Über Taylor-Modelle

Zur Erlangung des akademischen
Grades eines

Doktors Der
Naturwissenschaften

von der Fakultät für Mathematik der
Universität Karlsruhe
genehmigte

Dissertation

von

Dipl.-Math. Ingo Eble
aus Zell am Harmersbach

Tag der mündlichen Prüfung: 29. Januar 2007
Referent: Prof. Dr. G. Alefeld
Korreferent: apl. Prof. Dr. R. Lohner

Die vorliegende Arbeit wurde mit LaTeX und KOMA-Script erstellt.

Danksagung

Mein besonderer Dank gebührt Herrn Prof. Dr. G. Alefeld, der durch seine fortwährende Unterstützung die Fertigstellung dieser Arbeit erst ermöglicht hat.

Außerdem danke ich Herrn apl. Prof. Dr. R. Lohner für die freundliche Übernahme des Korreferats.

Ganz herzlich möchte ich mich auch bei Herrn Dr. M. Neher für seine wertvollen Anregungen und Ratschläge sowie seine unermüdliche Bereitschaft zur Diskussion bedanken.

Zu guter Letzt, aber nicht weniger herzlich, möchte ich auch noch all den Menschen danken, die mich in den Jahren der Entstehung dieser Arbeit auf meinem Weg begleitet und unterstützt haben.

Was immer Du auch tun kannst
oder erträumst zu können,
beginne es.
Kühnheit besitzt Genie,
Macht und magische Kraft.
Beginne es jetzt.

Johann Wolfgang Goethe (1747-1842)

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

Liste der Algorithmen

Einleitung

In den Naturwissenschaften werden Merkmale und deren Wechselbeziehung zu anderen Merkmalen unter anderem über mathematische Modelle studiert. Die Modelle entstehen aus einer vereinfachten mathematischen Beschreibung des tatsächlichen Sachverhaltes. Die Vereinfachungen sind dabei Annahmen über die Rahmenbedingungen sowie Annahmen über die betrachteten Merkmale und deren Wechselwirkung selbst. Häufig gelangt man bei der Modellbildung auf nichtlineare gewöhnliche Differentialgleichungen, für die nur in wenigen Spezialfällen die Möglichkeit der analytischen Lösung besteht. In aller Regel wird man deren Lösung nur numerisch gewinnen können. Dazu werden in erster Linie Näherungsverfahren herangezogen, die aber etliche Nachteile mit sich bringen.

Sofern durchführbar liefern Näherungsverfahren - wie der Name schon sagt - nur eine Näherungslösung, eventuell auch dann, wenn überhaupt keine Lösung existiert. Auch können in aller Regel keine Aussagen über den Fehler zur exakten Lösung gemacht werden. Bei einem unerwarteten Resultat ist damit nicht klar, ob die Ursachen im Modell stecken, d. h. ob die Vereinfachungen zu stark waren, oder ob die Ursache in dem gewählten numerischen Verfahren und der Berechnung auf dem Computer zu suchen ist.

Des Weiteren kommt hinzu, dass in der Praxis gegebene Anfangswerte meist nicht exakt vorliegen, sondern mit Messfehlern behaftet sind. Streng genommen ist demnach eine Menge von Anfangswertproblemen zu lösen. Abhilfe in allen genannten Punkten schaffen hier die Intervall-Methoden. Sie liefern im Falle der Durchführbarkeit neben verifizierten Schranken auch den Beweis über die Existenz einer Lösung und erlauben ohne zusätzlichen Aufwand, eine Menge von Anfangswertproblemen simultan zu lösen.

Den Grundstein für die Entwicklung von Intervall-Methoden zur verifizierten Lösung von Anfangswertproblemen legte Moore [47, 48] mit seinen Arbeiten aus den 1960er Jahren. Es folgten etliche Weiterentwicklungen, von denen wir besonders Krückeberg [34], sowie die Dissertationen von Eijgenraam [16] und Lohner [41] erwähnen wollen. In diesen Arbeiten werden Taylorentwicklungen eindimensionaler Funktionen mit Fehlerintervallen zur Lösungseinschließung verwendet. Varianten davon, die auf Tschebyscheff- und Bernsteinentwicklungen basieren, wurden um 1980 unter anderem von Kaucher und Miranker [30] vorgeschlagen und sind unter der Bezeichnung Ultra-Arithmetik oder Funktoid bekannt geworden.

Speziell die Taylorentwicklungen besitzen einige Vorteile gegenüber anderen Funktionenbasen. Der Fehlerterm ist von einfacher Gestalt, die Taylorkoeffizienten

können leicht berechnet werden und sind unabhängig von der gewählten Schrittweite. Die Ordnung der Entwicklung ist ohne großen Mehraufwand veränderbar, denn überschüssige Terme können einfach weggelassen und fehlende Terme einfach nachberechnet werden. Die Berechnung der Koeffizienten wird durch automatische Differentiation [19, 47, 48, 53] relativ einfach möglich.

Bei den herkömmlichen Intervall-Methoden zur verifizierten Lösung von Anfangswertproblemen unterteilt sich der Prozess der Lösungseinschließung üblicherweise in zwei Phasen. In der ersten Phase wird zu gegebener Schrittweite eine Grobeinschließung bestimmt. Dies geschieht auf der Grundlage des Banachschen Fixpunktsatzes, sodass im Erfolgsfall die Eindeutigkeit der Lösung auf dem Teilintervall gleich mit sichergestellt ist. In der zweiten Phase erfolgt die Berechnung einer engen Lösungseinschließung. Hier hat man mit dem sogenannten Wrapping-Effekt zu kämpfen, der entsteht, wenn die Lösungsmenge in eine für den nachfolgenden Integrationsschritt verwertbare Menge eingepackt werden muss. Dabei entsteht meist eine Überschätzung der tatsächlichen Lösungsmenge, was sich negativ auf die Integration auswirkt und die Lösungschranken verschlechtert. Moore [47, 48] rückt dem Wrapping-Effekt durch ein bewegtes Koordinatensystem zu Leibe, Krückeberg [34] greift Moores Idee auf und konzipiert lokal die Einschließung durch Parallelepipede, Lohner verwendet lokal ein orthogonales Koordinatensystem, das bei schlecht konditionierten Matrizen sehr gute Dienste leistet. Nicht unerwähnt lassen wollen wir die Zonotope von Kühn [37]. Alle genannten Methoden schließen die Lösungsmenge in konvexe Mengen ein, was aber spätestens bei nicht-konvexen Lösungsmengen, wie sie bei nichtlinearen Differentialgleichungen auftreten können, zu deutlichen Überschätzungen führt.

Die Taylor-Modelle von Berz und seinen Mitarbeitern [5, 9, 10, 28, 29, 43, 44, 45, 46] gehen hier einen Schritt weiter und zeigen speziell bei nichtlinearen Differentialgleichungen Vorteile gegenüber den bisherigen Verfahren. Zwar arbeitet auch die Methode von Makino und Berz mit Taylorentwicklung und Fehlerterm, der entscheidende Unterschied ist jedoch, dass die Lösung als multivariate Taylorentwicklung der Anfangswerte dargestellt wird. Die Lösungseinschließung geschieht wie üblich über den Fehlerterm. Verglichen mit dem (üblichen) Einschließungsprozess bei herkömmlichen Verfahren wird hier in einer statt in zwei Phasen eine Lösungeinschließung berechnet, wobei wie oben die Existenz der Lösung über das theoretische Fundament eines Fixpunktsatzes gleichermaßen abgesichert wird. Die multivariate Entwicklung der Lösung schafft eine nichtlineare Verbindung der Lösungsmenge mit den Anfangswerten und ermöglicht so eine formgerechte Darstellung der Lösungsmenge in jedem Schritt. Neben konvexen Einschließungen sind auch nicht-konvexe Mengen zur Einschließung möglich, wodurch die Wirkung des Wrapping-Effekts erheblich reduziert wird.

Seit der ersten Veröffentlichung über Taylor-Modelle 1996 gibt es immer wieder kontroverse Diskussionen (vgl. Neumaier [51]) über deren Anwendungsfelder und

über die jeweiligen Aussagen zu deren Qualität und Nutzen. Das liegt mitunter an fehlenden, aber doch wichtigen Details über die Taylor-Modelle an sich, sowie einem Mangel an Vergleichsrechnungen mit anderen Methoden der Intervallrechnung. Eine Implementierung der Taylor-Modelle sowie auch der Methode zur Lösungseinschließung bei gewöhnlichen Differentialgleichungen mit Taylor-Modellen existiert und ist gegen Registrierung erhältlich. Die Weitergabe der Software an andere Personen wie auch Veränderungen am Code selbst sind nur mit Zustimmung der Urheber erlaubt.

Diese Arbeit entstand aus dem Wunsch heraus, die Theorie der Taylor-Modelle in mathematisch vollständiger Form darzustellen, sowie deren Verwendung bei gewöhnlichen Differentialgleichungen im Detail verständlich zu machen. Hierbei haben wir im Vergleich zur Darstellung in der bisherigen Literatur an einigen Stellen Veränderungen vorgenommen. Wir beginnen in Kapitel 1 mit den für diese Arbeit relevanten Aussagen der Intervallrechnung. Anschließend beschreiben wir in Kapitel 2 das Konzept der Taylor-Modelle. Gegenüber den bisherigen Veröffentlichungen über Taylor-Modelle führen wir dort den Begriff Taylor-Modell als Funktionenmenge ein und schaffen damit gewissermaßen eine Verallgemeinerung des Intervallbegriffs auf Funktionenräume. Außerdem nehmen wir eine leicht abgewandelte Definition der Multiplikation und Integration von Taylor-Modellen vor, die im Vergleich zu den früheren Definitionen kleinere Ergebnismengen zur Folge hat. In Kapitel 3 folgt dann die Verwendung von Taylor-Modellen zur Lösungseinschließung bei gewöhnlichen Anfangswertproblemen. Anders als in der bisherigen Literatur wenden wir dort den Banachschen Fixpunktsatz zur Lösungseinschließung an. Dies wird durch eine Korrektur der in [5, 10, 42] gemachten Voraussetzungen für das Einschließungsverfahren von Makino und Berz möglich. Es gelingt uns außerdem noch, einige Detailverbesserungen an deren Einschließungsalgorithmus inklusive der von Makino und Berz erdachten Methode des Shrink-Wrapping anzubringen. Dieser modifizierte Algorithmus wurde im Rahmen dieser Arbeit in einer Software realisiert, die jedem Interessierten frei zugänglich ist und nach Belieben weiterentwickelt und verändert werden darf. In Kapitel 4 wird auf die Implementierung dieser Software näher eingegangen, dabei werden auch Konzepte der objektorientierten Softwareentwicklung vorgestellt und erläutert. Den Abschluss bildet das Kapitel 5 mit einem praktischen Vergleich der implementierten Methode zu den Computerprogrammen AWA und COSY-VI.

Die im Rahmen dieser Arbeit entstandene Software ist online erhältlich unter der Internetadresse

`http://iamlasun8.mathematik.uni-karlsruhe.de/~ae08/`

oder über eine e-Mail an `ingoeble@web.de`.

1 Intervallrechnung

Die höchsten Türme fangen beim Fundament an.

Thomas Alva Edison (1847-1931)

In diesem Kapitel werden die Grundlagen der Intervallrechnung bereitgestellt. Für eine ausführliche Behandlung der Thematik verweisen wir auf Alefeld und Herzberger [1], Moore [47, 48] oder Neumaier [50].

Ein Intervall ist definiert als kompakte und zusammenhängende Teilmenge von $\mathbb{R}$. Wir schreiben dafür

$$I = [\underline{x}, \overline{x}] := \{x \in \mathbb{R} \,|\, \underline{x} \leqslant x \leqslant \overline{x}\}.$$

Für die Schranken $\underline{x}$, $\overline{x}$ gilt

$$\underline{x} = \inf(I),$$
$$\overline{x} = \sup(I).$$

Sind $\underline{x}$ und $\overline{x}$ gleich, so spricht man von einem Punktintervall. Ein solches kann mit der reellen Zahl $x = \underline{x} = \overline{x}$ identifiziert werden. Umgekehrt kann eine reelle Zahl als Punktintervall aufgefasst werden. Die Menge aller Intervalle wird mit $\mathbb{IR}$ bezeichnet.

Unter Intervallvektoren und -matrizen versteht man Vektoren und Matrizen mit Intervallen als Komponenten. Die entsprechenden Mengen werden mit $\mathbb{IR}^{\nu}$ bzw. $\mathbb{IR}^{\mu\times\nu}$ bezeichnet.

Relationen wie Gleichheit und Inklusion sowie die Schnittbildung sind in $\mathbb{IR}$ im mengentheoretischen Sinne zu verstehen [1, S. 1]. Es gilt für $I_1 = [\underline{x}_1, \overline{x}_1]$, $I_2 = [\underline{x}_2, \overline{x}_2]$

$$I_1 = I_2 \;\Leftrightarrow\; \underline{x}_1 = \underline{x}_2 \text{ und } \overline{x}_1 = \overline{x}_2, \tag{1.1}$$

$$I_1 \subseteq I_2 \;\Leftrightarrow\; \underline{x}_2 \leqslant \underline{x}_1 \text{ und } \overline{x}_1 \leqslant \overline{x}_2, \tag{1.2}$$

$$I_1 \subset I_2 \;\Leftrightarrow\; I_1 \subseteq I_2 \text{ und } I_1 \neq I_2. \tag{1.3}$$

Für Intervallvektoren und -matrizen werden Gleichheit, Inklusion und Schnittbildung über die Komponenten definiert. So sind beispielsweise zwei Intervallvektoren gleich, wenn alle Komponentenintervalle gleich sind.

Die arithmetischen Operationen $\{+, -, \cdot, /\}$ werden auf $\mathbb{IR}$ wie folgt definiert [1, S. 2]:

Definition 1 (Grundarithmetik)
Sind I_1, I_2 zwei Intervalle aus $\mathbb{IR}$ und $\circ$ eine Verknüpfung aus $\{+, -, \cdot, /\}$, dann ist

$$I_1 \circ I_2 := \{x_1 \circ x_2 \,|\, x_1 \in I_1,\, x_2 \in I_2\}\,.$$

Im Falle der Division wird $0 \notin I_2$ vorausgesetzt.

Aufgrund der Stetigkeit der Operationen ist die Ergebnismenge wieder ein Intervall. Jedoch ist die Definition für die praktische Berechnung des Intervalls unbrauchbar, da unendlich viele Operationen durchzuführen sind. Eine Darstellung des Ergebnisintervalls durch Verknüpfungen in den Intervallschranken liefert [vgl. 1, S. 2]

Satz 1
Für I_1, $I_2 \in \mathbb{IR}$ gilt

$$I_1 + I_2 = [\underline{x}_1 + \underline{x}_2, \overline{x}_1 + \overline{x}_2]\,, \quad I_1 - I_2 = [\underline{x}_1 - \overline{x}_2, \overline{x}_1 - \underline{x}_2]\,,$$
$$I_1 \cdot I_2 = \big[\min\{\underline{x}_1\underline{x}_2, \underline{x}_1\overline{x}_2, \overline{x}_1\underline{x}_2, \overline{x}_1\overline{x}_2\}, \max\{\underline{x}_1\underline{x}_2, \underline{x}_1\overline{x}_2, \overline{x}_1\underline{x}_2, \overline{x}_1\overline{x}_2\}\big],$$
$$I_1/I_2 = I_1 \cdot \big[\frac{1}{\overline{x}_2}, \frac{1}{\underline{x}_2}\big], \quad 0 \notin I_2.$$

Während die Addition und Subtraktion mit jeweils zwei Verknüpfungen in den Grenzen auskommt, benötigt man für die Multiplikation vier Produkte und einige Vergleiche, um die Intervallgrenzen zu bestimmen. Es ist aber auch eine Variante möglich, welche höchstens drei Multiplikationen in den Intervallgrenzen benötigt und im Durchschnitt mit zwei Multiplikationen auskommt [23].

Addition und Multiplikation in $\mathbb{IR}$ sind kommutativ und assoziativ, allerdings gilt das Distributivgesetz nicht, sondern nur die sogenannte Subdistributivität [1, S. 3]

$$I_1 \cdot (I_2 + I_3) \subseteq I_1 \cdot I_2 + I_1 \cdot I_3.$$

Dies hat zur Folge, dass bei der Umformung von Intervalltermen im Allgemeinen die Gleichheit verlorengeht und man stattdessen Inklusionsbeziehungen erhält. Das Gleichheitszeichen erhält man nur in einigen Sonderfällen, zum Beispiel wenn I_1 ein Punktintervall ist.

Es folgt die Definition der Begriffe Mittelpunkt, Durchmesser, Radius und Absolutbetrag für Intervalle.

Definition 2
Ist $I = [\underline{x}, \overline{x}] \in \mathbb{IR}$*, dann heißt*

$$\mathrm{m}(I) := \frac{\underline{x} + \overline{x}}{2} \quad \textit{Mittelpunkt von } I, \qquad \mathrm{d}(I) := \overline{x} - \underline{x} \quad \textit{Durchmesser von } I,$$
$$\mathrm{r}(I) := \frac{\overline{x} - \underline{x}}{2} \quad \textit{Radius von } I, \qquad |I| := \max\{|\underline{x}|, |\overline{x}|\} \quad \textit{Betrag von } I.$$

Für Intervallvektoren und -matrizen werden diese Begriffe komponentenweise erklärt. Man erhält also im Falle der Mittelpunktsbildung bei einem Intervallvektor einen reellen Vektor mit den Mittelpunkten der Komponentenintervalle als Einträge.

Wie die Grundoperationen lassen sich auch Standardfunktionen auf $\mathbb{IR}$ definieren. Unter einer Standardfunktion verstehen wir in dieser Arbeit eine Funktion aus

$$\mathcal{SF} := \left\{\exp, \ln, \sin, \cos, \sinh, \cosh, \sqrt{\ }, \mathrm{power}\right\}, \tag{1.4}$$

wobei $\mathcal{SF}$ auch noch um weitere Funktionen erweitert werden kann. Die Potenzfunktion power ist definiert als

$$\mathrm{power}(x, k) := x^k, \quad k \in \mathbb{N}. \tag{1.5}$$

Ist $\mathcal{D}_\varphi \subseteq \mathbb{R}$ der Definitionsbereich von $\varphi \in \mathcal{SF}$, so definieren wir auf der Menge $\mathbb{ID}_\varphi := \{I \in \mathbb{IR} \,|\, I \subseteq \mathcal{D}_\varphi\}$ eine Erweiterung von φ auf $\mathbb{IR}$ durch

$$\varphi(I) := \left[\min_{x \in I} \varphi(x), \max_{x \in I} \varphi(x)\right]. \tag{1.6}$$

Die Berechnung der Intervallschranken in (1.6) ist beispielsweise für die monoton wachsende Funktion exp bei gegebenem I leicht zu realisieren, während bei sin oder cos deutlich mehr Aufwand betrieben werden muss [12, 33, 54].

Es gilt [vgl. 1, S. 6]

Satz 2 (Inklusionsmonotonie)
Seien I_1*,* I_2*,* I_3 *und* I_4 *Intervalle aus* $\mathbb{IR}$*,* $\circ$ *eine Verknüpfung aus* $\{+, -, \cdot, /\}$ *und* φ *eine Funktion aus* $\mathcal{SF}$*. Dann gilt:*

a) Für $I_1 \subseteq I_2$ *und* $I_3 \subseteq I_4$ *ist* $I_1 \circ I_3 \subseteq I_2 \circ I_4$*. Insbesondere gilt mit* $x_1 \in I_1$ *und* $x_2 \in I_2$ *die Inklusion* $x_1 \circ x_2 \in I_1 \circ I_2$*.*

b) Für $I_1 \subseteq I_2 \in \mathbb{ID}_\varphi$ *ist* $\varphi(I_1) \subseteq \varphi(I_2)$*. Insbesondere gilt mit* $x_1 \in I_1$ *die Inklusion* $\varphi(x_1) \in \varphi(I_1)$*.*

Sei nun f eine stetige, reellwertige Funktion. Ein Funktionsausdruck von f ist eine Vorschrift zur Berechnung des Funktionswertes für jedes Argument des Definitionsbereichs $\mathcal{D}_f$. Der Funktionsausdruck soll dabei aus endlich vielen Operationen und Standardfunktionen sowie reellen Konstanten und Variablen $x_1, \ldots, x_\nu$ zusammengesetzt sein.

Ersetzt man die Variablen durch Intervalle $I_1, \ldots, I_\nu$ und sind alle im Ausdruck auszuführenden Intervalloperationen durchführbar, so existiert die intervallmäßige Auswertung, die wir mit $f(I_1, \ldots, I_\nu)$ bezeichnen wollen. Konstanten werden dabei als Punktintervalle aufgefasst. Das Ergebnis ist aufgrund der Stetigkeit der Operationen und Standardfunktionen wieder ein Intervall.

Sei

$$W(f; I_1, \ldots, I_\nu) := \left\{ f(x_1, \ldots, x_\nu) \,|\, x_i \in I_i,\ i = 1, \ldots, \nu \right\}$$

der Wertebereich der Funktion f auf $(I_1, \ldots, I_\nu) \in \mathbb{IR}^\nu$. Es gilt [vgl. 1, S. 22]:

Satz 3
Ist f eine Funktion gegeben durch einen Funktionsausdruck in ν Variablen und existiert $f(I_1, \ldots, I_\nu)$ für ein $(I_1, \ldots, I_\nu) \in \mathbb{IR}^\nu$, dann gilt für $(\tilde{I}_1, \ldots, \tilde{I}_\nu) \subseteq (I_1, \ldots, I_\nu)$:

a) $f(\tilde{I}_1, \ldots, \tilde{I}_\nu)$ existiert und

b) $f(\tilde{I}_1, \ldots, \tilde{I}_\nu) \subseteq f(I_1, \ldots, I_\nu)$.

Weiter gilt

$$W(f; I_1, \ldots, I_\nu) \subseteq f(I_1, \ldots, I_\nu). \tag{1.7}$$

Der Wertebereich einer Funktion f auf $(I_1, \ldots, I_\nu) \in \mathbb{IR}^\nu$ ist also in der intervallmäßigen Auswertung eines zu f gehörenden Funktionsausdrucks enthalten.

Eine Aussage darüber, wann in (1.7) Gleichheit gilt, macht der folgende Satz [vgl. 1, S. 23].

Satz 4
Ist f eine Funktion gegeben durch einen Funktionsausdruck in ν Variablen und kommt darin jede Variable höchstens einmal vor, dann gilt für alle $(I_1, \ldots, I_\nu) \in \mathbb{IR}^\nu$, für die $f(I_1, \ldots, I_\nu)$ existiert,

$$f(I_1, \ldots, I_\nu) = W(f; I_1, \ldots, I_\nu).$$

Eine Funktion kann durch mehrere verschiedene Funktionsausdrücke dargestellt werden. Dies führt in der Intervallrechnung zum sogenannten Abhängigkeitsproblem (engl. dependency problem). Das bedeutet, dass der Wert der Funktion, genauer gesagt das Ergebnisintervall, davon abhängen kann, wie oft eine Variable im Funktionsausdruck vorkommt. Nach obigem Satz erhält man genau den Wertebereich, falls jede Variable höchstens einmal vorkommt. Ist dies nicht der Fall,

erhält man eventuell Überschätzungen des Wertebereichs, die, je nach Funktionsausdruck, sehr groß sein können.

Neben der einfachen Intervallauswertung eines Funktionsausdrucks gibt es auch noch andere Formen der Auswertung zur Einschließung des Wertebereichs. Eine solche Form ist die sogenannte Mittelwertform für stetig differenzierbare Funktionen f. Sie basiert auf dem Mittelwertsatz der Differentialrechnung und lautet (für $\nu = 1$)

$$F(I) := f(x_0) + f'(I) \cdot (I - x_0), \quad x_0 \in I,$$

wobei $f'(I)$ eine intervallmäßige Auswertung der Ableitung f' bezeichnet. Die Mittelwertform liefert ebenfalls eine Einschließung des Wertebereichs von f auf I, wobei gezeigt werden kann, dass unter gewissen Voraussetzungen an f' die Überschätzung von $W(f; I)$ durch $F(I)$ quadratisch gegen Null geht, wenn der Durchmesser $\mathrm{d}(I)$ gegen Null strebt [1, S. 28]. Da f' intervallmäßig ausgewertet wird, ist der Wert der Mittelwertform, neben der Wahl von x_0, auch abhängig von dem verwendeten Funktionsausdruck für f'.

Eine andere Strategie zur Berechnung einer möglichst engen Wertebereichseinschließung ist, Minimum und Maximum von f auf I aufzuspüren. Der folgende Satz von Neumaier [51] bietet ein Kriterium, mit dem die genannte Strategie verfolgt werden kann. Er stellt eine Verallgemeinerung des von Makino [42, S. 128-130] vorgestellten *Linear Dominated Bounder* (LDB) dar.

Satz 5
Sei f eine auf dem Intervall $\boldsymbol{I}_f = (I_{f,1}, \ldots, I_{f,\nu})^\mathsf{T} \in \mathbb{IR}^\nu$ definierte, reellwertige Funktion von ν Veränderlichen. Weiter seien $\boldsymbol{z} \in \boldsymbol{I}_f$, $I_C \in \mathbb{IR}$ und $\boldsymbol{I_L} = (I_{L,1}, \ldots, I_{L,\nu})^\mathsf{T} \in \mathbb{IR}^\nu$ derart, dass

$$f(\boldsymbol{x}) \in I_C + \boldsymbol{I_L}^\mathsf{T}(\boldsymbol{x} - \boldsymbol{z}) \tag{1.8}$$

für alle $\boldsymbol{x} \in \boldsymbol{I}_f$ gilt. Bezeichnen $\boldsymbol{x}'$, $\boldsymbol{x}'' \in \boldsymbol{I}_f$ Extremalstellen von f in $\boldsymbol{I}_f$ mit

$$f(\boldsymbol{x}') \leqslant f(\boldsymbol{x}) \leqslant f(\boldsymbol{x}'')$$

für alle $\boldsymbol{x} \in \boldsymbol{I}_f$ und gilt für ein $i \in \{1, \ldots, \nu\}$

$$|\,\mathrm{m}(I_{L,i})|\;\mathrm{d}(I_{f,i}) > \delta := \mathrm{d}(I_C) + \mathrm{d}(\boldsymbol{I_L})^\mathsf{T}\,|\boldsymbol{x} - \boldsymbol{z}|, \tag{1.9}$$

dann ist

$$x_i' \in \begin{cases} \left[\inf(I_{f,i})\,,\, \inf(I_{f,i}) + \delta/|\,\mathrm{m}(I_{L,i})|\right] & \text{falls } \mathrm{m}(I_{L,i}) > 0, \\ \left[\sup(I_{f,i}) - \delta/|\,\mathrm{m}(I_{L,i})|\,,\, \sup(I_{f,i})\right] & \text{falls } \mathrm{m}(I_{L,i}) < 0, \end{cases}$$

$$x_i'' \in \begin{cases} \left[\inf(I_{f,i})\,,\, \inf(I_{f,i}) + \delta/|\,\mathrm{m}(I_{L,i})|\right] & \text{falls } \mathrm{m}(I_{L,i}) < 0, \\ \left[\sup(I_{f,i}) - \delta/|\,\mathrm{m}(I_{L,i})|\,,\, \sup(I_{f,i})\right] & \text{falls } \mathrm{m}(I_{L,i}) > 0. \end{cases}$$

Ist die Bedingung (1.9) für ein $i \in \{1, \ldots, \nu\}$ erfüllt, dann kann das Problem der Wertebereichseinschließung auf $\boldsymbol{I}_f$ aufgeteilt werden in ein Minimierungs- und Maximierungsproblem auf zwei kleineren Intervallen aus $\mathbb{IR}^\nu$. Durch Berechnen einer Einschließung der Form (1.8) von f auf jedem der zwei erhaltenen Teilintervalle entsteht ein iterativer Prozess, welcher für gewöhnlich zu engeren Wertebereichseinschließungen führt [51].

Im weiteren Verlauf dieser Arbeit wird speziell die Wertebereichseinschließung von multivariaten Polynomen eine wichtige Rolle spielen. Deshalb sollte man gerade in diesem Bereich geeignete Methoden verwenden. Neben den eben genannten Methoden werden in der Dissertation von Stahl [58] noch weitere Vorgehensweisen angegeben und miteinander verglichen. Da allerdings der Zweck dieser Arbeit ein Anderer ist, wird auf eine weitere Diskussion dieser Problemstellung verzichtet und die oben erwähnten Methoden zur Wertebereichseinschließung verwendet.

2 Taylor-Modelle

Wer zur Quelle gehen kann, gehe
nicht zum Wassertopf.

Leonardo da Vinci (1452-1519)

Wir werden in diesem Kapitel in das Konzept des Taylor-Modells, wie es von Berz und Makino in [5, 45] erdacht wurde, einführen und die Grundlagen zur Lösungseinschließung bei gewöhnlichen Anfangswertproblemen bereitstellen. Wir machen dazu Gebrauch von ähnlich klingenden Begriffen wie Taylorpolynom, Taylorreihe, Satz von Taylor, Taylorentwicklung, etc., werden diese aber vor deren Verwendung erklären.

Die Taylor-Modelle sind aus einem praktischen Problem der Physik entstanden [45]. Neben der Verwendung zur Lösungseinschließung bei Anfangswertproblemen gewöhnlicher Differentialgleichungen, welche wir in dieser Arbeit untersuchen und beschreiben wollen, gibt es noch eine Vielzahl anderer Einsatzmöglichkeiten. So können Taylor-Modelle zur Wertebereichseinschließung von Funktionen verwendet werden [9, 44, 45], zur Quadratur [11, 45], zur Lösung impliziter Gleichungen [7] und zur Lösung impliziter gewöhnlicher Differentialgleichungen [26, 27, 28]. Ein großer Vorteil der Taylor-Modelle, der in allen genannten Bereichen ausgenutzt wird, ist die Verminderung des Abhängigkeitsproblems beim Auswerten von Funktionsausdrücken. Speziell bei deren Verwendung zur Lösung von Anfangswertproblemen führt die Darstellung der Lösungsmenge mit Taylor-Modellen zu einem verminderten Wrapping-Effekt [29, 41, 46].

Zur einfacheren Darstellung beschränken wir uns auf den Fall reellwertiger Funktionen von zwei Veränderlichen s und t. Im Allgemeinen Fall von mehr als zwei Veränderlichen ändert sich lediglich die Bezeichnungsweise. Es sei $n \geqslant 0$ und

$$\boldsymbol{D} := [\underline{s}, \overline{s}] \times [\underline{t}, \overline{t}], \quad \underline{s} < \overline{s},\ \underline{t} < \overline{t}$$

der Definitionsbereich der im Folgenden auftretenden Funktionen. Wir vereinbaren weiter, dass alle auftretenden Taylorpolynome, wenn nichts anderes gesagt wird, um $(s_0, t_0) \in \boldsymbol{D}$ entwickelt sind. Die Stelle (s_0, t_0) ist dabei beliebig, aber fest gewählt. Für den Wertebereich $W(y; \boldsymbol{D})$ einer auf $\boldsymbol{D}$ definierten Funktion y schreiben wir kürzer $W(y)$.

2.1 Definition

Für eine nichtleere offene Teilmenge $\mathcal{G}$ des $\mathbb{R}^2$ bezeichne $C^{n+1}(\mathcal{G})$ die Menge aller auf $\mathcal{G}$ definierten reellwertigen Funktionen, deren partielle Ableitungen bis zur Ordnung $n+1$ auf $\mathcal{G}$ existieren und dort stetig sind [24, S. 250]. Mit dem Differenzierbarkeitsbegriff aus [24, S. 259] den wir hier zugrunde legen wollen, handelt es sich hierbei um die Menge aller $(n+1)$-mal stetig differenzierbaren Funktionen auf $\mathcal{G}$ [24, S. 250, S. 262].

Für die Einführung des Taylor-Modells spielt der Satz von Taylor eine zentrale Rolle. Die folgende Formulierung ist dabei an Heuser [24, S. 282] angelehnt:

Satz 6 (Satz von Taylor)
Sei $\mathcal{G} \subset \mathbb{R}^2$ offen und $y \in C^{n+1}(\mathcal{G})$ $(n \geqslant 0)$. Liegen die Punkte (s_0, t_0) und (s, t) mitsamt ihrer Verbindungsstrecke in $\mathcal{G}$, so gibt es ein θ zwischen 0 und 1, so dass gilt:

$$y(s,t) = \sum_{j=0}^{n} \frac{1}{j!} \left((s-s_0)\frac{\partial}{\partial s} + (t-t_0)\frac{\partial}{\partial t} \right)^j y(s_0,t_0)$$

$$+ \frac{1}{(n+1)!} \left((s-s_0)\frac{\partial}{\partial s} + (t-t_0)\frac{\partial}{\partial t} \right)^{n+1} y\big(s_0 + \theta(s-s_0), t_0 + \theta(t-t_0)\big).$$

Dabei bedeutet

$$\left((s-s_0)\frac{\partial}{\partial s} + (t-t_0)\frac{\partial}{\partial t} \right)^k y(s,t) = \sum_{i=0}^{k} \binom{k}{i} (s-s_0)^{k-i} (t-t_0)^i \frac{\partial^k y(s,t)}{\partial s^{k-i} \partial t^i}.$$

Die Stellen (s_0, t_0) und $(s_0 + \theta(s-s_0), t_0 + \theta(t-t_0))$ werden jeweils nach der Bildung der Ableitungen eingesetzt.

Wir betrachten die Menge $C^{n+1}(\boldsymbol{D})$ mit dem kompakten Definitionsbereich $\boldsymbol{D}$. Das Innere $(\underline{s}, \overline{s}) \times (\underline{t}, \overline{t})$ von $\boldsymbol{D}$ ist eine offene Menge und nicht leer (da $\underline{s} < \overline{s},\ \underline{t} < \overline{t}$). $(n+1)$-mal stetige Differenzierbarkeit auf $\boldsymbol{D}$ bedeutet, dass jede Funktion auf dem Inneren von $\boldsymbol{D}$ $(n+1)$-mal stetig differenzierbar ist und die partiellen Ableitungen bis zur Ordnung $(n+1)$ stetig auf den Rand von $\boldsymbol{D}$ fortgesetzt werden können.

Für eine Funktion $y \in C^{n+1}(\boldsymbol{D})$ bezeichne im Folgenden

$$P_{n,y}(s-s_0, t-t_0) := \sum_{j=0}^{n} \frac{1}{j!} \left((s-s_0)\frac{\partial}{\partial s} + (t-t_0)\frac{\partial}{\partial t} \right)^j y(s_0,t_0) \tag{2.1}$$

das Taylorpolynom der Ordnung n und

$$R_{n,y}(s-s_0,t-t_0) := \frac{1}{(n+1)!}\left((s-s_0)\frac{\partial}{\partial s}+(t-t_0)\frac{\partial}{\partial t}\right)^{n+1} y\big(s_0+\theta(s-s_0),t_0+\theta(t-t_0)\big)$$

das zugehörige Restglied, welches den Approximationsfehler beschreibt. Es gilt damit die Gleichung

$$y(s,t) = P_{n,y}(s-s_0,t-t_0) + R_{n,y}(s-s_0,t-t_0) \quad \forall (s,t) \in \boldsymbol{D}.$$

Definition 3 (Fehlereinschließung)
Ein Intervall $I_{n,y} \in \mathbb{IR}$ heißt Fehlereinschließung n-ter Ordnung von y auf $\boldsymbol{D}$ genau dann, wenn $R_{n,y}(s-s_0,t-t_0) \in I_{n,y}$ für alle $(s,t) \in \boldsymbol{D}$ gilt.

Mit einer Fehlereinschließung $I_{n,y}$ n-ter Ordnung gilt

$$y(s,t) \in P_{n,y}(s-s_0,t-t_0) + I_{n,y} \quad \forall (s,t) \in \boldsymbol{D}, \tag{2.2}$$

d. h. für $(s,t) \in \boldsymbol{D}$ ist der Funktionswert $y(s,t)$ in dem Intervall

$$P_{n,y}(s-s_0,t-t_0) + I_{n,y} \in \mathbb{IR},$$

das sich aus der Addition des Wertes $P_{n,y}(s-s_0,t-t_0)$ und der Fehlereinschließung $I_{n,y}$ ergibt, enthalten. Eine Fehlereinschließung $I_{n,y}$ ist keineswegs eindeutig bestimmt. Beispielsweise alle Intervalle, die ein $I_{n,y}$ enthalten, sind ebenfalls Fehlereinschließungen.

Fehlereinschließungen können mittels Automatischer Differentiation unter Verwendung von Intervallarithmetik berechnet werden. Eine ausführliche Einführung und Diskussion der Automatischen Differentiation findet man bei Moore [47, 48], Rall [53], Griewank [19] oder Corliss und Griewank [13]. Eine kurze und übersichtliche Darstellung zur Berechnung einer Fehlereinschließung mit Hilfe der Automatischen Differentiation im Fall einer Veränderlichen gibt beispielsweise Lohner [41, S. 18-23]. Wie wir bald sehen werden, bietet der Formalismus der Taylor-Modelle eine weitere Möglichkeit zur Berechnung einer Fehlereinschließung für eine Funktion y.

Bemerkung 2.1 An dieser Stelle möchten wir zwei Anmerkungen zu den Bezeichnungen der eingeführten Begriffe loswerden, die für das weitere Verstehen der vorliegenden Arbeit nützlich sind.

a) Taylorpolynom, Restglied und Fehlereinschließung sind jeweils mit zwei Indizes versehen. Der erste Index ist die Ordnung n des Taylorpolynoms, der

zweite Index die dazugehörige Funktion. Wir können also z. B. an der Bezeichnung $P_{n,y}$ ablesen, dass es sich hierbei um das Taylorpolynom der Ordnung n zur Funktion y handelt. Die Entwicklungsstellen sind, wie zu Beginn vereinbart, fest gewählt. Ansonsten müssten wir diese auch noch in der Bezeichnung unterbringen.

Der Bedeutung dieser Schreibweise folgend, schreiben wir dann auch z. B. P_{n,y_1+y_2} für das Taylorpolynom der Ordnung n der Summe zweier Funktionen y_1 und y_2. Analoges gilt für Subtraktion, Multiplikation und Division zweier Funktionen. Im Falle einer Verkettung $f(y)$ schreiben wir $P_{n,f(y)}$ für das dazugehörige Taylorpolynom.

In manchen Fällen, in denen wir konkrete, durch einen Funktionsausdruck gegebene Funktionen betrachten, wie z. B. $y(s,t) = st - 1$ verwenden wir auch die Schreibweise $P_{n,st-1}$ anstelle von $P_{n,y}$. Wir ersetzen also im Index das Symbol y durch den Funktionsausdruck. Dies wird überwiegend in den nachfolgenden Beispielen der Fall sein.

Handelt es sich bei der betrachteten Funktion um eine vektorwertige Funktion, z. B.

$$\boldsymbol{y} = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}, \quad y_1,\, y_2 \in C^{n+1}(\boldsymbol{D}),$$

dann bezeichnet $\boldsymbol{P}_{n,\boldsymbol{y}}$ einen Vektor mit den Taylorpolynomen P_{n,y_1} und P_{n,y_2} als Einträgen. Diese Schreibweise wird in Kapitel 3 verstärkt auftreten.

Auch wenn wir die Besonderheiten dieser Schreibweise nur am Beispiel des Taylorpolynoms erläutert haben, gelten die gemachten Erklärungen auch analog für die Bezeichnung des Restglieds und der Fehlereinschließung.

b) An einigen Stellen in der vorliegenden Arbeit betrachten wir auch Funktionen y von nur einer Variablen. Dies wird z. B. im Abschnitt 2.2.2 so sein, wo wir Taylorentwicklungen der Standardfunktionen aus (1.4) betrachten. In Analogie zu der hier eingeführten Schreibweise bezeichnen wir auch in diesen Fällen das Taylorpolynom der Ordnung n mit $P_{n,y}$ und das zugehörige Restglied mit $R_{n,y}$. Es ändert sich nur die Zahl der Argumente. Es ist

$$y(s) = P_{n,y}(s - s_0) + R_{n,y}(s - s_0)$$

für alle $s \in [\underline{s}, \overline{s}]$.

Wir betrachten den Raum $C(\boldsymbol{D})$ der stetigen, reellwertigen Funktionen auf $\boldsymbol{D}$ versehen mit der Maximumnorm und den bekannten Operationen $\{+, -, \cdot, /\}$ sowie der unbestimmten Integration $\int$ bzgl. einer Variablen. Zusätzlich statten wir $C(\boldsymbol{D})$

noch mit der partiellen Ordnung

$$y_1 \leqslant y_2 \quad :\Leftrightarrow \quad y_1(s,t) \leqslant y_2(s,t) \quad \forall (s,t) \in \boldsymbol{D}$$

aus.

Im Rahmen einer problemangepassten Verifikation von Lösungen funktionaler Probleme stehen Darstellung und Arithmetik von Funktionenmengen im Vordergrund. Analog zur Intervallarithmetik werden wir aus der Potenzmenge $\mathcal{P}C(\boldsymbol{D})$ von $C(\boldsymbol{D})$ ein geeignetes Mengensystem herausgreifen und in diesem eine Arithmetik definieren.

Definition 4 (Taylor-Modell)
Sei $P_{n,y}$ das Taylorpolynom n-ter Ordnung einer Funktion $y \in C^{n+1}(\boldsymbol{D})$ und $I_{n,y} \in \mathbb{IR}$ eine Fehlereinschließung n-ter Ordnung von y auf $\boldsymbol{D}$. Dann bezeichnen wir mit

$$[\![P_{n,y}, I_{n,y}]\!] := \left\{ \tilde{y} \in C(\boldsymbol{D}) \,\middle|\, \tilde{y}(s,t) \in P_{n,y}(s-s_0, t-t_0) + I_{n,y} \;\forall (s,t) \in \boldsymbol{D} \right\} \quad (2.3)$$

die Menge aller auf $\boldsymbol{D}$ stetigen Funktionen $\tilde{y}$, deren Funktionswerte $\tilde{y}(s,t)$, $(s,t) \in \boldsymbol{D}$, durch die Intervalle

$$P_{n,y}(s-s_0, t-t_0) + I_{n,y} \in \mathbb{IR},$$

die sich aus der Addition des Wertes $P_{n,y}(s-s_0, t-t_0)$ und der Fehlereinschließung $I_{n,y}$ ergeben, erfasst werden. Mit anderen Worten sind das alle Funktionen $\tilde{y}$, die mit der Unterfunktion

$$\underline{p}_{n,y}(s,t) := P_{n,y}(s-s_0, t-t_0) + \inf(I_{n,y})$$

und der Oberfunktion

$$\overline{p}_{n,y}(s,t) := P_{n,y}(s-s_0, t-t_0) + \sup(I_{n,y})$$

die Ungleichung

$$\underline{p}_{n,y} \leqslant \tilde{y} \leqslant \overline{p}_{n,y} \quad (2.4)$$

erfüllen. Wir nennen $[\![P_{n,y}, I_{n,y}]\!]$ ein Taylor-Modell n-ter Ordnung von y. Die Menge aller Taylor-Modelle $[\![P_{n,y}, I_{n,y}]\!]$, $y \in C^{n+1}(\boldsymbol{D})$, bezeichnen wir mit $\mathcal{TM}_n(\boldsymbol{D})$.

Bemerkung 2.2 Ein Taylor-Modell $[\![P_{n,y}, I_{n,y}]\!]$ einer Funktion $y \in C^{n+1}(\boldsymbol{D})$ wird über das Taylorpolynom $P_{n,y}$ und über eine Fehlereinschließung $I_{n,y}$ n-ter Ordnung von y erklärt. Deshalb werden wir diese fortan auch als Komponenten eines Taylor-Modells bezeichnen. Eine Fehlereinschließung ist nicht eindeutig. Sind $I_{n,y}^{(1)}$ und $I_{n,y}^{(2)}$ zwei Fehlereinschließungen n-ter Ordnung von y mit $I_{n,y}^{(1)} \neq I_{n,y}^{(2)}$, so sind $[\![P_{n,y}, I_{n,y}^{(1)}]\!]$ und $[\![P_{n,y}, I_{n,y}^{(2)}]\!]$ zwei Taylor-Modelle n-ter Ordnung von y, die unterschiedliche Funktionenmengen darstellen. Nach Definition 3 bzw. (2.2) gilt $y \in [\![P_{n,y}, I_{n,y}^{(1)}]\!]$ und $y \in [\![P_{n,y}, I_{n,y}^{(2)}]\!]$.

Bemerkung 2.3 Ergänzend zu Bemerkung 2.1 machen wir zwei weitere Anmerkungen zu den Bezeichnungen und Sprechweisen der eingeführten Begriffe.

a) Gelegentlich verwenden wir in dieser Arbeit auch die umgangssprachliche Bezeichnung „Polynomanteil“ oder „polynomialer Anteil“ für das Taylorpolynom eines Taylor-Modells n-ter Ordnung, sofern aus dem Kontext heraus klar ist, welches Taylorpolynom damit gemeint ist. Genauso verhält es sich mit der Bezeichnung „Intervallterm“, die an einigen Stellen synonym für eine Fehlereinschließung n-ter Ordnung verwendet wird.

b) Die eingeführten Begriffe sind, wie aus den Definitionen ersichtlich wird, alle von der Polynomordnung n abhängig. Diese Abhängigkeit spiegelt sich in der Bezeichnung und der definierten Sprechweise dieser Begriffe wieder. Da sich aber die Ordnung n nur an wenigen Stellen in dieser Arbeit ändert, schreiben wir im Folgenden für „Taylor-Modell n-ter Ordnung“ bzw. „Taylor-Modell der Ordnung n“ kurz „Taylor-Modell“. Auch für den Begriff der Fehlereinschließung verwenden wir im weiteren Verlauf den kürzeren Ausdruck „Fehlereinschließung“ anstatt der Sprechweise „Fehlereinschließung n-ter Ordnung“. Lediglich an Stellen an denen n konkrete Werte annimmt oder wo eine explizite Nennung wichtig erscheint, kehren wir zu den ursprünglichen Sprechweisen zurück.

Beispiel 2.1 *Ein Taylor-Modell für die konstante Funktion $y(s,t) = c$, $c \in \mathbb{R}$, ist*

$$[\![P_{n,c}, [0,0]]\!] \text{ mit } P_{n,c}(s - s_0, t - t_0) = c.$$

Für $y(s,t) = s$ und jedes $n \geqslant 1$ ist

$$[\![P_{n,s}, [0,0]]\!] \text{ mit } P_{n,s}(s - s_0, t - t_0) = s_0 + (s - s_0)$$

ein Taylor-Modell, und für $y(s,t) = t$ und jedes $n \geqslant 1$ ist

$$[\![P_{n,t}, [0,0]]\!] \text{ mit } P_{n,t}(s - s_0, t - t_0) = t_0 + (t - t_0)$$

ein Taylor-Modell.

Bevor wir auf die Arithmetik zu sprechen kommen, werden wir noch auf Eigenschaften und Besonderheiten der Mengen (2.3) eingehen.

Sei $[\![P_{n,y}, I_{n,y}]\!] \in \mathcal{TM}_n(\boldsymbol{D})$ beliebig. Die Menge $[\![P_{n,y}, I_{n,y}]\!]$ ist wegen (2.4) beschränkt. Sind weiter $\tilde{y}_1$, $\tilde{y}_2 \in [\![P_{n,y}, I_{n,y}]\!]$, dann ist für $\lambda \in (0,1)$ auch

$$\underline{p}_{n,y} \leqslant \lambda\tilde{y}_1 + (1-\lambda)\tilde{y}_2 \leqslant \overline{p}_{n,y},$$

d. h. die Menge $[\![P_{n,y}, I_{n,y}]\!]$ ist zudem noch konvex. Wir zeigen als nächstes die Abgeschlossenheit von $[\![P_{n,y}, I_{n,y}]\!]$. Sei dazu $(\tilde{y}_k)_{k\in\mathbb{N}}$ eine konvergente Funktionenfolge aus $[\![P_{n,y}, I_{n,y}]\!]$ mit $\lim_{k\to\infty} \tilde{y}_k = \tilde{y} \in C(\boldsymbol{D})$ gleichmäßig auf $\boldsymbol{D}$. Angenommen die Grenzfunktion $\tilde{y}$ liegt nicht in $[\![P_{n,y}, I_{n,y}]\!]$. Dann existiert ein $(\hat{s}, \hat{t}) \in \boldsymbol{D}$ mit o. B. d. A. $\tilde{y}(\hat{s}, \hat{t}) < \underline{p}_{n,y}(\hat{s}, \hat{t})$. Da aus der gleichmäßigen Konvergenz die punktweise Konvergenz folgt, existiert ein Index k_0 mit $\tilde{y}_k(\hat{s}, \hat{t}) < \underline{p}_{n,y}(\hat{s}, \hat{t})$, wenn nur k größer als k_0 ist. Dies steht aber im Widerspruch zur Voraussetzung $\tilde{y}_k \in [\![P_{n,y}, I_{n,y}]\!]$ für alle k.

Die Mengen $[\![P_{n,y}, I_{n,y}]\!] \in \mathcal{TM}_n(\boldsymbol{D})$ sind also beschränkt, abgeschlossen und konvex. Zieht man einen Vergleich zu den reellen Intervallen, dann können die Mengen aus $\mathcal{TM}_n(\boldsymbol{D})$ als Verallgemeinerung des Intervallbegriffs auf den Funktionenraum $C(\boldsymbol{D})$ angesehen werden. Die Schranken dieser (Funktions-)Intervalle sind die Polynome $\underline{p}_{n,y}$ und $\overline{p}_{n,y}$. Das Besondere an diesen Mengen ist deren einfache Darstellung. Ober- und Unterschranke werden gemeinsam durch ein Polynom und ein Intervall dargestellt. Anders ausgedrückt unterscheiden sich die polynomialen Schranken nur um eine Konstante. Dies wird sich auch bei der Definition arithmetischer Operationen als günstig erweisen.

Aus (2.3) liest man die Äquivalenz

$$\tilde{y} \in [\![P_{n,y}, I_{n,y}]\!] \;\Leftrightarrow\; \tilde{y}(s,t) \in P_{n,y}(s - s_0, t - t_0) + I_{n,y} \quad \forall (s,t) \in \boldsymbol{D}, \tag{2.5}$$

ab. Die Inklusion

$$[\![P_{n,y_1}, I_{n,y_1}]\!] \subseteq [\![P_{n,y_2}, I_{n,y_2}]\!]$$

zweier Mengen $[\![P_{n,y_1}, I_{n,y_1}]\!]$, $[\![P_{n,y_2}, I_{n,y_2}]\!]$ aus $\mathcal{TM}_n(\boldsymbol{D})$ ist im üblichen Sinne zu verstehen, d. h. die linke Menge $[\![P_{n,y_1}, I_{n,y_1}]\!]$ ist Teilmenge der rechten Menge $[\![P_{n,y_2}, I_{n,y_2}]\!]$ genau dann, wenn jedes Element der linken Menge auch in der rechten Menge enthalten ist. Mit (2.5) gilt:

$$[\![P_{n,y_1}, I_{n,y_1}]\!] \subseteq [\![P_{n,y_2}, I_{n,y_2}]\!] \tag{2.6}$$

$$\Leftrightarrow\; P_{n,y_1}(s - s_0, t - t_0) + I_{n,y_1} \subseteq P_{n,y_2}(s - s_0, t - t_0) + I_{n,y_2} \quad \forall (s,t) \in \boldsymbol{D}$$

$$\Leftrightarrow\; P_{n,y_1}(s - s_0, t - t_0) - P_{n,y_2}(s - s_0, t - t_0) + I_{n,y_1} \subseteq I_{n,y_2} \quad \forall (s,t) \in \boldsymbol{D}. \tag{2.7}$$

Mit (2.7) hat man ein Kriterium zur Verfügung, mit dem die Inklusion (2.6) über die Darstellung der Mengen geprüft werden kann. Notwendig und hinreichend für (2.7) ist die Inklusion

$$W(P_{n,y_1} - P_{n,y_2}) + I_{n,y_1} \subseteq I_{n,y_2}, \tag{2.8}$$

die mit Hilfe der Intervallrechnung nachgeprüft werden kann. Es ist aufgrund der Stetigkeit des Polynoms $P_{n,y_1} - P_{n,y_2}$ und der Kompaktheit von $\boldsymbol{D}$ der Wertebereich $W(P_{n,y_1} - P_{n,y_2})$ ein Intervall aus $\mathbb{IR}$. (2.8) wird in Kapitel 3 für den Nachweis einer Inklusion von Funktionenmengen ein zentrale Rolle spielen.

2.2 Arithmetik

Wenden wir uns nun der Definition einer arithmetischen Struktur in $\mathcal{TM}_n(\boldsymbol{D})$ zu. Wir begeben uns zunächst wieder in die Potenzmenge von $C(\boldsymbol{D})$ und legen fest, wie Operationen auf Funktionenmengen zu verstehen sind.

Definition 5
Seien $\mathcal{Y}_1, \mathcal{Y}_2 \in \mathcal{P}C(\boldsymbol{D})$, $\circ \in \{+, -, \cdot, /\}$ eine Verknüpfung. Wir definieren

$$\mathcal{Y}_1 \circ_{\mathcal{P}} \mathcal{Y}_2 := \left\{ y_1 \circ y_2 \,\middle|\, y_1 \in \mathcal{Y}_1,\, y_2 \in \mathcal{Y}_2 \right\} \tag{2.9}$$

elementweise, wobei wir im Falle der Division voraussetzen, dass jedes $y_2 \in \mathcal{Y}_2$ auf $\boldsymbol{D}$ frei von Nullstellen ist. Die unbestimmte Integration bzgl. einer Variablen und auch die Standardfunktionen φ aus (1.4) werden für $\mathcal{Y} \in \mathcal{P}C(\boldsymbol{D})$ ebenfalls elementweise definiert:

$$\int_{\mathcal{P}} \mathcal{Y}\, ds := \left\{ \int_{s_0}^{s} y(\sigma, t)\, d\sigma \,\middle|\, y \in \mathcal{Y} \right\}, \quad \int_{\mathcal{P}} \mathcal{Y}\, dt := \left\{ \int_{t_0}^{t} y(s, \tau)\, d\tau \,\middle|\, y \in \mathcal{Y} \right\},$$

$$\varphi_{\mathcal{P}}(\mathcal{Y}) := \left\{ \varphi(y) \,\middle|\, y \in \mathcal{Y} \right\}.$$

Im Falle der unbestimmten Integration wird also zu jeder Funktion aus $\mathcal{Y}$ die in der Definition angegebene Stammfunktion bzgl. der gewählten Variablen bestimmt.

Es handelt sich bei $\mathcal{TM}_n(\boldsymbol{D})$ um eine Teilmenge der Potenzmenge $\mathcal{P}C(\boldsymbol{D})$. Mit obiger Definition ist damit nur teilweise eine Arithmetik in $\mathcal{TM}_n(\boldsymbol{D})$ erklärt, denn beispielsweise liegt im Falle der Multiplikation die Ergebnismenge nicht wieder in $\mathcal{TM}_n(\boldsymbol{D})$. Dazu später mehr. Außerdem sind obige Definitionen für praktische Zwecke nicht geeignet, da sie de facto nicht durchführbar sind. Es ist deshalb erforderlich, die elementweise definierten Verknüpfungen durch Verknüpfungen in den Intervallschranken, also der Darstellung, zu ersetzen. Das Ergebnis dieser Ersatzoperationen soll dabei wieder in $\mathcal{TM}_n(\boldsymbol{D})$ liegen.

Wir halten fest: die Ersatzoperationen $\left\{ \boxplus, \boxminus, \boxdot, \boxslash, \int\!\!\!\!\!\!\boxdot \right\}$ sind derart zu definieren, dass

a) deren Ergebnis ein Element aus $\mathcal{TM}_n(\boldsymbol{D})$ ist;

b) die Komponenten zur Darstellung der Ergebnismenge, also Taylorpolynom und Intervall, aus den Komponenten der Operanden errechnet werden;

c) die Inklusion ($\circ \in \{+, -, \cdot, /\}$)

$$[\![P_{n,y_1}, I_{n,y_1}]\!] \boxdot [\![P_{n,y_2}, I_{n,y_2}]\!] \supseteq [\![P_{n,y_1}, I_{n,y_1}]\!] \circ_{\mathcal{P}} [\![P_{n,y_2}, I_{n,y_2}]\!], \tag{2.10}$$

$$\oint\!\!\!\boxdot [\![P_{n,y}, I_{n,y}]\!] \begin{Bmatrix} ds \\ dt \end{Bmatrix} \supseteq \int_{\mathcal{P}} [\![P_{n,y}, I_{n,y}]\!] \begin{Bmatrix} ds \\ dt \end{Bmatrix}, \tag{2.11}$$

erfüllt wird (Einschließungseigenschaft).

Es wird sich zeigen, dass im Falle von Addition und Subtraktion in (2.10) sogar Gleichheit erreicht werden kann, während bei allen anderen Operationen in der Regel echte Inklusion vorliegt.

Wir werden in 2.2.1 Addition, Subtraktion und Multiplikation definieren, im darauf folgenden Abschnitt 2.2.2 dann die Division und in 2.2.3 abschließend noch die Integration bezüglich einer Variablen. In Abschnitt 2.2.2 werden wir zudem die Arithmetik auf $\mathcal{TM}_n(\boldsymbol{D})$ um einige Standardfunktionen ergänzen.

2.2.1 Addition, Subtraktion und Multiplikation

In diesem Abschnitt seien die Taylor-Modelle $[\![P_{n,y_1}, I_{n,y_1}]\!]$ und $[\![P_{n,y_2}, I_{n,y_2}]\!]$ aus $\mathcal{TM}_n(\boldsymbol{D})$ mit Funktionen y_1, $y_2 \in C^{n+1}(\boldsymbol{D})$, beliebig gewählt. Es ist nach (2.3)

$$[\![P_{n,y_1}, I_{n,y_1}]\!] = \left\{ \tilde{y}_1 \in C(\boldsymbol{D}) \,\middle|\, \tilde{y}_1(s,t) \in P_{n,y_1}(s - s_0, t - t_0) + I_{n,y_1} \ \forall (s,t) \in \boldsymbol{D} \right\},$$

$$[\![P_{n,y_2}, I_{n,y_2}]\!] = \left\{ \tilde{y}_2 \in C(\boldsymbol{D}) \,\middle|\, \tilde{y}_2(s,t) \in P_{n,y_2}(s - s_0, t - t_0) + I_{n,y_2} \ \forall (s,t) \in \boldsymbol{D} \right\}.$$

Die Definitionen der Verknüpfungen sind recht eingängig, weshalb wir bei deren Herleitung nicht übermäßig viele Worte verlieren wollen, sondern diese kurz und prägnant vorstellen.

2.2.1.1 Addition und Subtraktion

Wir gehen zuerst auf die Verknüpfungsvorschrift für die Addition ein und kommen anschließend auf die Subtraktion zu sprechen.

Wie oben erwähnt, möchten wir aus den Argumenten ein Element aus $\mathcal{TM}_n(\boldsymbol{D})$ konstruieren, welches das Ergebnis

$$[\![P_{n,y_1}, I_{n,y_1}]\!] +_{\mathcal{P}} [\![P_{n,y_2}, I_{n,y_2}]\!] = \left\{ \tilde{y}_1 + \tilde{y}_2 \,\middle|\, \tilde{y}_1 \in P_{n,y_1} + I_{n,y_1},\ \tilde{y}_2 \in P_{n,y_2} + I_{n,y_2} \right\} \tag{2.12}$$

der Potenzmengenoperation einschließt. Die Summe der Taylorpolynome P_{n,y_1} und P_{n,y_2} ergibt gerade das Taylorpolynom von $y_1 + y_2$, also

$$P_{n,y_1} + P_{n,y_2} = P_{n,y_1+y_2}.$$

Damit gilt für beliebiges $\tilde{y}_1 \in [\![P_{n,y_1}, I_{n,y_1}]\!]$ und beliebiges $\tilde{y}_2 \in [\![P_{n,y_2}, I_{n,y_2}]\!]$

$$\begin{aligned}(\tilde{y}_1 + \tilde{y}_2)(s,t) - P_{n,y_1+y_2}(s-s_0, t-t_0)& \\ = \tilde{y}_1(s,t) - P_{n,y_1}(s-s_0,t-t_0) \;+\; \tilde{y}_2(s,t) - P_{n,y_2}(s-s_0,t-t_0)& \\ \in I_{n,y_1} + I_{n,y_2}, \quad \forall (s,t) \in \boldsymbol{D},&\end{aligned} \tag{2.13}$$

wie man leicht an der Eigenschaft (2.5) ablesen kann. Mit dem Taylorpolynom P_{n,y_1+y_2} von $y_1 + y_2$ und dem Intervall

$$I_{n,y_1+y_2} := I_{n,y_1} + I_{n,y_2} \tag{2.14}$$

das insbesondere eine Fehlereinschließung für $y_1 + y_2$ auf $\boldsymbol{D}$ darstellt – (2.13) gilt ja speziell auch für $y_1 + y_2$ –, haben wir aus den Komponenten der zwei Summanden ein Taylor-Modell errechnet mit

$$[\![P_{n,y_1+y_2}, I_{n,y_1+y_2}]\!] \supseteq [\![P_{n,y_1}, I_{n,y_1}]\!] +_{\mathcal{P}} [\![P_{n,y_2}, I_{n,y_2}]\!]$$

und $[\![P_{n,y_1+y_2}, I_{n,y_1+y_2}]\!] \in \mathcal{TM}_n(\boldsymbol{D})$.

Da überdies für Elemente aus (2.12) die Ungleichung

$$\underline{p}_{n,y_1} + \underline{p}_{n,y_2} \leqslant \tilde{y}_1 + \tilde{y}_2 \leqslant \overline{p}_{n,y_1} + \overline{p}_{n,y_2}$$

scharf ist und da

$$\begin{aligned}\underline{p}_{n,y_1} + \underline{p}_{n,y_2} &= \underline{p}_{n,y_1+y_2}, \\ \overline{p}_{n,y_1} + \overline{p}_{n,y_2} &= \overline{p}_{n,y_1+y_2}\end{aligned}$$

gilt, ist $[\![P_{n,y_1+y_2}, I_{n,y_1+y_2}]\!]$ nicht nur eine Einschließung von (2.12), sondern sogar die Ergebnismenge der Addition von $[\![P_{n,y_1}, I_{n,y_1}]\!]$ und $[\![P_{n,y_2}, I_{n,y_2}]\!]$ in der Potenzmenge. Es ist also

$$[\![P_{n,y_1+y_2}, I_{n,y_1+y_2}]\!] = [\![P_{n,y_1}, I_{n,y_1}]\!] +_{\mathcal{P}} [\![P_{n,y_2}, I_{n,y_2}]\!].$$

Auf analogem Wege erhalten wir für die Subtraktion

$$[\![P_{n,y_1-y_2}, I_{n,y_1-y_2}]\!] = [\![P_{n,y_1}, I_{n,y_1}]\!] -_{\mathcal{P}} [\![P_{n,y_2}, I_{n,y_2}]\!]$$

mit dem Taylorpolynom $P_{n,y_1-y_2} = P_{n,y_1} - P_{n,y_2}$ der Differenz $y_1 - y_2$ und der Fehlereinschließung $I_{n,y_1-y_2} := I_{n,y_1} - I_{n,y_2}$.

Wir definieren

$$[\![P_{n,y_1}, I_{n,y_1}]\!] \boxplus [\![P_{n,y_2}, I_{n,y_2}]\!] := [\![P_{n,y_1+y_2}, I_{n,y_1+y_2}]\!], \tag{2.15}$$

$$[\![P_{n,y_1}, I_{n,y_1}]\!] \boxminus [\![P_{n,y_2}, I_{n,y_2}]\!] := [\![P_{n,y_1-y_2}, I_{n,y_1-y_2}]\!]. \tag{2.16}$$

2.2.1.2 Multiplikation

Im Falle der Multiplikation ist aus den Operanden ein Element aus $\mathcal{TM}_n(\boldsymbol{D})$ zu konstruieren, welches die Menge

$$\begin{aligned}&[\![P_{n,y_1}, I_{n,y_1}]\!] \cdot_{\mathcal{P}} [\![P_{n,y_2}, I_{n,y_2}]\!] \\ &\qquad = \left\{ \tilde{y}_1 \cdot \tilde{y}_2 \,\middle|\, \tilde{y}_1 \in P_{n,y_1} + I_{n,y_1},\ \tilde{y}_2 \in P_{n,y_2} + I_{n,y_2} \right\} \end{aligned} \tag{2.17}$$

einschließt. Wir verfahren wie bei der Addition und Subtraktion und verknüpfen zunächst beide Taylorpolynome, d. h. wir bilden das Produkt $P_{n,y_1} P_{n,y_2}$ der Polynome P_{n,y_1} und P_{n,y_2}. Es handelt sich dabei um ein Polynom vom Grad höchstens $2n$, dessen Terme bis zur Ordnung n gerade das Taylorpolynom der Funktion $y_1 y_2$ repräsentieren. Wir schreiben

$$P_{n,y_1} P_{n,y_2} = P_{n,y_1 y_2} + P_{>n}, \tag{2.18}$$

wobei sich das Polynom $P_{>n}$ aus den Termen mit Ordnung größer als n zusammensetzt. Damit erhalten wir für beliebige $\tilde{y}_1 \in [\![P_{n,y_1}, I_{n,y_1}]\!]$, $\tilde{y}_2 \in [\![P_{n,y_2}, I_{n,y_2}]\!]$ und beliebiges $(s,t) \in \boldsymbol{D}$

$$\begin{aligned}&(\tilde{y}_1\, \tilde{y}_2)(s,t) - P_{n,y_1 y_2}(s-s_0, t-t_0) \\ &\quad = \tilde{y}_1(s,t)\, \tilde{y}_2(s,t) - P_{n,y_1}(s-s_0,t-t_0)\, P_{n,y_2}(s-s_0,t-t_0) \\ &\qquad + P_{>n}(s-s_0,t-t_0) \\ &\quad = \big(\tilde{y}_1(s,t) - P_{n,y_1}(s-s_0,t-t_0)\big)\, \tilde{y}_2(s,t) + P_{n,y_1}(s-s_0,t-t_0)\, \tilde{y}_2(s,t) \\ &\qquad - P_{n,y_1}(s-s_0,t-t_0)\, P_{n,y_2}(s-s_0,t-t_0) + P_{>n}(s-s_0,t-t_0) \\ &\quad = \big(\tilde{y}_1(s,t) - P_{n,y_1}(s-s_0,t-t_0)\big)\, \tilde{y}_2(s,t) \\ &\qquad + P_{n,y_1}(s-s_0,t-t_0)\, \big(\tilde{y}_2(s,t) - P_{n,y_2}(s-s_0,t-t_0)\big) \\ &\qquad + P_{>n}(s-s_0,t-t_0) \\ &\quad \in I_{n,y_1}\, \big(P_{n,y_2}(s-s_0,t-t_0) + I_{n,y_2}\big) + P_{n,y_1}(s-s_0,t-t_0)\, I_{n,y_2} \\ &\qquad + P_{>n}(s-s_0,t-t_0) \\ &\quad \subseteq I_{n,y_1}\, \big(W(P_{n,y_2}) + I_{n,y_2}\big) + W(P_{n,y_1})\, I_{n,y_2} + W(P_{>n}). \end{aligned} \tag{2.19}$$

Die Wertebereiche sind aufgrund der Stetigkeit der Funktionen und der Kompaktheit von $\boldsymbol{D}$ Intervalle aus $\mathbb{IR}$. Das Intervall

$$I_{n,y_1 y_2} := W(P_{>n}) + W(P_{n,y_1})\, I_{n,y_2} + I_{n,y_1}\, \big(W(P_{n,y_2}) + I_{n,y_2}\big) \tag{2.20}$$

ist eine Fehlereinschließung von $y_1 y_2$, da (2.19) insbesondere auch für $y_1 y_2$ Gültigkeit besitzt. Es ist also

$$(\tilde{y}_1\, \tilde{y}_2)(s,t) \in P_{n,y_1y_2}(s-s_0, t-t_0) + I_{n,y_1y_2} \quad \forall (s,t) \in \boldsymbol{D}.$$

Die Menge $[\![P_{n,y_1y_2}, I_{n,y_1y_2}]\!]$ schließt demnach die Menge (2.17) ein.

Vertauschen wir die Reihenfolge von y_1 und y_2, dann erhalten wir analog durch

$$[\![P_{n,y_1y_2}, I_{n,y_2y_1}]\!],$$

mit

$$I_{n,y_2y_1} := W(P_{>n}) + W(P_{n,y_2})\, I_{n,y_1} + I_{n,y_2} \left(W(P_{n,y_1}) + I_{n,y_1}\right), \tag{2.21}$$

eine weitere Einschließung der Menge (2.17), wobei durchaus $I_{n,y_2y_1} \neq I_{n,y_1y_2}$ sein kann.

Wir definieren

$$[\![P_{n,y_1}, I_{n,y_1}]\!] \boxdot [\![P_{n,y_2}, I_{n,y_2}]\!] := [\![P_{n,y_1y_2}, I_{n,y_1y_2} \cap I_{n,y_2y_1}]\!] \tag{2.22}$$

und erhalten damit eine kommutative Multiplikation in $\mathcal{TM}_n(\boldsymbol{D})$.

Beispiel 2.2 *Sei $n = 1$, $\boldsymbol{D} = [0,1] \times [0,1]$ und $s_0 = t_0 = \frac{1}{2}$. Wir betrachten zwei Taylor-Modelle $[\![P_{1,y_1}, I_{1,y_1}]\!]$, $[\![P_{1,y_2}, I_{1,y_2}]\!] \in \mathcal{TM}_1(\boldsymbol{D})$ zu Funktionen y_1, $y_2 \in C^2(\boldsymbol{D})$ mit*

$$P_{1,y_1}(s-\frac{1}{2}, t-\frac{1}{2}) = \frac{1}{2} + (s-\frac{1}{2}), \quad I_{1,y_1} = [-1,1],$$

$$P_{1,y_2}(s-\frac{1}{2}, t-\frac{1}{2}) = \frac{1}{2} + (t-\frac{1}{2}), \quad I_{1,y_2} = [-1,1],$$

und berechnen mit diesen ein Taylor-Modell erster Ordnung für $y_1 y_2$ nach Definition (2.22). Es ist

$$(P_{1,y_1} P_{1,y_2})(s-\frac{1}{2}, t-\frac{1}{2}) = \underbrace{\frac{1}{4} + \frac{1}{2}(s-\frac{1}{2}) + \frac{1}{2}(t-\frac{1}{2})}_{=P_{1,y_1y_2}(s-\frac{1}{2}, t-\frac{1}{2})} + \underbrace{(s-\frac{1}{2})(t-\frac{1}{2})}_{=P_{>1}(s-\frac{1}{2}, t-\frac{1}{2})}.$$

Der Wertebereich von $P_{>1}$ ist $[-\frac{1}{4}, \frac{1}{4}]$. Der Wertebereich von P_{1,y_1} sowie von P_{1,y_2} ist $[0,1]$. Die Fehlereinschließungen I_{1,y_1y_2} und I_{1,y_2y_1} ergeben sich damit zu

$$\begin{aligned} I_{1,y_1y_2} = I_{1,y_2y_1} &= I_{1,y_2}\left(W(P_{1,y_1}) + I_{1,y_1}\right) + W(P_{1,y_2})\, I_{1,y_1} + W(P_{>1}) \\ &= [-1,1] \cdot \left([0,1] + [-1,1]\right) + [0,1] \cdot [-1,1] + [-\frac{1}{4}, \frac{1}{4}] \\ &= [-2,2] + [-1,1] + [-\frac{1}{4}, \frac{1}{4}] = [-\frac{13}{4}, \frac{13}{4}]. \end{aligned}$$

Damit gilt

$$[\![P_{1,y_1}, I_{1,y_1}]\!] \boxdot [\![P_{1,y_2}, I_{1,y_2}]\!] = [\![P_{1,y_1y_2}, I_{1,y_1y_2} \cap I_{1,y_2y_1}]\!],$$

mit

$$P_{1,y_1y_2}(s-\frac{1}{2}, t-\frac{1}{2}) = \frac{1}{4} + \frac{1}{2}(s-\frac{1}{2}) + \frac{1}{2}(t-\frac{1}{2}), \quad I_{1,y_1y_2} \cap I_{1,y_2y_1} = [-\frac{13}{4}, \frac{13}{4}].$$

Wir zeigen abschließend, dass diese Definition im Allgemeinen eine echte Obermenge von (2.17) liefert. Der Einfachheit wegen betrachten wir den Fall

$$0 \leqslant \underline{p}_{n,y_1}(s,t), \quad 0 \leqslant \underline{p}_{n,y_2}(s,t) \quad \forall (s,t) \in \boldsymbol{D},$$

d. h. keine Funktion aus $[\![P_{n,y_1}, I_{n,y_1}]\!]$ bzw. $[\![P_{n,y_2}, I_{n,y_2}]\!]$ wechselt auf $\boldsymbol{D}$ das Vorzeichen. Für beliebige Funktionen $\tilde{y}_1 \in [\![P_{n,y_1}, I_{n,y_1}]\!]$, $\tilde{y}_2 \in [\![P_{n,y_2}, I_{n,y_2}]\!]$ erhalten wir dann wegen

$$\underline{p}_{n,y_1} \leqslant \tilde{y}_1 \leqslant \overline{p}_{n,y_1}, \quad \underline{p}_{n,y_2} \leqslant \tilde{y}_2 \leqslant \overline{p}_{n,y_2}$$

die scharfe Abschätzung – Ober- und Unterfunktion gehören jeweils auch noch zur Menge dazu –

$$\begin{aligned}
\tilde{y}_1\tilde{y}_2 &\leqslant \overline{p}_{n,y_1}\overline{p}_{n,y_2} \\
&= \big(P_{n,y_1} + \sup(I_{n,y_1})\big)\big(P_{n,y_2} + \sup(I_{n,y_2})\big) \\
&= P_{n,y_1}P_{n,y_2} + P_{n,y_1}\sup(I_{n,y_2}) + P_{n,y_2}\sup(I_{n,y_1}) + \sup(I_{n,y_1})\sup(I_{n,y_2})
\end{aligned}$$

$$\begin{aligned}
\tilde{y}_1\tilde{y}_2 &\geqslant \underline{p}_{n,y_1}\underline{p}_{n,y_2} \\
&= \big(P_{n,y_1} + \inf(I_{n,y_1})\big)\big(P_{n,y_2} + \inf(I_{n,y_2})\big) \\
&= P_{n,y_1}P_{n,y_2} + P_{n,y_1}\inf(I_{n,y_2}) + P_{n,y_2}\inf(I_{n,y_1}) + \inf(I_{n,y_1})\inf(I_{n,y_2}).
\end{aligned}$$

Diese Schranken der Menge (2.17) sind aufgrund des Produkts $P_{n,y_1}P_{n,y_2}$ Polynome vom Grad höchstens $2n$ und sie unterscheiden sich im Allgemeinen in mehr als nur dem konstanten Term. Die Ergebnismenge kann also nicht wie bei Addition und Subtraktion durch ein Taylor-Modell exakt dargestellt werden. Betrachten wir den allgemeinen Fall, bei dem die Schranken von $[\![P_{n,y_1}, I_{n,y_1}]\!]$ und $[\![P_{n,y_2}, I_{n,y_2}]\!]$ auf $\boldsymbol{D}$ mehrmals das Vorzeichen wechseln, können Ober- und Unterfunktion der Menge (2.17) nur noch abschnittsweise angegeben werden. Auf ganz $\boldsymbol{D}$ betrachtet sind die Schranken dann stückweise Polynome und stetig.

2.2.1.3 Eigenschaften

Die bisher definierten Grundoperationen bilden das Fundament für alles weitere. Wir wollen deshalb an dieser Stelle einige Eigenschaften der definierten Operationen auflisten und, wenn nicht schon in anderer Literatur geschehen, auch beweisen.

Satz 7 (Inklusionsmonotonie)
Seien $[\![P_{n,y_i}, I_{n,y_i}]\!] \in \mathcal{TM}_n(\boldsymbol{D})$, $i = 1, \ldots, 4$, *mit*

$$[\![P_{n,y_1}, I_{n,y_1}]\!] \subseteq [\![P_{n,y_3}, I_{n,y_3}]\!],$$
$$[\![P_{n,y_2}, I_{n,y_2}]\!] \subseteq [\![P_{n,y_4}, I_{n,y_4}]\!],$$

und $\circ \in \{+, -\}$ *eine Verknüpfung. Dann gilt:*

$$[\![P_{n,y_1}, I_{n,y_1}]\!] \boxed{\circ} [\![P_{n,y_2}, I_{n,y_2}]\!] \subseteq [\![P_{n,y_3}, I_{n,y_3}]\!] \boxed{\circ} [\![P_{n,y_4}, I_{n,y_4}]\!]. \tag{2.23}$$

Gilt zusätzlich noch $P_{n,y_1} = P_{n,y_3}$ *und* $P_{n,y_2} = P_{n,y_4}$, *dann ist* (2.23) *auch für die Multiplikation* $\boxed{\cdot}$ *erfüllt, d. h. es gilt*

$$[\![P_{n,y_1}, I_{n,y_1}]\!] \boxed{\cdot} [\![P_{n,y_2}, I_{n,y_2}]\!] \subseteq [\![P_{n,y_3}, I_{n,y_3}]\!] \boxed{\cdot} [\![P_{n,y_4}, I_{n,y_4}]\!]. \tag{2.24}$$

Beweis: Es gilt

$$\begin{aligned}[\![P_{n,y_1}, I_{n,y_1}]\!] \circ_{\mathcal{P}} [\![P_{n,y_2}, I_{n,y_2}]\!] &\subseteq [\![P_{n,y_3}, I_{n,y_3}]\!] \circ_{\mathcal{P}} [\![P_{n,y_2}, I_{n,y_2}]\!] \\ &\subseteq [\![P_{n,y_3}, I_{n,y_3}]\!] \circ_{\mathcal{P}} [\![P_{n,y_4}, I_{n,y_4}]\!],\end{aligned}$$

woraus die Behauptung im Falle von Addition und Subtraktion leicht aus der Definition von $\boxplus$ und $\boxminus$ folgt. Im Falle der Multiplikation folgt die Behauptung aus der Inklusionsmonotonie der Intervalloperationen, denn wegen

$$P_{n,y_1y_2} + P_{>n} = P_{n,y_1} P_{n,y_2} = P_{n,y_3} P_{n,y_4} = P_{n,y_3y_4} + P_{>n}$$

erhält man die Behauptung mit $I_{n,y_1} \subseteq I_{n,y_3}$ und $I_{n,y_2} \subseteq I_{n,y_4}$ sowie den Definitionen (2.20) und (2.21) von I_{n,y_1y_2} und I_{n,y_2y_1} bzw. I_{n,y_3y_4} und I_{n,y_4y_3}. □

Kehren wir noch einmal zur Behauptung (2.24) und der Multiplikation $\boxed{\cdot}$ zurück. Das nachfolgende Beispiel 2.3 zeigt, dass die Inklusionsmonotonie bei voneinander verschiedenen Polynomen $P_{n,y_1} \neq P_{n,y_3}$ oder $P_{n,y_2} \neq P_{n,y_4}$ im Allgemeinen nicht gilt. Jede der Mengen

$$[\![P_{n,y_1}, I_{n,y_1}]\!] \boxed{\cdot} [\![P_{n,y_2}, I_{n,y_2}]\!]$$
$$[\![P_{n,y_3}, I_{n,y_3}]\!] \boxed{\cdot} [\![P_{n,y_4}, I_{n,y_4}]\!]$$

enthält aber das Ergebnis der Mengenmultiplikation

$$[\![P_{n,y_1}, I_{n,y_1}]\!] \cdot_{\mathcal{P}} [\![P_{n,y_2}, I_{n,y_2}]\!].$$

Beispiel 2.3 *Sei* $n = 1$, $D = [-1, 1]$ *und* $s_0 = 0$ *sowie*

$$P_{1,y_1}(s) = -\frac{s}{2} - \frac{1}{2},\ I_{1,y_1} = [0, 1], \quad P_{1,y_3}(s) = -\frac{s}{2},\ I_{1,y_3} = [-\frac{1}{2}, \frac{1}{2}],$$

$$P_{1,y_2}(s) = \frac{s}{2} + \frac{1}{2},\ I_{1,y_2} = [-1, 0], \quad P_{1,y_4}(s) = \frac{s}{2},\ I_{1,y_4} = [-\frac{1}{2}, \frac{1}{2}].$$

Die Voraussetzungen

$$[\![P_{1,y_1}, I_{1,y_1}]\!] \subseteq [\![P_{1,y_3}, I_{1,y_3}]\!]$$

$$[\![P_{1,y_2}, I_{1,y_2}]\!] \subseteq [\![P_{1,y_4}, I_{1,y_4}]\!]$$

sind erfüllt. Wie man leicht sieht, gilt in beiden Fällen sogar Gleichheit. Bilden wir jedoch die Produkte wie in (2.24) *aufgeführt, dann erhält man*

$$(P_{1,y_1} P_{1,y_2})(s) = -\frac{s^2}{4} - \frac{s}{2} - \frac{1}{4} \ \Rightarrow\ P_{1,y_1y_2}(s) = -\frac{s}{2} - \frac{1}{4},\ P^{(1)}_{>1}(s) = -\frac{s^2}{4},$$

$$(P_{1,y_3} P_{1,y_4})(s) = -\frac{s^2}{4} \qquad \Rightarrow\ P_{1,y_3y_4}(s) = 0,\ P^{(2)}_{>1}(s) = -\frac{s^2}{4},$$

und wegen

$$W(P^{(1)}_{>1}; D) = [-\frac{1}{4}, 0] = W(P^{(2)}_{>1}; D),$$

$$W(P_{1,y_3}; D) = [-\frac{1}{2}, \frac{1}{2}] = W(P_{1,y_4}; D),$$

$$W(P_{1,y_1}; D) = [-1, 0],$$

$$W(P_{1,y_2}; D) = [0, 1]$$

die Intervalle

$$\begin{aligned} I_{1,y_1y_2} &= [-\frac{1}{4}, 0] + [-1, 0] \cdot [-1, 0] + [0, 1] \cdot \big([0, 1] + [-1, 0]\big) \\ &= [-\frac{1}{4}, 0] + [0, 1] + [-1, 1] \\ &= [-\frac{5}{4}, 2] = I_{1,y_2y_1}, \end{aligned}$$

$$\begin{aligned} I_{1,y_3y_4} &= [-\frac{1}{4}, 0] + [-\frac{1}{2}, \frac{1}{2}] \cdot [-\frac{1}{2}, \frac{1}{2}] + [-\frac{1}{2}, \frac{1}{2}] \cdot \big([-\frac{1}{2}, \frac{1}{2}] + [-\frac{1}{2}, \frac{1}{2}]\big) \\ &= [-\frac{1}{4}, 0] + [-\frac{1}{4}, \frac{1}{4}] + [-\frac{1}{2}, \frac{1}{2}] \\ &= [-1, \frac{3}{4}] = I_{1,y_4y_3}. \end{aligned}$$

Damit ist

$$[\![P_{1,y_1}, I_{1,y_1}]\!] \boxdot [\![P_{1,y_2}, I_{1,y_2}]\!] = [\![P_{1,y_1y_2}, I_{1,y_1y_2} \cap I_{1,y_2y_1}]\!]$$
$$[\![P_{1,y_3}, I_{1,y_3}]\!] \boxdot [\![P_{1,y_4}, I_{1,y_4}]\!] = [\![P_{1,y_3y_4}, I_{1,y_3y_4} \cap I_{1,y_4y_3}]\!]$$

und es gilt z. B. für $\overline{p}_{1,y_1y_2}(s) = -\frac{s}{2} - \frac{1}{4} + 2$ *und* $\overline{p}_{1,y_3y_4}(s) = \frac{3}{4}$

$$\overline{p}_{1,y_1y_2}(s) > \overline{p}_{1,y_3y_4}(s) \quad \forall s \in [-1,1]$$

d. h.

$$\overline{p}_{1,y_1y_2} \notin [\![P_{1,y_3y_4}, I_{1,y_3y_4}]\!].$$

Die Inklusion (2.24) *gilt in diesem Beispiel also nicht, obwohl die Funktionenmengen identisch sind.*

Satz 8
Seien $[\![P_{n,y_i}, I_{n,y_i}]\!] \in \mathcal{TM}_n(\boldsymbol{D})$, $i = 1, 2, 3$, *dann gilt:*

a) Kommutativität:

$$[\![P_{n,y_1}, I_{n,y_1}]\!] \boxplus [\![P_{n,y_2}, I_{n,y_2}]\!] = [\![P_{n,y_2}, I_{n,y_2}]\!] \boxplus [\![P_{n,y_1}, I_{n,y_1}]\!],$$
$$[\![P_{n,y_1}, I_{n,y_1}]\!] \boxdot [\![P_{n,y_2}, I_{n,y_2}]\!] = [\![P_{n,y_2}, I_{n,y_2}]\!] \boxdot [\![P_{n,y_1}, I_{n,y_1}]\!];$$

b) Assoziativität:

$$\big([\![P_{n,y_1}, I_{n,y_1}]\!] \boxplus [\![P_{n,y_2}, I_{n,y_2}]\!]\big) \boxplus [\![P_{n,y_3}, I_{n,y_3}]\!] = [\![P_{n,y_1}, I_{n,y_1}]\!] \boxplus \big([\![P_{n,y_2}, I_{n,y_2}]\!] \boxplus [\![P_{n,y_3}, I_{n,y_3}]\!]\big);$$

c) Subdistributivität:

$$[\![P_{n,y_1}, I_{n,y_1}]\!] \boxdot \big([\![P_{n,y_2}, I_{n,y_2}]\!] \boxplus [\![P_{n,y_3}, I_{n,y_3}]\!]\big) \subseteq [\![P_{n,y_1}, I_{n,y_1}]\!] \boxdot [\![P_{n,y_2}, I_{n,y_2}]\!] \boxplus [\![P_{n,y_1}, I_{n,y_1}]\!] \boxdot [\![P_{n,y_3}, I_{n,y_3}]\!].$$

Beweis: Die Assoziativität und die Kommutativität ergeben sich leicht aus den Definitionen von $\boxplus$ und $\boxdot$ [vgl. auch 42, S. 83-84]). Die Subdistributivität erhält man ebenfalls aus den Definitionen mit der Subdistributivität für Intervalle. Allerdings hat man deutlich mehr zu schreiben als bei a) und b) [vgl. 42, S. 84]. □

2.2.1.4 Polynome

Mit den bisher definierten Grundoperationen $\boxplus$, $\boxminus$ und $\boxdot$ können nun beliebige, aus diesen Verknüpfungen aufgebaute Funktionsausdrücke mit Elementen aus $\mathcal{TM}_n(\boldsymbol{D})$ ausgewertet werden. Mit Blick auf den nächsten Abschnitt 2.2.2 betrachten wir hier speziell Polynome p der Gestalt

$$p(x) = \sum_{i=0}^{k} c_i\, x^i, \quad x \in \mathbb{R},\ c_i \in \mathbb{R}.$$

Da wir in der vorliegenden Arbeit alle Polynome einer Veränderlichen mit dem Horner-Schema auswerten wollen, schreiben wir

$$p(x) = c_0 + x \cdot (c_1 + x \cdot (c_2 + x \cdot (\ldots (c_{k-1} + x \cdot c_k) \ldots))).$$

Ersetzen wir in p die arithmetischen Operationen durch die entsprechenden Operationen aus der Potenzmenge $\mathcal{P}C(\boldsymbol{D})$, dann stellt für $\mathcal{Y} \in \mathcal{P}C(\boldsymbol{D})$ die Abbildung

$$p_{\mathcal{P}} : \mathcal{Y} \mapsto \mathcal{C}_0 +_{\mathcal{P}} \mathcal{Y} \cdot_{\mathcal{P}} (\mathcal{C}_1 +_{\mathcal{P}} \mathcal{Y} \cdot_{\mathcal{P}} (\mathcal{C}_2 +_{\mathcal{P}} \mathcal{Y} \cdot_{\mathcal{P}} (\ldots (\mathcal{C}_{k-1} +_{\mathcal{P}} \mathcal{Y} \cdot_{\mathcal{P}} \mathcal{C}_k) \ldots)))$$

eine Repräsentation von p auf der Potenzmenge $\mathcal{P}C(\boldsymbol{D})$ dar, wobei die Menge $\mathcal{C}_i \in \mathcal{P}C(\boldsymbol{D})$, $i = 1, \ldots, k$, jeweils nur die konstante Funktion $y(s,t) \equiv c_i$ enthält. Nach Definition 5 ist $p_{\mathcal{P}}(\mathcal{Y})$ die Menge

$$p_{\mathcal{P}}(\mathcal{Y}) = \big\{ p(y) \,|\, y \in \mathcal{Y} \big\}.$$

Wir weisen darauf hin, dass für $y \in C(\boldsymbol{D})$ die Funktion $p(y)$ eine stetige Funktion von zwei Veränderlichen ist.

Auf analoge Weise erklären wir p auf dem Teilsystem $\mathcal{TM}_n(\boldsymbol{D})$ von $\mathcal{P}C(\boldsymbol{D})$. Wir ersetzen die arithmetischen Operationen durch $\boxplus$ und $\boxdot$ und die Koeffizienten c_i durch die Mengen

$$[\![P_{n,c_i}, [0,0]]\!] \text{ mit } P_{n,c_i}(s - s_0, t - t_0) = c_i, \quad i = 1, \ldots, k.$$

Damit ist für $[\![P_{n,y}, I_{n,y}]\!] \in \mathcal{TM}_n(\boldsymbol{D})$ die Abbildung

$$\begin{aligned} p_{\boxdot} : [\![P_{n,y}, I_{n,y}]\!] \mapsto\ & [\![P_{n,c_0}, [0,0]]\!] \boxplus [\![P_{n,y}, I_{n,y}]\!] \boxdot \Big([\![P_{n,c_1}, [0,0]]\!] \boxplus [\![P_{n,y}, I_{n,y}]\!] \\ & \boxdot \Big([\![P_{n,c_2}, [0,0]]\!] \boxplus [\![P_{n,y}, I_{n,y}]\!] \boxdot \big(\ldots \big([\![P_{n,c_{k-1}}, [0,0]]\!] \\ & \boxplus [\![P_{n,y}, I_{n,y}]\!] \boxdot [\![P_{n,c_k}, [0,0]]\!] \big) \ldots \big) \Big) \Big) \end{aligned}$$

eine Repräsentation des Polynoms p auf der Menge $\mathcal{TM}_n(\boldsymbol{D})$. Das Resultat $p_\square([\![P_{n,y}, I_{n,y}]\!])$ ist aufgrund der Definition der Verknüpfungen wieder ein Element aus $\mathcal{TM}_n(\boldsymbol{D})$ und es gilt mit Satz 7 für alle Funktionen $\tilde{y} \in [\![P_{n,y}, I_{n,y}]\!]$

$$p(\tilde{y}) \in p_\mathcal{P}([\![P_{n,y}, I_{n,y}]\!]) \subseteq p_\square([\![P_{n,y}, I_{n,y}]\!]). \tag{2.25}$$

Sobald wir in den nächsten Abschnitten die Division, einige Standardfunktionen sowie die Integration auf $\mathcal{TM}_n(\boldsymbol{D})$ erklärt haben, können wir durch die hier dargestellte Vorgehensweise auch allgemeinere Funktionsausdrücke in der Menge $\mathcal{TM}_n(\boldsymbol{D})$ auswerten. Wir kennzeichnen solche Funktionen bzw. Funktionsausdrücke mit einem $\square$ im Index, so wie wir es hier mit dem Polynom p gemacht haben. Wir nennen die Auswertung eines Ausdrucks in $\mathcal{TM}_n(\boldsymbol{D})$ im Folgenden Taylor-Modell-Auswertung. Speziell für Polynome mit zwei statt einer Veränderlichen demonstrieren wir die Taylor-Modell-Auswertung in zwei Beispielen am Ende dieses Abschnitts.

Wir vereinbaren an dieser Stelle, dass wir eine reelle Zahl $c \in \mathbb{R}$, die innerhalb der Taylor-Modell-Auswertung als Operand bzgl. $\boxplus$, $\boxminus$, $\boxdot$ oder $\boxslash$ auftritt, mit dem Element $[\![P_{n,c}, [0,0]]\!] \in \mathcal{TM}_n(\boldsymbol{D})$ mit $P_{n,c}(s - s_0, t - t_0) = c$ identifizieren. Damit aber die weiteren Ausführungen übersichtlich bleiben, schreiben wir ($\circ \in \{+, -, \cdot, /\}$)

$$c \boxcirc [\![P_{n,y}, I_{n,y}]\!] \quad \text{anstelle von} \quad [\![P_{n,c}, [0,0]]\!] \boxcirc [\![P_{n,y}, I_{n,y}]\!]$$

bzw.

$$[\![P_{n,y}, I_{n,y}]\!] \boxcirc c \quad \text{anstelle von} \quad [\![P_{n,y}, I_{n,y}]\!] \boxcirc [\![P_{n,c}, [0,0]]\!].$$

Bevor wir nun diesen Abschnitt mit den Beispielen schließen, betrachten wir die Menge $p_\square([\![P_{n,y}, I_{n,y}]\!])$ etwas genauer. $p_\square([\![P_{n,y}, I_{n,y}]\!])$ ist, wie oben erwähnt, ein Element aus $\mathcal{TM}_n(\boldsymbol{D})$, d. h. die Menge wird durch ein Taylorpolynom und ein Intervall dargestellt. Nun gilt: Sind P_{n,y_1} und P_{n,y_2} zwei Taylorpolynome von Funktionen y_1, $y_2 \in C^{n+1}(\boldsymbol{D})$, dann erhält man aus

$$c_1 \cdot P_{n,y_1} + c_2 \cdot P_{n,y_2}, \quad c_1, c_2 \in \mathbb{R},$$

das Taylorpolynom $P_{n,c_1y_1+c_2y_2}$ der Funktion $c_1y_1 + c_2y_2$. Ebenso resultiert aus dem Produkt

$$P_{n,y_1} \cdot P_{n,y_2} = P_{n,y_1y_2} + P_{>n}$$

das Taylorpolynom von y_1y_2. Das bedeutet aber, dass sich das Taylorpolynom der Funktion $p(y)$ mit $y \in C^{n+1}(\boldsymbol{D})$ aus

$$p(P_{n,y}) = P_{n,p(y)} + P_{>n} \tag{2.26}$$

ergibt. Das Polynom $P_{>n}$ enthält dabei alle Terme mit Ordnung größer als n.

Kommen wir zurück zu unserer Menge $p_\square([\![P_{n,y}, I_{n,y}]\!])$. Aufgrund der Definition (2.15) und (2.22) der Verknüpfungen $\boxplus$ und $\boxdot$ sowie dem eben Gesagten erhalten wir aus der Taylor-Modell-Auswertung von $p_\square$ mit $[\![P_{n,y}, I_{n,y}]\!]$ die Darstellung

$$p_\square([\![P_{n,y}, I_{n,y}]\!]) = [\![P_{n,p(y)}, I_{n,p(y)}]\!]$$

mit dem Taylorpolynom $P_{n,p(y)}$, das sich aus (2.26) ergibt, und einer Fehlereinschließung $I_{n,p(y)}$, die unter anderem den Wertebereich $W(P_{>n})$ von $P_{>n}$ aus (2.26) enthält.

Wir halten fest, dass die Taylor-Modell-Auswertung eines Polynoms p als Ergebnis ein Taylor-Modell $[\![P_{n,p(y)}, I_{n,p(y)}]\!]$ aus $\mathcal{TM}_n(\boldsymbol{D})$ liefert, das sich aus dem Taylorpolynom von $p(y)$ und einer Fehlereinschließung $I_{n,p(y)}$ zusammensetzt. Das Gleiche gilt auch für Polynome in mehreren Variablen.

Es folgen die Beispiele zur Taylor-Modell-Auswertung von Polynomen mit zwei Veränderlichen.

Beispiel 2.4 *Sei* $n = 1$, $\boldsymbol{D} = [0,1] \times [0,1]$ *und* $s_0 = t_0 = \frac{1}{2}$. *Wir betrachten das Polynom* p *mit*

$$p(s,t) = s - st.$$

Taylor-Modelle der Ordnung 1 für die Funktionen $y_1(s,t) = s$ *und* $y_2(s,t) = t$ *sind (vgl. Beispiel 2.1)*

$$[\![P_{1,s}, [0,0]]\!] \text{ mit } P_{1,s}(s - \frac{1}{2}, t - \frac{1}{2}) = \frac{1}{2} + (s - \frac{1}{2}),$$

$$[\![P_{1,t}, [0,0]]\!] \text{ mit } P_{1,t}(s - \frac{1}{2}, t - \frac{1}{2}) = \frac{1}{2} + (t - \frac{1}{2}).$$

Ein Taylor-Modell erster Ordnung für p *erhalten wir nun aus*

$$p_\square([\![P_{1,s}, [0,0]]\!], [\![P_{1,t}, [0,0]]\!]) = [\![P_{1,s}, [0,0]]\!] \boxminus [\![P_{1,s}, [0,0]]\!] \boxdot [\![P_{1,t}, [0,0]]\!],$$

also durch Taylor-Modell-Auswertung von $p_\square$. *Zunächst berechnen wir das Produkt* $[\![P_{1,s}, [0,0]]\!] \boxdot [\![P_{1,t}, [0,0]]\!]$ *nach Definition* (2.22). *Es ist*

$$(P_{1,s}P_{1,t})(s - \frac{1}{2}, t - \frac{1}{2}) = \underbrace{\frac{1}{4} + \frac{1}{2}(s - \frac{1}{2}) + \frac{1}{2}(t - \frac{1}{2})}_{=P_{1,st}(s-\frac{1}{2}, t-\frac{1}{2})} + \underbrace{(s - \frac{1}{2})(t - \frac{1}{2})}_{=P_{>1}(s-\frac{1}{2}, t-\frac{1}{2})}.$$

Der Wertebereich von $P_{>1}$ *ist* $[-\frac{1}{4}, \frac{1}{4}]$. *Der Wertebereich von* $P_{1,s}$ *sowie von* $P_{1,t}$ *ist* $[0,1]$. *Die Fehlereinschließungen* $I_{1,st}$ *und* $I_{1,ts}$ *ergeben sich zu*

$$I_{1,st} = I_{1,ts} = W(P_{>1}) = [-\frac{1}{4}, \frac{1}{4}].$$

Damit gilt

$$[\![P_{1,s}, [0,0]]\!] \boxdot [\![P_{1,t}, [0,0]]\!] = [\![P_{1,st}, I_{1,st} \cap I_{1,ts}]\!],$$

mit

$$P_{1,st}(s-\frac{1}{2}, t-\frac{1}{2}) = \frac{1}{4} + \frac{1}{2}(s-\frac{1}{2}) + \frac{1}{2}(t-\frac{1}{2}), \quad I_{1,st} \cap I_{1,ts} = [-\frac{1}{4}, \frac{1}{4}].$$

Schließlich erhält man für p das Taylor-Modell erster Ordnung

$$\begin{aligned} p_{\square}([\![P_{1,s}, [0,0]]\!], [\![P_{1,t}, [0,0]]\!]) &= [\![P_{1,s}, [0,0]]\!] \boxminus [\![P_{1,s}, [0,0]]\!] \boxdot [\![P_{1,t}, [0,0]]\!] \\ &= [\![P_{1,s}, [0,0]]\!] \boxminus [\![P_{1,st}, [-\frac{1}{4}, \frac{1}{4}]]\!] \\ &= [\![P_{1,s-st}, [-\frac{1}{4}, \frac{1}{4}]]\!], \end{aligned}$$

mit

$$\begin{aligned} P_{1,s-st}(s-\frac{1}{2}, t-\frac{1}{2}) &= P_{1,s}(s-\frac{1}{2}, t-\frac{1}{2}) - P_{1,st}(s-\frac{1}{2}, t-\frac{1}{2}) \\ &= \left(\frac{1}{2} + (s-\frac{1}{2})\right) - \left(\frac{1}{4} + \frac{1}{2}(s-\frac{1}{2}) + \frac{1}{2}(t-\frac{1}{2})\right) \\ &= \frac{1}{4} + \frac{1}{2}(s-\frac{1}{2}) - \frac{1}{2}(t-\frac{1}{2}). \end{aligned}$$

Es gilt nach (2.2)

$$p(s,t) \in \frac{1}{4} - \frac{1}{2}(s-\frac{1}{2}) + \frac{1}{2}(t-\frac{1}{2}) + [-\frac{1}{4}, \frac{1}{4}] \quad \forall (s,t) \in \boldsymbol{D} = [0,1] \times [0,1].$$

Beispiel 2.5 *Unter den in Beispiel 2.4 gemachten Voraussetzungen betrachten wir hier ein Polynom p gegeben durch*

$$p(s,t) = \frac{1}{2} + s - 2st.$$

Mit dem Ergebnis für $s \cdot t$ aus Beispiel 2.4 erhält man

$$\begin{aligned} &p_{\square}([\![P_{1,s}, [0,0]]\!], [\![P_{1,t}, [0,0]]\!]) \\ &\quad = \frac{1}{2} \boxplus [\![P_{1,s}, [0,0]]\!] \boxminus 2 \boxdot [\![P_{1,s}, [0,0]]\!] \boxdot [\![P_{1,t}, [0,0]]\!] \\ &\quad = [\![P_{1,\frac{1}{2}+s}, [0,0]]\!] \boxminus [\![P_{1,2st}, [-\frac{1}{2}, \frac{1}{2}]]\!] \\ &\quad = [\![P_{1,\frac{1}{2}+s-2st}, [-\frac{1}{2}, \frac{1}{2}]]\!], \end{aligned}$$

mit

$$P_{1,\frac{1}{2}+s-2st}(s-\frac{1}{2},t-\frac{1}{2})=\frac{1}{2}-(t-\frac{1}{2}).$$

2.2.2 Standardfunktionen

In diesem Abschnitt wollen wir die Grundoperationen mit der Definition der Division vervollständigen und $\boxslash$ gemäß (2.10) definieren, sowie die Arithmetik um die Standardfunktionen (1.4) ergänzen. Den Ausführungen von Berz und Makino folgend, führen wir für eine Standardfunktion φ wie beispielsweise sin, cos oder auch exp, eine Operation $\varphi_\square$ ein, so dass für alle $[\![P_{n,y}, I_{n,y}]\!] \in \mathcal{TM}_n(\boldsymbol{D})$

$$\varphi_\square\big([\![P_{n,y}, I_{n,y}]\!]\big) \supseteq \varphi_\mathcal{P}\big([\![P_{n,y}, I_{n,y}]\!]\big) \tag{2.27}$$

erfüllt wird und $\varphi_\square\big([\![P_{n,y}, I_{n,y}]\!]\big) \in \mathcal{TM}_n(\boldsymbol{D})$ ist.

Doch zunächst muss man sich überlegen, wie man ein Taylorpolynom zur Darstellung von $\varphi_\square\big([\![P_{n,y}, I_{n,y}]\!]\big)$ für $[\![P_{n,y}, I_{n,y}]\!] \in \mathcal{TM}_n(\boldsymbol{D})$ geeignet berechnen kann. Wir betrachten das Taylorpolynom $P_{n,\varphi}$ einer Funktion $\varphi \in \mathcal{SF}$. Es handelt sich hierbei um ein Polynom von nur einer Veränderlichen. Basierend auf den Überlegungen des vorherigen Abschnittes 2.2.1.4 ergibt die Berechnung von $P_{n,\varphi}(P_{n,y})$ das Taylorpolynom der Funktion $P_{n,\varphi}(y)$ plus Terme höherer Ordnung. Es gilt also nach (2.26)

$$P_{n,\varphi}(P_{n,y}) = P_{n,P_{n,\varphi}(y)} + P_{>n},$$

wobei $P_{>n}$ alle Terme mit Ordnung größer als n enthält. Weil jetzt aber $P_{n,\varphi}$ das Taylorpolynom von φ ist, liegt es nahe zu fragen, unter welchen Bedingungen

$$P_{n,P_{n,\varphi}(y)} = P_{n,\varphi(y)}$$

gilt. Die Antwort auf diese Frage wollen wir mit nachfolgendem Beispiel motivieren.

Beispiel 2.6 *Wir betrachten die Funktionen $\varphi(s) := e^s$ und $y(s) := 2\cos s$ und möchten von $\varphi(y)$ das Taylorpolynom vom Grad 2 entwickelt bei $s_0 = 0$ berechnen. Es ist $P_{2,y}(s) = 2 - s^2$ und wir erhalten*

$$P_{2,\varphi}(P_{2,y}(s)) = 1 + P_{2,y}(s) + \frac{1}{2}(P_{2,y}(s))^2 = 5 - 3s^2 + \frac{1}{2}s^4.$$

Nun ist aber

$$P_{2,\varphi(y)}(s) = e^2 - e^2 s^2,$$

das Taylorpolynom von $\varphi(y)$ hat also mit dem erhaltenen Ausdruck nichts zu tun. Zur Begründung betrachten wir die Taylorreihe

$$r_\infty(s) = \sum_{k=0}^{\infty} \frac{s^k}{k!}$$

der Exponentialfunktion. Das Taylorpolynom $P_{2,\varphi}$ entspricht der bei $k = 2$ abgebrochenen Reihe r_∞. Demnach liegt die Überlegung nahe, auch die Verkettung $r_\infty(P_{2,y}(s))$ bei $k = 2$ abzubrechen, um $P_{2,\varphi(y)}$ zu erhalten. Setzen wir das Taylorpolynom $P_{2,y}$ in r_∞ ein, dann enthalten aufgrund des konstanten Terms von $P_{2,y}$ auch Terme $(P_{2,y}(s))^k$, $k > 2$, ein konstantes Glied sowie Potenzen in s niedrigerer Ordnung als k. In diesem Fall kann die Taylorreihe der Exponentialfunktion nicht bei $k = 2$ abgebrochen werden, denn auch Terme wie $(P_{2,y}(s))^3$ oder $(P_{2,y}(s))^{56}$ liefern noch einen Beitrag zu Termen niedriger Ordnung.
Um dieses Problem zu beheben, wenden Makino und Berz [45] den folgenden Trick an: Sie zerlegen y in

$$y(s) = y(s_0) + (y(s) - y(s_0)) = 2 + (2\cos s - 2)$$

und verwenden das Taylorpolynom von φ entwickelt bei $y(s_0) = 2$. Mit der Funktion $h(s) := 2\cos s - 2$, die das Taylorpolynom $P_{2,h}(s) = P_{2,y}(s) - y(s_0) = -s^2$ hat, also ein Polynom mit konstantem Term 0, erhalten wir

$$\begin{aligned} P_{2,\varphi}(P_{2,y}(s) - 2) &= e^2 + e^2(P_{2,y}(s) - 2) + \frac{e^2}{2}(P_{2,y}(s) - 2)^2 \\ &= e^2 + e^2(P_{2,h}(s)) + \frac{e^2}{2}(P_{2,h}(s))^2 \\ &= e^2 - e^2 s^2 + \frac{e^2}{2}s^4. \end{aligned}$$

Das Taylorpolynom $P_{2,\varphi(y)}$ ist Teil dieses Ausdrucks. Zur Begründung können wir wieder die Taylorreihe der Exponentialfunktion, diesmal entwickelt bei $y(s_0) = 2$ heranziehen. Diese lautet

$$r_\infty(s; 2) = \sum_{k=0}^{\infty} e^2 \frac{(s-2)^k}{k!}.$$

Setzt man in diese das Polynom $P_{2,y}$ ein, dann liefern die Terme $(P_{2,h}(s))^k$ keinen Beitrag zu Termen mit Ordnung kleiner gleich $2k$, denn der konstante Term von $P_{2,h}$ ist 0. Da die bei $k = 2$ abgebrochene Reihe $r_\infty(s; 2)$ gerade das Taylorpolynom $P_{2,\varphi}$ entwickelt bei $y(s_0) = 2$ ergibt und in diesem Fall mit dem Taylorpolynom $P_{2,\varphi(y)}$ von $\varphi(y)$ übereinstimmt, kann $r_\infty(P_{2,y}(s); 2)$ nach dem dritten Summanden abgebrochen werden, um $P_{2,\varphi(y)}$ zu erhalten.

Nachdem wir an einem Beispiel dargestellt haben, wie im Falle einer Verkettung $\varphi(y)$ das Taylorpolynom berechnet werden kann, wenn nur das Taylorpolynom der inneren Funktion y zur Verfügung steht, diskutieren wir jetzt die Definition der angekündigten Operationen in $\mathcal{TM}_n(\boldsymbol{D})$ am Beispiel der Exponentialfunktion und anschließend an der Division [45][43, S. 67-69][42, S. 85-87].

Es sei dazu $[\![P_{n,y}, I_{n,y}]\!] \in \mathcal{TM}_n(\boldsymbol{D})$ beliebig mit $y \in C^{n+1}(\boldsymbol{D})$, $\tilde{y} \in [\![P_{n,y}, I_{n,y}]\!]$ eine beliebig gewählte Funktion und $c := y(s_0, t_0)$ der Funktionswert von y an der Entwicklungsstelle des Taylorpolynoms $P_{n,y}$. Wir schreiben

$$y(s,t) = c + (y(s,t) - c), \quad h(s,t) := y(s,t) - c, \quad (s,t) \in \boldsymbol{D}. \tag{2.28}$$

Das Taylorpolynom von h

$$P_{n,h} = P_{n,y} - c$$

hat den konstanten Term 0 und es ist $I_{n,h} = I_{n,y}$. Dies bedeutet für das Taylor-Modell $[\![P_{n,h}, I_{n,h}]\!]$

$$[\![P_{n,h}, I_{n,h}]\!] = [\![P_{n,y}, I_{n,y}]\!] \boxminus [\![P_{n,c}, [0,0]]\!]. \tag{2.29}$$

Weiter bezeichne für $x \in \mathbb{R}$

$$\begin{aligned} p(x) &:= P_{n,\exp}(x - c) \\ &= \exp(c) \cdot \left\{ 1 + (x-c) + \frac{1}{2}(x-c)^2 + \ldots + \frac{1}{n!}(x-c)^n \right\} \end{aligned}$$

das Taylorpolynom von exp entwickelt bei c und

$$\begin{aligned} r(x) &:= R_{n,\exp}(x - c) \\ &= \exp(c) \cdot \frac{1}{(n+1)!}(x-c)^{n+1} \exp(\theta \cdot (x-c)), \quad \theta \in (0,1) \end{aligned}$$

das zugehörige Restglied. Es gilt

$$\exp(x) = P_{n,\exp}(x - c) + R_{n,\exp}(x - c), \quad x \in \mathbb{R}.$$

Bevor wir uns nun der Berechnungsvorschrift für $\exp_\square\left([\![P_{n,y}, I_{n,y}]\!]\right)$ widmen, machen wir hier noch einmal auf die folgenden Punkte aufmerksam:

a) die Menge $\exp_\square\left([\![P_{n,y}, I_{n,y}]\!]\right)$ muss ein Element aus $\mathcal{TM}_n(\boldsymbol{D})$ sein,

b) die Komponenten zur Darstellung der Menge sollen aus den Komponenten von $[\![P_{n,y}, I_{n,y}]\!]$ errechnet werden und

c) $\exp_\square\left([\![P_{n,y}, I_{n,y}]\!]\right)$ soll (2.27) erfüllen, was zu

$$\exp(\tilde{y}) \in \exp_\square\left([\![P_{n,y}, I_{n,y}]\!]\right) \qquad \forall \tilde{y} \in [\![P_{n,y}, I_{n,y}]\!] \tag{2.30}$$

äquivalent ist.

Das Vorgehen ist nun denkbar einfach und basiert auf der Beziehung

$$\exp(\tilde{y}) = p(\tilde{y}) + r(\tilde{y}). \tag{2.31}$$

Als erstes berechnet man das Taylor-Modell

$$p_{\square}\big([\![P_{n,y}, I_{n,y}]\!]\big) = [\![P_{n,p(y)}, I_{n,p(y)}]\!]$$

wie in Abschnitt 2.2.1.4 angegeben mit dem Horner-Schema, wobei (2.29) zu beachten ist. Es gilt nach (2.25)

$$p(\tilde{y}) \in [\![P_{n,p(y)}, I_{n,p(y)}]\!]. \tag{2.32}$$

Da das Polynom $P_{n,h}$ den konstanten Term 0 hat und p das Taylorpolynom der Ordnung n von exp entwickelt bei c ist, gilt nach obigen Überlegungen

$$P_{n,p(y)} = P_{n,\exp(y)},$$

d. h. das errechnete Taylorpolynom stimmt mit dem Taylorpolynom von $\exp(y)$ überein. Wir haben also durch die Taylor-Modell-Auswertung von $p_{\square}$ zweierlei erreicht: wir haben ein Element aus $\mathcal{TM}_n(\boldsymbol{D})$ mit (2.32) errechnet und zugleich das Taylorpolynom $P_{n,\exp(y)}$ von $\exp(y)$ erhalten. Da (2.31) und (2.32) gleichbedeutend sind mit

$$\begin{aligned}\exp(\tilde{y}(s,t)) &= p(\tilde{y}(s,t)) + r(\tilde{y}(s,t)) \\ &\in P_{n,\exp(y)}(s - s_0, t - t_0) + I_{n,p(y)} + r(\tilde{y}(s,t)), \quad (s,t) \in \boldsymbol{D},\end{aligned} \tag{2.33}$$

genügt es, eine von $\tilde{y}$ unabhängige Einschließung der Funktion $r(\tilde{y})$ zu berechnen, um $\exp_{\square}\big([\![P_{n,y}, I_{n,y}]\!]\big)$ definieren zu können. Dies folgt aus (2.30) und (2.5).

Wir berechnen also im nächsten Schritt eine Einschließung von $r(\tilde{y})$. Es ist

$$r(\tilde{y}(s,t)) = \frac{\exp(c)}{(n+1)!}\big(\tilde{y}(s,t) - c\big)^{n+1} \exp\big(\theta \cdot (\tilde{y}(s,t) - c)\big)$$

mit $\theta \in (0,1)$, $(s,t) \in \boldsymbol{D}$. Für $\tilde{y}$ gilt

$$\tilde{y}(s,t) \in P_{n,y}(s - s_0, t - t_0) + I_{n,y} \subseteq W(P_{n,y}) + I_{n,y} =: I_y \quad \forall (s,t) \in \boldsymbol{D}.$$

Damit erhalten wir die Einschließung

$$r(\tilde{y}(s,t)) \in \frac{\exp(c)}{(n+1)!}(I_y - c)^{n+1} \exp\big([0,1] \cdot (I_y - c)\big) =: I_{r(y)},$$

die für jedes $\tilde{y} \in [\![P_{n,y}, I_{n,y}]\!]$ und alle $(s,t) \in \boldsymbol{D}$ Gültigkeit besitzt.

Insgesamt ergibt sich aus (2.33) die Inklusionsbeziehung

$$\exp(\tilde{y}(s,t)) \in P_{n,\exp(y)}(s-s_0,t-t_0) + I_{n,p(y)} + I_{r(y)}$$

für alle $(s,t) \in \boldsymbol{D}$ und alle $\tilde{y} \in [\![P_{n,y}, I_{n,y}]\!]$, was nach (2.5) äquivalent ist zu

$$\exp(\tilde{y}) \in [\![P_{n,\exp(y)}, I_{n,\exp(y)}]\!]$$

mit dem Intervall $I_{n,\exp(y)} := I_{n,p(y)} + I_{r(y)}$. Damit wird die Exponentialfunktion $\exp(x)$ auf der Menge $\mathcal{TM}_n(\boldsymbol{D})$ definiert durch

$$\exp_\square\left([\![P_{n,y}, I_{n,y}]\!]\right) := [\![P_{n,\exp(y)}, I_{n,\exp(y)}]\!].$$

Beispiel 2.7 *Unter den Voraussetzungen* $n=1$, $\boldsymbol{D} = [0,1]\times[0,1]$ *und* $s_0 = t_0 = \frac{1}{2}$ *berechnen wir* $\exp_\square\left([\![P_{1,y}, I_{1,y}]\!]\right)$ *mit*

$$P_{1,y}(s-\frac{1}{2},t-\frac{1}{2}) = \frac{1}{2} - (t-\frac{1}{2}), \quad I_{1,y} = [-\frac{1}{2},\frac{1}{2}].$$

Es ist $c = y(s_0,t_0) = P_{1,y}(0,0) = \frac{1}{2}$, *womit sich*

$$P_{1,h}(s-\frac{1}{2},t-\frac{1}{2}) = -(t-\frac{1}{2}), \quad I_{1,h} = I_{1,y}$$

ergibt. Die Auswertung von $p_\square$ *liefert*

$$\begin{aligned} p_\square\big([\![P_{1,y}, I_{1,y}]\!]\big) &= e^{0.5} \boxdot \big\{1 \boxplus [\![P_{1,h}, I_{1,h}]\!]\big\} \\ &= [\![P_{1,e^{0.5}(1+h)}, \left[-0.5\,e^{0.5}, 0.5\,e^{0.5}\right]]\!], \end{aligned}$$

mit

$$\begin{aligned} P_{1,e^{0.5}(1+h)}(s-\frac{1}{2},t-\frac{1}{2}) &= e^{0.5}\cdot\big\{1 + P_{1,h}(s-\frac{1}{2},t-\frac{1}{2})\big\} \\ &= e^{0.5}\cdot\big\{1-(t-\frac{1}{2})\big\} \\ &= e^{0.5} - e^{0.5}(t-\frac{1}{2}) \end{aligned}$$

und $P_{1,\exp(y)} = P_{1,e^{0.5}(1+h)}$. *Mit* $W(P_{1,h}) = [-\frac{1}{2},\frac{1}{2}]$ *berechnet sich* $I_{r(y)}$ *unter Beachtung von* (1.6) *zu*

$$I_{r(y)} = \frac{1}{2}[-1,1]^2 \exp\left([0,1]\cdot[-1,1]\right) = \left[0, 0.5\,e^1\right].$$

Mit der resultierenden Fehlereinschließung erster Ordnung

$$I_{1,\exp(y)} = \left[-0.5\,e^{0.5}, 0.5\,e^{0.5}\right] + \left[0, 0.5\,e^{1}\right] = \left[-0.5\,e^{0.5}, 0.5(e^{1} + e^{0.5})\right]$$

erhalten wir

$$\exp_{\square}\left([\![P_{1,y}, I_{1,y}]\!]\right) = [\![P_{1,\exp(y)}, \left[-0.5\,e^{0.5}, 0.5(e^{1} + e^{0.5})\right]]\!]$$

mit

$$P_{1,\exp(y)}(s - \frac{1}{2}, t - \frac{1}{2}) = e^{0.5} - e^{0.5}(t - \frac{1}{2}).$$

Wenden wir uns nun der Division, wie sie von Berz und Makino vorgenommen wurde, zu. Zur Durchführung der Operation müssen wir sicherstellen, dass auf $\boldsymbol{D}$ verschwindende Funktionen nicht in $[\![P_{n,y}, I_{n,y}]\!]$ enthalten sind, da sonst durch Null dividiert wird. Wir können dies sicher ausschließen, wenn

$$0 \notin P_{n,y}(s - s_0, t - t_0) + I_{n,y}, \quad \forall (s,t) \in \boldsymbol{D}$$

gilt. In diesem Fall ist $c = y(s_0, t_0) = P_{n,y}(0,0) \neq 0$. Bezeichnet

$$\begin{aligned} p(x) &:= P_{n,\frac{1}{x}}(x - c) \\ &= \frac{1}{c} \cdot \left\{1 - \frac{(x-c)}{c} + \frac{(x-c)^2}{c^2} - \ldots + (-1)^n \frac{(x-c)^n}{c^n}\right\} \end{aligned}$$

das Taylorpolynom der Funktion $\dfrac{1}{x}$ vom Grad n entwickelt bei c und

$$\begin{aligned} r(x) &:= R_{n,\frac{1}{x}}(x - c) \\ &= \frac{(-1)^{n+1}}{c} \cdot \frac{(x-c)^{n+1}}{c^{n+1}} \cdot \frac{1}{\left(1 + \theta \cdot \dfrac{(x-c)}{c}\right)^{n+2}}, \quad \theta \in (0,1), \end{aligned}$$

das zugehörige Restglied, dann gilt für $x \neq 0$

$$\frac{1}{x} = P_{n,\frac{1}{x}}(x - c) + R_{n,\frac{1}{x}}(x - c)$$

und folglich

$$\frac{1}{\tilde{y}} = p(\tilde{y}) + r(\tilde{y}).$$

Mit einer Funktion h nach (2.28) ist das Vorgehen zur Berechnung eines Elements aus $\mathcal{TM}_n(\boldsymbol{D})$, das alle Funktionen $\dfrac{1}{\tilde{y}}$ enthält, von nun an analog zur Exponentialfunktion. Man berechnet

$$p_{\square}\left([\![P_{n,y}, I_{n,y}]\!]\right) = [\![P_{n,p(y)}, I_{n,p(y)}]\!],$$

wobei aus demselben Grund wie bei der Exponentialfunktion auch $P_{n,p(y)} = P_{n,\frac{1}{y}}$ gilt, und schließt den Restgliedterm $r(\tilde{y})$ in ein Intervall $I_{r(y)}$ ein. Daraus gewinnt man die noch fehlende Fehlereinschließung $I_{n,\frac{1}{y}}$ als Summe von $I_{n,p(y)}$ und $I_{r(y)}$. Man erhält dann

$$\frac{1}{\tilde{y}} \in [\![P_{n,\frac{1}{y}}, I_{n,\frac{1}{y}}]\!]$$

für alle $\tilde{y} \in [\![P_{n,y}, I_{n,y}]\!]$. Wir definieren damit für $[\![P_{n,y_1}, I_{n,y_1}]\!]$, $[\![P_{n,y_2}, I_{n,y_2}]\!] \in \mathcal{TM}_n(\boldsymbol{D})$:

$$[\![P_{n,y_1}, I_{n,y_1}]\!] \boxslash [\![P_{n,y_2}, I_{n,y_2}]\!] := [\![P_{n,y_1}, I_{n,y_1}]\!] \boxdot [\![P_{n,\frac{1}{y_2}}, I_{n,\frac{1}{y_2}}]\!]. \tag{2.34}$$

Diese an zwei Beispielen demonstrierte Vorgehensweise kann zur Definition weiterer Standardfunktionen auf $\mathcal{TM}_n(\boldsymbol{D})$ verwendet werden. Die entscheidende Stelle ist die Abspaltung der Konstanten c und die eindimensionale Taylorentwicklung der Standardfunktion. Wir werden im Folgenden jeweils die entsprechenden Entwicklungen der restlichen Standardfunktionen $\varphi \in \mathcal{SF}$ – ausgenommen die power-Funktion (1.5) – angeben. Die Berechnung der einschließenden Menge $\varphi_\square\big([\![P_{n,y}, I_{n,y}]\!]\big) \in \mathcal{TM}_n(\boldsymbol{D})$ für $[\![P_{n,y}, I_{n,y}]\!] \in \mathcal{TM}_n(\boldsymbol{D})$, also die Bestimmung des Taylorpolynoms und einer Fehlereinschließung, verläuft dann analog zu den Beispielen. Die folgenden Entwicklungen gehen auf Berz und Makino zurück [45][42, S. 85-91].

Quadratwurzel: Unter der Bedingung

$$P_{n,y}(s - s_0, t - t_0) + I_{n,y} \subset (0, \infty) \quad \forall (s,t) \in \boldsymbol{D}$$

gilt

$$\begin{aligned} \sqrt{y} \;=\; & \sqrt{c} \cdot \left\{ 1 + \frac{1}{2}\frac{h}{c} - \frac{1}{2!\,2^2}\frac{h^2}{c^2} + \ldots + (-1)^{n-1}\frac{(2n-3)!!}{n!\,2^n}\frac{h^n}{c^n} \right\} \\ & + (-1)^n \sqrt{c}\,\frac{(2n-1)!!}{(n+1)!\,2^{n+1}}\frac{h^{n+1}}{c^{n+1}}\frac{1}{\left(1 + \theta \cdot \dfrac{h}{c}\right)^{n+1/2}} \end{aligned}$$

mit $0 < \theta < 1$, wobei

$$k!! := \begin{cases} 1 \cdot 3 \cdot 5 \cdot \ldots \cdot k & \text{für } k \in \mathbb{N} \text{ ungerade,} \\ 1 \cdot 3 \cdot 5 \cdot \ldots \cdot (k-1) & \text{für } k \in \mathbb{N} \text{ gerade.} \end{cases}$$

Multiplikative Inverse der Quadratwurzel: Unter der Bedingung

$$P_{n,y}(s - s_0, t - t_0) + I_{n,y} \subset (0, \infty) \quad \forall (s,t) \in \boldsymbol{D}$$

gilt

$$\begin{aligned} \frac{1}{\sqrt{y}} &= \frac{1}{\sqrt{c}} \cdot \left\{ 1 - \frac{1}{2}\frac{h}{c} + \frac{3!!}{2!\,2^2}\frac{h^2}{c^2} + \ldots + (-1)^n \frac{(2n-1)!!}{n!\,2^n}\frac{h^n}{c^n} \right\} \\ &\quad + (-1)^{n+1}\frac{1}{\sqrt{c}}\frac{(2n+1)!!}{(n+1)!\,2^{n+1}}\frac{h^{n+1}}{c^{n+1}}\frac{1}{\left(1 + \theta \cdot \frac{h}{c}\right)^{n+3/2}} \end{aligned}$$

mit $0 < \theta < 1$.

Logarithmus: Unter der Bedingung

$$P_{n,y}(s - s_0, t - t_0) + I_{n,y} \subset (0, \infty) \quad \forall (s,t) \in \boldsymbol{D}$$

gilt

$$\begin{aligned} \ln(y) &= \ln\left(c \cdot \left(1 + \frac{h}{c}\right)\right) \\ &= \ln(c) + \ln\left(1 + \frac{h}{c}\right) \\ &= \ln(c) + \frac{h}{c} - \frac{1}{2}\frac{h^2}{c^2} + \ldots + (-1)^{n+1}\frac{1}{n}\frac{h^n}{c^n} \\ &\quad + (-1)^{n+2}\frac{1}{n+1}\frac{h^{n+1}}{c^{n+1}}\frac{1}{\left(1 + \theta \cdot \frac{h}{c}\right)^{n+1}} \end{aligned}$$

mit $0 < \theta < 1$.

Sinus:

$$\begin{aligned} \sin(y) &= \sin(c) + \cos(c) \cdot h - \frac{1}{2!}\sin(c) \cdot h^2 - \frac{1}{3!}\cos(c) \cdot h^3 \\ &\quad + \ldots + \frac{1}{(n+1)!}h^{n+1} \cdot \tilde{r} \end{aligned}$$

mit

$$\tilde{r} = \begin{cases} -\tilde{r}_0 & , \quad n \bmod 4 = 1, 2\,, \\ \tilde{r}_0 & , \quad \text{sonst}, \end{cases}$$

$$\tilde{r}_0 = \begin{cases} \cos(c + \theta \cdot h) & , \quad n \text{ gerade}, \\ \sin(c + \theta \cdot h) & , \quad \text{sonst}, \end{cases} \qquad 0 < \theta < 1.$$

Kosinus:

$$\begin{aligned}\cos(y) &= \cos(c) - \sin(c)\cdot h - \frac{1}{2!}\cos(c)\cdot h^2 + \frac{1}{3!}\sin(c)\cdot h^3 \\ &\quad + \ldots + \frac{1}{(n+1)!}h^{n+1}\cdot\tilde{r}\end{aligned}$$

mit

$$\tilde{r} = \begin{cases} -\tilde{r}_0 & , \quad n \bmod 4 = 0,1\ , \\ \tilde{r}_0 & , \quad \text{sonst}, \end{cases}$$

$$\tilde{r}_0 = \begin{cases} \sin(c+\theta\cdot h) & , \quad n \text{ gerade}, \\ \cos(c+\theta\cdot h) & , \quad \text{sonst}, \end{cases} \qquad 0<\theta<1.$$

Hyperbolischer Sinus:

$$\begin{aligned}\sinh(y) &= \sinh(c) + \cosh(c)\cdot h + \frac{1}{2!}\sinh(c)\cdot h^2 + \frac{1}{3!}\cosh(c)\cdot h^3 \\ &\quad + \ldots + \frac{1}{(n+1)!}h^{n+1}\cdot\tilde{r}\end{aligned}$$

mit

$$\tilde{r} = \begin{cases} \cosh(c+\theta\cdot h) & , \quad n \text{ gerade}, \\ \sinh(c+\theta\cdot h) & , \quad \text{sonst}, \end{cases} \qquad 0<\theta<1.$$

Hyperbolischer Kosinus:

$$\begin{aligned}\cosh(y) &= \cosh(c) + \sinh(c)\cdot h + \frac{1}{2!}\cosh(c)\cdot h^2 + \frac{1}{3!}\sinh(c)\cdot h^3 \\ &\quad + \ldots + \frac{1}{(n+1)!}h^{n+1}\cdot\tilde{r}\end{aligned}$$

mit

$$\tilde{r} = \begin{cases} \sinh(c+\theta\cdot h) & , \quad n \text{ gerade}, \\ \cosh(c+\theta\cdot h) & , \quad \text{sonst}, \end{cases} \qquad 0<\theta<1.$$

Entwicklungen weiterer Funktionen wie arcsin, arccos und arctan findet man bei Makino [42, S. 90-91].

2.2.3 Integration

Abschließen wollen wir dieses Kapitel mit der Definition der Integration in $\mathcal{TM}_n(\boldsymbol{D})$ gemäß (2.11) bezüglich einer Variablen. Für die Herleitung einer geeigneten Berechnungsvorschrift sei wie in den vorigen Abschnitten $[\![P_{n,y}, I_{n,y}]\!] \in \mathcal{TM}_n(\boldsymbol{D})$ beliebig mit $y \in C^{n+1}(\boldsymbol{D})$ und $\tilde{y} \in [\![P_{n,y}, I_{n,y}]\!]$ eine beliebig gewählte Funktion. Wir betrachten hier die unbestimmte Integration bzgl. der Variablen s und definieren

$$Y(s,t) := \int_{s_0}^{s} y(\sigma,t)\,d\sigma, \quad \tilde{Y}(s,t) := \int_{s_0}^{s} \tilde{y}(\sigma,t)\,d\sigma. \tag{2.35}$$

Das Ziel soll sein, ein Taylor-Modell aus $\mathcal{TM}_n(\boldsymbol{D})$ zu konstruieren, welches die Menge

$$\int_{\mathcal{P}} [\![P_{n,y}, I_{n,y}]\!]\,ds = \Big\{ \int_{s_0}^{s} \tilde{y}(\sigma,t)\,d\sigma \;\Big|\; \tilde{y} \in [\![P_{n,y}, I_{n,y}]\!] \Big\} \tag{2.36}$$

einschließt. Sei dazu im Folgenden $(s,t) \in \boldsymbol{D} = [\underline{s}, \overline{s}] \times [\underline{t}, \overline{t}]$. Wir beginnen mit der Integration des Taylorpolynoms $P_{n,y}$. Offensichtlich gilt

$$\int_{s_0}^{s} P_{n,y}(\sigma - s_0, t - t_0)\,d\sigma = P_{n+1,Y}(s - s_0, t - t_0)$$

mit dem Taylorpolynom $P_{n+1,Y}$ vom Grad $n+1$ der Stammfunktion Y aus (2.35). Wir schreiben $P_{n+1,Y} = P_{n,Y} + P_{>n}$ mit dem Polynom $P_{>n} := P_{n+1,Y} - P_{n,Y}$ bestehend aus allen Termen der Ordnung $n+1$ von $P_{n+1,Y}$.

Damit erhalten wir für $\tilde{y}$

$$\begin{aligned}
&\int_{s_0}^{s} \tilde{y}(\sigma,t)\,d\sigma - P_{n,Y}(s - s_0, t - t_0) \\
&\qquad = \int_{s_0}^{s} \tilde{y}(\sigma,t)\,d\sigma - P_{n+1,Y}(s - s_0, t - t_0) + P_{>n}(s - s_0, t - t_0) \\
&\qquad = \int_{s_0}^{s} \big(\tilde{y}(\sigma,t) - P_{n,y}(\sigma - s_0, t - t_0)\big)\,d\sigma + P_{>n}(s - s_0, t - t_0).
\end{aligned} \tag{2.37}$$

Es ist weiter

$$\int_{s_0}^{s} \big(\tilde{y}(\sigma,t) - P_{n,y}(\sigma - s_0, t - t_0)\big)\, d\sigma$$

$$\begin{cases} \leqslant & \displaystyle\int_{s_0}^{s} \sup(I_{n,y})\, d\sigma \;=\; \sup(I_{n,y}) \cdot (s - s_0) \\ \geqslant & \displaystyle\int_{s_0}^{s} \inf(I_{n,y})\, d\sigma \;=\; \inf(I_{n,y}) \cdot (s - s_0) \end{cases},$$

also

$$\int_{s_0}^{s} \big(\tilde{y}(\sigma,t) - P_{n,y}(\sigma - s_0, t - t_0)\big)\, d\sigma \in I_{n,y} \cdot (s - s_0).$$

Wir erhalten aus (2.37)

$$\begin{aligned} \int_{s_0}^{s} \tilde{y}(\sigma,t)\, d\sigma - P_{n,Y}(s - s_0, t - t_0) &\in I_{n,y} \cdot (s - s_0) + P_{>n}(s - s_0, t - t_0) \\ &\subseteq I_{n,y} \cdot \big([\underline{s}, \overline{s}] - s_0\big) + W(P_{>n}). \end{aligned}$$

Mit dem Intervall

$$I_{n,Y} := I_{n,y} \cdot \big([\underline{s}, \overline{s}] - s_0\big) + W(P_{>n}), \tag{2.38}$$

das eine Fehlereinschließung für Y darstellt, gilt dann

$$\tilde{Y}(s,t) - P_{n,Y}(s - s_0, t - t_0) \in I_{n,Y} \quad \forall (s,t) \in \boldsymbol{D},$$

was nach (2.5) äquivalent ist zu

$$\tilde{Y} \in [\![P_{n,Y}, I_{n,Y}]\!].$$

Hiermit definieren wir die unbestimmte Integration in $\mathcal{TM}_n(\boldsymbol{D})$ bezüglich s als

$$\int\!\!\!\!\square [\![P_{n,y}, I_{n,y}]\!]\, ds := [\![P_{n,Y}, I_{n,Y}]\!]. \tag{2.39}$$

Die Komponenten der Menge $[\![P_{n,Y}, I_{n,Y}]\!]$ berechnen sich also durch Integration des Taylorpolynoms $P_{n,y}$, Abspalten der Terme mit Ordnung $n+1$ vom integrierten Polynom und Berechnen der Fehlereinschließung nach (2.38).

Wir zeigen abschließend, dass diese Definition im Allgemeinen eine echte Obermenge von (2.36) liefert. Für beliebiges $\tilde{y} \in [\![P_{n,y}, I_{n,y}]\!]$ erhalten wir wegen

$$\underline{p}_{n,y} \leqslant \tilde{y} \leqslant \overline{p}_{n,y}$$

die scharfe Abschätzung – Ober- und Unterfunktion gehören jeweils auch noch zur Menge dazu –

$$\begin{aligned}
\int_{s_0}^{s} \tilde{y}(\sigma,t)\,d\sigma &\leqslant \int_{s_0}^{s} \overline{p}_{n,y}(\sigma,t)\,d\sigma \\
&= \int_{s_0}^{s} \big(P_{n,y}(\sigma - s_0, t - t_0) + \sup(I_{n,y})\big)\,d\sigma \\
&= P_{n+1,Y}(s - s_0, t - t_0) + \sup(I_{n,y}) \cdot (s - s_0) \\
\\
\int_{s_0}^{s} \tilde{y}(\sigma,t)\,d\sigma &\geqslant \int_{s_0}^{s} \underline{p}_{n,y}(\sigma,t)\,d\sigma \\
&= \int_{s_0}^{s} \big(P_{n,y}(\sigma - s_0, t - t_0) + \inf(I_{n,y})\big)\,d\sigma \\
&= P_{n+1,Y}(s - s_0, t - t_0) + \inf(I_{n,y}) \cdot (s - s_0).
\end{aligned}$$

Diese Schranken der Menge (2.36) sind Polynome vom Grad $n+1$. Die Schranken von Mengen aus $\mathcal{TM}_n(\boldsymbol{D})$ sind aber Polynome vom Grad n. Die Ergebnismenge kann also nicht durch ein Taylor-Modell exakt dargestellt werden.

Die Integration bezüglich der Variablen t wird auf analogem Wege definiert.

Bemerkung 2.4 Die hier vorgenommene Definition der Integration weicht von der von Makino und Berz [5, 10, 45][42, S. 92-93] angegebenen Definition ab. In diesen Arbeiten werden zuerst alle Terme des Polynoms $P_{n,y}$ mit Ordnung n abgespalten, eine Wertebereichseinschließung davon berechnet und diese zum Intervallterm hinzuaddiert. Erst dann wird integriert. Dies führt in der Regel zu schlechteren Fehlereinschließungen, da die Division durch die um eins erhöhte Potenz der Integrationsvariablen für jeden eingeschlossenen Term ausbleibt.

Beispiel 2.8 *Sei* $n = 1$, $\boldsymbol{D} = [0,1] \times [0,1]$ *und* $s_0 = t_0 = \frac{1}{2}$. *Wir integrieren* $[\![P_{n,y}, I_{n,y}]\!]$ *mit*

$$P_{1,y}(s - \frac{1}{2}, t - \frac{1}{2}) = \frac{1}{4} - \frac{1}{2}(s - \frac{1}{2}) + \frac{1}{2}(t - \frac{1}{2}), \quad I_{1,y} = [-\frac{1}{4}, \frac{1}{4}],$$

aus Beispiel 2.4 bezüglich der Variablen s. Es ist

$$\int_{\frac{1}{2}}^{s} P_{1,y}(\sigma - \frac{1}{2}, t - \frac{1}{2})\, d\sigma = \frac{1}{4}(s - \frac{1}{2}) - \frac{1}{4}(s - \frac{1}{2})^2 + \frac{1}{2}(s - \frac{1}{2})(t - \frac{1}{2}).$$

Die beiden letzten Terme sind von zweiter Ordnung und werden abgespalten. Sie ergeben das Polynom $P_{>1}$. Es gilt

$$W(P_{>1}) = [-\frac{3}{16}, \frac{1}{16}].$$

Wir erhalten damit als Fehlereinschließung

$$I_{1,Y} = [-\frac{3}{16}, \frac{1}{16}] + [-\frac{1}{4}, \frac{1}{4}] \cdot [-\frac{1}{2}, \frac{1}{2}] = [-\frac{5}{16}, \frac{3}{16}].$$

Das resultierende Taylor-Modell lautet

$$\int [\![P_{n,y}, I_{n,y}]\!]\, ds = [\![P_{n,Y}, I_{n,Y}]\!]$$

mit

$$P_{1,Y}(s - \frac{1}{2}, t - \frac{1}{2}) = \frac{1}{4}(s - \frac{1}{2}), \quad I_{1,Y} = [-\frac{5}{16}, \frac{3}{16}].$$

Integriert man $[\![P_{n,y}, I_{n,y}]\!]$ nach der Vorgehensweise von Makino und Berz (vgl. Bemerkung 2.4), dann erhält man die Fehlereinschließung

$$I_{1,Y} = [-\frac{3}{8}, \frac{3}{8}].$$

3 Lösungseinschließung bei gewöhnlichen Differentialgleichungen

Es ist nicht so, daß sie die Lösung nicht sehen. Es ist so, daß sie das Problem nicht sehen.

Gilbert Keith Chesterton (1874-1936)

Wir erläutern in diesem Kapitel die Vorgehensweise zur Lösungseinschließung an Differentialgleichungen mit zwei unbekannten Funktionen. Diese Reduktion auf zwei unbekannte Funktionen ermöglicht den Einsatz von Skizzen zur Veranschaulichung von Lösungsmengen und deren Einschließung. Außerdem wird durch diese Reduktion die Übersichtlichkeit der Darstellung erhöht, wodurch einige wichtige Punkte in der Methode von Makino und Berz, die wir hier vorstellen wollen, deutlicher herausgestellt werden können. Wir wollen damit das Verständnis insgesamt erleichtern. Der Transfer auf Differentialgleichungen mit mehr als zwei Unbekannten ist leicht herzustellen.

3.1 Problemstellung

Wir betrachten für $t \in D_t := [t_0, t_e]$, $t_0 < t_e$, die Anfangswertaufgabe

$$\begin{aligned} y_1'(t) &= f_1\big(y_1(t), y_2(t), t\big), \\ y_2'(t) &= f_2\big(y_1(t), y_2(t), t\big), \\ y_1(t_0) &= \eta_1,\ y_2(t_0) = \eta_2 \end{aligned} \tag{3.1}$$

mit Werten η_1 und η_2, von denen bekannt sei, dass sie in Intervallen I_{η_1} bzw. I_{η_2} liegen. Wir setzen voraus, dass die Funktionen $f_1,\ f_2 : \mathbb{R} \times \mathbb{R} \times D_t \to \mathbb{R}$ als Funktionen mehrerer Veränderlicher n mal (mit $n > 0$) stetig differenzierbar sind. Siehe hierzu auch die Erläuterung in Abschnitt 2.1 zum Begriff der Differenzierbarkeit.

Zwei Funktionen y_1, y_2 bilden eine Lösung von (3.1) auf einem Intervall $[t_0, t_0 + h] \subseteq D_t$, $h > 0$, wenn mit $j = 1, 2$ die Gleichungen $y_j'(t) = f_j\big(y_1(t), y_2(t), t\big)$ für alle $t \in [t_0, t_0 + h]$ und $y_j(t_0) = \eta_j$ erfüllt sind.

Unser Interesse gilt der Menge aller Lösungen der Anfangswertaufgabe (3.1) auf D_t, die man erhält, wenn die Anfangswerte in den gegebenen Intervallen variieren. Im Rahmen einer Lösungseinschließung fragen wir nach einer Einschließungsmenge für diese Menge.

Damit im Folgenden der Text übersichtlich bleibt, gehen wir auf vektorwertige Größen über, die wir durch Fettdruck von den skalaren Größen abheben. Wir schreiben mit

$$\boldsymbol{y} = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}, \quad \boldsymbol{y}' = \begin{pmatrix} y_1' \\ y_2' \end{pmatrix}, \quad \boldsymbol{f} = \begin{pmatrix} f_1 \\ f_2 \end{pmatrix}, \quad \boldsymbol{\eta} = \begin{pmatrix} \eta_1 \\ \eta_2 \end{pmatrix}, \quad \boldsymbol{I}_{\boldsymbol{\eta}} = \begin{pmatrix} I_{\eta_1} \\ I_{\eta_2} \end{pmatrix}$$

das Anfangswertproblem (3.1) in kompakter Form als

$$\boldsymbol{y}'(t) = \boldsymbol{f}\big(\boldsymbol{y}(t), t\big), \quad t \in D_t, \tag{3.2}$$

$$\boldsymbol{y}(t_0) = \boldsymbol{\eta} \in \boldsymbol{I}_{\boldsymbol{\eta}}. \tag{3.3}$$

Eine Lösung der Anfangswertaufgabe (3.1) zum Anfangswert $\boldsymbol{\eta}$ kennzeichnen wir mit dem Symbol $\boldsymbol{\eta}$ im Index. Wir vereinbaren weiter, dass Operationen mit vektorwertigen Größen komponentenweise zu verstehen sind, sofern nichts anderes gesagt wird. Aber auch Begriffe wie z. B. Taylor-Modell oder Taylor-Modell-Auswertung sind bei vektorwertigen Größen komponentenweise aufzufassen. Ein Taylor-Modell einer vektorwertigen Funktion $\boldsymbol{y}$ ist damit ein Vektor bestehend aus Taylor-Modellen der Komponentenfunktionen.

Wie oben erwähnt, suchen wir eine Einschließung der Menge

$$\mathcal{Y}_{D_t} := \big\{ \boldsymbol{y}_{\boldsymbol{\eta}} \,|\, t \in D_t,\ \boldsymbol{\eta} \in \boldsymbol{I}_{\boldsymbol{\eta}} \big\}$$

aller Lösungen $\boldsymbol{y}_{\boldsymbol{\eta}}$ der Anfangswertaufgabe (3.1). Insbesondere sind wir an einer Einschließung der Menge der Funktionswerte

$$\mathcal{W}_{t_e} := \big\{ \boldsymbol{y}_{\boldsymbol{\eta}}(t_e) \,|\, \boldsymbol{\eta} \in \boldsymbol{I}_{\boldsymbol{\eta}} \big\} \tag{3.4}$$

aller Lösungen $\boldsymbol{y}_{\boldsymbol{\eta}}$ von (3.1) an der Stelle t_e interessiert. Wir lösen beide Aufgaben in der vorliegenden Arbeit mit der Methode von Makino und Berz, die wir zur Motivation nachfolgend grob skizzieren und in Abschnitt 3.4 ausführlich vorstellen.

Ausgehend von einer vorgegebenen Schrittweite h_0 mit $0 < h_0 \leqslant t_e - t_0$, berechnet die Methode von Makino und Berz Einschließungen der Mengen

$$\mathcal{Y}_{[t_j, t_{j+1}]} := \big\{ \boldsymbol{y}_{\boldsymbol{\eta}} \,|\, t \in [t_j, t_{j+1}],\ \boldsymbol{\eta} \in \boldsymbol{I}_{\boldsymbol{\eta}} \big\}, \quad j = 0, \ldots, m-1 \tag{3.5}$$

mit Stellen

$$t_0 < t_1 < t_2 < \cdots < t_m := t_e.$$

Die Stellen $t_j := t_{j-1} + h_{j-1}$, $j = 2, \ldots, m-1$, sind a priori nicht bekannt und werden erst während der Durchführung des Algorithmus durch die Berechnung geeigneter Schrittweiten h_{j-1} ermittelt. Als erstes wird eine Einschließung von $\mathcal{Y}_{[t_0,t_1]}$ berechnet. Gegebenenfalls wird hierzu die Schrittweite h_0 vom Verfahren reduziert, die Stelle $t_1 := t_0 + h_0$ ändert sich dementsprechend. Anschließend wird aus der Einschließung von $\mathcal{Y}_{[t_0,t_1]}$ eine Einschließung der Menge der Funktionswerte

$$\mathcal{W}_{t_1} := \{\boldsymbol{y}_{\boldsymbol{\eta}}(t_1) \,|\, \boldsymbol{\eta} \in \boldsymbol{I}_{\boldsymbol{\eta}}\} \tag{3.6}$$

aller Lösungen $\boldsymbol{y}_{\boldsymbol{\eta}}$ von (3.1) an der Stelle t_1 ermittelt. Diese Einschließung fungiert im darauf folgenden Schritt, in dem eine Einschließung der Menge $\mathcal{Y}_{[t_1,t_2]}$ berechnet wird, als Anfangswertmenge der bei t_1 formulierten Anfangswertaufgabe (3.1). Aus der berechneten Einschließung von $\mathcal{Y}_{[t_1,t_2]}$ wird eine Einschließung für

$$\mathcal{W}_{t_2} := \{\boldsymbol{y}_{\boldsymbol{\eta}}(t_2) \,|\, \boldsymbol{\eta} \in \boldsymbol{I}_{\boldsymbol{\eta}}\}$$

gewonnen, die zur Einschließung von $\mathcal{Y}_{[t_2,t_3]}$ verwendet wird usw. So fortfahrend erzielt die Methode von Makino und Berz eine Einschließung aller Lösungen der Anfangswertaufgabe (3.1) auf D_t.

Wir beschreiben in den kommenden Abschnitten die Vorgehensweise zur Lösungseinschließung auf dem Teilintervall $[t_0, t_1]$ und in Abschnitt 3.4.3, wie die Methode von t_1 aus fortgesetzt wird.

Im nächsten Abschnitt stellen wir einige Vorüberlegungen an, um die Vorgehensweise im Verfahren von Makino und Berz für den Leser verständlich zu machen.

3.2 Existenz und Eindeutigkeit von Lösungen

Es sei $\boldsymbol{I} = (I_1, I_2)^{\mathsf{T}} \in \mathbb{IR}^2$ ein Intervall mit

$$\boldsymbol{I}_{\boldsymbol{\eta}} \subset \boldsymbol{I} \quad \text{und} \quad \inf(I_{\eta_j}) \neq \inf(I_j),\ \sup(I_{\eta_j}) \neq \sup(I_j),\ j = 1, 2. \tag{3.7}$$

Aufgrund der Glattheit von f_1 und f_2 gilt für alle

$$\boldsymbol{\xi} = \begin{pmatrix} \xi_1 \\ \xi_2 \end{pmatrix},\ \tilde{\boldsymbol{\xi}} = \begin{pmatrix} \tilde{\xi}_1 \\ \tilde{\xi}_2 \end{pmatrix} \in \boldsymbol{I}$$

und alle $t \in D_t$

$$\left| f_i(\boldsymbol{\xi}, t) - f_i(\tilde{\boldsymbol{\xi}}, t) \right| \leqslant L_i \cdot \left(|\xi_1 - \tilde{\xi}_1| + |\xi_2 - \tilde{\xi}_2| \right), \qquad i = 1, 2 \tag{3.8}$$

mit Konstanten $L_i > 0$. Dies folgt aus dem Mittelwertsatz und der Stetigkeit der partiellen Ableitungen von f_1 und f_2 sowie der Kompaktheit der Menge $\boldsymbol{I}$ [25, S. 516].

Wir definieren zu einem Anfangswert $\boldsymbol{\eta} \in \boldsymbol{I}_{\boldsymbol{\eta}}$ die Zahl

$$b_{\boldsymbol{\eta}} := \min_{j=1,2} \big\{ \sup(I_j) - \eta_j, \eta_j - \inf(I_j) \big\},$$

die wegen (3.7) positiv ist, und betrachten die rechte Seite $\boldsymbol{f}$ auf dem kompakten, achsenparallelen Quader

$$Q_{\boldsymbol{\eta}} := \big\{ (\boldsymbol{\xi}, t) \,\big|\, \|\boldsymbol{\xi} - \boldsymbol{\eta}\|_\infty \leqslant b_{\boldsymbol{\eta}},\ t \in D_t \big\}.$$

$Q_{\boldsymbol{\eta}}$ ist eine Teilmenge von $\boldsymbol{I} \times D_t$. Folglich gilt die Lipschitz-Bedingung (3.8) der Funktion f_1 bzw. f_2 auch auf $Q_{\boldsymbol{\eta}}$. Mit den Zahlen

$$M_{\boldsymbol{\eta}} := \max_{(\boldsymbol{\xi},t) \in Q_{\boldsymbol{\eta}}} \|\boldsymbol{f}(\boldsymbol{\xi}, t)\|_\infty, \quad h_{\boldsymbol{\eta}} := \min \big\{ t_e - t_0, \frac{b_{\boldsymbol{\eta}}}{M_{\boldsymbol{\eta}}} \big\}$$

gilt sodann nach dem Satz von Picard-Lindelöf, dass das Anfangswertproblem (3.1) genau eine auf dem Intervall $[t_0, t_0 + h_{\boldsymbol{\eta}}]$ definierte Lösung besitzt [vgl. 25, S. 516].

Ersetzen wir in der Definition von $h_{\boldsymbol{\eta}}$ den Wert $M_{\boldsymbol{\eta}}$ durch

$$M := \max_{(\boldsymbol{\xi},t) \in \boldsymbol{I} \times D_t} \|\boldsymbol{f}(\boldsymbol{\xi}, t)\|_\infty,$$

so erhalten wir mit

$$\tilde{h}_{\boldsymbol{\eta}} := \min \big\{ t_e - t_0, \frac{b_{\boldsymbol{\eta}}}{M} \big\}$$

ein evtl. kleineres Existenz- und Eindeutigkeitsintervall $[t_0, t_0 + \tilde{h}_{\boldsymbol{\eta}}]$.

Wir fragen als nächstes nach einem gemeinsamen Existenzintervall für die Menge der eindeutigen Lösungen von (3.1), die man erhält, wenn der Anfangswert $\boldsymbol{\eta}$ das Intervall $\boldsymbol{I}_{\boldsymbol{\eta}}$ durchläuft. An der Definition der Größen $b_{\boldsymbol{\eta}}$ und $\tilde{h}_{\boldsymbol{\eta}}$ sieht man leicht, dass diese stetig von $\boldsymbol{\eta}$ abhängen. Demzufolge existiert ein kleinster Wert

$$\hat{h} := \min_{\boldsymbol{\eta} \in \boldsymbol{I}_{\boldsymbol{\eta}}} \big\{ \tilde{h}_{\boldsymbol{\eta}} \big\},$$

der wegen der Positivität aller $b_{\boldsymbol{\eta}}$ bzw. $\tilde{h}_{\boldsymbol{\eta}}$ selbst auch positiv ist. Es ist also $[t_0, t_0 + \hat{h}]$ ein gemeinsames Existenzintervall für die Menge der eindeutigen Lösungen von (3.1).

Die (eindeutigen) Lösungen der Anfangswertaufgabe (3.1) hängen stetig vom Anfangswert $\boldsymbol{\eta}$ ab [25, S. 517]. Wir definieren durch die Zuordnung

$$\boldsymbol{y}^\star : (\eta_1, \eta_2, t) \mapsto \boldsymbol{y}_{\boldsymbol{\eta}}(t) \tag{3.9}$$

eine stetige Funktion auf $\boldsymbol{I}_{\boldsymbol{\eta}} \times [t_0, t_0 + \hat{h}]$, die jedem Tripel (η_1, η_2, t) aus $\boldsymbol{I}_{\boldsymbol{\eta}} \times [t_0, t_0 + \hat{h}]$ den Funktionswert der (eindeutigen) Lösung $\boldsymbol{y}_{\boldsymbol{\eta}}$ an der Stelle t zuordnet. Wir

erinnern nochmal daran, dass wir mit $\boldsymbol{y}_{\boldsymbol{\eta}}$ die Lösung der Anfangswertaufgabe (3.1) zu den Anfangswerten η_1, η_2 bezeichnen.

Die Funktion $\boldsymbol{y}^\star$ wird in der Literatur für gewöhnlich mit *Fluss* bezeichnet [20, S. 97]. Anschaulich gesprochen beschreibt der Fluss $\boldsymbol{y}^\star$ zur Differentialgleichung (3.2), wie die Anfangswertmenge $\boldsymbol{I}_{\boldsymbol{\eta}}$ unter Einwirkung der Differentialgleichung (3.2) in der Phasenebene verschoben und verformt wird. Für einen fest gewählten Zeitpunkt $\hat{t} \in [t_0, t_0 + \hat{h}]$ erhält man aus $\boldsymbol{y}^\star$ eine Funktion

$$\boldsymbol{y}_{\hat{t}}(\eta_1, \eta_2) := \boldsymbol{y}^\star(\eta_1, \eta_2, \hat{t}\,), \quad (\eta_1, \eta_2)^\mathsf{T} \in \boldsymbol{I}_{\boldsymbol{\eta}}$$

der zwei Veränderlichen η_1 und η_2, deren Wertebereich

$$W(\boldsymbol{y}_{\hat{t}}; \boldsymbol{I}_{\boldsymbol{\eta}})$$

das Ergebnis dieser Deformation zum Zeitpunkt $\hat{t}$ darstellt. So ist beispielsweise, die weiter oben eingeführte Menge (3.6) das Ergebnis dieser Einwirkung zum Zeitpunkt t_1. Die Menge (3.6) wird durch $\boldsymbol{y}_{t_1}(\eta_1, \eta_2) := \boldsymbol{y}^\star(\eta_1, \eta_2, t_1)$ beschrieben.

Die Methode von Makino und Berz berechnet im ersten Schritt eine Einschließung der Funktionenmenge $\mathcal{Y}_{[t_0,t_1]}$ mit geeignet gewähltem t_1. Dabei handelt es sich nach Definition des Flusses $\boldsymbol{y}^\star$ um eine Einschließung von $\boldsymbol{y}^\star$ auf $\boldsymbol{I}_{\boldsymbol{\eta}} \times [t_0, t_1]$.

3.3 Taylor-Modelle zur Lösungseinschließung

Eine gängige Praxis zur Lösungseinschließung bei Anfangswertproblemen ist, diese als Fixpunktproblem zu formulieren und mit Hilfe von Fixpunktsätzen eine Einschließung zu bestimmen.

Das Anfangswertproblem (3.1) mit fest gewählten Anfangswerten $\eta_1 \in I_{\eta_1}$, $\eta_2 \in I_{\eta_2}$ ist nach dem Hauptsatz der Differential- und Integralrechnung äquivalent zu dem System von Integralgleichungen

$$\boldsymbol{y}(t) = \boldsymbol{\eta} + \int_{t_0}^{t} \boldsymbol{f}\big(\boldsymbol{y}(\tau), \tau\big)\, d\tau = \begin{pmatrix} \eta_1 + \displaystyle\int_{t_0}^{t} f_1\big(y_1(\tau), y_2(\tau), \tau\big)\, d\tau \\ \eta_2 + \displaystyle\int_{t_0}^{t} f_2\big(y_1(\tau), y_2(\tau), \tau\big)\, d\tau \end{pmatrix}, \tag{3.10}$$

d. h. Lösungen von (3.1) sind Lösungen von (3.10) und umgekehrt. Mit dem Operator

$$\boldsymbol{K}_{\boldsymbol{\eta}} : C(D_t) \times C(D_t) \to C(D_t) \times C(D_t)$$

definiert durch

$$\boldsymbol{K}_{\boldsymbol{\eta}}(\boldsymbol{y})(t) := \boldsymbol{\eta} + \int_{t_0}^{t} \boldsymbol{f}\big(\boldsymbol{y}(\tau), \tau\big)\, d\tau$$

ist das Anfangswertproblem (3.1) äquivalent zum Fixpunktproblem

$$\boldsymbol{K}_{\boldsymbol{\eta}}(\boldsymbol{y}) = \boldsymbol{y}. \tag{3.11}$$

Betrachten wir die Anfangswertaufgabe (3.1) auf dem Intervall $[t_0, t_0 + \hat{h}]$ des vorigen Abschnittes, dann ist diese dort eindeutig lösbar und die Lösung $\boldsymbol{y}_{\boldsymbol{\eta}}$ ist Fixpunkt des zugehörigen Fixpunktproblems (3.11) auf $[t_0, t_0 + \hat{h}]$, d. h. es gilt

$$\boldsymbol{y}_{\boldsymbol{\eta}}(t) = \boldsymbol{\eta} + \int_{t_0}^{t} \boldsymbol{f}\big(\boldsymbol{y}_{\boldsymbol{\eta}}(\tau), \tau\big)\, d\tau, \quad t \in [t_0, t_0 + \hat{h}].$$

Ersetzt man in dieser Gleichung $\boldsymbol{y}_{\boldsymbol{\eta}}(t)$ nach (3.9) durch $\boldsymbol{y}^{\star}(\boldsymbol{\eta}, t)$, dann schreibt sich diese als

$$\boldsymbol{y}^{\star}(\boldsymbol{\eta}, t) = \boldsymbol{\eta} + \int_{t_0}^{t} \boldsymbol{f}\big(\boldsymbol{y}^{\star}(\boldsymbol{\eta}, \tau), \tau\big)\, d\tau, \quad t \in [t_0, t_0 + \hat{h}]. \tag{3.12}$$

Mit $\tilde{\boldsymbol{D}} := \boldsymbol{I}_{\boldsymbol{\eta}} \times [t_0, t_0 + \hat{h}]$ und dem Operator

$$\boldsymbol{K} = \begin{pmatrix} k_1 \\ k_2 \end{pmatrix} : C(\tilde{\boldsymbol{D}}) \times C(\tilde{\boldsymbol{D}}) \to C(\tilde{\boldsymbol{D}}) \times C(\tilde{\boldsymbol{D}}),$$

definiert durch

$$\boldsymbol{K}(\boldsymbol{y})(\boldsymbol{\eta}, t) := \boldsymbol{\eta} + \int_{t_0}^{t} \boldsymbol{f}\big(\boldsymbol{y}(\boldsymbol{\eta}, \tau), \tau\big)\, d\tau$$

erhält man mit

$$\boldsymbol{K}(\boldsymbol{y}) = \boldsymbol{y} \tag{3.13}$$

aus (3.12) eine äquivalente Formulierung des Anfangswertproblems (3.1) auf $\tilde{\boldsymbol{D}}$. Die eingeführte Funktion $\boldsymbol{y}^{\star}$ löst (3.13).

Die bisher verwendete Intervallgrenze $t_0 + \hat{h}$ bzw. der Wert $\hat{h}$ ist in aller Regel unbekannt. Deshalb wählen wir für das weitere Vorgehen ein $h_0 := \hat{h}$ mit $0 < h_0 \leqslant t_e - t_0$, welches die Rolle von $\hat{h}$ übernehmen soll, setzen $t_1 := t_0 + h_0$ und betrachten im Folgenden (3.13) für

$$\boldsymbol{D} := \boldsymbol{I}_{\boldsymbol{\eta}} \times [t_0, t_1].$$

Wir versuchen mit einem geeigneten Fixpunktsatz eine Einschließung von $\boldsymbol{y}^{\star}$ auf $\boldsymbol{D}$ zu berechnen.

Im Gegensatz zu Makino und Berz [5, 10][42, S. 139-147] verwenden wir in dieser Arbeit den Banachschen Fixpunktsatz zur Lösungseinschließung. Makino und Berz stützen sich auf den Schauderschen Fixpunktsatz, der aber nur die Existenz eines Fixpunktes sichert.

Wir geben im Folgenden beide Fixpunktsätze an und orientieren uns bei deren Formulierung an [24, S. 35, S. 608]:

Satz 9 (Schauderscher Fixpunktsatz)
Sei $\mathcal{M}$ eine (nichtleere) konvexe und kompakte Teilmenge eines normierten Raumes und $\boldsymbol{K}$ eine stetige Selbstabbildung von $\mathcal{M}$. Dann besitzt $\boldsymbol{K}$ einen Fixpunkt in $\mathcal{M}$.

Satz 10 (Banachscher Fixpunktsatz)
Sei $\mathcal{M}$ eine (nichtleere) abgeschlossene Teilmenge eines Banachraumes und $\boldsymbol{K}$ eine kontrahierende Selbstabbildung von $\mathcal{M}$, für alle $\boldsymbol{y}$, $\tilde{\boldsymbol{y}} \in \mathcal{M}$ gelte also

$$\|\boldsymbol{K}(\boldsymbol{y}) - \boldsymbol{K}(\tilde{\boldsymbol{y}})\| \leqslant q \, \|\boldsymbol{y} - \tilde{\boldsymbol{y}}\|$$

mit einem festen $q < 1$. Dann besitzt $\boldsymbol{K}$ genau einen Fixpunkt $\boldsymbol{y}^\star$ in $\mathcal{M}$. Derselbe kann iterativ gewonnen werden: Wählt man einen beliebigen Startpunkt $\boldsymbol{y}_0 \in \mathcal{M}$ und setzt $\boldsymbol{y}_{k+1} := \boldsymbol{K}(\boldsymbol{y}_k)$, $k \geqslant 0$, so konvergiert die Folge $\{\boldsymbol{y}_k\}_{k \geqslant 0}$ gegen $\boldsymbol{y}^\star$.

Wir gehen im Folgenden näher auf die Voraussetzungen des Banachschen Fixpunktsatzes ein und bringen dabei die Taylor-Modelle ins Spiel.

Banachraum: Mit Blick auf das Fixpunktproblem (3.13) betrachten wir den Raum $C(\boldsymbol{D})^2$. Für jedes $\alpha > 0$ ist die gewichtete Norm

$$\|y\|_\alpha := \max_{(\boldsymbol{\eta},t) \in \boldsymbol{D}} \left(e^{-\alpha(t-t_0)} |y(\boldsymbol{\eta},t)| \right)$$

in $C(\boldsymbol{D})$ äquivalent zur Maximumnorm. Analog wird $C(\boldsymbol{D})^2$ für beliebige $\alpha > 0$ durch die Norm

$$\|\boldsymbol{y}\| := \max \left\{ \|y_1\|_\alpha, \|y_2\|_\alpha \right\}$$

zu einem Banachraum.

Teilmenge $\mathcal{M}$: Die Teilmenge $\mathcal{M}$ des Banachraumes $\left(C(\boldsymbol{D})^2, \|\cdot\|\right)$ definieren wir wie Makino und Berz über ein Taylor-Modell. Es sei

$$[\![\boldsymbol{P}_{n,\boldsymbol{y}^\star}, \boldsymbol{I}_{n,\boldsymbol{y}^\star}]\!] = \begin{pmatrix} [\![P_{n,y_1^\star}, I_{n,y_1^\star}]\!] \\ [\![P_{n,y_2^\star}, I_{n,y_2^\star}]\!] \end{pmatrix} \in \mathcal{TM}_n(\boldsymbol{D})^2 \tag{3.14}$$

eine Funktionenmenge im Sinne von (2.3). Das Taylorpolynom sei bzgl. der drei Variablen η_1, η_2 und t bei

$$\big(\,\mathrm{m}(I_{\eta_1}), \mathrm{m}(I_{\eta_2}), t_0\,\big) \in \boldsymbol{D}$$

entwickelt. Für $\mathrm{m}(I_{\eta_1})$ und $\mathrm{m}(I_{\eta_2})$ schreiben wir im Folgenden kurz m_1 bzw. m_2.

Wir definieren

$$\mathcal{M} := [\![\boldsymbol{P}_{n,\boldsymbol{y}^\star}, \boldsymbol{I}_{n,\boldsymbol{y}^\star}]\!] \subseteq C(\boldsymbol{D})^2.$$

Die Menge $\mathcal{M}$ ist als kartesisches Produkt der abgeschlossenen Funktionenmengen $[\![P_{n,y_1^\star}, I_{n,y_1^\star}]\!]$ und $[\![P_{n,y_2^\star}, I_{n,y_2^\star}]\!]$ abgeschlossen. Zudem ist $\mathcal{M}$ nicht leer.

Selbstabbildung: Gilt für die in (3.14) eingeführte Funktionenmenge die Inklusion

$$\boldsymbol{K}\big([\![\boldsymbol{P}_{n,\boldsymbol{y}^\star}, \boldsymbol{I}_{n,\boldsymbol{y}^\star}]\!]\big) \subseteq [\![\boldsymbol{P}_{n,\boldsymbol{y}^\star}, \boldsymbol{I}_{n,\boldsymbol{y}^\star}]\!], \tag{3.15}$$

dann ist per Definition

$$\boldsymbol{K}(\mathcal{M}) \subseteq \mathcal{M}$$

und $\boldsymbol{K}$ bildet die Menge $\mathcal{M}$ in sich ab.

Kontraktionsbedingung: Mit einem Intervall $\boldsymbol{I}_{\boldsymbol{P}_{n,\boldsymbol{y}^\star}} \in \mathbb{IR}^2$, das den Wertebereich $W(\boldsymbol{P}_{n,\boldsymbol{y}^\star}; \boldsymbol{D})$ von $\boldsymbol{P}_{n,\boldsymbol{y}^\star}$ enthält, gilt für alle Funktionen $\tilde{\boldsymbol{y}} \in [\![\boldsymbol{P}_{n,\boldsymbol{y}^\star}, \boldsymbol{I}_{n,\boldsymbol{y}^\star}]\!]$

$$\begin{aligned}\tilde{\boldsymbol{y}}(\eta_1,\eta_2,t) &\in \boldsymbol{P}_{n,\boldsymbol{y}^\star}\big(\eta_1 - m_1, \eta_2 - m_2, t - t_0\big) + \boldsymbol{I}_{n,\boldsymbol{y}^\star} \\ &\subseteq \boldsymbol{I}_{\boldsymbol{P}_{n,\boldsymbol{y}^\star}} + \boldsymbol{I}_{n,\boldsymbol{y}^\star} =: \boldsymbol{I}_{\boldsymbol{y}^\star} \quad \forall \boldsymbol{\eta} \in \boldsymbol{I}_{\boldsymbol{\eta}},\ \forall t \in [t_0, t_1].\end{aligned}$$

Gilt (3.15), dann ist außerdem

$$\begin{aligned}\boldsymbol{K}(\tilde{\boldsymbol{y}})(\eta_1,\eta_2,t) &\in \boldsymbol{P}_{n,\boldsymbol{y}^\star}\big(\eta_1 - m_1, \eta_2 - m_2, t - t_0\big) + \boldsymbol{I}_{n,\boldsymbol{y}^\star} \\ &\subseteq \boldsymbol{I}_{\boldsymbol{y}^\star} \quad \forall \boldsymbol{\eta} \in \boldsymbol{I}_{\boldsymbol{\eta}},\ \forall t \in [t_0, t_1],\end{aligned}$$

woraus speziell für $t = t_0$

$$\boldsymbol{\eta} = \boldsymbol{K}(\tilde{\boldsymbol{y}})(\eta_1,\eta_2,t_0) \in \boldsymbol{I}_{\boldsymbol{y}^\star} \quad \forall \boldsymbol{\eta} \in \boldsymbol{I}_{\boldsymbol{\eta}}$$

folgt, d. h. es ist $\boldsymbol{I}_{\boldsymbol{\eta}} \subseteq \boldsymbol{I}_{\boldsymbol{y}^\star}$.

Hiermit genügen die rechten Seiten f_1 und f_2 nach Abschnitt 3.2 jeweils der Lipschitz-Bedingung (3.8). Daraus folgt für beliebige

$$\boldsymbol{y} = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix},\ \tilde{\boldsymbol{y}} = \begin{pmatrix} \tilde{y}_1 \\ \tilde{y}_2 \end{pmatrix} \in \mathcal{M}$$

und beliebiges $(\eta_1, \eta_2, t) \in \boldsymbol{D}$:

$$
\begin{aligned}
&\big|k_i(\boldsymbol{y})(\eta_1,\eta_2,t) - k_i(\tilde{\boldsymbol{y}})(\eta_1,\eta_2,t)\big| \\
&= \left| \int_{t_0}^{t} \Big[f_i\big(y_1(\eta_1,\eta_2,\tau), y_2(\eta_1,\eta_2,\tau),\tau\big) - f_i\big(\tilde{y}_1(\eta_1,\eta_2,\tau), \tilde{y}_2(\eta_1,\eta_2,\tau),\tau\big) \Big]\, d\tau \right| \\
&\leqslant \int_{t_0}^{t} \Big[L_i \cdot \big(|y_1(\eta_1,\eta_2,\tau) - \tilde{y}_1(\eta_1,\eta_2,\tau)| + |y_2(\eta_1,\eta_2,\tau) - \tilde{y}_2(\eta_1,\eta_2,\tau)| \big) \Big]\, d\tau \\
&\leqslant L_i \cdot \big(\|y_1 - \tilde{y}_1\|_\alpha + \|y_2 - \tilde{y}_2\|_\alpha \big) \cdot \int_{t_0}^{t} e^{\alpha(\tau - t_0)}\, d\tau \\
&\leqslant \frac{2L_i}{\alpha} \cdot \|\boldsymbol{y} - \tilde{\boldsymbol{y}}\| \cdot e^{\alpha(t-t_0)}, \quad i = 1, 2.
\end{aligned}
$$

Wir wählen nun $\alpha > 0$ so, dass $q := \max\left\{\frac{2L_1}{\alpha}, \frac{2L_2}{\alpha}\right\} < 1$ gilt, und dividieren beide Abschätzungen durch $e^{\alpha(t-t_0)}$. Dann gilt

$$\|k_i(\boldsymbol{y}) - k_i(\tilde{\boldsymbol{y}})\|_\alpha \leqslant q \cdot \|\boldsymbol{y} - \tilde{\boldsymbol{y}}\|, \quad i = 1, 2.$$

Daraus folgt die Kontraktion

$$\|\boldsymbol{K}(\boldsymbol{y}) - \boldsymbol{K}(\tilde{\boldsymbol{y}})\| \leqslant q \cdot \|\boldsymbol{y} - \tilde{\boldsymbol{y}}\|$$

des Operators $\boldsymbol{K}$ auf der Menge $\mathcal{M}$.

Wir halten fest: Unter der Voraussetzung, dass ein Taylor-Modell (3.14) mit der Eigenschaft (3.15) existiert, greift der Banachsche Fixpunktsatz und $\mathcal{M}$ enthält genau einen Fixpunkt $\boldsymbol{y}^\star$ von (3.13) auf $\boldsymbol{D} = \boldsymbol{I}_{\boldsymbol{\eta}} \times [t_0, t_1]$. In diesem Falle gilt

$$\boldsymbol{y}^\star(\eta_1,\eta_2,t) \in \boldsymbol{P}_{n,\boldsymbol{y}^\star}\big(\eta_1 - m_1, \eta_2 - m_2, t - t_0\big) + \boldsymbol{I}_{n,\boldsymbol{y}^\star}$$

für alle $\boldsymbol{\eta} \in \boldsymbol{I}_{\boldsymbol{\eta}}$, $t \in [t_0, t_1]$. Insbesondere gilt an der Stelle $t = t_1$

$$\boldsymbol{y}^\star(\eta_1,\eta_2,t_1) = \begin{pmatrix} y_1^\star(\eta_1,\eta_2,t_1) \\ y_2^\star(\eta_1,\eta_2,t_1) \end{pmatrix} \in \boldsymbol{P}_{n,\boldsymbol{y}^\star}\big(\eta_1 - m_1, \eta_2 - m_2, h_0\big) + \boldsymbol{I}_{n,\boldsymbol{y}^\star}$$

für alle $\boldsymbol{\eta} \in \boldsymbol{I}_{\boldsymbol{\eta}}$.

Wir definieren

$$\boldsymbol{P}^{(t_1)}_{n,\boldsymbol{y}^\star}\big(\eta_1 - m_1, \eta_2 - m_2\big) := \boldsymbol{P}_{n,\boldsymbol{y}^\star}\big(\eta_1 - m_1, \eta_2 - m_2, h_0\big),$$

$$\boldsymbol{I}^{(t_1)}_{n,\boldsymbol{y}^\star} := \boldsymbol{I}_{n,\boldsymbol{y}^\star}$$

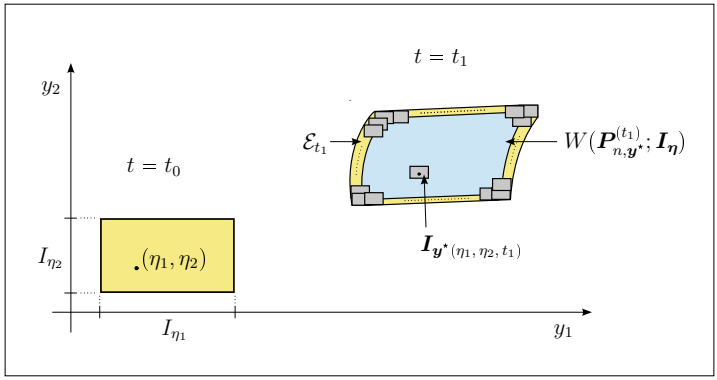

Abbildung 3.1: Die Mengen $\boldsymbol{I}_{\boldsymbol{y}^\star(\eta_1,\eta_2,t_1)}$, $\mathcal{E}_{t_1}$ und $W(\boldsymbol{P}_{n,\boldsymbol{y}^\star}^{(t_1)}; \boldsymbol{I}_{\boldsymbol{\eta}})$ beispielhaft skizziert.

und machen durch die Indizierung mit t_1 deutlich, dass diese Komponenten aus den Komponenten der Lösungseinschließung $[\![\boldsymbol{P}_{n,\boldsymbol{y}^\star}, \boldsymbol{I}_{n,\boldsymbol{y}^\star}]\!]$ durch Einsetzen der Stelle $t = t_1$ hervorgehen. Bezeichnet für $\boldsymbol{\eta} \in \boldsymbol{I}_{\boldsymbol{\eta}}$

$$\boldsymbol{I}_{\boldsymbol{y}^\star(\eta_1,\eta_2,t_1)} := \boldsymbol{P}_{n,\boldsymbol{y}^\star}^{(t_1)}\big(\eta_1 - m_1, \eta_2 - m_2\big) + \boldsymbol{I}_{n,\boldsymbol{y}^\star}^{(t_1)} \tag{3.16}$$

die Intervallbox zum Wert $\boldsymbol{y}^\star(\eta_1, \eta_2, t_1)$ in $\mathbb{R}^2$, dann ist

$$\mathcal{E}_{t_1} := \bigcup_{\boldsymbol{\eta} \in \boldsymbol{I}_{\boldsymbol{\eta}}} \boldsymbol{I}_{\boldsymbol{y}^\star(\eta_1,\eta_2,t_1)} \supseteq \mathcal{W}_{t_1} \tag{3.17}$$

eine simultane Einschließung aller Funktionswerte der Lösungen an der Stelle t_1. Abbildung 3.1 veranschaulicht beispielhaft die Mengen $\boldsymbol{I}_{\boldsymbol{y}^\star(\eta_1,\eta_2,t_1)}$, $\mathcal{E}_{t_1}$ und den Wertebereich $W(\boldsymbol{P}_{n,\boldsymbol{y}^\star}^{(t_1)}; \boldsymbol{I}_{\boldsymbol{\eta}})$.

Im nächsten Abschnitt stellen wir das Einschließungsverfahren von Makino und Berz vor, welches versucht ein Taylor-Modell (3.14) zu berechnen, das die Inklusion (3.15) erfüllt. Neben der reinen Beschreibung des Verfahrens geben wir außerdem die einzelnen Komponenten bzw. Teilberechnungen des Verfahrens in Pseudocode wieder. Es handelt sich dabei um deterministische Algorithmen, von denen einige in deren Ablaufsteuerung Heuristiken verwenden, die sich in der praktischen Anwendung des Einschließungsverfahrens bewährt haben.

3.4 Das Einschließungsverfahren von Makino und Berz

Das Einschließungsverfahren von Makino und Berz berechnet das gesuchte Taylor-Modell (3.14) in zwei Schritten. Zuerst wird das Taylorpolynom $\boldsymbol{P}_{n,\boldsymbol{y}^\star}$ berechnet, dann anschließend die Fehlereinschließungen $\boldsymbol{I}_{n,\boldsymbol{y}^\star}$.

Wir setzen voraus, dass die Funktionsausdrücke der rechten Seiten f_1 und f_2 mit Elementen aus $\mathcal{TM}_n(\boldsymbol{D})$ ausgewertet werden können. Dies ist der Fall, wenn sich die Ausdrücke aus endlich vielen Konstanten, Variablen, arithmetischen Grundoperationen sowie Standardfunktionen aus $\mathcal{SF}$ zusammensetzen. Folglich ist der Operator $\boldsymbol{K}$ mit Elementen aus $\mathcal{TM}_n(\boldsymbol{D})^2$ auswertbar. Wie in Abschnitt 2.2.1.4 vereinbart, bezeichnen wir die Repräsentation von $\boldsymbol{K}$ auf $\mathcal{TM}_n(\boldsymbol{D})^2$ mit einem $\square$ im Index.

Die Variablen $\boldsymbol{\eta}$ und t werden in der Repräsentation $\boldsymbol{K}_\square$ des Operators $\boldsymbol{K}$ durch die Taylor-Modelle

$$[\![P_{n,\eta_1}, [0,0]]\!], \quad [\![P_{n,\eta_2}, [0,0]]\!], \quad [\![P_{n,t}, [0,0]]\!] \tag{3.18}$$

aus $\mathcal{TM}_n(\boldsymbol{D})$ mit

$$\begin{aligned} P_{n,\eta_1}\big(\eta_1 - m_1, \eta_2 - m_2, t - t_0\big) &= m_1 + (\eta_1 - m_1), \\ P_{n,\eta_2}\big(\eta_1 - m_1, \eta_2 - m_2, t - t_0\big) &= m_2 + (\eta_2 - m_2), \\ P_{n,t}\big(\eta_1 - m_1, \eta_2 - m_2, t - t_0\big) &= t_0 + (t - t_0) \end{aligned}$$

ersetzt.

3.4.1 Berechnen des Taylorpolynoms

Nach dem Banachschen Fixpunktsatz kann mit der Vorschrift

$$\begin{aligned} \boldsymbol{y}_0 &\in \mathcal{M}, \\ \boldsymbol{y}_{k+1} &:= \boldsymbol{K}(\boldsymbol{y}_k), \quad k \geqslant 0, \end{aligned}$$

die auch Picard-Iteration genannt wird, der Fixpunkt $\boldsymbol{y}^\star$ iterativ berechnet werden. $\boldsymbol{y}^\star$ ist die in Abschnitt 3.2 eingeführte Funktion.

Für die Berechnung des Taylorpolynoms $\boldsymbol{P}_{n,\boldsymbol{y}^\star}$ führen wir die Picard-Iteration in $\mathcal{TM}_n(\boldsymbol{D})^2$ durch. Wir beginnen mit dem Taylor-Modell

$$[\![\boldsymbol{P}_{n,\boldsymbol{y}_0}, \boldsymbol{I}_{n,\boldsymbol{y}_0}]\!] = \begin{pmatrix} [\![P_{n,\eta_1}, [0,0]]\!] \\ [\![P_{n,\eta_2}, [0,0]]\!] \end{pmatrix}$$

für die Funktion

$$\boldsymbol{y}_0(\eta_1, \eta_2, t) := \begin{pmatrix} \eta_1 \\ \eta_2 \end{pmatrix}$$

und erhalten im $(k+1)$-ten Schritt der Iteration

$$[\![\boldsymbol{P}_{n,\boldsymbol{y}_{k+1}}, \boldsymbol{I}_{n,\boldsymbol{y}_{k+1}}]\!] = \boldsymbol{K}_{\square}\big([\![\boldsymbol{P}_{n,\boldsymbol{y}_k}, \boldsymbol{I}_{n,\boldsymbol{y}_k}]\!]\big), \quad k \geqslant 0$$

insbesondere das Taylorpolynom $\boldsymbol{P}_{n,\boldsymbol{y}_{k+1}}$ der $(k+1)$-ten Iterierten

$$\boldsymbol{y}_{k+1} = \boldsymbol{K}(\boldsymbol{y}_k)$$

entwickelt bei $\big(m_1, m_2, t_0\big)$.

Betrachtet man die Ableitungen der ersten $n+1$ Iterierten an der Entwicklungsstelle $\big(m_1, m_2, t_0\big)$ genauer – jede Iterierte ist aufgrund der Glattheit der rechten Seite $\boldsymbol{f}$ $(n+1)$-mal stetig differenzierbar auf $\boldsymbol{D}$, ebenso auch $\boldsymbol{y}^\star = \boldsymbol{K}(\boldsymbol{y}^\star)$ –, dann erhält man

Satz 11
Sei $k \leqslant n+1$. Dann gilt für alle $j = j_1 + j_2 + j_3 \leqslant k$ mit Zahlen $j_1, j_2, j_3 \in \mathbb{N} \cup \{0\}$

$$\frac{\partial^j \boldsymbol{y}_k}{\partial \eta_1^{j_1} \partial \eta_2^{j_2} \partial t^{j_3}}\big(m_1, m_2, t_0\big) = \frac{\partial^j \boldsymbol{y}^\star}{\partial \eta_1^{j_1} \partial \eta_2^{j_2} \partial t^{j_3}}\big(m_1, m_2, t_0\big). \tag{3.19}$$

Beweis: Wir beweisen die Aussage durch Induktion über k. Für $k = 0$ ist

$$\boldsymbol{y}_0\big(m_1, m_2, t_0\big) = \begin{pmatrix} m_1 \\ m_2 \end{pmatrix} = \boldsymbol{K}(\boldsymbol{y}^\star)\big(m_1, m_2, t_0\big) = \boldsymbol{y}^\star\big(m_1, m_2, t_0\big).$$

Für $k = 1$ erhalten wir mit $\boldsymbol{y}_1 = \boldsymbol{K}(\boldsymbol{y}_0)$

$$\boldsymbol{y}_1\big(m_1, m_2, t_0\big) = \begin{pmatrix} m_1 \\ m_2 \end{pmatrix} = \boldsymbol{y}^\star\big(m_1, m_2, t_0\big)$$

und

$$\begin{aligned}
\frac{\partial \boldsymbol{y}_1}{\partial \eta_1}\big(m_1, m_2, t_0\big) &= \frac{\partial \boldsymbol{K}(\boldsymbol{y}_0)}{\partial \eta_1}\big(m_1, m_2, t_0\big) = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \\
&= \frac{\partial \boldsymbol{K}(\boldsymbol{y}^\star)}{\partial \eta_1}\big(m_1, m_2, t_0\big) = \frac{\partial \boldsymbol{y}^\star}{\partial \eta_1}\big(m_1, m_2, t_0\big), \\
\frac{\partial \boldsymbol{y}_1}{\partial \eta_2}\big(m_1, m_2, t_0\big) &= \frac{\partial \boldsymbol{K}(\boldsymbol{y}_0)}{\partial \eta_2}\big(m_1, m_2, t_0\big) = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \\
&= \frac{\partial \boldsymbol{K}(\boldsymbol{y}^\star)}{\partial \eta_2}\big(m_1, m_2, t_0\big) = \frac{\partial \boldsymbol{y}^\star}{\partial \eta_2}\big(m_1, m_2, t_0\big),
\end{aligned}$$

$$\begin{aligned}\frac{\partial \boldsymbol{y}_1}{\partial t}\big(m_1, m_2, t_0\big) &= \frac{\partial \boldsymbol{K}(\boldsymbol{y}_0)}{\partial t}\big(m_1, m_2, t_0\big)\\ &= \begin{pmatrix} f_1\big(m_1, m_2, t_0\big)\\ f_2\big(m_1, m_2, t_0\big)\end{pmatrix}\\ &= \frac{\partial \boldsymbol{K}(\boldsymbol{y}^\star)}{\partial t}\big(m_1, m_2, t_0\big) = \frac{\partial \boldsymbol{y}^\star}{\partial t}\big(m_1, m_2, t_0\big).\end{aligned}$$

Die Aussage (3.19) gelte nun für ein $k \leqslant n$. Wir betrachten zunächst den Fall $j \leqslant k$, dann anschließend den Fall $j = k+1$. Die Zahlen j_1, j_2, $j_3 \in \mathbb{N} \cup \{0\}$ mit $j_1 + j_2 + j_3 = j$ seien dabei beliebig. Für $j \leqslant k$ erhält man auf der rechten Seite von

$$\frac{\partial^j \boldsymbol{y}_{k+1}}{\partial \eta_1^{j_1} \partial \eta_2^{j_2} \partial t^{j_3}}\big(\eta_1, \eta_2, t\big) = \frac{\partial^j \boldsymbol{K}(\boldsymbol{y}_k)}{\partial \eta_1^{j_1} \partial \eta_2^{j_2} \partial t^{j_3}}\big(\eta_1, \eta_2, t\big)$$

in jeder Komponente durch mehrmaliges Anwenden der Ketten- und Produktregel einen Ausdruck mit partiellen Ableitungen bis zur Ordnung $j \leqslant k$ von $\boldsymbol{y}_k$. Da diese partiellen Ableitungen an der Stelle $\big(m_1, m_2, t_0\big)$ nach Induktionsvoraussetzung mit den entsprechenden partiellen Ableitungen von $\boldsymbol{y}^\star$ übereinstimmen, gilt

$$\begin{aligned}\frac{\partial^j \boldsymbol{y}_{k+1}}{\partial \eta_1^{j_1} \partial \eta_2^{j_2} \partial t^{j_3}}\big(m_1, m_2, t_0\big) &= \frac{\partial^j \boldsymbol{K}(\boldsymbol{y}^\star)}{\partial \eta_1^{j_1} \partial \eta_2^{j_2} \partial t^{j_3}}\big(m_1, m_2, t_0\big)\\ &= \frac{\partial^j \boldsymbol{y}^\star}{\partial \eta_1^{j_1} \partial \eta_2^{j_2} \partial t^{j_3}}\big(m_1, m_2, t_0\big).\end{aligned}$$

Im Fall $j = k+1$ betrachten wir die Fälle $j_3 = 0$ und $j_3 > 0$. Für $j_3 = 0$ wird nicht nach t differenziert und es bleibt auf der rechten Seite von

$$\frac{\partial^{k+1} \boldsymbol{y}_{k+1}}{\partial \eta_1^{j_1} \partial \eta_2^{j_2}}\big(\eta_1, \eta_2, t\big) = \frac{\partial^{k+1} \boldsymbol{K}(\boldsymbol{y}_k)}{\partial \eta_1^{j_1} \partial \eta_2^{j_2}}\big(\eta_1, \eta_2, t\big)$$

das Integral erhalten. Setzt man dann im Anschluss an die Differentiation die Stelle $\big(m_1, m_2, t_0\big)$ ein, so verschwindet rechts der Integralterm und es bleibt

$$\begin{pmatrix} \dfrac{\partial^{k+1} \eta_1}{\partial \eta_1^{j_1} \partial \eta_2^{j_2}}\big(m_1, m_2, t_0\big)\\[2ex] \dfrac{\partial^{k+1} \eta_2}{\partial \eta_1^{j_1} \partial \eta_2^{j_2}}\big(m_1, m_2, t_0\big)\end{pmatrix}$$

übrig. Das gleiche Ergebnis erhält man für $\dfrac{\partial^{k+1} \boldsymbol{K}(\boldsymbol{y}^\star)}{\partial \eta_1^{j_1} \partial \eta_2^{j_2}}\big(m_1, m_2, t_0\big)$, sodass letztendlich

$$\frac{\partial^{k+1} \boldsymbol{y}_{k+1}}{\partial \eta_1^{j_1} \partial \eta_2^{j_2}}\big(m_1, m_2, t_0\big) = \frac{\partial^{k+1} \boldsymbol{y}^\star}{\partial \eta_1^{j_1} \partial \eta_2^{j_2}}\big(m_1, m_2, t_0\big)$$

gilt. Im Fall $j_3 > 0$ erhalten wir

$$\begin{aligned}\frac{\partial^{k+1}\boldsymbol{y}_{k+1}}{\partial\eta_1^{j_1}\partial\eta_2^{j_2}\partial t^{j_3}}\big(\eta_1,\eta_2,t\big) &= \frac{\partial^k}{\partial\eta_1^{j_1}\partial\eta_2^{j_2}\partial t^{j_3-1}}\left[\frac{\partial\boldsymbol{y}_{k+1}}{\partial t}\big(\eta_1,\eta_2,t\big)\right]\\ &= \frac{\partial^k}{\partial\eta_1^{j_1}\partial\eta_2^{j_2}\partial t^{j_3-1}}\left[\frac{\partial\boldsymbol{K}(\boldsymbol{y}_k)}{\partial t}\big(\eta_1,\eta_2,t\big)\right]\\ &= \begin{pmatrix}\dfrac{\partial^k}{\partial\eta_1^{j_1}\partial\eta_2^{j_2}\partial t^{j_3-1}} f_1\big(\boldsymbol{y}_k(\eta_1,\eta_2,t),t\big)\\ \dfrac{\partial^k}{\partial\eta_1^{j_1}\partial\eta_2^{j_2}\partial t^{j_3-1}} f_2\big(\boldsymbol{y}_k(\eta_1,\eta_2,t),t\big)\end{pmatrix}.\end{aligned}$$

Auf der rechten Seite entstehen durch Differentiation Ausdrücke, in denen partielle Ableitungen von $\boldsymbol{y}_k$ bis höchstens zur Ordnung k auftreten. Da diese aber nach Induktionsvoraussetzung an der Stelle $\big(m_1,m_2,t_0\big)$ mit den entsprechenden partiellen Ableitungen von $\boldsymbol{y}^\star$ übereinstimmen, gilt

$$\frac{\partial^{k+1}\boldsymbol{y}_{k+1}}{\partial\eta_1^{j_1}\partial\eta_2^{j_2}\partial t^{j_3}}\big(m_1,m_2,t_0\big) = \frac{\partial^{k+1}\boldsymbol{y}^\star}{\partial\eta_1^{j_1}\partial\eta_2^{j_2}\partial t^{j_3}}\big(m_1,m_2,t_0\big).$$

□

Nach (3.19) stimmen also die partiellen Ableitungen von $\boldsymbol{y}_k$ bis zur Ordnung k mit den entsprechenden partiellen Ableitungen von $\boldsymbol{y}^\star$ an der Stelle $\big(m_1,m_2,t_0\big)$ überein. Das Taylorpolynom der Ordnung k der k-ten Iterierten ist damit gleich dem Taylorpolynom der Ordnung k des Fixpunktes $\boldsymbol{y}^\star$.

Darüber hinaus erhalten wir also nach n Schritten ein Taylorpolynom vom Grad n, das nach dem obigen Ergebnis durch Anwenden von $\boldsymbol{K}_\square$ reproduziert wird. Das Taylorpolynom $\boldsymbol{P}_{n,\boldsymbol{y}_n}$ entspricht dem Taylorpolynom $\boldsymbol{P}_{n,\boldsymbol{y}^\star}$ der Lösung, d. h. es ist

$$\boldsymbol{P}_{n,\boldsymbol{y}^\star} = \boldsymbol{P}_{n,\boldsymbol{y}_n}.$$

Algorithmus 1 fasst die Berechnung des Taylorpolynoms in Pseudocode zusammen.

Bevor wir nun die Berechnung einer geeigneten Fehlereinschließung $\boldsymbol{I}_{n,\boldsymbol{y}^\star}$ erklären, demonstrieren wir in dem folgenden Beispiel die Durchführung der Picard-Iteration in $\mathcal{TM}_n(\boldsymbol{D})^2$.

Beispiel 3.1 *Wir betrachten das Anfangswertproblem*

$$\begin{aligned} y_1'(t) &= y_2(t),\\ y_2'(t) &= (y_1(t))^2,\\ y_1(0) &= \eta_1,\\ y_2(0) &= \eta_2 \end{aligned}$$

Algorithmus 1 : Picard-Iteration

Daten : Die Taylorpolynome P_{n,η_1} und P_{n,η_2}
Ergebnis : Das Taylorpolynom $\boldsymbol{P}_{n,\boldsymbol{y}^\star}$ des Fixpunktes $\boldsymbol{y}^\star$

$$[\![\boldsymbol{P}_{n,\boldsymbol{y}_0}, \boldsymbol{I}_{n,\boldsymbol{y}_0}]\!] := \begin{pmatrix} [\![P_{n,\eta_1}, [0,0]]\!] \\ [\![P_{n,\eta_2}, [0,0]]\!] \end{pmatrix};$$

for $k := 1$ **to** n **do**
  $[\![\boldsymbol{P}_{n,\boldsymbol{y}_k}, \boldsymbol{I}_{n,\boldsymbol{y}_k}]\!] := \boldsymbol{K}_\square\big([\![\boldsymbol{P}_{n,\boldsymbol{y}_{k-1}}, \boldsymbol{I}_{n,\boldsymbol{y}_{k-1}}]\!]\big)$;
end
$\boldsymbol{P}_{n,\boldsymbol{y}^\star} := \boldsymbol{P}_{n,\boldsymbol{y}_n}$;

mit $\eta_1, \eta_2 \in [-0.1, 0.1]$. Wir berechnen das Taylorpolynom vierten Grades der Lösung entwickelt um $m_1 = 0$, $m_2 = 0$, $t_0 = 0$. Da die auftretenden Fehlereinschließungen für die Berechnung des Taylorpolynoms unwichtig sind, betrachten wir hier nur den polynomialen Anteil der vorkommenden Taylor-Modelle. Wir beginnen mit

$$\boldsymbol{P}_{4,\boldsymbol{y}_0}(\eta_1, \eta_2, t) = \begin{pmatrix} \eta_1 \\ \eta_2 \end{pmatrix}.$$

Einsetzen in den Operator $\boldsymbol{K}_\square$ ergibt

$$\boldsymbol{P}_{4,\boldsymbol{y}_1} = \boldsymbol{K}_\square(\boldsymbol{P}_{4,\boldsymbol{y}_0}) = \begin{pmatrix} \eta_1 + \int_0^t \eta_2 \, d\tau \\ \eta_2 + \int_0^t \eta_1^2 \, d\tau \end{pmatrix} = \begin{pmatrix} \eta_1 + \eta_2 \, t \\ \eta_2 + \eta_1^2 \, t \end{pmatrix}.$$

Erneutes Einsetzen von $\boldsymbol{P}_{4,\boldsymbol{y}_1}$ liefert

$$\boldsymbol{P}_{4,\boldsymbol{y}_2} = \boldsymbol{K}_\square(\boldsymbol{P}_{4,\boldsymbol{y}_1}) = \begin{pmatrix} \eta_1 + \int_0^t (\eta_2 + \eta_1^2 \, \tau) \, d\tau \\ \eta_2 + \int_0^t (\eta_1 + \eta_2 \, \tau)^2 \, d\tau \end{pmatrix} = \begin{pmatrix} \eta_1 + \eta_2 \, t + \frac{1}{2} \, \eta_1^2 \, t^2 \\ \eta_2 + \eta_1^2 \, t + \eta_1 \, \eta_2 \, t^2 \end{pmatrix}.$$

So fortfahrend erhalten wir $\boldsymbol{P}_{4,\boldsymbol{y}_4} = \boldsymbol{P}_{4,\boldsymbol{y}_3} = \boldsymbol{P}_{4,\boldsymbol{y}_2}$. Man beachte, dass alle Terme mit Ordnung größer als 4 innerhalb der Taylor-Modell-Arithmetik zum Intervall-

term gepackt werden. Es ist also

$$\begin{aligned} P_{4,y_1^\star}(\eta_1,\eta_2,t) &= \eta_1 + \eta_2\, t + \tfrac{1}{2}\,\eta_1^2\, t^2, \\ P_{4,y_2^\star}(\eta_1,\eta_2,t) &= \eta_2 + \eta_1^2\, t + \eta_1\,\eta_2\, t^2. \end{aligned}$$

3.4.2 Berechnen der Fehlereinschließung

Als nächstes berechnen wir eine Fehlereinschließung $\boldsymbol{I}_{n,\boldsymbol{y}^\star}$, sodass $[\![\boldsymbol{P}_{n,\boldsymbol{y}^\star}, \boldsymbol{I}_{n,\boldsymbol{y}^\star}]\!]$ die Bedingung (3.15) erfüllt.

Doch zunächst nehmen wir ein geeignetes Intervall $\boldsymbol{I}_{n,\boldsymbol{y}^\star}$ als bekannt an und betrachten die Bedingung (3.15) genauer. Aufgrund der Inklusionsmonotonie der Taylor-Modell-Operationen gilt

$$\boldsymbol{K}\big([\![\boldsymbol{P}_{n,\boldsymbol{y}^\star}, \boldsymbol{I}_{n,\boldsymbol{y}^\star}]\!]\big) \subseteq \boldsymbol{K}_\square\big([\![\boldsymbol{P}_{n,\boldsymbol{y}^\star}, \boldsymbol{I}_{n,\boldsymbol{y}^\star}]\!]\big),$$

d. h. die Bedingung (3.15) ist sicher erfüllt, wenn

$$\boldsymbol{K}_\square\big([\![\boldsymbol{P}_{n,\boldsymbol{y}^\star}, \boldsymbol{I}_{n,\boldsymbol{y}^\star}]\!]\big) \subseteq [\![\boldsymbol{P}_{n,\boldsymbol{y}^\star}, \boldsymbol{I}_{n,\boldsymbol{y}^\star}]\!] \tag{3.20}$$

gilt. Aus dem vorherigen Abschnitt 3.4.1 wissen wir, dass das Taylorpolynom $\boldsymbol{P}_{n,\boldsymbol{y}^\star}$ von $\boldsymbol{K}_\square$ reproduziert wird. Demnach erhalten wir auf der linken Seite von (3.20) ein Taylor-Modell

$$[\![\boldsymbol{P}_{n,\boldsymbol{y}^\star}, \boldsymbol{I}_{n,\boldsymbol{K}(\boldsymbol{y}^\star)}]\!] = \begin{pmatrix} [\![P_{n,y_1^\star}, I_{n,k_1(y^\star)}]\!] \\ [\![P_{n,y_2^\star}, I_{n,k_2(y^\star)}]\!] \end{pmatrix} = \boldsymbol{K}_\square\big([\![\boldsymbol{P}_{n,\boldsymbol{y}^\star}, \boldsymbol{I}_{n,\boldsymbol{y}^\star}]\!]\big), \tag{3.21}$$

bei dem das Taylorpolynom $\boldsymbol{P}_{n,\boldsymbol{y}^\star}$ ebenfalls darstellende Komponente ist; da die Auswertung von $\boldsymbol{K}$ mit einem Taylor-Modell für $\boldsymbol{y}^\star$ ein Taylor-Modell für $\boldsymbol{K}(\boldsymbol{y}^\star)$ liefert, steht im Index der Fehlereinschließung die Bezeichnung $\boldsymbol{K}(\boldsymbol{y}^\star)$. Mit (3.21) erhält man nach (2.8) die zu (3.20) äquivalente Inklusion

$$\boldsymbol{I}_{n,\boldsymbol{K}(\boldsymbol{y}^\star)} = \begin{pmatrix} I_{n,k_1(y^\star)} \\ I_{n,k_2(y^\star)} \end{pmatrix} \subseteq \begin{pmatrix} I_{n,y_1^\star} \\ I_{n,y_2^\star} \end{pmatrix} = \boldsymbol{I}_{n,\boldsymbol{y}^\star}, \tag{3.22}$$

die leicht nachzuprüfen ist.

Können wir also mit einem noch zu definierenden Verfahren eine Fehlereinschließung $\boldsymbol{I}_{n,\boldsymbol{y}^\star}$ mit der Eigenschaft (3.22) berechnen, dann gilt (3.20) und die geforderte Bedingung (3.15) ist mit dem Taylor-Modell $[\![\boldsymbol{P}_{n,\boldsymbol{y}^\star}, \boldsymbol{I}_{n,\boldsymbol{y}^\star}]\!]$ erfüllt.

Wir kennen ferner einen Vertreter aus der Menge $[\![\boldsymbol{P}_{n,\boldsymbol{y}^\star}, \boldsymbol{I}_{n,\boldsymbol{y}^\star}]\!]$, nämlich das Taylorpolynom $\boldsymbol{P}_{n,\boldsymbol{y}^\star}$. Wir erhalten speziell mit

$$[\![\boldsymbol{P}_{n,\boldsymbol{y}^\star}, \boldsymbol{I}^{(0)}_{n,\boldsymbol{K}(\boldsymbol{y}^\star)}]\!] := \boldsymbol{K}_\square\big([\![P_{n,y_1^\star}, [0,0]]\!], [\![P_{n,y_2^\star}, [0,0]]\!]\big)$$

aus (3.20) die Inklusion

$$\boldsymbol{I}^{(0)}_{n,\boldsymbol{K}(\boldsymbol{y}^\star)} = \begin{pmatrix} I^{(0)}_{n,k_1(y^\star)} \\ I^{(0)}_{n,k_2(y^\star)} \end{pmatrix} \subseteq \begin{pmatrix} I_{n,y_1^\star} \\ I_{n,y_2^\star} \end{pmatrix} = \boldsymbol{I}_{n,\boldsymbol{y}^\star}.$$

Das Intervall $\boldsymbol{I}^{(0)}_{n,\boldsymbol{K}(\boldsymbol{y}^\star)}$ gibt ein Mindestmaß für die Fehlereinschließung $\boldsymbol{I}_{n,\boldsymbol{y}^\star}$ an und stellt damit eine innere Approximation an $\boldsymbol{I}_{n,\boldsymbol{y}^\star}$ dar. Diese innere Approximation wird für die Konstruktion von $\boldsymbol{I}_{n,\boldsymbol{y}^\star}$ herangezogen.

Wir lassen die getroffene Annahme über die Existenz der Fehlereinschließung $\boldsymbol{I}_{n,\boldsymbol{y}^\star}$ wieder fallen und gehen als Nächstes an deren Berechnung. Ausgehend von der inneren Approximation $\boldsymbol{I}^{(0)}_{n,\boldsymbol{K}(\boldsymbol{y}^\star)}$ definieren wir mittels ϵ-Inflation [57] eine Folge von Intervallen durch (ϵ_1, $\epsilon_2 > 0$)

$$\boldsymbol{I}^{(k)}_{n,\boldsymbol{y}^\star} := (1+\epsilon_1)\boldsymbol{I}^{(k-1)}_{n,\boldsymbol{K}(\boldsymbol{y}^\star)} - \epsilon_1 \boldsymbol{I}^{(k-1)}_{n,\boldsymbol{K}(\boldsymbol{y}^\star)} + \begin{pmatrix} [-\epsilon_2, \epsilon_2] \\ [-\epsilon_2, \epsilon_2] \end{pmatrix}, \quad k \geqslant 1, \tag{3.23}$$

nach der von Rihm [56, S. 30] empfohlenen Weise als Kombination einer relativen Aufweitung mit einer absoluten Aufweitung. Wir berechnen dann für jedes k die Menge

$$[\![\boldsymbol{P}_{n,\boldsymbol{y}^\star}, \boldsymbol{I}^{(k)}_{n,\boldsymbol{K}(\boldsymbol{y}^\star)}]\!] = \begin{pmatrix} [\![P_{n,y_1^\star}, I^{(k)}_{n,k_1(y^\star)}]\!] \\ [\![P_{n,y_2^\star}, I^{(k)}_{n,k_2(y^\star)}]\!] \end{pmatrix} = \boldsymbol{K}_\square\big([\![\boldsymbol{P}_{n,\boldsymbol{y}^\star}, \boldsymbol{I}^{(k)}_{n,\boldsymbol{y}^\star}]\!]\big) \tag{3.24}$$

und prüfen, ob die Inklusion

$$\boldsymbol{I}^{(k)}_{n,\boldsymbol{K}(\boldsymbol{y}^\star)} = \begin{pmatrix} I^{(k)}_{n,k_1(y^\star)} \\ I^{(k)}_{n,k_2(y^\star)} \end{pmatrix} \subseteq \begin{pmatrix} I^{(k)}_{n,y_1^\star} \\ I^{(k)}_{n,y_2^\star} \end{pmatrix} = \boldsymbol{I}^{(k)}_{n,\boldsymbol{y}^\star} \tag{3.25}$$

erfüllt wird. Ist dies für ein $k_0 \geqslant 1$ der Fall, dann beenden wir die Iteration und setzen

$$\boldsymbol{I}_{n,\boldsymbol{y}^\star} := \boldsymbol{I}^{(k_0)}_{n,\boldsymbol{K}(\boldsymbol{y}^\star)}.$$

In diesem Fall sind die Voraussetzungen für den Banachschen Fixpunktsatzes komplett. Das Taylor-Modell $[\![\boldsymbol{P}_{n,\boldsymbol{y}^\star}, \boldsymbol{I}_{n,\boldsymbol{y}^\star}]\!]$ erfüllt (3.15) und die Aussage des Banachschen Fixpunktsatzes gilt.

Ist die Inklusion (3.25) nach $k_{max} > 1$ Schritten noch nicht erfüllt, dann verringern wir analog zu Makino und Berz die Schrittweite h_0 durch Multiplikation mit dem Faktor[1] 0.7. Hierdurch verändert sich der Definitionsbereich $\boldsymbol{D}$ mit der

[1] mehr dazu in Kapitel 4.

Folge, dass die innere Approximation $\boldsymbol{I}^{(0)}_{n,\boldsymbol{K}(\boldsymbol{y}^\star)}$ neu berechnet werden muss – innerhalb der Taylor-Modell-Auswertung werden Einschließungen für alle $(\eta_1, \eta_2, t) \in \boldsymbol{D}$ bestimmt. Anschließend wird die eben vorgestellte Iteration erneut durchgeführt. Unterschreitet die Schrittweite h_0 eine vorgegebene Mindestschrittweite $h_{min} > 0$, dann brechen wir das Verfahren analog zur Vorgehensweise von Makino und Berz ab. In diesem Fall scheint eine Lösungseinschließung mit vertretbarem Aufwand nicht möglich. Algorithmus 2 gibt das vorgestellte Verfahren in übersichtlicher Form in Pseudocode wieder.

Bemerkung 3.1 a) Die Konstruktion der Folge $\boldsymbol{I}^{(k)}_{n,\boldsymbol{y}^\star}$ nach (3.23) stellt für das Verfahren von Makino und Berz eine Neuerung dar. Makino schlägt in ihrer Dissertation [42, S. 146] die Folge

$$\boldsymbol{I}^{(k)}_{n,\boldsymbol{y}^\star} := 2^k \cdot \boldsymbol{I}^{(0)}_{n,\boldsymbol{K}(\boldsymbol{y}^\star)}, \quad k \geqslant 1,$$

vor, d. h. es wird nach jedem erfolglosen Schritt die zuvor verwendete Fehlereinschließung verdoppelt. Kommt also im $(k-1)$-ten Schritt $(k > 1)$ die Inklusion (3.25) mit $\boldsymbol{I}^{(k-1)}_{n,\boldsymbol{y}^\star}$ nicht zustande, dann wird im nächsten Schritt das Intervall $\boldsymbol{I}^{(k)}_{n,\boldsymbol{y}^\star} = 2 \cdot \boldsymbol{I}^{(k-1)}_{n,\boldsymbol{y}^\star}$ probiert. Wir hingegen benutzen im k-ten Schritt die aus der Auswertung von $\boldsymbol{K}$ resultierende Fehlereinschließung $\boldsymbol{I}^{(k-1)}_{n,\boldsymbol{K}(\boldsymbol{y}^\star)}$ zur Inflation und nicht die zuvor verwendete Iterierte $\boldsymbol{I}^{(k-1)}_{n,\boldsymbol{y}^\star}$. Dieses Vorgehen wollen wir mit dem nachfolgenden Beispiel 3.2 motivieren. Die dort berechnete Inneneinschließung und damit auch die erste Iterierte $\boldsymbol{I}^{(1)}_{4,\boldsymbol{y}^\star}$ ist in der ersten Komponente symmetrisch zur Null. Die Auswertung von $\boldsymbol{K}_\square$ mit $\boldsymbol{I}^{(1)}_{4,\boldsymbol{y}^\star}$ ergibt in der resultierenden Fehlereinschließung $\boldsymbol{I}^{(1)}_{4,\boldsymbol{K}(\boldsymbol{y}^\star)}$ jedoch keine symmetrische erste Komponente mehr. Diese Verformung der Iterierten $\boldsymbol{I}^{(1)}_{4,\boldsymbol{y}^\star}$, die sich in $\boldsymbol{I}^{(1)}_{4,\boldsymbol{K}(\boldsymbol{y}^\star)}$ zeigt, kann als Hinweis auf die Lage und die Gestalt der gesuchten Einschließung $\boldsymbol{I}_{4,\boldsymbol{y}^\star}$ verstanden werden. Wir verwenden diese vom Operator $\boldsymbol{K}$ gelieferte Information in unserer ϵ-Inflation (3.23) für den nachfolgenden Schritt. Die Gestalt der in Beispiel 3.2 errechneten Lösungseinschließung $\boldsymbol{I}_{4,\boldsymbol{y}^\star}$ zeigt, dass es Sinn macht so zu verfahren.

b) Die Auswertung des Integraloperators geschieht komponentenweise. Dementsprechend können wir auch die Inklusionsbedingung (3.25) komponentenweise prüfen. Wir machen uns dies zunutze, indem wir bei Inklusion in der ersten Komponente

$$I^{(k)}_{n,k_1(y^\star)} \subseteq I^{(k)}_{n,y_1^\star},$$

die berechnete Fehlereinschließung $I^{(k)}_{n,k_1(y^\star)}$ für die Berechnung der zweiten Komponente – ähnlich dem Einzelschrittverfahren bei linearen Gleichungs-

Algorithmus 2 : Lösungseinschließung

Daten : Das Taylorpolynom $\boldsymbol{P}_{n,\boldsymbol{y}^\star}$, die Zahlen ϵ_1, ϵ_2, k_{max} und h_{min}
Ergebnis : Entweder eine Fehlereinschließung $\boldsymbol{I}_{n,\boldsymbol{y}^\star}$ oder Abbruch des Verfahrens

repeat

$[\![\boldsymbol{P}_{n,\boldsymbol{y}^\star}, \boldsymbol{I}^{(0)}_{n,\boldsymbol{K}(\boldsymbol{y}^\star)}]\!] := \boldsymbol{K}_\square\big([\![P_{n,y_1^\star}, [0,0]]\!], [\![P_{n,y_2^\star}, [0,0]]\!]\big)$;

$k := 0$;

$Inklusion := false$; `// boolsche Variable`

while $Inklusion \neq true \ \wedge \ k < k_{max}$ **do**

$k := k + 1$;

$\boldsymbol{I}^{(k)}_{n,\boldsymbol{y}^\star} := (1+\epsilon_1)\boldsymbol{I}^{(k-1)}_{n,\boldsymbol{K}(\boldsymbol{y}^\star)} - \epsilon_1 \boldsymbol{I}^{(k-1)}_{n,\boldsymbol{K}(\boldsymbol{y}^\star)} + \begin{pmatrix}[-\epsilon_2,\epsilon_2]\\ [-\epsilon_2,\epsilon_2]\end{pmatrix}$; `// ϵ-Inflation`

$[\![\boldsymbol{P}_{n,\boldsymbol{y}^\star}, \boldsymbol{I}^{(k)}_{n,\boldsymbol{K}(\boldsymbol{y}^\star)}]\!] := \boldsymbol{K}_\square\big([\![\boldsymbol{P}_{n,\boldsymbol{y}^\star}, \boldsymbol{I}^{(k)}_{n,\boldsymbol{y}^\star}]\!]\big)$;

if $\boldsymbol{I}^{(k)}_{n,\boldsymbol{K}(\boldsymbol{y}^\star)} \subseteq \boldsymbol{I}^{(k)}_{n,\boldsymbol{y}^\star}$ **then**

$Inklusion := true$;

end

end

if $Inklusion \neq true$ **then**

$h_0 := 0.7 \cdot h_0$;

if $h_0 < h_{min}$ **then**

exit ; `// Abbruch des Algorithmus`

end

$t_1 := t_0 + h_0$;

end

until $Inklusion = true$;

$\boldsymbol{I}_{n,\boldsymbol{y}^\star} := \boldsymbol{I}^{(k)}_{n,\boldsymbol{K}(\boldsymbol{y}^\star)}$;

systemen – gleich weiter verwenden. Wir berechnen also nicht

$$k_{2\square}\big([\![P_{n,y_1^\star}, I^{(k)}_{n,y_1^\star}]\!], [\![P_{n,y_2^\star}, I^{(k)}_{n,y_2^\star}]\!]\big) = [\![P_{n,k_2(y^\star)}, I^{(k)}_{n,k_2(y^\star)}]\!],$$

sondern

$$k_{2\square}\big([\![P_{n,y_1^\star}, I^{(k)}_{n,k_1(y^\star)}]\!], [\![P_{n,y_2^\star}, I^{(k)}_{n,y_2^\star}]\!]\big) = [\![P_{n,k_2(y^\star)}, \widetilde{I}^{(k)}_{n,k_2(y^\star)}]\!]$$

mit einer engeren Fehlereinschließung $\widetilde{I}^{(k)}_{n,k_2(y^\star)} \subseteq I^{(k)}_{n,k_2(y^\star)}$. Ist die Inklusions-

bedingung auch für $\widetilde{I}^{(k)}_{n,k_2(y^\star)}$ erfüllt, gilt also

$$\widetilde{I}^{(k)}_{n,k_2(y^\star)} \subseteq I^{(k)}_{n,y_2^\star},$$

dann ist aufgrund der Inklusionsmonotonie der Operationen

$$\begin{aligned} k_{1\square}\big([\![P_{n,y_1^\star}, I^{(k)}_{n,k_1(y^\star)}]\!], [\![P_{n,y_2^\star}, \widetilde{I}^{(k)}_{n,k_2(y^\star)}]\!]\big) &\subseteq k_{1\square}\big([\![P_{n,y_1^\star}, I^{(k)}_{n,y_1^\star}]\!], [\![P_{n,y_2^\star}, I^{(k)}_{n,y_2^\star}]\!]\big) \\ &= [\![P_{n,y_1^\star}, I^{(k)}_{n,k_1(y^\star)}]\!], \\ k_{2\square}\big([\![P_{n,y_1^\star}, I^{(k)}_{n,k_1(y^\star)}]\!], [\![P_{n,y_2^\star}, \widetilde{I}^{(k)}_{n,k_2(y^\star)}]\!]\big) &\subseteq k_{2\square}\big([\![P_{n,y_1^\star}, I^{(k)}_{n,k_1(y^\star)}]\!], [\![P_{n,y_2^\star}, I^{(k)}_{n,y_2^\star}]\!]\big) \\ &= [\![P_{n,y_2^\star}, \widetilde{I}^{(k)}_{n,k_2(y^\star)}]\!]. \end{aligned}$$

Die Inklusion (3.25) gilt also für die Fehlereinschließungen $I^{(k)}_{n,k_1(y^\star)}$ und $\widetilde{I}^{(k)}_{n,k_2(y^\star)}$. Im Allgemeinen Fall einer Differentialgleichung mit ν Gleichungen hat man entsprechend ν Fehlereinschließungen – gemeint sind die Komponenten der Fehlereinschließung $\boldsymbol{I}_{n,\boldsymbol{y}^\star}$ – zu berechnen und man kann analog für die Berechnung der k-ten ($2 \leqslant k \leqslant \nu$) Fehlereinschließung die schon berechneten $k-1$ Einschließungen verwenden – vorausgesetzt die ersten $k-1$ Inklusionen kamen zustande. Der Vorteil dieser Vorgehensweise liegt auf der Hand. Die gewünschte Inklusion kommt eventuell früher zustande als bei einer Berechnung ohne Berücksichtigung der schon berechneten Einschließungen.

Beispiel 3.2 (Fortsetzung von Beispiel 3.1) *Wir setzen $h_0 = 0.5$ und betrachten das Anfangswertproblem auf dem Intervall $[0, 0.5]$. Die innere Approximation $\boldsymbol{I}^{(0)}_{4,\boldsymbol{K}(\boldsymbol{y}^\star)}$ erhalten wir durch Auswerten von $\boldsymbol{K}_\square$ mit $[\![P_{4,y_1^\star}, [0,0]]\!]$, $[\![P_{4,y_2^\star}, [0,0]]\!]$. Es ist*

$$\begin{aligned} &\boldsymbol{K}_\square\big([\![P_{4,y_1^\star}, [0,0]]\!], [\![P_{4,y_2^\star}, [0,0]]\!]\big) \\ &\qquad = \begin{pmatrix} [\![P_{4,y_1^\star}, [0,0]]\!] \boxplus \boxdot [\![P_{4,y_2^\star}, [0,0]]\!]\, dt \\ [\![P_{4,y_2^\star}, [0,0]]\!] \boxplus \boxdot [\![P_{4,y_1^\star}, [0,0]]\!] \boxdot [\![P_{4,y_1^\star}, [0,0]]\!]\, dt \end{pmatrix} \end{aligned}$$

und wir erhalten als innere Approximation

$$\begin{pmatrix} I^{(0)}_{4,k_1(y^\star)} \\ I^{(0)}_{4,k_2(y^\star)} \end{pmatrix} = \begin{pmatrix} [-4.167\mathrm{E}{-}04, 4.167\mathrm{E}{-}04] \\ [-1.876\mathrm{E}{-}04, 6.050\mathrm{E}{-}04] \end{pmatrix}.$$

Die Intervallgrenzen wurden nach außen gerundet. Für die anschließende ϵ-Inflation verwenden wir die Größen $\epsilon_1 := 10^{-2}$ und $\epsilon_2 := 10^{-6}$. Mit den ersten Kandidaten

$$I^{(1)}_{4,y^\star_1} := (1+\epsilon_1) I^{(0)}_{4,k_1(y^\star)} - \epsilon_1 I^{(0)}_{4,k_1(y^\star)} + [-\epsilon_2, \epsilon_2] = [-4.261\text{E}{-}04, 4.261\text{E}{-}04],$$

$$I^{(1)}_{4,y^\star_2} := (1+\epsilon_1) I^{(0)}_{4,k_2(y^\star)} - \epsilon_1 I^{(0)}_{4,k_2(y^\star)} + [-\epsilon_2, \epsilon_2] = [-1.965\text{E}{-}04, 6.139\text{E}{-}04],$$

erhält man in (3.24) *die Fehlereinschließungen*

$$I^{(1)}_{4,k_1(y^\star)} = [-5.149\text{E}{-}04, 7.237\text{E}{-}04],$$

$$I^{(1)}_{4,k_2(y^\star)} = [-2.521\text{E}{-}04, 6.695\text{E}{-}04].$$

Inklusion liegt noch nicht vor, also berechnen wir die nächste Iterierte

$$I^{(2)}_{4,y^\star_1} := (1+\epsilon_1) I^{(1)}_{4,k_1(y^\star)} - \epsilon_1 I^{(1)}_{4,k_1(y^\star)} + [-\epsilon_2, \epsilon_2] = [-5.283\text{E}{-}04, 7.370\text{E}{-}04],$$

$$I^{(2)}_{4,y^\star_2} := (1+\epsilon_1) I^{(1)}_{4,k_2(y^\star)} - \epsilon_1 I^{(1)}_{4,k_2(y^\star)} + [-\epsilon_2, \epsilon_2] = [-2.623\text{E}{-}04, 6.797\text{E}{-}04].$$

Wir erhalten

$$I^{(2)}_{4,k_1(y^\star)} = [-5.478\text{E}{-}04, 7.566\text{E}{-}04],$$

$$I^{(2)}_{4,k_2(y^\star)} = [-2.983\text{E}{-}04, 7.167\text{E}{-}04].$$

Auch der zweite Schritt liefert keine Inklusion, ebenso der dritte, den wir hier nicht explizit angeben. Erst im vierten Schritt führt

$$I^{(4)}_{4,y^\star_1} := (1+\epsilon_1) I^{(3)}_{4,k_1(y^\star)} - \epsilon_1 I^{(3)}_{4,k_1(y^\star)} + [-\epsilon_2, \epsilon_2] = [-5.859\text{E}{-}04, 7.952\text{E}{-}04],$$

$$I^{(4)}_{4,y^\star_2} := (1+\epsilon_1) I^{(3)}_{4,k_2(y^\star)} - \epsilon_1 I^{(3)}_{4,k_2(y^\star)} + [-\epsilon_2, \epsilon_2] = [-3.146\text{E}{-}04, 7.331\text{E}{-}04],$$

auf

$$I^{(4)}_{4,k_1(y^\star)} = [-5.740\text{E}{-}04, 7.832\text{E}{-}04],$$

$$I^{(4)}_{4,k_2(y^\star)} = [-3.053\text{E}{-}04, 7.238\text{E}{-}04],$$

und damit zur Inklusion

$$\begin{pmatrix} I^{(4)}_{4,k_1(y^\star)} \\ I^{(4)}_{4,k_2(y^\star)} \end{pmatrix} \subseteq \begin{pmatrix} I^{(4)}_{4,y^\star_1} \\ I^{(4)}_{4,y^\star_1} \end{pmatrix}.$$

3.4.3 Fortsetzung der Integration

Wir beschreiben in diesem Abschnitt den zweiten Integrationsschritt, wobei wir uns im wesentlichen darauf beschränken, die Unterschiede zum ersten Integrationsschritt herauszustellen.

Wir nehmen also an, dass im ersten Integrationsschritt eine Lösungseinschließung $\mathcal{E}_{t_1}$ von

$$\mathcal{W}_{t_1} = \left\{ \boldsymbol{y}_{\boldsymbol{\eta}}(t_1) \,|\, \boldsymbol{\eta} \in \boldsymbol{I}_{\boldsymbol{\eta}} \right\}$$

erfolgreich berechnet wurde. Die Menge $\mathcal{E}_{t_1}$ wird nach (3.16) und (3.17) durch das Polynom $\boldsymbol{P}_{n,\boldsymbol{y}^\star}^{(t_1)}$ und das Intervall $\boldsymbol{I}_{n,\boldsymbol{y}^\star}^{(t_1)}$ beschrieben.

Wir formulieren das zu lösende Anfangswertproblem (3.1) an der Stelle t_1. Dies lautet hier

$$\begin{aligned} \boldsymbol{y}'(t) &= \boldsymbol{f}\big(\boldsymbol{y}(t), t\big), \quad t \in [t_1, t_2], \\ \boldsymbol{y}(t_1) &= \boldsymbol{y}_{\boldsymbol{\eta}}(t_1), \quad \boldsymbol{\eta} \in \boldsymbol{I}_{\boldsymbol{\eta}}, \end{aligned} \tag{3.26}$$

mit einem $t_2 = t_1 + h_1 \leqslant t_e$, $h_1 \geqslant h_{min}$. Im Gegensatz zum ersten Integrationsschritt ist die Anfangswertmenge nicht mehr ein Intervall aus $\mathbb{IR}^2$, sondern die Menge $\mathcal{W}_{t_1}$. Da aber die Menge $\mathcal{W}_{t_1}$ nicht bekannt ist, verwenden wir für die praktische Durchführung der Integration die berechnete Einschließung $\mathcal{E}_{t_1}$ als Anfangswertmenge. In der Regel handelt es sich bei $\mathcal{E}_{t_1}$ um eine echte Obermenge von $\mathcal{W}_{t_1}$, sodass wir durch die Verwendung von $\mathcal{E}_{t_1}$ als Anfangswertmenge auch Anfangswerte zulassen, die nicht in $\mathcal{W}_{t_1}$ liegen und wir so de facto ein umfangreicheres System von Anfangswertproblemen lösen. Doch bevor wir auf die praktische Durchführung zu sprechen kommen, wenden wir uns zunächst dem theoretischen Fundament der Lösungseinschließung zu.

In den Ausführungen zu den Voraussetzungen des Banachschen Fixpunktsatzes in Abschnitt 3.3 ändert sich der Definitionsbereich $\boldsymbol{D}$, der Integraloperator $\boldsymbol{K}$ und die Entwicklungsstelle des Taylor-Modells (3.14). Es ist hier

$$\boldsymbol{D} = \boldsymbol{I}_{\boldsymbol{\eta}} \times [t_1, t_2],$$

der Integraloperator $\boldsymbol{K}$ für (3.26) ist gegeben durch

$$\boldsymbol{K}(\boldsymbol{y})(\boldsymbol{\eta}, t) := \boldsymbol{y}^\star(\boldsymbol{\eta}, t_1) + \int_{t_1}^{t} \boldsymbol{f}\big(\boldsymbol{y}(\boldsymbol{\eta}, \tau), \tau\big)\, d\tau$$

und die Entwicklungsstelle des Taylor-Modells (3.14) ändert sich zu

$$\big(m_1, m_2, t_1\big) \in \boldsymbol{D}.$$

Im Verfahren von Makino und Berz ändert sich neben der Entwicklungsstelle der Taylorpolynome auch die Repräsentation des Operators $\boldsymbol{K}$ auf $\mathcal{TM}_n(\boldsymbol{D})^2$. Analog

zu (3.18) wird in $\boldsymbol{K}_{\square}$ der Term $\boldsymbol{y}^{\star}(\boldsymbol{\eta}, t_1)$ durch das Taylor-Modell

$$[\![\boldsymbol{P}_{n,\boldsymbol{y}^{\star}}^{(t_1)}, \boldsymbol{I}_{n,\boldsymbol{y}^{\star}}^{(t_1)}]\!]$$

ersetzt. Die Picard-Iteration wird mit

$$[\![\boldsymbol{P}_{n,\boldsymbol{y}_0}, \boldsymbol{I}_{n,\boldsymbol{y}_0}]\!] := [\![\boldsymbol{P}_{n,\boldsymbol{y}^{\star}}^{(t_1)}, \boldsymbol{I}_{n,\boldsymbol{y}^{\star}}^{(t_1)}]\!]$$

für die Funktion

$$\boldsymbol{y}_0(\eta_1, \eta_2, t) := \boldsymbol{y}^{\star}(\eta_1, \eta_2, t_1)$$

begonnen.

Gelingt es unter Berücksichtigung der eben vorgenommenen Änderungen, eine Menge $[\![\boldsymbol{P}_{n,\boldsymbol{y}^{\star}}, \boldsymbol{I}_{n,\boldsymbol{y}^{\star}}]\!] \in \mathcal{TM}_n(\boldsymbol{D})^2$ mit

$$\boldsymbol{K}\big([\![\boldsymbol{P}_{n,\boldsymbol{y}^{\star}}, \boldsymbol{I}_{n,\boldsymbol{y}^{\star}}]\!]\big) \subseteq [\![\boldsymbol{P}_{n,\boldsymbol{y}^{\star}}, \boldsymbol{I}_{n,\boldsymbol{y}^{\star}}]\!]$$

zu berechnen, dann enthält $[\![\boldsymbol{P}_{n,\boldsymbol{y}^{\star}}, \boldsymbol{I}_{n,\boldsymbol{y}^{\star}}]\!]$ den (eindeutigen) Fixpunkt $\boldsymbol{y}^{\star}$ von $\boldsymbol{K}$ auf $\boldsymbol{D}$ und ist damit eine Einschließung der Funktionenmenge

$$\mathcal{Y}_{[t_1,t_2]} := \big\{\boldsymbol{y}_{\boldsymbol{\eta}} \,\big|\, t \in [t_1, t_2],\ \boldsymbol{\eta} \in \boldsymbol{I}_{\boldsymbol{\eta}}\big\}.$$

In diesem Fall gilt

$$\boldsymbol{y}^{\star}(\eta_1, \eta_2, t) \in \boldsymbol{P}_{n,\boldsymbol{y}^{\star}}\big(\eta_1 - m_1, \eta_2 - m_2, t - t_1\big) + \boldsymbol{I}_{n,\boldsymbol{y}^{\star}}$$

für alle $\boldsymbol{\eta} \in \boldsymbol{I}_{\boldsymbol{\eta}}$, $t \in [t_1, t_2]$. Speziell für $t = t_2$ erhalten wir

$$\boldsymbol{y}^{\star}(\eta_1, \eta_2, t_2) \in \boldsymbol{P}_{n,\boldsymbol{y}^{\star}}\big(\eta_1 - m_1, \eta_2 - m_2, h_1\big) + \boldsymbol{I}_{n,\boldsymbol{y}^{\star}}$$

für alle $\boldsymbol{\eta} \in \boldsymbol{I}_{\boldsymbol{\eta}}$. Analog zum ersten Integrationsschritt bekommen wir mit

$$\boldsymbol{P}_{n,\boldsymbol{y}^{\star}}^{(t_2)}\big(\eta_1 - m_1, \eta_2 - m_2\big) := \boldsymbol{P}_{n,\boldsymbol{y}^{\star}}\big(\eta_1 - m_1, \eta_2 - m_2, h_1\big),$$

$$\boldsymbol{I}_{n,\boldsymbol{y}^{\star}}^{(t_2)} := \boldsymbol{I}_{n,\boldsymbol{y}^{\star}}$$

und

$$\boldsymbol{I}_{\boldsymbol{y}^{\star}(\eta_1,\eta_2,t_2)} := \boldsymbol{P}_{n,\boldsymbol{y}^{\star}}^{(t_2)}\big(\eta_1 - m_1, \eta_2 - m_2\big) + \boldsymbol{I}_{n,\boldsymbol{y}^{\star}}^{(t_2)}, \quad \boldsymbol{\eta} \in \boldsymbol{I}_{\boldsymbol{\eta}}$$

eine simultane Einschließung aller Funktionswerte der Lösungen $\boldsymbol{y}_{\boldsymbol{\eta}}$ an der Stelle t_2 durch

$$\mathcal{E}_{t_2} := \bigcup_{\boldsymbol{\eta} \in \boldsymbol{I}_{\boldsymbol{\eta}}} \boldsymbol{I}_{\boldsymbol{y}^{\star}(\eta_1,\eta_2,t_2)} \supseteq \mathcal{W}_{t_2}.$$

Nachfolgende Integrationsschritte werden analog durchgeführt, d. h. für den $(j+1)$-ten Integrationsschritt $(j > 0)$ ist in diesem Abschnitt t_1 durch t_j und t_2 durch t_{j+1} zu ersetzen.

Insgesamt erhalten wir nach jedem Integrationsschritt eine Einschließung, die durch ein Taylor-Modell abhängig von η_1 und η_2 dargestellt wird. Damit wird der Wrapping-Effekt [29, 46][41, S. 47] auf den (im Vergleich zu herkömmlichen Intervall-Einschließungsverfahren erheblich kleineren) Intervallterm reduziert. Die Darstellung mit Taylor-Modellen erlaubt eine genauere, problemangepasste Darstellung des Flusses, was gegenüber der Parallelepiped-Methode von Lohner [41, S. 48-49] den Vorteil einer geringeren Überschätzung mit sich bringt [46].

3.4.4 Zusammenfassung

Die einzelnen Schritte der simultanen Lösungseinschließung werden hier noch einmal in kompakter Form aufgelistet. Bevor wir das Einschließungsverfahren in Pseudocode formulieren geben wir zunächst eine Beschreibung der Vorgehensweise im Gesamten.

Gesucht ist eine Einschließung der Menge aller Lösungen von (3.1)

$$\begin{aligned} \boldsymbol{y}'(t) &= \boldsymbol{f}\big(\boldsymbol{y}(t), t\big), \quad t \in D_t = [t_0, t_e],\ t_0 < t_e, \\ \boldsymbol{y}(t_0) &= \boldsymbol{\eta} \in \boldsymbol{I}_{\boldsymbol{\eta}} \in \mathbb{IR}^2 \end{aligned}$$

auf D_t, die man erhält, wenn man den Anfangswert $\boldsymbol{\eta}$ im gegebenen Intervall $\boldsymbol{I}_{\boldsymbol{\eta}}$ variiert sowie eine Einschließung der Menge der Funktionswerte $\mathcal{W}_{t_e}$ aller Lösungen von (3.1) an der Stelle t_e. Die Komponentenfunktionen von $\boldsymbol{f}$ seien n mal stetig differenzierbar.

Wir wählen die Ordnung n der Taylorpolynome, eine Mindestschrittweite $h_{min} > 0$, eine Startschrittweite $h_0 \geqslant h_{min}$, die Zahl $k_{max} > 1$ der maximal zulässigen Iterationsschritte und die für die ϵ-Inflation benötigten Größen $\epsilon_1 > 0$, $\epsilon_2 > 0$. Anschließend initialisieren wir die Polynome P_{n,η_1} und P_{n,η_2}, die im ersten Schritt für die Auswertung von $\boldsymbol{K}_{\square}$ benötigt werden. Zusätzlich legen wir $t_1 := t_0 + h_0$ und das Polynom $P_{n,t}$ fest (vgl. Abschnitt 3.4).

Im ersten Integrationsschritt berechnen wir das Taylorpolynom $\boldsymbol{P}_{n,\boldsymbol{y}^\star}$ nach Algorithmus 1. Danach versuchen wir eine Fehlereinschließung $\boldsymbol{I}_{n,\boldsymbol{y}^\star}$ nach Algorithmus 2 zu berechnen. Im Erfolgsfall bestimmen wir $\boldsymbol{P}^{(t_1)}_{n,\boldsymbol{y}^\star}$ und $\boldsymbol{I}^{(t_1)}_{n,\boldsymbol{y}^\star}$.

Vorbereitend für den zweiten Integrationsschritt wählen wir eine Schrittweite $h_1 \geqslant h_{min}$ (wir gehen in Kapitel 4 näher auf die Schrittweitenwahl ein), setzen $t_2 := t_1 + h_1$ und initialisieren das Polynom $P_{n,t}$ neu.

Im zweiten Integrationsschritt starten wir den Algorithmus 1 zur Berechnung des Taylorpolynoms mit dem Taylor-Modell $[\![\boldsymbol{P}^{(t_1)}_{n,\boldsymbol{y}^\star}, \boldsymbol{I}^{(t_1)}_{n,\boldsymbol{y}^\star}]\!]$. Anschließend versuchen wir eine Fehlereinschließung $\boldsymbol{I}_{n,\boldsymbol{y}^\star}$ nach Algorithmus 2 zu berechnen. Dabei ist in Algorithmus 2 t_1 durch t_2, t_0 durch t_1 und h_0 durch h_1 zu ersetzen. Gelingt die Berechnung von $\boldsymbol{I}_{n,\boldsymbol{y}^\star}$, dann bestimmen wir $\boldsymbol{P}^{(t_2)}_{n,\boldsymbol{y}^\star}$ und $\boldsymbol{I}^{(t_2)}_{n,\boldsymbol{y}^\star}$.

Alle weiteren Integrationsschritte werden nun analog zu Schritt zwei durchgeführt (vgl. Abschnitt 3.4.3). Ist das Einschließungsverfahren erfolgreich, d. h.

klappt die Integration bis zur Stelle t_e, dann stellt die Menge $\mathcal{E}_{t_e}$ beziehungsweise das die Menge $\mathcal{E}_{t_e}$ beschreibende Taylor-Modell $\left(\boldsymbol{P}_{n,\boldsymbol{y}^\star}^{(t_e)}, \boldsymbol{I}_{n,\boldsymbol{y}^\star}^{(t_e)}\right)$ eine Einschließung der Menge $\mathcal{W}_{t_e}$ dar.

Das Verfahren scheitert, wenn

a) Lösungen $\boldsymbol{y}_{\boldsymbol{\eta}}$ von (3.1) nicht bis zur Stelle t_e existieren;

b) die Überschätzungen im Einschließungsverfahren so groß sind, dass die Anfangswertmenge $\mathcal{E}_{t_j}$ im $(j+1)$-ten Schritt Werte enthält, für die eine Lösung des zugehörigen Punkt-Anfangswertproblems nicht existiert;

c) die Überschätzungen im Verfahren so groß sind, dass in Algorithmus 2 die Mindestschrittweite h_{min} unterschritten wird.

In den Fällen b) und c) gelingt die Integration eventuell mit veränderten Parametern n, h_{min}, k_{max}, ϵ_1 und ϵ_2.

Algorithmus 3 gibt das Einschließungsverfahren in Pseudocode wieder.

3.5 Ergänzungen zum Einschließungsverfahren

Wir präsentieren in diesem Abschnitt zwei Ergänzungen zum vorgestellten Einschließungsverfahren, die wesentlich zur Qualität der Lösungseinschließung beitragen und die Durchführbarkeit des Verfahrens positiv beeinflussen können.

3.5.1 Verbesserung der Lösungseinschließung

Wir nehmen an, dass mit Algorithmus 2 eine Fehlereinschließung $\boldsymbol{I}_{n,\boldsymbol{y}^\star}$ berechnet werden konnte. In diesem Fall ist der Fixpunkt $\boldsymbol{y}^\star$ in der Menge $[\![\boldsymbol{P}_{n,\boldsymbol{y}^\star}, \boldsymbol{I}_{n,\boldsymbol{y}^\star}]\!]$ enthalten. $\boldsymbol{y}^\star$ ist aber auch – wie man leicht zeigt – in der Menge

$$[\![\boldsymbol{P}_{n,\boldsymbol{y}^\star}, \boldsymbol{I}_{n,\boldsymbol{K}(\boldsymbol{y}^\star)}]\!] = \boldsymbol{K}_{\square}\left([\![\boldsymbol{P}_{n,\boldsymbol{y}^\star}, \boldsymbol{I}_{n,\boldsymbol{y}^\star}]\!]\right)$$

enthalten, die wegen (3.22) eine engere Einschließung von $\boldsymbol{y}^\star$ darstellt. Erneutes Anwenden von $\boldsymbol{K}_{\square}$ auf $[\![\boldsymbol{P}_{n,\boldsymbol{y}^\star}, \boldsymbol{I}_{n,\boldsymbol{K}(\boldsymbol{y}^\star)}]\!]$ liefert eine Menge

$$[\![\boldsymbol{P}_{n,\boldsymbol{y}^\star}, \boldsymbol{I}_{n,\boldsymbol{K}(\boldsymbol{K}(\boldsymbol{y}^\star))}]\!] = \boldsymbol{K}_{\square}\left([\![\boldsymbol{P}_{n,\boldsymbol{y}^\star}, \boldsymbol{I}_{n,\boldsymbol{K}(\boldsymbol{y}^\star)}]\!]\right) \subseteq [\![\boldsymbol{P}_{n,\boldsymbol{y}^\star}, \boldsymbol{I}_{n,\boldsymbol{K}(\boldsymbol{y}^\star)}]\!],$$

die ebenfalls den Fixpunkt enthält. Es gilt

$$\boldsymbol{I}_{n,\boldsymbol{K}(\boldsymbol{K}(\boldsymbol{y}^\star))} \subseteq \boldsymbol{I}_{n,\boldsymbol{K}(\boldsymbol{y}^\star)} \subseteq \boldsymbol{I}_{n,\boldsymbol{y}^\star}.$$

Auf diese Weise kann eine errechnete Einschließung $[\![\boldsymbol{P}_{n,\boldsymbol{y}^\star}, \boldsymbol{I}_{n,\boldsymbol{y}^\star}]\!]$ von $\boldsymbol{y}^\star$ iterativ verbessert werden.

Algorithmus 3 : Einschließungsverfahren

Daten : Das Anfangswertproblem (3.1)
Ergebnis : Entweder eine Einschließung von $\mathcal{W}_{t_e}$ oder Abbruch des Verfahrens

Initialisiere die benötigten Parameter n, h_{min}, h_0, k_{max}, ϵ_1 und ϵ_2;

// Polynome P_{n,η_1} und P_{n,η_2} initialisieren
$m_1 := \mathrm{m}(I_{\eta_1})$, $m_2 := \mathrm{m}(I_{\eta_2})$;
$P_{n,\eta_1}\big(\eta_1 - m_1, \eta_2 - m_2, t - t_0\big) := m_1 + (\eta_1 - m_1)$;
$P_{n,\eta_2}\big(\eta_1 - m_1, \eta_2 - m_2, t - t_0\big) := m_2 + (\eta_2 - m_2)$;

$j := 0$;

while $t_j < t_e$ **do**
  $t_{j+1} := t_j + h_j$;
  if $t_{j+1} > t_e$ **then**
    $t_{j+1} := t_e$;
    $h_j := t_{j+1} - t_j$; // Korrektur der Schrittweite
  end
  // Polynom $P_{n,t}$ initialisieren
  $P_{n,t}\big(\eta_1 - m_1, \eta_2 - m_2, t - t_j\big) := t_j + (t - t_j)$;
  // Taylorpolynom $\boldsymbol{P}_{n,\boldsymbol{y}^\star}$ berechnen:
  // Sobald $j > 0$ setze im Alg. 1 $[\![\boldsymbol{P}_{n,\boldsymbol{y}_0}, \boldsymbol{I}_{n,\boldsymbol{y}_0}]\!] := [\![\boldsymbol{P}^{(t_j)}_{n,\boldsymbol{y}^\star}, \boldsymbol{I}^{(t_j)}_{n,\boldsymbol{y}^\star}]\!]$
  Algorithmus 1;
  // Fehlereinschließung $\boldsymbol{I}_{n,\boldsymbol{y}^\star}$ berechnen:
  // Ersetze im Algorithmus 2 t_1 durch t_{j+1}, t_0 durch t_j und
  // h_0 durch h_j
  Algorithmus 2;
  // Darstellung von $\mathcal{E}_{t_{j+1}}$
  $\boldsymbol{P}^{(t_{j+1})}_{n,\boldsymbol{y}^\star}\big(\eta_1 - m_1, \eta_2 - m_2\big) := \boldsymbol{P}_{n,\boldsymbol{y}^\star}\big(\eta_1 - m_1, \eta_2 - m_2, h_j\big)$; $\boldsymbol{I}^{(t_{j+1})}_{n,\boldsymbol{y}^\star} := \boldsymbol{I}_{n,\boldsymbol{y}^\star}$;
  Wähle $h_{j+1} \geqslant h_{min}$ (z. B. $h_{j+1} := h_j$);
  $j := j + 1$;
end

Wir definieren dazu eine Folge von Intervallen $\boldsymbol{I}^{(j)}_{n,\boldsymbol{y}^\star}$ durch [42, S. 146-147]

$$\begin{aligned} \boldsymbol{I}^{(0)}_{n,\boldsymbol{y}^\star} &:= \boldsymbol{I}_{n,\boldsymbol{y}^\star} \\ [\![\boldsymbol{P}_{n,\boldsymbol{y}^\star}, \boldsymbol{I}^{(j)}_{n,\boldsymbol{y}^\star}]\!] &:= \boldsymbol{K}_{\square}\big([\![\boldsymbol{P}_{n,\boldsymbol{y}^\star}, \boldsymbol{I}^{(j-1)}_{n,\boldsymbol{y}^\star}]\!]\big), \quad j \geqslant 1. \end{aligned} \tag{3.27}$$

Für die resultierenden Intervalle gilt die Inklusion

$$\boldsymbol{I}_{n,\boldsymbol{y}^\star}^{(0)} \supseteq \boldsymbol{I}_{n,\boldsymbol{y}^\star}^{(1)} \supseteq \ldots \supseteq \boldsymbol{I}_{n,\boldsymbol{y}^\star}^{(j-1)} \supseteq \boldsymbol{I}_{n,\boldsymbol{y}^\star}^{(j)},$$

die aus der entsprechenden Inklusionsbeziehung für die dazugehörigen Funktionenmengen folgt [42, S. 147].

Analog zu Makino und Berz beenden wir die Iteration (3.27) sobald in jeder Komponente einer Iterierten die Veränderung in beiden Intervallschranken jeweils bezogen auf den Intervalldurchmesser der vorherigen Iterierten unter eine vorgegebene Schranke $\epsilon_{rel} > 0$ fällt oder aber eine vorgegebene Anzahl $j_{max} > 1$ an Iterationen überschritten wurde.

Wird die Iteration für ein j_0 beendet, dann setzen wir

$$\boldsymbol{I}_{n,\boldsymbol{y}^\star} := \boldsymbol{I}_{n,\boldsymbol{y}^\star}^{(j_0)}.$$

Algorithmus 4 fasst die Vorgehensweise in Pseudocode zusammen.

Analog zur Bemerkung in Abschnitt 3.4.2 wird auch hier der Integraloperator in (3.27) komponentenweise ausgewertet und die schon berechneten, besseren Einschließungen für die Auswertung nachfolgender Komponenten verwendet.

3.5.2 Shrink-Wrapping

Nach Abschnitt 3.4.3 wird im $(j+1)$-ten Integrationsschritt $(j > 0)$ die Einschließung $\mathcal{E}_{t_j}$ als Anfangswertmenge für die bei t_j formulierte Anfangswertaufgabe (3.1) verwendet. Die Menge $\mathcal{E}_{t_j}$ wird durch das Paar

$$\boldsymbol{P}_{n,\boldsymbol{y}^\star}^{(t_j)},\ \boldsymbol{I}_{n,\boldsymbol{y}^\star}^{(t_j)}, \tag{3.28}$$

bestehend aus Polynom und Intervall, beschrieben, das aus der Lösungseinschließung des j-ten Integrationsschrittes durch Einsetzen von $t = t_j$ hervorgeht.

Die Fehlereinschließung $\boldsymbol{I}_{n,\boldsymbol{y}^\star}^{(t_j)}$, die dabei den lokalen Approximationsfehler, Fehler aus vorangegangenen Integrationsschritten und bei Durchführung des Verfahrens auf dem Rechner auch die Rundungsfehler aus Gleitpunktoperationen enthält (hierzu mehr in Kapitel 4), unterliegt dabei dem Wrapping-Effekt [29, 46][41, S. 47], so wie die Intervalle bei herkömmlichen Intervall-Einschließungsverfahren auch. Aufgrund der damit verbundenen negativen Konsequenzen für den Integrationsprozess ist für eine erfolgreiche Integration über große Bereiche D_t hinweg eine geeignete Behandlung des Intervallterms $\boldsymbol{I}_{n,\boldsymbol{y}^\star}^{(t_j)}$ erforderlich. Makino und Berz haben zu diesem Zweck eine Methode entwickelt, die den Namen „Shrink-Wrapping“ trägt [46, S. 15-23].

Für die Methode des Shrink-Wrapping setzen wir voraus, dass die Anfangswerte $\boldsymbol{\eta}$ aus $[-1,1] \times [-1,1]$ sind [46, S. 15]. Dies stellt keine Einschränkung dar, weil

Algorithmus 4 : Verbesserung der Lösungseinschließung

Daten : Die mit Algorithmus 2 berechnete Fehlereinschließung $\boldsymbol{I}_{n,\boldsymbol{y}^\star}$, die Zahlen ϵ_{rel} und j_{max}

Ergebnis : Eine engere Fehlereinschließung $\boldsymbol{I}_{n,\boldsymbol{y}^\star}$

$\boldsymbol{I}^{(0)}_{n,\boldsymbol{y}^\star} := \boldsymbol{I}_{n,\boldsymbol{y}^\star}$;

$j := 0$;

repeat

  $j := j + 1$;

  $[\![\boldsymbol{P}_{n,\boldsymbol{y}^\star}, \boldsymbol{I}^{(j)}_{n,\boldsymbol{y}^\star}]\!] := \boldsymbol{K}_\square\big([\![\boldsymbol{P}_{n,\boldsymbol{y}^\star}, \boldsymbol{I}^{(j-1)}_{n,\boldsymbol{y}^\star}]\!]\big)$;

  `// Berechne in jeder Komponente (hier:`
  `// zwei Komponenten) die relative Veränderung`
  `// in den Intervallschranken`

  $err := 0$;

  for $k := 1$ **to** 2 **do**

    $errinf := \big(\inf(I^{(j)}_{n,y^\star_k}) - \inf(I^{(j-1)}_{n,y^\star_k})\big) / \operatorname{d}\big(I^{(j-1)}_{n,y^\star_k}\big)$;

    $errsup := \big(\sup(I^{(j-1)}_{n,y^\star_k}) - \sup(I^{(j)}_{n,y^\star_k})\big) / \operatorname{d}\big(I^{(j-1)}_{n,y^\star_k}\big)$;

    if $err < \max\{errinf, errsup\}$ **then**

      $err := \max\{errinf, errsup\}$;

    end

  end

until $err < \epsilon_{rel} \ \vee \ j > j_{max}$;

$\boldsymbol{I}_{n,\boldsymbol{y}^\star} := \boldsymbol{I}^{(j)}_{n,\boldsymbol{y}^\star}$;

zum einen die Anfangswertmenge $\boldsymbol{I}_{\boldsymbol{\eta}}$ durch

$$\begin{aligned} \eta_1 &= m_1 + \mathrm{r}(I_{\eta_1}) \cdot \alpha_1, \\ \eta_2 &= m_2 + \mathrm{r}(I_{\eta_2}) \cdot \alpha_2, \end{aligned} \qquad \alpha_1, \alpha_2 \in [-1, 1] \tag{3.29}$$

als Bild von $[-1, 1] \times [-1, 1]$ dargestellt wird, und zum anderen das Einschließungsverfahren es erlaubt, mit Bildmengen polynomialer Abbildungen als Anfangswertmengen zu starten. Kurzum, die Anfangswertmenge im ersten Schritt wird dargestellt als polynomiales Bild von $[-1, 1] \times [-1, 1]$, wie es unabhängig davon auch in allen weiteren Integrationsschritten der Fall ist. Wir verwenden dann im ersten Integrationsschritt innerhalb der Repräsentation $\boldsymbol{K}_\square$ die Taylorpolynome (vgl. Ab-

schnitt 3.4)

$$\begin{aligned} P_{n,\eta_1}(\alpha_1, \alpha_2, t - t_0) &= m_1 + \mathrm{r}(I_{\eta_1}) \cdot \alpha_1, \\ P_{n,\eta_2}(\alpha_1, \alpha_2, t - t_0) &= m_2 + \mathrm{r}(I_{\eta_2}) \cdot \alpha_2 \end{aligned} \tag{3.30}$$

bezüglich der Entwicklungsstelle $(0, 0, t_0)$.

Es ist von nun an $I_{\eta_1} = I_{\eta_2} = [-1, 1]$ und damit $m_1 = m_2 = 0$.

Die Idee des Shrink-Wrapping besteht darin, den Intervallterm $\boldsymbol{I}_{n,\boldsymbol{y}^\star}^{(t_j)}$ durch Verändern der Polynomkoeffizienten in $\boldsymbol{P}_{n,\boldsymbol{y}^\star}^{(t_j)}$ zu absorbieren, d. h. die durch das Polynom und das Intervall aus (3.28) dargestellte Menge $\mathcal{E}_{t_j}$ durch eine polynomiale Abbildung von $[-1, 1] \times [-1, 1]$ einzuschließen. Die Konstruktion dieser Abbildung aus dem vorliegenden Paar (3.28) wird im Folgenden Schritt für Schritt erläutert und anhand von Skizzen veranschaulicht. Es sei dazu

$$\begin{aligned} \boldsymbol{P}(\eta_1, \eta_2) &:= \boldsymbol{P}_{n,\boldsymbol{y}^\star}^{(t_j)}(\eta_1, \eta_2), \\ \boldsymbol{I} &:= \boldsymbol{I}_{n,\boldsymbol{y}^\star}^{(t_j)} \end{aligned}$$

und $\eta_1,\ \eta_2 \in [-1, 1]$.

1. Schritt: Von dem Polynom $\boldsymbol{P}$ wird der konstante Term abgespalten und damit die Menge $\mathcal{E}_{t_j}$ in den Ursprung verschoben. Wir erhalten (Abb. 3.2)

$$\tilde{\mathcal{E}}_{t_j} = \bigcup_{\boldsymbol{\eta}} \left(\tilde{\boldsymbol{P}}(\eta_1, \eta_2) + \boldsymbol{I} \right)$$

mit $\tilde{\boldsymbol{P}}(\eta_1, \eta_2) = \boldsymbol{P}(\eta_1, \eta_2) - \boldsymbol{P}(0, 0)$.

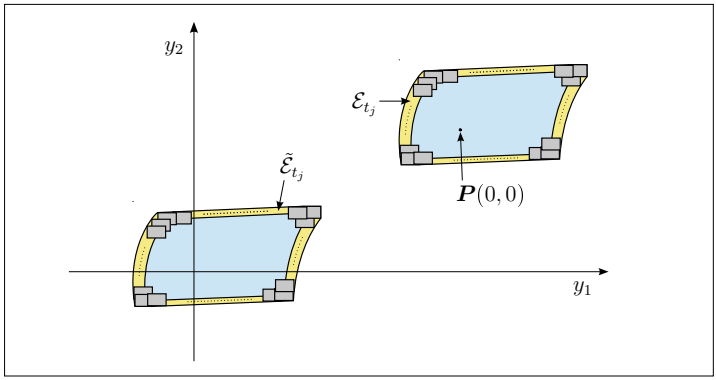

Abbildung 3.2: Skizze zu Schritt 1 des Shrink-Wrapping.

2. Schritt: Wir bestimmen die Koeffizientenmatrix $\boldsymbol{A} \in \mathbb{R}^{2\times 2}$ des linearen Anteils von $\tilde{\boldsymbol{P}}$:

$$\tilde{\boldsymbol{P}}(\eta_1, \eta_2) = \boldsymbol{A} \cdot \begin{pmatrix} \eta_1 \\ \eta_2 \end{pmatrix} + \boldsymbol{p}(\eta_1, \eta_2), \quad \boldsymbol{p} = \begin{pmatrix} p_1 \\ p_2 \end{pmatrix} : [-1,1]^2 \to \mathbb{R}^2.$$

Anschließend berechnen wir ihre Inverse $\boldsymbol{A}^{-1} \in \mathbb{R}^{2\times 2}$. Die Komponenten von $\boldsymbol{p}$ enthalten die Terme zweiter bis n-ter Ordnung des Polynoms $\tilde{\boldsymbol{P}}$ bzw. $\boldsymbol{P}$.

3. Schritt: Wir multiplizieren $\tilde{\boldsymbol{P}}$ von links mit $\boldsymbol{A}^{-1}$ und erhalten

$$\boldsymbol{A}^{-1}\tilde{\boldsymbol{P}}(\eta_1, \eta_2) = \begin{pmatrix} \eta_1 \\ \eta_2 \end{pmatrix} + \boldsymbol{A}^{-1}\boldsymbol{p}(\eta_1, \eta_2)$$

mit dem nichtlinearen Anteil

$$\boldsymbol{q} = \begin{pmatrix} q_1 \\ q_2 \end{pmatrix} := \boldsymbol{A}^{-1}\boldsymbol{p}.$$

Sind die Polynomkoeffizienten in $\boldsymbol{p}$ hinreichend klein, dann ist der Wertebereich $W\left(\boldsymbol{A}^{-1}\tilde{\boldsymbol{P}}\right) \approx [-1,1]^2$ (Abb. 3.3).

4. Schritt: Wir berechnen $\boldsymbol{A}^{-1}\boldsymbol{I}$ und bestimmen ein $\delta \geqslant 0$ mit

$$\boldsymbol{A}^{-1}\boldsymbol{I} \subseteq \delta \cdot [-1,1]^2.$$

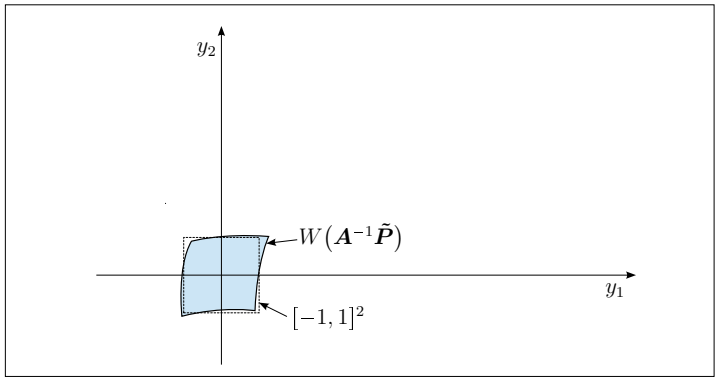

Abbildung 3.3: Skizze zu Schritt 3 des Shrink-Wrapping.

Anschließend berechnen wir für alle $\eta_1,\ \eta_2 \in [-1,1]$ und $i = 1,2$ die Schranken

$$\sigma \geqslant \Big|q_i(\eta_1,\eta_2)\Big|, \quad \tau \geqslant \Big|\frac{\partial q_i}{\partial \eta_1}(\eta_1,\eta_2)\Big|, \quad \tau \geqslant \Big|\frac{\partial q_i}{\partial \eta_2}(\eta_1,\eta_2)\Big|,$$

z. B. mittels Intervallauswertung der Polynome q_i, $\dfrac{\partial q_i}{\partial \eta_1}$, $\dfrac{\partial q_i}{\partial \eta_2}$.

Es bezeichne ν die Anzahl der Variablen von $\boldsymbol{P}$ (hier: $\nu = 2$). Werden die Bedingungen

$$1 - \sigma > 0 \quad \text{und} \quad 1 - \nu\tau > 0 \tag{3.31}$$

von den berechneten Größen σ und τ erfüllt, dann gilt mit dem Shrink-Faktor

$$\kappa := 1 + \frac{\delta}{(1-\sigma)(1-(\nu-1)\tau)}$$

der folgende Satz [46, S. 18]:

Satz 12 (Shrink-Wrapping)
Mit dem Shrink-Faktor κ aus Schritt 4 gilt die Inklusion

$$\bigcup_{\boldsymbol{\eta}} \big(\boldsymbol{A}^{-1}\tilde{\boldsymbol{P}}(\eta_1,\eta_2) + \boldsymbol{A}^{-1}\boldsymbol{I}\big) \ \subset\ \bigcup_{\boldsymbol{\eta}} \kappa\boldsymbol{A}^{-1}\tilde{\boldsymbol{P}}(\eta_1,\eta_2) = W\big(\kappa\boldsymbol{A}^{-1}\tilde{\boldsymbol{P}}\big)$$

d. h. der Wertebereich bzw. die Bildmenge der polynomialen Abbildung $\kappa\boldsymbol{A}^{-1}\tilde{\boldsymbol{P}}$ in $\mathbb{R}^2$ ist eine Obermenge der transformierten Ausgangsmenge $\boldsymbol{A}^{-1}\tilde{\mathcal{E}}_{t_j}$.

Beweis: Siehe Makino und Berz [46, S. 18-20]. □

5. Schritt: Indem wir das Polynom $\kappa\boldsymbol{A}^{-1}\tilde{\boldsymbol{P}}$ mit $\boldsymbol{A}$ multiplizieren

$$\boldsymbol{A}\cdot\big(\kappa\boldsymbol{A}^{-1}\tilde{\boldsymbol{P}}(\eta_1,\eta_2)\big) = \kappa\tilde{\boldsymbol{P}}(\eta_1,\eta_2)$$

und anschließend den Wert $\boldsymbol{P}(0,0)$ hinzu addieren, erhalten wir ein Polynom

$$\hat{\boldsymbol{P}}(\eta_1,\eta_2) := \boldsymbol{P}(0,0) + \kappa\tilde{\boldsymbol{P}}(\eta_1,\eta_2)$$

mit (Abb. 3.4)

$$\mathcal{E}_{t_j} \subset W\big(\hat{\boldsymbol{P}}\big).$$

Wir zeigen dies wie folgt: zu $\boldsymbol{x} \in \mathcal{E}_{t_j}$ beliebig existiert ein $\hat{\boldsymbol{\eta}} \in [-1,1]^2$ mit

$$\boldsymbol{x} \in \boldsymbol{P}(\hat{\eta}_1,\hat{\eta}_2) + \boldsymbol{I},$$

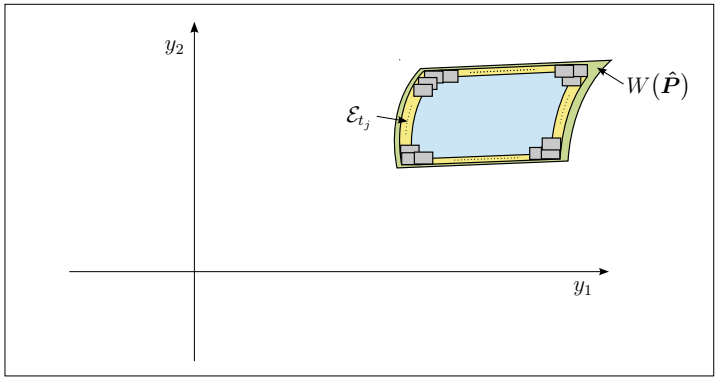

Abbildung 3.4: Skizze zu Schritt 5 des Shrink-Wrapping.

d. h. $\boldsymbol{x} = \boldsymbol{P}(\hat{\eta}_1, \hat{\eta}_2) + \boldsymbol{y}$ mit einem $\boldsymbol{y} \in \boldsymbol{I}$. Ferner gilt

$$\begin{aligned}\boldsymbol{x} &= \boldsymbol{P}(\hat{\eta}_1, \hat{\eta}_2) + \boldsymbol{y} = \tilde{\boldsymbol{P}}(\hat{\eta}_1, \hat{\eta}_2) + \boldsymbol{y} + \boldsymbol{P}(0,0) \\ &= \boldsymbol{A} \cdot \left(\boldsymbol{A}^{-1}\tilde{\boldsymbol{P}}(\hat{\eta}_1, \hat{\eta}_2) + \boldsymbol{A}^{-1}\boldsymbol{y}\right) + \boldsymbol{P}(0,0). \end{aligned} \tag{3.32}$$

Nach Satz 12 ist die Stelle $\boldsymbol{z} := \boldsymbol{A}^{-1}\tilde{\boldsymbol{P}}(\hat{\eta}_1, \hat{\eta}_2) + \boldsymbol{A}^{-1}\boldsymbol{y}$ im Wertebereich $W\left(\kappa\boldsymbol{A}^{-1}\tilde{\boldsymbol{P}}\right)$ von $\kappa\boldsymbol{A}^{-1}\tilde{\boldsymbol{P}}$ echt enthalten. Es existiert also ein $\tilde{\boldsymbol{\eta}} \in [-1,1]^2$ mit

$$\boldsymbol{z} = \kappa\boldsymbol{A}^{-1}\tilde{\boldsymbol{P}}(\tilde{\eta}_1, \tilde{\eta}_2). \tag{3.33}$$

Darüber hinaus besagt Satz 12 aber auch, dass es mindestens ein $\boldsymbol{\eta} \in [-1,1]^2$ gibt, zu dem kein $\boldsymbol{x} \in \mathcal{E}_{t_j}$ gefunden werden kann, sodass (3.33) gilt. Wir erhalten demnach aus (3.32)

$$\begin{aligned}\boldsymbol{x} &= \boldsymbol{A} \cdot \boldsymbol{z} + \boldsymbol{P}(0,0) \\ &= \boldsymbol{A} \cdot \kappa\boldsymbol{A}^{-1}\tilde{\boldsymbol{P}}(\tilde{\eta}_1, \tilde{\eta}_2) + \boldsymbol{P}(0,0) = \kappa\tilde{\boldsymbol{P}}(\tilde{\eta}_1, \tilde{\eta}_2) + \boldsymbol{P}(0,0) \\ &= \hat{\boldsymbol{P}}(\tilde{\eta}_1, \tilde{\eta}_2)\end{aligned}$$

und damit die Behauptung $\mathcal{E}_{t_j} \subset W\left(\hat{\boldsymbol{P}}\right)$.

Die fünf Schritte des Shrink-Wrapping sind in Algorithmus 5 noch einmal in Pseudocode aufgeführt. Darüber hinaus werden in Algorithmus 5 auch mögliche Schwierigkeiten, die bei der Durchführung von Shrink-Wrapping auftreten können und die wir in Abschnitt 3.5.3 diskutieren, berücksichtigt.

Algorithmus 5 : Shrink-Wrapping

Daten : Ein Polynom $\boldsymbol{P}$ und ein Intervall $\boldsymbol{I}$
Ergebnis : Ein Polynom $\hat{\boldsymbol{P}}$ oder Rückgabe eines Fehlercodes

`// 1. Schritt`
$\tilde{\boldsymbol{P}}(\eta_1, \eta_2) := \boldsymbol{P}(\eta_1, \eta_2) - \boldsymbol{P}(0, 0)$;

`// 2. Schritt`
Bestimme $\boldsymbol{A}$ und $\boldsymbol{p}$;
if $\boldsymbol{A}$ *regulär* **then**
  Berechne $\boldsymbol{A}^{-1}$;
else
  return *error* := 2;
end

`// 3. Schritt`
$\boldsymbol{q} := \boldsymbol{A}^{-1}\boldsymbol{p}$;

`// 4. Schritt`
Berechne $\boldsymbol{A}^{-1}\boldsymbol{I}$ und daraus δ;
Berechne σ und τ;
$\kappa := 2$;
if $1 - \sigma > 0 \ \wedge \ 1 - \nu\tau > 0$ **then**
  $\kappa := 1 + \dfrac{\delta}{(1-\sigma)(1-(\nu-1)\tau)}$;
end
if $\kappa \geqslant 1.01$ **then**
  return *error* := 1;
end

`// 5. Schritt`
$\hat{\boldsymbol{P}}(\eta_1, \eta_2) := \boldsymbol{P}(0, 0) + \kappa\tilde{\boldsymbol{P}}(\eta_1, \eta_2)$;

Kommen wir zurück zum Integrationsprozess und betrachten den Fall, dass am Ende des j-ten Integrationsschrittes für das Paar (3.28), welches die Einschließung $\mathcal{E}_{t_j}$ beschreibt, Shrink-Wrapping erfolgreich durchgeführt und ein Polynom $\hat{\boldsymbol{P}}^{(t_j)}$ mit

$$\mathcal{E}_{t_j} \subset W\big(\hat{\boldsymbol{P}}^{(t_j)}\big)$$

berechnet werden konnte. In diesem Fall wird im darauf folgenden Integrationsschritt die Menge $W\big(\hat{\boldsymbol{P}}^{(t_j)}\big)$ als Anfangswertmenge verwendet und in der Reprä-

sentation $\boldsymbol{K}_{\square}$ der Term $\boldsymbol{y}^{\star}(\boldsymbol{\eta}, t_j)$ durch

$$[\![\hat{\boldsymbol{P}}^{(t_j)}, \begin{pmatrix}[0,0]\\[0,0]\end{pmatrix}]\!]$$

ersetzt. Die Picard-Iteration wird entsprechend mit

$$[\![\boldsymbol{P}_{n,\boldsymbol{y}_0}, \boldsymbol{I}_{n,\boldsymbol{y}_0}]\!] := [\![\hat{\boldsymbol{P}}^{(t_j)}, \begin{pmatrix}[0,0]\\[0,0]\end{pmatrix}]\!]$$

gestartet.

Die zunächst als Nachteil erscheinende Tatsache, dass nach erfolgreichem Shrink-Wrapping mit einer absichtlich vergrößerten Anfangswertmenge weitergerechnet wird, entpuppt sich bei diesem Einschließungsverfahren infolge der Darstellung der Anfangswertmenge als Vorteil. Die Verbindung von Polynom- mit Intervallarithmetik innerhalb des Verfahrens bewirkt, dass im Falle eines nicht vorhandenen Intervallterms das Abhängigkeitsproblem (vgl. Kapitel 1) deutlich reduziert wird. Auch der Einfluss des Wrapping-Effekt wird durch den absorbierten Intervallterm vermindert.

3.5.3 Durchführbarkeit von Shrink-Wrapping

Wie Algorithmus 5 schon zeigt, kann die Durchführung des Shrink-Wrapping an zwei Stellen scheitern. Das ist zum einen die Invertierung der Matrix $\boldsymbol{A}$ und zum zweiten sind es die Bedingungen (3.31) für die Schranken σ und τ. Für das Ausmaß der Überschätzung ist zusätzlich noch die Größe des Shrink-Faktors von Bedeutung, weshalb in Algorithmus 5 auch bei einem zu großen Shrink-Faktor κ ausgestiegen wird. Wir stellen im Folgenden eine von Makino und Berz erdachte Heuristik vor, die zur Steuerung des Shrink-Wrapping verwendet werden kann[2].

Ist die Matrix $\boldsymbol{A}$ nicht invertierbar, so könnte man wie Lohner [41, S. 49-50] den orthogonalen Anteil der QR-Zerlegung von $\boldsymbol{A}$ verwenden. Diese extreme Methode kann aber dazu führen, dass anschließend die Größen σ und τ zu groß werden, und dadurch das Shrink-Wrapping an dieser Stelle scheitert. Eine sanftere Modifikation von $\boldsymbol{A}$ erreicht man durch eine geeignete Vergrößerung der Winkel der Spaltenvektoren aus $\boldsymbol{A}$, was Lohner in [41, S. 51] anspricht und Makino und Berz in [46, S. 20-22] unter der Bezeichnung *blunting* diskutieren. Daneben gibt es aber auch noch die Möglichkeit, die Anfangswertmenge in ein Intervall aus $\mathbb{IR}^2$ einzuschließen und die Integration damit fortzusetzen.

Sind die Bedingungen (3.31) an die Schranken σ und τ nicht erfüllt, dann liegt das an einem zu großen Einfluss des nichtlinearen Anteils $\boldsymbol{q}$. Makino und Berz verfahren in diesem Fall so, dass sie zunächst das Volumen der zu absorbierenden

[2] Diese Heuristik stammt aus einer Implementierung des Shrink-Wrapping in Software. Näheres dazu in Kapitel 4.

Fehlereinschließung $\boldsymbol{I}$ mit dem Volumen des durch die Spalten von $\boldsymbol{A}$ aufgespannten Parallelepipeds vergleichen. Ist das Volumen von $\boldsymbol{I}$ kleiner als das 0.01^{ν}-fache des Parallelepipedvolumens (ν bezeichnet die Anzahl der Variablen von $\boldsymbol{P}$; hier: $\nu = 2$), dann ist nach Makino und Berz der Intervallterm noch klein genug, um die Integration auch ohne Shrink-Wrapping fortführen zu können. Ist das Volumen von $\boldsymbol{I}$ größer, dann wird die Anfangswertmenge $\mathcal{E}_{t_j}$ sowohl in ein Intervall aus $\mathbb{IR}^2$ eingeschlossen als auch aus der Darstellung durch $\boldsymbol{P}$ und $\boldsymbol{I}$ eine Parallelepipedeinschließung von $\mathcal{E}_{t_j}$ berechnet. Für die Fortsetzung der Integration kommt dann die Einschließung mit dem geringsten Volumen zum Einsatz.

Die Einschließung von $\mathcal{E}_{t_j}$ in ein Parallelepiped kann über Shrink-Wrapping bestimmt werden. Hierzu wird der nichtlineare Anteil $\boldsymbol{p}$ im Intervallterm $\boldsymbol{I}$ absorbiert [46, S. 20], d. h. es wird eine Wertebereichseinschließung $\boldsymbol{I_p} \in \mathbb{IR}^2$ von $\boldsymbol{p}$ berechnet und diese zum Intervallterm $\boldsymbol{I}$ hinzuaddiert. Anschließend wird Shrink-Wrapping erneut auf das Paar $\bar{\boldsymbol{P}}$, $\boldsymbol{I} + \boldsymbol{I_p}$ mit dem jetzt linearen Polynom

$$\bar{\boldsymbol{P}}(\eta_1, \eta_2) = \boldsymbol{A} \cdot \begin{pmatrix} \eta_1 \\ \eta_2 \end{pmatrix}$$

angewendet.

Wird der nichtlineare Anteil $\boldsymbol{p}$ wie eben beschrieben im Intervallterm $\boldsymbol{I}$ absorbiert, dann berechnet sich im 4. Schritt des Shrink-Wrapping die Fehlereinschließung zu

$$\boldsymbol{A}^{-1}\left(\boldsymbol{I} + \boldsymbol{I_p}\right) = \boldsymbol{A}^{-1}\boldsymbol{I} + \boldsymbol{A}^{-1}\boldsymbol{I_p}. \tag{3.34}$$

Da $\boldsymbol{A}$ eine reelle Matrix ist, haben wir in (3.34) Gleichheit anstelle einer Inklusion (Subdistributivität) vorliegen [1, S. 122].

Die Berechnung des zweiten Summanden in (3.34) eröffnet uns eine zur Vorgehensweise von Makino und Berz alternative Berechnungsmöglichkeit. Wir schlagen vor, nicht $\boldsymbol{p}$ im Intervallterm $\boldsymbol{I}$, sondern den transformierten nichtlinearen Anteil $\boldsymbol{q} = \boldsymbol{A}^{-1}\boldsymbol{p}$ im Intervallterm $\boldsymbol{A}^{-1}\boldsymbol{I}$ zu absorbieren, also im 4. Schritt des Shrink-Wrapping die Fehlereinschließung

$$\boldsymbol{A}^{-1}\boldsymbol{I} + \boldsymbol{I}_{\boldsymbol{A^{-1}p}} \tag{3.35}$$

zu berechnen, wobei $\boldsymbol{I}_{\boldsymbol{A^{-1}p}} \in \mathbb{IR}^2$ eine Wertebereichseinschließung von $\boldsymbol{q}$ darstellt. Wir erhalten auch in diesem Fall $\sigma = \tau = 0$, jedoch kann wegen

$$W(\boldsymbol{q}) = \boldsymbol{A}^{-1}W(\boldsymbol{p}) \subseteq \boldsymbol{A}^{-1}\begin{pmatrix} W(p_1) \\ W(p_2) \end{pmatrix},$$

die Inklusion $\boldsymbol{I}_{\boldsymbol{A^{-1}p}} \subseteq \boldsymbol{A}^{-1}\boldsymbol{I_p}$ gelten, was dann im 4. Schritt eine engere Fehlereinschließung als (3.34) und damit ein kleineres δ bzw. κ zur Folge hat.

Im Falle, dass Shrink-Wrapping durchgeführt werden konnte, aber der Shrink-Faktor eine vorgegebenen Wert überschritten hat, verfahren Makino und Berz wie im Fall der Nichterfüllung von (3.31). Algorithmus 6 gibt die Struktur dieser Heuristik in Pseudocode wieder.

3.5.4 Umsetzung im Gesamtalgorithmus

In diesem Abschnitt präsentieren wir abschließend den Algorithmus des Einschließungsverfahrens aus Abschnitt 3.4.4 ergänzt um die eben vorgestellten Erweiterungen. Dabei ist zu beachten, dass für die Einbindung des Shrink-Wrapping die Anfangswertmenge $\boldsymbol{I_\eta}$ zu Beginn des Einschließungsverfahrens als polynomiales Bild von $[-1,1]^2$ dargestellt werden muss. Wir verwenden folglich im ersten Integrationsschritt (vgl. Abschnitt 3.5.2)

$$P_{n,\eta_i}(\alpha_1, \alpha_2, t - t_0) = m_i + \mathrm{r}(I_{\eta_i}) \cdot \alpha_i, \quad \alpha_i \in [-1,1],\ i = 1,2.$$

Algorithmus 7 zeigt das erweiterte Einschließungsverfahren in Pseudocode.

Algorithmus 6 : Steuerung des Shrink-Wrapping-Prozesses

Daten : Im j-ten Schritt das Paar $\boldsymbol{P}_{n,\boldsymbol{y}^\star}^{(t_j)}$, $\boldsymbol{I}_{n,\boldsymbol{y}^\star}^{(t_j)}$
Ergebnis : Ein Polynom $\hat{\boldsymbol{P}}^{(t_j)}$ und ein Intervall $\hat{\boldsymbol{I}}^{(t_j)}$

$error := 0$;
$\boldsymbol{P}(\eta_1, \eta_2) := \boldsymbol{P}_{n,\boldsymbol{y}^\star}^{(t_j)}(\eta_1, \eta_2)$;
$\boldsymbol{I} := \boldsymbol{I}_{n,\boldsymbol{y}^\star}^{(t_j)}$;
`// Versuche Shrink-Wrapping. Ergebnis:` $\hat{\boldsymbol{P}}$ `oder Fehlercode` $error$`.`
Algorithmus 5;
if $error = 0$ **then**
  $\hat{\boldsymbol{P}}^{(t_j)} := \hat{\boldsymbol{P}}$; $\hat{\boldsymbol{I}}^{(t_j)} := \begin{pmatrix}[0,0]\\[0,0]\end{pmatrix}$; `// Shrink-Wrapping war erfolgreich`
else
  Berechne Volumen V_I von $\boldsymbol{I}$;
  Berechne Volumen V_A des Parallelepipeds gegeben durch $\boldsymbol{A}$;
  if $V_I < 0.01^\nu \cdot V_A$ **then**
    $\hat{\boldsymbol{P}}^{(t_j)} := \boldsymbol{P}$; $\hat{\boldsymbol{I}}^{(t_j)} := \boldsymbol{I}$; `// Fahre ohne Shrink-Wrapping fort`
  else
    `// Intervalleinschließung`
    Berechne Intervalleinschließung von $\mathcal{E}_{t_j}$ und deren Volumen V_{Box};
    `// Parallelepipedeinschließung`
    if $error = 2$ **then**
      $V_{Par} := V_{Box} + 1$; `// Parallelepipedeinschl. nicht möglich`
    else
      Berechne Parallelepipedeinschließung von $\mathcal{E}_{t_j}$ und deren Volumen V_{Par};
    end
    if $V_{Box} < V_{Par}$ **then**
      Bestimme $\hat{\boldsymbol{P}}^{(t_j)}$ aus Intervalleinschließung;
    else
      Bestimme $\hat{\boldsymbol{P}}^{(t_j)}$ aus Parallelepipedeinschließung;
    end
    $\hat{\boldsymbol{I}}^{(t_j)} := \begin{pmatrix}[0,0]\\[0,0]\end{pmatrix}$;
  end
end

Algorithmus 7 : Erweitertes Einschließungsverfahren

Daten : Das Problem (3.1)
Ergebnis : Entweder eine Einschließung von $\mathcal{W}_{t_e}$ oder Abbruch des Verfahrens

Initialisiere die benötigten Parameter n, h_{min}, h_0, k_{max}, j_{max}, ϵ_{rel}, ϵ_1 und ϵ_2;

// Stelle $\boldsymbol{I}_{\boldsymbol{\eta}}$ als polynomiales Bild von $[-1,1]^2$ dar und
// initialisiere die Polynome P_{n,η_1} und P_{n,η_2} entsprechend
$P_{n,\eta_1}\big(\alpha_1,\alpha_2,t-t_0\big) := \mathrm{m}(I_{\eta_1}) + \mathrm{r}(I_{\eta_1})\cdot\alpha_1$;
$P_{n,\eta_2}\big(\alpha_1,\alpha_2,t-t_0\big) := \mathrm{m}(I_{\eta_2}) + \mathrm{r}(I_{\eta_2})\cdot\alpha_2$;

$j := 0$;
while $t_j < t_e$ **do**
  $t_{j+1} := t_j + h_j$;
  if $t_{j+1} > t_e$ **then**
    $t_{j+1} := t_e$;
    $h_j := t_{j+1} - t_j$; // Korrektur der Schrittweite
  end
  // Polynom $P_{n,t}$ initialisieren
  $P_{n,t}\big(\alpha_1,\alpha_2,t-t_j\big) := t_j + (t-t_j)$;
  // Taylorpolynom $\boldsymbol{P}_{n,\boldsymbol{y}^\star}$ berechnen:
  // Sobald $j>0$ setze im Alg. 1 $[\![\boldsymbol{P}_{n,\boldsymbol{y}_0},\boldsymbol{I}_{n,\boldsymbol{y}_0}]\!] := [\![\hat{\boldsymbol{P}}^{(t_j)},\hat{\boldsymbol{I}}^{(t_j)}]\!]$
  Algorithmus 1;
  // Fehlereinschließung $\boldsymbol{I}_{n,\boldsymbol{y}^\star}$ berechnen:
  // Ersetze im Algorithmus 2 t_1 durch t_{j+1}, t_0 durch t_j und
  // h_0 durch h_j
  Algorithmus 2;
  // Verbessere die berechnete Fehlereinschließung
  Algorithmus 4;
  // Darstellung von $\mathcal{E}_{t_{j+1}}$
  $\boldsymbol{P}^{(t_{j+1})}_{n,\boldsymbol{y}^\star}\big(\alpha_1,\alpha_2\big) := \boldsymbol{P}_{n,\boldsymbol{y}^\star}\big(\alpha_1,\alpha_2,h_j\big)$; $\boldsymbol{I}^{(t_{j+1})}_{n,\boldsymbol{y}^\star} := \boldsymbol{I}_{n,\boldsymbol{y}^\star}$;
  // Führe Shrink-Wrapping durch sofern Ende nicht schon
  // erreicht
  if $t_{j+1} \neq t_e$ **then**
    Algorithmus 6;
  end
  Wähle $h_{j+1} \geqslant h_{min}$ (z. B. $h_{j+1} := h_j$);
  $j := j+1$;
end

4 Implementierung

Wenn man zu verstehen gelernt hat,
fürchtet man nichts mehr.

Marie Curie (1867-1934)

Die Taylor-Modelle wie auch die Algorithmen zur Lösungseinschließung bei Differentialgleichungen wurden schon vor der Entstehung dieser Arbeit von Berz und seinen Mitarbeitern in Software realisiert. Aus dieser Implementierung ging das Basispaket COSY INFINITY [6, 8][42, S. 103-110] hervor, das in der Programmiersprache Fortran77 geschrieben wurde. Es stellt dem Benutzer neben einer Vielzahl an Routinen zur Lösung verschiedener physikalischer Probleme, die für diese Arbeit aber keine Bedeutung haben, das Konzept der Taylor-Modelle zur Verfügung. Der Differentialgleichungscode wurde hingegen in einem Zusatzpaket namens COSY-VI verwirklicht. Dieses ist allerdings nur schwer erhältlich und darf ohne Zustimmung der Autoren nicht verändert oder weitergegeben werden. Insgesamt handelt es sich bei diesen zwei Paketen um ein mächtiges und äußerst schnelles Tool, mit dem numerische Probleme im Bereich der Differentialgleichungen sowohl verifiziert als auch nicht verifiziert gelöst werden können.

Darüber hinaus wurde die korrekte Funktionsweise der Implementierung von COSY INFINITY durch verschiedene Personen unabhängig voneinander überprüft [45, 55]. Unter anderem wurden hierbei kritische Stellen wie die Implementierung der Intervallarithmetik oder die Behandlung von Rundungsfehlern bei Polynomoperationen getestet.

Leider ist das Paket COSY INFINITY nur unzureichend dokumentiert, was mitunter zu Diskussionen über den Nutzen der Taylor-Modelle beigetragen hat [siehe 51]. Zwar ist der Programmcode an einigen Stellen mit Kommentaren versehen, doch sind diese wenig hilfreich, wenn es darum geht, den Code zu verstehen bzw. aus diesem z. B. etwas über die Rundungsfehlerbehandlung oder die Wertebereichseinschließung von Polynomen in Erfahrung zu bringen. Der folgende, zufällig gewählte Codeausschnitt aus der Datei `dafox.f` des COSY INFINITY Programmpakets verdeutlicht dies beispielhaft.

```
      CALL INCOPO(CC(IA),CC(IA+1),CCR1,CCR2)
*
      IC = NBRDC+2*(NORD+1)
```

```
      NC(IC+1) = NVRD
      CALL INMINO(CCR1,CCR2,CCV1,CCV2,CC(IC),CC(IC+1))
      CALL INAINO(COBN1,COBN2,CC(IC),CC(IC+1),CC(IC),CC(IC+1))
*
      IC = IC+2
      NC(IC) = NEDAC-NBEG(INC)+1
      CC(IC) = IMD
*
      IF(IMD.GT.0) THEN
         CALL INFLOO(-RDT0,RDT0,CC1,CC2)
         CALL INMREO(CC1,CC2,2.D0*EPSM,CC1,CC2)
         CALL INAINO(CC(IC-2),CC(IC-1),CC1,CC2,CC(IC-2),CC(IC-1))
         CALL INFLOO(-RDS0,RDS0,CC1,CC2)
         CALL INMREO(CC1,CC2,2.D0,CC1,CC2)
         CALL INAINO(CC(IC-2),CC(IC-1),CC1,CC2,CC(IC-2),CC(IC-1))
      ENDIF
```

Ein wesentliches Ziel der vorliegenden Arbeit war, ein frei zugängliches Computerprogramm zu entwerfen, welches weitergegeben und verändert werden darf. Das Programm sollte möglichst gut strukturiert sein, wobei sich die Implementierung gegenüber Erweiterungen und Änderungen flexibel zeigen sollte. Außerdem sollte eine mit der Implementierung konforme Dokumentation angefertigt werden.

Zur Realisierung wurden objektorientierte Konzepte herangezogen. Bevor wir aber die Struktur und Funktionsweise der erstellten Software beschreiben, gehen wir zunächst in Abschnitt 4.1 auf die Darstellung von Taylor-Modellen auf dem Computer ein und geben anschließend in Abschnitt 4.2 eine kurze Einführung in die objektorientierte Softwareentwicklung, deren Konzepte und Notation. In den Abschnitten 4.3, 4.4 und 4.5 folgt dann eine Beschreibung des erstellten Programms.

Für die Programmierung wurde die Sprache C++ verwendet. Die Übersetzung der Software wurde mit dem GNU GCC Compiler der Version 4.0.2 unter Ubuntu Linux 5.10 vorgenommen. Die Software steht unter der GNU General Public License [17].

4.1 Taylor-Modelle auf der Maschine

Wie bei jeder Implementierung mathematischer Verfahren werden auch hier mathematische Konzepte auf den Computer übertragen. Reelle Zahlen werden durch Gleitkommazahlen ersetzt, Intervalle werden durch Maschinenintervalle repräsentiert. Auch der Begriff des Taylor-Modells ist in geeigneter Weise auf den Rechner zu übertragen. Bevor wir darauf näher eingehen, geben wir zunächst einen Einblick in die Repräsentation reeller Zahlen und reeller Intervalle auf dem Computer. Eine ausführliche Beschreibung der Rechnerarithmetik geben z. B. Kulisch [35] sowie Kulisch und Miranker [36].

Es ist allgemein bekannt, dass auf einem Computer nur eine endliche Zahlenmenge für die Darstellung reeller Zahlen zur Verfügung steht. Diese Zahlenmenge wird im Allgemeinen durch Gleitkommazahlen

$$x = \pm\, m \cdot b^e$$

zur Basis $b \in \mathbb{N}$ mit einer Mantisse

$$m = 0.d_1 d_2 \ldots d_l, \quad 1 \leqslant d_1 \leqslant b-1,\ 0 \leqslant d_i \leqslant b-1,\ i = 2, \ldots, l$$

der Länge $l \in \mathbb{N}$ in normalisierter Darstellung ($d_1 \neq 0$) und Exponenten $e \in \mathbb{Z}$ mit

$$e_{min} \leqslant e \leqslant e_{max}, \quad e_{min} < 0,\ e_{max} > 1,\ e_{min}, e_{max} \in \mathbb{Z}$$

gebildet. Wir kennzeichnen die Menge der obigen Gleitkommazahlen, die eine Teilmenge von $\mathbb{R}$ ist, mit $S = S(b, l, e_{min}, e_{max})$. S wird häufig auch als Gleitkommasystem oder Gleitkommaraster oder einfach nur als Raster bezeichnet [35].

Die Approximation reeller Zahlen $x \in \mathbb{R}$ durch Gleitkommazahlen $\tilde{x} \in S$ wird über eine Rundung vollzogen. Dies ist eine Abbildung

$$fl : \begin{cases} \mathbb{R} & \to & S \\ x & \mapsto & \tilde{x} = fl(x) \end{cases}$$

mit der Eigenschaft

$$x_1, x_2 \in \mathbb{R} : \quad x_1 \leqslant x_2 \ \Rightarrow\ fl(x_1) \leqslant fl(x_2)$$

der Monotonie. Gilt für eine Rundung fl zusätzlich noch die Eigenschaft

$$x \in S \ \Rightarrow\ fl(x) = x$$

der Rasterinvarianz, so wird fl optimale Rundung genannt [1, S. 39-40].

Werden zwei Gleitkommazahlen x_1, x_2 aus S durch eine arithmetische Operation $\circ \in \{+, -, \cdot, /\}$ aus $\mathbb{R}$ miteinander verknüpft, dann liegt das Ergebnis $x_1 \circ x_2$ im Allgemeinen nicht wieder in S. Im Zuge der Abbildung der reellen Zahlen auf den Computer sind also auch die arithmetischen Grundoperationen auf dem Computer geeignet zu approximieren. Man definiert für Gleitkommazahlen x_1, $x_2 \in S$

$$x_1 \circ_{fl} x_2 := fl(x_1 \circ x_2),$$

d. h. die durchzuführende Verknüpfung wird zunächst in $\mathbb{R}$ ausgeführt, anschließend wird das Verknüpfungsergebnis über die Rundung fl nach S abgebildet. Analog dazu wird eine Standardfunktion $\varphi \in \mathcal{SF}$ für Gleitkommazahlen $x \in S$ durch

$$\varphi_{fl}(x) := fl\big(\varphi(x)\big)$$

auf dem Computer definiert. Entsprechende Algorithmen hierzu, wie auch Informationen zur Implementierung findet der interessierte Leser in [49].

Maschinenintervalle sind reelle Intervalle, deren Schranken Gleitkommazahlen aus S sind. Wir bezeichnen die Menge aller Maschinenintervalle mit $\mathbb{IS}$. Es ist $\mathbb{IS}$ eine Teilmenge von $\mathbb{IR}$. Möchte man die reellen Intervalle auf das Gleitkommaraster abbilden, dann muss die zu definierende Intervall-Rundung

$$\updownarrow\colon \begin{cases} \mathbb{IR} & \to & \mathbb{IS} \\ I & \mapsto & \tilde{I} = \updownarrow I \end{cases}$$

sinnvollerweise die Eigenschaften

$$I \in \mathbb{IR} \;\Rightarrow\; I \subseteq \updownarrow I \tag{4.1}$$

und

$$I_1, I_2 \in \mathbb{IR}: \quad I_1 \subseteq I_2 \;\Rightarrow \updownarrow I_1 \subseteq \updownarrow I_2$$

erfüllen. Nur so ist sichergestellt, dass die grundlegenden Merkmale der Intervallrechnung auf den Computer übertragen werden. Damit dies gelingt benötigt man den Begriff der gerichteten Rundung. Eine Rundung $\downarrow$ mit der Eigenschaft

$$x \in \mathbb{R} \;\Rightarrow \downarrow x \leqslant x$$

heißt nach unten gerichtete Rundung. Analog dazu heißt eine Rundung $\uparrow$ mit der Eigenschaft

$$x \in \mathbb{R} \;\Rightarrow\; x \leqslant \uparrow x$$

nach oben gerichtete Rundung. Die Intervall-Rundung wird dann durch

$$\updownarrow I = \updownarrow [\underline{x}, \overline{x}] := [\downarrow \underline{x}, \uparrow \overline{x}]$$

definiert [1, S. 40].

Analog zu den Gleitkommazahlen definiert man die arithmetischen Operationen für Maschinenintervalle I_1, $I_2 \in \mathbb{IS}$ über die Intervall-Rundung durch

$$I_1 \circ_{fl} I_2 := \updownarrow (I_1 \circ I_2), \quad \circ \in \{+, -, \cdot, /\}. \tag{4.2}$$

Wir kennzeichnen auch hier die Maschinenoperationen mit *fl* im Index. Man erkennt an den Operanden, welche Operation gemeint ist und für den Fall, dass eine Gleitkommazahl $x \in S$ mit einem Maschinenintervall verknüpft werden soll, wird die Gleitkommazahl als Punktintervall aufgefasst. Wir verwenden für ein Punktintervall die Schreibweise $[x]$. Die nach (4.2) definierten Verknüpfungen sind Inklusionsmonoton. Außerdem gilt

$$I_1 \circ I_2 \subseteq I_1 \circ_{fl} I_2.$$

Ist überdies für die Rundung fl die Ungleichung

$$\downarrow x \leqslant fl(x) \leqslant \uparrow x, \quad x \in \mathbb{R}$$

immer erfüllt, so gilt für Gleitkommazahlen $x_1, x_2 \in S$ die Inklusion [1, S. 41]

$$x_1 \circ_{fl} x_2 \in [x_1] \circ_{fl} [x_2].$$

Ein Taylor-Modell ist ein Konstrukt, das sich aus mehreren Komponenten zusammensetzt. Dazu zählen – wir beschränken uns wie in Kapitel 2 auf zwei Veränderliche –

a) der Definitionsbereich $\boldsymbol{D} = [\underline{s}, \overline{s}] \times [\underline{t}, \overline{t}]$, der für die Berechnung der Fehlereinschließung wichtig ist (vgl. Def. 3);

b) die Entwicklungsstelle $(s_0, t_0) \in \boldsymbol{D}$;

c) die Polynomordnung n;

d) das Taylorpolynom $P_{n,y}$ einer Funktion $y \in C^{n+1}(\boldsymbol{D})$ (vgl. (2.1));

e) die Fehlereinschließung $I_{n,y} \in \mathbb{IR}$.

Will man das Gebilde Taylor-Modell auf den Computer übertragen und damit funktionale Probleme verifiziert lösen, dann sind die genannten Komponenten geeignet auf die Maschine abzubilden. Die Funktionenmenge $[\![P_{n,y}, I_{n,y}]\!]$ (vgl. Definition 4, S. 11) muss sich durch die abgebildeten Komponenten auf der Maschine ebenfalls darstellen bzw. im Mengensinne einschließen lassen.

Wir setzen voraus, dass der Definitionsbereich $\boldsymbol{D}$ im Gleitkommaraster darstellbar ist, d. h. dass die Intervallschranken Gleitkommazahlen sind. Dies stellt keine Einschränkung dar, da dies leicht durch eine geeignete Skalierung der Variablen erreicht werden kann. Die Entwicklungsstelle (s_0, t_0) ersetzen wir auf der Maschine durch das Tupel $\big(fl(s_0), fl(t_0)\big)$. Die Polynomordnung n ist als natürliche Zahl ein Element des Gleitkommarasters S und kann demnach im praktisch relevanten Fall unverändert auf dem Computer gespeichert werden. Wir gehen als Nächstes auf die Maschinen-Repräsentation der Komponenten Taylorpolynom und Fehlereinschließung ein.

Das Taylorpolynom

$$P_{n,y}(s - s_0, t - t_0) = \sum_{i+j \leqslant n} c_{ij} (s - s_0)^i (t - t_0)^j$$

wird auf dem Computer durch ein Polynom

$$\tilde{P}_{n,y}\big(s - fl(s_0), t - fl(t_0)\big) = \sum_{i+j \leqslant n} \tilde{c}_{ij} \big(s - fl(s_0)\big)^i \big(t - fl(t_0)\big)^j$$

mit Gleitkommazahlen $\tilde{c}_{ij} \in S$ als Koeffizienten ersetzt. Die genaue Darstellung bzw. Speicherung eines Polynoms auf der Maschine besprechen wir in Abschnitt 4.4.1.2. Hier ist im Moment nur wichtig, dass wir die Koeffizienten von $P_{n,y}$ durch Gleitkommazahlen austauschen und so das Polynom $P_{n,y}$ auf dem Rechner greifbar machen. Bei bekanntem Polynom $P_{n,y}$ können wir dessen Koeffizienten mit der Rundung fl auf das Raster abbilden und

$$\tilde{c}_{ij} := fl(c_{ij})$$

setzen. Für gewöhnlich ist das Taylorpolynom $P_{n,y}$ einer Funktion $y \in C^{n+1}(\boldsymbol{D})$ aber nicht gegeben und ein Repräsentant $\tilde{P}_{n,y}$ von $P_{n,y}$ auf der Maschine wird mit dem Computer numerisch berechnet, beispielsweise über die Auswertung eines Funktionsausdrucks von y mit Taylor-Modellen. In diesem Fall gilt häufig für viele der Koeffizienten $\tilde{c}_{ij}$ des errechneten Polynoms $\tilde{P}_{n,y}$

$$\tilde{c}_{ij} \neq fl(c_{ij}).$$

Das Polynom $\tilde{P}_{n,y}$ ist im Allgemeinen kein Taylorpolynom der Funktion y, wie es bei $P_{n,y}$ der Fall ist und wie die Indizierung von $\tilde{P}_{n,y}$ suggeriert. Dennoch behalten wir diese Indizierung bei, um deutlich zu machen, dass $\tilde{P}_{n,y}$ das Taylorpolynom $P_{n,y}$ auf der Maschine repräsentiert.

Die Fehlereinschließung $I_{n,y}$ wird auf dem Computer durch ein Maschinenintervall $\tilde{I}_{n,y}$ dargestellt. Allerdings kann nicht einfach $\tilde{I}_{n,y} =\updownarrow I_{n,y}$ gesetzt werden. Vielmehr ist darauf zu achten, dass beim Übergang auf den Computer die Funktionenmenge $[\![P_{n,y}, I_{n,y}]\!]$ von der Funktionenmenge $[\![\tilde{P}_{n,y}, \tilde{I}_{n,y}]\!]$ der zugehörigen Maschinen-Repräsentation eingeschlossen wird. Dies entspricht der Forderung (4.1) an die Intervall-Rundung. Wir gehen im Folgenden näher darauf ein.

Es bezeichne $[\![\tilde{P}_{n,y}, \tilde{I}_{n,y}]\!]$ eine Maschinen-Repräsentation von $[\![P_{n,y}, I_{n,y}]\!]$. Offenkundig handelt es sich dabei um ein reelles Taylor-Modell, das per Definition (vgl. (2.3)) die Funktionenmenge

$$[\![\tilde{P}_{n,y}, \tilde{I}_{n,y}]\!] = \left\{ \tilde{y} \in C(\boldsymbol{D}) \,\middle|\, \tilde{y}(s,t) \in \tilde{P}_{n,y}(s - s_0, t - t_0) + \tilde{I}_{n,y} \;\forall (s,t) \in \boldsymbol{D} \right\}$$

darstellt. Das Maschinenintervall $\tilde{I}_{n,y}$ sollte nun so gewählt sein, dass die Mengeninklusion

$$[\![P_{n,y}, I_{n,y}]\!] \subseteq [\![\tilde{P}_{n,y}, \tilde{I}_{n,y}]\!] \tag{4.3}$$

zustande kommt. Dazu muss jede Funktion $\tilde{y} \in [\![P_{n,y}, I_{n,y}]\!]$ in der rechts stehenden Menge enthalten sein. Es gilt nach (2.5) für alle $(s,t) \in \boldsymbol{D}$

$$\begin{aligned} \tilde{y}(s,t) &\in P_{n,y}(s - s_0, t - t_0) + I_{n,y} \\ &= \tilde{P}_{n,y}(s - s_0, t - t_0) + \big(P_{n,y}(s - s_0, t - t_0) - \tilde{P}_{n,y}(s - s_0, t - t_0) + I_{n,y}\big), \end{aligned}$$

d. h. definieren wir das Maschinenintervall $\tilde{I}_{n,y}$ als

$$\tilde{I}_{n,y} := \updownarrow I_{P_{n,y}-\tilde{P}_{n,y}} +_{fl} \updownarrow I_{n,y}$$

mit einer Einschließung $I_{P_{n,y}-\tilde{P}_{n,y}} \in \mathbb{IR}$ des Approximationsfehlers $P_{n,y} - \tilde{P}_{n,y}$ auf $\boldsymbol{D}$, dann gilt

$$\tilde{y}(s,t) \in \tilde{P}_{n,y}(s - s_0, t - t_0) + \tilde{I}_{n,y} \quad \forall (s,t) \in \boldsymbol{D}$$

und damit die geforderte Inklusion (4.3).

Ein auf dem Computer dargestelltes Taylor-Modell wird im weiteren Verlauf Maschinen-Taylor-Modell heißen und die Menge aller Maschinen-Taylor-Modelle wird mit $\mathcal{TM}_n^{fl}(\boldsymbol{D})$ bezeichnet. Es handelt sich bei $\mathcal{TM}_n^{fl}(\boldsymbol{D})$ um eine Teilmenge von $\mathcal{TM}_n(\boldsymbol{D})$.

Auch bei den Maschinen-Taylor-Modellen hat man das Problem vorliegen, dass das Verknüpfungsergebnis zweier Mengen $[\![P_{n,y_1}, I_{n,y_1}]\!]$, $[\![P_{n,y_2}, I_{n,y_2}]\!] \in \mathcal{TM}_n^{fl}(\boldsymbol{D})$

$$[\![P_{n,y_1}, I_{n,y_1}]\!] \boxdot [\![P_{n,y_2}, I_{n,y_2}]\!], \quad \circ \in \{+, -, \cdot, /\}$$

im Allgemeinen nicht durch ein Maschinen-Taylor-Modell dargestellt werden kann. Das Gleiche gilt auch für die Integration eines Maschinen-Taylor-Modells. Folglich sind die grundlegenden Operationen

$$\left\{\boxplus, \boxminus, \boxdot, \boxint\right\} \tag{4.4}$$

auf dem Computer geeignet zu approximieren. Die Division $\boxslash$ und die Standardfunktionen $\varphi \in \mathcal{SF}$ greifen auf die Grundoperationen (4.4) zurück. Wir führen demzufolge Operationen

$$\left\{\boxplus_{fl}, \boxminus_{fl}, \boxdot_{fl}, \boxint_{fl}\right\}$$

ein, sodass für Mengen $[\![P_{n,y}, I_{n,y}]\!]$, $[\![P_{n,y_1}, I_{n,y_1}]\!]$, $[\![P_{n,y_2}, I_{n,y_2}]\!] \in \mathcal{TM}_n^{fl}(\boldsymbol{D})$ die Inklusion ($\circ \in \{+, -, \cdot\}$)

$$[\![P_{n,y_1}, I_{n,y_1}]\!] \boxdot_{fl} [\![P_{n,y_2}, I_{n,y_2}]\!] \supseteq [\![P_{n,y_1}, I_{n,y_1}]\!] \boxdot [\![P_{n,y_2}, I_{n,y_2}]\!],$$

$$\boxint_{fl} [\![P_{n,y}, I_{n,y}]\!] \begin{Bmatrix} ds \\ dt \end{Bmatrix} \supseteq \boxint [\![P_{n,y}, I_{n,y}]\!] \begin{Bmatrix} ds \\ dt \end{Bmatrix}$$

erfüllt wird. Die Realisierung dieser Operationen auf dem Computer orientiert sich an den Definitionen der korrespondierenden Operationen (4.4) auf $\mathcal{TM}_n(\boldsymbol{D})$. So werden die in den Abschnitten 2.2.1 und 2.2.3 aufgeführten Berechnungsvorschriften mit Hilfe der Intervallarithmetik derart auf dem Computer implementiert, dass auftretende Approximations- und Rundungsfehler in der resultierenden Fehlereinschließung erfasst werden. Wir veranschaulichen dies im Folgenden am Beispiel

der Addition, geben aber eine endgültige Definition dieser Verknüpfung, wie auch der anderen Operationen, aufgrund der engen Verzahnung mit der Datenstruktur erst in den Abschnitten 4.4.2 und 4.4.2.4. Dort wird die konkrete Umsetzung aller genannten Operationen auf dem Computer im Detail erläutert.

Im Falle der Addition ist nach (2.15)

$$[\![P_{n,y_1}, I_{n,y_1}]\!] \boxplus [\![P_{n,y_2}, I_{n,y_2}]\!] = [\![P_{n,y_1+y_2}, I_{n,y_1+y_2}]\!]$$

mit

$$P_{n,y_1+y_2} = P_{n,y_1} + P_{n,y_2}, \quad I_{n,y_1+y_2} = I_{n,y_1} + I_{n,y_2}$$

wobei P_{n,y_1+y_2} im Allgemeinen kein Maschinen-Polynom ist, also kein Polynom, dessen Koeffizienten ausschließlich Gleitkommazahlen sind, und I_{n,y_1+y_2} ebenso im Allgemeinen kein Maschinenintervall darstellt. Es gilt mit $\tilde{s}_0 := fl(s_0)$ und $\tilde{t}_0 := fl(t_0)$ für $(s,t) \in \boldsymbol{D}$ beliebig

$$\begin{aligned} P_{n,y_1}(s-\tilde{s}_0, t-\tilde{t}_0) + P_{n,y_2}(s-\tilde{s}_0, t-\tilde{t}_0) & \\ = \sum_{i+j\leqslant n} c_{ij}^{(y_1)}(s-\tilde{s}_0)^i(t-\tilde{t}_0)^j + \sum_{i+j\leqslant n} c_{ij}^{(y_2)}(s-\tilde{s}_0)^i(t-\tilde{t}_0)^j & \\ \in \sum_{i+j\leqslant n} \underbrace{\left(\left[c_{ij}^{(y_1)}\right] +_{fl} \left[c_{ij}^{(y_2)}\right]\right)}_{=:I_{ij}^{(y_1+y_2)}} (s-\tilde{s}_0)^i(t-\tilde{t}_0)^j. & \end{aligned}$$

Hieraus definieren wir nun ein Maschinen-Polynom

$$\tilde{P}_{n,y_1+y_2}(s-\tilde{s}_0, t-\tilde{t}_0) := \sum_{i+j\leqslant n} \tilde{c}_{ij}^{(y_1+y_2)}(s-\tilde{s}_0)^i(t-\tilde{t}_0)^j$$

mit Koeffizienten $\tilde{c}_{ij}^{(y_1+y_2)} := fl(\mathrm{m}(I_{ij}^{(y_1+y_2)})) \in S$. Für den Approximationsfehler $P_{n,y_1+y_2} - \tilde{P}_{n,y_1+y_2}$ gilt auf $\boldsymbol{D}$

$$\begin{aligned} P_{n,y_1+y_2}(s-\tilde{s}_0, t-\tilde{t}_0) - \tilde{P}_{n,y_1+y_2}(s-\tilde{s}_0, t-\tilde{t}_0) & \\ = \sum_{i+j\leqslant n} \left(c_{ij}^{(y_1+y_2)} - \tilde{c}_{ij}^{(y_1+y_2)}\right)(s-\tilde{s}_0)^i(t-\tilde{t}_0)^j & \\ \in \sum_{i+j\leqslant n} \left(I_{ij}^{(y_1+y_2)} -_{fl} \tilde{c}_{ij}^{(y_1+y_2)}\right)(s-\tilde{s}_0)^i(t-\tilde{t}_0)^j, & \end{aligned}$$

d. h. mit einem Maschinenintervall $I \in \mathbb{IS}$ mit

$$\sum_{i+j\leqslant n} \left(I_{ij}^{(y_1+y_2)} -_{fl} \tilde{c}_{ij}^{(y_1+y_2)}\right)(s-\tilde{s}_0)^i(t-\tilde{t}_0)^j \in I \quad \forall (s,t) \in \boldsymbol{D}$$

erhält man durch

$$\tilde{I}_{n,y_1+y_2} := I +_{fl} I_{n,y_1} +_{fl} I_{n,y_2}$$

ein Maschinen-Taylor-Modell $[\![\tilde{P}_{n,y_1+y_2}, \tilde{I}_{n,y_1+y_2}]\!]$ mit

$$[\![P_{n,y_1+y_2}, I_{n,y_1+y_2}]\!] \subseteq [\![\tilde{P}_{n,y_1+y_2}, \tilde{I}_{n,y_1+y_2}]\!].$$

Eine mögliche Definition der Addition $\boxplus_{fl}$ für Maschinen-Taylor-Modelle kann demnach über das Maschinen-Taylor-Modell $[\![\tilde{P}_{n,y_1+y_2}, \tilde{I}_{n,y_1+y_2}]\!]$ erfolgen. Näheres zur Definition dieser als auch der anderen Operationen folgt wie oben schon erwähnt in den Abschnitten 4.4.2 und 4.4.2.4.

4.2 Objektorientierte Softwareentwicklung

Die Entwicklung objektorientierter Sprachen und Konzepte begann in den 60er Jahren mit der Sprache Simula67. Man versteht heute unter objektorientierter Softwareentwicklung die Verwendung objektorientierter Konzepte in den drei Entwicklungsphasen Analyse, Entwurf und Implementierung. Der Vorteil dabei ist, dass der Entwicklungsprozess konsistent bleibt, was wiederum für eine bessere Nachvollziehbarkeit sorgt. Des weiteren wird ein Strukturbruch beim Übergang von einer Phase in die Nächste verhindert, wodurch Änderungen beispielsweise im Entwurf leicht in die Implementierung übertragen werden können und umgekehrt [2, S. 2-4].

Innerhalb der objektorientierten Konzepte unterscheidet man zwischen Grundkonzepten und Erweiterungen aus dem Bereich der semantischen Datenmodellierung. Zu den Grundkonzepten zählen beispielsweise Klasse, Objekt, Vererbung. Diese stammen aus objektorientierten Programmiersprachen und sind in allen Phasen des Entwicklungsprozesses vorhanden. Unter die Erweiterungen fallen beispielsweise Konzepte wie Geschäftsprozess, Szenario und Zustandsautomat. Man verwendet sie, um das dynamische Verhalten eines Systems zu modellieren [2, S. 5].

In der Analysephase werden die Anforderungen an das zu erstellende Softwaresystem ermittelt und durch objektorientierte Konzepte und Notationen beschrieben. Dazu werden statische und dynamische Modelle verwendet. Das statische Modell ist aus den Grundkonzepten und deren Verbindungen untereinander aufgebaut. Mit dem dynamischen Modell wird die Kommunikation und das Verhalten von Objekten untereinander beschrieben. Insgesamt zeigt das resultierende Analysemodell die wesentliche Struktur des Systems, aber nicht dessen technische Realisierung. Dies übernimmt das Entwurfsmodell, welches auf der Grundlage des Analysemodells erstellt wird. In der Entwurfsphase werden Schnittstellen, Funktions- und Leistungsumfang der Komponenten festgelegt und in einem Entwurfsmodell

beschrieben. Das Entwurfsmodell ist damit ein Abbild des späteren Programms [2, S. 14-15].

Im Bereich der Implementierung hat sich mit Beginn der 90er Jahre die Sprache C++ etabliert. Seit 1996 nimmt Java neben C++ eine nicht minder wichtige Stellung ein [2, S. 3]. Beide Programmiersprachen unterstützen die Grundkonzepte der Objektorientierung vollständig und zählen somit zu den objektorientierten Sprachen. Programmiersprachen, welche die Grundkonzepte nur teilweise unterstützen, werden dagegen objektbasiert genannt.

Zur Beschreibung eines objektorientierten Modells hat sich als Standard-Notation die *Unified Modeling Language*, kurz UML, durchgesetzt [2, S. 6]. Diese wird auch in dieser Arbeit verwendet werden. Es folgt nun eine Beschreibung der Grundkonzepte der Objektorientierung und deren Darstellung in der UML.

Objekt

Ein Objekt in der objektorientierten Softwareentwicklung ist ein auf die relevanten Eigenschaften reduziertes, reales Objekt [2, S. 21]. Das reale Objekt kann ein Gegenstand, eine Person oder ein Begriff sein [2, S. 18]. Dabei bedeutet „relevant“ auf das zu lösende Problem bzw. das zu realisierende Softwaresystem bezogen.

Ein Objekt besitzt einen bestimmten Zustand und reagiert mit einem definierten Verhalten. Der Zustand des Objekts ist gegeben durch seine Attribute bzw. deren aktuelle Werte sowie der momentan bestehenden Verbindungen zu anderen Objekten. Das Verhalten des Objekts wird durch eine Menge von Operationen beschrieben. Neben dem Zustand und dem Verhalten besitzt ein Objekt auch noch eine Identität, die es von anderen Objekten unterscheidbar macht [2, S. 18].

Das Objektdiagramm dient der Darstellung von Objekten und deren Verbindungen untereinander zu einem bestimmten Zeitpunkt. Es ist eine Momentaufnahme des Systems und zählt somit zu den statischen Konzepten der Softwareentwicklung [2, S. 19].

Klasse

Eine Klasse ist der Konstruktionsplan für eine Sammlung von Objekten. Sie definiert die Struktur (Attribute) und das Verhalten (Operationen) der Objekte, sowie deren Beziehungen zu anderen Objekten. Zudem ist sie für die Erzeugung der Objekte zuständig. Jedes Objekt gehört demnach zu genau einer Klasse. Mehrere Klassen können untereinander durch Assoziationen und Vererbungsstrukturen in Verbindung stehen [2, S. 21].

In der UML wird ein Objekt durch ein, eventuell in zwei Felder aufgeteiltes, Rechteck dargestellt (Abb. 4.1). Im oberen Feld steht bei einem anonymen Objekt nur der Klassenname mit einem Doppelpunkt vorangestellt. Soll das Objekt mit seinem Namen angesprochen werden, so kann zusätzlich noch der Objektname vor dem Doppelpunkt aufgelistet werden. Kann der Name der Klasse aus dem

Abbildung 4.1: Darstellung von Objekten in der UML.

Kontext erschlossen werden, so ist die Angabe des Objektnamens ausreichend. Im unteren Feld werden, sofern dies wichtig erscheint, die für den betrachteten Sachverhalt wichtigen Attribute eingetragen. Dabei gibt es die Möglichkeit, neben dem Attributnamen auch dessen Typ und Wert anzugeben [2, S. 18-19].

Für die Darstellung von Klassen gibt es in der UML mehrere Möglichkeiten, wie die Abbildung 4.2 zeigt. Die Art der Darstellung hängt ab von der Relevanz der

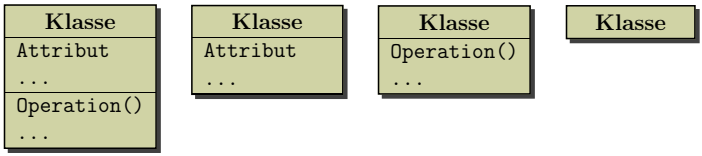

Abbildung 4.2: Darstellung von Klassen in der UML.

Details im zugrunde liegenden Kontext [2, S. 21-22].

Ein Klassendiagramm beschreibt Klassen und deren Beziehungen (Assoziationen, Vererbungsstrukturen) untereinander. Es gehört wie das Objektdiagramm auch zu den statischen Konzepten. Bei großen Systemen verwendet man zur Beschreibung oft mehrere Klassendiagramme [2, S. 22].

Eine besondere Bedeutung für die Vererbung hat das Konzept der abstrakten Klasse. Abstrakte Klassen können keine Objekte erzeugen und werden in der UML mit dem Merkmal `{abstrakt}` gekennzeichnet oder durch einen kursiven Klassennamen kenntlich gemacht [2, S. 23].

Ein Konzept des objektorientierten Entwurfs, welches später zur Beschreibung verwendet wird, ist die generische Klasse. Es handelt sich dabei um eine Klasse mit einem oder mehreren Parametern. In der UML wird eine generische Klasse wie in Abbildung 4.3 dargestellt. In der Parameterliste stehen die Parameter durch Kommata getrennt und werden mit Name und Typ spezifiziert [2, S. 228-229].

Attribut

Attribute beschreiben die Daten, die von Objekten einer Klasse angenommen werden können. Sie werden durch Name und Typ spezifiziert, wobei optional auch

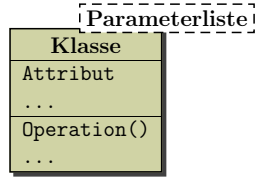

Abbildung 4.3: Darstellung von generischen Klassen in der UML.

noch ein Anfangswert sowie eine Liste von Merkmalen angegeben werden kann. Die vollständige Syntax lautet im Entwurf [2, S. 235]

```
Sichtbarkeit Attribut : Typ {Merkmalsliste}.
```

Der Name eines Attributs muss innerhalb der Klasse eindeutig sein. Ein Attribut heißt Klassenattribut, wenn dessen Wert für alle Objekte der Klasse gleich ist. Ein solches wird in der UML durch Unterstreichen kenntlich gemacht [2, S. 25-30].

Operation

Eine Operation ist eine ausführbare Tätigkeit. Operationen ermöglichen den Zugriff auf die Attribute einer Klasse. Sie stehen allen Objekten der Klasse zur Verfügung und beschreiben deren Verhalten. Die Operationen einer Klasse zusammengenommen bilden die Schnittstelle der Klasse. Man unterscheidet zwischen Objektoperationen, Konstruktoroperationen und Klassenoperationen. Die Erstgenannten werden auf einzelne Objekte angewandt. Konstruktoroperationen sind für die Erzeugung neuer Objekte und der damit einhergehenden Initialisierung und Datenerfassung zuständig. Klassenoperationen sind der jeweiligen Klasse zugeordnet und im Gegensatz zu Objektoperationen nicht auf einzelne Objekte anwendbar. In der UML werden Klassenoperationen analog zu Klassenattributen durch Unterstreichen kenntlich gemacht [2, S. 30-34].

Im Entwurf ist für jede Operation die komplette Signatur anzugeben. Die vollständige Syntax lautet

```
Sichtbarkeit Operation (Parameterliste) : Ergebnistyp {Merkmalsliste}.
```

Für die Parameter gilt die Syntax

```
[in|out|inout] Name : Typ = Anfangswert.
```

Um kompakte Klassendiagramme zu erhalten, können Angaben zur Parameterliste und zum Ergebnistyp auch separat gemacht werden [2, S. 238-240].

Sichtbarkeit bei Attributen und Operationen

Unter der Sichtbarkeit versteht man die Regelung des Zugriffs anderer Klassen auf Attribute und Operationen einer Klasse. Die Bezeichnung `public` steht für freien Zugriff, `protected` für Zugriff nur innerhalb der Klasse und aus deren Unterklassen und `private` für Zugriff nur innerhalb der Klasse. Die Kennzeichnung der Sichtbarkeit zeigt die Abbildung 4.4 [2, S. 235, S. 238].

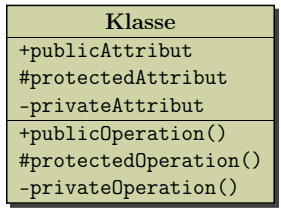

Abbildung 4.4: Kennzeichnung der Sichtbarkeit in der UML.

Assoziation

Assoziationen beschreiben Beziehungen zwischen Objekten verschiedener oder auch derselben Klasse. In der UML wird eine Assoziation durch eine Linie zwischen einer oder zwei Klassen kenntlich gemacht. An den Linienenden steht jeweils die Kardinalität der Assoziation, die besagt, wie viele Objekte ein bestimmtes Objekt kennen kann (Abb. 4.5) [2, S. 40-41]. Die Kardinalitäten können dabei einzelne

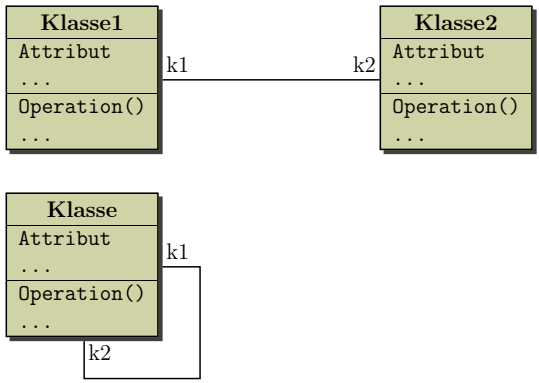

Abbildung 4.5: Darstellung von Assoziationen in der UML.

Zahlen oder auch Zahlenbereiche sein. Eine einzelne Zahl sagt aus, dass genau die angegebene Anzahl an Beziehungen besteht, während bei Angabe eines Bereiches die Zahl der Verbindungen innerhalb des Bereichs variieren kann. Beginnt die Bereichsangabe mit der Zahl 0, so kann eine Verbindung bestehen, muss aber nicht. Die Angabe „$\star$“ allein oder auch innerhalb einer Bereichsangabe steht für beliebig viele Verbindungen [2, S. 41].

Neben der einfachen Assoziation gibt es noch die Aggregation und die Komposition. Die Aggregation modelliert eine „ist Teil von“ bzw. „besteht aus“ Beziehung. Die Komposition stellt eine starke Form der Aggregation dar. Hier kann jedes Objekt der Teilklasse nur Teil eines einzigen Objekts der Aggregatklasse sein, und Aktionen wie beispielsweise kopieren oder löschen des Aggregatobjekts haben die entsprechende Aktion für die Teilobjekte zur Folge. Deren Darstellung in der UML zeigt die Abbildung 4.6 [2, S. 46-49].

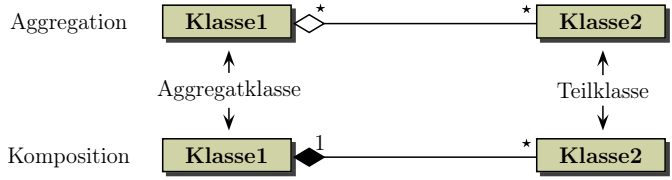

Abbildung 4.6: Darstellung von Aggregation und Komposition in der UML.

In der Entwurfsphase ist festzulegen, ob eine Assoziation nun uni- oder bidirektional implementiert wird. Somit wird innerhalb eines Klassendiagramms im Entwurf zusätzlich die Richtung, in der die Assoziation realisiert wird, mit einer Pfeilspitze kenntlich gemacht [2, S. 244-248].

Vererbung

Das Konzept der Vererbung wird verwendet, um eine Beziehung zwischen einer allgemeinen Klasse (Basisklasse) und einer spezialisierten Klasse zu beschreiben. Die spezialisierte Klasse ist dabei vollständig konsistent mit der Basisklasse, besitzt aber zusätzliche Attribute, Operationen und Assoziationen. In der UML wird diese Klassenhierarchie wie in Abbildung 4.7 gezeigt dargestellt [2, S. 51-55].

4.3 Aufbau des Programms RiOT

Das Programm RIOT, dessen Bezeichnung für „Rigorous integration of ODEs with Taylor models“ steht, setzt sich aus dem Paket TM-Base und dem Paket DGL zusammen (Abb. 4.8). Ersteres beinhaltet die Realisierung der Grundarithmetik und

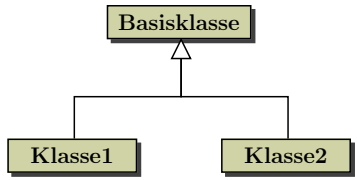

Abbildung 4.7: Darstellung von Vererbung in der UML.

benötigt eine externe Intervallbibliothek, das Paket DGL umfasst die Implementierung aller für die Lösung von Differentialgleichungen relevanten Teile und baut auf TM-Base auf.

Für die Intervallbibliothek wurde auf die gängigen Bibliotheken C-XSC [31, 61] und filib++ [39, 40, 61] zurückgegriffen. Während C-XSC eine sehr umfangreiche Bibliothek ist, die hochgenaue Einschließungen liefert, stellt filib++ nur das Nötigste zur Verfügung und liefert etwas großzügigere Einschließungen, sorgt dafür aber für wesentlich kürzere Rechenzeiten.

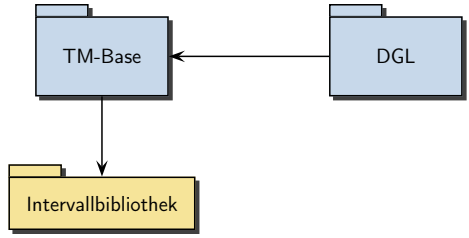

Abbildung 4.8: Aufbau der Implementierung.

4.4 Das Basispaket

Der Grundgedanke bei der softwaremäßigen Realisierung der Maschinen-Taylor-Modelle war, ihre Darstellung, d. h. die Programmierung der Datenstruktur mitsamt der dazugehörigen Hilfsstrukturen und Mechanismen, auf dem Rechner austauschbar zu machen, also von der Schnittstelle zu trennen. Dazu wurde das Objekt „Maschinen-Taylor-Modell“ in zwei Objekte aufgeteilt. Ein Objekt stellt die

Schnittstelle bereit, das andere Objekt verwaltet die Zustände. Zwei Klassen wurden demnach benötigt, eine Handle-Klasse [59, S. 841-845] für die Schnittstelle und eine Klasse für die Darstellung der Maschinen-Taylor-Modelle (Abb. 4.9). Diese Aufteilung ermöglicht eine Veränderung oder auch einen kompletten Austausch der Darstellung ohne Auswirkung auf die Schnittstelle. Hinzu kommt, dass ein Handle aufgrund der geringen Größe günstig verschoben und kopiert werden kann [59, S. 842], wovon wir im weiteren Verlauf noch Gebrauch machen werden.

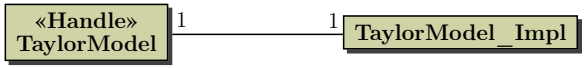

Abbildung 4.9: Klassen für Schnittstelle und Darstellung.

4.4.1 Datenstruktur der Maschinen-Taylor-Modelle

Ein Maschinen-Taylor-Modell besteht aus einem Polynom in mehreren Veränderlichen und einem Maschinenintervall für die Fehlereinschließung. Bei dem Polynom handelt es sich um ein Taylorpolynom, sodass neben den Polynomkoeffizienten und dem Polynomgrad auch die Entwicklungsstellen und der Definitionsbereich jeder Variablen als Information zu speichern ist.

Die Ordnung des Taylorpolynoms legen wir als Klassenattribut fest, wodurch sie für alle Maschinen-Taylor-Modelle gleich ist. Wird sie verändert, so wirkt sich dies auf alle bestehenden Maschinen-Taylor-Modelle aus.

4.4.1.1 Fehlereinschließung

Die Fehlereinschließung wird durch ein Maschinenintervall dargestellt. Der Datentyp für das Intervall wird in der externen Intervallbibliothek festgelegt und ist daher von dieser abhängig. Da C-XSC und filib++ den Intervall-Datentyp unterschiedlich definieren, müsste überall dort, wo ein Intervall verwendet wird, eine Fallunterscheidung nach der verwendeten Bibliothek vorgenommen werden, was den Code unnötig aufbläht und unübersichtlich macht.

Dieses Problem der Fallunterscheidung haben wir durch die Verwendung einer Adapterklasse aus dem eigentlichen Programmcode ausgelagert und an zentraler Stelle gekapselt. Hierdurch wird neben der Übersichtlichkeit des Programms auch noch die Wartbarkeit der Software verbessert. Außerdem hat man mit der Adapterklasse ein Modul, das auch in anderen Projekten wiederverwendet werden kann.

Damit also die Software auch mit anderen Intervallbibliotheken funktioniert, wurde eine Klasse `Interval` in Anlehnung an das Strukturmuster[1] Adapter [18,

[1] Ein Muster beschreibt die Lösung eines beständig wiederkehrenden Problems in abstrahierter Form [18, S. 3-5].

S. 171] konstruiert. Ihr Zweck ist, für die Verwendung von Intervallen innerhalb des TM-Base Pakets, aber auch innerhalb des DGL Pakets eine einheitliche Schnittstelle anzubieten. Zur Realisierung dieser Funktionalität wurde die Methode der Objektkomposition [18, S. 26] herangezogen. Die Klasse `Interval` enthält ein Exemplar der verwendeten Intervallklasse und leitet die Intervalloperationen an das Exemplar weiter (Abb. 4.10).

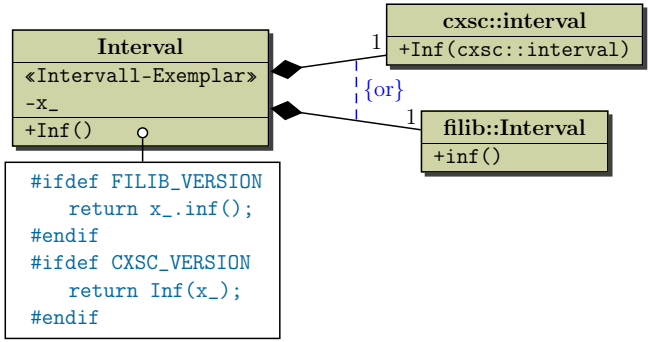

Abbildung 4.10: Delegation der Operationen am Beispiel der Operation `Inf()`.

4.4.1.2 Taylorpolynom

Das Taylorpolynom ist von der Form

$$P(x_1 - x_{1_0}, \ldots, x_\nu - x_{\nu_0}) = \sum_{|\boldsymbol{\iota}|=0}^{n} c_{\boldsymbol{\iota}} \, (\boldsymbol{x} - \boldsymbol{x}_0)^{\boldsymbol{\iota}}$$

mit

$$\begin{gathered} \boldsymbol{\iota} = (i_1, \ldots, i_\nu) \in \mathbb{N}^\nu, \quad |\boldsymbol{\iota}| = i_1 + \ldots + i_\nu, \\ \boldsymbol{x} = (x_1, \ldots, x_\nu) \in \mathbb{R}^\nu, \quad \boldsymbol{x}_0 = (x_{1_0}, \ldots, x_{\nu_0}) \in \mathbb{R}^\nu, \\ (\boldsymbol{x} - \boldsymbol{x}_0)^{\boldsymbol{\iota}} = (x_1 - x_{1_0})^{i_1} \cdots (x_\nu - x_{\nu_0})^{i_\nu}. \end{gathered}$$

Die $c_{\boldsymbol{\iota}}$ bezeichnen die Taylorkoeffizienten, die $(\boldsymbol{x}-\boldsymbol{x}_0)^{\boldsymbol{\iota}}$ die Monome und $\boldsymbol{x}_0$ bezeichnet die Entwicklungsstelle des Taylorpolynoms. Gespeichert werden die Koeffizienten und die Exponenten der Monome des Taylorpolynoms. Zur Entwicklungsstelle kommen wir in Abschnitt 4.4.1.3.

Die Anzahl der Koeffizienten hängt ab von der Anzahl ν der Veränderlichen und dem Polynomgrad n. Sie errechnet sich aus [44]

$$\left(\begin{array}{c} n+\nu \\ \nu \end{array} \right) = \frac{(n+\nu)!}{n!\,\nu!}.$$

In Tabelle 4.1 sind beispielhaft einige Werte aufgeführt. Man sieht deutlich, wie rasch die Anzahl der Koeffizienten im Fall von mehr als drei Veränderlichen mit der Polynomordnung zunimmt.

Im Allgemeinen ist in vielen Anwendungen zu beobachten, dass ein Großteil der Koeffizienten verschwindet, man aber auf dem Rechner das Problem hat, dass aufgrund von Rundungsfehlern der Wert Null nur selten exakt angenommen wird. Die Einführung eines Schwellenwertes, wie dies schon Berz und seine Mitarbeiter in deren Implementierung vorgemacht haben, kann hier Abhilfe schaffen. Dazu definieren wir in der Klasse `TaylorModel_Impl` ein Klassenattribut `sparsity_tol_`, setzen jeden Term $c_\iota(\boldsymbol{x} - \boldsymbol{x}_0)^\iota$ mit

$$|c_\iota| < \texttt{sparsity\_tol\_}$$

Null und modifizieren den Intervallterm so, dass die geforderte Inklusionseigenschaft erhalten bleibt (mehr dazu in Abschnitt 4.4.2). Dies sichert einerseits die Dünnbesetztheit der Polynome auch auf der Maschine, andererseits kann durch Verändern des Schwellenwertes die Anzahl der Terme im Polynom gesteuert werden [45, S. 394-395]. Die vorliegende, teilweise künstlich erzeugte dünnbesetzte Struktur der Polynome vermindert den Speicherbedarf für die Koeffizienten sowie

	Anzahl Variablen			
Ordnung	1	3	5	7
1	2	4	6	8
2	3	10	21	36
3	4	20	56	120
4	5	35	126	330
5	6	56	252	792
6	7	84	462	1716
7	8	120	792	3432
8	9	165	1287	6435
9	10	220	2002	11440
10	11	286	3003	19448

Tabelle 4.1: Anzahl der Koeffizienten eines Polynoms in ν Variablen vom Grad n.

den Zeitaufwand bei umfangreichen Polynomoperationen. Es ist klar, dass nur die nicht verschwindenden Koeffizienten gespeichert werden.

Innerhalb eines interdisziplinären Projekts an der Technischen Universität München [32] wurde der Frage nach der effizienten Speicherung multivariater, dünnbesetzter Polynome nachgegangen und mit verschiedenen Datenstrukturen Vergleiche durchgeführt. Als Datenstrukturen wurde die verkette Liste, der ausgeglichene Baum und die Hash-Tabelle verwendet. Verglichen wurde der Aufwand an Operationen in jeweils für die Datenstruktur angepassten Algorithmen, wie beispielsweise die Grundrechenarten $\{+, -, \cdot\}$ oder die Potenzierung für Polynome.

Von den Grundrechenarten kommt der Multiplikation wegen des hohen Aufwands und der wichtigen Rolle im Kontext der Maschinen-Taylor-Modelle besondere Bedeutung zu. Letzteres begründet sich aus der Definition der Standardfunktionen für Taylor-Modelle (siehe Kapitel 2, Abschnitt 2.2.2), die dann auch für Maschinen-Taylor-Modelle herangezogen wird. Die Berechnung eines Taylor-Modells für eine Standardfunktion $\varphi \in \mathcal{SF}$ basiert auf deren Taylorentwicklung und damit insbesondere auf der Multiplikation. Der Aufwand für die Berechnung eines Taylor-Modells der Ordnung n entspricht in diesem Fall n Multiplikationen und n Additionen von Taylor-Modellen. Somit hängt die Effizienz der Taylor-Modell-Arithmetik im wesentlichen von der Effizienz der Multiplikation ab.

In einem Vergleich der Datenstrukturen bezüglich des zeitlichen Aufwands für die Multiplikation zweier Polynome hat die Hash-Tabellen-Implementierung am Besten abgeschnitten [32, S. 71-75]. Selbst speziell für die verkettete Liste optimierte Multiplikationsalgorithmen konnten daran nichts ändern. Wir haben deshalb das Taylorpolynom in unserem Programm durch eine Hash-Tabelle dargestellt und uns dabei an den Ausführungen in [32, S. 11-20, S. 39-50] orientiert.

Eine Hash-Tabelle besteht aus einem Array mit verketteten Listen als Einträgen. Sie bietet sowohl eine effiziente Speicherung von Daten, als auch einen schnellen Zugriff darauf. Insbesondere können Einfüge-, Lese- und Löschoperationen schnell durchgeführt werden. Die Effizienz der Hash-Tabelle hängt dabei von der Größe des Arrays und von den Längen der einzelnen Listen ab. Damit die Tabelle nicht zu einer einzigen Liste degeneriert, ist die Verteilung der Daten über die zur Verfügung stehenden Arrayfelder entscheidend. Diese Aufgabe übernimmt die sogenannte Hash-Funktion, die jedem einzufügenden Element die passende Liste zuweist. Eine gute Hash-Funktion zeichnet sich demnach durch eine möglichst gleichmäßige Verteilung der Daten über die Arrayfelder aus. Solch eine Hash-Funktion zu bestimmen ist in der Regel alles andere als trivial. In ihre Konstruktion gehen neben der Arraygröße und der Gestalt der Schlüssel auch Besonderheiten über die zu speichernden Daten ein [32, S. 39-42].

In unserem Programm wurde die Hash-Tabelle als generische Klasse implementiert, wie auch alle dazugehörigen Bausteine und Hilfsstrukturen. Dazu zählt die

Struktur `HashTableEntry`, welche aus einem Schlüssel-Wert-Paar `HashTableData` und einem Zeiger auf ein `HashTableEntry`-Objekt besteht. Die Zeiger sind für die Konstruktion der Listen vorgesehen. Der Zeiger des letzten Listenelements wird auf Null gesetzt und definiert somit das Ende einer Liste. Die Hash-Tabelle selbst wird realisiert durch einen Array mit Zeigern auf `HashTableEntry`-Objekte, die den Anfang jeder Liste darstellen. Enthält ein Arrayfeld keine Liste, so wird der Zeiger auf Null gesetzt.

Die Klassen `HashTableIterator` und `HashTableConstIterator` als weitere Bausteine der Implementierung stellen zeigerähnliche Konstrukte dar, welche ein sequentielles Durchlaufen der Hash-Tabelle ermöglichen. Dieser Mechanismus ist nützlich, wenn man auf allen Einträgen operieren will, beispielsweise bei der Ausgabe der Hash-Tabelle oder später bei den arithmetischen Grundoperationen der Polynome.

Einen weiteren Hilfsmechanismus stellt die Basisklasse `BasicAction` zur Verfügung. Sie definiert Schnittstellen für Operationen, die beim Einfügen eines eventuell schon vorhandenen Elements entscheiden, wie verfahren werden soll. Diese Operationen stellen ein flexibles Werkzeug zur Manipulation der Daten dar. Die Klasse wurde nach dem sogenannten „Barton & Nackman Trick“ [3][60, S. 9] realisiert. Er bietet eine Möglichkeit für statischen Polymorphismus [59, S. 330, S. 370] und ermöglicht den Verzicht auf virtuelle Funktionen [59, S. 328], welche bei einer geringen Menge an auszuführendem Code und häufigen Aufrufen zu schlechteren Laufzeiten führen [60, S. 6-7].

Abbildung 4.11 zeigt die eben vorgestellten Elemente der Hash-Tabellen-Implementierung und deren Beziehungen untereinander.

Ein Polynom kann aufgefasst werden als Menge von Termen $c_{\boldsymbol{\iota}}(\boldsymbol{x}-\boldsymbol{x}_0)^{\boldsymbol{\iota}}$. Zur Speicherung der Terme in der Hash-Tabelle muss zunächst das Konstrukt des Monoms auf den Rechner abgebildet werden. Für die Realisierung eines entsprechenden Datentyps auf der Maschine ist die Kenntnis der im Monom vertretenen Variablen und deren Exponenten wichtig. Wir legen eine globale Symboltabelle `Variables` an, in der jedem Variablensymbol – das ist eine vom Benutzer festgelegte Zeichenkette – eine Zahl größer 1 bijektiv zugeordnet wird. Die Zuordnung erfolgt bei 1 beginnend in der Reihenfolge der Eintragungen. So erhält das erste eingetragene Symbol den Wert 1, das zweite den Wert 2, das dritte den Wert 3 und so weiter. Diese Werte wollen wir im Folgenden auch als Variablencode bezeichnen. Wir erhalten dadurch in der Menge der Variablenbezeichner eine Ordnung, die zur Darstellung der Monome herangezogen wird. Wir orientieren uns dabei an der Implementierung in [32, S. 11-15]. Die Klasse `Monom` (Abb. 4.12) speichert nun im Attribut `left_` den Code der kleinsten, im Attribut `right_` den Code der größten in einem Monom auftretenden Variablen. Die Exponenten aller im Bereich von `left_` bis `right_` liegenden Variablen werden in einem Feld abgelegt, auf welches

über die Variablencodes zugegriffen wird. Bei Variablen, die nicht im Monom vorhanden sind, aber durch die Angaben in `left_` und `right_` erfasst werden, wird logischerweise der Exponent 0 eingetragen.

Wie oben schon erwähnt, kommt der Multiplikation eine entscheidende Rolle in Bezug auf die Effizienz der Arithmetik zu. Werden zwei Polynome miteinander multipliziert, von denen das eine aus N_1 Termen, das andere aus N_2 Termen besteht, so werden insgesamt $N_1 \cdot N_2$ Terme miteinander multipliziert und anschließend gleiche Terme zusammengefasst. Die Multiplikation für Monome wird also $N_1 \cdot N_2$ mal aufgerufen. Für die Addition gleichartiger Terme werden mindestens genauso viele Monome miteinander verglichen.

In der derzeitigen Implementierung besteht das Exponentenfeld aus zwei ganzen Zahlen, die im Speicher insgesamt 8 Byte einnehmen. In jedem Byte wird der Exponent einer Variablen abgelegt. Die Darstellung verliert damit zwar an Flexibilität, ermöglicht dafür aber eine schnelle Realisierung der Multiplikation und des Vergleichs zweier Monome. So sind bei der Multiplikation nur zwei ganze Zahlen zu addieren und im Monomvergleich die entsprechenden Zahlen miteinander zu vergleichen. Die Flexibilitätseinbußen äußern sich in der Beschränkung auf Va-

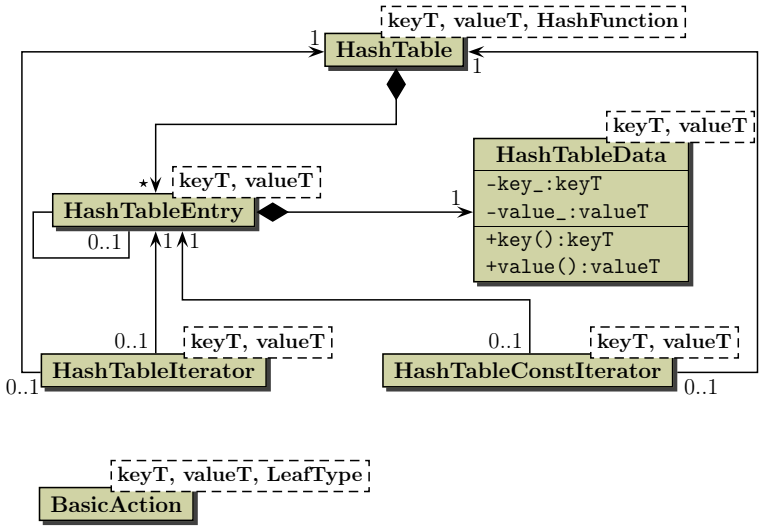

Abbildung 4.11: Die Elemente der Hash-Tabellen-Implementierung.

riablencodes bis maximal 8 sowie einer Einschränkung der Polynomordnung auf maximal[2] 127. Die Information in `left_` und `right_` wird verwendet, um in anderen, nicht so stark frequentierten Operationen Zugriffe auf nicht benutzte Bytes zu vermeiden.

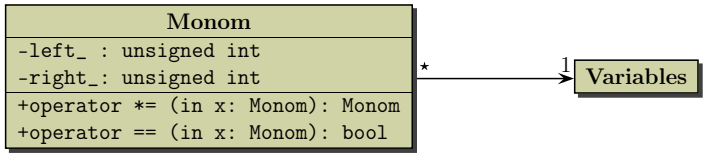

Abbildung 4.12: Die Klasse `Monom`.

Wenden wir uns nun der Hash-Funktion zu. Die Hash-Funktion weist einem Term $c_{\boldsymbol{\iota}}(\boldsymbol{x} - \boldsymbol{x}_0)^{\boldsymbol{\iota}}$ seine Liste zu, in die er eingefügt werden soll. Wir verwenden hierzu das Monom $(\boldsymbol{x} - \boldsymbol{x}_0)^{\boldsymbol{\iota}}$ bzw. den Exponenten $\boldsymbol{\iota}$ als Argument. Die Bildmenge der Funktion besteht aus den Indizes der Arrayfelder. Bezeichnet l die Länge des Arrays, so ist die Bildmenge $\{0, \ldots, l-1\}$. Aufgrund der vorliegenden Beschränkungen für den Datentyp `Monom` gestaltet sich die Konstruktion der Hash-Funktion einfach. Wir verwenden die sogenannte Divisionsmethode [32, S. 43], erweitert auf ein Tupel von Zahlen. Diese liefert eine nahezu gleichmäßige Verteilung der Terme auf die Listen. Wir betrachten dazu den Exponenten $\boldsymbol{\iota} = (i_1, \ldots, i_8)$ des Monoms als Darstellung einer Zahl a zur Basis $b = 127$,

$$a = i_1 + i_2 \cdot b + i_3 \cdot b^2 + \ldots + i_8 \cdot b^7.$$

Diese kann mittels des Horner-Schemas in 8 Schritten berechnet werden. Anschließend reduzieren wir die Zahl a modulo l und erhalten den Hash-Wert aus $\{0, \ldots, l-1\}$. Damit alle Einträge des Tupels in die Berechnung eingehen, ist es wichtig, dass l und b teilerfremd sind. Wir wählen demnach die Arraygröße l als Primzahl ungleich b.

Wird die Hash-Tabelle zu voll, so wird der Zugriff auf die Daten ineffizient [59, S. 533]. Die Arraygröße wird deshalb bei einer Auslastung von 70 % entsprechend vergrößert.

Die für die Arraygröße benötigen Primzahlen werden in einer Primzahlentabelle zur Verfügung gestellt. Die Klasse `PrimeNumbers` ist für deren Erzeugung zuständig. Da aber nur eine Primzahlentabelle im System benötigt wird, ist die Klasse

[2] In einem Byte können Werte bis maximal 255 gespeichert werden. Da aber bei der Multiplikation von Monomen die Exponenten addiert werden und das Ergebnis noch darstellbar sein muss, wird die Polynomordnung auf maximal 127 beschränkt.

so gebaut, dass von ihr nur ein Objekt existieren kann. Nach dem Singletonmuster [18, S. 157-166] werden hierzu die Konstruktoroperationen als `private` deklariert, wodurch nur Operationen innerhalb der Klasse Objekte erzeugen können. Für die Erzeugung des einen Objekts wird eine Klassenoperation eingerichtet, welche zudem für die Erreichbarkeit von außen einen definierten Bezugspunkt auf das Objekt bereitstellt. Daneben bietet die Realisierung über eine Klasse den Vorteil einer Schnittstelle, welche für einen definierten Zugriff auf die Daten sorgt. Die Klasse `Variables` ist ebenfalls nach dem Singleton-Prinzip aufgebaut.

4.4.1.3 Restliche Daten

Neben dem Polynom und der Fehlereinschließung sind in `TaylorModel_Impl` auch noch Informationen über die Entwicklungsstellen und der Gültigkeitsbereiche jeder im Polynom vertretenen Variablen zu speichern. Dies ist wichtig, da es in diesem Kontext nur Sinn macht, Polynome zu addieren und zu multiplizieren, deren gemeinsame Variablen die gleichen Entwicklungsstellen und Gültigkeitsbereiche haben. Diese Information wird auch zur Polynomauswertung und zur Integration benötigt.

Für jede Variable sind eine Gleitkommazahl für die Entwicklungsstelle und ein Maschinenintervall für den Gültigkeitsbereich zu speichern. Da es aber durchaus mehrere Maschinen-Taylor-Modelle geben kann, deren Variablen denselben Entwicklungspunkt und denselben Gültigkeitsbereich haben, können diese sich die Information auch teilen. Dazu wird im System eine Klasse `VarData` nach dem Singleton-Prinzip angelegt, welche die Variablendaten in einer Tabelle ablegt. Die Tabelle besteht aus einem Array mit Zeigern auf Objekte der Struktur `VarDataListNode`, welche alle Informationen kapselt. Jeder Variablen wird über den Variablencode ihre Liste im Array zugewiesen, in der dann die verschiedenen `VarDataListNode`-Objekte zu einer Liste verkettet werden.

Die Klasse `TaylorModel_Impl` speichert Zeiger auf die benötigten Listenelemente und greift somit direkt auf die Daten zu. Werden die Daten eines Listenelements von keinem Maschinen-Taylor-Modell mehr referenziert, wird es aus der Liste gelöscht. Hierzu ist ein Protokollieren der Referenzen nötig. Dieser Verwaltungsakt sowie auch das Löschen nicht mehr referenzierter Daten wurde in einer separaten Klasse realisiert.

4.4.1.4 Reference-Counting

Die Datenmengen, die mit jedem Maschinen-Taylor-Modell anfallen, sind in der Regel recht umfangreich. Ein ständiges Erzeugen, Kopieren und Löschen von Objekten, beispielsweise innerhalb der Auswertung von Funktionsausdrücken [15, Chapter 11], kann sich deshalb sehr schnell in einer verminderten Effizienz nie-

derschlagen. Aus diesem Grund wurde die generische Klasse `ReferenceCounter` geschrieben. Sie hat die Aufgabe, die Zugriffe auf ein Objekt zu zählen, sodass dieses nur bei Bedarf kopiert und gelöscht wird. Es gibt also im System mehrere Handles, die ein und dieselbe Darstellung im Speicher verwenden. Erst wenn eines der Handles diese Darstellung verändern möchte, wird eine Kopie davon angelegt [59, S. 842].

Reference-Counting [59, S. 842-844] kann eingesetzt werden, wenn wie hier die Schnittstelle von der Implementierung getrennt ist. Es stellt einen zusätzlichen Verwaltungsaufwand für die Handle-Klasse dar, der sich aber bei großen Objekten bezahlt macht. Damit das Funktionsprinzip des Reference-Counting auch leicht für andere Objekte verwendet werden kann, wurde es in einer eigenen Klasse gekapselt (Abb. 4.13).

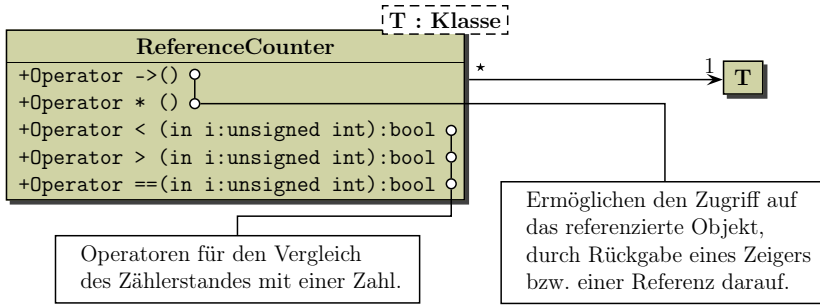

Abbildung 4.13: Die generische Klasse `ReferenceCounter`.

Die Klasse Reference-Counting legt für das referenzierte Objekt einen Zähler an und verwaltet einen Zeiger darauf. Der Zähler ist dem referenzierten Objekt zugeordnet und wird von allen `ReferenceCounter`-Objekten verwendet, die den Zugriff auf das Objekt zählen (Abb. 4.14).

Nach der Anwendung des Reference-Counting auf die Implementierung der Maschinen-Taylor-Modelle enthält die Handle-Klasse ein `ReferenceCounter`-Objekt, welches die Funktionalität für die Verwaltung der Zugriffe auf die Darstellung bereitstellt. Das Klassendiagramm aus Abbildung 4.9 wird um diesen Baustein ergänzt und entsprechend verändert (Abb. 4.15). Als Folge der Aufsplittung des Typs „Maschinen-Taylor-Modell“ in Handle und Darstellung werden nun bei der Auswertung arithmetischer Ausdrücke oder bei der Zuweisung von Maschinen-Taylor-Modellen Handle-Objekte kopiert bzw. zugewiesen, anstelle der in der Regel sehr umfangreichen Darstellung. Das miteingebaute `ReferenceCounter`-Objekt notiert die Anzahl der Handles, die gemeinsam eine Darstellung verwenden und kümmert

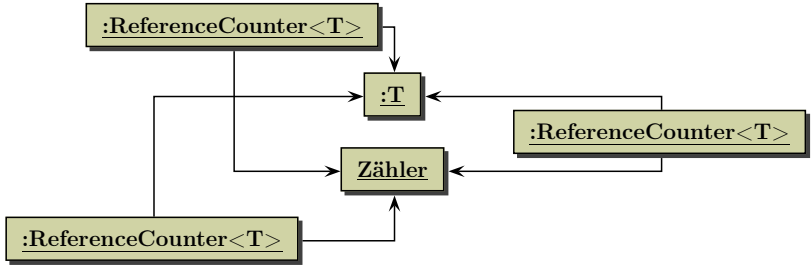

Abbildung 4.14: Ein Beispiel für Objektverbindungen beim Reference-Counting.

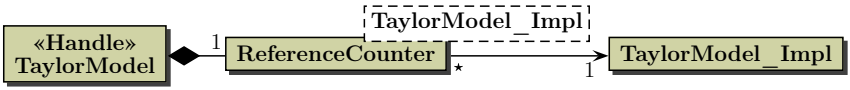

Abbildung 4.15: Verbindung zwischen Schnittstelle und Darstellung.

sich um das Aufräumen des Speichers, falls eine Darstellung nicht mehr benötigt wird.

4.4.1.5 Monomstruktur

In diesem Abschnitt wollen wir noch einen Baustein vorstellen, mit dem die Struktur der Monome beeinflusst werden kann. Darunter verstehen wir einen Mechanismus, der anhand des Exponenten $\boldsymbol{\iota}$ eines Monoms $(\boldsymbol{x} - \boldsymbol{x}_0)^{\boldsymbol{\iota}}$ entscheidet, ob dieses im Polynom vertreten sein soll oder nicht.

Realisiert wurde diese Funktionalität über die abstrakte Basisklasse `DegreeBase`. Die Klasse `DegreeBase` bietet eine Operation namens `check` an, die als Argument ein Objekt vom Typ `Monom` und die Polynomordnung n verlangt und als Ergebnis wahr oder falsch zurückliefert.

Implementiert und als default eingestellt ist die Klasse `TotalDegree`, welche von der Basisklasse abgeleitet und nach dem Singleton-Prinzip angelegt wurde. Die Operation `check` dieser Implementierung prüft für ein übergebenes Monom $(\boldsymbol{x} - \boldsymbol{x}_0)^{\boldsymbol{\iota}}$, ob

$$|\boldsymbol{\iota}| \leqslant n$$

gilt und liefert entsprechend wahr oder falsch zurück.

Die Funktion `check` wird in den Funktionen und Operationen der Maschinen-Taylor-Modelle verwendet, um zu entscheiden, ob ein Term in ein Polynom einge-

fügt werden soll oder nicht (siehe Abschnitt 4.4.2). Die Klasse `TaylorModel_Impl` enthält hierzu einen Zeiger namens `degree_` auf ein Objekt der Klasse `DegreeBase`, der per Default die Adresse des `TotalDegree`-Objekts enthält.

4.4.2 Arithmetik

In diesem Abschnitt werden wir die arithmetischen Operationen für Maschinen-Taylor-Modelle definieren, deren Implementierung im Detail erläutern und auf die Verwendung der Mechanismen zur Veränderung der Polynomstruktur eingehen.

Für die Definition der Maschinen-Operationen greifen wir auf die Definitionen aus Abschnitt 2.2 zurück, wobei wir hier im Rahmen der Verifikation verschiedene Fehlerquellen zu berücksichtigen haben. Einen ersten Eindruck über die Vorgehensweise zur Definition der Operationen haben wir schon in Abschnitt 4.1 am Beispiel der Addition gegeben.

Im Folgenden seien die Operanden $[\![P_{n,y_1}, I_{n,y_1}]\!]$ und $[\![P_{n,y_2}, I_{n,y_2}]\!]$ Maschinen-Taylor-Modelle aus $\mathcal{TM}_n^{fl}(\boldsymbol{D})$. Dabei ist

$$\boldsymbol{D} := [\underline{x}_1, \overline{x}_1] \times \ldots \times [\underline{x}_\nu, \overline{x}_\nu] \in \mathbb{IS}^\nu \quad \text{und} \quad \boldsymbol{x}_0 \in \boldsymbol{D}.$$

Es bezeichnen $+_{fl}$, $-_{fl}$ und $\cdot_{fl}$ die in Abschnitt 4.1 eingeführten Maschinen-Operationen in S und $\mathbb{IS}$.

4.4.2.1 Addition und Subtraktion

Wir gehen zunächst auf die Definition der Addition ein und kommen anschließend auf die Subtraktion zu sprechen.

Nach (2.15) ergibt sich in $\mathcal{TM}_n(\boldsymbol{D})$ die Summe der Maschinen-Taylor-Modelle $[\![P_{n,y_1}, I_{n,y_1}]\!]$ und $[\![P_{n,y_2}, I_{n,y_2}]\!]$ zu

$$[\![P_{n,y_1}, I_{n,y_1}]\!] \boxplus [\![P_{n,y_2}, I_{n,y_2}]\!] = [\![P_{n,y_1+y_2}, I_{n,y_1+y_2}]\!]$$

mit

$$P_{n,y_1+y_2} = P_{n,y_1} + P_{n,y_2}, \quad I_{n,y_1+y_2} = I_{n,y_1} + I_{n,y_2},$$

wobei P_{n,y_1+y_2} im Allgemeinen kein Maschinen-Polynom und I_{n,y_1+y_2} im Allgemeinen kein Maschinenintervall ist. Es ist für $\boldsymbol{x} \in \boldsymbol{D}$ beliebig

$$\begin{aligned} P_{n,y_1}(\boldsymbol{x} - \boldsymbol{x}_0) + P_{n,y_2}(\boldsymbol{x} - \boldsymbol{x}_0) &= \sum_{|\boldsymbol{\iota}| = 0}^{n} c_{\boldsymbol{\iota}}^{(y_1)} (\boldsymbol{x} - \boldsymbol{x}_0)^{\boldsymbol{\iota}} + \sum_{|\boldsymbol{\iota}| = 0}^{n} c_{\boldsymbol{\iota}}^{(y_2)} (\boldsymbol{x} - \boldsymbol{x}_0)^{\boldsymbol{\iota}} \\ &\in \sum_{|\boldsymbol{\iota}| = 0}^{n} \underbrace{\left(\left[c_{\boldsymbol{\iota}}^{(y_1)}\right] +_{fl} \left[c_{\boldsymbol{\iota}}^{(y_2)}\right] \right)}_{=: I_{\boldsymbol{\iota}}^{(y_1+y_2)}} (\boldsymbol{x} - \boldsymbol{x}_0)^{\boldsymbol{\iota}}. \end{aligned} \tag{4.5}$$

Mit den Indexmengen

$$\Upsilon_{y_1+y_2} := \left\{ \boldsymbol{\iota} : \left| fl(\mathrm{m}(I_{\boldsymbol{\iota}}^{(y_1+y_2)})) \right| \geqslant \texttt{sparsity\_tol\_} \right\},$$
$$\Upsilon_{r} := \left\{ \boldsymbol{\iota} : \left| fl(\mathrm{m}(I_{\boldsymbol{\iota}}^{(y_1+y_2)})) \right| < \texttt{sparsity\_tol\_} \right\}$$

spalten wir (4.5) auf in

$$P_{n,y_1}(\boldsymbol{x}-\boldsymbol{x}_0) + P_{n,y_2}(\boldsymbol{x}-\boldsymbol{x}_0) \in \sum_{\boldsymbol{\iota} \in \Upsilon_{y_1+y_2}} I_{\boldsymbol{\iota}}^{(y_1+y_2)} (\boldsymbol{x}-\boldsymbol{x}_0)^{\boldsymbol{\iota}} + \sum_{\boldsymbol{\iota} \in \Upsilon_r} I_{\boldsymbol{\iota}}^{(y_1+y_2)} (\boldsymbol{x}-\boldsymbol{x}_0)^{\boldsymbol{\iota}}.$$

Hieraus definieren wir mit den Koeffizienten $c_{\boldsymbol{\iota}}^{(y_1+y_2)} := fl(\mathrm{m}(I_{\boldsymbol{\iota}}^{(y_1+y_2)})) \in S$ ein Maschinen-Polynom

$$\tilde{P}_{n,y_1+y_2}(\boldsymbol{x}-\boldsymbol{x}_0) := \sum_{\boldsymbol{\iota} \in \Upsilon_{y_1+y_2}} c_{\boldsymbol{\iota}}^{(y_1+y_2)} (\boldsymbol{x}-\boldsymbol{x}_0)^{\boldsymbol{\iota}}.$$

Damit gilt

$$\begin{aligned} &P_{n,y_1+y_2}(\boldsymbol{x}-\boldsymbol{x}_0) - \tilde{P}_{n,y_1+y_2}(\boldsymbol{x}-\boldsymbol{x}_0) \\ &\qquad \in \sum_{\boldsymbol{\iota} \in \Upsilon_{y_1+y_2}} \left(I_{\boldsymbol{\iota}}^{(y_1+y_2)} -_{fl} c_{\boldsymbol{\iota}}^{(y_1+y_2)} \right) (\boldsymbol{x}-\boldsymbol{x}_0)^{\boldsymbol{\iota}} + \sum_{\boldsymbol{\iota} \in \Upsilon_r} I_{\boldsymbol{\iota}}^{(y_1+y_2)} (\boldsymbol{x}-\boldsymbol{x}_0)^{\boldsymbol{\iota}} \end{aligned}$$

für alle $\boldsymbol{x} \in \boldsymbol{D}$. Folglich erhalten wir mit einem Maschinenintervall I mit

$$\sum_{\boldsymbol{\iota} \in \Upsilon_{y_1+y_2}} \left(I_{\boldsymbol{\iota}}^{(y_1+y_2)} -_{fl} c_{\boldsymbol{\iota}}^{(y_1+y_2)} \right) (\boldsymbol{x}-\boldsymbol{x}_0)^{\boldsymbol{\iota}} + \sum_{\boldsymbol{\iota} \in \Upsilon_r} I_{\boldsymbol{\iota}}^{(y_1+y_2)} (\boldsymbol{x}-\boldsymbol{x}_0)^{\boldsymbol{\iota}} \in I \quad \forall \boldsymbol{x} \in \boldsymbol{D}$$

über die Definition

$$\tilde{I}_{n,y_1+y_2} := I +_{fl} I_{n,y_1} +_{fl} I_{n,y_2} \tag{4.6}$$

ein Maschinen-Taylor-Modell $[\![\tilde{P}_{n,y_1+y_2}, \tilde{I}_{n,y_1+y_2}]\!]$ mit

$$[\![P_{n,y_1+y_2}, I_{n,y_1+y_2}]\!] \subseteq [\![\tilde{P}_{n,y_1+y_2}, \tilde{I}_{n,y_1+y_2}]\!].$$

Das Intervall I enthält dabei alle während der Addition der Koeffizienten entstandenen Rundungsfehler sowie auch den Darstellungsfehler bzgl. der dünnbesetzten Struktur.

Die Fehlererfassung bei der Subtraktion zweier Taylorpolynome wird nach demselben Prinzip durchgeführt. Es ist dort

$$\tilde{P}_{n,y_1-y_2}(\boldsymbol{x}-\boldsymbol{x}_0) := \sum_{\boldsymbol{\iota} \in \Upsilon_{y_1-y_2}} c_{\boldsymbol{\iota}}^{(y_1-y_2)} (\boldsymbol{x}-\boldsymbol{x}_0)^{\boldsymbol{\iota}}, \quad \tilde{I}_{n,y_1-y_2} := I +_{fl} I_{n,y_1} -_{fl} I_{n,y_2}$$

mit Koeffizienten $c_{\boldsymbol{\iota}}^{(y_1-y_2)} := fl(\mathrm{m}(I_{\boldsymbol{\iota}}^{(y_1-y_2)})) \in S$.

Wir definieren

$$[\![P_{n,y_1}, I_{n,y_1}]\!] \boxplus_{fl} [\![P_{n,y_2}, I_{n,y_2}]\!] := [\![\tilde{P}_{n,y_1+y_2}, \tilde{I}_{n,y_1+y_2}]\!],$$

$$[\![P_{n,y_1}, I_{n,y_1}]\!] \boxminus_{fl} [\![P_{n,y_2}, I_{n,y_2}]\!] := [\![\tilde{P}_{n,y_1-y_2}, \tilde{I}_{n,y_1-y_2}]\!]$$

und beschreiben abschließend die Umsetzung der Addition in Software.

Bezogen auf die Hash-Tabellen-Implementierung der Polynome wurde die Addition folgendermaßen programmiert. Zunächst wird vom ersten der beiden Maschinen-Taylor-Modelle eine Kopie angelegt, die als Ergebnis der Addition dient. Wir bezeichnen die Kopie mit `result`. Auf diese Kopie wird nun das zweite Maschinen-Taylor-Modell aufaddiert. Wir beginnen mit dem Polynomanteil.

Die Hash-Tabelle des zweiten Maschinen-Taylor-Modells wird sequentiell mit Hilfe eines `HashTableIterator`-Objekts durchlaufen und Term für Term in die Hash-Tabelle von `result` eingefügt. Ist der einzufügende Term noch nicht in der Hash-Tabelle von `result` vertreten, wird er in die ihm zugewiesene Liste eingehängt. Gibt es schon einen Term mit dem gleichen Monom, dann werden die Koeffizienten als Punktintervalle addiert. Nach der anschließenden Mittelpunktsbildung wird überprüft, ob

$$|c_{\boldsymbol{\iota}}^{(y_1+y_2)}| < \texttt{sparsity\_tol\_}$$

gilt. Abhängig davon wird das Monom $(\boldsymbol{x} - \boldsymbol{x}_0)^{\boldsymbol{\iota}}$ entweder aus der Hash-Tabelle von `result` gelöscht und in einer zuvor angelegten Hash-Tabelle `rest_poly` mit $I_{\boldsymbol{\iota}}^{(y_1+y_2)}$ als Koeffizient eingefügt, oder es verbleibt mit neuem Koeffizient $c_{\boldsymbol{\iota}}^{(y_1+y_2)}$ in der Tabelle. In diesem Fall wird

$$\left(I_{\boldsymbol{\iota}}^{(y_1+y_2)} -_{fl} c_{\boldsymbol{\iota}}^{(y_1+y_2)}\right) (\boldsymbol{x} - \boldsymbol{x}_0)^{\boldsymbol{\iota}}$$

in `rest_poly` eingetragen.

Sind alle Terme des zweiten Summanden eingefügt, wird eine Einschließung I des durch `rest_poly` dargestellten Polynoms mit Intervallkoeffizienten berechnet und diese ebenso wie die Fehlereinschließung des zweiten Maschinen-Taylor-Modells zur Fehlereinschließung in `result` hinzuaddiert. Analog hierzu wird die Subtraktion implementiert.

4.4.2.2 Multiplikation

Die Multiplikation zweier Maschinen-Taylor-Modelle ist im Vergleich zur Addition und Subtraktion deutlich aufwendiger, verläuft aber bezüglich der Fehlererfassung nach dem gleichen Schema.

Nach (2.22) ergibt sich in $\mathcal{TM}_n(\boldsymbol{D})$ das Produkt der Maschinen-Taylor-Modelle $[\![P_{n,y_1}, I_{n,y_1}]\!]$ und $[\![P_{n,y_2}, I_{n,y_2}]\!]$ zu

$$[\![P_{n,y_1}, I_{n,y_1}]\!] \boxdot [\![P_{n,y_2}, I_{n,y_2}]\!] = [\![P_{n,y_1y_2}, I_{n,y_1y_2} \cap I_{n,y_2y_1}]\!]$$

mit

$$\begin{aligned}
P_{n,y_1y_2} &= P_{n,y_1} \cdot P_{n,y_2} - P_{>n},\\
I_{n,y_1y_2} &= W(P_{>n}) + W(P_{n,y_1})\, I_{n,y_2} + I_{n,y_1}\,\big(W(P_{n,y_2}) + I_{n,y_2}\big),\\
I_{n,y_2y_1} &= W(P_{>n}) + W(P_{n,y_2})\, I_{n,y_1} + I_{n,y_2}\,\big(W(P_{n,y_1}) + I_{n,y_1}\big).
\end{aligned}$$

Auch hier ist P_{n,y_1y_2} im Allgemeinen kein Maschinen-Polynom und $I_{n,y_1y_2} \cap I_{n,y_2y_1}$ im Allgemeinen kein Maschinenintervall. Es bezeichne im Folgenden das Symbol $\sum^{fl}$ die Summation bzgl. der Verknüpfung $+_{fl}$. Dann ist für $\boldsymbol{x} \in \boldsymbol{D}$ beliebig

$$\begin{aligned}
P_{n,y_1}(\boldsymbol{x}-\boldsymbol{x}_0)\cdot P_{n,y_2}(\boldsymbol{x}-\boldsymbol{x}_0) &= \Big(\sum_{|\boldsymbol{\iota_1}|=0}^{n} c_{\boldsymbol{\iota_1}}^{(y_1)}\,(\boldsymbol{x}-\boldsymbol{x}_0)^{\boldsymbol{\iota_1}}\Big)\cdot\Big(\sum_{|\boldsymbol{\iota_2}|=0}^{n} c_{\boldsymbol{\iota_2}}^{(y_2)}\,(\boldsymbol{x}-\boldsymbol{x}_0)^{\boldsymbol{\iota_2}}\Big)\\
&\in \sum_{|\boldsymbol{\iota}|=0}^{n} \underbrace{\Big(\sum_{\boldsymbol{\iota_1}+\boldsymbol{\iota_2}=\boldsymbol{\iota}}^{fl} \big[c_{\boldsymbol{\iota_1}}^{(y_1)}\big]\cdot_{fl}\big[c_{\boldsymbol{\iota_2}}^{(y_2)}\big]\Big)}_{=:I_{\boldsymbol{\iota}}^{(y_1y_2)}}(\boldsymbol{x}-\boldsymbol{x}_0)^{\boldsymbol{\iota}}.
\end{aligned}$$

Verwenden wir die Indexmengen

$$\begin{aligned}
\Upsilon_{y_1y_2} &:= \big\{\boldsymbol{\iota}:\ \texttt{check}(\boldsymbol{\iota}) = \texttt{true} \quad\wedge\quad \big|fl(\mathrm{m}(I_{\boldsymbol{\iota}}^{(y_1y_2)}))\big| \geqslant \texttt{sparsity_tol_}\big\},\\
\Upsilon_{r} &:= \big\{\boldsymbol{\iota}:\ \texttt{check}(\boldsymbol{\iota}) = \texttt{false} \quad\vee\quad \big|fl(\mathrm{m}(I_{\boldsymbol{\iota}}^{(y_1y_2)}))\big| < \texttt{sparsity_tol_}\big\},
\end{aligned}$$

so erhalten wir

$$P_{n,y_1}(\boldsymbol{x}-\boldsymbol{x}_0)\cdot P_{n,y_2}(\boldsymbol{x}-\boldsymbol{x}_0) \ \in \sum_{\boldsymbol{\iota}\,\in\,\Upsilon_{y_1y_2}} I_{\boldsymbol{\iota}}^{(y_1y_2)}\,(\boldsymbol{x}-\boldsymbol{x}_0)^{\boldsymbol{\iota}} + \sum_{\boldsymbol{\iota}\,\in\,\Upsilon_{r}} I_{\boldsymbol{\iota}}^{(y_1y_2)}\,(\boldsymbol{x}-\boldsymbol{x}_0)^{\boldsymbol{\iota}}.$$

Mit den Koeffizienten $c_{\boldsymbol{\iota}}^{(y_1y_2)} := fl(m(I_{\boldsymbol{\iota}}^{(y_1y_2)}))$ definieren wir ein Maschinen-Polynom

$$\tilde{P}_{n,y_1y_2}(\boldsymbol{x}-\boldsymbol{x}_0) := \sum_{\boldsymbol{\iota}\,\in\,\Upsilon_{y_1y_2}} c_{\boldsymbol{\iota}}^{(y_1y_2)}\,(\boldsymbol{x}-\boldsymbol{x}_0)^{\boldsymbol{\iota}},$$

für das

$$\begin{aligned}
&P_{n,y_1y_2}(\boldsymbol{x}-\boldsymbol{x}_0) - \tilde{P}_{n,y_1y_2}(\boldsymbol{x}-\boldsymbol{x}_0)\\
&\qquad\in \sum_{\boldsymbol{\iota}\,\in\,\Upsilon_{y_1y_2}} \big(I_{\boldsymbol{\iota}}^{(y_1y_2)} -_{fl} c_{\boldsymbol{\iota}}^{(y_1y_2)}\big)\,(\boldsymbol{x}-\boldsymbol{x}_0)^{\boldsymbol{\iota}} + \sum_{\boldsymbol{\iota}\,\in\,\Upsilon_{r}} I_{\boldsymbol{\iota}}^{(y_1y_2)}\,(\boldsymbol{x}-\boldsymbol{x}_0)^{\boldsymbol{\iota}}
\end{aligned}$$

für alle $\boldsymbol{x} \in \boldsymbol{D}$ gilt. Demzufolge erhalten wir mit einem Maschinenintervall I mit

$$\sum_{\boldsymbol{\iota}\,\in\,\Upsilon_{y_1y_2}} \big(I_{\boldsymbol{\iota}}^{(y_1y_2)} -_{fl} c_{\boldsymbol{\iota}}^{(y_1y_2)}\big)\,(\boldsymbol{x}-\boldsymbol{x}_0)^{\boldsymbol{\iota}} + \sum_{\boldsymbol{\iota}\,\in\,\Upsilon_{r}} I_{\boldsymbol{\iota}}^{(y_1y_2)}\,(\boldsymbol{x}-\boldsymbol{x}_0)^{\boldsymbol{\iota}} \in I \quad \forall \boldsymbol{x} \in \boldsymbol{D}$$

über die Definition

$$\tilde{I}_{n,y_1y_2} := I +_{fl} I_{n,y_1} \cdot_{fl} \left(I_{P_{n,y_2}} +_{fl} I_{n,y_2}\right) +_{fl} I_{P_{n,y_1}} \cdot_{fl} I_{n,y_2},$$
$$\tilde{I}_{n,y_2y_1} := I +_{fl} I_{n,y_2} \cdot_{fl} \left(I_{P_{n,y_1}} +_{fl} I_{n,y_1}\right) +_{fl} I_{P_{n,y_2}} \cdot_{fl} I_{n,y_1},$$

ein Maschinen-Taylor-Modell $[\![\tilde{P}_{n,y_1y_2}, \tilde{I}_{n,y_1y_2} \cap \tilde{I}_{n,y_2y_1}]\!]$ mit

$$[\![P_{n,y_1y_2}, I_{n,y_1y_2} \cap I_{n,y_2y_1}]\!] \subseteq [\![\tilde{P}_{n,y_1y_2}, \tilde{I}_{n,y_1y_2} \cap \tilde{I}_{n,y_2y_1}]\!].$$

Die Intervalle $I_{P_{n,y_1}}$ und $I_{P_{n,y_2}}$ sind Maschinenintervalle, die den Wertebereich von P_{n,y_1} bzw. P_{n,y_2} einschließen. Auftretende Rundungs- und Darstellungs- bzw. Abschneidefehler innerhalb der Polynommultiplikation werden durch I erfasst.

Wir definieren

$$[\![P_{n,y_1}, I_{n,y_1}]\!] \boxdot_{fl} [\![P_{n,y_2}, I_{n,y_2}]\!] := [\![\tilde{P}_{n,y_1y_2}, \tilde{I}_{n,y_1y_2} \cap \tilde{I}_{n,y_2y_1}]\!]$$

und beschreiben abschließend die Umsetzung der Multiplikation in Software.

Für die softwaremäßige Realisierung in der zugrundeliegenden Datenstruktur wurden drei Polynome, also drei Hash-Tabellen verwendet: zwei Polynome mit Intervallkoeffizienten, `iprod_poly` und `rest_poly`, und ein Polynom `prod_poly` mit Gleitkommakoeffizienten für die Speicherung des Ergebnisses. Die Tabelle `iprod_poly` speichert relevante Terme des Produkts der beiden Polynome.

Das Produkt wird gebildet, indem jeder Term des einen Polynoms mit jedem Term des anderen Polynoms multipliziert wird. Gilt für ein Monomprodukt $(\boldsymbol{x} - \boldsymbol{x}_0)^{\boldsymbol{\iota_1}} \cdot (\boldsymbol{x} - \boldsymbol{x}_0)^{\boldsymbol{\iota_2}} = (\boldsymbol{x} - \boldsymbol{x}_0)^{\boldsymbol{\iota_1} + \boldsymbol{\iota_2}} =: (\boldsymbol{x} - \boldsymbol{x}_0)^{\boldsymbol{\iota}}$

$$\texttt{check}(\boldsymbol{\iota}) = \texttt{true},$$

dann wird der Term

$$I_{\boldsymbol{\iota}}^{(y_1y_2)} (\boldsymbol{x} - \boldsymbol{x}_0)^{\boldsymbol{\iota}}$$

in `iprod_poly` abgelegt, andernfalls in `rest_poly`. Nachdem alle Terme miteinander multipliziert wurden, wird `iprod_poly` Term für Term durchlaufen und von den Koeffizienten $I_{\boldsymbol{\iota}}^{(y_1y_2)}$ der Mittelpunkt bestimmt. Abhängig davon ob

$$|c_{\boldsymbol{\iota}}^{(y_1y_2)}| < \texttt{sparsity\_tol\_}$$

gilt, wird entweder $I_{\boldsymbol{\iota}}^{(y_1y_2)} (\boldsymbol{x} - \boldsymbol{x}_0)^{\boldsymbol{\iota}}$ nach `rest_poly` verschoben oder $c_{\boldsymbol{\iota}}^{(y_1y_2)} (\boldsymbol{x} - \boldsymbol{x}_0)^{\boldsymbol{\iota}}$ in `prod_poly` abgelegt und $\left(I_{\boldsymbol{\iota}}^{(y_1y_2)} -_{fl} c_{\boldsymbol{\iota}}^{(y_1y_2)}\right) (\boldsymbol{x} - \boldsymbol{x}_0)^{\boldsymbol{\iota}}$ zu `rest_poly` addiert.

Sind alle Terme aus `iprod_poly` auf diese Weise verarbeitet worden, wird von dem Polynom in `rest_poly` eine Wertebereichseinschließung I berechnet, die nach obigem die Rundungsfehler und auch alle auftretenden Darstellungsfehler enthält.

4.4.2.3 Standardfunktionen

Die Implementierung der Standardfunktionen basiert auf der Addition und Multiplikation von Maschinen-Taylor-Modellen und greift demnach auf deren Implementierung zurück. So ist lediglich die Auswertung des jeweiligen Taylorpolynoms nach dem Horner-Schema sowie die zugehörige Berechnung der Fehlereinschließungen zu implementieren. Alle Standardfunktionen wurden nach der algorithmischen Beschreibung am Beispiel der Exponentialfunktion bzw. der Division auf den Seiten 28 bis 33 realisiert. Die benötigten Fakultäten werden von der Singleton-Klasse `Factorials` angelegt und zur Verfügung gestellt.

Bemerkung 4.1 Wir machen an dieser Stelle darauf aufmerksam, dass die Berechnung eines Maschinen-Taylor-Modells einer Funktion f über die Auswertung eines Funktionsausdrucks von f erfolgt. Eventuelle Darstellungsfehler in den Argumenten sind dabei in den Fehlereinschließungen der zugehörigen Maschinen-Taylor-Modelle zu berücksichtigen.

Sind beispielsweise einzelne Entwicklungsstellen nicht darstellbar, so ist vor der Auswertung des Funktionsausdrucks, also vor der Berechnung des Taylorpolynoms, bei den Maschinen-Taylor-Modellen der Variablen der (bekannte) Darstellungsfehler explizit in der zugehörigen Fehlereinschließung festzuhalten.

Als Resultat der Auswertung erhalten wir ein Taylorpolynom mit dünnbesetzter Struktur und Termen von der in `check` definierten Gestalt. Die Fehlereinschließung enthält sowohl die Rundungsfehler aus arithmetischen Operationen als auch den Darstellungsfehler des berechneten Taylorpolynoms zum tatsächlichen reellen Taylorpolynom von f.

4.4.2.4 Integration

Wir definieren in diesem Abschnitt die letzte noch verbleibende Operation in der Menge $\mathcal{TM}_n^{fl}(\boldsymbol{D})$ der Maschinen-Taylor-Modelle, die unbestimmte Integration bzgl. einer Variablen x_k, $k \in \{1, \ldots, \nu\}$. Sei dazu k beliebig aber fest gewählt und $[\![P_{n,y}, I_{n,y}]\!] \in \mathcal{TM}_n^{fl}(\boldsymbol{D})$ ein Maschinen-Taylor-Modell.

Nach (2.39) berechnet sich in $\mathcal{TM}_n(\boldsymbol{D})$ das unbestimmte Integral der Menge $[\![P_{n,y}, I_{n,y}]\!]$ zu

$$\int\!\!\!\!\square [\![P_{n,y}, I_{n,y}]\!]\, dx_k := [\![P_{n,Y}, I_{n,Y}]\!]$$

mit der Fehlereinschließung

$$I_{n,Y} = I_{n,y} \cdot \left([\underline{x}_k, \overline{x}_k] - x_{k_0}\right) + W(P_{>n})$$

und dem Polynom $P_{n,Y} = P_{n+1,Y} - P_{>n}$, das sich aus der Integration des Taylorpolynoms $P_{n,y}$ und anschließendem Abspalten der Terme mit Ordnung $n+1$

vom integrierten Polynom ergibt. Das Polynom wie auch die Fehlereinschließung sind im Allgemeinen nicht ohne einen Informationsverlust im Gleitkommaraster S darstellbar.

Bevor wir nun auf die Definition der unbestimmten Integration für Maschinen-Taylor-Modelle eingehen, führen wir noch eine Bezeichnung ein, welche die nachfolgende Darstellung übersichtlich macht. Es bezeichne $\mathbf{1}_k$ einen Vektor aus $\mathbb{N}^\nu$ der in der k-ten Komponente die Zahl 1 und ansonsten die Zahl 0 enthält. Damit ist

$$\boldsymbol{\iota} + \mathbf{1}_k = (i_1, \ldots, i_{k-1}, i_k + 1, i_{k+1}, \ldots, i_\nu).$$

Es gilt für beliebiges $\boldsymbol{x} \in \boldsymbol{D}$ mit $\tilde{\boldsymbol{x}} := (x_1, \ldots, x_{k-1}, \xi, x_{k+1}, \ldots, x_\nu)$

$$\begin{aligned}\int_{x_{k_0}}^{x_k} P_{n,y}(\tilde{\boldsymbol{x}} - \boldsymbol{x}_0)\, d\xi &= \int_{x_{k_0}}^{x_k} \sum_{|\boldsymbol{\iota}|=0}^{n} c_{\boldsymbol{\iota}} (\tilde{\boldsymbol{x}} - \boldsymbol{x}_0)^{\boldsymbol{\iota}}\, d\xi \\ &\in \sum_{|\boldsymbol{\iota}|=0}^{n} \underbrace{\left([c_{\boldsymbol{\iota}}] \cdot_{fl} \frac{1}{i_k + 1} \right)}_{=:I_{\boldsymbol{\iota}}} (\boldsymbol{x} - \boldsymbol{x}_0)^{\boldsymbol{\iota} + \mathbf{1}_k}.\end{aligned}$$

Verwenden wir die Indexmengen

$$\begin{aligned}\Upsilon_Y &:= \left\{ \boldsymbol{\iota} : \texttt{check}(\boldsymbol{\iota} + \mathbf{1}_k) = \texttt{true} \quad \wedge \quad \left| fl(\mathrm{m}(I_{\boldsymbol{\iota}})) \right| \geqslant \texttt{sparsity\_tol\_} \right\}, \\ \Upsilon_r &:= \left\{ \boldsymbol{\iota} : \texttt{check}(\boldsymbol{\iota} + \mathbf{1}_k) = \texttt{false} \quad \vee \quad \left| fl(\mathrm{m}(I_{\boldsymbol{\iota}})) \right| < \texttt{sparsity\_tol\_} \right\}\end{aligned}$$

so erhalten wir

$$\int_{x_{k_0}}^{x_k} P_{n,y}(\tilde{\boldsymbol{x}} - \boldsymbol{x}_0)\, d\xi \quad \in \quad \sum_{\boldsymbol{\iota} \in \Upsilon_Y} I_{\boldsymbol{\iota}} (\boldsymbol{x} - \boldsymbol{x}_0)^{\boldsymbol{\iota} + \mathbf{1}_k} + \sum_{\boldsymbol{\iota} \in \Upsilon_r} I_{\boldsymbol{\iota}} (\boldsymbol{x} - \boldsymbol{x}_0)^{\boldsymbol{\iota} + \mathbf{1}_k}.$$

Mit den Koeffizienten $c_{\boldsymbol{\iota}}^{(Y)} := fl(m(I_{\boldsymbol{\iota}}))$ definieren wir ein Maschinen-Polynom

$$\tilde{P}_{n,Y}(\boldsymbol{x} - \boldsymbol{x}_0) := \sum_{\boldsymbol{\iota} \in \Upsilon_Y} c_{\boldsymbol{\iota}}^{(Y)} (\boldsymbol{x} - \boldsymbol{x}_0)^{\boldsymbol{\iota} + \mathbf{1}_k},$$

für das

$$P_{n,Y}(\boldsymbol{x} - \boldsymbol{x}_0) - \tilde{P}_{n,Y}(\boldsymbol{x} - \boldsymbol{x}_0) \in \sum_{\boldsymbol{\iota} \in \Upsilon_Y} \left(I_{\boldsymbol{\iota}} -_{fl} c_{\boldsymbol{\iota}}^{(Y)} \right) (\boldsymbol{x} - \boldsymbol{x}_0)^{\boldsymbol{\iota} + \mathbf{1}_k} + \sum_{\boldsymbol{\iota} \in \Upsilon_r} I_{\boldsymbol{\iota}} (\boldsymbol{x} - \boldsymbol{x}_0)^{\boldsymbol{\iota} + \mathbf{1}_k}$$

für alle $\boldsymbol{x} \in \boldsymbol{D}$ gilt. Demzufolge erhalten wir mit einem Maschinenintervall I mit

$$\sum_{\boldsymbol{\iota} \in \Upsilon_Y} \left(I_{\boldsymbol{\iota}} -_{fl} c_{\boldsymbol{\iota}}^{(Y)} \right) (\boldsymbol{x} - \boldsymbol{x}_0)^{\boldsymbol{\iota} + \mathbf{1}_k} + \sum_{\boldsymbol{\iota} \in \Upsilon_r} I_{\boldsymbol{\iota}} (\boldsymbol{x} - \boldsymbol{x}_0)^{\boldsymbol{\iota} + \mathbf{1}_k} \in I \quad \forall \boldsymbol{x} \in \boldsymbol{D}$$

über die Definition

$$\tilde{I}_{n,Y} := I +_{fl} I_{n,y} \cdot_{fl} \left([\underline{x}_k, \overline{x}_k] -_{fl} x_{k_0} \right), \tag{4.7}$$

ein Maschinen-Taylor-Modell $[\![\tilde{P}_{n,Y}, \tilde{I}_{n,Y}]\!]$ mit

$$[\![P_{n,Y}, I_{n,Y}]\!] \subseteq [\![\tilde{P}_{n,Y}, \tilde{I}_{n,Y}]\!].$$

Das Intervall I enthält dabei alle Rundungs- und Darstellungsfehler.

Wir definieren

$$\oint\!\!\!\square_{fl} [\![P_{n,y}, I_{n,y}]\!] \, dx_k := [\![\tilde{P}_{n,Y}, \tilde{I}_{n,Y}]\!]$$

und beschreiben abschließend die Umsetzung der Integration in Software. Bezüglich der zugrundeliegenden Datenstruktur wurden für die Realisierung der Integration zwei Polynome bzw. Hash-Tabellen verwendet, und zwar ein Polynom `result`, welches am Ende $\tilde{P}_{n,Y}$ enthält, und ein Polynom `rest_poly` mit Intervallkoeffizienten. Wir beginnen mit dem Polynomanteil.

Die Hash-Tabelle von $P_{n,y}$ wird sequentiell mit Hilfe eines `HashTableIterator`-Objekts durchlaufen. Jeder Term wird dabei bzgl. der Variablen x_k integriert, es werden die Intervalle $I_{\boldsymbol{\iota}}$ berechnet und geprüft, ob

$$\texttt{check}(\boldsymbol{\iota} + \mathbf{1}_k) = \texttt{true}$$

gilt. Ist dies der Fall, dann wird der Term

$$I_{\boldsymbol{\iota}} \, (\boldsymbol{x} - \boldsymbol{x}_0)^{\boldsymbol{\iota} + \mathbf{1}_k}$$

in `rest_poly` eingefügt, andernfalls wird von $I_{\boldsymbol{\iota}}$ der Mittelpunkt bestimmt und abhängig davon, ob

$$|c_{\boldsymbol{\iota}}^{(Y)}| < \texttt{sparsity\_tol\_}$$

gilt, der Term $I_{\boldsymbol{\iota}}^{(Y)} \, (\boldsymbol{x} - \boldsymbol{x}_0)^{\boldsymbol{\iota} + \mathbf{1}_k}$ in `rest_poly` eingefügt oder $c_{\boldsymbol{\iota}}^{(Y)} \, (\boldsymbol{x} - \boldsymbol{x}_0)^{\boldsymbol{\iota} + \mathbf{1}_k}$ in die Tabelle `result` und

$$\left(I_{\boldsymbol{\iota}} -_{fl} c_{\boldsymbol{\iota}}^{(Y)} \right) \, (\boldsymbol{x} - \boldsymbol{x}_0)^{\boldsymbol{\iota} + \mathbf{1}_k}$$

in `rest_poly` eingetragen.

Sind alle Terme von $P_{n,y}$ abgearbeitet, wird eine Einschließung I des durch `rest_poly` dargestellten Polynoms bestimmt und die Fehlereinschließung nach (4.7) berechnet.

4.4.3 Besondere Mechanismen

In diesem Abschnitt stellen wir zwei besondere Mechanismen vor, die speziell in der Anwendung der Taylor-Modelle zur Lösungseinschließung bei gewöhnlichen Differentialgleichungen die Rechenzeit erheblich reduzieren.

4.4.3.1 Record-Feature

Die Auswertung von Funktionsausdrücken mit Maschinen-Taylor-Modellen kann je nach Ausdruck, Polynomgrad und Anzahl an Variablen im Polynom sehr zeitintensiv sein, wobei der Großteil der Zeit für die Berechnung des polynomialen Anteils verwendet wird.

Wird ein Ausdruck mehrmals hintereinander mit ein und demselben Taylorpolynom, aber unterschiedlichen Fehlereinschließungen, ausgewertet, so macht sich dies negativ bemerkbar. Beispielsweise tritt dies im Rahmen der Lösungseinschließung von Differentialgleichungen innerhalb der iterativen Berechnung der Fehlereinschließung (vgl. (3.24)) oder daran anschließend bei deren iterativen Verbesserung auf (vgl. (3.27)). Dort ist in beiden Fällen nur die berechnete Fehlereinschließung von Interesse. Das Taylorpolynom ist bekannt und für die dortige Fragestellung uninteressant. Einzig und allein der aus der Berechnung des Taylorpolynoms resultierende Beitrag zur Fehlereinschließung ist relevant. Dieser ist in allen Aufrufen gleich und unabhängig von der sich ändernden Fehlereinschließung des Arguments.

Das Record-Feature ermöglicht in den eben genannten Fällen, auf die unnötigen Polynomberechnungen zu verzichten. Dazu wird durch Setzen des Flags `REMAINDER_REC` die Berechnung der Fehlereinschließung in jeder aufgerufenen Funktion und Operation aufgezeichnet, d. h. es werden jeweils die Beiträge aus der Berechnung des Taylorpolynoms zur Fehlereinschließung gespeichert. So wird beispielsweise bei der Addition von Maschinen-Taylor-Modellen das Intervall I gespeichert (vgl. (4.6)). Zur Speicherung wird ein Array verwendet, da in der Regel die Addition mehrfach aufgerufen wird.

Ist die Auswertung beendet, also die Aufzeichnung abgeschlossen, so ist für alle nachfolgenden Auswertungen das Flag `REMAINDER_PLAY` zu setzen. Von nun an wird innerhalb der aufgerufenen Funktionen und Operatoren nur noch die Fehlereinschließung unter Verwendung der gespeicherten Werte berechnet. Am Beispiel der Addition heißt dies, dass die Fehlereinschließung nach (4.6) berechnet wird, indem das entsprechende Intervall I aus dem Array der gespeicherten Intervalle gewählt und mit den Fehlereinschließungen der Argumente verrechnet wird. Wird das Record-Feature nicht oder nicht mehr benötigt, muss das Flag `NORMAL_MODE` gesetzt werden.

4.4.3.2 Abschalten der Rundungsfehlererfassung

Für Intervalloperationen wird mindestens doppelt soviel Rechenzeit aufgewendet wie für die entsprechenden Operationen mit Gleitkommazahlen. Es wäre also nützlich, wenn man die Rundungsfehlererfassung mittels Intervallen ausschalten kann, wenn sie nicht gebraucht wird, wie zum Beispiel bei der Berechnung des Taylorpolynoms der Lösung mittels Picard-Iteration.

Zur Realisierung dieses Features haben wir die arithmetischen Grundoperationen $\{\boxplus_{fl}, \boxminus_{fl}, \boxdot_{fl}\}$ für Maschinen-Taylor-Modelle ein zweites Mal ohne die Erfassung der Rundungs- und Darstellungsfehler implementiert. Die mitgeführte Fehlereinschließung wird in diesem Fall sinnlos und bleibt deshalb unberücksichtigt.

Man gelangt in diesen Modus durch setzen des Flag `NO_REMAINDER`. Alle nachfolgend aufgerufenen Funktionen und Operationen werden ohne Fehlererfassung durchgeführt. Zurück zur verifizierten Rechnung kommt man durch Setzen des Flag `NORMAL_MODE`.

4.4.4 Wertebereichseinschließung

Die Berechnung der Wertebereichseinschließung von Polynomen spielt in der Implementierung der Grundarithmetik und der Standardfunktionen eine wichtige Rolle. Damit die Arithmetik unabhängig von den Einschließungsverfahren bleibt, wurden diese als separates und austauschbares Modul programmiert.

Die abstrakte Basisklasse `PolynomialRange` gibt die Schnittstelle vor. Konkrete Verfahren können durch Vererbung und Definition der Operationen in den Unterklassen verwirklicht werden. Die Basisklasse bleibt abstrakt, da die Menge an auszuführendem Code in den Operationen in der Regel recht umfangreich ist und damit der Overhead, der aus den virtuellen Funktionen [59, S. 328] resultiert, vernachlässigt werden kann.

Derzeit liegen Implementierungen der naiven Intervallauswertung, der Mittelwertform und des LDB (vgl. Satz 5, S. 5) vor. Die Unterklassen wurden nach dem Singleton-Prinzip konstruiert.

4.4.5 Übersicht

Im Folgenden wird der Inhalt des Basispakets nochmals in übersichtlicher Form dargestellt. Die eingeführten Klassen werden gruppiert und in Paketen zusammengefasst. Am Ende des Abschnitts werden alle Klassen in einem Klassendiagramm dargestellt und deren Verbindungen untereinander anschaulich präsentiert.

Wir gliedern das Paket TM-Base in die Pakete Sonstiges, Hash-Tabelle, Beobachter und Taylor-Modell, welches auf den drei anderen Paketen aufsetzt (Abb. 4.16).

Im Paket Sonstiges fassen wir die Klassen `PrimeNumbers`, `Factorials`, `Interval` und die generische Klasse `ReferenceCounter` zusammen. Es handelt sich hierbei um allgemeine Bausteine unabhängig vom Kontext der anderen Pakete.

Im Paket Hash-Tabelle werden alle in der Implementierung der Hash-Tabelle vorkommenden Klassen und Strukturen zusammengefasst. Dies sind die Klassen `HashTable`, `HashTableData`, `HashTableIterator`, `HashTableConstIterator`, `BasicAction` und die Struktur `HashTableEntry`. Die Beziehung der Klassen un-

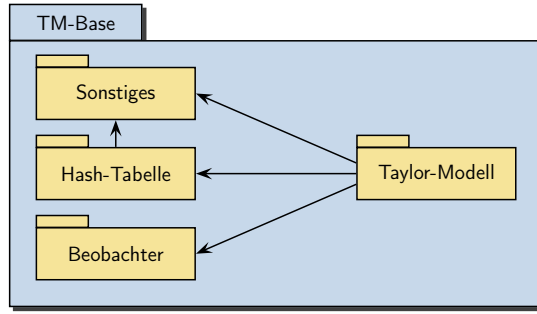

Abbildung 4.16: Gliederung von TM-Base.

tereinander kann der Abbildung 4.11 entnommen werden. Die Implementierung greift auf `PrimeNumbers` aus Sonstiges zurück (Abb. 4.16).

Das Paket Taylor-Modell wird aus den Klassen `TaylorModel`, `TaylorModel_Impl`, `Variables`, `VarData`, `Monom`, `PolynomialRange`, `DegreeBase`, `TotalDegree` und der Struktur `VarDataListNode` geschnürt. Die Verbindungen der Elemente untereinander können aus der Gesamtübersicht in Abbildung 4.17 entnommen werden.

Zu guter Letzt bleibt noch das Paket Beobachter. Der Inhalt setzt sich aus den Klassen `Observer`, `Subject` und `Action` zusammen, die nach dem Beobachtermuster [18, S. 287-300] konzipiert sind. Durch Vererbung kann den Unterklassen die Rolle des Beobachters oder des zu beobachtenden Subjekts zugewiesen werden. Beobachter-Objekte können sich bei `Subject`-Objekten in eine Liste eintragen lassen und werden bei Aktionen oder Zustandsänderungen von diesen darüber informiert. Denkbar ist ein Einsatz zusammen mit der Variablentabelle. Die Eintragung neuer Variablen, wie auch die Änderung eines Variablensymbols zur Laufzeit, kann an Objekte, die auf diese Information angewiesen sind, weitergegeben werden.

Abschließend zeigt Abbildung 4.17 ein Klassendiagramm, das die Beziehungen der Klassen untereinander darstellt. Der Übersichtlichkeit wegen wird auf ein vollständiges Klassendiagramm verzichtet und nur die primären Klassen und deren wesentliche Beziehungen untereinander dargestellt.

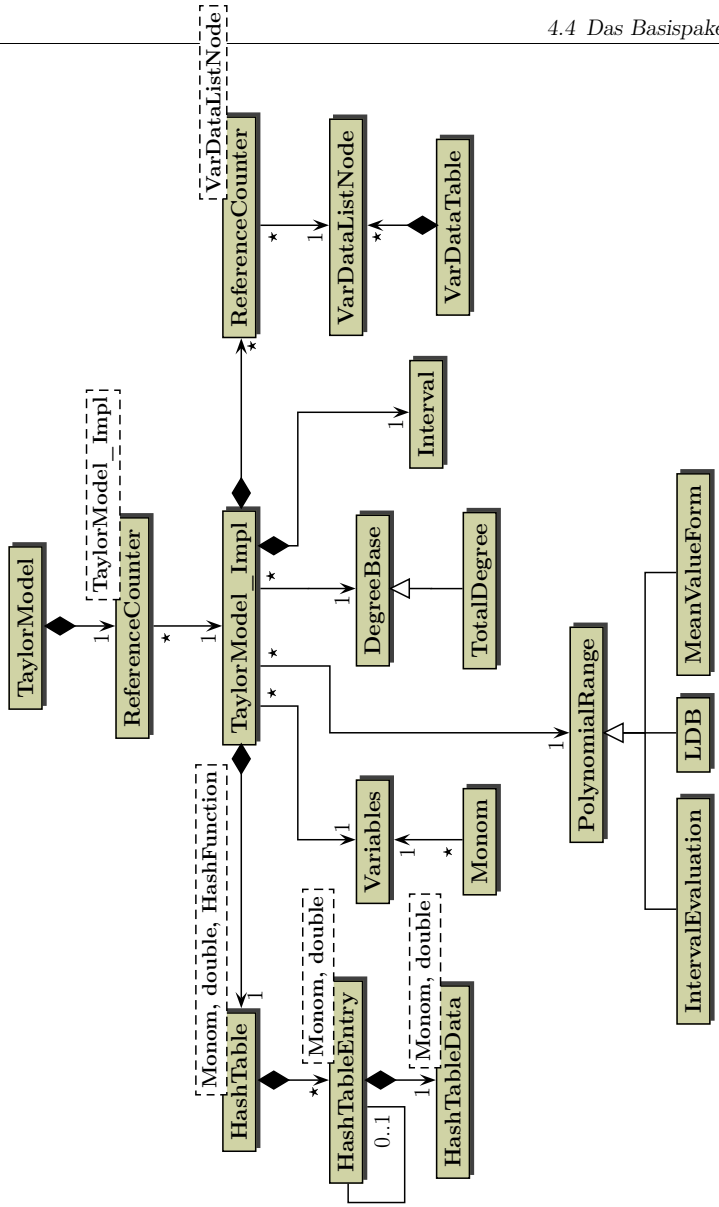

Abbildung 4.17: Übersichtsdiagramm.

4.5 Das DGL-Paket

Das Paket DGL setzt sich zusammen aus Klassen und Strukturen für die Lösungseinschließung bei gewöhnlichen Anfangswertproblemen mittels Taylor-Modellen. Der Algorithmus zur Lösungseinschließung und der Algorithmus des Shrink-Wrapping wurde jeweils in einer Klasse nach dem Singleton-Prinzip gekapselt. Die Schnittstelle beider Klassen bietet jeweils eine Operation zur Initialisierung und eine Operation zur Ausführung des Algorithmus an (Abb. 4.18).

Abbildung 4.18: Die Klassen `ShrinkWrapping` und `DGLSolver`.

Die Initialisierung von `DGLSolver` erfolgt über eine Struktur namens `Data`, welche die folgenden Informationen enthält:

- Funktionszeiger auf die Komponenten der rechten Seite der Differentialgleichung sowie ihre Dimension.
- Die Anfangswerte und die Stelle t_0, an der das Anfangswertproblem gegeben ist.
- Die Ordnung n der Maschinen-Taylor-Modelle, den Algorithmus zur Wertebereichseinschließung von Polynomen (vgl. Abschnitt 4.4.4), das Objekt zur Ordnungsbestimmung der Terme (vgl. Abschnitt 4.4.1.5) und die Toleranzgrenze `sparsity_tol_` für die Polynomkoeffizienten.
- Angaben zur Schrittweitensteuerung. Hier kann zwischen der Verwendung von konstanten Schrittweiten und vom Programm zu berechnenden Schrittweiten gewählt werden.

  Im Falle konstanter Schrittweiten kann der Benutzer entweder nur eine Schrittweite angegeben, die dann in jedem Integrationsschritt verwendet wird, oder aber eine Liste von mehreren Schrittweiten, mit der für jeden Integrationsschritt eine Schrittweite vorgegeben wird. Hierdurch wird gleichzeitig der Integrationsbereich abgesteckt, dessen rechte Grenze sich aus der Summe der Schrittweiten und t_0 ergibt.

  Entscheidet man sich für die Berechnung der Schrittweiten durch das Programm, dann ist hierfür eine Startschrittweite h_0, eine untere Schranke h_{min} für die Schrittweiten sowie zusätzlich noch eine Fehlertoleranzgrenze für den lokalen Fehlerzuwachs pro Integrationsschritt anzugeben.

- Eine Liste mit der Größe nach geordneten Knoten $t_0 < \tilde{t}_1 < \tilde{t}_2 < \ldots < \tilde{t}_m$, $m \in \mathbb{N}$, an denen eine Lösungseinschließung berechnet werden soll. Bei automatischer Schrittweitensteuerung sowie bei einer einzelnen konstanten Schrittweite ist der Integrationsbereich gegeben durch $[t_0, \tilde{t}_m]$. Kommt die Schrittweitenliste zur Anwendung, wird eine Lösungseinschließung an den Knoten $\tilde{t}_k$ berechnet, die innerhalb des durch die Schrittweiten abgesteckten Integrationsbereichs liegen.

- Vom Benutzer vorgegebene Bezeichner für die Anfangswerte sowie auch einen Bezeichner für die Zeitvariable.

Während der Initialisierungsphase werden die Eingabedaten auf Korrektheit geprüft, diverse (Hilfs-)Variablen angelegt und definiert sowie der Shrink-Wrapping-Algorithmus initialisiert. Erst nach erfolgreicher Initialisierung kann `DGLSolver` gestartet werden.

Anders als bei der Beschreibung des Basispakets wollen wir hier die Implementierung des Algorithmus zur Lösungseinschließung anhand des Quellcodes vorstellen. Dies soll einerseits als Einführung in den Quellcode dienen, andererseits aber auch alle Details des Einschließungsverfahrens aufzeigen.

Eine Vorstellung der verwendeten Hilfsklassen und -strukturen sowie die Zusammenfassung der Klassen zu Paketen werden wir in Abschnitt 4.5.3 vornehmen.

4.5.1 Der Algorithmus in `DGLSolver`

In `DGLSolver` ist der Algorithmus 7 des erweiterten Einschließungsverfahrens implementiert. Den Quellcode dazu werden wir im Folgenden sequentiell vorstellen, wobei wir das Codelisting immer wieder unterbrechen und die einzelnen Codeabschnitte ausreichend kommentieren werden. Auch die im Code auftretenden Variablen werden wir erläutern.

Das Codelisting zeigt die wesentlichen Teile aus `DGLSolver`. Codezeilen beispielsweise für die Ausgabe von Ergebnissen werden bewusst ausgeblendet, um den eigentlichen Algorithmus so übersichtlich wie möglich präsentieren zu können. Die Nummerierung am Rand veranschaulicht die Reihenfolge der einzelnen Codefragmente, stimmt aber nicht mit den entsprechenden Zeilennummern in RiOT überein.

Wir beginnen mit dem Einstieg in den Algorithmus, der Zeitschleife. Sie veranlasst die Integration des Anfangswertproblems über die Teilintervalle des Integrationsbereichs solange durchzuführen, bis das Ende t_e erreicht ist. Etwaige andere Abbruchkriterien sind zweitrangig und stehen innerhalb der Schleife.

```
1 bool Repeat_Step = false;
2 bool TransformInitVals = true;
```

```
4  while( t_curr_ < sup(t_end_) )
     {
6      ++step_cntr;   //Counts steps of integration.
```

Die Variable `t_curr_` bezeichnet den aktuellen Gitterpunkt und ist beim ersten Durchlauf auf den Wert von t_0 gesetzt. `t_end_` ist ein Intervall mit dem unter Umständen nicht darstellbaren rechten Ende des Integrationsbereichs. Die Variable `Repeat_Step` steuert im Inneren der Schleife den Kontrollfluss im Fall, dass ein schon durchgeführter Integrationsschritt wegen Genauigkeitsverlust erneut durchgeführt werden muss. Die Bedeutung der Variable `TransformInitVals` wird im Anschluss an das nächste Codesegment erläutert.

Bevor wir mit der Konstruktion der Maschinen-Taylor-Modelle beginnen, sind noch einige Vorbereitungen zu treffen. So wird zunächst nach (3.29) die Anfangswertmenge als Bildmenge eines Maschinen-Polynoms definiert auf $[-1,1]^\nu$ dargestellt. Wir führen diese Transformation innerhalb der Zeitschleife durch, damit auch Punktanfangswerte berücksichtigt werden können. In diesem Fall erhalten wir nach dem ersten Integrationsschritt eine Intervalleinschließung der Lösungsmenge, die dann als Anfangswertmenge für den anschließenden Integrationschritt verwendet wird. Diese Anfangswertmenge kann dann beim zweiten Durchlauf der Schleife durch die Komponenten eines Maschinen-Taylor-Modells mit nicht verschwindendem variablem Term dargestellt werden.

```
8      if( TransformInitVals and not Repeat_Step )
         {
10         for(int i=0; i < Data_Ptr_->Dimension_; i++)
             {
12             double mid_point = Data_Ptr_->InitVals_Value_[i].mid();
               Interval domain = Data_Ptr_->InitVals_Value_[i] -
                   mid_point;
14             double scale = domain.sup();

16             TransformInitVals &= (scale < Data_Ptr_->Sparsity_);
               InitValsDGLTM_[i] = mid_point + scale * InitValsTM_[i];
18           }
         }
```

Die boolsche Variable `TransformInitVals` steuert zusammen mit `Repeat_Step` die Durchführung der Transformation. Die dafür benötigten Maschinen-Taylor-Modelle der Form $[\![\alpha_i, [0,0]]\!]$, $\alpha_i \in [-1,1]$, werden von der Variablen `InitValsTM_` bereitgestellt. `Data_Ptr_` ist ein Zeiger auf die Datenstruktur `Data` und `Dimension_` ein Element der Struktur, welches die Dimension des DGL-Systems enthält. In

der Variablen `InitValuesDGLTM_` werden die Maschinen-Taylor-Modelle, welche gemäß (3.30) die Anfangswertmenge der Differentialgleichung darstellen, gespeichert. Fällt bei der Transformation der Skalierungsfaktor `scale` kleiner als die vorgegebene Sparsity-Toleranz `Sparsity_` aus, reduziert sich der Polynomanteil der Maschinen-Taylor-Modelle in `InitValuesDGLTM_` auf eine Konstante, da intern der variable Term in die Fehlereinschließung verschoben wird. Die Variable `TransformInitVals` bleibt dann auf `true` und die Transformation wird nach erfolgreich verlaufenem Integrationsschritt mit der berechneten Intervalleinschließungen der Lösung als Anfangswertmenge noch einmal durchgeführt.

Als nächsten Punkt der Vorbereitung definieren wir den Gitterpunkt, an dem die Lösung bestimmt werden soll.

```
20      /*
          Set 't_next_' and check if the step was too large.
22      */

24      t_next_ = t_curr_ + h_curr_;

26      if( t_next_ >= inf(t_end_) )
          {
28          t_next_ = sup(t_end_);

30          /*
              Calculate new step size.
32          */

34          h_curr_ = t_next_ - t_curr_;
          }
```

Die Variable `h_curr_` enthält die aktuelle Schrittweite und wird außerhalb der Zeitschleife mit einem Startwert belegt. Die Variable `t_next_` bezeichnet den nachfolgenden Gitterpunkt.

Der Integrationsbereich ist definiert und wir können als nächstes gemäß (3.18) ein Maschinen-Taylor-Modell für die Zeitvariable erzeugen.

```
36      /*
          Next we generate a Taylor model for the time variable:
38        TimeTM_=[ t_curr_+(t-t_curr_) , [0,0] ],
          t \in [t_curr_,t_next_].
40      */

42      Interval time_interval(t_curr_,t_next_);
```

```
44      if( Repeat_Step )
          {
46          TimeTM_.change_var_data(TimeCode_, t_curr_, time_interval);
          }
48      else
          {
50          TimeTM_ = t_curr_ +
             TaylorModel(
52                      Data_Ptr_->Time_Identifier_,
                        t_curr_,
54                      time_interval
                       );
56
          /*
58          Initialize 'Solution_' and store information about the time
            variable (the subdomain of integration) in it.
60          This will be needed for integration.
          */
62
          Solution_ = InitValsDGLTM_ + 0.0 * TimeTM_;
64      }
```

Muss der vorige Integrationsschritt wiederholt werden, dann setzen wir mit der Operation `change_var_data` den neuen Definitionsbereich der Zeitvariablen und ersparen uns hierdurch die Neuberechnung des Taylorpolynoms. Wir erinnern daran, dass jedes Taylorpolynom den Definitionsbereich seiner Variablen mitführt.

Nachdem dann alle Vorbereitungen abgeschlossen sind, berechnen wir das Taylorpolynom der Lösung über die Picard-Iteration nach Algorithmus 1. Wir beginnen die Iteration mit Polynomordnung 1 und erhöhen diese mit jedem Iterationsschritt um 1. Da im k-ten Iterationsschritt nur die Terme bis zur Ordnung k relevant sind, sparen wir durch diese Vorgehensweise Speicherplatz und Rechenzeit für ansonsten unnötig mitgeführte Terme. Außerdem lassen wir während der Berechnung des Taylorpolynoms die Fehlereinschließungen unberücksichtigt. Wir setzen hierfür das Flag `NO_REMAINDER` (vgl. Abschnitt 4.4.3.2).

```
66      if( not Repeat_Step )
          /*
68          Calculate the Taylor polynomial.
          */
70        {
```

```
        TaylorModel::set_mode( NO_REMAINDER );
72
        for(int k = 1; k <= Data_Ptr_->Order_; k++)
74        {
            TaylorModel::set_order( k );
76
            /*
78            Evaluate the Picard operator.
            */
80
            for(int i = 0; i < Data_Ptr_->Dimension_; i++)
82            {
                currTM_[i] = InitValsDGLTM_[i] +
84                integrate(
                        Data_Ptr_->Func_Ptr_RHS_[i]( Solution_,
                            TimeTM_ ),
86                      TimeCode_
                       );
88            }

90          Solution_ = currTM_;
          }
92      }
      else
94      /*
          The integration step will be repeated. We can reuse
96        the already calculated Taylor polynomial.
        */
98      {
          Solution_ = currTM_;
100     }

102   TaylorModel::set_mode( NORMAL_MODE );
```

Nachdem wir das Taylorpolynom berechnet haben, bestimmen wir eine geeignete Fehlereinschließung mit Algorithmus 2. Dabei ist zu beachten, dass auf der Maschine das Taylorpolynom $\boldsymbol{P}_{n,\boldsymbol{y}^\star}$ vom Operator $\boldsymbol{K}$ im Allgemeinen nicht reproduziert wird. Die Konsequenz daraus ist, dass beim Nachweis der Mengeninklusion (3.20), die nach (2.7) auf

$$\begin{aligned} &[\![\boldsymbol{P}_{n,\boldsymbol{K}(\boldsymbol{y}^\star)}, \boldsymbol{I}_{n,\boldsymbol{K}(\boldsymbol{y}^\star)}]\!] \subseteq [\![\boldsymbol{P}_{n,\boldsymbol{y}^\star}, \boldsymbol{I}_{n,\boldsymbol{y}^\star}]\!] \\ \Leftrightarrow\ &\boldsymbol{P}_{n,\boldsymbol{K}(\boldsymbol{y}^\star)}(\boldsymbol{x}-\boldsymbol{x}_0) - \boldsymbol{P}_{n,\boldsymbol{y}^\star}(\boldsymbol{x}-\boldsymbol{x}_0) + \boldsymbol{I}_{n,\boldsymbol{K}(\boldsymbol{y}^\star)} \subseteq \boldsymbol{I}_{n,\boldsymbol{y}^\star} \quad \forall \boldsymbol{x} \in \boldsymbol{D} \end{aligned}$$

basiert, nun die Differenz der Polynome explizit zu bilden ist. Folglich ist auf der Maschine die Mengeninklusion nach (2.8) über die Inklusion

$$\boldsymbol{I}_{\boldsymbol{P}_{n,\boldsymbol{K}(\boldsymbol{y}^\star)} - \boldsymbol{P}_{n,\boldsymbol{y}^\star}} +_{fl} \boldsymbol{I}_{n,\boldsymbol{K}(\boldsymbol{y}^\star)} \subseteq \boldsymbol{I}_{n,\boldsymbol{y}^\star},$$

mit einer Wertebereichseinschließung $\boldsymbol{I}_{\boldsymbol{P}_{n,\boldsymbol{K}(\boldsymbol{y}^\star)} - \boldsymbol{P}_{n,\boldsymbol{y}^\star}}$ von $\boldsymbol{P}_{n,\boldsymbol{K}(\boldsymbol{y}^\star)} - \boldsymbol{P}_{n,\boldsymbol{y}^\star}$ zu prüfen. Allerdings wird bei praktischen Rechnungen das Polynom $\boldsymbol{P}_{n,\boldsymbol{K}(\boldsymbol{y}^\star)}$ nur wenig von $\boldsymbol{P}_{n,\boldsymbol{y}^\star}$ abweichen, sodass der Durchmesser von $\boldsymbol{I}_{\boldsymbol{P}_{n,\boldsymbol{K}(\boldsymbol{y}^\star)} - \boldsymbol{P}_{n,\boldsymbol{y}^\star}}$ in der Regel sehr klein sein wird. Es folgt die Schleife, in der die Innen- und die Lösungseinschließung berechnet werden.

```
    int  iter_1, iter_1_sum = 0, iter_1_max = 3;
104 bool Inclusion = true;
    bool Decreased = false;
106 TaylorModel help;

108 do {
```

Die Inneneinschließung wird wie das Taylorpolynom komponentenweise berechnet. Wir setzen das Flag `REMAINDER_REC` und veranlassen die interne Aufzeichnung der Berechnungen innerhalb der Taylor-Modell-Operationen (vgl. Abschnitt 4.4.3.1).

```
      /*
110     Calculate the inner approximation.
      */
112   // Create P+[0,0].
      Solution_.replace_interval_part_with( ZERO_IVECTOR );
114   TaylorModel::set_mode( REMAINDER_REC ); // Set Record mode.

116   for(int i = 0; i < Data_Ptr_->Dimension_; i++)
        {
118       /*
            Evaluate the i-th component of the
120         integral operator: K_i( P+[0,0] ) - P_i.
            The calculated inner approximation will be stored
122         in the interval vector 'currIV_'.
          */
124
          help = InitValsDGLTM_[i];
126       help += integrate(
                      Data_Ptr_->Func_Ptr_RHS_[i]( Solution_,
                          TimeTM_ ),
128                   TimeCode_
```

```
                    );
130       help.subtract_polynomial_part_of( Solution_[i] );

132       currIV_[i] = help.eval();
        }
```

Für die Berechnung der Lösungseinschließung setzen wir das Flag `REMAINDER_PLAY`, wodurch von nun an nur noch Fehlereinschließungen berechnet werden. Entsprechend der Bemerkung in Abschnitt 3.4.2 (S. 58) werden bei der komponentenweisen Auswertung des Integraloperators $\boldsymbol{K}$ schon berechnete Fehlereinschließungen im Fall bestehender Inklusion sofort weiterverwendet. Wir begrenzen die Zahl der Iterationen auf `iter_1_max`.

```
134     /*
          Calculate enclosures of the solution.
136     */

138     TaylorModel::set_mode( REMAINDER_PLAY ); //Play mode.

140     iter_1 = 0;
        Inclusion = false;
142
        while( not Inclusion and iter_1 < iter_1_max )
144       {
            ++iter_1;
146
            /*
148           Replace the remainder part with the inflated
              inner approximation I_k.
150         */

152         for(int i = 0; i < Data_Ptr_->Dimension_; i++)
              {
154             inflate( currIV_[i] ); //Epsilon Inflation.

156             Solution_[i].replace_interval_part_with( currIV_[i] );
              }
158
            /*
160           Evaluate the integral operator: K( P + I_k ) - P =: J_k
            */
162
            Inclusion = true;
```

```
164
          for(int i = 0; i < Data_Ptr_->Dimension_; i++)
166         {
              help  = InitValsDGLTM_[i];
168           help += integrate(
                            Data_Ptr_->Func_Ptr_RHS_[i]( Solution_,
                                TimeTM_ ),
170                         TimeCode_
                           );
172           help.subtract_polynomial_part_of( Solution_[i] );

174           currIV_[i] = help.eval();

176           /*
                Check if there is inclusion.
178           */

180           Inclusion &= (currIV_[i]).subset( Solution_[i].
                  interval_part() );

182           /*
                If inclusion holds in the current component
184             use the calculated (better) enclosure in the following
                calculations.
186           */

188           if( Inclusion )
                Solution_[i].replace_interval_part_with( currIV_[i] );
190         }
        }
```

Wird die obige `while`-Schleife beendet, weil `iter_1_max` überschritten wurde, dann war der Integrationsbereich zu groß und die Schrittweite wird verringert. Im Falle einer konstanten Schrittweite steigen wir an dieser Stelle aus dem Algorithmus aus und verlassen das Programm mit einer Meldung an den Benutzer. Im Falle der automatischen Schrittweitensteuerung orientieren wir uns an den Erfahrungswerten von Berz und seinen Mitarbeitern und übernehmen deren Vorgehensweise zur Schrittweitenkorrektur[3].

```
192     /*
          If failure then decrease the step size or in
```

[3] siehe Variable `DECH` in Datei `TM.fox` des COSY-VI Pakets (Stand: 03/22/2004), Zeile 1 211.

```
194           case of constant step size exit the algorithm.
          */

196
          if( not Inclusion )
198         {
              /*
200             No inclusion has been reached, because the time interval
                of the integration step was too big.
202             Decrease the step size in case of automatic step
                size control and exit in case of a constant step size.
204           */

206           switch( Data_Ptr_->Step_cntrl_ )
                {
208             case AUTO:
                  {
210               h_curr_ *= 0.7; // Use the same factor as in COSY.

212               if( h_curr_ < Data_Ptr_->h_min_ )
                    {
214                  std::cout << "DROPOUT: *** "
                           << "Step size undergoes minimal step size."
216                        << "Inclusion is not possible. *** "
                           << std::endl;
218                 std::exit(1); //Quit.
                    }
220
                  Decreased = true;
222
                  t_next_ = t_curr_ + h_curr_;
224
                  /*
226                 Change the data of the time variable.
                  */
228
                  TimeTM_.change_var_data(
230                                       TimeCode_,
                                         t_curr_,
232                                      Interval(t_curr_,t_next_)
                                        );
234
                  break;
236             }
```

```
            case CONST:
              {
               std::cout << "DROPOUT: ***"
                     << "With the given step size"
                     << "inclusion is not possible. *** "
                     << std::endl;
               std::exit(1); //Quit.

               break;
              }
            }
        }

      iter_1_sum += iter_1;

    } while ( not Inclusion );
```

Nachdem eine Lösungseinschließung erfolgreich berechnet wurde, geht es im nächsten Schritt an die iterative Verbesserung derselben (Algorithmus 4). Wir setzen das Flag `REMAINDER_PLAY`, da auch hier nur Fehlereinschließungen berechnet werden. Der Integraloperator wird wie üblich komponentenweise ausgewertet und jede berechnete (bessere) Fehlereinschließung für die nachfolgenden Komponenten weiterverwendet. Es wird solange iteriert, bis die Verbesserung in den Intervallschranken jeder Komponente im Vergleich zur vorigen Iterierten kleiner als 1 % ist. Überprüft wird dies von der Funktion `terminate`.

```
    /*
      Improve the enclosure of the solution.
    */

    TaylorModel::set_mode( REMAINDER_PLAY ); //Play mode.

    int iter_2 = 0;
    Interval better_enc;

    do {

      /*
        Evaluate the integral operator.
      */

      for(int i = 0; i < Data_Ptr_->Dimension_; i++)
        {
```

```
270           //Save remainder interval.
              currIV_[i] = Solution_[i].interval_part();
272
              help = InitValsDGLTM_[i];
274           help += integrate(
                          Data_Ptr_->Func_Ptr_RHS_[i]( Solution_, TimeTM_
                              ),
276                       TimeCode_
                         );
278           help.subtract_polynomial_part_of( Solution_[i] );

280           better_enc = help.eval();

282           /*
                Set new remainder interval.
284           */

286           Solution_[i].replace_interval_part_with( better_enc );
            }
288
          ++iter_2;
290
        } while ( not terminate( currIV_, Solution_.interval_part() ) );
292
        TaylorModel::set_mode( NORMAL_MODE ); //Normal mode.
```

Die Lösung des Anfangswertproblems wurde auf dem zugrundeliegenden Teilintervall kontinuierlich eingeschlossen. Wir bestimmen nun als Nächstes eine Einschließung der Lösungsmenge am rechten Ende des Teilintervalls, indem wir im Taylorpolynom die Zeitvariable durch den entsprechenden Wert substituieren. Die berechnete Einschließung verwenden wir dann zur Abschätzung des lokalen Fehlerzuwachses in diesem Integrationsschritt, den wir nachfolgend definieren.

Wir betrachten den j-ten Zeitschritt. Für den in der i-ten Komponente ($i \in \{1, \ldots, \nu\}$) auftretenden lokalen (Verfahrens-)Fehler gilt für ein $\theta_i \in (0,1)$

$$\epsilon_i^{(j)}(\boldsymbol{\eta}, h_{j-1}) := \frac{y_i^{\star(n+1)}(\boldsymbol{\eta}, t_{j-1} + \theta_i h_{j-1})}{(n+1)!}\, h_{j-1}^{n+1} \in I_{n,y_i^\star}^{(t_j)}$$

für alle Anfangswerte $\boldsymbol{\eta} \in \boldsymbol{I}_{\boldsymbol{\eta}}$. Die Fehlereinschließung $I_{n,y_i^\star}^{(t_j)}$ ist aber mindestens so groß wie $I_{n,y_i^\star}^{(t_{j-1})}$, d. h. es gilt

$$I_{n,y_i^\star}^{(t_{j-1})} \subseteq I_{n,y_i^\star}^{(t_j)}.$$

Wir definieren den Fehlerzuwachs in der i-ten Komponente durch

$$\tilde{\epsilon}_i^{(j)} := \max\left\{\sup(I_{n,y_i^\star}^{(t_j)}) - \sup(I_{n,y_i^\star}^{(t_{j-1})}), \inf(I_{n,y_i^\star}^{(t_{j-1})}) - \inf(I_{n,y_i^\star}^{(t_j)})\right\} \geqslant 0$$

und damit den lokalen Fehlerzuwachs im j-ten Schritt als

$$\tilde{\epsilon}^{(j)} := \max_{i=1,\ldots,\nu} \tilde{\epsilon}_i^{(j)}. \tag{4.8}$$

```
294     /*
          Calculate the solution set at 't_next_' and
296       the local error increase.
        */
298
        Interval grid_point;
300     double local_err = 0.0;

302     if( t_next_ == sup(t_end_) ) grid_point = t_end_;
        else                         grid_point = t_next_;
304
        for(int i = 0; i < Data_Ptr_->Dimension_; i++)
306       {
            help = substitute( Solution_[i], TimeCode_, grid_point );
308
            Interval tmp = abs(
310                           subtract_bounds(
                                          help.interval_part(),
312                                       InitValsDGLTM_[i].
                                            interval_part()
                                      )
314                         );

316         local_err = ( local_err > tmp.sup() ) ? local_err : tmp.sup()
                ;
          }
```

Liegt der lokale Fehlerzuwachs oberhalb einer vorgegebenen Schranke, dann erklären wir die berechnete Einschließung als zu schlecht und wiederholen den Integrationsschritt mit entsprechend korrigierter Schrittweite. Der Schrittweitenkorrektur[4] liegen dabei die folgenden Überlegungen zugrunde. Für die Schrittweite $c \cdot h_{j-1}$, $c \leqslant 1$, gilt

$$\epsilon_i^{(j)}(\boldsymbol{\eta}, c \cdot h_{j-1}) \in c^{n+1} I_{n,y_i^\star}^{(t_j)} \subseteq c^{n+1}\left(I_{n,y_i^\star}^{(t_{j-1})} + [-\tilde{\epsilon}_i^{(j)}, \tilde{\epsilon}_i^{(j)}]\right).$$

[4] vgl. Datei `TM.fox` des COSY-VI Pakets (Stand: 03/22/2004), Zeilen 1292-1299.

Damit nun der Zuwachs $c^{n+1}\tilde{\epsilon}_i^{(j)}$ in jeder Komponente kleiner als eine vorgegebene Schranke $\epsilon_{tol} > 0$ bleibt, muss gelten:

$$c^{n+1} \cdot \tilde{\epsilon}_i^{(j)} \leqslant c^{n+1} \cdot \tilde{\epsilon}^{(j)} \leqslant \epsilon_{tol} \quad \Leftrightarrow \quad c \leqslant \left(\frac{\epsilon_{tol}}{\tilde{\epsilon}^{(j)}}\right)^{\frac{1}{n+1}} .$$

Da es sich hierbei jedoch nur um Schätzungen handelt, berechnen wir c aus

$$c = \left(\frac{\epsilon_{tol}}{10 \cdot \tilde{\epsilon}^{(j)}}\right)^{\frac{1}{n+1}} . \tag{4.9}$$

Der Faktor 10 ist willkürlich gewählt und gibt keine Garantie, dass die Wiederholung des Integrationsschrittes mit der um c reduzierten Schrittweite zum Erfolg führt.

Hat sich der Benutzer für eine konstante Schrittweite entschieden, bleibt der lokale Fehlerzuwachs unberücksichtigt. Die Integration wird mit konstanter Schrittweite fortgesetzt.

```
318     /*
          If automatic step size control was chosen then
320       decide on the local error increase if the time
          step should be repeated and correct the step
322       size due to the local error increase. In case
          of a constant step size the size of the local
324       error increase will be ignored.
        */
326
        switch( Data_Ptr_->Step_cntrl_ )
328       {
          case AUTO:
330         {
            Interval tmp = pow(
332             Interval(Data_Ptr_->Local_error_tol_) / (10*local_err),
                Interval(1.0)/(TaylorModel::order()+1)
334           );

336         double c = tmp.inf(); //Factor for step size correction.

338         if( local_err > Data_Ptr_->Local_error_tol_
                and h_curr_ > Data_Ptr_->h_min_ )
340           /*
              Repeat the time step.
342           */
```

```
            {
344           Repeat_Step = true;
              --step_cntr;
346
              //
348           // Reset 't_next_'.
              //
350           t_next_ = t_curr_;
            }
352       else
            /*
354           The local error increase is smaller than the user
              defined upper bound. The time step need not be
356           repeated.
            */
358         {
              Repeat_Step = false;
360           t_curr_    = t_next_;

362           //
              // Make sure the factor for
364           // step size correction is not too big.
              //
366           if( c > 1.1 ) c = 1.1;
            }
368
          /*
370         Correct the step size and make sure the step size is
            not too small.
372       */

374       h_curr_ *= c;
          if( h_curr_ < Data_Ptr_->h_min_ )
376         {
              h_curr_ = Data_Ptr_->h_min_;
378         }

380       if( c < 1.0 ) Decreased = true; //For output only.

382       break;
        }
384   case CONST:
        {
```

```
386           t_curr_ = t_next_;

388           if( Data_Ptr_->Step_sizes_ > 1 )
                h_curr_ = Data_Ptr_->h_sequel_[ step_cntr ];
390         }
          }
```

Wurde der Integrationsschritt akzeptiert, wenden wir den Shrink-Wrapping-Algorithmus auf die berechnete Lösungseinschließung an und setzen abschließend die Anfangswertmenge für den nächsten Schritt.

```
392
      if( not Repeat_Step )
394     {
          /*
396         Calculate the solution set.
          */
398
          if( t_next_ == sup(t_end_) ) grid_point = t_end_;
400       else                         grid_point = t_next_;

402       for(int i = 0; i < Data_Ptr_->Dimension_; i++)
            {
404           Solution_[i] = substitute( Solution_[i], TimeCode_,
                  grid_point );

406           /*
                'Enclosure' is an interval vector which bounds
408             the solution set, described by the calculated
                Taylor models.
410           */

412           Enclosure[i] = Solution_[i].eval();
            }
414
          /*
416         Use shrink wrapping only at intermediate grid points.
            When shrink wrapping at the end point the
418         enclosure property of the Taylor models for
            fixed initial values is lost.
420       */

422       if( Data_Ptr_->Shrink_Wrapping_ == ON and t_next_ < sup(
```

```
            t_end_) )
          {
424         ShrinkWrapping::Algorithm().run( &Solution_, file );
          }
426
        /*
428       An enclosure of the solution has been calculated.
          If the init values of the current time step are points,
430       then use the intervals in 'Enclosure' for the following
          time step. Otherwise use the calculated representation
432       of the shrink wrapped solution set by Taylor models.
        */
434
        if( TransformInitVals )
436       {
            Data_Ptr_->InitVals_Value_ = Enclosure;
438       }
        else
440       {
            InitValsDGLTM_ = Solution_;
442       }
      }
444
  } //End of while-loop (Time-Loop).
```

4.5.2 Bemerkungen zum Shrink-Wrapping

Shrink-Wrapping wurde gemäß den Algorithmen 5 und 6 implementiert. Wir verzichten hier auf eine Angabe des Quellcodes und gehen stattdessen auf einzelne Stellen der Implementierung ein.

Kommen wir zunächst auf die Implementierung von Algorithmus 5 zu sprechen. Für die Invertierung der Matrix $\boldsymbol{A} \in \mathbb{R}^{\nu \times \nu}$ im zweiten Schritt des Algorithmus greifen wir auf die Funktion `matinv` der C-XSC-Toolbox [22, 61] zurück. Diese berechnet eine Näherungsinverse von $\boldsymbol{A}$ mittels LU-Zerlegung der Matrix und anschließendem Lösen von ν Gleichungssystemen. Den nichtlinearen Anteil $\boldsymbol{q}$ in Schritt drei berechnen wir aus

$$\boldsymbol{q} = \boldsymbol{A}^{-1}\tilde{\boldsymbol{P}} - \boldsymbol{Id},$$

um Abweichungen von der Identität $\boldsymbol{Id}$, resultierend aus $\boldsymbol{A}^{-1}\boldsymbol{A} \approx \boldsymbol{Id}$, mit einzufangen [46, S. 16]. Im vierten Schritt des Algorithmus berechnen wir dann wie in

COSY-VI für jede Komponente $i \in \{1, \ldots, \nu\}$ einen Shrink-Faktor

$$\kappa_i := 1 + \frac{\delta_i}{(1-\sigma)(1-(\nu-1)\tau)}$$

mit $\left(\boldsymbol{A}^{-1}\boldsymbol{I}\right)_i \subseteq \delta_i \cdot [-1, 1]$ und multiplizieren bei der anschließenden Rücktransformation in Schritt fünf jede Komponente mit dem entsprechenden Faktor κ_i. Dies hat gegenüber einem einzigen, für alle Komponenten gleichen Shrink-Faktor κ den Vorteil, dass die Lösungsmenge in den einzelnen Dimensionen unterschiedlich stark erweitert wird, was wiederum die Überschätzung der Einschließung verringert. Für die Rücktransformation benötigen wir eine Einschließung der Inversen von $\boldsymbol{A}^{-1}$. Wir verwenden die Funktion `linsys`, ebenfalls eine Funktion aus der C-XSC-Toolbox, die für ein eindeutig lösbares Gleichungssystem eine Lösungseinschließung berechnet. Analog zur Funktion `matinv` berechnen wir die Einschließung der Inversen spaltenweise durch Lösen von ν Gleichungssystemen.

Zur Implementierung von Algorithmus 6 merken wir an, dass wir die Heuristik von Makino und Berz mit der Modifikation (3.35) in RIOT realisiert haben.

4.5.3 Übersicht

Das Paket DGL wird gebildet von den Paketen TM-Verfahren, Shrink-Wrapping und Hilfsmittel (Abb. 4.19).

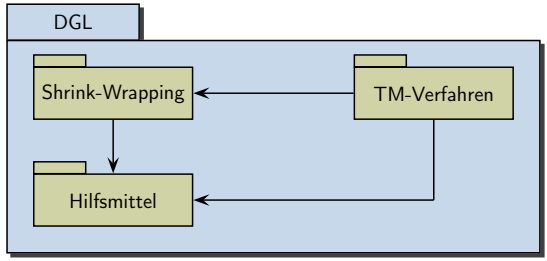

Abbildung 4.19: Gliederung von DGL.

Das Paket Shrink-Wrapping besteht nur aus der Klasse `ShrinkWrapping`, das Paket TM-Verfahren enthält die Klasse `DGLSolver` und die Struktur `Data`.

Das Paket Hilfsmittel umfasst Klassen und Algorithmen zu Matrizen und Vektoren sowie zur Zeitmessung und Verwaltung. Es besteht aus den Klassen `Vector`, `Matrix` für reelle Vektoren und Matrizen, den Klassen `IVector`, `IMatrix` für Intervallvektoren und -matrizen, sowie den Funktionen `linsys` und `matinv`. Hin-

zu kommt die Hilfsklasse `VVTaylorModel` für Vektoren von Maschinen-Taylor-Modellen, die für eine übersichtlichere Gestaltung des Codes eingeführt wurde und die Klasse `Files`, die das Anlegen der Ausgabedateien übernimmt und den Zugriff darauf ermöglicht. Des Weiteren enthält das Paket die Klasse `Stopwatch`, die für die Messung der Laufzeit zuständig ist, sowie die Klasse `TimeIdentifier`, die als Bindeglied zwischen Variablentabelle und Benutzer Informationen zur Zeitvariablen bereitstellt. Sie dient als zentraler Anlaufpunkt für alle Klassen und Strukturen, welche Informationen über die Zeitvariable benötigen.

5 Numerische Beispiele

Der höchste Lohn für unsere Bemühungen ist nicht das, was wir dafür bekommen, sondern das, was wir dadurch werden.

John Ruskin (1819-1900)

In diesem Kapitel wollen wir an Differentialgleichungen aus verschiedenen Anwendungsfeldern der Naturwissenschaften den Nutzen der in dieser Arbeit vorgestellten Methode praktisch überprüfen. Dazu vergleichen wir die Ergebnisse mit dem populären Programm AWA [41]. Außerdem wollen wir das mit dieser Arbeit entstandene Programmpaket RiOT mit dem von Berz und seinen Mitarbeitern entwickelten Programm COSY-VI [46] vergleichen. Wir setzen dazu die Parameter in COSY-VI so, dass ein Vergleich mit RiOT möglich ist, werden aber auch die volle Funktionalität von COSY-VI verwenden, um den Nutzen der Taylor-Modell-Methode aufzuzeigen.

Die numerischen Ergebnisse hängen z. T. stark von der Wahl gewisser Parameter (Ordnung der Taylorpolynome, Fehlertoleranz, Schrittweitenparameter, etc.) ab. Einige Parameter haben wir experimentell so bestimmt, dass ein bezüglich Rechenzeit und Ergebnisgenauigkeit möglichst austariertes Ergebnis erzielt wurde. Wir möchten an dieser Stelle aber betonen, dass wir uns nicht bemüht haben, eine bzgl. Genauigkeit und Rechenzeit optimale Parameterkonstellation zu finden.

Die Berechnungen wurden auf einem 1.83 GHz AMD Sempron 2600+ Prozessor unter Ubuntu Linux 5.10 durchgeführt.

5.1 Van-der-Pol-Gleichung

Die Van-der-Pol-Gleichung stammt ursprünglich aus den Untersuchungen des sogenannten Van-der-Pol-Oszillators, einem nichtlinearen Oszillator, der in den 1920er Jahren von B. van der Pol und J. van der Mark aufgebaut und eingehend untersucht wurde. Er gilt als Prototyp eines selbsterregten Oszillators [52]. Die Differentialgleichung, die das Verhalten des Oszillators beschreibt, lautet [20, S. 112]

$$y'' + \mu(y^2 - 1)y' + y = 0, \quad \mu > 0. \tag{5.1}$$

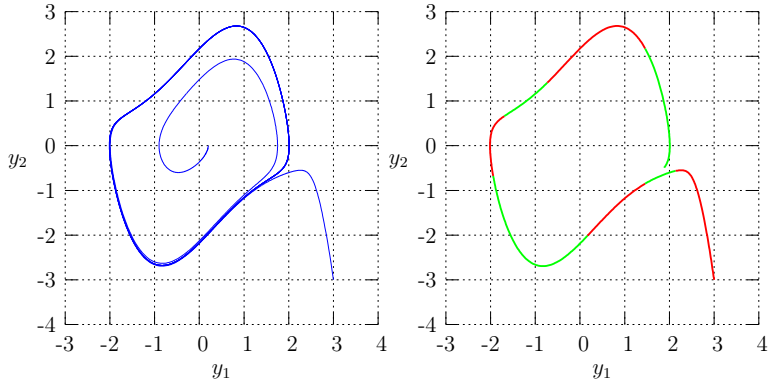

Abbildung 5.1: Lösungen der Van-der-Pol-Gleichung mit $\mu = 1$ (links) und farbliche Unterteilung der Trajektorie nach Zeiteinheiten der Länge 1 für $0 \leqslant t \leqslant 8$ (rechts).

Sie findet heute auch in vielen anderen Fragestellungen wie z. B. in Rad-Fahrbahn-Problemen oder der Modellierung des Herzrhythmus Verwendung [14].

Das Besondere an diesem Modell ist das selbstregulierende Verhalten. Eine Lösung y von (5.1) wird für $|y| > 1$ gedämpft und für $|y| < 1$ verstärkt. Das hat zur Folge, dass sich alle Lösungen bei einer stabilen, periodischen Lösung einschwingen. Man nennt so eine Lösung auch einen Grenzzyklus. Dieser ist für die Van-der-Pol-Gleichung eindeutig und hat in der Phasenebene für kleine Werte von μ nahezu die Gestalt einer Ellipse, wird jedoch mit wachsendem μ zunehmend eckiger. Abbildung 5.1 zeigt für $\mu = 1$, wie sich zwei Lösungen der Differentialgleichung an den Grenzzyklus anschmiegen. Neben der Gestalt hängt auch die Periode des Grenzzyklus von μ ab. Für den Wert $\mu = 1$ hat der Grenzzyklus die Periode $T \approx 6.663\,286\,859$ [20, S. 126].

Als System lautet die Van-der-Pol-Gleichung

$$\boxed{\begin{aligned} y_1' &= y_2, \\ y_2' &= \mu(1 - y_1^2)y_2 - y_1, \quad \mu > 0. \end{aligned}} \tag{5.2}$$

Wir wollen die Lösung dieses Systems für $\mu = 1$ zu den Anfangswerten

$$\begin{aligned} y_1(0) &\in [\ 2.999,\ \ 3.001], \\ y_2(0) &\in [-3.001, -2.999] \end{aligned} \tag{5.3}$$

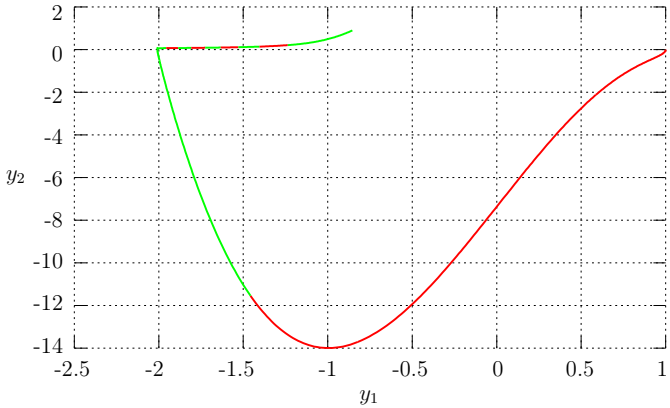

Abbildung 5.2: Farbliche Unterteilung einer Trajektorie nach Zeiteinheiten der Länge 1 für $\mu = 10$ und $0 \leqslant t \leqslant 10$.

berechnen und wählen dazu die Parameter in RiOT wie folgt:

RiOT	t_0	t_e	Order n	Sparsity tolerance	Step size control		
	0	10	10	1E-20	AUTO	$h_0 = 0.1$	$h_{min} = 0.005$

Local error tolerance	Order check	Bounder
1.E-11	TotalDegree	LDB

Abbildung 5.1 zeigt, wie unterschiedlich schnell sich die Lösung auf der Trajektorie entlang bewegt. Da sich Überschätzungen der Lösungsmenge nicht vermeiden lassen, wird diese mit der Zeit immer länger. Je kleiner die Überschätzungen bleiben, desto länger wird sich eine extreme Längung, die ein Intervallverfahren zum Erliegen bringt, hinauszögern. Noch deutlicher wird das beispielsweise für $\mu = 10$. Dort verweilt die Lösungsmenge in gewissen Bereichen recht lange, in anderen Bereichen wird die Trajektorie sehr zügig durchlaufen (vgl. Abbildung 5.2).

Das Ergebnis der Rechnung mit RiOT präsentieren wir in Abbildung 5.3. Wir sehen dort, dass der Durchmesser bei $t \approx 5$ in beiden Komponenten sprunghaft ansteigt.

In der Phasenebene zeigt dies Abbildung 5.4: die noch nahezu punktförmige Einschließung zur Zeit $t = 5.0$ geht in eine deutlich in die Länge gezogene „Gerade“ über ($t = 5.1$); eine weitere sprunghafte Vergrößerung der Lösungseinschließung ist zu sehen beim Übergang von $t = 6.7$ zu $t = 6.8$. Die Ursache dieser Sprünge liegt in dem stark anziehenden Grenzzyklus, der die zu Beginn noch enge Lösungsein-

schließung recht schnell zu einem „Strich“ werden lässt und somit auf eine schlecht konditionierte Matrix $\boldsymbol{A}$ des linearen Anteils führt. In den ersten 27 Integrationsschritten bis ca. $t \approx 1.05$ ist Shrink-Wrapping normal durchführbar, ab dann sind die Werte σ und τ zu groß. Die Konditionszahl von $\boldsymbol{A}$ liegt im 28. Schritt

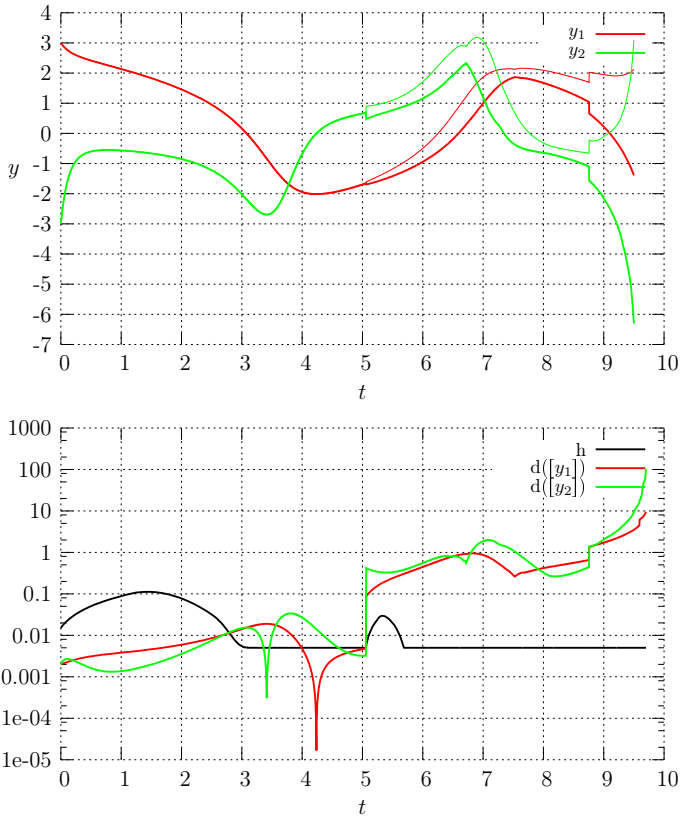

Abbildung 5.3: Einschließung der Lösungskurven zu den Anfangswerten (5.3) dargestellt bis $t = 9.5$ (oben) sowie Schrittweite und Durchmesser der Lösungseinschließungen (unten), berechnet mit RiOT.

bei 1 201 und steigt in den nachfolgenden Schritten weiter an. Shrink-Wrapping kommt erst wieder an der Stelle $t \approx 5.051\,34$ zum Einsatz. Die Fehlereinschließung ist bis dahin auf die Größenordnung 10^{-8} angewachsen, die Konditionszahl liegt bei 1.836E+5. Da nun die Fläche der Fehlereinschließung zu groß geworden ist, wird Shrink-Wrapping wieder durchgeführt und die Lösungsmenge in ein Parallelepiped eingeschlossen. Die Shrink-Faktoren liegen aber aufgrund der schlecht konditionierten Matrix bei $\kappa_1 \approx 17$ für die erste Komponente und $\kappa_2 \approx 130$ für die zweite Komponente. Die ursprüngliche Menge wird also drastisch vergrößert, wie man der Abbildung 5.4 entnehmen kann. Das nächste Mal wird Shrink-Wrapping dann bei $t \approx 6.714\,72$ durchgeführt und die Lösungsmenge dort in ein Intervall eingeschlossen. Die Fläche des einschließenden Parallelepipeds war aufgrund der hohen Konditionszahl von 2.98E+6 mit 7.94E+4 um ein Vielfaches größer ist als die Fläche der Intervalleinschließung. Wie stark die Einschließungen mit der Zeit in die Länge gezogen und geschert werden, ist den letzten Einschließungen in Abbildung 5.5 zu entnehmen.

In einem zweiten Anlauf bauen wir in die Heuristik des Shrink-Wrapping eine Schranke für die Konditionszahl ein. Sobald Shrink-Wrapping nicht durchgeführt werden kann und die Konditionszahl von $\boldsymbol{A}$ oberhalb einer vorgegebenen Schranke liegt, wird die Lösungsmenge in ein Intervall eingeschlossen. Auf diese Weise haben wir immer eine „gut“ konditionierte Matrix vorliegen, gehen dabei aber auch das Risiko ein, mit dieser Methode die Lösungsmenge zu stark aufzublähen. Eine erneute Berechnung mit einer Konditionsschranke von 1 000 brachte ein sehr gutes Ergebnis, wie wir Abbildung 5.6 entnehmen können. Nicht nur die Schrittweite liegt deutlich höher als bei der vorherigen Rechnung, auch die Durchmesser bleiben über den gesamten Integrationsbereich hinweg klein. Dies gilt, obwohl eine Zunahme der Durchmesser zu erkennen ist. Zu sehen ist dies an den Stellen der lokalen Extrema der Funktionen. Hier wechselt die einschließende Menge die Richtung und ist deshalb in einer Komponente deutlich schmäler als in der anderen Komponente. Für beide Funktionen ist an diesen Stellen ein Sprung im Durchmesser erkennbar.

Als nächstes wollen wir dieses Problem auch mit COSY-VI und AWA rechnen. Wir berechnen die Lösung des Problems jeweils mehrmals mit verschiedenen Programmparametern, die wir in den Tabellen durch „/“ voneinander trennen und die wir für die Berechnung, soweit möglich, miteinander kombinieren. Wir wählen die Parameter der beiden Programme wie folgt:

COSY-VI	t_0	t_e	Order n	Step sizes		
	0	10	10	$h_0 = 0.1$	$h_{min} = 0.005$	$h_{max} = 1$

Local error tolerance	Preconditioning	Shrink Wrapping	Weighted Order
1.E-11	None/None/QR	On/Blunting/Blunting	None

AWA	t_0	t_e	Order p	Step size	Enclosure Method
	0	10	10/20	$h_0 = 0.1$	1/3/4

Error tolerances	
ϵ_{abs} = 1.E-16	ϵ_{err} = 1.E-16

Tabelle 5.1 zeigt die berechneten Einschließungen von y_1 und y_2 an der Stelle $t_e = 10$ sowie deren Durchmesser und die benötigte Rechenzeit.

Aufgrund der schlecht konditionierten Matrizen bricht AWA mit Methode 1 (Parallelepiped) und Ordnung 10 die Integration sehr früh bei $t \approx 1.639$ ab. Methode 3 (Durchschnitt von Intervallvektor und Parallelepiped) kommt mit Ordnung 10 auch nur bis $t \approx 6.752$. Zwar schließt RIOT in den allermeisten Fällen des Shrink-Wrapping die Lösungsmenge auch nur in Parallelepipede oder Intervalle ein, doch

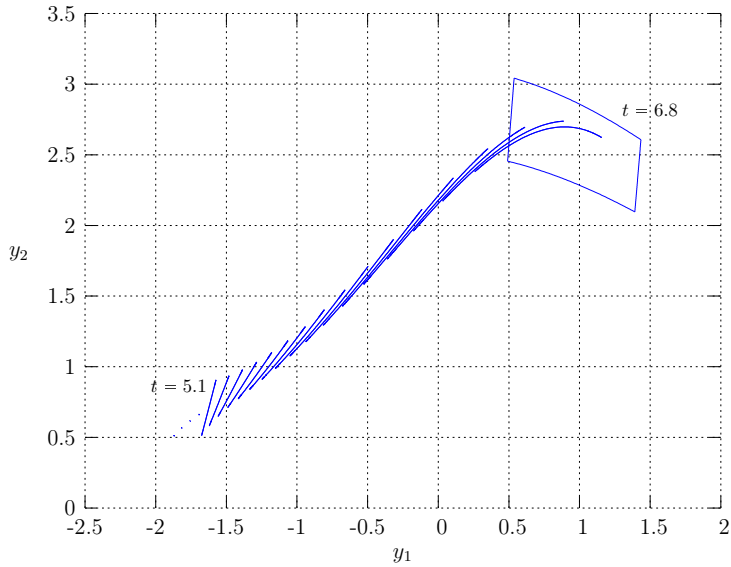

Abbildung 5.4: Einschließungen der Lösungsmenge zu den Zeiten $t = 4.7, 4.8, \ldots, 6.8$ berechnet mit RIOT, wobei die ersten vier Einschließungen dargestellt sind durch „Punkte“ links neben der Einschließung bei $t = 5.1$ („Strich“). Die Fehlereinschließungen liegen unterhalb der gewählten Auflösung und sind deshalb nicht sichtbar.

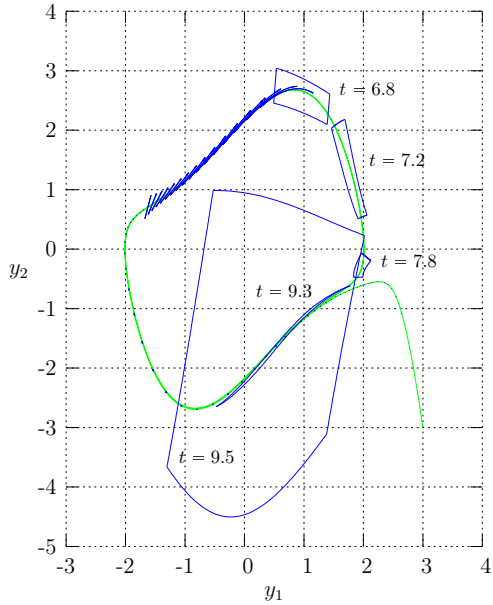

Abbildung 5.5: Einschließungen der Lösungsmenge (blau) entlang der Trajektorie (grün) zu den Zeitpunkten $t = i/10$, $i = 1, \ldots, 68$, $t = 7.2$, $t = 7.8$, $t = 9.3$ und $t = 9.5$ (größte Einschließung) berechnet mit RiOT. Die Fehlereinschließungen liegen unterhalb der gewählten Auflösung und sind deshalb nicht sichtbar.

geschieht dies nicht nach jedem Schritt. Ändern wir die Ordnung auf 20, dann gelingt AWA mit Methode 3 die Integration, jedoch liegt der Durchmesser der Lösungseinschließung von y_2 bei 1.96. Mit Methode 4 (Durchschnitt von QR-Methode und Intervallvektor) schafft AWA die Integration ohne große Mühe und liefert gute Einschließungen.

Verwenden wir COSY-VI ohne Vorkonditionierung und mit Shrink-Wrapping ohne Verwendung von *blunting* (vgl. Abschnitt 3.5.3), dann bricht die Integration bei $t \approx 6.548$ ab, da mit der vorgegebenen Mindestschrittweite keine Inklusion mehr erzielt werden kann. Es ist etwas verwunderlich, dass COSY-VI schon so früh abbricht, denn RiOT entspricht in seinem Leistungsumfang gerade dieser Einstellung von COSY-VI und integriert bis $t \approx 9.7$. Wir werden weiter unten versuchen, darauf

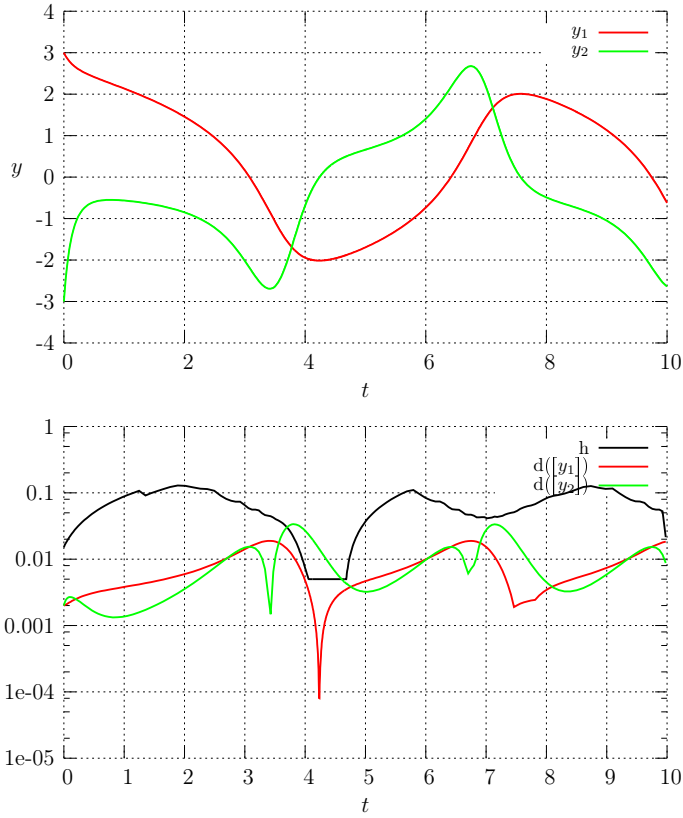

Abbildung 5.6: Einschließung der Lösungskurven zu den Anfangswerten (5.3) (oben) sowie Schrittweite und Durchmesser der Lösungseinschließungen (unten), berechnet mit RiOT unter Verwendung einer Schranke für die Konditionszahl.

eine Antwort zu finden. Verwendet man zusätzlich noch *blunting* zur Verbesserung des Shrink-Wrapping, dann erreicht man zwar t_e, doch sprechen die Ergebnisse nicht gerade für diese Wahl. Ganz anders sieht es aus, wenn man zusätzlich noch die Taylor-Modelle selbst vorkonditioniert (siehe [46, S. 21-28]). Hier stellt COSY-

	AWA	COSY-VI		RIOT
	4/10	None/Blunting	QR/Blunting	
$[y_1]$	$\begin{smallmatrix}-5.983\,616\\-6.288\,454\end{smallmatrix}$E-1	$\begin{smallmatrix}-5.245\,664\\-7.020\,950\end{smallmatrix}$E-1	$-6.\,\begin{smallmatrix}044\,821\\229\,791\end{smallmatrix}$E-1	$-6.\,\begin{smallmatrix}044\,489\\230\,122\end{smallmatrix}$E-1
$[y_2]$	$-2.6\,\begin{smallmatrix}27\,609\\40\,924\end{smallmatrix}$	$-2.\,\begin{smallmatrix}593\,496\\662\,436\end{smallmatrix}$	$-2.63\,\begin{smallmatrix}0\,638\\7\,859\end{smallmatrix}$	$-2.63\,\begin{smallmatrix}0\,625\\7\,879\end{smallmatrix}$
d($[y_1]$)	3.048 369E-2	1.775 285E-1	1.849 685E-2	1.856 318E-2
d($[y_2]$)	1.331 334E-2	6.893 918E-2	7.219 898E-3	7.253 348E-3
Zeit [s]	0.46	20.79	3.64	53.66

Tabelle 5.1: Ergebnisse der Berechnungen zur Van-der-Pol-Gleichung (5.2) mit $\mu = 1$ und den Anfangswerten (5.3) im Vergleich, wobei wir für AWA nur das beste Resultat und für RIOT nur die Ergebnisse aus dem zweiten Versuch mit Verwendung der Schranke für die Konditionszahl angeben. Zum Vergleich: die Rechenzeit bei RIOT ohne die Beschränkung liegt bei 295 Sekunden.

VI mehrere Varianten zur Verfügung, von denen wir aber nur die QR-Methode verwenden wollen. Wir erreichen damit sehr gute Einschließungen, und zwar in einer recht kurzen Rechenzeit.

Die guten Ergebnisse von RIOT, die wir mit der Beschränkung der Konditionszahl erzielt haben, erklären sich dadurch, dass diese Beschränkung sowohl die Taylor-Modelle als auch das Shrink-Wrapping „vorkonditioniert“. Dieses Vorgehen ist natürlich nicht zu vergleichen mit der Vorkonditionierung der Taylor-Modelle, wie sie in COSY-VI vorgenommen wird, jedoch wird das Verhalten durch diesen Eingriff in RIOT, zumindest für dieses Beispiel mit $\mu = 1$, recht gut nachgeahmt. Doch zeigte sich in weiteren Berechnungen mit $\mu = 10$, deren Ergebnisse wir hier aber nicht angeben wollen, die Problematik der Intervalleinschließung bei schlechter Konditionszahl von $\boldsymbol{A}$ sehr deutlich. Da für $\mu = 10$ die Konditionszahl der Matrix noch sehr viel schneller ansteigt als für $\mu = 1$, wird für $\mu = 10$ im Shrink-Wrapping auch sehr viel häufiger eine Intervalleinschließung berechnet, was die Lösungseinschließung sehr stark anwachsen lässt und zum Abbruch der Integration führt.

Zum Abschluss ein Erklärungsversuch, weshalb RIOT weiter integriert als COSY-VI „None/On“ (ohne Vorkonditionierung und ohne *blunting*). Mehrere Faktoren sind dafür verantwortlich. Zum Einen zeigt eine erste Analyse der Ausgaben von RIOT und COSY-VI, dass RIOT größere Schrittweiten wählt als COSY-VI. Führen wir die Berechnung erneut mit den Schrittweiten von COSY-VI „None/On“ durch, dann erhalten wir in jedem Schritt eine engere Fehlereinschließung und damit einen geringeren Fehlerzuwachs (vgl. Abbildung 5.7). RIOT rechnet also genauer und kann deshalb größere Schrittweiten wählen. Zum Zweiten macht sich auch die

Verwendung von (3.35) positiv bemerkbar. Wie Abbildung 5.7 zeigt, haben beide Verfahren zum Zeitpunkt $t \approx 5$ bei automatischer Schrittweitensteuerung ungefähr eine gleich große Fehlereinschließung und einen gleich großen Fehlerzuwachs zu verzeichnen. Shrink-Wrapping schaltet sich ein und schließt bei beiden Verfahren die Lösungsmenge in ein Parallelepiped ein. Im Falle von RiOT ergeben sich Shrink-Faktoren $\kappa_1 \approx 17$ und $\kappa_2 \approx 130$ und die resultierende Parallelepipedfläche hat eine Größe von $2.271\,8 \cdot 10^{-7}$. Im Falle von COSY-VI „None/On“ ergeben sich Shrink-Faktoren mit $\sqrt{\kappa_1 \kappa_2} \approx 219$ und einer Parallelepipedfläche von $4.482\,1 \cdot 10^{-6}$, also eine ungefähr 10 mal so große Fläche als bei RiOT. Dies ist auf die Verwendung von (3.34) in COSY-VI zurückzuführen. Um nun zu belegen, dass die zu große Aufblähung für das deutlich frühere Versagen von COSY-VI „None/On“ verantwortlich ist, haben wir in RiOT die gleiche Berechnung mit (3.34) durchgeführt. Wir erhalten Shrink-Faktoren $\kappa_1 \approx 25$ und $\kappa_2 \approx 192$ und eine resultierende Parallelepipedfläche von $4.874\,0 \cdot 10^{-7}$, also eine doppelt so große Fläche als mit Gleichung (3.35). Diese doppelt so große Fläche führt nun schon bei $t \approx 8.488\,9$ zum Abbruch der Integration. Hieraus dürfen wir vermuten, dass eine 10 mal so große Fläche durchaus zu einem noch früheren Scheitern der Integration führen kann.

Aus Abbildung 5.7 kann man auch entnehmen, dass RiOT mit den Schrittweiten von COSY-VI „None/On“ schon bei $t \approx 6.135\,2$ abbricht, anstatt bei $t \approx 6.548$. Der Grund dafür liegt ebenfalls in den engeren Fehlereinschließungen von RiOT. Diese sind an der Stelle, an der COSY-VI „None/On“ Shrink-Wrapping durchgeführt hat, noch zu klein und die Bedingung für die Durchführung wird nicht erfüllt. Erst später an der Stelle $t \approx 5.948\,1$ schaltet sich Shrink-Wrapping ein. Nur ist bis dahin die Konditionszahl von $\boldsymbol{A}$ noch um eine weitere Zehnerpotenz angestiegen und sorgt zusammen mit der Fehlereinschließung für einen Shrink-Faktor von $\kappa_2 \approx 1\,516$. Die Einschließung wird zu groß, um mit der vorgegebenen Mindestschrittweite die Integration fortführen zu können.

Offenbar hängt der Verlauf wie auch das Ergebnis der Integration mit nicht vorkonditionierten Methoden davon ab, zu welchem Zeitpunkt Shrink-Wrapping durchgeführt wird, d. h. wie groß die Konditionszahl und die Fehlereinschließung zu diesem Zeitpunkt ist. Wir führen die Integration daher erneut mit verschiedenen Fehlertoleranzen durch:

RiOT	t_0	t_e	Order n	Sparsity tolerance	Step size control		
	0	10	10	1.E-20	AUTO	$h_0 = 0.1$	$h_{min} = 0.005$

Local error tolerance	Order check	Bounder
1.E-10/1.E-9/1.E-8/1.E-7	TotalDegree	LDB

COSY-VI	t_0	t_e	Order n	Step sizes		
	0	10	10	$h_0 = 0.1$	$h_{min} = 0.005$	$h_{max} = 1$

Local error tolerance	Preconditioning	Shrink Wrapping	Weighted Order
1.E-10/1.E-9/1.E-8/1.E-7	None	On/Blunting	None

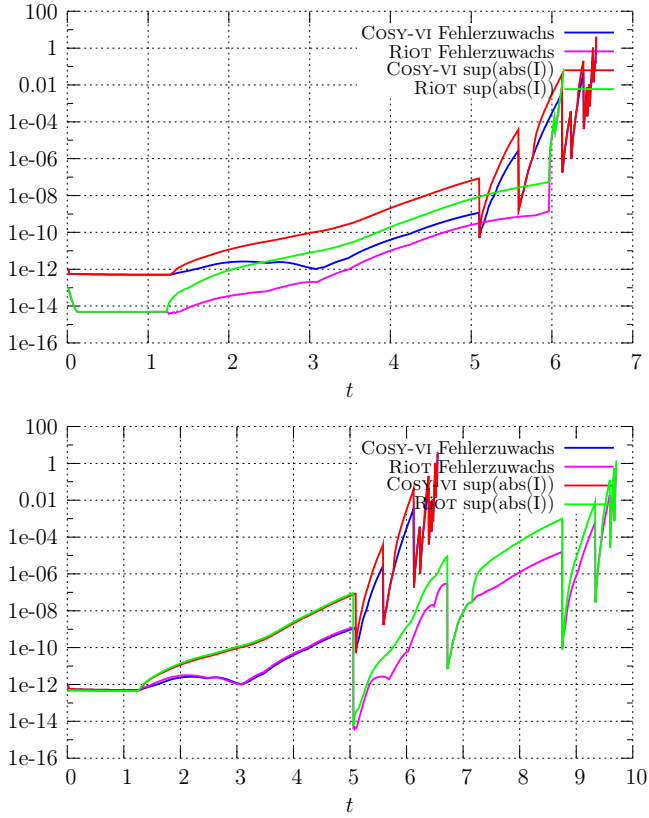

Abbildung 5.7: Betragsmäßige Oberschranke der Fehlereinschließung und lokaler Fehlerzuwachs von COSY-VI und RIOT bei gleichen Schrittweiten (oben) und mit Schrittweitensteuerung (unten).

Das Resultat zeigt Tabelle 5.2. RIOT schafft die Integration erst ab der Toleranzgrenze 1.E-7, während COSY-VI „None/On“ schon ab 1.E-10 die Integration vollständig beenden kann. Parallel dazu wurde auch COSY-VI „QR/Blunting“ mit den angegebenen Toleranzen verwendet. Da sich aber die Einschließungen nicht wesentlich verändert haben, führen wir die Ergebnisse nicht an.

An diesem Beispiel wird deutlich, dass die Kontrolle des Fehlerterms über die Schrittweitensteuerung als auch über das Shrink-Wrapping nur dann sinnvoll möglich ist, wenn die Konditionszahl des linearen Anteils „gut“ ist. Da dieser im wesentlichen für die Qualität des Shrink-Wrapping verantwortlich ist, muss dessen Kondition kontrolliert und gesteuert werden, wie es in COSY-VI „QR/Blunting“ gemacht wird.

	COSY-VI			RIOT
	None/Blunting	None/On		
	1.E-10	1.E-10	1.E-7	1.E-7
$[y_1]$	$-5.998\,294$ $-6.275\,799$E-1	$-6.\,044\,588$ $\,229\,925$E-1	$-5.446\,303$ $-6.828\,486$E-1	$-6\,043\,971$ $230\,585$E-1
$[y_2]$	$-2.6\,28\,730$ $\,39\,573$	$-2.63\,0\,629$ $\,7\,864$	$-2.6\,03\,807$ $\,57\,389$	$-2.63\,0\,607$ $\,7\,900$
d($[y_1]$)	2.775 034E-2	1.853 363E-2	1.382 181E-1	1.866 134E-2
d($[y_2]$)	1.084 186E-2	7.234 406E-3	5.358 023E-2	7.292 402E-3
Zeit [s]	10.59	4.21	2.17	14.69

Tabelle 5.2: Ergebnisse der Berechnungen mit verschiedenen Fehlertoleranzen im Vergleich. Für COSY-VI „None/Blunting“ präsentieren wir nur das beste Resultat. Im Falle von COSY-VI „None/On“ geben wir das beste Ergebnis und das Ergebnis aus der schnellsten Berechnung an.

5.2 Lotka-Volterra-Gleichung

Die Lotka-Volterra-Gleichungen modellieren die Entwicklung zweier Population, die miteinander in einer Räuber-Beute-Beziehung stehen. Es handelt sich dabei um ein einfaches Modell, welches sich auf die folgenden Annahmen stützt: die Beutepopulation y_1 vermehrt sich proportional zu ihrem Bestand, wird aber gleichsam proportional zur Zahl der Begegnungen mit Räubern reduziert, während sich der Bestand der Räuberpopulation y_2 ohne Beute proportional zu seinem Bestand verringert und sich proportional zu den Begegnungen mit Beute vergrößern kann. Die Gleichungen lauten [25, S. 512]

$$\begin{aligned} y_1' &= \alpha_1 y_1 - \beta_1 y_1 y_2, \\ y_2' &= -\alpha_2 y_2 + \beta_2 y_1 y_2, \end{aligned} \qquad \alpha_1, \alpha_2, \beta_1, \beta_2 > 0. \tag{5.4}$$

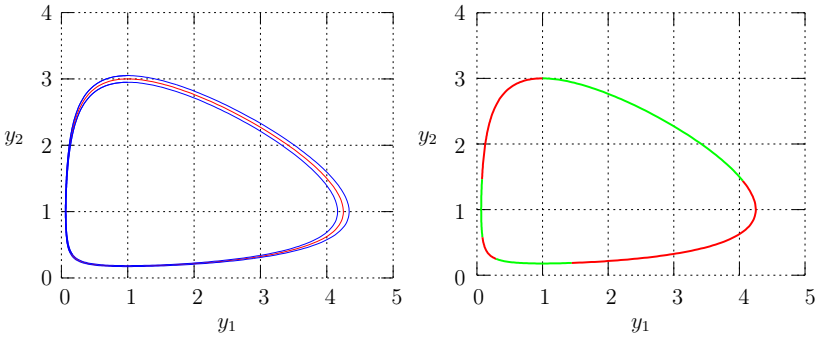

Abbildung 5.8: Lösungen von (5.4) mit $\alpha_1 = \beta_1 = 2$ und $\alpha_2 = \beta_2 = 1$ zu den Anfangswerten $y_1(0) = 1$, $y_2(0) = 3$ (links, rot), $y_1(0) = 1$, $y_2(0) = 3.05$ und $y_1(0) = 1$, $y_2(0) = 2.95$ (links, blau) und farbliche Unterteilung einer Trajektorie nach Zeiteinheiten der Länge 1 für $0 \leqslant t \leqslant T$ (rechts).

Wir setzen hier speziell $\alpha_1 = \beta_1 = 2$ und $\alpha_2 = \beta_2 = 1$ und betrachten für unsere Berechnungen die Anfangswerte

$$\begin{aligned} y_1(0) &\in [0.95, 1.05], \\ y_2(0) &\in [2.95, 3.05]. \end{aligned} \tag{5.5}$$

Die Lösungstrajektorie zu den Punktanfangswerten $y_1(0) = 1$, $y_2(0) = 3$ stellt eine geschlossene Kurve mit Periode $T \approx 5.488\,138\,468\,035$ dar. Wir wollen einen vollen Umlauf dieser Trajektorie mit den oben genannten Intervallanfangswerten berechnen (vgl. [46]). Abbildung 5.8 zeigt die Trajektorie zu drei Anfangswertpaaren aus der Menge in (5.5) und den zeitlichen Verlauf entlang einer Trajektorie.

Wir übernehmen die Programmparameter von COSY-VI und AWA aus [46]. Die Ordnung ist so gewählt, dass die Berechnung eines Zyklus möglich wird, ohne mit sehr kleinen Schrittweiten rechnen zu müssen. Wir setzen:

RIOT	t_0	t_e	Order n	Sparsity tolerance	Step size control		
	0	T	18	1.E-20	AUTO	$h_0 = 0.03$	$h_{min} = 0.003$

Local error tolerance	Order check	Bounder
1.E-11	TotalDegree	LDB

COSY-VI	t_0	t_e	Order n	Step sizes		
	0	T	18	$h_0 = 0.03$	$h_{min} = 0.003$	$h_{max} = 1$

Local error tolerance	Preconditioning	Shrink Wrapping	Weighted Order
1.E-11	None/None/QR	On/Blunting/Blunting	None

AWA	t_0	t_e	Order p	Step size	Enclosure Method
	0	T	12/18	$h_0 = 0.1$	1/3/4

Error tolerances	
ϵ_{abs} = 1.E-12	ϵ_{err} = 1.E-12

Das Ergebnis der Berechnung mit RIOT zeigen die Abbildungen 5.9, 5.10 und 5.11. Zahlenwerte für einen Vergleich der drei Programme zeigt Tabelle 5.3. Sie enthält die Ergebnisse von COSY-VI und RIOT. AWA konnte mit keiner Methode die Integration des Zyklus bewerkstelligen. Auch eine Erhöhung der Ordnung auf 18 schaffte keine Abhilfe, eher im Gegenteil, die Integration wurde damit schon etwas früher abgebrochen. Mit Methode 4 gelingt AWA die Integration bis zur Stelle $t \approx 5.06$.

Verantwortlich für den Abbruch ist die Tatsache, dass AWA die Lösungsmenge nur in Parallelepipede oder Intervalle einschließen kann, denn bei dieser Differentialgleichung haben schon kleine, sich aufsummierende Überschätzungen, verheerende Auswirkungen. Abbildung 5.9 versucht dies zu veranschaulichen. Sie zeigt eine Lösungseinschließung von AWA zur Zeit $t \approx 4.039\,46$ mit zwei Trajektorien, die dieser Menge entspringen, sowie Lösungseinschließungen berechnet mit RIOT zu Zeiten $t > 4.5$. Es wird deutlich, wie schon kleine Überschätzungen im Bereich $0 \leqslant y_1 \leqslant 2.5$, $0 \leqslant y_2 \leqslant 0.5$ zu einer enormen Vergrößerung der Lösungsmenge im Bereich $3.5 \leqslant y_1$ führen. Das Entscheidende ist nun aber, dass gerade in diesem Bereich überwiegend nichtlineare Verzerrungen der Lösungsmenge vorherrschen, wodurch jede weitere Einschließung durch konvexe Mengen unweigerlich zum Abbruch der Integration führt, unabhängig von der gewählten Ordnung.

	COSY-VI			RIOT
	None/On	None/Blunting	QR/Blunting	
$[y_1]$	1.240 265 0.816 719	1.454 922 0.718 909	1.240 265 0.816 719	1.240 280 0.798 904
$[y_2]$	3.045 759 2.935 927	3.077 687 2.873 835	3.045 759 2.935 927	3.054 377 2.933 777
d($[y_1]$)	4.235 454E-1	7.360 120E-1	4.235 454E-1	4.413 752E-1
d($[y_2]$)	1.098 307E-1	2.038 509E-1	1.098 307E-1	1.205 983E-1
Zeit [s]	23.36	42.88	7.17	696.49

Tabelle 5.3: Ergebnisse der Berechnungen im Vergleich. AWA konnte die Integration mit keiner Methode vollenden.

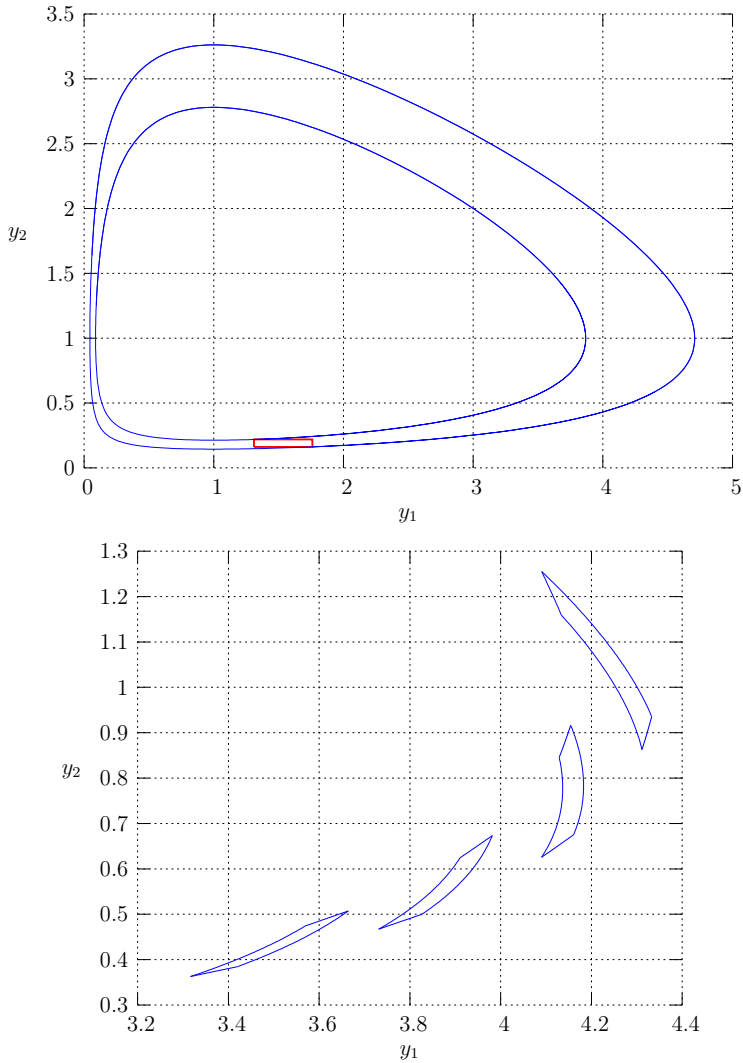

Abbildung 5.9: Trajektorien durch zwei Ecken der Lösungseinschließung an der Stelle $t \approx 4.039\,46$ berechnet mit AWA (oben) und Einschließungen der Lösungsmenge zu den Zeiten $t = 4.6, 4.7, 4.8, 4.9$ (unten, von links nach rechts) berechnet mit RIOT. Die Fehlereinschließungen liegen unterhalb der gewählten Auflösung und sind deshalb nicht sichtbar.

Vergleichen wir die Ergebnisse von COSY-VI und RIOT, dann fallen uns neben der schlechten Rechenzeit von RIOT auch die etwas schlechtere Einschließung von COSY-VI „None/Blunting“ auf. Erklären können wir dies jedoch nicht. Die Konditionszahl des linearen Anteils war in diesem Beispiel durchweg sehr gut und lag unter 50. Interessant ist noch die Wahl der Schrittweiten (vgl. Abbildung 5.10). Bis zur Stelle $t \approx 3.275\,4$ wählen alle Methoden die gleichen Schrittweiten. Erst danach unterscheiden sie sich, wobei COSY-VI “QR/Blunting“ besonders im letzten Teil der Integration, in dem die Nichtlinearitäten in Erscheinung treten, wesentlich größere Schrittweiten verwendet als die anderen Methoden.

5.3 Lorenz-Attraktor

Der Lorenz-Attraktor ist besonders in der Chaosforschung von Bedeutung. Es handelt sich hierbei um die Lösung des folgenden Systems nichtlinearer Differentialgleichungen:

$$\begin{aligned} y_1' &= -\sigma y_1 + \sigma y_2, \\ y_2' &= -y_1 y_3 + r y_1 - y_2, \\ y_3' &= y_1 y_2 - b y_3, \\ \sigma = 10, &\quad b = 1, \quad r = \tfrac{8}{3}, \end{aligned} \tag{5.6}$$

welches 1963 von Edward N. Lorenz aufgestellt wurde. Er wollte damit Aussagen über das Langzeitverhalten von Strömungen in Gasen und Flüssigkeiten gewinnen [38]. Für die angegebenen Parameterwerte für σ, b und r zeigen die Lösungen von (5.6) ein deterministisches chaotisches Verhalten. Darunter versteht man ein irreguläres, chaotisch erscheinendes Verhalten, dessen Dynamik aber von dem zugrundeliegenden System bestimmt ist und nicht durch zufällige äußere Einflüsse (z. B. Rauschen) zustande kommt [52].

Wir wählen die Anfangswerte

$$\begin{aligned} y_1(0) &\in [-8.001, -7.998], \\ y_2(0) &\in [\ 7.998,\ \ 8.001], \\ y_3(0) &\in [\ 26.998,\ 27.001], \end{aligned} \tag{5.7}$$

und setzen die Parameter der Programme wie folgt:

RIOT	t_0	t_e	Order n	Sparsity tolerance	Step size control		
	0	3	10	1.E-20	AUTO	$h_0 = 0.05$	$h_{min} = 0.001$

Local error tolerance	Order check	Bounder
1.E-11/1.E-9/1.E-7	TotalDegree	LDB

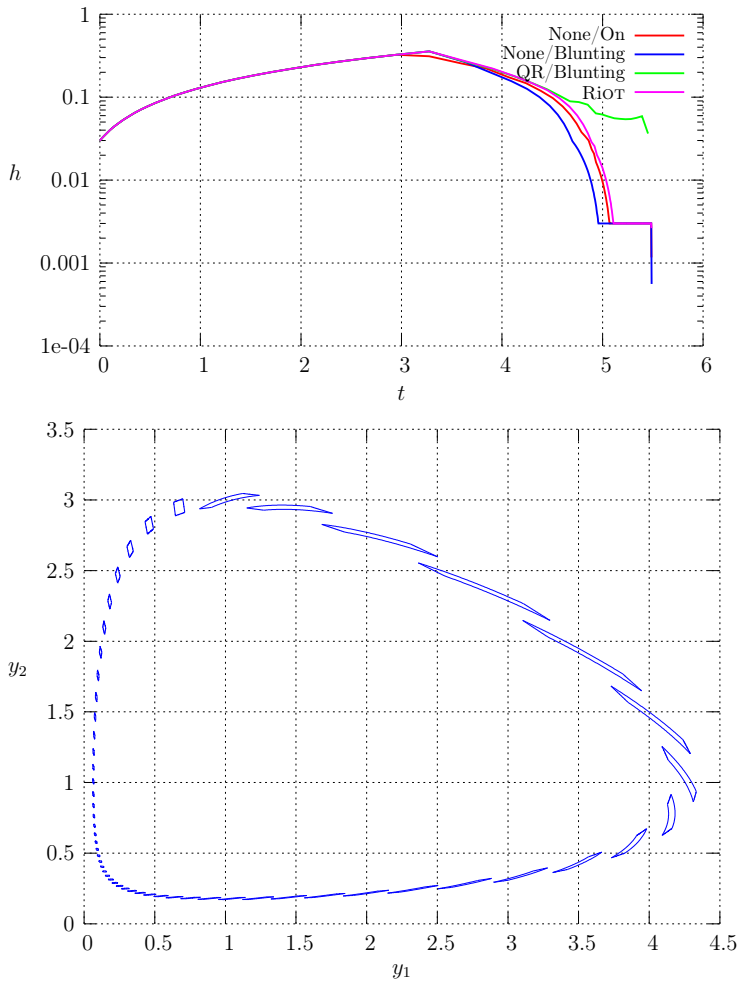

Abbildung 5.10: Die Schrittweiten der Verfahren im Vergleich (oben) und Einschließungen der Lösungsmenge entlang der Trajektorie berechnet mit RiOT (unten). Die Fehlereinschließungen liegen unterhalb der gewählten Auflösung und sind deshalb nicht sichtbar.

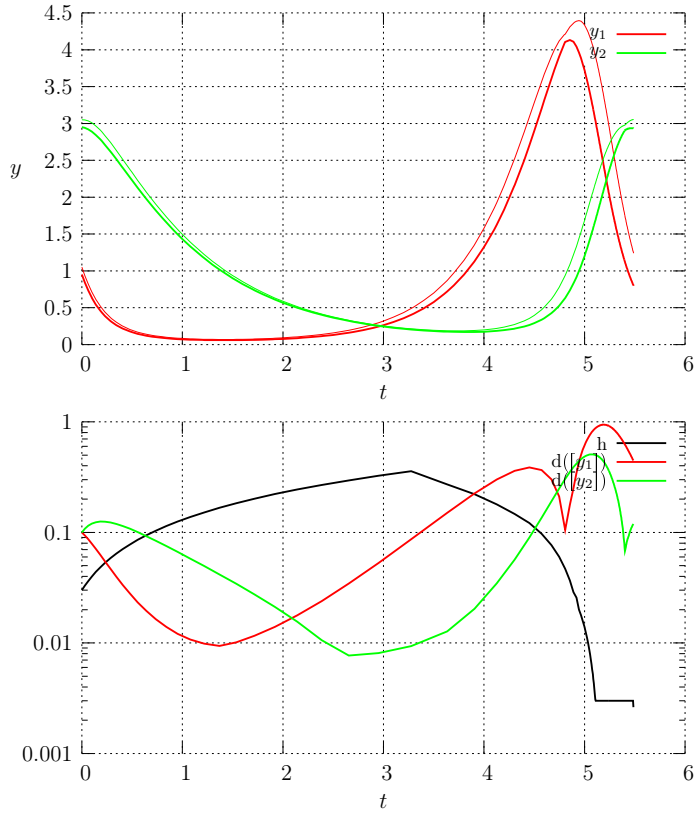

Abbildung 5.11: Einschließung der Lösungskurven zu den Anfangswerten (5.5) (oben) sowie Schrittweite und Durchmesser der Lösungseinschließungen (unten), berechnet mit RiOT.

COSY-VI	t_0	t_e	Order n	Step sizes		
	0	3	10	$h_0 = 0.05$	$h_{min} = 0.001$	$h_{max} = 1$

Local error tolerance	Preconditioning	Shrink Wrapping	Weighted Order
1.E-11/1.E-9/1.E-7	None/None/QR	On/Blunting/Blunting	None

AWA	t_0	t_e	Order p	Step size	Enclosure Method
	0	3	10/20	$h_0 = 0.1$	1/3/4

Error tolerances	
ϵ_{abs} = 1.E-16	ϵ_{err} = 1.E-16

Die Abbildungen 5.12 und 5.13 zeigen das Ergebnis der Integration mit RIOT. Ergebnisse für den Vergleich der Methoden sind in Tabelle 5.4 aufgeführt.

AWA gelingt mit Methode 1 die Integration bis $t \approx 0.57$ und mit Methode 3 immerhin bis $t \approx 1.96$. Erst Methode 4 bringt den ersehnten Erfolg. Es ist daher überraschend, dass RIOT mit den Toleranzen 1.E-11 und 1.E-9 die Integration erfolgreich durchführt und im Falle der Toleranz 1.E-11 auch noch ein Ergebnis liefert, das mit dem von AWA gleichzieht und besser ist als das Ergebnis von COSY-VI „None/On“ und „None/Blunting“.

Das Versagen von AWA mit Methode 1 und 3 im Vergleich zu den Taylor-Modell basierten Methoden lässt sich wie folgt begründen. Die Konditionszahlen von $\boldsymbol{A}$ liegen schon nach wenigen Schritten über 1 000 und nehmen sehr große Werte an. In den auf Taylor-Modellen basierenden Methoden bewirkt dies innerhalb des Shrink-Wrapping eine Verstärkung der nichtlinearen Terme, was sehr früh zum Aussetzen von Shrink-Wrapping führt. Die Fläche des vom linearen Anteil $\boldsymbol{A}$ aufgespannten Parallelepipeds ist sehr klein, doch bleibt auch die Fehlereinschließung lange Zeit klein, sodass während der Integration nur an sehr wenigen Stellen die Lösungsmenge in ein Intervall oder Parallelepiped eingeschlossen wird. In allen anderen Fällen kann die Darstellung der Lösungsmenge durch Polynome verwendet werden. Dieser Vorteil der Taylor-Modelle gegenüber AWA mit Methode 1 und 3 macht sich hier deutlich bemerkbar. Denn so kann sich die schlechte Konditionszahl nicht in jedem Schritt negativ auf die Einschließung auswirken, sondern wie in diesem Fall nur wenige Male.

Was bei diesem Beispiel auch deutlich wird ist, dass die Variation der Fehlertoleranz in COSY-VI und RIOT zu deutlich unterschiedlichen Ergebnissen führt, ausgenommen der Methode COSY-VI „QR/Blunting“. Sie weist gegenüber Veränderungen in den Toleranzen ein stabiles Verhalten auf und liefert gleichbleibend gute Einschließungen. Das kann auf die Vorkonditionierung der Taylor-Modelle zurückgeführt werden, denn ein Vorkonditionieren des Shrink-Wrapping allein bringt kein stabiles Verhalten, wie COSY-VI „None/Blunting“ zeigt. Rechnet man das Anfangswertproblem mit der Fehlertoleranz 1.E-8, erhält man bei COSY-VI „None/On“ und „None/Blunting“ Durchmesser in den Einschließungen größer eins. RIOT muss in diesem Fall die Integration bei $t \approx 1.9$ abbrechen.

	AWA	COSY-VI			RIOT
		None/On	None/Blunting	QR/Blunting	
	4/10	1.E-11	1.E-11	1.E-11	1.E-11
$[y_1]$	$\begin{matrix}-7.780\,253\\-8.352\,018\end{matrix}$E-1	$\begin{matrix}-7.645\,588\\-8.442\,697\end{matrix}$E-1	$\begin{matrix}-0.054\,046\\-1.475\,680\end{matrix}$	$\begin{matrix}-7.928\,041\\-8.160\,224\end{matrix}$E-1	$\begin{matrix}-7.785\,478\\-8.349\,144\end{matrix}$E-1
$[y_2]$	$-2.\,\begin{matrix}084\,474\\912\,699\end{matrix}$E-1	$\begin{matrix}-1.916\,456\\-3.023\,880\end{matrix}$E-1	$\begin{matrix}0.719\,785\\-1.065\,562\end{matrix}$	$-2.\,\begin{matrix}321\,196\\620\,030\end{matrix}$E-1	$-2.\,\begin{matrix}117\,617\\881\,471\end{matrix}$E-1
$[y_3]$	$1.9\,\begin{matrix}25\,738\\19\,608\end{matrix}$E+1	$1.9\,\begin{matrix}29\,640\\15\,530\end{matrix}$E+1	$1.\,\begin{matrix}974\,675\\885\,177\end{matrix}$E+1	$1.92\,\begin{matrix}3\,430\\1\,690\end{matrix}$E+1	$1.9\,\begin{matrix}27\,413\\17\,987\end{matrix}$E+1
$\mathrm{d}([y_1])$	5.717 648E-2	7.971 081E-2	1.421 633	2.321 820E-2	5.636 653E-2
$\mathrm{d}([y_2])$	8.282 238E-2	1.107 422E-1	1.785 347	2.988 328E-2	7.638 527E-2
$\mathrm{d}([y_3])$	6.128 616E-2	1.410 877E-1	8.949 629E-1	1.738 517E-2	9.424 673E-2
Zeit [s]	1.39	52.54	184.34	26.73	1638.26
		1.E-9	1.E-9	1.E-9	1.E-9
$[y_1]$		$\begin{matrix}-0.468\,767\\-1.148\,406\end{matrix}$	$\begin{matrix}-0.580\,310\\-1.019\,379\end{matrix}$	$\begin{matrix}-7.928\,174\\-8.160\,092\end{matrix}$E-1	$\begin{matrix}-0.132\,343\\-1.531\,286\end{matrix}$
$[y_2]$		$\begin{matrix}3.240\,442\\-8.452\,145\end{matrix}$E-1	$\begin{matrix}0.335\,304\\-5.235\,434\end{matrix}$E-1	$-2.\,\begin{matrix}321\,361\\619\,866\end{matrix}$E-1	$\begin{matrix}0.751\,316\\-1.289\,991\end{matrix}$
$[y_3]$		$1.\,\begin{matrix}988\,247\\857\,762\end{matrix}$E+1	$1.9\,\begin{matrix}37\,087\\09\,227\end{matrix}$E+1	$1.92\,\begin{matrix}3\,472\\1\,648\end{matrix}$E+1	$\begin{matrix}2.014\,470\\1.843\,598\end{matrix}$E+1
$\mathrm{d}([y_1])$		6.796 375E-1	4.390 677E-1	2.319 168E-2	1.398 941
$\mathrm{d}([y_2])$		1.169 258	5.570 738E-1	2.985 037E-2	2.041 305
$\mathrm{d}([y_3])$		1.304 837	2.780 871E-1	1.822 317E-2	1.708 708
Zeit [s]		44.84	106.59	25.8	1397.36
		1.E-7		1.E-7	
$[y_1]$		$\begin{matrix}-0.386\,234\\-1.190\,872\end{matrix}$		$\begin{matrix}-7.928\,158\\-8.160\,109\end{matrix}$E-1	
$[y_2]$		$\begin{matrix}2.936\,956\\-7.246\,710\end{matrix}$E-1		$-2.\,\begin{matrix}321\,341\\619\,885\end{matrix}$E-1	
$[y_3]$		$1.\,\begin{matrix}959\,512\\885\,734\end{matrix}$E+1		$1.92\,\begin{matrix}3\,466\\1\,654\end{matrix}$E+1	
$\mathrm{d}([y_1])$		8.046 375E-1		2.319 498E-2	
$\mathrm{d}([y_2])$		1.018 366		2.985 429E-2	
$\mathrm{d}([y_3])$		7.377 693E-1		1.811 206E-2	
Zeit [s]		25.69		27.71	

Tabelle 5.4: Ergebnisse der Berechnungen im Vergleich, wobei für AWA nur das beste Ergebnis aufgeführt ist.

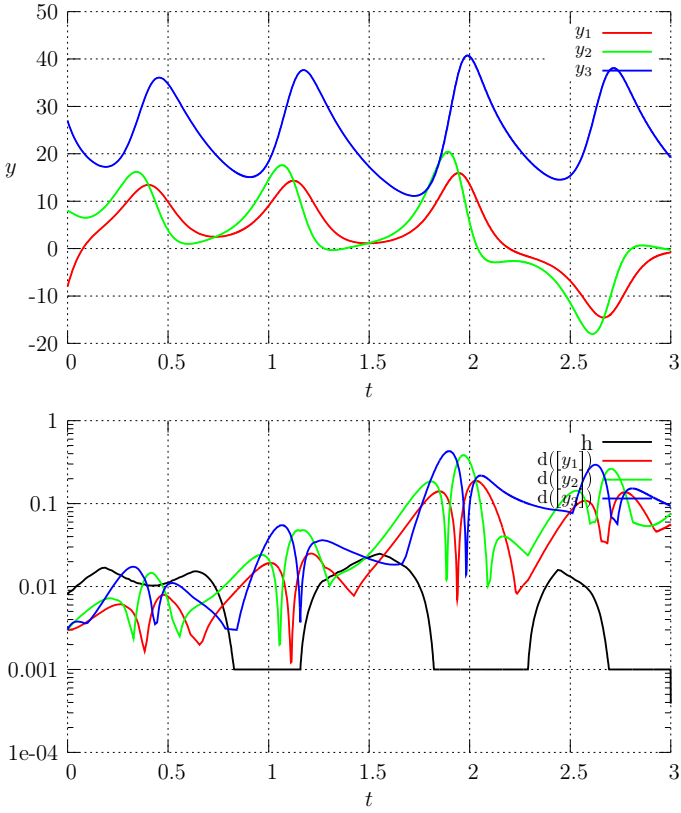

Abbildung 5.12: Einschließung der Lösungskurven zu den Anfangswerten (5.7) (oben) sowie Schrittweite und Durchmesser der Lösungseinschließungen (unten), berechnet mit RiOT „1.E-11“.

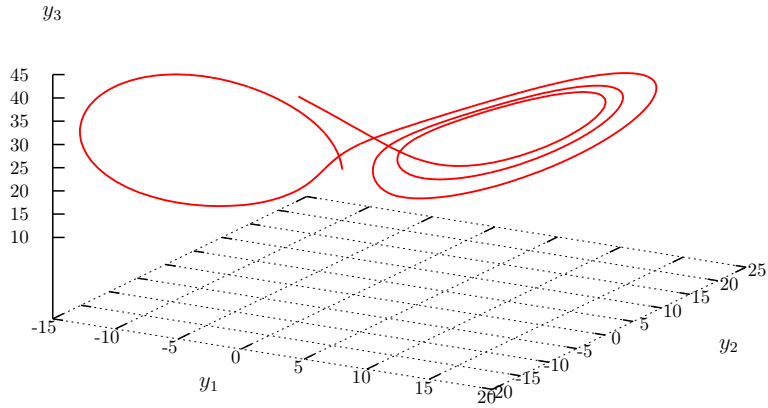

Abbildung 5.13: Phasenplot des Lorenz-Attraktors zu den Anfangswerten (5.7) bis $t = 3$ berechnet mit RiOT „1.E-11“.

Integrieren wir über die Grenze $t_e = 3$ hinaus, zeigen sich deutliche Unterschiede bei AWA und COSY-VI. AWA muss die Integration schon bei $t \approx 3.66$ abbrechen, da die Einschließungen zu explodieren beginnen. COSY-VI „QR/Blunting“ schafft es mit Toleranz 1.E-11 immerhin bis $t \approx 6.94$, bevor auch dort die Einschließungen über alle Grenzen wachsen. Auch hier macht sich der Unterschied in der Darstellung der Lösung bemerkbar.

5.4 Chemische Reaktionskinetik

Die Reaktionskinetik als Teildisziplin der physikalischen Chemie befasst sich unter anderem mit der mathematischen Beschreibung chemischer Reaktionen auf der Grundlage des Massenwirkungsgesetzes. Betrachtet wird nicht das Verhalten einzelner Moleküle, sondern die Konzentrationen der beteiligten Stoffe. Zentral in diesem Zusammenhang ist der Begriff der Reaktionsgeschwindigkeit. Er gibt

an, wie schnell die Ausgangsstoffe eines im Ungleichgewicht befindlichen Systems dem chemischen Gleichgewichtszustand zustreben. Zur Vereinfachung der Modelle werden die Rahmenbedingungen Druck, Volumen und Temperatur als konstant angenommen [52].

Wir betrachten m Teilreaktionen von n Substanzen A_i

$$\sum_{i=1}^{n} \alpha_{ij} A_i \xrightarrow{k_j} \sum_{i=1}^{n} \gamma_{ij} A_i, \quad j = 1, \ldots, m.$$

Die Werte k_j bezeichnen jeweils die Reaktionsgeschwindigkeit der j-ten Teilreaktion, die nicht negativen, ganzen Zahlen α_{ij} und γ_{ij} jeweils den Anteil der Substanz A_i an der j-ten Teilreaktion. Dann ergibt sich für die Konzentration $y_i(t)$ der Substanz A_i die Differentialgleichung

$$y_i'(t) = \sum_{j=1}^{m} (\gamma_{ij} - \alpha_{ij})\, k_j \prod_{l=1}^{n} \big(y_l(t)\big)^{\alpha_{lj}}, \quad i = 1, \ldots, n\,, \tag{5.8}$$

mit polynomialer rechter Seite.

Nun gibt es innerhalb einer Reaktion langsame und schnell ablaufende Teilreaktionen. Diese rapide Änderung der Stoffkonzentrationen schlägt sich in steifen Differentialgleichungen nieder. Wir wollen im folgenden ein solches System mit RiOT, Cosy-vi und Awa rechnen und betrachten die Reaktion

$$\begin{aligned} A &\xrightarrow{0.04} B \\ B + B &\xrightarrow{3\cdot 10^7} C + B \\ B + C &\xrightarrow{10^4} A + C \end{aligned} \tag{5.9}$$

der Stoffe A, B und C [21, S. 3]. Beschreibt $y_1(t)$ die Konzentration von A, $y_2(t)$ die Konzentration von B und $y_3(t)$ die Konzentration von C, dann werden gemäß (5.8) die Reaktionsgleichungen (5.9) durch das Differentialgleichungssystem

$$\boxed{\begin{aligned} y_1' &= -0.04 y_1 + 10^4 y_2 y_3, \\ y_2' &= 0.04 y_1 - 3 \cdot 10^7 y_2^2 - 10^4 y_2 y_3, \\ y_3' &= 3 \cdot 10^7 y_2^2 \end{aligned}} \tag{5.10}$$

modelliert. Wir gehen für unsere Berechnung von den Anfangswerten

$$\begin{aligned} y_1(0) &\in [0.999\,95, 1.000\,05], \\ y_2(0) &\in [0, 0.000\,000\,01], \\ y_3(0) &\in [0, 0.000\,000\,01] \end{aligned} \tag{5.11}$$

aus und wählen die Parameter der Programme wie folgt:

RiOT	t_0	t_e	Order n	Sparsity tolerance	Step size control		
	0	1	10	1.E-20	AUTO	$h_0 = 0.1$	$h_{min} = 0.0001$

Local error tolerance	Order check	Bounder
1.E-11/1.E-9/1.E-7	TotalDegree	LDB

COSY-VI	t_0	t_e	Order n	Step sizes		
	0	1	10	$h_0 = 0.1$	$h_{min} = 0.0001$	$h_{max} = 1$

Local error tolerance	Preconditioning	Shrink Wrapping	Weighted Order
1.E-11/1.E-9/1.E-7	None/None/QR	On/Blunting/Blunting	None

AWA	t_0	t_e	Order p	Step size	Enclosure Method
	0	1	10/20	$h_0 = 0.1$	3/4

Error tolerances	
ϵ_{abs} = 1.E-16	ϵ_{err} = 1.E-16

Die Ergebnisse verändern sich mit der Variation der Fehlertoleranzen und Ordnungen nur unwesentlich. Tabelle 5.5 zeigt die besten erzielten Einschließungen mit jeweils dazugehöriger Toleranz. Das einzig Nennenswerte dabei ist, dass RiOT auch hier besser abschneidet als COSY-VI „None/On“. Einziger Wermutstropfen ist und bleibt die schlechte Laufzeit.

Der Anstieg des Durchmessers in y_3 (vgl. Abbildung 5.14) ist problembedingt. Nach (5.9) wird der Stoff C aus A erzeugt, wobei B dabei als Zwischenprodukt entsteht und sehr schnell wieder zu A und C umgesetzt wird. Dies erklärt, weshalb die Konzentration von A fällt und die von C steigt, die Konzentration von B aber klein bleibt. Die Unsicherheiten in den Anfangswerten sorgen nun für das rapide Anwachsen der Einschließung von y_3, wobei sich der Durchmesser allerdings zu stabilisieren scheint.

Während der Berechnungen zeigen die auf Taylor-Modellen basierenden Verfahren das für steife Differentialgleichungen charakteristische Verhalten expliziter Verfahren (vgl. Abbildung 5.15). Die Lösung verhält sich völlig harmlos, trotzdem zeigt das numerische Verfahren große Instabilität in der Schrittweitenwahl. Wählt man die Mindestschrittweite in COSY-VI und RiOT oberhalb der kleinsten verwendeten Schrittweite muss das Verfahren abbrechen, da mit Erreichen der Mindestschrittweite keine Lösungseinschließung mehr erzielt werden kann. AWA dagegen verhält sich bzgl. der Schrittweiten sehr stabil. Ein „flattern“ wie in Abbildung 5.15 liegt erstaunlicherweise nicht vor.

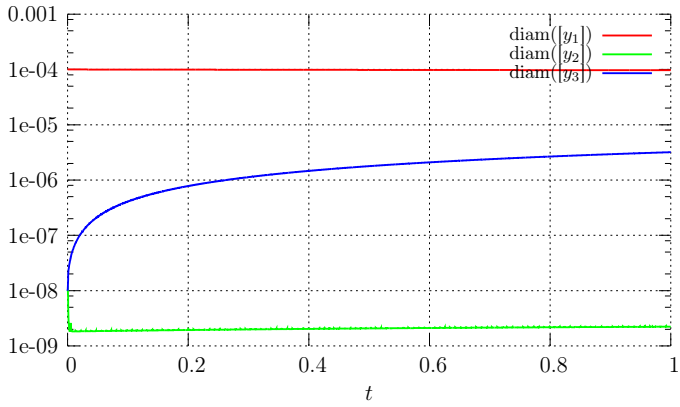

Abbildung 5.14: Durchmesser der Einschließungen zu den Anfangswerten (5.11) berechnet mit RiOT.

	AWA	COSY-VI			RIOT
		None/On	None/Blunting	QR/Blunting	
	4/10	1.E-7	1.E-11	1.E-11	1.E-11
$[y_1]$	$9.66\,\frac{5\,082}{4\,112}$E-1	$9.66\,\frac{5\,192}{4\,003}$E-1	$9.66\,\frac{5\,083}{4\,112}$E-1	$9.66\,\frac{5\,082}{4\,112}$E-1	$9.66\,\frac{5\,082}{4\,112}$E-1
$[y_2]$	$3.074\,\frac{694}{559}$E-5	$3.074\,\frac{763}{490}$E-5	$3.074\,\frac{695}{558}$E-5	$3.074\,\frac{694}{559}$E-5	$3.074\,\frac{738}{515}$E-5
$[y_3]$	$3.35\,\frac{1\,110}{0\,795}$E-2	$3.35\,\frac{1\,148}{0\,756}$E-2	$3.35\,\frac{1\,118}{0\,787}$E-1	$3.35\,\frac{1\,110}{0\,795}$E-2	$3.35\,\frac{1\,112}{0\,792}$E-2
d($[y_1]$)	9.688 205E-5	1.187 290E-4	9.695 129E-5	9.688 286E-5	9.689 507E-5
d($[y_2]$)	1.339 585E-9	2.725 914E-9	1.360 885E-9	1.339 638E-9	2.221 435E-9
d($[y_3]$)	3.136 998E-6	3.916 508E-6	3.294 264E-6	3.136 764E-6	3.191 946E-6
Zeit [s]	17.34	74.01	90.48	140.65	1518.77

Tabelle 5.5: Ergebnisse der Berechnungen im Vergleich.

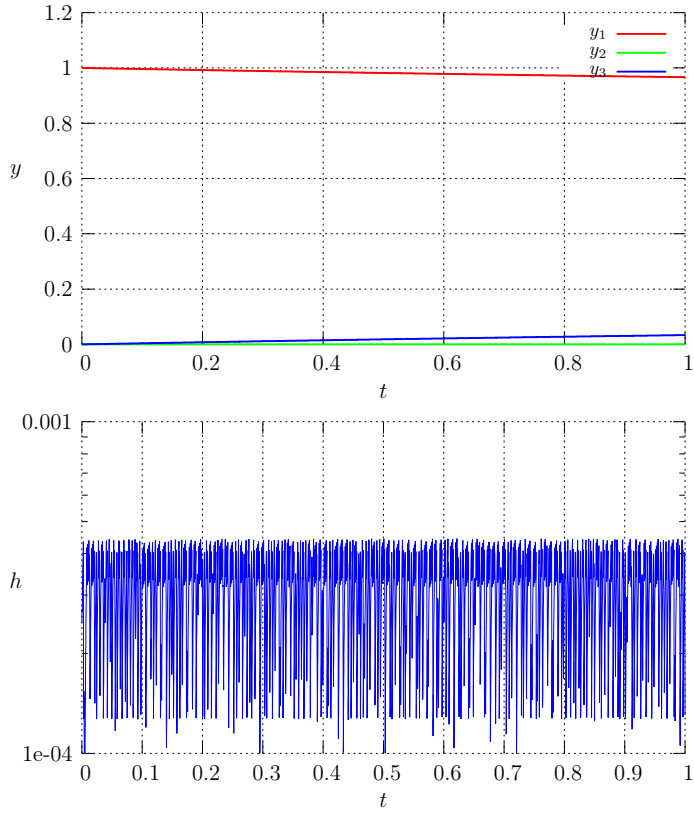

Abbildung 5.15: Einschließung der Lösungskurven zu den Anfangswerten (5.11) (oben) sowie die dazu gewählte Schrittweite (unten), berechnet mit RiOT.

5.5 Himmelsmechanik

Die Himmelsmechanik ist ein Teilgebiet der Astronomie und beschäftigt sich mit der Bewegungsbeschreibung von Himmelskörpern unter dem alleinigen Einfluss der Massenanziehung anderer Körper. Hierbei handelt es sich streng genommen immer um ein Mehrkörperproblem, da die Bewegung eines Himmelskörpers um die Sonne immer auch von allen anderen Planeten beeinflusst wird. Die Kraft $\vec{F}$, die N Massen m_i auf einen Körper der Masse m ausüben, ergibt sich aus der Superposition der Kräfte, die jede einzelne der Massen auf m ausübt, zu

$$\vec{F} = \sum_{i=1}^{N} \vec{F}_i = \sum_{i=1}^{N} G \frac{m_i\, m}{\|\vec{r}_i\|^3}\, \vec{r}_i.$$

Dabei ist $\vec{r}_i$ der Vektor, der m mit m_i verbindet, und G die Gravitationskonstante. In den allermeisten Fällen vernachlässigt man z. B. bei Bewegungen von Kometen oder Planeten um die Sonne die störende Wirkung der restlichen Planeten, da deren Massen im Verhältnis zur Sonne klein und deren gegenseitigen Entfernungen groß sind [38].

Wir betrachten hier speziell die Bewegung eines Asteroiden um die Sonne ohne Berücksichtigung der Wirkung anderer Planeten. Diese Art von Problem, bei dem die Bewegung von nur zwei Massen unter dem Einfluss der Gravitationskraft betrachtet wird, heißt auch Kepler-Problem. In unserem Fall ist die Masse des Asteroiden um ein Vielfaches kleiner als die Masse der Sonne, weshalb wir davon ausgehen, dass die Sonne an ihrer Position verharrt und sich der Asteroid um die Sonne bewegt. Die Bahn, die der Asteroid unter dem Einfluss der Sonne durchläuft, wird beschrieben durch die Differentialgleichung

$$\vec{r}\,'' = -\gamma \frac{\vec{r}}{\left(r_1^2 + r_2^2 + r_3^2\right)^{3/2}} \tag{5.12}$$

für die Position $\vec{r}(t) = \big(r_1(t), r_2(t), r_3(t)\big)$ des Asteroiden und der Sonne im Koordinatenursprung. Die Konstante $\gamma = 0.998\,6$ ergibt sich aus dem Produkt der Gravitationskonstanten G mit der Masse der Sonne. Die Einheiten sind in diesem Beispiel für Distanzen Astronomische Einheiten (AE), wobei eine Astronomische Einheit dem durchschnittlichen Abstand von Erde und Sonne entspricht. Für die Zeit wurde eine Skalierung gewählt, in dem 2π Zeiteinheiten (ZE) einem Erdenjahr entsprechen [29].

Wir erhalten mit $y_1 := r_1$, $y_2 := r_2$, $y_3 := r_3$, $y_4 := r_1'$, $y_5 := r_2'$, $y_6 := r_3'$ das zu

(5.12) äquivalente System erster Ordnung

$$\begin{aligned} y_1' &= y_4, \\ y_2' &= y_5, \\ y_3' &= y_6, \\ y_4' &= -\gamma \frac{y_1}{d}, \\ y_5' &= -\gamma \frac{y_2}{d}, \\ y_6' &= -\gamma \frac{y_3}{d}, \\ d &:= \left(y_1^2 + y_2^2 + y_3^2\right)^{3/2}. \end{aligned} \tag{5.13}$$

Wir wollen dieses System mit Anfangswerten

$$\begin{aligned} y_1(0) &\in -1.772\,690\,981\,915\,12 && \pm 0.5 \cdot 10^{-7} \text{ AE}, \\ y_2(0) &\in 0.148\,721\,485\,234\,2955 && \pm 0.5 \cdot 10^{-7} \text{ AE}, \\ y_3(0) &\in -0.079\,283\,504\,622\,441\,94 && \pm 0.5 \cdot 10^{-7} \text{ AE}, \\ y_4(0) &\in 0.237\,203\,191\,651\,623\,7 && \pm 0.5 \cdot 10^{-6} \text{ AE/ZE}, \\ y_5(0) &\in -0.612\,524\,538\,758\,628 && \pm 0.5 \cdot 10^{-6} \text{ AE/ZE}, \\ y_6(0) &\in 0.045\,832\,175\,721\,656\,24 && \pm 0.5 \cdot 10^{-6} \text{ AE/ZE} \end{aligned} \tag{5.14}$$

rechnen, die der gemessenen Position und Geschwindigkeit des Asteroiden 1997 XF11 am 17. Januar 1997 entsprechen. Dabei ist der Durchmesser der Intervalle hinreichend um alle Meßungenauigkeiten zu erfassen [29]. Uns interessiert hier besonders die Zuverlässigkeit der Methoden über lange Integrationsbereiche hinweg. Wie in Hoefkens et al. [29] versuchen wir die Berechnung einer Lösungseinschließung über 23 Jahre, was 46π ZE entspricht. Wir wählen die Eingabeparameter für RiOT, COSY-VI und AWA wie folgt:

RiOT	t_0	t_e	Order n	Sparsity tolerance	Step size control		
	0	46π	10	1.E-15/1.E-20	AUTO	$h_0 = 0.1$	$h_{min} = 0.001$

Local error tolerance	Order check	Bounder
1.E-11	TotalDegree	LDB

COSY-VI	t_0	t_e	Order n	Step sizes		
	0	46π	10	$h_0 = 0.1$	$h_{min} = 0.001$	$h_{max} = 1$

Local error tolerance	Preconditioning	Shrink Wrapping	Weighted Order
1.E-11/1.E-9	None/QR	On/Blunting	None

Awa	t_0	t_e	Order p	Step size	Enclosure Method
	0	46π	18	$h_0 = 0.0001$	1/3/4

Error tolerances	
ϵ_{abs} = 1.E-16	ϵ_{err} = 1.E-16

Wegen der langen Rechenzeiten verwenden wir in RiOT hier eine etwas größere Toleranzgrenze für die Koeffizienten der Polynome, belassen dafür aber die Fehlertoleranz bei 1.E-11. Interessant wird sein, wie sich dies auf die Laufzeit und auf die Einschließungen auswirkt.

Die recht lange Integration konnte nur von Cosy-vi „QR/Blunting“ über das volle Zeitintervall bewältigt werden. Alle anderen Verfahren steigen früher aus. Awa integriert mit Methode 4 bis $t \approx 46.754\,9$ ZE am weitesten. Das entspricht ungefähr 7.44 Jahren. Mit Methode 1 ist schon nach ca. 3.72 Jahren Schluss und mit Methode 3 gelingt die Integration bis ca. 4.52 Jahre. Die Intervalleinschließungen werden in allen Fällen so groß, dass das Programm durch Null dividiert und abbrechen muss. Das bedeutet aber auch, dass die Taylor-Modell-Methode ohne Vorkonditionierung die gesamte Integration ebenfalls nicht bewältigen kann. Deshalb und wegen der sehr langen Rechenzeiten von RiOT führen wir die Integration nur bis $t \approx 34.557\,5$ ZE durch, was ca. 5.5 Jahren entspricht. Der Zeitbedarf ist zwar immer noch hoch, doch bleibt die Rechenzeit für unsere Testungen akzeptabel.

Tabelle 5.6 zeigt das Resultat der Integration von Cosy-vi „QR/Blunting/1.E-11“ über die vollen 23 Jahre. Die Durchmesser der Einschließungen sind, bezogen auf die Durchmesser der Anfangswertmenge, um maximal drei Zehnerpotenzen angestiegen und das ist, wenn man den langen Integrationsbereich wie auch die Schwere des Problems berücksichtigt, ein sehr gutes Ergebnis (siehe auch [29]). Dass es sich hierbei um eine schwieriges Problem handelt, zeigt unter anderem das Verhalten von Awa.

Cosy-vi					
QR/Blunting/1.E-11					
$[y_1]$	$[y_2]$	$[y_3]$	$[y_4]$	$[y_5]$	$[y_6]$
$-9.0\,\frac{29\,650}{39\,513}$E-1	$-7.49\,\frac{4\,944}{9\,257}$E-1	$8.6\,\frac{91\,249}{24\,310}$E-3	$9.2\,\frac{33\,148}{27\,239}$E-1	$-3.9\,\frac{66\,078}{71\,035}$E-1	$6.02\,\frac{7\,035}{6\,357}$E-2
d($[y_1]$)	d($[y_2]$)	d($[y_3]$)	d($[y_4]$)	d($[y_5]$)	d($[y_6]$)
9.861 620E-4	4.312 183E-4	6.693 830E-5	5.908 668E-4	4.956 302E-4	6.776 288E-6
Zeit [s]	16 316				

Tabelle 5.6: Ergebnis von Cosy-vi „QR/Blunting“ nach 23 Jahren.

Tabelle 5.7 zeigt die Ergebnisse an der Stelle $t \approx 17.278\,7$ ZE (entspricht ca. 2.75 Jahren) und das Ergebnis nach 5.5 Jahren. Sie enthält jeweils das mit den vorgegeben Toleranzen erzielte beste Resultat. Für einen Vergleich sei aber dennoch erwähnt, dass COSY-VI „None/On“ mit der Fehlertoleranz 1.E-9 das Problem in nur 1 541.8 Sekunden integriert, sich dies jedoch in einer teilweise erheblichen Verschlechterung der Einschließungsdurchmesser bemerkbar macht (Faktor 2-6). Bei COSY-VI „QR/Blunting“ hingegen macht sich der Wechsel in den Fehlertoleranzen zumindest bis 2.75 Jahre nicht so deutlich bemerkbar. Mit der Fehlertoleranz 1.E-9 ist die Methode ca. 8 Minuten langsamer und es zeigt sich eine geringfügige Verschlechterung in den Einschließungsdurchmessern (Faktor 1.05-1.2). Nach 5.5 Jahren erhalten wir nach einer ungefähr 1.5 mal so langen Rechnung eine Verschlechterung in den Durchmessern (je nach Komponente) um einen Faktor zwischen 45 und 70. Die längeren Rechenzeiten bei höherer Fehlertoleranzgrenze entstehen, weil aufgrund einer zu optimistischen Schrittweitenwahl mehrere Schritte wiederholt werden müssen.

Der Grund für das gute Abschneiden der Taylor-Modell-Methode liegt in der zunehmenden Bedeutung des nichtlinearen Anteils in der Darstellung der Lösungsmenge. Betrachten wir die Gestalt des Taylorpolynoms der ersten Komponente y_1 über die Integration hinweg, dann sind nach einem Jahr nur Terme bis zur Ordnung 2 vorhanden. Die Koeffizienten der linearen Terme sind dabei von der Größenordnung 10^{-6} bis 10^{-8}, die Koeffizienten der Terme zweiter Ordnung sind von der Größenordnung 10^{-11} bis 10^{-14}. Nach 5 Jahren hat der Einfluss der nichtlinearen Terme deutlich zugenommen. Die Terme zweiter Ordnung sind bis auf 10^{-7} angewachsen und auch Terme dritter Ordnung mit Koeffizienten im Bereich 10^{-11} bis 10^{-15} treten nun im Polynom auf. Eine Vernachlässigung dieser Terme in jedem Schritt führt zu signifikanten Überschätzungen der Lösungsmenge, was ein Anwachsen der Durchmesser und ein Abbruch der Integration zur Folge hat, wie wir es an der Integration mit AWA gesehen haben.

Wie schon in den vorigen Beispielen liefert RIOT ein weiteres Mal bessere Ergebnisse als COSY-VI „None/On“ und das auch noch mit beiden Toleranzgrenzen. Das spricht für die Detailverbesserungen, die wir an dem Verfahren vorgenommen haben. Leider schafft COSY-VI „None/On“ die Integration wegen Speicherüberlauf auf dem verwendeten Rechner nicht bis $t \approx 34.557\,5$ ZE, sodass uns ein weiterer Vergleich fehlt. Die Erhöhung der Toleranzschwelle für die Polynomkoeffizienten auf 1.E-15 reduzierte die Rechenzeit gegenüber der niedrigeren Schwelle deutlich, ohne dabei ein wesentliche Verschlechterung in den Einschließungen zu verursachen.

Die Abbildungen 5.16 und 5.17 veranschaulichen die Ergebnisse aus der Integration mit RIOT. Die Sprünge in den Durchmessern (vgl. Abbildung 5.17, unten) zu den Zeiten $t \approx 17.26$ und $t \approx 26.1$ rühren von Parallelepipedeinschließungen der

	Awa	Cosy-vi		Riot	
		None/On	QR/Blunting		
	4	1.E-11	1.E-11	1.E-15	1.E-20
Jahre	2.75	2.75	2.75	2.75	2.75
$[y_1]$	$-5.67\,0\,980\,1\,439$E-1	$-5.67\,0\,240\,2\,180$E-1	$-5.67\,0\,966\,1\,453$E-1	$-5.67\,0\,845\,1\,575$E-1	$-5.671\,006\,413$E-1
$[y_2]$	$1.838\,7\,40\,32$	$1.838\,7\,47\,24$	$1.838\,73\,\frac{9}{2}$	$1.838\,7\,46\,25$	$1.838\,73\,\frac{9}{3}$
$[y_3]$	$-1.318\,2\,13\,63$E-1	$-1.318\,134\,343$E-1	$-1.318\,2\,14\,63$E-1	$-1.318\,2\,11\,66$E-1	$-1.318\,2\,16\,61$E-1
$[y_4]$	$-5.867\,5\,03\,52$E-1	$-5.867\,444\,612$E-1	$-5.867\,5\,06\,50$E-1	$-5.867\,5\,08\,48$E-1	$-5.867\,5\,11\,45$E-1
$[y_5]$	$4.99\,8\,817\,6\,838$E-2	$5.001\,823\,4.993\,833$E-2	$4.99\,8\,851\,6\,810$E-2	$4.99\,8\,696\,6\,959$E-2	$4.99\,8\,686\,6\,969$E-2
$[y_6]$	$-2.628\,649\,789$E-2	$-2.628\,441\,997$E-2	$-2.628\,648\,790$E-1	$-2.628\,660\,778$E-2	$-2.628\,660\,778$E-2
d($[y_1]$)	4.581 125E-5	1.938 527E-4	4.856 704E-5	7.295 694E-5	4.063 959E-5
d($[y_2]$)	6.982 230E-6	2.204 949E-5	6.316 541E-6	1.974 290E-5	4.990 423E-6
d($[y_3]$)	4.919 242E-6	2.081 581E-5	4.863 480E-6	5.460 399E-6	4.448 172E-6
d($[y_4]$)	4.791 893E-6	1.670 766E-5	4.245 541E-6	3.907 514E-6	3.361 500E-6
d($[y_5]$)	1.977 308E-5	7.989 099E-5	2.040 883E-5	1.735 772E-5	1.716 141E-5
d($[y_6]$)	1.393 621E-6	5.554 181E-6	1.414 084E-6	1.177 793E-6	1.168 779E-6
Zeit [s]	3.01	2454.26	1702.76	5457.82	34336.20
	4		1.E-11	1.E-15	1.E-20
Jahre	5.5		5.5	5.5	5.5
$[y_1]$	$-9.1\,83\,529\,92\,449$E-1		$-9.18\,7\,245\,8\,739$E-1	$-9.\,139\,402\,236\,494$E-1	$-9.\,144\,781\,231\,145$E-1
$[y_2]$	$-7.4\,27\,962\,32\,376$E-1		$-7.4\,29\,834\,30\,500$E-1	$-7.4\,09\,598\,50\,619$E-1	$-7.4\,11\,222\,49\,016$E-1
$[y_3]$	$7.6\,80\,225\,20\,470$E-3		$7.6\,55\,686\,44\,973$E-3	$7.\,989\,522\,310\,627$E-3	$7.\,961\,098\,339\,076$E-3
$[y_4]$	$9.13\,9\,898\,4\,469$E-1		$9.13\,7\,633\,6\,730$E-1	$9.1\,64\,971\,09\,395$E-1	$9.1\,61\,645\,12\,709$E-1
$[y_5]$	$-4.04\,2\,339\,7\,165$E-1		$-4.04\,4\,378\,5\,129$E-1	$-4.0\,23\,144\,66\,174$E-1	$-4.0\,25\,455\,63\,907$E-1
$[y_6]$	$6.03\,5\,479\,4\,566$E-2		$6.03\,5\,094\,4\,952$E-2	$6.03\,1\,915\,8\,019$E-2	$6.03\,7\,988\,1\,965$E-2
d($[y_1]$)	8.918 362E-4		1.492 804E-4	9.709 145E-3	8.636 351E-3
d($[y_2]$)	4.412 134E-4		6.645 066E-5	4.102 046E-3	3.779 277E-3
d($[y_3]$)	5.975 463E-5		1.071 215E-5	6.788 934E-4	6.220 211E-4
d($[y_4]$)	5.427 170E-4		9.014 262E-5	5.557 502E-3	4.893 542E-3
d($[y_5]$)	4.824 878E-4		7.501 268E-5	4.302 904E-3	3.845 040E-3
d($[y_6]$)	9.119 810E-6		1.413 859E-6	6.103 263E-5	6.021 407E-5
Zeit [s]	5.26		3 427.04	27 941.79	111 640.48

Tabelle 5.7: Ergebnisse der Berechnungen im Vergleich.

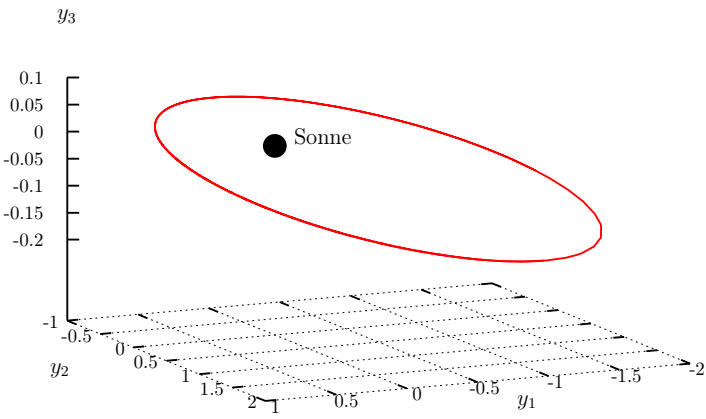

Abbildung 5.16: Asteroidenbahn um die Sonne zu den Anfangswerten (5.14) berechnet mit RiOT.

Lösungsmenge her. Näher an der Sonne befindliche Körper bewegen sich schneller als weiter entfernt liegende Körper, d. h. die anfängliche Intervallanfangswertmenge wird im Laufe der Integration mehr und mehr geschert und gekrümmt. Die Scherung verursacht eine mit der Zeit immer schlechter werdende Konditionszahl des linearen Anteils $\boldsymbol{A}$. Die Krümmung lässt die nichtlinearen Terme bzgl. der Darstellung der anwachsenden Lösungsmenge immer wichtiger werden. Überwiegen diese zu stark bzw. ist die Konditionszahl zu hoch, kann Shrink-Wrapping nicht mehr „normal“ vollzogen werden und es bleibt im Bedarfsfall nur noch die Einschließung in Intervalle oder Parallelepipede. An der Ersten der genannten Stellen ist der Shrink-Faktor der zweiten Komponente mit $\kappa_2 \approx 4.12$ fast viermal größer als der jeweilige Faktor in den anderen Komponenten. Deswegen ist der Sprung des Durchmessers in y_2 auch fast viermal größer als in den anderen Komponenten. Ähnliches kann auch an der zweiten Stelle beobachtet werden.

5.6 Zusammenfassung

Die gerechneten Beispiele haben deutlich gemacht, wie wichtig eine formgerechte Darstellung der Lösungsmenge während der Integration nichtlinearer Differentialgleichungen ist. Doch hat sich auch gezeigt, wie schnell dieser Vorteil der Taylor-

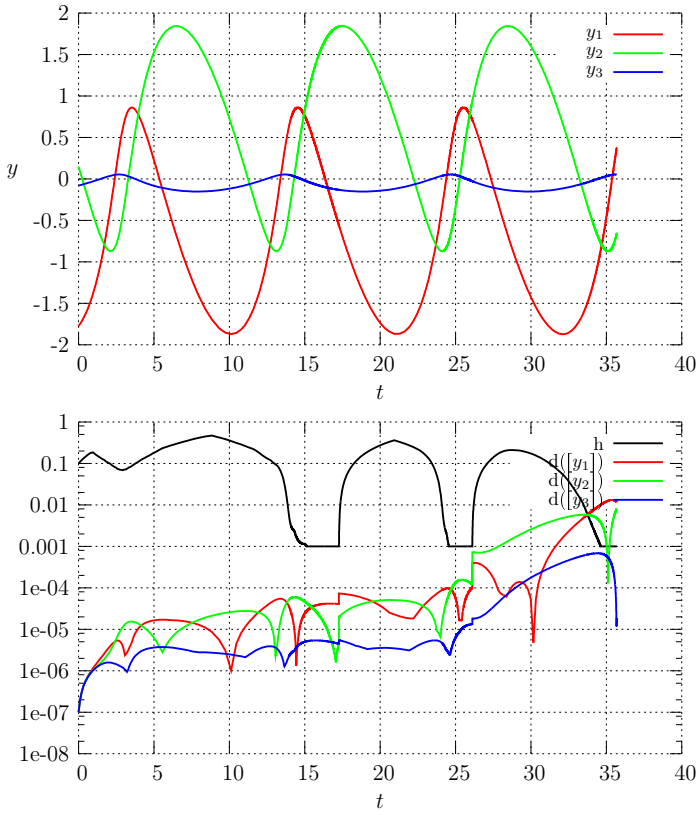

Abbildung 5.17: Einschließung der Ortskurven zu den Anfangswerten (5.14) (oben) sowie der Durchmesser der Einschließungen und die gewählte Schrittweite (unten), berechnet mit RiOT „1.E-15/1.E-11“.

Modell-Methode durch einen schlecht konditionierten linearen Anteil $\boldsymbol{A}$ über das Shrink-Wrapping wieder zunichte gemacht wird. Erst durch entsprechende Vorkonditionierung der Taylor-Modelle kann das Verfahren seine Stärke voll ausschöpfen.

Ist die Anfangswertmenge „groß“ bzw. wird lange genug integriert, um eine entsprechend große Einschließung während der Integration zu erhalten, dann zeigt die Verwendung nichtlinearer Terme zur Lösungsdarstellung ihre volle Wirkung. Deutlich wurde dies unter anderem bei der Berechnung der Asteroidenbahn. Dort sind wir mit einer „kleinen“ Menge gestartet und haben über sehr lange Zeit integriert. Aber auch bei der Lotka-Volterra-Gleichung war sehr schön die Bedeutung der formgerechten Darstellung der Lösungsmenge in den Zwischenschritten zu sehen. Schon kleinste Überschätzungen haben dort die Einschließungen von AWA durch Intervalle und Parallelepipede „explodieren“ lassen. Das Beispiel aus der chemischen Reaktionskinetik hat gezeigt, dass auch das Taylor-Modell-Verfahren bei steifen Differentialgleichungen das gleiche Verhalten wie jedes andere explizite Verfahren aufweist.

In der Reduktion des Wrapping-Effekts nimmt das Shrink-Wrapping eine sehr wichtige Rolle ein. Doch Shrink-Wrapping allein führt nicht immer zum Erfolg, wie man an der Van-der-Pol-Gleichung und dem Lorenz-Attraktor sehen konnte. Sobald die Konditionszahl zu schlecht wird, kommt Shrink-Wrapping an seine Grenzen. Zwar kann die Integration über die Heuristik von Berz und seinen Mitarbeitern noch einige Zeit aufrecht erhalten werden, doch nach einigen Intervall- bzw. Parallelepipedeinschließungen kann die Menge so groß geworden sein, dass ein Fortsetzen der Integration nicht mehr möglich ist. Die alleinige Vorkonditionierung des linearen Anteils im Shrink-Wrapping hilft hier nicht immer weiter, wie die Ergebnisse der „None/Blunting“ Variante von COSY-VI gezeigt haben. Erst die zusätzliche Vorkonditionierung der Taylor-Modelle führt die Integration zum Erfolg. Bei allen gerechneten Beispielen zeichnet sich die „QR/Blunting“ Variante durch konstant gute Einschließungen und „gute“ Rechenzeiten aus. Zwar dauern die Berechnungen länger als bei AWA, doch gelingt der Taylor-Modell-Methode die Integration auch mit großen Anfangswertmengen bzw. über sehr große Zeitintervalle hinweg.

RIOT berechnete in fast allen Beispielen engere Einschließungen als COSY-VI „None/On“. Wir möchten im Folgenden die Unterschiede zwischen RIOT und COSY-VI „None/On“ auflisten:

Datenstruktur: RIOT verwendet zur Darstellung der dünnbesetzten Polynome Hash-Tabellen, während COSY-VI hierfür die Speicherung in Arrays mit spezieller Indizierung bevorzugt [4, 45].

Grundarithmetik: Die Integration von Taylor-Modellen wird in RIOT nach (2.39) vorgenommen. COSY-VI verschenkt an dieser Stelle etwas an Genauigkeit,

wie schon in der Bemerkung auf Seite 37 erläutert wurde. Außerdem ist RiOT mit dem Record-Feature ausgestattet, das bei mehrmaliger Auswertung eines Ausdrucks mit gleichem Taylorpolynom aber unterschiedlichen Fehlereinschließungen die ausschließliche Berechnung der Fehlereinschließung ermöglicht. Das Record-Feature wird zur Einsparung von Rechenzeit bei der Berechnung einer Lösungseinschließung und bei deren Verbesserung eingesetzt. Ob COSY-VI so etwas ähnliches auch macht ist nicht bekannt. Des Weiteren greift RiOT auf eine externe Intervallbibliothek zurück und verwendet diese überall dort im Programm, wo Fehler erfasst werden müssen. COSY-VI dagegen bringt eine eigene Implementierung der Intervallarithmetik mit, wobei die Rundungsfehlererfassung in den Grundoperationen sowie auch die Wertebereichseinschließung bei Polynomen über sogenannte *tallying*-Variablen erfolgt [45]. Dieser Verzicht auf Intervallarithmetik macht sich in kürzeren Rechenzeiten bemerkbar.

Taylorpolynom: Für die Berechnung des Taylorpolynoms über die Picard-Iteration kann in RiOT die Berücksichtigung der Fehlereinschließung ausgeschaltet werden, was sich noch einmal positiv auf die Rechenzeit auswirkt. Wir vermuten, dass dies in COSY-VI nicht gemacht wird, da dies aufgrund der Verwendung von *tallying*-Variablen keine Verbesserung der Rechenzeit ergeben würde.

Inneneinschließung: In RiOT setzen wir die Fehlereinschließung des Taylorpolynoms vor dem Berechnen der Inneneinschließung explizit auf Null, was eine engere Inneneinschließung zur Folge hat. Aufgrund von Testrechungen und Vergleichen mit COSY-VI liegt die Vermutung nahe, dass dies in COSY-VI nicht gemacht wird.

Lösungseinschließung: In der Berechnung der Lösungseinschließung verwenden wir schon berechnete Lösungseinschließungen in den Komponenten für nachfolgende Komponenten weiter. In COSY-VI wird das nicht gemacht. Außerdem verwenden wir eine Folge $\boldsymbol{I}^{(k)}_{n,\boldsymbol{y}^\star}$ nach (3.23), wohingegen in COSY-VI, ähnlich zur Dissertation von Makino [42, S. 146], die Folge

$$\boldsymbol{I}^{(k)}_{n,\boldsymbol{y}^\star} := 1.1^k \cdot \boldsymbol{I}^{(0)}_{n,\boldsymbol{K}(\boldsymbol{y}^\star)}, \quad k \geqslant 1,$$

verwendet wird[1]. Die Anzahl an möglichen Iterationsschritten wird in RiOT durch die Größe k_{max} begrenzt. k_{max} war in den gemachten Berechnungen auf den Wert 3 gesetzt. In COSY-VI richtet sich die Begrenzung der Iterationen nach der Aufweitung der Inneneinschließung $\boldsymbol{I}^{(0)}_{n,\boldsymbol{K}(\boldsymbol{y}^\star)}$. Es wird in

[1] siehe Variable `FAC` in Datei `TM.fox` des COSY-VI Pakets (Stand: 03/22/2004), Zeile 1 214.

Gleichung	Beispiel	RiOT	RiOT mit Strategie von Cosy-vi
Van-der-Pol	„1.E-11“ mit Schranke für Konditionszahl	315	2928
Lorenz-Attraktor	„1.E-11“	1341	3081
Lotka-Volterra		211	484

Tabelle 5.8: Gesamtzahl an Iterationen zur Lösungseinschließung im Vergleich.

Cosy-vi höchstens solange iteriert bis die Inneneinschließung um mehr als das fünffache aufgebläht wurde, also solange $k < \ln(5)/\ln(1.1) \approx 16.89$ ist[2].

Um einen Eindruck davon zu bekommen, wie sich die Wahl der Folge $\boldsymbol{I}^{(k)}_{n,\boldsymbol{y}^\star}$ auf die benötigten Iterationen auswirkt, haben wir RiOT mit der eben vorgestellten Strategie aus Cosy-vi ausgestattet und drei Beispiele erneut gerechnet. Tabelle 5.8 zeigt jeweils die Summe aller im Integrationsprozess durchgeführten Iterationsschritte zur Lösungseinschließung. Wie man sieht benötigt unsere Wahl der Folge $\boldsymbol{I}^{(k)}_{n,\boldsymbol{y}^\star}$ deutlich weniger Fixpunktiterationen. Allerdings sind die Auswirkungen auf die berechneten Lösungseinschließungen in allen drei Beispielen gering. Auch in Bezug auf die Rechenzeit sind die Unterschiede nicht von Bedeutung.

Verbesserung der Lösungseinschließung: Auch hier gehen schon berechnete Fehlereinschließungen direkt in die Berechnung nachfolgender Fehlereinschließungen ein. Des Weiteren führen wir die Iteration solange durch, bis die gewünschte Genauigkeit erreicht wurde. In Cosy-vi wird die iterative Verbesserung spätestens nach 10 Iterationen abgebrochen[3].

Lokaler Fehlerzuwachs: RiOT berechnet den lokalen Fehlerzuwachs nach (4.8). In Cosy-vi wird der Zuwachs nach

$$\epsilon^{(j)} = \max_{i=1,\ldots,\nu} \left\{ \sup(|I^{(t_j)}_{n,y^\star_i}|) - \sup(|I^{(t_{j-1})}_{n,y^\star_i}|) \right\}, \quad j > 0$$

berechnet, was bei nicht symmetrischen Intervallen auf kleinere Fehlerzuwächse und größere Schrittweiten führt.

Shrink-Wrapping: In der Heuristik des Shrink-Wrapping verwenden wir im Fall einer Parallelepipedeinschließung die Einschließung der nichtlinearen Terme nach (3.35) und nicht wie in Cosy-vi nach (3.34). Dieses Vorgehen hat sich

[2] siehe Datei `TM.fox` des Cosy-vi Pakets (Stand: 03/22/2004), Zeile 1 262.
[3] siehe Variable `NTOT` in Datei `TM.fox` des Cosy-vi Pakets (Stand: 03/22/2004), Zeile 1 214.

in wesentlich kleineren Shrink-Faktoren bemerkbar gemacht, wie sich am Beispiel der Van-der-Pol-Gleichung herausgestellt hat.

Rechenzeit: In puncto Rechenzeit hat RiOT ganz klar das Nachsehen. COSY-VI ist trotz gleichen Aufwands um ein Vielfaches schneller. Unklar ist, ob und wieviel die Wahl der Programmiersprache zu den schlechten Laufzeiten beiträgt. Vermutlich ist auch die in RiOT gewählte Datenstruktur nicht optimal. Positiv zu erwähnen sind aber gewisse Strukturen wie das in der Grundarithmetik angesprochene Record-Feature, das im Zuge der Verbesserung der Rechenzeit entstanden ist.

Ebenfalls positiv zu erwähnen sind die Verbesserungen bzgl. der Einschließungsqualität. Diese haben immerhin in vier von fünf Beispielen zu engeren Einschließungen im Vergleich zu COSY-VI „None/On“ geführt. Wir halten fest, dass RiOT als frei verfügbare Software eine durchaus gute Grundlage für weitere Implementierung in diesem Bereich darstellt.

Zu guter Letzt wollen wir noch darauf hinweisen, dass bei Differentialgleichungssystemen die Reihenfolge der Gleichungen einen Einfluss auf die Größe einer berechneten Einschließung haben kann [vgl. 41, S. 49], wir dies hier aber nicht diskutiert haben.

Literaturverzeichnis

[1] Alefeld, G. und J. Herzberger. *Introduction to Interval Computations*. Academic Press, New York, 1983.

[2] Balzert, H. *Lehrbuch der Objektmodellierung: Analyse und Entwurf*. Spektrum Akad. Verlag, Heidelberg, 1999.

[3] Barton, J. und L. Nackman. *Scientific and Engineering C++: An Introduction with Advanced Techniques and Examples*. Addison-Wesley, Boston, 6. Auflage, 2000.

[4] Berz, M. *Algorithms for Higher Order Automatic Differentiation in Many Variables with Applications to Beam Physics*. In: Griewank, A. und G. Corliss (Hg.), *Automatic Differentiation of Algorithms: Theory, Implementation, and Application*. SIAM, 1991.

[5] Berz, M. *From Taylor Series to Taylor Models*. In: *Beam Stability and Nonlinear Dynamics*. AIP Conference Proceedings, 1997.

[6] Berz, M. *COSY INFINITY Version 8.1 User's Guide and Reference Manual*. Technical Report MSUHEP-20704, Department of Physics and Astronomy, Michigan State University, East Lansing, MI 48824, 2002.
URL `http://cosy.pa.msu.edu/`

[7] Berz, M. und J. Hoefkens. *Verified High-Order Inversion of Functional Dependencies and Interval Newton Methods*. Reliable Computing, 7(5), 379–398, 2001.

[8] Berz, M. und J. Hoefkens. *COSY INFINITY Version 8.1 Programming Manual*. Technical Report MSUHEP-20703, Department of Physics and Astronomy, Michigan State University, East Lansing, MI 48824, 2002.
URL `http://cosy.pa.msu.edu/`

[9] Berz, M. und G. Hoffstätter. *Computation and Application of Taylor Polynomials with Interval Remainder Bounds*. Reliable Computing, 4, 83–97, 1998.

[10] Berz, M. und K. Makino. *Verified Integration of ODEs and Flows Using Differential Algebraic Methods on High-Order Taylor Models*. Reliable Computing, 4, 361–369, 1998.

[11] Berz, M. und K. Makino. *New Methods for High-Dimensional Verified Quadrature*. Reliable Computing, 5, 13–22, 1999.

[12] Braune, D. *Hochgenaue Standardfunktionen für reelle und komplexe Punkte und Intervalle in beliebigen Gleitpunktrastern*. Dissertation, Universität Karlsruhe, 1987.

[13] Corliss, G. und A. Griewank (Hg.). *Automatic Differentiation of Algorithms: Theory, Implementation, and Application*. SIAM, Philadelphia, 1991.

[14] Eberl, W. *Modellierung und Steuerung nichtlinearer Oszillatoren mit diskreten Abbildungen*. Dissertation, Technische Universität München, 1992.

[15] Eckel, B. *Thinking in C++, Second Edition, Volume One: Introduction to Standard C++*. Prentice Hall Inc., New Jersey, 2000.
URL `http://mindview.net/Books/TICPP/ThinkingInCPP2e.html`

[16] Eijgenraam, P. *The Solution of Initial Value Problems Using Interval Arithmetic*. Mathematical Centre Tracts No. 144. Stichting Mathematisch Centrum, 1981.

[17] Free Software Foundation. *GNU General Public License*. Version 2, 1991.
URL `http://www.gnu.org/licenses/licenses.html`

[18] Gamma, E., R. Helm, R. Johnson und J. Vlissides. *Entwurfsmuster: Elemente wiederverwendbarer objektorientierter Software*. Addison-Wesley-Longman, Bonn, 1996.

[19] Griewank, A. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. SIAM, Philadelphia, 2000.

[20] Hairer, E., S. Nørsett und G. Wanner. *Solving Ordinary Differential Equations I, Nonstiff Problems*. Springer, Berlin, Heidelberg, New York, 2. Auflage, 1993.

[21] Hairer, E. und G. Wanner. *Solving Ordinary Differential Equations II, Stiff and Differential-Algebraic Problems*. Springer, Berlin, Heidelberg, New York, 2. Auflage, 1996.

[22] Hammer, R., M. Hocks, U. Kulisch und D. Ratz. *C++ Toolbox for Verified Computing I - Basic Numerical Problems*. Springer, Heidelberg, New York, 1995.

[23] Heindl, G. *An Improved Algorithm for Computing the Product of Two Machine Intervals*. Interner Bericht IAGMPI-9304, Fachbereich Mathematik, Gesamthochschule Wuppertal, 1993.

[24] Heuser, H. *Lehrbuch der Analysis, Teil 2*. Teubner, Stuttgart, 8. Auflage, 1993.

[25] Heuser, H. *Gewöhnliche Differentialgleichungen: Einführung in Lehre und Gebrauch*. Teubner, Stuttgart, 3. Auflage, 1995.

[26] Hoefkens, J. *Rigorous Numerical Analysis With High-Order Taylor Models*. Dissertation, Michigan State University, East Lansing, Michigan, USA, 2001.

[27] Hoefkens, J., M. Berz und K. Makino. *Efficient High-Order Methods for ODEs and DAEs*. In: Corliss, G., C. Faure, A. Griewank, L. Hascoët und U. Naumann (Hg.), *Automatic Differentiation of Algorithms from Simulation to Optimization*. Springer, 2002, 343–348.

[28] Hoefkens, J., M. Berz und K. Makino. *Computing Validated Solutions of Implicit Differential Equations*. Adv. Comput. Math., 19(1-3), 231–253, 2003.

[29] Hoefkens, J., M. Berz und K. Makino. *Controlling the Wrapping Effect in the Solution of ODEs for Asteroids*. Reliable Computing, 8, 21–41, 2003.

[30] Kaucher, E. W. und W. L. Miranker. *Self-Validating Numerics for Function Space Problems*. Academic Press, New York, 1984.

[31] Klatte, R., U. Kulisch, C. Lawo, M. Rauch und A. Wiethoff. *C-XSC: A C++ Class Library for Extended Scientific Computing*. Springer, Berlin, 1993.

[32] Klein, G., M. Rott, S. Schäfler und S. Wimmer. *Schnelle Algorithmen für dünnbesetzte Polynome*. Interdisziplinäres Projekt, Betreuer: Dr. Michael Kaplan, Fakultät für Mathematik, Technische Universität München, 1997. URL `http://www.mathematik.tu-muenchen.de/\textasciitildekaplan/ca/spock/`

[33] Krämer, W. *Inverse Standardfunktionen für reelle und komplexe Intervallargumente mit a priori Fehlerabschätzungen für beliebige Datenformate*. Dissertation, Universität Karlsruhe, 1987.

[34] Krückeberg, F. *Ordinary Differential Equations*. In: Hansen, E. (Hg.), *Topics in Interval Analysis*. Clarendon Press, Oxford, 1969, 91–97.

[35] Kulisch, U. *Grundlagen des numerischen Rechnens: mathematische Begründung der Rechnerarithmetik*. Nummer 19 in Reihe Informatik. Bibliographisches Institut, Mannheim, Wien, Zürich, 1976.

[36] Kulisch, U. und W. Miranker. *Computer Arithmetic in Theory and Practice*. Academic Press, New York, 1981.

[37] Kühn, W. *Rigorously Computed Orbits of Dynamical Systems without the Wrapping Effect*. Computing, (61), 47–67, 1998.

[38] Lenk, R. und W. Gellert (Hg.). *Fachlexikon ABC Physik, Band 1 A-Ma*. Harri Deutsch, Thun, Frankfurt/Main, 2. Auflage, 1989.

[39] Lerch, M., G. Tischler und J. Wolff von Gudenberg. *filib++ - Interval Library Specification and Reference Manual*. Technical Report 279, Universität Würzburg, 2001.

[40] Lerch, M., G. Tischler, J. Wolff von Gudenberg, W. Hofschuster und W. Krämer. *The Interval Library filib++ 2.0: Design, Features and Sample Programs*. Preprint 2001/4, Wissenschaftliches Rechnen/Softwaretechnologie, Universität Wuppertal, 2001.

[41] Lohner, R. *Einschließung der Lösung gewöhnlicher Anfangs- und Randwertaufgaben*. Dissertation, Universität Karlsruhe, 1988.

[42] Makino, K. *Rigorous Analysis of Nonlinear Motion in Particle Accelerators*. Dissertation, Michigan State University, East Lansing, Michigan, USA, 1998.

[43] Makino, K. und M. Berz. *Remainder Differential Algebras and Their Applications*. In: Berz, M., C. Bischof, G. Corliss und A. Griewank (Hg.), *Computational Differentiation: Techniques, Application, and Tools*. SIAM, 1996, 63–75.

[44] Makino, K. und M. Berz. *Efficient Control of the Dependency Problem Based on Taylor Model Methods*. Reliable Computing, 5, 3–12, 1999.

[45] Makino, K. und M. Berz. *Taylor Models and Other Validated Functional Inclusion Methods*. International Journal of Pure and Applied Mathematics, 4(4), 379–456, 2003.

[46] Makino, K. und M. Berz. *Suppression of the Wrapping Effect by Taylor Model-Based Validated Integrators*. Report MSUHEP 40910, Department of Physics, Michigan State University, East Lansing, MI 48824, 2004.
URL `http://bt.pa.msu.edu/pub/`

[47] Moore, R. *Interval Analysis*. Prentice-Hall, Engelwood Cliffs, N.J., 1966.

[48] Moore, R. *Methods and Applications of Interval Analysis*. SIAM, Philadelphia, USA, 1979.

[49] Muller, J.-M. *Elementary functions: algorithms and implementation*. Birkhäuser, Boston, 1997.

[50] Neumaier, A. *Interval Methods for Systems of Equations*. Cambridge University Press, Cambridge, 1990.

[51] Neumaier, A. *Taylor Forms - Use and Limits*. Reliable Computing, 9, 43–79, 2003.

[52] Online Lexika von Elsevier/Spektrum Akademischer Verlag, Stand: Februar 2005.
URL `http://www.wissenschaft-online.de`

[53] Rall, L. *Automatic Differentiation: Techniques and Applications*. Springer, New York, 1981.

[54] Ratschek, H. und J. Rokne. *Computer Methods for the Range of Functions*. Ellis Horwood, Chichester, 1984.

[55] Revol, N., K. Makino und M. Berz. *Taylor Models and Floating-Point Arithmetic: Proof that Arithmetic Operations are Validated in COSY*. Technical Report RR 4737, INRIA, LIP, École Normale Supérieure de Lyon, 2003.

[56] Rihm, R. *Über Einschließungsverfahren für gewöhnliche Anfangswertprobleme und ihre Anwendung auf Differentialgeichungen mit unstetiger rechter Seite*. Dissertation, Universität Karlsruhe, 1993.

[57] Rump, S. *A Note on Epsilon-Inflation*. Reliable Computing, 4, 371–375, 1998.

[58] Stahl, V. *Interval Methods for Bounding the Range of Polynomials and Solving Systems of Nonlinear Equations*. Dissertation, Technisch-Naturwiss. Fakultät, Universität Linz, 1995.

[59] Stroustrup, B. *Die C++-Programmiersprache*. Addison-Wesley-Longman, Bonn, 3. Auflage, 1998.

[60] Veldhuizen, T. *Techniques for Scientific C++*. Technical Report 542, Computer Science, Indiana University, Bloomington, 2000.

[61] XSC Webseite, Stand: November 2004.
URL `http://www.xsc.de`

Lebenslauf

Name:	Ingo Eble
Geburtsdatum:	16. Oktober 1974
Geburtsort:	Zell am Harmersbach
Eltern:	Berthold Eble Loretta Eble, geb. Zimmerer
Schulbildung:	Sep. 1980 – Jul. 1985: Grundschule in Nordrach
	Sep. 1985 – Jul. 1991: Realschule in Zell am Harmersbach Mittlere Reife: 5. Juli 1991
	Sep. 1991 – Jun. 1994: Technisches Gymnasium in Offenburg Abiturprüfung: 14. Juni 1994
Studium:	Okt. 1994 – Jun. 2001: Studium der Mathematik an der Universität Karlsruhe mit Nebenfach Mechanik Diplomprüfung: 11. Juni 2001
	Okt. 2003 – Apr. 2007: Zweitstudium Sportwissenschaft an der Universität Karlsruhe. Bachelorprüfung: 20. April 2007
Berufstätigkeit:	Dez. 1998 – Jun. 2001: Wissenschaftliche Hilfskraft am Institut für Angewandte Mathematik
	Jun. 2001 – Aug. 2001: Geprüfte wissenschaftliche Hilfskraft am Institut für Angewandte Mathematik
	Sep. 2001 – Aug. 2003: Wissenschaftlicher Angestellter am Institut für Angewandte Mathematik
	Sep. 2003 – Mär. 2007: Geprüfte wissenschaftliche Hilfskraft am Institut für Angewandte Mathematik