Ein rekursives Verfahren zur Berechnung von Strudeln für Differentialgleichungen $y' = -\frac{A(x,y)}{B(x,y)}$ um eine Unbestimmtheitsstelle

Diplomarbeit
vorgelegt von
Kay Moritzen
am Lehrstuhl Mathematik VI
der Universität Bayreuth

Bayreuth, den 1.12.2000

Inhaltsverzeichnis

1 Einleitung und Bezeichnungen

Poincaré veröffentlichte schon 1880 im ersten Band seiner "Œvres" [11] frühe Arbeiten zur Behandlung von Differentialgleichung der Form $\frac{dx}{X} = \frac{dy}{Y}$ [1]. Dort folgerte er schon erste Erkenntnisse und der noch später verwendete Begriff bzw. die Definition der Wirbelbedingung an einer Unbestimmtheitsstelle findet dort seinen Ursprung. Jahrzehnte später folgte eine Idee von Dehn, die sich ausschließlich mit Differentialgleichungen der Form $y' = -\frac{A(x,y)}{B(x,y)}$ [2] befaßte. Dabei galt bei beiden für X, Y bzw. A, B, daß beide Polynome im Nullpunkt verschwanden, jedoch der Vektor $(X,Y)^T$ bzw. $(A,B)^T$ in einer punktierten (gelochten) Umgebung $U \setminus \{0\}$ nie verschwand.

1933 wurde von Frommer, dessen Doktorvater Dehn war, eine Arbeit angefertigt, die noch tiefere Erkenntnisse lieferte. zum ersten Mal wurden dort feste Begriffe, wie die Strudelgröße, und Beweise vorgeschlagen, die das Lösungsverhalten solcher Differentialgleichungen untersuchen. Auch wurde dort zum ersten Mal ein rekursives Verfahren vorgestellt, das der Nachprüfung dient, ob eine bestimmte Differentialgleichung vom Wirbel- oder Strudeltyp ist. Auf dieses Verfahren bezieht sich diese Arbeit und möchte es in seiner Durchführung verfeinern.

In Anbetracht der vielen Jahrzehnte die diese Arbeiten in den Annalen der Mathematik vergessen ließen, präsentiert diese Arbeit nun eine moderne Betrachtung o.g. Differentialgleichungen. Ein rekursives Verfahren zur Berechnung der von Frommer definierten "Strudelgrößen" ist Basis für eine Computerunterstützte Überprüfung von Differentialgleichungen genannten Typs. In den folgenden Kapiteln sollen also Formeldarstellungen für ein rekursives Verfahren ermittelt und zusammengestellt werden.

Natürlich darf in einer solchen Arbeit nicht eine reale Implementierung solcher Formeln fehlen, so daß die zu Beginn erschlossenen Erkenntnisse im Schlußteil in einem Java-Programm zum Einsatz kommen sollen.

Die Arbeiten von Poincaré und Frommer wurden an geeigneter Stelle mitbedacht. Zitierte Äußerungen dieser Autoren dienen der Hervorhebung der geschichtlichen Bedeutung des Themas, dessen Bearbeitung nunmehr schon hundert Jahre andauert und - in eingeweihten Kreisen - immer noch nicht als abgeschlossen gilt.

[1] X, Y Polynome in x und y
[2] $A(x,y), B(x,y)$ Polynome in x, y

2 Differentialgleichungen, Strudel und Wirbel

Bevor die Wirbelbedingungen von Poincaré dargestellt werden, sei ein einfacher Wirbel bzw. Strudel und seine notwendigen Notationen kurz umrissen.

2.1 Differentialgleichungen

Definition 2.1 (Differentialgleichung). *Sei $G \subset \mathbb{R}^2$ eine offene Teilmenge und*

$$f : G \to \mathbb{R}, \quad (x, y) \mapsto f(x, y)$$

eine stetige Funktion. Dann heißt

$$y' = f(x, y) \tag{2.1}$$

eine Differentialgleichung erster Ordnung .

Definition 2.2 (Lösung einer Differentialgleichung). *Unter einer Lösung einer Differentialgleichung von (2.1) versteht man eine auf einem Intervall $I \subset \mathbb{R}$ definierte stetig differenzierbare Funktion*

$$\varphi : I \to \mathbb{R}$$

mit den Eigenschaften

- *Der Graph φ ist in G enthalten, d.h.*

$$\Gamma_\varphi = \{(x, y) \in I \times \mathbb{R} : y = \varphi(x)\} \subset G$$

- *Es gilt*

$$\varphi'(x) = f(x, \varphi(x)) \qquad \forall x \in I$$

Damit die zweite Bedingung überhaupt formuliert werden kann muß die erste gestellt werden.

Bemerkung 2.3 (Zur Eindeutigkeit). *Für die Eindeutigkeit einer Lösung wird die Lipschitzbedingung benötigt. Dieser Begriff und auch der Beweis der Eindeutigkeit kann auf den Seiten 96ff in [4], Bd.II, nachgelesen werden. Im weiteren sei der Einfachheit halber stets ausreichende Glattheit vorausgesetzt, was im Falle der hier betrachteten Differentialgleichungen keine spektakuläre Forderung ist.*

2.2 Autonome Systeme und Pfaffsche Formen

Die Darstellung eines einfachen Wirbels ist häufig in einführender Literatur zu Differentialgleichungen zu finden[1].

[1] in [4] §10, [9] Bd. II, §11, [13] S.379 und [2] §8

Beispiel 2.1. *Sei $G \subset \mathbb{R} \times \mathbb{R}_{>0}$. Sei weiterhin eine Differentialgleichung*

$$y' = -\frac{x}{y} \qquad \text{in } G \tag{2.2}$$

gegeben. Lösungen von (2.2) sind die Halbkreise

$$y = \pm\sqrt{c - x^2}, \qquad |x| < \sqrt{c} \tag{2.3}$$

In dieser Lösung fehlen die beiden Punkte $(-\sqrt{c}, 0)$ und $(+\sqrt{c}, 0)$. Durch eine Betrachtung von $x'(y) = \frac{dx}{dy}$, dem umgekehrten Fall von $y'(x) = \frac{dy}{dx}$, könnte man die fehlenden Punkte herbeischaffen. Pfaffsche Formen legitimieren so eine Verfahrensweise. Diese definiert man wie folgt:

Definition 2.4 (Pfaffsche Form). *Unter einer Pfaffschen Form*[2] *auf einer offenen Menge $U \subset \mathbb{R}^n$ versteht man eine Abbildung*

$$\omega : U \to L(\mathbb{R}^n, \mathbb{C}).$$

Für jedes $x \in U$ ist also $\omega(x)$ eine $\mathbb{R}$-lineare Abbildung

$$\omega(x) : \mathbb{R}^n \to \mathbb{C}.$$

Sei $h : U \to \mathbb{R}$ eine Funktion.

- *Je nach dem, ob der Wert von $h \cdot \omega(x)$ reell oder komplex ist für alle $x \in U$ und $h \in \mathbb{R}^n$, heißt die Pfaffsche Form reell oder komplex.*
- *Ist h stetig und nirgends verschwindend auf U, so heißt h Multiplikator.*

Sei an dieser Stelle der Begriff des Tangentenfeldes definiert. Das Ziel ist nun eine eindeutige Korrespondenz zwischen Lösungen von Differentialgleichungen (Funktionen) und Lösungen von Differentialen (Kurven) zu schaffen.

Definition 2.5 (Richtungsfeld, Tangentenfeld). *Sei $G \subset \mathbb{R}^2$ offen und $f : G \to \mathbb{R}$ eine Abbildung in G. Die Gesamtheit der Geraden durch $(x, y) \in G \subset \mathbb{R}^2$ mit der Steigung $y' = f(x, y)$ heißt Richtungsfeld zu $y' = f(x, y)$. Sie bilden das Tangentenfeld zu den Lösungen von $y' = f(x, y)$.*

Die Lösung (2.3) kann nun als Niveaulinienfunktion geschrieben werden. Es gilt

$$F(x, y) = c \qquad \Longleftrightarrow \qquad F(x, y) = x^2 + y^2$$

Hieraus ergibt sich als Gradient für $F(x, y)$

$$\operatorname{grad}(F) = \begin{pmatrix} \frac{\partial F}{\partial x} \\ \frac{\partial F}{\partial y} \end{pmatrix} = \begin{pmatrix} 2x \\ 2y \end{pmatrix}$$

Definition 2.6 (Totales Differential). *Sei $U \subset \mathbb{R}^n$ offen. Das totale Differential $dF(x)$ einer stetig differenzierbaren Funktion $F : U \to \mathbb{C}$ ist für jeden Punkt $x \in U \subset \mathbb{R}^n$ eine (komplexwertige) Linearform*[3]*:*

$$dF(x)h = \sum_{i=1}^{n} \frac{\partial}{\partial x_i} F(x) h_i = \langle grad(F(x)), h \rangle, \qquad h \in \mathbb{R}^n, \tag{2.4}$$

wobei mit $\langle \cdot, \cdot \rangle$ ein Skalarprodukt gemeint ist. Im Falle, daß die Totalität nur auf U definiert ist, heißt sie lokal total.

[2] Vgl. [9] Bd.II, §11. Auch "Differentialform ersten Grades" oder "1-Form" genannt.
[3] Vgl. [9] Bd.II, S. 52.

Seien $(dx_i(\xi))_{i=1,\dots,n} = (dx_i)_{i=1,\dots,n}$, die Differentiale der kanonischen Koordinatenfunktionen des $\mathbb{R}^n$ an der Stelle $\xi \in U$. Wegen $h_i = dx_i h$ und unter Verwendung der Schreibweise (2.4) folgt

$$dF(\xi)h = \sum_{i=1}^{n} \frac{\partial F}{\partial x_i}(\xi)h_i = \sum_{i=1}^{n} \frac{\partial F}{\partial x_i}(\xi)dx_i(\xi)h,$$

wofür man abkürzend schreibt

$$dF = \sum_{i=1}^{n} \frac{\partial F}{\partial x_i}dx_i$$

Somit läßt sich das Tangentenfeld des Wirbels (Wirbelfeld) aus (2.2) als Pfaffsche Form (Windungsform) schreiben[4]

$$dF = \left\langle \operatorname{grad}(F(x,y)), \begin{pmatrix} dx \\ dy \end{pmatrix} \right\rangle = 2xdx + 2ydy = \omega \tag{2.5}$$

Setzt man $\omega = 0$ erhält man folgende Äquivalenzen des Tangentenfeldes

$$0 = 2xdx+2ydy \quad \Longleftrightarrow \quad y'(x) = \frac{dy}{dx} = -\frac{x}{y} \quad \Longleftrightarrow \quad x'(y) = \frac{dx}{dy} = -\frac{y}{x}$$

Daß diese Äquivalenzen richtig sind, sollen die folgenden Erläuterungen zeigen.

Im folgenden werden reguläre Pfaffsche Formen

$$\omega = A(x,y)dx + B(x,y)dy$$

auf einer offenen Umgebung $G \subset \mathbb{R}^2$ betrachtet. Die Funktionen $A, B : G \to \mathbb{R}$ seien hinreichend glatt. ω sei auf G regulär, d.h. $(A,B) \neq 0$ auf G.

Definition 2.7 (Lösung einer Pfaffschen Form[5]). *Sei $I \subset \mathbb{R}$ ein Intervall. Eine glatte Kurve W heißt Lösung der Gleichung $\omega = 0$, wenn es eine glatte Parametrisierung $\Phi : I \to \mathbb{R}^2$ von W gibt mit*

- $\Phi(I) \subset G$
- $\omega \circ \Phi = 0$

Bemerkung 2.8. *Die zweite Forderung aus Definition (2.7) wird ausführlicher formuliert mit $\Phi = (\varphi_1, \varphi_2)$. Der Parameter auf I ist mit t bezeichnet. So gilt*[6]

$$\omega \circ \Phi = ((A \circ \Phi)\varphi_1' + (B \circ \Phi)\varphi_2')\, dt$$

Die Gleichung $\omega \circ \Phi = 0$ bedeutet also

$$A(\varphi_1(t), \varphi_2(t)) \cdot \varphi_1'(t) + B(\varphi_1(t), \varphi_2(t)) \cdot \varphi_2'(t) = 0$$

Daraus ergibt sich für $y' = f(x,y)$ der wichtige Satz

Satz 2.9. *Die Lösungsfunktion der Differentialgleichung $y' = f(x,y)$ und die Lösungskurve der zugehörigen Pfaffschen Form $\omega = dy - f(x,y)dx$ mit $\omega = 0$ sind gleich*[7].

[4] Sehr schön läßt sich an diesem Beispiel der Multiplikator $h = 2$ erkennen, der sich, wie weiter unten zu sehen, im Falle der Pfaffschen Formen aufgrund der Invarianz von $\omega = 0$ wegkürzen läßt.

[5] [6] S.164

[6] nach Bemerkung in [6] S.164 und Regeln aus [6], S.88ff.

[7] Da Kurven und Funktionen nicht dasselbe sind, ist dieser Satz unpräzise formuliert. Wie sich jedoch die Funktionen und Kurven einander entsprechen wird aus dem Beweis ersichtlich.

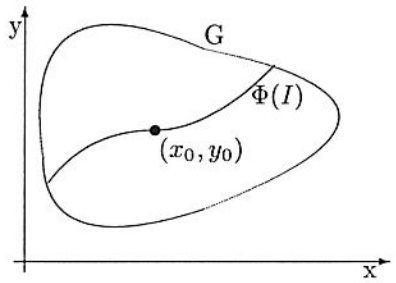

Abbildung 2.1: $\Phi(I)$ verläuft von Rand zu Rand auf G.

Beweis: [6], S.164

Bemerkung 2.10. *Nicht jede Pfaffsche Form kann man einer Differentialgleichung so zuordnen, daß die Lösungsmengen übereinstimmen. Ein Gegenbeispiel ist die Windungsform*[8]*. Sei* $G = \mathbb{R}^2 \setminus \{0\}$ *und* $\omega = xdx + ydy$ *wie in (2.5). Lösungen von* $\omega = 0$ *sind nur die Kreise um den Nullpunkt. Definiert man für festes* $r \in \mathbb{R}$, $r > 0$, *eine parametrisierte Kurve* $\Phi : \mathbb{R} \to G$ *durch* $\Phi(t) = (r\cos(t), r\sin(t))$, *so ist*

$$\omega \circ \Phi = \big(r^2\cos(t)(-\sin(t)) + r^2\sin(t)\cos(t)\big)\,dt = 0$$

Diese Kurven können aber nicht Integralkurven einer über G *definierten Differentialgleichung sein, denn als solche müßten sie in* G *von Rand zu Rand laufen.*

Für die Lösung von $\omega = 0$ werden nun verschiedene andere Schreibweisen eingeführt. Dazu wird das autonome System betrachtet

$$\dot{x} = \frac{dx}{dt} = B(x,y), \qquad \dot{y} = \frac{dy}{dt} = -A(x,y) \tag{2.6}$$

und eine Lösungskurve $\left(I, \Phi(\cdot) = \binom{x(\cdot)}{y(\cdot)}, \Phi(I)\right)$ gewählt durch $(x_0 := x(0), y_0 := y(0)) \in G$. I kann ohne weiteres das maximale (offene) Existenzintervall[9] sein, so daß $\Phi(I)$ von Rand zu Rand in G verläuft, wenn I und G beschränkt sind[10] (siehe Abb. (2.1)). Im Phasenraum hängen A und B nur von den Variablen x und y des Phasenraums ab. Dies eröffnet weitgehende Möglichkeiten zur Wahl von anderen Parametrisierungen von Teilstücken von $\Phi(I)$.

Zunächst ist $(I, \Phi, \Phi(I))$ Lösung von $\omega = 0$. Ist in (x_0, y_0), oder in irgendeinem anderen Punkt von $\Phi(I)$, $B(x_0, y_0) \neq 0$, so gilt dies auch in einer Umgebung von (x_0, y_0). Wegen $\dot{x} \neq 0$ kann man dort x als Parameter der Lösungskurve wählen. Dann wird das entsprechende Kurvenstück durch die Lösungen von $y' = -\frac{A(x,y)}{B(x,y)}$ beschrieben. Ist $A(x_1, y_1) \neq 0$ in einem Punkt $(x_1, y_1) \in \Phi(I)$, so kann ein Stück von $\Phi(I)$ durch die Lösungen von $x' = -\frac{B(x,y)}{A(x,y)}$ beschrieben werden, indem y als Parameter gewählt wird (siehe Abb. (2.2)). Mit Satz (2.9) erkennt man, daß die Lösungskurven von $\omega = 0$ genau die Lösungskurven von (2.6) sind, indem man noch die Invarianz der Lösungen von $\omega = 0$ gegen Multiplikation mit einem Multiplikator h in jedem offenen Teilstück von G benutzt[11].

Sei I das maximale Existenzintervall zu (2.6). Aus den vorhergehenden Betrachtungen folgt, daß bei beschränktem G und unbeschränktem I die folgenden Situationen eintreten können.
O.E. sei $0 \in I$, $I = (t^-, t^+)$ mit den Intervallenden t^-, t^+: $-\infty \leq t^- < 0 < t^+ \leq +\infty$.

[8] [6], S.165, [4] Bd.III, S.197f
[9] Definition aus [17] und [6]
[10] Definition aus [6].
[11] [6] S.168

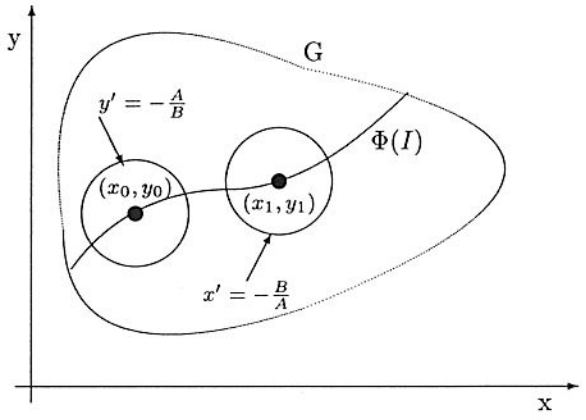

Abbildung 2.2: verschiedene Steigungen verschiedener Punkte auf $\Phi(I)$

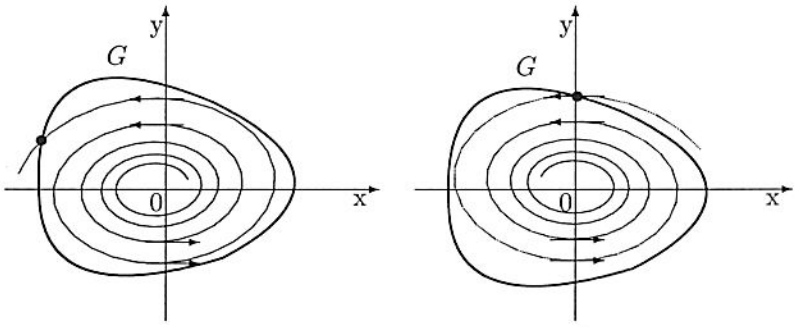

Abbildung 2.3: Beispiel einer sich herauswindenden (links) und hineinwindenden (rechts) Spirale. In beiden Fällen ist t nach dem Polarwinkel φ parametrisiert. Mit 0 ist der kritische Punkt markiert. Der Punkt am Rande des Gebietes G soll den Eintritts- respektive den Austrittspunkt der Kurve kennzeichnen.

a.) Eine Spirale die sich aus einem kritischen Punkt herauswindet, wenn t wächst.
$t^- = -\infty, \quad t^+ < +\infty$
(Abb. (2.3), links)

b.) Eine Spirale die sich in einen kritischen Punkt hineinwindet, wenn t wächst.
$t^- > -\infty, \quad t^+ = +\infty$
(Abb. (2.3), rechts)

c.) Eine periodische Lösung, die in G verläuft.
$I = \mathbb{R}$, d.h. $t^- = -\infty, \quad t^+ = +\infty$
(Abb. (2.4))

Ist I an einem Ende endlich, so muß die Lösung von (2.6) bei beschränktem G in einen Punkt aus ∂G laufen, wenn t sich diesem Ende annähert.
Die Lösung $\binom{x(\cdot)}{y(\cdot)}$ kann nur an einem "unendlichen Ende" von I in einen kritischen Punkt einmünden.
In allen Fällen liegen Überdeckungen von Teilen von $\Phi(I)$ oder $\Phi(I)$ der folgenden Form vor:
Sei $I' \subset I$, I' kompakt. Dann ist

$$\Phi(I') = \bigcup_{i=1}^{N} K_i$$

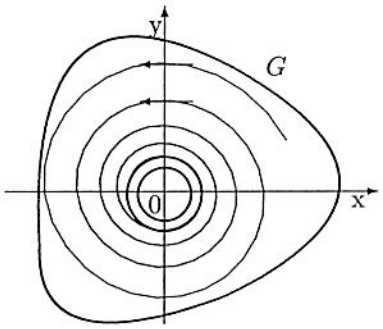

Abbildung 2.4: Periodische Lösung um den kritischen Punkt. Im Inneren der periodischen Lösung können nur weitere periodische Lösungen liegen (Wirbelfall), aber auch Spiralen und weitere periodische Lösungen, wie etwa Grenzzykel.

mit

$$K_i = \left\{ (x, y(x)) |\ x \in I_i \text{ offen},\ y' = -\frac{A}{B} \right\}$$

oder

$$K_i = \left\{ (x(y), y) |\ y \in I_i \text{ offen},\ x' = -\frac{B}{A} \right\}.$$

Und daraus folgt für ganz I

$$\Phi(I) = \bigcup_{i=1}^{\infty} K_i, \qquad K_i \text{ wie oben.}$$

Im folgenden wird die Schreibweise $y' = -\frac{A}{B}$ für das Problem (2.6) bzw. $\omega = 0$ verwendet, indem sozusagen formal dt herausgekürzt wird. Diese Schreibweise hat zwar den Nachteil, daß sie nur lokal und nicht einmal überall lokal gilt, doch enthält die rechte Seite keine Unbestimmtheit mehr hinsichtlich eines Multiplikators h wie etwa die Gleichung $\omega = 0$, die dieselbe Lösungsmenge liefert wie $h\omega = 0$. Weiter dient $y' = -\frac{A}{B}$ (bzw. $x' = -\frac{B}{A}$) dazu, aus $\omega = 0$ auf das System (2.6) zu schließen. Sei also der Fall betrachtet

i. $0 \in G$,

ii. 0 kritischer Punkt von (2.6),

iii. $(x_0 = x(0), y_0 = y(0)) \neq 0$ als Anfangswert zu (2.6).

Dann liegt $\Phi(I)$ ganz in $G \setminus \{0\}$. Sei $h : G \setminus \{0\} \to \mathbb{R} \setminus \{0\}$ stetig differenzierbar. Auch hier ist die Schreibweise $y' = -\frac{A}{B}$ (bzw. $x' = -\frac{B}{A}$) von Nutzen, denn man findet, daß im Sinn des Anfangs dieses Absatzes für das System (2.6) und für

$$\dot{x} = h(x,y)B(x,y), \qquad \dot{y} = -h(x,y)A(x,y) \tag{2.7}$$

mit $x_0 = x(0)$ und $y_0 = y(0)$ und maximalem Existenzintervall I, das 0 enthalte, $\Phi(I)$ und die Spur $\Psi(I)$ der Lösungskurve $(I, \Psi, \Psi(I))$ von (2.7) übereinstimmen.

Damit sind die Möglichkeiten diskutiert, Differentialgleichungen oder Gleichungen $\omega = 0$ für zugehörige Pfaffsche Formen ω des zu erörternden Problems zu lösen.

2.3 Poincarésche Wirbelbedingung

Gegeben seien die reell analytischen Funktionen in x und y

$$\begin{aligned} A(x,y) &:= x + p(x,y), \qquad p(x,y) := \sum_{i=2}^{\infty} p_i(x,y), \\ B(x,y) &:= y + q(x,y), \qquad q(x,y) := \sum_{i=2}^{\infty} q_i(x,y), \end{aligned} \tag{2.8}$$

wobei p_i und q_i homogene Polynome vom Grad i sind. $A(x,y)$ und $B(x,y)$ sind konvergente Potenzreihen in einer offenen Umgebung G von Null. Mit linearem Anteil werden die homogenen Polynome ersten Grades aus A und B bezeichnet, d.h. für $A(x,y)$ ist der lineare Anteil x und für $B(x,y)$ ist der lineare Anteil y. Diese linearen Anteile sind markant für das nun folgende

Definition 2.11 (Poincarésche Problem). *Seien A, B und $G \subset \mathbb{R}^2$ wie in (2.6) und (2.7) bzw. konstruiert wie in (2.8). Die Pfaffsche Form*

$$\omega := A(x,y)dx + B(x,y)dy = 0$$

über G sei regulär in $G\backslash\{0\}$. Das Poincarésche Problem[12] besteht in der Charakterisierung der Lösungen von $\omega = 0$ in der Nähe des kritischen Punktes 0. Nach dem vorherigen Abschnitt sind die Lösungen durch die Differentialgleichungen

$$y' := \frac{dy}{dx} = -\frac{A(x,y)}{B(x,y)}, \qquad B(x,y) \neq 0 \tag{2.9}$$

bzw.

$$x' := \frac{dx}{dy} = -\frac{B(x,y)}{A(x,y)}, \qquad A(x,y) \neq 0 \tag{2.10}$$

bestimmt.

Die linearen Anteile des Poincaréschen Problems sind das Struktur-Merkmal dieser Klasse von Differentialgleichungen. Andere Probleme die dieses Strukturmerkmal nicht aufweisen sind bisher noch ungelöst. Im 5. Kapitel dieser Arbeit soll ein Ausblick auf die Analyse der Differentialgleichungen ohne dieses Strukturmerkmal gewagt werden, um die Einsetzbarkeit des noch zu erörternden rekursiven Verfahrens auch auf solche Probleme zu zeigen.
Bisher wurden die Begriffe Wirbel und Strudel bzw. Wirbel- und Strudelfall ohne jegliches Fundament einer Definition verwand. Dieser Umstand wird also an dieser Stelle mit Hilfe der Definition des Poincaréschen Problems nachgebessert.

Definition 2.12 (Wirbel, Strudel, Wirbelfall, Strudelfall). *Sei das Poincarésche Problem aus Definition (2.11) betrachtet. Sind in einer gelochten Umgebung $G \setminus \{0\}$ des kritischen Punktes 0 alle in $G \setminus \{0\}$ liegenden Integralkurven geschlossen, so sei von Wirbel gesprochen oder: es läge der Wirbelfall in 0 vor. Liegt nicht der Wirbelfall vor, so sei von einem Strudelfall gesprochen bzw. es handle sich um einen Strudel. In [15] wird aufgezeigt, daß im Strudelfall die Integralkurven Spiralen um den Nullpunkt bilden.*

Poincaré ging von der Voraussetzung aus, daß sich im Wirbelfall die Lösungskurven als Niveaulinien $F(x,y) = c$ (c eine Konstante) darstellen lassen. $F(x,y)$ ist dabei eine in einer Umgebung des Nullpunktes konvergente Potenzreihe[13]. Es gibt insbesondere keine Integralkurve in G, die in den Nullpunkt

[12] H.Poincaré, Journal de Mathematiques 1885
[13] [5] S.397

mit einer bestimmten Richtung einmündet. Und genau diese Eigenschaft ist es, die auf die Klassifizierung der Strudel und Wirbel als Lösungen von $\omega = 0$ führt.

Frommer konnte mit dem Vergleich der Tangentenfelder zweier Differentialgleichungen, wobei eines jedenfalls geschlossene Integralkurven hat, auf die Poincaréschen Bedingungen für einen Wirbel schließen, ohne dabei die Voraussetzung der Existenz einer Funktion F wie oben zu fordern. Daß diese Bedingungen zugleich notwendig als auch hinreichend sind, wurde in [5], §2 gerechtfertigt[14] und wird im Satz 3.1 im nächsten Kapitel endgültig bewiesen. Im nächsten Abschnitt dieses Kapitels wird die Verfahrensweise dieses Vergleiches aufzeigen.

2.4 Vergleichsdifferentialgleichung und Verfahren nach Dehn

Um die Bedingungen für einen Strudel bzw. einen Wirbel zu bestimmen, schlug Dehn 1929 ein Verfahren vor, welches sich einer konstruierten Vergleichsdifferentialgleichung bediente, um die Differenz zwischen gegebener und konstruierter Differentialgleichung als Kriterium für das Vorhandensein eines Wirbels oder Strudels zu nutzen. Es wird sich zeigen, daß bei Nicht-Vorliegen des Wirbelfalls die Gestalt (2.8) für A, B das Vorliegen des Strudelfalls nach sich zieht, d.h. alle Integralkurven von $y' = -\frac{A}{B}$, die in der Nähe von 0 beginnen, sind Spiralen, die mit positiver oder negativer Durchlaufrichtung des Polarwinkels φ in den kritischen Punkt 0 einmünden.
Es sei daher

$$F(x,y) := x^2 + y^2 + \sum_{i=3}^{n} F_i(x,y) \tag{2.11}$$

für $n \geq 3$ betrachtet, wobei F_i ein homogenes Polynom in x und y vom Grad i ist. Für die Ableitungen nach x bzw. nach y ergibt sich für $F(x,y)$

$$F_x(x,y) := \frac{\partial}{\partial x} F(x,y) = 2x + \sum_{i=3}^{n} F_{xi}(x,y),$$

$$F_y(x,y) := \frac{\partial}{\partial y} F(x,y) = 2y + \sum_{i=3}^{n} F_{yi}(x,y)$$

bzw. wenn Mißverständnisse ausgeschlossen werden können

$$F_x := F_x(x,y), \qquad F_y := F_y(x,y)$$

Das Verfahren[15] beruht nun auf der Konstruktion einer Vergleichsdifferentialgleichung

$$y_1' := f(x,y),$$

wobei $f(x,y)$ ein Tangentenfeld beschreibt, dessen zugehörige Lösungskurven in der gelochten Umgebung des Nullpunktes $G \setminus \{0\}$ sicher geschlossene Kurven darstellen, somit also Wirbel bilden. Sei nun von der folgenden geometrisch einleuchtenden Annahme ausgegangen:
Gelingt es eine solche Vergleichsdifferentialgleichung so zu bilden, daß die Feldrichtungen der Ausgangs- und der Vergleichsdifferentialgleichung nie in einer bestimmten Umgebung des Nullpunktes übereinstimmen, so ist der Nachweis erbracht, daß nicht beide vom Wirbeltyp sind. Da diese Eigenschaft jedoch von der Vergleichsdifferentialgleichung vorausgesetzt wird, muß die Ausgangsdifferentialgleichung ein Strudel sein.
Sei für die Vergleichsdifferentialgleichung die Pfaffsche Form $\omega_1 = 0$ angesetzt

$$\omega_1 = F_x dx + F_y dy = 0,$$

[14] Auf S.405 in [5] befindet sich eine Beweislücke, die jedoch nach [15] geschlossen werden kann.
[15] Vgl. [5] S.398ff

welche die Niveaulinien von F beschreibt. Für $y_1' = f(x,y)$ folgt

$$y_1' = f(x,y) := \frac{dy}{dx} = -\frac{F_x}{F_y}, \qquad F_y \neq 0 \tag{2.12}$$

und weiter

$$x_1' = \frac{1}{f(x,y)} = \frac{dx}{dy} = -\frac{F_y}{F_x}, \qquad F_x \neq 0. \tag{2.13}$$

Die Lösungen von x_1 bzw. y_1 bilden sicher geschlossene Kurven in einer Umgebung $G \setminus \{0\}$ [16]. Somit kann die Differenz für den Vergleich aus (2.9) und (2.12) gebildet werden

$$\begin{aligned} y' - y_1' &= -\frac{A(x,y)}{B(x,y)} - \left(-\frac{F_x(x,y)}{F_y(x,y)}\right) \\ &= \frac{-A(x,y)F_y(x,y) + B(x,y)F_x(x,y)}{B(x,y)F_y(x,y)}, \qquad (B(x,y)F_y(x,y)) \neq 0. \\ &=: \frac{Z(x,y)}{N(x,y)} \end{aligned} \tag{2.14}$$

bzw. analog für den Vergleich nach x aus (2.10) und (2.13)

$$\begin{aligned} x' - x_1' &= -\frac{B(x,y)}{A(x,y)} - \left(-\frac{F_y(x,y)}{F_x(x,y)}\right) \\ &= \frac{-B(x,y)F_x(x,y) + A(x,y)F_y(x,y)}{A(x,y)F_x(x,y)}, \qquad (A(x,y)F_x(x,y)) \neq 0. \end{aligned} \tag{2.15}$$

Mit der Notation der autonomen Systeme aus dem vorherigen Abschnitt bekommt man für Ausgangs- und Vergleichsdifferentialgleichung $\binom{\dot{x}}{\dot{y}} = \binom{B}{-A}$ und $\binom{\dot{x}_1}{\dot{y}_1} = \binom{F_y}{-F_x}$. Analog zu (2.14) und (2.15) kann man auch die darstellende Matrix dieser beiden Systeme heranziehen. Da in (2.14) und (2.15) lediglich der Zähler für die nähere Betrachtung relevant sein wird, ist dieser bis auf das Vorzeichen gleich mit der Determinante von

$$\begin{pmatrix} \dot{x} & \dot{x}_1 \\ \dot{y} & \dot{y}_1 \end{pmatrix} = \begin{pmatrix} B & F_y \\ -A & -F_x \end{pmatrix} \quad \Longrightarrow \quad \det \begin{pmatrix} B & F_y \\ -A & -F_x \end{pmatrix} = -BF_x + AF_y$$

Geometrisch bedeutet dies, daß die darstellenden Vektoren der Matrix linear abhängig sind, wenn die Determinante Null ist. Daraus folgt, daß Ausgangs- und Vergleichsdifferentialgleichung in einer gelochten Umgebung von Null dasselbe Richtungsfeld besitzen. Da die Vergleichsdifferentialgleichung bereits als Wirbeltyp vorausgesetzt wurde, muß für die Ausgangsdifferentialgleichung ebenfalls der Wirbeltyp folgen. Im Falle einer Determinante ungleich Null folgt der Strudeltyp.

Um dieses Kapitel zu beenden sei noch zur Invarianz von $Z(x,y)$ ein Satz gezeigt.

Satz 2.13. *Sei $Z(x,y)$ wie in (2.14) mit $Z(x,y) = -A(x,y)F_y(x,y) + B(x,y)F_x(x,y)$ und $Z(x,y) \neq 0$ in der gelochten Umgebung des Nullpunktes $G \setminus \{0\}$. Dann sind die Tangentialvektoren an die Integralkurven im Phasenraum durch $(x,y) \in G \setminus \{0\}$ linear unabhängig und umgekehrt.*

Beweis: Folgt unmittelbar aus dem vorherigen Abschnitt.

Im nächsten Kapitel soll ein Algorithmus vorgestellt werden, der y_1' bestimmt. Der Begriff der Strudelgröße wird dann das Kriterium dafür sein, ob ein Wirbel oder Strudel in der Ausgangsdifferentialgleichung vorliegt.

[16] [1] S.21f, Satz 3.2

3 Die Strudelgrößen und ihre rekursive Berechnung

Im vorigen Kapitel ist der Begriff der Vergleichsdifferentialgleichung bereits eingeführt worden. Dieses Kapitel zeigt nun ein Rekursionsverfahren, das die gesuchte Vergleichsdifferentialgleichung der Form (2.11) ermittelt. Die folgenden Darstellungen werden sich anfänglich im Wesentlichen derer aus [5] und [1] ähneln. In [5] und [1] ist diese Rekursion bis zu den maximalen homogenen Polynomen 6. respektive 8. Grades schon vorgestellt worden. Für eine vollständige Betrachtung beliebiger Differentialgleichungen der Form aus dem Poincaréschen Problem ist es jedoch nötig homogene Polynome $n$ten also beliebigen Grades berechnen zu können. Letztendlich sollen die Erörterungen aus [5] zu diesem Verfahren hier durch einen Hauptsatz abgeschlossen werden, der besagt, daß man die zur Konstruktion benötigten homogenen Polynome in beliebiger Ordnung berechnen kann.

Seien die Darstellungen für $y' = -\frac{A(x,y)}{B(x,y)}$ aus (2.8) und $y_1' = -\frac{F_x(x,y)}{F_y(x,y)}$ aus (2.11) wie im vorherigen Kapitel gegeben.

3.1 Ein Kriterium dafür, daß die Differentialgleichung $y' = -\frac{A(x,y)}{B(x,y)}$ nicht vom Wirbeltyp ist

Die Differenz aus Ausgangs- und Vergleichsdifferentialgleichung sei wie in (2.14)

$$\begin{aligned} y' - y_1' &= -\frac{A(x,y)}{B(x,y)} - \left(-\frac{F_x(x,y)}{F_y(x,y)}\right) \\ &=: \frac{-AF_y + BF_x}{BF_y} \quad =: \quad \frac{Z}{N} \end{aligned} \tag{3.1}$$

wobei Z und N als Abkürzungen für Zähler bzw. Nenner dienen sollen. Der Zählerterm Z läßt sich umformen zu

$$\begin{aligned} Z &= -AF_y + BF_x \\ &= \sum_{n=3}^{\infty} \underbrace{\sum_{i+j=n} Q_{ij}^{(n)} x^i y^j}_{=:Q^{(n)}(x,y)} \quad = \quad \sum_{n=3}^{\infty} Q^{(n)}(x,y) \quad =: \sum_{n=3}^{\infty} Q^{(n)} \end{aligned} \tag{3.2}$$

Die zu ermittelnden Koeffizienten sind die des Polynoms $F(x,y)$, dessen Ableitungen nach x und y mit A und B (3.1) bestimmen. Für das n-te homogene Polynom $F_n(x,y)$ vom Grad n $(n \geq 0)$ soll also gelten

$$F_n(x,y) = \sum_{i+j=n} A_{ij}^{(n)} x^i y^j, \qquad i,j = 0,\ldots,n$$

Die Koeffizienten $A_{ij}^{(n)}$ stehen dabei in keinerlei Beziehung zum Polynom $A(x,y)$. Die Rekursion erfolgt über den Laufparameter n. Für $n \leq 2$ folgt bereits aus (2.11)

$$F_0(x,y) = F_1(x,y) = 0,$$
$$F_2(x,y) = x^2 + y^2.$$

Die Rekursion kann also mit $n = 3$ analog wie in [5] beginnen. Zwei wesentliche Schritte sollen innerhalb eines Rekursionsschrittes erfolgen

1. C-Schritt: Ansetzen von

$$Q^{(n)} = -xF_{ny} + yF_{nx} + \sum_{k=2}^{n-1}(-p_k F_{(n-k+1)y} + q_k F_{(n-k+1)x}). \tag{3.3}$$

   Für F_n werden die gesuchten $A_{ij}^{(n)}$-Koeffizienten in $(xF_{ny} + yF_{nx})$ angegeben. Der C-Schritt führt die hintere Summation durch, und weist so jedem $C_{ij}^{(n)}$-Koeffizienten einen Wert zu, so daß gilt

$$\sum_{i+j=n} C_{ij}^{(n)} x^i y^j := \sum_{k=2}^{n-1}(-p_k F_{(n-k+1)y} + q_k F_{(n-k+1)x}), \qquad i,j = 0,\ldots,n \tag{3.4}$$

2. A-Schritt: Mit (3.4) ergibt sich für den $Q^{(n)}$-Term (3.3) nun die neue Darstellung

$$\begin{aligned} Q^{(n)} &= -xF_{ny} + yF_{nx} + \sum_{i+j=n} C_{ij}^{(n)} x^i y^j \\ &= \sum_{i+j=n} A_{ij}^{(n)} x^i y^j + \sum_{i+j=n} C_{ij}^{(n)} x^i y^j, \qquad i,j = 0,\ldots;n \end{aligned}$$

   Wenn $Q^{(n)}$ Null gesetzt wird, bedeutet das für die neuen Koeffizienten für F_n, daß sie sich ausschließlich aus den homogenen Teilpolynomen von A und B und denen aus früheren Schritten bereits ermittelten $(F_i)_{(i\leq n-1)}$ berechnen lassen.
   Für $Q^{(n)}$-Polynome soll folgendes gelten:

   - Im Falle n ungerade sollen die Koeffizienten von F_n so gewählt werden, daß $Q^{(n)}$ ein Nullpolynom wird. Daß man immer solche Koeffizienten finden kann, wird der Satz 3.2 zeigen.
   - Im Falle n gerade kann man für F_n immer Koeffizienten finden, so daß $Q^{(n)}$ bis auf die Koeffizienten von x^n und y^n immer zu einem Nullpolynom wird. Mit Hilfe einer Zusatzbedingung werden diese beiden noch fehlenden Koeffizienten gleich gemacht. Ihr gemeinsamer Wert ist die Strudelgröße, die, wenn sie ebenfalls Null ist, $Q^{(n)}$ zu einem Nullpolynom macht. Im anderen Fall ist sie das Kriterium dafür, daß die Ausgangsdifferentialgleichung vom Strudeltyp ist. Der Satz 3.3 zeigt, daß man für so ein $Q^{(n)}$ immer Koeffizienten finden kann.

   Die Ermittlung der Koeffizienten für F_n erfolgt durch Lösen eines linearen Gleichungssystems.

Nachdem diese Schritte erfolgt sind, kann von n nach $n+1$ übergegangen werden. Abgebrochen wird die Rekursion, wenn eine Strudelgröße verschieden von Null im A-Schritt vorliegt. Wenn keine solche Strudelgröße gefunden wird, muß die Rekursion unendlich oft durchgeführt werden.
Aus [14] §27 ist bereits bekannt, daß bei einer Differentialgleichung des Poincaréschen Problems mit endlichen Polynomen A und B bereits nach endlich vielen Strudelgrößen entscheidbar ist, ob ein Wirbel- oder Strudelfall vorliegt. Da zum Zeitpunkt dieser Arbeit noch keine feste Gesetzmäßigkeit existierte, mit welcher Strudelgröße bei Vorliegen einer Ausgangsdifferentialgleichung abgebrochen werden kann, sei hier noch angenommen, daß unendlich viele solcher Strudelgröße ermittelt werden müssen.

Das oben bereits Gesagte wird im folgenden Hauptsatz formuliert:

Satz 3.1 (Hauptsatz). *Sei $\epsilon > 0$. Zu jedem $n \in \mathbb{N}$, $n \geq 4$ und n gerade, existieren homogene Polynome $F_3(x,y), \ldots, F_n(x,y)$ vom Grad $3, \ldots, n$, derart daß mit*

$$F(x,y) = x^2 + y^2 + F_3(x,y) + \cdots + F_n(x,y)$$

der Ausdruck $(-AF_y + BF_x)$ eine in $|x| < \epsilon$, $|y| < \epsilon$ konvergente Potenzreihe

$$-AF_y + BF_x = \sum_{j=1}^{\frac{n}{2}-1} d_j \left(x^{2j+2} + y^{2j+2}\right) + (\textit{Terme höherer Ordnung } \geq (n+1))$$

ist. Insbesondere gilt dann die Darstellung als Taylorpolynom mit Restglied

$$\begin{aligned} -AF_y + BF_x &= \sum_{j=1}^{\frac{n}{2}-1} d_j \left(x^{2j+2} + y^{2j+2}\right) + \\ &\quad + \sum_{k+l=n+1} \frac{n+1}{k!l!} \left(\int_0^1 \frac{\partial^{k+l}}{\partial x^k \partial y^l}(0,0) \left(-AF_y + BF_x\right)(t_x, t_y)(1-t)^n dt \right) x^k y^l. \end{aligned}$$

Bei Übergang von n zu $n+2$ bleiben $F_3, \ldots, F_n$ ungeändert. Nur F_{n+1} und F_{n+2} werden neu bestimmt. d_j heißt die j-te Strudelgröße der Differentialgleichung $y' = -\frac{A}{B}$. Das Vorzeichen von d_j ist eindeutig bestimmt.

Beweis: Die Aussage folgt unmittelbar aus den Sätzen 3.2 und 3.3.

Die folgenden Abschnitte unterteilen sich in der Betrachtung der Q_n-Polynome für n ungerader (Abschnitt 3.2) bzw. gerader Ordnung (Abschnitt 3.3).

3.2 Eine Formel für Polynome ungerader Ordnung

Satz 3.2. *Sei $n \in \mathbb{N}$ ungerade und $n \geq 3$. $F_n(x,y)$ läßt sich so wählen, daß $Q_n(x,y) = 0$ ist.*

Beweis: Der Beweis ist erfolgt wenn die $A_{ij}^{(n)}$ sich durch die $C_{ij}^{(n)}$ eindeutig bestimmen lassen. Es gilt die folgende Matrizen-Darstellung für den Koeffizientenvergleich wie er schon in [5] vorgestellt wurde

$$L^{(n)} := \begin{pmatrix} 0 & -1 & & & \\ n & 0 & -2 & & 0 \\ & n-1 & 0 & \ddots & \\ & & n-2 & \ddots & -(n-1) \\ & & & \ddots & 0 & -n \\ & 0 & & & 1 & 0 \end{pmatrix} \tag{3.5}$$

$$A^{(n)} := \begin{pmatrix} A_{n,0} \\ A_{n-1,1} \\ A_{n-2,2} \\ \vdots \\ A_{2,n-2} \\ A_{1,n-1} \\ A_{0,n} \end{pmatrix}, \qquad C^{(n)} := \begin{pmatrix} C_{n,0} \\ C_{n-1,1} \\ C_{n-2,2} \\ \vdots \\ C_{2,n-2} \\ C_{1,n-1} \\ C_{0,n} \end{pmatrix} \tag{3.6}$$

Mit Hilfe von (3.5) und (3.6) läßt sich ein allgemeines Lösungsgleichungssystem aufstellen

$$L^{(n)} \cdot A^{(n)} = -C^{(n)} \qquad \Leftrightarrow \tag{3.7}$$

$$\Leftrightarrow \begin{pmatrix} 0 & -1 & & & & \\ n & 0 & -2 & & 0 & \\ & n-1 & 0 & \ddots & & \\ & & n-2 & \ddots & -(n-1) & \\ & & & \ddots & 0 & -n \\ & 0 & & & 1 & 0 \end{pmatrix} \begin{pmatrix} A_{n,0} \\ A_{n-1,1} \\ A_{n-2,2} \\ \vdots \\ A_{2,n-2} \\ A_{1,n-1} \\ A_{0,n} \end{pmatrix} = - \begin{pmatrix} C_{n,0} \\ C_{n-1,1} \\ C_{n-2,2} \\ \vdots \\ C_{2,n-2} \\ C_{1,n-1} \\ C_{0,n} \end{pmatrix}$$

$A^{(n)}$ repräsentiert den Lösungsvektor. $-(L^{(n)})^{-1} \cdot C^{(n)}$ ist die Lösung zu $A^{(n)}$ sofern $L^{(n)}$ invertierbar ist. Daß $L^{(n)}$ immer invertierbar ist für ungerades n wird nachfolgend noch gezeigt werden.
Die Koeffizienten-Matrix $L^{(n)}$ läßt sich zu einer Vier-Block-Matrix transformieren. Durch Verschieben jeder zweiten Zeile und jeder zweiten Spalte in die obere bzw. linke Hälfte der Matrix läßt sich der "Makel" der Null-Diagonalen aufheben. Diese Transformation führen zwei geeignete Permutationsmatrizen durch. Die Permutationsmatrix des Zeilentauschs

$$P_{ZT}^{(n)} = \begin{pmatrix} 1 & 0 & 0 & 0 & \cdots & & & 0 & \\ 0 & 0 & 1 & 0 & \cdots & & & 0 & \\ & & & & \ddots & & & & \\ & 0 & & & \cdots & 1 & 0 & 0 & 0 \\ & 0 & & & \cdots & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & \cdots & & & 0 & \\ 0 & 0 & 0 & 1 & \cdots & & & 0 & \\ & & & & \ddots & & & & \\ & 0 & & & \cdots & 0 & 1 & 0 & 0 \\ & 0 & & & \cdots & 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.8}$$

wird von links auf beiden Seiten des linearen Gleichungssystems heranmultipliziert. Die Permutationsmatrix des Spaltentauschs

$$P_{ST}^{(n)} = \begin{pmatrix} 0 & 0 & & & & 1 & 0 & & & \\ 1 & 0 & & 0 & 0 & 0 & 0 & & 0 & 0 \\ 0 & 0 & & & & 0 & 1 & & & \\ 0 & 1 & & & & 0 & 0 & & & \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ & & & 0 & 0 & & & & 1 & 0 \\ & & & 1 & 0 & & & & 0 & 0 \\ 0 & 0 & & 0 & 0 & 0 & 0 & & 0 & 1 \\ & & & 0 & 1 & & & & 0 & 0 \end{pmatrix} \tag{3.9}$$

wird zusammen mit ihrer Inversen[1] zwischen $L^{(n)}$ und $A^{(n)}$ sozusagen als "geschickte Einheitsmatrix" eingeschoben

$$\begin{aligned} P_{ZT}^{(n)} L^{(n)} P_{ST}^{(n)} (P_{ST}^{(n)})^{-1} A^{(n)} &= -P_{ZT}^{(n)} C^{(n)} \\ \underbrace{-P_{ZT}^{(n)} L^{(n)} P_{ST}^{(n)}}_{=:L'^{(n)}} (P_{ST}^{(n)})^{T} A^{(n)} &= P_{ZT}^{(n)} C^{(n)} \\ L'^{(n)} (P_{ST}^{(n)})^{T} A^{(n)} &= P_{ZT}^{(n)} C^{(n)} \end{aligned} \tag{3.10}$$

[1] Inverse von Permutationsmatrizen existieren immer da sie lediglich ihre Transponierten sind, wie aus der lin. Algebra bekannt. Einen exakten Beleg dieser Aussage gibt [18] auf S.21f.

wobei auf das Umdrehen des Vorzeichens an dieser Stelle hingewiesen sei.
Das Ergebnis der Transformation von $L^{(n)}$ in $L'^{(n)}$ ist

$$L'^{(n)} = \underbrace{\left(\begin{array}{ccccc|ccccc} 1 & 0 & & & & & & & & \\ -(n-1) & 3 & \ddots & & & & & & & \\ & -(n-3) & \ddots & 0 & & & & 0 & & \\ & & \ddots & n-2 & 0 & & & & & \\ & & & -2 & n & & & & & \\ \hline & & & & & -n & 2 & & & \\ & & & & & 0 & -(n-2) & \ddots & & \\ & & 0 & & & & 0 & \ddots & n-3 & \\ & & & & & & & \ddots & -3 & n-1 \\ & & & & & & & & 0 & -1 \end{array}\right)}_{=:\left(\begin{array}{c|c} L_1'^{(n)} & 0 \\ \hline 0 & L_2'^{(n)} \end{array}\right)}$$

Leicht abzulesen ist, daß $L'^{(n)}$ sich die Determinante durch Multiplikation der Diagonalelemente ergibt. Die Determinante ist mit ungeradem n immer ungleich Null, da die Diagonalelemente nie Null sind, was eine notwendige Bedingung für die eindeutige Lösbarkeit eines linearen quadratischen Gleichungssystems ist. Wie man aus der Matrix ebenfalls erkennen kann, sind die Koordinaten von $L_2'^{(n)}$ gleich denen von $L_1'^{(n)}$, wenn sie um 180^O gedreht und mit umgekehrten Vorzeichen versehen werden. Die Transformationen der Vektoren $A^{(n)}$ und $C^{(n)}$ ergeben

$$A'^{(n)} := (P_{ST}^{(n)})^T A^{(n)} = \left(\begin{array}{c} A_{n-1,1} \\ A_{n-3,3} \\ \vdots \\ A_{2,n-2} \\ \hline A_{0,n} \\ A_{n,0} \\ A_{n-2,2} \\ \vdots \\ A_{3,n-3} \\ A_{1,n-1} \end{array}\right) =: \left(\begin{array}{c} A_1'^{(n)} \\ \hline A_2'^{(n)} \end{array}\right) \tag{3.11}$$

$$C'^{(n)} := P_{ZT}^{(n)} C^{(n)} = \left(\begin{array}{c} C_{n,0} \\ C_{n-2,2} \\ \vdots \\ C_{3,n-3} \\ C_{1,n-1} \\ \hline C_{n-1,1} \\ C_{n-3,3} \\ \vdots \\ C_{2,n-2} \\ C_{0,n} \end{array}\right) =: \left(\begin{array}{c} C_1'^{(n)} \\ \hline C_2'^{(n)} \end{array}\right) \tag{3.12}$$

Das Problem hat sich also dahingehend vereinfacht, daß man zur Lösung eines $A_{ij}^{(n)}$ bereits die Lösung von $-A_{ji}^{(n)}$ ermittelt hat, indem man lediglich die Indizes in den $C^{(n)}$ vertauscht und das Vorzeichen

wechselt. Daher wird im folgenden nur noch das Teilproblem, $L_1^{(n)}$ auf Gestalt einer Einheitsmatrix zu bringen, betrachtet.
Durch Multiplizieren mit $(n-2i+1)$ und Addieren der i-ten Zeile auf die $(i+1)$-te und nachfolgendem Dividieren durch $(2i+1)$ in $L_1'^{(n)}$ erhält man die gewünschte Einheitsmatrix in $L_1'^{(n)}$. Die Anwendung auf der linken Seite des Gleichungssystems führt zur Einheitsmatrix, jedoch auf der rechten Seite zu einer neuen Komposition der $C_{ij}^{(n)}$ aus $C_1'^{(n)}$ in der jeweiligen Zeile

$$\begin{pmatrix} A_{n-1,1} \\ A_{n-3,3} \\ A_{n-5,5} \\ \vdots \\ A_{0,n} \end{pmatrix} = \begin{pmatrix} 1 & & & & & \\ \frac{(\frac{n-1}{1})\frac{1}{3}}{} & \frac{1}{3} & & & 0 & \\ (\frac{n-1}{1})(\frac{n-3}{3})\frac{1}{5} & (\frac{n-3}{3})\frac{1}{5} & \frac{1}{5} & & & \\ \vdots & \vdots & \vdots & \ddots & & \\ \prod_{j=0}^{m-1} \frac{n-2j-1}{2j+1}\frac{1}{(2m+1)} & b_{m1}^{(n)} & b_{m2}^{(n)} & \cdots & \frac{1}{(2m+1)} & \end{pmatrix} \begin{pmatrix} C_{n,0} \\ C_{n-2,2} \\ C_{n-4,4} \\ \vdots \\ C_{1,n-1} \end{pmatrix}, \tag{3.13}$$

wobei $m := \frac{n-1}{2}$ bezeichnen soll und für $b_{ij}^{(n)}$ das Produkt gilt

$$b_{ij}^{(n)} := \frac{1}{(2i+1)} \prod_{k=j}^{i-1} \frac{n-2k-1}{2k+1} \tag{3.14}$$

$b_{ij}^{(n)}$ gibt die Komponente der erzeugten Inversen in der j-ten Spalte und i-ten Zeile an $(i,j=0,\ldots,m,\ j<i)$.
Somit läßt sich die Lösung eines $A_{ij}^{(n)}$-Koeffizienten explizit aus einer Formeldarstellung angeben

$$A_{n-(2m+1),(2m+1)}^{(n)} = \sum_{i=0}^{m} \frac{C_{n-2i,2i}^{(n)}}{2m+1} \prod_{j=i}^{m-1} \frac{n-2j-1}{2j+1},$$

was für den Einsatz in einer Computer-Implementierung ein Vorteil ist, wie sich im nächsten Kapitel zeigt. Es ist schon angemerkt worden, daß sich die Lösung für $L_2^{(n)} A_2^{(n)} = C_2^{(n)}$ analog aus der Lösung des vorigen Gleichungssystems ergibt, jedoch unter der Berücksichtigung, daß $C_{ij}^{(n)}$ und $C_{ji}^{(n)}$ i.a. nicht identisch sind. Es gilt letztendlich für beide Fälle

$$A_{n-l,l}^{(n)} = \begin{cases} \sum_{i=0}^{\frac{l-1}{2}} \frac{C_{n-2i,2i}^{(n)}}{l} \prod_{j=i}^{\frac{l-3}{2}} \frac{n-2j-1}{2j+1}, & \text{für } l \text{ ungerade,} \\ -\sum_{i=0}^{\frac{n-l-1}{2}} \frac{C_{2i,n-2i}^{(n)}}{n-l} \prod_{j=i}^{\frac{n-l-3}{2}} \frac{n-2j-1}{2j+1}, & \text{für } l \text{ gerade} \end{cases} \tag{3.15}$$

Die Koeffizienten für $F_n(x,y)$ lassen sich also immer so wählen, daß $Q_n(x,y)=0$ gilt, wenn n ungerade ist. □

Im nächsten Abschnitt betrachten wir nun den Fall, daß für $Q_n(x,y)$ n gerade gilt.

3.3 Zwei Formeln für Polynome gerader Ordnung

Satz 3.3. *Sei $n \in \mathbb{N}, n>3$ gerade. $F_n(x,y)$ läßt sich so wählen, daß $Q_n(x,y) = d_{\frac{n}{2}-1}(x^n+y^n)$ gilt. $d_{\frac{n}{2}-1}$ ist dabei die $(\frac{n}{2}-1)$-te Strudelgröße aus Satz 3.1.*

Beweis: Der Beweis ist erfolgt, wenn die $A_{ij}^{(n)}$ sich durch die $C_{ij}^{(n)}$ in der gewünschten Weise bestimmen

lassen. Es gelten die Matrix und die Vektoren aus (3.5) und (3.6)

$$L^{(n)} := \begin{pmatrix} 0 & -1 & & & & \\ n & 0 & -2 & & 0 & \\ & n-1 & 0 & \ddots & & \\ & & n-2 & \ddots & -(n-1) & \\ & & & \ddots & 0 & -n \\ & 0 & & & 1 & 0 \end{pmatrix} \tag{3.16}$$

$$A^{(n)} := \begin{pmatrix} A_{n,0} \\ A_{n-1,1} \\ A_{n-2,2} \\ \vdots \\ A_{2,n-2} \\ A_{1,n-1} \\ A_{0,n} \end{pmatrix}, \qquad C^{(n)} := \begin{pmatrix} C_{n,0} \\ C_{n-1,1} \\ C_{n-2,2} \\ \vdots \\ C_{2,n-2} \\ C_{1,n-1} \\ C_{0,n} \end{pmatrix} \tag{3.17}$$

Im Unterschied zum Ansatz des vorigen Kapitels besitzt $L^{(n)}$ nun Zeilen- und Spaltenzahl ungerader Ordnung. Diese Eigenschaft führt dazu, daß sich eine eindeutige Lösung wie im Falle ungerader Ordnung nicht ohne weiteres finden läßt.
Zunächst soll $L^{(n)}$ wieder auf Block-Gestalt gebracht werden um den Lösungsweg zu vereinfachen. Dazu nutzt man wieder Permutationsmatrizen ähnlich denen aus (3.9) und (3.8). Die Permutationsmatrix des Zeilentauschs ergibt sich zu

$$P_{ZT}^{(n)} = \left(\begin{array}{ccccc|ccccc} 1 & 0 & 0 & 0 & \cdots & & & 0 & & \\ 0 & 0 & 1 & 0 & \cdots & & & 0 & & \\ & & & & \ddots & & & & & \\ & 0 & & & \cdots & 1 & 0 & 0 & 0 & 0 \\ & 0 & & & \cdots & 0 & 0 & 1 & 0 & 0 \\ \hline & 0 & & & \cdots & 0 & 0 & 0 & 0 & 1 \\ \hline 0 & 1 & 0 & 0 & \cdots & & & 0 & & \\ 0 & 0 & 0 & 1 & \cdots & & & 0 & & \\ & & & & \ddots & & & & & \\ & 0 & & & \cdots & 0 & 1 & 0 & 0 & 0 \\ & 0 & & & \cdots & 0 & 0 & 0 & 1 & 0 \end{array}\right) \tag{3.18}$$

Analog lautet die Permutationsmatrix des Spaltentauschs

$$P_{ST}^{(n)} = \left(\begin{array}{ccccccccccc|c} 0 & 0 & & & & & 1 & 0 & & & & \\ 1 & 0 & & 0 & 0 & 0 & 0 & & 0 & 0 & 0 \\ 0 & 0 & & & & & 0 & 1 & & & & \\ 0 & 1 & & & & & 0 & 0 & & & & \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & & \\ & & & 0 & 0 & & & & 1 & 0 & 0 \\ & & & 1 & 0 & & & & 0 & 0 & 0 \\ 0 & 0 & & 0 & 0 & 0 & 0 & & 0 & 1 & 0 \\ & & & 0 & 1 & & & & 0 & 0 & 0 \\ & & & 0 & 0 & & & & 0 & 0 & 1 \end{array}\right) \tag{3.19}$$

Die eingekasteten Spalten/Zeilen ergeben sich aufgrund des geraden n, woraus ein ungerader Rang der Permutationsmatrix folgt. Werden die Permutationsmatrizen wie in (3.10) angewendet auf das Glei-

chungssystem, so ergibt sich die gewünschte Block-Gestalt für $L'^{(n)} =$

$$\left(\begin{array}{ccccc|ccccc} 1 & 0 & & & & & & & & \\ -(n-1) & 3 & \ddots & & & & & & & \\ & -(n-3) & \ddots & 0 & & & & 0 & & \\ & & \ddots & n-3 & 0 & & & & & \\ & & & -3 & n-1 & & & & & \\ \hline & & & & -1 & \mathbf{0} & & & & \\ \hline & & & & & -n & 2 & & & \\ & & & & & 0 & -(n-2) & \ddots & & \\ & & 0 & & & & 0 & \ddots & n-2 & \\ & & & & & & & \ddots & -2 & n \end{array}\right) \tag{3.20}$$

Ebenso verändert sich die Gestalt der Vektoren $A'^{(n)}$ und $C'^{(n)}$ aus (3.11) zu

$$A'^{(n)} := (P_{ST}^{(n)})^T A^{(n)} = \left(\begin{array}{c} A_{n-1,1} \\ A_{n-3,3} \\ \vdots \\ A_{3,n-3} \\ A_{1,n-1} \\ \hline A_{n,0} \\ A_{n-2,2} \\ \vdots \\ A_{2,n-2} \\ A_{0,n} \end{array}\right) =: \left(\begin{array}{c} A_1'^{(n)} \\ \hline A_2'^{(n)} \end{array}\right) \tag{3.21}$$

$$C'^{(n)} := P_{ZT}^{(n)} C^{(n)} = \left(\begin{array}{c} C_{n,0} \\ C_{n-2,2} \\ \vdots \\ C_{2,n-2} \\ C_{0,n} \\ \hline C_{n-1,1} \\ C_{n-3,3} \\ \vdots \\ C_{3,n-3} \\ C_{1,n-1} \end{array}\right) =: \left(\begin{array}{c} C_1'^{(n)} \\ \hline C_2'^{(n)} \end{array}\right) \tag{3.22}$$

In (3.20) liefert die linke obere Block-Matrix, definiert als $L_1'^{(n)}$, und die rechte untere Block-Matrix, definiert als $L_2'^{(n)}$, keine eindeutige Lösung. Aufgrund der Null in der eingerahmten Zeile ist die Determinante der Matrix Null, d.h. das Gleichungssystem kann keine Inverse und somit keine eindeutige Lösung liefern. Während der $L_1'^{(n)}$-Teil des Gleichungssystems unter einer Überbestimmtheit "leidet", ist der $L_2'^{(n)}$-Teil mit seiner Unterbestimmtheit noch für das Gleichungssystem einer Lösung dienlich. Folgende Eingriffe führen daher zu einer Lösung des Problems

- für $L_1'^{(n)}$: Durch Ansetzen einer zusätzlichen Bedingung kann $L_1'^{(n)}$ wieder auf eine quadratische Matrix[2] und somit lösbar für ein lineares Gleichungssystem gemacht werden. Es soll analog zu Frommer angenommen werden, daß die Koeffizienten der x^n und y^n also $A_{n-1,1}^{(n)}$ und $A_{1,n-1}^{(n)}$ gleich sind.

[2]mit vollem Rang

- für $L_2^{\prime(n)}$: Für das Gleichungssystem wird lediglich eine Koordinate des Lösungsvektors in der transformierten Form $A_2^{\prime(n)}$ auf Null gesetzt. Ohne Einschränkung wird angenommen, daß $A_{0,n}^{(n)}$ gewählt wird. Diese Auswahl bringt beim Beweis Vereinfachungen mit sich.

Diese Ansätze führen in keinem Fall zu einer Einschränkung in der Ermittlung der Strudelgrößen. Frommer, und auch Birnmeyer, praktizierten diese Eingriffe ohne jegliche Einschränkung. Die Legitimation ist in [5] und [1] nachzulesen. Der Eingriff bedeutet für das lineare Gleichungssystem daß die eingerahmte Zeile in (3.20) auf linker wie auf rechter Seite von der ersten abgezogen und dann entfernt wird.
Die Gestalt der neuen Gleichungssysteme ergeben dann für $L_1^{\prime(n)} A_1^{\prime(n)} = C_1^{\prime(n)}$

$$\begin{pmatrix} 1 & & & 0 & 1 \\ -(n-1) & 3 & & & 0 \\ & -(n-3) & \ddots & & \vdots \\ & & \ddots & n-3 & 0 \\ 0 & & & -3 & n-1 \end{pmatrix} \begin{pmatrix} A_{n-1,1} \\ A_{n-3,3} \\ \vdots \\ A_{3,n-3} \\ A_{1,n-1} \end{pmatrix} =$$

$$= \begin{pmatrix} 1 & & 0 & -1 \\ & 1 & & 0 \\ & & \ddots & \vdots \\ 0 & & 1 & 0 \end{pmatrix} \begin{pmatrix} C_{n,0} \\ C_{n-2,2} \\ \vdots \\ C_{2,n-2} \\ C_{0,n} \end{pmatrix} \tag{3.23}$$

und für $L_2^{\prime(n)} A_2^{\prime(n)} = C_2^{\prime(n)}$

$$\begin{pmatrix} -n & 2 & & 0 & 0 \\ & -(n-2) & \ddots & & \vdots \\ & & \ddots & n-2 & 0 \\ 0 & & & -2 & 0 \end{pmatrix} \begin{pmatrix} A_{n,0} \\ A_{n-2,2} \\ \vdots \\ A_{2,n-2} \\ A_{0,n} \end{pmatrix} = \begin{pmatrix} C_{n-1,1} \\ C_{n-3,3} \\ \vdots \\ C_{3,n-3} \\ C_{1,n-1} \end{pmatrix} \tag{3.24}$$

Das zweite Gleichungssystem (3.24) wurde in ähnlicher Weise bereits in (3.13) gelöst. Somit kann im wesentlichen die Formeldarstellung aus (3.15) als Lösung übernommen werden, wenn berücksichtigt wird, daß der Koeffizient $A_{0,n}^{(n)}$ bereits mit Null vorbelegt wurde

$$A_{n-l,l}^{(n)} = -\sum_{i=1}^{\frac{n-l}{2}} \frac{C_{2i-1,n-(2i-1)}^{(n)}}{n-l} \prod_{j=i}^{\frac{n-l-2}{2}} \frac{n-2j}{2j}, \qquad \text{für } l > 1 \text{ gerade, (l=0,2,...,n-2).} \tag{3.25}$$

Als letztes bleibt nur noch zu zeigen, daß (3.23) eine Lösung besitzt. Es wird eine Hilfsvariable definiert

$$a_i^{(n)} := \prod_{k=1}^{i} \frac{n-(2k-1)}{2k-1}, \qquad \text{für } 0 \le i \le 2m. \tag{3.26}$$

Es sei

$$a_i^{(n)} = \begin{cases} 1 & \text{für } i = 0 \\ \prod_{k=1}^{i} \frac{n-(2k-1)}{2k-1}, & \text{für } 1 \le i \le m \\ a_{\frac{n}{2}-i}^{(n)}, & \text{für } m < i \le (2m+1) \end{cases}$$

wobei $m = \lfloor \frac{n}{4} \rfloor$ gilt und die Symmetrieeigenschaft der Produktterme ausgenutzt wurde[3]. Im folgenden wird stets a_i an Stelle von $a_i^{(n)}$ verwendet, wenn dies zu keinen Mißverständnissen führt. Durch

[3] leichtes Nachrechnen bestätigt, daß $a_i = a_{\frac{n}{2}-1-i}$ ist.

Abwärtselimination erhalten wir für $L_1'^{(n)}$ und $C_1'^{(n)}$ die folgende Gestalt

$$\left(\begin{array}{ccccc} 1 & & 0 & & \frac{a_0}{1} \\ & 1 & & & \frac{a_1}{3} \\ & & \ddots & & \vdots \\ \hline & & 1 & & \frac{a_m}{2m+1} \\ \hline & & & 1 & \frac{a_{m-1}}{2(m-1)+1} \\ & & & \ddots & \vdots \\ 0 & & & 1 & \frac{a_1}{3} \\ & & & & 1+\frac{a_0}{1} \end{array}\right) \left(\begin{array}{c} A_{n-1,1} \\ A_{n-3,3} \\ \vdots \\ A_{3,n-3} \\ A_{1,n-1} \end{array}\right) =$$

$$= \left(\begin{array}{ccccccc} a_0 & & & & & & \\ \frac{a_1}{3a_0} & \frac{a_1}{3a_1} & & & & & 0 \\ \vdots & \vdots & \ddots & & & & \\ \hline \frac{a_m}{a_0(2m+1)} & \frac{a_m}{a_1(2m+1)} & \cdots & \frac{a_m}{a_m(2m+1)} & & & \\ \hline \frac{a_{m-1}}{a_0(2(m-1)+1)} & \frac{a_{m-1}}{a_1(2(m-1)+1)} & \cdots & \frac{a_{m-1}}{a_m(2(m-1)+1)} & \frac{a_{m-1}}{a_{m-1}(2(m-1)+1)} & & \\ \vdots & \vdots & & \vdots & \vdots & \ddots & \\ \frac{a_1}{3a_0} & \frac{a_1}{3a_1} & \cdots & \frac{a_1}{3a_m} & \frac{a_1}{3a_{m-1}} & \cdots & \frac{a_1}{3a_2} \\ a_0 & \frac{a_0}{a_1} & \cdots & \frac{a_0}{a_m} & \frac{a_0}{a_{m-1}} & \cdots & \frac{a_0}{a_2} \end{array}\right. \left.\begin{array}{c} \\ \\ \\ \\ \\ \\ \frac{a_0}{a_1} \end{array}\right) \cdot$$

$$\cdot \left(\begin{array}{c} C_{n,0}-C_{0,n} \\ C_{n-2,2} \\ \vdots \\ C_{4,n-4} \\ C_{2,n-2} \end{array}\right) \qquad (3.27)$$

Um das Vorwärtslösen abzuschließen muß noch die letzte Zeile durch zwei dividiert werden, woraus folgt daß der erste Koeffizient $A_{1,n-1}^{(n)}$ bereits ermittelt ist. Das Rückwärtseinsetzen eliminiert die letzte Spalte aus $L_1'^{(n)}$ und liefert das Endergebnis.
Sei im folgenden also nur noch die Koeffizientenmatrix der $C_1'^{(n)}$ dargestellt

$$\left(\begin{array}{ccccccc} \frac{a_0}{2a_0} & -\frac{a_0}{2a_1} & \cdots & -\frac{a_0}{2a_m} & -\frac{a_0}{2a_{m-1}} & \cdots & -\frac{a_0}{2a_2} & -\frac{a_0}{2a_1} \\ \frac{a_1}{2\cdot 3a_0} & \frac{a_1}{2\cdot 3a_1} & \cdots & -\frac{a_1}{2\cdot 3a_m} & -\frac{a_1}{2\cdot 3a_{m-1}} & \cdots & -\frac{a_1}{2\cdot 3a_2} & -\frac{a_1}{2\cdot 3a_1} \\ \vdots & \vdots & \ddots & \vdots & \vdots & & \vdots & \vdots \\ \frac{a_{m-1}}{2a_0(2m-1)} & \frac{a_{m-1}}{2a_1(2m-1)} & \cdots & -\frac{a_{m-1}}{2a_m(2m-1)} & -\frac{a_{m-1}}{2a_{m-1}(2m-1)} & \cdots & -\frac{a_{m-1}}{2a_1(2m-1)} & -\frac{a_{m-1}}{2a_0(2m-1)} \\ \hline \frac{a_m}{2a_0(2m+1)} & \frac{a_m}{2a_1(2m+1)} & \cdots & \frac{a_m}{2a_m(2m+1)} & -\frac{a_m}{2a_m(2m+1)} & \cdots & -\frac{a_m}{2a_1(2m+1)} & -\frac{a_m}{2a_0(2m+1)} \\ \hline \frac{a_{m-1}}{2a_0(2m-1)} & \frac{a_{m-1}}{2a_1(2m-1)} & \cdots & \frac{a_{m-1}}{2a_m(2m-1)} & \frac{a_{m-1}}{2a_{m-1}(2m-1)} & \cdots & -\frac{a_{m-1}}{2a_1(2m-1)} & -\frac{a_{m-1}}{2a_0(2m-1)} \\ \vdots & \vdots & & \vdots & \vdots & \ddots & \vdots & \vdots \\ \frac{a_1}{2\cdot 3a_0} & \frac{a_1}{2\cdot 3a_1} & \cdots & \frac{a_1}{2\cdot 3a_m} & \frac{a_1}{2\cdot 3a_{m-1}} & \cdots & \frac{a_1}{2\cdot 3a_2} & -\frac{a_1}{2\cdot 3a_1} \\ \frac{a_0}{2a_0} & \frac{a_0}{2a_1} & \cdots & \frac{a_0}{2a_m} & \frac{a_0}{2a_{m-1}} & \cdots & \frac{a_0}{2a_2} & \frac{a_0}{2a_1} \end{array}\right) \qquad (3.28)$$

Dabei ist zu beachten, daß die eingerahmten Zeilen[4] nur dann in (3.27) und (3.28) existieren, wenn n nicht durch 4 teilbar ist.
Die Lösung läßt sich also in die Formeldarstellung bringen

$$A_{n-l,l}^{(n)} = \frac{1}{2} \cdot \frac{a_{\min\{\frac{l-1}{2},\frac{n-l-1}{2}\}}}{\min\{l, n-l\}} \cdot \left(\sum_{i=0}^{\frac{l-1}{2}} \frac{C_{n-2i,2i}^{(n)}}{a_i} - \sum_{j=\frac{l-1}{2}+1}^{\frac{n}{2}} \frac{C_{n-2j,2j}^{(n)}}{a_j} \right), \qquad (3.29)$$

[4] $(m+1)$-te Zeile; Zählung bei eins begonnen.

$$\text{für } l \text{ ungerade, } (l = 1, 3, \ldots, n-1).$$

Die Strudelgröße $d_{\frac{n}{2}-1}$ definiert sich (unter Berücksichtigung der bereits erfolgten Definition in Satz 3.1) analog zu Frommer wie folgt

$$d_{\frac{n}{2}-1} := -A^{(n)}_{n-1,1} + C^{(n)}_{n,0} = A^{(n)}_{1,n-1} + C^{(n)}_{0,n}$$

Frommer verstand unter der Strudelgröße ein Aufaddieren aller $C^{(n)}_{ij}$ $(i, j$ gerade) mit ihren jeweiligen Vorkoeffizienten, abzulesen aus der letzten bzw. ersten Zeile aus (3.28)

$$d_{\frac{n}{2}-1} = \frac{1}{2} \sum_{i=0}^{\frac{n}{2}} \frac{C^{(n)}_{n-2i,2i}}{a_i}, \qquad \text{für } n \text{ gerade, } (n \geq 4). \tag{3.30}$$

□

Nun sind also alle Formeln bereitsgestellt um das rekursive Verfahren zu implementieren. Im nachfolgenden Kapitel sind nun (3.15), (3.25), (3.29) und zusätzlich (3.26) zur Berechnung der Koeffizienten im A-Schritt und (3.30) als Abbruchbedingung für den Algorithmus aus dem einführenden Abschnitt dieses Kapitels realisiert.

4 Eine einfache Realisierung unter JAVA im Fall von Polynomen A und B

Im vorhergehenden Kapitel wurden die benötigten Hilfsmittel für eine Computer-Implementierung zur Berechnung der Strudelgrößen bereits erörtert. Dieses Kapitel enthält eine reale Umsetzung der Formeln (3.15), (3.25), (3.29) und (3.26) in einem rekursiven Algorithmus.

In einer praktischen Realisierung wie z.B. unter JAVA stößt man auf das für rekursive Algorithmen typische Problem der Fehlerfortpflanzung. Wie sich dieser Fehler bildet, kann man an den Beispielen im nächsten Kapitel sehen. Die theoretischen Grundlagen sowie auch Möglichkeiten zur Verbesserung sollen dabei am Ende dieses Kapitels dargelegt werden.

4.1 Der Algorithmus

Der Algorithmus zur Berechnung der Strudelgrößen aus Abschnitt 3.1 wird nochmals als Struktogramm in Abb. 4.1 dargestellt. Aus Satz 3.1 und im Ablauf ist dabei ersichtlich, daß grundsätzlich unendlich viele Schritte ausgeführt werden müßten um einen Wirbelfall feststellen zu können. Bei der praktischen Realisierung gibt es jedoch die obere Iterationsschranke *nmax*.

Die Praxis hat gezeigt, daß immer der Algorithmus abbricht, wenn nicht-triviale Polynome für $A(x,y)$ und $B(x,y)$ vorliegen. Der Grund dafür liegt in der Fehlerfortpflanzung früherer Ungenauigkeiten in den Koeffizienten des berechneten $F(x,y)$. Im Abschnitt über das Konvergenzverhalten sei darüber mehr berichtet. In diesem Abschnitt werden die wesentlichen Teile des Algorithmus unter JAVA genauer betrachtet. Der Algorithmus wurde unter JAVA in der Routine `doAlgo(DGL a)` realisiert. Übergeben wird dabei eine Differentialgleichung, die in einem Objekt der Klasse `DGL` alle Informationen über die Differentialgleichung enthält. Wichtige Informationen sind dabei die beiden Polynome $A(x,y)$ und $B(x,y)$. Zudem wird die obere Iterationsschranke `maxiter` und die untere Schranke der Strudelgrößen `mineps` übergeben.

```
public String doAlgo(DGL a, int maxiter, double mineps)
   {
   String temp=new String();
   Polynom2DDiag p=Polynom2DDiag.make(a.A), q=Polynom2DDiag.make(a.B);

   double sg=0;
   int xmin=p.getFirst().getDegree(), ymin=q.getFirst().getDegree();
   int n=Math.max(2,Math.min(xmin,ymin));
   Polynom2DDiag F=new Polynom2DDiag();

   do
      {
   // Teil der Initialisierung:
   // Für n==2 wird F_2=x^2+y^2 gesetzt.
```

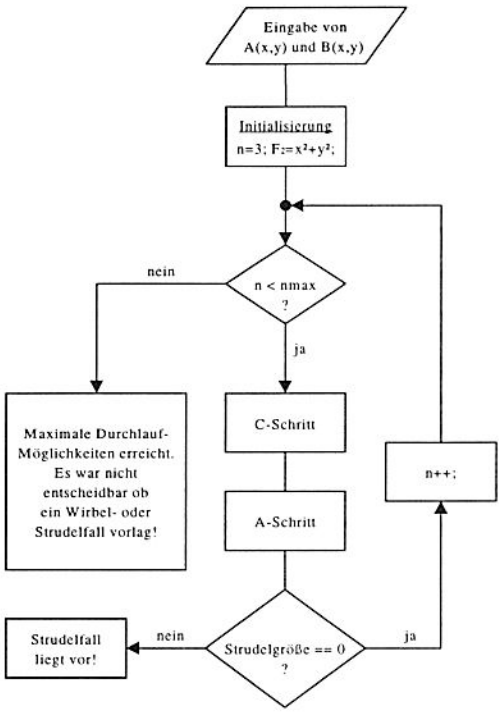

Abbildung 4.1: Struktogramm des Algorithmus aus Abschnitt 3.1.

- n ist die Iterations- oder Laufvariable. Im Poincaréschen Fall wird n immer mit 2 vorbelegt.

Im Falle, daß die Differentialgleichung a kein Poincarésches Problem beschreibt, somit also der lineare Anteil fehlt, wird mit n als der niedrigsten gemeinsamen und ungeraden Potenz von x bzw. y der Polynome $A(x,y)$ und $B(x,y)$ gestartet. Somit ist also schon die Möglichkeit, Differentialgleichungen anderer Klassen als die des Poincareschen Problems zu analysieren, realisiert. Die Zeilen

```
[...]
// Teil der Initialisierung:
// Für n==2 wird F_2=x^2+y^2 gesetzt.
   if (xmin+1==n) F.add(n,0,1);
   if (ymin+1==n) F.add(0,n,1);
[...]
```

stellen sicher, daß das Startpolynom von F mit jeweils dem niedrigsten Grad (**plus eins**) initialisiert wird.
Nachdem n um eins auf 3 heraufgezählt wurde, ist der eigentliche Startwert für n erreicht und die Initialisierung ist abgeschlossen.
Die nächsten Abschnitte zeigen nun die einzelnen Schritte des Algorithmus.

```
        if (xmin+1==n) F.add(n,0,1);
        if (ymin+1==n) F.add(0,n,1);

    // n heraufzählen
        n++;

    // C-Schritt: Neue C-Koeffizienten berechnen
        Polynom2D grossC= makeGrossCs(F, p, q, n);
        temp+=n+". C-Schritt: "+grossC+"\n";

    // A-Schritt: neue A-Koeffizienten berechnen
        Polynom2D grossA=makeGrossAs(n, grossC);
        temp+=n+". A-Schritt: "+grossA+"\n";

    // Strudelgroesse berechnen wenn n durch 2 teilbar ist.
        if (n%2==0)
            {
            sg=getStrudelgroesse(n, grossC);
            temp+=n+".: "+(n/2-1)+". Strudelgroesse: "+sg+"\n";
            }

    // Die berechneten neuen Koeffizienten aus F_n werden kopiert.
        for (int i=0; i<=n; i++)
            F.set(i, n-i, grossA.get(i,n-i));

    // der Algorithmus bricht ab, wenn die maximale Anzahl
    // an Iterationen erreicht, oder eine Strudelgrösse
    // berechnet wurde, die nicht mehr so klein ist, dass sie
    // als Null angesehen wird.
        } while (n<maxiter && Math.abs(sg)<mineps );

    return temp;
    }
```

4.1.1 Initialisierung

Die Initialisierung setzt folgende Variablen auf ihre Startwerte

- `temp` ist der Text-`String` der Kommentare und Zwischenergebnisse als Text aufnimmt und am Ende der Iterationen zurückgegeben wird.
- `p, q` sind verkettete Listen von homogenen Polynomen. Es wird mit `Polynom2DDiag.make(a.A)` die Klassenmethode `make` mit dem Parameter `A` für das Zähler-Polynom der Differentialgleichung `a` aufgerufen, die eine Liste von homogenen Polynomen, angeordnet nach ihrem Grad, erzeugt. Analog eerfolgt der Schritt mit `B`.
- `sg` enthält die Strudelgröße jedes zweiten Iterationsschrittes. Ist der absolute Wert von `sg` nicht mehr klein genug, bricht der Algorithmus ab. Es wird dann eine Strudelgröße angenommen, die nicht Null ist.
- `xmin, ymin` geben die niedrigsten Potenzen der Listen der homogenen Polynome aus `p, q` an. Im Poincaréschen Fall sind `xmin, ymin` immer 2.

4.1.2 Implementierung der Schritte

Die einzelnen Schritte teilen sich in drei Hauptaufgaben auf. Nachdem im C-Schritt die $C_{ij}^{(n)}$-Koeffizienten bereitstehen, kann im Falle geraden ns sofort die Strudelgröße berechnet werden. Die Berechnung der $A_{ij}^{(n)}$-Koeffizienten erfolgen dann direkt auf Basis der $C_{ij}^{(n)}$-Koeffizienten. Der dritte Schritt ist dann die Berechnung der Strudelgröße, sofern der Iterationsschritt n gerade ist.

C-Schritt

Die Realisierung des C-Schrittes wurde in die Routine `makeGrossCs` ausgelagert, wie aus den Zeilen im Algorithmus ersichtlich.

```
[...]
// C-Schritt: Neue C-Koeffizienten berechnen
   Polynom2D grossC= makeGrossCs(F, p, q, n);
   temp+=n+". C-Schritt: "+grossC+"\n";
[...]
```

Der Routine `makeGrossCs` werden die schon aus früheren Schritten berechneten $(F_i)_{i<n}$ und die Listen der homogenen Polynome aus A und B in Form von `p, q` übergeben. `n` gibt den homogenen Grad des zu berechnenden neuen Polynoms an. Zurückgegeben wird ein neues homogenes Polynom vom Grad `n`. Die Implementierung lautet

```
public Polynom2D makeGrossCs(Polynom2DDiag F, Polynom2DDiag p, Polynom2DDiag q, int n)
   {
   if (n<3) return null;
   Polynom2D grossC=new Polynom2D();

   for (int i=2; i<n; i++)
      {
      Polynom2DDiag f=F.getAtDegree(n-i+1);
      Polynom2D fx=null, fy=null;
      if (f!=null)
         {
         fx=f.getPolynom().Dx();
         fy=f.getPolynom().Dy();
         }
      grossC.sub(Polynom2D.multipliziere(p.getAtDegreePol(i), fy));
      grossC.add(Polynom2D.multipliziere(q.getAtDegreePol(i), fx));
      }
   return grossC;
   }
```

Zu Beginn wird überprüft ob der zu berechnende Grad `n` mindestens 3 ist. Dann wird ein leeres Nullpolynom erzeugt, das mit den neuen Koeffizienten "gefüllt" wird. Ein beliebiges Feld hätte genügt. Die Klasse `Polynom2D` besitzt jedoch besondere Methoden zum Herauslesen der Koeffizienten.
Über die Laufvariable `i` werden die benötigten Teilableitungen nach x und y von den homogenen Polynomen F_{n-i+1} mit den homogenen Polynomen aus `p, q` entsprechend multipliziert und voneinander abgezogen. Daß für jedes `i` die Ableitungen der homogenen Polynome von F berechnet werden, weist noch keinen redundanten Aufwand auf. Daß jedoch für jeden neuen Aufruf von `makeGrossCs` diese Ableitungen neu berechneten werden, obwohl sie gespeichert werden könnten, zeigt dies eine Optimierungsmöglichkeit, die bei der Berechnung des Aufwands im nächsten Abschnitt Berücksichtigung findet. Letztendlich kann diese Realisierung unter Berücksichtigung geringen Speicherplatzes als optimal angesehen werden.

```
    strudelgroesse+=grossC.get(n-2*i, 2*i)/getKleinA(n, i);
return strudelgroesse/2.0;
}
```

n gibt dabei an, daß die $(\frac{n}{2}-1)$-te Strudelgröße berechnet werden soll. Nachdem überprüft wurde, daß `grossC` kein Null-Objekt ist, n durch 2 teilbar und größer 3 ist wird die Zwischenergebnis-Variable `temp` mit Null vorbesetzt. Danach werden über die Laufvariable `i` alle C-Koeffizienten aufsummiert, gemäß (3.30), und zurückgegeben.

4.1.3 Abbruchbedingungen

Als letzten Unterabschnitt werden noch die Abbruchbedingungen aus `doAlgo` kommentiert.

```
[...]
// der Algorithmus bricht ab, wenn die maximale Anzahl
// an Iterationen erreicht, oder eine Strudelgrösse
// berechnet wurde, die nicht mehr so klein ist, dass sie
// als Null angesehen wird.
   } while (n<maxiter && Math.abs(sg)<mineps );
[...]
```

Zwei Abbruchbedingungen sind dabei vorhanden

- `maxiter` als obere Iterationsschranke:
  Die Anzahl der maximalen Iterationsschritte ist mit 100 bei Polynomen bis 20-sten als höchstem Grades völlig ausreichend. Bis 5-ten als höchstem Grad ist sogar eine Schranke von 10 genügend.
- `mineps` als untere Genauigkeitsschranke der Strudelgrößen:
  Mit der unteren Schranke der Strudelgrößen von 0.0001 konnte der Algorithmus immer genügend Auskunft über ein Strudel- oder Wirbelverhalten angeben.

Die Parameter werden durch die Übergabe an `doAlgo` gesetzt und sind somit variabel ansetzbar.

4.2 Komplexität und Aufwand

Im Anhang B können einige hier verwendete Definitionen wie z.B. die Landausymbole nachgelesen werden. Für die Betrachtung der Komplexität wird sich hier auf die Komplexität im schlechtesten Fall beschränkt. Für den Speicherbedarf muß eine Kodierungslänge angegeben werden die genau eine dargestellte Zahl bzw. einen Koeffizienten bemißt. Sie gilt dann als Faktor für die Summe aller benötigten Koeffizienten im Algorithmus. Das Ergbnis wurde bereits als Laufzeit eines Algorithmus im Anhang B definiert (Definition B.3).

4.2.1 Speicherbedarf und Zeitaufwand

Für den Speicherbedarf seien die benötigten Daten in folgender Weise aufgezählt

- Die Eingabedaten sind die Koeffizienten der Anfangsdifferentialgleichung. Sie sind in den Polynomen $A(x,y)$ und $B(x,y)$ abgelegt (siehe (2.8)).
- Die Koeffizienten des "Design"-$F(x,y)$ aus (2.11) wird zur Laufzeit gebildet. Von diesem $F(x,y)$ werden die Ableitungen F_x und F_y benötigt.

A-Schritt

Im A-Schritt werden alle benötigten A-Koeffizienten, aufgerufen in `doAlgo`, durch

```
[...]
// A-Schritt: neue A-Koeffizienten berechnen
   Polynom2D grossA=makeGrossAs(n, grossC);
   temp+=n+". A-Schritt: "+grossA+"\n";
[...]
```

in der Routine `makeGrossAs` berechnet. Die Lösungen werden abgelegt in einem `Polynom2D`-Objekt, und analog zum C-Schritt zurückgegeben.

```
public Polynom2D makeGrossAs(int n, Polynom2D grossC)
   {
   if (grossC==null) return null;
   Polynom2D grossA=new Polynom2D();

   for (int l=0; l<=n; l++)
      grossA.set(n-l, l, getGrossA(n, l, grossC));

   return grossA;
   }
```

Während im C-Schritt Polynome miteinander multipliziert und danach addiert respektive voneinander subtrahiert werden konnten, sind im A-Schritt die Berechnungen jedes A-Koeffizienten nach (3.15), (3.25) und (3.29) in etwas komplexerer Weise durchgeführt. `getGrossA` realisiert diese Berechnung in

```
public double getGrossA(int n, int l, Polynom2D grossC)
   {
   if (grossC==null || n<2 || l>n) return 0;
   double temp=0, temp1=0;
   int lmax=(l-1)/2;
   int nlmax=(n-l-1)/2;
   if (n%2==0)
      {
      if (l%2==1)
         {
         for (int i=0; i<=lmax; i++)
            temp+=grossC.get(n-2*i, 2*i)/getKleinA(n, i);
         for (int i=lmax+1; i<=n/2; i++)
            temp-=grossC.get(n-2*i, 2*i)/getKleinA(n, i);
         return temp*getKleinA(n, (l<n-l ? l: n-l)/2)*0.5/(double)(l<n-l ? l: n-l);
         }
      else
         {
         if (n-l==0) return 0;
         for (int i=1; i<=nlmax+1; i++)
            {
            temp1=1.0;
            for (int j=i; j<=nlmax; j++) temp1*=(double)(n-2*j)/(double)(2*j);
            temp+=grossC.get(2*i-1,n-(2*i-1))*temp1/(double)(n-l);
            }
         return -temp;
```

```
            }
         }
      else
         {
         if (l%2!=0)
            {
            for (int i=0; i<=lmax; i++)
               {
               temp1=1.0;
               for (int j=i; j<=lmax-1; j++) temp1*=(double)(n-2*j-1)/(double)(2*j+1);
               temp+=grossC.get(n-2*i,2*i)*temp1;
               }
            temp/=(double)l;
            }
         else
            {
            for (int i=0; i<=(n-l-1)/2; i++)
               {
               temp1=1.0;
               for (int j=i; j<=(n-l-1)/2-1; j++) temp1*=(double)(n-2*j-1)/(double)(2*j+1);
               temp+=grossC.get(2*i,n-2*i)*temp1;
               }
            temp/=-(double)(n-l);
            }
         }
      return temp;
      }
```

`kleinA` berechnet dabei den Hilfskoeffizienten aus (3.26). Die jeweiligen vier Fälle richten sich danach, ob `n` gerade oder ungerade bzw. ob `l` gerade oder ungerade ist.

Die Strudelgröße

Nachdem nun die $C_{ij}^{(n)}$-Koeffizienten bereitstehen, könnte eine Strudelgröße berechnet werden. In `doAlgo` wird mit den Zeilen

```
      [...]
      // Strudelgroesse berechnen wenn n durch 2 teilbar ist.
         if (n%2==0)
            {
            sg=getStrudelgroesse(n, grossC);
            temp+=n+".: "+(n/2-1)+". Strudelgroesse: "+sg+"\n";
            }
      [...]
```

überprüft ob eine Strudelgröße für gegebenes n berechnet werden kann. Auch hier wurde die Berechnung, obwohl sehr einfach, ausgelagert in `getStrudelgroesse`

```
   public double getStrudelgroesse(int n, Polynom2D grossC)
      {
      if (grossC==null || n%2==1 || n<3) return 0;
      double strudelgroesse=0;
      for (int i=0; i<=n/2; i++)
```

- Im C-Schritt werden zwei Polynom-Multiplikationen und eine nachfolgende Subtraktion benötigt. Das Ergebnis ist ein homogenes Polynom vom Grade des Iterationsschrittes.
- Im A-Schritt wird ein homogenes Polynom vom Grade des Iterationsschrittes erzeugt, das an das $F(x,y)$ additiv angefügt wird.

Ein paar Hilfskoeffizienten und Notationen seien noch eigeführt um die Berchnung der Komplexität zu erleichtern.

Definition 4.1. *Sei $n \in \mathbb{N}$ eine ganze Zahl. Dann sei unter $n!!$ das Produkt verstanden*

$$n!! := \begin{cases} 1 \cdot 3 \cdot \cdots \cdot (n-2) \cdot n, & \text{für } n \text{ ungerade,} \\ 2 \cdot 4 \cdot \cdots \cdot (n-2) \cdot n, & \text{für } n \text{ gerade.} \end{cases}$$

Somit ließe sich für (3.14) mit $i, j \in \mathbb{N}$: $i \neq 0$, $j \leq i$, schreiben

$$b_{ij}^{(n)} = \frac{1}{(2i+1)} \cdot \frac{(n-2j-1)!!}{(n-(2j+2i+1))!!(2i+1)!!}$$

Definition 4.2 (niedrigster und höchster homogener Grad.). *Sei P ein Polynom beliebigen Grades. Mit $Gr(P)$ sei der höchste homogene Polynomgrad des Polynoms P bezeichnet und mit $gr(P)$ der kleinste.*

Stets soll immer $\mathrm{gr}(A) = 1$ und $\mathrm{gr}(B) = 1$ gelten, da grundsätzlich Poincarésche Probleme betrachtet werden sollen. Für das $F(x,y)$ folgt dann immer $\mathrm{gr}(F) = 2$. Auch soll angenommen sein, daß $\mathrm{Gr}(A) < \infty$ und $\mathrm{Gr}(B) < \infty$ gilt.

Weiterhin sei M die Menge der verschiedenen homogenen Grade eines Polynoms.

$$\begin{aligned} M_P \quad := \{ i \mid \; & p_i \text{ ist homogenes Polynom von } P \text{ vom Grad } i, \\ & p_i \text{ hat mindenstens einen Koeffizienten, der nicht Null ist.} \}. \end{aligned}$$

Somit gilt also für ein Polynom P

$$\mathrm{gr}(P) = \min M_P, \qquad \mathrm{Gr}(P) = \max M_P$$

Die Mächtigkeit der Menge M_P ist die Anzahl der verschiedenen homogenen Grade des Polynoms P, bezeichnet mit $|M_P|$.

Die Komplexität des Gesamt-Algorithmus berechnet sich aus der Komplexität der einzelnen Schritte. Im schlechtesten Fall sind für jedes homogene Polynom vom Grad i gerade $(i+1)$ Koeffizienten nicht Null. Dieses sei der Vereinfachung der Rechnungen wegen stets angenommen. n sei der betrachtete Iterationsschritt.

- **C-Schritt**
  - Speicher:
    Die Ableitungen von F F_{ix} und F_{iy} benötigen jeweils $(i+1)$ Multiplikationen (Multiplikation des alten Koeffizienten mit dem Exponenten von x respektive y) und i Dekrementierungen des Exponenten um 1. Es werden $|M_A|$ verschiedene Ableitungen von F_{iy} und $|M_B|$ verschiedene Ableitungen von F_{ix} benötigt, da im anderen Fall die Zwischenergebnisse stets Nullpolynome sind. Die vorgelegte Realisation unter JAVA berechnet jedesmal alle benötigten Ableitungen von F erneut, obwohl nur das homogene Polynom $(n-1)$-ten Grades von F nach x respektive y neu abgeleitet werden müßte. Die fehlenden Polynome können aus früheren Schritten bezogen

werden. Sei also dieser (zeitoptimale) Fall vorausgesetzt, dann werden für das neue homogene Polynom F_n $(n+1)$ neue Koeffizienten benötigt und für $F_{n-1,x}$ und $F_{n-1,y}$ jeweils n. Insgesamt wird also in jedem neuen C-Schritt Speicherplatz für $(3n+1)$ neue Koeffizienten benötigt.

- Auswerten von $-(\sum_i p_i F_{n-i+1,y} - q_i F_{n-i+1,x})$:
  Für das Ableiten von F_{n-1} wurden $4n$ Rechenschritte benötigt (2 mal (n Multiplikationen + n Dekrementierungen)). Sei m gesetzt als

$$m := \min\left(\max\left(\mathrm{Gr}\left(A\right), \mathrm{Gr}\left(B\right)\right), n-1\right).$$

  Dann erfolgt die Auswertung von

$$-\left(\sum_{i=2}^{m} p_i F_{n-i+1,y} - q_i F_{n-i+1,x}\right)$$

  Diese Auswertung ist endlich da A und B endliche Eingabe-Polynome sind. Daraus folgt also, daß m immer beschränkt ist. Weiterhin kann man die Anzahl der Summationen bzw. Produkte auf die Anzahl $|M_A \cup M_B|$ (die Anzahl der verschiedenen homogenen Grade) beschränkt werden. Der Einfachheit halber nehme man jedoch an, daß die Polynome A und B keinen Null-Koeffizienten besitzen zwischen den Graden 2 und $\mathrm{Gr}\left(A\right)$ bzw. $\mathrm{Gr}\left(B\right)$. Daraus ergibt sich die Abschätzung zur Komplexität im schlechtesten Fall wie folgt bei $(i+1)$ Koeffizienten von p_i bzw. q_i und $(n-i+2)$ Koeffizienten von $F_{n-i+1,x}$ bzw. $F_{n-i+1,y}$

$$\begin{aligned}
T_C^S(n) &= 2\sum_{i=2}^{m}(i+1)(n-i+2) \\
&= 2\sum_{i=2}^{m}(in - i^2 + 2i + n - i + 2) \\
&= 2\left(\sum_{i=2}^{m} n(i+1) + \sum_{i=2}^{m}(i-i^2) + 2(m-1)\right) \\
&= 2\left(n\frac{m(m+1)}{2} - 3 + \sum_{i=2}^{m}(i-i^2) + 2(m-1)\right) \\
&= 2\left(n\frac{m(m+1)}{2} - 3 + \left(\frac{m(m-1)}{2} - \frac{m(m+1)(2m+1)}{6}\right) + 2(m-1)\right)
\end{aligned}$$

  woraus sich eine Komplexität von $T_C^S(n) = O(nm^2)$ ergibt bzw. $T_C^S(n) = O(n)$ da m beschränkt ist. Die Komplexität des Ableitens ist mit $O(4nm) = O(n)$ unter $T_C^S(n)$ genügend abgedeckt, da sich der Aufwand des Ableitens lediglich dazuaddiert.

- Kürzen:
  Insgesamt sind im C-Schritt maximal

$$T^S(n) = (n+1)\left(3 + \sum_{i=4}^{n}\left(2 + \frac{1}{3}(i-4)\right)\right) = (n+1)\left(1 + \frac{1}{6}(4(n-3) + n(n+1))\right)$$

  Terme durch die Auswertung erzeugt worden. Zu mindestens $(n+1)$ Termen können sie sich bei gleichen Potenzen in x und y zusammenfassen lassen. Das bedeutet, daß

$$(n+1)\left(1 + \frac{1}{6}(4(n-3) + n(n+1)) - 1\right)$$

  Additionen immer noch benötigt werden um $(n+1)$ Koeffizienten im Ergebnis zu erhalten. Daraus folgt, daß eine Komplexität von $O(n^3)$ immer bestehen bleibt. Wie oben jedoch bereits gesehen, ist das Kürzen analog zum Auswerten auf $O(nm^2)$ beschränkt, da die Polynome A und B immer endlich sind. Also gilt für die Komplexität $T_C^S(n) = O(n)$.

- **A-Schritt**

  - Speicher:
    Da die Koeffizienten aufgrund der Formeldarstellungen direkt mit Hilfe der C-Koeffizienten ermittelt werden können, wird kein zusätzlicher Speicher bis auf den Speicher für die Lösungs-Koeffizienten benötigt: Also $(n+1)$.
  - Auswerten:
    Beim Auswerten ist auf die Ermittlung der Hilfskoeffizienten a_i und b_{ij} zu achten. Für beide Hilfskoeffizienten kann man die Anzahl der Multiplikationen und eine Divisionen abschätzen zu
    $$T^S_{a_i,b_{ij}}(n) = 2\frac{l_n(l_n+1)}{2} + 1 = O(n^2)$$
    wobei $l_n := \lceil \frac{n}{4} \rceil$. Sei angenommen, daß diese Koeffizienten wegen ihres vielfachen Aufrufs einmal errechnet und gespeichert werden. Der Aufwand der Berechnung dieser Koeffizienten kann jedoch auf $O(n)$ gemindert werden, da a_{i+1} bzw. $b_{i+1,j}$ durch heranmultiplizieren und dividieren zweier Zahlen an a_i bzw. $b_{i,j}$ erzeugt werden kann. Für einen A-Koeffizienten werden n Multiplikationen dieser Hilfskoeffizienten mit C_{ij}-Koeffizienten durchgeführt und dann aufsummiert. $n+1$ A-Koeffizienten werden berechnet. Dann gilt für den A-Schritt die Komplexität von $O(n^2)$. Einsparungen ergeben sich zwar im Falle ungeraden ns (Inverse von $L'^{(n)}$ ist untere Dreiecksmatrix) jedoch vermindern sie das Problem nicht sehr (nur um ein achtel). Also gilt
    $$T^S_A(n) = O(n^2).$$

Für den Gesamtalgorithmus ergibt sich dann die Komplexität aus der Komplexität des C- und A-Schritts, d.h.
$$T^S_{gesamt}(n) = T^S_C(n) + T^S_A(n) = O(n) + O(n^2) = O(n^2).$$

4.2.2 Zur Auslöschung und Konditionierung

Auslöschung ist ein großes Problem der Numerik. Die Phänomene der Auslöschung betrafen auch den hier vorgestellten Algorithmus wie man in den aufgezeigten Fallstudien im nächsten Kapitel sehen kann. Dort sind in der Betrachtung des Wirbelfalles aufgrund sich fortpflanzender Ungenauigkeiten in den Koeffizienten des Polynoms F mit wachsendem n immer stärker wachsende Strudelgrößen berechnet worden, die im theoretischen Fall Null sein müßten.
Wie solche Fehler sich grundsätzlich entwickeln kann in [16] im Einführungskapitel eingehender nachgelesen werden. Es sollen hier lediglich Kommentierungen zu Lösungsvorschlägen folgen, da letztendlich eine Aufhebung von Ausschlöschungen im Falle eingesetzter Gleitkommazahlen nicht zu erwarten ist. Wie jedoch trotzdem auf ein exaktes Ergebnis unter Einsatz von Computer-Algorithmen ermitteln werden kann wird erst im nächsten Abschnitt erörtert werden.
Seien nochmals die Hilfskoeffizienten a_i und b_{ij} aus dem vorigen Unterabschnitt betrachtet. Der Kurvenverlauf für wachsendes n ist in Abb. 4.2 abgebildet. Es ist dabei das Maximum in $2\lceil \frac{n-1}{4} \rceil + 1$ zu erkennen. Es ergibt sich also für die Kodierungslänge der Zahl $a^{(n)}_k$
$$\langle a^{(n)} \rangle = \left\langle \frac{(n-1)!!}{(n-k)!!(k-2)!!} \right\rangle$$
mit $k := 2\lceil \frac{n-1}{4} \rceil + 1$. Diese Folge ist exponentiell steigend. Für b ergibt sich ein ähnlicher Koeffizient, dessen Betrachtung an dieser Stelle als vernachlässigbar gelten kann, da die Kodierungslänge für b sich analog zu a konstruiert.

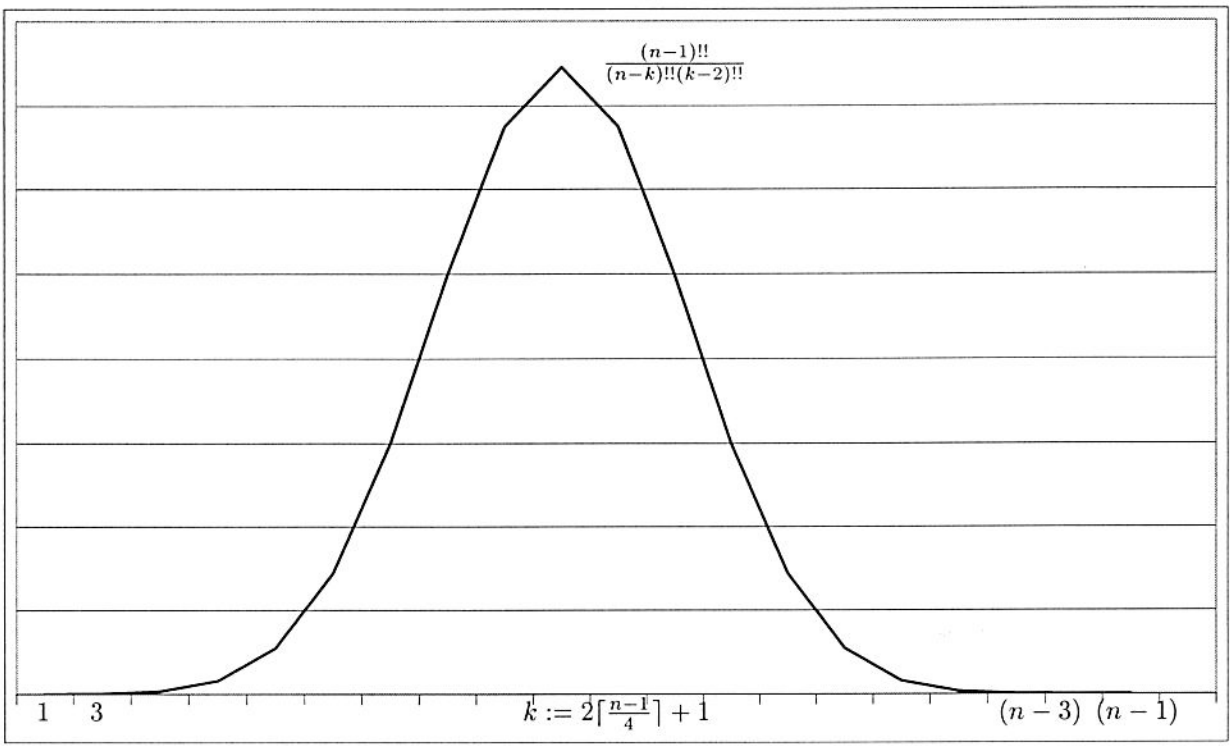

Abbildung 4.2: Die Kurve zeigt den Kurvenverlauf bzw. Wachstumsverhalten des Hilfskoeffizienten a_i im Falle gerader n. Das Maximum der Kurve liegt für $k := 2\lceil\frac{n-1}{4}\rceil + 1$ bei $\frac{n!!}{(n-k)!!(k-2)!!}$. Für b_{ij} ergibt sich der gleiche Kurvenverlauf.

Die Abbildung 4.3 zeigt eine Tabelle die das exponentielle Wachstum der Koeffizienten veranschaulicht. In der letzten Spalte ist an den Exponentenwerten der Zahlen zu erkennen, daß sie sich proportional zum linear steigendem n verhalten. Die Auslöschung verschlimmert sich linear, da die Mantissen im (z.B. im Falle einer Aufsummierung) nicht genügend Ziffern kleinerer Werte aufnehmen können. In so einem Fall kann man von schlechter Konditionierung des Problems sprechen, da im Rechnereinsatz stets von fester Mantissenlänge ausgegangen wird.

4.3 JAVA, C/C++ und algebraische Löser

Aufgrund der Entwicklungsgeschichte von JAVA können alle hier vorgestellten Routinen auch in C++ übernommen werden, da die Notationen und Grundbefehle, bis auf Ausnahmen die die objektorientierte (OO) Strukturierung betrifft, gleich sind. Da C++ eine schnellere Berechnung garantieren kann aufgrund der größeren Nähe zur Maschinensprache ist dieser Weg nicht nur sehr attraktiv als auch mit der vorgelegten Programmversion unter JAVA sehr einfach. Eine Adaptierung in C wird aufgrund der fehlenden Objektorientiertheit ein schwieriges Vorhaben sein. Jedoch kann eine "Pseudeo"-OO-Umgebung realisiert werden, wenn anstelle der Klassen `struct`-Umgebungen definiert werden, die ebenfalls Routinen in sich aufnehmen können.

n	$k := 2\lceil\frac{n-1}{4}\rceil + 1$	$\frac{(n-1)!!}{(n-k)!!(k-2)!!}$
4	3	3.0
6	5	5.0
8	5	11.666666666666666
10	7	21.0
12	7	46.2
14	9	85.8
16	9	183.85714285714286
18	11	347.2857142857143
20	11	733.1587301587301
30	17	22598.706293706295
40	21	746100.4179566563
50	27	2.3373393849643294E7
100	51	7.981240450360928E14
150	77	2.6580506530457407E22
200	101	8.974854588761086E29
250	127	2.9986901088435136E37
300	151	1.0100578753228138E45
350	177	3.379120010327191E52
400	201	1.136987177828599E60
450	227	3.8063630095410493E67
500	251	1.2799737537258833E75
550	277	4.2868826084042343E82
598	301	7.192982815511595E89

Abbildung 4.3: In der ersten Spalte ist n der Laufparameter der Polynomgrade. Für k wird in der zweiten Spalte die Stelle des Maximums von a_k angegeben. In der dritten Spalte wird das Maximum $a_k^{(n)}$ berechnet. Für steigendes k wird der Kurvenverlauf um der Stelle des Maximums steiler. Für den Hilfskoeffizienten b_{ij}, der nur für ungerade n gilt, ergibt sich ein analoges Verhalten.

Was die Datentypen betrifft so sind nur wenige Unterschiede zu vermerken. Alle hier genannten Umgebungen und auch FORTRAN speichern Fließkommazahlen/Gleitkommazahlen im IEEE-754-Standard ab. Die angebotenen Genauigkeiten werden im unteren Kasten abgebildet.

Datentyp	Vorzeichen	Mantisse	Exponent
`float`	1 Bit	24 Bit	7 Bit
`double`	1 Bit	53 Bit	10 Bit

Eine Lösung mit rationalen Zahlen, die natürlich eine entsprechende Klasse der rationalen Zahlen und ihre Verknüpfungen anbieten müßte, könnte bessere Werte unter JAVA/C/C++ liefern, weil dort der Datentyp `long integer` vorhanden wäre mit einer Genauigkeitsdarstellung von 64 Bits, das heißt für den maximalen darstellbaren Wert $2^{64} - 1$ ohne Vorzeichen. Der nachfolgende Kasten stellt die vorhandenen Datentypen und ihre Speicherbreite nochmals dar, sowie darstellbare min. und max. Werte.

Datentyp	Speicherbreite	Minimum	Maximum
`byte`	8 Bits (1 Byte)	-128	127
`short`	16 Bits (2 Byte)	-32768	32767
`int`	32 Bits (4 Byte)	-2147483648	2147483647
`long`	64 Bits (8 Byte)	-9223372036854775808	9223372036854775807
`float`	32 Bits (4 Byte)	$\pm 1.40239846 \cdot 10^{-45}$	$\pm 3.4028347 \cdot 10^{38}$
`double`	64 Bits (8 Byte)	$\pm 4.9406\ldots \cdot 10^{-324}$	$\pm 1.7976931348623157 \cdot 10^{308}$

Nur im Falle des Einsatzes von Koeffizienten variabler Mantissenlänge ist eine exakte Lösung beliebiger aber endlicher Iterationsschritte des Algorithmus möglich. So eine Programmier-Umgebung können algebraische Löser wie MapleV oder Mathematica bieten. Dort ist es nicht nur auf einfache Weise möglich rationale Zahlen aus $\mathbb{Q}$ als Koeffizienten einzusetzen und auszuwerten, sondern auch die Darstellungen der Zahlen in (fast) beliebiger aber endlicher Länge festzulegen.

1a) Diplom- oder Facharbeit

1b) $y' = -\dfrac{x^{2n-1} + p(x,y)}{y^{2n-1} + q(x,y)}$

1c) Kreiszyklen

5 Beispiele und Fallstudien

Das letzte Kapitel soll ein paar praktische Beispiele zeigen, die mit den aus dieser Arbeit gewonnenen Erkenntnissen analysiert werden konnten. Eine Vielzahl von Beispielen sind dabei aus [5][1] übernommen. Aber auch andere interessante Differentialgleichungen, die bisher noch nicht betrachtet wurden, sollen hier der Analyse unterzogen werden.
Allgemein gilt für die Ausgabe des Algorithmus das Format

```
[...]
n. C-Schritt: (Polynom der C-Koeffizienten)
n. A-Schritt: (neu berechnetes n-tes homogenes Polynom vom Design-F)
n.: Strudelgrösse: (Angabe wenn n durch 2 teilbar ist.)
[...]
```

Beim Vergleichen der Ergebnisse mit denen aus [5] ist in Bezug auf die Strudelgröße ein Faktor von 2 (bzw. $\frac{1}{2}$) zu berücksichtigen. Frommer schien, ganz entgegen seiner Definition, stets seine Strudelgrößen zu halbieren. Der Grund zur Entstehung dieses Unterschieds war zum Zeitpunkt der Arbeit nicht herauszufinden.

Um die Vielseitigkeit des Algorithmus zu zeigen, wird am Ende ein Ausblick auf die Analyse von Differentialgleichungen des Poincaréschen Typs jedoch ohne Vorliegen eines linearen Anteils gewagt. Eine vollständige Analyse soll diese Arbeit nicht mehr aufzeigen.

5.1 Beispiele von Frommer

5.1.1 Polynome 2. Grades

Sei als Ausgangsdifferentialgleichung gegeben[2]

$$y' = -\frac{x + 4x^2 + y^2}{y + x^2 - 2y^2}$$

Der Algorithmus berechnet dann einen Strudelfall

```
3. C-Schritt: (-2.0*y^3 -4.0*x*y^2 -8.0*x^2*y +2.0*x^3 )
3. A-Schritt: (2.0*x*y^2 +2.0*x^2*y +4.0*x^3 )
4. C-Schritt: (-4.0*y^4 -12.0*x*y^3 -24.0*x^2*y^2
               -12.0*x^3*y +4.0*x^4)
4. A-Schritt: (6.0*x^2*y^2 +8.0*x^3*y +6.0*x^4 )
4.: 1. Strudelgroesse: -4.0
```

[1] Vgl. S. 406f.
[2] [5], S.406, Strudelfall in Beispiel 1

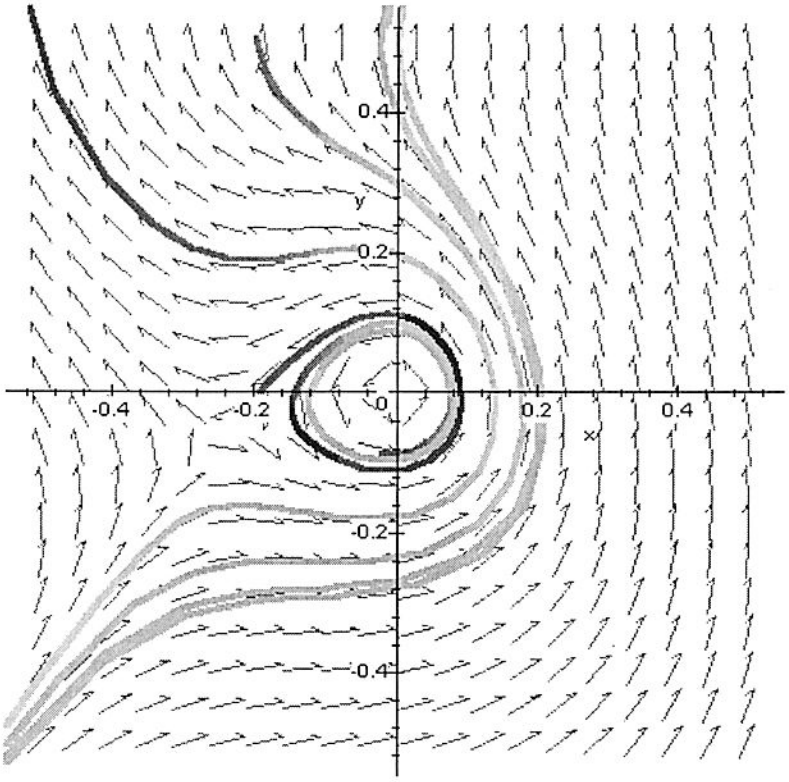

Abbildung 5.1: $y' = -\frac{x+4x^2+y^2}{y+x^2-2y^2}$. Die Nieveaulinien um den kritischen Punkt Null bilden Spiralen.

Strudel 2. Grades
Auch die nächste Differentialgleichung beschreibt einen Strudelfall[3].

$$y' = -\frac{x + x^2 + 2xy - y^2}{y - 2xy + y^2}$$

Als Auswertung ergibt sich

```
3. C-Schritt: (2.0*y^3 -2.0*x*y^2 -6.0*x^2*y )
3. A-Schritt: (-0.6666666666666666*y^3 -2.0*x*y^2 +0.6666666666666666*x^3 )
4. C-Schritt: (-4.0*y^4 +4.0*x*y^3 +12.0*x^2*y^2 )
4. A-Schritt: (4.0*x*y^3 -2.0*x^2*y^2 -1.0*x^4 )
4.: 1. Strudelgroesse: 0.0
5. C-Schritt: (4.0*y^5-20.0*x^2*y^3-8.0*x^3*y^2 +12.0*x^4*y )
5. A-Schritt: (-1.0666666666666667*y^5 -4.0*x*y^4 -2.6666666666666665*x^2*y^3
   +1.3333333333333333*x^3*y^2 -1.8666666666666667*x^5 )
6. C-Schritt: (-9.333333333333332*y^6 -2.666666666666667*x*y^5 +44.0*x^2*y^4
   +26.666666666666664*x^3*y^3 -6.666666666666667*x^4*y^2 +16.0*x^5*y )
6. A-Schritt: ( +8.399999999999999*x*y^5 +1.3333333333333335*x^2*y^4
   -0.6666666666666687*x^3*y^3 -5.333333333333332*x^4*y^2
   +0.9333333333333327*x^5*y -4.444444444444444*x^6 )
6.: 2. Strudelgroesse: -0.9333333333333327
```

[3][5], S.407 unten

5.1.2 Polynome 3. Grades

Das nächste Beispiel wird einen Wirbelfall beschreiben[4]. Sehr interessant dabei ist wie die schlechte Konditionierung des Problems die Schwäche des Algorithmus im Wirbelfall aufzeigt.

$$y' = -\frac{x + 4x^2 + y^2 + 2x^3 - 2y^3}{y + x^2 - 2y^2 + 2x^3 - 2y^3}$$

Die nächsten Seiten zeigen die Auswertung des Algorithmus:

```
3. C-Schritt: (-2.0*x^0*y^3-4.0*x^1*y^2-8.0*x^2*y^1 +2.0*x^3*y^0 )
3. A-Schritt: ( +2.0*x^1*y^2 +2.0*x^2*y^1 +4.0*x^3*y^0 )
4. C-Schritt: (-16.0*x^1*y^3-24.0*x^2*y^2-16.0*x^3*y^1 +8.0*x^4*y^0 )
4. A-Schritt: ( +8.0*x^2*y^2 +8.0*x^3*y^1 +8.0*x^4*y^0 )
4.: 1.Strudelgroesse: 0.0
5. C-Schritt: (-4.0*x^0*y^5-32.0*x^1*y^4-84.0*x^2*y^3-52.0*x^3*y^2-40.0*x^4*y^1
   +20.0*x^5*y^0 )
5. A-Schritt: (-2.666666666666667*x^0*y^5 +4.0*x^1*y^4 +9.333333333333334*x^2*y^3
   +33.333333333333336*x^3*y^2 +20.0*x^4*y^1 +21.333333333333336*x^5*y^0 )
6. C-Schritt: (5.333333333333336*x^0*y^6 -85.33333333333334*x^1*y^5
   -186.66666666666666*x^2*y^4 -320.0*x^3*y^3 -213.33333333333337*x^4*y^2
   -170.66666666666669*x^5*y^1 +74.66666666666669*x^6*y^0 )
6. A-Schritt: (-5.333333333333325*x^1*y^5 +42.66666666666667*x^2*y^4
   +53.333333333333336*x^3*y^3 +122.66666666666667*x^4*y^2
   +74.66666666666667*x^5*y^1 +69.33333333333334*x^6*y^0 )
6.: 2.Strudelgroesse: 1.0658141036401503E-14
7. C-Schritt: (-24.00000000000002*x^0*y^7 -149.3333333333334*x^1*y^6 -640.0*x^2*y^5
  -941.3333333333335*x^3*y^4 -1682.666666666667*x^4*y^3 -912.0*x^5*y^2
  -581.3333333333334*x^6*y^1 +290.66666666666674*x^7*y^0 )
7. A-Schritt: (-11.733333333333304*x^0*y^7 +24.00000000000002*x^1*y^6
   +33.600000000000094*x^2*y^5 +261.33333333333337*x^3*y^4
   +277.3333333333335*x^4*y^3 +545.6000000000001*x^5*y^2
   +290.66666666666674*x^6*y^1 +238.9333333333334*x^7*y^0 )
8. C-Schritt: (44.799999999999734*x^0*y^8 -502.40000000000043*x^1*y^7
   -1362.1333333333346*x^2*y^6 -4444.800000000001*x^3*y^5
   -6208.000000000004*x^4*y^4 -8355.200000000003*x^5*y^3
   -3574.400000000002*x^6*y^2 -2364.8*x^7*y^1 +1192.5333333333335*x^8*y^0 )
8. A-Schritt: (-44.800000000000196*x^1*y^7 +251.20000000000022*x^2*y^6
   +349.5111111111111*x^3*y^5 +1488.0000000000005*x^4*y^4
   +1591.111111111112*x^5*y^3 +2384.533333333334*x^6*y^2
   +1192.5333333333338*x^7*y^1 +891.7333333333336*x^8*y^0 )
8.: 3.Strudelgroesse: -4.654054919228656E-13
9. C-Schritt: (-122.6666666666659*x^0*y^9 -537.5999999999997*x^1*y^8
   -4881.0666666666675*x^2*y^7 -11810.488888888889*x^3*y^6
   -30788.977777777796*x^4*y^5 -34499.55555555557*x^5*y^4
   -39952.71111111113*x^6*y^3 -16454.40000000001*x^7*y^2
   -9422.933333333338*x^8*y^1 +5127.466666666667*x^9*y^0 )
9. A-Schritt: (-63.085714285715135*x^0*y^9 +122.6666666666659*x^1*y^8
   -15.085714285716156*x^2*y^7 +1954.1333333333316*x^3*y^6
   +2926.2222222222176*x^4*y^5 +8502.755555555557*x^5*y^4
   +8188.4444444444425*x^6*y^3 +10566.247619047625*x^7*y^2
   +5127.466666666667*x^8*y^1 +3395.04761904762*x^9*y^0 )
```

[4][5], S.406, Wirbelfall in Beispiel 1

```
10. C-Schritt: (412.0380952381048*x^0*y^10-2552.9904761904663*x^1*y^9
   -8308.11428571423*x^2*y^8-47588.57142857137*x^3*y^7
   -95748.97777777772*x^4*y^6-187451.7333333333*x^5*y^5
   -187251.91111111105*x^6*y^4-197960.93968253976*x^7*y^3
   -71468.19047619044*x^8*y^2-36352.91428571432*x^9*y^1
   +21928.228571428583*x^10*y^0 )
10. A-Schritt: (-412.038095238088*x^1*y^9 +1276.4952380952332*x^2*y^8
    +1533.2571428571457*x^3*y^7 +14450.13333333331*x^4*y^6
    +21296.35555555555*x^5*y^5 +45692.08888888886*x^6*y^4
    +41961.95555555555*x^7*y^3 +47591.1619047619*x^8*y^2
    +21928.228571428568*x^9*y^1 +13153.523809523813*x^10*y^0 )
10.: 4. Strudelgroesse: 1.6825651982799172E-11
11. C-Schritt: (-556.8000000000283*x^0*y^11 +625.3714285713781*x^1*y^10
    -31759.542857142842*x^2*y^9-107526.80634920642*x^3*y^8
    -393700.7746031743*x^4*y^7-658220.7999999997*x^5*y^6
    -1109393.777777777*x^6*y^5-1023128.1777777777*x^7*y^4
    -947846.806349206*x^8*y^3-309016.07619047613*x^9*y^2
    -143600.76190476192*x^10*y^1 +94678.24761904769*x^11*y^0 )
11. A-Schritt: (-327.494885361529*x^0*y^11 +556.8000000000283*x^1*y^10
    -2113.907583774087*x^2*y^9 +12442.514285714375*x^3*y^8
    +22125.409523809878*x^4*y^7 +98648.17777777786*x^5*y^6
    +135516.44444444482*x^6*y^5 +243040.4063492063*x^7*y^4
    +212588.80000000028*x^8*y^3 +213334.27019400347*x^9*y^2
    +94678.24761904769*x^10*y^1 +51842.66384479717*x^11*y^0 )
12. C-Schritt: (+3312.919929452938*x^0*y^12-9635.03633157045*x^1*y^11
    -29438.961552031305*x^2*y^10-398002.74850088597*x^3*y^9
    -1054970.7174603269*x^4*y^8-2874311.111111117*x^5*y^7
    -4334213.6888889*x^6*y^6-6398164.92698413*x^7*y^5
    -5355831.263492071*x^8*y^4-4505452.946737211*x^9*y^3
    -1356587.1633157006*x^10*y^2-555548.2186948843*x^11*y^1
    +410770.3308641972*x^12*y^0 )
12. A-Schritt: (-3312.9199294539494*x^1*y^11 +4817.518165785225*x^2*y^10
    -2334.3858906540454*x^3*y^9 +111544.48253968456*x^4*y^8
    +206792.2488888881*x^5*y^7 +627777.8285714323*x^6*y^6
    +825965.6330158738*x^7*y^5 +1270603.9873015904*x^8*y^4
    +1053962.1587301597*x^9*y^3 +958786.8895943573*x^10*y^2
    +410770.3308641983*x^11*y^1 +206093.49982363323*x^12*y^0 )
12.: 5. Strudelgroesse: -1.011130734696053E-9
13. C-Schritt: (-1692.6476190457959*x^0*y^13 +36763.67689594946*x^1*y^12
    -150187.20846560175*x^2*y^11-685547.4342151411*x^3*y^10
    -3856299.9082892663*x^4*y^9-8780377.640634904*x^5*y^8
    -1.9935126227301687E7*x^6*y^7-2.7263379464127E7*x^7*y^6
    -3.546070812444455E7*x^8*y^5-2.768493165714288E7*x^9*y^4
    -2.1292609986596175E7*x^10*y^3-5852207.271957679*x^11*y^2
    -2111593.605643738*x^12*y^1+1781222.7837742479*x^13*y^0 )
13. A-Schritt: (-1520.3165624503044*x^0*y^13 +1692.6476190457959*x^1*y^12
    -28263.896103902265*x^2*y^11 +56832.99329805043*x^3*y^10
    +93661.14426805306*x^4*y^9 +884925.9682539541*x^5*y^8
    +1603887.9898412318*x^6*y^7 +3859219.1390476176*x^7*y^6
    +4811324.424126952*x^8*y^5 +6512891.439858917*x^9*y^4
    +5174155.3777777655*x^10*y^3+4304015.976911985*x^11*y^2
    +1781222.7837742479*x^12*y^1+824586.5814975163*x^13*y^0 )
14. C-Schritt: (+23004.659932670264*x^0*y^14 +589.5018759317209*x^1*y^13
```

```
    +161010.68488071277*x^2*y^12-2396394.611255089*x^3*y^11
    -8469156.029628968*x^4*y^10-3.2973337916048605E7*x^5*y^9
    -6.72992653409512E7*x^6*y^8-1.3061760536380884E8*x^7*y^7
    -1.633512422806338E8*x^8*y^6-1.9248711717925933E8*x^9*y^5
    -1.4095307540317395E8*x^10*y^4-9.914030439967959E7*x^11*y^3
    -2.5114197850504834E7*x^12*y^2-7855654.689369975*x^13*y^1
    +7719437.758409521*x^14*y^0 )
14. A-Schritt:(-23004.65993260925*x^1*y^13-294.75093796586043*x^2*y^12
    -153357.08800154435*x^3*y^11 +598214.3999998746*x^4*y^10
    +1356445.612322396*x^5*y^9 +6492580.319341226*x^6*y^8
    +1.1358182264550397E7*x^7*y^7+2.281978098981733E7*x^8*y^6
    +2.698427979249851E7*x^9*y^5+3.2940580311816327E7*x^10*y^4
    +2.507949766960604E7*x^11*y^3+1.9241885470578745E7*x^12*y^2
    +7719437.758409459*x^13*y^1+3309958.9736091043*x^14*y^0 )
14.: 6. Strudelgroesse: 6.101254257373512E-8
15. C-Schritt: ( +3095.794003418993*x^0*y^15 +453918.71014849184*x^1*y^14
    -62128.794741905236*x^2*y^13-1472850.179683876*x^3*y^12
    -2.730956155519227E7*x^4*y^11-8.510344158288616E7*x^5*y^10
    -2.6006354899752435E8*x^6*y^9-4.780075338482342E8*x^7*y^8
    -8.14921977861103E8*x^8*y^7-9.490092006354862E8*x^9*y^6
    -1.0213003537096648E9*x^10*y^5-7.03738018261258E8*x^11*y^4
    -4.5686079279790115E8*x^12*y^3-1.0680621618239483E8*x^13*y^2
    -2.804911000237298E7*x^14*y^1+3.333848014827655E7*x^15*y^0 )
15. A-Schritt: (-4679.436001090698*x^0*y^15-3095.794003418993*x^1*y^14
    -262055.12508242327*x^2*y^13+6262.559564679778*x^3*y^12
    -483466.6115969094*x^4*y^11+5476942.453993686*x^5*y^10
    +1.3297551475886716E7*x^6*y^9+4.497613907678018E7*x^7*y^8
    +7.471068714140178E7*x^8*y^7+1.3052567671948272E8*x^9*y^6
    +1.4719840106252986E8*x^10*y^5+1.64041310366051E8*x^11*y^4
    +1.1997750196449228E8*x^12*y^3+8.561738725093117E7*x^13*y^2
    +3.333848014827655E7*x^14*y^1+1.3285592300282355E7*x^15*y^0 )
16. C-Schritt: (+122392.44788841697*x^0*y^16 +494619.461881582*x^1*y^15
    +4559880.140243532*x^2*y^14-4763744.063872457*x^3*y^13
    -3.680886359812129E7*x^4*y^12-2.709825228718594E8*x^5*y^11
    -7.476603626358835E8*x^6*y^10-1.8989742721912422E9*x^7*y^9
    -3.1947005921118355E9*x^8*y^8-4.902508576610341E9*x^9*y^7
    -5.344191344953107E9*x^10*y^6-5.295706120143393E9*x^11*y^5
    -3.460267213166713E9*x^12*y^4-2.078658830443948E9*x^13*y^3
    -4.472819731922957E8*x^14*y^2-9.446253609524673E7*x^15*y^1
    +1.4316993965536514E8*x^16*y^0 )
16. A-Schritt: (-122392.4478908358*x^1*y^15-247309.730940791*x^2*y^14
    -2131922.286202023*x^3*y^13+325351.9576753457*x^4*y^12
    +1818774.7754989988*x^5*y^11 +4.581445772732725E7*x^6*y^10
    +1.0966669788091035E8*x^7*y^9+2.9463985618306434E8*x^8*y^8
    +4.6463343033778083E8*x^9*y^7 +7.259627426074855E8*x^10*y^6
    +7.815113961197795E8*x^11*y^5+8.042902146490254E8*x^12*y^4
    +5.667557072127393E8*x^13*y^3+3.782728349314321E8*x^14*y^2
    +1.4316993965536758E8*x^15*y^1+5.318801287238194E7*x^16*y^0 )
16.: 7. Strudelgroesse: -2.4188339011743665E-6
17. C-Schritt: (+110593.40375578866*x^0*y^17 +3786663.910359662*x^1*y^16
    +9280468.892961292*x^2*y^15+3.6113327285145566E7*x^3*y^14
    -8.100562143631874E7*x^4*y^13-5.0079940856644523E8*x^5*y^12
    -2.3943294225282283E9*x^6*y^11-5.916724402654093E9*x^7*y^10
```

```
    -1.3038912811811584E10*x^8*y^9-2.0349921018870277E10*x^9*y^8
    -2.8456973743491966E10*x^10*y^7-2.9224247758806858E10*x^11*y^6
    -2.6907816867509853E10*x^12*y^5-1.6724734924045795E10*x^13*y^4
    -9.317224344635197E9*x^14*y^3-1.844248092347155E9*x^15*y^2
    -2.8762568947292423E8*x^16*y^1 +6.102192560485582E8*x^17*y^0 )
17. A-Schritt: (+12393.8640009648*x^0*y^17-110593.40375578866*x^1*y^16
    -1787984.111171787*x^2*y^15-3683321.1176846367*x^3*y^14
    -1.5733272238180336E7*x^4*y^13 +5887825.157746764*x^5*y^12
    +4.9377811578351624E7*x^6*y^11 +3.5214047491731274E8*x^7*y^10
    +8.074850412519938E8*x^8*y^9 +1.8400352845538568E9*x^9*y^8
    +2.7617286390138183E9*x^10*y^7+3.9252050927202563E9*x^11*y^6
    +4.0463623526586328E9*x^12*y^5+3.8814651864485683E9*x^13*y^4
    +2.6397533348099256E9*x^14*y^3+1.6562056726952982E9*x^15*y^2
    +6.102192560485582E8*x^16*y^1+2.1176688440373647E8*x^17*y^0 )
18. C-Schritt: (+255276.01527684732*x^0*y^18 +6238896.391817856*x^1*y^17
    +5.383317346873322E7*x^2*y^16 +1.2265710317261745E8*x^3*y^15
    +2.3418660457831025E8*x^4*y^14-1.0354820561133813E9*x^5*y^13
    -5.18656511471368E9*x^6*y^12-1.9157371983608627E10*x^7*y^11
    -4.323504447774773E10*x^8*y^10-8.497593593501836E10*x^9*y^9
    -1.2393721710330211E11*x^10*y^8-1.5976645687894174E11*x^11*y^7
    -1.555879923263383E11*x^12*y^6-1.3379693544531308E11*x^13*y^5
    -7.936778547418646E10*x^14*y^4-4.110067369704906E10*x^15*y^3
    -7.453143118261568E9*x^16*y^2-7.041304348501549E8*x^17*y^1
    +2.574836543274774E9*x^18*y^0 )
18. A-Schritt: (-255276.01547755598*x^1*y^17-3119448.195908928*x^2*y^16
    -1.9390955243950557E7*x^3*y^15-4.314206857679007E7*x^4*y^14
    -1.0501018664751372E8*x^5*y^13 +7.19155160063867E7*x^6*y^12
    +5.459189554708574E8*x^7*y^11+2.5025447719606586E9*x^8*y^10
    +5.471128109769684E9*x^9*y^9+1.1000138365462494E10*x^10*y^8
    +1.5743397281020845E10*x^11*y^7+2.0647296983553474E10*x^12*y^6
    +2.0445521022575706E10*x^13*y^5+1.8405765524759567E10*x^14*y^4
    +1.2106359372471E10*x^15*y^3+7.170233487255458E9*x^16*y^2
    +2.5748365432749753E9*x^17*y^1+8.35810967186726E8*x^18*y^0 )
18.: 8. Strudelgroesse: -2.007086732191965E-4
```

Mit jeder neuen Strudelgröße haben sich die Werte um mindestens eine 10er Potenz verschlechtert. Daher ist der Algorithmus gezwungen, schon früh eine Strudelgröße als Nichtnull zu werten.

1. Strudel 3. Grades
Im folgenden Beispiel[5] sei ein Strudelfall gezeigt.

$$y' = -\frac{x + 2x^3 + y^3}{y + x^2y + y^3}$$

```
3. C-Schritt: ( )
3. A-Schritt: ( )
4. C-Schritt: (-2.0*y^4 +2.0*x*y^3-2.0*x^3*y )
4. A-Schritt: (1.0*x*y^3-1.0*x^2*y^2 +1.0*x^3*y )
4.: 1. Strudelgroesse: -1.0
```

[5][5], S.406, Beispiel 2

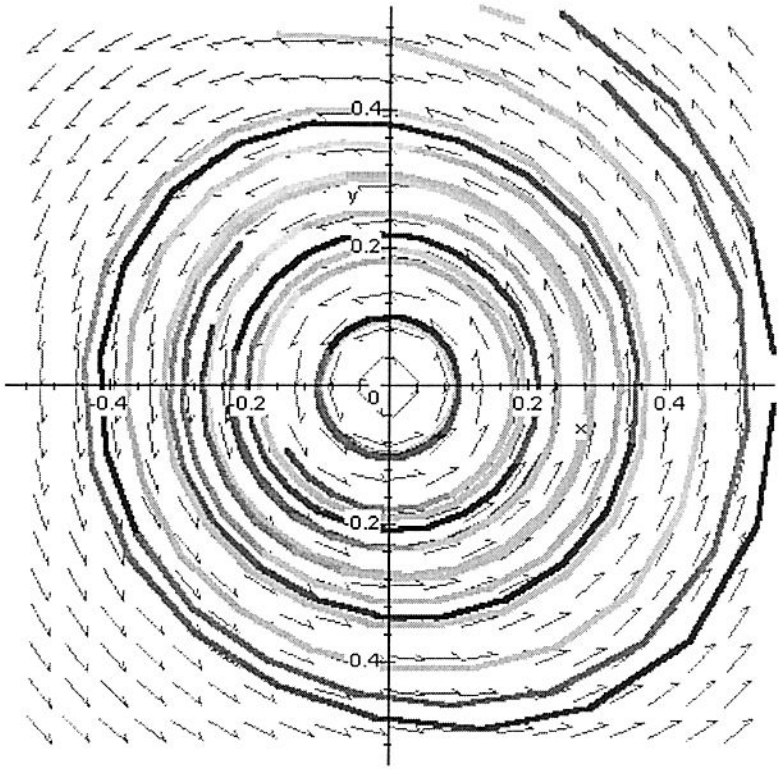

Abbildung 5.2: Tangentenfeld und Trajektorien von $y' = -\frac{x+2x^3+y^3}{y+x^2y+y^3}$.

2. Strudel 3. Grades

Zum Vergleich sei noch ein Beispiel[6] eines Strudelfalles gezeigt.

$$y' = -\frac{x + 4x^2y + y^3}{y + x^3 + xy^2}$$

```
3. C-Schritt: ( )
3. A-Schritt: ( )
4. C-Schritt: (-2.0y^4-6.0*x^2*y^2 +4.0*x^4 )
4. A-Schritt: (2.0*x*y^3 +4.0*x^3*y )
4.: 1. Strudelgroesse: 0.0
5. C-Schritt: ( )
5. A-Schritt: ( )
6. C-Schritt: (-4.0*x*y^5-12.0*x^3*y^3 +8.0*x^5*y )
6. A-Schritt: (2.0*x^2*y^4 +5.0*x^4*y^2 +0.33333333333333326*x^6 )
6.: 2. Strudelgroesse: 0.0
7. C-Schritt: ( )
7. A-Schritt: ( )
8. C-Schritt: (-4.0*x^2*y^6-14.0*x^4*y^4 +2.0*x^6*y^2
   +3.999999999999999*x^8 )
8. A-Schritt: (1.2571428571428567*x*y^7 +4.266666666666665*x^3*y^5
   +7.066666666666655*x^5*y^3 +2.7428571428571424*x^7*y )
```

[6][5], S.406, Strudelfall in Beispiel 3

```
 8.: 3. Strudelgroesse: 1.2571428571428567
```

5.1.3 Polynome 5. Grades

Die nächste Differentialgleichung zeigt einen Wirbelfall[7]

$$y' = -\frac{x + 4x^2y + y^3 + 2x^3y^2}{y + 2x^3 + xy^2 + 2x^4y}$$

Sei aus Platzgründen nur noch die Strudelgrößen und die letzten Schritte gezeigt.

```
  3. C-Schritt: ( )
  3. A-Schritt: ( )
  4. C-Schritt: (-2.0*x^0*y^4-6.0*x^2*y^2 +4.0*x^4*y^0 )
  4. A-Schritt: (2.0*x^1*y^3 +4.0*x^3*y^1 )
  4.:  1. Strudelgroesse: 0.0
  6.:  2. Strudelgroesse: 0.0
  8.:  3. Strudelgroesse: -1.1102230246251565E-16
 10.:  4. Strudelgroesse: 0.0
 12.:  5. Strudelgroesse: -2.5979849584765384E-15
 14.:  6. Strudelgroesse: 0.0
 16.:  7. Strudelgroesse: -1.516359194062639E-14
 18.:  8. Strudelgroesse: 0.0
 20.:  9. Strudelgroesse: 8.544126330536161E-15
 22.: 10. Strudelgroesse: 0.0
 24.: 11. Strudelgroesse: 1.4519389952746118E-12
 26.: 12. Strudelgroesse: 0.0
 28.: 13. Strudelgroesse: 1.3101097901904932E-11
 30.: 14. Strudelgroesse: 0.0
 32.: 15. Strudelgroesse: -1.278980059765441E-10
 34.: 16. Strudelgroesse: 0.0
 36.: 17. Strudelgroesse: -4.227684185053371E-9
 38.: 18. Strudelgroesse: 0.0
 40.: 19. Strudelgroesse: -2.432166333046484E-9
 42.: 20. Strudelgroesse: 0.0
 44.: 21. Strudelgroesse: 1.509389448302279E-6
 46.: 22. Strudelgroesse: 0.0
 47. C-Schritt: ( )
 47. A-Schritt: ( )
 48. C-Schritt: (-0.00133128149340261*x^2*y^46-0.02886456384000359 4*x^4*y^44
     -0.28713379504057046*x^6*y^42-1.6968933051262551*x^8*y^40
     -6.265007461952673*x^10*y^38-12.592849770758939*x^12*y^36
     +3.489531128148757*x^14*y^34 +120.54748148879348*x^16*y^32
     +465.1343703261745*x^18*y^30 +1094.0540295221565*x^20*y^28
     +1815.3872733122967*x^22*y^26 +2195.0996407122816*x^24*y^24
     +1900.4017581190478*x^26*y^22 +1069.7831053986677*x^28*y^20
     +224.31665343722847*x^30*y^18-220.9471234990557*x^32*y^16
     -261.54425382420237*x^34*y^14-140.4697054022555*x^36*y^12
     -43.65674114995073*x^38*y^10-7.04966645696048*x^40*y^8
     -0.187710799678834*x^42*y^6+0.09635274250230215*x^44*y^4
     +0.0081341258187677 7*x^46*y^2 +5.2684386519516154E-5*x^48*y^0 )
 48. A-Schritt: (1.5077066577490255E-5*x^1*y^47 +6.799678741815506E-4*x^3*y^45
```

[7] [5], S.407, Wirbelfall in Beispiel 3

```
    +0.011892623635634675*x^5*y^43 +0.1140738016246945*x^7*y^41
    +0.7082132413043034*x^9*y^39 +3.080483988438228*x^11*y^37
    +9.736212103305643*x^13*y^35 +22.485192832503248*x^15*y^33
    +36.55669894022434*x^17*y^31+35.164384043198936*x^19*y^29
    -3.5374710604470003*x^21*y^27-83.08256486714632*x^23*y^25
    -170.88655049563758*x^25*y^23-215.95527479698944*x^27*y^21
    -193.2704784874291*x^29*y^19-125.69212079672198*x^31*y^17
    -58.05511909227935*x^33*y^15-17.408072358856796*x^35*y^13
    -2.3198712233211576*x^37*y^11+0.4650809664978977*x^39*y^9
    +0.27403402818150135*x^41*y^7+0.0489755806858939*x^43*y^5
    +0.0033005566186809967*x^45*y^3+3.760731994202589E-5*x^47*y^1 )
48.: 23. Strudelgroesse: 1.5077066577490255E-5
```

5.2 Beispiele höheren Grades

$$y' = -\frac{x + x^7 + 2x^8y - y^{11}}{y - 2x^{10}y + y^{11}}$$

```
3. C-Schritt: ( )
3. A-Schritt: ( )
4. C-Schritt: ( )
4. A-Schritt: ( )
4.: 1. Strudelgroesse: 0.0
5. C-Schritt: ( )
5. A-Schritt: ( )
6. C-Schritt: ( )
6. A-Schritt: ( )
6.: 2. Strudelgroesse: 0.0
7. C-Schritt: ( )
7. A-Schritt: ( )
8. C-Schritt: (-2.0*x^7*y^1 )
8. A-Schritt: ( +0.25*x^8*y^0 )
8.: 3. Strudelgroesse: 0.0
9. C-Schritt: ( )
9. A-Schritt: ( )
10. C-Schritt: (-4.0*x^8*y^2 )
10. A-Schritt: (-0.2222222222222222*x^1*y^9-0.6666666666666666*x^3*y^7
    -0.9333333333333332*x^5*y^5-0.6666666666666666*x^7*y^3
    +0.2222222222222222*x^9*y^1 )
10.: 4. Strudelgroesse: -0.2222222222222222
```

5.3 Ausblick

Trotz fehlen des Charakteristikums eines linearen Anteils ist der Algorithmus im Stande eine Analyse für Differentialgleichungen, die nicht dem Poincaréschen Problem angehören, durchzuführen.

5.3.1 Polynome ohne linearen Anteil in x und y

Die Differentialgleichung 7.Grades

$$y' = -\frac{x^5 + xy^6 + x^4y^3}{y^5 + 3x^6y + 6xy^6}$$

beschreibt einen Strudelfall ohne linearen Anteil. Das Ergebnis laut Algorithmus ist

```
6. C-Schritt: ( )
6. A-Schritt: ( )
6.: 2. Strudelgroesse: 0.0
7. C-Schritt: ( )
7. A-Schritt: ( )
8. C-Schritt: ( )
8. A-Schritt: ( )
8.: 3. Strudelgroesse: 0.0
9. C-Schritt: ( )
9. A-Schritt: ( )
10. C-Schritt: ( +12.0*x^5*y^5 )
10. A-Schritt: (-2.0*x^6*y^4-1.0*x^8*y^2-0.2*x^10*y^0 )
10.: 4. Strudelgroesse: 0.0
11. C-Schritt: ( )
11. A-Schritt: ( )
12. C-Schritt: (-6.0*x^1*y^11-6.0*x^4*y^8 +36.0*x^6*y^6
    +18.0*x^11*y^1 )
12. A-Schritt: ( +0.2987012987012987*x^1*y^11 +3.0*x^2*y^10
    +1.0952380952380951*x^3*y^9 +7.5*x^4*y^8
    +3.1714285714285713*x^5*y^7
    +10.0*x^6*y^6-1.9714285714285715*x^7*y^5
    +7.5*x^8*y^4 -1.0952380952380951*x^9*y^3
    +3.0*x^10*y^2-0.2987012987012987*x^11*y^1-1.0*x^12*y^0 )
12.: 5. Strudelgroesse: 0.2987012987012987
```

A Symbolverzeichnis

Matrizen

A', A^T : Transponierte von A
$\det(A), |A|$: Determinate von A
$tr(A)$: Spur von A
A^{-1} : Inverse von A

Räume

$\mathbb{N}$: Menge der natürlichen Zahlen
$\mathbb{Z}$: Menge der ganzen Zahlen
$\mathbb{Q}$: Körper der rationalen Zahlen
$\mathbb{R}$: Körper der reellen Zahlen
$\mathbb{C}$: Körper der komplexen Zahlen
$\mathbb{K}$: vollständiger Körper
$\mathbb{K}^m$: m-dimensionaler Vektorraum über $\mathbb{K}$
$\mathbb{K}_{\neq 0}$: alle Elemente sind nicht null
$\mathbb{R}_{\geq 0}$: alle Elemente sind pos. oder null
$\mathbb{R}_{>0}$: alle Elemente sind echt pos.

Funktionenräume

$C^0(S)$: Menge der stetigen Funktionen $f : S \to \mathbb{K}$.
$C^n(S)$: Menge der n-mal stetigdifferenzierbaren Funktionen $f : S \to \mathbb{K}$.
$L(\mathbb{K}^n, \mathbb{K})$: Menge der linearen Abbildungen $\mathbb{K}^n \to \mathbb{K}$,
wobei $\mathbb{K}$ bestenfalls ein Körper ist.

Autonomes System

ω : Pfaffsche Form, Definition 2.4
dF : Differential, Definition 2.4
$\dot{x}$: 1. Ableitung
$\Phi(t)$: Parameterabbildung, Definition 2.7
$A(x,y), B(x,y)$: Funktionen als Abbildungen von $Teilmengendes\mathbb{R}^2 \to \mathbb{R}$, S.5

Koeffizienten

$d_{(\frac{n}{2}-1)}$: $(\frac{n}{2}-1)$te Strudelgröße des n-ten homogenen Polynoms, Definition in Satz 3.1, Formel (3.30)

$C_{ij}^{(n)}$: Koeffizienten des C-Schrittes, S.13

$A_{ij}^{(n)}$: Koeffizienten des neuen homogen Polynoms n-ten Grades des A-Schrittes. Formeln: (3.15), (3.25), (3.29)

$F(x,y)$: Design Stammfunktion bzw. Stammfunktion der Vergleichsdifferentialgleichung. Definition (2.11)

$a_i^{(n)}, b_{ij}^{(n)}$: Hilfskoeffizienten nach (3.26) und (3.14)

Komplexitätsanalyse

$o(\cdot), O(\cdot)$: Landausymbole, Definition B.2.

$T_A^S(n)$: Komplexität im schlechtesten Fall, Definition B.1.

$\langle a \rangle, \langle\langle a \rangle\rangle$: Kodierungslänge einer Zahl in Bits und Bytes, siehe (B.2) und (B.3).

$t_A^S(P)$: Laufzeit eines Algorithmus A zur Lösung von P, Definition B.3.

$n!!$: In Anlehnung an eine durch Fakultät bestimmte Zahl, Definition 4.1.

$\mathrm{Gr}(\cdot), \mathrm{gr}(\cdot)$: Größter bzw. kleinster Grad, Definition 4.2.

weitere Bezeichnungen

$\mathrm{grad}(f)$: Gradient von f

$\langle a,b \rangle := \sum_{i=1}^{n} \bar{a}_i b_i$: Skalarprodukt für $a, b \in \mathbb{C}^n$

$|a|$: Betrag von a

$||a||_p$: $(< a, a >^p)^{\frac{1}{p}}$ p-te Norm von a für $p < \infty$

$\lceil x \rceil$: für $n \in \mathbb{N} : n < x \leq n+1$ gilt $\lceil x \rceil := n+1$.

$\lfloor x \rfloor$: für $n \in \mathbb{N} : n \leq x < n+1$ gilt $\lfloor x \rfloor := n$.

B Ergänzungen

B.1 Notationsunterschiede zu Frommer

Im wesentlichen wurde versucht, die Notationen, die Frommer in [5] verwendete, auch zu übernehmen. Jedoch hatte sich in wenigen Fällen gezeigt, daß es der Übersichtlichkeit wegen ratsam war von den Vorgaben von Frommer abzuweichen. Diese Abweichungen seien im folgenden erklärt

- **Strudelgröße**
  in Satz 3.1 wurde die Strudelgröße definiert und in (3.30) als Formel dargestellt. Frommer bezeichnet mit D_i die i-te Strudelgröße. Das i berechnet sich dabei aus dem n welches den $(2i+2)$-ten Schritt bzw. das homogene Polynom mit Grad $(2i+2)$ bezeichnet. Es gilt also $n = (2i+2)$ woraus mit Auflösung nach i folgt $i = \frac{n}{2} - 1$. Daraus ergibt sich die Strudelgröße $d_{\frac{n}{2}-1}$ in direkter Abhängigkeit des Grades des neu zu konstruierenden homogenen Polynoms F_n. Um den Unterschied noch zu unterstreichen wurde die hier verwendete Strudelgröße klein d bezeichnet.

- **Koeffizienten von $Q^{(n)}(x,y)$**
  Die Koeffizienten Des $Q^{(n)}(x,y)$-Polynom aus (3.2) bezeichnete Frommer mit B_{ij}. Um Verwechslungen mit dem Polynom $B(x,y)$ aus (2.8) zu vermeiden seien die Koeffizienten aus $Q^{(n)}(x,y)$ mit $Q_{ij}^{(n)}$ bzeichnet bzw. wenn klar ist um welches n es sich handelt mit Q_{ij}

- **Stammfunktion der Vergleichsdifferentialgleichung $F(x,y)$**
  Frommer konstruiert die Stammfunktion der Vergleichsdifferentialgleichung als $f(x,y)$. Da jedoch gemeinhin Stammfunktionen gerne mit Großbuchstaben definiert werden und auch mit $f(x,y)$ bereits ein Tangentialfeld definiert wurde ist hier $F(x,y)$ für die Stammfunktion bezeichnet. (Vgl. (2.11))

B.2 Landausymbole

Definition B.1 (Komplexität[1]). *Es sei A ein Algorithmus zur Lösung eines Problems P mit $n \in \mathbb{N}$ Eingabedaten. Die Abbildung*

$$T_A : \mathbb{N} \to \mathbb{N},$$

die jeder Anzahl von Eingabedaten die Anzahl der Grundoperationen beim Ablauf des Algorithmus zuordnet, heißt Komplexität von A.

Diese Definition der Komplexität erfaßt noch nicht alle Aspekte, die ein Gütemaß für einen Algorithmus besitzen sollte. So hängt die Laufzeit nicht nur von der Anzahl der Eingabedaten, sondern auch wesentlich von deren Codierungslänge ab. Ebenso ist das Laufzeitverhalten eines Algorithmus nicht unabhängig von dem Maschinentyp, der benutzt wird. Was die Überlegungen zur Komplexität betrifft wird gemeinhin ein einheitlicher Maschinentyp wie etwa eine Turing-Maschine[2] zugrundegelegt. Zur Beschreibung des Verhaltens der Komplexitätsfunktion wird das Landau-Symbol definiert. Insbesondere für große n ist diese Notation in der Numerik gebräuchlich.

[1] Vgl. [8] S.41
[2] Siehe einführende Literatur zur Informatik.

Definition B.2 (Landausymbol). *Sei $D \subset \mathbb{R}$ und $f, g : D \to \mathbb{R}$. Für g gelte zusätzlich $g(x) \neq 0$ für $x \in D$.*

- *f heißt von der Ordnung groß O von g für x gegen x_0, wenn es eine Konstante $\varepsilon > 0$ und ein $\delta > 0$ gibt, so daß für alle $x \in D$ mit $x \neq x_0$ und $|x - x_0| < \delta$ die Abschätzung*

$$\left|\frac{f(x)}{g(x)}\right| \leq \varepsilon$$

gilt. In Kurzschreibweise

$$f(x) = O(g(x)) \text{ für } x \to x_0$$

- *f heißt von der Ordnung klein o von g für x gegen x_0, wenn es für jede Konstante $\varepsilon > 0$ ein $\delta > 0$ gibt, so daß für alle $x \in D$ mit $x \neq x_0$ und $|x - x_0| < \delta$ die Abschätzung*

$$\left|\frac{f(x)}{g(x)}\right| \leq \varepsilon$$

gilt. In Kurzschreibweise

$$f(x) = o(g(x)) \text{ für } x \to x_0$$

Für die Landausymbole gelten dabei folgende Eigenschaften, deren Beweise an dieser Stelle nicht gezeigt werden. Leichte Nachprüfungen bestätigen die Aussagen.

a.) $f(x) = O(g(x))$

b.) $f(x) = o(g(x)) \implies f(x) = O(g(x))$

c.) $f(x) = K \cdot O(g(x))$ für ein $K \in \mathbb{R} \implies f(x) = O(g(x))$

d.) $f(x) = O(g_1(x))$ und $g_1(x) = O(g_2(x)) \implies f(x) = O(g_2(x))$

e.) $f_1(x) = O(g_1(x))$ und $f_2(x) = O(g_2(x)) \implies f_1(x) \cdot f_2(x) = O(g_1(x) \cdot g_2(x))$

f.) $f(x) = O(g_1(x) \cdot g_2(x)) \implies f(x) = g_1(x) \cdot O(g_2(x))$

Die Beziehungen c) bis f) gelten dabei ebenfalls entsprechend für o.

Beispiel B.1 (Beispiel zu Landausymbolen). *Sei $f : [0,1] \to \mathbb{R}$ eine Funktion mit $f(0) = 0$. Wenn f stetig bzw. einmal stetig differenzierbar auf dem Intervall $[0,1]$ ist, so gilt $f(x) = o(1)$ bzw. $f(x) = O(x)$ für $x \to 0$.*

Ein Algorithmus soll grundsätzlich auf eine ganze Klasse von Problemen anwendbar sein. D.h. er soll eine ganze Menge $\Omega := \{w_1, w_2, \dots\}$ von Eingabedaten bearbeiten können. Daraus stellt sich die Frage nach der Komplexität bzgl. der Eingabedaten der Menge Ω, die eine feste Größe $g(w_i) = n$ haben. Somit kann man zum einen nach der Zeitkomplexität im schlechtesten Fall ("worst case") und nach der Zeitkomplexität im Mittel fragen. Dazu setzt man

$$T_A^S(n) := \sup\{T_A(w) | w \in \Omega, \quad g(w) = n\} \tag{B.1}$$

und

$$T_A^M(n) := E\{T_A(w) | w \in \Omega, \quad g(w) = n\}$$

E im zweiten Fall fungiert dabei als Erwartungswert über die bedingte Verteilung $V(w|g(w) = n)$. Es soll aber für die Betrachtungen hier nur die Komplexität im schlechtesten Fall von Interesse sein, da Betrachtungen der Komplexität im Mittel Fragen nach den Wahrscheinlichkeitsmaßen beantworten müßte. Diese Fragen würden jedoch zu weit führen. Die Komplexität im schlechtesten Fall liefert hier genügend Information.

Beispiel B.2 (Naive Auswertung eines Polynoms). *Der naive Algorithmus zur Auswertung eines Polynoms vom Grade n an der Stelle $x_0 \in \mathbb{R}$ lautet*

- *Eingabe:* $n \in \mathbb{N}$, $(a_0, \ldots, a_n) \in \mathbb{R}^{n+1}$, $x_0 \in \mathbb{R}$.
- *Ausgabe:* $p := \sum_{i=0}^{n} a_i x_0^i$
- *Verarbeitung:*

  i. $p := a_0$

  ii. *Für* $i = 1$ *bis* n *{*
  $b := a_i$
  für $j = 1$ *bis* i *{*
  $b := bx_0$
  $p = p + b$ *}*
  }

Der Algorithmus hat bei Zählung der Additionen und Multiplikationen die Komplexität

$$T_A^S(n) = \frac{1}{2}n(n+3) = O(n^2).$$

Zuletzt soll noch die Frage nach dem Laufzeitverhalten[3] eines Algorithmus im schlechtesten Fall erörtert werden. Dazu sei der Begriff der Laufzeit des Algorithmus nachfolgend definiert. Die Komplexität eines Algorithmus war die Anzahl der auszuführenden elementaren Rechenschritte. Zur Ermittlung des Laufzeitverhaltens spielt die Kodierungslänge eine wichtige Rolle, da Algorithmen mit großen Zahlen aufwendiger sind, als mit kleinen Zahlen. Da Rechenanlagen nur rationale Zahlen darstellen können seien die Betrachtungen nur auf Zahlen aus $\mathbb{Q}$ beschränkt. Rechenanlagen codieren Zahlen im binären Format (Nullen und Einsen). Eine Zahl n aus $\mathbb{N}$ wird somit im binären Format mit $\lceil \log_2(n+1) \rceil$ Bits dargestellt bzw. mit $\lceil \log_{256}(n+1) \rceil$ Bytes. Zusätzlich wird ein Bit für das Vorzeichen benötigt. Daraus ergibt sich die Bezeichnung für die Kodierungslänge einer Zahl $n \in \mathbb{Z}$ als

$$\langle n \rangle := \lceil \log_2(|n|+1) \rceil + 1 \tag{B.2}$$

bzw. in Byte-Darstellung

$$\langle\langle n \rangle\rangle := \lceil \log_{256}(|n|+1) + \log_{256}(2) \rceil \tag{B.3}$$

Beispiel B.3. *Sei $r \in \mathbb{Q}$ und $p, q \in \mathbb{Z}$ mit $r := \frac{p}{q}$ und $q > 0$. Dann folgt daraus für die Kodierungslänge*

$$\langle r \rangle = \langle p \rangle + \langle q \rangle$$

in Bits und

$$\langle\langle r \rangle\rangle = \langle\langle p \rangle\rangle + \langle\langle q \rangle\rangle$$

in Bytes. Für eine Matrix $A \in \mathbb{Q}^{m \times n}$, $A = (A_{ij})$ gilt die Kodierungslänge

$$\langle A \rangle = \sum_i^m \sum_j^n \langle A_{ij} \rangle,$$

bzw.

$$\langle\langle A \rangle\rangle = \sum_i^m \sum_j^n \langle\langle A_{ij} \rangle\rangle.$$

[3] Vgl. [8] S.423f

Mit diesen beiden Notationen der Kodierungslänge und der Komplexität im schlechtesten Fall läßt sich nun die Laufzeit eines Algorithmus definieren

Definition B.3 (Laufzeit eines Algorithmus.). *Sei A der Algorithmus um das Problem P zu lösen*

- *Die Laufzeit $t_A^S(P)$ des Algorithmus A zur Lösung des Problems P ist die Anzahl der elementaren Rechenschritte multipliziert mit der maximalen Kodierungslänge der verwendeten Zahlen die der Algorithmus benutzt.*
- *Für den Algorithmus A zur Lösung der Klasse von Problemen $\mathcal{P}$ ist die Laufzeitfunktion die Abbildung*

$$T_A^S : \mathbb{N} \to \mathbb{N},$$

$$T_A^S(n) := \max\{t_A^S(P) | P \in \mathcal{P}, \quad \langle P \rangle \leq n\}$$

  $\langle P \rangle$ gilt dabei als Summe aller Kodierungslängen der Daten, die das Problem P beschreiben.

- *Ein Algorithmus A besitzt polynomialer Laufzeit, wenn es ein Polynom $p : \mathbb{N} \to \mathbb{N}$ gibt, so daß für alle $n \in \mathbb{N}$ die Abschätzung*

$$T_A^S(n) \leq p(n)$$

  gilt. Der Algorithmus A wird dann polynomiale Algorithmus genannt.

C Programm-Code

Nicht der gesamte Programm-Code konnte an dieser Stelle präsentiert werden. Dafür hätte der Platz nicht ausgereicht. Jedoch Auszüge bzw. ein Diagramm, welches die Abhängigkeit der einzelnen Klassen darstellen, sollen hier einen Gesamtüberblick der Struktur bringen. Der Programmcode selbst und eine Dokumentation im HTML-Format können der beiliegenden Diskette entnommen werden.

Zwei wesentliche Felder sind in dem Softwarepaket realisiert. Zum einen Klassen zur Berechnung von Differentialgleichungen und zum anderen Klassen zur grafischen Ausgabe.

- **Magno**
  Die Startklasse Magno ist ein Frontend-Programm zur Berechnung der Frommerschen Strudelgrößen. Es wird dabei der Algorithmus zur Analyse in `doAlgo` bereitgestellt. Mit `java Magno` wird das Startfenster (hinter einem DOS-Prompt) gestartet.
- Differentialgleichungen
  - **DGL**
    Die Klasse DGL hält die Polynome $A(x, y)$ und $B(x, y)$ der Pfaffschen Form $\omega = Adx + Bdy$ bereit in Form von `Polynom2D`-Objekten. Zudem wird sie zum Zeichnen des Gradientenfeldes bzw. einzelner Trajektorien verwendet. Zum Berechnen von Integralkurven für die Zeichnungen wurde ein einfaches Runge-Kutta-Verfahren implementiert.
  - **Polynom2D**
    Die Klasse `Polynom2D` ist die Polynom-Basisklasse. 2D soll dabei für zweidimensional stehen. Die beiden Dimensionen sollen hier also x- und y-Richtung verstanden werden. Die beiden Dimensionen spannen ein Gitter unterschiedlicher Potenzen von x und y auf. Die einzelnen Koeffizienten werden in den Gitterpunkten (`Polynom2DNode`-Objekte) abgelegt. Die Klasse realisiert eine dünnbesetzte Matrix von Gitterpunkten. In vertikaler Ebene werden Zeilen von Koeffizienten gleicher x-Potenz geispeichert. In jeder Zeile sind die Potenzen von y aufsteigend von links nach rechts. In vertikaler Ebene sind die x-Potenzen von oben absteigend nach unten. Es stehen in der Klasse nicht nur get-/set-Routinen für Setzen und Auslesen der Werte, sowie ihre Verbindungen, sondern auch Routinen zur Multiplikation und Addition zur Verfügung.
  - **Polynom2DDiag**
    Die Klasse `Polynom2DDiag` listet all die Koeffizienten auf, in denen die Summe der Potenzen für x und y gleich sind. Die einzelnen homogenen Polynome sind dabei in `Polynom2D`-Objekte für gleiche Potenzen-Summe gespeichert. `Polynom2DDiag` sorgt also letztendlich nur für die ordnungsgemäße Verknüpfung der `Polynom2D`-Objekte untereinander.
  - **Polynom2DNode**
    Die Klasse `Polynom2DNode` speichert einen Koeffizienten eines zweidimensionalen Polynoms. Dabei werden die Daten für die Potenz von x, die Potenz von y und der Wert des Koeffizienten aufgenommen. Der Knoten kann in vier Richtungen verknüpft werden. In `left` ist der Knoten mit der nächsten kleineren Potenz von y gespeichert, in `right` der der nächsten höheren. In `up` wird der Knoten mit der nächsten kleineren Potenz von x geispeichert, analog zu y in `down` die nächste größere. Zum Auslesen bzw. setzen der Werte und der Verknüpfungen stehen get-/set-Routinen zur Verfügung.

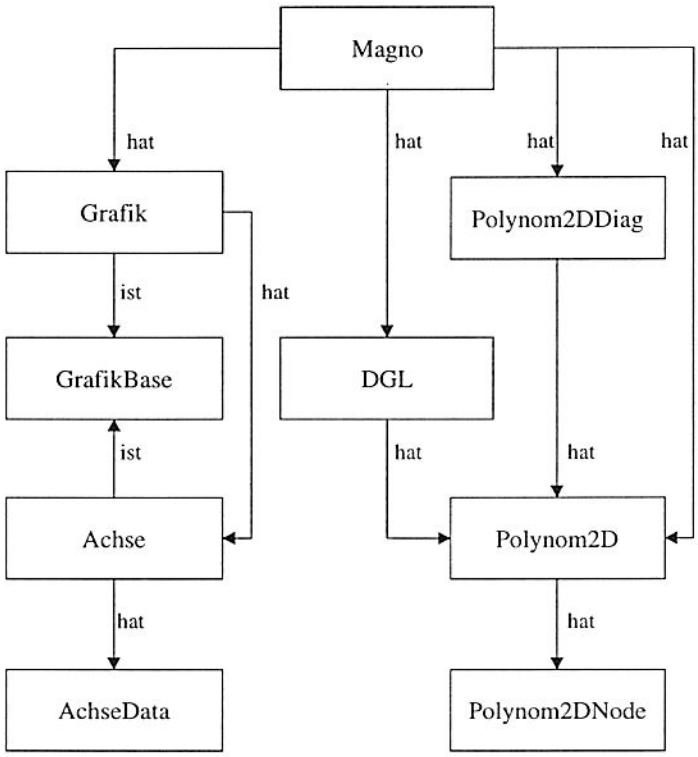

Abbildung C.1: Die Klassenhierarchie teilt sich auf in zwei wesentliche Bereiche. Auf der linken Seite sind die Grafik-Klassen zu sehen mit ihren Beziehungen. Auf der rechten Seite die Beziehungen der Differentialgleichungsklassen. Mit "hat" und "ist" wurde gekenntzeichnet, ob eine Klasse von einer anderen abgeleitet ist (Ist-Beziehung) oder ob ein Objekt einer Klasse Objekte einer anderen Klasse nutzt (Hat-Beziehung).

- Grafik
  - **Grafik**
    Die Grafik-Klasse stellt die Ausgabefläche zur Verfügung in der das Achsenkreuz mit den Trajektorien einer Differentialgleichung und ein Vektorfeld ausgegeben werden kann. Die eigentlich Ausgabe erfolgt in der nächsten Klasse `Achse`.
  - **Achse**
    Durch ein übergebenes Graphic-Objekt wird ein Achsenkreuz und Trajektorien einer Differentialgleichung ausgegeben (siehe `drawImage(Graphics2D)`). Die Klasse Achse stellt dabei lediglich die wesentlichen Besatndteile einer Grafik zusammen (Achsen, Achsenkreuz, Differentialgleichung, Vektorfeld) und gibt sie aus.
  - **GrafikBase**
    Grafiken können zu unterschiedlichen Bezugssystemen gezeichnet werden. So werden z.B. Funktionen in solchen Achsenkreuzen gezeichnet die den Nullpunkt in der linken unteren Ecke eines Bildes besitzen und die fortlaufenden Abtragungen auf den Achsen in positiver Richtung entweder nach rechts (unabhängige Variable) oder nach oben (abhängige Variabel) gesetzt werden. In der Bildverarbeitung bzw. im Einsatz von Grafiken an einem Computer ist man daran gewöhnt

den Nullpunkt in der linken oberen Ecke zu setzen und die Richtungen der positive steigenden Werte der beiden Seiten nach unten und nach rechts festzulegen. GrafikBase bietet nun an eine 2x3 Matrix so mit Koeffizienten zu besetzen, daß man gewöhnliche Grafikfunktionen (Malen von Linien, Kreisen,...) so nutzt als würde man, wie im Beispiel von Achsenkreuzen, den Nullpunkt in der linken unteren Ecke vorfinden. Natürlich können auch beliebige andere Bezugssysteme erstellt werden (Setzen des Nullpunktes in den Mittelpunkt einer Grafik). Mit jedem Aufruf einer elementaren Grafikfunktion werden dann die übergebenen Parameter auf das interne Bezugssystem umgerechnet.

- **AchseData**
  Eine Achse bemißt sich durch ihren Anfangs- und Endpunkt des Ausschnitts einer Dimension, den eine Grafik anzeigen soll. Dabei werden zwei unterschideliche Skalierungen unterschieden: Zum einen die reale, echte Skalierung aus der Natur (z.B. cm, mm, Zeit,...) und zum anderen die Skalierung in Bildpunkten für die interne Darstellung. `AchseData` stellt eine Beziehung her zu echten Skala-Daten und ihre interne Darstellung in einer Grafik. Es werden dabei optimale Abstände und Offsets berechnet für die Formatierung einer Achse.

Abbildungsverzeichnis

Literaturverzeichnis

[1] Birnmeyer, M. [2000]: „*Die Poincaréschen Wirbelbedingungen an der Stelle einer Unbestimmtheit*“, Diplomarbeit am Mathematik-Lehrstuhl VI der Universität Bayreuth

[2] Bronstein, I. N.; Semendjajew, K. A.; Musiol, G.; Mühlig, H. [1993]: „*Taschenbuch der Mathematik*“, Verlag Harri Deutsch, Thun - Frankfurt a. M.

[3] Fischer, W.; Lieb, I. [1990]: „*Funktionentheorie*“, F. Vieweg & Sohn Verlagsgesellschaft mbH, Braunschweig - Wiesbaden

[4] Forster, O. [1981, 1992]: „*Analysis, Bd. I, II & III*“, F. Vieweg & Sohn Verlagsgesellschaft mbH, Braunschweig - Wiesbaden

[5] Frommer, M. [1935]: „*in Mathamtische Analen 109, S.395-424* “, Springer-Verlag, Berlin - Heidelberg - New York - London - Paris - Tokyo - Hong-Kong

[6] Grauert, H.; Fischer, W. [1973]: „*Differential- und Integralrechnung, Bd. II*“, Springer-Verlag, Berlin - Heidelberg - New York - London - Paris - Tokyo - Hong-Kong

[7] Haage, P. [1980]: „*Wilhelm Busch*“, Meyster Verlag GmbH, Wien - München

[8] Hämmerlin, G.; Hoffmann, K.-H. [1994]: „*Numerische Mathematik*“, Springer-Verlag, Berlin - Heidelberg - New York - London - Paris - Tokyo - Hong-Kong

[9] Königsberger, K. [1991]: „*Analysis, Bd. I & II*“, Springer-Verlag, Berlin - Heidelberg - New York - London - Paris - Tokyo - Hong-Kong

[10] Kopka, H. [1994]: „*LaTeX eine Einführung & Erweiterungsmöglichkeiten*“, Addison Wesley, Deutschland

[11] Poincaré, H. [1951]: „*Œvres, Bd. I, "Une équation différentielle"*“, Imprimerie Gauthier-Villares et C^ie, Paris

[12] Press, W. H.; Teukolsky, S. A.; Vetterling, W. T.; Flannery, B. P. [1993]: „*Numerical Recipes in C*“, Cambridge University Press

[13] Reinhardt, F.; Soeder, H. [1974]: „*dtv-Atlas zur Mathematik, Bd. I & II*“, Deutscher Taschenbuch Verlag GmbH & Co KG, München

[14] Siegel, C. L.; Moser, J. K. [1971]: „*Lectures in Celestial Mechanics*“, Springer-Verlag, Berlin - Heidelberg - New York - London - Paris - Tokyo - Hong-Kong

[15] Teichert, A. [2000]: „*Die Strudelbedingungen für $y' = -\frac{A}{B}$ an einer Stelle der Unbestimmtheit*“, Zulassungsarbeit am Mathematik-Lehrstuhl VI der Universität Bayreuth

[16] Stoer, J. [1994]: „*Numerische Mathematik, Bd. I & II*“, Springer-Verlag, Berlin - Heidelberg - New York - London - Paris - Tokyo - Hong-Kong

[17] von Wahl, W. [SS 1998]: „*Gewöhnliche Differentialgleichung*“,
Vorlesung an der Universität Bayreuth

[18] Werner, J. [1992]: „*Numerik, Bd. I & II*“,
F. Vieweg & Sohn Verlagsgesellschaft mbH, Braunschweig - Wiesbaden

Index