BAM
Bundesanstalt für
Materialforschung
und -prüfung

Dr. rer. nat. Frank Liebenau
Dr.-Ing. Reinald Rudolphi

Das Verfahren von Gear mit BICG-Löser – Grundlagen, Algorithmen und Anwendungen auf Temperaturberechnungen aus dem Brand- und Wärmeschutz

Forschungsbericht
247

Die Aufgaben der Bundesanstalt stehen unter der Leitlinie:

Sicherheit und Zuverlässigkeit in Chemie- und Materialtechnik

Die Aufgaben

Die BAM ist im Aufgabenverbund „Material – Chemie –Umwelt – Sicherheit" zuständig für:

- Hoheitliche Funktionen zur öffentlichen technischen Sicherheit, insbesondere im Gefahrstoff- und Gefahrgutrechtsbereich;
- Mitarbeit bei der Entwicklung entsprechender gesetzlicher Regelungen, z. B. bei der Festlegung von Sicherheitsstandards und Grenzwerten;
- Beratung der Bundesregierung, der Wirtschaft sowie der nationalen und internationalen Organisationen im Bereich der Materialtechnik und Chemie;
- Entwicklung und Bereitstellung von Referenzmaterialien und -verfahren, insbesondere der analytischen Chemie und der Prüftechnik;
- Unterstützung der Normung und anderer technischer Regeln für die Beurteilung von Stoffen, Materialien, Konstruktionen und Verfahren im Hinblick auf die Schadensfrüherkennung bzw. -vermeidung, den Umweitschutz und den Erhalt volkswirtschaftlicher Werte.

Die Tätigkeitsbereiche

Das Tätigkeitsspektrum der BAM umfasst die sich ergänzenden und aufeinander bezogenen Tätigkeiten:

- Forschung und Entwicklung
- Prüfung, Analyse und Zulassung
- Beratung und Information.

Die nationale und internationale Zusammenarbeit

Die Aufgaben der BAM für Technik, Wissenschaft, Wirtschaft und Gesellschaft erfordern eine interdisziplinäre Zusammenarbeit. Insofern arbeitet die Bundesanstalt mit Technologieinstitutionen des In- und Auslandes, insbesondere den nationalen Schwesterinstituten eng zusammen. Sie berät Bundesministerien, Wirtschaftsverbände, Industrieunternehmen sowie Verbraucherorganisationen und unterstützt mit Fachgutachten Verwaltungsbehörden sowie Gerichte. Daneben ist sie in die internationale technische Zusammenarbeit eingebunden und im Bereich „Messwesen – Normung – Prüftechnik – Qualitätssicherung" (MNPQ) als nationale Institution für die Prüftechnik zuständig. Die Mitarbeiter der Bundesanstalt wirken in zahlreichen Fachgremien, gesetzlichen Körperschaften und normensetzenden Institutionen an der Aufstellung von technischen Regeln und Sicherheitsbestimmungen mit und vertreten die Bundesrepublik in nationalen und supranationalen Einrichtungen.

Der Status

Die BAM ist als technisch-wissenschaftliche Bundesoberbehörde im Geschäftsbereich des Bundesministeriums für Wirtschaft und Technologie Nachfolgeinstitution des 1871 gegründeten Staatlichen Materialprüfungsamtes sowie der 1920 gebildeten Chemisch-Technischen Reichsanstalt (CTR). Sie hat dementsprechend die Funktion einer materialtechnischen und chemisch-technischen Bundesanstalt. In ihr sind etwa 1600 Mitarbeiter, darunter mehr als 700 Wissenschaftler und Ingenieure, auf dem Stammgelände in Berlin-Lichterfelde sowie auf den Zweiggeländen in Berlin-Steglitz und Berlin-Adlershof tätig.

Dr. rer. nat. Frank Liebenau
Technische Universität Berlin
Fachbereich Mathematik

Dr. Ing. Reinald Rudolphi
Bundesanstalt für Materialforschung
und -prüfung

Das Verfahren von Gear mit BICG-Löser Grundlagen, Algorithmen und Anwendungen auf Temperaturberechnungen aus dem Brand- und Wärmeschutz

Forschungsbericht 247
Berlin, 2001

Autor address:
Frank Liebenau, Technische Universität Berlin, Strasse des 17. Juni 136, Berlin, Germany
E-mail address: liebau@dr-liebau.de
URL: www.dr–liebau.de

Reinald Rudolphi, BAM, 12205 Berlin, Germany
E-mail address: Reinald.Rudolphi@gmx.de

Impressum

Forschungsbericht 247
Oktober 2000
Herausgegeben von
Bundesanstalt für Materialforschung und -prüfung (BAM)
Unter den Eichen 87, 12205 Berlin
Telefon (030) 81 04-0
Telefax (030) 8 11 20 29

Herstellung und Verlag:
Wirtschaftsverlag NW
Verlag für neue Wissenschaft GmbH
Bürgermeister-Smidt-Str. 74-76, 27511 Bremerhaven
Postfach 10 11 10, 27611 Bremerhaven
Telefon (0471) 9 46 44-0
Telefax (0471) 9 45 44 77/88

Layout: Referat G.3

ISSN 0938-5533
ISBN 3-89701-632-X

ZUSAMMENFASSUNG. Wird das Mehrschrittverfahren von W. Gear [8] zur Integration eines Systems gewöhnlicher Differentialgleichungen in der Implementation EPISODE [16] auf die nichtlineare Wärmeleitungsgleichung angewandt, so ergeben sich aufgrund der Bandstruktur der Jacobi–Matrix Speicherplatzprobleme, da EPISODE eine LU–Zerlegung mit Pivot–Wahl benutzt. Der Bericht beschreibt Maßnahmen, mit denen das Verfahren an die hier vorliegende Situation angepaßt wird. Durch die Verwendung des Speicherschemas für *Sparse*–Matrizen aus Liebau, Rudolphi [20] werden nur die Nichtnullelemente der Jacobi–Matrix gespeichert. Liegt insbesondere ein lineares Problem vor, so braucht diese Matrix nur einmal berechnet zu werden.

Das BI–CGSTAB–Verfahren, also eine iterative Methode, ersetzt die LU–Zerlegung bzw. das Lösen des aus dem Korrektor–Schritt stammenden (evtl. nichtsymmetrischen) linearen Gleichungssystems.

Sind τ die Zeit– und h die Ortsschrittweite, so besitzt diese BICG–Variante für moderate Verhältnisse von τ/h^2 günstige Konvergenzeigenschaften.

Weiterhin ist auch eine kurze Beschreibung des in Borland Delphi 4/Object Pascal entwickelten Programms INSTATCP [28] enthalten. Anhand von zwei Anwendungsbeispielen aus dem Wärme– und Brandschutz (Hochlochziegel und brandbeanspruchte bekleidete Stahlstütze) wird gezeigt, daß die Modifizierungen des Gear–Verfahrens effektiv arbeiten.

ABSTRACT. If we apply the multistep method of W. Gear [8] for the integration of a system of ordinary differential equations to the nonlinear heat conduction equation in the implementation EPISODE [16], this results in memory problems due to the band structure of the Jacobian, because EPISODE uses a LU–decomposition with pivoting. The report describes measures which suit this method to the actual conditions. If we use the storage scheme for sparse matrices from Liebau, Rudolphi [20], only the nonzero elements of the Jacobian are stored. In case of a linear problem, this matrix only has to be computed once.

The BI–CGSTAB–method, an iterative method, replaces the LU–decomposition resp. the solution of the linear (and possibly non–symmetric) system of equations resulting from the corrector step.

If τ is the time step length and h the space step length, this BICG–variant has good convergence properties for a moderate ratio of τ/h^2.

The report also includes a short description of the program INSTATCP [28] developed in Borland Delphi 4/Object Pascal. Two applications from the field of heat and fire protection (brick and fire resistant covered steal support) show that the modifications of the Gear method work effectively.

Inhaltsverzeichnis

KAPITEL 1

Einleitung

Die Diskretisierung der instationären nichtlinearen Wärmeleitungsgleichung mittels einer Finiten–Volumen– bzw. Finite–Elemente–Methode führt auf ein nichtlineares Gleichungssystem der Form (siehe z. B. Liebau und Rudolphi [**20**])

$$\text{(P)} \qquad M_h u_h'(t) - A_h(u_h)u_h(t) = f_h(t)$$

für den unbekannten Knotenvektor $u_h(t)$ der Temperaturen zur Zeit t. In vielen praktischen Anwendungen ändern sich die Materialeigenschaften mit der Temperatur, damit sind die Massematrix M_h und auch die Steifigkeitsmatrix $A_h(u_h)$ Funktionen bzgl. der Temperatur.

(P) ist ein System gewöhnlicher Differentialgleichungen (*ODE*s) für $u_h(t)$ und kann deshalb mit einer entsprechenden numerischen Methode gelöst werden. Eine Möglichkeit stellt das Verfahren von Gear [**8**] dar, alternativ wäre z. B. an ein θ–Verfahren zu denken. Will man Gears Algorithmus zur Zeitintegration benutzen, so ergeben sich aufgrund der evtl. sehr großen Dimension des Knotenvektors (d. h. der feinen Auflösung in "Ortsrichtung") Speicherplatz– und Rechenzeitprobleme, die im Zusammenhang mit der Jacobi–Matrix und dem Newton–Verfahren auftreten. Die Ursache hierfür liegt in der Verwendung eines direkten Verfahrens zur Lösung von linearen Gleichungssystemen.

Der Algorithmus von Gear ist primär *nicht* darauf ausgerichtet Problemstellungen der Art (P), d. h. parabolische Probleme, zu lösen, sondern ist allgemeiner konzipiert. Somit ist die Benutzung einer LU–Zerlegung zur Behandlung der linearen Gleichungssysteme nicht weiter verwunderlich. Beschränkt man sich allerdings auf Probleme der Form (P), so ist eine Modifizierung des Speicherformats, angepaßt auf die schwachbesetzte Struktur der Matrizen, und des Lösers, angepaßt auf die Eigenschaften des entstehenden Gleichungssystems, von Vorteil.

Das zweite Kapitel des Berichtes widmet sich den Mehrschrittverfahren, und es wird u. a. auf die Begriffe Steifheit und Stabilität eingegangen. Der dritte Abschnitt beschreibt dann den Algorithmus von Gear. Ein Vorschlag für ein Speicherformat für schwachbesetzte Matrizen ist Gegenstand des Kapitels 4. In den folgenden Abschnitten 5 und 6 werden eine CG–Variante — das direkte Verfahren zur Lösung von linearen Gleichungssystemen wird durch eine iterative Methode ersetzt — und zwei Anwendungsbeispiele vorgestellt.

Das Computerprogramm. Das PC–Computerprogramm INSTATCP arbeitet auf der Basis der beschriebenen Algorithmen (Algorithmen 5, 6 und 8). Es benutzt eine Finite–Volumen–Diskretisierung in Ortsrichtung und den Gear–Löser in Verbindung mit dem BICG-Gleichungsauflöser für die Zeitintegration. Mit dem in Borland Delphi 4/Object Pascal entwickelten Programm können unter Windows 95/98/NT geometrisch komplizierte, dreidimensionale Brand- und Wärmeschutzprobleme gelöst werden. Die maximale Anzahl der Unbekannten beträgt bei dieser Fassung 131040. Es werden gerade prismatische (2 parallele Dreieckselemente spannen im Raum das Volumenelement auf) und Rechteckelemente mit Knotenpunkten in den Ecken verwendet. Zur Vereinfachung der Knotennumerierung wurde eine Bandbreitenoptimierung nach Cuthill–McKee mit der RCM–Methode in das Programm

integriert. Die Implementierung dieses Verfahrens gestattet eine beliebige Knotennumerierung, wobei auch Knotennummern weggelassen oder nicht benutzt werden können.

Das zur Integration eingesetzte Mehrschrittverfahren, das nach der Prädiktor-Korrektor-Methode mit automatischer Schrittweitensteuerung arbeitet, liefert auch bei sehr unterschiedlichen Schichtdicken und Wärmekapazitäten der Baustoffe (steife Differentialgleichungen) verläßliche Resultate.

INSTATCP vermeidet die bei der Lösung großer Problemstellungen auftretenden Speicherplatzprobleme elegant durch Einsatz eines iterativen BICG-Verfahrens bei kompakter Speicherung der Jacobi-Matrix. Nur so ist überhaupt eine Anzahl von 131040 Unbekannten noch lösbar. Im affinen Fall, d. h. bei nicht temperaturabhängigen Funktionen (z.B. konstante Wärmeleitfähigkeiten, kein Wärmeübergang durch Strahlung), wird die Jacobi-Matrix nur einmal aufgestellt und so bei Wärmebrückenberechnungen viel Rechenzeit gespart.

Es ist anwendbar auf aus verschiedenen Baustoffen zusammengesetzte Bauteilquerschnitte, berücksichtigt nichtlineare Wärmeübergangs- und Strahlungsrandbedingungen zwischen dreiecks- und rechteckförmigen Oberflächen sowie Temperatur- und Zeitabhängigkeiten der vorzugebenden Kennlinien. Es ist möglich, Phasenumwandlungen, d. h. Änderungen des Aggregatzustandes (Verdampfung oder Gefrieren der in Baustoffen vorhandenen Feuchte, Schmelzen von Eisschichten, Gefrieren von Wasserschichten u.a.), ebenso numerisch zu erfassen wie die konvektive Wärmeübertragung strömender Medien in Rohren und Kanälen und Wärmestromeinspeisungen.

Ein Anwendungsbeispiel für die Berechnung der Temperaturen und Wärmeströme bei einer komplizierten Baukonstruktion zeigt Abbildung 1.1. Die Visualisierung erfolgte unter Benutzung von VTK [**29**].

Danksagungen

Wir danken der Georg Rimmele KG, Ehingen, für die Überlassung der technischen Daten zum Hochlochziegel und Herrn Dipl.–Math. Guido Büttner, der die Rechnungen zum BICG–Verfahren bzgl. der Modellprobleme (5.9) durchgeführt hat.

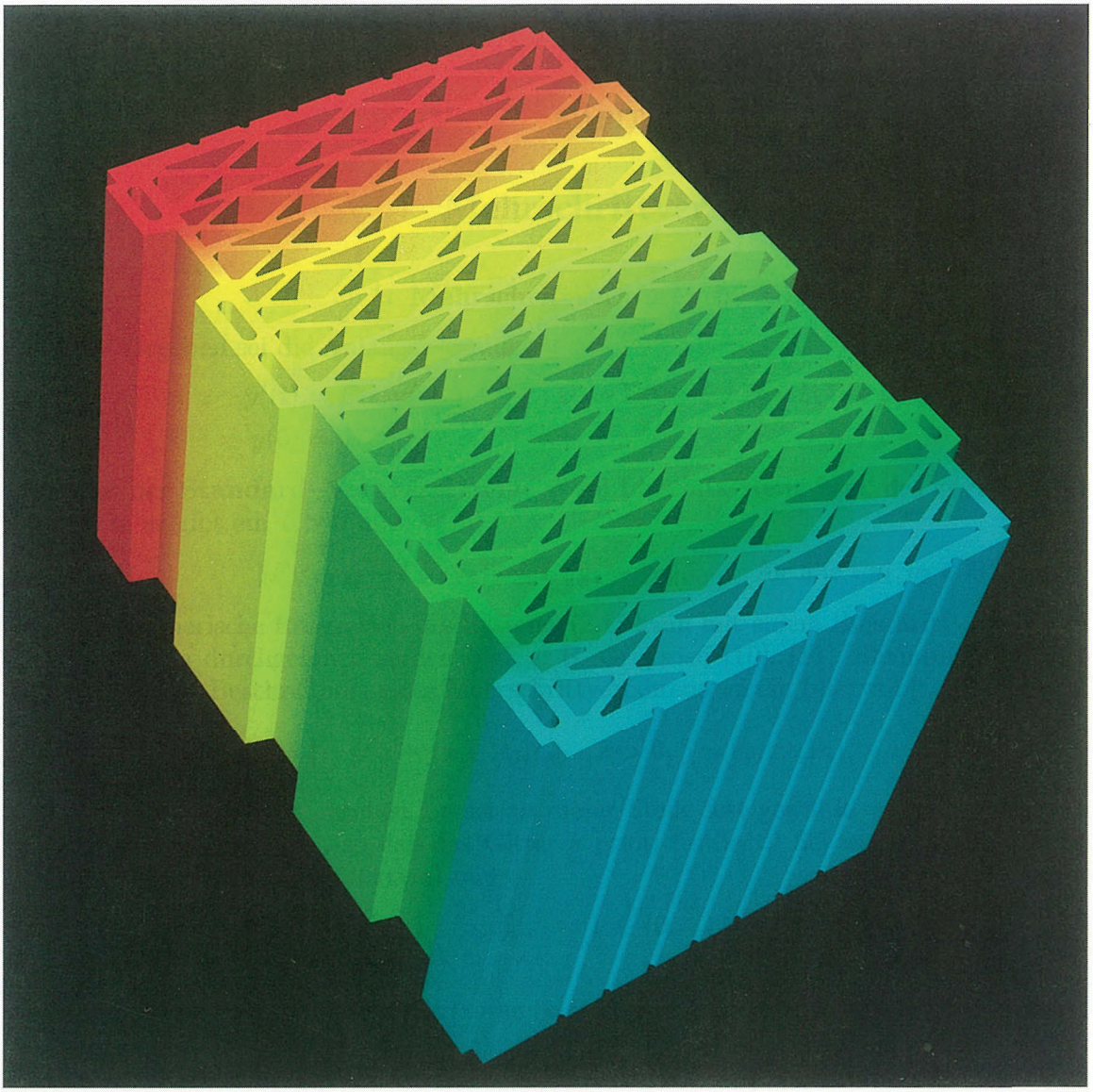


Abbildung 1.1: Mit INSTATCP errechnete Temperaturverteilung in einem Hochlochziegel. Visualisierung unter Benutzung von vtk [**29**].

KAPITEL 2

Mehrschrittverfahren

1. Mehrschrittverfahren im Überblick

Gegeben sei die Anfangswertaufgabe

$$\begin{aligned} &\mathbf{y}'(t) = F(t, \mathbf{y}(t)), \qquad \mathbf{y} : \mathbb{R} \to \mathbb{R}^n, \quad F : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}^n, \quad t > t_0 \\ &y(0) = y_0 \end{aligned} \tag{2.1}$$

mit der **Standard–Voraussetzung:** F sei Lipschitz–stetig bzgl. des zweiten Argumentes, d. h. es gibt ein $L > 0$ mit

$$\|F(t, \mathbf{y}_1) - F(t, \mathbf{y}_2)\| \leq L \, \|\mathbf{y}_1 - \mathbf{y}_2\|, \qquad \text{für alle} \quad y_1, y_2 \in \mathbb{R}^n. \tag{2.2}$$

Die numerische Integration des Systems (2.1) kann mit einem Einschritt– oder Mehrschrittverfahren durchgeführt werden (vgl. z. B. Grigorieff [**10**], [**11**], Shampine [**30**] und Werner [**35**]). Zur Beschreibung eines Mehrschrittverfahrens sei ein Intervall $[t_0, t_e]$ und das Gitter

$$I_\tau := \{t_j = a + j\,\tau \in [t_0, t_e] \; : \; \tau > 0, \, j = 0, \ldots, m\}, \quad \tau \leq \frac{t_e - t_0}{m},$$

zur Schrittweite τ gegeben. τ sei an dieser Stelle fest gewählt, ändert sich also im Laufe der Rechnung nicht, so daß das Gitter a priori festliegt. Sei weiter u_τ eine Funktion mit $u_\tau : I_\tau \to \mathbb{R}^n$ (Gitterfunktion) und

$$u_\nu, \qquad \nu \in \{0, \ldots, n\},$$

die Abkürzung für $u_\tau(t_\nu)$.

Ein lineares Mehrschrittverfahren (k–Schrittverfahren) zur Bestimmung einer Näherungslösung u_τ des Systems (2.1) ist gegeben durch die Vorgabe der Startwerte

$$u_\nu, \quad \nu = 0, \ldots, k-1, \tag{2.3a}$$

und einer Differenzengleichung

$$\sum_{\nu=0}^{k} \alpha_\nu \, u_{j+\nu} = \tau \sum_{\nu=0}^{k} \beta_\nu F(t_{j+\nu}, u_{j+\nu}), \quad t_j \in I'_\tau := I_\tau \setminus \{t_{m-k+1}, \ldots, t_m\}. \tag{2.3b}$$

Die Wahl $t_j \in I'_\tau$ hat formale Gründe: $t_j \in I'_\tau \Rightarrow t_{j+\nu} \in I_\tau$. Dem durch (2.3b) spezifizierten Mehrschrittverfahren werden zwei Polynome zugeordnet:

$$\rho(z) := \sum_{\nu=0}^{k} \alpha_\nu \, z^\nu, \qquad \text{erstes charakteristisches Polynom,} \tag{2.4a}$$

und

$$\sigma(z) := \sum_{\nu=0}^{k} \beta_\nu \, z^\nu, \qquad \text{zweites charakteristisches Polynom.} \tag{2.4b}$$

Umgekehrt beschreiben auch die beiden Polynome (ρ, σ) ein Mehrschrittverfahren. In einer Rechnung wird ausgehend von der Kenntnis der k Näherungen

$$u_j, u_{j+1}, \ldots, u_{j+k-1}$$

aus der Differenzengleichung (2.3b) die neue Näherung u_{j+k} im Punkt t_{j+k} bestimmt. Ist der Koeffizient $\beta_k = 0$, so liegt ein explizites Mehrschrittverfahren vor, ansonsten heißt das Verfahren implizit, da der gesuchte Wert u_{j+k} auch in der rechten Seite von (2.3b) auftritt.

Man nennt ein Verfahren (2.3) *konsistent* mit der Aufgabe (2.1), falls (2.3) das Problem (2.1) approximiert, d. h. die diskreten Gleichungen "streben" gegen die kontinuierliche Gleichung, falls die Schrittweite τ gegen Null strebt: "(2.3) $\to$ (2.1) für $\tau \to 0$". Die Konsistenzordnung gibt die Güte dieser Approximation an. Setzt man die Lösung y des Systems (2.1) in die Gleichungen (2.3b) ein, so gilt

$$\frac{1}{\tau} \sum_{\nu=0}^{k} \alpha_\nu \, y(t + \nu\tau) = \sum_{\nu=0}^{k} \beta_\nu F(t + \nu\tau, y(t + \nu\tau) + r_\tau(t), \quad t \in I'_\tau,$$

mit einem Rest r_τ, der als lokaler Verfahrensfehler bzw. lokaler Diskretisierungsfehler bezeichnet wird. Durch die Wahl der Koeffizienten α_ν und β_ν wird nun versucht, r_τ möglichst klein ausfallen zu lassen (vgl. auch Grigorieff [**11**]).

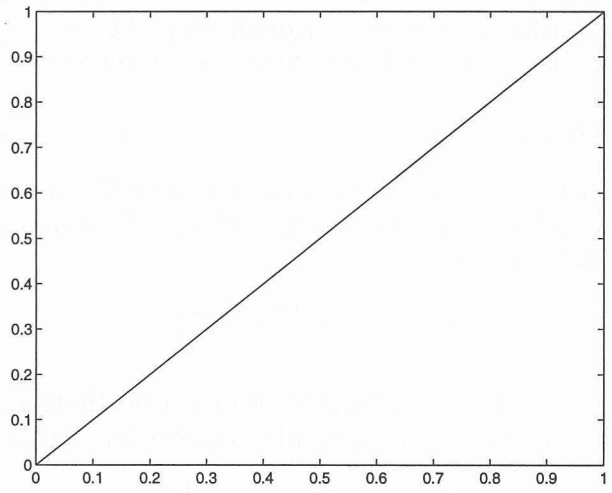


Abbildung 2.1: Lösung von (2.5).

DEFINITION 2.1. Konsistenz. Das Verfahren (2.3) heißt *konsistent* mit der Aufgabe (2.1), wenn für $\tau \to 0$

$$|u_\nu - y(t_\nu)| \to 0, \quad \nu = 0, \ldots, k-1,$$

und

$$|r_\tau(t)| \to 0, \quad t \in I'_\tau$$

gilt. Sind für ein $p > 0$ die Bedingungen

$$|u_\nu - y(t_\nu)| = \mathcal{O}(\tau^p), \quad \nu = 0, \ldots, k-1, \quad \text{und} \quad |r_\tau(t)| = \mathcal{O}(\tau^p), \quad t \in I'_\tau,$$

erfüllt, so besitzt das Verfahren die Konsistenzordnung p.

Mit Hilfe der Polynome (2.4) können Aussagen zur Konsistenzordnung des Verfahrens formuliert werden (Grigorieff [**11**], Werner [**35**]): Seien die Startwerte hinreichend genau, dann gilt

- sind die Gleichungen $\rho(1) = 0$ und $\rho'(1) - \sigma(1) = 0$ erfüllt, so ist das entsprechende Mehrschrittverfahren konsistent,
- das lineare Mehrschrittverfahren (ρ, σ) besitzt mindestens die Konsistenzordnung p, falls $\xi = 1$ p–fache Nullstelle von $\rho(\xi)/\log\xi - \sigma(\xi)$ ist.

Die Hoffnung ist nun, daß mit einem konsistenten Verfahren, d. h. einen mit der Schrittweite τ fallenden lokalen Verfahrensfehler, auch die Lösung u_τ der Differenzengleichungen (2.3) der kontinuierlichen Lösung y möglichst nahekommt (Konvergenz). Dieses ist bei Mehrschrittverfahren, im Gegensatz zu den Einschrittverfahren, nicht mehr unbedingt der Fall.

DEFINITION 2.2. Konvergenz. Das Verfahren (2.3) heißt *konvergent* zur Aufgabe (2.1), falls

$$\|e_\tau\| := \|u_\tau - y\| \to 0, \qquad \text{für} \quad \tau \to 0.$$

Das Verfahren hat die Konvergenzordnung $p > 0$, wenn $\|e_\tau\| = \mathcal{O}(\tau^p)$ gilt.

Das folgende Beispiel zeigt, daß Konsistenz nicht hinreichend für die Konvergenz eines Verfahrens ist.

BEISPIEL 2.1. Gegeben sei das (sehr einfache) Anfangswertproblem

$$y'(x) = 1, \qquad x \in (0,1], \qquad y(0) = 0, \tag{2.5}$$

mit der exakten Lösung $y(x) = x$. Eine kleine Störung in den Daten, etwa

$$z'(x) = 1 + \delta, \qquad x \in (0,1], \qquad y(0) = \epsilon,$$

führt zur Lösung $z(x) = (1+\delta)\,x + \epsilon = y(x) + \delta\,x + \epsilon$.

$$|y(x) - z(x)| \le |\delta|\,x + |\epsilon| \le |\delta| + |\epsilon|. \tag{2.6}$$

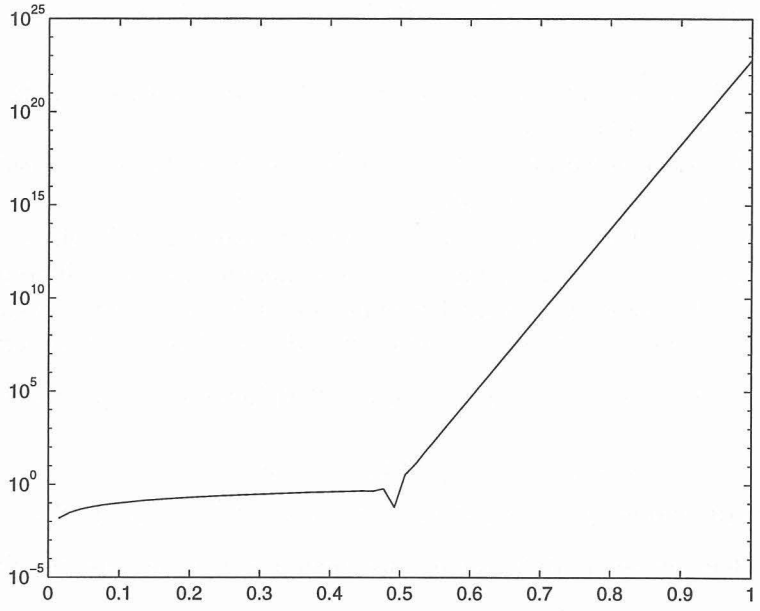
Die Störungen in der Lösung sind klein, d. h. die Aufgabe selbst ist stabil.

Das explizite Mehrschrittverfahren

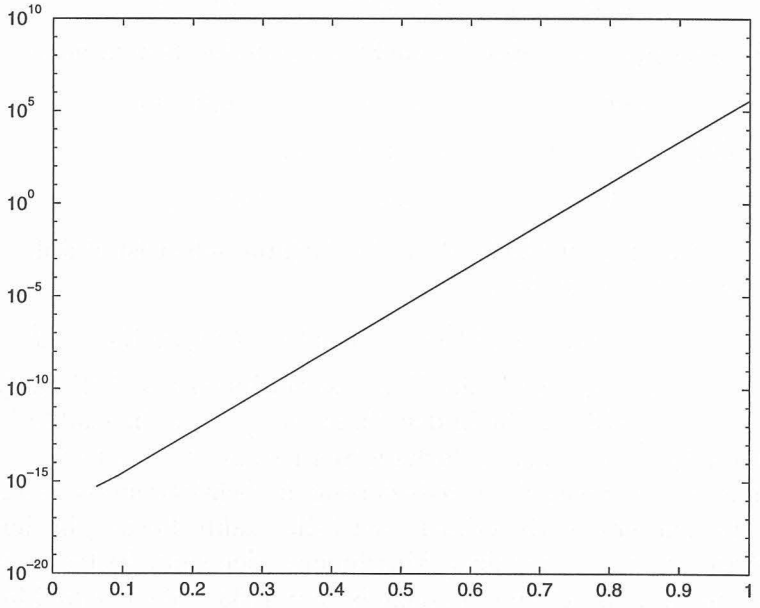
$$\eta_{j+2} + 4\,\eta_{j+1} - 5\,\eta_j = \tau\,\left(4\,F(x_{j+1}, \eta_{j+1}) + 2\,F(x_j, \eta_j)\right), \qquad j \ge 0, \tag{2.7}$$

mit den Startwerten η_0 und η_1 besitzt die Konsistenzordnung $p = 3$ (Stoer [**33**]). Eine Auswertung von (2.7) für $\tau = 1/32$ ergibt in den Punkten $x_j = j\,\tau$ die exakte Lösung: $\eta_j = x_j$, $j = 0, \ldots, 32$ (Abb. 2.1). Ändert man jedoch die Schrittweite zu $\tau = 1/33$, so ist das Ergebnis der Rechnung völlig unbrauchbar. Eine Reduzierung der Schrittweite verschärft die Situation nur (Abb. 2(a)). In dem ersten Experiment wird die exakte Lösung in den Gitterpunkten erreicht, da alle Werte $x_j = j\,\tau$ und alle η_j Maschinenzahlen sind. Es treten keine Rundungsfehler in der Rechnung auf. Im Gegensatz dazu ist $\tau = 1/33 = 0.\overline{03}$ *keine* Maschinenzahl und kann nur mit einem Rundungsfehler dargestellt werden, dessen Fortpflanzung die Rechnung wertlos macht. Stört man in der Rechnung zur Schrittweite $\tau = 1/32$ den Anfangswert η_0 um $\epsilon = 10^{-16}$, so liegt der Fehler im Punkt $x = 1$ in der Größenordnung $\mathcal{O}(10^6)$ (Abb. 2(b)) statt der nach (2.6) zu erwarteten Größenordnung ϵ: Das Verfahren ist instabil.

Von einem Mehrschrittverfahren, von einem numerischen Verfahren i. a. , wird man verlangen, daß es stabil ist. Für ein lineares Mehrschrittverfahren kann diese Forderung mit Hilfe der sogenannten Wurzelbedingung, sie stammt aus der Theorie zu den Differenzengleichungen, formuliert werden.



(a) Lösung von (2.7) mit $\tau = 1/65$.



(b) Fehler $|\eta_j - x_j|$ zum gestörten Anfangswert $\eta_0 = \epsilon$, $\tau = 1/32$.

Abbildung 2.2: Verfahren (2.7) ist instabil.

DEFINITION 2.3. Wurzelbedingung. Ein Polynom Ψ erfüllt die *Wurzelbedingung*, falls $|\lambda| \leq 1$ für jede Nullstelle λ von Ψ gilt und im Fall $|\lambda| = 1$ die Wurzel einfach ist.

Stabilität und Konsistenz eines Verfahrens ergeben Konvergenz.

SATZ 2.1. *Sei F hinreichend oft differenzierbar. Ein lineares Mehrschrittverfahren (ρ, σ) ist genau dann konvergent, wenn ρ die Wurzelbedingung erfüllt und es konsistent ist.*

BEMERKUNG 2.1. (Satz von Dahlquist) Ein konvergentes lineares k–Schrittverfahren besitzt maximal die Konsistenzordnung

$$p \leq \begin{cases} k+1 & \text{falls } k \text{ ungerade} \\ k+2 & \text{falls } k \text{ gerade} \end{cases}.$$

BEISPIEL 2.2. (*Explizites lineares Mehrschrittverfahren*) Für $k = 4$ gibt Grigorieff [11] das folgende Nyström–Verfahren zur Lösung von (2.1) an $(j = 0, \ldots, n-4)$

$$u_{j+4} - u_{j+2} = \tau \left(-F(t_j, u_j) + 4\,F(t_{j+1}, u_{j+1}) - 5\,F(t_{j+2}, u_{j+2}) + 8\,F(t_{j+3}, u_{j+3})\right).$$

Es ist demnach

$$\rho(z) = z^4 - z^2 \quad \text{und} \quad \sigma(z) = 8\,z^3 - 5\,z^2 + 4z - 1.$$

Eine numerische Umsetzung des Verfahrens ist Algorithmus 2.2.

Algorithmus 1 Beispiel: explizites lineares Mehrschrittverfahren

```
procedure F(j:integer)
{ berechnet F(t_j, u_j)}
...
Setze u_0 := y(0);
Berechne mit einem geeigneten Verfahren die Startwerte u_1, u_2 und u_3;
for  j = 0 to n − 4 do
  u_{j+4} = u_{j+2} + τ (−F(j) + 4F(j + 1) − 5F(j + 2) + 8F(j + 3));
end for
```

BEISPIEL 2.3. (*implizites lineares Mehrschrittverfahren*) Das Simpson–Verfahren ($k = 2$) lautet

$$u_{j+2} - u_j = \frac{\tau}{3}\left(F(t_j, u_j) + 4\,F(t_{j+1}, u_{j+1}) + F(t_{j+2}, u_{j+2})\right), \qquad j = 0, \ldots, n-2.$$

Diese Gleichung kann i. a. nicht mehr nach u_{j+2} aufgelöst werden, sondern stellt ein nichtlineares Problem dar:

$$u_{j+2} - \frac{\tau}{3}F(t_{j+2}, u_{j+2}) = u_j + \frac{\tau}{3}\left(F(t_j, u_j) + 4\,F(t_{j+1}, u_{j+1})\right).$$

Algorithmus 2 beschreibt (stark vereinfacht) eine numerische Implementation des Verfahrens.

Algorithmus 2 Beispiel: implizites lineares Mehrschrittverfahren

```
procedure F(j:integer)
{ berechnet F(t_j, u_j)}
procedure NonLinSolve( u, f);
{ löst das nichtlineare Problem u - (τ/3) F(t,u) - f = 0}
...
Setze u_0 := y(0)
Berechne mit einem geeigneten Verfahren den Wert u_1
for  j = 0 to n - 2 do
  f := u_j + (τ/3) (F(t_j, u_j) + 4 F(t_{j+1}, u_{j+1}));
  wähle eine Schätzung u^(0)_{j+2} für u_{j+2}
  NonLinSolve(u^(0)_{j+2}, f);
end for
```

2. Fixpunktiteration und Newton–Verfahren

Das im impliziten Mehrschrittverfahren entstehende nichtlineare Gleichungssystem für den gesuchten Wert u_{j+k}

$$u_{j+k} + \sum_{\nu=0}^{k-1} \alpha_\nu u_{j+\nu} = \tau \beta_k F(t_{j+k}, u_{j+k}) + \tau \sum_{\nu=0}^{k-1} \beta_\nu F_{j+\nu} \tag{2.8}$$

ist iterativ zu lösen. D. h. ausgehend von einer Schätzung $u_{j+k}^{(0)}$ für u_{j+k} werden sukzessive weitere Näherungen $u_{j+k}^{(m)}$, $m = 1, \ldots$, mit einer Fixpunktiteration oder einer Variante des Newton–Verfahrens erzeugt, bis (hoffentlich) eine gewünschte Genauigkeit erreicht wird. Faßt man die bekannten Größen im Vektor $q := -\sum_{\nu=0}^{k-1} \alpha_\nu u_{j+\nu} + \tau \sum_{\nu=0}^{k-1} \beta_\nu F_{j+\nu}$ zusammen und definiert noch

$$G : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}^n, \quad G(t,u) := \tau\, \beta_k F(t,u) + q,$$

so schreibt sich (2.8) als Fixpunktgleichung

$$u_{j+k} = G(t_{j+k}, u_{j+k})$$

bzw. als Nullstellenproblem

$$u_{j+k} - G(t_{j+k}, u_{j+k}) = 0.$$

Gilt die Standard–Voraussetzung (2.2), so kann durch Beschränkung der Schrittweite τ die Konvergenz der Fixpunktiteration garantiert werden.

BEMERKUNG 2.2. **Fixpunktiteration.** F aus (2.1) erfülle die Standard–Voraussetzung (2.2) mit der Lipschitz–Konstanten L. Die Schrittweite τ sei so gewählt, daß

$$\tau\, |\beta_k|\, L < 1$$

gilt, dann konvergiert die Fixpunktiteration

$$u^{(m+1)} - \tau\, \beta_k\, F(t, u^{(m)}) = \mathbf{q}, \qquad m = 0, 1, \ldots, \tag{2.9}$$

für alle Startwerte $u^{(0)}$ und alle Vektoren $\mathbf{q} \in \mathbb{R}^n$ (denn $\tau\, \beta_k\, F$ ist eine Kontraktion).

Üblicherweise wird für die zweidimensionale Wärmeleitungsgleichung die Lipschitz–Konstante in der Größenordnung $\mathcal{O}(h^{-2})$ liegen, wobei h der typische Diskretisierungsparameter bzgl. des Ortes ist, so daß (2.9) die Wahl der Zeitschrittweite an h koppelt.

Das Newton–Verfahren zur Lösung eines Nullstellenproblems

$$H(x) = 0, \qquad H : \mathbb{R}^n \to \mathbb{R}^n, \quad x \in \mathbb{R}^n,$$

lautet:

- wähle einen Startwert $x^{(0)} \in \mathbb{R}^n$,
- berechne

$$x^{(m+1)} = x^{(m)} - \left(D\,H(x^{(m)})\right)^{-1} H(x^{(m)}), \qquad m = 0, 1, \ldots, \tag{2.10}$$

bis eine hinreichend genaue Näherung gefunden ist oder eine maximale Anzahl von Iterationen überschritten wird.

Dabei ist $D\,H(x^{(m)})$ die *Jacobi*– oder Funktionalmatrix der Abbildung H an der Stelle $x^{(m)}$. In jedem Iterationsschritt ist die Jacobi–Matrix neu zu berechnen und ein Gleichungssystem der Form

$$D\,H(x^{(m)})\,\delta = H(x^{(m)}) \tag{2.11}$$

zu lösen. Um diesen eventuell großen Rechenaufwand zu reduzieren, ist es möglich, in der Iteration (2.10) die Jacobi–Matrix $D\,H(x^{(m)})$ durch eine geeignete Approximation zu ersetzen, z. B. $D\,H(x^{(0)})$ statt $D\,H(x^{(m)})$ (*chord*–method). Eine Übersicht der Newton–Varianten findet sich in Kelley [**19**].

Zu (2.8) gehört die Funktionalmatrix

$$P := D(u - G(t,u)) = I - DG(t,u) = I - \tau\,\beta_k DF(t,u). \tag{2.12}$$

Die Chord–Methode ist dann durch die Vorschrift

$$u^{(m+1)} = u^{(m)} - \left(I - \tau\,\beta_k DF(t,u^{(0)})\right)^{-1} (u^{(m)} - \tau\,\beta_k F(t,u^{(m)}) - q), \qquad m = 0, 1, \ldots,$$

gegeben (Algorithmus 3). Im Fall der BDF–Formeln (3.2) ist $q = -\sum_{\nu=0}^{k-1} \alpha_\nu u_{j+\nu}$.

Algorithmus 3 Chord–Methode

```
Sei u_c ein fest gewählter Vektor
step=0;
repeat
  löse das Gleichungssystem
```

$$(I - \tau\,\beta_k DF(t,u_c))\;\delta = u^{(m)} - \tau\,\beta_k F(t,u^{(m)}) - q;$$

```
  setze dann:  u^(m+1) = u^(m) - δ;
  step++;
until  (‖δ‖ < tol) or (step > maxIt)
```

3. Prädiktor–Korrektor–Verfahren

Berechnet man den Startwert $u_{j+k}^{(0)}$ mit einem expliziten Mehrschrittverfahren $(\rho^\star, \sigma^\star)$ und führt anschließend eine feste Anzahl von Fixpunktiterationen bzgl. des impliziten Mehrschrittverfahrens (ρ, σ) durch, so gelangt man zu den Prädiktor–Korrektor–Verfahren (Algorithmus 4). Diese Verfahren bezeichnet man auch als PC–Verfahren bzw. $P(EC)^lE$–Verfahren. Eine genaue Diskussion findet sich wieder in Grigorieff [**11**] und Werner [**35**], insbesondere auch die folgende Aussage: Ist $p^\star$ die Konsistenzordnung des Prädiktors und p

Algorithmus 4 Prädiktor–Korrektor–Verfahren

1. Berechne mit einem geeigneten Verfahren die Werte $u_{j+\nu}$ und setze dann
$$F_{j+\nu} = F(t_{j+\nu}, u_{j+\nu}), \quad \nu = 0, \dots, k-1.$$
2. Zur Berechnung des Wertes u_{j+k} verfahre wie folgt:

   Bestimme $u_{j+k}^{(0)}$ aus
$$\text{(P)} \qquad u_{j+k}^{(0)} + \sum_{\nu=0}^{k-1} \alpha_\nu^\star u_{j+\nu} = \tau \sum_{\nu=0}^{k-1} \beta_\nu^\star F_{j+\nu}.$$
   Für $\mu = 1, \dots, l$ sei
$$\text{(E)} \qquad F_{j+k} = F(t_{j+k}, u_{j+k}^{(\mu-1)})$$
$$\text{(C)} \qquad u_{j+k}^{(\mu)} + \sum_{\nu=0}^{k-1} \alpha_\nu u_{j+\nu} = \tau\beta_k F_{j+k} + \tau \sum_{\nu=0}^{k-1} \beta_\nu F_{j+\nu}.$$
   Setze dann $u_{j+k} := u_{j+k}^{(l)}$ und $F_{j+k} = F(t_{j+k}, u_{j+k})$.
3. $j := j+1$, falls $t_j < T$ `Goto 2`, sonst `stop`.

die des Korrektors, so erreicht man mit bestimmten Voraussetzungen (u. a. Stetigkeit von F, Lipschitz–Stetigkeit von $F(t,.)$) für das $P(EC)^l E$–Verfahren die Konvergenzordnung

$$p_l = \min(p^\star + l, p). \tag{2.13}$$

4. Steifheit und Stabilität

Wird das lineare Mehrschrittverfahren (ρ, σ) auf die "Testgleichung" $y' = Ay$, mit diagonalisierbarer Matrix A, angewandt, ergibt sich

$$\sum_{\nu=0}^{k} \alpha_\nu u_{j+\nu} = \tau \sum_{\nu=0}^{k} \beta_\nu F(t_{j+\nu}, u_{j+\nu}) = \tau \sum_{\nu=0}^{k} \beta_\nu A u_{j+\nu} = \tau A \sum_{\nu=0}^{k} \beta_\nu u_{j+\nu}$$

oder

$$\sum_{\nu=0}^{k} (\alpha_\nu - \tau A \beta_\nu)\, u_{j+\nu} = 0.$$

Da $A = Q^{-1} D Q$, wobei D die Diagonalmatrix mit den Eigenwerten von A ist, geht die letzte Gleichung mit der neuen Variablen $w := Qu$ über in $\sum_\nu (\alpha_\nu - \tau D \beta_\nu)\, w_{j+\nu} = 0$. Dieses ist ein entkoppeltes System von homogenen Differenzengleichungen, so daß für die i–te Komponente des Vektors w

$$\sum_{\nu=0}^{k} (\alpha_\nu - \tau \lambda_i \beta_\nu)\, w_{i,j+\nu} = 0$$

gilt (die Schrittweite sei dabei fest gewählt). Das charakteristische Polynom dieser Differenzengleichung ist

$$\chi(\zeta; \tau\lambda) := \sum_{\nu=0}^{k} (\alpha_\nu - \tau \lambda_i \beta_\nu)\, \zeta^\nu = \rho(\zeta) - \tau \lambda_i \sigma(\zeta).$$

Satz 2.1 garantiert die Konvergenz eines linearen Mehrschrittverfahrens, falls es konsistent und die Stabilitätsbedingung in Form des Wurzelkriteriums erfüllt ist. Die Konvergenz eines Verfahrens ist eine asymptotische Aussage bzgl. der Schrittweite ($\tau \to 0$). Kleine Schrittweiten, die dann Konvergenz des Verfahrens sichern, können aber für die praktische Anwendung viel zu klein sein (vgl. Beispiel 4). Der Ansatz einer Stabilitätsdiskussion bzgl. der Testgleichung $y' = A\,y$ bietet die Möglichkeit, Aussagen für Schrittweiten $\tau > 0$ zu erhalten, da man nun den Einfluß von $\tau\lambda$ auf die Wurzeln des charakteristischen Polynoms berücksichtigt. Dem linearen Mehrschrittverfahren (ρ, σ) wird das *Stabilitätspolynom*

$$\chi(\xi, z) = \rho(\xi) - z\,\sigma(\xi)$$

zugeordnet. Das Verfahren heißt absolut stabil für ein $z \in \mathbb{C}$, falls $|\xi| < 1$ für alle Nullstellen ξ von $\chi(., z)$ gilt. Der Stabilitätsbereich S eines Verfahrens (ρ, σ) ist definiert durch

$$S := \{z \in \mathbb{C} \,:\, \text{Aus } \chi(\xi_k, z) = 0 \quad \Rightarrow \quad |\xi_k| < 1\}.$$

Liegt $\tau\,\lambda$, $\Re(\lambda) < 0$, im Bereich der absoluten Stabilität, so folgt, daß die Lösungen der Differenzengleichung beschränkt sind, welches den Lösungen der Testgleichung entspricht. Liegt $\tau\,\lambda$ hingegen nicht in S, so wächst die numerische Lösung und stimmt damit quantitativ nicht mit der gesuchten Lösung überein. Das lineare Mehrschrittverfahren (ρ, σ) heißt A–stabil, falls

$$S = \{z \in \mathbb{C} \,:\, \Re z < 0\},$$

d. h., es liegt absolute Stabilität in der gesamten Halbebene $\{z \in \mathbb{C} \,:\, \Re z < 0\}$ vor.

Steife Differentialgleichungssysteme zeichnen sich dadurch aus, daß in der Lösungsschar Lösungen mit stark unterschiedlichem Wachstumsverhalten enthalten sind (siehe Stoer [**33**], Grigorieff [**10**] und z. B. auch Werner [**35**], Shampine [**30**] sowie die Diskussion in Aiken [**1**]). Das folgende System aus Grigorieff [**10**] ist dafür ein Beispiel.

BEISPIEL 2.4. Gegeben sei

$$\begin{pmatrix} y_1' \\ y_2' \end{pmatrix} = \frac{1}{2} \begin{pmatrix} \lambda_1 + \lambda_2 & \lambda_1 - \lambda_2 \\ \lambda_1 - \lambda_2 & \lambda_1 + \lambda_2 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}, \qquad \lambda_1, \lambda_2 < 0, \tag{2.14}$$

wobei λ_1 und λ_2 die Eigenwerte der Koeffizientenmatrix sind. Die allgemeine Lösung dieses Systems ist ($x \geq 0$)

$$y_1(x) = C_1\, e^{\lambda_1 x} + C_2\, e^{\lambda_2 x}$$
$$y_2(x) = C_1\, e^{\lambda_1 x} - C_2\, e^{\lambda_2 x}$$

mit Konstanten C_1 und C_2, die durch die Anfangsbedingung festgelegt sind. Da $\lambda_1, \lambda_2 < 0$, folgt, daß beide Lösungskomponenten beschränkt sind: $|y_i(x)| < |C_1| + |C_2|$ für alle $x \geq 0$. Wird zur numerischen Lösung ein explizites Euler–Verfahren benutzt, so gilt für die Näherungen

$$\eta_{1j} = C_1\,(1 + \tau\lambda_1)^j + C_2(1 + \tau\lambda_2)^j$$
$$\eta_{2j} = C_1\,(1 + \tau\lambda_1)^j - C_2(1 + \tau\lambda_2)^j.$$

Die numerische Lösung η konvergiert nur, falls die Schrittweite so klein gewählt wird, daß

$$|1 + \tau\lambda_1| < 1 \quad \text{und} \quad |1 + \tau\lambda_2| < 1. \tag{2.15}$$

Ist z. B. $\lambda_1 = -1$ und $\lambda_2 = -50$, so wird der Einfluß von $e^{\lambda_2 x}$ auf die Lösung mit wachsenden x immer geringer (für $x = 0.2$ ist $e^{\lambda_2 x} = e^{-10} = 4.5{\cdot}10^{-5}$) und kann schließlich ignoriert werden. Dieses gilt *nicht* für die numerische Integration, da die Stabilitätsbedingungen (2.15) für alle

Integrationsschritte eingehalten werden müssen (vgl. auch die Rechnungen in Grigorieff [**10**]). Aus (2.15) folgt für die Zeitschrittweite

$$\tau \leq \min\{2/|\lambda_1|, 2/|\lambda_2|\} = \frac{2}{|\lambda_2|}.$$

In dem letzten Beispiel trägt die Lösungskomponente zu λ_2 (die steife Komponente zum stark negativen Eigenwert der Systemmatrix) wenig zur Lösung bei, dominiert aber die Wahl der Schrittweite. Bei einer Anwendung des *impliziten* Euler–Verfahrens auf ein lineares System mit konstanten Koeffizienten (z. B. auf (2.14)) zeigt sich, daß die steifen Lösungskomponenten hier keine kleinen Schrittweiten erzwingen. Also wird man von einem Verfahren zur Integration steifer Systeme fordern, daß es die stark abklingenden Lösungskomponenten *unabhängig* von der Schrittweite τ stabil integriert. Die Konkretisierung dieser Forderung führt zu weiteren Stabilitätsbegriffen, wie A(α)–stabil und mehreren anderen, auf deren Erläuterung aber an dieser Stelle verzichtet werden soll, statt dessen sei wieder auf Grigorieff [**10**], [**11**] und Werner [**35**] verwiesen. Dennoch einige Feststellungen:

- Zur numerischen Integration von steifen Differentialgleichungen ist man an A–stabilen Verfahren interessiert (das implizite Euler–Verfahren ist so ein Verfahren). Das folgende Ergebnis von Dahlquist deshalb ist besonders erschreckend:

  SATZ 2.2 (Dahlquist). *Die Konsistenzordnung eines A–stabilen lin. Mehrschrittverfahrens ist höchstens 2.*

  Eine höhere Konsistenzordnung wird mit nichtlinearen Methoden oder linearen Verfahren, die höhere Ableitungen verwenden, erreicht.
- Die Trapezregel (Crank–Nicolson–Verfahren) ist A–stabil mit Konsistenzordnung 2. Die numerischen Lösungen sind zwar beschränkt, aber es können Oszillationen für $\Re(\tau\lambda) << -1$ auftreten, die Trapezregel ist nicht stark absolut stabil.
- Das Verfahren von GEAR ist L–stabil, und damit besonders für steife Probleme geeignet.

KAPITEL 3

Der GEAR–Algorithmus

Eine Verallgemeinerung des impliziten Euler–Verfahrens führt zur Klasse der *backward differentiation formulas* (BDFs). Im Programm DIFSUB aus Gear [**8**] werden diese Verfahren als Korrektor verwendet und deshalb oft auch als *Gear's methods* bezeichnet. Auf die systematische Darstellung sei wieder verzichtet und auf die Literatur (z. B. Werner [**35**]) verwiesen.

1. BDF–Formeln

Der Differenzenoperator

$$\nabla^\nu : \mathbb{R}^n \to \mathbb{R}^n$$

ist rekursiv durch die Vorschrift (u_j und u_{j-1} bekannt)

$$\nabla^0 u_j = u_j, \qquad \nabla^{\nu+1} u_j = \nabla^\nu u_j - \nabla^\nu u_{j-1}$$

erklärt.

BEMERKUNG 3.1. Die explizite Darstellung für ∇^ν ist

$$\nabla^\nu u_{j+k} = \sum_{\mu=0}^{\nu} (-1)^\mu \binom{\nu}{\mu} u_{j+k-\mu}. \tag{3.1}$$

SATZ 3.1 ([**35**]). *Die rückwärtigen Differentiationsformeln (BDF) lauten*

$$\sum_{\nu=1}^{k} \frac{1}{\nu} \nabla^\nu u_{j+\nu} = \tau F_{j+k}, \qquad k \in \mathbb{N}. \tag{3.2}$$

Sie sind konsistent von k–ter Ordnung, genau für $1 \le k \le 6$ sind sie stabil und deshalb auch konvergent von dieser Ordnung. Die BDFs sind für diese Werte von k steif stabil und für $k = 1, 2$ A–stabil.

Die ersten beiden BDF–Formeln sind

$$u_{j+1} = u_j + \tau F_{j+1}, \qquad \text{(implizites Euler–Verfahren) und}$$
$$u_{j+2} = \frac{4}{3} u_{j+1} - \frac{1}{3} u_j + \tau \frac{2}{3} F_{j+1}.$$

Aus (3.1) ergibt sich für das erste charakteristische Polynom

$$\rho(\xi) = \sum_{\nu=0}^{k} \alpha_\nu \xi^\nu = \xi^k \sum_{\nu=1}^{k} \frac{1}{\nu} \left(1 - \frac{1}{\xi}\right)^\nu$$

und entsprechend

$$\sigma(\xi) = \xi^k$$

für das zweite charakteristische Polynom. Somit sind die BDF–Verfahren durch

$$\sum_{\nu=0}^{k} \alpha_\nu u_{j+\nu} = \tau F(t_{j+k}, u_{j+k}), \qquad j \geq k, \tag{3.3}$$

gegeben. In der Tabelle 3.1 sind die Koeffizienten α_ν des ersten charakteristischen Polynoms aufgelistet, sie ist mit den Tabellen 11.1 und 11.2 in Gear [8] identisch (dort ist nur α_k auf 1 normiert, statt wie hier $\beta_k = 1$).

k \ α_ν	α_0	α_1	α_2	α_3	α_4	α_5	α_6
1	-1	1					
2	$\frac{1}{2}$	-2	$\frac{3}{2}$				
3	$-\frac{1}{3}$	$\frac{3}{2}$	-3	$\frac{11}{6}$			
4	$\frac{1}{4}$	$-\frac{4}{3}$	3	-4	$\frac{25}{12}$		
5	$-\frac{1}{5}$	$\frac{5}{4}$	$-\frac{10}{3}$	5	-5	$\frac{137}{60}$	
6	$-\frac{1}{6}$	$-\frac{6}{5}$	$\frac{15}{4}$	$-\frac{20}{3}$	$\frac{15}{2}$	-6	$\frac{147}{60}$

Tabelle 3.1: Koeffizienten des ersten charakteristischen Polynoms der BDFs.

2. Adams–Verfahren

Eine weitere Klasse spezieller Mehrschrittverfahren sind die *Adams*–Verfahren. Sie sind von der Form

$$u_{j+k} - u_{j+k-1} = \tau \sum_{\nu=0}^{\kappa} \gamma_\nu \nabla^\nu F_{j+\kappa}, \quad \kappa \in \{k, k-1\}. \tag{3.4}$$

Für $\kappa = k-1$ ist (3.4) eine explizite Formel:

$$u_{j+k} - u_{j+k-1} = \tau \sum_{\nu=0}^{k-1} \gamma_\nu \nabla^\nu F_{j+k-1}, \tag{3.5}$$

diese Verfahren werden *Adams–Bashforth*–Verfahren genannt. Im Fall $\kappa = k$ liegt ein implizites Verfahren vor, das sogenannte *Adams–Moulton*–Verfahren.

In der Form (2.3b), d. h.

$$u_{j+k} - u_{j+k-1} = \tau \sum_{\nu=0}^{k-1} \beta_\nu F_{j+\nu},$$

sind die Koeffizienten β_ν der ersten sechs Adams–Bashforth–Verfahren durch die Tabelle 3.2 gegeben. Das erste charakteristische Polynom ist

$$\rho(\xi) = \xi^k - \xi^{k-1}$$

und für das zweite charakteristische Polynom findet man

$$\sigma(\xi) = \sum_{\nu=0}^{k-1} \beta_\nu \xi^\nu = \xi^{k-1} \sum_{\nu=0}^{k-1} \gamma_k \left(1 - \frac{1}{\xi}\right)^\nu .$$

k \ β_ν	β_0	β_1	β_2	β_3	β_4	β_5
1	1					
2	$-\frac{1}{2}$	$\frac{3}{2}$				
3	$\frac{5}{12}$	$-\frac{16}{12}$	$\frac{23}{12}$			
4	$-\frac{9}{24}$	$\frac{37}{27}$	$-\frac{59}{24}$	$\frac{55}{24}$		
5	$\frac{251}{720}$	$-\frac{1274}{720}$	$\frac{2616}{720}$	$-\frac{2774}{720}$	$\frac{1901}{720}$	
6	$-\frac{475}{1440}$	$\frac{2877}{1440}$	$-\frac{7298}{1440}$	$\frac{9982}{1440}$	$-\frac{7223}{1440}$	$\frac{4277}{1440}$

Tabelle 3.2: Koeffizienten des 2. char. Polynoms der Adams–Bashforth–Verfahren.

Die entsprechenden Koeffizienten der impliziten *Adams–Moulton*–Verfahren sind z. B. in Grigorieff [**11**], Tabelle 6, zu finden.

SATZ 3.2. *Sei F hinreichend oft differenzierbar. Die Konsistenzordnung der Adams–Verfahren ist $\kappa + 1$.*

BEMERKUNG 3.2. Sei $u_{k+j}^{(k)}$ das Ergebnis der k–Schrittformel (3.5). Es gilt die Rekursionsformel

$$u_{k+j}^{(k)} = u_{k+j}^{(k-1)} + \tau\,\gamma_{k-1}\nabla^{k-1}F_{j+k-1}, \qquad k = 2, 3, \ldots \tag{3.6}$$

Die Koeffizienten γ_ν bestimmen sich aus

$$\gamma_0 = 1, \qquad \gamma_k + \frac{1}{2}\gamma_{k-1} + \frac{1}{3}\gamma_{k-2} + \ldots + \frac{1}{k+1}\gamma_0 = 1, \quad k = 0, \ldots$$

k	0	1	2	3	4	5	6
γ_k	1	$\frac{1}{2}$	$\frac{5}{12}$	$\frac{3}{8}$	$\frac{251}{720}$	$\frac{95}{288}$	$\frac{19087}{60480}$

Tabelle 3.3: γ_k für (3.5) bzw. (3.6).

3. Nordsiek–Form

Sollen zur Berechnung der nächsten Iterierten eine explizite oder implizite Variante der Adams–Verfahren (3.4) oder eine BDF–Formel eingesetzt werden, so sind die Vektoren

$$u_{j+k-1}, u_{j+k-2}, \ldots, u_j \quad \text{und} \quad \tau\,F_{j+k-1}, \ldots, \tau\,F_j$$

notwendig. Setzt man $u'_{j+k-\nu} := F_{j+k-\nu} = F(t_{j+k-\nu}, u_{j+k-\nu})$, so geht dieser Satz von Vektoren über in

$$u_{j+k-1}, u_{j+k-2}, \ldots, u_j \quad \text{und} \quad \tau\,u'_{j+k-1}, \ldots, \tau\,u'_j,$$

oder zusammengefaßt in einem Vektor der Länge $2\,k\,n$

$$\mathbf{y} := \left(u_{j+k-1}, u_{j+k-2}, \dots, u_j, \tau\, u'_{j+k-1}, \dots, \tau\, u'_j\right) \in \mathbb{R}^{2\,k\,n}. \tag{3.7}$$

Damit kann der q–te Schritt eines Mehrschrittverfahrens in der Form (Gear [8])

$$\begin{aligned} \mathbf{y}^{q+1,(0)} &= \mathbf{B}\,\mathbf{y}^{q}, \\ \mathbf{y}^{q+1,(m+1)} &= \mathbf{y}^{q+1,(m)} + \mathbf{c}\,\mathbf{G}(\mathbf{y}^{q+1,(m)}), \qquad m = 0, 1, \dots, l-1 \end{aligned} \tag{3.8}$$

angegeben werden. Die nächste Näherung im Mehrschrittverfahren ist dann $\mathbf{y}^{q+1} := \mathbf{y}^{q+1,(l)}$. Die Matrix $\mathbf{B} \in \mathbb{R}^{2\,k\,n \times 2\,k\,n}$ und der Vektor $\mathbf{c} \in \mathbb{R}^{2\,k}$ sind je nach Verfahren zu wählen. Die Abbildung

$$\mathbf{G} : \mathbb{R}^{2\,k\,n} \to \mathbb{R}^{n}, \qquad \mathbf{G}(\mathbf{y}) := \tau\, F(\mathbf{y}_1) - \mathbf{y}_{k+1} = \tau\, F(u_{j+k-1}) - \tau\, u'_{j+k-1}$$

zeigt an, wie "gut" $\mathbf{y}$ das System (2.1) erfüllt. Das Produkt $\mathbf{c}\,\mathbf{G}$ sei durch

$$\mathbf{c}\,\mathbf{G}(\mathbf{y}) := (c_i\,\mathbf{y}_i)_{i=1}^{2\,k} \in \mathbb{R}^{2\,k\,n}$$

festgelegt (man beachte, daß die i–te Komponente von $\mathbf{y}$ ein Vektor ist: $\mathbf{y}_i \in \mathbb{R}^n$).

BEISPIEL 3.1. Das Adams–Bashforth–Verfahren (3.5) für $k = 3$ und $n = 1$ lautet

$$u_{j+3} = u_{j+2} + \tau \left(\frac{5}{12} F_j - \frac{16}{12} F_{j+1} + \frac{23}{12} F_{j+2} \right).$$

Mit $u'_{j+\nu} := F_{j+\nu}$, $\nu = 0, 1, 2$, sei

$$\mathbf{y} = \left(u_{j+2}, u_{j+1}, u_j, \tau\, u'_{j+2}, \tau\, u'_{j+1}, \tau\, u'_j\right)^T,$$

$$\mathbf{B} = \begin{pmatrix} 1 & 0 & 0 & \frac{23}{12} & -\frac{16}{12} & \frac{5}{12} \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}, \qquad \mathbf{c} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}$$

und $\mathbf{G}(y) = \tau\, F(\mathbf{y}_1) - \mathbf{y}_4$. Für die nächste Iterierte $\mathbf{v}$ in (3.8) folgt mit

$$\mathbf{B}\mathbf{y} = \begin{pmatrix} 1 & 0 & 0 & \frac{23}{12} & -\frac{16}{12} & \frac{5}{12} \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} u_{j+2} \\ u_{j+1} \\ u_j \\ \tau\, u'_{j+2} \\ \tau\, u'_{j+1} \\ \tau\, u'_j \end{pmatrix} = \begin{pmatrix} u_{j+3} \\ u_{j+2} \\ u_{j+1} \\ 0 \\ \tau\, u'_{j+2} \\ \tau\, u'_{j+1} \end{pmatrix} \tag{3.9}$$

und (zur Funktionsauswertung)

$$\begin{aligned} \mathbf{c}\,\mathbf{G}(\mathbf{B}\mathbf{y}) &= (0, 0, 0, \tau F(u_{j+3}), 0, 0)^T \\ \mathbf{v} = \mathbf{B}\mathbf{y} + \mathbf{c}\,\mathbf{G}(\mathbf{B}\mathbf{y}) &= \left(u_{j+3}, u_{j+2}, u_{j+1}, \tau\, F(u_{j+3}), \tau\, u'_{j+2}, \tau\, u'_{j+1}\right)^T \\ &= \left(u_{j+3}, u_{j+2}, u_{j+1}, \tau\, u'_{j+3}, \tau\, u'_{j+2}, \tau\, u'_{j+1}\right)^T. \end{aligned}$$

D. h. (3.8) mit der Wahl (3.9) ist äquivalent zu (3.5). Für Dimensionen $n > 1$ sind die Einträge der Matrix $\mathbf{B}$ durch Einheitsmatrizen mit entsprechenden Vorfaktoren zu ersetzen.

Im Fall der Adams–Methoden wird nicht der ganze Satz von Vektoren in (3.7) benötigt, sondern nur $k+1$ Einträge an den k Punkten $t_j, \ldots, t_{j+k-1}$. Im Beispiel 3.1 werden u_{j+1} und u_j zur Berechnung der nächsten Näherung u_j nicht benötigt und können somit aus dem Vektor $\mathbf{y}$ gestrichen werden. Die Darstellung (3.8) bleibt mit entsprechen verkleinerten Größen $\mathbf{B}$ und $\mathbf{G}$ gültig. Mit Hilfe eines Interpolationsarguments ist es nun möglich, zu dem Basis–Verfahren (also eine der Iterationen (3.4) in der Form (3.8)) ein äquivalentes Verfahren anzugeben, welches mit den $k+1$ Vektoren

$$u_{j+k-1}, \tau\, u'_{j+k-1}, \tau^2\, u''_{j+k-1}/2!, \ldots, \tau^k\, u^{(k)}_{j+k-1}/k! \tag{3.10}$$

den Vektor u_{k+j} berechnet. Dabei ist $u_n^{(r)}/r!$ die skalierte r–te Ableitung von u an der Stelle t_n. Es werden also die Ableitungen von u im Gitterpunkt t_{j+k-1} gespeichert, statt sich die Werte von u bzw. F in k Gitterpunkten zu merken. Grigorieff verweist in [**11**] darauf, daß dieses bei variabler Schrittweite günstig ist. Eine Verfahren, das die Vektoren (3.10) benutzt, liegt in der sogenannten *Nordsiek*–Form vor. Sie werden dabei in einem Vektor (Nordsiek–Vektor)

$$\mathbf{z} := [u_{j+k-1}, \tau\, u'_{j+k-1}, \tau^2\, u''_{j+k-1}/2!, \ldots, \tau^s\, u^{(s)}_{j+k-1}/s!], \quad s \in \mathbb{N}, \tag{3.11}$$

zusammengefaßt. Ist der Vektor $\mathbf{z}$ bekannt, so gilt mittels Taylorentwicklung für die Näherung $u_{j+\nu}$ im Punkt $t_{j+\nu}$

$$u_{j+\nu} = \sum_{i=0}^{s} \mathbf{z}_i \frac{(t_{j+\nu} - t_{j+k-1})^i}{\tau^i} = \sum_{i=0}^{s} \mathbf{z}_i\, (\nu - k + 1)^i.$$

Der Zusammenhang zwischen beiden Formulierungen eines Verfahrens wird durch eine lineare Transformation, also durch eine nichtsinguläre Matrix, hergestellt: Ist $\mathbf{y}$ der (evtl. verkleinerte) Vektor aus (3.7), so existiert eine reguläre Matrix T mit $\mathbf{z} = T\mathbf{y}$. Das Mehrschrittverfahren in der Form (3.8) kann nun für den Nordsiek–Vektor formuliert werden

$$\begin{aligned} \mathbf{z}^{q+1,(0)} &= \mathbf{A}\,\mathbf{z}^q, \\ \mathbf{z}^{q+1,(m+1)} &= \mathbf{z}^{q+1,(m)} + \mathbf{l}\,\mathbf{F}(\mathbf{z}^{q+1,(m)}), \qquad m = 0, 1, \ldots, l-1, \end{aligned} \tag{3.12}$$

wobei $\mathbf{A} = \mathbf{T}\,\mathbf{B}\,\mathbf{T}^{-1}$, $\mathbf{l} = \mathbf{T}\,\mathbf{c}$ und $\mathbf{F}(\mathbf{z}) = \mathbf{G}(\mathbf{T}^{-1}\,\mathbf{z})$. Der Vorteil der Formulierung (3.12) ist nun, daß ein Wechsel der Schrittweite für einen Vektor der Gestalt (3.11) sehr einfach ist. Mit Blick auf (3.11) zeigt sich, daß die Modifikation der Schrittweite um den Faktor $\alpha = \tau_{\text{neu}}/\tau_{\text{alt}}$ einer Multiplikation des Vektor $\mathbf{z}$ mit einer Diagonalmatrix entspricht:

$$\mathbf{z}_{\text{neu}} = C(\alpha)\,\mathbf{z}_{\text{alt}}, \qquad C(\alpha) := \operatorname{diag}(1, \alpha, \alpha^2, \ldots, \alpha^s).$$

Genauere Ausführungen zur Nordsiek–Form gibt Gear in [**8**] und [**7**].

4. Schrittweiten– und Ordnungssteuerung

Durch Steuerung der Schrittweite und der Ordnung des Verfahrens soll erreicht werden, daß mit möglichst großer Schrittweite gerechnet werden kann, d. h. auch mit möglichst geringem Aufwand, ohne eine vorgegebene Fehlertoleranz zu überschreiten. Die Berechnungen der Schrittweite und der Ordnung beruhen dabei auf einer Schätzung des lokalen Diskretisierungsfehlers.

BEMERKUNG 3.3. Für ein lineares Mehrschrittverfahren der Konsistenzordnung p besitzt der lokale Diskretisierungsfehler $r_\tau(t)$ (Regularität vorausgesetzt) die Darstellung

$$r_\tau(t) = C_{p+1}\,\tau^p\,y^{(p+1)}(t) + \mathcal{O}(\tau^{p+1})$$

mit der sogenannten Fehlerkonstanten C_{p+1} (zur Bestimmung von C_{p+1} für das Verfahren (σ,ρ) vgl. Grigorieff [**11**]).

Aus der Differenz der letzten Komponenten zweier aufeinanderfolgender Nordsiek–Vektoren $\mathbf{z}^q$ und $\mathbf{z}^{q+1}$ (hier mit $\nabla^1\mathbf{z}_s$ bezeichnet) erhält man eine Schätzung für die $(s+1)$–te Ableitung:

$$\nabla^1\mathbf{z}_s := (z^{\text{neu}})_s - (z^{\text{alt}})_s = \frac{\tau^s}{s!}\left(u^{(s)}_{j+k-1} - u^{(s)}_{j+k-1}\right) = \frac{\tau^s}{s!}\,\nabla^1 u^{(s)}_{j+k-1} \approx \frac{\tau^{s+1}}{s!}\,u^{(s+1)}_{j+k-1}.$$

Es folgt $(p=s)$

$$\tau\,r_\tau(t) \approx C_{p+1}\,p!\,\nabla^1\mathbf{z}_s.$$

Der Ausdruck auf der rechten Seite kann nun zur Kontrolle der Schrittweite τ benutzt werden. Erfüllt eine Schrittweite den Test

$$C_{p+1}\,p!\,\|\nabla\mathbf{z}_s\|_2 \le \text{tol} \tag{3.13}$$

für $\text{tol} > 0$, so wird sie akzeptiert, sonst verworfen. Eine Schätzung für eine Schrittweite $\alpha\,\tau$, die (3.13) erfüllt, ergibt sich, falls angenommen wird, daß der lokale Diskretisierungsfehler r_τ proportional zu τ^p ist (Gear [**8**]):

$$C_{p+1}\,p!\,\alpha^{p+1}\,\|\nabla\mathbf{z}_s\|_2 = \text{tol}.$$

Entsprechende Rechnungen für die Ordnungen $p+1$ und $p-1$ führen insgesamt auf die Schätzungen

$$\alpha = \begin{cases} \frac{1}{1.4}\left(\frac{\text{tol}}{C_{p+2}\,(p-1)!}\|\nabla^2\mathbf{z}_s\|_2^{-1}\right)^{\frac{1}{p+1}} & \text{für Ordnung } p+1, \\ \frac{1}{1.2}\left(\frac{\text{tol}}{C_{p+1}\,(p-1)!}\|\nabla^1\mathbf{z}_s\|_2^{-1}\right)^{\frac{1}{p}} & \text{für Ordnung } p, \\ \frac{1}{1.3}\left(\frac{\text{tol}}{C_p\,(p-1)!}\|\mathbf{z}_s\|_2^{-1}\right)^{\frac{1}{p-1}} & \text{für Ordnung } p-1. \end{cases} \tag{3.14}$$

Die Zahlen 1.2, 1.3 und 1.4 in (3.14) sind dabei Erfahrungswerte. Gear [**8**] schlägt zur Schrittweiten– und Ordnungssteuerung die folgende Strategie vor:

1. Berechne α nach (3.14), falls
   (a) der Test (3.13) nicht erfüllt ist,
   (b) nach $p+1$ Schritte weder die Schrittweite noch die Ordnung geändert wurde.
2. Falls α neu berechnet worden ist, wähle die Ordnung und die Schrittweite entsprechend (3.14) mit dem größten α Wert. Änderte sich die Ordnung nicht und ist $1 < \alpha \le 1.1$, so behalte die alte Schrittweite bei.

5. Der Algorithmus

Gear wählt in [**8**] (vgl. auch Gear [**7**]) als Prädiktor ein Adams–Bashforth–Verfahren und stellt als Korrektor entweder ein Adams–Moulton–Verfahren oder (für steife Probleme) die ersten fünf BDF–Formeln zur Verfügung. Gears Algorithmus läßt sich somit prinzipiell mittels Algorithmus 4 darstellen (die Schrittweitensteuerung ist in Alg. 4 nicht berücksichtigt).

Algorithmus 5 ist eine grobe Skizze des Verfahrens, ein genaueres Flußdiagramm, welches insbesondere weitere Angaben zum Update der Jacobi–Matrix, zur Ordnungs– und Schrittweitensteuerung und zu den Fehlermeldungen bzw. den möglichen Programmabbrüchen macht, findet sich in Hindmarsh [**16**].

Algorithmus 5 Skizze des Gear–Verfahrens

{Startphase}
Setze die Ordnung auf 1 und berechne u_0'; {u_0 bekannt}
{PC–Schritt mit Stepcontrol}
repeat
label 1:
{Prädiktor–Schritt in Nordsiek–Form:}

$$\mathbf{z}^{n,(0)} = \mathbf{A}\,\mathbf{z}^{(n-1)};$$

{Korrektor–Schritt. Zur Bestimmung von u_{j+k} löse (2.12), d. h. }
$u_{j+k} = (\mathbf{z}^{n,(0)})_1$; $e^n = 0$;
bestimme die Jacobi–Matrix P und evtl. deren LU–Zerlegung;
for $m = 0$ to $M-1$ **do**
$P\,\delta^{(m)} = \mathbf{G}(u_{j+k})$; $u_{j+k} -= \delta^{(m)}$; $e^n -= \delta^{(m)}$;
end for

{Zeitschritt und Schrittweitensteuerung}
if Konvergenz des Korrektors **then**
{ jetzt ist $u_{j+k} = (\mathbf{z}^{n,(0)})_1 + e^n$}
update Nordsiek–Vektor: $\mathbf{z}^n = \mathbf{z}^{n,(0)} + \mathbf{l}\,e^n$;
Ordnungs– und Schrittweitensteuerung;
evtl. Skalierung und Dimensionswechsel von $\mathbf{z}^n$;
else
verkleinere die Schrittweite, bestimme $\mathbf{z}^{(n-1)}$ neu (Skalierung);
if mehrfach keine Konvergenz **then**
Abbruch;
else
Goto 1;
end if
end if

until $t > t_{\text{end}}$ oder ein Abbruchkriterium ist erfüllt

KAPITEL 4

Die Jacobi–Matrix

1. Das Speicherplatzproblem

Zur numerischen Integration (2.1) verwendet der Gear–Algorithmus aus [**14**], [**15**] bzw. [**16**] die Lösung eines Gleichungssystems der Form

$$P x = f, \qquad P = I - \mu J \in \mathbb{R}^{n\times n}, \quad x, f \in \mathbb{R}^n, \quad \mu \in \mathbb{R}. \tag{4.1}$$

Die Matrix J ist dabei eine Approximation der Jacobi–Matrix DF:

$$J \approx DF := \left(\frac{\partial f_i}{\partial y_j}\right)_{i,j=1}^{n}, \quad F \text{ die rechte Seite aus (2.1).}$$

Spaltenweise kann J aus einem Differenzenquotienten bestimmt werden:

BEMERKUNG 4.1. Sei e_j der Einheitsvektor mit $(e_j)_i = \delta_{ij}$.

$$J_j := \frac{F(\mathbf{y} + \eta_j e_j) - F(\mathbf{y})}{\eta_j} \in \mathbb{R}^n, \qquad \eta_j > 0, \tag{4.2}$$

ist eine Näherung für die j–te Spalte von DF, d. h.

$$J := [J_1, \ldots, J_n]. \tag{4.3}$$

BEWEIS. Mit

$$\frac{\partial f_i}{\partial y_j}(t, \mathbf{y}) = \frac{f_i(t, \mathbf{y} + \eta_j e_j) - f_i(t, \mathbf{y})}{\eta_j} + \mathcal{O}(\eta_j)$$

folgt

$$\frac{F(\mathbf{y} + \eta_j e_j) - F(\mathbf{y})}{\eta_j} = \left(\frac{f_i(t, \mathbf{y} + \eta_j e_j) - f_i(t, \mathbf{y})}{\eta_j}\right)_{i=1}^{n} \approx (DF_{ij})_{i=1}^{n}.$$

□

F ist eine Abbildung von $\mathbb{R} \times \mathbb{R}^n$ nach $\mathbb{R}^n$, also von der Form

$$F(t, u) = \begin{pmatrix} f_1(t, u_1, \ldots, u_n) \\ f_2(t, u_1, \ldots, u_n) \\ \vdots \\ f_n(t, u_1, \ldots, u_n) \end{pmatrix}. \tag{4.4}$$

BEMERKUNG 4.2. Seien $\mathcal{I} := \{1, \ldots, n\}$ die Indexmenge zu (2.1) und $j_1(i), j_2(i) \in \mathbb{N}$ mit $j_1(i) \le i \le j_2(i)$, für alle $i \in \mathcal{I}$. Ist die i–te Komponentenfunktion von F nur von den Variablen $u_{j_1(i)}, \ldots, u_{j_2(i)}$ abhängig, d. h.

$$f_i(t, u) = f_1(t, u_{j_1(i)}, u_{j_1(i)+1}, \ldots, u_{j_2(i)}), \quad i \in \mathcal{I},$$

so gilt

$$\frac{\partial f_i}{\partial y_j} \equiv 0, \qquad \text{für } j < j_1(i) \text{ oder } j > j_2(i),\ i \in \mathcal{I}, \tag{4.5}$$

und deshalb auch $J_{ij} = 0$.

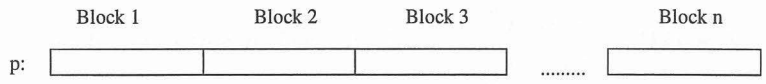
DEFINITION 4.1. Eine Abbildung $F : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}^n$ besitzt eine sparse–Struktur, falls jede Komponentenfunktion nur von $\mathcal{O}(1)$ Variablen abhängt.

$$f_i(t,u) = f_i(t, u_{j_1}, u_{j_2}, \ldots, u_{j_q}), \qquad \text{mit} \quad q << n.$$

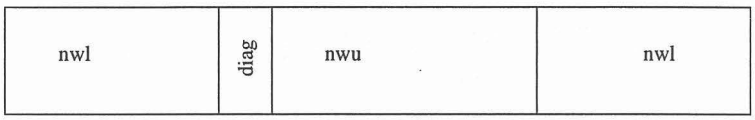
Seien

$$\begin{aligned} m_L &:= \max\{i-j \,:\, i \geq j, J_{ij} \neq 0, i,j \in \mathcal{I}\}, \\ m_U &:= \max\{j-i \,:\, j \geq i, J_{ij} \neq 0, i,j \in \mathcal{I}\}. \end{aligned} \tag{4.6}$$

Die Zahlen m_L und m_U (kurz (m_L, m_U)) charakterisieren die linksseitige bzw. rechtsseitige Bandbreite der Matrix J. Die Bandbreite von J ist dann $m = m_L + m_U + 1$.



(a) Speicher für J



(b) Aufteilung eines Blockes der Länge $2\,m_L + m_U + 1$

Abbildung 4.1: Speicherorganisation

Hindmarsh benutzt in [**16**] für EPISODE das in der Abbildung 1 skizzierte Speicherschema für die Funktionalmatrix. Die folgende Bemerkung kommentiert diese Wahl unter dem Gesichtspunkt, daß das System (2.1) das Ergebnis einer Diskretisierung einer zeitabhängigen partiellen Differentialgleichung ist und F damit eine Besetzungsstruktur aufweist, die als *sparse* bezeichnet werden kann.

BEMERKUNG 4.3. Die Matrix J wird in einem Vektor p gespeichert, der in n Blöcke (je Zeile ein Block) der Länge $2\,m_L + m_U + 1$ zerfällt: $\dim p = n\,(2\,m_L + m_U + 1)$. Jeder Block zerfällt in vier Teile $[p_1, p_2, p_3, p_4]$. Der Vektor p_1 im Block i enthält die Einträge J_{ij}, $-m_l < j - i < 0$, in p_2 ist das Diagonalelement und in p_3 sind die Elemente "rechts" von der Diagonale gespeichert. p_4 wird für eine eventuelle Pivot–Wahl benötigt. Da die Zahl $n_w := 2\,m_L + m_U + 1$ eine globale (also auf die gesamte Matrix bezogene) Größe ist, ergibt sich, daß der angeforderte Speicherplatz für ein Problem mit sparse–Struktur wesentlich größer ist, als die Anzahl der Nichtnullelemente der Matrix J. Es werden weder die Bandbreiten für eine konkrete Zeile $i \in \mathcal{I}$

$$\begin{aligned} m_L(i) &:= \max\{i-j \,:\, i \geq j, J_{ij} \neq 0, j \in \mathcal{I}\} \quad \text{und} \\ m_U(i) &:= \max\{j-i \,:\, j \geq i, J_{ij} \neq 0, j \in \mathcal{I}\}, \end{aligned} \tag{4.7}$$

noch das wirklich entstehende Besetzungsmuster berücksichtigt, sondern für alle Zeilen Speicherplatz entsprechend n_w für die maximal mögliche Bandbreite zur Verfügung gestellt.

BEMERKUNG 4.4. $i \in \mathcal{I}$. Es gilt $m_L(i) \leq m_L$, $m_U(i) \leq m_U$ und

$$\max\{m_U(i) + m_U(i) + 1 : i \in \mathcal{I}\} \leq m,$$
$$\max\{m_U(i) : i \in \mathcal{I}\} + \max\{m_L(i) : i \in \mathcal{I}\} + 1 = m.$$

Der konkrete Zugriff auf einen Eintrag der Jacobi–Matrix innerhalb der Bandbreite ist beschrieben durch die

BEMERKUNG 4.5. Sei $n_w := 2m_L + m_U + 1 \leq n$.

$$p_{(i-1)n_w+j-i+m_L+1} = J_{ij}, \qquad \text{falls} \quad -m_L \leq j - i \leq m_U, \quad i,j \in \mathcal{I}. \tag{4.8}$$

2. Vorschlag zur Reduzierung des Speicherbedarfs

Wird anstelle des direkten Verfahrens ein iterativer Gleichungslöser eingesetzt, so entfällt der *fill in* der LU–Zerlegung und es braucht auch kein entsprechender Speicherplatz vorhanden zu sein. Das folgende Vorgehen beseitigt die beschriebenen Speicherplatzprobleme, indem nur die Nichtnullelemente der Jacobi–Matrix gespeichert werden. Dabei wird das Speicherschema für Sparse–Matrizen, das schon im Zusammenhang mit dem algebraischen Mehrgitterverfahren (Liebau und Rudolphi[**20**]) verwendet wurde, benutzt.

Sei $M = (m_{ij})$ eine $n \times m$–Matrix und n_z die Anzahl der Nichtnullelemente von M. M wird durch die n_z Tripel der Form (i, j, m_{ij}) vollständig beschrieben: sei

$$T := \{(i, j, m_{ij}) : m_{ij} \neq 0\},$$

so gilt

$$M_{ij} = \begin{cases} m_{ij} & \text{falls } (i,j,m_{ij}) \in T \\ 0 & \text{sonst} \end{cases}.$$

Die Elemente von T lassen sich "zeilenweise" anordnen:

$$(i, j, m_{ij}) <_T (p, q, m_{pq}) \quad :\Leftrightarrow \quad i < p \vee (i = p \wedge j < q),$$

so daß $(T; <_T)$ eine streng geordnete Menge ist, die sich dann aufzählen läßt: es bezeichne T_k das k–te Element in $(T; <_T)$. Weiter seien $T_k(1)$, $T_k(2)$, und $T_k(3)$ die erste, zweite und dritte Komponente von T_k. Die geordneten Tripel können durch Vektoren $A_j, A \in \mathbb{R}^{n_z}$ und $A_i \in \mathbb{R}^{n+1}$ dargestellt werden, indem

$$A_k = T_k(3) \quad \text{und} \quad (A_j)_k = T_k(2), \quad k = 1, \ldots, n_z$$

gesetzt wird. Die Zeilenindizes müssen nicht in dieser Art gespeichert werden, sondern es ist ausreichend, sich die Positionen der "Zeilenwechsel" in den Vektoren A_j und A zu merken, d. h. $(A_i)_1 = 1$ und

$$(A_i)_i = k, \quad \text{falls} \quad T_{k-1}(1) = i - 1 \ \text{ und } \ T_k(1) = i, \quad i = 2, \ldots, n.$$

Aus rechentechnischen Gründen wird noch die Setzung $(A_i)_{n+1} = n_z + 1$ hinzugefügt. Ist eine Vektor–Klasse `Vektor` gegeben (diese kann z. B in *C++* aus der Standard–Template–Library abgeleitet werden), so kann eine Klasse `SparseMatrix` wie folgt aussehen:

```
class SparseMatrix
    {   long getLongIndex  (long, long)    ;
        public:
        long                n,m,NumNonZeros;
        Boolean             sym, iluIt     ;
```

```
    Vektor<long>        iA, jA, diag   ;
    Vektor<double>      A, ilu         ;
    // Methoden: ...
};
```

Der Vektor `ilu` enthält dabei die Koeffizienten der ILU–Zerlegung, in `diag` sind die Indizes der Diagonalelemente gespeichert (vorteilhaft, falls die Matrix nicht symmetrisch ist).

BEISPIEL 4.1. Sei die Matrix

$$M = \begin{pmatrix} 7 & 0 & -3 & -1 \\ 2 & 8 & 0 & 4 \\ 0 & 0 & 1 & 0 \\ -3 & 0 & 0 & 1 \end{pmatrix}$$

mit $n_z = 9$ Nichtnullelementen gegeben. $(T; <_T)$ ist dann

$$\Big((1,1,7),(1,3,-3),(1,4,-1),(2,1,2),(2,2,8),(2,4,4),(3,3,1),(4,1,-3),(4,4,1)\Big)$$

und die zugehörigen Vektoren A, A_j und A_i sind

$$\begin{aligned} A &= (\ 7,\ -3,\ -1,\ 2,\ 8,\ 4,\ 1,\ -3,\ 1) \\ A_j &= (\ 1,\ 3,\ 4,\ 1,\ 2,\ 4,\ 3,\ 1,\ 4) \\ A_i &= (\ 1,\ \ \ \ \ \ 4,\ \ \ \ \ \ 7,\ 8,\ \ \ \ \ 10). \end{aligned}$$

3. Zur Berechnung der Jacobi–Matrix

Die ganzzahlige Division werde mit $\div$ bezeichnet, d. h. für ganze Zahlen z, q gilt

$$z = (z \div q)\, q + z \bmod q.$$

LEMMA 4.1. *Seien* $j \in 1, \ldots, m$,

$$k : \{1, \ldots, m\} \to \mathbb{N} \qquad \textit{mit} \qquad k(j) = (n - j) \div m + 1$$

und

$$i : \mathbb{N} \times \mathbb{N} \to \mathbb{N} \qquad \textit{mit} \qquad i(j,p) = j + (p-1)m.$$

Sei weiter

$$I(j) := \{i(j,p) \ : 0 < p \leq k(j)\} \subset \mathbb{N}$$

eine Indexmenge. Es gelten

1. $l, k \in I(j)$ *für* $l \neq k$, *so folgt* $|l - k| \geq m$,
2. $I(j) \subset \mathcal{I}$ *für* $j \in 1, \ldots, m$,
3. $$I(j_1) \cap I(j_2) = \emptyset, \qquad \textit{für} \quad j_1 \neq j_2,$$

   und
4. $$\bigcup_{j=1}^{m} I(j) = \mathcal{I}.$$

BEWEIS. **zu 1.:** Da $l = i(j,p_1)$ und $k = i(j,p_2)$ verschieden sind, ist auch $p_1 \neq p_2$, was sofort aus der Definition von i folgt.

$$|l-k| = |(p_1-1)m - (p_2-1)m| = m|p_1-p_2| \geq m.$$

zu 2.: Nach den Definitionen von i und $I(j)$ sind alle Elemente von $I(j)$ positiv. Das maximale Element von $I(j)$ ist $i(j,k(j))$, denn $i(j,p_1) < i(j,p_2)$ für $p_1 < p_2$.

$$\begin{aligned} i(j,k(j)) &= j + (k(j)-1)m = j - m + k(j)m = j - m + ((n-j) \div m + 1)\, m \\ &= j - m + m + ((n-j) \div m)\, m \leq j + (n-j) = n. \end{aligned}$$

Damit ist $\max I(j) \leq n$ und $I(j) \subset \mathcal{I}$.

zu 3.: Annahme: sei $i(j_1,p_1) = i(j_2,p_2)$ für $j_1 \neq j_2$. O. B. d. A. sei weiter $j_1 < j_2$.

$$\begin{aligned} i(j_1,p_1) = i(j_2,p_2) \quad &\Leftrightarrow \quad j_1 + (p_1-1)m = j_2 + (p_2-1)m \\ &\Leftrightarrow \quad j_2 - j_1 = (p_2-p_1)m. \end{aligned}$$

Sind nun p_1 und p_2 verschieden, so ist die Differenz $j_2 - j_1$ ein Vielfaches von m. Dieses kann jedoch nicht sein, da $j_1, j_2 \in \{1, \ldots, m\}$ und somit $|j_2 - j_1| < m$ gilt. Also ist $p_1 = p_2$ und damit $j_1 = j_2$. Widerspruch zur Voraussetzung $j_1 \neq j_2$.

zu 4.: Nach 1. , 2. und 3. reicht es, die Elemente in $\cup_{j=1}^{m} I(j)$ zu zählen.

$$\begin{aligned} \sum_{j=1}^{m} |I(j)| &= \sum_{j=1}^{m} k(j) = \sum_{j=1}^{m} ((n-j) \div m + 1) \\ &= m + \sum_{j=1}^{m} ((n-j) \div m) = m + \sum_{j=n-m}^{n-1} j \div m. \end{aligned}$$

Das Intervall $[n-m, n-1]$ enthält m natürliche Zahlen. Es gibt genau ein $j_1 \in [n-m, n-1]$ mit $mq = j_1$, $q \in \mathbb{N}$, sonst würden mehr als ein Vielfaches von m in einem Intervall der Länge $m-1$ enthalten sein. Es folgt

$$j \div m = \begin{cases} q-1 & \text{für } j < j_1 \\ q & \text{für } j \geq j_1 \end{cases}, \qquad j \in [n-m, n-1].$$

$$\begin{aligned} m + \sum_{j=n-m}^{n-1} j \div m &= m + \sum_{j=n-m}^{j_1-1} j \div m + \sum_{j=j_1}^{n-1} j \div m \\ &= m + (j_1 - 1 - (n-m) + 1)(q-1) + (n-1-j_1+1)q \\ &= m + j_1(q-1) - (n-m)(q-1) + (n-j_1)q \\ &= m - j_1 - (n-m)(q-1) + nq \\ &= m - j_1 + n + m(q-1) = m - j_1 + n + j_1 - m = n, \end{aligned}$$

d. h. $\sum_{j=1}^{m} |I(j)| = n$.

□

LEMMA 4.2. *Sei F aus (4.4) gegeben mit*

$$\frac{\partial f_i}{\partial y_j} \equiv 0, \qquad \text{für } j < i - j_1 \text{ oder } j > i + j_2,\ i \in \mathcal{I}.$$

F besitzt demnach die Bandbreite $m := j_1 + j_2 + 1$. Die Matrix J aus der Bemerkung 4.1 ist dann mit $m+1$ Auswertungen von F berechenbar.

BEWEIS. Seien die Einheitsvektoren für $k \in \mathcal{I}$ mit e_k bezeichnet und ω_j der Vektor

$$\omega_j = y + \sum_{p=1}^{k(j)} h_{i(j,p)}\, e_{i(j,p)} = y + \sum_{l \in I(j)} h_l\, e_l.$$

ω_j ist der, in den Komponenten mit den Indizes aus $I(j)$, gestörte Vektor y. Mit der Setzung $h := \max\{h_{i(j,p)} : 0 < p \leq k(j)\}$ ergibt eine Taylorentwicklung

$$F(t,\omega_j) = F(t,y) + DF(t,y) \sum_{l \in I(j)} h_l\, e_l + \mathcal{O}(h^2) = F(t,y) + \sum_{l \in I(j)} h_l\, DF(t,y) e_l + \mathcal{O}(h^2)$$

oder

$$F(t,\omega_j) - F(t,y) = \sum_{l \in I(j)} h_l\, DF_l + \mathcal{O}(h^2).$$

Dabei bezeichnet DF_l die l–te Spalte der Matrix $DF(t,y)$. Nach Voraussetzung ist $DF_{il} = 0$ für $l < i - j_1$ oder $l > i + j_2$ und somit auch

$$DF_{il} = 0 \quad \text{für} \quad i < l - j_2 \;\text{ oder }\; i > l + j_1.$$

Die l–te Spalte besitzt maximal $m = j_1 + j_2 + 1$ Nichtnullelemente und ist von der Form

$$DF_l = (0, 0, \ldots, 0, DF_{l-j_2,l}, \ldots, DF_{l,l}, \ldots, DF_{l+j_1,l}, 0, \ldots, 0)^T.$$

Liegen nun zwei Indizes $l, k \in I(j)$ mit $l < k$ vor, dann folgt aus Lemma 4.1 $j_1 + j_2 < m \leq k - l$ und damit $l + j_1 < k - j_2$. Bildet man nun die Summe $s = DF_l + DF_k$, so "überlagern" sich ihre Nichtnullelemente nicht, da ja $l + j_1 < k - j_2$ gilt. DF_l und DF_k können aus s wieder zurückgewonnen werden. Mit dem gleichen Argument erkennt man, daß dieses auch für die Summe

$$\sum_{l \in I(j)} h_l\, DF e_l$$

gilt. Der Differenzvektor $F(t,\omega_j) - F(t,y)$ enthält demzufolge die wesentlichen Informationen zu den Spalten J_l, $l \in I(j)$, der (diskreten) Jacobi–Matrix. Mit einer Funktionsauswertung werden $|I(j)| = k(j)$ Spalten von J berechnet. Die gesamte Jacobi–Matrix ist dann durch die Differenzen

$$F(t,\omega_j) - F(t,y), \qquad j = 1, \ldots, m$$

($m + 1$ Funktionsauswertungen) gegeben. □

Der nachfolgende Algorithmus 6 ist eine Umsetzung des letzten Lemmas.

BEMERKUNG 4.6. Algorithmus 6 ist parallel durchführbar.

4. Zur Wahl von η_j

Die Wahl der Schrittweite η_j in der Differenzenapproximation (4.2) ist kritisch (vgl. auch Kelley[**19**]). Wird bei der Auswertung von F ein Fehler e verursacht, d. h. statt $F(t,y)$ erhält man $F(t,y) + e(t,y)$, so gilt:

$$\begin{aligned} J_j &= \frac{F(t, y + \eta_j e_j) + e(t, y + \eta_j e_j) - F(t,y) - e(t,y)}{\eta_j} \\ &= DF(t,y) + \mathcal{O}(\eta_j) + \frac{e(t, y + \eta_j e_j) - e(t,y)}{\eta_j}. \end{aligned}$$

Algorithmus 6 Berechnung der Jacobi–Matrix nach Lemma 4.2

```
t ∈ ℝ, y ∈ ℝ^n und F : ℝ × ℝ^n → ℝ^n gegeben;
// F erfülle die Voraussetzungen von Lemma 4.2;
ω, z, Fy ∈ ℝ^n;
Fy := F(t, y);
for j := 1 to m do
  ω := y;
  for p = 1 to k(j) do
    l := i(j, p); wähle h_l;
    ω_l := ω_l + h_l;
  end for
  z := F(t, ω);
  for p := 1 to k(j) do
    l := i(j, p);
    for i := l − j_2 to l + j_1 do
      J_il := (z_i − Fy_i) / h_l;
    end for
  end for
end for
```

Ist $\|e(t,y)\| \leq \bar{\epsilon}$, so kann der letzte Term durch $2\,\bar{\epsilon}/\eta_j$ abgeschätzt werden. Insgesamt hat man

$$J_j = DF_j(t,y) + \mathcal{O}(\eta_j + \frac{2\,\bar{\epsilon}}{\eta_j}).$$

Die Funktion

$$x + \frac{2\,\bar{\epsilon}}{x}, \qquad x > 0,$$

ist für $x_0 = \sqrt{2\,\bar{\epsilon}}$ minimal, so daß $\eta_j = \sqrt{2\,\bar{\epsilon}}$ den kleinsten Fehler erwarten läßt.

BEMERKUNG 4.7. m sei die Bandbreite der Matrix DF, $n = \dim J$, x die Lösung des linearen Systems (4.1), $x^\star$ die von

$$P^\star x := (I - \mu\, DF)x = f, \qquad \mu > 0,$$

und $\|(P^\star)^{-1}\|_2 = \|(I - \mu\, DF)^{-1}\|_2 < 1$ (vgl. (5.6)). Für die Spalten der Matrizen J und DF gelte

$$J_j = DF_j(t,y) + \mathcal{O}(\delta_j).$$

Dann ist mit hinreichend kleinem $\delta = \max \delta_j$

$$\frac{\|x - x^\star\|_2}{\|x\|_2} \leq \mathcal{O}(\mu\,\delta\,\sqrt{n\,m}). \tag{4.9}$$

BEWEIS. Nach Voraussetzung kann jede Komponente der Matrix $E := J - DF$ durch $c_{ij}\delta_j$ abgeschätzt werden: $|E_ij| \leq c_{ij}\,\delta_j$, $0 \leq c_{ij} \leq 1$.

$$\begin{aligned}\|J - DF\|_2 \leq \|J - DF\|_F &\leq \left(\sum_{i=1}^{n}\sum_{j=1}^{n} |c_{ij}\delta_j|^2\right)^{1/2} = \left(\sum_{j}\sum_{i=j-m_L}^{j+m_U} |c_{ij}\delta_j|^2\right)^{1/2} \\ &\leq \delta\,\sqrt{m\,n}\,\max c_{ij} < \delta\,\sqrt{n\,m}.\end{aligned}$$

Dann gilt

$$\|E\|_2 = \|(I - \mu DF) - (I - \mu J)\|_2 = \mu \|J - DF\|_2 \leq \mu\, \delta \sqrt{n\, m} < 1,$$

falls die Zahlen δ_j klein genug gewählt werden und es folgt

$$\|(I - \mu\, DF)^{-1} E\|_2 \leq \|(P^\star)^{-1}\| \cdot \|E\|_2 \leq 1.$$

Damit sind die Vorausetzungen des Störungsatzes 1.8.10 in Bunse[4] erfüllt und es gilt

$$\frac{\|x - x^\star\|_2}{\|x\|_2} \leq \frac{\|(P^\star)^{-1} E\|_2}{1 - \|(P^\star)^{-1} E\|_2} \leq \frac{\|E\|_2}{1 - \|E\|_2} \leq C\, \mu\, \delta \sqrt{n\, m} + \mathcal{O}(\delta^2\, \mu^2\, n\, m).$$

□

5. Spezialfall: F affin

BEMERKUNG 4.8. Sei die Abbildung F aus (2.1) affin in y, dann gilt (in exakter Arithmetik) $J = DF(t, y)$.

BEWEIS. Nach Voraussetzung hat F die Gestalt

$$F(t, y) = \mathcal{F}(t)\, y + q(t), \qquad \text{mit einer der Matrix} \quad \mathcal{F}(t) \in \mathbb{R}^{n \times n},$$

und $DF = \mathcal{F}$ folgt sofort. Für jeden Vektor $\omega = y + \eta\, e_j$ hat man

$$\begin{aligned} F(t, \omega) - F(t, y) &= \mathcal{F}(t)\, \omega + q(t) - (\mathcal{F}(t)\, y + q(t)) \\ &= \mathcal{F}(t)\, \eta\, e_j = \eta\, \mathcal{F}_j(t) = \eta\, DF_j. \end{aligned}$$

Demnach liefert der Differenzenquotient (4.2) die exakte j–te Spalte von DF. □

BEMERKUNG 4.9. Sind zusätzlich die Koeffizienten von $\mathcal{F}(t)$ nicht vom Parameter t abhängig, so ändert sich die Jacobi–Matrix während der Rechnung nicht. Sie braucht deshalb auch nur einmal bestimmt zu werden.

In dieser Situatuation, d. h. F hat die Form $\mathbf{y}' = \mathcal{F}\, y + q$, besitzt (2.1) die Lösung

$$\mathbf{y}(t) = e^{(t - t_0)\, \mathcal{F}} y(t_0) + \int_{t_0}^{t} e^{(t - \tau)\, \mathcal{F}} q(t)\, d\tau, \qquad t > t_0.$$

Diese Formel ist aber numerisch nicht brauchbar, da die Matrixexponentialfunktion $e^A = \sum_k A^k / k!$ nur sehr aufwendig zu berechnen ist (es sei denn, es exisitiert ein schneller Algorithmus, der A^k hinreichend genau bestimmen kann).

KAPITEL 5

Gleichungslöser

1. Zum CG–Verfahren

Zur iterativen Lösung eines linearen Gleichungssystems

$$Ax = b, \qquad A \in \mathbb{C}^{n\times n}, \quad x, b \in \mathbb{C}^n \tag{5.1}$$

mit hermitescher (im reellen Fall symmetrischer) positiv definiter Matrix wird häufig das CG–Verfahren eingesetzt. Es minimiert ein zu (5.1) äquivalentes quadratisches Funktional sukzessive bzgl. bestimmter Richtungen, den sogenannten konjugierten Gradienten (**c**onjugate **g**radient). Das CG–Verfahren zeichnet sich dadurch aus, daß es nach maximal n Schritten (bei rundungsfehlerfreier Arithmetik) sogar die exakte Lösung liefern würde. Jedoch ist im allgemeinen die Iterierte nach $m << n$ Schritten bereits eine hinreichend gute Näherung der exakten Lösung, wodurch das Verfahren einen iterativen Charakter bekommt.

BEMERKUNG 5.1. Sei A positiv definit mit der Konditionszahl $\kappa(A) = \lambda_{\max}/\lambda_{\min}$. Für die Iterierten x^m des CG–Verfahrens gilt

$$\|x^m - x\|_A \le \frac{2\,c^m}{1 + c^{2m}}\,\|x^0 - x\|_A, \qquad c = \frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1} = \frac{\sqrt{\lambda_{\max}}-\sqrt{\lambda_{\min}}}{\sqrt{\lambda_{\max}}+\sqrt{\lambda_{\min}}}, \tag{5.2}$$

wobei x^0 der Startvektor und $\|x\|_A := \sqrt{\langle A\,x, x\rangle}$ die Energienorm sind. Da $c \approx 1 - 1/\sqrt{\kappa}$, ist die Kondition der Matrix A die für die Konvergenz des CG–Verfahrens entscheidende Größe.

Abb. 5.1 zeigt die Anzahl der Iteration in Abhängigkeit von κ, die nach (5.2) nötig sind, um den Anfangsfehler um 10^{-6} zu reduzieren.

Algorithmus 7 CG–Algorithmus [**13**]

procedure CG--Verfahren (var A: SparseMatrix; var x, b:Vektor)
$r := b - A\,x$; {Residuum bilden}
$p := r$; {erste Richtung wählen}
$\rho := \langle p, r\rangle$;
while $\|r\| > \max\{\mathrm{tol}\,\|b\|, 1e-16\}$ **and** $(k < k_{\max})$ **do**
  $k++$;
  $v := A\,p$; $\lambda := \rho/\langle v, r\rangle$;
  $x := x + \lambda\,p$; {neue Näherung}
  $r := r - \lambda\,v$; $\rho_{\mathrm{old}} = \rho$; $\rho = \langle p, r\rangle$; {neues Residuum}
  $p := r + \rho/\rho_{\mathrm{old}}\,p$;
end while

Ausführliche Beschreibungen des CG–Verfahrens, insbesondere auch Beschleunigungen durch Präkonditionierung, finden sich z. B. in Hackbusch [**13**] oder Braess [**2**]. Für die Praxis vorteilhaft ist die einfache Gestalt von Algorithmus 7, die wiederum eine einfache (schnelle) Programmierung zur Folge hat; dieses kann der entscheidende Vorteil eines Verfahrens sein.

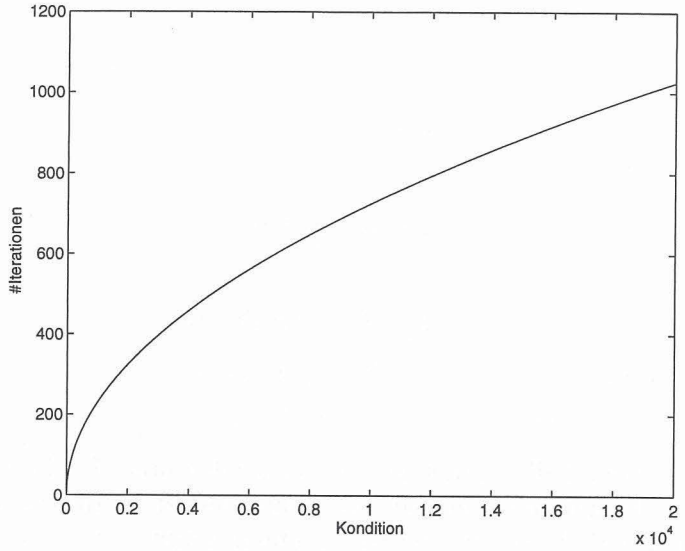


Abbildung 5.1: Anzahl der Iteration in Abhängigkeit von κ nach (5.2)

Weiterhin braucht für einen Vektor x nur die Vorschrift $A : x \to A\,x$ bekannt sein, nicht die Abbildung A selbst in Form einer Matrix.

Dem CG–Verfahren "verwandte" Methoden sind die Algorithmen Orthomin, Orthdir, Minres und das Lanczos–Verfahren (vgl. hierzu auch Greenbaum [**9**], Krylov–Raum–Approximationen).

Eine Verallgemeinerung des CG–Verfahrens für Systeme mit nichtsymmetrischer Matrix ist das BiCG–Verfahren bzw. die numerisch stabilere Variante Bi–CGSTAB, Algorithmus 8. Auf eine genauere Darstellung des Bi–CGSTAB–Verfahrens sei an dieser Stelle verzichtet und auf van der Vorst [**34**] verwiesen. Bemerkenswert ist, daß dieser Algorithmus ohne Kenntnis der transponierten Matrix auskommt, d. h. eine Matrix–Vektor–Operation der Form $A^H\,x$ ist nicht notwendig (die Multiplikation mit der transponierten Matrix kann durchaus zu Problemen im Zusammenhang mit der gewählten Struktur für die Sparse–Matrizen führen). Pro Iterationsschritt sind zwei Matrix–Vektor–Multiplikationen, zwei Aufrufe des Präkonditionierers und vier Skalarprodukte zu berechnen. Neben der rechten Seite b und den Vektoren x und $r = b - A\,x$ sind noch sechs weitere Hilfsvektoren zur Durchführung der Iteration nötig (Speicherplatz: $9\,n$; Algorithmus 8 ist hier nicht ganz optimal aufgeschrieben). Eine Alternative zu Algorithmus 8 ist das GMRES–Verfahren, welches in der nichtlinearen Variante gleich auf das Problem (2.8) angewandt werden könnte (Newton–GMRES), vgl. auch Bemerkung 5.3.

Die Iteration im Algorithmus 8 wird beendet, falls die Norm des Residuums relativ zur rechten Seite kleiner als eine Toleranz ist

$$\|r^{(m)}\|_h := \|A\,x^{(m)} - b\|_h \leq \eta\,\|b\|_h \tag{5.3}$$

oder die maximal zulässige Anzahl von Iterationen überschritten wird. Sei x die Lösung von (5.1), so gilt

$$\|x^{(m)} - x\|_h = \|A^{-1}\,A(x^{(m)} - x)\|_h \leq \|A^{-1}\|_2\,\|r^{(m)}\|_h.$$

Algorithmus 8 BI–CGSTAB–Algorithmus [34]

procedure BI–CGSTAB--`Verfahren (var` A: `SparseMatrix; var` x, b:`Vektor)`
$r := b - A\,x$; {`Residuum bilden`}
$p := r$; {`erste Richtung wählen`}
$r_0 := \frac{r}{\|r\|}$; {`wähle` r_0 `mit` $\langle r_0, r\rangle \neq 0$}
$\alpha = 1$;
while $\|r\| > \max\{\mathrm{tol}\,\|b\|, 1e-16\}$ **and** $(k < k_{\max})$ **do**
  $k++$;
  $\bar{p} = K^{-1}\,p$; {`Präkonditionierer:` K}
  $v := A\,\bar{p}$;
  $\lambda := \frac{\alpha}{\langle v, r_0\rangle}$;
  $s := r - \lambda v$;
  $\bar{s} = K^{-1}\,s$;
  $t := A\,\bar{s}$; $\bar{t} := K^{-1}\,t$;
  $\omega := \langle \bar{t}, \bar{s}\rangle / \langle \bar{t}, \bar{t}\rangle$ {`hier ist auch` $\omega := \frac{\langle t,s\rangle}{\langle t,t\rangle}$ `möglich`}
  $x := x + \lambda\bar{p} + \omega\,\bar{s}$;
  $r := s - \omega\,t$; $\alpha_{\mathrm{old}} = \alpha$; $\alpha = \langle r, r_0\rangle$ {`neues Residuum`}
  $p := r + \frac{\lambda\,\alpha}{\omega\,\alpha_{\mathrm{old}}}\,(p - \omega v)$;
end while

Mit (5.3) und $A\,x = b$ folgt $\|x^{(m)} - x\|_h \leq \eta \|A^{-1}\|_2 \|A\|_2 \|x\|_h$. Der relative Fehler kann demnach durch

$$\frac{\|x^{(m)} - x\|_h}{\|x\|_h} \leq \kappa_2(A)\,\eta \tag{5.4}$$

abgeschätzt werden.

2. Numerik

Sei h die typische Schrittweite für die Diskretisierung in Ortsrichtung. Die Gleichung (P) lautet in der Form (2.1)

$$u_h'(t) = M_h^{-1}\,A_h(u_h)u_h(t) + M_h^{-1}\,f_h(t), \tag{5.5}$$

d. h. in diesem Fall ist

$$F(t, u_h) = M_h^{-1}\,A_h(u_h)u_h(t) + M_h^{-1}\,f_h(t).$$

Nehmen wir als Testproblem an, daß die Steifigkeitsmatrix nicht von u_h abhängt und symmetrisch ist (lineares instationäres Problem), so ist $DF(t, u_h) = M_h^{-1}\,A_h\,u_h$ und das Gleichungssystem (2.11) hat die Form

$$\left(I_h - \beta\,\tau M_h^{-1}\,A_h\right)\delta = F(t, u_h), \tag{5.6}$$

wobei im Fall der k–ten BDF–Formel $\beta = 1/\alpha_k$ ist (Tabelle 3.1).

BEMERKUNG 5.2. Die Matrizen M_h und $-A_h$ seien positiv definit: $M_h > 0$ und $-A_h > 0$. Erfüllen ihre Eigenwerte (2–D–Fall einer Diskretisierung der instationären Wärmeleitungsgleichung)

$$c_1\,h^2 \leq \lambda(M_h) \leq c_2\,h^2, \qquad c_3\,h^2 \leq |\lambda(A_h)| \leq c_4 \quad \text{mit} \quad \lambda(A_h) < 0,$$

dann gilt für die Eigenwerte der Matrix $I_h - \beta\tau M_h^{-1} A_h$

$$1 < \lambda(I_h - \beta\tau M_h^{-1} A_h) \leq 1 + \mathcal{O}(\tau h^{-2}).$$

Die Kondition der Matrix im Gleichungssystem (5.6) ist somit

$$\kappa_2(I_h - \beta\tau M_h^{-1} A_h) = \mathcal{O}(\tau h^{-2}). \tag{5.7}$$

Diese Kondionsabschätzung gilt auch für eine 3–D Diskretisierung der instationären Wärmeleitungsgleichung.

BEWEIS. Sei $\theta = \beta\tau > 0$.

$$\begin{aligned}\lambda(I_h - \theta M_h^{-1} A_h) &= \lambda\left(M_h^{-1/2}(I_h - \theta M_h^{-1/2} A_h M_h^{-1/2})M_h^{1/2}\right) \\ &= \lambda\left(I_h - \theta M_h^{-1/2} A_h M_h^{-1/2}\right) = 1 - \theta\lambda\left(M_h^{-1/2} A_h M_h^{-1/2}\right).\end{aligned}$$

Mit $A_h < 0$ ist auch die Matrix $M_h^{-1/2} A_h M_h^{-1/2}$ negativ definit: $\lambda(I_h - \theta M_h^{-1} A_h) > 1$. Aus den Abschätzungen zu den Eigenwerten und den Rechenregeln für positiv definite Matrizen folgt

$$-A_h \leq c_4 I = \frac{c_4}{c_1 h^2} c_1 h^2 I \leq \frac{c_4}{c_1 h^2} M_h$$

und weiter

$$-M_h^{-1/2} A_h M_h^{-1/2} \leq \frac{c_4}{c_1 h^2} I_h$$

bzw. $-\lambda\left(M_h^{-1/2} A_h M_h^{-1/2}\right) < c_4/c_1 h^{-2}$. Die Matrix $X := I_h - \theta M_h^{-1} A_h$ ist nicht symmetrisch, aber ähnlich zu der symmetrischen Matrix $Y = I_h - \theta M_h^{-1/2} A_h M_h^{-1/2}$. Für die Kondition $\kappa_2(X)$ gilt dann

$$\kappa_2(X) = \kappa(M_h^{-1/2} Y M_h^{1/2}) = \|M_h^{-1/2} Y M_h^{1/2}\|_2 \, \|M_h^{-1/2} Y^{-1} M_h^{1/2}\|_2.$$

Da

$$\|M_h^{-1/2} Y M_h^{1/2}\|_2^2 = \rho(M_h^{-1/2} Y Y^H M_h^{1/2}) = \rho(Y Y^H) = \rho(Y)^2,$$

folgt

$$\kappa_2(X) = \rho(Y)\,\rho(Y^{-1})$$

und insgesamt die Behauptung

$$1 < \lambda(Y) = \lambda(I_h - \theta M_h^{-1/2} A_h M_h^{-1/2}) < 1 + \theta \frac{c_4}{c_1 h^2} = 1 + \mathcal{O}(\theta h^{-2}). \tag{5.8}$$

Im 3–D–Fall ist (bis auf evtl. Skalierung; vgl. Hackbusch [**12**], Braess [**2**])

$$c_1 h^3 \leq \lambda(M_h) \leq c_2 h^3, \qquad c_3 h^3 \leq |\lambda(A_h)| \leq c_4 h \quad \text{mit} \quad \lambda(A_h) < 0.$$

Es bleibt also $\kappa(M_h) = \mathcal{O}(1)$ und $\kappa(A_h) = \mathcal{O}(h^{-2})$. Wie eben folgt

$$\widehat{c}_3 < -\lambda\left(M_h^{-1/2} A_h M_h^{-1/2}\right) < \widehat{c}_4 h^{-2}.$$

□

Der Algorithmus 8 wurde an einigen Beispielen getestet, um das Konvergenzverhalten zu untersuchen. In der Tabelle 5.1 sind die Ergebnisse für eine Diskretisierung mittels Box–Methode des (elliptischen) stationären Modellproblems

$$\begin{aligned}-\Delta u &= 4 \qquad \text{in } \Omega = [0,1]^2 \\ u|_{\partial\Omega} &= x^2 + y^2\end{aligned}$$

für verschiedene Feinheitsstufen des Gitters dargestellt. Um die Zeitschrittweite zu simulieren, wurde danach die Differentialgleichung

$$
\begin{aligned}
u - \epsilon\, \Delta\, u &= 4 \qquad \text{in } \Omega = [0,1]^2, \quad \epsilon = \frac{\tau}{h^2}, \\
u|_{\partial\Omega} &= x^2 + y^2
\end{aligned}
\tag{5.9}
$$

für die Zeitschrittweiten $\tau = 10.0, 0.01, 0.0001$ betrachtet. Die Ergebnisse sind in den Tabellen 5.2 - 5.4 dargestellt. Nimmt man noch an, daß der Algorithmus 8 sich qualitativ wie das CG–Verfahren verhält, also die Kondition des Gleichungssystems die für die Konvergenz entscheidende Größe ist, so ist aus den Abschätzungen (5.2) und (5.7) zu erwarten, daß für kleine Verhältnisse von τ/h^2 (kleine Zeitschrittweiten) das Verfahren schnell, mit wachsenden τ/h^2 langsamer konvergiert. Die Tabellen 5.2 - 5.4 bestätigen genau dieses Verhalten. Außerdem wirkt sich eine kleine Zeitschrittweite günstig auf die relativen Fehler (4.9) und (5.4) aus. Für die wärmetechnische Simulation eines Hochlochziegels (vgl. Kapitel 6) zeigen die Abbildungen 5.2 und 5.3, daß Theorie und Praxis hier sehr schön zusammenpassen. Abbildung 5.2 zeigt den Vergleich zwischen der Zeitschrittweite τ und der Anzahl der Iterationen: Mit wachsenden τ nimmt der Aufwand für das Bi–CGSTAB–Verfahren zu (erste Flanke/erstes Maximum). In Abbildung 5.3 ist ein Vergleich zwischen β (d. h. der gewählten Methode) und der Anzahl der Iterationen dargestellt: Der Wechsel in der Methode (von $k = 2$ auf $k = 1$) bewirkt, daß der Aufwand für das Bi–CGSTAB–Verfahren kurzfristig zunimmt (zweites Maximum).

Damit sollte die Zeitschrittweite im Hinblick auf die Genauigkeit der Ergebnisse und die Konvergenzgeschwindigkeit des BiCG–Verfahrens nicht zu groß gewählt werden. Für die instationäre Wärmeleitungsgleichung geht in die Massematrix M_h das Produkt $\rho\, c$ aus Dichte und Kapazität ein, während die Wärmeleitfähigkeit der Parameter zur Steifigkeitsmatrix ist. Homogenes Material vorausgesetzt, ist $\rho\, c$ eine Schätzung für c_1 in Bemerkung 5.7, für c_4 kann z. B. $\|A_h\|_\infty$ genommen werden. Aus (5.8) folgt somit

$$
\kappa(I_h - \beta\, \tau M_h^{-1}\, A_h) \leq 1 + |\beta|\, \tau \frac{\|A_h\|_\infty}{\rho\, c\, h^2}.
$$

Oft wird die Massematrix durch eine geeignete Diagonalmatrix ersetzt (lumping the mass matrix), so daß $c_1\, h^2 = \lambda_{\min}(M_h) = \min_i (M_h)_{ii}$. Dann ist

$$
\kappa(I_h - \beta\, \tau M_h^{-1}\, A_h) \leq 1 + |\beta|\, \tau \|M_h^{-1}\, A_h\|_\infty \leq 1 + |\beta|\, \tau \frac{\|A_h\|_\infty}{\min_i (M_h)_{ii}}.
$$

Die Forderung, die Kondition zu beschränken, also etwa $\kappa(I_h - \beta\, \tau M_h^{-1}\, A_h) < C$, ergibt eine Bedingung an die Zeitschrittweite:

$$
\tau < \frac{(C-1)\rho\, c\, h^2}{|\beta|\, \|A_h\|_\infty} \quad \text{bzw.} \quad \tau < \frac{(C-1)}{|\beta|\, \|M_h^{-1}\, A_h\|_\infty}.
$$

Es liegt ein Zielkonflikt vor: Damit der Endzeitpunkt schnell erreicht wird, ist man an möglichst großen Zeitschrittweiten interessiert, diese verursachen aber ein langsame Konvergenz des BiCG–Verfahrens und, was schwerwiegender ist, ungenauere Ergebnisse (ein stabiler Zustand wird evtl. "zu spät" erreicht). Kleine Zeitschrittweiten hingegen garantieren eine größere Genauigkeit und schnelle Konvergenz des BiCG–Verfahren, aber es werden eben viele Zeitschritte benötigt.

Wird zur Lösung von (2.11) ein direktes Verfahren benutzt, so liegt bzgl. der Genauigkeit grundsätzlich keine andere Situation vor, denn auch hier geht die Kondition des Gleichungssystems als die entscheidende Größe in die Störungssätze ein ((5.4), auch Bunse [4]).

Dimension A	Konvergenzrate (geom. Mittel)	Defekt	# Schritte
81	0.10814	9.61496e-14	7
289	0.41257	3.70854e-13	16
1089	0.64772	2.62132e-13	33
4225	0.77761	6.28988e-13	56

Tabelle 5.1: Beispielrechnung für $\Delta\, u = 4$

Dimension A	Konvergenzrate (geom. Mittel)	Defekt	# Schritte
81	0.10820	9.40271654e-14	7
289	0.41008	3.07190835e-13	16
1089	0.64588	2.18146518e-13	33
4225	0.77653	1.96099578e-13	58
16641	0.86617	3.29315477e-13	102

Tabelle 5.2: Beispielrechnung für $u - \frac{10}{h^2}\Delta\, u = 4$

Dimension A	Konvergenzrate (geom. Mittel)	Defekt	# Schritte
81	0.13896	2.82349230e-13	7
289	0.09264	9.03881923e-14	6
1089	0.21875	8.04183647e-14	10
4225	0.49587	2.79669400e-13	21
16641	0.72668	7.48475169e-13	45

Tabelle 5.3: Beispielrechnung für $u - \frac{0.01}{h^2}\Delta\, u = 4$

Dimension A	Konvergenzrate (geom. Mittel)	Defekt	# Schritte
81	0.28561	2.38147762e-13	11
289	0.33774	1.58455403e-13	13
1089	0.36207	1.79943398e-13	14
4225	0.27021	2.31133041e-13	11
16641	0.11122	7.22919503e-14	7

Tabelle 5.4: Beispielrechnung für $u - \frac{0.0001}{h^2}\Delta\, u = 4$

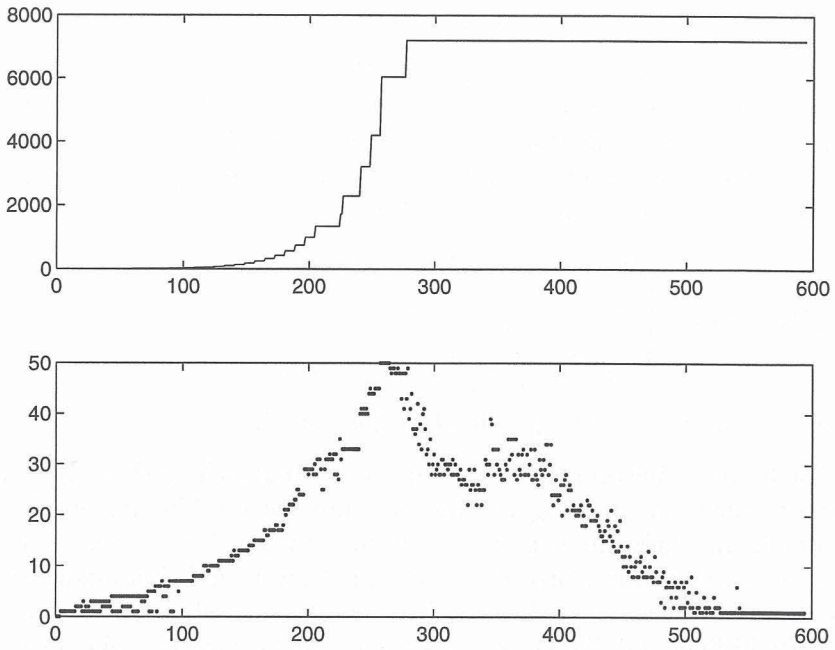


Abbildung 5.2: Vergleich: τ–Anzahl der BI–CGSTAB–Schritte.

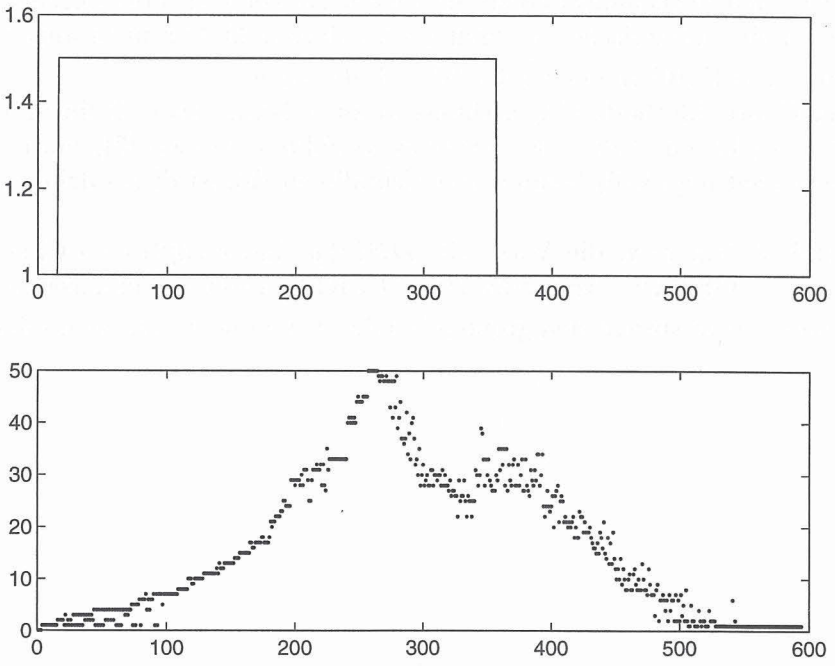


Abbildung 5.3: Vergleich: $-\beta$–Anzahl der BI–CGSTAB–Schritte.

3. Ausblick

BEMERKUNG 5.3. Es kann auf eine explizite Berechnung der Jacobi–Matrix verzichtet werden, falls das lineare Gleichungssystem (2.11) mit einem iterativen Verfahren gelöst wird, das nur Matrix–Vektor–Multiplikationen benötigt (CG–Methoden oder ein GMRES–Verfahren; vgl. Kelley [**19**]). Der Vektor $D\,H(z)\,y$ kann als Richtungsableitung $D\,H(z:y)$ von H angesehen werden. Eine Approximation hierfür ist (Kelley [**19**])

$$D_\mu H(z:y) := \begin{cases} 0, & y = 0, \\ \|y\| \frac{H(z+\mu y^1)-H(z)}{\mu \|z\|}, & z, y \neq 0, \\ \|y\| \frac{H(\mu y^1)-H(z)}{\mu}, & z = 0,\ y \neq 0 \end{cases}, \qquad \text{mit } y^1 := y/\|y\|. \tag{5.10}$$

Ist H affin in y, so stimmt $D_\mu H(z:y)$ mit $DH(z)\,y$ überein.

Besondere Bedeutung bekommt diese Bemerkung dadurch, daß es bei einem Einsatz einer CG–Variante nicht mehr nötig ist, die gesamte Jacobi–Matrix, wie im Abschnitt 3 beschrieben, mit einem Aufwand von $m+1$ Auswertungen der rechten Seite F von (2.1) zu berechnen, sondern es genügt, pro Iterationsschritt zwei Funktionsauswertungen[1] durchzuführen (Jacobian–free method). Ähnliche Konvergenzeigenschaften vorausgesetzt, ist der Einsatz von (5.10) billiger als die Verwendung der Jacobi–Matrix, solange die Anzahl der CG–Iterationen $(m+1)/2$ nicht übersteigt, welches insbesondere in der Startphase der Fall ist, da hier nur wenige Iterationen aufgrund der guten Kondition ausgeführt werden. Dieses Vorgehen besitzt allerdings den Nachteil, daß ein "globaler" Präkonditionierer (z. B. ILU–Iteration) nun nicht mehr eingesetzt werden kann, da die Matrix $DH(x)$ nicht bekannt ist.

Ist es das Ziel, den Rechenaufwand zu reduzieren, so bieten sich die folgenden Vorgehensweisen an, die auf eine numerische Bestimmung der Jacobi–Matrix nicht angewiesen sind:

- Benutzung von (5.10) in der BI–CGSTAB–Iteration,
- statt der Chord–Methode (Algorithmus 3) zur Lösung des nichtlinearen Gleichungssystems (2.8) ist ein NEWTON–GMRES–Verfahren (Brown [**3**]) einsetzbar, das die Richtungsableitung (5.10) benutzt (im nichtaffinen Fall ist dieses dringend zu empfehlen),
- die Jacobi–Matrix bzw. die Vorschrift $DH(z)\,y$ sind analytisch zu beschreiben und dann im BI–CGSTAB– bzw. NEWTON–GMRES–Verfahren einzusetzen.

Die letzte Alternative verspricht den größten Erfolg, insbesondere im nichtaffinen Fall.

[1] Falls (2.11) aus der Chord–Methode stammt und das BICG–Verfahren zur Lösung benutzt wird.

KAPITEL 6

Anwendungsspektrum und –beispiele

Das Anwendungsspektrum des Programms INSTATCP ist sehr weitgefächert. Es reicht von der Berechnung brandbeanspruchter Bauteile mit Berücksichtigung der Verdampfungsvorgänge über fast alle Arten von 3D–Wärmebrückenproblemen, die Simulation von Sonneneinstrahlungsproblemen und Gefrier– und Auftauvorgängen bis zur Erfassung strömender Medien, die nicht nur in Rohrleitungen, Bodenheizungen, erdverlegten oder oberirdischen Wasser– und Fernwärmeleitungen, brandbeanspruchten Lüftungsleitungen, Rauchgasabführungen, Haus– und Industrieschornsteinen, sondern auch bei technischen Wärmetauschern und geothermischen Anwendungen auftreten (Rudolphi [**26**], Rudolphi [**27**], Rudolphi, Kownatzki [**28**]). Da bei der Methode der finiten Volumen auch Dreieckselemente mit Winkeln $\alpha > 90°$ benutzt werden können, vgl. Liebau, Rudolphi [**20**], ist neben einer wesentlich flexibleren Elementierung komplizierter Geometrien (Schrägen, kreisförmige Bereiche) die Möglichkeit für eine automatische Triangulierung, etwa mit TRIANGLE , geschaffen. Dieses Programm von Richard Shewchuk [**31**] liefert ausgehend von 2D–Polygonzügen eine Delaunay–Triangulierung der beschriebenen Gebiete. Die Feinheit der Triangulierung kann dabei durch Vorgabe von Minimalwinkeln oder der maximal zulässigen Dreiecksfläche gesteuert werden. Zur Darstellung der Möglichkeiten des Programms sollen hier eine bekleidete HEM300–Stahlstütze mit Simulation der Verdampfung unter Normbrandbedingungen (Abb. 6.1 bis 6.8) und ein Hochlochziegel als Anwendungsbeispiele dienen (Abb. 6.9 bis 6.11).

1. Bekleidete Stahlstütze

Eine rechnerische Simulation kann generell nur erfolgen, wenn die thermischen Stoffeigenschaften wie Wärmeleitfähigkeit, spezifische Wärmekapazität und Rohdichte aller zum Bauteil gehörenden Baustoffe und die Wärmeübergangsverhältnisse im Brandraum und im Bauteil bekannt sind. Für die Berechnung diene als Beispiel eine im Stützenprüfofen der Bundesanstalt für Materialforschung und -prüfung (BAM) durchgeführte Brandprüfung an einer HEM200–Stahlstütze mit 15 mm dicker plattenförmiger Bekleidung aus Calcium–Silikat–Platten (Promatect–H). Die Beflammung erfolgte hierbei nach der Einheitstemperaturzeitkurve (DIN 4102 [**5**], ISO 834 [**18**]) bei einer Anfangstemperatur von $\vartheta_0 = 20$ °C,:

$$\vartheta = \vartheta_0 + 345 \ \log_{10}(8\,t + 1) \qquad \text{mit } \vartheta, \vartheta_0 \text{ in °C und } t \text{ in min.}$$

Die thermischen Stoffeigenschaften von Baustahl wurden wie folgt zugrunde gelegt:

- die Wärmeleitfähigkeit als Funktion der Stahltemperatur nach Abb. 6.1,
- die spezifische Wärmekapazität als Funktion der Stahltemperatur nach Abb. 6.2,
- die Rohdichte konstant mit $\rho = 7850$ kg/m^3.

Die thermischen Stoffeigenschaften und die Rohdichte der Calcium–Silikat–Platten (Promatect–H) wurden nach Angaben des Herstellers wie folgt angesetzt:

- die Wärmeleitfähigkeit als Funktion der Temperatur nach Abb. 6.3,
- die spez. Wärmekapazität als Funktion der Temperatur nach Abb. 6.4,
- die Rohdichte konstant mit ρ =870 kg/m^3.

Die den Wärmeübergang durch Konvektion und Strahlung kennzeichnenden Größen werden für den Stützenprüfofen der BAM entsprechend der Beziehung

$$\frac{\dot{Q}_{\mathrm{FS}}}{A} = \alpha_{\mathrm{K}} \left(\vartheta_{\mathrm{F}} - \vartheta_{\mathrm{S}}\right) + C_{\mathrm{S}}\, \varphi\, \epsilon_{\mathrm{F}}\, \epsilon_{\mathrm{S}} \left(\left(\frac{\vartheta_{\mathrm{F}} + 273}{100} \right)^4 - \left(\frac{\vartheta_{\mathrm{S}} + 273}{100} \right)^4 \right)$$

bzw.

$$\frac{\dot{Q}_{\mathrm{FS}}}{A} = \alpha_{\mathrm{K}} \left(\vartheta_{\mathrm{F}} - \vartheta_{\mathrm{S}}\right) + C_{\mathrm{S}}\, \varphi\, \epsilon_{\mathrm{F}} \left(\left(\frac{\vartheta_{\mathrm{F}} + 273}{100} \right)^4 - A_{\mathrm{S}} \left(\frac{\vartheta_{\mathrm{S}} + 273}{100} \right)^4 \right)$$

für den Strahlungsautausch zwischen dem Brandraumgas (F ... furnace) und der Prüfkörperoberfläche (S ... surface) nach einer Optimierungsrechnung zur Bestimmung der Wärmeübergangsverhältnisse Brandraum–Stütze Müller [**21**], Müller et al. [**24**], Paasch et al. [**25**] wie folgt angenommen:

- konvektiver Wärmeübergangskoeffizient $\alpha_{\mathrm{K}} = 31.6\ \mathrm{W/(m^2\,K)}$ (Konvektion),
- Emissionskoeffizient des Gases im Brandraum $\epsilon_{\mathrm{F}} = 0.516$ (Strahlung),
- Emissionskoeffizient der Prüfkörperoberfläche $\epsilon_{\mathrm{S}} = 0.95$ (Strahlung), also als Produkt der Emissionskoeffizienten $\epsilon_{\mathrm{F}}\, \epsilon_{\mathrm{S}} = 0.49$ (Strahlung),
- Konfigurationsfaktor $\varphi = 1.0$ (Strahlung).

Der Konfigurationsfaktor φ ist eine rein geometrische Konstante und ergibt sich aus Flächen, Abmessungen, Abstand und gegenseitiger Anordnung der im Strahlungsaustausch stehenden Körper (VDI–Wärmeatlas [**17**], Siegel [**32**]), hier des strahlenden Gaskörpers der Rauch– und Verbrennungsgase, der strahlenden Innenoberflächen des Stützenprüfofens und der Außenoberfläche der Bekleidung. Da an der Flanschaußenseite keine Abschattungseffekte auftreten, wurde dieser Konfigurationsfaktor mit $\varphi_{\text{Bekleidung außen}} = 1$ angesetzt. Im Luftraum zwischen der Flanschinnenseite, der Steginnenseite und der Innenseite der Stützenbekleidung wird eine nach Müller, Rudolphi [**22**], Müller, Rudolphi [**23**] ermittelte temperaturabhängige äquivalente Wärmeleitfähigkeitsfunktion angesetzt, Abb. 6.5. Abb. 6.6 sind die bei der Rechnung angesetzten Werte für die spezifische Wärmekapazität und die Dichte der Luft als Funktion der Temperatur zu entnehmen. Aus Symmetriegründen ist die Simulation eines Viertelausschnitts (bezogen auf die x–y–Frontebene) nach Abb. 6.7 ausreichend. Bei der automatischen Triangulierung der Frontebene mit TRIANGLE entstanden dort 285 Knoten und somit bei 2 Knotenebenen in der Tiefe (= 1 Tiefenschicht) insgesamt 570 Knoten mit 499 Wärmeleitungssechseckelementen und 32 Wärmeübergangsrechteckelementen. Die Gesamtzahl der Elemente ergab sich damit zu 531 Elementen. Die Breite des Teilausschnitts betrug 0.118 m ($b/2 + d_{\mathrm{Bekl}} = 103\ \mathrm{mm} + 15\ \mathrm{mm}$). Er reichte von der halben Stegdicke bis zur Außenseite der Bekleidung. Die Tiefe des Teilausschnitts betrug 0.1 m. Als Höhe des Teilausschnitts wurden (halbe Steghöhe bis zur Außenseite der Bekleidung, $h/2 + d_{\mathrm{Bekl}} = 110\ \mathrm{mm} + 15\ \mathrm{mm}$) 0.125 m zugrunde gelegt. Den für eine Brandprüfung im Stützenprüfofen der BAM an einer mit 15 mm dicken Calcium–Silikat–Platten bekleideten HEM200–Stahlstütze im Knoten 5 unter Berücksichtigung der Plattenfeuchtigkeit $u_{\mathrm{m}} = 3.5\ \%$ errechneten Temperatur–Zeit–Verlauf im Vergleich zu den gemessenen Werten (Knoten 285) zeigt Abb. 6.8. Die CPU–Rechenzeit auf einem PC mit 450–MHz–Pentium–Prozessor und 256 MB Hauptspeicher unter Borland Delphi 4 und Windows 98 betrug für eine Integrationszeitdauer von 1.5 Stunden unter Berücksichtigung der Plattenfeuchtigkeit 10.4 min, ohne ihre Berücksichtigung 156 s.

2. Hochlochziegel

Das zweite Anwendungsbeispiel zeigt eine dreidimensionale Berechnung des Wärmedurchgangskoeffizienten k für einen Hochlochziegel unter Einsatz von AutoCAD und TRIANGLE. Durch die automatische Triangulierung der Frontebene mit TRIANGLE entstanden dort 5798 Knoten und somit bei 20 Knotenebenen in der Tiefe (= 19 Tiefenschichten) insgesamt 115960 Knoten mit 214852 prismatischen Elementen, Abb. 6.9. Die Gesamtzahl der Elemente ergab sich damit zu 216144 Elementen. Bei wesentlich feinerer Triangulierung der Frontebene (20003 Knoten, insgesamt 196945 Elemente) ergibt sich für den Hochlochziegel das in Abbildung 6.12 gezeigte Gitter. Für den Ziegel wurde ohne Berücksichtigung des Steinversatzes ein 0.248 m breiter und 0.365 m tiefer Teilausschnitt, der in der Höhe von Mitte Ziegel (halbe Ziegelhöhe 0.119 m) bis Mitte Mörtelschicht (halbe Schichtdicke 6 mm = 0.006 m) reichte, also von 0.125 m Gesamthöhe, zugrunde gelegt. In den Lufthohlräumen wurde die äquivalente Wärmeleitfähigkeit der Luft nach DIN EN ISO 6946 [**6**] unter Zugrundelegung eines Emissionskoeffizienten von 0.9 berechnet. Damit wurden für den Hochlochziegel mit einem Lochbild nach Abb. 6.9 folgende Wärmeleitfähigkeiten angesetzt: $\lambda_{\text{Luft}} = 0.058$ W/(mK), $\lambda_{\text{Ton}} = 0.35$ W/(mK), $\lambda_{\text{Leichtmauermörtel}} = 0.21$ W/(mK). Die Randbedingungen wurden wie folgt angenommen: Außenlufttemperatur $\vartheta_{\text{La}} = 0$ °C, Innenlufttemperatur $\vartheta_{\text{Li}} = 20$ °C, äußerer Wärmeübergangskoeffizient $\alpha_{\text{a}} = 25$ W/(m^{2}K), innerer Wärmeübergangskoeffizient $\alpha_{\text{i}} = 7.69$ W/(m^{2}K). Bei Durchrechnung bis zum Beharrungszustand (30 Tage) betrug die CPU–Rechenzeit auf einem PC mit 450–MHz–Pentium–Prozessor und 256 MB Hauptspeicher unter Borland Delphi 4 (Windows 98) 8 Stunden. Für die einmalige Aufstellung der Jacobi–Matrix (affiner Fall) wurden dabei 6.5 Stunden gebraucht. Der für den Teilausschnitt errechnete Wärmestrom betrug $\Phi = 0.187858$ W bei einer Querschnittsfläche $A = 0.0125 \cdot 0.248 = 0.031$ m^2. Abb. 6.10 zeigt für den untersuchten Hochlochziegel den Temperaturverlauf im Knoten 4522 an der Warmseite bei einer Lufttemperatur auf der Innenseite von $\vartheta_{\text{Li}} = 20$ °C, von Mitte Ziegel bis Mitte Mörtelschicht, Abb. 6.11 den Temperaturverlauf von Mitte Ziegel bis Mitte Mörtelschicht im Knoten 4425 an der Kaltseite bei einer Lufttemperatur auf der Kaltseite von $\vartheta_{\text{La}} = 0$ °C.

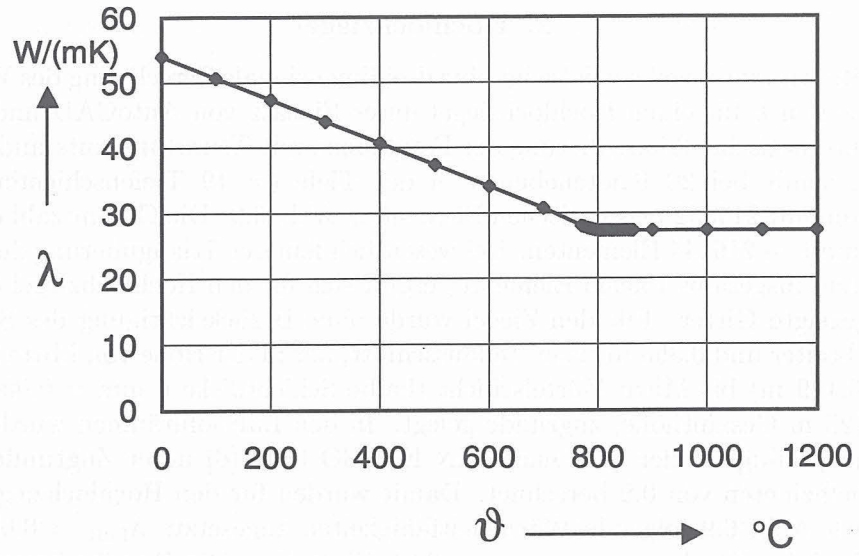


Abbildung 6.1: Wärmeleitfähigkeit von Baustahl als Funktion der Stahltemperatur.

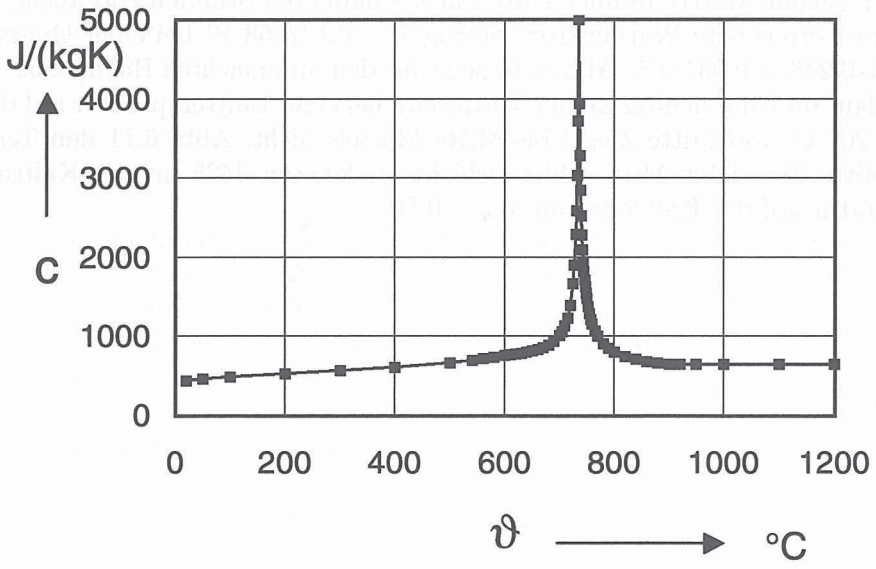


Abbildung 6.2: Spezifische Wärmekapazität von Baustahl als Funktion der Stahltemperatur.

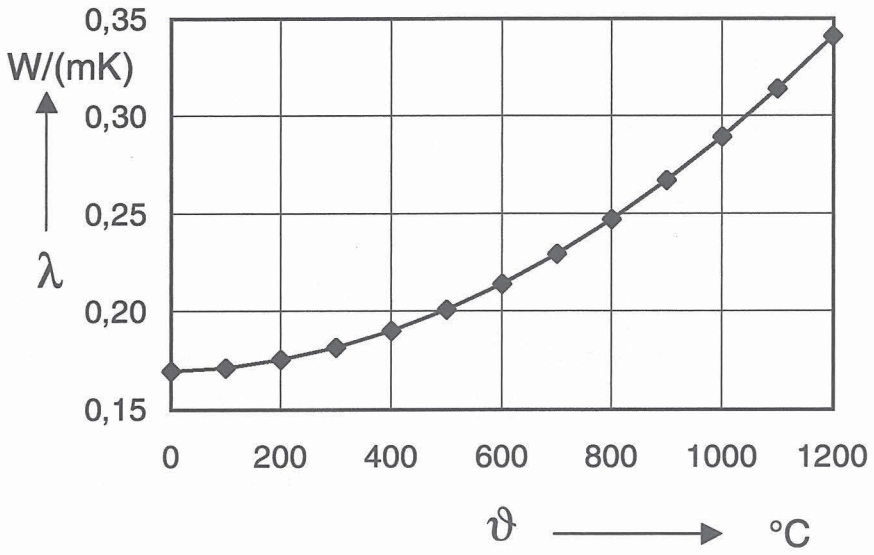


Abbildung 6.3: Wärmeleitfähigkeit der Calcium–Silikat–Platten (Promatect–H) als Funktion der Temperatur.

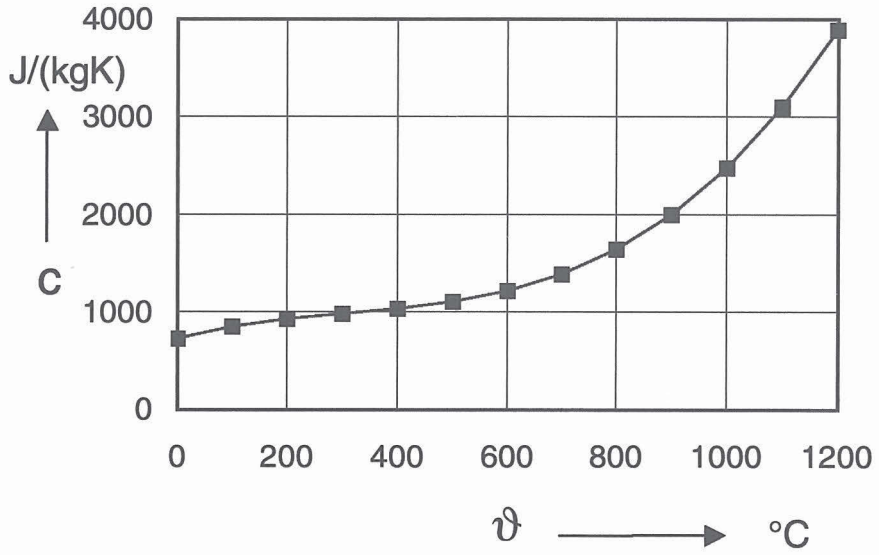


Abbildung 6.4: Spezifische Wärmekapazität der Calcium–Silikat–Platten (Promatect–H) als Funktion der Temperatur.

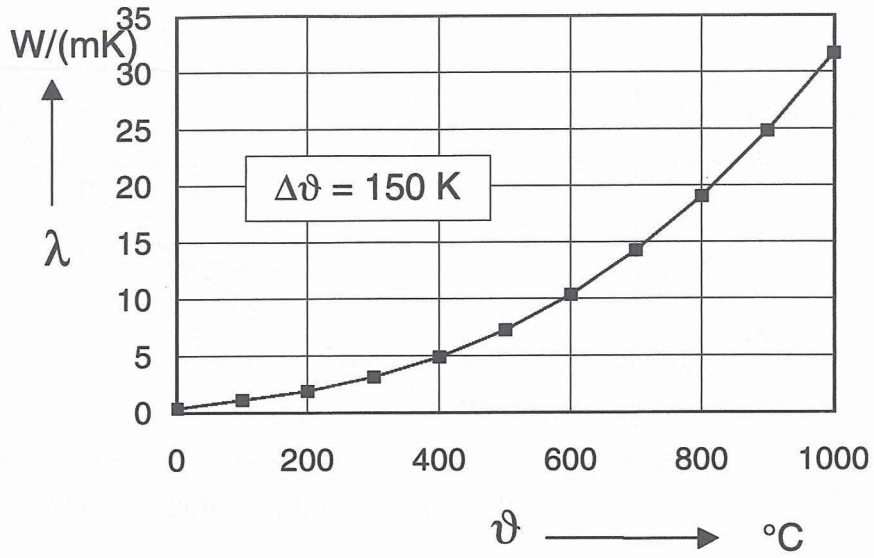
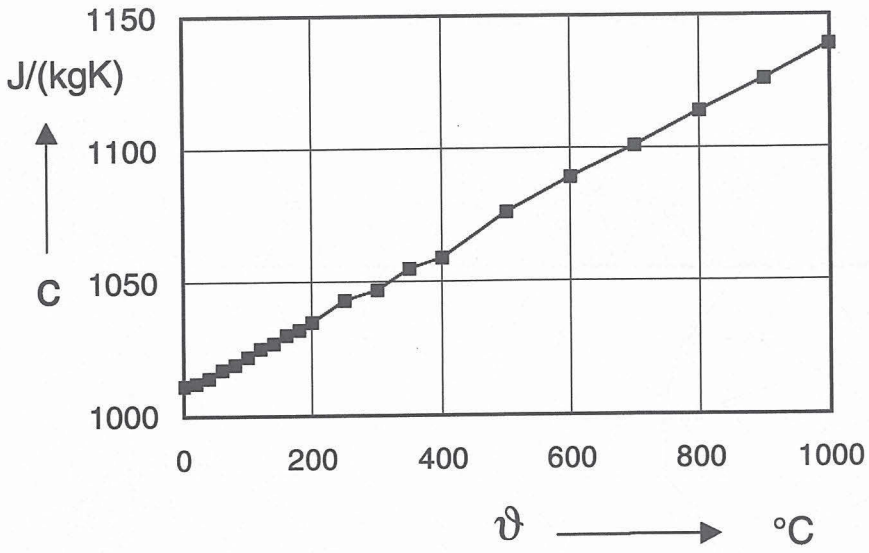
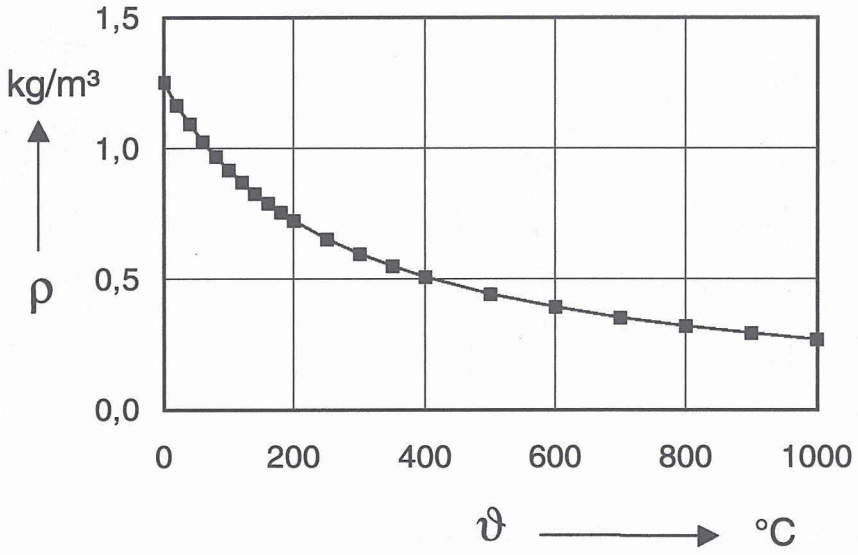


Abbildung 6.5: Äquivalente Wärmeleitfähigkeit der Luftschicht zwischen Stütze und Bekleidung als Funktion der Temperatur.



(a) Spezifische Wärmekapazität der Luft als Funktion der Temeratur.



(b) Dichte der Luft als Funktion der Temperatur.

Abbildung 6.6: Spezifische Wärmekapazität und Dichte der Luft als Funktion der Temperatur.

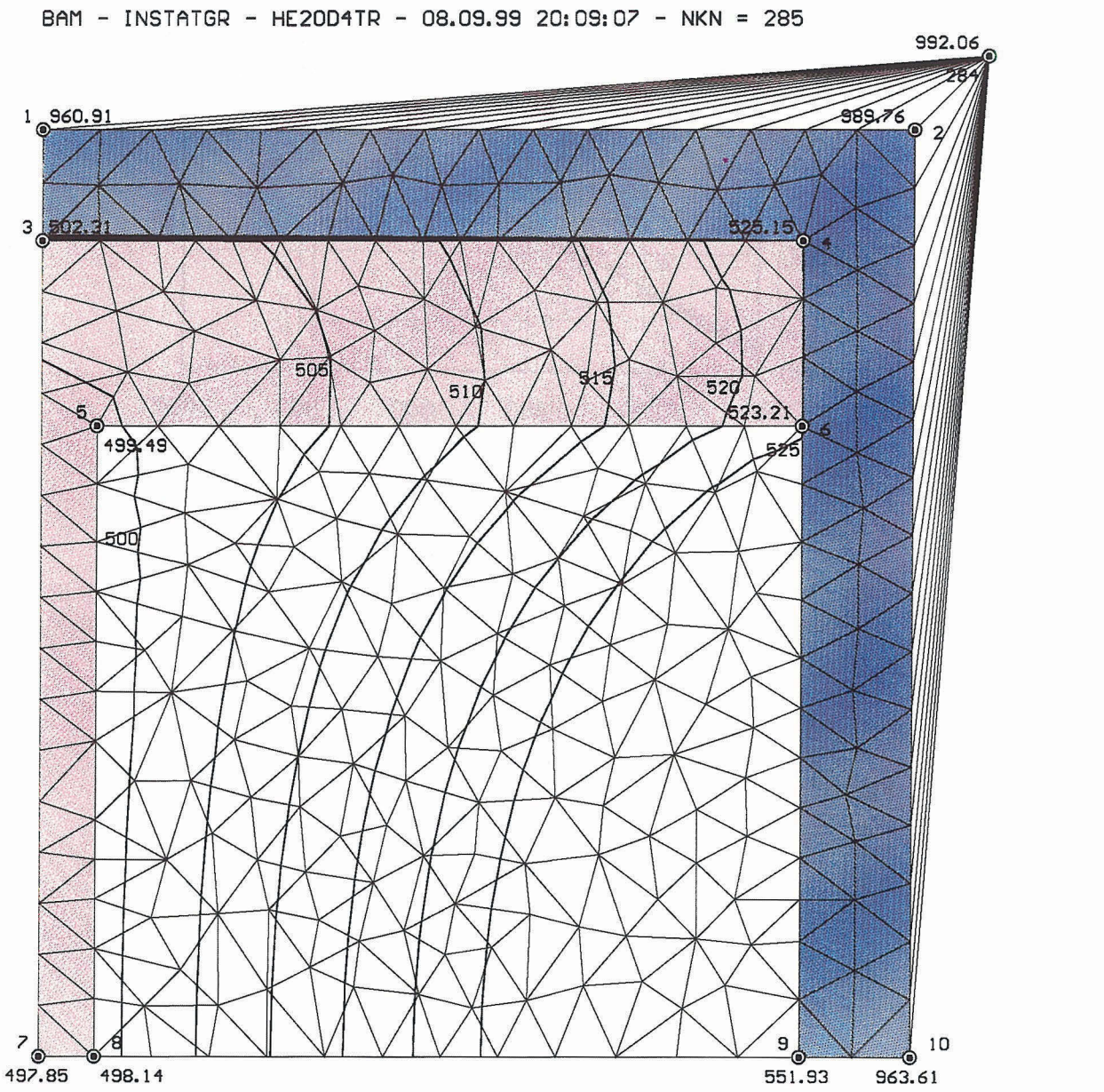


Abbildung 6.7: Untersuchter Ausschnitt mit automatisch generierter Elementaufteilung, Isothermen und Temperaturen an diskreten Stellen für eine mit 15 mm dicken Platten bekleidete HEM200–Stahlstütze nach 90 Minuten Beflammung im Stützenprüfofen der BAM.

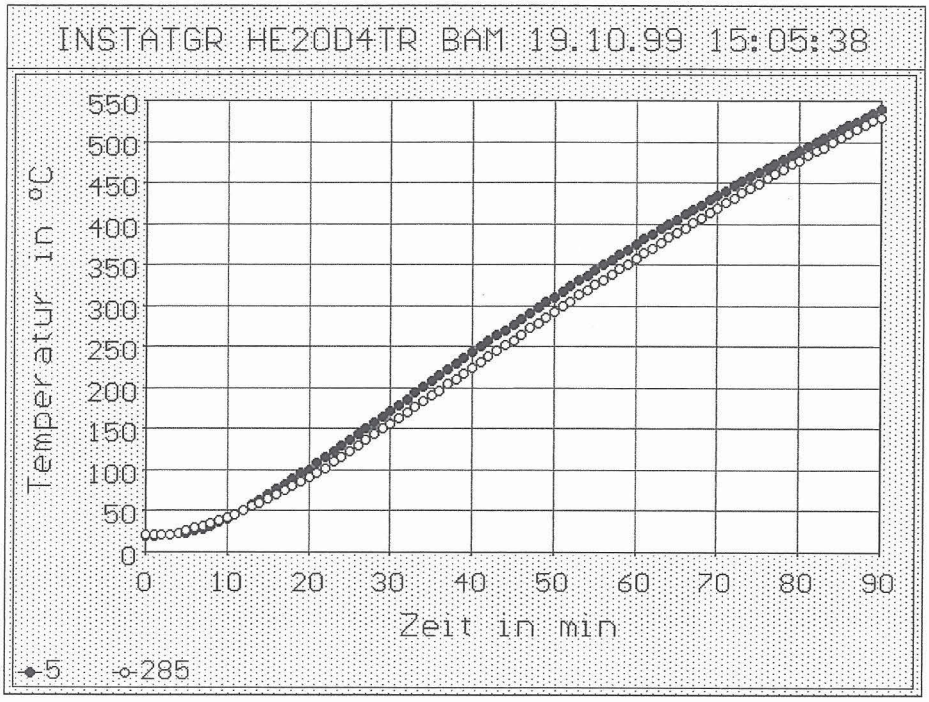


Abbildung 6.8: Für eine Brandprüfung im Stützenprüfofen der BAM an einer mit 15 mm dicken Platten bekleideten HEM200–Stahlstütze im Knoten 5 unter Berücksichtigung der Plattenfeuchtigkeit errechneter Temperatur–Zeit–Verlauf im Vergleich zu den gemessenen Werten (Knoten 285).

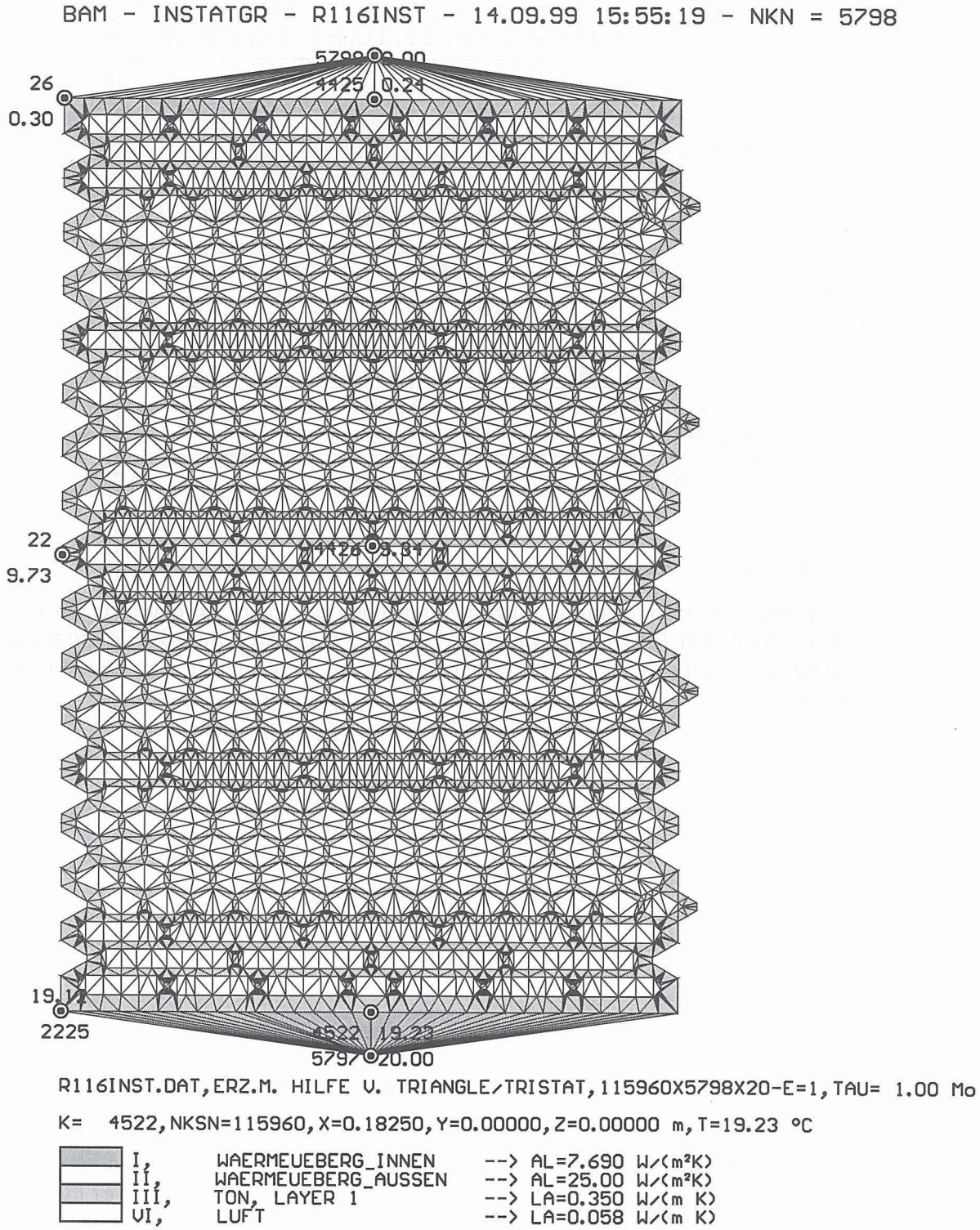


Abbildung 6.9: Elementaufteilung eines Hochlochziegels mit Temperaturen an diskreten Stellen in der Frontebene. Dreidimensionale Berechnung des Wärmedurchgangskoeffizienten k unter Einsatz von AutoCAD und TRIANGLE (115960 Knoten und 216144 Elemente).

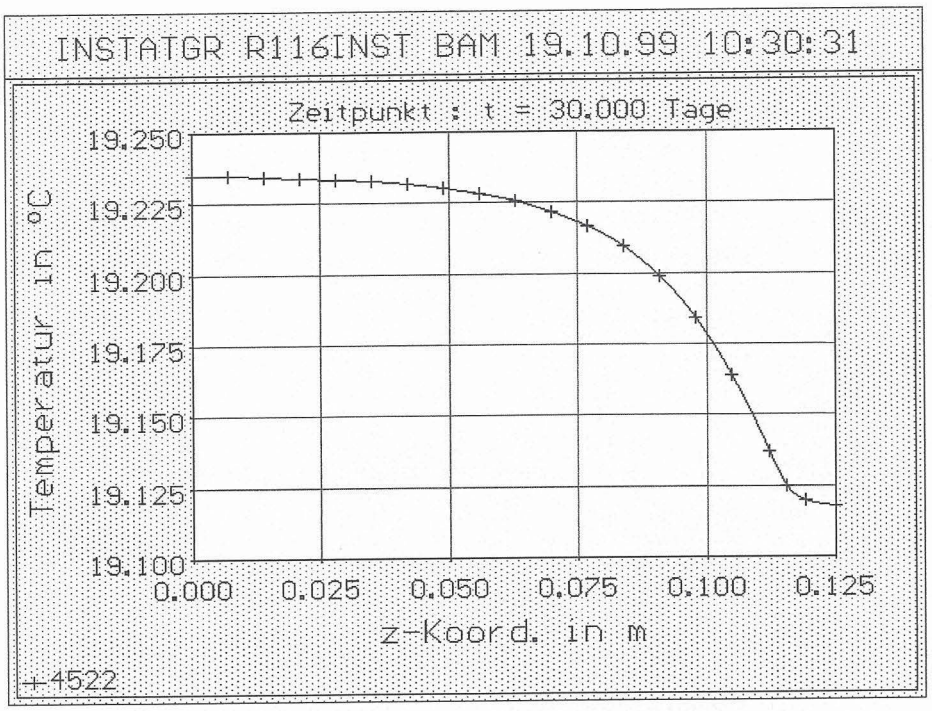


Abbildung 6.10: Temperaturverlauf von Mitte Ziegel bis Mitte Mörtelschicht im Knoten 4522 an der Warmseite $\vartheta_{\mathrm{Li}} = 20$ °C für einen Hochlochziegel.

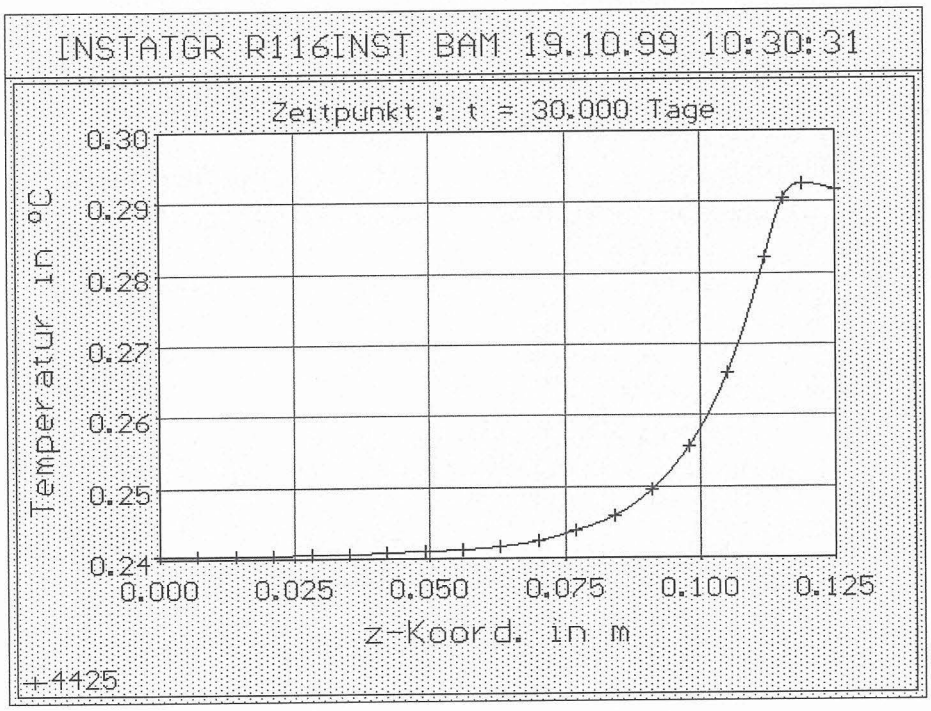


Abbildung 6.11: Temperaturverlauf von Mitte Ziegel bis Mitte Mörtelschicht im Knoten 4425 an der Kaltseite $\vartheta_{\mathrm{La}} = 0$ °C für einen Hochlochziegel.

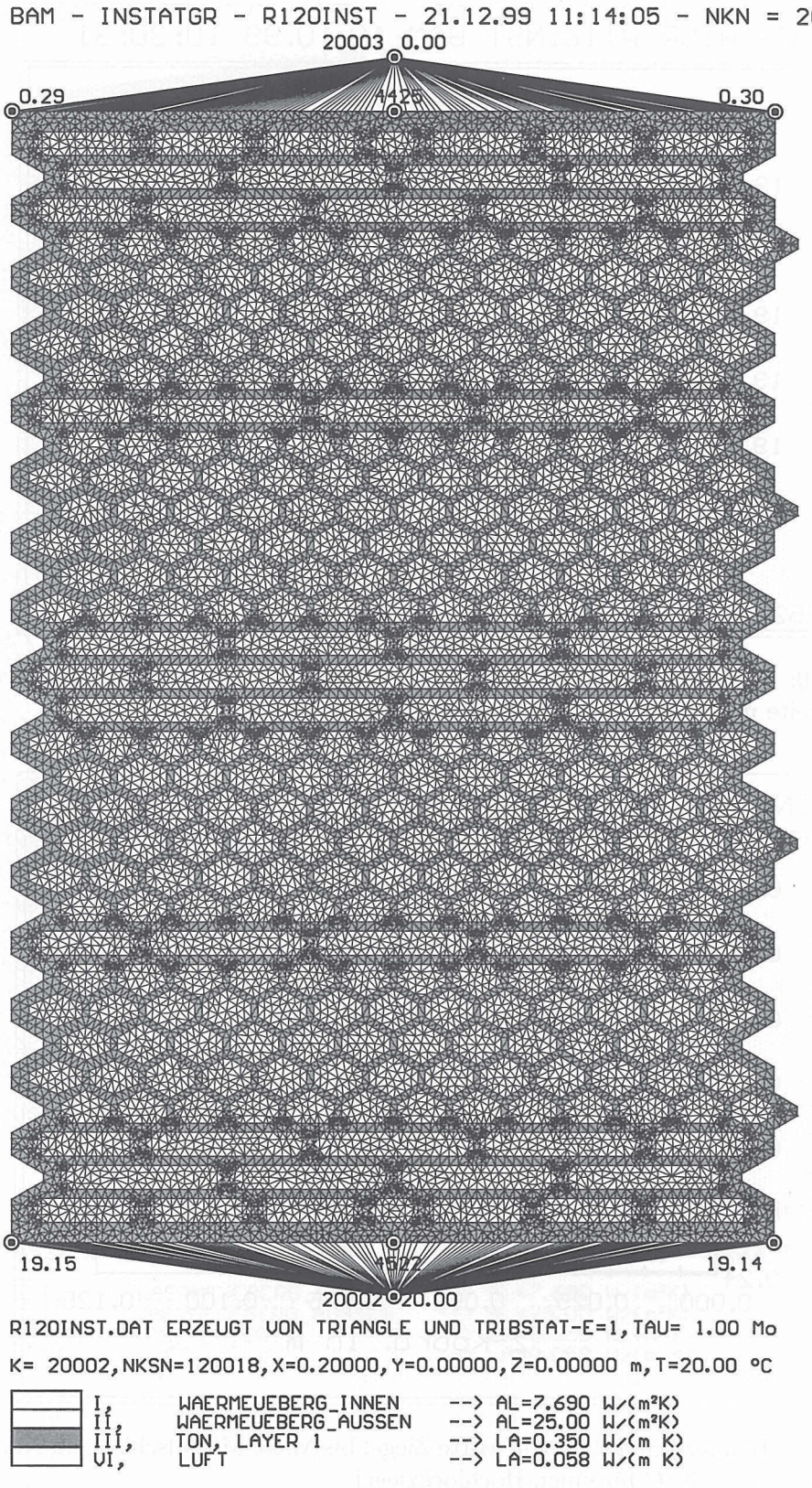
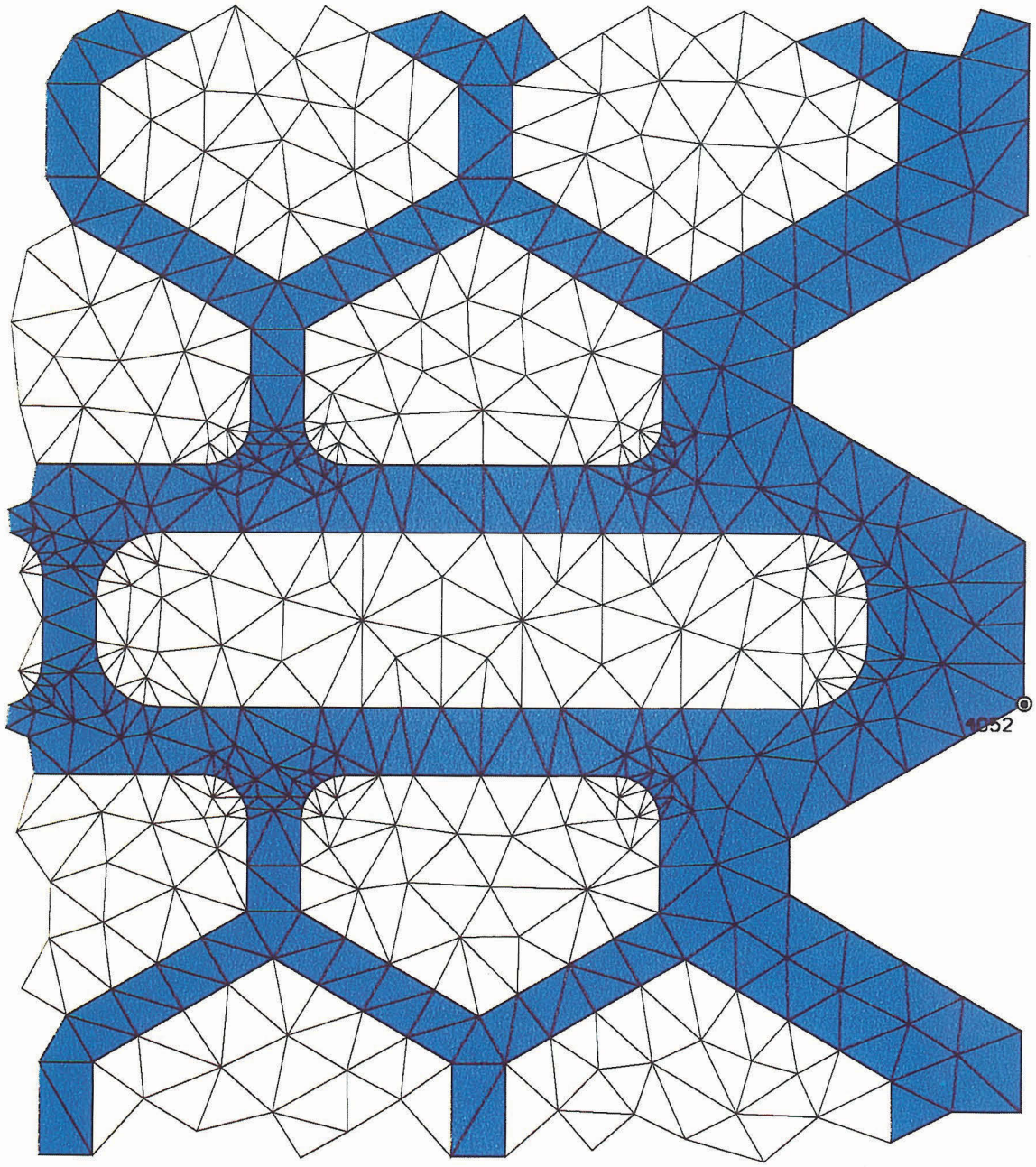


Abbildung 6.12: Elementaufteilung eines Hochlochziegels (20003 Knoten in der Frontebene).

BAM - INSTATGR - R120INST - 21.12.99 11:28:51 - NKN = 20003



R120INST.DAT ERZEUGT VON TRIANGLE UND TRIBSTAT-E=1,TAU= 1.00 Mo

K= 4052,NKSN=120018,X=0.08960,Y=0.12400,Z=0.00000 m,T=14.29 °C

III, TON, LAYER 1 --> LA=0.350 W/(m K)
VI, LUFT --> LA=0.058 W/(m K)

ANHANG A

Notation

Die wichtigsten Bezeichnungen und Symbole.

A	Matrix eines Gleichungssystems
A_h	Steifigkeitsmatrix
$\mathbf{A}$, $\mathbf{B}$, $\mathbf{G}$	Matrizen bzw. Operatoren zur Nordsiek–Form eines Mehrschrittverfahrens, siehe Seite 17ff
b	rechte Seite eines Gleichungssystems
C, C_j, c, c_{ij}, ...	Konstanten
$C(\alpha)$	Diagonalmatrix zur Anpassung der Schrittweite; vgl. Seite 17ff
$\mathbf{c}$, $\mathbf{l}$	siehe Seite 17ff
DF	Jacobi– oder Funktionalmatrix der Abbildung F
$F(.,.)$	rechte Seite in (2.1)
h	Ortsschrittweite
$\mathcal{I}$	Indexmenge
I,I_h	Identität, Einheitsmatrix
I_τ	Gitter in "Zeitrichtung"
$i(.,.)$, $k(.)$	Indexabbildungen
J_h	Jacobi–Matrix
L	Lipschitz–Konstante
M_h	Massematrix
m, m_L, m_U	Bandbreite; vgl. (4.6)
$\mathcal{O}(.)$	Landau–Symbol
p	Konsistenzordnung; vgl. Definition 2.1
$\mathbf{q}$	gegebener Vektor
$\Re z$	Realteil einer komplexen Zahl z
r_h	Residuum
$r_\tau(t)$	lokaler Diskretisierungsfehler

S	Stabilitätsbereich
t, t_j	Zeit, $t_j = t_0 + j * \tau$
u_h	Knotenvektor
u_τ, u_ν	Gitterfunktion bzgl. I_τ, Näherung für y; $u_\nu = u_\tau(t_\nu)$, vgl. 2
$x^{(m)}$	m–te Iterierte eines Verfahrens
$y(.)$	Lösung von (2.1)
$\mathbf{z}$	Nordsiek–Vektor
α, β	Koeffizienten der char. Polynome
$\chi(.,.)$	Stabilitätspolynom
ϵ, $\bar{\epsilon}$	kleine Zahl, Toleranz
η_j, η	Näherung für y oder kleine Zahl, Toleranz
$\kappa(.)$	Kondition; $\kappa(A) = \|A\| \, \|A^{-1}\|$
$\kappa_2(.)$	$\kappa_2(A) = \|A\|_2 \, \|A^{-1}\|_2$
$\lambda(X)$	Eigenwert von X
$\lambda_{\max}$, $\lambda_{\min}$	maximaler (minimaler) Eigenwert
$\rho(.), \sigma(.)$	charakteristische Polynome; vgl. (2.4)
$\rho(X)$	Spektralradius der Matrix X
τ	Zeitschrittweite
∇^ν	Differenzenoperator; vgl. 3
$\frac{\partial}{\partial y_i}$	partielle Ableitung bzgl. y_i
Δ	Laplace–Operator
Ω, $\partial\Omega$	Gebiet, Rand des Gebietes
$\langle ., . \rangle$	Euklidisches Skalarprodukt
$\|.\|$	Betrag
$\|\|.\|\|$	Vektor– oder Operatornorm
$\|\|.\|\|_2$	Euklidische Norm
$\|\|.\|\|_h$	Gitter–Norm; $\|\|.\|\|_h = \|\|h^d .\|\|_2$
$\|\|.\|\|_\infty$	Maximumnorm
$\|\|.\|\|_F$	Frobenius Norm

ANHANG B

Nordsiek–Form und Wahl der Verfahren im Programm EPISODE

Die Vektoren (3.11) werden auch im Programm DIFSUB aus Gear [**8**] verwendet:

```
Y       AN 8 BY N ARRAY CONTAINING THE DEPENDENT VARIABLES
        AND THEIR SCALED DERIVATIVES. Y(I,J+1) CONTAINS THE
        J-TH DERIVATIVE OF Y(I), SCALED BY H**J/FACTORIAL(J)
        WHERE H IS THE CURRENT STEP SIZE.
```

In dem zu Hindmarsh [**16**] gehörenden Programmpaket EPISODE wird dieser Zugang übernommen:

```
Y       AN NO BY LMAX ARRAY CONTAINING THE DEPENDENT VARIABLES
        AND THEIR SCALED DERIVATIVES.  LMAX IS 13 FOR THE
        ADAMS METHODS AND 6 FOR THE BDF METHODS.
        LMAX - 1 = MAXDER IS THE MAXIMUM ORDER AVAILABLE.
        SEE SUBROUTINE COSET.
        Y(I,J+1) CONTAINS THE J-TH DERIVATIVE OF Y(I), SCALED
        BY H**J/FACTORIAL(J)  (J = 0,1,...,NQ).
```

Es zeigt sich nun, daß die Schätzung für den Startwert des Korrektors, d. h. der Prädiktor–Prozeß in transformierter Form (die Matrix **A**), durch eine Rekursionsformel gegeben ist, die der für die Binomialkoeffizienten entspricht (PASCALsches–Dreieck). Die Programmzeilen

```
{------------------------------------------------------------
 THIS SECTION COMPUTES THE PREDICTED VALUES BY EFFECTIVELY
 MULTIPLYING THE Y ARRAY BY THE PASCAL TRIANGLE MATRIX.
 ------------------------------------------------------------}
200   T=T+H
      DO 260 J = 2,K
       DO 260 J1 = J,K
        J2 = K - J1 + J - 1
        DO 260 I = 1,N
260       Y(J2,I) = Y(J2,I) + Y(J2+I,I)
```

aus DIFSUB beschreiben den Prädiktor (in der Nordsiek–Form (3.12) ist dieses die Operation $\mathbf{z}^{q+1,(0)} = \mathbf{A}\,\mathbf{z}^{q}$). EPISODE berechnet $u_{j+k}^{(0)}$ genauso.

Die Wahl des Korrektors und die der Methode, mit der die nichtlineare Gleichung gelöst wird, beschreibt Hindmarsh [**16**] wie folgt:

```
MF    =  THE METHOD FLAG  (USED ONLY ON FIRST CALL, UNLESS
             INDEX = -1).  ALLOWED VALUES ARE 10, 11, 12, 13,
             20, 21, 22, 23.  MF HAS TWO DECIMAL DIGITS, METH
             AND MITER  (MF = 10*METH + MITER).
             METH IS THE BASIC METHOD INDICATOR..
```

```
      METH = 1  MEANS THE ADAMS METHODS.
      METH = 2  MEANS THE BACKWARD DIFFERENTIATION
                FORMULAS (BDF), OR STIFF METHODS OF GEAR.
    MITER IS THE ITERATION METHOD INDICATOR..
      MITER = 0 MEANS FUNCTIONAL ITERATION (NO PARTIAL
                DERIVATIVES NEEDED).
      MITER = 1 MEANS CHORD METHOD WITH ANALYTIC JACOBIAN.
                FOR THIS USER SUPPLIES SUBROUTINE
                PDB  (SEE DESCRIPTION BELOW).
      MITER = 2 MEANS CHORD METHOD WITH JACOBIAN CALCULATED
                INTERNALLY BY FINITE DIFFERENCES.
      MITER = 3 MEANS CHORD METHOD WITH JACOBIAN REPLACED
                BY A DIAGONAL APPROXIMATION BASED ON A
                DIRECTIONAL DERIVATIVE.
```

Hieraus ergibt sich:

- wähle als Korrektor
  - ein Adams–Moulton–Verfahren (3.4) mit $\kappa = k$, falls `METH = 1`
  - oder eine BDF–Formel (3.2), falls `METH = 2`,
- wähle zur Lösung der nichtlinearen Gleichung
  - die Fixpunktiteration (2.9), falls `MITER = 0`
  - oder ein Newton–Verfahren (2.10), falls `MITER > 0` .

Beide möglichen Korrektor–Verfahren sind implizite Mehrschrittverfahren der Form (2.3b). Hindmarsh benutzt für `MITER > 0` die Chord–Methode (Algorithmus 3) zur Bestimmung der nächsten Iterierten in (C), Algorithmus 4.

1. Implementierung

Nachfolgend ist die Modifikation der Prozedur `Psetb` (DELPHI–Version von `Psetb` aus EPISODE [**16**]) angegeben und kommentiert. Der hauptsächliche Aspekt ist, daß der in Abschnitt 4 beschriebene Vektor `p` nicht mehr auftritt, sondern die Jacobi–Matrix in einer Variablen `A` vom Type `SparseMatrix` abgelegt wird. Statt einer LU–Zerlegung und anschließender Lösung durch Vorwärts– und Rückwärtseinsetzen wird eine ILU–Zerlegung auf dem Muster von `A` durchgeführt und das Gleichungssystem mit einem BI–CGSTAB–Verfahren gelöst. Die Tabelle 2.1 listet den Speicherplatzbedarf für einige Modellprobleme (Poisson–Gleichung im Quadrat) auf. Dabei berechnet sich der Speicherplatz in Bytes für `p` aus (Faktor 1.5, da

(n,n_z,m)	(64,288,16)	(2304,11328,96)	(9604,47628,196)	(39204,195228,396)
S(p)	12.0 K	2592 K	22 M	177.6 M
S(A)	6.125 K	239.25 K	1 M	4 M

Tabelle 2.1: Speicherbedarf

die untere Bandbreite doppelt gezählt wird)

$$S(\texttt{p}) = 1.5 \cdot m \cdot n \cdot \texttt{sizeof(double)}, \qquad m = m_U + m_L + 1,$$

und für `A` aus

$$S(\texttt{A}) = 2 \cdot n_z \cdot \texttt{sizeof(double)} + (n_z + 2\,n) \cdot \texttt{sizeof(LongInt)}.$$

Typischerweise ist $n_z = cn$ (jeder Knoten einer Diskretisierung besitzt eine feste Anzahl von Nachbarn). Eine Abschätzung für $S(\mathtt{A})$ ist dann $S(\mathtt{A}) < (20c+8)n$, falls `sizeof(double)` $= 8$ und `sizeof(LongInt)` $= 4$ angenommen wird.

```
PROCEDURE Psetb_new (VAR n,ml,mu    : LONGINT;
                     VAR ym         : MATRIX ;
                     VAR con        : DOUBLE ;
                     VAR miter,ier  : LONGINT);
  // MW      = ML + MU + 1.
VAR
  ...
  index,permutation          : LongIntVector;
  werte                      : Vector       ;
  ...
BEGIN
    ....
    (* IF MITER = 2, MAKE MW CALLS TO DIFFUN TO APPROXIMATE J. *)
    de := 0;
    FOR i := 1 TO n DO de := de + SQR (save2^[i]);
    r0 := abs (h) * SQRT (de) * 1e3 * uround;
    (* THE ORIGINAL VALUES OF Y(,1) ARE SAVED TEMPORARILY IN ERROR *)
    Move (ym[1]^, error^, n * SizeOf (double));
    mwl := min_int (n,mw);
    FOR j := 1 TO mwl DO
    BEGIN
      kmax := (n - j) DIV mwl + 1;
      // Stoerung von y in den Komponenten I(j), vgl.\  Lemma 2
      // kmax entspricht k(j)
      FOR ka := 1 TO kmax DO
      BEGIN
        i := j + (ka - 1) * mwl;
        ym[1]^[i] := ym[1]^[i] + fmax(r0,epsj * ymaxi^[i]);
      END;
      // Funktionsauswertung
      Diffun (n, tvar, ym[1], save1) ;
      FOR ka := 1 TO kmax DO
      BEGIN
         // jj: entspricht i(j,p) aus Lemma 1
         // Spaltenindex der JacobiMatrix
        jj := j + (ka - 1) * mwl;
        de := con/fmax(r0,epsj * ymaxi^[jj]);
        q:=0;
         // i: Zeilenindex
         // Die Eintraege der jj-ten Spalte werden in den Vektoren
         // werte und index gespeichert
        FOR i := max_int(1,jj-mu) TO min_int(n,jj+ml) DO
        BEGIN
          DF_Eintrag := (save1^[i] - save2^[i]) * de ;
          if i=jj then DF_Eintrag:=DF_Eintrag + 1.0;
          if abs(DF_Eintrag) > 1.0E-15 then
           begin
             inc(q);
             index.yp[q] := i;
```

```
            werte.yp[q] := DF_Eintrag;
          end;
        END ; // of i
        // kopiere index und Werte nach Zw
        // Spalte jj findet sich in Zeile p von Zw
        Zw.SetZeile(p,q,index,werte);
        // deshalb wird sich ein Permutationsvektor gemerkt
        permutation.yp[jj]:=p;
        inc(p);
        for i:=1 to q do begin  index.yp[i] :=0; werte.yp[i] :=0.0; end;
        ym[1]^[jj] := error^[jj];  error^[jj] := 0 ;
      END; // of  ka
    END;   // of j
...
 werte.myFree; index.myFree;
 if A=nil then A:=SparseMatrix.init ;
 if ((Zw.NumNonZeros <> A.NumNonZeros) or (A.n <>Zw.n)) then
 begin
  if A.n > 0 then A.myFree;
  A.mycreate (Zw.n,Zw.n,Zw.NumNonzeros,false);
 end;
 // (PA)^T ist die gesuchte Jacobi-Matrix
 A.HalfPerm (Zw,permutation);
 // loesche Zw
 Zw.myFree;
 A.SetDiagonale;  A.sym  := false;
 // bestimme die ILU-Zerlegung
 A.SetIlu;
END ;  (* Psetb_new*)
```

Zur Lösung des linearen Gleichungssystems aus Algorithmus 3, also zur Bestimmung der nächsten Iterierten im Korrektor–Schritt, benutzt Hindmarsh in EPISODE [**16**] eine rechen– und speicherplatzaufwendige, in jedem Zeitschritt notwendige LU–Zerlegung (evtl. mit Pivot–Wahl) der entstehenden Funktionalmatrix $DF(t, u_c)$. In der modifizierten Fassung der DELPHI–Version der Prozeduren aus [**16**] wird in `Psetb_new` nur die Jacobi–Matrix

```
if _OLD then
    Psetb      (n0, ml, mu, ym, con, miter, ier)
 else
    Psetb_new  (n0, ml, mu, ym, con, miter, ier) ;
```

bestimmt, und der BI–CGSTAB–Algorithmus 8 ersetzt das Lösen mittels LU–Zerlegung.

```
if _OLD then
    Solbr (neband, n, ml, mu, pw, ipiv, save1)
 else
  begin
    ...
    Start_Bicg(A,xx,ff);
    ...
  end;
```

Literaturverzeichnis

[1] R.C. Aiken, editor. *Stiff Computation.* Oxford University Press, New York – Oxford, 1985.

[2] D. Braess. *Finite Elemente. Theorie, schnelle Löser und Anwendungen in der Elastizitätstheorie.* Springer, Berlin – Heidelberg – New York, 1997.

[3] P.N. Brown and Y. Saad. Hybrid krylov methods for nonlinear systems of equations. *SIAM J. Sci. Stat. Comput.*, 11(3):450–481, May 1990.

[4] W. Bunse and A. Bunse-Gerstner. *Numerische lineare Algebra.* Teubner–Verlag, Stuttgart, 1985.

[5] *DIN 4102, Teil 2: Brandverhalten von Baustoffen und Bauteilen–Bauteile–Begriffe, Anforderungen und Prüfungen*, September 1977.

[6] *DIN EN ISO 6946: Bauteile. Wärmedurchlaßwiderstand und Wärmedurchgangskoeffizient. Berechnungsverfahren*, 1996.

[7] C. W. Gear. The numerical integration of ordinary differential equations. *Mathematics of Computation*, 21:146–156, 1967.

[8] G. W. Gear. *Numerical Initial Value Problems in Ordinary Differential Equations.* Series in Automatic Computation. Prentice–Hall, Englewood Cliffs, N.J., 1971.

[9] A. Greenbaum. *Iterative Methods for Solving Linear Systems*, volume 17 of *Frontiers in Applied Mathematics.* SIAM, Philadelphia, 1997.

[10] R.D. Grigorieff. *Numerik gewöhnlicher Differentialgleichungen. Band 1: Einschrittverfahren.* Teubner, Stuttgart, 1972.

[11] R.D. Grigorieff. *Numerik gewöhnlicher Differentialgleichungen . Band 2: Mehrschrittverfahren.* Teubner, Stuttgart, 1977.

[12] W. Hackbusch. *Theorie und Numerik elliptischer Differentialgleichungen.* Teubner, Stuttgart, 1986.

[13] W. Hackbusch. *Iterative Lösung großer schwachbesetzter Gleichungssysteme.* Teubner, Stuttgart, 1991.

[14] A. C. Hindmarsh. GEAR: Ordinary differential equation system solver. Computer Documentation UCID–30001, University of California, Lawrence Livermore Lab., Livermore/California, December 1974.

[15] A. C. Hindmarsh. GEARB: Solution of ordinary differential equations having banded Jacobian. Computer Documentation UCID–30059, University of California, Lawrence Livermore Lab., Livermore/California, March 1975.

[16] A. C. Hindmarsh and G.D. Byrne. An experimental package for the integration of systems of ordinary differential equations. Computer Documentation UCID–30112, University of California, Lawrence Livermore Lab., Livermore/California, Mai 1975.

[17] Verein Deutscher Ingenieure, editor. *VDI–Wärmeatlas. Berechnungsblätter für den Wärmeübergang. 4. Auflage.* VDI–Verlag, Düsseldorf, 1984.

[18] *ISO 834, Part 1: Fire Resistance Tests–Elements of Building Construction–General Requirements to Fire Resistance Testing*, 1995.

[19] C. T. Kelley. *Iterative Methods for Linear and Nonlinear Equations*, volume 16 of *Frontiers in Applied Mathematics.* SIAM, Philadelphia, 1995.

[20] F. Liebau and R. Rudolphi. Berechnung von Temperatur– und Wärmestromverteilungen mittels algebraischer Mehrgittermethoden. Forschungsbericht 227, Bundesanstalt für Materialforschung und –prüfung, Berlin, 1998.

[21] R. Müller. Ein numerisches Verfahren zur simultanen Bestimmung thermischer Stoffeigenschaften oder Größen aus Versuchen — Anwendung auf das Heißdraht–Parallelverfahren und auf Versuche an Hausschornsteinen. Forschungsbericht 185, Bundesanstalt für Materialforschung und –prüfung, Berlin, 1992.

[22] R. Müller and R. Rudolphi. Berechnung des Wärmedurchlaßwiderstandes und der Temperaturverteilung im Querschnitt von Hausschornsteinen nach DIN 18160 Teil 6 mit abgedruckten FORTRAN–Sourcelistings sowie unter MS–DOS lauffähigen Programmen und kompletten Anwendungsbeispielen auf Diskette. Forschungsbericht 159, Bundesanstalt für Materialforschung und –prüfung, Berlin, Mai 1989.

[23] R. Müller and R. Rudolphi. Zur numerischen Ermittlung des Wärmedurchlaßwiderstandes von Hausschornsteinen. *Bauphysik*, 11(6):211–218, 1989.

[24] R. Müller, R. Rudolphi, and R. Schriever. Numerische Ableitung von thermischen Stoffeigenschaften oder Größen aus Brandversuchen. *wksb*, 44(42), 1999.

[25] S. Paasch, R. Müller, R. Rudolphi, and R. Schriever. Numerische Ableitung von thermischen Stoffeigenschaften oder Größen aus Brandprüfungen an unbekleideten und bekleideten Stahlstützen — Möglichkeiten und Grenzen —. Forschungsbericht 236, Bundesanstalt für Materialforschung und -prüfung, Berlin, 2000.

[26] R. Rudolphi. Anspruchsvolle 3D–Temperaturberechnungen bei instationären Randbedingungen. *wksb*, 43(41):40 – 46, Juli 1998.

[27] R. Rudolphi. Fortschritte bei bauphysikalischen Temperatur– und Wärmebrückenberechnungen. In *Ingenieurhochbau, Berichte aus Forschung und Praxis, Festschrift zum 60. Geburtstag von Prof. Dr. Erich Cziesielski*, pages 81–91. Werner Verlag, Düsseldorf, 1998.

[28] R. Rudolphi and B. Kownatzki. Rechenprogramm INSTATCP für 3D–Temperaturberechnungen bei instationären Randbedingungen unter Delphi und Windows. *http://www.bam.de/a_vii/calcfire/project.html*, April 1999.

[29] W. Schroeder, K. Martin, and B. Lorensen. *The Visualization Toolkit. An Object–Oriented Approach to 3D Graphics*. Prentice Hall, New Jersey, 1996.

[30] L.F. Shampine. *Numerical Solution of Ordinary Differential Equations*. Chapman & Hall, New York, 1994.

[31] Jonathan Richard Shewchuk. Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator. In *First Workshop on Applied Computational Geometry*, pages 124–133. Association for Computing Machinery, May 1996.

[32] R. Siegel, J. R. Howell, and J. Lohrengel. *Wärmeübertragung durch Strahlung. Teil 2: Strahlungsaustausch zwischen Oberflächen und in Umhüllungen*. Wärme– und Stoffübertragung. Springer–Verlag, Berlin, 1991.

[33] J. Stoer and R. Bulirsch. *Einführung in die Numerische Mathematik II*. Springer–Verlag, Berlin – Heidelberg – New York, 2. edition, 1978.

[34] H.A. Van der Vorst. BiCGSTAB: A fast and smoothly converging variant of bi–cg for the solution of nonsymmetric linear systems. *SIAM J. Sci. Statist. Comput.*, 13(2):631–644, 1992.

[35] H. Werner and H. Arendt. *Gewöhnliche Differentialgleichungen. Eine Einführung in Theorie und Praxis*. Springer–Verlag, Berlin, 1986.

ISSN 0938-5533
ISBN 3-89701-632-X