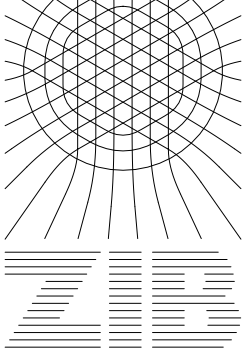
Konrad-Zuse-Zentrum für Informationstechnik Berlin



U. Nowak

Adaptive Linienmethoden für nichtlineare parabolische Systeme in einer Raumdimension

Technical Report TR 93–14 (December 1993)

U. Nowak

Adaptive Linienmethoden für nichtlineare parabolische Systeme in einer Raumdimension

Abstract

A new method for the numerical solution of highly nonlinear, coupled systems of parabolic differential equations in one space dimension is presented. The approach is based on a classical method of lines treatment. Time discretization is done by means of the semi–implicit Euler discretization. Space discretization is done with finite differences on non–uniform grids.

Both basic discretizations are coupled with extrapolation techniques. With respect to time the extrapolation is of variable order whereas just one extrapolation step is done in space. Based on local error estimates for both, the time and the space discretization error, the accuracy of the numerical approximation is controlled and the discretization stepsizes are adapted automatically and simultaneously. Besides the local adaptation of the space grids after each integration step (static regridding), the grid may even move within each integration step (dynamic regridding).

Thus, the whole algorithm has a high degree of adaptivity. Due to this fact, challenging problems from applications can be solved in an efficient and robust way.

Inhaltsverzeichnis

1 Einleitung **1**

2 Nichtuniforme Diskretisierungen **6**

2.1 Genaue Problemformulierung 6

2.2 Ortsdiskretisierung mit Finiten Differenzen 8

2.2.1 Untersuchung eines Modellproblems 9

2.2.2 Der allgemeine Fall 15

2.2.3 Zusammenfassung 23

2.3 Zeitdiskretisierung mit semi–implizitem Euler 26

2.3.1 Gewöhnliche Differentialgleichungen 27

2.3.2 Differentiell–algebraische Systeme 31

2.3.3 Basisdiskretisierung des Gesamtverfahrens 36

2.3.4 Angenommene asymptotische Entwicklung des globalen Fehlers 38

2.3.5 Ein numerisches Experiment – Teil I 39

3 Statische Gittererneuerung **47**

3.1 Gekoppelte Extrapolation in Raum und Zeit 47

3.1.1 2–D–Polynomextrapolation 49

3.1.2 Ein numerisches Experiment – Teil II 54

3.2 Fehlerschätzung 60

3.3 Adaptive Wahl von Zeitschrittweite und Raumgitter 65

3.3.1 Simultane Bestimmung der Grundschrittweiten 65

3.3.2 Konsequenzen aus Gitterwechsel 69

3.3.3 Das Interpolationsproblem 71

3.3.4 Lokale Gitteranpassung 77

3.4 Gesamtalgorithmus 81

4 Mitbewegtes Gitter **86**

4.1 Transformation der Gleichung 87

4.1.1 Ein Modellproblem 87

4.1.2 Der allgemeine Fall 89

4.1.3 A priori Bestimmung des Gitters 90

4.2 Gleichungen zur Gitterankopplung 91
4.2.1 Minimierung eines Zielfunktionals 91
4.2.2 Das Problem der Knotenüberkreuzung 93
4.2.3 Weitere Regularisierung der Gittergleichungen 94
4.2.4 Zwei allgemein verwendbare Gittergleichungen 96
4.2.5 Analyse und Vergleich 99
4.2.6 Spezielle Gitterbewegungsvarianten 104
4.3 Einbindung in das Gesamtverfahren 107
4.3.1 Die semi–diskreten Gleichungen 107
4.3.2 Fehlerkontrolle 108
4.3.3 Illustration 109

5 Programmpaket PDEX1M 112
5.1 Details der algorithmischen Realisierung 112
5.1.1 Wahl der Norm und interne Skalierung 112
5.1.2 Lineare Algebra 114
5.1.3 Berechnung der Jacobimatrix 115
5.2 Programmstruktur 116
5.2.1 Allgemeines 116
5.2.2 Standard–Benutzerschnittstelle 117

6 Numerische Ergebnisse 123
6.1 Ein Effizienzvergleich 124
6.2 Verhalten an typischen Testproblemen 129
6.2.1 Die Beispiele von Sincovec/Madsen 130
6.2.2 Wandernde Fronten 143
6.3 Ein Problem aus der Praxis 152

Zusammenfassung 169

Literaturverzeichnis 170

1. Einleitung

Die mathematische Modellierung vieler interessanter Phänomene aus Natur– und Ingenieurwissenschaften führt auf zeitabhängige partielle Differentialgleichungen, deren numerische Lösung eine herausfordernde Aufgabe ist. Auch die Simulation von räumlich nur eindimensionalen Modellen kann bereits ein wertvolles Instrument bei der Untersuchung dieser Phänomene sein. Bei zunehmender Realitätsnähe der Modellierung entstehen sehr rasch hoch nichtlineare, gekoppelte Systeme.

Damit die numerische Simulation ein hilfreiches Werkzeug in der Hand des Naturwissenschaftlers und Ingenieurs ist, muß das Lösungsverfahren sicher und effizient arbeiten. Dies bedingt *Adaptivität* des Verfahrens und Implementierung in ein einfach zu bedienendes Programmpaket. Die vorliegende Arbeit hat genau diese Zielsetzung.

Die zentrale Rolle der Adaptivität hat im wesentlichen zwei Gründe. Nur mit adaptiver Wahl der Diskretisierungsschrittweiten kann ein zumindest hohes Maß an Sicherheit in folgendem Sinn erreicht werden: die bestmöglich schätzbaren Fehler von Raum– *und* Zeitdiskretisiering bleiben unter einer vorgegebenen Schranke. Nur mit lokaler Anpassung der Schrittweiten – in Raum *und* Zeit – ist eine effiziente Lösung von Problemen mit lokalen Effekten möglich.

Insbesondere letzteres Ziel streben die meisten in der Literatur beschriebenen Lösungsverfahren für parabolische Systeme an. Betrachtet man diese Verfahren etwas genauer, so läßt sich ein gewisser Trend feststellen. Bedingt durch das hohe Maß an Qualität, welches Integrationsverfahren zur Lösung von gewöhnlichen Differentialgleichungssystemen (und differentiell–algebraischen Systemen) inzwischen erreicht haben, ist der Lösungsansatz "Linienmethode" offensichtlich sehr attraktiv.

Bei einer klassischen Linienmethode wird zunächst eine Ortsdiskretisierung durchgeführt, etwa mit Finiten Differenzen, Finiten Elementen oder Kollokation. Das erhaltene sog. semi–diskrete System stellt ein System von i.a. steifen Differentialgleichungen dar. Führt man die Ortsdiskretisierung nur einmal, d.h. vor Beginn der Zeitintegration aus, so kann anschließend praktisch jeder moderne steife Zeitintegrator sehr effizient als Löser verwendet werden. Diese Integratoren zeigen ein sehr hohes Maß an Adaptivität: so bieten sowohl Verfahren vom BDF–Typ als auch Extrapolationsverfahren nicht nur die adaptive, automatische Wahl der Zeitschrittweite an, sondern auch die der Verfahrensordnung.

Die klassische Linienmethode legt a priori ein zu wählendes Gitter fest (uniform oder auch bereits lokal angepaßt), was ein Nachteil gerade für solche Probleme ist, bei denen sich der kritische räumliche Bereich im Laufe der Zeit ändert oder vorher nicht bekannt ist. Insbesondere ist dabei zu beden-

ken, daß die Fehlerkontrolle des Zeitintegrators den Ortsdiskretisierungsfehler praktisch nicht berücksichtigt. Erst wenn die Ortsdiskretisierung so schlecht ist, daß das semi–diskrete System instabil wird, macht sich die Güte der Ortsdiskretisierung auch im Zeitintegrationsprozeß bemerkbar.

Um auch Adaptivität in der Ortsdikretisierung zu erreichen, wurden in den letzten Jahren eine ganze Reihe von Verfahren entwickelt, die dieses Ziel mit Hilfe von sog. statischem Regridding, dynamischem Regridding, oder einer Kombination beider Typen zu erreichen versuchen. Dabei ist die Art der Ortsdiskretisierung von eher untergeordneter Bedeutung.

Beim *dynamischen* Regridding handelt es sich um eine Technik, bei der ein Ortsdiskretisierungsgitter, mit einer festen Anzahl von Knoten, im Laufe der Zeitintegration mitbewegt wird. Für diese Technik werden häufig die Arbeiten Harten/Hymann [39], Miller [55], Miller/Miller [56] zitiert. Zwei Ziele können damit verfolgt werden. Zum einen eine Bewegung der Knoten in "kritische Bereiche", zum anderen eine Bewegung, um die Schwierigkeit des Zeitintegrationsproblems zu entschärfen. Dabei schließen sich diese Ziele nicht unbedingt aus. In seiner reinen Form, d.h. ohne Adaptivität in der Knotenzahl des Gitters, wird dieses Konzept z.B. von Verwer/Blom/Sanz–Serna [78], Zegeling/Blom [80] verwendet, um auch einige neuere Arbeiten zu nennen. Da sich die Dimension des semi–diskreten Systems nicht ändert, ist die Zeitintegration, zumindest formal, nicht gestört. Die Grenze dieser Technik ist offensichtlich. Es muß eine a priori Wahl der Knotenzahl getroffen werden. Diese Anzahl kann, über die gesamte Integrationszeit gesehen, sowohl zu klein sein (was zu einem großen Ortsdiskretisierungsfehler, d.h. Verlust an Robustheit, führt) oder zu groß sein (was eine ineffiziente Integration zur Folge hat).

Beim *statischen* Regridding wird hingegen versucht, durch Einfügen bzw. Eliminieren von Knoten auch während der Zeitintegration, das Gitter "möglichst gut" anzupassen. Eine entscheidende Frage bei dieser Technik ist, wie oft ein derartiges Regridding durchzuführen ist. Es ergibt sich ein Konflikt mit der Zeitintegration, die nur auf einem nicht geänderten semi–diskreten Problem ungestört fortschreiten kann.

Bei beiden Arten der Gittersteuerung ergibt sich die Frage, wie die Begriffe "möglichst gut" oder "kritischer Bereich" zu quantifizieren sind. In der Regel werden dazu mehr oder weniger heuristische Kriterien verwendet. Oft erfolgt eine Quantifizierung mittels einer sog. Monitorfunktion, in welche die Größe des Gradienten und/oder der Krümmung eingehen.

Eine ganze Reihe von Verfahren verwenden auch beide Arten der Gitteranpassung. Als ein moderner Vertreter dieser Gattung, ist das von Petzold [66, 67] entwickelte Verfahren zu nennen. In Verbindung mit dem auf einem BDF–Verfahren basierenden Code DASSL [64] wird dabei ein spezieller

Moving–Grid–Ansatz ausgewählt. Zusätzlich erfolgt ein statisches Regridding mit einer Monitorfunktion des oben genannten Typs. Das Verfahren ist auch im Ort adaptiv, aber es erfolgt keine Ortsfehlerschätzung oder gar Fehlerkontrolle im Ort. Über die Häufigkeit der statischen Gitteranpassung wird nichts gesagt.

Was die Ortsfehlerkontrolle anbelangt, so ist die Methode von Biermann/Babuška [8], im Vergleich mit allen bisher genannten Verfahren, besonders ausgereift. Hier wird die Zeitintegration mit dem BDF–Code LSODI [40] durchgeführt. Für den Ortsdiskretisierungsfehler wird in [8] ein neuer Fehlerschätzer entwickelt. Dabei ist die Schätzung örtlich lokal, d.h. für jeden Knoten des Gitters liegt eine eigene Schätzung vor. Basierend auf dieser Information wird ein statisches Regridding derart durchgeführt, daß der (geschätzte) Fehler kleiner als eine vorgegebene Toleranz wird. Diese Schätzung und Adaptierung wird allerdings nur an fest vorzugebenden Zeitpunkten durchgeführt. Auch wenn dabei der geschätzte Fehler größer als die vorgegebene Toleranz ist, wird die Lösung akzeptiert. Zwischen den Testzeitpunkten wird der Ortsfehler nicht kontrolliert und das Gitter nicht angepaßt. Dieser Nachteil der Methode ist durch die Wahl eines Mehrschrittverfahrens als Zeitintegrator bedingt. Das Verfahren muß nach jeder Adaptierung des Gitters mit niedrigster Ordnung neu gestartet werden. Passiert dies zu häufig, so ist der eigentliche Vorteil des Linienmethodenansatzes verloren.

Dreht man die Reihenfolge der Diskretisierung um, so erhält man statt eines reinen Anfangswertproblems in der Zeit eine Folge von reinen Randwertproblemen im Ort. Diese könnten nun mit einem entsprechenden hochentwickelten, adaptiven Löser (etwa COLSYS [3, 5]) gelöst werden. Damit bliebe jedoch wiederum die Frage nach Adaptivität in der Zeit übrig. Zumindest vom Prinzip her ist die von Smooke/Koszykowski [74] entwickelte Methode von diesem Typ. Allerdings wird die adaptive Schrittweitensteuerung, sowohl im Ort als auch in der Zeit, mittels heuristischer Kriterien durchgeführt.

Die letztgenannte Reihenfolge der Diskretisierung liegt auch einer erst kürzlich begonnenen Verfahrensentwicklung zugrunde. Bei diesem Verfahrenstyp, wie er erstmals in Bornemann [9, 10] für parabolische Probleme vorgestellt wurde, werden die höchst erfolgreichen adaptiven Multi–Level–Konzepte zur Lösung von elliptischen Problemen in mehreren Raumdimensionen, vgl. etwa Bank/Dupont/Yserentant [7], Deuflhard/Leinen/Yserentant [21], mit einer Zeitdiskretisierung im Funktionenraum (Rothe-Methode) kombiniert. Vorläufig ist die in [9, 10] behandelte Problemklasse noch recht einfach (eine skalare, lineare parabolische Gleichung), wenn auch in Bornemann [11] bereits der zweidimensionale Fall betrachtet wird. Erste Erweiterungen in Richtung auf die hier betrachtete Problemklasse sind jedoch erfolgversprechend, vgl. Lang/Walter [46]. Das Konzept dieser neuen Verfahrensklasse ist von einem höchsten Maß an Adaptivität geprägt. Auch in-

nerhalb eines Zeitschrittes werden die verwendeten Gitter noch adaptiv angepaßt. Insbesondere für den räumlich mehrdimensionalen Fall dürfte sich dies in Zukunft auszahlen.

In der Mehrzahl naturwissenschaftlich–technischer Anwendungen wird bisher nahezu ausschließlich die Linienmethode verwendet. In der vorliegenden Arbeit soll deshalb das Potential der Linienmethode bezüglich Adaptivität voll ausgeschöpft werden. Sowohl statisches als auch dynamisches Regridding wird durchgeführt. Darüber hinaus soll der Fehler im Ort kontrolliert werden, und zwar mit gleicher Qualität wie der Zeitfehler. Für die Diskretisierungen in der Zeit wird auf Extrapolationsmethoden zurückgegriffen (semi–implizite Euler–Diskretisierung). Für die Diskretisierung im Ort wird, wegen der natürlichen Nähe zur Extrapolation, ein finites Differenzenverfahren ausgewählt. Interessante Anwendungsprobleme aus den Natur– und Ingenieurwissenschaften enthalten häufig äußerst lokalisierte Phänomene. Deshalb soll von Anfang an konzeptionell auf im allgemeinen *nicht-uniforme* Gitter gesetzt werden. Für diesen Fall von Diskretisierung existieren in der Literatur kaum globale Konvergenzaussagen, allenfalls für einfache Spezialfälle, siehe etwa Manteuffel/White [54]. Will man eine hinreichend weite Klasse von interessanten partiellen Differentialgleichungen aus den Anwendungen erfassen, so bleibt nur der Rückgriff auf *lokale* Fehlerstrukturen. Anstelle von Theoremen für eher einfache, aber kaum anwendungsrelevante Probleme tritt deshalb in der vorliegenden Arbeit die sorgfältige mathematische Untersuchung lokaler Fehlerentwicklungen (nach Diskretisierungschrittweiten) und deren penible Umsetzung in einen funktionsfähigen, effizienten Algorithmus. Von Fall zu Fall werden für einfache Probleme globale Diskretisierungsordnungen illustriert.

Natürlich ist es nicht möglich, mit diesem Ansatz alle interessierenden partiellen Differentialgleichungen gleichermaßen befriedigend zu erfassen. Immerhin eignet sich der hier vorgestellte Ansatz jedoch für eine Klasse von partiellen Differentialgleichungen, die als "parabolisch dominant" bezeichnet werden kann. Für diese Klasse gilt:

Der Charakter der partiellen Differentialgleichungen ist dissipativ genug, um den Einfluß lokaler Fehler nicht zu verstärken.

Dies ist eine etwas schwächere Form der Annahme über behandelbare Probleme als die von Bietermann/Babuška [8] verwendete. Sie fordern, daß die lokalen Fehler sogar gedämpft werden. Entsprechend genügt die Kontrolle lokaler Fehler. Dieses Vorgehen ist in Übereinstimmung mit dem üblichen algorithmischen Vorgehen bei gewöhnlichen Differentialgleichungen.

Die Arbeit gliedert sich in sechs Kapitel.

Im Anschluß an diese Einleitung folgt in **Kapitel 2** zunächst die Darstellung nicht–uniformer Diskretisierungen in Zeit und Raum im synoptischen

Vergleich. Die lokalen Fehlerentwicklungen, als Grundlage des darzustellenden Algorithmus, werden bereitgestellt. Im Raum wird auf finite Differenzen, in der Zeit auf semi–implizite Euler–Diskretisierung spezialisiert.

Die beiden folgenden Kapitel behandeln die Anpassung der Raumgitter in Abhängigkeit von der Zeit. In **Kapitel 3** werden zunächst Techniken der statischen Gittererneuerung ("static regridding") behandelt. Bei Wechsel der Gitter innerhalb einer Zeitschicht ist besonderes Augenmerk auf die Interpolation zu richten sowie auf die dadurch eingeschleppten Interpolationsfehler. Glattheit und Monotonie von Interpolationsformeln sind für partielle Differentialgleichungen wichtige Eigenschaften, deren Implementierung jedoch wiederum die lokalen Fehlerentwicklungen beeinflußt. Insbesondere können sich hier Fehlerordnungen verschlechtern, obwohl sich der Fehler insgesamt verbessert! **Kapitel 4** widmet sich der dynamischen Gitteranpassung ("moving grid"). Diese Technik empfiehlt sich speziell für Probleme, bei denen aus dem naturwissenschaftlichen Kontext wandernde Fronten zu erwarten sind. Im Rahmen der Linienmethode entstehen dabei implizite, blockstrukturierte gewöhnliche Differentialgleichungssysteme. Darüber hinaus erfordert dieser Ansatz noch eine zusätzliche Regularisierung, um ein "Übereinanderlaufen" von Gitterknoten zu verhindern.

Auf dieser Basis wird in **Kapitel 5** das hier propagierte adaptive Verfahren vorgestellt und zwar in der präzisen Form eines umfangreichen Programmpakets (Code PDEX1M). Zahlreiche Überlegungen werden angeführt, welche für die Effizienz und Robustheit des vorgestellten Algorithmus unverzichtbar sind. Abschließend, in **Kapitel 6**, wird das Verhalten des Algorithmus an einer Reihe von partiellen Differentialgleichungen illustriert. Der Nachweis der Effizienz ist für die komplizierte Problemklasse nicht mehr in allgemeiner, methodisch sauberer Form zu führen. Deswegen wird in Kapitel 6.1 anhand eines in der Literatur wohl dokumentierten und häufig als Testproblem herangezogenen Beispiels ein Vergleich mit einer schon recht effizienten, rein zeitadaptiven Linienmethode vorgestellt (basierend alternativ auf Linienmethode bei festem Raumgitter mit Integratoren DASSL von Petzold [64] bzw. LIMEX von Deuflhard/Nowak [23]). In Kapitel 6.2.1 wird die von Sincovec/Madsen [72] zusammengestellte Beispielsammlung mit PDEX1M duchgerechnet und in Kapitel 6.2.2 einige weitere häufig verwendete Testprobleme, deren Charakteristikum wandernde Fronten sind. Dies geschieht mit einem einzigen Code ohne problemabhängiges "Tuning" von Parametern. Die Robustheit des Verfahrens wird somit dokumentiert. Schließlich wird ein Beispiel aus dem Ingenieurbereich gelöst, nämlich die katalytische Abgasreinigung in einem Autokatalysator – mit voller Komplexität eines realen Problems. Da der vorliegende Algorithmus für eben solche Probleme entwickelt worden ist, kommt der erfolgreichen Bewältigung dieses speziellen Anwendungsproblems eine wesentliche Rolle zu.

2. Nichtuniforme Diskretisierungen

Im folgenden sollen die grundlegenden Approximationseigenschaften der verwendeten Ort–und Zeitdiskretisierungsschemata untersucht werden. Die Analyse beschränkt sich i.w. auf die Untersuchung des Diskretisierungsfehlers eines Zeitintegrationsschrittes auf einem nicht uniformen Ortsdiskretisierungsgitter. Zunächst wird die verwendete Ortsdiskretisierung vorgestellt und der daraus herrührende lokale Diskretisierungsfehler abgeleitet. Danach wird auf die Zeitdiskretisierung eingegangen und abschließend wird der angenommene globale Diskretisierungsfehler der Gesamtdiskretisierung angegeben. Vorab soll jedoch die behandelbare Problemklasse im notwendigen Detail beschrieben werden.

2.1 Genaue Problemformulierung

Die vorliegende Arbeit beschäftigt sich mit der Entwicklung von adaptiven numerischen Verfahren zur Lösung von Systemen gekoppelter, nichtlinearer partieller Differentialgleichungen (PDG) in einer Raumdimension. Das Problem soll in kartesischen, zylindrischen oder sphärischen Koordinaten formulierbar sein und die Modellgleichungen in eine allgemeine Problemklasse fallen, die in der folgenden Form beschreibbar ist:

Integrationsintervall:

$$t \in [t_0, t_{end}] \subset \mathbb{R}^1 \ , \quad -\infty < t_0 < t_{end} < \infty \tag{2.1}$$

Ortsgebiet:

$$x \in \Omega := [x_L, x_R] \subset \mathbb{R}^1 \ , \quad -\infty < x_L < x_R < \infty \tag{2.2}$$

Dimension:

$$u(x,t) \in \mathbb{R}^{n_{pde}} \ , \quad n_{pde} \geq 1 \tag{2.3}$$

Gleichung:

$$\begin{gathered} B(x,t,u,u_x)u_t = f(x,t,u,u_x,\mathcal{D}(x,t,u)) \\ t \in (t_0, t_{end}] \ , \quad x \in \dot{\Omega} \end{gathered} \tag{2.4}$$

Anfangsbedingung:

$$u(x,t_0) = u_0(x) \ \ , t = t_0 \ , \quad x \in \Omega \tag{2.5}$$

Randbedingung:

$$\begin{gathered} \text{a)} \qquad \alpha(x)u + \beta(x,t,u)u_x = \gamma(x,t,u) \\ \text{oder} \\ \text{b)} \qquad u_t(x,t) = \delta(x,t,u) \\ x = x_L \text{ oder } x_R \, , \ t \in (t_0, t_{end}] \ . \end{gathered} \tag{2.6}$$

Hierbei bezeichnet $\mathcal{D}$ einen eventuell von Ort, Zeit und Lösung nichtlinear abhängenden elliptischen Differentialoperator der Form

$$\mathcal{D}(x,t,u) := \frac{1}{x^c}(x^c D(x,t,u)\cdot u_x)_x \ . \tag{2.7}$$

mit

$$c = \begin{cases} 0\,, & \text{für kartesische Koordinaten} \\ 1\,, & \text{für Zylinderkoordinaten} \\ 2\,, & \text{für Kugelkoordinaten}\ . \end{cases} \tag{2.8}$$

Die Funktionen und Matrizen in (2.4–2.6) sollen folgende Dimensionen bzw. Formen haben:

$$\begin{aligned} B &: \mathbb{R}\times\mathbb{R}\times\mathbb{R}^{n_{pde}}\times\mathbb{R}^{n_{pde}} \to \mathbb{R}^{n_{pde}\times n_{pde}} \\ f &: \mathbb{R}\times\mathbb{R}\times\mathbb{R}^{n_{pde}}\times\mathbb{R}^{n_{pde}}\times\mathbb{R}^{n_{pde}\times n_{pde}} \to \mathbb{R}^{n_{pde}} \\ D &: \mathbb{R}\times\mathbb{R}\times\mathbb{R}^{n_{pde}} \to \mathbb{R}^{n_{pde}\times n_{pde}} \\ u_0 &: \mathbb{R} \to \mathbb{R}^{n_{pde}} \\ \alpha &: \mathbb{R} \to \mathbb{R}^{n_{pde}\times n_{pde}} \\ \beta &: \mathbb{R}\times\mathbb{R}\times\mathbb{R}^{n_{pde}} \to \mathbb{R}^{n_{pde}\times n_{pde}} \\ \gamma &: \mathbb{R}\times\mathbb{R}\times\mathbb{R}^{n_{pde}} \to \mathbb{R}^{n_{pde}\times n_{pde}} \\ \delta &: \mathbb{R}\times\mathbb{R}\times\mathbb{R}^{n_{pde}} \to \mathbb{R}^{n_{pde}} \end{aligned}$$

wobei die Matrizen α, β, γ Diagonalgestalt haben sollen:

$$\begin{aligned} \alpha &:= \operatorname{diag}(\alpha_1,\ldots,\alpha_{n_{pde}}) \\ \beta &:= \operatorname{diag}(\beta_1,\ldots,\beta_{n_{pde}}) \\ \gamma &:= \operatorname{diag}(\gamma_1,\ldots,\gamma_{n_{pde}})\ . \end{aligned}$$

Die redundante Form der Randbedingung (2.6.a) wurde gewählt, um Problemstellung und Softwareschnittstelle in Einklang zu bringen. Für die Benutzerschnittstelle ist gerade die Form (2.6.a) weit verbreitet. Auf Randbedingungen der Form (2.6.b) wird im folgenden nicht näher eingegangen, da ihre Behandlung im Kontext des hier vorgestellten Verfahrens trivial ist.

Handelt es sich bei dem durch (2.1–2.7) gegebenen Problem um ein System von Gleichungen, so bezeichne u_j die j–te Komponente des Vektors

$$u = (u_1, ..., u_{n_{pde}})^T \in \mathbb{R}^{n_{pde}}\ . \tag{2.9}$$

Unter dem Argument u_x in der Matrix B bzw. in der Funktion f ist dann der gesamte Vektor $u_x \in \mathbb{R}^{n_{pde}}$ zu verstehen, und unter dem Differentialoperator (2.7) soll es sich um eine (n_{pde}, n_{pde})–Matrix $\mathcal{D}$ mit Elementen

$$\mathcal{D}_{j,k} = \frac{1}{x^c}\frac{\partial}{\partial x}\left(x^c D_{j,k}(x,t,u)\frac{\partial u_k}{\partial x}\right) \tag{2.10}$$

handeln. Unter der Verknüpfung

$$D(x,t,u) \cdot u_x$$

ist also das normale Matrizenprodukt

$$D(x,t,u) \cdot \mathrm{diag}\left(\partial u_1/\partial x, \ldots, \partial u_{n_x}/\partial x\right)$$

zu verstehen.

Üblicherweise wird der Differentialoperator $\mathcal{D}$ linear in die rechte Seite der PDG eingehen. Bei (2.4) handelt es sich also meist um quasi–lineare Gleichungen.

Auf die Fragen der Existenz und Eindeutigkeut einer Lösung soll hier nicht eingegangen werden; stattdessen sei auf das Buch von Smoller [73] verwiesen. Vielmehr wird angenommen, daß eine Lösung $u(x,t)$ existiert und eindeutig ist. Auch sollen keine klassisch formulierten Voraussetzungen an die Eigenschaften der das Problem beschreibenden Funktionen gemacht werden. Das System soll jedoch einen dominant parabolischen Charakter haben, wie in der Einleitung bereits ausgeführt.

Als formal notwendige Voraussetzungen für die Anwendbarkeit des hier entwickelten Algorithmus sind zu nennen:

- $B \neq 0$
- B, f, D seien mehrfach stetig differenzierbar nach allen Argumenten

Zugelassen ist jedoch:

- B singulär, aber derart, daß für den Index des hier betrachteten, zugehörigen semi–diskreten Systems gilt: Index ≤ 1
- Ortsableitungsterme dürfen in einem Teil der Gleichungen völlig fehlen

2.2 Ortsdiskretisierung mit Finiten Differenzen

Um die angestellten Betrachtungen nicht durch zunächst unnötige Notation zu überladen, wird anstelle der recht allgemeinen Problemformulierung (2.1–2.6) zunächst ein vereinfachtes Modellproblem betrachtet. Inwieweit die daran hergeleiteten Ergebnisse auch für den allgemeinen Fall gültig sind, wird anschließend diskutiert.

2.2.1 Untersuchung eines Modellproblems

Als Modellproblem wird eine skalare, nichtlineare parabolische Gleichung mit nichtlinearen Randbedingungen vom Dirichlet–Typ betrachtet.

$$\begin{aligned}
&a) \quad x \in [x_L, x_R] \ , \quad t \in [t_0, t_{end}] \\
&b) \quad u_t = d(x,t,u)u_{xx} + c(x,t,u)u_x + g(x,t,u) \\
&\qquad =: L(x,t,u,u_x,u_{xx)} \\
&\qquad \text{für } x \in (x_L, x_R) \text{ und } t \in (t_0, t_{end}] \\
&c) \quad u(x,t_0) = u_0(x) \\
&d) \quad \alpha_L u = \gamma_L(t,u) \ ; \ x = x_L \ , t > t_0 \\
&\qquad \alpha_R u = \gamma_R(t,u) \ ; \ x = x_R \ , t > t_0 \ .
\end{aligned} \tag{2.11}$$

Um aus (2.11) ein semi–diskretes System von gewöhnlichen Differentialgleichungen zu erzeugen, müssen die Differentialoperatoren $\partial/\partial x$ bzw. $\partial^2/(\partial x)^2$ durch geeignete Approximationen ersetzt werden. Dazu sei ein Ortsgitter

$$\mathcal{G} := \{x_i, \ i = 1, ..., n_x\} \tag{2.12}$$

gegeben. Dabei bezeichnet n_x die Anzahl der Knoten oder Gitterpunkte, für die gelte:

$$x_L = x_1 < x_2 < ... < x_{n_x - 1} < x_{n_x} = x_R \ . \tag{2.13}$$

Im folgenden bezeichne

$$\begin{aligned}
\bar{\mathcal{G}} &:= \{x_1, x_{n_x}\} \\
\dot{\mathcal{G}} &:= \{x_i\}_{i=2,...,n_x-1}
\end{aligned} \tag{2.14}$$

die Menge der Randpunkte bzw. die inneren Punkte des Gitters $\mathcal{G}$. Die Gitterabstände seien durch

$$\Delta x_i := x_{i+1} - x_i \ , \quad i = 1, \ldots, n_x - 1$$

definiert.

Approximation des Gradienten

Zur Approximation der 1. Ableitung $u_x(x,t)$ für $x \in \dot{\mathcal{G}}$ mit zentralen finiten Differenzen über maximal 3 Knoten stehen nun zwei Varianten zur Verfügung. Die Diskretisierung

$$\frac{\partial}{\partial x} u(x_i, t) \to \Delta_x u(x_i, t) \tag{2.15}$$

kann mit der "klassischen" Approximation

$$\begin{aligned} \Delta_x^0 u(x_i,t) &:= \frac{u_{i+1}-u_{i-1}}{x_{i+1}-x_{i-1}} \\ &= \frac{u_{i+1}-u_{i-1}}{\Delta x_{i-1}+\Delta x_i} \end{aligned} \tag{2.16}$$

oder durch eine gewichtete Version (siehe z.B. PEARSON [61])

$$\begin{aligned} \Delta_x u(x_i,t) &:= \frac{1}{\Delta x_{i-1}+\Delta x_i}\left[\Delta x_{i-1}\frac{u_{i+1}-u_i}{\Delta x_i}+\Delta x_i\frac{u_i-u_{i-1}}{\Delta x_{i-1}}\right] \\ &= \frac{1}{\Delta x_{i-1}+\Delta x_i}\left[-\frac{\Delta x_i}{\Delta x_{i-1}}u_{i-1}+\frac{\Delta x_i^2-\Delta x_{i-1}^2}{\Delta x_{i-1}\Delta x_i}u_i+\frac{\Delta x_{i-1}}{\Delta x_i}u_{i+1}\right] \end{aligned} \tag{2.17}$$

erfolgen. Dabei ist u_i eine abkürzende Schreibweise für $u(x_i,t)$. Im Fall eines äquidistanten Gitters, d.h.

$$\Delta x_i = \Delta x\ ,\ i=1,\ldots,n_x-1\ ;\ \Delta x = \frac{x_R-x_L}{n_x-1}$$

stimmt (2.16) mit (2.17) überein. Für nichtuniforme Gitter unterscheiden sich dagegen die beiden Approximationen in ihrer Approximationsgüte. Der lokale Diskretisierungsfehler der Diskretisierungen (2.16) bzw. (2.17) ergibt sich zu:

$$\begin{aligned} \tau^0(x_i,t) &:= \Delta_x^0 u(x_i,t) - \frac{\partial}{\partial x}u(x_i,t) \\ &= \sum_{k=2}^{K}\frac{1}{k!}\frac{(-1)^{k-1}\Delta x_{i-1}^k+\Delta x_i^k}{\Delta x_{i-1}+\Delta x_i}u^{(k)}(x_i,t)\ +\ O(\widehat{\Delta x}^K) \end{aligned}$$

bzw.

$$\begin{aligned} \tau^1(x_i,t) &:= \Delta_x u(x_i,t) - \frac{\partial}{\partial x}u(x_i,t) \\ &= \sum_{k=3}^{K}\frac{1}{k!}\frac{(-1)^{k-1}\Delta x_{i-1}^{k-1}\Delta x_i+\Delta x_{i-1}\Delta x_i^{k-1}}{\Delta x_{i-1}+\Delta x_i}u^{(k)}(x_i,t)\ +\ O(\widehat{\Delta x}^K)\ . \end{aligned} \tag{2.18}$$

Die obigen Entwicklungen der lokalen Diskretisierungsfehler ergeben sich durch Taylorentwicklungen von u_{i-1} bzw. u_{i+1} um u_i und Zusammenfassung der entsprechenden Glieder. Es wird also angenommen, daß $u(x,t)$ entsprechend oft ($(K+1)$–mal) stetig differenzierbar ist und die Restglieder der Taylorentwicklung beschränkt bleiben. Dabei sei $\widehat{\Delta x} = \max\{\Delta x_{i-1},\Delta x_i\}$ und $u^{(k)}$ die k–te partielle Ableitung von u nach x.

Der wesentliche Unterschied der beiden Entwicklungen ist die Ordnung des führenden Fehlerterms. Für (2.16) erhält man

$$\tau^0(x_i,t) \doteq \frac{1}{2}(\Delta x_i - \Delta x_{i-1})u^{(2)}(x_i,t) \ ,$$

während sich für die Approximation (2.17)

$$\tau^1(x_i,t) \doteq \frac{1}{6}\Delta x_{i-1}\Delta x_i u^{(3)}(x_i,t) \tag{2.19}$$

ergibt, d.h. ein quasi quadratischer Fehler in den Diskretisierungsschrittweiten.

Definiert man zur Ersetzung des Differentialquotienten $\partial/\partial x$ einen finiten Differenzenoperator δ_x, einen Mittelungsoperator μ und einen gewichteten Mittelungsoperator $\bar{\mu}$ durch

$$\begin{aligned} \delta_x u_i &:= 2\frac{u_{i+1/2} - u_{i-1/2}}{\Delta x_i + \Delta x_{i-1}} \\ \mu u_{i+1/2} &:= \frac{1}{2}(u_{i+1} + u_i) \\ \bar{\mu} u_{i+1/2} &:= \frac{1}{2}\left(\frac{\Delta x_{i-1}}{\Delta x_i}u_{i+1} + \frac{\Delta x_i}{\Delta x_{i-1}}u_i\right) \end{aligned}$$

so gilt gerade

$$\begin{aligned} \Delta_x^0 u(x_i,t) &= \mu\delta_x u_i \\ \Delta_x u(x_i,t) &= \bar{\mu}\delta_x u_i \ . \end{aligned}$$

In dem hier entwickelten Verfahren wird, im Gegensatz zu SINCOVEC/MADSEN [72] und DEW/WALSH [24], die Diskretisierung (2.17) verwendet. Weniger weil der Approximationsfehler etwas günstiger ist, sondern da (2.17) mit der unten angegebenen Diskretisierung des Diffusionsterms in folgendem Sinne konsistent ist. Approximiert man $u(x,t)$ im Intervall $[x_{i-1}, x_{i+1}]$ durch eine Parabel, die duch die Wertepaare $(x_{i-1}, u_{i-1}), (x_i, u_i), (x_{i+1}, u_{i+1})$ definiert sei, dann ist deren 1. bzw. 2. Ableitung gerade durch (2.17) bzw. (2.21) gegeben.

Approximation der 2.Ableitung

Die Approximation der 2. Ableitung $u_{xx}(x,t)$ für $x \in \dot{\mathcal{G}}$ mit zentralen finiten Differenzen über 3 Knoten erfolgt demnach durch (siehe [61])

$$\frac{\partial^2}{(\partial x)^2}u(x_i,t) \to \Delta_{xx}u(x_i,t) \tag{2.20}$$

mit

$$\begin{aligned}\Delta_{xx}u(x_i,t) &:= \frac{2}{\Delta x_{i-1}+\Delta x_i}\left[\frac{u_{i+1}-u_i}{\Delta x_i}-\frac{u_i-u_{i-1}}{\Delta x_{i-1}}\right] \\ &= \frac{2}{\Delta x_{i-1}+\Delta x_i}\left[\frac{u_{i-1}}{\Delta x_{i-1}}-\frac{(\Delta x_{i-1}+\Delta x_i)u_i}{\Delta x_{i-1}\Delta x_i}+\frac{u_{i+1}}{\Delta x_i}\right] .\end{aligned} \tag{2.21}$$

Genau diese Approximation ergibt sich auch aus der Ersetzung

$$\frac{\partial^2}{(\partial x)^2}u(x_i,t) \to \delta_x\delta_x u(x_i,t) . \tag{2.22}$$

Der lokale Diskretisierungsfehler dieser Näherung ergibt sich zu

$$\begin{aligned}\tau^2(x_i,t) &:= \Delta_{xx}u(x_i,t)-\frac{\partial^2}{(\partial x)^2}u(x_i,t) \\ &= \sum_{k=3}^{K}\frac{2}{k!}\frac{(-1)^k\Delta x_{i-1}^{k-1}+\Delta x_i^{k-1}}{\Delta x_{i-1}+\Delta x_i}u^{(k)}(x_i,t) \;+\; O(\widehat{\Delta x}^{K-1})\end{aligned} \tag{2.23}$$

mit dem führenden Fehlerterm

$$\tau^2(x_i,t) \doteq \frac{1}{3}(\Delta x_i-\Delta x_{i-1})u^{(3)}(x_i,t) . \tag{2.24}$$

Im Gegensatz zum äquidistanten Fall besitzen also alle obigen Approximationen auf nichtuniformen Gittern keine quadratische Entwicklung des lokalen Fehlers. Das Paradoxon, daß trotz der nur linearen lokalen Diskretisierungsfehler, die Lösung einen quadratischen globalen Fehler in einem geeignet definiertem Δx_{max} besitzt, wird für den Fall linearer, skalarer 2–Punkt–Randwertprobleme in Manteuffel/White [54] ausführlich erörtert.

Eine Möglichkeit, wenigstens einen führenden quadratischen Fehlerterm zu zeigen, besteht darin, den Begriff eines nur schwach variierenden Gitters einzuführen (vgl. z.B. Chong [13], Hoffmann [41]), etwa durch die Forderung

$$\begin{aligned}\Delta x_i-\Delta x_{i-1} &= O(\overline{\Delta x}^2) \\ \overline{\Delta x} &:= \max_i\{\Delta x_i\} .\end{aligned}$$

Damit ist dann der führende Fehler in (2.18) und (2.23) quadratisch in $\overline{\Delta x}$. Eine damit verwandte Methode, siehe etwa Ablow/Schechter [1], Riva [69], um nur einige frühe Arbeiten zu nennen, besteht nun darin anzunehmen, daß eine glatte, streng monoton wachsende Gitterfunktion

$$\xi(r) \in C^\infty[0,1] \tag{2.25}$$

existiert derart, daß gilt:

$$\begin{aligned} x_i &= \xi(r_i)\ ,\ \ i = 1, ..., n_x \\ r_i &= (i-1)\cdot\Delta r\ ,\ \Delta r = 1/(n_x-1)\ ,\ \ i = 1, \ldots, n_x\ . \end{aligned} \tag{2.26}$$

Durch Taylorentwicklung erhält man

$$x(r_{i+1}) = x(r_i) + \sum_{l=1}^{L} \frac{1}{l!}\Delta r^l \xi^{(l)}(r_i, t)\ +\ O(\Delta r^{L+1}) \tag{2.27}$$

bzw.

$$x(r_{i-1}) = x(r_i) + \sum_{l=1}^{L} \frac{1}{l!}(-1)^l \Delta r^l \xi^{(l)}(r_i, t)\ +\ O(\Delta r^{L+1})\ . \tag{2.28}$$

Daraus folgt sofort (L sei als gerade angenommen, $L \geq K$)

$$\Delta x_i - \Delta x_{i-1} = \sum_{l=1}^{L/2} \frac{1}{(2l)!}\Delta r^{2l} \xi^{(2l)}(r_i, t)\ +\ O(\Delta r^{L+1})\ . \tag{2.29}$$

Damit sind die führenden Fehler (2.19, 2.24) quadratisch, allerdings in Δr und nicht in Δx – wie etwa in [69] gezeigt.

Für die hier entwickelte Schrittweitensteuerung und Fehlerschätzung wird sogar eine quadratische Entwicklung benötigt, zumindest muß auch der Fehler der Ordnung 3 in (2.18) und (2.23) eliminiert werden. Dazu wird die eben verwendete Technik auch auf die höheren Fehlerterme angewendet, was die hohe Glattheitsforderung an $\xi(r)$ bedingt. Einsetzen der Entwicklungen (2.27) und (2.28) in die Fehlerentwicklungen (2.18) bzw. (2.23) liefert, nach geeigneter Umformung, das Resultat:

$$\tau^1(x_i, t) = \sum_{k=1}^{K/2} \Delta r^{2k} \chi^1_{2k}(r_i, t)\ +\ O(\Delta r^{K+1})$$

bzw.

$$\tau^2(x_i, t) = \sum_{k=1}^{K/2} \Delta r^{2k} \chi^2_{2k}(r_i, t)\ +\ O(\Delta r^{K+1})\ ,$$

wobei die Koeffizienten χ^1_{2k} bzw. χ^2_{2k} von $u^{(j)}, j = 3, \ldots, 2(k+1)$ und $\xi^{(j)}$, $j = 1, \ldots, 2k$ abhängen. Zur Illustration seien χ^2_2 und χ^2_4 angegeben:

$$\begin{aligned} \chi^2_2(x_i, t) &= \frac{1}{12}(\xi^{(1)})^2 u^{(4)} + \frac{1}{3}\xi^{(2)} u^{(3)} \\ \chi^2_4(x_i, t) &= \frac{2}{6!}(\xi^{(1)})^4 u^{(6)} + \frac{2}{5!}2(\xi^{(1)})^2 \xi^{(2)} u^{(5)} + \\ &\quad \frac{2}{4!}(\frac{1}{3}\xi^{(1)}\xi^{(3)} + \frac{3}{4}(\xi^{(2)})^2) u^{(4)} + \frac{2}{3!}\frac{1}{12}(\xi^{(1)})^4 u^{(3)}\ . \end{aligned}$$

Randbedingung

Um das Modellproblem (2.11) nun vollständig in ein semi–diskretes System zu überführen, müssen noch die Randbedingungen (2.11.d) behandelt werden. Dazu wird, wie auch bei den später folgenden Untersuchungen der allgemeinen Randbedingungen (2.6.a), exemplarisch der linke Rand betrachtet. Für den rechten Rand gelten die entsprechenden Überlegungen.

Da sich keine Ortsableitungsterme in den Randbedingungen (2.11.d) befinden, stellt (2.11.d) eine approximationsfehlerfreie, nichtlineare Gleichung in u_1 dar:

$$R_L(t, u_1) := \gamma_L(t, u_1) - \alpha_L u_1 = 0 \; . \tag{2.30}$$

Das semi–diskrete System

Setzt man die Approximationen (2.17, 2.21) und die Randbedingung (2.30) (und ihr Analogon für den rechten Rand) in (2.11.b,d) ein, erhält man das zugehörige semi–diskrete System (über dem Gitter $\mathcal{G}$) :

$$\begin{aligned}
&a) && 0 &&=\;& \gamma_L(t, u_1) - \alpha_L u_1 \\
&i = 2, \ldots, n_x - 1 \\
&b) && u_t(x_i, t) &=\;& d(x_i, t)\Delta_{xx} u_i + c(x_i, t)\Delta_x u_i + g(x_i, t, u_i) \\
& && &=:\;& L_i^{\Delta}(t, u_{i-1}, u_i, u_{i+1}) \\
&c) && 0 &=\;& \gamma_R(t, u_{n_x}) - \alpha_R u_{n_x} \; .
\end{aligned} \tag{2.31}$$

Dabei besitzt das System eine sehr einfache Bandstruktur, da die i–te rechte Seite von (2.31) nur von u_{i-1}, u_i, u_{i+1} abhängt.

Der lokale Ortsdiskretisierungsfehler dieser semi–diskreten Gleichungen besitzt auch für ein nichtuniformes Gitter an jedem Knoten $x_i \in \dot{\mathcal{G}}$ eine quadratische Fehlerentwicklung

$$\begin{aligned}
\tau^L(x_i, t) \;&:=\; L_i^{\Delta}(t, u_{i-1}, u_i, u_{i+1}) - L(x, t, u, u_x, u_{xx})|_{x=x_i} \\
&=\; \sum_{k=1}^{K/2} \Delta r^{2k} (d(x_i, t)\chi_{2k}^2(r_i, t) \;+\; c(x_i, t)\chi_{2k}^1(r_i, t)) \\
&\quad + O(\Delta r^{K+1}) \; .
\end{aligned} \tag{2.32}$$

Rein formal gilt diese Entwicklung natürlich auch für $i = 1$ und $i = n_x$, allerdings mit verschwindenden Koeffizientenfunktionen und ohne Restglied.

Faßt man nun noch die Komponenten zusammen, etwa

$$\begin{aligned} \mathbf{u} &:= (u_1, \ldots, u_{n_x})^T \\ \mathbf{L}^\Delta &:= (R_L, L_2^\Delta, \ldots, L_{n_x-1}^\Delta, R_R)^T \\ \mathbf{u_t} &:= (u_t(x_1,t), \ldots, u_t(x_{n_x},t))^T \ , \end{aligned}$$

so läßt sich (2.31) auch in verkürzter Schreibweise angeben:

$$\mathbf{u}_t = \mathbf{L}^\Delta(t, \mathbf{u}) \ .$$

2.2.2 Der allgemeine Fall

Nachdem für das Modellproblem (2.11) eine quadratische Fehlerentwicklung für jeden Knoten eines nichtäquidistanten Gitters gezeigt wurde, soll nun untersucht werden, ob auch für das allgemeine Problem (2.1–2.6) eine derartige Form, etwa

$$\tau^\Delta(x_i,t) = \sum_{k=1}^{K/2} \Delta r^{2k} \Xi_k(r_i,t) \ + \ O(\Delta r^{K+1}) \ , i = 1, \ldots, n_x \ , \tag{2.33}$$

hergeleitet werden kann. Dazu soll schrittweise vorgegangen werden. Als erste Verallgemeinerung des Modellproblems (2.11) werden nun Randbedingungen der Form (2.6.a) – mit $\beta(x,t,u) \neq 0$ – betrachtet.

Randbedingungen vom allgemeinen Typ

Da der Spezialfall einer reinen Neumann–Randbedingung keine Sonderbehandlung erfährt, kann gleich der allgemeine Fall

$$R_L(t,u_1) := \gamma_L(t,u_1) - \alpha_L u_1 - \beta_L(t,u_1) u_x(x_1,t) = 0 \tag{2.34}$$

betrachtet werden. Dabei sind $\alpha_L, \beta_L, \gamma_L$ mit $\alpha(x_L), \beta(x_L,.,.), \gamma(x_L,.,.)$ zu identifizieren.
Eine sehr einfache Möglichkeit der Diskretisierung von u_x in (2.34) stellt die Approximation durch einen einseitigen Differenzenquotienten dar, etwa

$$\frac{\partial}{\partial x} u(x_1,t) \to \Delta_x^+ u(x_1,t) := \frac{u_2 - u_1}{\Delta x_1} \ . \tag{2.35}$$

Einsetzen in (2.34) liefert wieder eine algebraische Gleichung für u_1,

$$R_L^{\Delta,+}(t,u) := \gamma_L(t,u_1) - \alpha_L - \beta_L(t,u_1) \frac{u_2 - u_1}{\Delta x_1} = 0 \ , \tag{2.36}$$

die nun, im Gegensatz zu (2.30), vom Zustand des Systems im Inneren des Gebietes abhängt. Per Konstruktion besitzt diese Gleichung einen linearen Approximationsfehler. Durch einseitige Approximation höherer Ordung kann zwar ein zumindest quadratischer Diskretisierungsfehler erzwungen werden, jedoch zerstört ein derartiges Vorgehen die einfache Band–Struktur des semi–diskreten Systems.

Alternativ zu (2.34) wird daher meist wie folgt vorgegangen. Man setzt die Gültigkeit der PDG auch auf dem Rand $\delta\Omega$ voraus und diskretisiert die PDG auch an $x_1 = x_L$ mit (2.17) und (2.21), wobei ein virtueller Knoten $x_0 := x_L - \Delta x_1$ verwendet wird. Die Randbedingung (2.36) wird nun, ebenfalls auf dem Randknoten x_1, zentral durch

$$R_L^\Delta(t,u) := \gamma_L(t,u_1) - \alpha_L u_1 - \beta_L(t,u_1)\frac{u_2-u_0}{\Delta x_1} = 0 \tag{2.37}$$

approximiert. Der lokale Diskretisierungsfehler dieser algebraischen Gleichung

$$\begin{aligned} \tau^R(x_1,t) &:= R_L^\Delta(t,u) - R_L(t,u_1) \\ &= \sum_{k=1}^{K/2} \frac{-\beta_L(t,u_1)}{(2k+1)!}\Delta x_1^{2k} u^{(2k+1)}(x_1,t) + O(\Delta x_1^{K+1}) \end{aligned} \tag{2.38}$$

zeigt also eine quadratische Entwicklung in Δx_1. Um die dazu nötige Taylorentwicklung durchführen zu können, muß allerdings die Existenz einer exakten Lösung $u(x,t)$ mit entsprechender Differenzierbarkeitsordnung auch für $x \in [x_L - \Delta x_1, x_L]$ vorausgesetzt werden. Im Prinzip könnte die Gleichung (2.37) in Verbindung mit den Gleichungen (2.31.b), diese jetzt jedoch auch für $i = 1$ und $i = n_x$, zur Ortsdiskretisierung des Gesamtsystems verwendet werden. Allerdings ist dann wieder die Bandstruktur des Systems (2.31) gestört und die Annahme der Gültigkeit der PDG auch über den Definitionsbereich hinaus wird explizit verwendet. Daher wird (2.37) üblicherweise nach u_0 aufgelöst und in die Ortsdiskretisierung (2.31.b) (für $x_i = x_L$) eingesetzt. Damit ergibt sich die semi–diskrete Gleichung (differentieller Art) an $x_1 = x_L$ zu

$$\begin{aligned} u_t(x_1,t) &= d(x_1,t)\frac{1}{\Delta x_1}\left[\frac{u_2-u_1}{\Delta x_1} - \frac{u_1-u_0}{\Delta x_1}\right] \\ &\quad + c(x_1,t)\frac{u_2-u_0}{2\Delta x_1} + g(x_1,t,u_1) \\ &= d(x_1,t)\frac{2}{\Delta x_1}\left[\frac{u_2-u_1}{\Delta x_1} - \frac{\gamma_L(t,u_1)-\alpha_L u_1}{\beta_L(t,u_1)}\right] \\ &\quad + c(x_1,t)\frac{\gamma_L(t,u_1)-\alpha_L u_1}{\beta_L(t,u_1)} + g(x_1,t,u_1)\ . \end{aligned} \tag{2.39}$$

Wegen der damit wieder eingeführten Asymmetrie liegt jedoch i.a. wiederum nur eine Gleichung mit einem formal linearen, lokalen Fehler in Δx_1 vor. Für den häufigen Fall, daß es sich bei der Randbedingung um eine homogene Neumann–Bedingung handelt, und dies als Bedingung für Spiegelsymmetrie der Lösung $u(x,t)$ an $x = x_L$ angesehen werden kann, d.h.

$$u^{(2k+1)}(x_L, t) = 0 \ , \ k = 0, 1, \ldots, K/2 \ ,$$

besitzt (2.39) jedoch eine lokale Fehlerentwicklung, die quadratisch in Δx_1 (bzw. Δr) ist, da dann die störenden Terme der Taylorentwicklung von $u(x_2, t)$ um x_1 verschwinden.

Für den allgemeinen Fall läßt sich im numerischen Experiment allerdings beobachten, daß die lineare, lokale Randstörung einen diffusiven Charakter der PDG vorausgesetzt, rasch wegtransportiert wird und somit auch am Rand ein praktisch quadratisches Fehlerverhalten bzgl. der Ortsdiskretisierung zu beobachten ist. Für die einfache einseitige Approximation (2.35) gilt dies jedoch nicht.

Diffusionsterm

Als nächster Schritt der Verallgemeinerung des Modellproblems (2.11) wird das Auftreten eines nicht ausdifferenzierten, orts– und lösungsabhängigen Diffusionsterms in kartesischen ($c = 0$), zylindrischen ($c = 1$) oder sphärischen Koordinaten ($c = 2$) betrachtet. Zu seiner Approximation kann analog zur Ersetzung (2.22) vorgegangen werden, d.h.

$$\begin{aligned}
&\mathcal{D}(x_i, t, u_i) = \\
&x^{-c}\frac{\partial}{\partial x}(x^c D(x,t,u)\frac{\partial u}{\partial x})\Big|_{x=x_i} \rightarrow \\
&x^{-c}\delta_x(x^c D(x,t,u)\frac{\partial u}{\partial x})\Big|_{x=x_i} = \\
&x_i^{-c}\frac{2}{\Delta x_i + \Delta x_{i-1}}(x^c_{i+1/2}D_{i+1/2}\frac{\partial}{\partial x}u_{i+1/2} - x^c_{i-1/2}D_{i-1/2}\frac{\partial}{\partial x}u_{i-1/2}) \rightarrow \\
&x_i^{-c}\frac{2}{\Delta x_i + \Delta x_{i-1}}(x^c_{i+1/2}D_{i+1/2}\delta_x u_{i+1/2} - x^c_{i-1/2}D_{i-1/2}\delta_x u_{i-1/2}) = \\
&x_i^{-c}\frac{2}{\Delta x_i + \Delta x_{i-1}}(x^c_{i+1/2}D_{i+1/2}\frac{u_{i+1}-u_i}{\Delta x_i} - x^c_{i-1/2}D_{i-1/2}\frac{u_i - u_{i-1}}{\Delta x_{i-1}}) \\
&=: \mathcal{D}^{\Delta}(x_i, t, u_i)
\end{aligned} \tag{2.40}$$

Dabei kann $D_{i\pm1/2}$ entweder durch

$$D_{i\pm1/2} = D(x_{i\pm1/2}, t, u_{i\pm1/2}) := D(x_{i\pm1/2}, t, \frac{u_{i\pm1} + u_i}{2}) \tag{2.41}$$

oder

$$D_{i\pm1/2} := \frac{D_{i\pm1} + D_i}{2} \tag{2.42}$$

approximiert werden. Ohne dies zunächst zu spezifizieren, ist zu bemerken, daß die in (2.40) verwendete Vorgehensweise nicht die einzig natürliche ist. Im Gegensatz zu (2.22) kann hier die Reihenfolge der Ersetzung $\partial/\partial x \to \delta_x$ nicht ohne weiteres vertauscht werden. Wird nämlich zuerst $\partial u/\partial x$ durch $\delta_x u$ ersetzt, ergibt sich anschließend der Term ($c = 0$ zur Vereinfachung)

$$\delta_x \left(D(x_i, t, u_i) \frac{2(u_{i+1/2} - u_{i-1/2})}{\Delta x_i + \Delta x_{i-1}} \right) ,$$

und es muß geklärt werden, wie das Ergebnis einer Anwendung des finiten Differenzenoperators δ_x auf ein Produkt von Funktionen auszusehen hat. Setzt man z.B.

$$\delta_x(D_i v_{i+1/2}) := \frac{D_{i+1/2}(v_{i+1} - v_i)}{\Delta x_i} ,$$

so ergibt sich gerade das obige Resultat (2.40). Definiert man dagegen eine diskrete Produktregel

$$\delta_x(D_i v_{i+1/2}) := (\delta_x D_i) v_{i+1} + D_i(\delta_x v_{i+1/2}) , \tag{2.43}$$

so erhält man

$$\begin{aligned} &\frac{\partial}{\partial x}(D(x,t,u)\frac{\partial}{\partial x}u)\bigg|_{x=x_i} \to \\ &\frac{2(D_{i+1/2} - D_{i-1/2})}{\Delta x_i + \Delta x_{i-1}} \frac{2(u_{i+1/2} - u_{i-1/2})}{\Delta x_i + \Delta x_{i-1}} + \frac{2D_i}{\Delta x_i + \Delta x_{i-1}} \left(\frac{u_{i+1} - u_i}{\Delta x_i} + \frac{u_i - u_{i-1}}{\Delta x_{i-1}} \right) . \end{aligned} \tag{2.44}$$

Diese Approximation ist auch nicht identisch mit einer Diskretisierung des ausdifferenzierten Diffusionsterms mittels der Approximationen (2.21) und (2.17), wenn letztere auch zur Diskretisierung von D_x verwendet wird, d.h.

$$(D_x u_x + D u_{xx})|_{x=x_i} \to (\Delta_x D_i)(\Delta_x u_i) + D_i \Delta_{xx} u_i . \tag{2.45}$$

Welche von diesen (und weiteren möglichen) Approximationen für ein spezielles Problem die "beste" ist, kann a priori nicht entschieden werden. Gegen (2.45), und somit auch gegen (2.44), spricht jedoch, daß zunächst eine "aufrauhende" Differentiation durchgeführt wird, bevor dann die Approximation erfolgt.

Die Diskretisierung (2.40) ist für den Fall eines kartesischen Koordinatensystems identisch mit der in [24, 72] verwendeten:

$$\begin{aligned} &x^{-c}\frac{\partial}{\partial x}(x^c D(x,t,u)\frac{\partial}{\partial x}u)\bigg|_{x=x_i} \to \\ &\frac{c+1}{x_{i+1/2}^{c+1} - x_{i-1/2}^{c+1}} \left(x_{i+1/2}^c D_{i+1/2} \frac{u_{i+1} - u_i}{\Delta x_i} x_{i-1/2}^c D_{i-1/2} - \frac{u_i - u_{i-1}}{\Delta x_{i-1}} \right) . \end{aligned} \tag{2.46}$$

Diese Diskretisierung, die als spezielle Finite–Volumen–Diskretisierung interpretiert werden kann, stimmt für $c = 1$ auf nichtuniformen Gitter, bzw. für $c = 2$ schon auf äquidistantem, nicht mehr vollständig mit (2.40) überein.

Ein wesentlicher Unterschied zeigt sich allerdings nur, wenn der Punkt $x = 0$ in Ω enthalten ist. Für (2.40) überträgt sich die Singularität des Diffusionsterms auch auf die Diskretisierung, während (2.46) auch für $x_i = 0$ auswertbar bleibt. Es sei denn, man betrachtet den etwas pathologischen Fall, daß $x_i = 0$ im Innern des Gebietes Ω liegt (also $x_{i-1/2} = -x_{i+1/2}, c = 1$). Der integrale Charakter von (2.46) und die nicht notwendige Sonderbehandlung des Falls $x = 0$ sprechen für die Verwendung von (2.46). Da jedoch typischerweise für $x = 0$ eine homogene Neumann–Randbedingung vorliegt, kann es günstiger sein, in dieser Situation ($x = 0, u_x(0,t) = 0$) die analytisch gewonnenen Grenzwerte

$$\lim_{x\to 0}\left(x^{-c}\frac{\partial}{\partial x}(x^c D(x,t,u)\frac{\partial}{\partial x}u)\right) = (c+1)D(0,t,u)u_{xx} \tag{2.47}$$

zu approximieren und in (2.40) für $x = 0$ einzusetzen. In der aktuellen Version der entwickelten Software wird so vorgegangen.

Dabei wird, ebenso wie in [72], (2.41) zur Approximation eines lösungsabhängigen Diffusionskoeffizienten $D_{i\pm 1/2}$ verwendet. Der wesentliche Grund dafür ist darin zu sehen, daß die Behandlung von unstetigen Koeffizienten einfacher ist, solange die Gitterwahl so erfolgt, daß die Sprungstelle auf einen Gitterpunkt fällt. In [24] wird dagegen (2.42) der Vorzug gegeben, da sich dadurch die Benutzerschnittstelle des Codes etwas vereinfacht.

Für die Approximation (2.40) kann auch für nichtuniformes Gitter eine quadratische Entwicklung des lokalen Diskretisierungsfehlers gezeigt werden, etwa der Form

$$\begin{aligned} \tau^{\mathcal{P}}(x_i,t) &:= \mathcal{D}^{\Delta}(x_i,u_i) - \mathcal{D}(x_i,u_i) \\ &= \sum_{k=1}^{K/2} \Delta r^{2k}\psi_{2k}(r_i,t) \;+\; O(\Delta r^{K+1})\,. \end{aligned} \tag{2.48}$$

Auf eine genaue Herleitung und Darstellung soll hier verzichtet werden. Neben den Entwicklungen (2.27, 2.28) benötigt man nun auch Taylorentwicklungen von $D_{i\pm 1/2}$ um $x = x_i$. Damit hängen die Koeffizientenfunktionen ψ_{2k} auch von höheren Ableitungen des Diffusionskoeffizienten $D(x,t,u)$ (nach x und/oder u) ab. Entsprechende Bedingungen an die Differenzierbarkeitsordnungen und Beschränktheit der Restglieder seien also vorausgesetzt.

Damit erübrigt sich eigentlich die obige Bemerkung über die Behandlung von

unstetigen Diffusionskoeffizienten, die häufig in der Form

$$D(x) = \begin{cases} D_L & , \quad x < \bar{x} \\ D_R & , \quad x > \bar{x} \end{cases}$$

auftreten. Allerdings verkraften die hier entwickelten adaptiven Kontroll– und Steuermechanismen in vielen Fällen das Auftreten von kleinen, lokalen Störungen, so etwa auch derartige Diffusionskoeffizienten.

Nichtlinearität von f und B

Gehen Ortsableitungsterme nichtlinear in die rechte Seite der PDG ein, so wird die Funktion f aus (2.4) an jedem inneren Knoten $x_i \in \dot{\mathcal{G}}$ durch die Diskretisierung

$$f \to f_i^\Delta := f(x_i, t, u_i, \Delta_x u_i, \mathcal{D}^\Delta(x_i, t, u_i)) \tag{2.49}$$

ersetzt. Nimmt man nun auch die Funktion f als genügend oft differenzierbar bzgl. aller Argumente $x, u, u_x, \mathcal{D}$ an, so kann durch Taylorentwicklung um die exakte Lösung u der lokale Diskretisierungsfehler wieder auf die gewünschte Form gebracht werden. Mit (2.18) und (2.48) erhält man (ohne Spezifikation des speziellen Knotens):

$$\begin{aligned} f^\Delta &= f(x, t, u, u_x + \tau^1, \mathcal{D} + \tau^D) \\ &= f(x, t, u, u_x, \mathcal{D}) \\ &+ f_{u_x}(x, t, u, u_x, \mathcal{D})\tau^1 + f_{\boldsymbol{\mathcal{D}}}(x, t, u, u_x, \mathcal{D})\tau^{\boldsymbol{\mathcal{D}}} \\ &+ f_{u_x, \boldsymbol{\mathcal{D}}}(x, t, u, u_x, \mathcal{D})\tau^1\tau^{\boldsymbol{\mathcal{D}}} \\ &+ \ldots \ . \end{aligned} \tag{2.50}$$

Nach dem Zusammenfassen der Terme mit gleichen Δr–Potenzen besitzt der durch die Ersetzung (2.49) hervorgerufene Ortsdiskretisierungsfehler

$$\tau^f(x_i, t) := f_i^\Delta - f(x, t, u, u_x, \mathcal{D}(x, t, u))|_{x=x_i} \tag{2.51}$$

wieder die Form (2.33).

Für die Ortsdiskretisierung ist ein lösungsabhängiger Term $B(x, t, u, u_x)$ in (2.4) nur dann von Relevanz, wenn auch tatsächlich eine Abhängigkeit von u_x vorliegt. Man ersetzt:

$$B \to B_i^\Delta := B(x_i, t, u_i, \Delta_x u_i) \ , \tag{2.52}$$

und wie schon für nichtlineares f, muß gegebenenfalls entwickelt werden und man erhält

$$\begin{aligned} B^{\Delta} &= B(x,t,u,u_x+\tau^1) \\ &= B(x,t,u,u_x,) \\ &+ B_{u_x}(x,t,u,u_x)\tau^1 + \tfrac{1}{2}B_{u_x,u_x}(x,t,u,u_x)(\tau^1)^2 \\ &+ \ldots \ . \end{aligned} \tag{2.53}$$

Analog zum Vorgehen bei f erhält man dann auch für den Ortsdiskretisierungsfehler von B,

$$\tau^B(x_i,t) := B_i^{\Delta} - B(x,t,u,u_x)|_{x=x_i} \ , \tag{2.54}$$

wieder die Form (2.33).

Systeme

a) Differentialgleichungen

Ist das betrachtete Problem ein gekoppeltes System von n_{pde} Gleichungen, in n_{pde} Unbekannten, also

$$u = (u_1, u_2, ..., u_{n_{pde}})^T \ , \tag{2.55}$$

so ergibt sich zunächst ein gewisses Notationsproblem. Die bereits in (2.9) eingeführte Notation, unter $u_j(x,t)$ die j–te Komponente des Vektors u zu verstehen, widerspricht der oben verwendeten vereinfachenden Schreibweise, mit u_i die (exakte) Lösung am i–ten Knoten des Gitters $\mathcal{G}$ zu bezeichnen. Da jedoch beide Notationen sehr gebräuchlich sind, soll auch hier keine unnatürliche Notation (etwa $u_j^i, u_i^j, u_i^{[j]}$) verwendet werden. In der Regel wird die Bedeutung eines einzelnen, unteren Index aus dem Zusammenhang sofort ersichtlich sein. Wird zur abkürzenden Schreibweise für die j–te Komponente am i–ten Knoten die doppelt indizierte Größe $u_{j,i}$ verwendet, so bezeichnet in dieser Arbeit grundsätzlich der erste Index die Nummer der Komponente und der zweite Index die des Gitterknotens. Darüber hinaus wird, wenn irgend möglich, der Buchstabe j zur Indizierung der Komponente und der Buchstabe i zur Indizierung (bzw. als Laufindex) der Knoten verwendet. Entsprechend sind die Differenzenoperatoren Δ_x bzw. Δ_{xx} im Systemfall wie folgt zu verstehen ($x_i \in \dot{\mathcal{G}}$):

$$\begin{aligned} \Delta_x u_i = \Delta_x u(x_i,t) &= (\Delta_x u_1(x_i,t), \Delta_x u_2(x_i,t), \ldots, \Delta_x u_{n_{pde}}(x_i,t))^T \\ &= (\Delta_x u_{1,i}, \Delta_x u_{2,i}, \ldots, \Delta_x u_{n_{pde},i})^T \ , \end{aligned} \tag{2.56}$$

bzw.

$$\Delta_{xx} u_i = (\Delta_{xx} u_{1,i}, \Delta_{xx} u_{2,i}, \ldots, \Delta_{xx} u_{n_{pde},i})^T \ . \tag{2.57}$$

Für das Element (j,k) des diskretisierten Diffusionsoperators $\mathcal{D}^\Delta$ an inneren Punkten erhält man, vgl. (2.10) zur Definition des kontinuierlichen Operators,

$$\begin{aligned} \mathcal{D}^\Delta_{j,k,i} := \frac{2x_i^{-c}}{\Delta x_i + \Delta x_{i-1}} \quad & \left(x^c_{i+1/2} D_{j,k,i+1/2} \frac{u_{k,i+1} - u_{k,i}}{\Delta x_i} \right. \\ & \left. - x^c_{i-1/2} D_{j,k,i-1/2} \frac{u_{k,i} - u_{k,i-1}}{\Delta x_{i-1}} \right) \end{aligned} \tag{2.58}$$

mit

$$D_{j,k,i\pm1/2} := D_{j,k} \left(x_{i\pm1/2}, t, \frac{1}{2}(u(x_{i\pm1}) + u(x_i)) \right) \ .$$

Da man auch im Systemfall gemäß (2.50, 2.53) vorgehen kann, ergibt sich nun an jedem Knoten $x_i \in \dot{\mathcal{G}}$ ein Vektor

$$\tau^f(x_i, t) \in \mathbb{R}^{n_{pde}}$$

bzw. eine Matrix

$$\tau^B(x_i, t) \in \mathbb{R}^{n_{pde} \times n_{pde}}$$

von lokalen Diskretisierungsfehlern, wobei für jede Komponente bzw. jedes Matrixelement eine quadratische Entwicklung gilt. Für den Fall, daß in die j–te Differentialgleichung auch Ortsableitungsterme der k–ten Komponente eingehen, werden die Koeffizientenfunktionen der Fehlerentwicklungen $\tau^f_{j,i}, \tau^B_{j,k,i}$ auch Anteile enthalten, die von Ableitungen der Komponente $u_k(x_i, t)$ abhängen. Die Art der Kopplung des Gesamtsystems überträgt sich also bereits auf die Kopplung der Koeffizientenfunktionen des lokalen Ortsdiskretisierungsfehlers.

b) Randbedingungen

Die Entscheidung, welche Art von Randbedingung vorliegt, ist jetzt komponentenweise zu treffen. Abhängig von $\beta_{L,j}$ ergibt sich für die j–te Komponente von u entweder eine Gleichung der Form (2.30) oder eine Verallgemeinerung von (2.39). Allerdings bereitet der Fall $\beta_{L,j} \neq 0$ jetzt eine offensichtliche Schwierigkeit. Um (2.39) herzuleiten, wurde zwar die Lösung u formal über den Rand hinaus fortgesetzt, aber die letzlich verwendete Gleichung setzt nur die Gültigkeit der PDG auf dem Rand $\delta\Omega$ voraus. Liegt nun aber ein allgemeiner Differentialoperator $\mathcal{D}(x, t, u)$ vor, würde die direkte Verallgemeinerung von (2.39) die Auswertung von

$$D_{j,k,-1/2} = D_{j,k} \left(x_1 - \Delta x_1/2), t, u((x_1 - \Delta x_1/2) \right)$$

in (2.58) erfordern. Um dies zu vermeiden, wird statt dessen

$$D_{j,k,-1/2} := D_{j,k,1} = D_{j,k}(x_1, t, u(x_1))$$

ausgewertet. Damit, und mit der Randdiskretisierung der 1. Ableitung (wenn $\beta_{L,j} \neq 0$),

$$\frac{\partial u_j(x_1,t)}{\partial x} \to \Delta_x^L u_j := \frac{\gamma_{L,j} - \alpha_{L,j} u_{j,1}}{\beta_{L,j}} \tag{2.59}$$

erhält man die Randapproximation

$$\begin{aligned} \mathcal{D}_{j,k}^L := \frac{x_1^{-c}}{\Delta x_1} \quad & \left(x_{1+1/2}^c D_{j,k,1+1/2} \frac{u_{k,2} - u_{k,1}}{\Delta x_1} \right. \\ & \left. - x_1^c D_{k,j,1} \frac{\gamma_{L,k}(t,u_{k,1}) - \alpha_{L,k} u_{k,1}}{\beta_{L,k}(t,u_{k,1})} \right) . \end{aligned} \tag{2.60}$$

Der Fall $c > 1, x_1 = 0, u_x(x_1) = 0$ erfährt weiterhin eine Sonderbehandlung, $c > 1, x_1 = 0, u_x(x_1) \neq 0$ wird ausgeschlossen.

Eine zweite Schwierigkeit entsteht, wenn $\beta_{L,k} \neq 0$, $\beta_{L,j} = 0$ gilt, und die k–te Komponente der Funktion f, bzw. die k–te Zeile von B, am (linken) Rand von $\partial u_j / \partial x$ abhängt. In einer derartigen Situation wird die k–te PDG am Rand ausgewertet, aber es kann (2.59) nicht zur Bestimmung von $\partial u_j(x_1,t)/\partial x$ verwendet werden. Stattdessen wird, wie auch in [72], die einseitige Approximation (2.35) zur Diskretisierung von $\partial u_j(x_1,t)/\partial x$ in f_k bzw. B verwendet. Wie oben ausgeführt, ist damit die Konsistenzordnung auf 1 reduziert. Damit wird, wie bei Sprüngen im Diffusionskoeffizienten, letztendlich der in Kapitel 3 entwickelte Kontroll– und Steueralgorithmus zumindest lokal gestört. Die Problemformulierung (2.4, 2.6) erlaubt formal den Fall, daß f_k auch von $\mathcal{D}_{k,j}$ ($\beta_{L,k} \neq 0, \beta_{L,j} = 0, j \neq k$) abhängt. Da sich aber mit nur zwei Werten $(u_{j,1}, u_{j,2})$ keine konsistente Diskretisierung für $\mathcal{D}_{k,j}$ erreichen läßt, soll dieser Fall ausgeschlossen sein.

Am rechten Rand gilt entsprechendes und man erhält die komponenten– bzw. elementweise definierten Differenzenoperatoren

$$\Delta_x^R u_j := \begin{cases} \dfrac{\gamma_{R,j} - \alpha_{R,j} u_{n_x,j}}{\beta_{R,j}} & , \text{ wenn } \beta_{R,j} \neq 0 \\ \dfrac{u_{j,n_x} - u_{j,n_x-1}}{\Delta_{n_x-1}} & , \text{ wenn } \beta_{R,j} = 0 \end{cases} \tag{2.61}$$

und

$$\mathcal{D}_{j,k}^R := \frac{x_{n_x}^{-c}}{\Delta x_{n_x-1}} \left(x_{n_x}^c D_{j,k,n_x} \Delta_x^R - x_{n_x-1/2}^c D_{k,j,n_x-1/2} \frac{u_{k,n_x} - u_{k,n_x-1}}{\Delta x_{n_x-1}} \right) . \tag{2.62}$$

2.2.3 Zusammenfassung

Die semi–diskreten Gleichungen

Die mit Hilfe der oben definierten Approximationen durchgeführte Semi–Diskretisierung soll nun zusammengefaßt dargestellt werden. Dazu wird der

Gesamtvektor $\mathbf{u}(t)$ der Unbekannten des semi–diskreten Systems

$$\mathbf{u}(t) := (u(x_1,t),...,u(x_{nx},t))^T \in \mathbb{R}^{n_x \cdot n_{npde}}$$

und dessen Zeitableitung

$$\mathbf{u}_t(t) := (u_t(x_1,t),...,u_t(x_{nx},t))^T \in \mathbb{R}^{n_x \cdot n_{npde}}$$

eingeführt. Die Anordnung der Unbekannten $\{\mathbf{u}_{j,i}(t)\}_{j=1,...,n_{pde}}^{i=1,...,n_x}$ erfolgt also derart, daß alle Komponenten des Vektors $u(x,t)$ am ersten Knoten x_1, gefolgt von allen Komponenten des Vektors $u(x,t)$ am zweiten Knoten x_2, usw., aufgeschrieben werden. Die Semi–Diskretisierung der PDG (2.1–2.6) ist dann durch das folgende System von differentiell–algebraischen Gleichungen (DAG) gegeben:

$$\mathbf{B}^{\Delta}(t,\mathbf{u}(t)) \cdot \mathbf{u}_t(t) = \mathbf{f}^{\Delta}(t,\mathbf{u}) \ , \quad t \in [t_0, t_{end}] \tag{2.63}$$

mit t_0, t_{end} aus (2.1) und den Anfangswerten nach (2.5):

$$\mathbf{u}(t_0) = (u_0^T(x_1), \ldots, u_0^T(x_{n_x}))^T \in \mathbb{R}^{n_x \cdot n_{npde}} \ . \tag{2.64}$$

Dabei ist $\mathbf{B}^{\Delta}$ eine Blockdiagonalmatrix der Dimension $(n_x \cdot n_{pde}) \times (n_x \cdot n_{pde})$:

$$\mathbf{B}^{\Delta}(t,\mathbf{u}) = \text{diag}\,(B_1^{\Delta}(t,\mathbf{u}), B_2^{\Delta}(t,\mathbf{u}), \ldots, B_{n_x}^{\Delta}(t,\mathbf{u})) \tag{2.65}$$

mit

$$\begin{aligned}
a) \quad B_i^{\Delta} &= B(x_i,t,u_i,\Delta_x u_i) \\
&\quad i = 2,\ldots,n_x - 1 \\
b) \quad B_1^{\Delta} &= (B_1^{\Delta})_{j,k} = \begin{cases} 0 & , \text{ wenn } \beta_{L,j} = 0 \\ (B(x_1,t,u_1,\Delta_x^L u_1))_{j,k} & , \text{ wenn } \beta_{L,j} \neq 0 \end{cases} \\
&\quad j = 1,\ldots,n_{pde} \ ; \ \ k = 1,\ldots,n_{pde} \\
c) \quad B_{n_x}^{\Delta} &= (B_{n_x}^{\Delta})_{j,k} = \begin{cases} 0 & , \text{ wenn } \beta_{R,j} = 0 \\ (B(x_{n_x},t,u_{n_x},\Delta_x^R u_{n_x}))_{j,k} & , \text{ wenn } \beta_{R,j} \neq 0 \end{cases} \\
&\quad j = 1,\ldots,n_{pde} \ ; \ \ k = 1,\ldots,n_{pde} \ .
\end{aligned} \tag{2.66}$$

Die dabei verwendeten Differenzenoperatoren Δ_x, Δ_x^L und Δ_x^R sind durch (2.17), (2.59) bzw. (2.61) gegeben.
Die rechte Seite $\mathbf{f}^{\Delta}(t,\mathbf{u})$ der Gleichung (2.63) ist ebenfalls blockweise gegeben durch:

$$\mathbf{f}^{\Delta}(t,\mathbf{u}) = (f_1^{\Delta}(t,\mathbf{u}), f_2^{\Delta}(t,\mathbf{u}), \ldots, f_{n_x}^{\Delta}(t,\mathbf{u}))^T \in \mathbb{R}^{n_x \cdot n_{npde}} \tag{2.67}$$

mit

$$
\begin{aligned}
a) \quad f_i^{\Delta} &= f(x_i, t, u_i, \Delta_x u_i, \mathcal{D}^{\Delta}(x_i, t, u_i)) \\
&\quad i = 2, \ldots, n_x - 1 \\
b) \quad (f_1^{\Delta})_j &= \begin{cases} \gamma_L(t, u_1) - \alpha_{L,j} \cdot u_{j,1} & , \text{ wenn } \beta_{L,j} = 0 \\ (f_j(x_1, t, u_1, \Delta_x^L u_1, \mathcal{D}^L(x_1, t, u_1))) & , \text{ wenn } \beta_{L,j} \neq 0 \end{cases} \\
&\quad j = 1, \ldots, n_{pde} \\
c) \quad (f_{n_x}^{\Delta})_j &= \begin{cases} \gamma_R(t, u_{n_x}) - \alpha_{R,j} \cdot u_{j,n_x} & , \text{ wenn } \beta_{R,j} = 0 \\ (f_j(x_{n_x}, t, u_{n_x}, \Delta_x^R u_{n_x}, \mathcal{D}^R(x_{n_x}, u_{n_x}))) & , \text{ wenn } \beta_{R,j} \neq 0 \end{cases} \\
&\quad j = 1, \ldots, n_{pde} \, .
\end{aligned}
\tag{2.68}
$$

Die diskreten Diffusionsoperatoren $\mathcal{D}^{\Delta}, \mathcal{D}^L$ und $\mathcal{D}^R$ sind dabei durch (2.58), (2.60) bzw. (2.62) gegeben.

Der lokale Ortsdiskretisierungsfehler

Bezeichne $\mathbf{u}(t)$ den Vektor der Restriktionen der exakten Lösung $u(x,t)$ der PDG auf das durch (2.12) definierte Gitter $\mathcal{G}$ mit den Eigenschaften (2.25, 2.26). Es seien, analog zu $\mathbf{B}^{\Delta}, \mathbf{f}^{\Delta}$ die Blockdiagonalmatrix $\mathbf{B}$ und der Blockvektor $\mathbf{f}$ definiert, also

$$
\begin{aligned}
\mathbf{B} &= \operatorname{diag}(B_1, B_2, \ldots, B_{n_x}) \\
\mathbf{f} &= (f_1, f_2, \ldots, f_{n_x})^T .
\end{aligned}
$$

Dabei sind die Komponenten B_i, f_i entsprechend (2.66) und (2.68) definiert, d.h. es sind die dabei verwendeten diskreten Operatoren durch die entsprechenden kontinuierlichen zu ersetzen. Faßt man nun noch die Diskretisierungsfehler τ^f, τ^B zum entspechenden Vektor $\tau^{\mathbf{f}}$ bzw. der Matrix $\tau^{\mathbf{B}}$ zusammen, dann ist der lokale (Orts–)Diskretisierungsfehler der Semi–Diskretisierung (2.63) durch den $(n_x \cdot n_{pde})$–Vektor

$$
\begin{aligned}
\kappa^{\mathbf{\Delta R}}(t) &:= \mathbf{f}^{\Delta}(t, \mathbf{u}(t)) - \mathbf{B}^{\Delta}(t, \mathbf{u}(t)) \cdot \mathbf{u}_t(t) \\
&= \mathbf{f} + \tau^{\mathbf{f}} - (\mathbf{B} + \tau^{\mathbf{B}}) \cdot \mathbf{u}_t \\
&= \tau^{\mathbf{f}} - \tau^{\mathbf{B}} \cdot \mathbf{u}_t
\end{aligned}
\tag{2.69}
$$

gegeben. Der durch (2.69) definierte Fehler gibt also den Defekt der (restringierten) exakten Lösung $u(x,t)$ bzgl. der Gleichung (2.63) und dem Gitter $\mathcal{G}$ an. Der Vektor $\kappa^{\mathbf{\Delta R}}(t)$ setze sich dabei aus n_x Vektoren $\kappa^{\Delta R}(x_i, t) \in \mathbb{R}^{n_{pde}}$, $x_i \in \mathcal{G}$, zusammen.

Die oben gemachten Überlegungen, inwieweit sich die quadratische Fehlerentwicklung (2.32) für das Modellproblem (2.11) auch auf die allgemeine Problemklasse (2.1–2.6) überträgt, haben gezeigt, daß zumindest für die inneren Knoten $x_i \in \dot{\mathcal{G}}$ eine quadratische Entwicklung existiert, wenn nur genügend starke Forderungen an die Taylorentwickelbarkeit aller in (2.4) auftretenden Funktionen gestellt werden. Insbesondere läßt sich der lokale Ortsdiskretisierungsfehler auch örtlich lokal, d.h. getrennt für jeden Knoten $x_i \in \mathcal{G}$, angeben:

$$\kappa_i^{\mathbf{\Delta R}}(t) = \begin{cases} \tau^L := f_1^\Delta - B_1^\Delta \cdot u_{t,i} & , i = 1 \\ \tau_i^f - \tau_i^B \cdot u_{t,i} & , i = 2, \ldots, n_x - 1 \\ \tau^R := f_{n_x}^\Delta - B_{n_x}^\Delta \cdot u_{t,i} & , i = n_x \ . \end{cases} \tag{2.70}$$

Der Defekt am Rand kann dabei sowohl verschwinden ($\beta = 0$), eine nur lineare, eine gestörte quadratische, aber auch eine ungestörte quadratische Fehlerentwicklung in Δr besitzen. Nach den obigen Ausführungen ist jedoch klar, daß er, zumindest für innere Knoten, die gewünschte Form hat:

$$\kappa^{\mathbf{\Delta R}}(x_i, t) = \sum_{k=1}^{K/2} \Delta r^{2k} \Xi_k(x_i, t) \ + \ O(\Delta r^{K+1}) \ , i = 2, ..., n_x - 1 \ . \tag{2.71}$$

Die (für $n_{pde} > 1$ vektorwertigen) Koeffizientenfunktionen Ξ_k können dabei in sehr komplizierter Weise von allen in (2.4) auftretenden Funktionen, deren (gemischten, höheren) partiellen Ableitungen nach allen Argumenten $(x, t, u, u_x, \mathcal{D})$ und der Gitterfunktion $\xi(r)$ bzw. deren Ableitungen abhängen. Falls B von u_x abhängig ist, so treten nicht nur Ortsableitungen von u, sondern auch die Zeitableitung u_t in den Funktionen Ξ_k auf. Insbesondere können dabei in der j–ten Komponente von Ξ_k auch alle anderen Komponenten der Funktionen aus (2.4) und deren Ableitungen auftreten.

Im weiteren wird sogar davon ausgegangen, daß eine quadratische Entwicklung der Form (2.71) formal auch für die Randknoten gilt. Die hier eventuell auftretenden linearen Fehleranteile werden als den Gesamtalgorithmus nicht wesentlich beeinflussende Störterme angesehen.

Für die Anwendung der im weiteren vorgestellten Steuerungsmechanismen ist die genaue Form der Entwicklungskoeffizienten unerheblich. Deshalb können die entwickelten Algorithmen auch für andere Diskretisierungsvarianten erfolgreich eingesetzt werden. Etwa (2.16) statt (2.17), (2.46) statt (2.40) oder direkte Diskretisierung von Termen der Form $h(u)_x$ mit (2.16) oder (2.17). Wichtig ist jedoch, daß die Entwicklungen weiterhin örtlich lokal gelten. Ebenso kann an die Verwendung von geeigneten upwind-Schemata gedacht werden, falls die Gleichung einen starken hyperbolischen Anteil hat. Falls dazu eine Ortsdiskretisierung gewählt wird, die keine quadratische Entwicklung

besitzt (sondern nur eine lineare), können die unten dargestellten adaptiven Konzepte prinzipiell weiterverwendet werden. Allerdings erfordert dies evtl. eine sorgfältige Anpassung der einzelnen algorithmischen Bausteine. Gleiches gilt für Fehlerentwicklungen in anderen Potenzen.

2.3 Zeitdiskretisierung mit semi–implizitem Euler

Die zeitliche Integration des semi–diskreten Problems (2.63) wird mit Hilfe einer (modifizierten) semi–impliziten Euler–Methode (oft auch als linear–implizite Euler–Methode bezeichnet) durchgeführt. Wie schon bei der Darstellung der Ortsdiskretisierung, sollen ihre Herleitung und Eigenschaften an Hand von vereinfachten Modellproblemen diskutiert werden.

2.3.1 Gewöhnliche Differentialgleichungen

Klassische Eulerdiskretisierungen

Es wird zunächst ein System von gewöhnlichen Differentialgleichungen (in autonomer Form) betrachtet

$$y' = f(y) \ , \ y(t_0) = y_0 \ , \ f \in C^{L+1}[t_0, t_{end}] \ . \tag{2.72}$$

Ein klassisches Lösungsverfahren ist die explizite Euler–Diskretisierung, in der die Zeitableitung durch einen einfachen Vorwärtsdifferenzenquotienten approximiert wird. Sei die Lösung zu einem Zeitpunkt, etwa $t = t_k$, gegeben, dann wird, mit der in Abschnitt 2.2.1 eingeführten Notation (vgl. (2.35)), die Ersetzung

$$y'(t_k) \to \Delta_t^+ y(t_k) := \frac{y(t_{k+1}) - y(t_k)}{\Delta t} \tag{2.73}$$

durchgeführt. Durch Einsetzen der exakten Lösung $y(t)$ und Taylorentwicklung ergibt sich der Approximationsfehler der Diskretisierung (2.73) zu

$$\begin{aligned} \tau_k^+ := \tau^+(t_k) \ &:= \ \Delta_t^+ y(t_k) - y'(t_k) \\ &= \ \textstyle\sum_{l=2}^{L} \frac{\Delta t^{l-1}}{l!} y^{(l)}(t_k) \ + \ O(\Delta t^L) \ . \end{aligned} \tag{2.74}$$

Einsetzen von (2.73) in (2.72) und Auswerten der rechten Seite der DG an t_k liefert die explizite Euler–Diskretisierung von (2.72):

$$\frac{y_{k+1} - y_k}{\Delta t} = f(y_k) \ , \ t_{k+1} := t_k + \Delta t \ . \tag{2.75}$$

Schreibt man (2.75) in der üblichen Notation für Einschrittverfahren (ESV) zur Lösung von gewöhnlichen DG-Systemen,

$$\begin{aligned} y_{k+1} \ &= \ y_k + \Delta t \Phi(y_k, y_{k+1}; \Delta t) \\ &\qquad \Phi \ : \ \text{Inkrementfunktion} \ , \end{aligned} \tag{2.76}$$

und definiert den Defekt ("Konsistenzfehler", Residuumfehler) eines ESV durch

$$\kappa_k := y(t_{k+1}) - y(t_k) - \Delta t \Phi(y(t_k), y(t_{k+1}); \Delta t) \; , \tag{2.77}$$

so ergibt sich derjenige der expliziten Euler–Diskretisierung von (2.72) zu

$$\kappa_k^{EE} = \Delta t \cdot \tau_k^+ = \frac{1}{2} y''(t_k) \cdot \Delta t^2 + O(\Delta t^3) \; . \tag{2.78}$$

Das Verfahren hat also die Konsistenzordnung $q = 1$. Der lokale Diskretisierungsfehler der Methode, gemäß der üblichen Definition, vgl. etwa STREHMEL/WEINER [75],

Differenz der numerischen Approximation y_{k+1} und der exakten Lösung $y(t_{k+1})$ nach einem Schritt, gestartet auf exakter Lösung $y(t_k)$

ist also für das explizite Euler–Verfahren ebenfalls durch (2.78) gegeben:

$$\begin{aligned} \tau_{k+1}^{EE} &:= y(t_{k+1}) - y_{k+1} \\ &= y(t_{k+1}) - y(t_k) - \Delta t f(y(t_k)) = \kappa_k^{EE} \; . \end{aligned} \tag{2.79}$$

Eine andere klassische Diskretisierungmethode für (2.72) ist das implizite Euler–Verfahren:

$$y_{k+1} = y_k + \Delta t f(y_{k+1}) \; . \tag{2.80}$$

Bezüglich der Zeitschicht t_{k+1} wird die Zeitableitung also durch eine Rückwärtsdifferenz ($\Delta_t^- y(t_{k+1})$) approximiert. Der Defekt dieses Verfahrens ist am einfachsten bzgl. der Zeitschicht t_{k+1} anzugeben:

$$\kappa_{k+1}^{IE} = \Delta t \cdot \tau_{k+1}^- = -\tfrac{1}{2} y''(t_{k+1}) \cdot \Delta t^2 + \textstyle\sum_{l=3}^{L} \frac{(-1)^{l-1} \Delta t^l}{l!} y^{(l)}(t_{k+1}) \; + \; O(\Delta t^{L+1}) \; . \tag{2.81}$$

Bezüglich des Defektes ist (2.80) also nicht "besser" als (2.75). Der lokale Diskretisierungsfehler, gemäß obiger Definition, ergibt sich als Lösung der nichtlinearen Gleichung:

$$\begin{aligned} \tau_{k+1}^{IE} &:= y(t_{k+1}) - y(t_k) - \Delta t f(y_{k+1}) \\ &= y(t_{k+1}) - y(t_k) - \Delta t f(y(t_{k+1}) - \tau_{k+1}^{IE}) \; . \end{aligned} \tag{2.82}$$

Durch Taylorentwicklung und Vernachlässigung des Restgliedes erhält man in erster Näherung:

$$\tau_{k+1}^{IE} \doteq (I - \Delta t f_y(y(t_{k+1}))^{-1} \kappa_{k+1}^{IE} \; . \tag{2.83}$$

Steife Differentialgleichungen

Der entscheidende Unterschied von (2.80) zu (2.75) ist die implizite Struktur der Diskretisierung, die allerdings in jedem Zeitschritt $t_k \to t_{k+1}$ die Lösung

eines nichtlinearen Gleichungssystems erfordert. Diesem Nachteil steht der Vorteil gegenüber, eine geeignete Diskretisierungsform für sogenannte "*steife*" Differentialgleichungen zu sein. Eine sehr pragmatische und gleichzeitig auch die älteste Definition dieses Begriffs stammt von CURTISS/HIRSCHFELDER [14] und lautet:

stiff equations are equations where certain implicit methods, in particular BDF, perform better, usually tremendously better, than explicit methods.

Sie ist interessanterweise auch in dem Lehrbuch von HAIRER/WANNER [38] dem Kapitel über steife Differentialgleichungen vorangestellt. Auf eine exakte Definition wird in [38] jedoch verzichtet.

In DEUFLHARD [20] findet sich der Satz:

Differentialgleichungen, zu deren Beschreibung $f_y(\cdot)$ wichtig ist, heißen "steife" Differentialgleichungen.

Damit dürfte der Kern der Sache noch etwas besser getroffen sein. Diese "Definition" des Begriffs *Steifheit* wurde durch die von DEUFLHARD in [19] entwickelte Charakterisierung steifer Systeme und semi–impliziter Verfahren motiviert.

Neben den klassischen Diskretisierungstypen "explizit" und "implizit" gewinnt daher die Klasse der "linear–impliziten" Verfahren zunehmend an Bedeutung. In semi–impliziten (linear–impliziten) Diskretisierungen versucht man zwar weiterhin die Information der Jacobimatrix zu verwenden, aber es sollen nur lineare statt nichtlineare Gleichungssysteme in der Diskretisierung auftreten. Eine ausführliche Darstellung von linear–impliziten Runge–Kutta–Methoden findet sich in z.B. in [75].

Semi–implizite Euler–Diskretisierung

Das hier verwendete Extrapolationsverfahren ist, im Prinzip, von diesem Typ, da es sich auch als Runge-Kutta-Verfahren (variabler Ordnung!) interpretieren läßt. Dabei wird als Basisdiskretisierung die sog. semi–implizite Euler–Diskretisierung verwendet. Angewendet auf (2.72) lautet sie:

$$(I - \Delta t A)(y_{k+1} - y_k) = \Delta t f(y_k) \tag{2.84}$$

mit der nicht notwendigerweise exakten und an einem evtl. zeitlich weiter zurückliegenden Punkt $t_l \leq t_k$ ausgewerteten Jacobimatrx

$$A :\approx f_y(y_l) \; . \tag{2.85}$$

Der Defekt der Methode ergibt sich zu:

$$\begin{aligned} \kappa_k^{SI} &:= \kappa_k^{EE} - \Delta t^2 A(\tau_k^+ + y'(t_k)) \\ &= (I - \Delta t A)\kappa_k^{EE} - \Delta t^2 A y'(t_k) \; . \end{aligned} \tag{2.86}$$

Als eigenständige Diskretisierungsmethode wurde sie erstmals von DEUFLHARD [18] vorgeschlagen. Davor wurde sie als Startschritt in der semi–impliziten Mittelpunktsregel von BADER/DEUFLHARD [6] verwendet. Die Diskretisierung (2.84) liegt nun gewissermaßen zwischen (2.75) und (2.80), allerdings recht "nahe" bei (2.80), was folgende Interpretation der Methode belegt.

Verwendet man zur Lösung des durch die implizite Euler–Diskretisierung aufgestellten nichtlinearen Gleichungssystems ein Newtonverfahren, so erhält man die Iteration

$$\begin{aligned} & i = 0, 1, 2, ... \\ & A :\approx f_y(y^i_{k+1}) \\ & (I - \Delta t A)(y^{i+1}_{k+1} - y^i_{k+1}) = -y^i_{k+1} + y_k + \Delta t f(y^i_{k+1}) \ . \end{aligned} \tag{2.87}$$

Der Typ des Newtonverfahrens wird durch die genaue Spezifikation der Matrix A festgelegt:

$$\begin{array}{lllll} \text{a)} & A & := & f_y(y^i_{k+1}) & : \text{ gewöhnliches Newtonverfahren} \\ \text{b)} & A & := & f_y(y^0_{k+1}) & : \text{ vereinfachtes Newtonverfahren} \\ \text{c)} & A & := & f_y(y^0_l) & : \text{ Newton–ähnliches Verfahren .} \end{array} \tag{2.88}$$

Wählt man in (2.87) den natürlichen Startwert $y^0_{k+1} = y_k$ und die Matrix A nach (2.88.c), so ist der erste Schritt der obigen Newtoniteration gerade mit der semi–impliziten Euler–Diskretisierung (2.84) identisch, wenn (2.85) exakt gilt. Ist dagegen in (2.85) bzw. (2.84) gerade $l = k$, so entspricht dies dem ersten Schritt eines vereinfachten Newtonverfahrens. Für lineare Probleme (2.72) sind implizite und linear–implizite Diskretisierung natürlich identisch.

Auf der anderen Seite reduziert sich die semi–implizite Euler–Diskretisierung zur expliziten Diskretisierung (2.75), wenn man $A \equiv 0$ setzt. Wenn man so will, ist dies der Extremfall einer anderen Interpretation. Wendet man eine Euler–Diskretisierung auf ein modifizierte System

$$y' - Ay = f(y) - Ay \ \ , \ A \text{ regulär} \tag{2.89}$$

an, so erhält man

$$\frac{y_{k+1} - y_k}{\Delta t} - A y_{k+1} = f(y_k) - A y_k \ . \tag{2.90}$$

Für $A :\approx f_y(y_l)$ ergibt sich gerade (2.84). Die Matrix A könnte, im Prinzip, auch eine beliebige (reguläre) Matrix sein. Die Diskretisierung bleibt wegen (2.86) weiterhin konsistent mit der DG, solange $\Delta t A \to 0$, für $\Delta t \to 0$, gilt. Die Tatsache, daß die semi–implizite Euler–Methode Manipulationen an A zuläßt, wird z.B. in NOWAK [57] erfolgreich ausgenutzt.

Der zuzätzliche Aufwand, um einen Integrationsschritt $t_k \to t_{k+1}$ mit (2.84) statt mit (2.75) durchzuführen, ist das Lösen eines linearen Gleichungssystems. Üblicherweise wird dies mit Gaußelimination (LU–Zerlegung, gefolgt von einer Vorwärts/Rückwärtssubstitution) durchgeführt. In darauf folgenden Schritten ändert sich für konstantes Δt und festes A die in (2.84) zu zerlegende Matrix nicht. Es müssen nur Vorwärts/Rückwärtssubstitutionen mit den neuen rechten Seiten $f_{k+1}, f_{k+2}, ...$ durchgeführt werden. Wird die Basis–Diskretisierung (2.84) in Verbindung mit Extrapolation verwendet, ist dies eine wichtige Eigenschaft für die Effizienz des Gesamtverfahrens. Würde man dagegen die implizite Euler–Methode als Basisdiskretisierung wählen, so müßte nicht nur pro Integrationsschritt ein nichtlineares Gleichungssystem anstelle eines linearen gelöst werden, sondern gerade diese wichtige Eigenschaft ginge überdies verloren.

2.3.2 Differentiell–algebraische Systeme

Diskretisierungsvarianten

Die eben vorgestellten Interpretationen zeigen, daß es sich bei der semi–impliziten Euler–Diskretisierung eigentlich um eine Schar von Diskretisierungen handelt. Dies zeigt sich erneut, wenn man sie auf linear–implizite Systeme von differentiell–algebraischen Gleichungen (DAG) anwendet, d.h. auf Systeme der Form

$$B(y)y' = f(y);\ ,\ y(t_0) = y_0\ , \tag{2.91}$$

mit lösungsabhängiger, evtl. singulärer Matrix B. Es wird angenommen, daß das Problem vom "index ≤ 1" ist und konsistente Startdaten y_0 vorliegen.

Nimmt man zur Herleitung einer Diskretisierung dieser Gleichung mit Hilfe der semi–impliziten Euler–Diskretisierung zunächst B als regulär an, und wendet (2.84) direkt auf das transformierte System

$$y' = B(y)^{-1} f(y) \tag{2.92}$$

an, so erhält man, siehe Deuflhard/Nowak [23], als eine mögliche Form der Diskretisierung

$$(B_l - \Delta t \hat{A})(y_{k+1} - y_k) = \Delta t B_l B_k^{-1} f_k \tag{2.93}$$

mit der an $t_l \leq t_k$ ausgewerteten Jacobimatrix des Residuums $r := f - By'$:

$$\hat{A} :\approx f_y(y_l) - B_y(y_l) \cdot y_l'\ . \tag{2.94}$$

Für konstantes B ergibt sich die bereits in [18] vorgeschlagene Diskretisierung. Für den allgemeinen Fall zeigt ein Vergleich mit (2.84), daß sich der

Rechenaufwand zur Ausführung eines Schrittes mehr als verdoppelt hat, da in jedem Schritt ein zusätzliches Gleichungssystem (mit der von k abhängiger Matrix B_k) gelöst werden muß. Schreibt man (2.93) in einer anderen Form,

$$(B_k - \Delta t B_k B_l^{-1} \hat{A})(y_{k+1} - y_k) = \Delta t f_k$$

und ersetzt nun

$$B_k B_l^{-1} \to I \tag{2.95}$$

so kann dies als Störung der Jacobimatrix $\hat{A}$ interpretiert werden. Die entstandene Diskretisierung

$$(B_k - \Delta t \hat{A})(y_{k+1} - y_k) = \Delta t f_k \tag{2.96}$$

ist nun auch wieder für singuläres B formal anwendbar. Die Ersetzung (2.95) läßt sich auch als Ersetzung $B_l \to B_k$ an geeigneter Stelle der Jacobimatrix des transformierten Systems (2.92) ansehen. Dies bedeutet, daß die Diskretisierung (2.96), sogar "näher" an einer impliziten Euler–Diskretisierung von (2.91) ist als (2.93).

Im Vergleich zu (2.84) hat die Diskretisierung (2.96) allerdings immer noch einen wesentlichen Nachteil. Da die Matrix des linearen Gleichungssystems nicht nur von Δt und $y(t_l)$ (über (2.94)), sondern auch von $y(t_k)$ abhängt, muß nun bei einer direkten Lösung in jedem Schritt, $k \to k+1$, eine LU–Zerlegung durchgeführt werden. In (2.84) mußte dagegen nur bei einer Neuberechnung der Jacobimatrix oder Wechsel der Schrittweite zerlegt werden. Als Alternative bietet sich daher eine iterative Gleichungslösung (für $k \neq l$) an, siehe [18]. In [23] wurde eine adaptive Realisierung entwickelt, die, abhängig vom Konvergenzverhalten der Iteration und dem (geschätzten) Aufwand einer Zerlegung bzw. Vorwärts/Rückwärtssubstitution, zwischen iterativer und direkter Lösung umschaltet. Die Iterationsvorschrift wurde in Lubich [48] noch etwas verbessert, so daß die iterative Lösung von (2.96) für $k > l$ lautet:

$$\begin{aligned} y_{k+1}^0 &= y_k + (y_k - y_{k-1}) \\ &i = 0, 1, 2, \ldots \\ &(B_l - \Delta t \hat{A})(y_{k+1}^{i+1} - y_k) = \Delta t f_k - (B_k - B_l)(y_{k+1}^i - y_k) \quad . \end{aligned} \tag{2.97}$$

Durch eine alternative Herleitung [48] läßt sich noch eine andere Variante der semi–impliziten Euler–Diskretisierung für Systeme der Form (2.91) begründen. Sie kann letztendlich als die iterative Version von (2.96) mit nur einem Iterationsschritt in (2.97) interpretiert werden und zeigt sowohl in der Theorie als auch in der Praxis deutlich schlechtere Eigenschaften.

Liegt statt (2.91) ein nicht–autonomes Problem vor, etwa

$$B(t, y) y' = f(t, y) \ , \ y(t_0) = y_0 \ , \tag{2.98}$$

so stehen wiederum mehrere Diskretisierungsvarianten zur Verfügung. Die einfachste Diskretisierungsmöglichkeit ist durch

$$(B(t_k, y_k) - \Delta t\hat{A})(y_{k+1} - y_k) = \Delta t f(t_k, y_k) \tag{2.99}$$

gegeben. Dies entspricht einer expliziten Euler–Diskretisierung des Problems bzgl. der t–Abhängigkeit. Für rein differentielle Probleme, die ein nicht–steifes Verhalten bzgl. t zeigen, ist (2.99) daher eine durchaus adäquate Diskretisierung. Allerdings kann dieses Verfahren nicht mehr als ein Schritt eines speziellen Newton–ähnlichen Verfahrens zur Lösung des durch eine implizite Euler–Diskretisierung von (2.98) entstandenen Gleichungssystems angesehen werden. Um eine derartige Interpretation zuzulassen, müßte man z.B. die Diskretisierung (implizit in der t–Abhängigkeit)

$$(B(t_{k+1}, y_k) - \Delta t\hat{A})(y_{k+1} - y_k) = \Delta t f(t_{k+1}, y_k) \tag{2.100}$$

wählen. In Verbindung mit Extrapolation erweist sich diese Form jedoch als ungünstig für die Implementierung. Im Vergleich zu (2.99) sind zusätzliche Auswertungen von f und B erforderlich.

Als Alternative bietet sich daher die Diskretisierung

$$(B(t_k, y_k) - \Delta t\hat{A})(y_{k+1} - y_k) = \Delta t f(t_k, y_k) + \Delta t^2(f_t(t_l, y_l) - B_t(t_l, y_l)y') \tag{2.101}$$

an. Diese Form ergibt sich durch Transformation des Systems (2.98) auf autonome Form, Anwendung der Diskretisierung (2.96) und anschließender Rücktransformation.

In dem hier vorgestellten Verfahren zur Lösung von partiellen Differentialgleichungen der Form (2.1)–(2.6) wird zur Zeitdiskretisierung des zugehörigen semi–diskreten Systems (2.63) die Variante (2.101) mit selbstadaptivem Umschalten zwischen der direkten bzw. iterativen Gleichungslösung verwendet.

Für die hier angestellten Betrachtungen ist es jedoch ausreichend, den Defekt der Basis–Diskretisierung (2.96), angewendet auf das System (2.91), anzugeben:

$$\begin{aligned} \kappa_k^{LI} &:= (B_k - \Delta t\hat{A})(y_{k+1} - y_k) - \Delta t f_k \\ &= \Delta t B_k \tau_k^+ - \Delta t^2 \hat{A}(y'_k + \tau_k^+) \ . \end{aligned} \tag{2.102}$$

Es liegt also eine Entwicklung in Potenzen von Δt vor.

Extrapolation

Aus der Basis–Diskretisierung erhält man ein effizientes Gesamt–Verfahren durch Anwendung des Extrapolationsprinzips. Dies soll nun in aller Kürze

dargestellt werden. Eine ausführliche Darstellung des dann zweidimensionalen Extrapolationsverfahrens in Ort und Zeit findet sich in Abschnitt 3.1.1.

Prinzip eines Extrapolationsverfahrens ist die wiederholte Integration über eine sog. Grundschrittweite, etwa

$$t_l \to t_l + \Delta T =: \bar{t}\,,$$

mit sukzessiv kleiner werdenden sog. internen Schrittweiten

$$\Delta t_j := \frac{\Delta T}{n_j}\,.$$

Die Folge $\mathcal{F} := \{n_j\}_{j=1,2,\ldots}$ wird dabei als Schrittweitenfolge bezeichnet und die geeignete Wahl der speziellen Werte hängt im wesentlichen von der Struktur der verwendeten Basisdiskretisierung ab. Für die semi–implizite Euler–Diskretisierung wurden sehr gute Erfahrungen mit der harmonischen Folge

$$\mathcal{F}_H := \{1, 2, 3, \ldots\} \tag{2.103}$$

gemacht. Durch die wiederholte Integration von t_0 nach $\bar{t}$ liegen verschiedene numerische Approximationen $Y(\bar{t}; \Delta t_j)$ an die exakte Lösung $y(\bar{t})$ vor. Mit Hilfe von Extrapolation sollen daraus Approximationen höherer Ordnung (d.h. genauere) konstruiert werden. Dies ist nur unter der Voraussetzung möglich, daß der globale Diskretisierungsfehler der Basisdiskretisierung

$$\varepsilon(\bar{t}) := Y(\bar{t}; \Delta t_j) - y(\bar{t}) \tag{2.104}$$

eine asymptotische Entwicklung der Form

$$\varepsilon(\bar{t}) = \sum_{k=1}^{N} e_k(\bar{t}) \Delta t_j^{qk} + O(\Delta t_j^{N+1}) \tag{2.105}$$

besitzt. Das Restglied sei dabei gleichmäßig beschränkt und die Potenz q gibt die Konsistenzordnung der Basisdiskretisierung an, also hier $q = 1$. Wird nun Polynomextrapolation verwendet, d.h. wird ein interpolierendes Polynom $P(\Delta t)$ mit sukzessiv wachsendem Grad über den Stützwerten

$$\{\Delta t_j, Y(\bar{t}_j; \Delta t_j)\}_{j=1,2,\ldots}$$

aufgebaut und an der Stelle $\Delta t = 0$ ausgewertet, so entspricht dies gerade einer Elimination der führenden Fehler der Entwicklung (2.105). Üblicherweise werden die nichtextrapolierten Approximationen mit $T_{j,1} := Y(\bar{t}; \Delta t_j)$ bezeichnet und die durch Extrapolation gewonnenen Approximationen mit $T_{j,k}, k \leq j$. Pro Extrapolationsstufe, $k \to k+1$, wird jeweils ein Term der Fehlerentwicklung eliminiert, so daß in erster Näherung gilt:

$$T_{j,k} - y(\bar{t}) \doteq (-1)^{k-1} \gamma_{j,k} e_k(\bar{t}) \Delta t_j^k \;\;, k \leq j\,.$$

Der Wert der Konstante $\gamma_{j,k}$ hängt dabei von der Schrittweitenfolge und der Konsistenzordnung q ab

$$\gamma_{j,k} = (n_{j-k+1} \cdot \ldots \cdot n_j)^{-q} \; .$$

Ein besonderer Reiz von Extrapolationsverfahren ist sicherlich, daß die Anzahl der Extrapolationsstufen nicht festgelegt ist. Man hat also ein Verfahren mit variabler Ordnung. Um ein robustes und effizientes Gesamtverfahren zu erhalten, benötigt man eine Ordnungs– und Schrittweitenkontrolle, die den Fehler (2.104) schätzt, kontolliert und geeignete Grundschrittweiten ΔT und eine dazu passende Ordnung (d.h. die Anzahl von Extrapolationsstufen) automatisch wählt. Ein für eine ganze Kette von Basisdiskretisierungen verwendbarer Kontroll– und Steuerungsalgorithmus wurde in DEUFLHARD [17] entwickelt und hat sich bei der Integration mit nicht–steifen, steifen und differentiell–algebraischen Integratoren als äußerst effizient und robust erwiesen. Auf eine genaue Darstellung kann an dieser Stelle verzichtet werden, da die wesentlichen Aspekte in Abschnitt 3.3.1 angesprochen werden.

Der globale Diskretisierungsfehler

Die Voraussetzung für die Anwendung von Extrapolationstechniken in Verbindung mit der semi–impliziten Euler–Diskretisierung ist die Existenz einer asymptotischen Entwicklung der Form (2.105). Der Existenzbeweis stellt dabei bereits für gewöhnliche Differentialgleichungen ein schwieriges analytisches Problem dar. Die z.Z. wohl eleganteste Beweistechnik geht auf HAIRER/LUBICH [35] zurück und hat eine einfache rekursive Struktur. Mit ihrer Hilfe wurden die verschiedenen Varianten der semi–impliziten Euler–Diskretisierung bei Anwendung auf diverse Typen von Gleichungen untersucht, siehe etwa [22, 36, 38, 48]. Dabei hat sich gezeigt, daß für singulär gestörte Probleme und differentiell–algebraische Gleichungen, insbesondere in der allgemeinen Formulierung (2.98), nur noch die Existenz von gestörten Entwicklungen gezeigt werden kann. Interessanterweise besitzt die hier verwendete Variante (2.101) die "besten" theoretischen Eigenschaften, obwohl bei ihrer algorithmisch orientierten Entwicklung in [23] diese Resultate noch nicht bekannt waren. Für diese Diskretisierung, angewendet auf Systeme der Form (2.98), gilt unter gewissen Voraussetzungen, z.B. an die Matrix B, die folgende gestörte Entwicklung [48]:

$$\varepsilon(\bar{t}) = e_1(\bar{t})\Delta t_j + (e_2(\bar{t}) + \epsilon_{2,j})\Delta t_j^2 + \ldots + (e_N(\bar{t}) + \epsilon_{N,j})\Delta t_j^N + O(\Delta t_j^{N+1}) \; . \tag{2.106}$$

Dabei sind die Koeffizientenfunktionen $e_k(t)$ Lösung von linearen differentiell–algebraischen Systemen der Form

$$B(y)e_k' = (f_y(y) - B_y(y)y')e_k(t) - d_k(t) \; .$$

Die Inhomogenitäten $d_k(t)$ stellen den Defekt von im Beweis rekursiv definierten Approximationsschemata höherer Ordnung dar ($d_1(t)$ ist der führende Fehlerterm von κ^{LI}). Je nachdem, ob die differentiellen und algebraischen Komponenten des Systems separiert sind oder nicht, verschwindet eine gewisse Anzahl der ersten Störterme in der Entwicklung, so daß das Gesamtverfahren dann eine Maximalordnung $p = 4$ oder $p = 5$ besitzt. Dennoch können auch höhere Extrapolationsstufen erfolgreich verwendet werden, da die Genauigkeit der Approximationen trotz formal gleicher Ordnung zunimmt. Dies liegt daran, daß die Koeffizienten vor den Störtermen für wachsendes k deutlich kleiner werden.

Abschließend ist noch zu bemerken, daß die angegebenen Ordungsschranken nur für die modifizierte Schrittweitenfolge $\mathcal{F} := \{2, 3, 4, \ldots\}$ gezeigt werden können. Dennoch wird hier (wie bereits im DAG–Löser LIMEX [23]) die Schrittweitenfolge (2.103) verwendet, was einen formalen Ordnungsverlust von einer weiteren Potenz bedeutet, da dann fast alle Störterme in der Entwicklung (2.106) nicht–verschwindend sind.

2.3.3 Basisdiskretisierung des Gesamtverfahrens

Wendet man nun die Zeitdiskretisierung (2.101) auf das semi–diskrete System (2.63) an, so erhält man die vollständig diskretisierte Form der PDG. Zur Vereinfachung der Notation wird hier der Term $\Delta t^2 r_t$ aus (2.101) weggelassen.

$$\left(\mathbf{B}^{\Delta}(t_k, \mathbf{u}_k) - \Delta t \mathbf{A}^{\Delta}\right)(\mathbf{u}_{k+1} - \mathbf{u}_k) = \Delta t \mathbf{f}^{\Delta}(t_k, \mathbf{u}_k) \tag{2.107}$$

mit

$$\mathbf{A}^{\Delta} :\approx \left.\frac{\partial \mathbf{r}^{\Delta}(t, \mathbf{u})}{\partial \mathbf{u}}\right|_{t=t_l, \mathbf{u}=\mathbf{u}_l} , \; t_l \leq t_k \; , \; \mathbf{r}^{\Delta} = \mathbf{f}^{\Delta} - \mathbf{B}^{\Delta}\mathbf{u}_t \; . \tag{2.108}$$

Diese Basisdiskretisierung soll nun in Verbindung mit Extrapolationstechniken in Zeit und Raum verwendet werden, um ein effizientes, adaptives Gesamtverfahren zu erhalten. Bevor darauf eingegangen wird, ist zunächst der Diskretisierungsfehler von (2.107) allein zu betrachten.

Der Defekt der Basisdiskretisierung

Einsetzen der exakten Lösung $u(x, t_{k+1}), u(x, t_k)$ bzw. $u(x, t_l)$, jeweils restringiert auf das verwendete Gitter, also $\mathbf{u}(t_{k+1}), \mathbf{u}(t_k), \mathbf{u}(t_l)$, liefert den Defekt der Gesamtmethode. Eine mögliche Definition ist etwa

$$\kappa^{\Delta}(t_k) := \Delta t \mathbf{f}^{\Delta}(t_k, \mathbf{u}(t_k)) - \left(\mathbf{B}^{\Delta}(t_k, \mathbf{u}(t_k)) - \Delta t \mathbf{A}^{\Delta}\right)(\mathbf{u}(t_{k+1}) - \mathbf{u}(t_k)) \tag{2.109}$$

mit

$$\mathbf{A}^{\Delta} :\approx \left.\frac{\partial \mathbf{r}^{\Delta}(t,\mathbf{u})}{\partial \mathbf{u}}\right|_{t=t_l,\mathbf{u}=\mathbf{u}(t_l)} . \tag{2.110}$$

Die erste Frage ist nun, ob die Gesamtdiskretisierung eine asymptotische Entwicklung des Defekts in Potenzen von Δr^2 und Δt besitzt. Dazu ist zunächst der Diskretisierungsfehler eines Matrix–Vektor Produkts $\mathbf{w}^{\Delta}(t_l,t_k) := \mathbf{A}^{\Delta}\mathbf{v}(t_k)$ zu betrachten. Mit der Definition (2.110) und der Gleichung (2.69) ergibt sich.

$$\mathbf{w}^{\Delta} = \frac{\partial \mathbf{r}^{\Delta}}{\partial \mathbf{u}}\mathbf{v} = \frac{\partial(\mathbf{f}-\mathbf{B}\mathbf{u}_t)}{\partial \mathbf{u}}\mathbf{v} + \frac{\partial \kappa^{\mathbf{\Delta R}}}{\partial \mathbf{u}}\mathbf{v} .$$

Also gilt ($\mathbf{w} := \mathbf{A}\mathbf{v}$):

$$\mathbf{w}^{\mathbf{\Delta}} - \mathbf{w} = \frac{\partial \kappa^{\mathbf{\Delta R}}}{\partial \mathbf{u}}\mathbf{v} =: \kappa^{\mathbf{w}}(t_j,t_k;\mathbf{u},\mathbf{v}) . \tag{2.111}$$

Die quadratische Form der Entwicklung $\kappa^{\mathbf{\Delta R}}$ überträgt sich auf die Entwicklung $\kappa^{\mathbf{w}}$, wenn die Approximation von $\mathbf{A}^{\Delta}$ bzw. $\mathbf{A}$ in (2.108) bzw. (2.110) nicht zu grob ist.

Da auch für die Vektoren $\mathbf{u}(t_{k+1}), \mathbf{u}(t_k)$ die Entwicklung (2.74) gilt, d.h.

$$\mathbf{u}(t_{k+1}) - \mathbf{u}(t_k) = \Delta t(\mathbf{u}_t(t_k) + \tau^{+}(t_k;\mathbf{u}(t_k))) ,$$

ergibt sich aus (2.69) und (2.102) für (2.109) (Argumente wie oben angegeben):

$$\begin{aligned}
\kappa^{\Delta} &= \Delta t(\mathbf{f}^{\Delta} - \mathbf{B}^{\Delta}\mathbf{u}_t) - \Delta t\mathbf{B}^{\Delta}\tau^{+} + \Delta t^2\mathbf{A}^{\Delta}(\mathbf{u}_t + \tau^{+}) \\
&= \Delta t\kappa^{\mathbf{\Delta R}} - \Delta t(\mathbf{B} + \tau^{\mathbf{B}})\tau^{+} + \Delta t^2\mathbf{A}^{\Delta}(\mathbf{u}_t + \tau^{+}) \\
&= \Delta t\kappa^{\mathbf{\Delta R}} - \Delta t\mathbf{B}\tau^{+} + \Delta t^2\mathbf{A}(\mathbf{u}_t + \tau^{+}) + \Delta t\tau^{\mathbf{B}}\tau^{+} + \Delta t^2\tau^{\mathbf{w}} \\
&= \Delta t\kappa^{\mathbf{\Delta R}} + \kappa^{\mathbf{\Delta T}} + \Delta t\tau^{\mathbf{B}}\Delta t^2\tau^{\mathbf{w}} .
\end{aligned} \tag{2.112}$$

Dabei bezeichnet $\kappa^{\mathbf{\Delta T}}$ also den reinen Zeitdefekt, vgl. (2.102), und $\kappa^{\mathbf{\Delta R}}$ war ja gerade der reine Ortsdefekt (2.69). Ist also B unabhängig von u_x, so rühren die einzigen nichttrivialen gemischten Orts–Zeitfehler aus der semi–impliziten Struktur der Diskretisierung ($A \neq 0$) her.

Nach der oben zitierten Definition ist der lokale Diskretisierungsfehler eines Schrittes des Gesamtverfahrens durch die Differenz

$$\tau_{k+1} = \mathbf{u}_{k+1} - \mathbf{u}(t_{k+1})$$

gegeben. Einsetzen der Diskretisierung (2.107) und Berücksichtigung, daß auf der exakten Lösung gestartet wird ($\mathbf{u}_k \equiv \mathbf{u}(t_k)$), liefert:

$$\begin{aligned}
\tau_{k+1} &= -(\mathbf{u}(t_{k+1}) - \mathbf{u}(t_k)) + \left(\mathbf{B}^{\Delta}(t_k,\mathbf{u}(t_k)) - \Delta t\mathbf{A}^{\Delta}\right)^{-1}\Delta t\mathbf{f}^{\Delta}(t_k,\mathbf{u}(t_k)) \\
&= \left(\mathbf{B}^{\Delta}(t_k,\mathbf{u}(t_k)) - \Delta t\mathbf{A}^{\Delta}\right)^{-1}\kappa^{\Delta}(t_k) .
\end{aligned} \tag{2.113}$$

Hat das zu lösende System eine dissipative Natur, so wird der Defektfehler nicht durch die Übertragungsmatrix $\left(\mathbf{B}_k^{\Delta} - \Delta t \mathbf{A}^{\Delta}\right)^{-1}$ verstärkt. Auch bei Hintereinanderausführung mehrerer Schritte findet keine Fehlerverstärkung statt. Das Verfahren ist "stabil".

2.3.4 Angenommene asymptotische Entwicklung des globalen Fehlers

Bezeichne $U(\bar{x}, \bar{t}; \Delta r, \Delta t)$ die numerische Lösung mit der Methode (2.107) zum Zeitpunkt $\bar{t} = t_0 + n \cdot \Delta t$, d.h. nach n Schritten mit der Zeitschrittweite Δt. Dabei sei $\bar{x}$ ein Knoten eines festen Gitters $\bar{\mathcal{G}}$ das durch die Ortsschrittweite Δr und eine nicht näher spezifizierte Gitterfunktion $\xi(r)$ definiert sei. Es wird angenommen, daß der globale Diskretisierungsfehler, d.h. $U(\bar{t}) - u(\bar{t})$, lokal an jedem Knoten $\bar{x}$ eine aymptotische Entwicklung besitzt:

$$\begin{aligned} U(\bar{x}, \bar{t}; \Delta r, \Delta t) - u(\bar{x}, \bar{t}) \quad = \quad & \sum_{\rho=0, l=0; (\rho,l) \neq (0,0)}^{\rho=M, l=N} e_{\rho,l}(\bar{x}, \bar{t}) \Delta R^{2\rho} \Delta T^{l} \\ & + O(\Delta r^{2M+1}, \Delta t^{N+1}) \; . \end{aligned} \tag{2.114}$$

Dabei seien die Koeffizientenfunktionen $e_{\rho,l}(\bar{x}, t)$ auf das Gitter $\bar{\mathcal{G}}$ restringierte Lösungen der linearen, inhomogenen PDG

$$B e'_{\rho,l} = (f_u + f_{u_x} + f_{\mathcal{D}} - (B_u + B_{u_x}) u_t) e_{\rho,l} - d_{\rho,l}(x, t) \tag{2.115}$$

mit

$$e_{\rho,l}(x, t_0) = 0 \tag{2.116}$$

und die in dem Term $O(\Delta r^{2M+1}, \Delta t^{N+1})$ zusammengefaßten Restglieder seien beschränkt.

Ziel dieser Arbeit ist es nun nicht, unter geeigneten gewählten Annahmen an die Eigenschaften des Problems, diese Entwicklung, bzw. um es realistischer zu formulieren, eine gestörte Entwicklung, zu beweisen und geeignete Voraussetzungen zu finden, unter denen diese Störungen "klein" sind. Dieses Thema sei den Spezialisten dieses Gebietes der numerischen Analysis überlassen. Daß bestenfalls gestörte Entwicklungen zu erwarten sind, zeigen die Resultate einer gemeinsamen Arbeit zweier dieser Spezialisten, Lubich/Ostermann [50]. Es werden für ganze Klassen von impliziten Runge–Kutta Methoden sehr drastische, nichtganzzahlige Ordnungsschranken bei Anwendung zur Lösung gewisser Typen von parabolischen Gleichungen (lineare, semi–lineare) gezeigt. Die spezielle Art der Ortsdiskretisierung spielt dabei eine eher untergeordnete Rolle, während die Art der Randbedingung die Stärke der Ordnungsreduktion praktisch determiniert. Interpretiert man die Ergebnisse für die hier verwendete Extrapolationsmethode, so ist eine

Ordungsschranke von $q_{max} = 3 + \varepsilon$ zu erwarten (LUBICH [49]). Dies bedeutet jedoch nicht, daß nicht mit höheren Extrapolationsstufen dennoch genauere numerische Approximationen erreicht werden können. Sie besitzen nur nicht die klassische, beweisbare höhere Ordnung. Eine Situation, wie sie schon im Bereich der Integration von rein differentiell–algebraischen Gleichungen auftritt.

Die genaue Form einer (gestörten) asymptotischen Entwicklung ist natürlich auch eng verknüpft mit der Frage nach den Konvergenzeigenschaften des Zeitdiskretisierungsverfahrens, angewandt auf ein semi–diskretes System der Form (2.63). So wird z.B. in STREHMEL/HUNDSDORFER/WEINER/ARNOLD [76] das Konvergenzverhalten einiger Varianten von linear–impliziten Euler–Diskretisierungen, angewandt auf einfache semi–diskrete Gleichungen, die aus der Diskretisierung von semi–linearen und quasi–linearen parabolischen Gleichungen stammen, betrachtet. Dabei werden z.T. recht restriktive Forderungen, sowohl an die semi–diskreten Operatoren als auch an die Operatoren in der PDG, gestellt. So etwa die Forderung nach klassischer Lipschitzstetigkeit eines Quellterms in der PDG.

Anstelle einer genaueren Darstellung obiger Resultate und einer weiteren (i.w. fruchtlosen) theoretischen Diskussion, soll hier eine experimentelle Untersuchung durchgeführt werden.

2.3.5 Ein numerisches Experiment – Teil I

Mit Hilfe eines nichttrivialen Testproblems soll untersucht werden, ob die Annahme gerechtfertigt ist, daß der globale Diskretisierungsfehler einer Entwicklung der Form (2.114) genügt. Zunächst wird dabei geprüft, ob sich der führende Fehler linear in ΔT und quadratisch in ΔR verhält. Dazu wird das folgende Kunstproblem betrachtet:

KUNSTPROBLEM (**spherical combustion**):

$$
\begin{aligned}
&\text{a)} \quad x \in [0,1]\ , \quad t \in [0, 580] \\
&\text{b)} \quad \frac{\partial u_1}{\partial x} &&= k_1 x^{-2} \frac{\partial}{\partial x}\left(x^2 2 e^{-0.01 u_1} \frac{\partial u_1}{\partial x} \right) + k_2 e^{\frac{-30800}{1.987(u_1+273.16)}} \\
&\qquad \frac{\partial u_2}{\partial x} &&= 10^{-4} x^{-2} \frac{\partial}{\partial x}\left(x^2 \frac{\partial u_2}{\partial x} \right) - k_4 u_2 \\
&\qquad \frac{\partial u_3}{\partial x} &&= 5 \cdot 10^{-5} x^{-2} \frac{\partial}{\partial x}\left(x^2 \frac{\partial u_3}{\partial x} \right) + k_4 u_2 - 2 k_5 u_3^2 \\
&\qquad 0 &&= 10^{-3} x^{-2} \frac{\partial}{\partial x}\left(x^2 \frac{\partial u_3}{\partial x} \right) + 2 k_5 u_3^2 - k_3 u_4 \\
&\text{c)} \quad u_1(x, t_0) &&= 25\ ; \quad u_2(x, t_0) = 5 \\
&\qquad u_3(x, t_0) &&= u_4(x, t_0) = 0 \\
&\text{d)} \quad \frac{\partial}{\partial x} u_j(x_L, t) &&= 0\ ,\ j = 1, \ldots, 4 \\
&\qquad u_1(x_R, t) &&= 25 + 125(1 - e^{-0.01 t}) \\
&\qquad \frac{\partial}{\partial x} u_j(x_R, t) &&= 0\ ,\ j = 2, 3, 4\ .
\end{aligned}
\tag{2.117}
$$

Die Konstruktion dieses Kunstproblems hat zum Ziel, ein Problem zu erzeugen, in dem mehrere der in den vorigen Abschnitten angesprochenen Diskretisierungsschwierigkeiten gleichzeitig auftreten. Bei dem System (2.117) handelt es sich um ein gemischt parabolisch–elliptisches System in Kugelkoordinaten mit z.T. hochnichtlinearen Quelltermen und nichtlinearer Kopplung der Gleichungen. Eine der Randbedingungen führt auf eine zeitabhängige algebraische Bedingungsgleichung im zugehörigen semi–diskreten System (2.63) und die anderen Randbedingungen am rechten Rand stellen keine Symmetriebedingungen dar.

Das Testproblem (2.117) entstand durch Modifikation bzw. Erweiterung einer skalaren Modellgleichung für ein Wärmeleitungsproblem mit exothermer Reaktion in sphärischen Koordinaten, wie es in SCHIESSER [71] angegeben ist. Am Originalproblem kann die physikalische Fragestellung untersucht werden, ob das Aufheizen an der Kugeloberfläche ausreicht, um die Reaktion in Gang zu setzen, was dann letztendlich zur Explosion führt. Obwohl auch für das modifizierte Problem (2.117) diese Explosion zu beobachten ist (bei $t \approx 578$), soll diese Problematik hier nicht näher betrachtet werden. Vielmehr soll ein Integrationsintervall studiert werden, in dem das Gesamtsystem eine starke zeitliche Dynamik aufweist und steile Ortsgradienten auch den Einfluß der Ortsdiskretisierung auf das Gesamtfehlerverhalten sichtbar machen. Da-

zu wird nun der Zeitintegrationspunkt $t = 444$ herausgegriffen, an dem das System sich in einem geeigneten Zustand befindet. Exemplarisch ist in Abbildung 2.1 der örtliche Verlauf der 3. Lösungskomponente für die Zeitpunkte $t_l = 0, 100, 200, 300, 400, 444$ dargestellt.

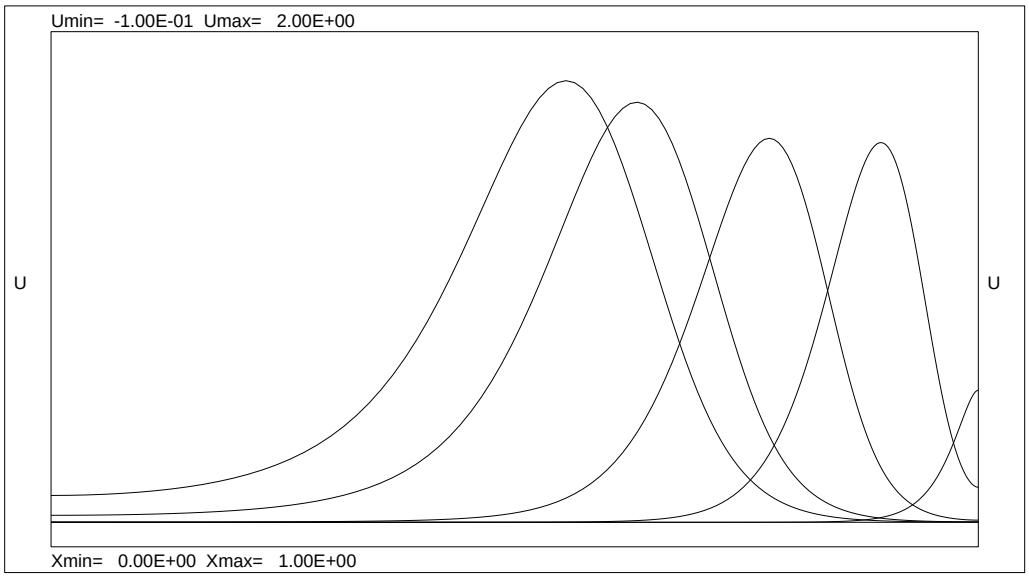


Abbildung 2.1: Lösungskomponente $u_3(x, t_l)$ des Kunstproblems **spherical combustion** zu verschiedenen Zeitpunkten t_l

Von diesem Anfangswert $\bar{t}_0 := 444$ soll nun mit der Diskretisierung (2.107) bis zu einer Endzeit $\bar{t}_{end} := 445$ integriert werden. Als Zeitschrittweite wird zunächst $\Delta t = 1$ verwendet, d.h. es wird nur ein Integrationsschritt durchgeführt. Als Ortsschrittweite wird dabei $\Delta r = 1/20$ gewählt. Die Integration erfolgt allerdings über einem nichtuniformen Gitter $\mathcal{G}^1$, dessen Knoten x_i^1 durch die Wahl einer speziellen Gitterfunktion der Form (2.25,2.26) definiert sind:

$$x_i^1 = \xi((i-1)\Delta r) \;\; ; \; i = 1, \ldots, 21 \tag{2.118}$$

mit

$$\xi(r) = (e^3 - e^{3(1-r)})/(e^3 - e^0) \; ; \;\; r \in [0, 1] \; . \tag{2.119}$$

Diese spezielle Gitterfunktion ist in Abbildung 2.2 dargestellt. Aus der äquidistanten Unterteilung des Einheitsintervalls wird durch (2.119) ein Gitter erzeugt, dessen Knotenverteilung sich am rechten Rand des Gebietes Ω stark verdichtet.

Dieses Gitter bietet in der ersten Phase der Integration (etwa bis $t = 300$) eine hohe Knotendichte in dem Bereich, wo sich steile Ortsgradienten ausbilden. Für den hier herausgegriffenen Zeitpunkt $t = 444$ ist es jedoch nicht

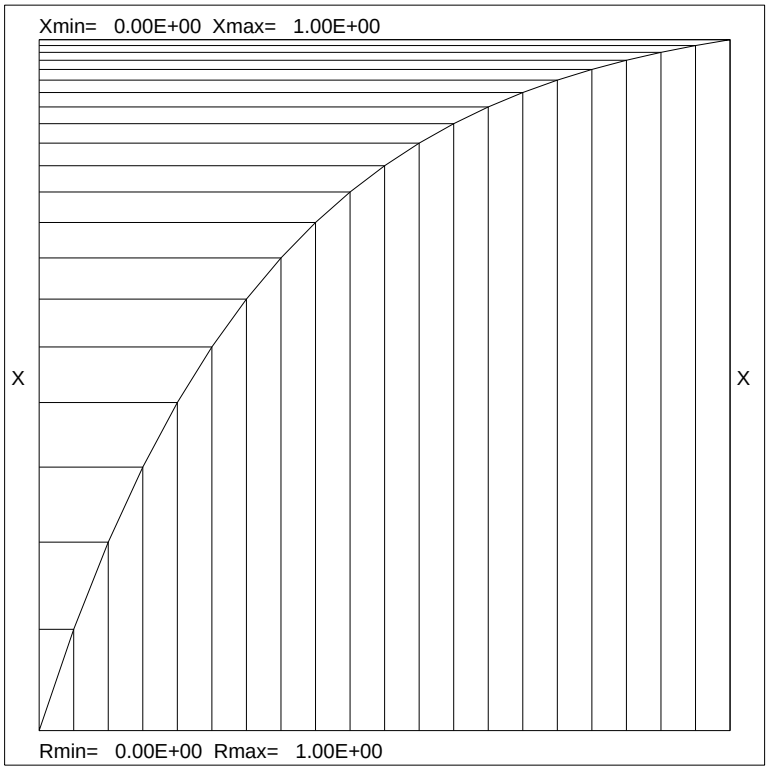


Abbildung 2.2: Die zur Lösung des Kunstproblems **spherical combustion** verwendete Gitterfunktion $\xi(r)$

mehr optimal, ja sogar weniger günstig als ein äquidistantes Gitter gleicher Dimension. In diesem Experiment soll jedoch das Fehlerverhalten untersucht werden und dabei ist die absolute Größe des Fehlers von untergeordneter Bedeutung.

Da für das Kunstproblem (2.117) keine analytische Lösung bekannt ist, müssen sowohl der Startwert $u(x, \bar{t}_0)$ als auch die "exakte" Lösung $u(x, \bar{t}_{end})$ durch eine (möglichst genaue) numerische Integration des Problems (2.117) erzeugt werden. Dazu wird eine klassische Linienmethode verwendet. Das auf einem durch die Gitterfunktion (2.119) definierten nichtuniformen Gitter $\bar{\mathcal{G}}$ der Dimension $n_x = 1281$ gegebene semi–diskrete System der Form (2.63) wird mit einem modernen DAG–Löser bis zum Zeitpunkt $t = 454$ integriert. Hierfür wurde das Mehrschrittverfahren DASSL von Petzold [64] verwendet (Code DDASSL aus der Bibliothek SLATEC, latest change 910624). Als Fehlertoleranz wurde $RTOL = ATOL = 10^{-8}$ vogegeben.

Da die damit erzeugte numerische Lösung eine globale Darstellung bzgl. t

besitzt, liegt eine recht genaue Referenzlösung

$$u^{ref}(x,t)\ ;\ \ x \in \bar{\mathcal{G}}, t \in [0,454] \tag{2.120}$$

für das Problem (2.117) vor, die als Ersatz für die nicht vorhandene analytische Lösung dient.

Um das Fehlerverhalten zu studieren, wird nun mehrmals, mit sukzessiv kleiner werdenden Zeitschrittweiten Δt_l (und entsprechend wachsender Schrittzahl), das Kunstproblem von $\bar{t}_0$ nach $\bar{t}_{end}$ integriert. Dabei wird in allen Fällen das grobe Gitter $\mathcal{G}^1$ verwendet. Da $\mathcal{G}^1 \subset \bar{\mathcal{G}}$, ist die Wahl geeigneter Anfangswerte problemlos (d.h. ohne eine fehlererzeugende Interpolation) möglich. Es werden die Zeitschrittweiten

$$\Delta t_l := \Delta t \cdot \left(\frac{1}{2}\right)^{l-1}, l = 1, ..., 7 \tag{2.121}$$

verwendet. Die dabei erhaltenen numerischen Apporoximationen an die Lösung $u(x, \bar{t}_{end}) \in \mathbb{R}^4$ seien mit

$$U(x_i, \bar{t}_{end}; \Delta r_1, \Delta t_l) \in \mathbb{R}^4\ \ ,\ x_i \in \mathcal{G}^1 \tag{2.122}$$

bezeichnet. Um den Fehler auch bzgl. kleiner werdenden Ortsschrittweiten betrachten zu können, wird die obige Kette von Zeitintegrationen jeweils über sukzessiv feiner werdenden Ortsgittern $\mathcal{G}^\rho$ durchgeführt. Es werden dazu die Schrittweiten

$$\Delta r_\rho := \Delta r \cdot \left(\frac{1}{2}\right)^{\rho-1}, \rho = 1, ..., 7 \tag{2.123}$$

verwendet. Die Lösungen seien mit

$$U(x_i, \bar{t}_{end}; \Delta r_\rho, \Delta t_l)\ \ ,\ x_i \in \mathcal{G}^\rho \tag{2.124}$$

bezeichnet. Es gilt

$$\mathcal{G}^\rho \subset \mathcal{G}^{\rho+1}, \rho = 1, ..., 6$$

sowie

$$\mathcal{G}^7 \equiv \bar{\mathcal{G}}\ ,$$

so daß die Wahl der Anfangswerte und die Restriktion der Lösungen (2.124) auf das Grobgitter $\mathcal{G}^1$ fehlerfrei durchgeführt werden können.

In Abbildung 2.3 sind die beobachteten relativen Fehler der 4. Komponente u_4 am Knoten $x_7 = 0.62452...$

$$\varepsilon_{l,\rho} := \frac{|U_4(x_7^1, \bar{t}_{end}; \Delta r_\rho, \Delta t_l) - u_4^{ref}(x_7^1, \bar{t}_{end})|}{|u_4^{ref}(x_7^1, \bar{t}_{end})|} \tag{2.125}$$

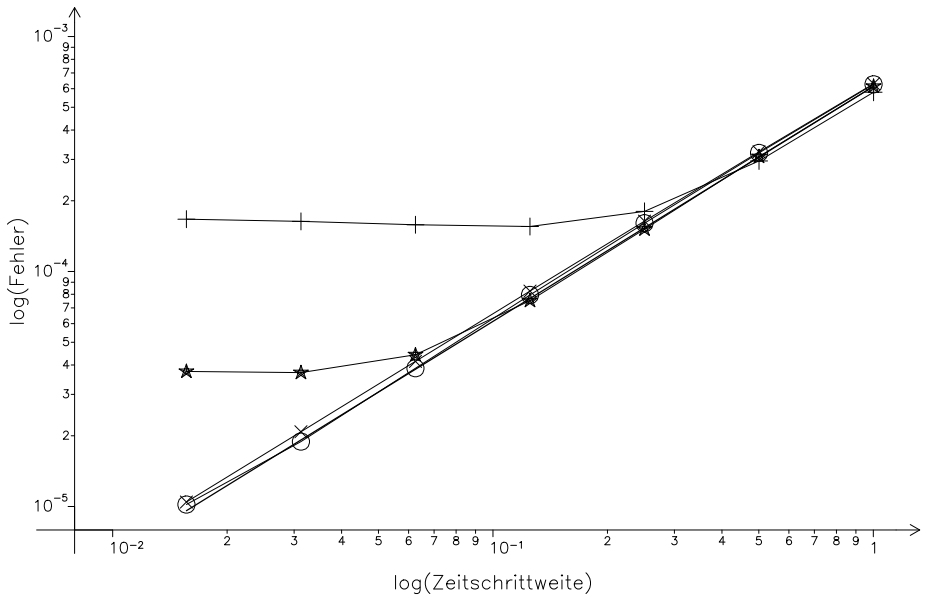


Abbildung 2.3: Fehlerreduktion durch Reduktion von Δt

für

$$\begin{aligned} \rho &= 4 \text{ (Symbol “+”)} \\ \rho &= 5 \text{ (Symbol “}\star\text{”)} \\ \rho &= 6 \text{ (Symbol “}\circ\text{”)} \\ \rho &= 7 \text{ (Symbol “}\times\text{”)} \end{aligned}$$

über alle verwendeten Zeitschrittweiten dargestellt. Zusätzlich ist noch die Kurve

$$\tilde{\varepsilon}_{l,5} := \varepsilon_{1,5} \cdot \left(\frac{1}{2}\right)^{l-1}, l = 1, ..., 7$$

dargestellt (ohne Markierung), die das theoretisch erwartete Fehlerverhalten angibt. Offensichtlich stimmen das beobachtete Fehlerverhalten und das erwartete Fehlerverhalten fast perfekt überein, wenn der Ortsdiskretisierungsfehler nur klein genug ist.

Die spezielle Wahl des Fehlermaßes (2.125) erfolgte, weil für fast alle Approximationen (2.124) der maximale Fehler am 7. Knoten und für die algebraische (4.) Komponente zu beobachten ist. Aber auch in anderen Normen verhält sich der Fehler entsprechend. Insbesondere auch in einer Norm, die global im Ort ist.

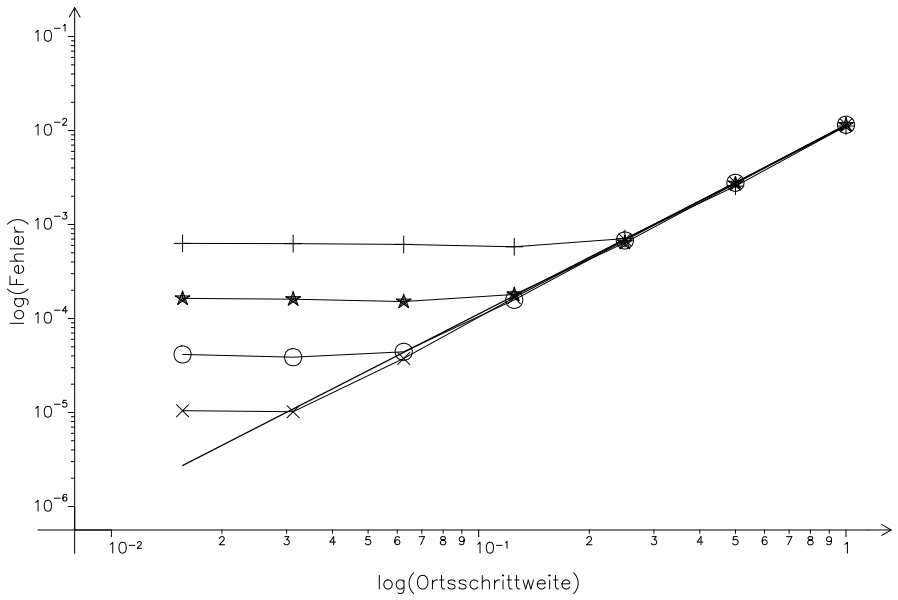


Abbildung 2.4: Fehlerreduktion durch Reduktion von Δr

In Abbildung 2.4 sind nun die beobachteten relativen Fehler $\varepsilon_{l,\rho}$ für

$$\begin{aligned} l &= 1 \text{ (Symbol “}+''\text{)} \\ l &= 3 \text{ (Symbol “}\star''\text{)} \\ l &= 5 \text{ (Symbol “}\circ''\text{)} \\ l &= 7 \text{ (Symbol “}\times''\text{)} \end{aligned}$$

über alle verwendeten Ortsschrittweiten Δr_ρ dargestellt. Zusätzlich ist noch die Kurve

$$\tilde{\varepsilon}_{1,\rho} := \varepsilon_{1,1} \cdot \left(\frac{1}{2}\right)^{2(\rho-1)}, \rho = 1, ..., 7 \tag{2.126}$$

dargestellt (ohne Markierung), die wiederum das theoretisch zu erwartende Fehlerverhalten angibt. Auch hier stimmen das beobachtete Fehlerverhalten und das erwartete Fehlerverhalten fast perfekt überein. Da in der genauesten Approximation $U(x_i, \bar{t}_{end}; \Delta r_7, \Delta t_7)$ der Zeitfehleranteil dominant ist, stellt sich allerding für die letzte Verfeinerung der Ortsdiskretisierungsschrittweite keine Fehlerreduktion mehr ein. Auch bei einer weiteren Verfeinerung der Zeitschrittweite reduziert sich der (beobachtbare!) Ortsfehler beim Übergang $\Delta r_6 \to \Delta r_7$ nicht um den erwarteten Faktor 1/4, da die Referenzlösung nur auf dem gleichen Gitter vorliegt.

Da in der ungenauesten Approximation $U(x^1, \bar{t}_{end}; \Delta r_1, \Delta t_1)$ der Ortsdiskretisierungsfehler offensichtlich dominant ist, läßt sich der Ortsdiskretisierungsfehler auch der Referenzlösung durch $\tilde{\varepsilon}_{1,7}$ nach (2.126) abschätzen. Damit

dürfte zwischen der Referenzlösung und der exakten Lösung von Problem (2.117) folgende Beziehung bestehen:

$$||u^{ref} - u^{exakt}|| \approx 10^{-6} \ . \tag{2.127}$$

Die Ergebnisse dieses numerischen Experiments belegen, daß sich, innerhalb eines gewissen Zeitintervalls (hier $[\bar{t}_0, \bar{t}_{end}]$), der (zeitlich) globale Fehler der Basisdiskretisierung (2.107), örtlich sowohl lokal als auch global, linear in Δt und quadratisch in Δr verhält.

3. Statische Gittererneuerung

In diesem Kapitel soll nun das eigentliche Gesamtverfahren beschrieben werden. Ausgehend von der Basisdiskretisierung (2.107) sollen mit Hilfe eines einheitlichen Steueralgorithmus, folgende Ziele erreicht werden:

- Konstruktion von numerischen Approximationen höherer Ordnung
- interne Fehlerschätzung dieser Approximationen
- optimale Ordnungs– und Schrittweitenwahl für die Zeitdiskretisierung
- statische Erneuerung eines nichtuniformen Ortsgitters.

3.1 Gekoppelte Extrapolation in Raum und Zeit

Die bei der Erörterung der reinen Zeitdiskretisierung schon kurz dargestellte Extrapolationstechnik (vergleiche Abschnitt 2.3.2) wird nun auf die Ortsdiskretisierung ausgedehnt. Es bezeichne weiterhin ΔT die Grundschrittweite der Zeitdiskretisierung, d.h., es wird ein Integrationsschritt

$$\hat{t} \to \hat{t} + \Delta T =: \bar{t} \ , \quad t_0 \leq \hat{t} < \bar{t} \leq t_{end}$$

betrachtet. In Analogie dazu bezeichne ΔR die Grundschrittweite bzgl. der Ortsdiskretisierung. Dabei gelte (n_x sei vorgegeben)

$$\Delta R := \frac{1}{n_x - 1} \ ; \ n_x \geq 2 \ .$$

Durch ΔR sei im Einheitsintervall $[0, 1]$ ein äquidistantes Gitter $\mathcal{R}$ mit Knoten r_i definiert:

$$\mathcal{R} := \{r_i | r_i = (i-1) \cdot \Delta R, i = 1, \ldots, n_x\} \ .$$

Das für die tatsächliche Rechnung verwendete Gitter $\mathcal{G}$ mit den Knoten x_i ist dann durch die Vorgabe einer streng monoton wachsenden Gitterfunktion $\xi(r)$ – siehe (2.25) bzw. (2.26) – definiert. Es wird zunächst angenommen, daß $\xi(r)$ explizit vorgegeben ist.

Um einen Extrapolationsprozeß durchführen zu können, muß das diskretisierte Problem mehrfach mit verschiedenen sogenannten internen Schrittweiten Δt bzw. Δr gelöst werden. Exakte Anfangswerte $u(x, \hat{t})$ für alle Gitterpunkte x_i seien gegeben.

Die Folge der unabhängig voneinander kleiner werdenden internen Schrittweiten $\{\Delta r_\rho\}$ bzw. $\{\Delta t_l\}$ wird durch die Vorgabe zweier Schrittweitenfolgen $\mathcal{F}_r := \{m_\rho\}$ bzw. $\mathcal{F}_t := \{n_l\}$ festgelegt:

$$\begin{aligned} \Delta t_l &:= \frac{\Delta T}{n_l}, \quad l = 1, \ldots, l_{max}, \quad n_{l+1} > n_l, \qquad n_l \in \mathbb{N}^+ \\ \Delta r_\rho &:= \frac{\Delta R}{m_\rho}, \quad \rho = 1, \ldots, \rho_{max}, \quad m_{\rho+1} > m_\rho, \quad m_\rho \in \mathbb{N}^+ \ . \end{aligned}$$

Führt man für ein beliebiges Schrittweitenpaar $(\Delta r_\rho, \Delta t_l)$ zunächst die Ortsdiskretisierung auf dem zugehörigen Ortsgitter

$$\mathcal{G}^\rho = \{x_i | x_i = \xi(r_i), r_i = (i-1)\Delta r_\rho, i = 1, \ldots, m_\rho \cdot n_x - 1\}$$

durch und integriert mit der Zeitdiskretisierung (2.101) das entstandene semi–diskrete System der Form (2.63) durch n_l Schritte mit der Schrittweite Δt_l, so erhält man für die Zeitpunkte $\tilde{t}_{j,l} = \hat{t} + j\Delta t_l, j = 1, \ldots, n_l$ und jeden Knoten $x_{i,\rho} \in \mathcal{G}^\rho$ eine numerische Approximation

$$U(x_{i,\rho}, \tilde{t}_{j,l}; \Delta r_\rho, \Delta t_l) \tag{3.1}$$

an die exakte Lösung $u(x_{i,\rho}, \tilde{t}_{j,l})$. Will man mit Hilfe von Extrapolation daraus Approximationen höherer Ordnung (zumindest "genauere" Approximationen) gewinnen, so ist dies nur an denjenigen Zeitpunkten bzw. Ortsknoten möglich, die allen Approximationen

$$\{U(x_{i,\rho}, \tilde{t}_{j,l}; \Delta r_\rho, \Delta t_l)\}_{\rho=1,\ldots,\rho_{max}}^{l=1,\ldots,l_{max}}$$

gemeinsam sind (und für die eine Entwicklung der Form (2.114) gilt). Für einen Extrapolationsprozeß, der die Approximationen aller Unterteilungen verwenden soll, sind dies gerade der Zeitpunkt $\bar{t} = \hat{t} + \Delta T$ und die Knoten $\{\bar{x}_i\}$ des gröbsten Gitters $\mathcal{G}^1$.

Daraus ergibt sich sofort ein wesentlicher Unterschied zwischen Orts– und Zeitextrapolation bzgl. der Anzahl der praktisch verwendbaren Extrapolationsstufen. Nach einem abgeschlossenem Schritt wird ja i.a. ein weiterer Integrationsschritt folgen. Während zur Weiterintegration die am weitesten t–extrapolierte Approximation als "Anfangswert" verwendet werden kann (die auf allen Gittern vorliegt), liegen die am weitesten r–extrapolierten Werte nur auf dem Grobgitter vor. Zur Weiterintegration mit feineren Gittern müßte daher interpoliert werden. Dies hat letztendlich zur Konsequenz, daß die Tiefe der r–Extrapolation gering sein muß (ρ_{max} klein). Dennoch wird der $r-t$–Extrapolationsprozeß zunächst in voller Allgemeinheit beschrieben und untersucht.

Die eben angesprochene Eigenschaft bedeutet gerade, daß Extrapolationsverfahren keine "natürliche" globale Darstellung der Lösung besitzen. Dies

gilt natürlich auch für die t–Extrapolation. Hier liegen nämlich innerhalb des Intervalls $[\hat{t}, \bar{t}\,]$ keine genaueren Approximationen der Lösung vor. Wird auch innerhalb eines Integrationsschrittes eine in der Zeit globale Darstellung der Lösung benötigt, so läßt sich dies durch einen von HAIRER/OSTERMANN [37] vorgeschlagenen zusätzlichen Extrapolations– und Interpolationprozeß erreichen. Diese Technik kann zwar auf die r–Extrapolation übertragen werden, aber auch damit verschwinden nicht die in Abschnitt 3.3.3 diskutierten Interpolationsprobleme.

3.1.1 2–D–Polynomextrapolation

In Erweiterung der üblichen Schreibweise für Extrapolationsverfahren in einer unabhängigen Variablen wird definiert ($\bar{x}_i \in \mathcal{G}^1$):

$$T_{\rho,1,l,1} := U(\bar{x}_i, \bar{t}; \Delta r_\rho, \Delta t_l)\ . \tag{3.2}$$

Handelt es sich bei der betrachteten PDG um eine skalare Gleichung, so ist $T_{\rho,1,l,1}$ ebenfalls eine skalare Größe. Wird ein System von Gleichungen betrachtet, so ist $T_{\rho,1,l,1}$ eigentlich ein Vektor der Länge n_{pde}. Man kann $T_{\rho,1,l,1}$ auch als Vektor der Länge $n := n_x \cdot n_{pde}$ interpretieren, wenn man statt (3.2)

$$T_{\rho,1,l,1} := (U(\bar{x}_1, \bar{t}; \Delta r_\rho, \Delta t_l), \ldots, U(\bar{x}_{n_x}, \bar{t}; \Delta r_\rho, \Delta t_l))^T \tag{3.3}$$

definiert. Da der unten beschriebene Extrapolationsprozeß komponentenweise (in Größen $T.,.,.,.$) zu verstehen ist, ist eine genaue Unterscheidung im Moment noch nicht notwendig. Im weiteren wird deshalb zunächst $T.,.,.,.$ als skalare Größe aufgefasst. Ebenso soll $U(x, t; \Delta r, \Delta t)$ als skalare Größe angesehen werden, wobei sich aus dem Zusammenhang ergibt, um welchen Knoten x_i, welches Gitter $\mathcal{G}$ und um welche Lösungskomponente es sich handelt.

Gemäß der Annahme, daß für den globalen Diskretisierungsfehler eine asymptotische Entwicklung der Form (2.114) gilt, erhält man für (3.2)

$$T_{\rho,1,l,1} = \sum_{\rho=0}^{M} \sum_{l=0}^{N} e_{\rho,l}(\bar{x}, \bar{t}) \left(\frac{\Delta R}{m_\rho}\right)^{2\rho} \left(\frac{\Delta T}{n_l}\right)^{l} + O(\Delta R^{2M+1}, \Delta T^{N+1})$$

mit

$$e_{0,0}(\bar{x}, \bar{t}\ ) \equiv u(\bar{x}, \bar{t}\ )\ .$$

Von dieser Entwicklung sollen nun die führenden Fehler eliminiert werden. Da neben gemischten Fehlertermen natürlich auch die nur zu $\Delta r^{2\rho}$ oder Δt^l proportionalen Terme zu eliminieren sind, ist dazu eine Extrapolation in Raum und Zeit notwendig. Die Fehlerterme der Entwicklung (2.114) und die Wirkung einer Extrapolation in r und t ist in Abbildung 3.1 illustriert.

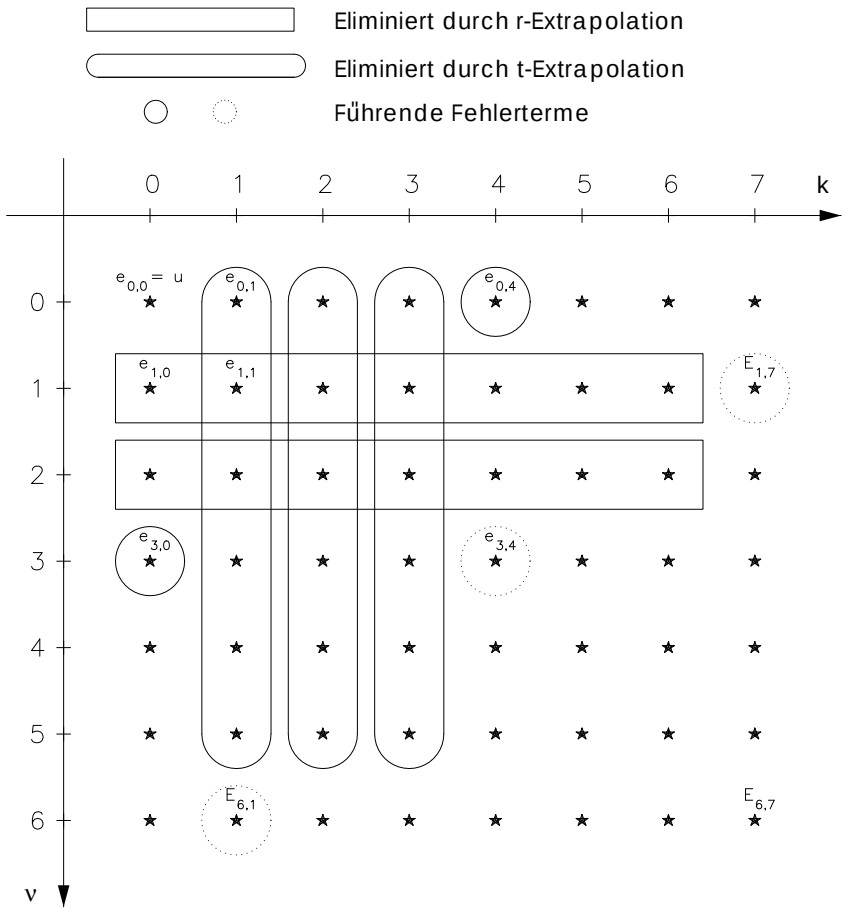


Abbildung 3.1: Fehlerelimination durch 2-D-Extrapolation

Von der Entwicklung (2.114) werden nun die führenden Terme eliminiert, indem ein zweidimensionales interpolierendes Polynom in Δr^2 und Δt aufgestellt und an $\Delta t = 0, \Delta r = 0$ ausgewertet wird. Als Ansatz für das Interpolationspolynom wird ein Polynom vom Grad $(\nu - 1)$ bzgl. Δr^2 und $(k-1)$ bzgl. Δt gewählt

$$P^{\nu,k}(\Delta r^2, \Delta t) := \Delta \mathbf{r}^T \mathbf{P} \Delta \mathbf{t} \tag{3.4}$$

mit

$$\begin{aligned} \Delta \mathbf{r} &= (1, \Delta r^2, \ldots, \Delta r^{2\nu-2})^T \\ \Delta \mathbf{t} &= (1, \Delta t, \ldots, \Delta t^{k-1})^T \\ \mathbf{P} &= \begin{bmatrix} p_{00} & \cdots & p_{0,k-1} \\ \vdots & & \vdots \\ p_{\nu-1,0} & \cdots & p_{\nu-1,k-1} \end{bmatrix}. \end{aligned}$$

Zur Bestimmung der $\nu \cdot k$ Koeffizienten $p_{\rho,l}$ sind mindestens $\nu \cdot k$ Bedingungen notwendig. Es werden dazu die folgenden Interpolationsbedingungen gewählt:

$$\begin{aligned} P^{\nu,k}(\Delta r^2, \Delta t) = T_{\rho,1,l,1} \quad & \text{für} \quad \rho = \mu - \nu + 1, \ldots, \mu \\ & \qquad\; l = j - k + 1, \ldots, j \\ & \text{mit} \quad \mu > \nu, j \geq k \; . \end{aligned} \tag{3.5}$$

Abhängig von der Wahl der Interpolationsbedingungen (d.h. der Spezifikation von μ und j) kann es verschiedene Interpolationspolynome gleichen Grades geben. Um dies zu verdeutlichen, bezeichne $P^{\nu,k}_{\mu,j}$ das Interpolationspolynom, das den Ansatz (3.4) und den Bedingungen (3.5) genügt. Der extrapolierte Wert wird dann mit

$$T_{\mu,\nu,j,k} := P^{\nu,k}_{\mu,j}(\Delta r^2 = 0, \Delta t = 0) = p_{00}$$

bezeichnet.
Das 2–D–Interpolationspolynom $P^{\nu,k}_{\mu,j}(\Delta r^2, \Delta t)$ besitzt folgende Eigenschaften:

- Es existiert und ist eindeutig.
- Es ist längs der Linien $\Delta r^2 = \Delta r^2_\rho, \rho = \mu - \nu + 1, \ldots, \mu$ bzw. $\Delta t = \Delta t_l, l = j-k+1, \ldots, j$ mit dort entsprechend nach (3.4, 3.5) definierten eindimensionalen Interpolationspolynomen $\bar{Q}^k_j(\Delta t; \Delta r_\rho)$ bzw. $\hat{Q}^\nu_\mu(\Delta r^2; \Delta t_l)$ identisch.
- Das Interpolationspolynom $\overline{\overline{Q}}^{\nu}_{\mu}(\Delta r^2)$ über den Stützstellen $(\Delta r_\rho, \bar{Q}^k_j(\Delta t = 0; \Delta r_\rho)), \rho = \mu - \nu + 1, \ldots, \mu$ bzw. das Polynom $\hat{\hat{Q}}^{k}_{j}(\Delta t)$ über den Stützstellen $(\Delta t_l, \hat{Q}^\nu_\mu(\Delta r^2 = 0; \Delta t_l)), l = j - k + 1, \ldots, j$ stimmt mit $P^{\nu,k}_{\mu,j}(\Delta r^2, \Delta t = 0)$ bzw. $P^{\nu,k}_{\mu,j}(\Delta r^2 = 0, \Delta t)$ überein.

Die Vertauschbarkeit der Reihenfolge der r– bzw. t–Extrapolation ist in Abbildung 3.2 exemplarisch für die Berechnung von $T_{2,2,2,2}$ dargestellt.

Für den Approximationsfehler des extrapolierten Wertes $P^{\nu,k}_{\mu,j}(0,0)$ erhält

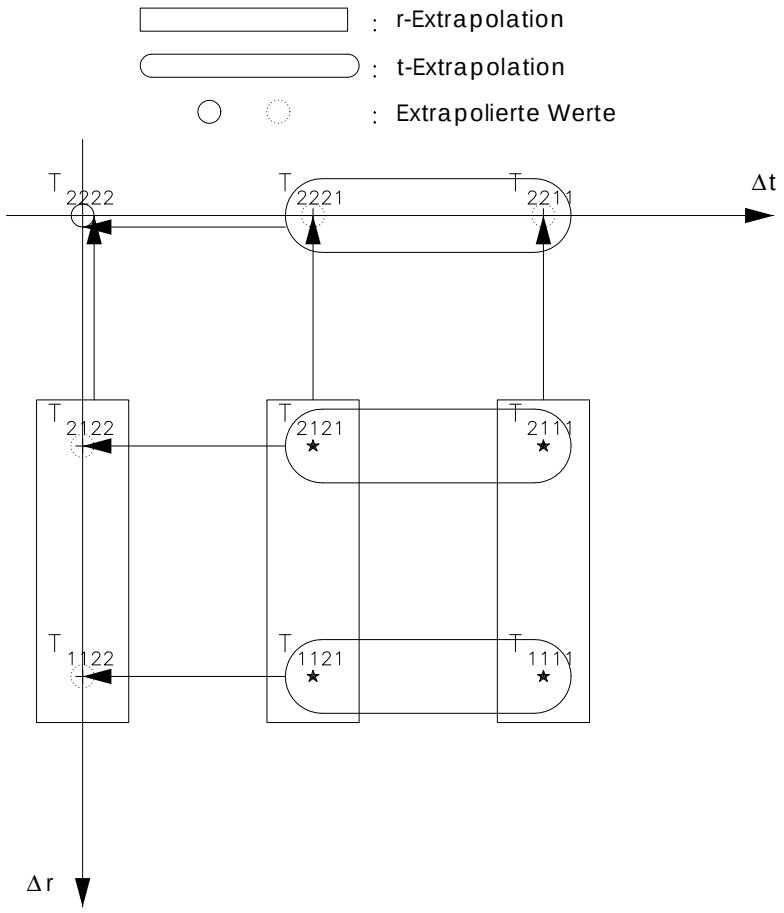


Abbildung 3.2: Mögliche Extrapolationswege um $T_{2,2,2,2}$ zu berechnen.

man (für $\nu \leq M, k \leq N$, es gelte die asymptotische Entwicklung (2.114)):

$$\begin{aligned}
\varepsilon_{\mu,j}^{\nu,k} &:= |P_{\mu,j}^{\nu,k}(0,0) - u(\bar{x},\bar{t})| \\
&= |(-1)^{\nu+1}\beta_{\mu,\nu}e_{\nu,0}(\bar{x},\bar{t})\Delta R^{2\nu} \\
&+ (-1)^{k+1}\gamma_{j,k}e_{0,k}(\bar{x},\bar{t})\Delta T^{k} \\
&+ (-1)^{\nu+k}\beta_{\mu,\nu}\gamma_{j,k}e_{\nu,k}(\bar{x},\bar{t})\Delta R^{2\nu}\Delta T^{k} \\
&+ O(\Delta R^{2\nu+2}) + O(\Delta T^{k+1}) \\
&+ O(\Delta R^{2\nu}\Delta T^{k+1}) + O(\Delta R^{2\nu+2}\Delta T^{k}) \\
&+ O(\Delta R^{2M+1}\Delta T) + O(\Delta T^{N+1}\Delta R^{2})|
\end{aligned} \tag{3.6}$$

mit

$$\begin{aligned}
\beta_{\mu,\nu} &= (m_{\mu-\nu+1}\cdot\ldots\cdot m_{\mu})^{-2} \\
\gamma_{j,k} &= (n_{j-k+1}\cdot\ldots\cdot n_{j})^{-1}\ .
\end{aligned} \tag{3.7}$$

Aufgrund der Eigenschaften von $P_{\mu,j}^{\nu,k}$ kann man zur expliziten Berechnung der extrapolierten Werte $T(\mu,\nu,j,k) \equiv P_{\mu,j}^{\nu,k}(0,0)$ auch hier das bei eindimensionaler Extrapolation üblicherweise verwendete Aitken–Neville–Schema einsetzen. Man berechnet vermöge

$$T_{\rho,1,j,k} := T_{\rho,1,j,k-1} + \frac{T_{\rho,1,j,k-1} - T_{\rho,1,j-1,k-1}}{(n_j/n_{j-k+1}) - 1} \tag{3.8}$$

mit

$$\begin{aligned} & j = 2,3,... \ ; \ k = 2,...,j \\ & \rho \text{ fest } ; \rho \in \{\mu-\nu+1,\ldots,\mu\} \end{aligned}$$

die t–extrapolierten Approximationen und erhält zusammen mit (3.2) das übliche t–Extrapolationstableau (über einem Gitter $\mathcal{G}^\rho$):

$$\begin{array}{llll} T_{\rho,1,1,1} & & & \\ T_{\rho,1,2,1} & T_{\rho,1,2,2} & & \\ T_{\rho,1,3,1} & T_{\rho,1,3,2} & T_{\rho,1,3,3} & \\ \quad\cdot & & \cdot & \\ \quad\cdot & & & \cdot \\ T_{\rho,1,k,1} & \ldots & T_{\rho,1,k,k-1} & T_{\rho,1,k,k}\ . \end{array} \tag{3.9}$$

Führt man dies für alle $\rho \in \{\mu-\nu+1,\ldots,\mu\}$ durch, so kann dann ein r–Extrapolationstableau für jede (j,k)–Kombination berechnet werden:

$$\begin{array}{llll} T_{1,1,j,k} & & & \\ T_{2,1,j,k} & T_{2,2,j,k} & & \\ \quad\cdot & & \cdot & \\ \quad\cdot & & & \cdot \\ T_{\nu,1,j,k} & \ldots & T_{\nu,\nu-1,j,k} & T_{\nu,\nu,j,k}\ . \end{array} \tag{3.10}$$

Auch diese Berechnung erfolgt wiederum sehr effizient durch ein Aitken–Neville–Schema:

$$T_{\mu,\nu,j,k} := T_{\mu,\nu-1,j,k} + \frac{T_{\mu,\nu-1,j,k} - T_{\mu-1,\nu-1,j,k}}{(m_\mu/m_{\mu-\nu+1})^2 - 1} \tag{3.11}$$

mit

$$\begin{aligned} & \mu = 2,3,... \ ; \ \nu = 2,...,\mu \\ & j,k \text{ fest.} \end{aligned}$$

Durch eine entsprechende Reihenfolge des Ausführens von (3.8) und (3.11) können, abhängig von der Reihenfolge der Berechnungen der numerischen Lösungen $T_{\rho,1,l,1}$ während eines Grundschrittes, sukzessiv verschiedene Approximationen $T_{\mu,\nu,j,k}, \mu \geq \nu, j \geq k$ für die Lösung $u(\bar{x},\bar{t})$ berechnet werden.

Inwieweit man mit deren Hilfe Fehlerschätzung, Schrittweitenwahl und Kontrolle des Extrapolationsalgorithmus durchführen kann, soll in den folgenden Abschnitten untersucht werden. Vorab sei jedoch jetzt schon bemerkt, daß dabei nicht alle der oben angegebenen Extrapolationen durchgeführt werden müssen. Und selbst wenn dies der Fall wäre, so ist der dafür benötigte Aufwand immer noch klein im Vergleich zu demjenigen, der zur Berechnung der nicht extrapolierten Approximationen notwendig ist.

Zunächst soll jedoch nochmals auf die Frage eingegangen werden, inwieweit der Extrapolationsprozeß überhaupt Sinn macht, wenn bereits für recht einfache Problemklassen die angenommene Entwicklung (2.114) zumindest gestört ist.

3.1.2 Ein numerisches Experiment – Teil II

Es wird nochmals das bereits in Abschnitt 2.3.6 verwendete Kunstproblem (2.117) betrachtet. Für die Schrittweitenfolgen

$$\begin{aligned} \mathcal{F}_r &= \{1, 2, 4, 8, 16, 32, \ldots\} \\ \mathcal{F}_t &= \{1, 2, 3, 4, 5, 6, \ldots\} \end{aligned} \tag{3.12}$$

wird, wiederum vom "Startwert" $t = 444$, ein Zeitintegrationsschritt mit den Grundschrittweiten $\Delta R = 1/20$ und $\Delta T = 10$ durchgeführt. Bzgl. r werden dabei zunächst $\rho_{max} = 5$ und bzgl. t werden $l_{max} = 7$ Verfeinerungen durchgeführt. Es liegen also insgesamt 35 Approximatiuonen der Form (3.1) an die exakte Lösung $u(\bar{x}, \bar{t})$ bzw. an die Referenzlösung $u^{ref}(\bar{x}, \bar{t})$ vor. Durch Extrapolation nach (3.8) bzw. (3.11) werden daraus weitere Approximationen

$$T_{\mu,\nu,j,k} \ , \ \mu = 1, \ldots, 5; \nu = 1, \ldots, \mu; \ j = 1, \ldots, 7; k = 1, \ldots, j \tag{3.13}$$

erzeugt. In Abbildung 3.3 ist der "Fehler" einiger Approximationen bzgl. der Referenzlösung (2.120) aufgetragen. Als Fehlermaß wird dabei wie schon in den Abbildungen 2.3, 2.4 der Absolutbetrag des relativen Fehlers der 4. Komponente am 7. Knoten des Grundgitters $\mathcal{G}^1$ verwendet.

Die mit dem Symbol "+" markierte Linie gibt den Fehlerverlauf der Approximationen $T_{\mu,\mu,7,1}$ für $\mu = 1, ..., 5$ an, d.h. die bzgl. r voll extrapolierten Werte, während bzgl. der Zeit nicht extrapoliert wurde. Dabei werden die Werte betrachtet, die sich bei der Integration mit der kleinsten internen Schrittweite $\Delta t = \Delta T/7$ ergeben. Offensichtlich dominiert für diese Approximationen der durch die Zeitdiskretisierung hervorgerufene Fehler den Gesamtfehler. Ab $\mu = 2$ reduzieren weitere r–Extrapolationsschritte den Gesamtfehler nicht mehr.

Ein ähnliches Verhalten zeigen die Diagonalelemente des t–Extrapolationstableaus (3.9), wenn bzgl. der Ortsdiskretisierung nicht extrapoliert wird.

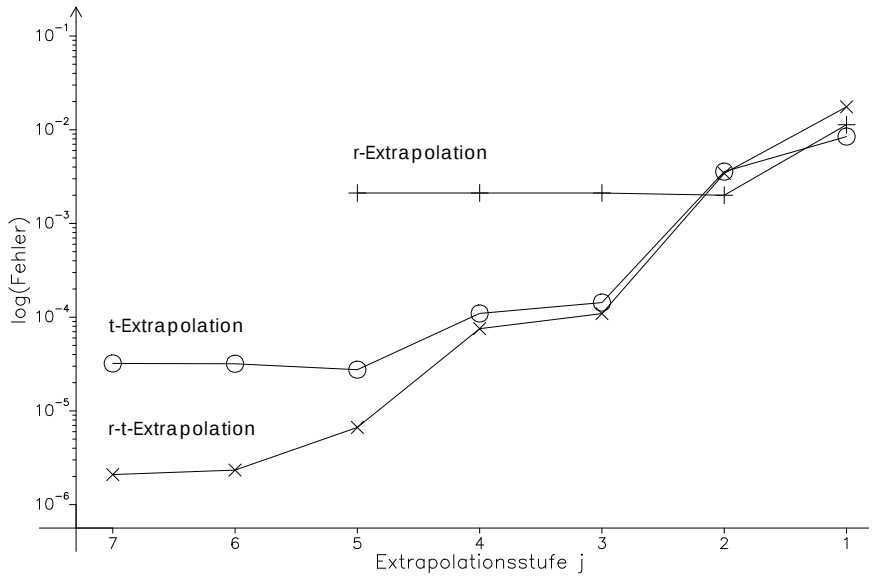


Abbildung 3.3: Fehlerreduktion durch Extrapolation

Dabei sind die Approximationen auf der feinsten, nicht r–extrapolierten Ortsdiskretisierung $T_{5,1,j,j}$ aufgetragen (Symbol "$\circ$"). Da der Ortsdiskretisierungsfehler auf dem feinsten Gitter bereits recht klein ist, führt nur der letzte t–Extrapolationsschritt zu keiner weiteren Fehlerreduktion.

Eine Fehlerreduktion des Gesamtfehlers bis auf das Niveau der Referenzlösung (ca. 10^{-6}, vgl. (2.127)) erhält man nur, wenn bzgl. r und t extrapoliert wird. Exemplarisch ist in Abbildung 3.3 der Fehlerverlauf von $\{T_{max\{j,5\},max\{j,5\},j,j}\}_{j=1,\ldots,7}$ angegeben (Symbol "$\times$"). Wiederum ergeben sich sehr ähnliche Bilder, wenn andere Fehlermaße verwendet werden.

Will man das Fehlerverhalten für noch höher extrapolierte Werte oder von Approximationen studieren, die sich bei Verkleinerung der Grundschrittweite(n) ergeben, so muß als Referenzlösung eine noch weiter extrapolierte Approximation verwendet werden. Die Integration des Kunstproblems mit noch deutlich feinerem Gitter und stringenteren Fehlertoleranzen ist zu rechenzeitintensiv (bereits mehrere Tage um eine Genauigkeit von 10^{-9} anstelle von 10^{-6} in (2.127) zu erreichen).

Daß z.B. eine Ersetzung

$$u^{ref} := T_{7,7,7,7} \tag{3.14}$$

durchaus berechtigt ist, zeigt sich, wenn man zur Fehlerberechnung der in Abbildung 3.3 angegebenen Approximationen die durch (3.14) definierte Approximation als Referenzlösung verwendet. Es ergeben sich praktisch identische

Resultate, mit zwei Ausnahmen. Die Fehler von $T_{5,5,6,6}$ bzw. $T_{5,5,7,7}$ ergeben sich zu $2 \cdot 10^{-7}$ bzw. $1 \cdot 10^{-11}$. Generell ist bei einem derartigen Vorgehen zu beachten, daß die Startwerte $u^{ref}(x_i, \hat{t}), x_i \in \mathcal{G}^7$ nicht exakt, sondern fehlerbehaftet sind.

Nachdem bereits eine Fehlerreduktion für die nicht extrapolierten Approximationen bei kleiner werdenden Diskretisierungsschrittweiten zu beobachten war – siehe Abbildungen 2.3,2.4 –, hat das obige Experiment gezeigt, daß auch durch die Wahl höherer Extrapolationsstufen der globale Diskretisierungsfehler eines Integrationsschrittes deutlich reduziert werden kann. Dies muß nicht notwendigerweise bedeuten, daß das Verfahren wirklich eine höhere Ordnung besitzt, aber für die algorithmische Realisierung des Gesamtverfahrens ist die Frage nach der wahren (klassischen) Ordnung ohnehin weniger wichtig als die Tatsache, daß höhere Extrapolationsstufen zur Fehlerreduktion führen.

Dennoch ist es interessant, mit Hilfe eines weiteren numerischen Experiments die tatsächlich beobachtbare Ordnung zu bestimmen. Die Ordnung bei r–Extrapolation kann dabei recht einfach bestimmt werden. Dazu wird das Problem mit anderer Grundschrittweite, etwa $\Delta\tilde{R}$, erneut gelöst und die beobachteten Fehler der interessierenden Approximationen werden in geeigneter Weise miteinander verglichen.

Nimmt man an, daß die Grundschrittweiten und Extrapolationsstufen derart gewählt sind, daß im führenden Fehler der zu untersuchenden Approximationen der Ortsdiskretisierungsfehler dominant ist, so folgt aus (3.6):

$$\varepsilon_{\mu,j}^{\nu,k} \doteq \beta_{\mu,\nu} |e_{\nu,0}(\bar{x}, \bar{t})| \Delta R^{2\nu} \ , \quad \bar{x} \in \mathcal{G}^1 \tag{3.15}$$

bzw.

$$\tilde{\varepsilon}_{\mu,j}^{\nu,k} \doteq \beta_{\mu,\nu} |e_{\nu,0}(\tilde{x}, \bar{t})| \Delta \tilde{R}^{2\nu} \ , \quad \tilde{x} \in \tilde{\mathcal{G}}^1 \ . \tag{3.16}$$

Hat man sinnvollerweise die modifizierte Grundschrittweite $\Delta\tilde{R}$ so gewählt, daß $\mathcal{G}^1 \subset \tilde{\mathcal{G}}^1$ oder $\tilde{\mathcal{G}}^1 \subset \mathcal{G}^1$, dann kann für alle Knoten $x \in \mathcal{G}^1 \cap \tilde{\mathcal{G}}^1$ die beobachtete Ordnung in natürlicher Weise definiert werden:

$$q_{\mu,\nu}^{obs} := \frac{log(\varepsilon_{\mu,j}^{\nu,k}) - log(\tilde{\varepsilon}_{\mu,j}^{\nu,k})}{log(\Delta R) - log(\Delta\tilde{R})} \ . \tag{3.17}$$

Gelten die Beziehungen (3.15, 3.16) exakt, so erhält man:

$$\begin{aligned} q_{\mu,\nu}^{obs} &= \frac{log(\beta_{\mu,\nu}|e_{\nu,0}(x,\bar{t})|\Delta R^{2\nu}) - log(\beta_{\mu,\nu}|e_{\nu,0}(x,\bar{t})|\Delta\tilde{R}^{2\nu})}{log(\Delta R) - log(\Delta\tilde{R})} \\ &= \frac{2\nu log(\Delta R) - 2\nu log(\Delta\tilde{R})}{log(\Delta R) - log(\Delta\tilde{R})} \\ &= 2\nu \ . \end{aligned} \tag{3.18}$$

Eine alternative Definition, die nur die durch Unterteilung der Grundschrittweite ΔR berechneten Werte verwendet, ergibt sich durch folgende Überlegung.

Für alle Extrapolationsstufen, die kleiner als die maximale sind, liegen mehrere Werte gleicher Ordnung vor (in der Spalte eines Extrapolationstableaus). Sie unterscheiden sich nur in den dabei verwendeten internen Schrittweiten. Nimmt man nun wieder Dominanz des Ortsdiskretisierungsfehlers in (3.6) an, so gilt wegen (3.7) und (3.15) für $\mu = \nu + 1, ..., \rho_{max}$ die folgende Beziehung zwischen den Fehlern der Approximationen $T_{\mu,\nu,.,.}$ und $T_{\mu-1,\nu,.,.}$:

$$\varepsilon_{\mu,j}^{\nu,k} = \varepsilon_{\mu-1,j}^{\nu,k} \left(\frac{m_{\mu-\nu}}{m_\mu} \right)^2 . \tag{3.19}$$

Einsetzen der Schrittweitenfolge $\{m_\rho\}$ nach (3.12) liefert:

$$\varepsilon_{\mu,j}^{\nu,k} = \varepsilon_{\mu-1,j}^{\nu,k} \left(\frac{2^{\mu-\nu-1}}{2^{\mu-1}} \right)^2 = \varepsilon_{\mu-1,j}^{\nu,k} 2^{-2\nu} . \tag{3.20}$$

Daraus ergibt sich als alternative Definition für die beobachtete Ordnung:

$$\hat{q}_{\mu,\nu}^{obs} := ld(\varepsilon_{\mu-1,j}^{\nu,k}) - ld(\varepsilon_{\mu,j}^{\nu,k}) . \tag{3.21}$$

Wählt man in (3.17) gerade $\Delta\tilde{R} = 2\Delta R$ so ist $\tilde{\varepsilon}_{\mu,j}^{\nu,k} \equiv \varepsilon_{\mu-1,j}^{\nu,k}$ und (3.17) und (3.21) stimmen überein. Die beobachteten Ordnungen $\hat{q}^{obs}$, die sich bei dem oben beschriebenen numerischen Experiment ergeben – allerdings mit einer Grundschrittweite $\Delta T = 0.1$, um Dominanz des Ortsdiskretisierungsfehlers zu erreichen – sind in Tabelle 3.1 zusammengefaßt. Als Referenzlösung wird $T_{7,7,7,7}$ verwendet.

μ	$\nu = 1$	$\nu = 2$	$\nu = 3$	$\nu = 4$	$\nu = 5$
2	2.07				
3	2.01	4.17			
4	2.00	4.04	6.07		
5	2.00	4.01	6.06	6.37	
6	2.00	4.00	5.92	5.64	4.83

Tabelle 3.1: Beobachtete Ordnungen $\hat{q}_{\mu,\nu}^{obs}$ bei r–Extrapolation.

Bis einschließlich $\nu = 3$ ergeben sich dabei die erwarteten Ordnungen mit hoher Genauigkeit. Daß sich für die noch weiter extrapolierten Approximationen das erwartete Fehlerverhalten nicht mehr einstellt, liegt (zumindest auch) daran, daß sich der beobachtete Fehler dann in Größenordnungen bewegt, wo sich der Einfluß der doch recht ungenauen Anfangswerte (bestenfalls eine Genauigkeit im Bereich $[10^{-7}, 10^{-6}]$) störend bemerkbar macht. Diese Genauigkeit ist in Relation zur geschätzten Genauigkeit $\varepsilon_{4,4,7,7} = 1.05855 \times 10^{-9}$ und

$\varepsilon_{5,4,7,7} = 1.27923 \times 10^{-11}$ zu sehen, den "ungenauesten" Werten die verwendet werden, um eine beobachtete Ordnung $\hat{q}^{obs}$ für $\nu = 4$ zu berechnen.

Geht man analog zu (3.19, 3.20) vor, um eine Definition der beobachteten Ordnung p^{obs} der t–Extrapolation abzuleiten, so erhält man für die Schrittweitenfolge $\mathcal{F}_t$ aus (3.12) die Definition:

$$\hat{p}^{obs}_{j,k} := j(1 - \varepsilon^{\nu,k}_{\mu,j})/\varepsilon^{\nu,k}_{\mu,j-1}) \; . \tag{3.22}$$

Die beobachteten Ordnungen $\hat{p}^{obs}$, die sich in obigem numerischen Experiment ergeben – nun wieder mit $\Delta T = 10$ – sind in Tabelle 3.2 zusammengefaßt. Als Referenzlösung kann nun wieder die übliche Referenzlösung (2.120) verwendet werden, da die extrapolierten Werte nicht genauer als diese werden. Dadurch kann auch noch für $j = 7$ in sinnvoller Weise $\hat{p}^{obs}_{7,6}$ berechnet werden.

j	$k = 1$	k =2	k =3	$k = 4$	k =5	k=6
2	0.58					
3	0.79	1.94				
4	0.85	1.86	0.90			
5	0.88	1.88	2.58	4.23		
6	0.90	1.89	2.76	4.31	4.78	
7	0.91	1.91	2.83	4.31	5.04	6.68

Tabelle 3.2: Beobachtete Ordnungen $\hat{p}^{obs}_{j,k}$ bei t–Extrapolation.

Ersetzt man das Fehlermaß (2.125) durch eine (bzgl.x) globale Norm, ist die Übereinstimmung mit den erwarteten Ordnungen sogar noch etwas besser, insbesondere der stark abweichende Wert $\hat{p}^{obs}_{4,3} = 0.9$ verschwindet.

Eine zu (3.17) analoge Definition ist natürlich ebenfalls möglich. Allerdings ergibt sich dabei das Problem, daß dann Approximationen zu verschiedenen Zeitpunkten verglichen werden müßen. Statt dessen kann man z.B. untersuchen, inwieweit der globale Fehler der maximal t–extrapolierten Approximation, d.h. $\varepsilon^{\nu,k}_{\mu,k}(x, t + \Delta T)$, der Annahme

$$||\varepsilon^{\nu,k}_{\mu,k}|| \sim \Delta T^{\bar{p}}$$

genügt. Die so "definierte" Ordnung $\bar{p}$ wird häufig als Ordnung des lokalen Verfahrensfehlers bezeichnet und ist nicht identisch mit der klassischen Konvergenzordnung eines Verfahrens.

Für den Fehler der maximal t–extrapolierten Approximation folgt aus (3.6), nun unter der Annahme, daß ΔR bzw. μ, ν so gewählt sind, daß der Zeitdiskretisierungsfehler dominant ist,

$$\varepsilon^{\nu,k}_{\mu,k} \doteq \gamma_{k,k}||e_{0,k}(x, t + \Delta T)||\Delta T^k \; .$$

Taylorentwicklung von $e_{0,k}(x, t+\Delta T)$ um den Anfangszeitpunkt t des Schrittes und mit der Annahme (2.116), erhält man als zu erwartende Ordnung:

$$\bar{p} = k + 1 \; . \tag{3.23}$$

Ob diese Ordnung tatsächlich beobachtbar ist, soll nun, analog zu dem Vorgehen in LUBICH [48], untersucht werden.

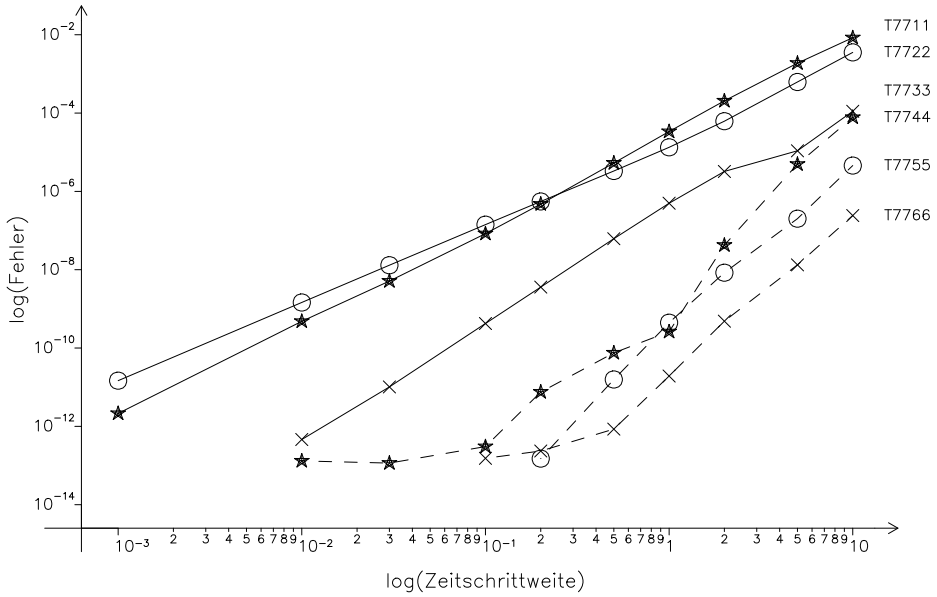


Abbildung 3.4: Schrittweiten/Fehlerdiagramm für die Approximationen $T_{7,7,k,k}$ $(n_1 = 1)$

Dazu wird (2.117) mit sukzessiv kleiner werdenden Grundschrittweiten ΔT und jeweils voller Extrapolation in Raum und Zeit gelöst. Der Fehler der Approximationen $T_{7,7,k,k}, k = 1, \ldots, 6$ (verglichen mit $T_{7,7,7,7}$) ist in Abbildung 3.4 als Funktion von ΔT aufgetragen. Da dazu ein doppelt logarithmischer Maßstab verwendet wurde, sollten sich bei Gültigkeit von (3.23) Geraden mit Steigung $\bar{p}$ ergeben. Dies ist allerdings nicht zu erwarten, da hier ein gemischt parabolisch/elliptisches Problem vorliegt, und dessen Semi–Diskretisierung ein differentiell–algebraisches System darstellt, für das die Zeitdiskretisierung nur eine gestörte asymptotische Entwicklung besitzt. Darüber hinaus liegen weder exakte Anfangswerte noch eine exakte Referenzlösung vor. Doch in Bereichen, wo das Problem mit "vernünftigen" Schrittweiten gelöst wird, zeigt sich ein insgesamt doch sehr befriedigendes Verhalten. Insbesondere ist auch zu sehen, daß bei gleicher Schrittweite eine Erhöhung der Extrapolationsstufe einen Genauigkeitsgewinn zur Folge hat.

Interessant ist nun noch ein Vergleich mit den Resultaten eines entsprechenden Experiments, in dem die theoretisch vorzuziehende Schrittweitenfolge $\mathcal{F}_t = \{2, 3, 4, \ldots\}$ verwendet wird. Die Ergebnisse sind in Abbildung 3.5 dargestellt und zeigen kein signifikant besseres Verhalten. Wiederum sind die in beiden Experimenten beobachteten Ergebnisse relativ stabil gegen eine Änderung der Norm. Betrachtet man z.B. das Fehlerverhalten einer differentiellen Komponente, so ergibt sich ein sehr ähnliches Verhalten.

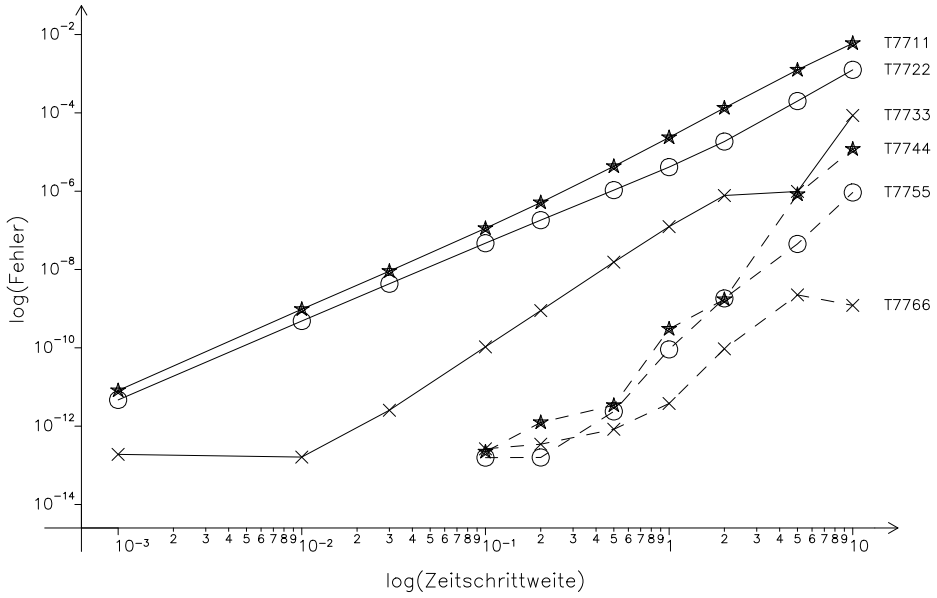


Abbildung 3.5: Schrittweiten/Fehlerdiagramm für die Approximationen $T_{7,7,k,k}$ $(n_1 = 2)$

Da die hier dargestellten experimentellen Resultate sich auch in vielen weiteren Rechnungen bestätigen, kann man davon ausgehen, daß die angenommene asymptotische Entwicklung (2.114) eine sichere Basis für die nun hergeleitete Fehlerschätzung und Schrittweitenkontrolle bietet. Man kann aber auch umgekehrt folgern. Nur wenn die Güte der Diskretisierung und der theoretischen Voraussetzungen durch eine permanente adaptive Kontrolle und Anpassung der Schrittweiten überprüft wird, bietet das Gesamtverfahren ein hohes Maß an Sicherheit.

3.2 Fehlerschätzung

Sei nun unter $T_{\mu,\nu,j,k}$ der durch (3.3) definierte n–Vektor der extrapolierten Werte an allen $\bar{x} \in \mathcal{G}^1$ zu verstehen $(n = n_x \cdot n_{pde})$. Ebenso bezeichne

$u \equiv u(\bar{x},\bar{t})$ bzw. $e_{\rho,l} \equiv e_{\rho,l}(\bar{x},\bar{t})$ die n–Vektoren der Werte der exakten Lösung $u(\bar{x},\bar{t})$ bzw. die der Koeffizientenfunktionen $e_{\rho,l}(\bar{x},\bar{t})$ an allen Knoten $\bar{x}_i, i = 1,\ldots,n_x$ des Grobgitters $\mathcal{G}^1$ zum Zeitpunkt $\bar{t}$. Bezeichne $\|\cdot\|$ eine Vektornorm, die hier zunächst nicht weiter spezifiziert sei, jedoch als differenzierbar angenommen werde. Zur Wahl der Norm siehe Abschnitt 5.1.1. Damit ergibt sich als natürliche Definition des wahren Fehlers ε^0 einer numerischen Approximation $T_{\mu,\nu,j,k}$:

$$\varepsilon^0_{\mu,\nu,j,k} := \|T_{\mu,\nu,j,k} - u\| \; . \tag{3.24}$$

Mit Hilfe der komponentenweisen Fehlerdarstellung (3.6) erhält man:

$$\begin{aligned} \varepsilon^0_{\mu,\nu,j,k} \;=\; & \|(-1)^{\nu+1}\beta_{\mu,\nu}e_{\nu,0}\Delta R^{2\nu} + O(\Delta R^{2\nu+2}) \\ & +(-1)^{k+1}\gamma_{j,k}e_{n,k}\Delta T^k + O(\Delta T^{k+1}) \\ & +(-1)^{\nu+k}\beta_{\mu,\nu}\gamma_{j,k}e_{\nu,k}\Delta R^{2\nu}\Delta T^k \\ & +O(\Delta R^{2\nu+2}\Delta T^k) + O(\Delta R^{2\nu}\Delta T^{k+1}) \\ & +O(\Delta R^{2M+2}\Delta T) + O(\Delta R^2\,,\Delta T^{N+1})\| \end{aligned} \tag{3.25}$$

wobei

$$\mu,\nu < M \;;\, j,k < N \;;\; \beta_{\mu,\nu},\gamma_{j,k} \text{ nach (3.7)} \; .$$

Für die folgenden Überlegungen wird angenommen, daß die Terme

$$O(\Delta R^{2M+2}\Delta T) \;,\; O(\Delta R^2\Delta T^{N+1}) \tag{3.26}$$

vernachlässigt werden können. Damit erhält man den führenden Fehler von $\varepsilon^0_{\mu,\nu,j,k}$

$$\varepsilon^1_{\mu,\nu,j,k} :\doteq \varepsilon^0_{\mu,\nu,j,k} \tag{3.27}$$

als

$$\begin{aligned} \varepsilon^1_{\mu,\nu,j,k} \;=\; & \|(-1)^{\nu+1}\beta_{\mu,\nu}e_{\nu,0}\Delta R^{2\nu} + (-1)^{k+1}\gamma_{j,k}e_{0,k}\Delta T^k \\ & +(-1)^{\nu+k}\beta_{\mu,\nu}\gamma_{j,k}e_{\nu,k}\Delta R^{2\nu}\Delta T^k\| \; . \end{aligned} \tag{3.28}$$

Aufgrund der additiven und vorzeichenbehafteten Struktur des ε^1 definierenden Ausdrucks, ist $\varepsilon^1_{\mu,\nu,j,k}$ als algorithmisch brauchbares Maß für den wahren Fehler von $T_{\mu,\nu,j,k}$ ungeeignet. Denn selbst unter den Annahmen, daß die Fehleranteile der höheren Potenzen klein gegen die der führenden Potenzen sind, d.h.

$$\varepsilon^1_{\mu,\nu,j,k} \approx \varepsilon^0_{\mu,\nu,j,k} \; , \tag{3.29}$$

und daß eine gute numerische Schätzung $\bar{\varepsilon}^1$ von ε^1, z.B. durch

$$\bar{\varepsilon}^1_{\mu,\nu,j,k} := \|T_{\mu,\nu,j,k} - T_{\mu,\nu+1,j,k+1}\| \;;\, \nu+1 \le \mu,\; k+1 \le j \tag{3.30}$$

möglich ist, also

$$\bar{\varepsilon}^1_{\mu,\nu,j,k} \approx \varepsilon^1 \approx \varepsilon^0$$

gilt, hat ε^1 (und damit auch $\bar{\varepsilon}^1$) immer noch zwei entscheidende Nachteile. Zum einen läßt sich daraus keine getrennte Schrittweitenschätzung für ΔR und ΔT ableiten, zum anderen stellt bereits ε^0 (und somit auch $\bar{\varepsilon}^1$) kein monotones Fehlermaß dar. Gerade ein streng monoton fallendes Verhalten des Fehlers bzgl. wachsender Ordnung, kleineren internen Schrittweiten und kleineren Grundschrittweiten ist jedoch die grundlegende Voraussetzung für eine Ordnungs– und Schrittweitenkontrolle. Deshalb wird hier, basierend auf der Darstellung (3.25) und der Annahme (3.26), eine modifizierte Definition des wahren Fehlers verwendet:

$$\text{a)} \quad \hat{\varepsilon}_{\mu,\nu,j,k} := \hat{\varepsilon}^x_{\mu,\nu,j,k} + \hat{\varepsilon}^t_{\mu,\nu,j,k} + \hat{\varepsilon}^{xt}_{\mu,\nu,j,k}$$

mit

$$\begin{aligned}
\text{b)} \quad \hat{\varepsilon}^x_{\mu,\nu,j,k} &:= \|(-1)^{\nu+1}\beta_{\mu,\nu}e_{\nu,0}\Delta R^{2\nu} + O(\Delta R^{2\nu+2})\| \\
\text{c)} \quad \hat{\varepsilon}^t_{\mu,\nu,j,k} &:= \|(-1)^{k+1}\gamma_{j,k}e_{0,k}\Delta T^{k} + O(\Delta T^{k+1})\| \\
\text{d)} \quad \hat{\varepsilon}^{xt}_{\mu,\nu,j,k} &:= \|(-1)^{\nu k+2}\beta_{\mu,\nu}\gamma_{j,k}\Delta R^{2\nu}\Delta T^{k} + O(\Delta R^{2\nu+2}\Delta T^{k}) \\
&\quad + O(\Delta R^{2\nu}\Delta T^{k+1})\| \; .
\end{aligned} \tag{3.31}$$

Das heißt, $\hat{\varepsilon}^x$ beschreibt den reinen Fehleranteil durch Ortsdiskretisierung, $\hat{\varepsilon}^t$ beschreibt den reinen Fehleranteil durch Zeitdiskretisierung und ε^{xt} alle gemischten Anteile.

Unter Verwendung eines derartigen Fehlermaßes erhält man als führenden Fehler:

$$\text{a)} \quad \varepsilon_{\mu,\nu,j,k} := \varepsilon^x_{\mu,\nu,j,k} + \varepsilon^t_{\mu,\nu,j,k} + \varepsilon^{xt}_{\mu,\nu,j,k} \doteq \hat{\varepsilon}_{\mu,\nu,j,k}$$

mit

$$\begin{aligned}
\text{b)} \quad \varepsilon^x_{\mu,\nu,j,k} &= \beta_{\mu,\nu}\|e_{\nu,0}\|\Delta R^{2\nu} &&\doteq \hat{\varepsilon}^x_{\mu,\nu,j,k} \\
\text{c)} \quad \varepsilon^t_{\mu,\nu,j,k} &= \gamma_{j,k}\|e_{0,k}\|\Delta T^{k} &&\doteq \hat{\varepsilon}^t_{\mu,\nu,j,k} \\
\text{d)} \quad \varepsilon^{xt}_{\mu,\nu,j,k} &= \beta_{\mu,\nu}\gamma_{j,k}\|e_{\nu,k}\|\Delta R^{2\nu}\Delta T^{k} &&\doteq \hat{\varepsilon}^{xt}_{\mu,\nu,j,k} \; .
\end{aligned} \tag{3.32}$$

Ein derartig definierter Fehler ε hat nun im Gegensatz zu ε^1 die gewünschten Eigenschaften. Zunächst ergibt sich durch Vergleich mit (3.28) unmittelbar:

$$\varepsilon_{\mu,\nu,j,k} \geq \varepsilon^1_{\mu,\nu,j,k} \; .$$

Das modifizierte Fehlermaß ist also evtl. zu pessimistisch, aber sicherer. Aufgrund der Definition von $\beta_{\mu,\nu}$ und $\gamma_{j,k}$ (vgl. (3.7) gilt für alle Tupel (μ,ν,j,k) mit $(\nu \leq \mu - 1,\ k \leq j - 1)$:

$$\begin{aligned}
\text{a)} \quad \varepsilon_{\mu,\nu,j,k} &< \varepsilon_{\mu-1,\nu,j,k} \\
\text{b)} \quad \varepsilon_{\mu,\nu,j,k} &< \varepsilon_{\mu,\nu,j-1,k}
\end{aligned} \tag{3.33}$$

und es folgt:

$$\varepsilon_{\mu,\nu,j,k} < \varepsilon_{\mu-1,\nu,j-1,k} \ . \tag{3.34}$$

Man hat damit die gewünschte Eigenschaft eines streng monoton fallenden Fehlerverhaltens bzgl. Verfeinerung der internen Diskretisierungsschrittweiten Δr und/oder Δt bei gleichbleibender Ordnung 2ν bzw. k. Die Frage nach Monotonie bzgl. wachsender Ordnung muß auch hier, wie schon bei 1–D–Extrapolationsverfahren, durch die entsprechende Voraussetzung beantwortet werden. Für den durch (3.32) definierten Fehler genügen allerdings die bzgl. ΔR^2 und ΔT separierten Bedingungen. Es gelte für alle Tupel (μ, ν, j, k) mit $(\nu \leq \mu,\ k \leq j)$:

$$\begin{aligned} &\text{a)} \quad \varepsilon_{\mu,\nu,j,k} \ < \ \varepsilon_{\mu,\nu-1,j,k} \\ &\text{b)} \quad \varepsilon_{\mu,\nu,j,k} \ < \ \varepsilon_{\mu,\nu,j,k-1} \ . \end{aligned} \tag{3.35}$$

Daraus folgt:

$$\varepsilon_{\mu,\nu,j,k} < \varepsilon_{\mu,\nu-1,j,k-1} \ . \tag{3.36}$$

Die Voraussetzung (3.35) ist aufgrund der Form von $\beta_{\mu,\nu}$ und $\gamma_{j,k}$ durch die hinreichende Forderung, daß die Fehlerkoeffizienten $e_{\rho,l}$ mit wachsender Ordnung nicht zu stark wachsen, gewährleistet:

Es gelte:

$$\begin{aligned} \|e_{\nu-1,0}\| &\approx \|e_{\nu,0}\| \\ \|e_{0,k-1}\| &\approx \|e_{0,k}\| \\ \|e_{\nu-1,k-1}\| &\approx \|e_{\nu,k-1}\| \approx \|e_{\nu-1,k}\| \approx \|e_{\nu,k}\| \ . \end{aligned}$$

Dann folgen die Bedingungen (3.35).

Eine Ordnung bzgl. "<" läßt sich zwischen den Fehlern von Approximation der Extrapolationsstufe $(\mu, \nu-1, j, k-1)$ bzw. zwischen $(\mu-1, \nu, j, k)$ und $(\mu, \nu, j-1, k)$ nicht angeben, es sei denn, man fordert zu starke Bedingungen an das Verhalten der Fehlerkoeffizienten. Solange der gemischte Fehler ε^{xt} als vergleichsweise klein angesehen werden kann, wird eine derartige Ordnung auch nicht benötigt.

Mit Hilfe der modifizierten Fehlerdefinition und der Voraussetzung (3.35) läßt sich nun die Art der Fehlerschätzung, wie sie sich bei 1–D–Extrapolationsverfahren inzwischen durchgesetzt hat, auch auf den 2–D–Fall übertragen. Man verwende die beste Näherung des Extrapolationstableaus z.B. $T_{\nu,\nu,k,k}$ $(\nu, k \geq 2)$ als Ersatz für die exakte Lösung in (3.24). Damit wird dann der Fehler der besten Approximation niedrigerer Ordnung (bzgl. ΔR^2 und ΔT) abgeschätzt. In Erweiterung des in Deuflhard [17] eingeführten sogenannten subdiagonalen Fehlerkriteriums, bedeutet dies, hier ein "doppelt subdiagonales" Fehlerkriterium zu verwenden, also den Fehler der Approximation $T_{\nu,\nu-1,k,k-1}$ abzuschätzen. Anstelle der nicht geeigneten Schätzung

$\bar{\varepsilon}^1_{\nu-1,\nu-1,k,k-1}$, vgl. (3.30), muß nun noch ein einfach berechenbarer und "guter" Schätzer für $\varepsilon_{\nu,\nu-1,k,k-1}$ (nach (3.32)) angegeben werden. Zur Vereinfachung der Notation wird im weiteren für das doppelt subdiagonale Index–Tupel $(\nu, \nu-1, k, k-1)$ das Index–Paar $(\nu-1, k-1)$ verwendet.

Man betrachte nun die Schätzer für (3.32):

$$\begin{aligned}
&\text{a)} \quad \bar{\varepsilon}_{\nu-1,k-1} &:=\ & \bar{\varepsilon}^x_{\nu-1,k} + \bar{\varepsilon}^t_{\nu,k-1} + \bar{\varepsilon}^{xt}_{\nu-1,k-1} \\
&\text{b)} \quad \bar{\varepsilon}^x_{\nu-1,k} &:=\ & \|T_{\nu,\nu-1,k,k} - T_{\nu,\nu,k,k}\| \\
&\text{c)} \quad \bar{\varepsilon}^t_{\nu,k-1} &:=\ & \|T_{\nu,\nu,k,k-1} - T_{\nu,\nu,k,k}\| \\
&\text{c)} \quad \bar{\varepsilon}^{xt}_{\nu,k-1} &:=\ & \|T_{\nu,\nu-1,k,k-1} - T_{\nu,\nu-1,k,k} - T_{\nu,\nu,k,k-1} + T_{\nu,\nu,k,k}\| \ .
\end{aligned} \tag{3.37}$$

Um die Güte der Schätzungen (3.37) im Vergleich zu den wahren Fehlern nach (3.32) beurteilen zu können, setze man wieder die Darstellung (3.6) in (3.37) ein. Man erhält $(\nu \geq 2, k \geq 2)$:

$$\begin{aligned}
&\text{b)} \quad \bar{\varepsilon}^x_{\nu-1,k} &=\ & \|(-1)^\nu \beta_{\nu,\nu-1} e_{\nu-1,0} \Delta R^{2\nu-2} + O(\Delta R^{2\nu}) + O(\Delta T^{N+1}) \\
&& & \quad + O(\Delta R^{2\nu-2} \Delta T^k)\| \\
&\text{c)} \quad \bar{\varepsilon}^t_{\nu,k-1} &=\ & \|(-1)^k \gamma_{k,k-1} e_{0,k-1} \Delta T^{k-1} + O(\Delta T^k) + O(\Delta R^{2M+2}) \\
&& & \quad + O(\Delta R^{2\nu} \Delta T^{k-1})\| \\
&\text{d)} \quad \bar{\varepsilon}^{xt}_{\nu-1,k-1} &=\ & \|(-1)^{\nu+k} \beta_{\nu,\nu-1} \gamma_{k,k-1} e_{\nu-1,k-1} \Delta R^{2\nu-2} \Delta T^{k-1} \\
&& & \quad + O(\Delta R^{2\nu} \Delta T^{k-1}) + O(\Delta R^{2\nu} \Delta T^k) \\
&& & \quad + O(\Delta R^{2M+2}) + O(\Delta T^{N+1})\| \ .
\end{aligned} \tag{3.38}$$

Da in diesen Schätzern für den jeweiligen dominanten Fehleranteil (bzgl. der reinen ΔR^2 bzw. ΔT bzw. gemischten Potenzen) die jeweils fremden Anteile von hoher Potenz sind, kann man also schreiben:

$$\begin{aligned}
&\text{b)} \quad \bar{\varepsilon}^x_{\nu-1,k} &\ \dot{\approx}\ & \varepsilon^x_{\nu-1,k} \\
&\text{c)} \quad \bar{\varepsilon}^t_{\nu-1,k} &\ \dot{\approx}\ & \varepsilon^t_{\nu-1,k} \\
&\text{d)} \quad \bar{\varepsilon}^{xt}_{\nu-1,k-1} &\ \dot{\approx}\ & \varepsilon^{xt}_{\nu-1,k-1}
\end{aligned} \tag{3.39}$$

und damit

$$\text{a)} \quad \bar{\varepsilon}_{\nu-1,k-1} \ \dot{\approx}\ \varepsilon_{\nu-1,k-1} \ .$$

Fehlerschätzer und führende wahre Fehler stimmen also gut überein.

Es sei hier noch kurz auf die Konsequenzen hingewiesen, die sich bei Aufgabe der Annahme (3.26) ergeben. Man erhält dann als zusätzliche gemischte

Störung in (3.38) Terme von $O(\Delta R^{2M+2}\Delta T)$ und $O(\Delta R^2 \Delta T^{N+1})$. Abhängig vom Verhältnis (ν, k) zu (M, N) müssen dann die Beziehungen (3.39) evtl. relativiert werden. Von Extremfällen abgesehen, stellen die durch (3.37) definierten Fehlerschätzer jedoch eine recht gute Approximation für die tatsächlichen Fehler (nach (3.31)) dar und können somit als Basis für eine Fehler– und Schrittweitenkontrolle dienen.

Als natürliches Konvergenzkriterium ergibt sich somit für das 2–D–Extrapolationsverfahren die Forderung:

$$\bar{\varepsilon}_{\nu-1,k-1} \leq tol \ . \tag{3.40}$$

Dabei sei *tol* die verlangte Genauigkeit für die numerische Lösung. Die Interpretation von (3.40) als absolutes oder relatives Fehlerkriterium ist eng mit der Frage nach der verwendeten Norm und ihrer Skalierung verbunden. Zu dieser, in schwierigen Anwendungen ganz entscheidenden, Problematik sei hier auf die Überlegungen in Abschnitt 5.1.1 verwiesen.

Wegen (3.37) wäre (3.40) also durch die Konvergenzforderung

$$\bar{\varepsilon}^x_{\nu-1,k} \leq \frac{tol}{3} \ \wedge \ \bar{\varepsilon}^t_{\nu,k-1} \leq \frac{tol}{3} \ \wedge \ \bar{\varepsilon}^{xt}_{\nu-1,k-1} \leq \frac{tol}{3} \tag{3.41}$$

zu realisieren. Um jedoch auch eine getrennte Vorgabe von verlangter Genauigkeit für die Orts– bzw. Zeitdiskretisierung zu erlauben, wird anstelle von (3.41) eine Konvergenzprüfung der Art

$$\bar{\varepsilon}^x_{\nu-1,k} \leq tol_x \ \wedge \ \bar{\varepsilon}^t_{\nu,k-1} \leq tol_t \tag{3.42}$$

durchgeführt, d.h. der gemischte Fehlerterm wird vernachlässigt. Aufgrund der Potenzgrade bzgl. ΔR bzw. ΔT in (3.32) bzw. (3.37) liegt es aber nahe, anzunehmen, daß für den gemischten Fehleranteil $\varepsilon^{xt}_{\nu-1,k-1}$ gilt:

$$\varepsilon^{xt}_{\nu-1,k-1} \ll \min\{\varepsilon^x_{\nu-1,k}, \varepsilon^t_{\nu,k-1}\} \ . \tag{3.43}$$

Abschließend sei noch darauf hingewiesen, daß aufgrund der in (3.33–3.36) dargestellten Fehlerbeziehungen der Fortgang der Zeitintegration mit den genaueren, aber nicht kontrollierbaren, numerischen Lösungen $T_{\nu,\nu,k,k}$, $T_{\nu,\nu-1,k,k}$ oder $T_{\nu,\nu,k,k-1}$ anstelle von $T_{\nu,\nu-1,k,k-1}$ erfolgen kann – und i.a. mit $T_{\nu,\nu-1,k,k}$ erfolgt.

3.3 Adaptive Wahl von Zeitschrittweite und Raumgitter

3.3.1 Simultane Bestimmung der Grundschrittweiten

Zur Ableitung einer Schrittweitenschätzung nehme man an, daß mit den Schrittweiten $\Delta R, \Delta T$ Konvergenz im Sinn von (3.42) für die Approximation $T_{\nu,\nu-1,k,k-1}$ erzielt wurde. Damit liegen die numerischen Werte von $\bar{\varepsilon}^x_{\nu-1,k}$, $\bar{\varepsilon}^t_{\nu,k-1}$ als Schätzer der dominanten Fehleranteile von Orts–bzw. Zeitdiskretisierung (d.h. $\varepsilon^x_{\nu-1,k-1}$, $\varepsilon^t_{\nu,k-1}$) vor. Auch die hier entwickelte Schrittweitensteuerung verwendet das Prinzip, die eigentlich optimalen Schrittweiten des gerade akzeptierten Schrittes zu bestimmen und dann im nächsten Schritt zu verwenden. Gesucht sind also die Schrittweiten $\widetilde{\Delta R}$, $\widetilde{\Delta T}$ für die gilt,

$$\begin{aligned} \varepsilon^x_{\nu-1,k} &\stackrel{!}{=} tol_x \\ \varepsilon^t_{\nu,k-1} &\stackrel{!}{=} tol_t \ . \end{aligned} \tag{3.44}$$

Einsetzen von (3.38) liefert die Bestimmungsgleichungen

$$\begin{aligned} \beta_{\nu,\nu-1}\|e_{\nu-1,0}(\tilde{x},\tilde{t})\|\widetilde{\Delta R}^{2\nu-2} &\stackrel{!}{=} tol_x \\ \gamma_{k,k-1}\|e_{0,k-1}(\tilde{x},\tilde{t})\|\widetilde{\Delta T}^{k-1} &\stackrel{!}{=} tol_t \end{aligned} \tag{3.45}$$

mit

$$\begin{aligned} \tilde{x}_i &:= \xi(\tilde{r}_i), \tilde{r}_i := (i-1)\widetilde{\Delta R}, i = 1,\ldots,n_x; n_x := \frac{1}{\widetilde{\Delta R}} + 1 \\ \tilde{\mathcal{G}}^1 &:= \{\tilde{x}_i\} \\ \tilde{t} &:= \hat{t} + \widetilde{\Delta T} \ . \end{aligned} \tag{3.46}$$

In Kombination mit (3.38) und (3.39) erhält man daraus als Schrittweitenschätzungen

$$\widetilde{\Delta R} = \sqrt[2\nu-2]{\frac{tol_x}{\bar{\varepsilon}^x_{\nu-1,k}} \cdot \frac{\|e_{\nu-1,0}(\bar{x},\bar{t})\|}{\|e_{\nu-1,0}(\tilde{x},\tilde{t})\|}} \cdot \Delta R \tag{3.47}$$

$$\widetilde{\Delta T} = \sqrt[k-1]{\frac{tol_t}{\bar{\varepsilon}^t_{\nu,k-1}} \cdot \frac{\|e_{0,k}(\bar{x},\bar{t})\|}{\|e_{0,k}(\tilde{x},\tilde{t})\|}} \cdot \Delta T \ . \tag{3.48}$$

Um berechenbare Größen zu erhalten, entwickelt man die Funktionen $e_{\nu-1,0}$ und $e_{0,k-1}$ um den Anfangspunkt des Integrationsschrittes, und mit der an-

genommenen Eigenschaft (2.116) ergibt sich:

$$\begin{aligned} e_{\nu-1,0}(\bar{x},\bar{t}) &\doteq 0 + e'_{\nu-1,0}(\bar{x},0)\Delta T \\ e_{\nu-1,0}(\tilde{x},\tilde{t}) &\doteq 0 + e'_{\nu-1,0}(\tilde{x},0)\widetilde{\Delta T} \\ e_{0,k-1}(\bar{x},\bar{t}) &\doteq 0 + e'_{0,k-1}(\bar{x},0)\Delta T \\ e_{0,k-1}(\tilde{x},\tilde{t}) &\doteq 0 + e'_{0,k-1}(\tilde{x},0)\widetilde{\Delta T}\ . \end{aligned} \tag{3.49}$$

Nimmt man nun noch an, daß die gewählte Norm das globale Verhalten bzgl. x im wesentlichen unabhängig von der Knotenwahl $\bar{x}_i$ oder $\tilde{x}_i$ widerspiegelt, d.h. es gelte

$$\|e'.,.(\tilde{x},0)\| \simeq \|e'.,.(\bar{x},0)\|\ , \tag{3.50}$$

dann ergibt sich anstelle von (3.47, 3.48):

$$\widetilde{\Delta R} = \sqrt[2\nu-2]{\frac{tol_x}{\bar{\varepsilon}^x_{\nu-1,k}} \cdot \frac{\Delta T}{\widetilde{\Delta T}}}\Delta R \tag{3.51}$$

und

$$\widetilde{\Delta T} := \sqrt[k]{\frac{tol_t}{\bar{\varepsilon}^t_{\nu,k-1}}}\Delta T\ . \tag{3.52}$$

Die Tatsache, daß das zu lösende Problem eine partielle Differentialgleichung, d.h. mit Kopplung von Raum– und Zeitableitung(en), darstellt, zeigt sich also nicht nur in einem gekoppelten Diskretisierungsfehler, sondern auch in einer gekoppelten Schrittweitensteuerung. Wegen der teilweisen Entkopplung der gegenseitigen Abhängigkeit ergibt sich jedoch eine ganz natürliche Art der Steuerung. Mit (3.52) wird die neue Zeitschrittweite gewählt, und dann ergibt sich die neue Ortsschrittweite direkt aus (3.51). Doch ist diese einfache Vorschrift für den numerischen Ablauf nicht ohne weiteres geeignet. Während ein Wechsel der Zeitschrittweite keine numerischen Probleme für das Verfahren beinhaltet (Einschrittverfahren als Zeitdiskretisierung), bringt ein Wechsel der Ortsschrittweite einen Gitterwechsel $\mathcal{G}^1 \longrightarrow \tilde{\mathcal{G}}^1$ mit sich (vgl. (3.46)). Für die Integration selbst stellt dies ebenfalls kein Problem dar (ESV als Zeitdiskretisierung), aber der Transfer der Werte von $\mathcal{G}^1$ nach $\tilde{\mathcal{G}}^1$ erzeugt Schwierigkeiten. Wenn $\tilde{\mathcal{G}}^1 \not\subseteq \mathcal{G}^1$, benötigt man dazu eine Interpolationsvorschrift und man erhält für die Werte auf $\mathcal{G}^1$ dadurch einen zusätzlichen Interpolationsfehler. Zwar sind die Startwerte für den neuen Integrationsschritt, im Gegensatz zu den theoretischen Überlegungen, sowieso mit einem "alten" Integrationsfehler behaftet, doch ist dieser zumindest kleiner als der geschätzte Approximationsfehler (Fortschreiten der Integration z.B. mit der besseren Approximation $T_{\nu,\nu-1,k,k}$). Eine Möglichkeit, dieses Problem

zu umgehen, wäre die Konstruktion einer Interpolation, die einen (schätzbaren!) Interpolationsfehler hat, der kleiner als der Diskretisierungsfehler der aktuellen Lösung ist. Mit Hinblick auf die Tatsache, daß die globale Ortsschrittweitenwahl ΔR durch eine lokale Gitterwahl ergänzt wird, kann hier ein einfacherer Weg eingeschlagen werden.

Man schränkt die Anzahl der möglichen Werte von $\widetilde{\Delta R}$ derart ein, daß sich bzgl. der Interpolation möglichst wenig Schwierigkeiten ergeben. Darüber hinaus wird nun auch die Ordnung der Ortsdiskretisierung fixiert. Wegen des schon in Kap. 3.1 angesprochenen Problems, eine genaue Approximation auf einem groben Gitter $\mathcal{G}^1$ auf die zum Extrapolieren nötigen feineren Gitter $\mathcal{G}^\rho, \rho = 2, \ldots, \nu$ zu prolongieren, wird hier gesetzt

$$\nu = \nu_{max} = \nu_{min} = 2 \; .$$

Im Gegensatz dazu wird die Ordnungs– und Schrittweitenwahl bzgl. ΔT zunächst nicht eingeschränkt, sondern es soll die effiziente Ordnungs- und Schrittweitensteuerung nach DEUFLHARD [17] so weit wie möglich übernommen bzw. ergänzt werden. Ein wesentlicher Gesichtspunkt dabei ist, daß bei drastischer Änderung des Verhaltens der Lösung $u(x,t)$ für die Zeitdiskretisierung ein Wechsel von Schrittweite und Ordnung zur Verfügung stehen soll, während für die Ortsdiskretisierung neben der eingeschränkten Wahl der Grundschrittweite im Ort noch die unten angegebene lokale Gitteranpassung zur Verfügung steht. Mit der Schrittweitenfolge (3.12) erhält man folgende Restriktion an die zu bestimmende Schrittweite $\widetilde{\Delta R}$:

$$\widetilde{\Delta R} \in \left\{ 2 \cdot \Delta R, \; \Delta R, \frac{\Delta R}{2} \right\} \; . \tag{3.53}$$

Zur Vereinfachung der Notation seien die möglichen neuen Grundschrittweiten $\widetilde{\Delta R}$ bezeichnet durch:

$$\begin{aligned} \widetilde{\Delta R}^1 &:= \Delta R \cdot 2 \\ \widetilde{\Delta R}^2 &:= \Delta R \\ \widetilde{\Delta R}^3 &:= \Delta R/2 \; . \end{aligned}$$

Damit muß nur für den Fall $\widetilde{\Delta R} = \widetilde{\Delta R}^3$ interpoliert werden. Für $\widetilde{\Delta R} = \widetilde{\Delta R}^1$ ist das alte Grobgitter das neue feine Gitter und das neue Grobgitter entsteht durch das Weglassen jeden zweiten Knotens. Die Schrittweitenschätzung (3.51) wird nun, bei gegebenen Werten von $\bar{\varepsilon}^x_{1,k}$, tol_x, ΔR, $\{\widetilde{\Delta R}^\rho\}_{\rho=1,2,3}$ und ΔT verwendet, um Schrittweitenbeschränkungen

$$\widetilde{\Delta T}^\rho_{max}, \; \rho = 1, 2, 3 \tag{3.54}$$

zu berechnen. Unter diesen Restriktionen kann nun die Wahl der optimalen Ordnungs– und Schrittweitenkombination für das Extrapolationsverfahren bzgl. der Zeitdiskretisierung erfolgen. Ohne hier auf Details einzugehen (siehe dazu wieder [17]), werden in diesem Prozeß die zu den Ordnungen $2, \ldots, k+1$ gehörenden Zeitschrittweiten

$$\begin{aligned} &\widetilde{\Delta T}_l; \quad l = 2, \ldots, k+1 \; ; \quad l \leq k_{max} \\ &\qquad k_{max} := \text{maximale Stufe der } t\text{–Extrapolation} \end{aligned} \tag{3.55}$$

berechnet (für $l \leq k$ durch (3.52), für $l = k+1$ durch eine Abschätzung, die auf der Basis Shannonscher Informationstheorie in [17] abgeleitet wurde). Werden diese Schrittweitenschätzungen unter der Restriktion (3.54) ausgeführt, so erhält man die möglichen Zeitschrittweiten

$$\{\widetilde{\Delta T}_l^{\rho}\}_{l=2,\ldots,k+1}^{\rho=1,2,3} \; , \tag{3.56}$$

für die Konvergenz der zugehörigen Approximationen $T_{2,1,l,l-1}$ auf dem durch $\widetilde{\Delta R}^{\rho}$ nach (3.46.a) definierten Gitter $\tilde{\mathcal{G}}^1 =: \tilde{\mathcal{G}}_{\rho}^1$ erwartet wird. Die Auswahl erfolgt auch hier nach dem Prinzip der Minimierung des normalisierten Aufwandes pro Integrationsschritt. Dazu wird eine Bewertung durch

$$W_l^{\rho} := \frac{\Delta T}{\widetilde{\Delta T}_l^{\rho}} \cdot A_l^{\rho}, \; \rho = 1, 2, 3 \; ; \quad l = 2, \ldots, k+1$$

eingeführt, wobei A_l^{ρ} den Aufwand mißt, der benötigt wird, um $T_{2,2,l,l}$ über dem Grundgitter $\tilde{\mathcal{G}}_{\rho}^1$ zu berechnen. Eine zunächst noch grobe, aber in den Anwendungen durchaus befriedigende Wahl der A_l^{ρ} erhält man, wenn man etwa für A_j^2 die Definition nach [17] verwendet und dann setzt:

$$\begin{aligned} A_j^1 &:= A_j^2/2 \\ A_j^3 &:= A_j^2 \cdot 2 \; . \end{aligned} \quad j = 1, 2, \ldots ,$$

Die optimale Kombination von Zeitschrittweite/Ordnung und Ortsschrittweite erhält man durch Wahl von $\widetilde{\Delta T}_{\tilde{k}}^{\tilde{\rho}}$ und $\widetilde{\Delta R}^{\tilde{\rho}}$ mit $\tilde{k}$,$\tilde{\rho}$ gegeben durch

$$W_{\tilde{k}}^{\tilde{\rho}} := \min_{\rho,l}\{W_l^{\rho}\} \; .$$

Ehe nun die lokale Gitteranpassung beschrieben wird, soll auf zwei wesentliche Aspekte eingegangen werden, die bei einem Wechsel des Ortsgitters zu beachten sind.

3.3.2 Konsequenzen aus Gitterwechsel

Es bezeichne weiterhin ΔR die Ortsgrundschrittweite des aktuellen Schrittes, in dem Konvergenz erzielt worden sei. Das zugehörige Grobgitter und das einmal verfeinerte Gitter werden im folgenden mit $\mathcal{G}^G$ bzw. $\mathcal{G}^F$ bezeichnet. Die Dimension des Grobgitters sei n_x^G (n_x^G ungerade). Mit der Schrittweitenfolge (3.12) ist die Dimension n_x^F von $\mathcal{G}^F$ durch $n_x^F = 2n_x^G - 1$ gegeben. Sei $\widetilde{\Delta R} := 1/(\tilde{n}_x^G - 1)$ die neue Ortsgrundschrittweite und $\tilde{\mathcal{G}}^G$ bzw. $\tilde{\mathcal{G}}^F$ das zugehörige Grob– bzw. Feingitter (mit Dimensionen $\tilde{n}_x^G$ und $\tilde{n}_x^F = 2\tilde{n}_x^G - 1$). Wegen der Restriktion (3.53) sind nur 3 Fälle zu unterscheiden:

$$\begin{aligned} &\text{(i)} && \tilde{n}_x^G &&= && (n_x^G + 1)/2 \\ &\text{(ii)} && \tilde{n}_x^G &&= && n_x^G \\ &\text{(iii)} && \tilde{n}_x^G &&= && 2n_x^G - 1 \ . \end{aligned}$$

Für (i) und insbesondere (iii) stellt sich die Frage, wie die Knoten der neuen Gitter zu bestimmen sind. Nur bei einer explizit gegebenen Gitterfunktion können die neuen Gitter durch Auswerten der Gitterfunktion $\xi(r)$ erzeugt werden. Seien r_i^G bzw. r_i^F die Knoten der (virtuellen) äquidistanten Gitter

$$\tilde{\mathcal{R}}^G := \left\{ r_i^G | r_i^G = (i-1)\widetilde{\Delta R},\ i = 1, \ldots, \tilde{n}_x^G \right\}$$

bzw.

$$\tilde{\mathcal{R}}^F := \left\{ r_i^F | r_i^F = (i-1)\frac{\widetilde{\Delta R}}{2},\ i = 1, \ldots, \tilde{n}_x^F \right\} \ ,$$

dann erhält man

$$\begin{aligned} &\text{a.)} \quad \tilde{\mathcal{G}}^G := \left\{ \tilde{x}_i^G | \tilde{x}_i^G = \xi(r_i^G),\ i = 1, \ldots, \tilde{n}_x^G \right\} \\ &\text{bzw.} \\ &\text{b.)} \quad \tilde{\mathcal{G}}^F := \left\{ \tilde{x}_i^F | \tilde{x}_i^F = \xi(r_i^F),\ i = 1, \ldots, \tilde{n}_x^F \right\} \ . \end{aligned} \tag{3.57}$$

In zeitabhängigen Problemen wird es nur selten eine für alle Integrationszeiten gleichbleibende optimale Gitterfunktion $\xi(r)$ geben. Vielmehr muß die Gitterfunktion adaptiv angepaßt werden. Eine Möglichkeit wäre die Wahl einer parameterabhängigen Funktion mit Anpassung des (oder der) Parameter an das zu lösende Problem. Dieser Weg kann für spezielle Problemklassen, wo spezielle Typen von Gitterfunktionen "optimal" sind (etwa $\xi(r) := r^\alpha, \alpha \geq 1$, für Probleme, die ein sehr feines Gitter nur am rechten Rand des Einheitsintervalls benötigen), sehr erfolgreich sein. Diese Möglichkeit wird hier jedoch nicht weiter verfolgt.

Auf die Konsequenzen der Anpassung einer Gitterfunktion von Schritt zu Schritt sei aber bereits an dieser Stelle hingewiesen. Die in Abschnitt 3.3.1 hergeleiteten Schrittweitenschätzer sind nur dann "gut", wenn sich die Fehlerkoeffizientenfunktionen $e_{.,.}(x,t)$ der Fehlerentwicklungen von Schritt zu Schritt nicht zu stark ändern. Da die (virtuelle) Gitterfunktion (bzw. ihre Ortsableitungen) über die lokalen Defektentwicklungen letztlich auch in die Bestimmungsgleichungen der Koeffizientenfunktionen eingehen, bringt ein Wechsel der Gitterfunktion ein zusätzliches Maß an Unsicherheit in die Schrittweitenschätzungen. Dies ist allerdings vor dem Hintergrund zu sehen, daß die Koeffizientenfunktionen bereits von den Orts– und Zeitableitungen der sich i.a. von Schritt zu Schritt stark ändernden Lösung $u(x,t)$ abhängen. Ein (möglichst glatter) Wechsel der Gitterfunktion von Schritt zu Schritt (und natürlich auch während eines Schrittes, vgl. Kapitel 4) bringt also keine neue Qualität an Unsicherheit in die Schrittweitenschätzungen.

Vor diesem Hintergrund sind die folgenden einfachen Regeln zu sehen, die anstelle von (3.57.a,b) verwendet werden.

(i) *Globale Gittervergröberung*

$$\begin{aligned} \text{a)} \quad \tilde{\mathcal{G}}^F &:= \mathcal{G}^G \\ \text{b)} \quad \tilde{\mathcal{G}}^G &:= \mathcal{G}^G \setminus \{x_i^G | i = 2,4,6,\ldots,n_x^G - 1\} \end{aligned} \tag{3.58}$$

(ii) *Beibehalten des Gitters*

$$\begin{aligned} \text{a)} \quad \tilde{\mathcal{G}}^G &:= \mathcal{G}^G \\ \text{b)} \quad \tilde{\mathcal{G}}^F &:= \mathcal{G}^F \end{aligned} \tag{3.59}$$

(iii) *Globale Gitterverfeinerung*

$$\begin{aligned} \text{a)} \quad \tilde{\mathcal{G}}^G &:= \mathcal{G}^F \\ \text{b)} \quad \tilde{\mathcal{G}}^F &:= \mathcal{G}^F \cup \{\frac{x_i^F + x_{i+1}^F}{2} | i = 1,\ldots,n_x^F - 1\} \; . \end{aligned} \tag{3.60}$$

Diese einfachen Regeln, die sich problemlos auf den Fall lokaler Gitterverfeinerung und mitbewegter Gitter verallgemeinern lassen, führen in der Praxis zu einem deutlich besseren Verhalten des Gesamtalgorithmus als Varianten, bei denen man versucht, die virtuelle Gitterfunktion mittels einer (streng monotonen!) Interpolationsvorschrift höherer Ordnung – etwa die im folgenden Abschnitt beschriebene – tatsächlich zu approximieren und zur Gitterknotenbestimmung zu verwenden.

3.3.3 Das Interpolationsproblem

Wahl der Interpolationsvorschrift

Von wesentlicher Bedeutung für den Gesamtalgorithmus ist die Wahl einer geeigneten Interpolationsvorschrift für die Lösungsapproximationen $U_{j,i}$. Dabei bezeichne $U_{j,i}$ die j–te Komponente der Approximation $T_{2,1,k,k} \in \mathbb{R}^{n_{pde}}$ am Knoten x_i des Feingitters $\mathcal{G}^F$ (k sei die aktuelle t–Extrapolationsstufe).

Bei der Interpolationsvorschrift für die Lösung ist zu beachten, daß bei globaler Gitterverfeinerung (und/oder lokaler Gitterverfeinerung, siehe Abschnitt 3.3.4) zur Fortsetzung der Integration Anfangswerte an den neuen Knoten benötigt werden. Für die (komponentenweise) Interpolation der Lösung $U_{j,i}$, und somit globalen Darstellung bzgl. x, wird eine monotone Interpolation verwendet. Obwohl hier die Monotonie weder eine notwendige noch hinreichende Bedingung für das möglichst ungestörte Fortschreiten der Integration ist, verhindert diese Art der Interpolation das häufig beobachtbare Überschwingen z.B. einer klassischen Spline–Interpolation bei steilen Ortsgradienten in der Lösung und vergleichsweise groben Gittern.

Abgesehen von dem i.a. recht großen Interpolationsfehler derartiger Überschwinger (trotz formal hoher Approximationsordnung), kann der interpolierte Wert dadurch sogar in physikalisch unsinnigen Bereichen liegen, was zu Schwierigkeiten bei der numerischen Auswertung der Problemfunktionen $\mathbf{f}^{\Delta}, \mathbf{B}^{\Delta}$ in (2.107) führen kann. Ohne eine monotone Interpolationsvorschrift wären z.B. für die Simulation der komplexen Modelle von Flammenausbreitungsphänomenen in Maas/Warnatz[53] ungleich feinere Gitter nötig gewesen [52].

Aus der Vielzahl von monotonen Interpolationsvorschriften (siehe etwa Rasch /Williamson [68] für einen Überblick und groben qualitativen Vergleich) wurde für die hier vorgestellte Anwendung die monotone, stückweise kubische Hermite–Interpolation nach Fritsch/Carlson [33], Fritsch/Butland [31] gewählt. Neben gewissen strukturellen Vorteilen der Methode, wie einfache Form, Unabhängigkeit von der Verarbeitungsrichtung (aufsteigende oder absteigende Indizes), Lokalität von Störungen, Abarbeitung in einem Durchlauf und vergleichsweise wenig Rechenaufwand, existiert dazu ein ausgetestetes und robust implementiertes Softwarepaket (PCHIP) von Fritsch/Butland [32].

Zur Erzeugung und Auswertung der Interpolierenden wird diese Software zwar als Black–Box–Programm verwendet, aber zur Steuerung des Gesamtalgorithmus wird zumindest eine grobe Fehlerschätzung für die interpolierten Werte der Lösung benötigt. Daher wird im folgenden der Interpolationsalgorithmus kurz dargestellt, ein einfacher Fehlerschätzer vorgestellt und die Konsequenzen für die Gitterkontrolle diskutiert.

Monotone Hermite-Interpolation

Sei $x_1 < \cdots < x_n$ ein gegebenes Gitter ($n \geq 2$) und sei $f(x)$ die zu interpolierende Funktion mit den Stützwerten f_i an den Knoten x_i. Ziel ist es, ein auf dem Intervall $I := [x_1, x_n]$ stückweise kubisches Interpolationspolynom $P(x) \in C^1[I]$ zu konstruieren, das für monotone Daten eine monotone Funktion auf I ist. Dazu wird $P(x)$ in jedem Intervall $\Delta_i := x_{i+1} - x_i$ als ein Polynom mit folgender Darstellung angesetzt:

$$P(x) = f_i H_1(x) + f_{i+1} H_2(x) + d_i H_3(x) + d_{i+1} H_4(x) \ , \tag{3.61}$$

wobei $d_j = P'(x_j), j = i, i+1$, und $H_k(x)$ die kubischen Hermite-Basisfunktionen für das Intervall Δ_i sind; d.h.:

$$\begin{array}{ll} H_1(x) = \phi(\frac{x_{i+1}-x}{\Delta_i}) & H_2(x) = \phi(\frac{x-x_i}{\Delta_i}) \\ H_3(x) = -\Delta_i \psi(\frac{x_{i+1}-x}{\Delta_i}) & H_4(x) = \Delta_i \psi(\frac{x-x_i}{\Delta_i}) \end{array} \tag{3.62}$$

mit

$$\phi(y) = 3y^3 - 2y^2 \ \ , \ \psi(y) = y^3 - y^2 \ . \tag{3.63}$$

Setzt man $d_i := f'(x_i)$ oder verwendet man Standardapproximationen zur Berechnung der d_i, etwa die in (2.16) oder (2.17) angegebenen, so erhält man eine stückweise kubische Interpolierende, die jedoch nicht notwendigerweise in jedem Intervall Δ_i monoton ist. In [33] werden notwendige und hinreichende Bedingungen für die Wahl der Werte d_i hergeleitet, unter denen das Polynom (3.61) monotone Daten auch monoton interpoliert. In [31] werden Approximationsvorschriften für d_i untersucht, die alle eine monotone Interpolation garantieren. Sehr befriedigende Resultate zeigt eine von BRODLIE [12] vorgeschlagene Approximation, die auch die Nichtuniformität eines Gitters berücksichtigt. Bezeichne S_i^L, S_i^R die links–bzw. rechtsseitige Sekante am Knoten x_i,

$$\begin{gathered} S_i^L := \frac{f_i - f_{i-1}}{\Delta_{i-1}} \ \ , \ S_i^R := \frac{f_{i+1} - f_i}{\Delta_i} \ , \\ i = 2, ..., n-1 \ , \end{gathered} \tag{3.64}$$

dann sind die Approximationen an die 1.Ableitung an den inneren Knoten gemäß

$$\begin{gathered} d_i := \begin{cases} \dfrac{S_i^L S_i^R}{\alpha S_i^R + (1-\alpha) S_i^L} & , \text{wenn } S_i^L S_i^R > 0 \ , \\ 0 & , \text{sonst} \ , \end{cases} \\ i = 2, ..., n-1 \end{gathered} \tag{3.65}$$

mit

$$\alpha = \frac{1}{3}(1 + \frac{\Delta_i}{\Delta_{i-1} + \Delta_i}) \tag{3.66}$$

zu wählen. Auf äquidistantem Gitter reduziert sich diese Approximation auf das harmonische Mittel:

$$d_i := \begin{cases} \dfrac{2S_i^L S_i^R}{S_i^L + S_i^R} & \text{, wenn } S_i^L S_i^R > 0 \text{ ,} \\ 0 & \text{, sonst.} \end{cases} \tag{3.67}$$

An den Rändern wird zur Bestimmung von d_1 bzw. d_n eine klassische einseitige Differenzenapproximation über 3 Gitterpunkte durchgeführt. Dabei wird auch dieser Wert gegebenenfalls modifiziert, um Monotonie zu erzwingen.

Bei der Interpolation der Lösung ist nun die Frage nach der Approximationsgenauigkeit von großer Bedeutung.

Interpolationsfehler

Obwohl der Gesamtalgorithmus nicht das Ziel hat, auch den Fehler der örtlich globalen Darstellung (mittels obiger Interpolationsvorschrift) zu kontrollieren, soll auf das Fehlerverhalten kurz eingegangen werden, da an durch Regridding evtl. neu erzeugten Knoten die Anfangswerte für den nächsten Zeitintegrationschritt mit diesem Interpolationsfehler behaftet sind. Dazu müssen zwei Fälle betrachtet werden. Zum einen die Interpolation in einem Intervall $I_i := [x_i, x_{i+1}]$, für das die Vorzeichenbedingungen

$$S_i^L S_i^R > 0 \text{ und } S_{i+1}^L S_{i+1}^R > 0 \tag{3.68}$$

erfüllt sind, und zum anderen eine Interpolation bei der zumindest eine der beiden Bedingungen in (3.68) verletzt ist. Um die folgende Darstellung nicht unnötig zu komplizieren, wird nun ein äquidistantes Gitter mit Maschenweite Δ betrachtet. Bei exakten Daten f_i und $d_i = f'(x_i)$ ist der Interpolationsfehler von (3.61) $O(\Delta^4)$. Ersetzt man die exakte Ableitung durch die Approximation (3.65), so besitzt diese für den ersten der oben dargestellten Fälle einen Diskretisierungsfehler $O(\Delta^2)$. Bezeichne nun $P(x)$ das zu den exakten Daten gehörende Interpolationspolynom und $\tilde{P}(x)$ das Polynom mit approximierter 1. Ableitung, so gilt (für $x \in I_i$):

$$\begin{aligned} f(x) - \tilde{P}(x) &= (f(x) - P(x)) + (P(x) - \tilde{P}(x)) \\ &\doteq O(\Delta^4) + O(\Delta^2)[H_3(x) - H_4(x)] \\ &\doteq O(\Delta^3) \text{ .} \end{aligned} \tag{3.69}$$

Diese Approximationsordnung ist im Vergleich zu den nicht exakten Daten $U_{j,i}$ zu sehen, deren Fehler durch die Ortsdiskretisierung der PDG ja proportional zu Δ^2 sind. Selbst unter Vernachlässigung der Tatsache, daß dadurch die Stützwerte f_i keine exakten Daten mehr darstellen, kann nicht gefolgert

werden, daß zumindest der zusätzliche Interpolationsfehler (3.69) klein im Vergleich zu dem Diskretisierungsfehler der PDG ist. Da die Fehlerkoeffizienten in den verschiedenen Fehlertermen von völlig unterschiedlicher Natur sind, kann gerade für die bei praktischen Rechnungen verwendeten Schrittweiten Δ der Interpolationsfehler, trotz formal höherer Ordnung, durchaus größer sein als der Diskretisierungsfehler bei Integration der PDG.

Für den zweiten der oben genannten Fälle führt die Erzwingung der Monotonie überdies zu einem Ordnungsverlust in der Interpolationsvorschrift. O.B.d.A. sei angenommen, daß die Vorzeichenbedingung (3.68) am linken Rand von I_i verletzt ist, d.h. $d_i = 0$. Sei das durch (3.61) und(3.65) definierte Interpolationspolynom mit $\hat{P}(x)$ bezeichnet. Dann gilt (für $x \in I_i$):

$$\begin{aligned} f(x) - \hat{P}(x) &= (f(x) - \tilde{P}(x)) + (\tilde{P}(x) - \hat{P}(x)) \\ &\doteq O(\Delta^3) + d_i H_3(x) \\ &\doteq d_i \frac{(x_{i+1} - x)^2 (x - x_i)}{\Delta^2} = O(\Delta) \ . \end{aligned} \tag{3.70}$$

Der Interpolationsfehler ist also linear in der Schrittweite Δ. Im Intervall I_i ist der führende Fehlerterm maximal für den Punkt $x = x_i + \frac{1}{3}\Delta$ und minimal für $x = x_{i+1}$.

Sind sogar beide Vorzeichenbedingungen (3.68) verletzt, so erhält man i.a. wiederum einen Interpolationsfehler mit linearem Fehlerterm. Bezeichne $\bar{P}(x)$ dieses Polynom, so gilt

$$\begin{aligned} f(x) - \bar{P}(x) &= (f(x) - \tilde{P}(x)) + (\tilde{P}(x) - \bar{P}(x)) \\ &\doteq d_i H_3(x) + d_{i+1} H_4(x) \\ &\doteq d_i \frac{(x_{i+1} - x)^2 (x - x_i)}{\Delta^2} - d_{i+1} \frac{(x_{i+1} - x)(x - x_i)^2}{\Delta^2} \\ &= O(\Delta) \ . \end{aligned} \tag{3.71}$$

Entwickelt man d_i und d_{i+1} um $x_M := (x_{i+1} - x_i)/2$ so zeigt sich, daß für den Fehler am Intervallmittelpunkt gilt:

$$f(x_M) - \bar{P}(x_M) \doteq \Delta^2 \ .$$

Fehlerschätzer und algorithmische Konsequenzen

Um auf große Interpolationsfehler algorithmisch reagieren zu können, wird ein Fehlerschätzer benötigt, der zumindest Fehler anzeigt, die deutlich größer als der vorhandene Diskretisierungsfehler sind. Einen vergleichsweise billigen Fehlerschätzer für den zusätzlichen Interpolationsfehler erhält man wie

folgt: man erzeugt (getrennt für jede Komponente) ein Interpolationspolynom $U_P(x)$ durch Interpolation der numerischen Lösung $U(x)$ über dem Grobgitter $\mathcal{G}^G$, dabei werden allerdings als Stützwerte die Lösungswerte U_i der Feingitterlösung verwendet. Diese Darstellung wird auf den restlichen Knoten $\bar{x}_i$ des Feingitters, also $\bar{x}_i \in \bar{\mathcal{G}} := \mathcal{G}^F \setminus \mathcal{G}^G$, ausgewertet und dann mit den dort vorhandenen nicht interpolierten Werten verglichen. Mit einer geeigneten Norm, die z.B. auch den Fehler aller Lösungskomponenten berücksichtigt, erhält man lokale Schätzer

$$\bar{\varepsilon}_P(\bar{x}_i) := ||U_P(\bar{x}_i) - U(\bar{x}_i)|| \ , \ x_i \in \bar{\mathcal{G}} \tag{3.72}$$

für den wahren (punktweisen) Interpolationsfehler

$$\varepsilon_P(\bar{x}_i) := ||u_P(\bar{x}_i) - u(\bar{x}_i)|| \ , \ x_i \in \bar{\mathcal{G}} \ ,$$

der bei der Interpolation der exakten Lösung $u(x)$ über dem Gitter $\mathcal{G}^G$ entstehen würde. Durch Wahl einer geeigneten örtlich globalen Norm erhält man einen Schätzer $\bar{\varepsilon}_P^{glo}$ für den wahren globalen Fehler

$$\varepsilon_P^{glo} := ||u_P(x) - u(x)||.$$

Zur Anfangswerterzeugung wird allerdings die Feingitterlösung auf dem feinen Gitter $\mathcal{G}^F$ interpoliert und (fast) immer an den Intervallmittelpunkten des feinen Gitters ausgewertet. Je nach Art, wie die Vorzeichenbedingung (3.68) erfüllt bzw. verletzt ist, muß die Schätzung (3.72) also um einen Faktor $\frac{1}{8}, \frac{1}{4}$ oder $\frac{1}{2}$ korrigiert werden. Es zeigt sich jedoch, daß die Verwendung eines konstanten Faktors $\frac{1}{4}$ einer genauen Fallunterscheidung, wann welche Fehler wo zu erwarten sind, zumindest gleichwertig ist.

Diese Schätzer werden allerdings nur bei der unten beschriebenen lokalen Gittererneuerung mitverwendet, um durch Einfügen neuer Knoten den nach dem nächsten Integrationschritt evtl. eingeschleppten Interpolationsfehler klein zu halten.

Der lokale Gittererneuerungsalgorithmus basiert nun auf Fehlerschätzungen, die auf den Grobgitterknoten gegeben sind. Also ist ein Fehlerschätzer für den Interpolationsfehler an den Grobgitterknoten $x_i \in \mathcal{G}^G$ anzugeben. Dies wird über eine einfache Mittelung erreicht. Der im Algorithmus letztendlich verwendete Fehlerschätzer ist durch

$$\hat{\varepsilon}_p(x_i) := \frac{1}{4} \frac{\bar{\varepsilon}_P(\bar{x}_i) + \bar{\varepsilon}_P(\bar{x}_{i+1})}{2} \tag{3.73}$$

gegeben. Numerische Experimente belegen, daß diese Schätzer zumindest die Größenordnung der wahren Fehler erfassen.

Durch den in (3.73) durchgeführten Mittelungsprozeß werden große lokale Fehler verschmiert, was ein Verfeinern mehrerer benachbarter Intervalle zur

Folge haben kann. Dies führt dazu, daß nach dem nächsten Zeitschritt auch bei wandernden steilen Fronten ein adäquates Interpolationsgitter vorliegt.

Der bei der aktuell durchzuführenden Interpolation eingeschleppte Fehler wird akzeptiert. Normalerweise werden die an einigen wenigen Knoten erzeugten zusätzlichen Fehler durch den parabolischen Charakter des Problems rasch gedämpft. Schlimmstenfalls wird die Zeitschrittweite im folgenden Integrationsschritt reduziert oder die nächste Fehlerschätzung (für den Zeit– und/oder Ortsdiskretisierungfehler) kann etwas zu pessimistisch ausfallen. Eine Wiederholung des bereits akzeptierten Schrittes wäre dagegen zu aufwendig und die Wahl einer geeigneten Zeitschrittweite und/oder eines modifizierten Gitters durchaus problematisch.

3.3.4 Lokale Gitteranpassung

Die in Abschnitt 3.3.1 beschriebene globale Gittersteuerung (vollständige Verfeinerung oder Vergröberung) muß in der Regel um eine lokale Gitteranpassung ergänzt werden. Damit soll erreicht werden, daß in kritischen Bereichen der Lösung möglichst kleine Gitterabstände vorliegen, während in unkritschen Bereichen mit vergleichsweise großen Schrittweiten Δx_i gearbeitet werden kann. Oder anders ausgedrückt: Es soll durch lokales Einfügen von Knoten eine dem Problem möglichst gut angepaßte (virtuelle) Gitterfunktion $\xi(r)$ erzeugt werden. "Möglichst gut" heißt in diesem Zusammenhang auch, daß bei einer Integration auf möglichst wenig Knoten ein (Orts-) Diskretisierungsfehler erzielt wird, der viel kleiner als die vorgegebene Genauigkeit ist.

Das verwendete Anpassungsprinzip ist dabei vergleichsweise einfach. Es soll der lokal an jedem Knoten auftretende Diskretisierungsfehler gleichverteilt werden. Dazu werden also lokale Fehlerschätzer $\bar{\varepsilon}_i^x$, d.h. Schätzer für den wahren durch Ortsdiskretisierung hervorgerufenen Fehler an jedem Knoten x_i, benötigt. Derartige lokale Fehlerschätzer lassen sich analog zu dem durch (3.37) definierten örtlich globalen Schätzer definieren:

$$\bar{\varepsilon}_i^x := \|U^F(x_i) - U^E(x_i)\| \ , \ x_i \in \mathcal{G}^G \ , \ i = 1, ..., n_x^G \ . \tag{3.74}$$

Es wird also wiederum die Differenz zwischen der auf dem feinen Ortsgitter berechnenten numerischen Lösung U^F und der orts–extrapolierten Approximation U^E (jeweils vollständig t–extrapoliert) verwendet. Da die r–extrapolierte Lösung nur auf dem Grobgitter $\mathcal{G}^G$ vorhanden ist, stellt (3.74) eine Fehlerschätzung für die Feingitterlösung an den Grobgitterknoten $x_i \in \mathcal{G}^G$ dar.

Gleichverteilung der lokalen Fehler(schätzer) heißt nun, daß ein neues Gitter $\tilde{\mathcal{G}}^G$ gesucht wird, auf dem für die zugehörigen Fehlerschätzungen $\tilde{\bar{\varepsilon}}_i^x$ gilt:

$$\tilde{\bar{\varepsilon}}_i^x \approx \text{const. für alle } x_i \in \tilde{\mathcal{G}}^G \ . \tag{3.75}$$

Damit die Äquilibrierung der lokalen Fehler ein sinnvolles Prinzip darstellt, muß allerdings gewährleistet sein, daß die verwendeten Normen in (3.37) bzw. (3.74) zumindest in folgendem Sinne zueinander passen. Sind alle lokalen Fehler kleiner als die vorgegebene Toleranz tol_x, so muß auch der globale Fehlerschätzer diese Eigenschaft haben, d.h. es soll gelten:

$$\bar{\varepsilon}_i^x \leq tol_x \text{ für alle } x_i \in \mathcal{G}^G \Rightarrow \bar{\varepsilon}^x \leq tol_x \ .$$

Zur Vereinfachung der Notation bezeichne dabei $\bar{\varepsilon}^x$ den durch (3.37) definierten globalen Fehlerschätzer.

Das Ziel der Äquilibrierung soll nun durch einen sehr einfachen Mechanismus erreicht werden. In Bereichen, wo der Fehler "groß " ist, sollen Knoten eingefügt und in Bereichen mit "kleinen" Fehlern Knoten eliminiert werden. Damit dies sinnvoll ist, muß davon ausgegangen werden, daß auch bei nur lokalem Einfügen von Knoten die Diskretisierungsfehler (sowohl $\bar{\varepsilon}_i^x$ als auch $\bar{\varepsilon}^x$) reduziert werden. Dies ist sicher eine vertretbare Annahme, allerdings kann – im Gegensatz zum lokalen Ortsdefekt $\kappa^{\mathbf{\Delta R}}$ – kein Verhalten

$$\varepsilon_i^x \to \frac{\varepsilon_i^x}{4} \ , \text{ wenn } \Delta x_i, \Delta x_{i-1} \to \frac{\Delta x_i}{2}, \frac{\Delta x_{i-1}}{2} \tag{3.76}$$

erwartet werden. Denn die Fehler ε_i^x hängen nicht nur vom lokalen Defekt $\kappa_i^{\mathbf{\Delta R}}$ sondern vom Gesamtvektor $\kappa^{\mathbf{\Delta R}}$ ab. Ist dabei der Einfluß von $\kappa_i^{\mathbf{\Delta R}}$ dominant, so wird sich das Verhalten (3.76) in etwa einstellen, aber als Basis für ein algorithmisches Konzept kann (3.76) nicht dienen.

Da hier der geschätzte globale Diskretisierungsfehler eines Schrittes äquilibriert werden soll, sind klassische Gleichverteilungstechniken, siehe etwa [62, 79, 70], nicht anwendbar. Bei ihrer Anwendung wird davon ausgegangen, daß ein Fehlerindikator ϵ, etwa durch eine Monitorfunktion der Art

$$\begin{aligned} \epsilon_i = \ & \omega_1 \frac{\partial u_i}{\partial x} + \omega_2 \frac{\partial^2 u_i}{\partial x^2} \\ & \omega_1, \, \omega_2 \text{ zu wählende Parameter} \ , \end{aligned}$$

gegeben ist. Diese Werte ändern sich bei Umverteilung der Knoten praktisch nicht, während der hier verwendete Fehlerschätzer ε_i^x sich bei anderer Knotenwahl völlig ändern kann (und soll).

Wie im letzten Abschnitt dargestellt, soll der geschätzte Interpolationsfehler (3.73) auch zur Steuerung des lokalen Verfeinerns herangezogen werden. Dazu werden die lokalen Schätzer des Diskretisierungsfehlers wie folgt korrigiert:

$$\tilde{\varepsilon}_i^x := \max \{\bar{\varepsilon}_i^x, \hat{\varepsilon}_p(x_i)\} \ .$$

Nun müssen noch die Begriffe "groß " und "klein" quantifiziert werden. Sehr gute Erfahrungen wurden mit folgenden Schwellwerten für das Einfügen (σ^+) bzw. Löschen (σ^-) von Knoten gemacht:

$$\begin{aligned} \sigma^+ &:= \tilde{\varepsilon}^x_{max} \,, && \text{wenn } \tilde{\varepsilon}^x_{max} \leq tol_x \\ \sigma^+ &:= \sqrt{\tilde{\varepsilon}^x_{max} \cdot tol_x} \,, && \text{wenn } \tilde{\varepsilon}^x_{max} > tol_x \\ \sigma^- &:= \tfrac{1}{6} tol_x \end{aligned}$$

mit

$$\tilde{\varepsilon}^x_{max} := \max_i \{\tilde{\varepsilon}^x_i; x_i \in \mathcal{G}^G\} \,.$$

Die allgemeinen Gitteranpassungsregeln sind nun sehr einfach:

- wenn $\tilde{\varepsilon}^x_i < \sigma^-$: Der Knoten x_i kann eliminiert werden.
- wenn $\tilde{\varepsilon}^x_i \geq \sigma^-$: Der Knoten x_i bleibt bestehen.
- wenn $\tilde{\varepsilon}^x_i > \sigma^+$: Unterteile Δ_{i-1} und Δ_i.

Unter welchen Bedingungen ein eliminierbarer Knoten auch tatsächlich gelöscht wird, ist, ebenso wie weitere Restriktionen, weiter unten angegeben.

Im Prinzip könnte man nun daran denken, die lokale Anpassung mehrmals hintereinander auszuführen und damit letztlich einen Adaptierungsprozeß im Sinne der von Babuška/Rheinboldt [4] vorgeschlagenen allgemeinen Strategie zu realisieren. Hier sprechen allerdings zwei Gründe gegen ein mehrmaliges Verfeinern. Es wird dann eine Fehlerschätzung für die Lösung auf dem bereits mindestens einmal angepaßten Gitter benötigt. Die Annahme eines Verhaltens gemäß (3.76) läßt aber globale Effekte außer acht und kann daher nicht verwendet werden. Darüber hinaus müßten an allen neuen Knoten Anfangswerte für den nächsten Zeitschritt durch Interpolation gewonnen werden. Gerade an stark zu verfeinernden Intervallen kann aber auch der Interpolationsfehler groß sein. Während Störungen an einigen wenigen Knoten beim Weiterintegrieren meist problemlos verkraftetet werden, kann bei einer zu großen Zahl von gestörten Anfangswerten die Zeitintegration erheblich (negativ) beeinflußt werden. Die einzige Alternative wäre eine Neuintegration des bereits akzeptierten Zeitschrittes auf dem modifizierten Gitter. Dies stellt jedoch einen unverhältnismäßig hohen Zusatzaufwand dar.

Da die lokale Anpassung ohnehin nach jedem Integrationsschritt erfolgt, kann davon ausgegangen werden, daß bereits nach einigen wenigen Schritten das Ausgangsgitter eine nicht allzu schlechte Knotenverteilung aufweisen dürfte. Ein einmaliges Durchführen des lokalen Regriddings wird also i.a. ausreichen.

Wie bereits oben angedeutet, läuft der lokale Gitteranpassungsprozeß unter Einhaltung gewisser Nebenbedingungen ab:

- die neue Knotenzahl $\tilde{n}_x$ soll in einem vorgegebenen Bereich liegen, d.h.

$$n_x^{min} \leq \tilde{n}_x \leq n_x^{max} \ .$$

  Typische Werte: $n_x^{min} = 21$, $n_x^{max} \leq 1281$ (i.w. speicherplatzbedingter Wert)

- Die neue Knotenzahl soll ungerade sein.

- Es werden die Knoten mit den kleinsten lokalen Fehlern zuerst eliminiert.

- Es wird kein Knoten eliminiert, der Randpunkt eines zu verfeinernden Intervalls ist.

- Es wird kein Knoten eliminiert, wenn dadurch der zu erwartende Fehler an einem der beiden Nachbarknoten größer als der aktuelle Einfügeschwellwert wird.

- Es wird kein Knoten eliminiert, wenn dadurch das Verhältnis benachbarter (neuer) Intervalle außerhalb eines vorgeschriebenen Bereichs $[1/q^{\Delta}, q^{\Delta}]$ liegt.

- Es wird durch Einfügen neuer Knoten erreicht, daß das Verhältnis benachbarter (neuer) Intervalle innerhalb eines vorgeschriebenen Bereichs $[1/\hat{q}^{\Delta}, \hat{q}^{\Delta}]$ liegt. Die Werte $q^{\Delta} = 3.0$, $\hat{q}^{\Delta} = 2.2$ haben sich bewährt.

Durch die beiden letzten Bedingungen wird eine Quasiuniformität der Gitter gewährleistet. Durch die Verwendung zweier Regeln mit unterschiedlichen Schwellwerten wird erreicht, daß die Elimination von Knoten zunächst nicht zu stark verhindert wird, andererseit aber nicht zu viele neue Knoten an Stellen eingefügt werden, wo vorher gerade eliminiert wurde (Interpolationsfehler!).

In Abbildung 3.6 ist ein lokaler Gittererneuerungsprozeß exemplarisch dargestellt. Da der lokale Fehlerschätzer ε_3^x größer als die Einfügegrenze σ^+ ist, werden die angrenzenden Intervalle unterteilt. Die Fehlerschätzer an den Knoten 6, 7 und 8 sind kleiner als die Löschgrenze σ^- und sind daher prinzipiell eliminierbar. Da keine benachbarten Knoten gleichzeitig gelöscht werden dürfen, bleibt der 7. Knoten bestehen. Wäre der Fehler am 7. Knoten minimal gewesen, so wäre nur dieser Knoten eliminiert worden. Da der Fehler der Knotens am rechten Rand auch noch größer als σ^+ ist, wird schließlich auch noch das letzte Intervall unterteilt. Nun ist allerdings die 2. Quasiuniformitätsbedingung verletzt, und deshalb wird noch der 9. Knoten des neuen Gitters eingefügt.

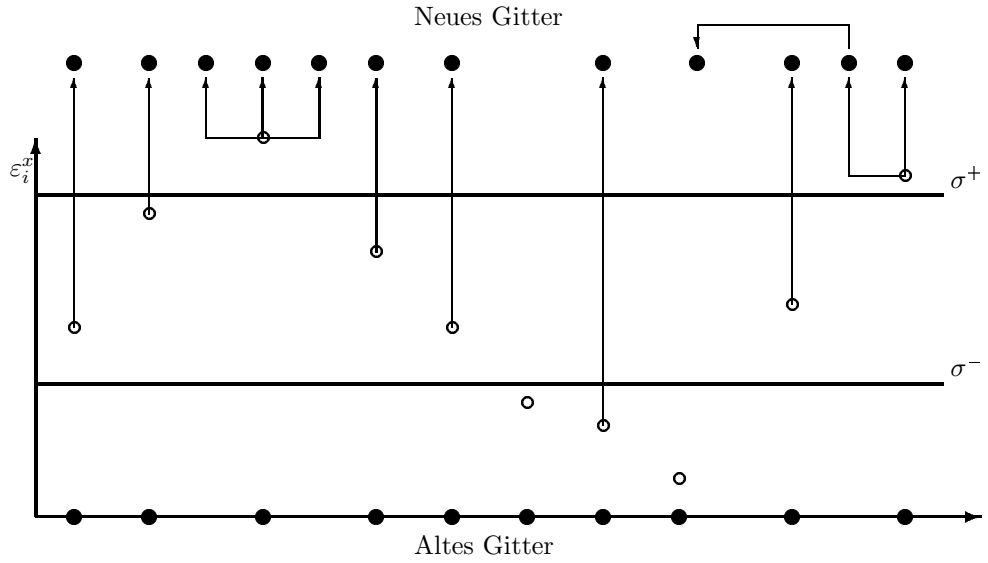


Abbildung 3.6: Lokale Gitteranpassung mit lokalen Fehlerschätzern

Abschließend muß noch darauf eingegangen werden, daß die beiden Gitteranpassungsstrategien (globale Anpassung, vgl. Abschnitt 3.3.1, und die eben beschriebene lokale Anpassung) natürlich nicht unabhängig voneinander durchgeführt werden können.

Da ein mehrmaliges Verfeinern eines Gitterintervalls zu Interpolationsproblemen führen kann, wird eine globale Gitterverfeinerung verhindert, wenn die Anzahl der neu eingefügten Knoten einen gewissen Schwellwert überschreitet. Ebenso wird keine globale Gittervergröberung ausgeführt, wenn die Anzahl der eliminierten Knoten zu groß ist. Schließlich wird noch die Schätzung des Ortsdiskretisierungsfehlers, die ja für das noch nicht lokal angepasste Gitter gilt, modifiziert. Bezeichne $\tilde{n}_x$ die neue Grobgitterknotenzahl nach lokalem Regridding, so hat sich die folgende, eher heuristisch anzusehende, Regel gut bewährt:

$$\bar{\varepsilon}^x \to \bar{\varepsilon}^x \left(\frac{n_x}{\tilde{n}_x} \right)^{3/2} . \tag{3.77}$$

3.4 Gesamtalgorithmus

Der Gesamtablauf eines Zeitintegrationsschrittes des Gesamtverfahrens ist in Abbildung (3.7) schematisch dargestellt. Von einer Feingitterlösung $U^F(x,t)$ ausgehend, wird zunächst die vollständige Zeitintegration des semi–diskreten Systems auf dem Grobgitter durchgeführt. Dies bedeutet, daß während dieser Integration die aus dem reinen Zeitintegrationsverfahren herstammenden

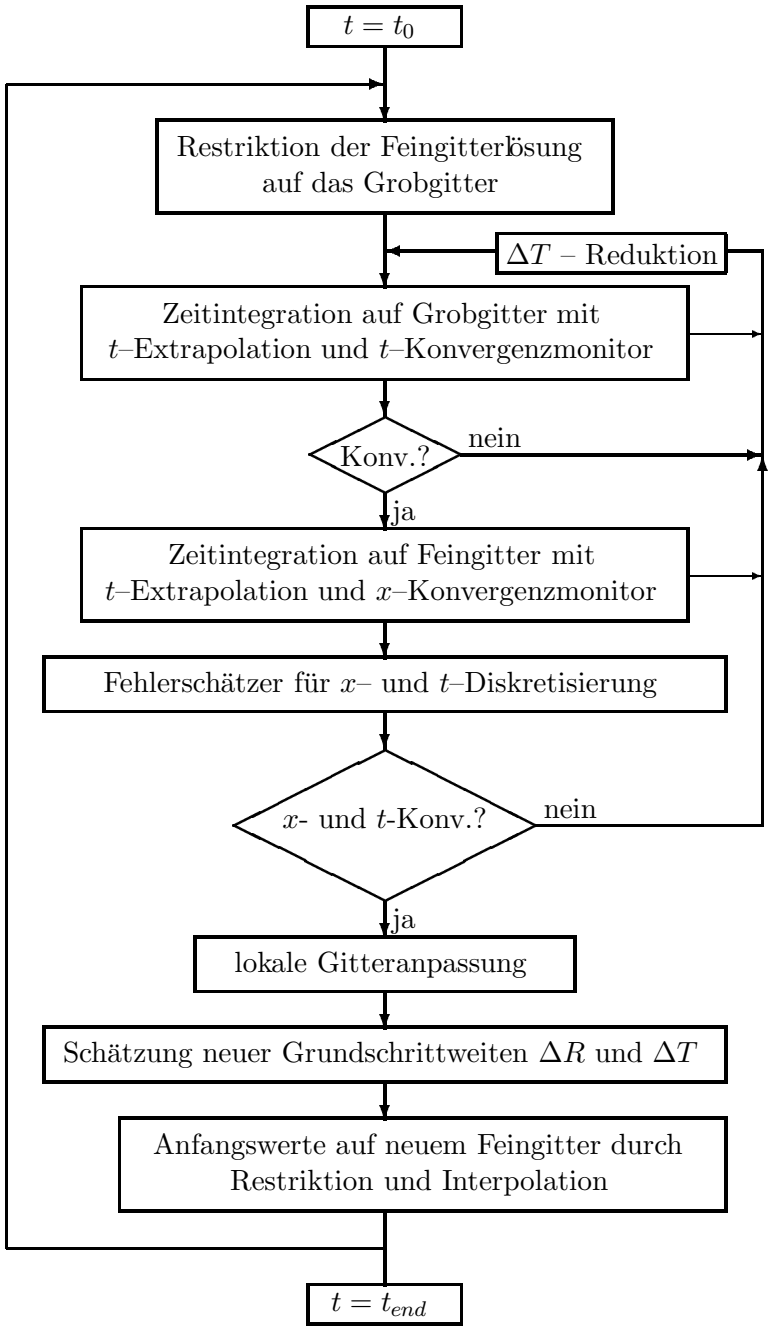


Abbildung 3.7: Ablauf eines Integrationsschrittes

Kontrollmechanismen aktiv sind. So kann die für diesen Schritt als optimal geschätzte Ordnung $\tilde{k}$ noch innerhalb des Schrittes, abhängig vom Konvergenzverhalten, variieren (innerhalb des sog. Ordnungsfensters $[\tilde{k}-1, \tilde{k}+1]$). Wenn sich keine Konvergenz einstellt, bzw. dies nicht mehr zu erwarten ist, wird die Grundschrittweite ΔT reduziert. Insbesondere wegen letzterer

Möglichkeit wird mit der weniger aufwendigen Integration auf dem Grobgitter begonnen.

Ist der erste Zeitschritt akzeptiert (z.B. mit Ordnung k), d.h. für den dabei schätzbaren Fehler,

$$\epsilon^t_{1,k-1} := ||T_{1,1,k,k-1} - T_{1,1,k,k}|| \ , \tag{3.78}$$

gilt

$$\epsilon^t_{1,k-1} \leq tol_t \ ,$$

so wird die Zeitintegration auf dem Feingitter durchgeführt. Dabei wird nun keine simultane t–Konvergenzkontrolle durchgeführt, sondern in jedem Fall $T_{2,1,k,k}$ berechnet. Analog zu (3.78) kann dabei auch eine Fehlerschätzung $\epsilon^t_{2,k-1}$ berechnet werden.

Es stimmen aber weder $\epsilon^t_{1,k-1}$ noch $\epsilon^t_{2,k-1}$ mit dem eigentlich zu schätzenden Fehler $\bar{\varepsilon}^t_{2,k-1}$ überein, da sie noch gemischte Fehleranteile

$$C\Delta R^2 \Delta T^{k-1} \ \text{ bzw. } \ C\frac{\Delta R^2}{4}\Delta T^{k-1}$$

enthalten. Erst wenn aus $T_{1,1,k,k}$ und $T_{2,1,k,k}$ die r–extrapolierte Approximation $T_{2,2,k,k}$ berechnet ist, kann $\bar{\varepsilon}^t_{2,k-1}$ bestimmt werden. Da der gemischte Fehlerterm i.a. klein ist, ist die durch ihn hervorgerufene Differenz der Fehlerschätzer meist ebenfalls klein. Nur relativ selten (typischerweise bei zu groben Gittern) wird also einer der Fälle

$$\epsilon^t_{1,k-1} \leq tol_t \ \wedge \ \bar{\varepsilon}^t_{2,k-1} > tol_t$$

oder

$$\epsilon^t_{1,k-1} \leq tol_t \ \wedge \ \epsilon^t_{2,k-1} > tol_t$$

eintreten, die eine Reduktion der Zeitschrittweite bedingen.

Mit den nun zur Verfügung stehenden Approximationen kann der Schätzer des Ortsdiskretisierungsfehlers, $\bar{\varepsilon}^x_{1,k}$, berechnet und somit der Konvergenztest (3.42) durchgeführt werden.

Um einen zu groß werdenden Ortsdiskretisierungsfehler ebenfalls "früh", d.h. bei niedriger t–Extrapolationsstufe, festzustellen, müßte man ein komplizierteres Muster der Hintereinanderausführung von t bzw. r–Extrapolation realisieren. Dies ist jedoch aus Gründen der Implementierung ungünstg. Man müßte z.B die Jacobimatrizen der Fein– und Grobgitterintegration gleichzeitig speichern. Statt dessen wird, sobald $T_{2,1,2,2}$ vorliegt, getestet, ob x–Konvergenz zu erwarten ist. Entweder mit der Konsequenz eines Abbruchs des aktuellen Schrittes oder mit der Konsequenz, daß auf jeden Fall $T_{2,1,k,k}$ berechnet wird.

Ist der Konvergenztest (3.42) erfolgreich, so wird zunächt die lokale Gitteranpassung durchgeführt. Nach Anpassung der Fehlerschätzung gemäß (3.77) und gegebenenfalls unter Beachtung der in Abschnitt 3.3.4 angegebenen Restriktionen, werden die neuen Grundschrittweiten bestimmt. Allerdings wird dazu als Zeitdiskretisierungsfehler nicht der Wert $\bar{\varepsilon}^t_{2,k-1}$, sondern

$$\tilde{\varepsilon}^t_2 := \max\{\epsilon^t_{1,k-1}, \epsilon^t_{2,k-1}, \bar{\varepsilon}^t_{2,k-1}\} \tag{3.79}$$

verwendet. Dadurch wird die simultane t–Konvergenzkontrolle, gerade in Fällen wo die Werte der drei Schätzer variieren, sicherer. Dies ist insbesondere bei globalem Gitterwechsel (altes feines Gitter ist neues Grobgitter bzw. altes Grobgitter wird zum neuen Feingitter) ein wichtiges algorithmisches Glättungsinstrument.

Während bei zu großem Zeitfehler die Zeitschrittweitenreduktion die natürliche Konsequenz ist, kann bei zu großem Ortsfehler die Ortsschrittweite nicht so einfach reduziert werden. Ein sich dadurch reduzierender Ortsfehler ist nur zu erwarten, wenn für die Startwerte des Integrationsschrittes eine im Ort globale Darstellung vorliegt, die überdies auch lokal, d.h. an jedem Punkt x des Gebietes Ω einen Fehler besitzt, der kleiner als die vorgegebene Genauigkeit ist. Dies ist hier nicht der Fall. Selbst wenn zur Ortsdiskretisierung ein Ansatz mit globaler Darstellung gewählt wird, ist diese Forderung nur zu erfüllen, wenn zur Fehlerschätzung eine (nicht differenzierbare!) Maximumsnorm verwendet wird. Daher wird hier anstelle einer Reduktion der Ortsschrittweite, auch bei zu großem Ortsfehler, die Zeitschrittweite reduziert. Gilt die Eigenschaft (3.49), so wird sich dadurch auch der Ortsfehler reduzieren.

Damit ist auch die Grenze der Anwendung des Verfahrens klargelegt. Ist eine der zu lösenden Gleichungen zeitunabhängig, so ist das zugehörige semi–diskrete Problem ein differentiell–algebraisches. Eine erfolgreiche Zeitintegration setzt konsistente Startwerte voraus. Durch die Interpolation werden aber kleine Störungen produziert. Abhängig vom lokalen Charakter des Problems, werden diese Störungen gedämpft oder ungedämpft weiterintegriert, was bei erfolglosem Konvergenztest zur erfolglosen Schrittweitenreduktion führen kann. Dies ist unabhängig davon, welcher Diskretisierungsfehler zur Nichtkonvergenz führt.

Das grundsätzliche Problem der Beschaffung konsistenter Anfangswerte bei der Lösung von differentiell–algebraischen Systemen vom Index 1 tritt natürlich schon bei Beginn der Integration auf. Um konsistente Startwerte zu erhalten, ist die Zeitintegration eines kleinen Zeitschrittes mit einer voll impliziten Euler–Diskretisierung (mit einem gewöhnlichen Newton–Verfahren zur Lösung des nichtlinearen Gleichungssystems) ein probates Mittel. Zwar gibt es elegantere Möglichkeiten zur Berechnung konsistenter Anfangswerte, siehe etwa Leimkuhler/Petzold/Gear [47], aber im hier betrachteten Kontext

ist die implizite Euler–Variante (mit Extrapolation und Fehlerschätzung) das natürliche Vorgehen.

Wenn die Integration eines Zeitschrittes wegen erfolgloser Schrittweitenreduktion abgebrochen wird, kann, nach einem Zwischenschritt mit dieser impliziten Euler–Diskretisierung, mit dem Standardverfahren meist erfolgreich weiterintegriert werden [30].

Ist der Abbruch durch einen nicht reduzierbaren Ortsdiskretisierungsfehler bedingt, kann man auch einen alternativen Weg gehen. Ein Konsistenzproblem zeigt sich typischerweise, wenn die Schätzer $\epsilon^t_{1,k-1}, \epsilon^t_{2,k-1}, \bar{\varepsilon}^t_{2,k-1}$ deutlich differieren. Verzichtet man in einem derartigen Fall zumindest auf den x–Konvergenztest, so stellt sich häufig bereits im nächsten Schritt wieder x–Konvergenz ein. Der Grund liegt darin, daß für algebraische Komponenten mehrere Schritte der linear–impliziten Euler–Diskretisierung die Konvergenzeigenschaften des Newton–Verfahrens eines voll–impliziten Eulerschrittes haben.

Auch bei einem zeitweiligen Wegfall des x–Konvergenztests, wird sowohl die globale als auch die lokale Gitteranpassung durchgeführt. Zudem wird zur Gittersteuerung ein interner Sicherheitsfaktor verwendet, der auf Abweichungen des tatsächlichen Fehlerverhaltens vom erwarteten Fehlerverhalten reagiert. Damit hat der partielle Verzicht auf den x–Konvergenztest (der als optionale Variante des Verfahrens realisiert ist) noch zu keinem beobachtbaren Robustheitsverlust des Gesamtverfahrens geführt.

Ein interessantes Phänomen taucht auf, wenn der stationäre Zustand eines Problems erreicht wird. Zur Vereinfachung sei angenommen, daß keine Gitteranpassung mehr erfolgt. Bei der Integration auf dem Feingitter wird dann der geschätzte Zeitfehler praktisch verschwinden, da die Lösung nicht mehr variiert. Die Integration auf dem Grobgitter startet jedoch auf der restringierten Feingitterlösung. Die zum Grobgitter gehörende stationäre Lösung ist (abhängig von der Güte der Ortsdiskretisierung) verschieden von der stationären Feingitterlösung. Dieser Unterschied zeigt sich in jedem Integrationsschritt auf dem Grobgitter als "Scheindynamik" und führt zu nicht verschwindenden Fehlerschätzungen $\epsilon^t_{1,k-1}, k = 2, \ldots$ und einem deutlich gestörten Ortsfehlerschätzer $\bar{\varepsilon}^x_{1,k}$. Wirkliche Probleme ergeben sich dadurch allerdings nur, wenn $tol_t \ll tol_x$ vorgegeben wird. Wegen (3.79) ist allerdings die Vergrößerung der Zeitschrittweiten bei Erreichen eines stationären Zustands nicht so schnell möglich, wie bei einer Zeitintegration ohne Ortsfehlerkontrolle.

Im Vergleich zu einer Linienmethode ohne Ortsfehlerkontrolle (mit gut gewähltem Gitter!), besteht der Mehraufwand des hier entwickelten Verfahrens also nicht nur in der zusätzlichen Integration auf dem Grobgitter. Auch die Zeitschrittweiten können kleiner werden, was zu einem insgesamt weiter gesteiger-

ten Aufwand führt. Da eine Nichtkonvergenz der Zeitdiskretisierung bereits auf dem Grobgitter festgestellt werden kann, relativiert sich jedoch dieser simple Vergleich. Letztendlich ist ein derartiger Vergleich kaum zu führen, denn wie soll die durch Ortsfehlerkontrolle gewonnene Sicherheit bewertet werden ?

4. Mitbewegtes Gitter

Für gewisse Typen von parabolischen Systemen kann die Effizienz und Robustheit des numerischen Lösungsverfahrens durch die Verwendung von zeitlich mitbewegten Gittern deutlich gesteigert werden. Dabei ist es von eher untergeordneter Bedeutung, welche Art der Ortsdiskretisierung vorgenommen wird. Dagegen läßt das Motiv, aufgrund dessen eine Mitbewegung erfolgen soll, eine charakterisierende Einteilung der zahlreichen in der Literatur beschriebenen Methoden in zwei Kategorien zu.

Bei der einen erfolgt die Gitterbewegung, um die Werte einer Monitorfunktion, die meist den Ortsdiskretisierungsfehler beschreiben soll, gleichzuverteilen – siehe etwa [2, 16, 26, 80]. Häufig soll dadurch eine statische Gittererneuerung überflüssig gemacht werden, die die Zeitintegration doch erheblich stören kann. Man denke nur an die Problematik der eingeschleppten Interpolationsfehler an neu einzufügenden Knoten und an die Tatsache, daß sich die Dimension des semi–diskreten Systems praktisch von Schritt zu Schritt ändert. Letzteres ist insbesondere für eine Zeitintegration mit einem Mehrschrittverfahren von großer Bedeutung. Bei zu häufigem Neustart wird die Effizienz eines BDF-Integrators, etwa DASSL, drastisch reduziert.

Bei der anderen Kategorie von Methoden wird das Gitter mitbewegt, um in den neuen Koordinaten eine insgesamt geringere Dynamik – und somit größere Zeitschrittweiten – zu erhalten. Siehe hierzu etwa [39, 42, 43, 55, 56, 66, 67].

Es soll hier nicht näher auf die prinzipiellen Unterschiede, Gemeinsamkeiten, Vor–und Nachteile dieser beiden Ansätze eingegangen werden. Ein Überblick über einige wesentlichen Aspekte, ebenso wie einige numerische Vergleichsrechnungen finden sich z.B. in Furzeland/Verwer/Zegeling [34].

Da der hier vorgestellte Algorithmus bereits eine statische Gitteranpassung besitzt, ist es ein ganz natürliches Vorgehen, das mitbewegte Gitter mit dem (Haupt–)Ziel einzuführen, dadurch größere Zeitschrittweiten verwenden zu können. Damit soll eine Effizienzsteigerung insbesondere für die Problemklassen erreicht werden, bei denen wandernde Fronten ein Charakteristikum der Lösung sind.

Ein derartiger Lösungsverlauf ist in Abbildung 4.1 skizziert. Hier wandert eine Front $u(x, t_1)$ (durchgezogene Linie) mit konstanter Geschwindigkeit von rechts nach links $u(x, t_2)\,,\ t_2 > t_1$ (gepunktete Linie). Bei der Integration auf einem festen Gitter ändern sich die Werte an allen in der Front liegenden Knoten. Das Problem besitzt eine hohe Dynamik und es werden sich kleine Zeitschrittweiten einstellen. Wird die Gleichung dagegen auf einem mitbewegten Gitter integriert, wie es etwa in Abbildung 4.1 rechts unten angegeben ist, so ändern sich die Werte an den Knoten in der Front nicht. Die

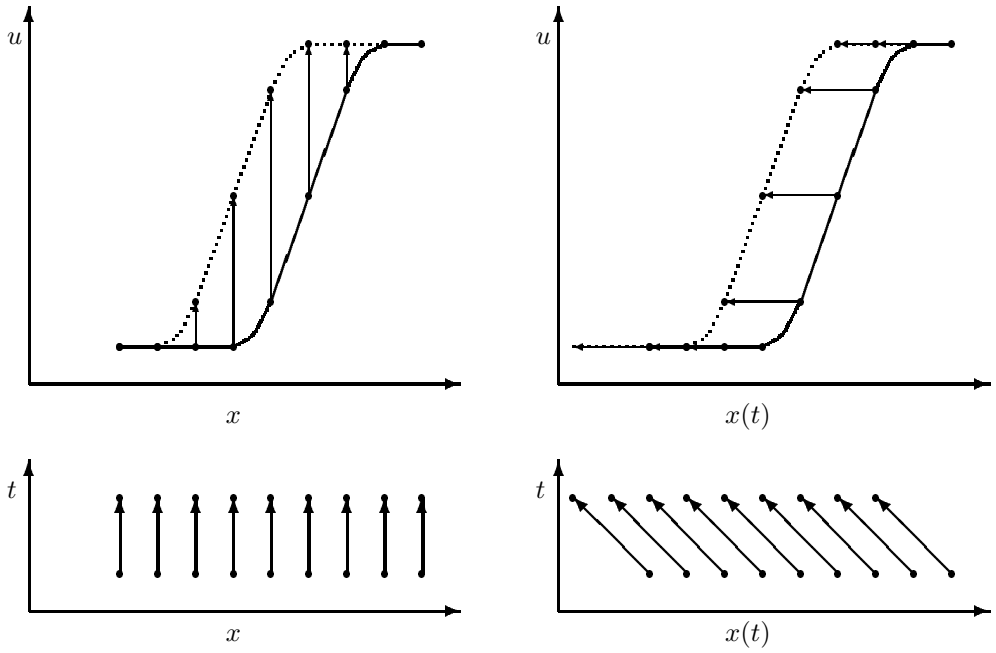


Abbildung 4.1: Wandernde Front über festem und mitbewegten Gitter

Dynamik des Gesamtsystems (transformierte PDG und zusätzliche Gittergleichung) wird nun von der Gitterbewegung dominiert. Können dabei alle Knoten mit der gleichen Geschwindigkeit wandern, so hat sich die Gesamtdynamik des Problems deutlich reduziert

Jedoch wird i.a. eine derart günstige Wahl nicht möglich sein, da das mitbewegte Gitter z.B. an den Rand des Integrationsgebietes stößt. Auf die daraus resultierenden Konsequenzen wird weiter unten ausführlich eingegangen. Zunächst sollen die transformierte Differentialgleichung und einige Gitterbewegungsgleichungen hergeleitet werden.

4.1 Transformation der Gleichung

4.1.1 Ein Modellproblem

Lagrangesche Form der Gleichung

Zur Illustration wird zunächst das Modellproblem einer skalaren, expliziten, parabolischen Gleichung betrachtet:

$$u_t = f(u, u_x, u_{xx}) \ . \tag{4.1}$$

Wird zur Ortsdiskretisierung ein Gitter verwendet, das innerhalb eines Schrittes des Zeitdiskretisierungsverfahrens nicht mehr fest, sondern zeitabhängig ist, d.h. $x_i \to x_i(t)$, so spricht man von einem mitbewegten Gitter (Moving Grid). Die Integration schreitet also längs der Trajektorien $x_i(t)$ fort. Für die kontinuierliche Gleichung (4.1) heißt dies, daß die totale Zeitableitung $\dot{u}$ nicht mehr mit der partiellen Zeitableitung u_t identisch ist. Vielmehr gilt für die Lösung $u(x(t), t)$ auf einer mitbewegten Koordinate $x(t)$ die Beziehung

$$\dot{u} = u_x \dot{x} + u_t \ . \tag{4.2}$$

Auflösen nach u_t und einsetzen in (4.1) liefert die transformierte Gleichung (Lagrangesche Form von (4.1))

$$\dot{u} - u_x \dot{x} = f(u, u_x, u_{xx}) \ . \tag{4.3}$$

Allgemeine Koordinatentransformation

Diese Gleichung ergibt sich auch, wenn man eine spezielle Variablensubstitution durchführt. Seien (s, τ) zwei neue unabhängige Variable, die durch die Gleichungen

$$x = \phi(s, \tau) \ , \ t = \tau$$

mit den alten Koordinaten (x, t) verbunden seien. Dann hängen die partiellen Ableitungen u_t, u_x, u_{xx} mit den partiellen Ableitungen u_τ, u_s, u_{ss} wie folgt zusammen. Aus

$$u_s = u_x \phi_s \ , \ u_\tau = u_x \phi_\tau + u_t$$

folgt unmittelbar (sei $\phi_s \neq 0$)

$$u_x = u_s \phi_s^{-1} \ , \ u_t = u_\tau - u_x \phi_\tau .$$

Für die 2. Ortsableitung ergibt sich daraus

$$u_{xx} = (u_x)_x = (u_s \phi_s^{-1})_s \phi_s^{-1} \ . \tag{4.4}$$

Ausdifferenzieren ergibt

$$u_{xx} = u_{ss} \phi_s^{-2} - u_s \phi_s^{-3} \phi_{ss} \ . \tag{4.5}$$

Einsetzen in (4.1) liefert das Modellproblem in den Koordinaten (s, τ):

$$u_\tau - u_s \phi_s^{-1} \phi_\tau = f(u, u_s \phi_s^{-1}, (u_s \phi_s^{-1})_s \phi_s^{-1}) \ . \tag{4.6}$$

Ist nun ϕ derart, daß für jeden festen Zeitpunkt $\hat{t}$ die Koordinaten s und x identisch sind, so gilt

$$\phi_s = 1\,,\ \phi_{ss} = 0.$$

Man erhält also wiederum (4.3), wenn man bei der Ortsdiskretisierung der vollständig transformierten Gleichung (4.6) die Knoten $s_i = x_i(t)$ verwendet. Obwohl dies das tatsächliche Vorgehen ist, ist es durchaus interessant, das folgende Gedankenexperiment durchzuführen. Man kann die Koordinate s mit der in (2.26) eingeführten virtuellen Koordinate r identifizieren, bzw. die Knoten s_i mit dem virtuellen Gitter $\mathcal{R}$, und die Funktion $\phi(s,t)$ als zeitabhängige Erweiterung der Gitterfunktion $\xi(r)$ – vgl. (2.25). Führt man nun eine Ortsdiskretisierung der Gleichung (4.6) auf einem äquidistanten Gitter $\mathcal{S}$ mit Knoten $s_i = (i-1)\cdot\Delta s\,,\ \Delta s = 1/(n_x-1)$ durch, so erhält man als semi–diskrete Gleichung $(x_i(t) = \phi(s_i,t))$:

$$\begin{aligned}
&u_{i,\tau} - \frac{u_{i+1}-u_{i-1}}{s_{i+1}-s_{i-1}}\left(\frac{x_{i+1}-x_{i-1}}{s_{i+1}-s_{i-1}}\right)^{-1}\phi_{i,\tau}\\
&= f(u_i,\\
&\tfrac{u_{i+1}-u_{i-1}}{s_{i+1}-s_{i-1}}\left(\tfrac{x_{i+1}-x_{i-1}}{s_{i+1}-s_{i-1}}\right)^{-1},\\
&\frac{\frac{u_{i+1}-u_i}{s_{i+1}-s_i}\left(\frac{x_{i+1}-x_i}{s_{i+1}-s_i}\right)^{-1} - \frac{u_i-u_{i-1}}{s_i-s_{i-1}}\left(\frac{x_i-x_{i-1}}{s_i-s_{i-1}}\right)^{-1}}{(s_{i+1}-s_{i-1})/2}\left(\frac{x_i-x_{i-1}}{s_i-s_{i-1}}\right)^{-1})\,,
\end{aligned}$$

also

$$\begin{aligned}
&u_{i,\tau} - \tfrac{u_{i+1}-u_{i-1}}{x_{i+1}-x_{i-1}}\phi_{i,\tau}\\
&= f(u_i, \tfrac{u_{i+1}-u_{i-1}}{x_{i+1}-x_{i-1}}, \tfrac{2(u_{i+1}(x_i-x_{i-1})-u_i(x_{i+1}-x_{i-1})+u_{i-1}(x_{i+1}-x_i))}{(x_{i+1}-x_i)(x_i-x_{i-1})(x_{i+1}-x_{i-1})})\,.
\end{aligned}$$

Diese Diskretisierung ist nun identisch mit derjenigen, die man erhält, wenn man das Problem (4.3) auf dem nichtuniformen Gitter mit Knoten $x_i(t)$ diskretisiert und dabei die einfache Approximation (2.16) für die 1. Ortsableitung verwendet. Diese Identität ergibt sich nicht, wenn anstelle von (4.4) der ausdifferenzierte Ausdruck (4.5) (mit den Approximationen (2.16, 2.21)) diskretisiert wird.

4.1.2 Der allgemeine Fall

Für den allgemeinen Fall eines linear–impliziten Systems von partiellen Differentialgleichungen, wie es in (2.1–2.6) als die hier behandelte Problemklasse angegeben ist, bedeutet der Übergang zu einem mitbewegten Gitter, daß anstelle von (2.4), d.h.

$$B(u)u_t = f(u)\,, \tag{4.7}$$

nun das System

$$B(u)(\dot{u} - u_x\dot{x}) = f(u) \tag{4.8}$$

zu lösen ist. Dabei soll die vereinfachte Schreibweise $B(u), f(u)$ die in (2.4) angegebenen Abhängigkeiten von Ortsableitungstermen sowie von x und t implizieren. Damit dieses System rein formal überhaupt lösbar ist, fehlt noch eine Gleichung zur Bestimmung von $x(t)$, bzw. es fehlen in der ortsdiskretisierten Form von (4.8) noch n_x Gleichungen zur Bestimmung der neuen abhängigen Variablen $\boldsymbol{x}(t) := (x_1(t), ..., x_{n_x}(t))^T$.

Auf die Frage nach der Wahl geeigneter Gleichungen zur Bestimmung der neuen Unbekannten wird in den nächsten Abschnitten detailliert eingegangen. Grundsätzlich kann man jedoch zwei Möglichkeiten unterscheiden. Entweder bestimmt man $x(t)$ durch simultanes Lösen eines erweiterten Systems von Gleichungen

$$\begin{aligned} B(u)(\dot{u} - u_x\dot{x}) &= f(u) \\ h(x, \dot{x}, u, \dot{u}) &= 0 \ , \end{aligned} \tag{4.9}$$

wobei die Funktion h noch zu spezifizieren ist, oder man führt die Wahl von $x(t)$ a priori durch. Dies muß nicht heißen, daß die Wahl vor dem Beginn der Gesamtintegration erfolgen muß, aber zumindest vor jedem Zeitintegrationsschritt müssen für den gesamten folgenden Schritt die Gitterlinien $x_i(t)$ fest vorgegeben werden.

4.1.3 A priori Bestimmung des Gitters

Üblicherweise wird dazu ein in der Zeit konstanter Ansatz verwendet, d.h. $\dot{x}(t) := v$. Vor jedem Zeitschritt werden nun zuerst geeignete Geschwindigkeiten v_i bestimmt und dann der Integrationsschritt auf dem gegebenen Gitter $x_i(t) = x_i(\bar{t}_0) + (t - \bar{t}_0) \cdot v_i$ durchgeführt. Diese sog. sequentielle Technik, wie sie z.B. von Davis/Flaherty [16] und Smooke/Koszykowski [74] verwendet wurde, hat zwei klare Vorteile gegenüber der simultanen Lösung von (4.9).

Die Dimension des zu lösenden Problems wächst nicht und, falls das ursprüngliche Problem ein explizites System von parabolischen Gleichungen war ($B = I$), so wird der Typ der Gleichung nicht geändert, da in diesem Fall ein System

$$\dot{u} = f(u) + v(x)u_x =: \tilde{f}(u)$$

zu lösen ist, wobei $v(x)$ eine gegebene Funktion ist.

Diesen Vorteilen stehen allerdings auch Nachteile gegenüber. Nach erfolgter Ortsdiskretisierung bedeutet die Berechnung der Knotenwerte $x_i(t)$ zwar praktisch keinen zusätzlichen Aufwand, aber es ist darauf zu achten, daß das

Gitter $\mathcal{G}$ für den gesamten Integrationschritt der Konsistenzbedingung

$$x_{i-1}(t) < x_i(t) < x_{i+1}(t) \text{ für alle } x_i \in \dot{\mathcal{G}} \tag{4.10}$$

genügt. In anderen Worten, es dürfen keine Überschneidungen von Gitterlinien auftreten. Dies läßt sich bei einer a priori Wahl von $x_i(t)$ leicht realisieren. Eine gute (automatische) Anpassung, selbst wenn die optimale Geschwindigkeit linear in t ist, ist jedoch meist recht schwierig zu erreichen. So hängt z.B. die optimale Geschwindigkeit für das zu integrierende semi–diskrete System von dem dann auftretenden Ortsdiskretisierungsfehler ab, vgl. Abschnitt 4.2.5. Ist die optimale Gitterbewegung darüber hinaus noch nichtlinear, so bedeutet das Hinterherhinken in der Adaptierung, daß die Integrationsschrittweiten oft kaum größer sind als im Fall eines festen Gitters.

Da hier die Bestimmung des mitbewegten Gitters derart erfolgen soll, daß (für bestimmte Problemklassen) die Zeitschrittweiten deutlich größer werden können, wird deshalb der Weg der simultanen Lösung von (4.9) beschritten.Somit muß noch eine geeignete Wahl von $h(x, \dot{x}, u, \dot{u})$ in (4.9) erfolgen.

4.2 Gleichungen zur Gitterankopplung

4.2.1 Minimierung eines Zielfunktionals

Ein häufig verwendeter Ansatz, vgl. etwa Hyman [42], Miller/Miller [56], Petzold [66], um das gesteckte Ziel größerer Zeitschrittweiten zu erreichen, kann wie folgt formuliert werden. Wähle $\dot{x}$ derart, daß die Zeitableitung des Gesamtsystems

$$\dot{w} := (\dot{u}, \dot{x})^T$$

in den neuen Koordinaten minimal wird, d.h.

$$||\dot{w}||^2 \stackrel{!}{=} \min \ . \tag{4.11}$$

Wählt man

$$||\dot{w}||^2 := \dot{w}^T \dot{w} \ ,$$

so erhält man als zu minimierendes Funktional:

$$\dot{u}^T \dot{u} + \dot{x}^T \dot{x} \stackrel{!}{=} \min \ . \tag{4.12}$$

Da die Beziehung (4.2), d.h

$$\dot{u} = u_t + u_x \dot{x} \ , \tag{4.13}$$

auch für die allgemeine Problemklasse (4.7) Gültigkeit hat, erhält man aus (4.12), unter Einführen eines skalaren Parameters α,

$$\min_{\dot{x}} \{\dot{u}^T \dot{u} + \alpha \dot{x}^2\} = \min_{\dot{x}} \{(u_t + u_x \dot{x})^T (u_t + u_x \dot{x}) + \alpha \dot{x}^2\} \ .$$

Als Lösung dieses einfachen Minimierungsproblems ergibt sich die Bedingungsgleichung zur Bestimmung von $x(t)$:

$$(u_t + u_x\dot{x})^T u_x + \alpha\dot{x} = 0 \ .$$

Zusammen mit (4.13) erhält man daraus die in [66] hergeleitete Gitterbewegungsgleichung

$$\dot{u}^T u_x + \alpha\dot{x} = 0 \ . \tag{4.14}$$

Das zu lösende Gesamtsystem hat also wieder eine linear–implizite Struktur:

$$\tilde{B}(w)\dot{w} = \tilde{f}(w) \tag{4.15}$$

mit

$$\begin{aligned} \tilde{B} &= \begin{pmatrix} B & Bu_x \\ u_x^T & \alpha \end{pmatrix} \\ \tilde{f} &= \begin{pmatrix} f(u) \\ 0 \end{pmatrix} \ . \end{aligned}$$

Für den häufigen Spezialfall $B = I$ reduziert sich (4.15) zur (immer noch linear–impliziten) Form

$$\begin{aligned} \dot{u} - u_x\dot{x} &= f(u) \\ \dot{u}^T u_x + \alpha\dot{x} &= 0 \ . \end{aligned} \tag{4.16}$$

Auflösen der ersten Gleichung nach $\dot{u}$ und Einsetzen in die zweite liefert die halbexplizite Form

$$\begin{aligned} \dot{u} - u_x\dot{x} &= f(u) \\ \dot{x} &= -f(u)^T u_x/(\alpha + u_x^T u_x) \ . \end{aligned} \tag{4.17}$$

Setzt man jetzt noch die zweite Gleichung in die erste ein, so ergibt sich die rein explizite Form von (4.16), vgl. etwa [42]:

$$\begin{aligned} \dot{u} &= f(u) - u_x(f(u)^T u_x/(\alpha + u_x^T u_x)) \\ \dot{x} &= -f(u)^T u_x/(\alpha + u_x^T u_x) \ . \end{aligned} \tag{4.18}$$

Obwohl es sich um algebraisch äquivalente Formen handelt, hängt es nun vom Zeitdiskretisierungsverfahren ab, ob die Äquivalenz auch nach Zeitdiskretisierung erhalten bleibt und welche Form für die numerische Integration die günstigste ist. In der oben angegebenen Form benötigen alle Formulierungen, daß $\alpha \neq 0$ gilt, damit auch für $u_x = 0$ eine Bestimmung von $x(t)$ möglich ist.

In [66, 67] wurden die obigen drei Formulierungen miteinander verglichen und der Form (4.16) der Vorzug gegeben. Dabei wurde zur Zeitintegration

das Mehrschrittverfahren DASSL verwendet und schon allein deshalb sind die gemachten Erfahrungen nicht unmittelbar übertragbar. Darüber hinaus ist zu bemerken, daß mit dem System (4.15), egal in welcher speziellen Formulierung, noch kein befriedigendes Verhalten des mitbewegten Gitters zu erreichen ist. Dies liegt i.w. daran, daß dem bereits angesprochenen Problem der Gitterlinienüberkreuzung noch keine Aufmerksamkeit geschenkt wurde.

4.2.2 Das Problem der Knotenüberkreuzung

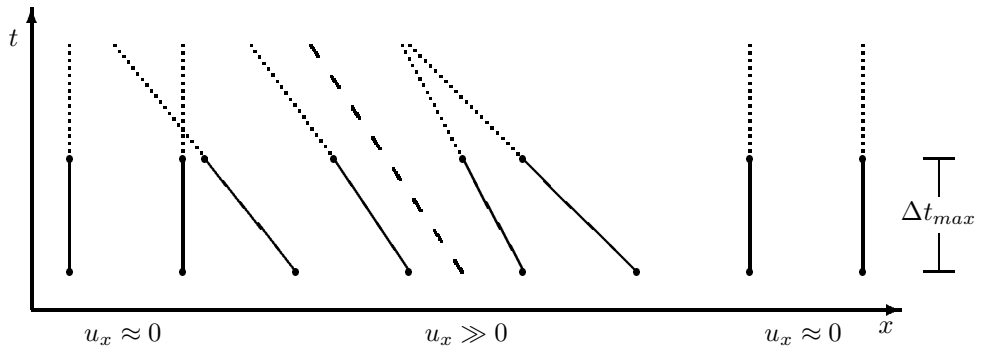


Abbildung 4.2: Knotenfluß wenn "α klein " in (4.18)

Der typische Knotenfluß, der sich bei Verwendung der obigen Gittergleichungen innerhalb eines Integrationsschrittes einstellt, ist in den Abbildungen 4.2 bzw. 4.3 illustriert. Dabei wird angenommen, daß es sich bei dem zu integrierenden Problem um eine Gleichung handelt, deren Lösung sich so, wie in Abbildung 4.1 dargestellt, verhält. Während also für Knoten, die sich innerhalb der Front befinden, der Gradient $u_x(x,t)$ groß ist, wird er vor und hinter der Front rasch verschwinden. Die dann z.B. aus (4.18) mit kleinen α–Werten gewonnene Knotenbewegung ist exemplarisch in Abbildung 4.2 dargestellt. Vor und hinter der Front bewegen sich die Knoten nicht, während sich innerhalb der Front sehr ähnliche Geschwindigkeiten einstellen. Der maximale Geschwindigkeitsunterschied zwischen zwei Nachbarknoten ist dann typischerweise an den Rändern der Front zu finden. Während das "Aufreißen" der Knoten hinter der Front i.a. "nur" zu Ortsdiskretisierungproblemen führt, so bedingt das Zusammenlaufen der Knoten vor der Front eine Zeitschrittweitenbeschränkung. Sei zur Vereinfachung angenommen, daß die Knotengeschwindigkeiten $\dot{x}_i$ vor dem Integrationsschritt aus (4.18) gewonnen werden. Dann führt die Konsistenzbedingung (4.10) zu folgender Schrittweitenbeschränkung:

$$x_i + \dot{x}_i \Delta t \overset{!}{<} x_{i+1} + \dot{x}_{i+1} \Delta t \;\text{ und }\; \dot{x}_{i+1} < \dot{x}_i \;\; \Rightarrow$$

$$\Delta t_{max} < \frac{x_{i+1} - x_i}{\dot{x}_i - \dot{x}_{i+1}} \; . \tag{4.19}$$

Die Differenz der Geschwindigkeit aufeinanderzulaufender Knoten darf also nicht zu groß sein.

Wird nun das Gitter währed der Integration simultan mitberechnet, so geht zur Bestimmung von $x_i(t+\Delta t)$ evtl. schon Information von dem Zustand auf der neuen Zeitschicht ein. Am bisher hemmenden Knoten wird der Gradient größer und der Knoten bewegt sich dann mit. Aber bei jeder Zeitintegration wird auch Information aus der Vergangenheit verwendet, und dies führt in jedem Fall zu einer Schrittweitenschranke der Form (4.19).

Die Wahl von α beeinflußt die Schrittweitenschranke nun dadurch, daß für größer werdendes α die aus (4.18) gewonnenen Geschwindigkeiten kleiner werden. Folglich läuft der Randknoten der Front langsamer auf den hemmenden Knoten zu. Größere Schrittweiten sind möglich. Dieser Effekt ist in Abbildung 4.3 illustriert. Allerdings wird dadurch die Knotengeschwindig-

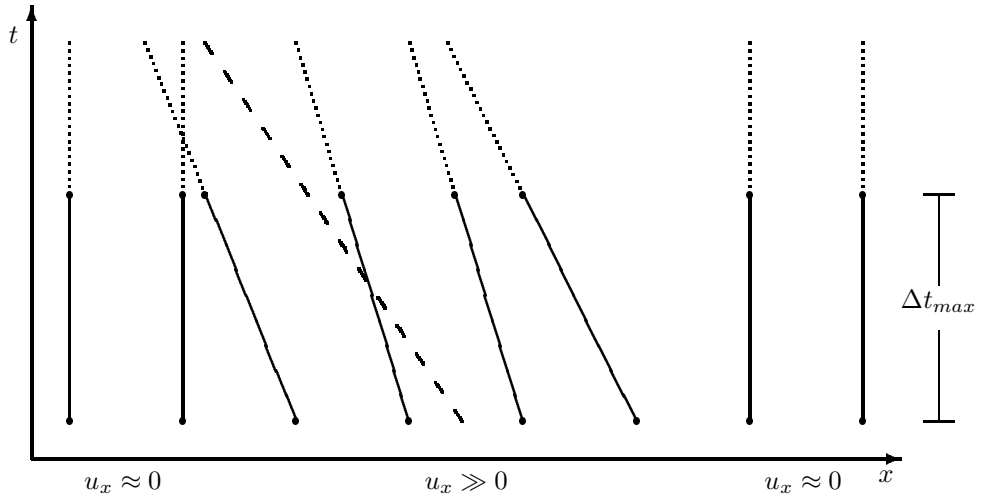


Abbildung 4.3: Knotenfluß wenn "α groß " in (4.18)

keit innerhalb der ganzen Front (evtl. deutlich) kleiner als die tatsächliche Frontgeschwindigkeit (gestrichelte Linie). Die Zeitableitung $\dot{u}$ wird größer – die Zeitschrittweiten meist kleiner.

4.2.3 Weitere Regularisierung der Gittergleichungen

Um dieses Problem zu mildern, wird in praktisch allen Verfahren, die mitbewegte Gitter verwenden, eine zusätzliche Regularisierung eingeführt, die das

Kreuzen von Gitterlinien verhindern soll. Dies kann, motiviert durch (4.19), durch die Forderung erreicht werden, daß die Differenz der Geschwindigkeit benachbarter Knoten um so kleiner wird, je näher die Knoten aneinanderliegen – vgl. etwa [55, 66]. Es soll also das Funktional

$$\sum_{i=2}^{n_x} \left\| \frac{\dot{x}_i - \dot{x}_{i-1}}{x_i - x_{i-1}} \right\|^2$$

minimiert werden. Mit der l_2–Norm erhält man daraus $n_x - 2$ Regularisierungsgleichungen für die inneren Knoten $x_i(t)$ des Gitters $\mathcal{G}$:

$$\frac{\dot{x}_i - \dot{x}_{i-1}}{(x_i - x_{i-1})^2} - \frac{\dot{x}_{i+1} - \dot{x}_i}{(x_{i+1} - x_i)^2} = 0 \ , \ i = 2, ..., n_x - 1 \ . \tag{4.20}$$

Diese Gleichungen sind in geeigneter Weise mit der diskretisierten Form von (4.15) zu koppeln, z.B. durch einfache Addition. Bezeichne nach Ortsdiskretisierung von (4.15) wieder $u_i = u(x_i, t)$ den Vektor der Unbekannten am Knoten i, so lautet die i–te Gitterbewegungsgleichung des semi–diskreten Systems:

$$\dot{u}_i^T(\Delta_x u_i) + \alpha \dot{x}_i + \lambda \left(\frac{\dot{x}_i - \dot{x}_{i-1}}{(x_i - x_{i-1})^2} - \frac{\dot{x}_{i+1} - \dot{x}_i}{(x_{i+1} - x_i)^2} \right) = 0 \ . \tag{4.21}$$

Ein Knotenfluß, der sich aus diesem Typ von Bewegungsgleichung ergibt, ist in Abbildung 4.4 illustriert. Durch die Ankopplung von (4.20) bewegen sich auch die Knoten vor und hinter der Front mit. Es entsteht eine fächerförmige Bewegung, um den Geschwindigkeitsunterschied zwischen den Knoten in der Front und den festen Randknoten möglichst gleichmäßig zu verteilen. Frontgeschwindigkeit und Frontknotengeschwindigkeit stimmen gut überein, die Schrittweiten werden i.a. groß. Allerding ist die Dynamik in der Gitterbewegung größer.

Interessant ist nun, daß auf einem äquidistantem Gitter die Gleichungen (4.20) einer Ortsdiskretisierung der Gleichung

$$-\dot{x}_{xx} = 0 \tag{4.22}$$

entsprechen. Da durch Hinzunahme eines Diffusionsterms in die Gitterbewegungsgleichung (4.14) gerade der gewünschte Dämpfungseffekt entsteht, kann man auch direkt die Gleichung

$$\dot{u}^T u_x + \alpha \dot{x} - \lambda \dot{x}_{xx} = 0 \tag{4.23}$$

diskretisieren und dann

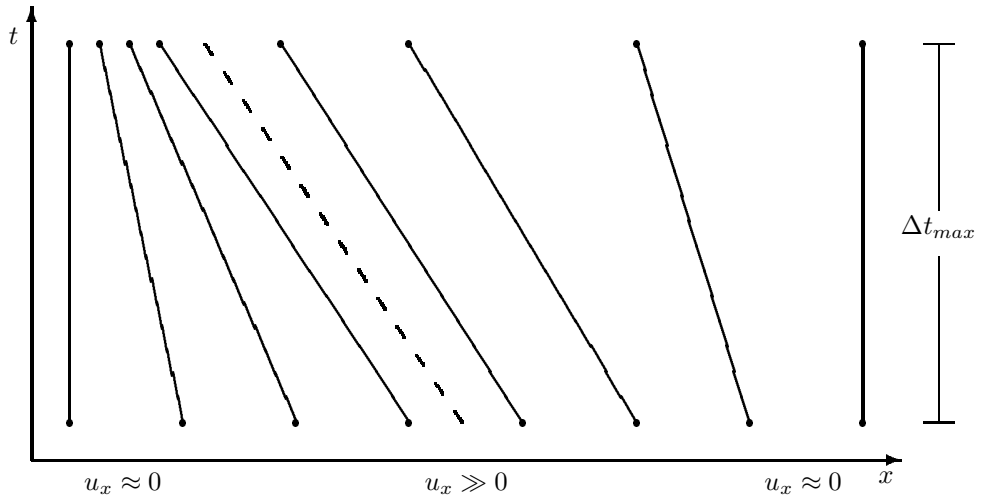


Abbildung 4.4: Knotenfluß wenn "α, λ optimal" in (4.21)

$$\begin{aligned} \dot{u}_i^T(\Delta_x u_i) + \alpha\dot{x} - \lambda\,(\Delta_{xx}\dot{x}_i) = \\ \dot{u}_i^T(\Delta_x u_i) + \alpha\dot{x}_i - \lambda \left(\frac{\frac{\dot{x}_i - \dot{x}_{i-1}}{x_i - x_{i-1}} - \frac{\dot{x}_{i+1} - \dot{x}_i}{x_{i+1} - x_i}}{\frac{x_{i+1} - x_{i-1}}{2}} \right) \quad = \quad 0 \end{aligned} \tag{4.24}$$

anstelle von (4.21) verwenden. Im Prinzip wird die Gleichung (4.23) auch in [67] verwendet, allerdings ist dort ein insgesamt positiver Diffusionsterm für die Geschwindigkeiten $\dot{x}$ angegeben, d.h. die Gleichung

$$\dot{u}^T u_x + \alpha\dot{x} + \lambda\dot{x}_{xx} = 0 \quad , \lambda > 0 \; . \tag{4.25}$$

Nur bei sorgfältiger Wahl der Parameterwerte α, λ (α groß , λ klein) läßt sich auch mit (4.25) ein akzeptables Gitterbewegungsverhalten erzielen. Da der Diffusionsterm $\lambda\dot{x}_{xx}$ auf der linken Seite der PDG steht, ist er nur bei negativem Vorzeichen von seiner typischen dämpfenden Natur und liefert für $\lambda \gg 1$ ein praktisch festes Gitter.

4.2.4 Zwei allgemein verwendbare Gittergleichungen

Damit ist aus dem ursprünglich so einfachen Ansatz (4.11) bereits eine Vielzahl von möglichen Realisierungen entstanden (und weitere sind denkbar). Ohne hier auf alle Details einzugehen, haben sich – im betrachteten Kontext – letztlich nur zwei Varianten als robust und effizient erwiesen. Zum einen die implizite Form (4.16) mit $\alpha = 0$ und in Verbindung mit der

Regularisierung über einen Diffusionsterm (auch künstliche "Knotenviskosität" genannt). Zum anderen (nur für $B = I$ möglich) eine Modifikation der halbexpliziten Form (4.17), wieder mit $\alpha = 0$, aber mit der Regularisierung (4.20). Damit lauten die transformierten semi–diskreten Gleichungen an inneren Knoten $x_i\ ,\ i = 2, ..., n_x - 1\ :$

$$\begin{array}{lrcl} \text{a)} & B(\dot{u}_i - (\Delta_x u_i)\dot{x}) & = & f_i^\Delta \\ \text{b)} & \dot{u}_i^T(\Delta_x u_i) - \lambda(\Delta_{xx}\dot{x}_i) & = & 0\ , \end{array} \tag{4.26}$$

bzw. $(B = I)$

$$\begin{array}{lrcl} \text{a)} & \dot{u}_i - (\Delta_x u_i)\dot{x}_i & = & f_i^\Delta \\ \text{b)} & \hat{\lambda}\left(\frac{\dot{x}_i - \dot{x}_{i-1}}{(x_i - x_{i-1})^2} - \frac{\dot{x}_{i+1} - \dot{x}_i}{(x_{i+1} - x_i)^2}\right) + (\Delta_x u_i)^T(\Delta_x u_i)\dot{x}_i & = & -(f_i^\Delta)^T u_x\ . \end{array} \tag{4.27}$$

Dabei läßt sich nur (4.26) als kontinuierliches Problem (im Inneren des Gebietes) formulieren:

$$\begin{array}{lrcl} \text{a)} & B(\dot{u} - u_x\dot{x}) & = & f(u) \\ \text{b)} & \dot{u}^T u_x - \lambda\dot{x}_{xx} & = & 0\ . \end{array} \tag{4.28}$$

Beide Varianten werden normalerweise zusammen mit trivialen Gleichungen für die Randknoten verwendet.

$$\dot{x}_1 = 0\ ,\quad \dot{x}_{n_x} = 0\ \ . \tag{4.29}$$

Im Kontext des hier entwickelten Verfahrens werden gute Erfahrungen mit der Parameterwahl

$$\lambda, \hat{\lambda} \in [0.01, 0.5] \tag{4.30}$$

gemacht. Um die Moving–Grid–Technik aber wirklich effektiv bei sehr unterschiedlichen Problemen anwenden zu können, ist ein Fein-Tuning letztlich unumgänglich. Als "Startwerte" können

$$\lambda = 0.1\ \ ,\quad \hat{\lambda} = 0.25 \tag{4.31}$$

empfohlen werden.

Ohne die genaue Form anzugeben, siehe hierzu (4.45), sei bereits jetzt die abkürzende Schreibweise für das semi–diskrete Gesamtsystem in mitbewegten Koordinaten angegeben:

$$\tilde{\mathbf{B}}^\Delta(\mathbf{w}) \cdot \dot{\mathbf{w}} = \tilde{\mathbf{f}}^\Delta(\mathbf{w})\ . \tag{4.32}$$

Für die Entscheidung, den Term $\alpha\dot{x}$ in der Gitterbewegungsgleichung grundsätzlich wegzulassen, waren 3 Gründe ausschlaggebend. Zum einen ist er zur

formalen Regularisierung der Gittergleichung für den Fall $u_x \to 0$ nicht mehr notwendig. Zum anderen ist der Term, bei dem Versuch das Gitter möglichst optimal mitzubewegen, eher hinderlich. Und die Tatsache, daß letztlich nur noch ein einziger, gut interpretierbarer Parameter anzupassen ist, ist ein Vorteil, der die Gleichungen (4.26) und (4.27) von den meisten anderen in der Literatur zu findenden Ankopplungsgleichungen abhebt.

Ehe die Mitbewegung eines Gitters nach (4.26) einer weiteren Analyse unterzogen wird, ist noch zu bemerken, daß auch die in den zu minimierenden Funktionalen verwendete Norm, und somit auch die Gittergleichungen, in geeigneter Weise zu gewichten ist. In unskalierter Form sind die enthaltenen Parameter nicht dimensionslos und damit von den in der PDG verwendeten physikalischen Einheiten abhängig.

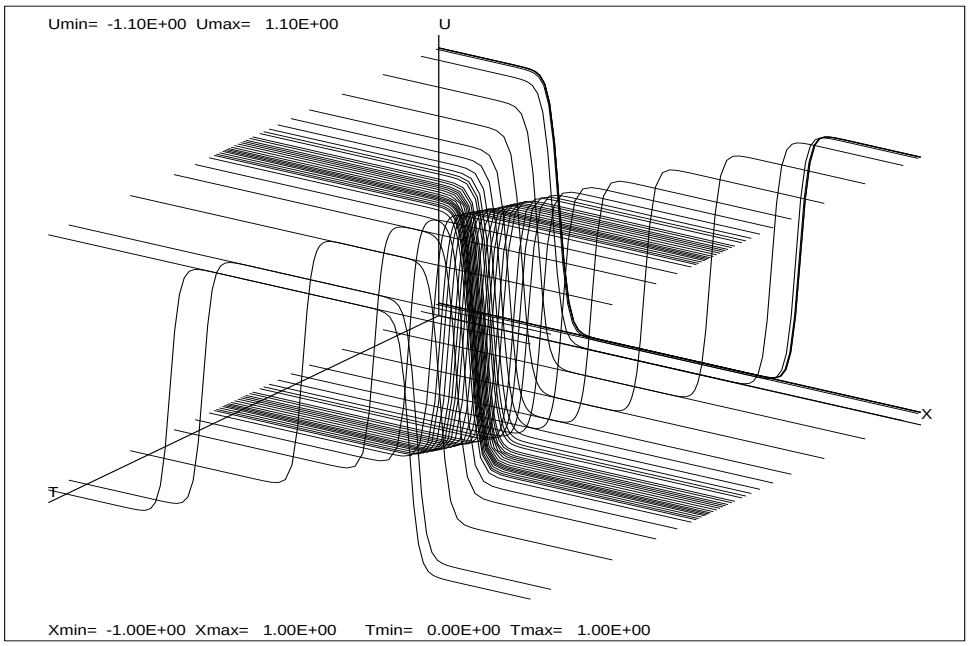


Abbildung 4.5: Kreuzen zweier wandernder Fronten

Mit den obigen Gleichungen lassen sich auch mehrere Fronten gleichzeitig verfolgen. Insbesondere wenn die Fronten aufeinanderzulaufen, ist die Gittermitbewegung extremen Anforderungen unterworfen. In Abbildung 4.5 ist ein derartiges Problem (Überkreuzung zweier identisch aussehender Fronten) illustriert. Der durch (4.26) erzeugte Knotenfluß ist in Abbildung 4.6 dargestellt. Es zeigt sich, daß auch in einer solchen Situation die Gitter fächerförmig gestaucht bzw. auseinandergezogen werden. Ob eine derartige Frontverfolgung jedoch mit nur einem Gitter erfolgen soll, oder ob dann nicht die Verwendung mehrerer Gitter, was prinzipiell möglich ist, vorteilhaf-

ter wäre, hängt stark vom gegebenen Problem ab und soll hier nicht weiter diskutiert werden. Eine wesentliche Schwierigkeit stellt etwa die Notwendigkeit einer in x globalen Lösungsdarstellung auch während eines Integrationsschrittes dar.

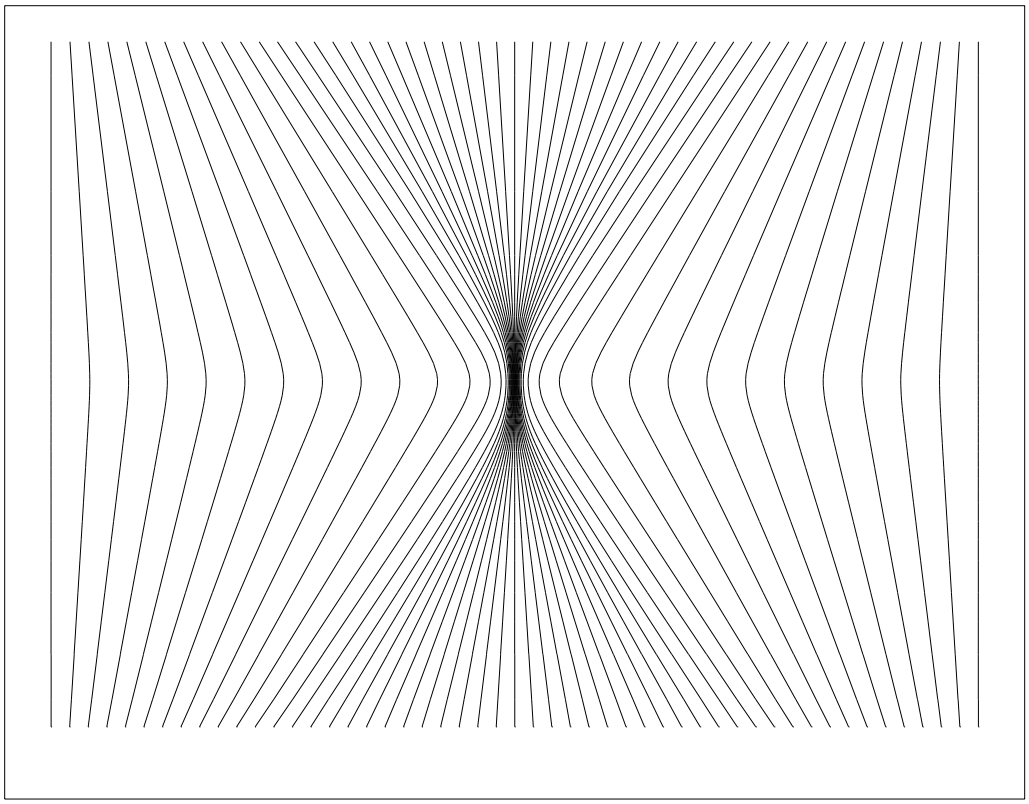


Abbildung 4.6: Knotenfluß in der x–t–Ebene beim Kreuzen zweier Fronten

4.2.5 Analyse und Vergleich

Alternative Zielfunktionale

Die Wahl $\alpha = 0$ hat auch einen recht grundsätzlichen Aspekt. Anstelle des ursprünglichen Ziels, alle Zeitableitungen im neuen Koordinatensystem zu minimieren (d.h. (4.11)), werden nun nur die Zeitableitungen der ursprünglichen Unbekannten, d.h.

$$\|\dot{u}\|^2 \stackrel{!}{=} \min , \tag{4.33}$$

minimiert. Doch dies muß nicht unbedingt ein Nachteil sein. Denn bereits das Maß $\|\dot{w}\|$ kann zur Charakterisierung der Schwierigkeit des Problems

ungeeignet sein. So wurde z.B. bereits in [42] die Frage aufgeworfen, ob nicht anstelle von (4.11) besser ein Funktional der Form

$$|| \ddot{w} ||^2 \stackrel{!}{=} \min \tag{4.34}$$

minimiert werden sollte. Das Problem einer robusten und effizienten algorithmischen Realisierung blieb allerdings offen und soll auch hier nicht näher untersucht werden. Wichtig ist jedoch ein anderer in [42] angesprochener Gesichtspunkt. Durch die Ersetzung von (4.11) durch (4.33) kann im Prinzip das Problem entstehen, daß die transformierten Gleichungen zur Berechnung von u zwar einfach zu lösen sind, sich die ganze Schwierigkeit des Problems dadurch aber in der Bestimmung der Knoten x_i wiederfindet. Nur durch einen sehr restriktiven Parameterwert $\alpha > 0$ kann dies generell vermieden werden. Nun wird es i.a. ohnehin eine Ja/Nein–Entscheidung sein, ob die Verwendung eines mitbewegten Gitters dem Problem angemessen ist. Wenn "Ja", dann soll aber eine Gitterbewegung erfolgen, die möglicht nahe an der optimal möglichen ist. Wie die optimale Mitbewegung aussieht, und wie sie zu erreichen ist, soll nun anhand einer idealen Problemklasse untersucht werden.

Problemtyp "Traveling Wave"

Wie zu Beginn dieses Kapitels erwähnt, hat der hier verwendete Moving–Grid–Ansatz zum Ziel, einer wandernden Front zu folgen. Oft ändert sich dabei die Gestalt nur wenig, so daß "$\|\dot{u}\|$ = klein " mit "$\|\dot{x}\|$ = groß " erreicht werden kann. Und nur mit dem Ansatz (4.33) kann man im Grenzfall, daß die Lösung $u(x,t)$ der PDG eine sog. "traveling wave" ist, tatsächlich die optimale Gitterbewegung erreichen. Zwar ist ein Problem, dessen Lösung im gesamten Integrationsintervall eine reine "traveling wave" ist, nicht unbedingt die hier interessierende Problemklasse (und läßt sich besser als parameterabhängiges Randwertproblem formulieren und lösen), aber bei vielen auch praktisch relevanten Problemen bildet sich nach einer gewissen Integrationszeit eine wandernde Front aus. Und dann ist es von großem Vorteil, wenn eine Gitterankopplungsgleichung gewählt wurde, mit der man dieser Front möglichst optimal folgen kann. Zur Veranschaulichung betrachte man ein allgemeines, nichtlineares Problem (wieder in vereinfachter Schreibweise)

$$u_t = f(u) \ , \tag{4.35}$$

dessen Lösung, zumindest ab einem gewissen Zeitpunkt $\bar{t} \geq t_0$, eine sich mit konstanter Geschwindigkeit v fortbewegende "Welle" ist. Dann gilt

$$u(x,t) = u(\bar{x} + v(t-\bar{t}), \bar{t}) \ \text{ für } t \geq \bar{t}$$

und damit hat die Lösung $u(x,t)$ die Eigenschaft

$$0 = \dot{u} = u_x \dot{x} + u_t \ . \tag{4.36}$$

Anstelle von (4.35) kann man also schreiben

$$u_t = f(u) = -u_x v \ . \tag{4.37}$$

Löst man (4.35) auf mitbewegtem Gitter, so ergibt mit (4.36, 4.37), rein formal, die zu lösende Gleichung

$$\dot{u} = u_x(\dot{x} - v) \ .$$

Die im Sinne von (4.33) optimale Wahl

$$\dot{x}_i = \begin{cases} v & , \quad \text{wenn } u_x(x_i, t) \neq 0 \\ \text{beliebig} & , \quad \text{sonst} \end{cases}$$

ist aber nur dann realisierbar, wenn sich daraus kein Widerspruch zur verwendeten Gitterbewegungsgleichung ergibt.

Für die hier verwendeten Gittergleichungen ist dies möglich, wenn man anstelle der Randbedingung (4.29) ebenfalls mitbewegte Randknoten verwendet, also etwa die Gleichungen

$$\dot{x}_1 - \dot{x}_2 = 0 \ , \quad \dot{x}_{n_x} - \dot{x}_{n_x - 1} = 0 \tag{4.38}$$

benutzt. Die verwendete Regularisierung verhindert nicht die optimale Wahl, sondern erlaubt für den Fall einer "traveling wave" sogar die natürliche Mitbewegung aller Knoten (auch der Randknoten) mit gleicher Geschwindigkeit. Für die im Sinne des Funktionals (4.11) optimalen Gitterbewegungsgleichungen, ergibt sich dies nur mit der Wahl $\alpha = 0$.

Am Rande sei noch bemerkt, daß nach Ortsdiskretisierung die optimale Geschwindigkeit ungleich der exakten Geschwindigkeit ist. Oder umgekehrt, bei Integration des transformierten Systems (mit den Randgleichungen (4.38)) stellt sich eine vom Ortsdiskretisierungfehler abhängige Geschwindigkeit $v_\Delta \neq v$ ein.

In der Praxis läßt sich die Wahl freier Ränder häufig nicht durchführen, insbesondere nicht für den gesamten Integrationszeitraum. Inwieweit allein dadurch die Effizienz einer Moving–Grid–Technik reduziert wird, und wie sensitiv die Gittergleichungen auf Änderung der Parameter λ, $\hat{\lambda}$ reagieren, soll nun anhand eines einfachen numerischen Experiments verdeutlicht werden.

Ein numerisches Experiment

Zu Lösen sei das Problem

TESTPROBLEM **tanh**:

$$\begin{array}{lrcl} \text{a)} & x & \in & [-1,1]\ ,\ \ t \in [0,1] \\ \text{b)} & u_t & = & u_{xx} + p_3(1-u^2) + 2p_2^2(u-u^3) \\ \text{c)} & u(x,t_0) & = & \tanh(p_2(x-p_1), p_3 t_0) \\ \text{d)} & u(x_L,t) & = & -1\ ,\ \ u(x_R,t) = 1\ ;\ (t > t_0) \end{array} \tag{4.39}$$

mit

$$p_1 = 0.5\ ,\ \ p_2 = 25.0 \quad p_3 = 25.0\ .$$

Für die betrachtete Integrationszeit stellt die Funktion

$$u(x,t) = \tanh(p_2(x-p_1), p_3 t)$$

praktisch die exakte Lösung dar, da der Effekt der gewählten Dirichlet-Randbedingungen vernachlässigbar ist.

Dieses Problem wird nun auf einem gemäß (4.26) mitbewegten Gitter zeitintegriert. Um den Effekt der Gittermitbewegung auf die Schwierigkeit der Zeitintegration in einfacher Weise studieren zu können, wird auf eine Kontrolle des Ortsdiskretisierungsfehlers und auf eine statische Gitteranpassung verzichtet. Das Verhalten der verbleibenden reinen Zeitschrittweitenkontrolle gibt dann Aufschluß darüber, inwieweit die Mitbewegung des Gitters das Problem tatsächlich vereinfacht. Um den Einfluß des Ortsdiskretisierungsfehlers möglichst klein zu halten, wird ein dem Problem angepaßtes nichtäquidistantes Startgitter verwendet. Es wird mit einer Startschrittweite $\Delta T_0 = 2 \cdot 10^{-4}$ begonnen und eine vorgegebene Zeitintegrationsgenauigkeit von $tol_t = 10^{-4}$ ist einzuhalten.

Integriert man nun mit mitbewegten Randknoten, so hat man die bestmögliche Problemvereinfachung durchgeführt und die Zeitintegration sollte ein triviales Problem sein. Tatsächlich stellt sich auch fast das bestmögliche Verhalten ein. Zur numerischen Integration werden nur 3 Schritte mit insgesamt 12 Auswertungen der das semi–diskrete System beschreibenden Problemfunktionen $\tilde{\mathbf{B}}^{\Delta}$,$\tilde{\mathbf{f}}^{\Delta}$ benötigt. Dabei stimmt am Endzeitpunkt die numerische Frontposition mit der exakten Frontposition $x_f = -0.5$ bis auf einen (absoluten) Fehler von $\varepsilon = 1.2 \cdot 10^{-2}$ überein. Als Frontposition ist dabei der Wert $x_f(t)$ zu verstehen, an dem $u(x_f,t) = 0$ gilt. Da dieser Wert i.a. nicht genau auf einem der Knoten der numerischen Lösung $U(X_i,t)$ zu finden ist, wird er durch Nullstellenbestimmung einer linear Interpolierenden ermittelt. Im vorliegenden Fall ist die Genauigkeit der numerischen Frontposition X_f praktisch vollständig von der (unkontrollierten) Ortsdiskretisierung abhängig.

Dieses sehr günstige Integrationsverhalten ist im Vergleich zu einer Integration mit festem Gitter zu sehen. Mit einem, nun notwendigerweise äquidistanten, feinen Gitter der Dimension $n_x > 200$ ergibt sich, praktisch unabhängig

von der Feinheit des Gitters, eine Schrittzahl von ca. 315 mit 1350 Auswertungen von $\tilde{\mathbf{B}}^{\Delta}$,$\tilde{\mathbf{f}}^{\Delta}$. Der durch optimale Gittermitbewegung erzielte Beschleunigungsfaktor von ca. 100 ist zwar dramatisch, aber für diesen idealen Testfall letztlich nicht verwunderlich.

Interessanter ist nun, welches Integrationsverhalten sich einstellt, wenn man die Randknoten festhält und in der Gitterbewegungsgleichung verschiedene Werte für den Parameter λ wählt. In Tabelle 4.1 sind die Ergebnisse einiger dieser Integrationen dargestellt. Sie sind repräsentativ für das Ergebnis, das man erhält, wenn man einen Bereich $\lambda \in [10^{-20}, 10^{2}]$ sehr fein abtastet.

λ	*nstep*	*nfcn*	Fehler	Position
10.0	48	631	$1.3 \cdot 10^{-2}$	17/18
2.0	12	189	$6.0 \cdot 10^{-3}$	25/26
1.0	8	101	$1.5 \cdot 10^{-2}$	28/29
0.5	6	63	$4.0 \cdot 10^{-2}$	30/31
0.1	5	47	$2.5 \cdot 10^{-2}$	33/34
0.01	13	58	$2.5 \cdot 10^{-2}$	34/35
10^{-4}	12	36	$1.1 \cdot 10^{-2}$	34/35
-10^{-16}	9	55	$1.2 \cdot 10^{-2}$	34/35
$0.1/\alpha$	16	121	$1.0 \cdot 10^{-1}$	32/33

Tabelle 4.1: Verhalten der Zeitintegration für verschiedene Regularisierungswerte λ in (4.26)

Als Indikatoren, wie einfach dadurch das Zeitintegrationsproblem geworden ist, sind die Anzahl der benötigten Integrationschritte (*nstep*) und Funktionsauswertungen (*nfcn*) angegeben. Daneben sind noch der beobachtete Fehler in der Frontposition und die Nummer der Knoten, zwischen denen sich x_f befindet, aufgeführt.

Bezüglich des Verhaltens von Aufwand in Abhängigkeit von λ stellt sich das erwartete Verhalten ein. Je kleiner λ gewählt wird, desto mehr sinkt der benötigte Aufwand zunächst. Unter ein gewisses, von λ dann praktisch unabhängiges, Niveau kann der Aufwand jedoch nicht reduziert werden. Da die Schrittweitenkontrolle variable Ordnungs– und Schrittweitenwahl besitzt, kann es dabei zu Schwankungen in der Anzahl der benötigten Schritte kommen. Die angegebene Frontposition zeigt sehr deutlich den Effekt des Diffusionsterms in der Gittergleichung. Wählt man große Werte für λ, so besitzt das Gitter eine zu große Viskosität. Die optimale Bewegung der Knoten, an denen die Front zu Beginn der Integration war ($x_f = x_{35}$, sowie dessen nächste Nachbarn), wird dadurch behindert. Die Lösungsfront läuft schneller als diese "Frontknoten" und somit entsteht an allen Knoten, über die die

Lösungsfront hinwegläuft, genau der Typ von Dynamik, der bei einem festen Gitter dem Problem innewohnt – wenn auch in abgeschwächter Form. Die generelle Tendenz ist offensichtlich. Je kleiner λ wird, um so besser folgen die ursprünglichen Frontknoten der Lösungsfront. Geringe Abweichungen der beiden Geschwindigkeiten führen zu kaum gesteigertem Aufwand, aber wenn die Differenz zu groß wird, ist der gewünschte Effekt rasch verloren. Dabei besteht zwischen der Größe der Differenz zur optimalen Geschwindigkeit und der daraus resultierenden Aufwandserhöhung ein nichtlinearer Zusammenhang. Insgesamt läßt sich aber auch bei festen Rändern eine drastische Effizienzsteigerung feststellen.

Interessant ist, daß das Verhalten der Gitterlinien auch für sehr kleine Werte von λ so robust ist. Es sind praktisch keine Überschneidungen von Gitterlinien (was eine automatische Wiederholung des aktuellen Zeitschrittes mit kleinerer Schrittweite zur Folge hat) zu beobachten. Für realistische Probleme, wo sich die Gestalt der Front noch ändert, können derartig kleine Werte nicht verwendet werden. Die geringe Regularisierungskraft eines sehr kleinen Diffusionsterms verhindert das Überkreuzen von Gitterlinien nicht mehr, wenn die reine Minimierung von $||\dot{u}||$ zu deutlich unterschiedlichen Geschwindigkeiten an Nachbarknoten führt.

Eine deutlich geringere Robustheit gegenüber Variation von λ läßt sich bei einem vergleichenden Experiment mit der Gitterbewegungsgleichung (4.27) feststellen. Für den in (4.30) angegebenen Wertebereich ergeben sich jedoch ähnliche Ergebnisse. In der letzten Zeile der Tabelle ist noch das Resultat einer Integration angegeben, in der der Regularisierungsterm $\alpha\dot{x}\,,\ \alpha = 1$, der Gittergleichung hinzugefügt ist. Der benötigte Aufwand ist mehr als doppelt so groß wie in der vergleichbaren Variante ohne diesen Term. Darüber hinaus führt das dadurch forcierte Aufreißen des Gitters hinter der Front zu einer schlechter werdenden Ortsdiskretisierung und somit zu einem größeren Fehler in der Frontposition.

4.2.6 Spezielle Gitterbewegungsvarianten

Verzichtet man auf den Anspruch, die Gitterankopplung mittels einer (formal) allgemein anwendbaren Gittergleichung durchzuführen, so lassen sich aus (4.28) noch interessante Spezialisierungen ableiten. Ist man z.B. daran interessiert, die Position einer wandernden Front mit einer Gitterlinie genau zu verfolgen, so ist dazu der allgemeine Ansatz nicht geeignet. Allerdings bietet sich folgende Alternative an. Man koppelt einen ausgezeichneten Knoten durch eine spezielle Bedingung an die Front an und bewegt alle anderen durch eine Regularisierungsbedingung mit. Wählt man als Definition für den Begriff "Frontposition" etwa

$$u(x_f(t), t) = \bar{u} \quad , \ \bar{u} \text{ vorgegeben} \,, \tag{4.40}$$

und nimmt an, daß es einen Knoten x_i im aktuellen Gitter $\mathcal{G}$ mit $x_i \approx x_f$ gibt, so kann man für den folgenden Integrationsschritt als i_f–te Gittergleichung zur Bestimmung des Knotens x_f die Gleichung

$$0 = u_{i_f} - \bar{u} \tag{4.41}$$

verwenden. Zur Vereinfachung der Notation wird in diesem Abschnitt $u(x,t)$ wieder als skalare Größe angesehen. Im Systemfall ist dann (4.40) als Bedingung an eine spezielle Komponente, etwa $\bar{j}$, zu verstehen. Eine automatische Ankopplung ist natürlich nur dann möglich, wenn x_f durch (4.40) eindeutig festgelegt ist. Gleiches gilt für andere interessante Bedingungen, etwa

$$\begin{aligned} \text{a)} \quad 0 &= u_x(x_f(t),t) - \bar{u}_x \\ \text{b)} \quad 0 &= u_{xx}(x_f(t),t) - \bar{u}_{xx} \\ & \quad \bar{u}_x, \bar{u}_{xx} \text{ vorgegeben .} \end{aligned} \tag{4.42}$$

In der aktuellen Implementierung des Verfahrens sind obige spezielle Gitterankopplungen als optionale Varianten realisiert, allerdings wird die eindeutige Möglichkeit der Bestimmung von x_f vorausgesetzt. Algorithmisch und softwaretechnisch sind auch allgemeinere Fälle handhabbar, aber z.Z. nicht realisiert.

Zur Mitbewegung der anderen Knoten bietet es sich an, eine reine Diffusionsgleichung für die Geschwindigkeiten, also (4.22) bzw. deren Semi–Diskretisierung, in Verbindung mit festen oder freien Randknoten zu verwenden. Diese spezielle Ankopplungstechnik ergibt sich auch direkt aus der Gittergleichung (4.28.b), wenn man dort einen ortsabhängigen Parameter λ zuläßt und den Grenzprozeß

$$\begin{aligned} \lambda(x) &\to \infty \quad \text{für } x \neq x_f \\ \lambda(x) &= 0 \quad \text{für } x = x_f \end{aligned}$$

durchführt. Die erste Gleichung liefert (4.22) und aus der zweiten folgt (sei $x = x_f, u_x(x_f,t) \neq 0$)

$$u\dot{u} = 0 \Leftrightarrow \dot{u} = 0 \Leftrightarrow u = \text{const} ,$$

was gerade (4.41) entspricht. Durch diese Art der Ankopplung löst man praktisch ein Problem mit einem freien inneren Rand, dessen Position sich aus einer Zusatzbedingung ergibt. Als zweite Gleichung, zur Bestimmung der Lösung $u(x_f,t)$, ist am Knoten x_{i_f} weiterhin die transformierte PDG (4.28.a) zu lösen.

Man kann diese beiden Gleichungen nun durch ein Gleichungspaar der Art

$$\begin{aligned} \text{a)} \quad 0 &= u(x_f(t),t) - \hat{u}(x_f(t),t) \\ \text{b)} \quad \hat{b}\dot{x} &= \hat{\alpha} + \hat{\beta^+}u_x^+ - \hat{\beta^-}u_x^- , \end{aligned} \tag{4.43}$$

ersetzen, wobei $\hat{b}, \hat{\alpha}, \hat{\beta^+}, \hat{\beta^+}$ vorgegebene (evtl. von x_f, t, u abhängige) Funktionen sind, und u_x^+, u_x^- die rechts– bzw. linksseitige 1. Ableitung bezeichnen. Damit kann das Gesamtsystem als ein nichtlineares zweiphasiges Stefan–Problem angesehen werden. Prinzipiell ist man also mit dieser ganz speziellen Moving–Grid–Technik in der Lage, einen sehr interessanten und schwierigen Problemtyp elegant zu lösen. Auf eine genauere Ausarbeitung dieser Möglichkeit, im Kontext des Gesamtverfahrens, wurde bisher verzichtet, da sich u.a. in Verbindung mit dem statischen Regridding prinzipielle Probleme einstellen können, z.B. wenn die links–und rechtseitigen Ableitungen nur durch einseitige Differenzenapproximationen 1. Ordnung ersetzt werden. Darüber hinaus wird die Lösung an x_f i.a. nicht mehr stetig differenzierbar sein, was zu Problemen bei der Interpolation führen kann.

Wird mit der Regularisierung (4.22) und einer der Ankopplungsbedingungen (4.40, 4.42a,b) eine wandernde Front verfolgt, so bewegt sich bei festen Rändern nur der Frontknoten x_{i_f} mit der im Sinne des Funktionals (4.33) optimalen Geschwindigkeit. Alle anderen sich auch in der Front befindenden Knoten werden bereits zur Dämpfung des Geschwindigkeitsunterschiedes verwendet. Daher ist es interessant, eine alternative Gittermitbewegung für die restlichen Knoten zu realisieren. Dazu seien zwei geeignete Indizes i_l, i_r gegeben. Dann lauten die restlichen Gittergleichungen

$$\begin{aligned} -\Delta_{xx}\dot{x}_i &= 0 \quad & i = 2, ..., i_l \\ \dot{x}_i - \dot{x}_{i+1} &= 0 \quad & i = i_l + 1, ..., i_f - 1 \\ \dot{x}_i - \dot{x}_{i-1} &= 0 \quad & i = i_f + 1, ..., i_r - 1 \\ -\Delta_{xx}\dot{x}_i &= 0 \quad & i = i_r, ..., n_x - 1 \ . \end{aligned} \tag{4.44}$$

Für den oben betrachteten "Traveling Wave"-Fall (mit einem Frontverlauf wie in Abbildung 4.1 angegeben) ist eine natürliche Wahl von i_l, i_r offensichtlich. Man bestimmt i_l, i_r derart, daß gilt:

$$u_x(x_{i_l}, t) < \hat{u}_x \leq u_x(x_{i_l+1}, t)$$

bzw.

$$u_x(x_{i_r}, t) < \hat{u}_x \leq u_x(x_{i_r-1}, t) \ .$$

Dabei ist $\hat{u}_x$ "geeignet" zu wählen. Offensichtlich ist eine automatische, und gleichzeitig gute, Bestimmung von i_l, i_r bereits für sehr einfache Probleme recht schwierig. Auf einige implementierte heuristische Kriterien zur Bestimmung von geeigneten Werten i_l, i_r soll hier nicht weiter eingegangen werden. Sie sind, ebenso wie alle hier aufgeführten speziellen Ankopplungsvarianten, nicht allgemein anwendbar. Für spezielle Problemtypen, die durchaus von praktischer Relevanz sind, können die hier dargestellten Techniken jedoch sehr effektiv sein.

4.3 Einbindung in das Gesamtverfahren

4.3.1 Die semi–diskreten Gleichungen

Fügt man die neuen Unbekannten $x_i(t), i = 1, \ldots, n_x$ den bereits vorhandenen Vektoren $u_i = (u_{1,i}, \ldots, u_{n_{pde},i})^T$ am jeweiligen Knoten i hinzu, so erhält man die neuen Vektoren

$$w_i := (u_i, x_i)^T \ , \ i = 1, \ldots, n_x \ .$$

Der neue Gesamtvektor der Unbekannten ist dann durch

$$\mathbf{w}(t) := (w_1(t), \ldots, w_{nx}(t))^T \in \mathbb{R}^{n_x \cdot (n_{npde}+1)}$$

gegeben. Als zu integrierendes semi–diskretes Gesamtsystem erhält man (vgl. (2.63 – 2.68)):

$$\tilde{\mathbf{B}}^{\Delta}(t, \mathbf{w}(t)) \cdot \dot{\mathbf{w}}(t) = \tilde{\mathbf{f}}^{\Delta}(t, \mathbf{w}) \ , \quad t \in [t_0, t_{end}] \tag{4.45}$$

Im Gegensatz zu (2.65) ist $\tilde{\mathbf{B}}^{\Delta}$ eine Blocktridiagonalmatrix. Für die Gitterankopplung nach (4.26.a) haben die Diagonal– und Nebendiagonalblöcke folgende Form:

$$i = 2, \ldots, n_x - 1$$

$$\tilde{\mathbf{B}}^{\Delta}_{i,i} = \begin{pmatrix} \mathbf{B}_i^{\Delta} & \mathbf{B}_i^{\Delta}(\Delta_x u_i) \\ (\Delta_x u_i)^T & b_i^M \end{pmatrix} \ ,$$

$$\tilde{\mathbf{B}}^{\Delta}_{i-1,i} = \begin{pmatrix} 0 & 0 \\ 0 & b_i^L \end{pmatrix} \ , \ \tilde{\mathbf{B}}^{\Delta}_{i,i+1} = \begin{pmatrix} 0 & 0 \\ 0 & b_i^R \end{pmatrix}$$

mit

$$b_i^M = \frac{2\lambda}{(x_{i+1} - x_i)(x_i - x_{i-1})} \ ,$$

$$b_i^L = \frac{-2\lambda}{(x_{i+1} - x_{i-1})(x_i - x_{i-1})} \ , \quad b_i^R = \frac{-2\lambda}{(x_{i+1} - x_{i-1})(x_{i+1} - x_i)} \ ,$$

und am Rand

$$i = 1, n_x$$

$$\tilde{\mathbf{B}}^{\Delta}_{i,i} = \begin{pmatrix} \mathbf{B}_i^{\Delta} & \mathbf{B}_i^{\Delta}(\Delta_x u_i) \\ 0 & 1 \end{pmatrix} \ ,$$

$$\tilde{\mathbf{B}}^{\Delta}_{1,2} = \tilde{\mathbf{B}}^{\Delta}_{n_x-1,n_x} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} \ .$$

Die transformierte rechte Seite der PDG ergibt sich zu

$$\tilde{\mathbf{f}}_i^{\Delta} = \begin{pmatrix} \mathbf{f}_i^{\Delta} \\ 0 \end{pmatrix} \ , \ i = 1, \ldots, n_x \ .$$

Durch die Gitterankopplung erhält man in jedem Fall ein semi–diskretes System, das von linear–impliziter Struktur (bzgl. $\dot{\mathbf{w}}$) ist.

4.3.2 Fehlerkontrolle

Am prinzipiellen algorithmischen Ablauf, wie er in Abschnitt 3.4 zusammenfassend beschrieben ist, ändert sich durch die Ankopplung einer Gittergleichung nichts. Der wesentliche Unterschied ist in der Tatsache zu sehen, daß die jeweiligen Gitter nun selbst Lösung einer PDG sind. Damit ändert sich die virtuelle Gitterfunktion (2.25) nicht nur von Schritt zu Schritt, sondern auch innerhalb eines Zeitintegrationsschrittes, d.h.

$$\xi(r) \rightarrow \xi(r,t) \ .$$

Die Gültigkeit der lokalen Defektentwicklungen aus Abschnitt 2.2 und die der globalen asymptotischen Entwicklung (2.114) bleibt davon unberührt. Dennoch wird die Fehlerschätzung und –kontrolle nicht unmodifiziert auf das erweiterte System übertragen. Vielmehr werden die Gitterkomponenten des erweiterten Systems davon ausgenommen. Dieses Vorgehen wird durch die Tatsache motiviert, daß das eigentliche Problem die (fehlerkontrollierte) Berechnung einer numerischen Approximation $U(x,t)$ and die exakte Lösung $u(x,t)$ ist. Durch die Diskretisierung mit finiten Differenzen liegt eine derartige Approximation ohnehin nur an gewissen Knoten x_i vor. Deren genaue Position ist dabei unerheblich.

Wenn die Gittermitbewegung das angestrebte Ziel erreicht, daß die Dynamik in den U–Komponenten drastisch reduziert wird, tritt häufig das bereits in Abschnitt 3.4 angesprochene Problem einer Scheindynamik auf. Die Güte der eigentlich zur Fehlerkontrolle zu verwendenden Schätzer $\bar{\varepsilon}^t_{2,k-1}$ und $\bar{\varepsilon}^x_{1,k}$ wird durch den Einfluß der inkonsistenten Startdaten (Feingitterlösung auf dem Grobgitter) deutlich gestört. Insbesondere für restriktive Toleranzvorgaben oder deutlich unterschiedliche Vorgaben für die Orts– und Zeitdiskretisierungsgenauigkeiten kann dieser Effekt zu sehr kleinen Schrittweiten führen (in Ort und Zeit), obwohl der wahre Fehler der Feingitterlösung bereits unterhalb der geforderten Toleranz liegt. Tritt ein derartiges Phänomen auf (ein interner Verfahrensmonitor zeigt stark differierende Fehlerschätzer auf Grob– und Feingitter an), so empfiehlt es sich, nur eine abgeschwächte Form der Fehlerkontrolle zu verwenden. Dabei wird anstelle des Konvergenztests (3.42) nur die t–Konvergenz der Grobgitterintegration verlangt, d.h

$$\epsilon^t_{1,k-1} \leq tol_t \ .$$

Mit diesem Schätzer anstelle von $\tilde{\varepsilon}_2^t$ (vgl. 3.79) und dem gewohnten x–Fehlerschätzer $\bar{\varepsilon}_{1,k}^x$ wird anschließend die simultane Schrittweitenbestimmung und lokale Gitteranpassung durchgeführt. Auch mit dieser abgeschwächten Form der Fehlerkontrolle wurden gute Erfahrungen bzgl. Robustheit und Genauigkeit der Gesamtintegration gemacht. Sie stellt allerdings nur eine optionale Verfahrensvariante dar.

Bzgl. der Anfangswertverwendung im nächsten Integrationsschritt hat sich gezeigt, daß, im Gegensatz zum Vorgehen für die Lösungskomponenten U, für die Gitterkomponenten X ein alternatives Vorgehen vorzuziehen ist. Es werden die durch einmalige r–Extrapolation gewonnenen Knotenwerte als neues Grobgitter verwendet und das zugehörige Feingitter durch lineare Interpolation konstruiert.

4.3.3 Illustration

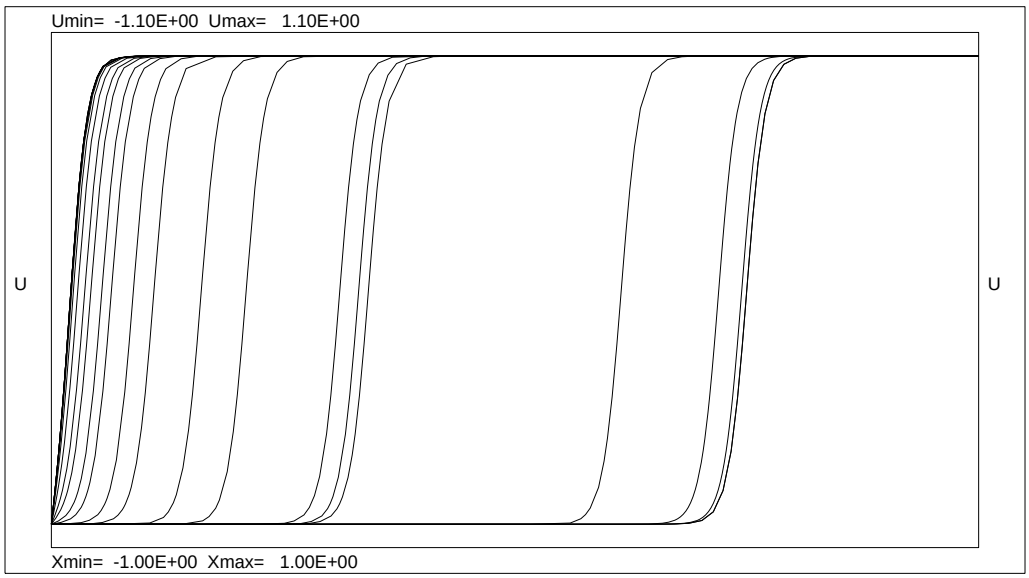


Abbildung 4.7: Lösung an allen Integrationszeitpunkten für Testproblem **tanh–2**

Wie bei der Kopplung von statischer und dynamischer Gitteranpassung ein typische Knotenfluß aussieht, ist in Abbildung 4.8 dargestellt. Es handelt sich um den Knotenfluß, der sich bei der Integration des Testproblems **tanh**, vgl. (4.39), bis zu einer Endzeit $t_{end} = 2$ ergibt. Dadurch läuft die wandernde Front ganz nahe an den linken Rand des Gebietes heran und wird dort durch die konstante Dirichlet-Randbedingung abrupt zum Stillstand gebracht. Nachdem sich die Gestalt der Lösung verändert hat, wird das Problem

dann zeitlich stationär. Diese zeitliche Entwicklung der Lösung ist in Abbildung 4.7 gezeigt, wobei die Lösung (auf dem Grobgitter) an allen Integrationszeitpunkten dargestellt ist. Man sieht, daß nach einer kurzen Einschwingphase die Integrationsschrittweiten dank der mitbewegten Gitter rasch sehr groß werden. Da sich hier der Einfluß des festen Randes noch nicht in einer anwachsenden Dynamik in der Gittergleichung niederschlägt, wird auch einmal eine Schrittweite geschätzt, die wesentlich zu groß ist. Die dadurch erzwungene starke Schrittweitenreduktion ist deutlich an den nahe beieinander liegenden Lösungen zu erkennen. Nach dann zunächst wieder größer werdenden Schrittweiten führt die von der Wand ausgehende Hemmung der Gitterbewegung zu einer rasch kleiner werdenden Geschwindigkeit der Knotenbewegung und durch diese Dynamik werden die verwendeten Schrittweiten nun wieder kleiner. Außerdem ändert sich noch die Gestalt der Lösung, so daß die Schrittweiten auch bei festem Gitter klein wären. Nachdem die Lösung stationär geworden ist, stellt sich auch in der Gittergleichung ein Gleichgewicht zwischen dem noch immer herrschenden Druck, wegen der steilen Front weiterhin die Knoten nach links zu bewegen, und der vom festen Rand ausgehenden, und von der Regularisierung weitergegebenen, Bewegungshemmung ein. Da dies in der gewählten Gittergleichung ein stabiler Zustand ist, können die Zeitschrittweiten nun wieder rasch groß werden.

Der Knotenfluß zeigt, daß in einer Anfangsphase sehr viele Knoten aus den unkritischen Bereichen der Lösung eliminiert werden. Danach brauchen nur noch relativ wenig Knoten eingefügt bzw. eliminiert zu werden, da sich die Gestalt der Front kaum ändert und durch die Knotenmitbewegung auch die Werte an den einzelnen Knoten kaum variieren. Der gegen Ende erreichte stationäre Zustand schlägt sich auch in einem nicht mehr veränderten Gitter nieder.

Ein derartig gutes Zusammenspiel der einzelnen Steuerungen läßt sich nicht immer erreichen. Aber die in Kapitel 6 exemplarisch dargestellten Ergebnisse für einige sehr anspruchsvolle Probleme zeigen, daß das Verfahren nicht nur für die bisher zur Illustration verwendeten “Spielprobleme” robust und effizient arbeitet.

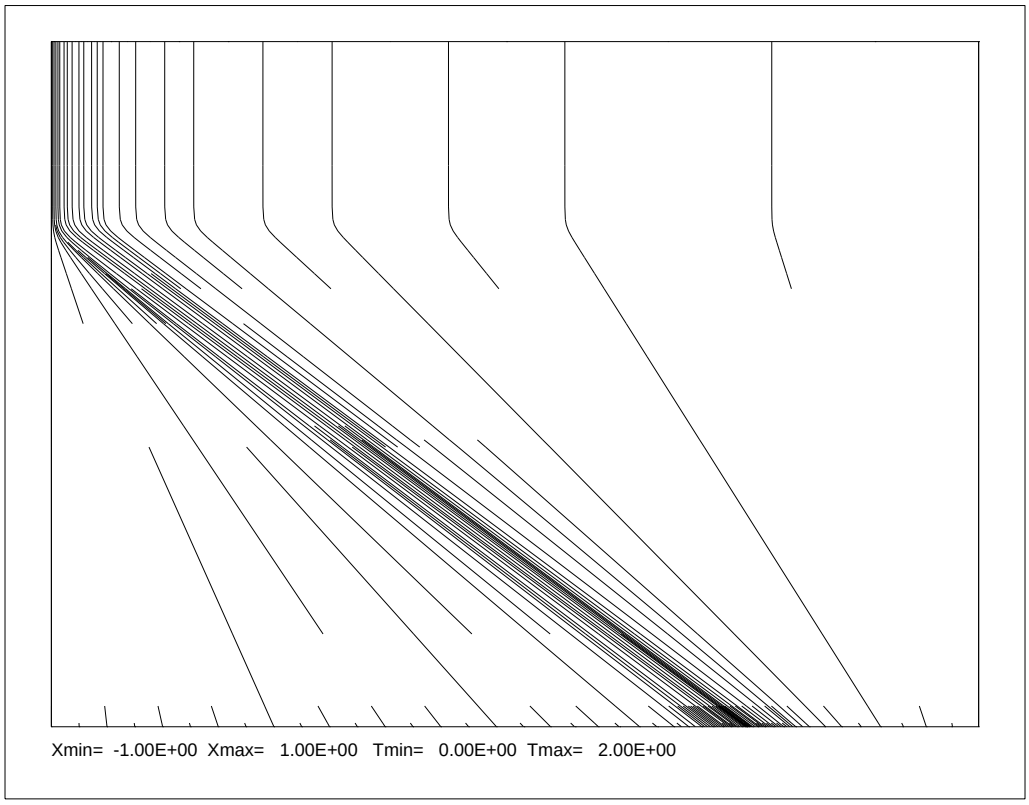


Abbildung 4.8: Knotenfluß für Testproblem **tanh–2** in einer x–t–Ebene

5. Programmpaket PDEX1M

Das in den vorangehenden Kapiteln hergeleitete und diskutierte Verfahren zur adaptiven Lösung von Problemen des Typs (2.1–2.6) ist in dem Programmpaket PDEX1M implementiert. Dabei steht PDEX1M für **P***arabolic* **D***ifferential* **E***quation e***X***trapolation solver in* **1***–D with (optionally)* **M***oving grid techniques.* Das sechsbuchstabige Kürzel deutet an, daß es sich dabei um eine Implementierung in FORTAN77 handelt.

Ehe auf Aspekte der Implementierung eingegangen werden kann, müssen noch einige für die praktische Anwendbarkeit des Verfahrens sehr wichtige Fragen diskutiert werden. Bei der Formulierung des Verfahrens in der Sprache der Mathematik wurden implizit einige Annahmen gemacht, z.B. daß die Norm "geeignet gewählt" ist, die bei einer Realisierung des Verfahren beachtet werden müssen.

5.1 Details der algorithmischen Realisierung

5.1.1 Wahl der Norm und interne Skalierung

Ein für die Problemstellung natürliche Wahl der Norm wäre z.B. eine L_2–Norm:

$$||u(x,t)|| := \sqrt{\int_{x=x_L}^{x=x_R} u(x,t)^2 dx} \; . \tag{5.1}$$

Deren numerische Auswertung im Verfahren ist etwa durch eine Approximation mit der Trapezregel möglich. Dem Charakter des hier verwendeten Diskretisierungsverfahrens ist dagegen eine rein knotenorientierte Norm adäquat. Da die Norm differenzierbar sein soll, bietet sich die Wahl einer l_2–Norm für den Vektor $\mathbf{u}(t)$ an, d.h.

$$||u(x,t)|| := ||\mathbf{u}(t)|| := \sqrt{\sum_{i=1}^{n_x} \sum_{j=1}^{n_{pde}} u_{j,i}^2} \; . \tag{5.2}$$

Im Gegensatz zu (5.1) ist diese Norm invariant gegen Umskalieren der unabhängigen Variablen x und t. Allerdings ist der Wert der Norm (5.2) nicht unabhängig von der Anzahl der Knoten des gewählten Gitters. Außerdem ist der Wert nicht invariant gegen komponentenweises Umeichen der abhängigen Variablen ($u_j \to s_j \cdot u_j$). Deshalb wird anstelle von (5.2) eine mittelnde und gewichtete (skalierte) Norm verwendet:

$$||\mathbf{u}(t)|| := \sqrt{\frac{1}{n_x \cdot n_{pde}} \sum_{i=1}^{n_x} \sum_{j=1}^{n_{pde}} \left(\frac{u_{j,i}}{u_{j,i}^w} \right)^2} \; . \tag{5.3}$$

Der Vektor $\mathbf{u}^w$ ist dabei ein noch zu spezifizierender Wichtungsvektor (Vektor mit internen Skalierungsgrößen).

Als lokale Norm an einem Knoten x_i wird, in Analogie zu (5.3),

$$||\mathbf{u}_i(t)|| := \sqrt{\frac{1}{n_{pde}} \sum_{j=1}^{n_{pde}} \left(\frac{u_{j,i}}{u^w_{j,i}} \right)^2}$$

verwendet. Die Werte des Wichtungsvektors $\mathbf{u}^w$ werden durch folgende interne Skalierungsprozedur bestimmt.

Zu Beginn der Integration:

$$u^w_{j,i} := \max\{|u_j(x_i, t_0)|, u^s_j\}$$
$$u^s_j := \text{ Skalierungschwellwerte (vom Anwender vorzugeben)}$$

Während eines Integrationsschrittes:

$$u^w_{j,i} := \max\{|U_j(x_i, t)|, u^w_{j,i}\}$$
$$U := \text{ beste vorliegende Lösungsapproximation}$$

Nach jedem Integrationsschritt:

$$u^w_{j,i} := \max\{|U_j(x_i, t)|, u^w_{j,i}\}$$
$$U := \text{ akzeptierte Feingitterlösung } T_{2,1,k,k}$$

Dies stellt i.w. die für steife Extrapolationsintegratoren typische interne Skalierungsprozedur dar. Die gewünschte Invarianz des Integrationsverhaltens gegenüber Umeichen der abhängigen Variablen ist damit zwar nicht vollständig, aber in den meisten Fällen ausreichend, realisiert. Mehrere Varianten dieser Standardskalierung sind im Programmpaket PDEX1M optional einsetztbar, sollen aber hier nicht weiter beschrieben werden. Alle in dieser Arbeit vorgestellten numerischen Rechnungen wurden mit der oben angegebenen internen Skalierungsprozedur durchgeführt. Für die nach statischem Regriddung evtl. neu hinzugekommenen Knoten des Gitters wird der zugehörige Skalierungswert durch Interpolation gewonnen.

Abschließend soll noch kurz auf die Skalierung der Moving–Grid–Gleichung eingegangen werden. Es wird die Gleichung (4.26.b) ersetzt durch

$$\dot{v}_i^T(\Delta_x v_i) - \lambda(\Delta_{xx}\dot{x}_i) = 0$$

mit

$$v_{j,i} := \frac{u_{j,i}}{u^w_{j,i}} \ .$$

Damit ist die Gleichung zwar noch abhängig von den für x und t verwendeten Einheiten, ihre Lösung, und damit das mitbewegte Gitter, ist jedoch invariant gegen komponentenweises Umskalieren in u bzw. x und t.

5.1.2 Lineare Algebra

Die bei der Zeitintegration zu lösenden linearen Gleichungssysteme haben bei der gewählten Nummerierung der Unbekannten eine Block–Tridiagonalgestalt wie sie in Abbildung 5.1 illustriert ist. Man könnte die Systeme zwar mit einem speziellen Blocklöser behandelt, dies wird sich jedoch nur lohnen, wenn die einzelne Blockgröße vergleichsweise sehr groß ist. Statt dessen wird in PDEX1M eine Gaußsche LU–Zerlegung mit partieller Pivotsuche im Bandmodus durchgeführt. Dazu werden die Routinen DGBFA und DGBSL des LINPACK–Paketes [25] verwendet. Für den allgemeinen Fall ergeben sich die

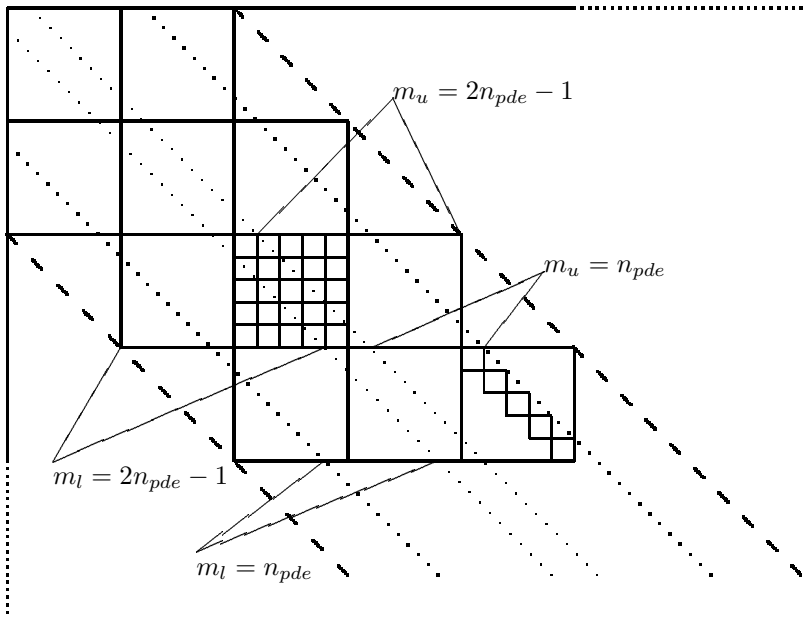


Abbildung 5.1: Matrixstruktur der linearen Gleichungssysteme ($n_{pde} = 5$)

untere (m_l) und obere (m_u) Bandbreite als

$$m_l = m_u = 2n_{pde} - 1 \; . \tag{5.4}$$

Im Systemfall hat das zu lösende Problem oft eine spezielle Struktur. Typischerweise hängt die j–te PDG nur von Ortsableitungstermen der Komponente u_j selbst ab. Dann haben die Nebendiagonalblöcke der Gesamtmatrix selbst Diagonalgestalt. Die maximal notwendigen Bandbreiten, um alle Nichtnullelemente der Gesamtmatrix zu erfassen, reduzieren sich zu

$$m_l = m_u = n_{pde} \; . \tag{5.5}$$

Um dieser Tatsache gerecht zu werden, hat der Benutzer von PDEX1M die Möglichkeit, die intern automatisch gemäß (5.4) gewählten Bandbreiten zu

korrigieren, d.h. (5.5) zu verwenden. Man sollte den dadurch zu erzielenden Beschleunigungseffekt nicht unterschätzen. Der Aufwand einer Bandmodus–Zerlegung wächst linear in der Dimension des Gesamtsystems und quadratisch in der Gesamtbandbreite m_b ($m_b = m_l + m_u + 1$). Zu beachten ist, daß die obige Voraussetzung zur Bandbreitenreduktion im Falle eines mitbewegten Gitters in jedem Fall verletzt ist, da die Knotenwerte in allen Ortsdiskretisierungstermen auftreten.

5.1.3 Berechnung der Jacobimatrix

Zur Zeitintegration wird die Jacobimatrix A des Residuums, d.h.

$$A := \frac{\partial r(u)}{\partial u} \ , \ \text{ mit } r := f - B\dot{u}$$

benötigt. Sie (bzw. die des semi–diskreten Systems) wird intern durch eine Approximation mittels finiter Differenzen gewonnen (numerische Differentiation). Auf die Möglichkeit, daß der Benutzer von PDEX1M eine analytisch gegebene Jacobimatrix des kontinuierlichen Problems zur Verfügung stellen kann, wird aus praktischen Gründen verzichtet. Im vorliegenden Fall würde dies bedeuten, daß der Benutzer neben den das Problem beschreibenden Funktionen (i.w. f, B) nun zusätzlich auch die partiellen Ableitungen dieser Funktionen nach allen Argumenten $u, u_x, (D(u)u_x)_x$ sowie D_u bereitstellen müßte. Da dies z.T. bereits Tensoren sind, wäre dieses Vorgehen sehr unhandlich und auch ineffizient in der anschließenden Auswertung.

Im allgemeinen ist die hier gewählte Zeitdiskretisierung robust gegen kleine Störungen in der Jacobimatrix. Daher wird die exakte Matrix ohnehin meist nur benötigt, um die Auswertung eventuell zu beschleunigen. Bei einer Realisierung wie oben angegeben, ist dies nicht zu erwarten.

Eine Schnittstelle, über die die Jacobimatrix des semi–diskreten Systems direkt zur Verfügung gestellt werden kann, ist jedoch vorhanden. Die Bereitstellung dieser Jacobimatrix erfordert allerdings Kenntnis der intern verwendeten Ortsdiskretisierungsschemata.

Da die interne numerische Differentiation im Bandmodus (simultane Auslenkung voneinander unabhängiger Komponenten) erfolgt, wird zur Berechnung der Matrix A nur eine Anzahl von m_b Auswertungen von $\mathbf{f}^{\Delta}, \mathbf{B}^{\mathbf{\Delta}}$ benötigt. Die einzelnen Bandbreiten, d.h. m_l und m_u, sind dabei entweder durch (5.4) oder (5.5) gegeben. Zwar wäre auch eine Realisierung möglich, die die Blockstruktur der Matrix berücksichtigt, dies würde aber nur im ersteren Fall zu einer Reduktion um $n_{pde} - 1$ Auswertungen führen. Da das Verfahren insgesamt vergleichsweise wenig Auswertungen der Jacobimatrix benötigt, wurde bisher auf eine Implementierung der numerische Differentiation im Bandmodus verzichtet.

Natürlich erfolgt auch die numerische Differentiation in geeignet skalierter Form, so daß die Approximationsgüte i.w. invariant gegen Umskalieren der Variablen ist.

5.2 Programmstruktur

5.2.1 Allgemeines

Das Programmpakete PDEX1M ist in ANSI–FORTRAN77 implementiert und somit auf praktisch jedem Rechner mit FORTRAN77–Compiler unverändert einsetzbar.

Nach außen hin bietet das Programmpaket PDEX1M mehrere Benutzerschnittstellen. Hier soll allerdings nur kurz auf die Standard–Benutzerschnittstelle eingegangen werden. In ihrer Art ähnelt diese Schnittstelle den Schnittstellen wie sie bei numerischer Software zur Lösung von gewöhnlichen Differentialgleichungen oder nicht ortsadaptiven Linienmethodencodes verwendet werden. Hierbei hat der Benutzer einige Unterprogramme zur Verfügung zu stellen, in denen die Problemstellung, d.h. die Funktionen f, B der PDG, die Matrix D der Diffusionskoeffizienten, und die Funktionen bzw. Konstanten $\alpha, \beta, \gamma, \delta$ der Randbedingungen, implementiert ist. Diese Unterprogramme, das Anfangsfeingitter und die zugehörigen Anfangswerte $u(x_i, t_0)$ sind dann, neben weiteren Input–Argumenten wie etwa verlangte Genauigkeit und Skalierungswerte, der Input für die Schnittstellenroutine PDEX1M, die von einem benutzergeschriebenen Hauptprogramm aufzurufen ist.

Das hier vorgestellte Lösungsverfahren, und dabei insbesondere die Moving-Grid–Technik, besitzt eine Vielzahl von algorithmischen Varianten und Steuerparametern. Durch deren Anpassung kann die Effizienz der numerischen Lösung von speziellen Problemklassen deutlich gesteigert werden. Die Wahl der Parameter und der Varianten geschieht durch eine entsprechende Setzung in sog. Optionsvektoren. Da eine sinnvolle Anpassung dieser Optionen nur bei Kenntnis des algorithmischen Verhaltens des Verfahrens erfolgen kann, bietet das Programmpaket eine Vielzahl von Monitorfunktionen an, die optional eingeschaltet werden können.

Eine mit der Rechnung simultan arbeitende graphische Ausgabe (Lösung, Gitter und algorithmische Indikatoren) steht ebenfalls optional zur Verfügung. Fast alle in dieser Arbeit gezeigten Abbildungen von Lösungsverläufen und Gitterdarstellungen wurden mit diesem Graphikpaket erzeugt. Das Paket kann auch zur graphischen Nachbehandlung eingesetzt werden. Es setzt auf die am Konrad–Zuse–Zentrum für Informationstechnik Berlin entwickelten graphischen Toolbox “Minigrafik” auf, und ist somit unter mehreren Windowsystemen ablauffähig.

Das Paket PDEX1M hat einen insgesamt modularen Aufbau. Technische Dinge wie Workspace–Verwaltung, Datenausgabe und graphische Darstellung finden außerhalb der numerischen Kernroutine statt. Leicht zu trennende algorithmische Teilaufgaben, wie etwa Normberechnung, lokales Regridding und natürlich die lineare Algebra werden in eigenen Modulen durchgeführt. Auf eine, im Prinzip wünschenswerte, Modularisierung aller algorithmischen Komponenten des Verfahrens wurde allerdings verzichtet. Sind nämlich die Einzelteile stark miteinander verwoben, so bringt eine rein fomale Modularisierung keinen echten Vorteil – insbesondere wenn das Verfahren in einer nicht objektorientierten Sprache wie FORTRAN realisiert ist. Dieser prinzipielle Nachteil der Programmiersprache FORTRAN bedeuted für den hier realisierten örtlich eindimensionalen Fall allerdings noch keine wesentliche Behinderung bei der Implementierung.

5.2.2 Standard–Benutzerschnittstelle

In diesem Abschnitt wird die oben angesprochene Standard–Benutzerschnittstelle kurz vorgestellt. Es handelt sich dabei um Auszüge aus dem Quell-Code des Gesamtpaketes. Eine detailliertere Beschreibung würde den Rahmen der Arbeit sprengen.

Die angegebenen Benutzerroutinen DFUN, PDEFUN, ABGFUN sind dabei knotenorientiert, d.h. sie werden für jeden Gitterpunkt (DFUN an den Intervallmittelpunkten) aufgerufen. Auf Vektorrechnern wird dadurch eine Vektorisierung der internen Routine verhindert, die die rechte und linke Seite ($\mathbf{f}^{\Delta}, \mathbf{B}^{\boldsymbol{\Delta}}$ bzw. $\tilde{\mathbf{f}}^{\Delta}, \tilde{\mathbf{B}}^{\boldsymbol{\Delta}}$) des semi–diskreten Systems aufstellt. Daher steht im Gesamtpaket PDEX1M z.B. auch eine vektorisierende Variante zur Verfügung, die dann allerdings Benutzerroutinen aufruft, in denen die benötigten Größen an allen Gitterpunkten gleichzeitig besetzt werden müssen.

```
      SUBROUTINE PDEX1M (NPDE,NX,NPDEMX,NXMAX,T,TEND,XL,XR,X,XDOT,
     1                   U,UDOT,BCFUN,DIFFUN,PROFUN,USCALE,TTOL,XTOL,
     2                   ROPT,IOPT,RWK,LRWK,IWK,LIWK,PRONAM,ICALL,IERR)
C
      DOUBLE PRECISION T,TEND,XL,XR,X,XDOT,U,UDOT,USCALE,TTOL,XTOL
      DOUBLE PRECISION ROPT,RWK
      INTEGER NPDE,NX,NPDEMX,NXMAX,IOPT,LRWK,IWK,LIWK,ICALL,IERR
      DIMENSION X(NPDEMX,NXMAX),XDOT(NPDEMX,NXMAX)
      DIMENSION U(NPDEMX,NXMAX),UDOT(NPDEMX,NXMAX)
      DIMENSION USCALE(NPDE),ROPT(50),RWK(LRWK)
      DIMENSION IOPT(50),IWK(LIWK)
      CHARACTER*12 PRONAM
      EXTERNAL BCFUN,DIFFUN,PROFUN
C
C
```

```
C* Parameters list description (* marks IN/OUT arguments)
C  ====================================================
C
C* External subroutines (to be supplied by the user)
C  ================================================
C
C     BCFUN,DIFFUN,PROFUN
C
C
C* Input arguments
C  ===============
C
C     NPDE           Int    Number of partial differential equations
C   * NX             Int    Initial number of space discretization points
C                           (fine grid)
C     NPDEMX         Int    Leading dimension of arrays u,udot
C     NXMAX          Int    Maximum dimension for a grid
C                           declared dimension of array x
C                           second dimension of arrays u,udot
C   * T              Dble   Starting point of integration
C     TEND           Dble   Final point of integration
C     XL             Dble   Left boundary
C     XR             Dble   Right boundary
C   * X(NXMAX)       Dble   Array for space discretization points
C                           X(1),...,X(NX) must hold the initial grid
C   * U(NPDEMX,NXMAX)Dble   Array for initial values at t on the
C                           initial grid
C   * UDOT(NPDEMX,NXMAX)
C                    Dble   Associated derivatives (may be zero in case
C                           of regular and constant left-hand side B)
C     USCALE(NPDEMX) Dble   User scaling (lower threshold) for the
C                           error norm used in PDEX1M
C     TTOL           Dble   Required relative precision for the time
C                           discretization
C     XTOL           Dble   Required relative precision for the space
C                           discretization
C   * ROPT(50)       Dble   Array of run-time options. Set to zero
C                           to get default values (details see below)
C   * IOPT(50)       Int    Array of run-time options. Set to zero
C                           to get default values (details see below)
C     PRONAM         C*12   Problem name
C     ICALL          Int    Indicator for type of call
C                           (=0: first call, details see below)
C
C* Output arguments
C  =================
C
C   * NX             Int    Current number of space discretization points
C                           (fine grid)
C   * T              Dble   Current time integration point
C   * X(NXMAX)       Dble   Array for space discretization points
```

```
C                          X(1),...,X(NX) hold the current (fine) grid
C   * U(NPDEMX,NXMAX)Dble  Array of solution values at t on the
C                          currentfine grid
C   * UDOT(NPDEMX,NXMAX)
C                    Dble  Associated derivatives
C   * ROPT(50)       Dble  Array of run-time options. Non zero default
C                          values ( induced by initial zero setting)
C                          are set.
C   * IOPT(50)       Int   Array of run-time options. Non zero default
C                          values ( induced by initial zero setting)
C                          are set.
C     IERR           Int   Error Code
C                          = 0 successfull completion of job,
C                              solution has been computed
C                          else: see list of error messages below
C
C* Workspace arguments
C  ===================
C
C  *  RWK(LRWK)      Dble  Real Workspace
C     LRWK           Int   Declared dimension of real workspace.
C  *  IWK(LIWK)      Int   Integer Workspace
C     LIWK           Int   Declared dimension of integer workspace.
C
C                    A test on sufficient workspace is made. If this
C                    test fails an error message
C                    is issued from which the minimum required
C                    workspace size can be obtained.
C
C                    The first elements of IWK and RWK can be used to
C                    pass integer and real parameter to the problem
C                    routines bcfun,dfun,pfun.
C                    The number of elements used for that purpose
C                    must be given in IOPT(49) and IOPT(50).
C


      SUBROUTINE DFUN (NDCL,NPDE,NX,IX,X,T,U,DVAL,
     1                 RPAR,LRPAR,IPAR,LIPAR,KFLAG)
C
      DOUBLE PRECISION X,T,U,DVAL,RPAR
      INTEGER NDCL,NPDE,NX,IX,LRPAR,IPAR,LIPAR,KFLAG
      DIMENSION U(NPDE),DVAL(NDCL,NDCL),RPAR(LRPAR),IPAR(LIPAR)
C
C=======================================================================
C
C  User routine for the local definition of the
C  diffusion coefficient matrix DVAL
C
C=======================================================================
C
C  Input arguments ( * marks IN/OUT arguments)
```

```
C-----------------------------------------
C
C     NDCL         declared dimension of local arrays and matrices
C     NPDE         dimension of partial differential equation
C     NX           current dimension of grid (number of nodes)
C     IX           position in current grid
C     X            current position in space (=X(IX+1/2)
C     T            current time
C     U(NDCL)      U(1:NPDE) holds the solution for the
C                  current (X,T)-values
C                  (linear interp. of  U(X(IX),T) and U(X(IX+1),T)
C     RPAR(LRPAR)  may contain real user parameter
C     LRPAR        dimension of array rpar
C     IPAR(LIPAR)  may contain integer user parameter
C     LIPAR        dimension of array ipar
C    * KFLAG       error indicator
C                  = 0  on input
C
C
C  Output arguments
C------------------
C
C     DVAL(NDCL,NDCL)
C                  DVAL(1:NPDE,1:NPDE) must hold the matrix of
C                  coefficients
C    * KFLAG       error code
C                  = 0 routine ended successfully
C                  < 0 error occured/detected in this routine
C
C=======================================================================
C

      SUBROUTINE PDEFUN (NDCL,NPDE,NX,IX,X,T,U,UX,DUXX,FFUN,BMAT,
     1                   RPAR,LRPAR,IPAR,LIPAR,KFLAG)
C
      DOUBLE PRECISION X,T,U,UX,DUXX,FFUN,BMAT,RPAR
      INTEGER NDCL,NPDE,NX,IX,LRPAR,IPAR,LIPAR,KFLAG
      DIMENSION U(NDCL),UX(NDCL),DUXX(NDCL,NDCL),FFUN(NDCL)
      DIMENSION BMAT(NDCL,NDCL),RPAR(LRPAR),IPAR(LIPAR)
C
C=======================================================================
C
C  User routine for the local definition of the right
C  and left hand side of the PDE.
C
C=======================================================================
C
C  Input arguments ( * marks IN/OUT arguments)
C-------------------------------------------
C
C     NDCL         declared dimension of local arrays and matrices
```

```
C     NPDE         dimension of partial differential equation
C     NX           current dimension of grid (number of nodes)
C     IX           position in current grid
C     X            current position in space
C     T            current time
C     U(NDCL)      U(1:NPDE) holds the solution for the
C                  current (X,T)-values
C     UX(NDCL)     U(1:NPDE) holds the first space derivative
C                  of the solution
C     DUXX(NDCL,NDCL)
C                  UDXX(1:NPDE,1:NPDE) holds the values of the
C                  discretized diffusion operator
C     RPAR(LRPAR)  may contain real user parameter
C     LRPAR        dimension of array rpar
C     IPAR(LIPAR)  may contain integer user parameter
C     LIPAR        dimension of array ipar
C    * KFLAG       error indicator
C                  = 0  on input
C
C
C  Output arguments
C-----------------
C
C     FFUN(NDCL)   FFUN(1:NPDE) must hold the right hand side vector
C                  of the PDE
C     BMAT(NDCL,NDCL)
C                  BMAT(1:NPDE,1:NPDE) must hold the left hand side
C                  matrix of the PDE
C     KFLAG        error code
C                  = 0: routine ended successfully
C                  < 0: error occured/detected in this routine
C
C
C=======================================================================
C

      SUBROUTINE ABGFUN (NDCL,NPDE,NX,IX,X,T,U,ALPHA,BETA,GAMMA,
     1                   DELTA,RPAR,LRPAR,IPAR,LIPAR,KFLAG)
C
      DOUBLE PRECISION X,T,U,ALPHA,BETA,GAMMA,DELTA
      INTEGER NDCL,NPDE,NX,IX,LRPAR,IPAR,LIPAR,KFLAG
      DIMENSION U(NDCL),ALPHA(NDCL),BETA(NDCL,NDCL),GAMMA(NDCL)
      DIMENSION DELTA(NDCL)
C
C=======================================================================
C
C  User routine for the local definition of the diagonal matrices
C  ALPHA,BETA,GAMMA (or the vector DELTA) describing the
C  boundary conditions.
C
C=======================================================================
```

```
C
C  Input arguments ( * marks IN/OUT arguments)
C-----------------------------------------------
C
C     NDCL          declared dimension of local arrays and matrices
C     NPDE          dimension of partial differential equation
C     NX            current dimension of grid (number of nodes)
C     IX            position in current grid
C                   IX=1 indicates left boundary
C                   IX=NX indicates right boundary
C     X             current position in space
C     T             current time
C     U(NDCL)       U(1:NPDE) holds the solution for the
C                   current (X,T)-values
C     RPAR(LRPAR)   may contain real user parameter
C     LRPAR         dimension of array rpar
C     IPAR(LIPAR)   may contain integer user parameter
C     LIPAR         dimension of array ipar
C    * KFLAG        error indicator
C                   = 0  on input
C
C
C  Output arguments
C------------------
C
C     ALPHA(NDCL)   ALPHA(1:NPDE) must contain the diagonal elements
C                   of the BC matrix alpha
C     BETA(NDCL)    BETA(1:NPDE) must contain the diagonal elements
C                   of the BC matrix beta
C     GAMMA(NDCL)   GAMMA(1:NPDE) must contain the diagonal elements
C                   of the BC matrix gamma
C     DELTA(NDCL)   DELTA(1:NPDE) must contain the vektor function
C                   deltae for ODE-type BC
C    * KFLAG        error code
C                   = 0: routine ended successfully
C                   < 0: error occured/detected in this routine
C
C
C=======================================================================
C
```

6. Numerische Ergebnisse

Bei der Angabe des benötigten Aufwands zur Lösung eines Problems werden die üblichen Aufwandsindikatoren verwendet, d.h. es wird die Gesamtrechenzeit und die Anzahl der Auswertungen der wesentlichen funktionalen Einheiten angegeben. Ihre Bedeutung ist in Tabelle 6.1 zusammengefaßt. Mit Ausnahme von $nstep$ wird bei der Zählung nicht zwischen einer Auswertung (Ausführung) bei Grob– bzw. Feingitterintegration unterschieden, sondern es werden alle Auswertungen unterschiedslos berücksichtigt. Als weiterer Indikator ist die durchschnittliche Knotenzahl des Feingitters angegeben:

$$\bar{n}_x^F := \frac{\sum_{l=0}^{nstep-1} n_{x,l}^F}{nstep}$$

wobei

$$\begin{aligned} n_{x,l}^F := &\quad \text{Feingitterknotenzahl des Integrationsschrittes } t_l \to t_{l+1} \\ nstep := &\quad \text{Anzahl der (erfolgreichen) Gesamtintegrationsschritte .} \end{aligned}$$

Die CPU–Zeit bezieht sich auf eine Workstation der Fa. SUN vom Typ SPARCstation IPC (Abschnitt 6.1 und 6.2) bzw. SPARCstation IPX (Abschnitt 6.3). Es wurde der SUN-FORTRAN-Compiler (Version 1.4.1) mit Standardoptionen und Optimierungsstufe $OPT = 1$ verwendet. Auf die Wahl einer höheren Optimierungsstufe mußte verzichtet werden, da für $OPT > 1$ kein stabiles Verhalten der Optimierungsgüte gegen algorithmisch bedeutungslose Änderungen am Quellcode zu beobachten waren.

Indikator	Bedeutung
$nstep$	Anzahl der (erfolgreichen) Integrationsschritte
$njac$	Anzahl der Jacobi–Matrix Auswertungen
$nfcnj$	Anzahl der $(\mathbf{f}^\Delta, \mathbf{B}^\Delta)$–Auswertungen für num. Differentiation
$nfcn$	Anzahl der sonstigen $(\mathbf{f}^\Delta, \mathbf{B}^\Delta)$–Auswertungen
$ndec$	Anzahl der LU–Zerlegungen
$nsol$	Anzahl der Vorwärts/Rückwärts–Substitutionen
$\bar{n}_x^F$	durchschnittliche Knotenzahl des Feingitters
CPU	benötigte Gesamtrechenzeit

Tabelle 6.1: Bedeutung der Aufwandsindikatoren

6.1 Ein Effizienzvergleich

Ein sehr schwieriges Testproblem stellt das folgende Modell eines diffusiven Verbrennungsprozesses mit Einschrittreaktion dar. Die Temperaturgleichung lautet:

TESTPROBLEM **hot–spot**:

$$\begin{aligned} a) \quad & x \in [0,1] \ , \ \ t \in [0, 0.29] \\ b) \quad & T_t = T_{xx} + D(1+\alpha - T)e^{-\delta/T} \\ c) \quad & T(x,t_0) = 1 \\ d) \quad & T_x(x_L,t) = 0 \ , \ \ T(x_R,t) = 1 \end{aligned} \tag{6.1}$$

mit der Damköhler Zahl

$$D = R\frac{e^{\delta}}{\alpha\delta}$$

und den Parameterwerten

$$\alpha = 1 \ , \ \ \delta = 30 \ , \ \ R = 5 \ .$$

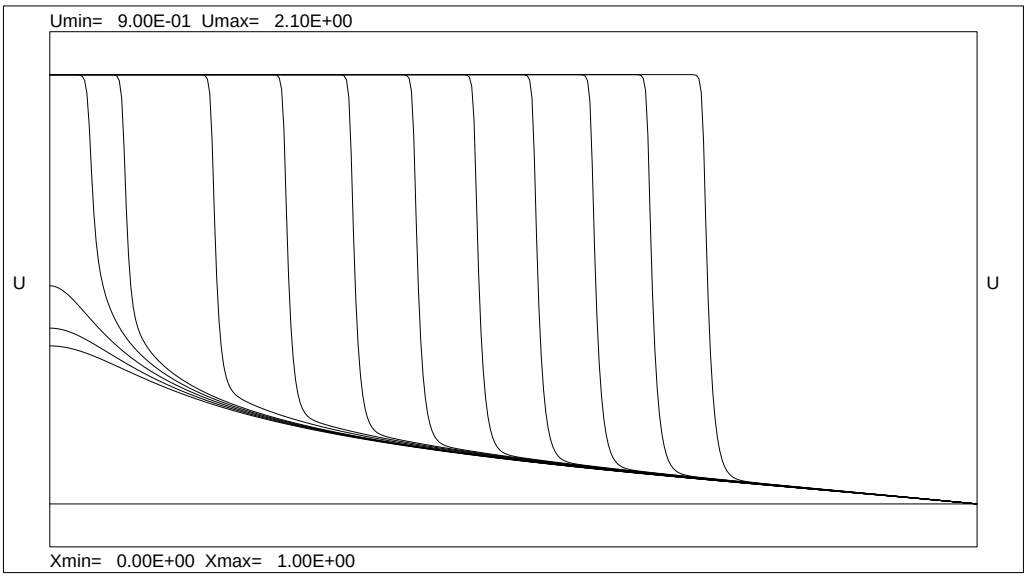


Abbildung 6.1: **hot–spot**: Lösung an einigen ausgewählten Zeitpunkten

In dieser Form ist die Brennstoffmenge durch $Y = (1+\alpha-T)/\alpha$ gegeben, und es braucht keine partielle Differentialgleichung für Y gelöst zu werden. Auf die Variante des Problems, in der auch eine PDG für die Brennstoffmenge

mitberücksichtigt ist (vgl. etwa [44], [2]), soll hier nicht eingegangen werden. Der Charakter des Problems ändert sich dadurch nur unwesentlich. In der hier angegebenen Form wird das Problem z.B. in [67] als Testbeispiel verwendet. Die Lösung des Problems an einigen geeignet gewählten Zeitpunkten t_j ist in Abbildung 6.1 dargestellt (t_j = 0.0, 0.2400, 0.2401, 0.2402, 0.2403,0.2404, 0.2408, 0.2412, ..., 0.2436, 0.2440). Vergleicht man die gewählten Zeitpunkte und die zugehörigen Positionen, so erkennt man, daß das Problem extrem kritisch ist. Nach einer langsamen Anlaufphase beginnt die Temperatur am linken Rand explosionsartig anzusteigen. Nachdem das Temperaturmaximum erreicht ist, beginnt die Temperaturfront sehr rasch nach rechts zu wandern. In dieser Anfangsphase der Bewegung zeigt die Frontgeschwindigkeit ein ausgeprägtes Maximum. Anschließend wandert die Front mit einer deutlich niedrigeren und dabei langsam abnehmenden Geschwindigkeit zum rechten Rand.

Prüft man die Genauigkeit einer numerischen Lösung nur für die in (6.1.a) angegebene Endzeit, so sind Fehler, die in der kritischen Anfangsphase auftreten ($t \approx 0.2403$), nicht mehr erkennbar. Daher soll hier ein spezielles numerisches Experiment durchgeführt werden.

Statt der Endzeit $t_{end} = 0.29$ wird die Endzeit $t_{end} = 0.244$ vorgegeben. Die Güte einer numerischen Lösung wird dadurch bewertet, daß die Genauigkeit der Frontposition zu diesem Zeitpunkt bestimmt wird. Mit Hilfe einer sehr genauen Referenzlösung erhält man:

$$T(x_f, 0.244) \stackrel{!}{=} 1.8 \Rightarrow x_f = 0.71053... \ .$$

Zuerst soll der Aufwand gefunden werden, den eine klassische Linienmethode mit Zeitschrittweitensteuerung aber nur äquidistantem, festem Gitter benötigt, um die Frontposition mit einer gewissen Genauigkeit zu berechnen.

	DASSL				EULSIM			
tol_t/n_x	101	201	401	801	101	201	401	801
10^{-3}	–	–	–	–	–	–	–	–
10^{-4}	–	–	–	–	–	○	○	●
10^{-5}	–	○	○	○	–	○	$\bullet^E$	●
10^{-6}	–	○	$\bullet^D$	●	–	○	●	●

Tabelle 6.2: Genauigkeit der Frontposition bei Zeitintegration mit Linienmethode

Dazu werden die Integratoren DASSL und EULSIM in Verbindung mit der auch in PDEX1M verwendeten Ortsdiskretisierung gewählt. Das Problem wird dann mit verschiedenen vorgegebenen Zeitintegrationsgenauigkeiten und

immer feiner werdenden Gittern gelöst. Die numerische Frontposition x_f^N wird dann durch Interpolation ermittelt. Der Interpolationsfehler ist dabei kleiner als 1%. Das Resultat des Experiments ist in Tabelle 6.2 zusammengefaßt. Die Bedeutung der Symbole ist wie folgt:

$$\begin{aligned} \bullet &: \; x_f^{err} \leq 1\% \\ \circ &: \; x_f^{err} \leq 10\% \\ - &: \; x_f^{err} > 10\% \; . \end{aligned}$$

Dabei ist unter x_f^{err} der relative Fehler der numerischen Frontposition zu verstehen:

$$x_f^{err} := |x_f - x_f^N|/x_f \; .$$

Die jeweils am wenigsten aufwendige Integration ist durch $\bullet^D$ bzw. $\bullet^E$ hervorgehoben. Für die dabei benötigte CPU–Zeit ergibt sich:

$$\bullet^D : 1156 \text{ Sek.} \;\; , \;\; \bullet^E : 862 \text{ Sek.} \; .$$

Löst man das modifizierte Problem mit PDEX1M, wobei zunächst die im folgenden Abschnitt angegebenen Standardoptionen verwendet werden ($tol_x = 2.5 \cdot 10^{-3}$, $tol_t = 1.0 \cdot 10^{-3}$, so zeigt sich bereits deutlich der Gewinn an Effizienz und Sicherheit den eine in Raum und Zeit adaptive und kontrollierte Integration besitzt. Mit einem Aufwand von nur 71 Sekunden Rechenzeit wird die Frontposition mit einer Abweichung von nur 7% berechnet (Eintrag V_S in Tabelle 6.3). Bei der vergleichbaren Linienmethoden–Integration mit $tol = 10^{-3}$ ist der Fehler auf allen Gittern mindestens 30%.

PDEX1M		
Variante	CPU	x_f^{err}
V_S	71	6.8%
V_2	34	7.0%
V_3	133	0.9%
V_4	74	0.3%
V_5	57	0.7%

Tabelle 6.3: Genauigkeit der Frontposition bzw. Aufwand bei Integration mit PDEX1M–Varianten

Bei der Integration mit PDEX1M meldet der Integrationsmonitor deutliche Diskrepanzen in den Fehlerschätzungen auf Grob– bzw. Feingitter. Das in Abschnitt 4.3.2 angesprochene Problem der Scheindynamik tritt auf. Schaltet man daher auf die abgeschwächte Fehlerkontrolle um, so zeigt sich eine

beschleunigte Integration bei praktisch gleicher Genauigkeit (Eintrag V_2 in Tabelle 6.3).

Experimentiert man nun noch etwas mit den Genauigkeitsforderungen und Gittermitbewegungsgleichungen für PDEX1M, so lassen sich rasch auch genauere Frontpositionen bei weiterhin sehr geringen Rechenzeiten erzielen. Exemplarisch sind die Resultate von drei dieser Integrationen in Tabelle 6.3 angegeben. Bei allen drei Rechnungen wurde eine Toleranzvorgabe von $tol_x = 2.5 \cdot 10^{-4}$, $tol_t = 1.0 \cdot 10^{-4}$ gewählt.

Bei V^3 wurde nochmals die Standardgittermitbewegung gewählt.

Bei V_4 wird die Gittermitbewegung durch eine Ankopplungsgleichung der Form (4.41) – mit $\bar{u} = 1.8$ – durchgeführt. Die Bestimmung des "Frontknotens" i_f erfolgt automatisch. Alle anderen Knoten werden durch eine diskretisierte Form der Gleichung (4.22) bewegt.

Bei der letzten Variante (V_5) ist gegenüber V_4 nur die Behandlung der "Restknoten" geändert. Es wird dazu eine spezielle Realisierung von (4.44) verwendet.

Insgesamt läßt sich also eine drastische Effizienzsteigerung feststellen. Um die potentiellen Möglichkeiten, die gerade die Gittermitbewegung bietet, voll auszuschöpfen, bedarf es allerdings eines gewissen Fein–Tunings.

Den deutlichen Unterschied im Integrationsverhalten zwischen einer nur zeitadaptiven Linienmethode und einer Methode mit zusätzlich statischem und dynamischen Regridding illustrieren die Abbildungen 6.2 (EULSIM mit $tol = 10^{-4}, n_x = 401$) und 6.3 (PDEX1M–V_5). In beiden Fällen ist die Lösung an jeweils allen intern gewählten Integrationszeitpunkten dargestellt.

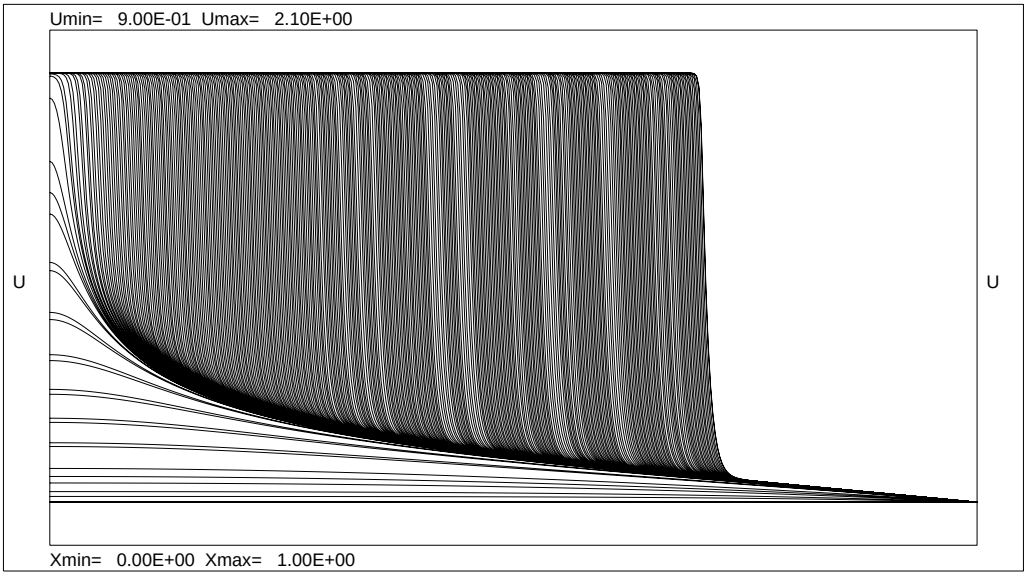


Abbildung 6.2: **hot–spot**: Lösung mit zeitadaptiver Linienmethode (EULSIM)

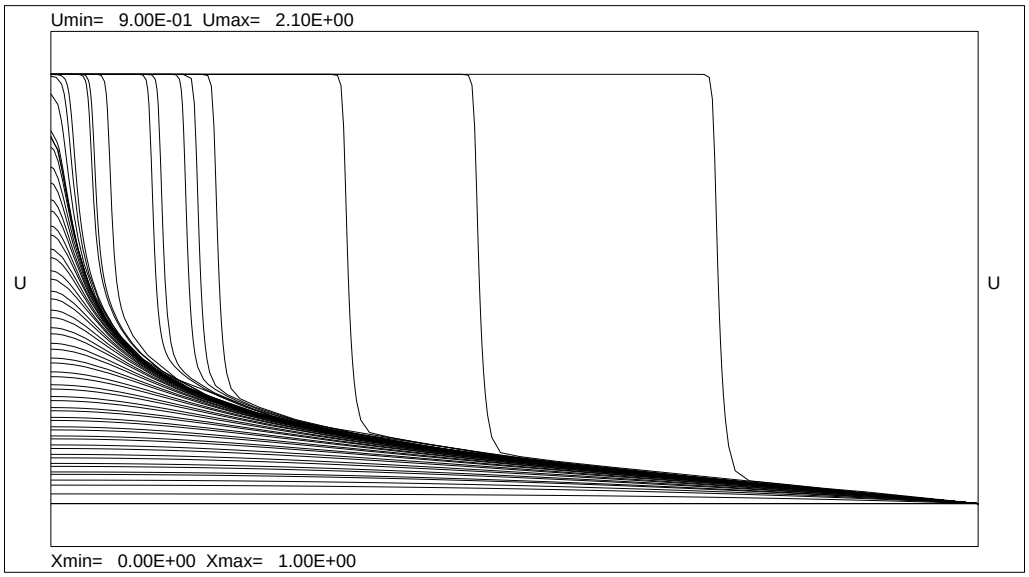


Abbildung 6.3: **hot–spot**: Lösung mit PDEX1M

6.2 Verhalten an typischen Testproblemen

In diesem Abschnitt soll nun insbesondere die Robustheit des Verfahrens bzw. die des Programmpaketes PDEX1M demonstriert werden. Dazu wird, ohne eine spezielle Adaptierung von Verfahrensparametern oder Wahl algorithmischer Varianten, eine möglichst breite Palette von Testproblemen gelöst. Es wird zunächst der bereits klassische Satz von 7 Testproblemen aus SINCOVEC/MADSEN [72] verwendet. Bei diesen Beispielen handelt es sich um typische Modellprobleme aus sehr unterschiedlichen Anwendungsbereichen. Die Probleme zeigen durchaus unterschiedliche Arten von Schwierigkeiten bei der numerischen Lösung. Anschließend werden einige häufig verwendete Testbeispiele mit wandernden Fronten gelöst.

Die Frage, ob zur Lösung ein mitbewegtes Gitter verwendet werden soll, ist dabei natürlich nicht generell zu beantworten, sondern ist vom Problem her vorab zu entscheiden. Wird jedoch ein mitbewegtes Gitter verwendet, so wird die allgemeine Gittermitbewegungsgleichung (4.26) mit dem Standardparameter $\lambda = 0.1$, vgl. (4.31), verwendet.

Andere Parameter sind jedoch von der Sache her problemabhängig und bedürfen eigentlich der Setzung durch den Benutzer der Software. Dennoch werden hier, von wenigen angezeigten Ausnahmen abgesehen, folgende Spezifikationen durchgängig verwendet:

- Anfangszeitschrittweite: $\Delta T_0 := 10^{-5}$
- Anfangsfeingitter: $n^F_{x,0} := 81$
- Skalierunsschwellwerte: $u^s_j := 1, j = 1, \ldots, n_{pde}$
- Verlangte Ortsgenauigkeit: $tol_x := 2.5 \cdot 10^{-3}$
- Verlangte Zeitgenauigkeit: $tol_t := 1.0 \cdot 10^{-3}$

Diese Kombination der verlangten Genauigkeiten reflektiert die Tatsache, daß die Zeitintegration wegen ihrer variablen Ordnung auf stringentere Genauigkeitsforderungen mit vergleichsweise geringerem Mehraufwand als die Ortsdiskretisierung reagiert. Da die Grobgitterdiskretisierung einen viermal größeren Fehler als tol_x besitzt, ist eine weniger genaue Vorgabe von tol_x bei einer "black box"–Anwendung von PDEX1M nicht zu empfehlen.

6.2.1 Die Beispiele von Sincovec/Madsen

TESTPROBLEM **SM–A**:

$$
\begin{aligned}
&a) \quad x \in [0,1]\ , \quad t \in [0, 1.1] \\
&b) \quad u_t = \epsilon u_{xx} - u u_x \\
&c) \quad u(x, t_0) = u_E(x, t_0) \\
&d) \quad u(x_L, t) = 1\ , \quad u(x_R, t) = 0.1\ .
\end{aligned}
$$

Die exakte Lösung dieser Burgers–Gleichung ist durch

$$u_E(x,t) = (0.1e^{-A} + 0.5e^{-B} + e^{-C})/(e^{-A} + e^{-B} + e^{-C})$$

mit

$$
\begin{aligned}
A &= (x - 0.5 + 4.95t)/(20\epsilon) \\
B &= (x - 0.5 + 0.75t)/(4\epsilon) \\
C &= (x - 0.375)/(2\epsilon)
\end{aligned}
$$

gegeben. Verwendet man den Parameterwert $\epsilon = 0.003$ aus [72], so ist genügend Diffusion im Problem, um es mit zentralen Differenzen sinnvoll diskretisieren zu können.

Da die Lösung eine wandernde Welle von variabler Gestalt darstellt, die am rechten Rand durch eine Dirichlet–Bedingung zum Stillstand gebracht wird, ist dies ein interessanter Test für die Güte der hier vorgeschlagenen allgemeinen Moving–Grid–Technik.

Der Vergleich zu einer Integration mit nur statisch angepaßtem Gitter findet sich in Tabelle 6.4. Es zeigt sich, daß bei Mitbewegung des Gitters die dadurch reduzierte Anzahl an Integrationsschritten durch die verdoppelte Problemdimension fast völlig kompensiert wird.

Die numerische Lösung an jeweils allen intern gewählten Integrationszeitpunkten ist in Abbildung 6.4 bzw. 6.6 dargestellt. Dabei ist wieder die Feingitterlösung an allen Grobgitterknoten dargestellt (wie in allen folgenden Lösungsdarstellungen). Das statisch angepaßte Grobgitter ist in Abbildung 6.5 illustriert. Das mitbewegte (Grob–) Gitter ist in Abbildung 6.7 gezeigt.

mog	*nstep*	*njac*	*nfcnj*	*nfcn*	*ndec*	*nsol*	$\bar{n}_x^F$	*CPU*
nein	46	112	336	716	429	1033	80	27.1
ja	19	38	266	141	159	420	79	22.1

Tabelle 6.4: Aufwand zur Lösung von Testproblem **SM–A**

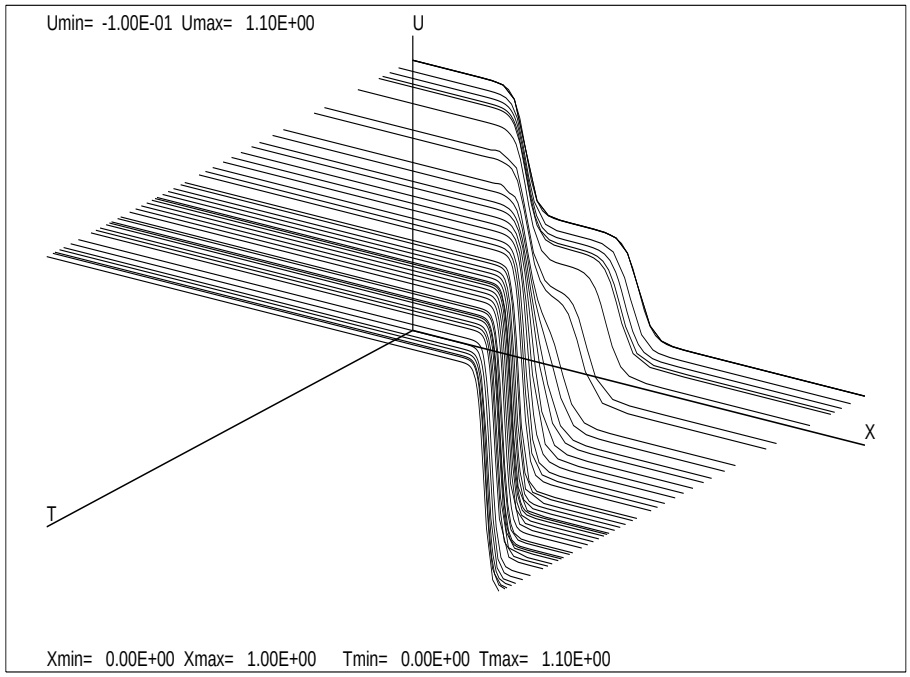


Abbildung 6.4: Lösung von Testproblem **SM–A**.

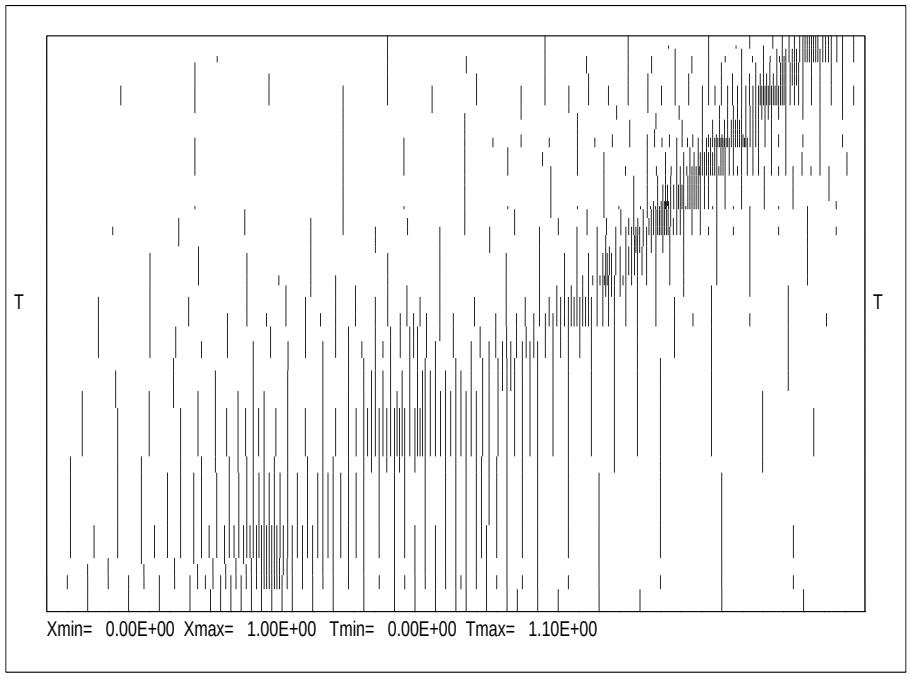


Abbildung 6.5: Nur statisch angepaßtes Gitter für Testproblem **SM–A**

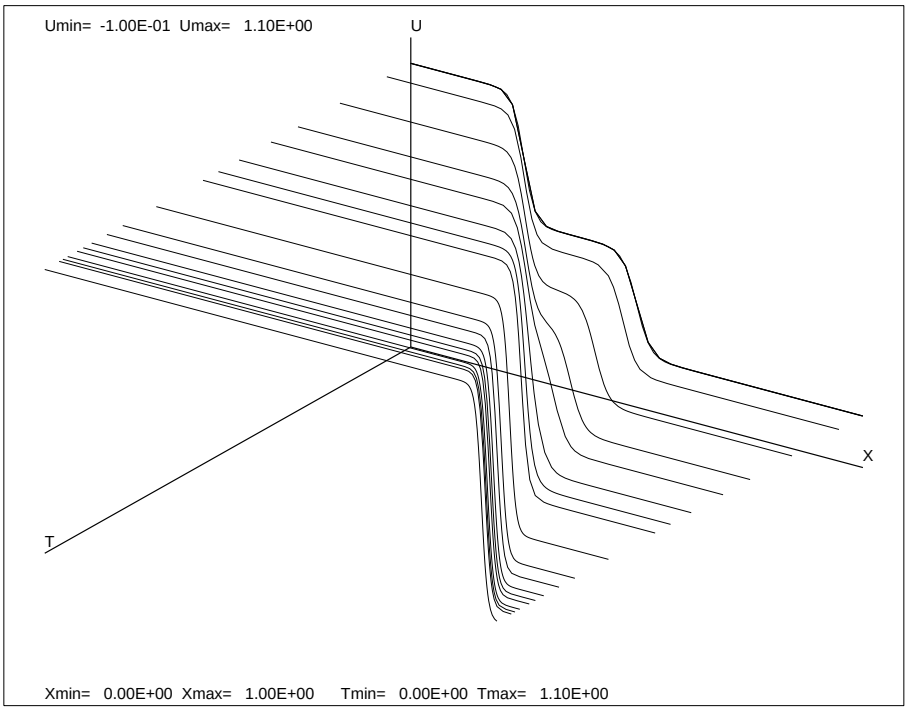


Abbildung 6.6: Lösung von Testproblem **SM–A** (bei mitbewegtem Gitter).

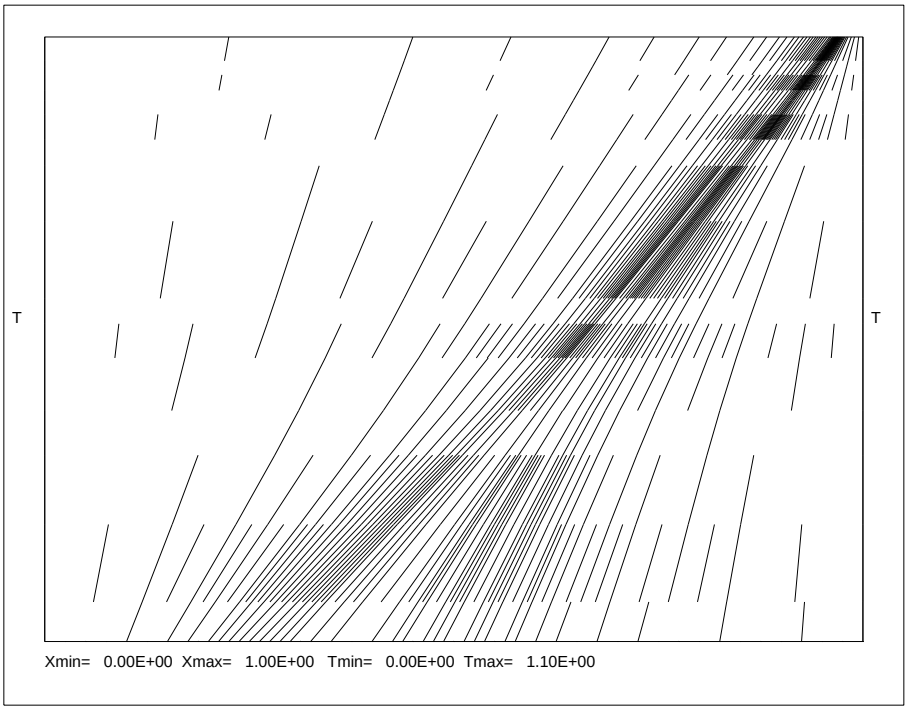


Abbildung 6.7: Mitbewegtes Gitter für Testproblem **SM–A**

TESTPROBLEM **SM–B**:

$$
\begin{aligned}
a) \quad & x \in [-400, 400] \ , \quad t \in [0, 1.83] \\
b) \quad & v_t = \nu v_{xx} - v v_x - g(w_x + H_x) \\
& w_t = \eta w_{xx} - v w_x - w v_x \\
c) \quad & v(x, t_0) = 40 \ , \quad w(x, t_0) = 20 - H(x) \\
d) \quad & v(x_L, t) = v(x_R, t) = 40 \ , \quad w(x_L, t) = w(x_R, t) = 20
\end{aligned}
$$

mit

$$
\begin{aligned}
& H = H(x) = \max\{0, 10 - 10(x/40)^2\} \\
& g = 980 \ .
\end{aligned}
$$

Bei diesen Modellgleichungen für ein Problem aus der Strömungsmechanik ("shallow water flow") handelt es sich nochmals um ein eigentlich hyperbolisches Problem. Um den Gleichungen wenigstens eine gewisse parabolische Natur zu geben, müssen kleine diffusive Terme hinzugefügt werden ("künstliche Viskosität"). In [72] werden allerdings keine Angaben über deren Größe gemacht. Mit der Wahl

$$\nu = \eta = 200 \tag{6.2}$$

lassen sich die in [72] dargestellten Lösungen zwar nicht ganz reproduzieren, aber man erhält ein für einen parabolischen Löser adäquates Problem.

In Abbildung 6.8 ist nicht nur die Lösung $v(x, t)$ (horizontale Strömungsgeschwindigkeit) an allen Integrationszeitpunkten, sondern auch das jeweils zugehörige Grobgitter dargestellt. Gleiches gilt für die Darstellung der Komponente $w(x, t)$ (Wassertiefe) in Abbildung 6.9.

mog	$nstep$	$njac$	$nfcnj$	$nfcn$	$ndec$	$nsol$	$\bar{n}_x^F$	CPU
nein	9	18	126	84	56	122	163	21.4

Tabelle 6.5: Aufwand zur Lösung von Testproblem **SM–B**

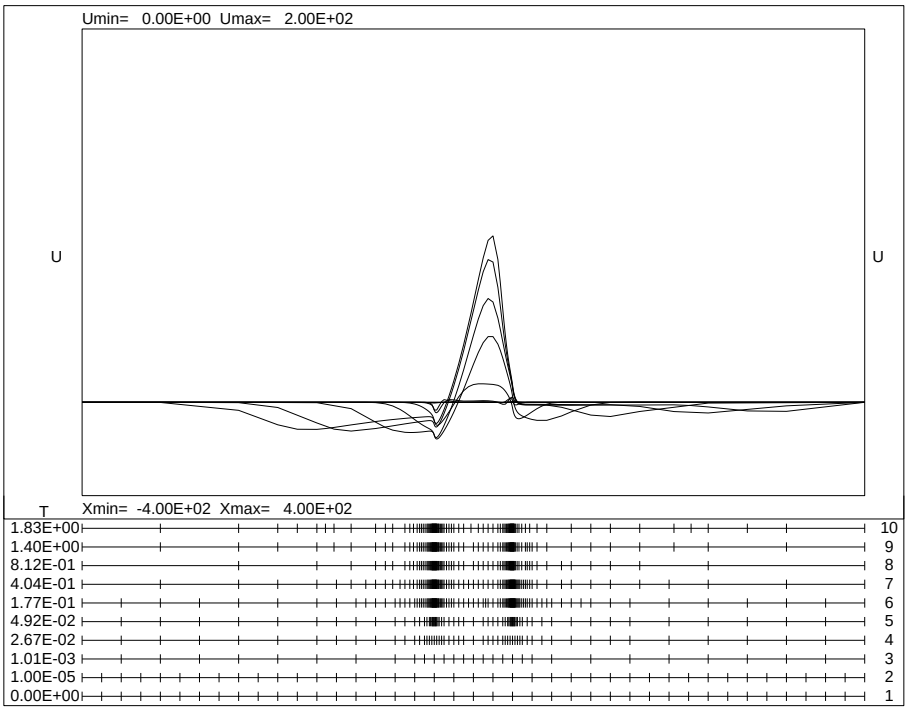


Abbildung 6.8: Lösung v von Testproblem **SM–B**

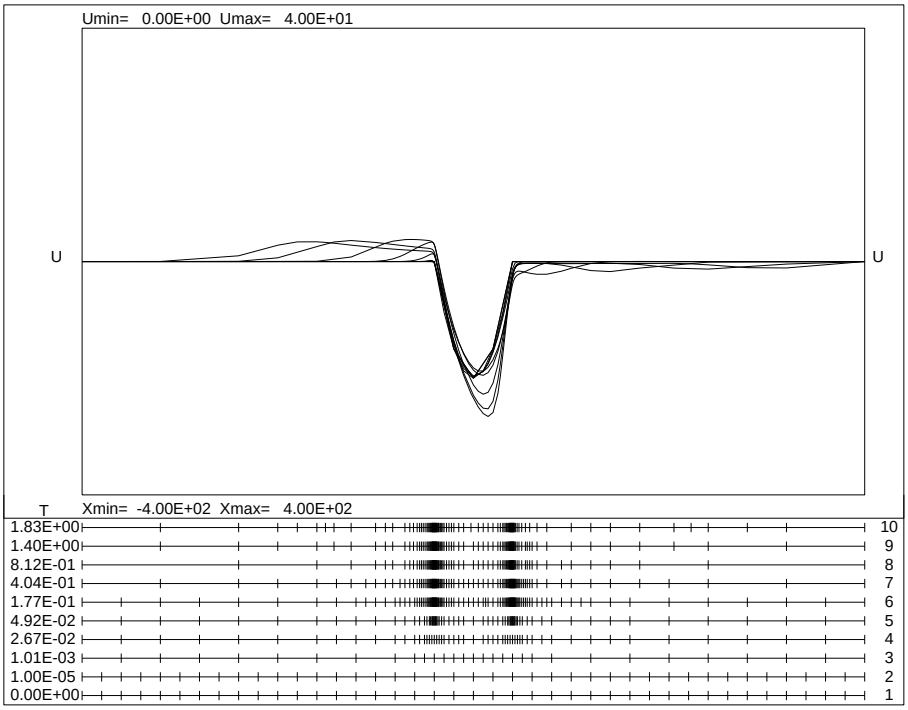


Abbildung 6.9: Lösung w von Testproblem **SM–B**

TESTPROBLEM **SM–C**:

$$
\begin{aligned}
a) \quad & x \in [0,1]\ , \quad t \in [0,1] \\
b) \quad & u_t = (D(x)u_x)_x - v(x)u_x \\
c) \quad & u(x,t_0) = 0 \ \text{ für } x > 0 \\
& u(x,t_0) = 1 \ \text{ für } x = 0 \\
d) \quad & u(x_L,t) = 1\ , \quad u(x_R,t) = 0
\end{aligned}
$$

mit

$$
D(x) = \begin{cases} 5 & \text{für} \quad 0 \quad \leq \quad x \quad < \quad 0.5 \\ 1 & \text{für} \quad 0.5 \quad < \quad x \quad \leq \quad 1 \end{cases}
$$

und

$$
v(x) = \begin{cases} 1000 & \text{für} \quad 0 \quad \leq \quad x \quad < \quad 0.5 \\ 1 & \text{für} \quad 0.5 \quad < \quad x \quad \leq \quad 1 \end{cases} \quad .
$$

Die Besonderheit dieser Diffusion–Konvektions–Gleichung liegt in den unstetigen Koeffizienten $D(x)$ bzw. $v(x)$ und der unstetigen Anfangsbedingung. Für den Wert $x = 0.5$ wird hier der einfache Mittelwert $v = 500.5$ verwendet. Ob bei der Mittelwertbildung eine Berücksichtigung der Größe der angenzenden Gitterintervalle sinnvoll ist, hängt vom physikalischen Hintergrund des Problems ab, und kann nicht im Verfahren entschieden werden. Der Koeffizient $D(x)$ wird an der Unstetigkeitsstelle nicht ausgewertet.

Obwohl diese partielle Differentialgleichung nicht die für die Anwendung des Verfahrens eigentlich notwendigen Differenzierbarkeitseigenschaften besitzt, ist das Verfahren robust genug, auch dieses Testbeispiel problemlos zu lösen. In Abbildung 6.10 sind Lösung und Gitter an einigen ausgewählten Integrationszeitpunkten dargestellt.

Zwar ist die Lösung (zunächst) eine wandernde Front, aber die Unstetigkeit der Funktionen $D(x)$ und $v(x)$ verbietet eine Anwendung der allgemeinen Moving–Grid–Technik.

Bei Integration dieses Problems wurde eine Anfangsschrittweite $\Delta T = 10^{-7}$ vorgegeben.

mog	$nstep$	$njac$	$nfcnj$	$nfcn$	$ndec$	$nsol$	$\bar{n}_x^F$	CPU
nein	28	62	186	326	210	474	71	12.8

Tabelle 6.6: Aufwand zur Lösung von Testproblem **SM–C**

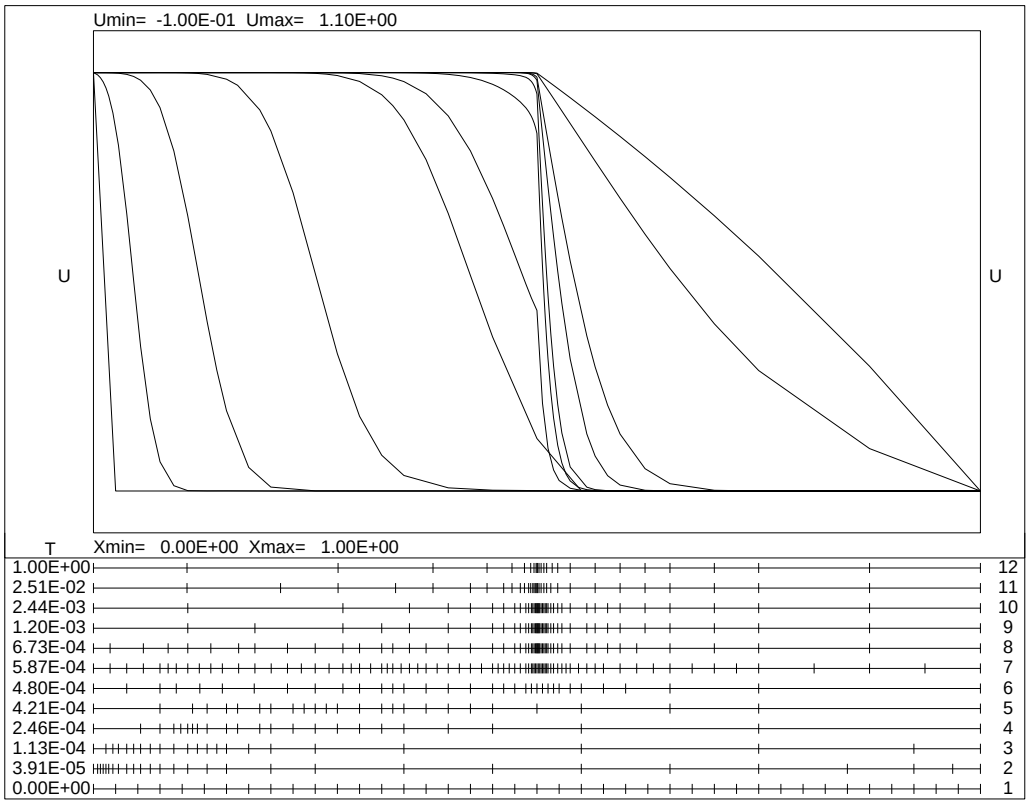


Abbildung 6.10: Lösung von Testproblem **SM–C**

TESTPROBLEM **SM–D**:

$$
\begin{aligned}
a) \quad & x \in [0,1]\ , \quad t \in [0, 0.1] \\
b) \quad & u_t = u_{xx} - 10 sinh(10u) \\
c) \quad & u(x, t_0) = 0 \ \text{ für } x < 1 \\
& u(x, t_0) = 1 \ \text{ für } x = 1 \\
d) \quad & u(x_L, t) = 0\ , \quad u(x_R, t) = 1
\end{aligned}
$$

Wiederum liegt ein Problem mit unstetiger Anfangsbelegung vor. Allerdings sind die Anfangswerte eigentlich unerheblich, da das Originalproblem (Tröschs Problem) die rein elliptische Randwertaufgabe ist. In [72] wird daraus jedoch die obige Anfangs–Randwertaufgabe konstruiert. Für die angegebene Endzeit ist der stationäre Zustand bereits erreicht.

Verwendet man zur Integration eine sehr kleine Anfangszeitschrittweite, etwa $\Delta T = 10^{-7}$, so wird ein durch den unstetigen Anfangszustand hervorgerufenes Problem deutlich. Da die Fehlerkontrolle neue Knoten am rechten Rand einfügt, läuft die Lösungsfront zunächst "rückwärts", d.h. auf den rechten Rand zu, und erst anschließend beginnt die Bewegung nach links.

Die in den Abbildungen 6.11 und 6.12 dargestellten Lösungen – wieder an allen Integrationszeitpunkten – lassen dieses Verhalten erkennen. Sie zeigen auch, daß die verlangte Genauigkeit, insbesondere in Verbindung mit dem gewählten Skalierungsschwellwert, recht grobe Gitter in einem eigentlich interessierenden Bereich der Lösung zuläßt. Ist man an einer genaueren Lösung interessiert, so kann man z.B. die Integration mit einem modifizierten Schwellwert $u_1^s = 10^{-2}$ wiederholen. Die zugehörige Lösung, nun an jedem 5. Integrationspunkt, ist in Abbildung 6.12 gezeigt. Auch hier läuft die Front zunächst noch nach rechts. Um dies zu vermeiden, müßte ein sehr viel feineres Anfangsgitter vorgegeben werden.

Die Aufwandsindikatoren in Tabelle 6.7 zeigen, daß die restriktivere Genauigkeitsforderung zu einem deutlich gestiegenen Integrationsaufwand führt.

scale	*nstep*	*njac*	*nfcnj*	*nfcn*	*ndec*	*nsol*	$\bar{n}_x^F$	*CPU*
S	9	18	54	60	48	90	66	3.1
R	25	52	156	290	182	420	111	18.4

Tabelle 6.7: Aufwand zur Lösung von Testproblem **SM–D**. Standardskalierung (S) und restriktive Skalierung (R)

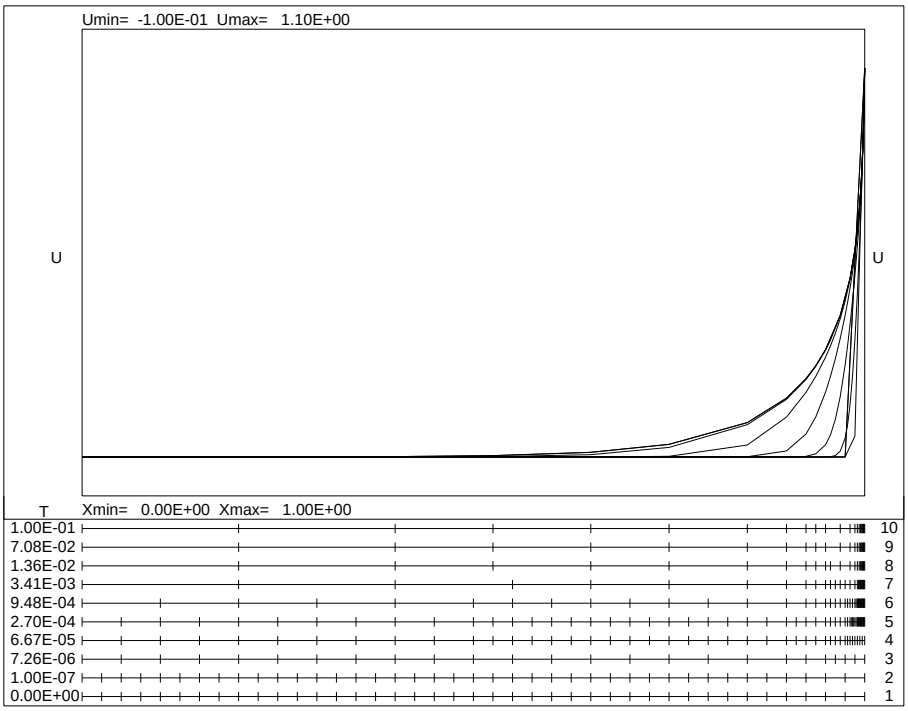


Abbildung 6.11: Lösung von Testproblem **SM–D** bei Standardskalierung

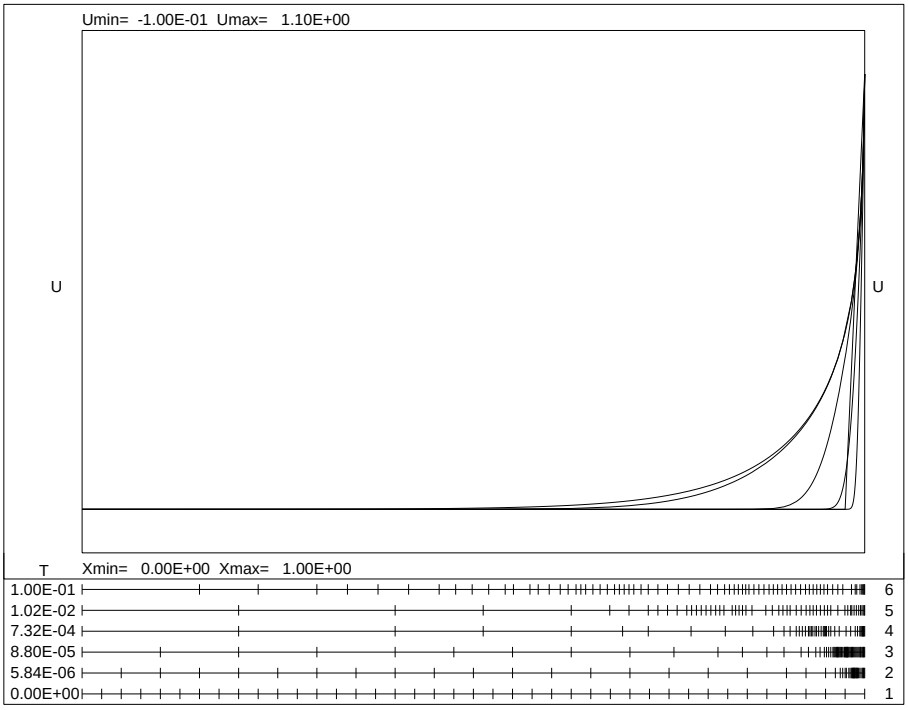


Abbildung 6.12: Lösung von Testproblem **SM–D** bei restriktiver Skalierung

TESTPROBLEM **SM–E**:

$$
\begin{aligned}
a) \quad & x \in [0,1] \ , \quad t \in [0,5] \\
b) \quad & u_t = x^{-1}(x u_x)_x \\
c) \quad & u(x,t_0) = 600 \\
d) \quad & u_x(x_L,t) = 0 \ , \quad u_x(x_R,t) = \phi(x_R,t,u)
\end{aligned}
$$

mit

$$\phi(x,t,u) = 1.73 \cdot 10^{-9}(6.25 \cdot 10^{10} - u(x,t)^4) \ .$$

Hierbei handelt es sich um eine einfache Wärmeleitungsgleichung, allerdings in Zylinderkoordinaten. Eine weitere Besonderheit ist die nichtlineare Randbedingung vom Neumann–Typ. Bei der numerischen Lösung mit PDEX1M ergeben sich keinerlei interessante Phänomene.

mog	$nstep$	$njac$	$nfcnj$	$nfcn$	$ndec$	$nsol$	$\bar{n}_x^F$	CPU
nein	7	14	42	56	40	82	47	1.7

Tabelle 6.8: Aufwand zur Lösung von Testproblem **SM–E**

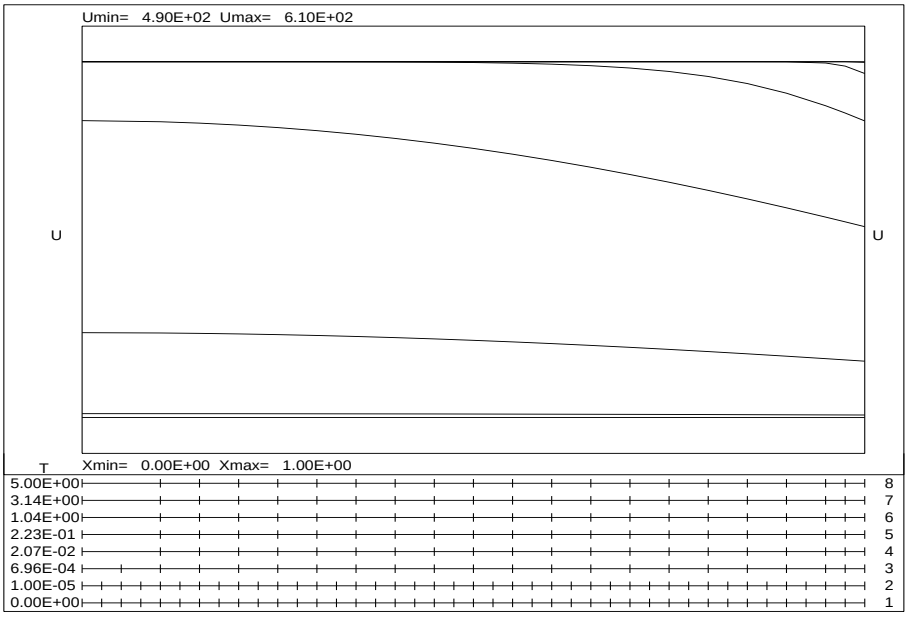


Abbildung 6.13: Lösung von Testproblem **SM–E**

TESTPROBLEM **SM–F**:

$$
\begin{aligned}
a) \quad & x \in [0,1]\ , \quad t \in [0, 10^{10}] \\
b) \quad & u_t = Du_{xx} + a_1 - a_2 u + a_3 v + a_4 y - a_5 uv + a_6 uy \\
& v_t = Dv_{xx} + b_1 - b_2 v + b_3 uv - b_4 vw \\
& w_t = Dw_{xx} - c_1 w + c_2 y + c_3 uy - c_4 vw + 800 + \mathrm{SST1}(x) \\
& y_t = Dy_{xx} - d_1 y + d_2 vw - d_3 uy + 800 \\
c) \quad & u(x,t_0) = 1.306028 \cdot 10^6 \\
& v(x,t_0) = 1.076508 \cdot 10^{12} \\
& w(x,t_0) = 6.457715 \cdot 10^{10} \\
& y(x,t_0) = 3.542285 \cdot 10^{10} \\
d) \quad & u_x(x_L,t) = v_x(x_L,t) = w_x(x_L,t) = y_x(x_L,t) = 0
\end{aligned}
$$

mit den Parameterwerten

$a_1 = 4 \cdot 10^5$, $a_2 = 272.443800016$, $a_3 = 10^{-4}$, $a_4 = 0.007$, $a_5 = 3.67 \cdot 10^{-16}$,
$a_6 = 4.13 \cdot 10^{-12}$

$b_1 = 272.4438$, $b_2 = 1.00016 \cdot 10^{-4}$, $b_3 = 3.67 \cdot 10^{-16}$, $b_4 = 3.57 \cdot 10^{-15}$

$c_1 = 1.6 \cdot 10^{-8}$, $c_2 = 0.007$, $c_3 = 4.1283 \cdot 10^{-12}$, $c_4 = 3.57 \cdot 10^{-15}$

$d_1 = 7.000016 \cdot 10^{-3}$, $d_2 = 3.57 \cdot 10^{-15}$, $d_3 = 4.1283 \cdot 10^{-12}$

$D = 10^{-9}$

$$
\mathrm{SST1}(x) = \begin{cases} 3250 & \text{für} \quad 0.475 \ \leq \ x \ \leq \ 0.575 \\ 360 & \text{sonst} \end{cases}
$$

Da das Verfahren PDEX1M intern in skalierten Größen arbeitet, machen sich die extremen Größenordungen dieses gekoppelten Systems von Reaktions–Diffusions–Gleichungen nicht negativ bemerkbar. Der unstetige Koeffizient SST1 erfährt keine Sonderbehandlung. Der in Abbildung 6.14 angegebene Lösunsverlauf der Komponente w ist typisch für das Verhalten aller Komponenten.

mog	$nstep$	$njac$	$nfcnj$	$nfcn$	$ndec$	$nsol$	$\bar{n}_x^F$	CPU
nein	14	28	420	106	76	154	104	55.2

Tabelle 6.9: Aufwand zur Lösung von Testproblem **SM–F**

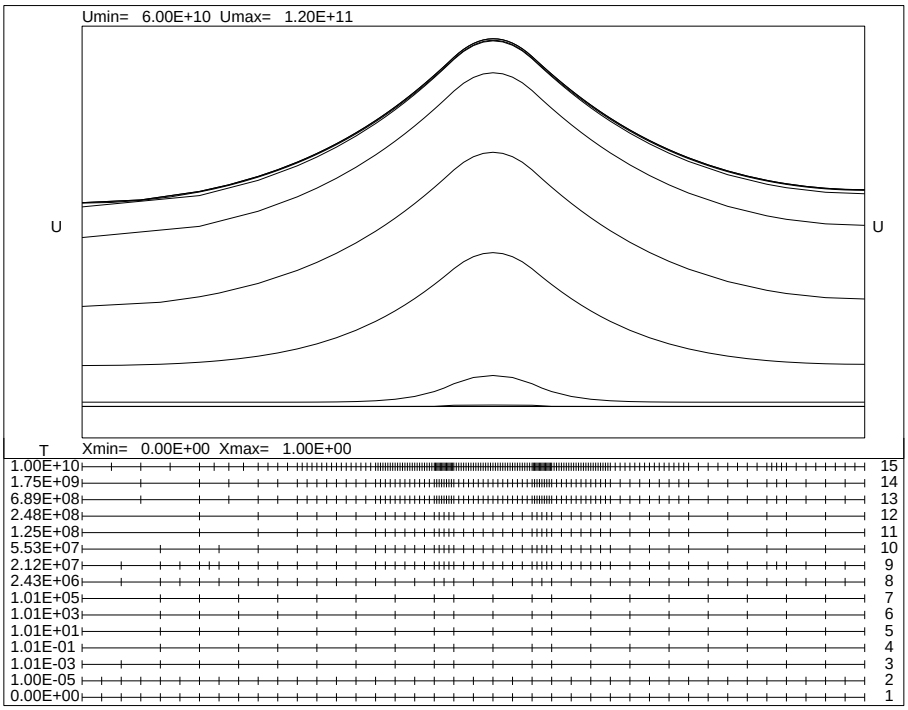


Abbildung 6.14: Lösungskomponente w von Testproblem **SM–F**

TESTPROBLEM **SM–G**:

$$
\begin{aligned}
a) \quad & x \in [0,1]\ , \quad t \in [0, 0.1] \\
b) \quad & u_t = (u u_x)_x - u^2 \\
c) \quad & u(x, t_0) = 100 \\
d) \quad & u(x_L, t) = 50\ , \quad u_x(x_R, t) = 1 - sin(u(x_R, t))
\end{aligned}
$$

Die Besonderheit dieses Problems ist im Auftreten eines lösungsabhängigen Diffusionsterms zu sehen. Nochmals tritt eine nichtlineare Randbedingung und eine unstetige Anfangsbelegung auf. Daher wird wieder eine Anfangsschrittweite von 10^{-7} verwendet. In Abbildung 6.15 ist die Lösung an allen Integrationspunkten mit $t \leq 3.4 \cdot 10^{-3}$ angegeben.

mog	$nstep$	$njac$	$nfcnj$	$nfcn$	$ndec$	$nsol$	$\bar{n}_x^F$	CPU
nein	15	30	90	124	90	184	48	3.8

Tabelle 6.10: Aufwand zur Lösung von Testproblem **SM–G**

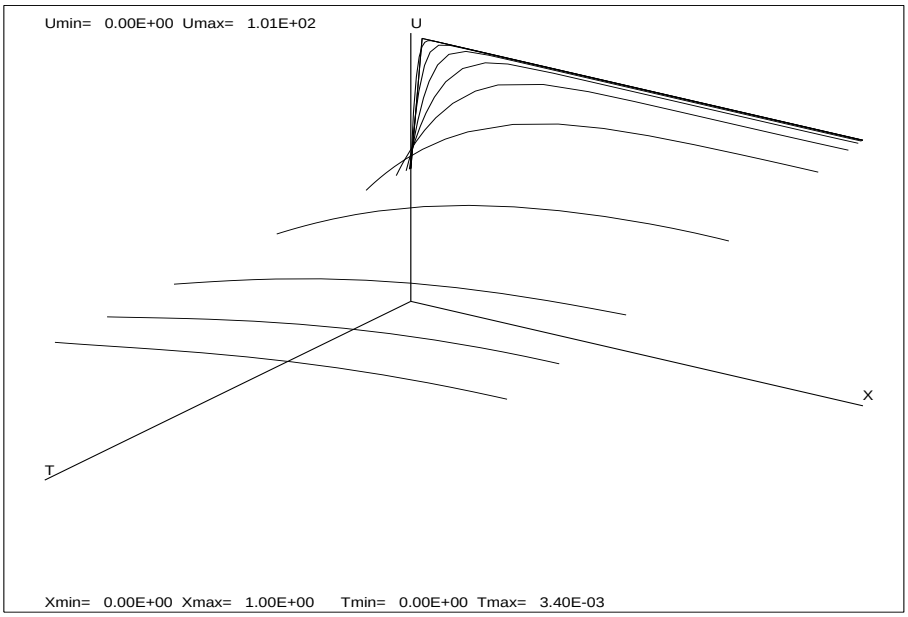


Abbildung 6.15: Lösung von Testproblem **SM–G**

6.2.2 Wandernde Fronten

TESTPROBLEM **nerve–pulse**:

$$
\begin{aligned}
a) \quad & x \in [0, 120] \ , \ \ t \in [0, 200] \\
b) \quad & u_t = u_{xx} + u(u - 0.139)(1 - u) - v \\
& v_t = 0.008(u - 2.54v) \\
c) \quad & u(x, t_0) = v(x, t_0) = 0 \\
d) \quad & u_x(x_L, t) = -0.225 \ \ , \ u_x(x_R, t) = 0
\end{aligned}
$$

Das Problem dient z.B. in [8, 78] als Testproblem. Das Modell besitzt zwei besondere Schwierigkeiten. Zum einen stellt eine der Gleichungen eine gewöhnliche Differentialgleichung dar. Nach Transformation auf mitbewegte Koordinaten liegt also eine formal hyperbolische Gleichung vor. Dies führt hier allerdings zu keinen Stabilitätsproblemen. Zum anderen tauchen am linken Rand immer neue Impulse auf, und dies führt zu einem nichttrivialen Lösungsverhalten hinter der sich nach rechts wegbewegenden Front.

mog	$nstep$	$njac$	$nfcnj$	$nfcn$	$ndec$	$nsol$	$\bar{n}_x^F$	CPU
ja	59	130	1430	712	794	1807	79	161.8

Tabelle 6.11: Aufwand zur Lösung von Testproblem **nerve–pulse**

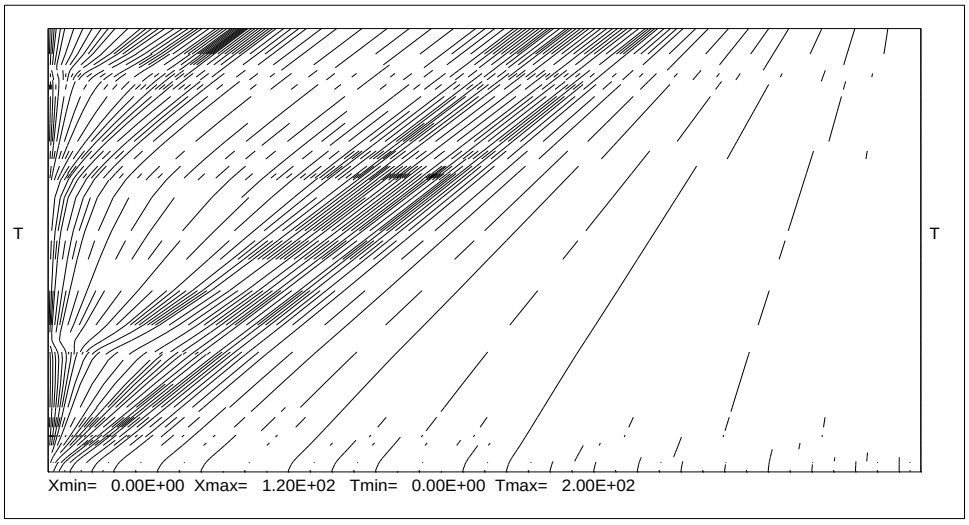


Abbildung 6.16: Mitbewegtes Gitter für Testproblem **nerve–pulse**

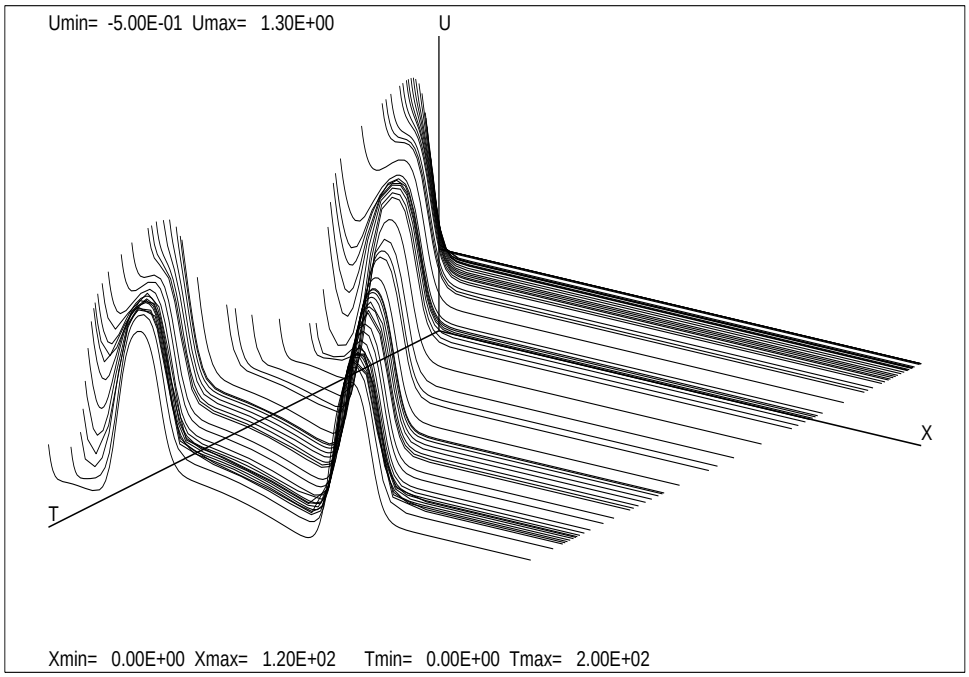


Abbildung 6.17: Lösungskomponente u von Testproblem **nerve–pulse**

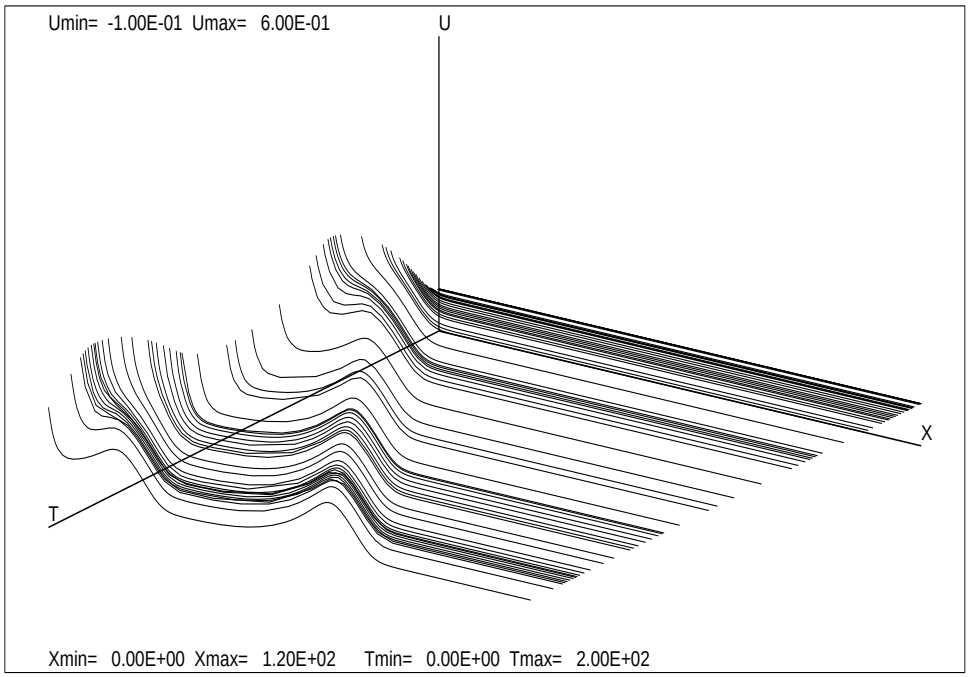


Abbildung 6.18: Lösungskomponente v von Testproblem **nerve–pulse**

TESTPROBLEM **slab–2**:

$$
\begin{aligned}
a) \quad & x \in [0,1] \ , \quad t \in [0, 0.025] \\
b) \quad & T_t = T_{xx} + \tfrac{\beta}{1+\beta} A_1 \rho e^{-\theta_1/T} + \tfrac{\beta}{1+\beta} A_2 \sigma e^{-\theta_2/T} \\
& \rho_t = \rho_{xx} - A_1 \rho e^{-\theta_1/T} \\
& \sigma_t = \sigma_{xx} + A_1 \rho e^{-\theta_1/T} - A_2 \sigma e^{-\theta_2/T} \\
c) \quad & T(x,t_0) = T_0 \ , \quad \rho(x,t_0) = 1 \ , \quad \sigma(x,t_0) = 0 \\
d) \quad & T_x(x_L,t) = \rho_x(x_L,t) = \sigma_x(x_L,t) = 0 \\
& T(x_R,t) = \min\{T_0 + Ct, T_f\} \\
& \rho_x(x_R,t) = \sigma_x(x_R,t) = 0
\end{aligned}
$$

Dieses Flammenmodell nach OTEY/DWYER [60] ist eine Erweiterung des sehr häufig verwendeten Testproblems von DWYER/SANDERS [27]. Es handelt sich um ein Modell mit einer Zweischritt–Reaktion. Die Paramter sind so gewählt, daß damit die Aufheizung eines aus dem Material ρ bestehenden Stabes simuliert wird, wobei der Stoff ρ dann durch einen exothermen Prozeß gemäß $\rho \to \sigma \to \psi$ zerfällt. Die Aufheizung wird durch eine einfache zeitabhängige Randbedingung modelliert. Die hier angegebenen Parameterwerte und die Integrationszeit entsprechen den von [59] verwendeten:

$$
\begin{aligned}
& A_1 = 2.2 \cdot 10^5, \quad A_2 = 2.2 \cdot 10^{10}, \quad \beta = 1, \\
& \theta_1 = 4, \qquad\qquad \theta_2 = 4, \\
& T_0 = 0.2, \qquad\quad T_f = 1.2, \qquad\quad C = 2000.
\end{aligned}
$$

Die Lösungen ρ (σ ist fast identisch Null, T ähnlich wie in Abbildung 6.21), zu allen Integrationszeitpunkten, findet sich in Abbildung 6.19; das statisch und dynamisch angepaßte Gitter in Abbildung 6.20.

mog	$nstep$	$njac$	$nfcnj$	$nfcn$	$ndec$	$nsol$	$\bar{n}_x^F$	CPU
ja	39	78	1170	301	289	929	61	107.1

Tabelle 6.12: Aufwand zur Lösung von Testproblem **slab–2**

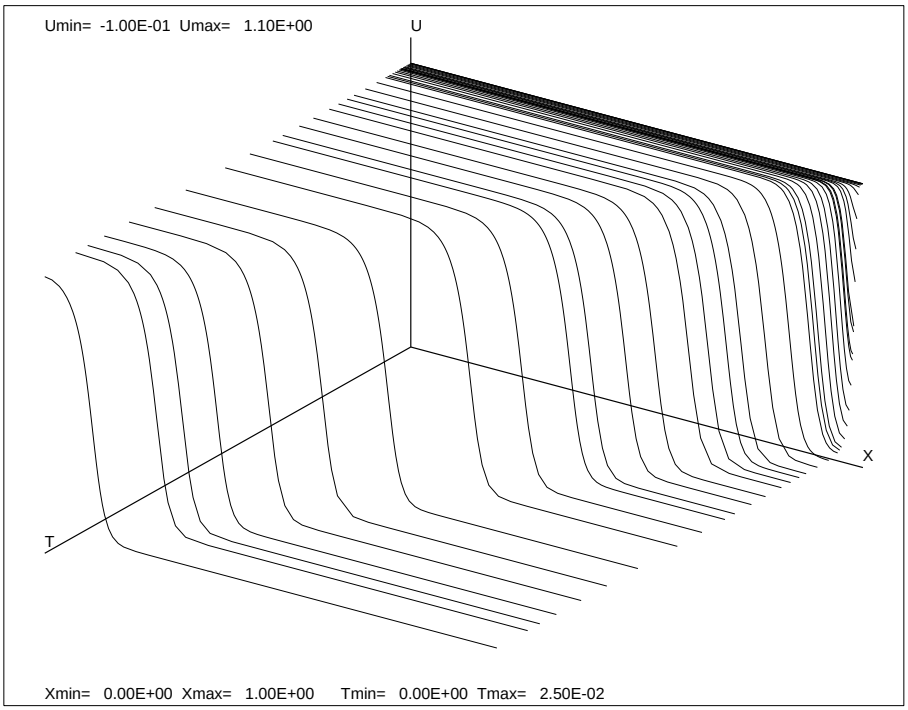


Abbildung 6.19: Lösungskomponente ρ von Testproblem **slab–2**

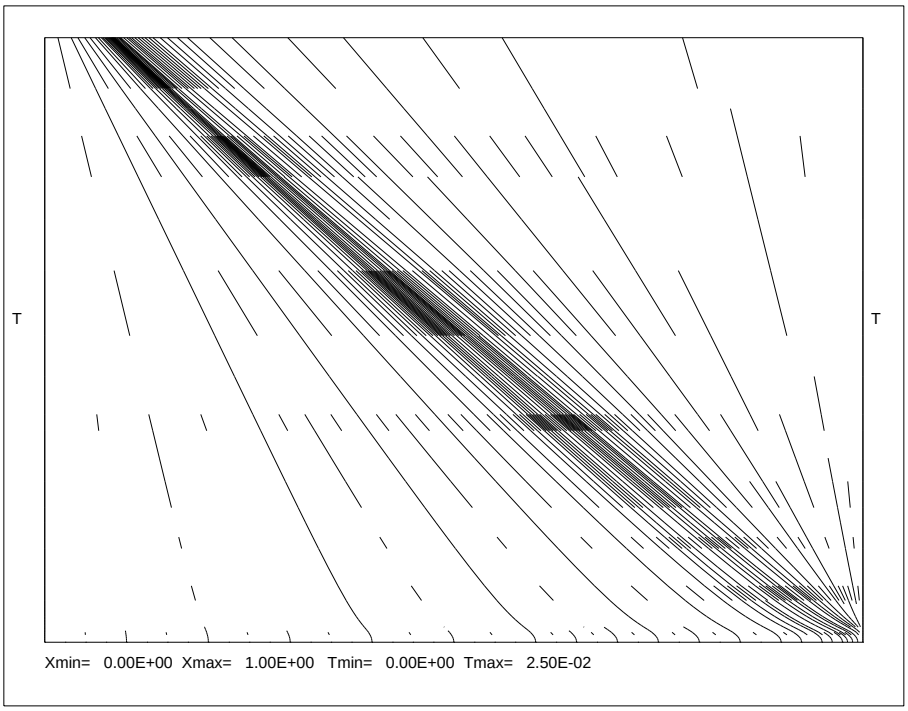


Abbildung 6.20: Mitbewegtes Gitter für Testproblem **slab–2**

TESTPROBLEM **slab–1**:

$$
\begin{aligned}
&a) \qquad x \in [0,1]\ , \quad t \in [0, 0.006] \\
&b) \qquad T_t = T_{xx} + A\rho e^{-\theta/T} \\
&\qquad\quad \rho_t = \rho_{xx} - A\rho e^{-\theta/T} \\
&c) \qquad T(x,t_0) = T_0\ , \quad \rho(x,t_0) = 1 \\
&d) \qquad T_x(x_L,t) = \rho_x(x_L,t) = 0 \\
&\qquad\quad T(x_R,t) = \min\{T_0 + Ct, T_f\} \\
&\qquad\quad \rho_x(x_R,t) = 0
\end{aligned}
\tag{6.3}
$$

Mit den Parameterwerten

$$
\begin{aligned}
&A = 3.52 \cdot 10^6, \quad \theta = 4, \\
&T_0 = 0.2, \qquad\quad T_f = 1.2, \quad C = 5000.
\end{aligned}
\tag{6.4}
$$

Es handelt sich um das eben erwähnte Testproblem aus [27]. Mit dieser Parameterwahl ist sowohl der Aufheizvorgang, als auch die Frontbewegung, deutlich schneller als im vorherigen Beispiel. Es handelt sich hierbei um ein außerst beliebtes Testproblem. Es wird z.B. in [8, 65, 66, 78, 77, 80] als Testbeispiel verwendet.

Die im Vergleich zu Problem **slab–2** noch schnellere Frontbewegung macht eine Lösung mit mitbewegtem Gitter sehr attraktiv.

mog	$nstep$	$njac$	$nfcnj$	$nfcn$	$ndec$	$nsol$	$\bar{n}_x^F$	CPU
ja	22	46	506	329	347	852	61	55.3

Tabelle 6.13: Aufwand zur Lösung von Testproblem **slab–1**

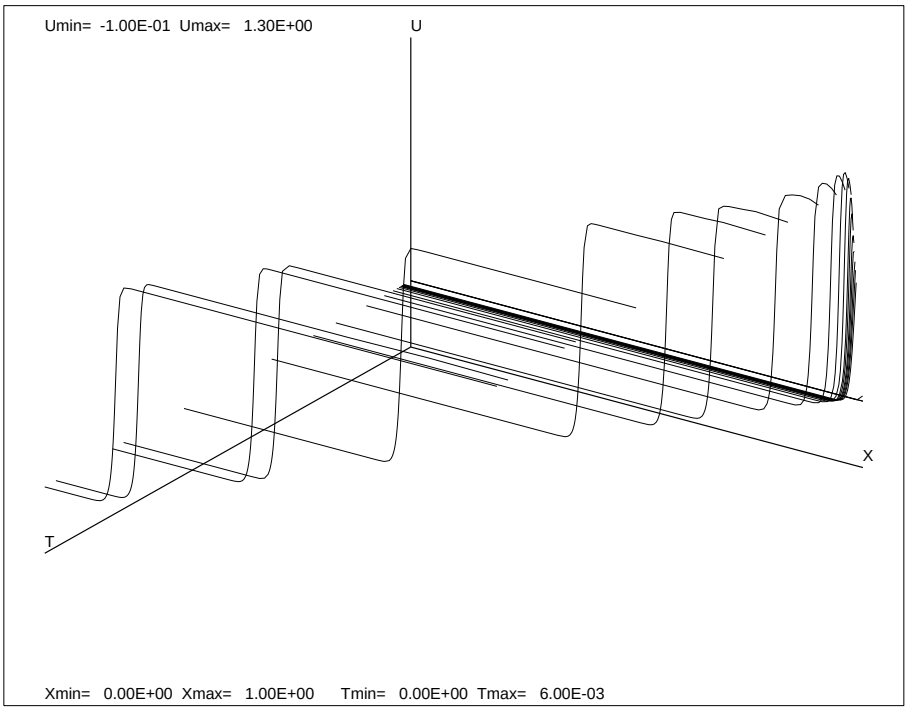


Abbildung 6.21: Lösungskomponente T von Testproblem **slab–1**

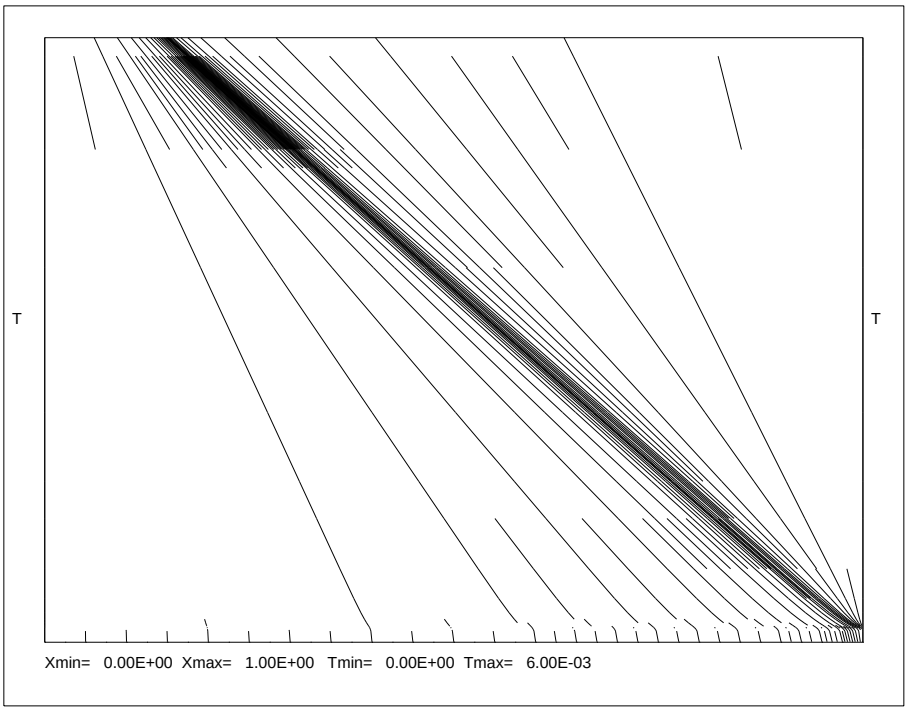


Abbildung 6.22: Mitbewegtes Gitter von Testproblem **slab–1**

TESTPROBLEM **PW–4**:

$$
\begin{aligned}
a.) \quad & x \in [-25, 10] \ , \ \ t \in [0, 15] \\
b.) \quad & T_t = T_{xx} + R(T, Y) \\
& Y_t = L^{-1} Y_{xx} - R(T, Y) \\
c.) \quad & x \in [x_L, 0) : T(x, t_0) = exp(x) \ , \ \ Y(x, t_0) = 1 - exp(L\ x) \\
& x \in [0, x_R] : T(x, t_0) = 1 \ , \ \ Y(x, t_0) = 0 \\
d.) \quad & T(x_L, t) = 0 \ , \ \ Y(x_L, t) = 1 \\
& T_x(x_R, t) = 0 \ , \ \ Y_x(x_R, t) = 0
\end{aligned}
\tag{6.5}
$$

mit der Reaktionsrate

$$
R(T) = \frac{\beta^2}{2\ L} Y e^{-\frac{\beta(1-T)}{1-\alpha(1-T)}} \tag{6.6}
$$

Für die dimensionslosen physikalischen Paramter α (Wärmefreigabe), β (Aktivierungsenergie) und L (Lewis–Zahl) können, z.B. wie in [74], die Werte

$$
\alpha = 0.8 \ , \ \ \beta = 20 \ , \ \ L = 2.0 \tag{6.7}
$$

verwendet werden. Dieses Beispiel ist dem Buch [63] entnommen. Das hier angegebene Testbeispiel entspricht dem Testproblem 4 aus [63]. Allerdings sind die eigentlich asymptotischen Randbedingungen auf die Ränder eines endlichen Intervalls gesetzt, die "weit genug" von der Position der Brennfront an $t = t_0$ entfernt sind. Die besondere Schwierigkeit bei der numerischen Lösung ist das Auftreten einer Flammenfront von variabler Gestalt und mit nichtkonstanter Geschwindigkeit. Die Lösung ist in den Abbildungen 6.23 und 6.24 gezeigt. Das mitbewegte Gitter in Abbildung 6.25. Um die Robustheit der rein statischen Gitteranpassung zu illustrierten, ist das nur statisch angepaßte Gitter einer Vergleichsrechnung in Abbildung 6.26 gezeigt. Der Aufwand dieser Vergleichsrechnungen ist in Tabelle 6.14 zusammengefaßt.

mog	$nstep$	$njac$	$nfcnj$	$nfcn$	$ndec$	$nsol$	$\bar{n}_x^F$	CPU
ja	35	74	814	366	332	1240	92	110.6
nein	249	498	3486	1961	1614	3077	86	276.0

Tabelle 6.14: Aufwand zur Lösung von Testproblem **PW–4**

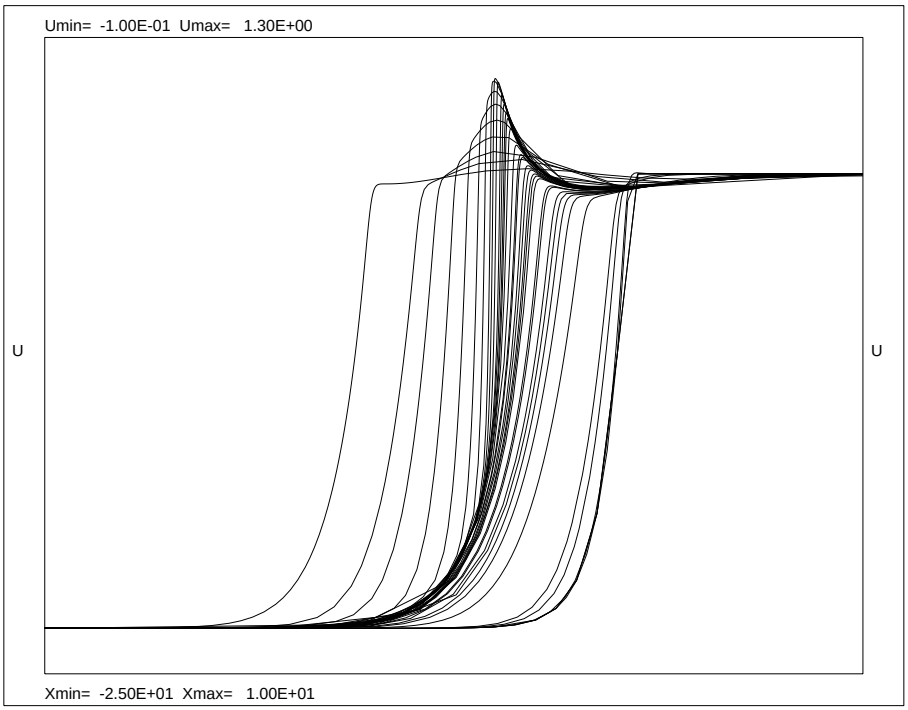


Abbildung 6.23: Lösungskomponente T von Testproblem **PW–4**

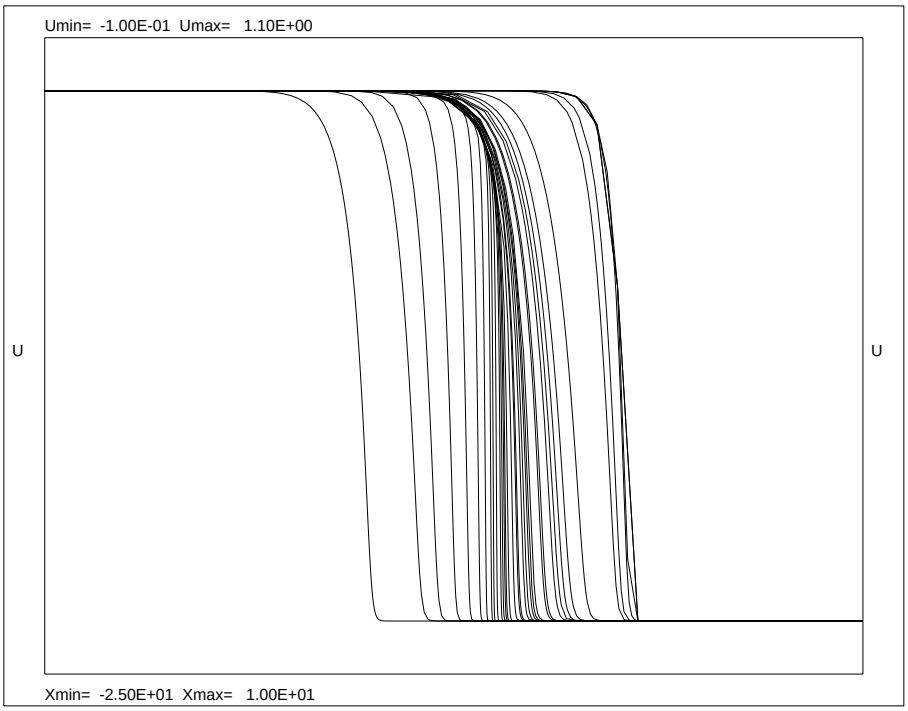


Abbildung 6.24: Lösungskomponente Y von Testproblem **PW–4**

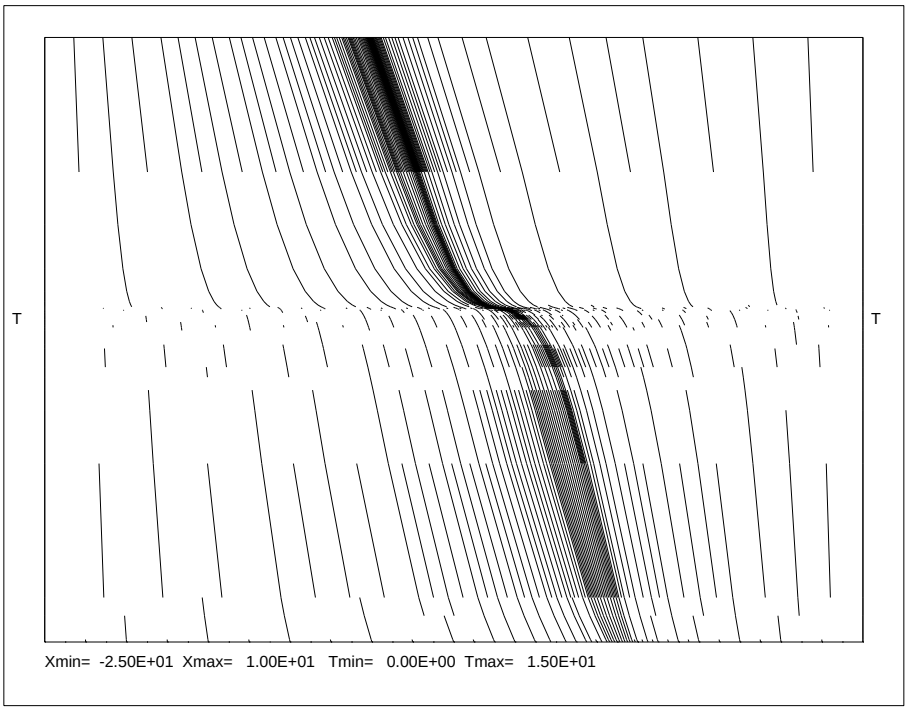


Abbildung 6.25: Mitbewegtes Gitter von Testproblem **PW–4**

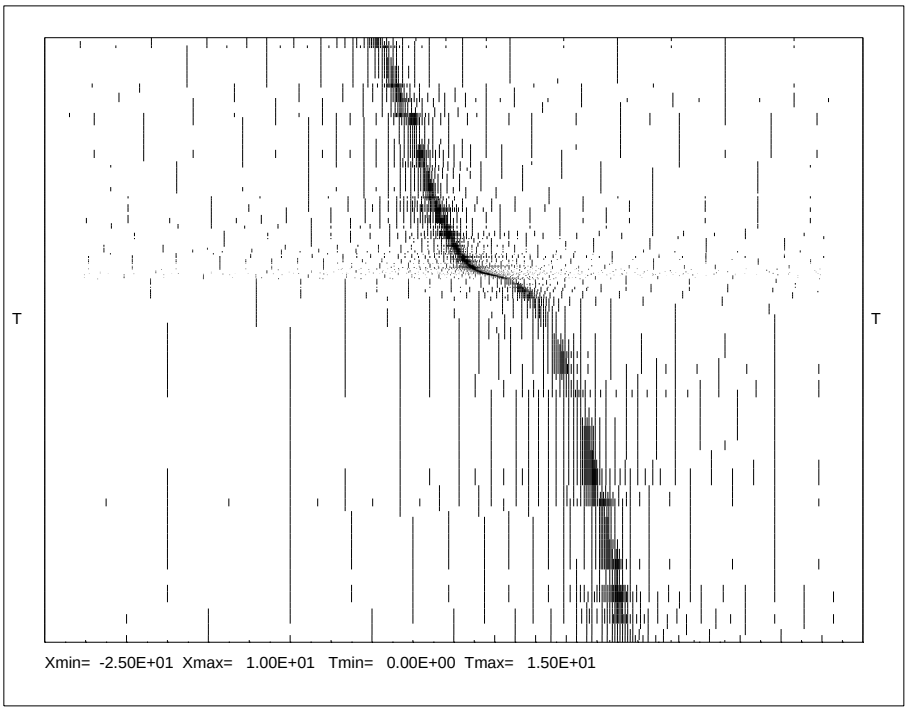


Abbildung 6.26: Nur statisch angepaßtes Gitter von Testproblem **PW–4**

6.3 Ein Problem aus der Praxis

Katalytische Abgasreinigung

Zum Verständnis der Vorgänge, die bei der Abgasreinigung in Katalysatoren stattfinden, können numerische Simulationen ein wesentliches Hilfsmittel sein. Auch mit räumlich nur eindimensionalen Modellen können bereits wichtige Erkenntnisse gewonnen werden. Insbesondere bei der Weiter– und Neuentwicklung der z. Z. verwendeten Katalysatoren können umfangreiche Simulationsstudien ein wichtiges Werkzeug sein.

Ein sehr wichtiges Problem stellt dabei fraglos die Abgasreinigung mit Hilfe von Autokatalysatoren dar. Jede, auch noch so geringe, Verbesserung des Wirkungsgrades ist ein wesentlicher Beitrag für die Reduzierung der durch den Individualverkehr hervorgerufenen Umweltverschmutzung. So werden z.B. am Institut für Chemische Verfahrenstechnik der Technischen Universität Stuttgart, in Zusammenarbeit mit einem deutschen Automobilkonzern, Möglichkeiten untersucht, u.a. durch ein neues Design den Wirkungsgrad der heutigen Auto–Katalysatoren weiter zu steigern. Die dazu notwendige realitätsnahe Modellierung führt dann allerdings auf Modellgleichungen, die den Rahmen der hier angegebenen Problemklasse sprengen.

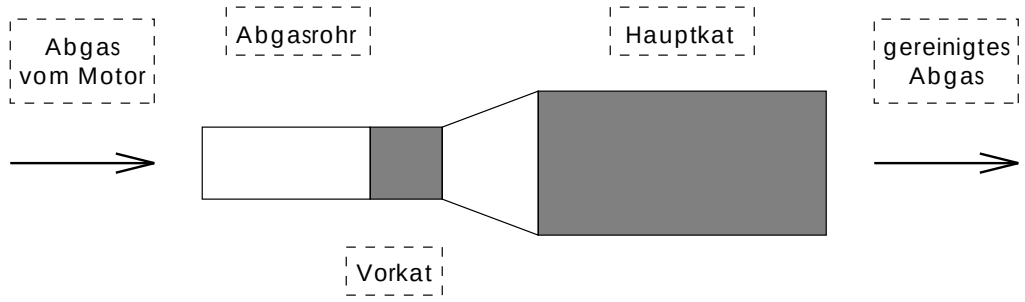


Abbildung 6.27: Zweistufiger Autokatalysator (nach KIRCHNER [45])

Um das Verfahren dennoch einsetzen zu können, wurde in Zusammenarbeit mit Mitarbeiten des o.g. Instituts eine Weiterentwicklung vorgenommen, die auch den ingenieurmäßigen Anforderungen genügt [58]. Ohne daß dabei Abstriche an den numerisch–algorithmischen Konzepten gemacht wurden, ist das weiterentwickelte Verfahren in der Lage, auch Probleme mit z.B. (mehreren) inneren Randbedingungen, nicht überall definierten abhängigen Variablen oder gebietsabhängigen Quellfunktionen zu lösen. Details finden sich in

Frauhammer [29]. Durch diese Erweiterungen ist das Verfahren zu einem robusten Simulationsinstrument in den Händen des Ingenieurs geworden. Ein typischer Versuchsaufbau, dessen mathematisches Modell mit diesem erweiterten Programmpaket simuliert wurde, ist in Abbildung 6.27 angegeben.

Ein vereinfachtes Modell

Zur Illustration der Komplexität und Schwierigkeit derartiger Modelle kann jedoch auch ein Problem dienen, das noch in die hier betrachtete Problemklasse fällt. Dazu wird ein im Ort homogenes Modell betrachtet, mit dem der Grad der Schadstoffreinigung in einem herkömmlichen Katalysator beim Start eines Autos untersucht werden kann.

Zur Modellierung wird ein 2–phasiges, kontinuierliches Katalysatormodell verwendet, in dem die radialen Effekte vernachlässigt werden. Damit ergibt sich ein räumlich eindimensionales Problem. Ohne hier auf weitere Details der Modellierung eingehen zu wollen, siehe hierzu etwa Eigenberger/Nieken [28], ist ein 2–phasiges Modell dadurch gekennzeichnet, daß zwischen Temperatur und Spezieskonzentration im Gas und am Feststoff (Katalysatormonolith) unterschieden wird. Ohne auch hier auf die gemachten Vereinfachungen (wie etwa konstante Wärmekapazität des Gases, Vernachlässigung der Strahlung und des Impulsaustausches) näher einzugehen, ergibt sich aus der Bilanzierung von Energie und Masse, sowohl im Gas als auch am Katalysator, ein komplexes System von gekoppelten, parabolischen Gleichungen, in denen auch konvektive Terme auftreten.

In dem hier verwendeten Modell soll der Schadstoffabbau von Kohlenmonoxyd, CO, und Propen, C_3H_6, untersucht werden. Dabei wird das Verhalten von Propen als repräsentativ für das Verhalten von sehr vielen Kohlenwasserstoffverbindungen angesehen. Die die chemischen Reaktionen beschreibenden Kinetiken sind dabei keine Elementarkinetiken, sondern stellen vereinfachte Bruttoreaktionen dar. Die Kopplung mit der Temperatur ist dabei hoch nichtlinear und sehr kritisch für das Verhalten des Systems.

Insgesamt ergibt sich ein Modellgleichungssystem in 11 Zustandsgrößen mit

folgender Bedeutung:

$$
\begin{aligned}
u_1 &:= T^W && : \text{Temperatur der Wand (Auspuffrohr)} \\
u_2 &:= T^G && : \text{Temperatur des Gasgemisches} \\
u_3 &:= T^K && : \text{Temperatur des Katalysators} \\
u_4 &:= g^K_{CO} && : \text{Anteil von CO an der Kat.–Oberfläche} \\
u_5 &:= g^G_{CO} && : \text{Anteil von CO im Gasgemisch} \\
u_6 &:= g^K_{C_3H_6} && : \text{Anteil von C3H6 an der Kat.–Oberfläche} \\
u_7 &:= g^G_{C_3H_6} && : \text{Anteil von C3H6 im Gasgemisch} \\
u_8 &:= g^K_{H_2} && : \text{Anteil von H2 an der Kat.–Oberfläche} \\
u_9 &:= g^G_{H_2} && : \text{Anteil von H2 im Gasgemisch} \\
u_{10} &:= g^K_{O_2} && : \text{Anteil von O2 an der Kat.–Oberfläche} \\
u_{11} &:= g^G_{O_2} && : \text{Anteil von O2 im Gasgemisch}
\end{aligned}
$$

Der Anteil des wesentlichen Abfallproduktes, Kohlendioxyd, braucht nicht mitintegriert zu werden, sondern läßt sich aus den obigen Größen rückrechnen.

Die Modellgleichungen bilden ein gekoppeltes System von linear–impliziten parabolischen Differentialgleichungen:

$$
\begin{aligned}
\varepsilon^W \rho^W c_p^W \frac{\partial T^W}{\partial t} &= \varepsilon^W \lambda^W \frac{\partial^2 T^W}{\partial x^2} && + \; q_T^W \\
\varepsilon^G \rho^G c_p^G \frac{\partial T^G}{\partial t} &= \varepsilon^G \lambda^G \frac{\partial^2 T^G}{\partial x^2} - \frac{\dot{m} c_p^G}{A} \frac{\partial T^G}{\partial x} && + \; q_T^G \\
(1-\varepsilon^G) \rho^K c_p^K \frac{\partial T^K}{\partial t} &= (1-\varepsilon^G) \lambda^K \frac{\partial^2 T^K}{\partial x^2} && + \; q_T^K
\end{aligned}
$$

$S = CO, C_3H_6, H_2, O_2$:

$$
\begin{aligned}
(1-\varepsilon^G) \rho^G \frac{\partial g_S^K}{\partial t} &= (1-\varepsilon^G) \rho^G D_{eff,S} \frac{\partial^2 g_S^K}{\partial x^2} && + \; q_S^K \\
\varepsilon^G \rho^G \frac{\partial g_S^G}{\partial t} &= \varepsilon^G \rho^G D_{eff,S} \frac{\partial^2 g_S^G}{\partial x^2} - \frac{\dot{m}}{A} \frac{\partial g_S^G}{\partial x} && + \; q_S^G \, ,
\end{aligned}
$$

wobei die Dichte des Gases durch die Gleichung

$$\rho^G = \frac{p \bar{M}}{R(T^G + 273.15)}$$

bestimmt wird. Die Bedeutung der Konstanten ist in Tabelle 6.15 zusammengefaßt.

Symbol	Bedeutung
ε^W	Geometriefaktor
ε^G	Leerraumanteil im Kat
ρ^W, ρ^K	Dichte Rohrwerkstoff bzw. Kat
c_p^W, c_p^G, c_p^K	spezifische Wärmekapazität Rohr, Gas, Kat
$\lambda^W, \lambda^G, \lambda^K$	Wärmeleitungskoeffizient Rohr, Gas, Kat
$D_{eff,S}$	Diffusionskoeffizient Stoff S
$\dot{m}$	Massenstrom
A	Fläche Rohrinnenquerschnitt
k_1^W, k_2^W	zusammengesetzte Konstanten; abhängig von weiteren physikalischen/geometrischen Konstanten
p	Gesamtdruck
R	allgemeine Gaskonstante
T^U	Umgebungstemperatur
α	Wärmeübergangskoeffizient Rohr/Umgebung
β_S	Stoffübergangskoeffizient Gas/Kat von Stoff S
a_v	geometrische Oberfläche Kat
a_x	katalytisch aktive Oberfläche Kat
H_S	Wärmetönung der Reaktion von Stoff S
A_S	Stoßfaktor Stoff S
E_S	Aktivierungsenergie Stoff S
$\bar{M}$	molare Masse Luft
M_S	molare Masse Stoff S

Tabelle 6.15: Bedeutung der physikalischen Konstanten in den Modellgleichungen.

Die Quellterme sind durch folgende Modellgleichungen gegeben:

$$\begin{aligned} q_T^W &= k_1^W(T^K - T^W) - k_2^W(T^W - T^U) \\ q_T^G &= \alpha a_v(T^K - T^G) \\ q_T^K &= -k_1^W(T^K - T^W) - \alpha a_v(T^K - T^G) \\ &\quad + a_x(H_{CO}R_{CO} + H_{C3H6}R_{C3H6} + H_{H2}R_{H2}) \end{aligned}$$

$$\begin{aligned} S = CO, C_3H_6, H_2, O_2 : \\ q_S^K &= \rho^G \beta_S a_v(g_S^G - g_S^K) - a_x M_S R_S \\ q_S^G &= -\rho^G \beta_S a_v(g_S^G - g_S^K) \ . \end{aligned}$$

Als Reaktionsgeschwindigkeiten sind hier gewählt:

$$\begin{aligned}
R_{CO} &= \frac{A_{CO} exp\left(\frac{-E_{CO}}{T^K+273.15}\right) \frac{\bar{M}}{M_{CO}} g_{CO}^K \frac{\bar{M}}{M_{O2}} g_{O2}^K}{T^K + 273.15} \\
R_{C3H6} &= \frac{A_{C3H6} exp\left(\frac{-E_{C3H6}}{T^K+273.15}\right) \frac{\bar{M}}{M_{C3H6}} g_{C3H6}^K \frac{\bar{M}}{M_{O2}} g_{O2}^K}{T^K + 273.15} \\
R_{H2} &= \frac{A_{CO} exp\left(\frac{-E_{CO}}{T^K+273.15}\right) \frac{\bar{M}}{M_{H2}} g_{H2}^K \frac{\bar{M}}{M_{O2}} g_{O2}^K}{T^K + 273.15} \\
R_{O2} &= 0.5 R_{CO} + 4.5 R_{C3H6} + 0.5 R_{H2}
\end{aligned}$$

Faßt man die Gleichungen zusammen, so erhält man ein zu lösendes System der Form

$$B(u)u' = Du_{xx} - Cu_x + q(x,u) \; ,$$

mit konstanten Diffusions–und Konvektionsmatrizen D und C sowie zustandsabhängiger Matrix B. Alle Matrizen haben strukturelle Diagonalgestalt:

$$\begin{aligned}
B &= \mathrm{diag}(b_1, \ldots, b_{11}) \\
D &= \mathrm{diag}(d_1, \ldots, d_{11}) \\
C &= \mathrm{diag}(0, c_2, 0, 0, c_5, 0, c_7, 0, c_9, 0, c_{11}) \; .
\end{aligned}$$

Als Randbedingungen sind vorgegeben:
Einströmseite ($x_L = 0.0$):

$$\left.\frac{\partial T^W}{\partial x}\right|_{x=x_L} = \left.\frac{\partial T^K}{\partial x}\right|_{x=x_L} = \left.\frac{\partial g_S^K}{\partial x}\right|_{x=x_L} = 0$$

$$\varepsilon^G \lambda^G \left.\frac{\partial T^G}{\partial x}\right|_{x=x_L} = \frac{\dot{m}}{A} c_p^G \left(T^G\Big|_{x=x_L} - T^{G,zu} \right)$$

$$\varepsilon^G D_{eff,S} \left.\frac{\partial g_S^G}{\partial x}\right|_{x=x_L} = \frac{\dot{m}}{A} \left(g_S^G\Big|_{x=x_L} - g_S^{G,zu} \right) \; ,$$

mit

$$T^{G,zu} = \begin{cases} 20 + t \cdot 3.8 & , t \in [0, 100] \\ 400 & , t > 100 \end{cases}$$

bzw.

$$g_{CO}^{G,zu} = 0.05 \, , \; g_{C3H6}^{G,zu} = 0.01 \, , \; g_{H2}^{G,zu} = 0.001 \, , \; g_{O2}^{G,zu} = 0.1 \; .$$

Ausströmseite ($x_R = 0.16$):

$$\left.\frac{\partial T^W}{\partial x}\right|_{x=x_R} = \left.\frac{\partial T^G}{\partial x}\right|_{x=x_R} = \left.\frac{\partial T^K}{\partial x}\right|_{x=x_R} = \left.\frac{\partial g_S^K}{\partial x}\right|_{x=x_R} = \left.\frac{\partial g_S^G}{\partial x}\right|_{x=x_R} = 0$$

Am rechten Rand unterliegen also alle Komponenten einer homogenen Neumann–Randbedingung. Am linken Rand, d.h. der Zulaufseite des Katalysators, gilt dies nur noch für die Feststofftemperaturen und die Konzentrationen an der Kat–Oberfläche. Für die anderen Komponenten beschreiben die üblichen Danckwerts–Randbedingungen [15] den Zulauf des Gases und der Energie.

Als Anfangsbedingung ist der Ruhezustand des Katalysators vorgegeben, d.h. Umgebungstemperatur ($T^W = T^G = T^K = 20$) und ein Luftgemisch, das den normalen Sauerstoffanteil besitzt ($g^K_{O2} = g^G_{O2} = 0.2\,; g^K_S = g^G_S = 0, S = CO, C_3H_6, H_2$).

Nach Starten des Motors wird, in idealisierter Form, Schadstoff zugeführt, in diesem Modell mit konstanter Geschwindigkeit und gleichbleibender Zusammensetzung. Da der Motor zunächst kalt ist, steigt die Temperatur des in den Katalysator einströmenden Gasgemisches erst allmählich an. Dies wird hier mit einem linearen Ansteigen modelliert. Es wird angenommen, daß erst nach 100 Sekunden eine, dann konstant bleibende, Temperatur von $400^o\ C$ im einströmenden Gasgemisch erreicht wird. Als Integrationsintervall wird hier $[0, 1000]$ betrachtet.

Numerische Simulation

Zur Lösung des Problems wird PDEX1M ohne mitbewegtes Gitter verwendet. Da die Dimension der PDG doch beachtlich groß ist, lohnt es sich, bei der Auswertung der Funktionen $\mathbf{f}^\Delta$ und $\mathbf{B}^\Delta$, die strukturelle Diagonalgestalt der Matrizen B und D auszunutzen. Dies kann durch Wahl entsprechender Optionen einfach erreicht werden. Eine spezielle Anpassung weiterer Verfahrensoptionen erfolgt nicht. Als verlangte Genauigkeiten sind $tol_x = tol_t = 10^{-3}$ vorgegeben. Als Schwellwert für die interne Skalierung aller Zustandsgrößen wird 10^{-3} verwendet.

In den Abbildungen 6.28 – 6.38 ist der Lösungsverlauf in Raum und Zeit aller Zustandsgrößen dargestellt. Da die wesentliche Dynamik innerhalb der ersten 65 Sekunden abläuft, ist nur dieser Bereich angegeben. Man beachte auch den unterschiedlichen Maßstab und den unterschiedlichen Blickwinkel für die Temperatur– bzw. Konzentrationsprofile. Es ist wieder die auf das Grobgitter restringierte Feingitterlösung dargestellt. Die Feingitterlösung selbst, evtl. noch mit zusätzlicher Interpolation, enthält keine optisch sichtbaren Knicke mehr. Da die Lösung an allen Zeitintegrationspunkten dargestellt ist, läßt sich das Verhalten der Zeitschrittweitensteuerung gut aus den Abbildungen ablesen.

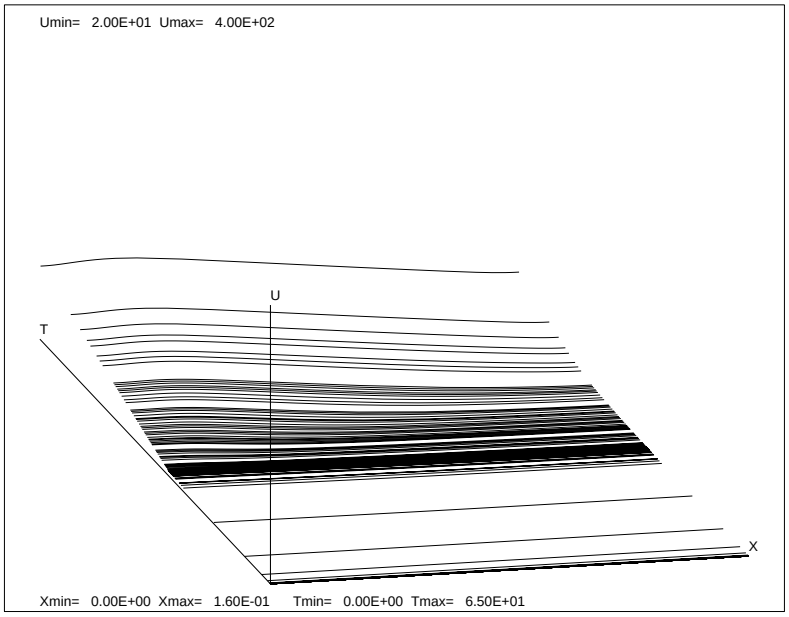


Abbildung 6.28: Autokat: Temperaturverlauf in der Auspuffrohrwand (T^W)

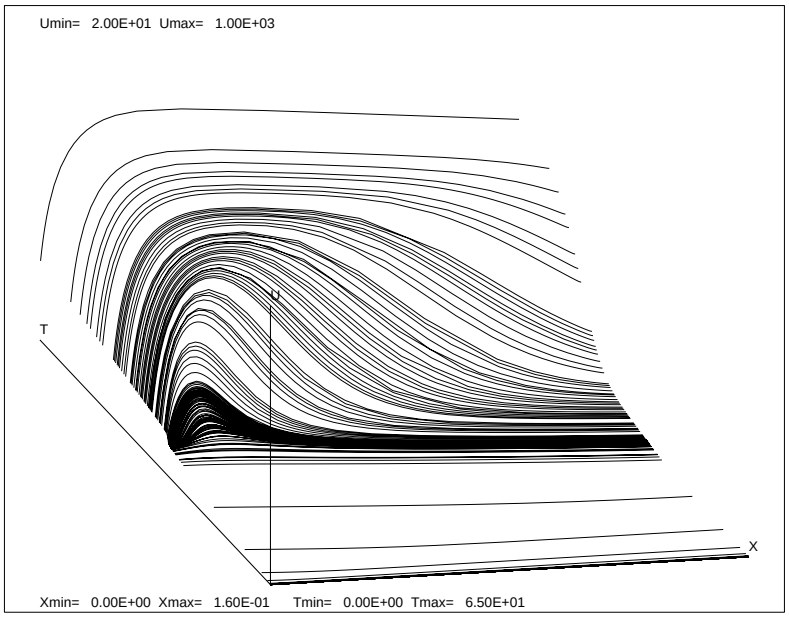


Abbildung 6.29: Autokat: Temperaturverlauf im Gasgemisch (T^G)

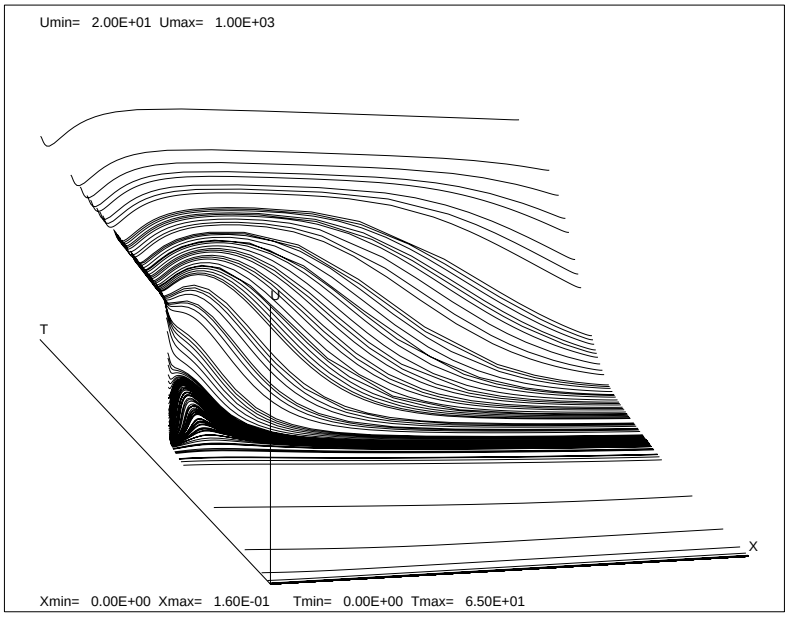


Abbildung 6.30: Autokat: Temperaturverlauf im Katalysator (T^K)

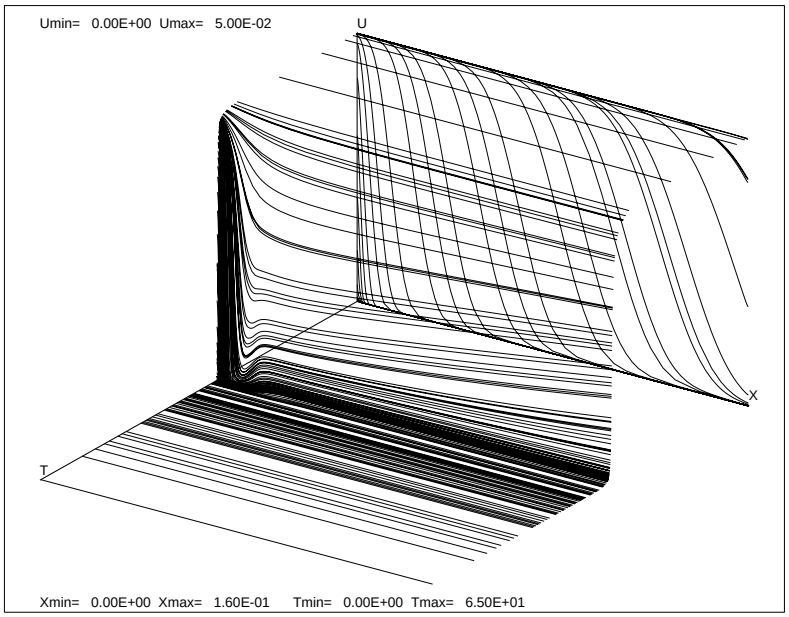


Abbildung 6.31: Autokat: Anteil von CO an der Kat–Oberfläche (g^K_{CO})

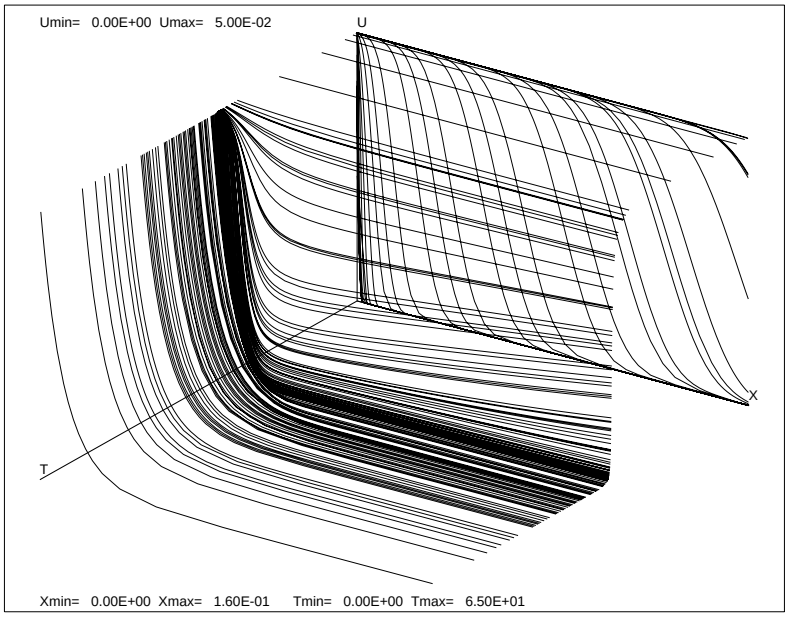


Abbildung 6.32: Autokat: Anteil von CO im Gasgemisch (g^G_{CO})

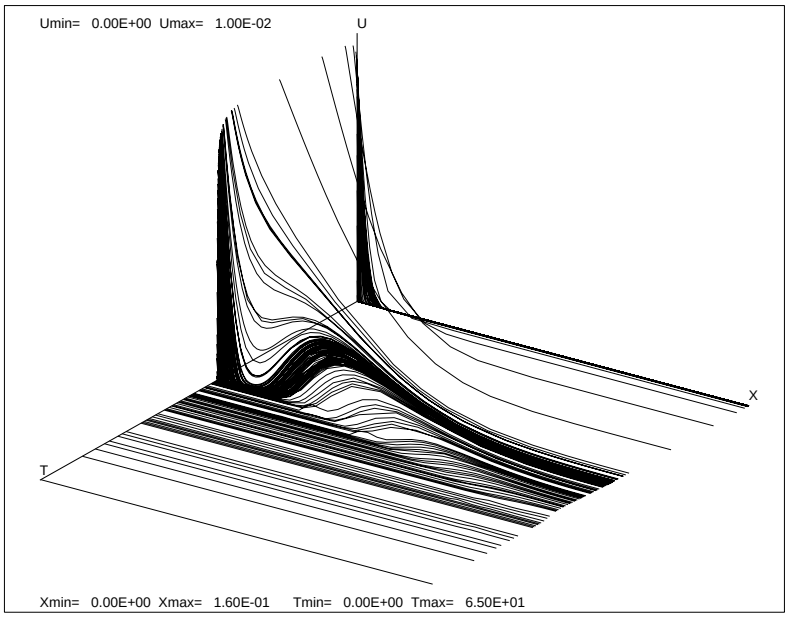


Abbildung 6.33: Autokat: Anteil von C_3H_6 an der Kat–Oberfläche (g^K_{C3H6})

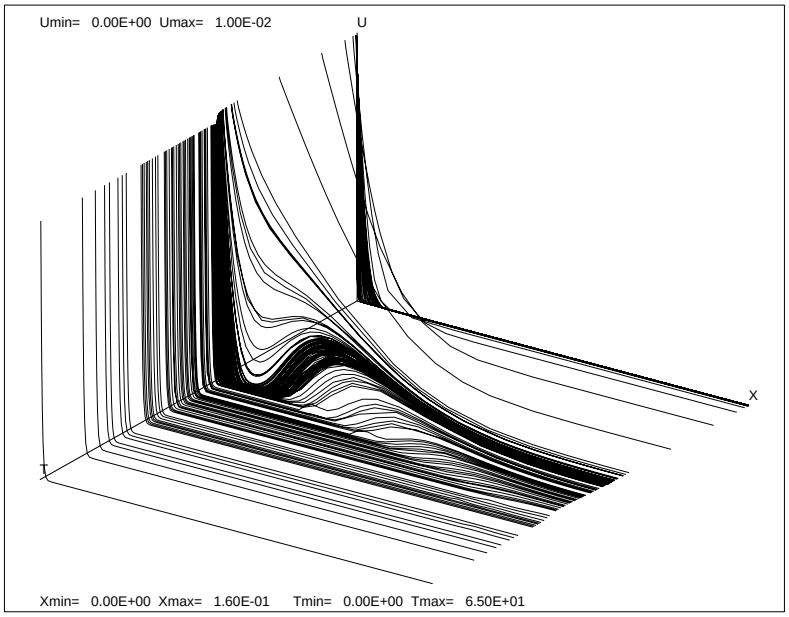


Abbildung 6.34: Autokat: Anteil von C_3H_6 im Gasgemisch (g^G_{C3H6})

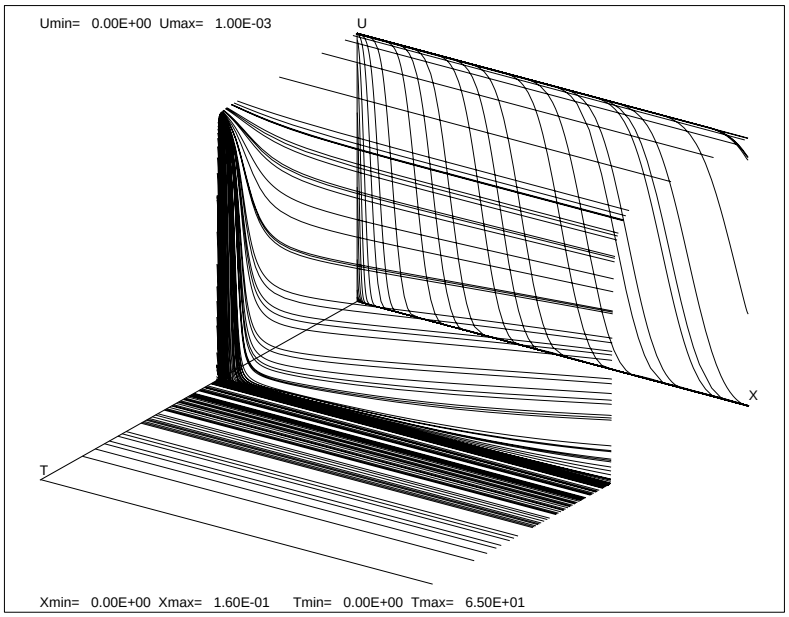


Abbildung 6.35: Autokat: Anteil von H_2 an der Kat–Oberfläche (g^K_{H2})

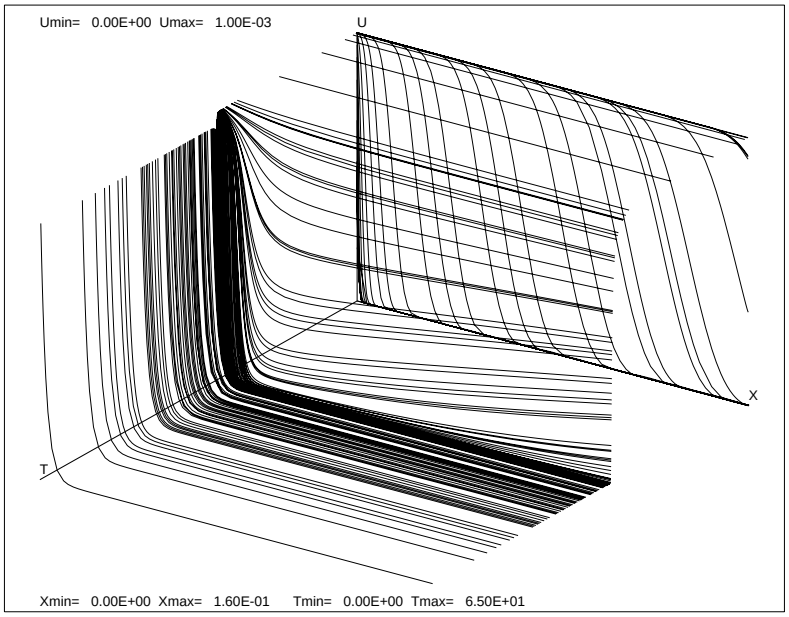


Abbildung 6.36: Autokat: Anteil von H_2 im Gasgemisch (g^G_{H2})

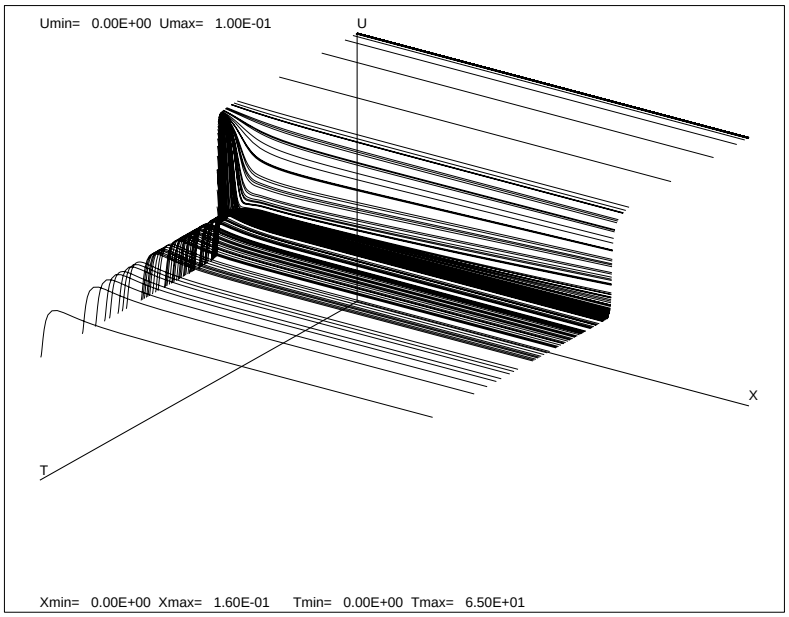


Abbildung 6.37: Autokat: Anteil von O_2 an der Kat–Oberfläche (g^K_{O2})

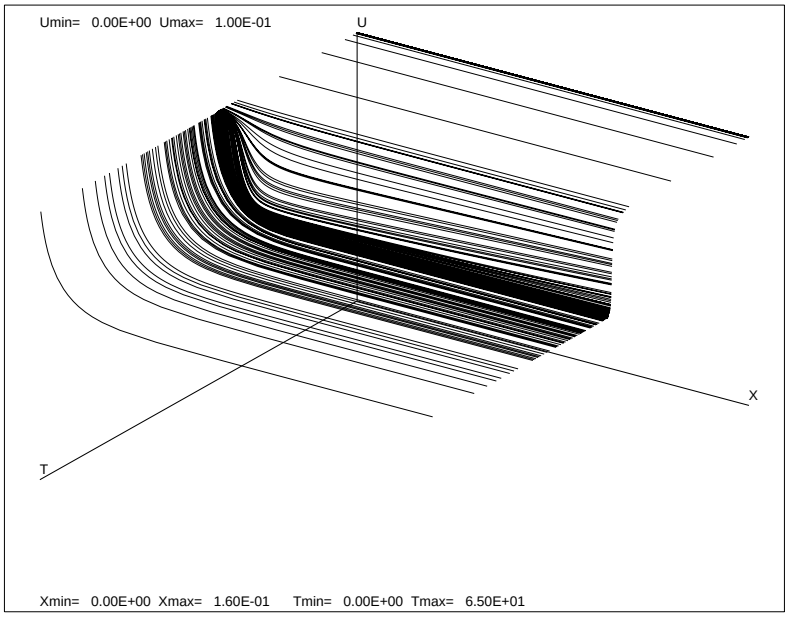


Abbildung 6.38: Autokat: Anteil von O_2 im Gasgemisch (g^G_{O2})

Die Abbildungen illustrieren zwar die hohe Dynamik des Problems, Details sind aber nur schwer zu erkennen. Auf einige, auch physikalisch wichtige, Vorgänge soll daher etwas näher eingegangen werden.

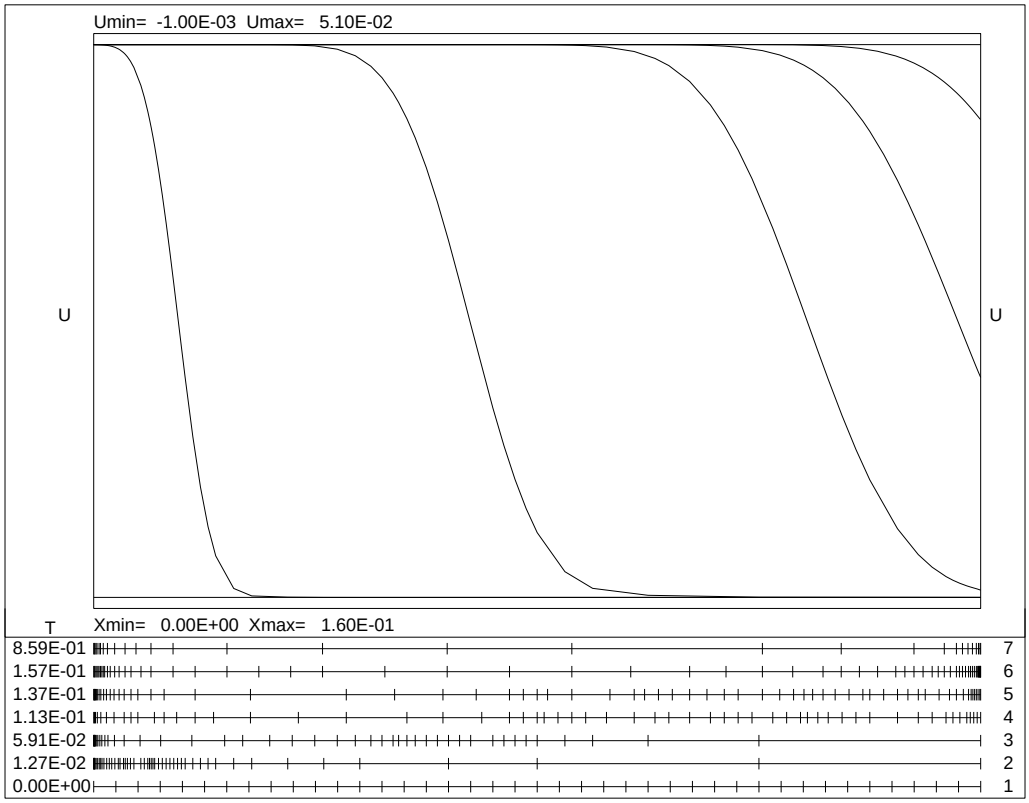


Abbildung 6.39: Autokat: Anteil von CO im Gasgemisch (g^G_{CO})

In Abbildung 6.39 ist das Konzentrationsprofil von $u_5 = g^G_{CO}$ zu einigen Zeitpunkten gleich nach Beginn der Simulation angegeben. Innerhalb von Bruchteilen von Sekunden durchläuft die Konzentrationsfront den Kat. Dies bedeutet, daß zunächst keine Reinigung stattfindet, sondern das Gas, so wie es in den Kat einströmt, auch wieder austritt. Die adaptiv gewählten Gitter zu allen dargestellten Zeitpunkten sind ebenfalls angegeben. Sie zeigen, wie sich das Gitter automatisch den sich zunächst sehr rasch ändernden Verhältnissen anpasst. Nachdem der CO–Anteil überall maximal geworden ist, befindet sich das Gesamtsystem fast in Ruhe. Die Zeitschrittweiten werden dadurch deutlich größer.

Nach ca. 25 Sek. ist die Temperatur weit genug angestiegen, daß die katalytisch bedingte "Reinigungs"–Reaktion anlaufen kann. Gemäß der gewählten Modellierung, wird nun CO fast im gesamten Bereich des Kats abgebaut,

und es bildet sich eine Reaktionsfront am linken Rand aus. Dieser Vorgang ist in Abbildung 6.40 dargestellt. Dort beginnt das Gesamtsystem immer stärker exotherm zu reagieren und die Temperatur in der Reaktionsfront steigt rasch an, wird aber noch eine Zeitlang vom kalten, hereinströmenden Gas gekühlt. Dennoch wandert die Reaktionsfront immer weiter an den linken Rand heran, was zu immer steiler werdenden Gradienten in den meisten Lösungskomponenten führt.

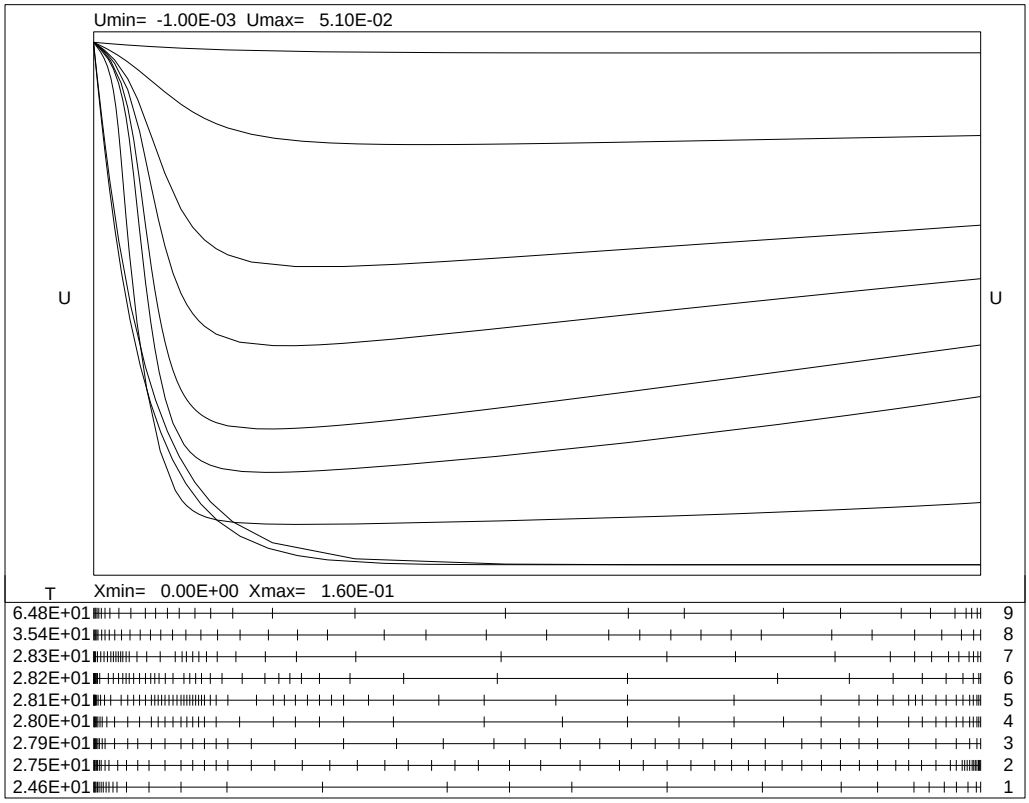


Abbildung 6.40: Autokat: Anteil von CO im Gasgemisch (g^G_{CO})

Während nach ca. 35 Sek. keine austretende CO-Konzentration mehr vorhanden ist, zeigt sich bei der Propen-Konzentration ein sehr kritisches Verhalten. Der Schadstoff dringt deutlich langsamer als CO ein. Bevor signifikante Mengen austreten können, beginnt bereits die Reaktion. Der Konzentrationsverlauf von C_3H_6 im Gasgemisch ist für einige Zeitpunkte in Abbildung 6.41 dargestellt. Während das Propen in der immer heißer werdenden Reaktionsfront rasch abgebaut wird, bleibt ein "Berg" von Propen-Konzentration rechts von der Reaktionsfront vorhanden (sowohl im Gasgemisch als auch an der Kat–Oberfläche). Dieser Berg wandert langsam durch den Kat und wird dabei weiter abgebaut, so daß, unter den hier gewählten Simulationsbedin-

gungen, kaum ein Schadstoffaustritt stattfindet. Dieser Prozeß läßt sich in Abbildung 6.41 gut nachverfolgen. Allerdings konzentrieren sich die Knoten des Gitters dabei immer mehr am linken Rand, da die dortige Dynamik den Diskretisierungsfehler dominiert.

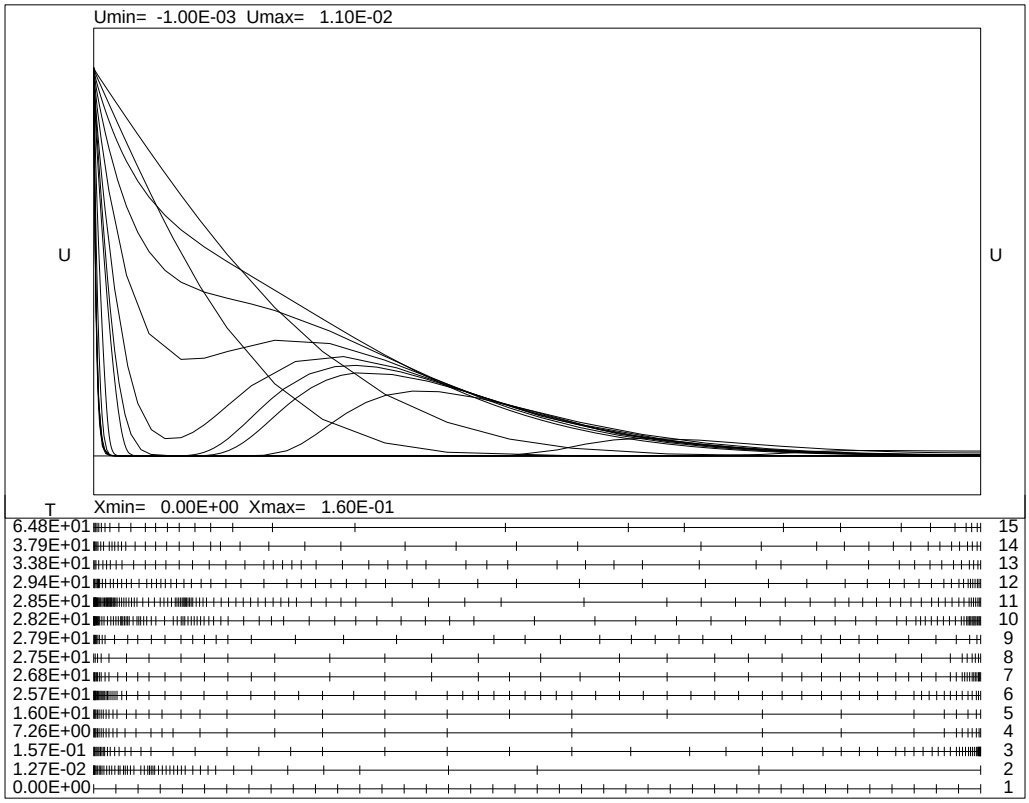


Abbildung 6.41: Autokat: Anteil von C_3H_6 im Gasgemisch (g^G_{C3H6})

Der wandernde Berg führt dagegen kaum zum Einfügen von Knoten. Dies ist nicht zuletzt auch ein Skalierungsproblem. Die Werte in diesem Bereich sind kleiner als die zuvor schon maximal erreichten und die zugehörigen Fehler werden daher nicht mehr so stark gewichtet. Gerade dieser wandernde Berg ist aber der numerisch kritische Bereich. Der hyperbolische Anteil in der Gleichung führt hier zu ersten Instabilitäten. Dies zeigen einige "Überschwinger" in der Lösung – vgl. Abbildung 6.33 bzw. 6.34. Da der Effekt lokal ist, wird diese Lösung dennoch akzeptiert (globale Norm für Konvergenzkontrolle!). Durch die lokale Gitteranpaßung, in die die lokalen Fehlerschätzer eingehen, werden allerdings sofort einige Knoten eingefügt, so daß die Lösung im nächsten Schritt wieder glatt ist. Ohne eine adaptive Kontrolle verstärken sich die Fehler und dieser Effekt kann bis zum Zusammenbruch der Integration führen.

Dieses instabile Verhalten der Lösung bedingt u.a. die klein bleibenden Zeitschrittweiten. Erst wenn die kritische Zustandsgröße g^G_{C3H6} im kritischen Bereich praktisch zu Null geworden ist, ist das Gesamtsystem wieder dissipativ.

Der in allen Speziesgleichungen der Gasphase vorhandene konvektive Term führt auch zu Instabilitäten am rechten Rand. Insbesondere, wenn die Lösung in Randnähe einen nicht verschwindenden Gradienten hat, und die dann nicht als Symmetriebedingung anzusehenden homogenen Neumann–Randbedingungen einen nicht mehr quadratischen Konsistenzfehler erzeugen. Obwohl hier keine quadratische Fehlerentwicklung mehr vorliegt, ist der Fehlerschätzer robust genug, zumindest die Größenordung des Fehlers zu erkennen, was dann letztlich zum Einfügen von Knoten führt.

Nach ca. 40 Sekunden befindet sich das System in einem fast stationären und stabilen Zustand. Es können wieder deutlich größere Schrittweiten gewählt werden. Das weitere Ansteigen der Temperaturen, insbesondere nun auch die der Außenwand des Kats, stellt keine kritische Dynamik mehr dar. Mit nur wenigen Integrationsschritten wird dann bis zur Endzeit $t = 1000$ Sekunden integriert. Der dann herrschende Zustand entspricht schon fast dem endgültigen stationären Zustand des Systems.

Der Aufwand zur numerischen Lösung dieses Problems ist beträchtlich. Der benötigte Aufwand für einige Rechnungen mit unterschiedlichen Genauigkeitsanforderungen ist in Tabelle 6.16 angegeben.

Bei der etwas weniger stringenten Anforderung $tol = 2.5 \cdot 10^{-3}, tol_x = tol_t = tol$ sind die beobachtbaren Instabilitäten noch etwas deutlicher zu sehen. Die adaptive Schrittweitenkontrolle bringt durch Einfügen neuer Knoten auch hier die Oszillationen nach jeweils einem Schritt zum Verschwinden.

tol	$2.5 \cdot 10^{-3}$	$1.0 \cdot 10^{-3}$	$2.5 \cdot 10^{-4}$	$10^{-3}/voll$
CPU	1335	2356	5753	5668
$nstep$	150	176	208	174
$njac$	316	386	450	374
$nfcnj$	8878	7268	10350	15652
$nfcn$	1275	1804	2760	1600
$ndec$	1316	1001	1687	1186
$nsol$	4511	3086	7122	3961
$\bar{n}_x^F$	83	115	205	117

Tabelle 6.16: Aufwandsvergleich bei Simulation des Autokat–Problems

Wie wichtig bei größeren Problemen das Ausnutzen struktureller Eigenschaften ist, zeigen die Ergebnisse, die in der letzten Spalte der Tabelle angegeben sind. Sie sind das Resultat einer Simulation, ohne daß dabei die strukturelle

Diagonalgestalt von B, D und das Vorliegen einer Gesamtbandbreite von nur $m_b = 2n_{pde} + 1$ in der Jacobimatrix des semi–diskreten Systems ausgenutzt wird. Die Gesamtrechenzeit wird mehr als verdoppelt, wobei sich die Tatsache, daß sich der durchschnittliche Aufwand zur Berechnung einer Jacobimatrix um einen Faktor 3.5 erhöht, und damit dann 63% der Gesamtrechenzeit ausmacht, stark bemerkbar macht. Dieser Faktor setzt sich aus einer knappen Verdopplung der Rechenzeit für eine $\mathbf{B}^\Delta, \mathbf{f}^\Delta$–Auswertung und der Verdopplung der Bandbreite (und damit notwendigen Verdopplung der Anzahl der $\mathbf{B}^\Delta, \mathbf{f}^\Delta$–Auswertungen zur numerischen Differentiation) zusammen. Interessanterweise steigt der Aufwand einer LU–Zerlegung im Bandmodus nicht, wie theoretisch zu erwarten, quadratisch, sondern nur linear an. Diese überraschende Beobachtung stimmt mit der für volle Matrizen in [51] gemachten Beobachtung überein. Der Aufwand einer Vorwärts/Rückwärtssubstitution steigt dagegen wie erwartet linear an.

Die geringen Abweichungen in den anderen Indikatoren (natürlich mit Ausnahme von $nfcnj$) rühren daher, daß die etwas unterschiedliche numerische Differentiation zu einer etwas anders gestörten Jacobimatrix führt, und sich somit der Gesamtablauf der Integration geringfügig ändert.

Der Vergleich des Aufwands der Simulation mit $tol = 10^{-3}$ und $tol = \frac{1}{4}10^{-3}$ zeigt, daß sich die durchschnittlich verwendete Knotenzahl des Feingitters fast verdoppelt (zu erwarten für ein Verfahren mit quadratischem Fehlerverhalten). Die Erhöhung des Aufwands zur Zeitintegration ist dagegen etwas geringer (ca. 50% mehr $\mathbf{B}^\Delta, \mathbf{f}^\Delta$–Auswertungen, ca. 20% mehr Schritte). Bei einer Abschwächung der Genauigkeitsforderung um den Faktor 2.5 ist die (beobachtbare) Gittervergröberung um den Faktor 1.4 gar nicht so weit entfernt von dem zu erwartenden Wert $\sqrt{2.5} = 1.6$. Wenn man bedenkt, daß Zeit–und Ortsschrittweitenwahl gekoppelt sind, und in der gesamten Steuerung einige heuristische Kriterien und Nebenbedingungen (z.B. $n_x^F \geq 41$) verwendet werden, so zeigt dieses Verhalten doch überraschend gut den mathematischen Hintergrund der verwendeten algorithmischen Konzepte auf.

Zusammenfassung

Ausgehend von einem klassischen Linienmethodenansatz wurde ein neues numerisches Verfahren zur Lösung von hochnichtlinearen, gekoppelten Systemen von parabolisch dominanten Differentialgleichungen in einer Raumdimension entwickelt. In der Zeit wird eine semi–implizite Euler–Diskretisierung verwendet, und im Ort wird mittels finiter Differenzen auf nicht–uniformen Gittern diskretisiert. Beide Basis–Diskretisierungen sind mit Extrapolation verbunden. Während die Zeit–Extrapolation von lokaler, variabler Ordnung ist, wird im Raum nur einmal extrapoliert. Basierend auf lokalen Fehlerschätzungen für beide Diskretisierungen, wird der Diskretisierungsfehler kontrolliert und die Diskretisierungsschrittweiten simultan und automatisch angepaßt. Neben der lokalen Anpassung der verwendeten Ortsgitter nach jedem Zeitschritt (statisches Regriddung) ist zusätzlich auch eine Mitbewegung des Gitters während des Zeitintegrationschritts möglich (dynamisches Regridding). Damit besitzt das Gesamtverfahren ein hohes Maß an Adaptivität, und ist somit in der Lage, schwierige Probleme aus den praktischen Anwendungen robust und effizient zu lösen.

Literaturverzeichnis

[1] C.M. Ablow, S. Schechter: *Campylotropic Coordinates*. J. Comput. Phys. **27**, p. 351–362 (1978)

[2] S. Adjerid, J.E. Flaherty: *A Moving Finite Element Method with Error Estimation and Refinement for One–Dimensional Time Dependent Partial Differential Equations*. SIAM J. Numer. Anal. **23**, p. 778–796 (1986)

[3] U. Ascher, J. Christiansen, R.D. Russell: *Collocation Software for Boundary-Value ODEs*. ACM Trans. Math. Software **7**, p. 209–222 (1981)

[4] I. Babuška, W.C. Rheinboldt: *Error Estimates for Adaptive Finite Element Computations*. SIAM J. Numer. Anal. **15**, p. 736–754 (1978)

[5] G. Bader, U. Ascher: *A New Basis Implementation for a Mixed Order Boundary Value ODE Solver*. SIAM J. Sci. Stat. Comput. **8**, p. 483–500 (1987)

[6] G. Bader, P. Deuflhard: *A Semi–Implicit Mid–Point Rule for Stiff Systems of Ordinary Differential Equations*. Numer. Math. **41**, p. 373–398 (1983)

[7] R.E. Bank, T. Dupont, H. Yserentant: *The Hierarchical Basis Multgrid Method*. Numer. Math. **52**, p. 427–458 (1988)

[8] M. Bieterman, I. Babuška: *An Adaptive Method of Lines with Error Control for Parabolic Equations of the Reaction–Diffusion Type*. J. Comput. Phys. **63**, p. 33–66 (1986)

[9] F.A. Bornemann: *An Adaptive Multilevel Approach to Parabolic Equations. I: General Theory and 1D–Implementation*. IMPACT Comput. Sci. Engrg. **2**, p. 279–317 (1990)

[10] F.A. Bornemann: *An Adaptive Multilevel Approach to Parabolic Equations. II: Variable Order Time Discretization Based on Multiplicative Error Correction*. IMPACT Comput. Sci. Engrg. **3**, p. 93–122 (1991)

[11] F.A. Bornemann: *An Adaptive Multilevel Approach to Parabolic Equations. III: 2D Error Estimation and Multilevel Preconditioning*. IMPACT Comput. Sci. Engrg. **4**, p. 1–45 (1992)

[12] K.W. Brodlie: *A Rewiew of Methods for Curve and Function Drawing*. In: K.W. Brodlie (ed.): Mathematical Methods in Computer Graphics and Design, p. 1–37. Academic Press, London (1980)

[13] T.H. Chong: *A Variable Mesh Finite Difference Method for Solving a Class of Parabolic Differential Equations in One Space Variable*. SIAM J. Numer. Anal. **15**, p. 835–857 (1978)

[14] C.F. Curtiss, J.O. Hirschfelder: *Integration of Stiff Equations*. Proc. Nat. Acad. Sci. **38**, p. 235–243 (1952)

[15] P. V. Danckwerts: *Continuous Flow Systems: Distribution and Residence Times*. Chem. Engng. Sci. **2**, p. 1–13 (1953)

[16] S.F. Davis, J.E. Flaherty: *An Adaptive Finite Element Method for Initial–Boundary Value Problems for Parabolic Equation*. SIAM J. Sci. Stat. Comput. **3**, p. 6–27 (1982)

[17] P. Deuflhard: *Order and Stepsize Control in Extrapolation Methods*. Numer. Math. **41**, p. 399–422 (1983)

[18] P. Deuflhard: *Recent Progress in Extrapolation Methods for ODE's*. SIAM Review **27**, p. 505–535 (1985)

[19] P. Deuflhard: *Uniqueness Theorems for Stiff ODE Initial Value Problems*. In: D.F. Griffith, G.A. Watson (eds.): Numerical Analysis, Dundee 1989, Pitman Research Notes in Mathematics Series 288, p. 74–88 (1989)

[20] P. Deuflhard: *Numerik von Anfangswertmethoden für gewöhnliche Differentialgleichungen*. Technical Report TR 89–2, Konrad–Zuse–Zentrum Berlin (1989)

[21] P. Deuflhard, P. Leinen, H. Yserentant: *Concepts of an Adaptive Hierarchical Finite Element Code*. IMPACT Comput. Sci. Engrg. **1**, p. 3–35 (1989)

[22] P. Deuflhard, E. Hairer, J. Zugck: *One–Step and Extrapolation Methods for Differential–Algebraic Systems*. Numer. Math. **51**, p. 501–516 (1987)

[23] P. Deuflhard, U. Nowak: *Extrapolation Integrators for Quasilinear Implicit ODE's*. In: P. Deuflhard, B. Enquist (eds.): Large Scale Scientific Computing. Progress in Scientific Computing **7**, p. 37–50. Birkhäuser (1987)

[24] P.M. Dew, J.E. Walsh: *A Set of Library Routines for Solving Parabolic Equations in One Space Variable*. ACM Trans. Math. Software **7**, p. 232–260 (1981)

[25] J.J. Dongarra, C.B. Moler, J.R. Bunch, G.W. Stewart: LINPACK. SIAM, Philadelphia (1979)

[26] E. A. Dorfi, L. O'C. Drury: *Simple Adaptive Grids for 1–D Initial Value Problems*. J. Comput. Phys. **69**, p. 175–195 (1987)

[27] H.A. Dwyer, B.R. Sanders: *Numerical Modeling of Unsteady Flame Propagation*. Acta Astronaut. **5**, p. 1171–1184 (1978)

[28] G. Eigenberger, U. Nieken: *Katalytische Abluftreinigung: Verfahrenstechnische Aufgaben und neue Lösungen*. Chem. Ing. Techn. **63(8)**, (1991)

[29] J. Frauhammer: *Numerische Lösung von eindimensionalen parabolischen Systemen mit adaptiven Gittern.* Diplomarbeit, Inst. für Chemische Verfahrenstechnik, Universität Stuttgart (1992)

[30] J. Frauhammer: *Persönliche Mitteilung*

[31] F.N. Fritsch, J. Butland: *A Method for Constructing Local Monotone Piecewise Cubic Interpolants*. SIAM J. Sci. Stat. Comput. **5**, p. 300–304 (1984)

[32] F.N. Fritsch, J. Butland: *Piecewise Cubic Hermite Intepolation Package*. Preprint UCRL–87285, Lawrence Livermore Nat. Lab. (1985)

[33] F.N. Fritsch, R.E. Carlson: *Monotone Piecewise Cubic Interpolation*. SIAM J. Numer. Anal. **17**, p. 238–246 (1980)

[34] R.M. Furzeland, J.G. Verwer, P.A. Zegeling: *A Numerical Study of Three Moving Grid Methods for One–Dimensional Partial Differential Equations which are Based on the Method of Lines*. J. Comput. Phys. **89**, p. 349–388 (1990)

[35] E. Hairer, Ch. Lubich: *Asymptotic Expansions of the Global Error of Fixed–Stepsize Methods*. Numer. Math. **45**, p. 345–360 (1984)

[36] E. Hairer, Ch. Lubich: *Extrapolation at Stiff Differential Equations*. Numer. Math. **52**, p. 377–400 (1988)

[37] E. Hairer, A. Ostermann: *Dense Output for Extrapolation Methods*. Numer. Math. **58**, p. 419–439 (1990)

[38] E. Hairer, G. Wanner: *Solving Ordinary Differential Equations II. – Stiff and Differential–Algebraic Problems*. Springer Series in Computational Mathematics **14**, Springer Verlag, Berlin–Heidelberg (1991)

[39] A. Harten, J.M. Hyman: *Self–Adjusting Grid Methods for One–Dimensional Hyperbolic Conservation Laws*. J. Comput. Phys. **50**, p. 235–269 (1983)

[40] A.C. Hindmarsh: LSODE *and* LSODI, *Two New Initial Value Ordinary Differential Equation Solvers*. ACM SIGNUM Newsletter **15**, p. 10 (1980)

[41] J.D. Hoffmann: *Relationship between the Truncation Errors of Centered Finite Difference Approximations on Uniform and Nonuniform Meshes*. J. Comput. Phys. **46**, p. 469–474 (1982)

[42] J.M. Hyman: *Moving Mesh Methods for Initial Boundary Value Problems*. Preprint LA–UR–84–61, Los Alamos Nat. Lab. (1984)

[43] J.M. Hyman: *Moving Mesh Methods for Partial Differential Equations*. In: L. Goldstein, S. Rosencrans, G. Sod (eds.): Mathematics Applied to Science, p. 129–153. Academic Press, Inc. (1988)

[44] A.K. Kapila: *Asymptotic Treatment of Chemically Reacting Systems*. Pitman Applicable Mathematics Series, Pitman, New York (1983)

[45] T. Kirchner: *Simulation des Anspringverhaltens eines Autoabgaskatalysators*. Forschungsvorhaben am Inst. für Chemische Verfahrenstechnik, Universität Stuttgart (1993)

[46] J. Lang, A. Walter: *A Finite Element Method Adaptive in Space and Time for Nonlinear Reaction–Diffusion Systems*. IMPACT Comput. Sci. Engrg. **4**, p. 269–314 (1992)

[47] B. Leimkuhler, L.R. Petzold, C.W. Gear: *Approximation Methods for the Consistent Initialization of Differential–Algebraic Equations*. SIAM J. Numer. Anal. **28**, p. 203–225 (1991)

[48] Ch. Lubich: *Linearly Implicit Extrapolation Methods for Differential–Algebraic Systems*. Numer. Math. **55**, p. 129–145 (1989)

[49] Ch. Lubich: *Persönliche Mitteilung*. (1992)

[50] Ch. Lubich, A. Ostermann: *Runge–Kutta Methods for Parabolic Equations and Convolution Quadrature*. ETH Zürich, Seminar für Angewandte Mathematik, Research Report No. 91–06 (1991)

[51] Ch. Lubich, U. Nowak, U. Pöhle, Ch. Engstler: MEXX – *Numerical Software for the Integration of Constrained Mechanical Multibody Systems*. Preprint SC 92–12, Konrad–Zuse–Zentrum Berlin (1992)

[52] U. Maas: *Persönliche Mitteilung*.

[53] U. Maas, J. Warnatz: *Simulation of Chemically Reacting Flows in Two-Dimensional Geometries*. IMPACT Comput. Sci. and Engrg. **1**, p. 394–420 (1989)

[54] Th.A. Manteuffel, A.B. White jr.: *The Numerical Solution of Second Order Boundary Value Problems on Nonuniform Meshes*. Math. Comp. **47**, p. 511–535 (1986)

[55] K. Miller: *Moving Finite Elements II*. SIAM J. Numer. Anal. **18**, p. 1033–1057 (1981)

[56] K. Miller, R.N. Miller: *Moving Finite Elements I*. SIAM J. Numer. Anal. **18**, p. 1019–1032 (1981)

[57] U. Nowak: *Dynamic Sparsing in Stiff Extrapolation Methods*. IMPACT Comput. Sci. Engrg. **5**, p. 53–74 (1993)

[58] U. Nowak, J, Frauhammer, U. Nieken, G. Eigenberger: *A Fully Adaptive Algorithm for Parabolic Partial Differential Equations in One Space Dimension*. Submitted for publication

[59] A. Ostermann, P. Kaps, T.D. Bui: *The Solution of a Combustion Problem with Rosenbrock Methods*. ACM Trans. Math. Software **12**, p. 354–361 (1986)

[60] G.R. Otey, H.A. Dwyer: *Numerical Study of the Interaction of Fast Chemistry and Diffusion*. AIAA J. **17**, p. 606–613 (1979)

[61] C.E. Pearson: *On a Differential Equation of Boundary Layer Type*. J. Math. Phys. **47**, p. 134–154, (1968)

[62] V. Pereyra, E.G. Sewell: *Mesh Selection for Discrete Solution of Boundary Value Problems in Ordinary Differential Equations*. Numer. Math. **23**, p. 261–268 (1975)

[63] N. Peters, J. Warnatz (eds.): *Numerical Methods in Laminar Flame Propagation*. Notes on Numerical Fluid Dynamics **6**, Vieweg (1982)

[64] L.R. Petzold: *A Description of DASSL: a differential–algebraic system solver*. Proc. IMACS World Congress, Montreal, Canada (1982)

[65] L.R. Petzold: *Order Results for Implicit Runge–Kutta Methods Applied to Differential/Algebraic Systems*. SIAM J. Numer. Anal. **23**, p. 837–852 (1986)

[66] L.R. Petzold: *Observations on an Adaptive Moving Grid Method for One–Dimensional Systems of Partial Differential Equations*. Appl. Numer. Math. **3**, p. 347–360 (1987)

[67] L.R. Petzold: *An adaptive Moving Grid Method for One–Dimensional Systems of Partial Differential Equations and its Numerical Solution*. Proc. Workshop on Adaptive Methods for Partial Differential Equations, Renselaer Polytechnic Institute (1988)

[68] PH.J. Rasch, D.L. Williamson: *On Shape–Preserving Interpolation and Semi–Largangian Transport*. SIAM J. Sci. Stat. Comput. **11**, p. 656–687 (1990)

[69] R.K. de Rivas: *On the Use of Nonuniform Grids in Finite Difference Equations*. J. Comput. Phys. **10**, p. 202–210 (1972)

[70] R.D. Russell: *Mesh Selection Methods*. Proceedings of the Conference for Working Codes for Boundary Value Problems in ODE's, Springer–Verlag, New York (1979)

[71] W.E. Schiesser: *The Numerical Method of Lines*. Academic Press, Inc., San Diego (1991)

[72] R.F. Sincovec, N.K. Madsen: *Software for Nonlinear Partial Differential Equations*. ACM Trans. Math. Software **1**, No. 3, p. 232–260 (1975)

[73] J. Smoller: *Shock Waves and Reaction–Diffusion Equations*. Grundlehren der mathematischen Wissenschaften 258, Springer Verlag (1983)

[74] M.D. Smooke, M.L. Koszykowski: *Fully Adaptive Solutions of One–Dimensional Mixed Initial–Boundary Value Problems with Applications to Unstable Problems in Combustion*. SIAM J. Sci. Stat. Comput. **7**, p. 301–321 (1986)

[75] K. Strehmel, R. Weiner: *Linear–implizite Runge–Kutta–Methoden und ihre Anwendungen*. Teubner–Texte zur Mathematik 127, B.G. Teubner Verlagsgesellschaft, Leipzig (1992)

[76] K. Strehmel, W.H. Hundsdorfer, R. Weiner, A. Arnold: *The Linearly Implicit Euler Method for Quasi–Linear Parabolic Differential Equations*. Centrum voor Wiskunde en Infomatica (CWI), Amsterdam, Report NM–R9002 (1990)

[77] J.G. Verwer, J.G. Blom, R.M. Furzeland, P.A. Zegeling: *A Moving–Grid Method for One–Dimensional PDEs based on the Method of Lines*. In: J.E. Flaherty, P.J. Paslow, M.S. Shepard, J.D. Vasilakis (eds.): Adaptive Methods for Partial Differential Equations, p. 160–175. SIAM, Philadelphia (1989)

[78] J.G. Verwer, J.G. Blom, J.M. Sanz–Serna: *An Adaptive Moving Grid Method for One–Dimensional Systems of Partial Differential Equations*. J. Comput. Phys. **82**, p. 454–486 (1989)

[79] A.B. White jr: *On Selection of Equidistributing Meshes for Two–Point Boundary Value Problems*. SIAM J. Numer. Anal. **16**, p. 472–502 (1979)

[80] P.A. Zegeling, J.G. Blom: *An Evaluation of the Gradient–Weighted Moving–Finite–Element Method in One Space Dimension*. Report NM–R9006, Centrum voor Wiskunde en Infomatica (CWI), Amsterdam (1990)