191             (1+b100*x23+b010*y23+b001*z23+b110*x23*y23+b011*y23*z23+...
192                 b101*x23*z23+b200*x23^2+b020*y23^2+b002*z23^2)>c
193             (1+b100*x24+b010*y24+b001*z24+b110*x24*y24+b011*y24*z24+...
194                 b101*x24*z24+b200*x24^2+b020*y24^2+b002*z24^2)>c
195             (1+b100*x25+b010*y25+b001*z25+b110*x25*y25+b011*y25*z25+...
196                 b101*x25*z25+b200*x25^2+b020*y25^2+b002*z25^2)>c
197 
198 
199     cvx_end
200 
201     q100=b100;
202     q010=b010;
203     q001=b001;
204     q110=b110;
205     q011=b011;
206     q101=b101;
207     q200=b200;
208     q020=b020;
209     q002=b002;
210 
211     %Intervallabschätzung von q
212     I=(1+q100*Ix+q010*Iy+q001*Iz+q110*Ix*Iy+q011*Iy*Iz+q101*Ix*Iz+...
213         q200*power(Ix,2)+q020*power(Iy,2)+q002*power(Iz,2));
214 
215 
216 end
217 
218 
219 p000=a000;
220 p100=a100;
221 p010=a010;
222 p001=a001;
223 p110=a110;
224 p011=a011;
225 p101=a101;
226 p200=a200;
```

```
227 p020=a020;
228 p002=a002;
```

A.23 dreiDfehler.m

```
1  function [u,o] = dreiDfehler ...
2      (pa000,pa100,pa010,pa001,pa110,pa101,pa011,pa200,pa020,pa002,...
3       qa100,qa010,qa001,qa110,qa101,qa011,qa200,qa020,qa002,ua,oa,operation,...
4       pb000,pb100,pb010,pb001,pb110,pb101,pb011,pb200,pb020,pb002,...
5       qb100,qb010,qb001,qb110,qb101,qb011,qb200,qb020,qb002,ub,ob,...
6       p000,p100,p010,p001,p110,p101,p011,p200,p020,p002,...
7       q100,q010,q001,q110,q101,q011,q200,q020,q002,Ixu,Ixo,Iyu,Iyo,Izu,Izo)
8
9  Ix=infsup(Ixu,Ixo);
10 Iy=infsup(Iyu,Iyo);
11 Iz=infsup(Izu,Izo);
12 Ia=infsup(ua,oa);
13 Ib=infsup(ub,ob);
14
15 if(operation==1)%Addition
16
17     %Einsetzen des Defintionsbereiches in f(x)
18     I1=((pa000+pa100*Ix+pa010*Iy+pa001*Iz+pa110*Ix*Iy+pa011*Iy*Iz+...
19         pa101*Ix*Iz+pa200*power(Ix,2)+pa020*power(Iy,2)+...
20         pa002*power(Iz,2)+Ia)/...
21         (1+qa100*Ix+qa010*Iy+qa001*Iz+qa110*Ix*Iy+qa011*Iy*Iz+qa101*Ix*Iz+...
22         qa200*power(Ix,2)+qa020*power(Iy,2)+qa002*power(Iz,2))+...
23         (pb000+pb100*Ix+pb010*Iy+pb001*Iz+pb110*Ix*Iy+pb011*Iy*Iz+...
24         pb101*Ix*Iz+pb200*power(Ix,2)+pb020*power(Iy,2)+...
25         pb002*power(Iz,2)+Ib)/...
26         (1+qb100*Ix+qb010*Iy+qb001*Iz+qb110*Ix*Iy+qb011*Iy*Iz+qb101*Ix*Iz+...
27         qb200*power(Ix,2)+qb020*power(Iy,2)+qb002*power(Iz,2)))*...
28         (1+q100*Ix+q010*Iy+q001*Iz+q110*Ix*Iy+q011*Iy*Iz+q101*Ix*Iz+...
29         q200*power(Ix,2)+q020*power(Iy,2)+q002*power(Iz,2))...
30         -(p000+p100*Ix+p010*Iy+p001*Iz+p110*Ix*Iy+p011*Iy*Iz+p101*Ix*Iz+...
31         p200*power(Ix,2)+p020*power(Iy,2)+p002*power(Iz,2));
32
33 elseif(operation==2) %Subtraktion
34
35     I1=((pa000+pa100*Ix+pa010*Iy+pa001*Iz+pa110*Ix*Iy+pa011*Iy*Iz+...
36         pa101*Ix*Iz+pa200*power(Ix,2)+pa020*power(Iy,2)+...
37         pa002*power(Iz,2)+Ia)/...
38         (1+qa100*Ix+qa010*Iy+qa001*Iz+qa110*Ix*Iy+qa011*Iy*Iz+qa101*Ix*Iz+...
39         qa200*power(Ix,2)+qa020*power(Iy,2)+qa002*power(Iz,2))-...
40         (pb000+pb100*Ix+pb010*Iy+pb001*Iz+pb110*Ix*Iy+pb011*Iy*Iz+...
41         pb101*Ix*Iz+pb200*power(Ix,2)+pb020*power(Iy,2)+...
42         pb002*power(Iz,2)+Ib)/...
43         (1+qb100*Ix+qb010*Iy+qb001*Iz+qb110*Ix*Iy+qb011*Iy*Iz+qb101*Ix*Iz+...
44         qb200*power(Ix,2)+qb020*power(Iy,2)+qb002*power(Iz,2)))*...
45         (1+q100*Ix+q010*Iy+q001*Iz+q110*Ix*Iy+q011*Iy*Iz+q101*Ix*Iz+...
46         q200*power(Ix,2)+q020*power(Iy,2)+q002*power(Iz,2))...
```

```
        -(p000+p100*Ix+p010*Iy+p001*Iz+p110*Ix*Iy+p011*Iy*Iz+p101*Ix*Iz+...
        p200*power(Ix,2)+p020*power(Iy,2)+p002*power(Iz,2));

elseif(operation==3) %Multiplikation

    I1=((pa000+pa100*Ix+pa010*Iy+pa001*Iz+pa110*Ix*Iy+pa011*Iy*Iz+...
        pa101*Ix*Iz+pa200*power(Ix,2)+pa020*power(Iy,2)+...
        pa002*power(Iz,2)+Ia)/...
        (1+qa100*Ix+qa010*Iy+qa001*Iz+qa110*Ix*Iy+qa011*Iy*Iz+qa101*Ix*Iz+...
        qa200*power(Ix,2)+qa020*power(Iy,2)+qa002*power(Iz,2))*...
        (pb000+pb100*Ix+pb010*Iy+pb001*Iz+pb110*Ix*Iy+pb011*Iy*Iz+...
        pb101*Ix*Iz+pb200*power(Ix,2)+pb020*power(Iy,2)+...
        pb002*power(Iz,2)+Ib)/...
        (1+qb100*Ix+qb010*Iy+qb001*Iz+qb110*Ix*Iy+qb011*Iy*Iz+qb101*Ix*Iz+...
        qb200*power(Ix,2)+qb020*power(Iy,2)+qb002*power(Iz,2)))*...
        (1+q100*Ix+q010*Iy+q001*Iz+q110*Ix*Iy+q011*Iy*Iz+q101*Ix*Iz+...
        q200*power(Ix,2)+q020*power(Iy,2)+q002*power(Iz,2))...
        -(p000+p100*Ix+p010*Iy+p001*Iz+p110*Ix*Iy+p011*Iy*Iz+p101*Ix*Iz+...
        p200*power(Ix,2)+p020*power(Iy,2)+p002*power(Iz,2));

elseif(operation==4) %Division

    I1=((pa000+pa100*Ix+pa010*Iy+pa001*Iz+pa110*Ix*Iy+pa011*Iy*Iz+...
        pa101*Ix*Iz+pa200*power(Ix,2)+pa020*power(Iy,2)+...
        pa002*power(Iz,2)+Ia)/...
        (1+qa100*Ix+qa010*Iy+qa001*Iz+qa110*Ix*Iy+qa011*Iy*Iz+...
        qa101*Ix*Iz+qa200*power(Ix,2)+qa020*power(Iy,2)+qa002*power(Iz,2))/...
        (pb000+pb100*Ix+pb010*Iy+pb001*Iz+pb110*Ix*Iy+pb011*Iy*Iz+...
        pb101*Ix*Iz+pb200*power(Ix,2)+pb020*power(Iy,2)+...
        pb002*power(Iz,2)+Ib)/...
        (1+qb100*Ix+qb010*Iy+qb001*Iz+qb110*Ix*Iy+qb011*Iy*Iz+...
        qb101*Ix*Iz+qb200*power(Ix,2)+qb020*power(Iy,2)+qb002*power(Iz,2)))*...
        (1+q100*Ix+q010*Iy+q001*Iz+q110*Ix*Iy+q011*Iy*Iz+q101*Ix*Iz+...
        q200*power(Ix,2)+q020*power(Iy,2)+q002*power(Iz,2))...
        -(p000+p100*Ix+p010*Iy+p001*Iz+p110*Ix*Iy+p011*Iy*Iz+p101*Ix*Iz+...
        p200*power(Ix,2)+p020*power(Iy,2)+p002*power(Iz,2));

end;

u=inf_(I1);
o=sup(I1);
```

A.24 dreiDfehlerslope.m

```
function [u,o] = dreiDfehlerslope ...
    (pa000,pa100,pa010,pa001,pa110,pa101,pa011,pa200,pa020,pa002,...
     qa100,qa010,qa001,qa110,qa101,qa011,qa200,qa020,qa002,ua,oa,operation,...
     pb000,pb100,pb010,pb001,pb110,pb101,pb011,pb200,pb020,pb002,...
     qb100,qb010,qb001,qb110,qb101,qb011,qb200,qb020,qb002,ub,ob,...
     p000,p100,p010,p001,p110,p101,p011,p200,p020,p002,...
     q100,q010,q001,q110,q101,q011,q200,q020,q002,Ixu,Ixo,Iyu,Iyo,Izu,Izo)
```

```
8
9
10 za1=Ixu;
11 za2=Iyu;
12 za3=Izu;
13 zb1=Ixo;
14 zb2=Iyo;
15 zb3=Izo;
16
17 Ix=infsup(Ixu,Ixo);
18 Iy=infsup(Iyu,Iyo);
19 Iz=infsup(Izu,Izo);
20 Ia=infsup(ua,oa);
21 Ib=infsup(ub,ob);
22
23 %Berechnung der Slopes erster und zweiter Ordnung (f[za,x],f[za,zb,x])
24 [sa000za,sa100za,sa010za,sa001za,sb000za,sb010za,...
25  sb001za,sc000za,sc001za,s2za] = ...
26     slope3(pa000,pa100,pa010,pa001,pa110,pa101,pa011,pa200,pa020,pa002,...
27            qa100,qa010,qa001,qa110,qa101,qa011,qa200,qa020,qa002,Ia,...
28            pb000,pb100,pb010,pb001,pb110,pb101,pb011,pb200,pb020,pb002,...
29            qb100,qb010,qb001,qb110,qb101,qb011,qb200,qb020,qb002,Ib,...
30            p000,p100,p010,p001,p110,p101,p011,p200,p020,p002,q100,q010,...
31            q001,q110,q101,q011,q200,q020,q002,...
32            operation,za1,za2,za3,zb1,zb2,zb3,Ixu,Ixo,Iyu,Iyo,Izu,Izo);
33
34 %Berechnung der Slopes erster und zweiter Ordnung (f[zb,x],f[zb,za,x])
35 [sa000zb,sa100zb,sa010zb,sa001zb,sb000zb,sb010zb,...
36  sb001zb,sc000zb,sc001zb,s2zb] = ...
37     slope3(pa000,pa100,pa010,pa001,pa110,pa101,pa011,pa200,pa020,pa002,...
38            qa100,qa010,qa001,qa110,qa101,qa011,qa200,qa020,qa002,Ia,...
39            pb000,pb100,pb010,pb001,pb110,pb101,pb011,pb200,pb020,pb002,...
40            qb100,qb010,qb001,qb110,qb101,qb011,qb200,qb020,qb002,Ib,...
41            p000,p100,p010,p001,p110,p101,p011,p200,p020,p002,q100,q010,...
42            q001,q110,q101,q011,q200,q020,q002,...
43            operation,zb1,zb2,zb3,za1,za2,za3,Ixu,Ixo,Iyu,Iyo,Izu,Izo);
44
45 %Berechnung von f[za,zb]
46 fzazb=[sa000za+sa100za*zb1+sa010za*zb2+sa001za*zb3,...
47        sb000za+sb010za*zb2+sb001za*zb3,sc000za+sc001za*zb3];
48
49
50 if(operation==1)%Addition
51
52
53     fza=((pa000+pa100*za1+pa010*za2+pa001*za3+pa110*za1*za2+pa011*za2*za3+...
54         pa101*za1*za3+pa200*za1^2+pa020*za2^2+pa002*za3^2+Ia)/...
55         (1+qa100*za1+qa010*za2+qa001*za3+qa110*za1*za2+qa011*za2*za3+...
56         qa101*za1*za3+qa200*za1^2+qa020*za2^2+qa002*za3^2)+...
57         (pb000+pb100*za1+pb010*za2+pb001*za3+pb110*za1*za2+pb011*za2*za3+...
58         pb101*za1*za3+pb200*za1^2+pb020*za2^2+pb002*za3^2+Ib)/...
59         (1+qb100*za1+qb010*za2+qb001*za3+qb110*za1*za2+qb011*za2*za3+...
60         qb101*za1*za3+qb200*za1^2+qb020*za2^2+qb002*za3^2))*...
61         (1+q100*za1+q010*za2+q001*za3+q110*za1*za2+q011*za2*za3+...
```

```
62          q101*za1*za3+q200*za1^2+q020*za2^2+q002*za3^2)-...
63          (p000+p100*za1+p010*za2+p001*za3+p110*za1*za2+p011*za2*za3+...
64          p101*za1*za3+p200*za1^2+p020*za2^2+p002*za3^2);
65
66      fzb=((pa000+pa100*zb1+pa010*zb2+pa001*zb3+pa110*zb1*zb2+pa011*zb2*zb3+...
67          pa101*zb1*zb3+pa200*zb1^2+pa020*zb2^2+pa002*zb3^2+Ia)/...
68          (1+qa100*zb1+qa010*zb2+qa001*zb3+qa110*zb1*zb2+qa011*zb2*zb3+...
69          qa101*zb1*zb3+qa200*zb1^2+qa020*zb2^2+qa002*zb3^2)+...
70          (pb000+pb100*zb1+pb010*zb2+pb001*zb3+pb110*zb1*zb2+pb011*zb2*zb3+...
71          pb101*zb1*zb3+pb200*zb1^2+pb020*zb2^2+pb002*zb3^2+Ib)/...
72          (1+qb100*zb1+qb010*zb2+qb001*zb3+qb110*zb1*zb2+qb011*zb2*zb3+...
73          qb101*zb1*zb3+qb200*zb1^2+qb020*zb2^2+qb002*zb3^2))*...
74          (1+q100*zb1+q010*zb2+q001*zb3+q110*zb1*zb2+q011*zb2*zb3+...
75          q101*zb1*zb3+q200*zb1^2+q020*zb2^2+q002*zb3^2)-...
76          (p000+p100*zb1+p010*zb2+p001*zb3+p110*zb1*zb2+p011*zb2*zb3+...
77          p101*zb1*zb3+p200*zb1^2+p020*zb2^2+p002*zb3^2);
78
79      %Einsetzen des Defintionsbereiches in f(x)
80      I1=((pa000+pa100*Ix+pa010*Iy+pa001*Iz+pa110*Ix*Iy+pa011*Iy*Iz+...
81          pa101*Ix*Iz+pa200*power(Ix,2)+pa020*power(Iy,2)+...
82          pa002*power(Iz,2)+Ia)/...
83          (1+qa100*Ix+qa010*Iy+qa001*Iz+qa110*Ix*Iy+qa011*Iy*Iz+qa101*Ix*Iz+...
84          qa200*power(Ix,2)+qa020*power(Iy,2)+qa002*power(Iz,2))+...
85          (pb000+pb100*Ix+pb010*Iy+pb001*Iz+pb110*Ix*Iy+pb011*Iy*Iz+...
86          pb101*Ix*Iz+pb200*power(Ix,2)+pb020*power(Iy,2)+...
87          pb002*power(Iz,2)+Ib)/...
88          (1+qb100*Ix+qb010*Iy+qb001*Iz+qb110*Ix*Iy+qb011*Iy*Iz+qb101*Ix*Iz+...
89          qb200*power(Ix,2)+qb020*power(Iy,2)+qb002*power(Iz,2)))*...
90          (1+q100*Ix+q010*Iy+q001*Iz+q110*Ix*Iy+q011*Iy*Iz+q101*Ix*Iz+...
91          q200*power(Ix,2)+q020*power(Iy,2)+q002*power(Iz,2))...
92          -(p000+p100*Ix+p010*Iy+p001*Iz+p110*Ix*Iy+p011*Iy*Iz+p101*Ix*Iz+...
93          p200*power(Ix,2)+p020*power(Iy,2)+p002*power(Iz,2));
94
95      %Abschätzen des Fehlerintervalls mittels Slopes erster Ordnung
96      %an den Punkte za bzw. zb.
97      I2=fza+[sa000za+sa100za*Ix+sa010za*Iy+sa001za*Iz,sb000za+sb010za*Iy+...
98          sb001za*Iz,sc000za+sc001za*Iz]*[Ix-za1;Iy-za2;Iz-za3];
99      I3=fzb+[sa000zb+sa100zb*Ix+sa010zb*Iy+sa001zb*Iz,sb000zb+sb010zb*Iy+...
100         sb001zb*Iz,sc000zb+sc001zb*Iz]*[Ix-zb1;Iy-zb2;Iz-zb3];
101
102     %Abschätzen des Fehlerintervalls mittels Slopes zweiter Ordnung
103     I4=fza+(fzazb+[Ix-zb1;Iy-zb2;Iz-zb3]'*s2za)*[Ix-za1;Iy-za2;Iz-za3];
104
105     %Schneiden der Intervalle
106     I5=intersect(intersect(I1,I2),intersect(I3,I4));
107
108
109
110 elseif(operation==2) %Subtraktion
111
112
113     fza=((pa000+pa100*za1+pa010*za2+pa001*za3+pa110*za1*za2+pa011*za2*za3+...
114         pa101*za1*za3+pa200*za1^2+pa020*za2^2+pa002*za3^2+Ia)/...
115         (1+qa100*za1+qa010*za2+qa001*za3+qa110*za1*za2+qa011*za2*za3+...
```

```
        qa101*za1*za3+qa200*za1^2+qa020*za2^2+qa002*za3^2)-...
        (pb000+pb100*za1+pb010*za2+pb001*za3+pb110*za1*za2+pb011*za2*za3+...
        pb101*za1*za3+pb200*za1^2+pb020*za2^2+pb002*za3^2+Ib)/...
        (1+qb100*za1+qb010*za2+qb001*za3+qb110*za1*za2+qb011*za2*za3+...
        qb101*za1*za3+qb200*za1^2+qb020*za2^2+qb002*za3^2))*...
        (1+q100*za1+q010*za2+q001*za3+q110*za1*za2+q011*za2*za3+...
        q101*za1*za3+q200*za1^2+q020*za2^2+q002*za3^2)-...
        (p000+p100*za1+p010*za2+p001*za3+p110*za1*za2+p011*za2*za3+...
        p101*za1*za3+p200*za1^2+p020*za2^2+p002*za3^2);

    fzb=((pa000+pa100*zb1+pa010*zb2+pa001*zb3+pa110*zb1*zb2+pa011*zb2*zb3+...
        pa101*zb1*zb3+pa200*zb1^2+pa020*zb2^2+pa002*zb3^2+Ia)/...
        (1+qa100*zb1+qa010*zb2+qa001*zb3+qa110*zb1*zb2+qa011*zb2*zb3+...
        qa101*zb1*zb3+qa200*zb1^2+qa020*zb2^2+qa002*zb3^2)-...
        (pb000+pb100*zb1+pb010*zb2+pb001*zb3+pb110*zb1*zb2+pb011*zb2*zb3+...
        pb101*zb1*zb3+pb200*zb1^2+pb020*zb2^2+pb002*zb3^2+Ib)/...
        (1+qb100*zb1+qb010*zb2+qb001*zb3+qb110*zb1*zb2+qb011*zb2*zb3+...
        qb101*zb1*zb3+qb200*zb1^2+qb020*zb2^2+qb002*zb3^2))*...
        (1+q100*zb1+q010*zb2+q001*zb3+q110*zb1*zb2+q011*zb2*zb3+...
        q101*zb1*zb3+q200*zb1^2+q020*zb2^2+q002*zb3^2)-...
        (p000+p100*zb1+p010*zb2+p001*zb3+p110*zb1*zb2+p011*zb2*zb3+...
        p101*zb1*zb3+p200*zb1^2+p020*zb2^2+p002*zb3^2);


    I1=((pa000+pa100*Ix+pa010*Iy+pa001*Iz+pa110*Ix*Iy+pa011*Iy*Iz+...
        pa101*Ix*Iz+pa200*power(Ix,2)+pa020*power(Iy,2)+...
        pa002*power(Iz,2)+Ia)/...
        (1+qa100*Ix+qa010*Iy+qa001*Iz+qa110*Ix*Iy+qa011*Iy*Iz+qa101*Ix*Iz+...
        qa200*power(Ix,2)+qa020*power(Iy,2)+qa002*power(Iz,2))-...
        (pb000+pb100*Ix+pb010*Iy+pb001*Iz+pb110*Ix*Iy+pb011*Iy*Iz+...
        pb101*Ix*Iz+pb200*power(Ix,2)+pb020*power(Iy,2)+...
        pb002*power(Iz,2)+Ib)/...
        (1+qb100*Ix+qb010*Iy+qb001*Iz+qb110*Ix*Iy+qb011*Iy*Iz+qb101*Ix*Iz+...
        qb200*power(Ix,2)+qb020*power(Iy,2)+qb002*power(Iz,2)))*...
        (1+q100*Ix+q010*Iy+q001*Iz+q110*Ix*Iy+q011*Iy*Iz+q101*Ix*Iz+...
        q200*power(Ix,2)+q020*power(Iy,2)+q002*power(Iz,2))...
        -(p000+p100*Ix+p010*Iy+p001*Iz+p110*Ix*Iy+p011*Iy*Iz+p101*Ix*Iz+...
        p200*power(Ix,2)+p020*power(Iy,2)+p002*power(Iz,2));


    I2=fza+[sa000za+sa100za*Ix+sa010za*Iy+sa001za*Iz,sb000za+sb010za*Iy+...
        sb001za*Iz,sc000za+sc001za*Iz]*[Ix-za1;Iy-za2;Iz-za3];
    I3=fzb+[sa000zb+sa100zb*Ix+sa010zb*Iy+sa001zb*Iz,sb000zb+sb010zb*Iy+...
        sb001zb*Iz,sc000zb+sc001zb*Iz]*[Ix-zb1;Iy-zb2;Iz-zb3];


    I4=fza+(fzazb+[Ix-zb1;Iy-zb2;Iz-zb3]'*s2za)*[Ix-za1;Iy-za2;Iz-za3];
    I5=intersect(intersect(I1,I2),intersect(I3,I4));



elseif(operation==3) %Multiplikation


```

```
    fza=((pa000+pa100*za1+pa010*za2+pa001*za3+pa110*za1*za2+pa011*za2*za3+...
        pa101*za1*za3+pa200*za1^2+pa020*za2^2+pa002*za3+Ia)/...
        (1+qa100*za1+qa010*za2+qa001*za3+qa110*za1*za2+qa011*za2*za3+...
        qa101*za1*za3+qa200*za1^2+qa020*za2^2+qa002*za3)*...
        (pb000+pb100*za1+pb010*za2+pb001*za3+pb110*za1*za2+pb011*za2*za3+...
        pb101*za1*za3+pb200*za1^2+pb020*za2^2+pb002*za3+Ib)/...
        (1+qb100*za1+qb010*za2+qb001*za3+qb110*za1*za2+qb011*za2*za3+...
        qb101*za1*za3+qb200*za1^2+qb020*za2^2+qb002*za3))*...
        (1+q100*za1+q010*za2+q001*za3+q110*za1*za2+q011*za2*za3+...
        q101*za1*za3+q200*za1^2+q020*za2^2+q002*za3)-...
        (p000+p100*za1+p010*za2+p001*za3+p110*za1*za2+p011*za2*za3+...
        p101*za1*za3+p200*za1^2+p020*za2^2+p002*za3);

    fzb=((pa000+pa100*zb1+pa010*zb2+pa001*zb3+pa110*zb1*zb2+pa011*zb2*zb3+...
        pa101*zb1*zb3+pa200*zb1^2+pa020*zb2^2+pa002*zb3+Ia)/...
        (1+qa100*zb1+qa010*zb2+qa001*zb3+qa110*zb1*zb2+qa011*zb2*zb3+...
        qa101*zb1*zb3+qa200*zb1^2+qa020*zb2^2+qa002*zb3)*...
        (pb000+pb100*zb1+pb010*zb2+pb001*zb3+pb110*zb1*zb2+pb011*zb2*zb3+...
        pb101*zb1*zb3+pb200*zb1^2+pb020*zb2^2+pb002*zb3+Ib)/...
        (1+qb100*zb1+qb010*zb2+qb001*zb3+qb110*zb1*zb2+qb011*zb2*zb3+...
        qb101*zb1*zb3+qb200*zb1^2+qb020*zb2^2+qb002*zb3))*...
        (1+q100*zb1+q010*zb2+q001*zb3+q110*zb1*zb2+q011*zb2*zb3+...
        q101*zb1*zb3+q200*zb1^2+q020*zb2^2+q002*zb3)-...
        (p000+p100*zb1+p010*zb2+p001*zb3+p110*zb1*zb2+p011*zb2*zb3+...
        p101*zb1*zb3+p200*zb1^2+p020*zb2^2+p002*zb3);


    I1=((pa000+pa100*Ix+pa010*Iy+pa001*Iz+pa110*Ix*Iy+pa011*Iy*Iz+...
        pa101*Ix*Iz+pa200*power(Ix,2)+pa020*power(Iy,2)+...
        pa002*power(Iz,2)+Ia)/...
        (1+qa100*Ix+qa010*Iy+qa001*Iz+qa110*Ix*Iy+qa011*Iy*Iz+qa101*Ix*Iz+...
        qa200*power(Ix,2)+qa020*power(Iy,2)+qa002*power(Iz,2))*...
        (pb000+pb100*Ix+pb010*Iy+pb001*Iz+pb110*Ix*Iy+pb011*Iy*Iz+...
        pb101*Ix*Iz+pb200*power(Ix,2)+pb020*power(Iy,2)+...
        pb002*power(Iz,2)+Ib)/...
        (1+qb100*Ix+qb010*Iy+qb001*Iz+qb110*Ix*Iy+qb011*Iy*Iz+qb101*Ix*Iz+...
        qb200*power(Ix,2)+qb020*power(Iy,2)+qb002*power(Iz,2)))*...
        (1+q100*Ix+q010*Iy+q001*Iz+q110*Ix*Iy+q011*Iy*Iz+q101*Ix*Iz+...
        q200*power(Ix,2)+q020*power(Iy,2)+q002*power(Iz,2))...
        -(p000+p100*Ix+p010*Iy+p001*Iz+p110*Ix*Iy+p011*Iy*Iz+p101*Ix*Iz+...
        p200*power(Ix,2)+p020*power(Iy,2)+p002*power(Iz,2));


    I2=fza+[sa000za+sa100za*Ix+sa010za*Iy+sa001za*Iz,sb000za+sb010za*Iy+...
        sb001za*Iz,sc000za+sc001za*Iz]*[Ix-za1;Iy-za2;Iz-za3];
    I3=fzb+[sa000zb+sa100zb*Ix+sa010zb*Iy+sa001zb*Iz,sb000zb+sb010zb*Iy+...
        sb001zb*Iz,sc000zb+sc001zb*Iz]*[Ix-zb1;Iy-zb2;Iz-zb3];


    I4=fza+(fzazb+[Ix-zb1;Iy-zb2;Iz-zb3]'*s2za)*[Ix-za1;Iy-za2;Iz-za3];
    I5=intersect(intersect(I1,I2),intersect(I3,I4));

elseif(operation==4) %Division

```

```
224
225     fza=((pa000+pa100*za1+pa010*za2+pa001*za3+pa110*za1*za2+pa011*za2*za3+...
226         pa101*za1*za3+pa200*za1^2+pa020*za2^2+pa002*za3+Ia)/...
227         (1+qa100*za1+qa010*za2+qa001*za3+qa110*za1*za2+qa011*za2*za3+...
228         qa101*za1*za3+qa200*za1^2+qa020*za2^2+qa002*za3)/...
229         (pb000+pb100*za1+pb010*za2+pb001*za3+pb110*za1*za2+pb011*za2*za3+...
230         pb101*za1*za3+pb200*za1^2+pb020*za2^2+pb002*za3+Ib)/...
231         (1+qb100*za1+qb010*za2+qb001*za3+qb110*za1*za2+qb011*za2*za3+...
232         qb101*za1*za3+qb200*za1^2+qb020*za2^2+qb002*za3))*...
233         (1+q100*za1+q010*za2+q001*za3+q110*za1*za2+q011*za2*za3+...
234         q101*za1*za3+q200*za1^2+q020*za2^2+q002*za3)-...
235         (p000+p100*za1+p010*za2+p001*za3+p110*za1*za2+p011*za2*za3+...
236         p101*za1*za3+p200*za1^2+p020*za2^2+p002*za3);
237
238     fzb=((pa000+pa100*zb1+pa010*zb2+pa001*zb3+pa110*zb1*zb2+pa011*zb2*zb3+...
239         pa101*zb1*zb3+pa200*zb1^2+pa020*zb2^2+pa002*zb3+Ia)/...
240         (1+qa100*zb1+qa010*zb2+qa001*zb3+qa110*zb1*zb2+qa011*zb2*zb3+...
241         qa101*zb1*zb3+qa200*zb1^2+qa020*zb2^2+qa002*zb3)/...
242         (pb000+pb100*zb1+pb010*zb2+pb001*zb3+pb110*zb1*zb2+pb011*zb2*zb3+...
243         pb101*zb1*zb3+pb200*zb1^2+pb020*zb2^2+pb002*zb3+Ib)/...
244         (1+qb100*zb1+qb010*zb2+qb001*zb3+qb110*zb1*zb2+qb011*zb2*zb3+...
245         qb101*zb1*zb3+qb200*zb1^2+qb020*zb2^2+qb002*zb3))*...
246         (1+q100*zb1+q010*zb2+q001*zb3+q110*zb1*zb2+q011*zb2*zb3+...
247         q101*zb1*zb3+q200*zb1^2+q020*zb2^2+q002*zb3)-...
248         (p000+p100*zb1+p010*zb2+p001*zb3+p110*zb1*zb2+p011*zb2*zb3+...
249         p101*zb1*zb3+p200*zb1^2+p020*zb2^2+p002*zb3);
250
251
252     I1=((pa000+pa100*Ix+pa010*Iy+pa001*Iz+pa110*Ix*Iy+pa011*Iy*Iz+...
253         pa101*Ix*Iz+pa200*power(Ix,2)+pa020*power(Iy,2)+...
254         pa002*power(Iz,2)+Ia)/...
255         (1+qa100*Ix+qa010*Iy+qa001*Iz+qa110*Ix*Iy+qa011*Iy*Iz+...
256         qa101*Ix*Iz+qa200*power(Ix,2)+qa020*power(Iy,2)+qa002*power(Iz,2))/...
257         (pb000+pb100*Ix+pb010*Iy+pb001*Iz+pb110*Ix*Iy+pb011*Iy*Iz+...
258         pb101*Ix*Iz+pb200*power(Ix,2)+pb020*power(Iy,2)+...
259         pb002*power(Iz,2)+Ib)/...
260         (1+qb100*Ix+qb010*Iy+qb001*Iz+qb110*Ix*Iy+qb011*Iy*Iz+...
261         qb101*Ix*Iz+qb200*power(Ix,2)+qb020*power(Iy,2)+qb002*power(Iz,2)))*...
262         (1+q100*Ix+q010*Iy+q001*Iz+q110*Ix*Iy+q011*Iy*Iz+q101*Ix*Iz+...
263         q200*power(Ix,2)+q020*power(Iy,2)+q002*power(Iz,2))...
264         -(p000+p100*Ix+p010*Iy+p001*Iz+p110*Ix*Iy+p011*Iy*Iz+p101*Ix*Iz+...
265         p200*power(Ix,2)+p020*power(Iy,2)+p002*power(Iz,2));
266
267
268     I2=fza+[sa000za+sa100za*Ix+sa010za*Iy+sa001za*Iz,sb000za+sb010za*Iy+...
269         sb001za*Iz,sc000za+sc001za*Iz]*[Ix-za1;Iy-za2;Iz-za3];
270     I3=fzb+[sa000zb+sa100zb*Ix+sa010zb*Iy+sa001zb*Iz,sb000zb+sb010zb*Iy+...
271         sb001zb*Iz,sc000zb+sc001zb*Iz]*[Ix-zb1;Iy-zb2;Iz-zb3];
272
273
274     I4=fza+(fzazb+[Ix-zb1;Iy-zb2;Iz-zb3]'*s2za)*[Ix-za1;Iy-za2;Iz-za3];
275     I5=intersect(intersect(I1,I2),intersect(I3,I4));
276
277 end;
```

```
278
279 u=inf_(I5);
280 o=sup(I5);
```

A.25 slope3.m

```
1  function [sa000,sa100,sa010,sa001,sb000,sb010,sb001,sc000,sc001,s2] = ...
2      slope3(pa000,pa100,pa010,pa001,pa110,pa101,pa011,pa200,pa020,pa002,...
3             qa100,qa010,qa001,qa110,qa101,qa011,qa200,qa020,qa002,Ia,...
4             pb000,pb100,pb010,pb001,pb110,pb101,pb011,pb200,pb020,pb002,...
5             qb100,qb010,qb001,qb110,qb101,qb011,qb200,qb020,qb002,Ib,...
6             p000,p100,p010,p001,p110,p101,p011,p200,p020,p002,q100,q010,...
7             q001,q110,q101,q011,q200,q020,q002,operation,...
8             za1,za2,za3,zb1,zb2,zb3,Ixu,Ixo,Iyu,Iyo,Izu,Izo)
9
10 %Definitionsbereich
11 Ix=infsup(Ixu,Ixo);
12 Iy=infsup(Iyu,Iyo);
13 Iz=infsup(Izu,Izo);
14
15 %Berechnung von Slopes erster Ordnung der verschiedenen Polynome
16 [pazaa000,pazaa100,pazaa010,pazaa001,pazab000,...
17  pazab010,pazab001,pazac000,pazac001,paza2]=...
18                 slopepoly3(pa000,pa100,pa010,pa001,pa110,pa011,pa101,...
19                            pa200,pa020,pa002,za1,za2,za3);
20 [qazaa000,qazaa100,qazaa010,qazaa001,qazab000,...
21  qazab010,qazab001,qazac000,qazac001,qaza2]=...
22                 slopepoly3(1,qa100,qa010,qa001,qa110,qa011,qa101,...
23                            qa200,qa020,qa002,za1,za2,za3);
24 [qazba000,qazba100,qazba010,qazba001,qazbb000,...
25  qazbb010,qazbb001,qazbc000,qazbc001,qazb2]=...
26                 slopepoly3(1,qa100,qa010,qa001,qa110,qa011,qa101,...
27                            qa200,qa020,qa002,zb1,zb2,zb3);
28 [pbzaa000,pbzaa100,pbzaa010,pbzaa001,pbzab000,...
29  pbzab010,pbzab001,pbzac000,pbzac001,pbza2]=...
30                 slopepoly3(pb000,pb100,pb010,pb001,pb110,pb011,pb101,...
31                            pb200,pb020,pb002,za1,za2,za3);
32 [pbzba000,pbzba100,pbzba010,pbzba001,pbzbb000,...
33  pbzbb010,pbzbb001,pbzbc000,pbzbc001,pbzb2]=...
34                 slopepoly3(pb000,pb100,pb010,pb001,pb110,pb011,pb101,...
35                            pb200,pb020,pb002,zb1,zb2,zb3);
36 [qbzaa000,qbzaa100,qbzaa010,qbzaa001,qbzab000,...
37  qbzab010,qbzab001,qbzac000,qbzac001,qbza2]=...
38                 slopepoly3(1,qb100,qb010,qb001,qb110,qb011,qb101,...
39                            qb200,qb020,qb002,za1,za2,za3);
40 [qbzba000,qbzba100,qbzba010,qbzba001,qbzbb000,...
41  qbzbb010,qbzbb001,qbzbc000,qbzbc001,qbzb2]=...
42                 slopepoly3(1,qb100,qb010,qb001,qb110,qb011,qb101,...
43                             qb200,qb020,qb002,zb1,zb2,zb3);
44 [pzaa000,pzaa100,pzaa010,pzaa001,pzab000,pzab010,...
45  pzab001,pzac000,pzac001,pza2]=...
```

```
                slopepoly3(p000,p100,p010,p001,p110,p011,p101,p200,...
                           p020,p002,za1,za2,za3);
[qzaa000,qzaa100,qzaa010,qzaa001,qzab000,qzab010,...
 qzab001,qzac000,qzac001,qza2]=...
                slopepoly3(1,q100,q010,q001,q110,q011,q101,q200,q020,...
                           q002,za1,za2,za3);
[qzba000,qzba100,qzba010,qzba001,qzbb000,qzbb010,...
 qzbb001,qzbc000,qzbc001,qzb2]=...
                slopepoly3(1,q100,q010,q001,q110,q011,q101,q200,...
                           q020,q002,zb1,zb2,zb3);


ha=(pa000+pa100*za1+pa010*za2+pa001*za3+pa110*za1*za2+pa011*za2*za3+...
    pa101*za1*za3+pa200*za1^2+pa020*za2^2+pa002*za3^2+Ia)/...
    (1+qa100*za1+qa010*za2+qa001*za3+qa110*za1*za2+qa011*za2*za3+...
    qa101*za1*za3+qa200*za1^2+qa020*za2^2+qa002*za3^2);
hb=(pb000+pb100*za1+pb010*za2+pb001*za3+pb110*za1*za2+pb011*za2*za3+...
    pb101*za1*za3+pb200*za1^2+pb020*za2^2+pb002*za3^2+Ib)/...
    (1+qb100*za1+qb010*za2+qb001*za3+qb110*za1*za2+qb011*za2*za3+...
    qb101*za1*za3+qb200*za1^2+qb020*za2^2+qb002*za3^2);

%Berechnen der Koeffizienten des Slopes erster Ordnung
%von pa/qa an der Stelle za, anhand bekannter Rechenregeln für Slopes
paqas1=([pazaa000,pazaa100,pazaa010,pazaa001,pazab000,pazab010,pazab001,...
         pazac000,pazac001]-(ha)*...
        [qazaa000,qazaa100,qazaa010,qazaa001,qazab000,qazab010,qazab001,...
         qazac000,qazac001])/...
       (1+qa100*Ix+qa010*Iy+qa001*Iz+qa110*Ix*Iy+qa011*Iy*Iz+qa101*Ix*Iz+...
        qa200*power(Ix,2)+qa020*power(Iy,2)+qa002*power(Iz,2));

%Berechnen der Koeffizienten des Slopes erster Ordnung
%von pb/qb an der Stelle za, anhand bekannter Rechenregeln für Slopes
pbqbs1=([pbzaa000,pbzaa100,pbzaa010,pbzaa001,pbzab000,pbzab010,pbzab001,...
         pbzac000,pbzac001]-(hb)*...
        [qbzaa000,qbzaa100,qbzaa010,qbzaa001,qbzab000,qbzab010,qbzab001,...
         qbzac000,qbzac001])/...
       (1+qb100*Ix+qb010*Iy+qb001*Iz+qb110*Ix*Iy+qb011*Iy*Iz+qb101*Ix*Iz+...
        qb200*power(Ix,2)+qb020*power(Iy,2)+qb002*power(Iz,2));

%Berechnen der Koeffizienten des Slopes erster Ordnung
%von pb/qb an der Stelle zb, anhand bekannter Rechenregeln für Slopes
pbqbzbs1=([pbzba000,pbzba100,pbzba010,pbzba001,pbzbb000,pbzbb010,pbzbb001,...
           pbzbc000,pbzbc001]-(hb)*...
          [qbzba000,qbzba100,qbzba010,qbzba001,qbzbb000,qbzbb010,qbzbb001,...
           qbzbc000,qbzbc001])/...
         (1+qb100*Ix+qb010*Iy+qb001*Iz+qb110*Ix*Iy+qb011*Iy*Iz+qb101*Ix*Iz+...
          qb200*power(Ix,2)+qb020*power(Iy,2)+qb002*power(Iz,2));


hbzb=(pb000+pb100*zb1+pb010*zb2+pb001*zb3+pb110*zb1*zb2+pb011*zb2*zb3+...
      pb101*zb1*zb3+pb200*zb1^2+pb020*zb2^2+pb002*zb3^2+Ib)/...
     (1+qb100*zb1+qb010*zb2+qb001*zb3+qb110*zb1*zb2+qb011*zb2*zb3+...
      qb101*zb1*zb3+qb200*zb1^2+qb020*zb2^2+qb002*zb3^2);

```

```
%Berechne die Koeffizienten des Slopes zweiter Ordnung
%von pa/qa[za,zb,x]
paqa2=(paza2-(ha)*qaza2+([qazba000+qazba100*Ix+qazba010*Iy+qazba001*Iz,...
       qazbb000+qazbb010*Iy+qazbb001*Iz,qazbc000+qazbc001*Iz]')*...
      ([qazaa000+qazaa100*zb1+qazaa010*zb2+qazaa001*zb3,...
        qazab000+qazab010*zb2+qazab001*zb3,qazac000+qazac001*zb3]*ha-...
      [pazaa000+pazaa100*zb1+pazaa010*zb2+pazaa001*zb3,...
       pazab000+pazab010*zb2+pazab001*zb3,pazac000+pazac001*zb3])/...
      (1+qa100*zb1+qa010*zb2+qa001*zb3+qa110*zb1*zb2+qa011*zb2*zb3+...
       qa101*zb1*zb3+qa200*zb1^2+qa020*zb2^2+qa002*zb3^2))/...
      (1+qa100*Ix+qa010*Iy+qa001*Iz+qa110*Ix*Iy+qa011*Iy*Iz+qa101*Ix*Iz+...
       qa200*power(Ix,2)+qa020*power(Iy,2)+qa002*power(Iz,2));

%Berechne die Koeffizienten des Slopes zweiter Ordnung
%von pb/qb[za,zb,x]
pbqb2=(pbza2-(hb)*qbza2+([qbzba000+qbzba100*Ix+qbzba010*Iy+qbzba001*Iz,...
       qbzbb000+qbzbb010*Iy+qbzbb001*Iz,qbzbc000+qbzbc001*Iz]')*...
      ([qbzaa000+qbzaa100*zb1+qbzaa010*zb2+qbzaa001*zb3,qbzab000+...
        qbzab010*zb2+qbzab001*zb3,qbzac000+qbzac001*zb3]*hb-...
       [pbzaa000+pbzaa100*zb1+pbzaa010*zb2+pbzaa001*zb3,pbzab000+...
        pbzab010*zb2+pbzab001*zb3,pbzac000+pbzac001*zb3])/...
      (1+qb100*zb1+qb010*zb2+qb001*zb3+qb110*zb1*zb2+qb011*zb2*zb3+...
       qb101*zb1*zb3+qb200*zb1^2+qb020*zb2^2+qb002*zb3^2))/...
      (1+qb100*Ix+qb010*Iy+qb001*Iz+qb110*Ix*Iy+qb011*Iy*Iz+...
       qb101*Ix*Iz+qb200*power(Ix,2)+qb020*power(Iy,2)+qb002*power(Iz,2));


%Es folgen die Berechnungen der Slopes erster und zweiter Ordnung
%für die verschiedenen Operationen anhand den bekannten Rechenregeln
%für Slopes:
if(operation==1)
    a=paqas1+pbqbs1;
    c=a*(1+q100*Ix+q010*Iy+q001*Iz+q110*Ix*Iy+q011*Iy*Iz+q101*Ix*Iz+...
         q200*power(Ix,2)+q020*power(Iy,2)+q002*power(Iz,2))+(ha+hb)*...
      [qzaa000,qzaa100,qzaa010,qzaa001,qzab000,...
       qzab010,qzab001,qzac000,qzac001];
    c=c-[pzaa000,pzaa100,pzaa010,pzaa001,pzab000,...
         pzab010,pzab001,pzac000,pzac001];

    sa000=c(1);
    sa100=c(2);
    sa010=c(3);
    sa001=c(4);
    sb000=c(5);
    sb010=c(6);
    sb001=c(7);
    sc000=c(8);
    sc001=c(9);


    E=paqa2+pbqb2;
    E=E*(1+q100*Ix+q010*Iy+q001*Iz+q110*Ix*Iy+q011*Iy*Iz+q101*Ix*Iz+...
         q200*power(Ix,2)+q020*power(Iy,2)+q002*power(Iz,2))+...
        (ha+hb)*qza2+[a(1)+a(2)*zb1+a(3)*zb2+a(4)*zb3,...
```

```
                      a(5)+a(6)*zb2+a(7)*zb3,a(8)+a(9)*zb3]'*...
        [qzba000+qzba100*Ix+qzba010*Iy+qzba001*Iz,...
         qzbb000+qzbb010*Iy+qzbb001*Iz,qzbc000+qzbc001*Iz];

    E=E-pza2;
    s2=E;


elseif(operation==2)

    a=paqas1-pbqbs1;
    c=a*(1+q100*Ix+q010*Iy+q001*Iz+q110*Ix*Iy+q011*Iy*Iz+q101*Ix*Iz+...
         q200*power(Ix,2)+q020*power(Iy,2)+q002*power(Iz,2))+...
        (ha-hb)*[qzaa000,qzaa100,qzaa010,qzaa001,...
                 qzab000,qzab010,qzab001,qzac000,qzac001];
    c=c-[pzaa000,pzaa100,pzaa010,pzaa001,pzab000,...
         pzab010,pzab001,pzac000,pzac001];

    sa000=c(1);
    sa100=c(2);
    sa010=c(3);
    sa001=c(4);
    sb000=c(5);
    sb010=c(6);
    sb001=c(7);
    sc000=c(8);
    sc001=c(9);

    E=paqa2-pbqb2;
    E=E*(1+q100*Ix+q010*Iy+q001*Iz+q110*Ix*Iy+q011*Iy*Iz+q101*Ix*Iz+...
         q200*power(Ix,2)+q020*power(Iy,2)+q002*power(Iz,2))+...
        (ha-hb)*qza2+[a(1)+a(2)*zb1+a(3)*zb2+a(4)*zb3,...
                      a(5)+a(6)*zb2+a(7)*zb3,a(8)+a(9)*zb3]'*...
        [qzba000+qzba100*Ix+qzba010*Iy+qzba001*Iz,...
         qzbb000+qzbb010*Iy+qzbb001*Iz,qzbc000+qzbc001*Iz];

    E=E-pza2;
    s2=E;

elseif(operation==3)

    a=paqas1*(pb000+pb100*Ix+pb010*Iy+pb001*Iz+pb110*Ix*Iy+pb011*Iy*Iz+...
      pb101*Ix*Iz+pb200*power(Ix,2)+pb020*power(Iy,2)+pb002*power(Iz,2)+Ib)/...
      (1+qb100*Ix+qb010*Iy+qb001*Iz+qb110*Ix*Iy+qb011*Iy*Iz+...
      qb101*Ix*Iz+qb200*power(Ix,2)+qb020*power(Iy,2)+qb002*power(Iz,2))+...
      ha*pbqbs1;
    c=a*(1+q100*Ix+q010*Iy+q001*Iz+q110*Ix*Iy+q011*Iy*Iz+q101*Ix*Iz+...
      q200*power(Ix,2)+q020*power(Iy,2)+q002*power(Iz,2))+...
      (ha*hb)*[qzaa000,qzaa100,qzaa010,qzaa001,qzab000,...
               qzab010,qzab001,qzac000,qzac001];

    c=c-[pzaa000,pzaa100,pzaa010,pzaa001,...
         pzab000,pzab010,pzab001,pzac000,pzac001];
```

```
208      sa000=c(1);
209      sa100=c(2);
210      sa010=c(3);
211      sa001=c(4);
212      sb000=c(5);
213      sb010=c(6);
214      sb001=c(7);
215      sc000=c(8);
216      sc001=c(9);
217
218      E=paqa2*(pb000+pb100*Ix+pb010*Iy+pb001*Iz+pb110*Ix*Iy+pb011*Iy*Iz+...
219        pb101*Ix*Iz+pb200*power(Ix,2)+pb020*power(Iy,2)+pb002*power(Iz,2)+Ib)/...
220        (1+qb100*Ix+qb010*Iy+qb001*Iz+qb110*Ix*Iy+qb011*Iy*Iz+qb101*Ix*Iz+...
221        qb200*power(Ix,2)+qb020*power(Iy,2)+qb002*power(Iz,2))+...
222        pbqb2*ha+[paqas1(1)+paqas1(2)*zb1+paqas1(3)*zb2+paqas1(4)*zb3,...
223        paqas1(5)+paqas1(6)*zb2+paqas1(7)*zb3,paqas1(8)+paqas1(9)*zb3]'*...
224        [pbqbs1(1)+pbqbs1(2)*Ix+pbqbs1(3)*Iy+pbqbs1(4)*Iz,pbqbs1(5)+...
225        pbqbs1(6)*Iy+pbqbs1(7)*Iz,pbqbs1(8)+pbqbs1(9)*Iz];
226
227      E=E*(1+q100*Ix+q010*Iy+q001*Iz+q110*Ix*Iy+q011*Iy*Iz+q101*Ix*Iz+...
228        q200*power(Ix,2)+q020*power(Iy,2)+q002*power(Iz,2))+...
229        (ha-hb)*qza2+[a(1)+a(2)*zb1+a(3)*zb2+a(4)*zb3,a(5)+a(6)*zb2+a(7)*zb3,...
230        a(8)+a(9)*zb3]'*[qzba000+qzba100*Ix+qzba010*Iy+qzba001*Iz,...
231        qzbb000+qzbb010*Iy+qzbb001*Iz,qzbc000+qzbc001*Iz];
232
233      E=E-pza2;
234      s2=E;
235
236
237  elseif(operation==4)
238
239      a=(paqas1-(ha/hb)*pbqbs1)/((pb000+pb100*Ix+pb010*Iy+pb001*Iz+...
240         pb110*Ix*Iy+pb011*Iy*Iz+pb101*Ix*Iz+pb200*power(Ix,2)+...
241         pb020*power(Iy,2)+pb002*power(Iz,2)+Ib)/...
242        (1+qb100*Ix+qb010*Iy+qb001*Iz+qb110*Ix*Iy+qb011*Iy*Iz+qb101*Ix*Iz+...
243         qb200*power(Ix,2)+qb020*power(Iy,2)+qb002*power(Iz,2)));
244      c=a*(1+q100*Ix+q010*Iy+q001*Iz+q110*Ix*Iy+q011*Iy*Iz+q101*Ix*Iz+...
245        q200*power(Ix,2)+q020*power(Iy,2)+q002*power(Iz,2))+...
246        (ha*hb)*[qzaa000,qzaa100,qzaa010,qzaa001,qzab000,qzab010,...
247         qzab001,qzac000,qzac001];
248      c=c-[pzaa000,pzaa100,pzaa010,pzaa001,pzab000,...
249           pzab010,pzab001,pzac000,pzac001];
250
251      sa000=c(1);
252      sa100=c(2);
253      sa010=c(3);
254      sa001=c(4);
255      sb000=c(5);
256      sb010=c(6);
257      sb001=c(7);
258      sc000=c(8);
259      sc001=c(9);
260
261      E=(paqa2-(ha/hb)*pbqb2+[pbqbzbs1(1)+pbqbzbs1(2)*Ix+pbqbzbs1(3)*Iy+...
```

```
        pbqbzbs1(4)*Iz,pbqbzbs1(5)+pbqbzbs1(6)*Iy+pbqbzbs1(7)*Iz,pbqbzbs1(8)+...
        pbqbzbs1(9)*Iz]'*([pbqbs1(1)+pbqbs1(2)*zb1+pbqbs1(3)*zb2+...
        pbqbs1(4)*zb3,pbqbs1(5)+pbqbs1(6)*zb2+pbqbs1(7)*zb3,pbqbs1(8)+...
        pbqbs1(9)*zb3]*(ha/hb)-[paqas1(1)+paqas1(2)*zb1+paqas1(3)*zb2+...
        paqas1(4)*zb3,paqas1(5)+paqas1(6)*zb2+paqas1(7)*zb3,paqas1(8)+...
        paqas1(9)*zb3])/(hbzb))/ ((pb000+pb100*Ix+pb010*Iy+pb001*Iz+...
        pb110*Ix*Iy+pb011*Iy*Iz+pb101*Ix*Iz+pb200*power(Ix,2)+...
        pb020*power(Iy,2)+pb002*power(Iz,2)+Ib)/...
       (1+qb100*Ix+qb010*Iy+qb001*Iz+qb110*Ix*Iy+qb011*Iy*Iz+qb101*Ix*Iz+...
        qb200*power(Ix,2)+qb020*power(Iy,2)+qb002*power(Iz,2)));

    E=E*(1+q100*Ix+q010*Iy+q001*Iz+q110*Ix*Iy+q011*Iy*Iz+q101*Ix*Iz+...
         q200*power(Ix,2)+q020*power(Iy,2)+q002*power(Iz,2))+...
        (ha-hb)*qza2+[a(1)+a(2)*zb1+a(3)*zb2+a(4)*zb3,a(5)+a(6)*zb2+...
         a(7)*zb3,a(8)+a(9)*zb3]'*[qzba000+qzba100*Ix+qzba010*Iy+...
         qzba001*Iz,qzbb000+qzbb010*Iy+qzbb001*Iz,qzbc000+qzbc001*Iz];

    E=E-pza2;
    s2=E;

end;
```

A.26 slopepoly3.m

```
function [sa000,sa100,sa010,sa001,sb000,sb010,sb001,sc000,sc001,s2]=...
    slopepoly3(e000,e100,e010,e001,e110,e011,e101,e200,e020,e002,z1,z2,z3)

sa000=e100+e200*z1;
sa100=e200;
sa010=e110;
sa001=e101;

sb000=e010+e020*z2+e110*z1;
sb010=e020;
sb001=e011;

sc000=e001+e011*z2+e101*z1+e002*z3;
sc001=e002;

s2=zeros(3);
s2(1,1)=e200;
s2(2,1)=e110;
s2(2,2)=e020;
s2(3,1)=e101;
s2(3,2)=e011;
s2(3,3)=e002;
```

Literatur

[1] Martin Berz and Georg Hofstätter. *Computation and Application of Taylor Polynomials with Interval Remainder Bounds.* Reliable Computing 4, 1998.

[2] Martin Berz and Kyoko Makino. *Remainder Diffenetial Algebras and Their Applications.* In: Berz, M., C. Bischof, G. Corliss und A. Griewank (Hg.), Computational Differentiation: Techniques, Application, and Tools. SIAM, 1996.

[3] Martin Berz and Kyoko Makino. *Taylor Models and other validated functional inclusion methods.* International Journal of Pure and Applied Mathematics, 2003.

[4] Ingo Eble. *Über Taylor-Modelle.* Diplomarbeit, Karlsruhe, 2007.

[5] Jr. George A. Baker. *Essentials of Padé Approximants.* Academic Press, Inc., 1975.

[6] Jr. George A. Baker and Peter Graves-Morris. *Padé Approximants Part I: Basic Theory.* Cambridge University Press, 1984.

[7] Jr. George A. Baker and Peter Graves-Morris. *Padé Approximants Part II: Extensions and Applications.* Cambridge University Press, 1984.

[8] Michael Grant and Stephen Boyd. *CVX: Matlab software for disciplined convex programming (web page and software).* http://stanford.edu/ boyd/cvx, 2009.

[9] Eldon Hansen. *Global Optimization using Interval Analysis.* Marcel Dekker, Inc., 1992.

[10] Werner Jank. *Grundlagen einer Intervallarithmetik.* Mathematisches Institut Graz, Bericht Nr. 68-1, 1968.

[11] Siegfried M. Rump. *Intlab-interval laboratory.* www.ti3.tuharburg.de/rump/intlab/.

[12] Marco Schnurr. *The Automatic Computation of Second-Order Slope Tuples for Some Nonsmooth Functions.* Electronic Transactions on Numerical Analysis 30, 2008.

[13] Moritz Wurnig. *Semilineare Relaxationen.* Diplomarbeit, Wien, 2007.

B Lebenslauf

Name	Thomas Kammerhuber
Geburtsdatum	25. März 1985
Geburtsort	Linz
Schulbildung	
1991-1995	Volksschule Wartberg ob der Aist
1995-1999	Hauptschule 1 Pregarten
1999-2004	HTBLA Linz 2 - Linzer Technikum, Abteilung Elektrotechnik
Zivildienst	
2004-2005	Diakoniewerk, Werkstätte Wartberg ob der Aist
Studium	
2005-2006	Technische Mathematik, Technische Universität Wien
2006-2010	Diplomstudium Mathematik, Universität Wien