Weierstraß–Institut
für Angewandte Analysis und Stochastik

im Forschungsverbund Berlin e.V.

Numerische Lösung großer strukturierter DAE–Systeme der chemischen Prozeßsimulation

Jürgen Borchardt[1] , Ludger Bruell[2] , Friedrich Grund[1],

Dietmar Horn[1], Frank Hubbuch[2], Tino Michael[1],

Horst Sandmann[1], Robert Zeller[3]

submitted: 22nd March 1996

[1] Weierstraß–Institut
für Angewandte Analysis
und Stochastik
Mohrenstraße 39
D – 10117 Berlin

[2] Bayer AG
Zentrale Forschung
ZF–TST
Geb. E41
D – 51368 Leverkusen

[3] Cray Research GmbH
Riesstraße 25
D – 80992 München

Preprint No. 231
Berlin 1996

1991 Mathematics Subject Classification. Primary 65Y05, 65L05, 65H10, 65F50; Secondary 80A30, 92E20.

Key words and phrases. Algebraic–differential equations, systems of partitioned nonlinear equations, systems of sparse linear algebraic equations, parallelization of numerical methods, simulation of chemical processes in chemical plants.

Edited by
Weierstraß–Institut für Angewandte Analysis und Stochastik (WIAS)
Mohrenstraße 39
D — 10117 Berlin
Germany

Fax: + 49 30 2044975
e–mail (X.400): c=de;a=d400-gw;p=WIAS-BERLIN;s=preprint
e–mail (Internet): preprint@wias-berlin.de

Abstract. Parallelizable numerical methods for solving large scale DAE systems are developed at the level of differential, nonlinear and linear equations. For this the subsystem–wise structure of the DAE systems based on unit–oriented modelling is explored. Partitionings are used to parallelize waveform relaxation and structured Newton methods. Initial values are computed with a modified Newton method. To solve large sparse systems of linear equations a special Gaussian elimination method is used. The algorithms were implemented on a CRAY C90 vector computer, as well as on both, moderately parallel CRAY J90 vector computers and massively parallel CRAY T3D machines. The methods were tested using several real life examples.

1 Einleitung

Um den ständig steigenden Anforderungen an die Qualität von Produkten der Chemischen Industrie und immer größer werdenden Erwartungen im Bereich des Umweltschutzes gerecht werden zu können, reicht es nicht mehr aus, Entwicklungs- und Produktionsverfahren schrittweise zu beschleunigen und zu verbessern: Quantensprünge zur Reduktion der Entwicklungszeiten und bei der Optimierung komplexer Chemie–Anlagen sind erforderlich. Um dieses Vorhaben zu verwirklichen, ist es notwendig, durch verfahrenstechnische Simulationen die verschiedenen Aufgaben in der Anlagenplanung und -optimierung zu unterstützen. Die Vernetzung chemischer Anlagen zur Energie- und Rohstoffreduktion, zur Reduktion der Nebenprodukte oder zum Erzielen geringerer Anlageninvestitionskosten führt zu höheren Komplexitäten der Gesamtanlagen (vgl. Abbildung 1), die isolierte Betrachtungen einzelner Verfahrenseinheiten zur Optimierung nicht mehr ausreichen lassen. Um die hohen Dimensionen von Simulationsmodellen von Gesamtanlagen auch bei wachsender Modelltiefe weiterhin numerisch im Griff zu haben, ist es notwendig, über den Einsatz von sequentiellen Höchstleistungsrechnern hinaus Konzepte zur Parallelisierung der mathematischen Lösungsverfahren zu entwickeln.

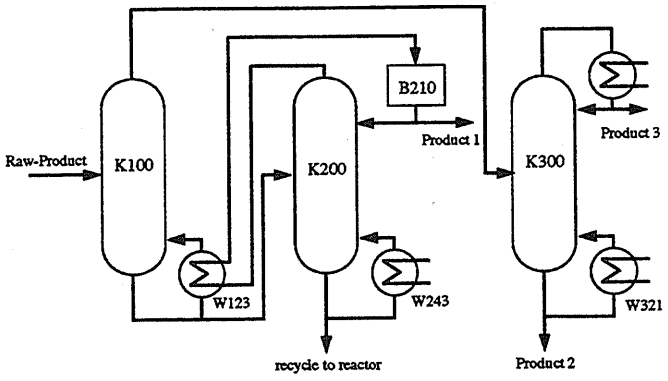


Abb. 1. Flowsheet einer wärme- und stromverkoppelten Destillationsanlage

Die mathematische Modellierung verfahrenstechnischer Prozesse in chemischen Anlagen, etwa bei der dynamischen Simulation komplexer chemischer und physikalischer Vorgänge in wärme– und stromverkoppelten Destillationskolonnen, führt auf Anfangswertprobleme für große Systeme von Algebro–Differentialgleichungen (DAE)

$$F(t, y(t), \dot{y}(t), u(t)) = 0, \quad y(t_0) = y_0, \tag{1}$$

$$F : \mathrm{R} \times \mathrm{R}^n \times \mathrm{R}^n \times \mathrm{R}^q \to \mathrm{R}^n, t \in [t_0, t_{END}],$$

wobei die Parameterfunktion $u(t)$ gegeben und $y(t) = (x_1(t), ..., x_n(t))^T$ gesucht ist.

Durch eine geeignete Modellierung kann in der Regel gesichert werden, daß das System (1) den differentiellen Index 1 hat. Es ist i. allg. ein System mit steifen Differentialgleichungen, dessen Diskretisierung und Linearisierung zu Gleichungssystemen mit schwach besetzter und nicht symmetrischer Jacobi–Matrix führt. Die Systeme können mehrere 10 000 Gleichungen umfassen und sind entsprechend der Modellierung der Gesamtanlage nach Funktionsblöcken in Teilsysteme strukturiert:

$$F_i(t, y_i, \dot{y}_i, v_i, \dot{v}_i, u) = 0, \quad y_i(t_0) = y_{i,0}, \tag{2}$$

$$F_i : \mathrm{R} \times \mathrm{R}^{n_i} \times \mathrm{R}^{n_i} \times \mathrm{R}^{(n-n_i)} \times \mathrm{R}^{(n-n_i)} \times \mathrm{R}^q \to \mathrm{R}^{n_i}, \sum_{i=1}^{m} n_i = n,$$

$$v_i = (y_1, \ldots, y_{i-1}, y_{i+1}, \ldots, y_m)^T, \quad i = 1(1)m.$$

Sowohl bei der Lösung des Anfangswertproblems als auch bei der Bestimmung der Anfangswerte y_0 betrachten wir numerische Verfahren und Algorithmen, die diese Struktureigenschaften der DAE–Systeme ausnutzen und für die Implementierung auf Parallelrechnern geeignet sind.

Parallelisierbare numerische Verfahren zur Lösung des Systems (1) werden auf drei Stufen des Lösungsprozesses – Differentialgleichungen, nichtlineare Gleichungen und lineare Gleichungen – betrachtet. Dabei wird von einer entsprechend der Struktur (2) des DAE–Systems verteilten Auswertung der Gleichungen ausgegangen.

Block–Waveform–Iterationsverfahren [4] weisen günstige Voraussetzungen für eine Parallelisierung auf der Ebene der DAE–Systeme auf (Kapitel 2). Sie sind jedoch nur für DAE–Systeme geeignet, für die eine geeignete Blockzerlegung bestimmt werden kann. Falls aufgrund starker Kopplungen zwischen den Teilsystemen keine solche Blockzerlegungen existieren, können parallelisierbare strukturierte Newton–Verfahren eingesetzt werden (Kapitel 3). Zur Berechnung der Anfangswerte $y_0 = y(t_0)$ des DAE–Systems (1) werden Gauß–Seidel–Newton–artige Verfahren verwendet (Kapitel 4). Für die Lösung der schwach besetzten linearen Gleichungssysteme wurden spezielle Gaußsche Eliminationsverfahren entwickelt (Kapitel 5).

Die numerischen Verfahren wurden bisher im wesentlichen an Modellproblemen getestet, die im Prozeßsimulator SPEEDUP [1] enthalten sind.

Hierfür wurde ein Programm entwickelt, welches aus den von SPEEDUP gelieferten Informationen eine Datenschnittstelle für unsere Verfahren erzeugt. Diese Schnittstelle beinhaltet das DAE–System in einer strukturierten Darstellung, die für eine teilsystemweise Berechnung des Funktionsvektors und der Jacobi–Matrix benötigt wird.

Mit den entwickelten Verfahren wurden Testrechnungen auch an Modellen aktueller Anlagen der Bayer AG durchgeführt. Durch Unterstützung von Cray Research konnte der lineare Solver in SPEEDUP eingebunden werden, womit Simulationen für große Produktionsanlagen durchgeführt wurden.

2 DAE–Systeme

Wir setzen voraus, daß das DAE–System (1) entsprechend (2) in Teilsysteme strukturiert ist. Die Teilvektorfunktionen $F_i(t, y(t), \dot{y}(t), u(t))$ seien ebenso wie die Hyperzeilen $\frac{\partial \tilde{F}_i}{\partial y} = \frac{\partial F_i}{\partial y} + c_i * \frac{\partial F_i}{\partial \dot{y}}$ der Jacobi–Matrix des diskretisierten Systems für jedes der m Teilsysteme separat berechenbar.

Durch eine eineindeutige Zuordnung von Variablen $y_i = (x_{i1}, ..., x_{in_i})^T$ zu den Teilsystemen $F_i = (f_{i1}, ..., f_{in_i})^T$ und durch Zusammenfassung von stark miteinander gekoppelten Teilsystemen zu Blöcken $\mathcal{F}_j = (F_{j_1}, ..., F_{j_{m_j}})^T$ mit $Y_j = (y_{j_1}, ..., y_{j_{m_j}})^T$ und $V_j = (v_{j_1}, ..., v_{j_{m_j}})$ erhalten wir eine Blockzerlegung des Problems (1) in

$$\mathcal{F}_j(t, Y_j(t), \dot{Y}_j(t), U_j(t), u(t)) = 0, \quad Y_j(t_0) = Y_{j,0}, \quad j = 1(1)M. \tag{3}$$

Der folgende Algorithmus eines Block–Waveform–Iterationsverfahrens [4] nutzt geeignete Prädiktorwerte für die Koppelvariablen $U_j = (V_j, \dot{V}_j)^T$. Dadurch können in jedem Iterationsschritt die einzelnen Teilsystem–Blöcke unabhängig voneinander über sukzessiv fortschreitenden Teilintervallen [7] des Zeitintervalls mit bekannten Verfahren, z.B. BDF, gelöst werden.

do for $p = 0, 1, 2, ...$
  set $Y_j^0[t_p, t_{p+1}]$ for $j = 1(1)M$
  do for $k = 1, 2, ...$
    do for $j = 1(1)M$
      solve for $t \in [t_p, t_{p+1}]$

$$\mathcal{F}_j(t, Y_j^k(t), \dot{Y}_j^k(t), U_j^{k-1}(t), u(t)) = 0$$

$$Y_j^k(t_p) = \tilde{Y}_j(t_p)$$

    enddo
  until $\|Y^k - Y^{k-1}\|_{[t_p, t_{p+1}]} < \epsilon$
enddo

Block–Waveform–Iterationsverfahren lassen sich günstig parallelisieren, sind jedoch nur für eingeschränkte Aufgabenklassen geeignet. Die Konvergenzeigenschaften hängen wesentlich von der zugrundeliegenden Blockzerlegung ab [3].

Es wurden Algorithmen zur Blockzerlegung von strukturierten DAE–Systemen entwickelt und implementiert. Dabei werden sowohl bei der erforderlichen Zuordnung von Variablen zu Gleichungen als auch bei der Zusammenfassung von Teilsystemen zu Blöcken "Gewichte" für die Kopplung zwischen den Variablen, Gleichungen bzw. Teilsystemen definiert, die aus der Jacobi–Matrix $A := \frac{\partial \tilde{F}(y)}{\partial y}|_{\,y=\tilde{y}}$, $A = (a_{pq}) \in \mathrm{R}^{n\times n}$ des nichtlinearen Systems $\tilde{F}(y) = 0$, $\tilde{F} : \mathrm{R}^n \to \mathrm{R}^n$ gewonnen werden, das durch Diskretisierung des DAE–Systems im Zeitpunkt $t = \tilde{t}$, $\tilde{y} \sim y(\tilde{t})$ entsteht.

Das Ziel des Zuordnungsprozesses ist, jeder Variablen x_q eineindeutig eine Gleichung f_p zuzuordnen, so daß die für die Teilsysteme $i = 1(1)m$ mit $y_i, F_i \in \mathrm{R}^{n_i}$ resultierende Zuordnung $y_i \to F_i$ konsistent bezüglich der Zustandsvariablen ist und die $n_i \times n_i$ Teilmatrizen $\frac{\partial \tilde{F}_i}{\partial y_i}$ regulär sind.

Hierzu formulieren wir das lineare gewichtete Zuordnungsproblem

$$\sum_{p=1}^{n}\sum_{q=1}^{n} w_{pq}s_{pq} \longrightarrow max, \quad \sum_{k=1}^{n} s_{pk} = 1, \sum_{k=1}^{n} s_{kq} = 1,$$

$$s_{pq} = \begin{cases} 1 : \text{Variable } x_q \text{ ist Gleichung } f_p \text{ zugeordnet,} \\ 0 : \text{sonst,} \end{cases}$$

$$w_{pq} = \begin{cases} 0 & : f_p \text{ hängt weder von } x_q \text{ noch von } \dot{x}_q \text{ ab,} \\ 1 + \frac{|a_{pq}|}{\sum_{r=1}^{n}|a_{pr}|} & : f_p \text{ hängt von } x_q \text{, nicht aber von } \dot{x}_q \text{ ab,} \\ 3 + \frac{|a_{pq}|}{\sum_{r=1}^{n}|a_{pr}|} & : f_p \text{ hängt von } \dot{x}_q \text{ ab.} \end{cases}$$

Ausgehend von der Jacobi–Matrix des Gesamtsystems erzeugen wir einen parametrisierten gerichteten Graphen und lösen das Zuordnungsproblem mittels Graphenalgorithmen aus LEDA [18] .

Bei der Zusammenfassung von Teilsystemen zu Blöcken definieren wir zunächst, was wir unter einer "starken" Kopplung zwischen Gleichungen bzw. Teilsystemen verstehen wollen. Wir nennen eine **Zeile** p der Jacobi–Matrix A mit $f_p \in F_i$ **dominant bezüglich Teilsystem i**, wenn

$$\sum_{q\notin K_i} |a_{pq}|/|a_{pp}| < 1, \quad K_i = \{r \mid f_r \in F_i\}$$

gilt. Das **Teilsystem i** wird dann als **starker Input des Teilsystems j** bezeichnet, wenn ein $p_j \in \{p \mid f_p \in F_j$, Zeile p ist nicht dominant bezüglich Teilsystem j$\}$ existiert, so daß

$$\frac{\sum_{k\in K_i} |a_{p_j k}|}{|a_{p_j p_j}|} \geq \beta_{p_j}, \quad 0 < \beta_{p_j} \leq 1$$

erfüllt ist.

Nachdem wir die starken Inputs der Teilsysteme bestimmt haben, initialisieren wir die Blöcke mit jeweils einem Teilsystem und fassen dann sukzessiv Blöcke mit starken Input–Teilsystemen zusammen. Im allgemeinen wird diese Blockzerlegung nur einmal vor Beginn des Iterationsprozesses für $t = t_0$ durchgeführt, sie kann aber für $t > t_0$ wiederholt werden, falls Konvergenzprobleme auftreten.

Gegenwärtig benutzt das Block–Waveform–Iterationsverfahren zur Integration der Blocksysteme eine Modifikation des Programms DASSL [8].

Für die Beispiele DYNEVAP (Doppel–Effekt–Verdampfer mit 13 Teilsystemen) und BTX (Destillationskolonne mit 52 Teilsystemen) wurden Blockzerlegungen bestimmt. Während das Waveform–Verfahren für DYNEVAP konvergiert, treten bei BTX aufgrund starker Rückkopplungen zwischen den Stufen der Destillationskolonnen Konvergenzprobleme auf.

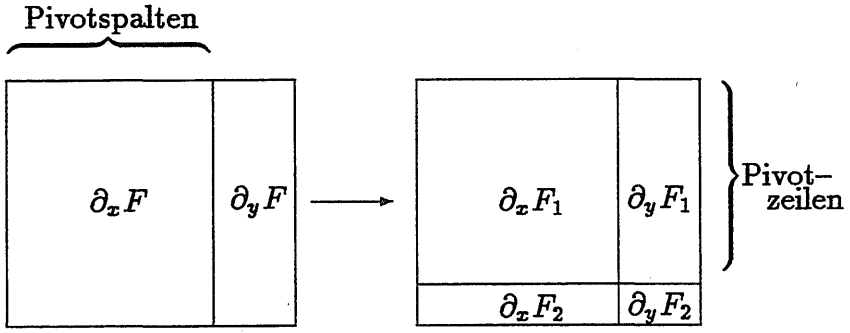
3 Nichtlineare Gleichungen

Falls die Teilsysteme (2) des DAE–Systems untereinander stark gekoppelt sind, kann durch den Partitionierungsalgorithmus keine sinnvolle Blockzerlegung für das Waveform–Verfahren gefunden werden. Es werden deshalb zur Lösung der aus (2) entstehenden nichtlinearen Gleichungssyteme strukturierte Newton–Verfahren [5, 11, 13] eingesetzt. Bei einstufig strukturierten Verfahren wird von Systemen der Form

$$\begin{aligned} F(x,y) &= 0 \\ G(y) &= 0 \end{aligned} \tag{4}$$

mit $F : \mathrm{R}^k \times \mathrm{R}^l \to \mathrm{R}^n$, $G : \mathrm{R}^l \to \mathrm{R}^p$, $k + l = p + n$, ausgegangen.

Während in [13] vorausgesetzt wird, daß sich $y = (y_1, y_2)^T$ so wählen läßt, daß $[\partial_x F \ \partial_{y_1} F]$ regulär ist, partitionieren wir in Anlehnung an [5, 11] F so in $F = (F_1, F_2)^T$, daß nur $\partial_x F_1$ regulär sein muß. Die Auswahl der Gleichungen F_1 erfolgt mit Hilfe eines Algorithmus zur Pivotwahl in der überbestimmten Teilmatrix $\partial_x F$.



Der Newton–Ansatz für (4) mit $F = (F_1, F_2)^T$ liefert

$$\begin{aligned} 0 &= F_1 + \partial_x F_1 \Delta x + \partial_y F_1 \Delta y \\ 0 &= F_2 + \partial_x F_2 \Delta x + \partial_y F_2 \Delta y \\ 0 &= G + \partial_y G \Delta y. \end{aligned}$$

Aus den beiden ersten Gleichungen erhält man

$$\begin{aligned} \Delta x &= \Delta \hat{x} + B \Delta y \\ 0 &= \hat{F}_2 + C \Delta y \end{aligned} \tag{5}$$

mit den Vektoren $\Delta \hat{x} = -(\partial_x F_1)^{-1} F_1$ und $\hat{F}_2 = F_2 + \partial_x F_2 \Delta \hat{x}$ sowie den Matrizen $B = -(\partial_x F_1)^{-1} \partial_y F_1$ und $C = \partial_x F_2 B + \partial_y F_2$. Somit kann die Berechnung der Δx und Δy eines Newton–Iterationsschrittes in drei Teilschritten erfolgen:

(i) bestimme $\Delta \hat{x}, \hat{F}_2, B$ und C aus

$$\begin{bmatrix} \partial_x F_1 & \emptyset \\ \partial_x F_2 & I \end{bmatrix} \begin{pmatrix} \Delta \hat{x} \\ \hat{F}_2 \end{pmatrix} = - \begin{pmatrix} F_1 \\ F_2 \end{pmatrix}$$

$$\begin{bmatrix} \partial_x F_1 & \emptyset \\ \partial_x F_2 & I \end{bmatrix} \begin{bmatrix} B \\ C \end{bmatrix} = - \begin{bmatrix} \partial_y F_1 \\ \partial_y F_2 \end{bmatrix},$$

(ii) bestimme Δy aus

$$\begin{bmatrix} C \\ \partial_y G \end{bmatrix} \Delta y = - \begin{pmatrix} \hat{F}_2 \\ G \end{pmatrix},$$

(iii) berechne Δx aus

$$\Delta x = \Delta \hat{x} + B \Delta y.$$

Besitzt das Gleichungssystem (4) mit $x = (x_1, ..., x_m)^T$, $y = (y_1, ..., y_m)^T$, $F = (F_1, ..., F_m)^T$ die Struktur

$$\begin{aligned} F_i(x_i, y_i) &= 0, \qquad i = 1(1)m \\ G(y) &= 0 \end{aligned} \tag{6}$$

und sind die Argumente unterschiedlicher F_i disjunkt, so können die Schritte **(i)** und **(iii)** für alle $i = 1(1)m$ unabhängig voneinander und damit parallel ausgeführt werden.

Bei der Lösung der linearen Gleichungssysteme in Schritt **(i)** kann man ihre spezielle Struktur ausnutzen. Bei ihrer LU–Faktorisierung

$$\begin{bmatrix} \partial_x F_1 & \emptyset \\ \partial_x F_2 & I \end{bmatrix} = \begin{bmatrix} L_{1,1} & \emptyset \\ L_{2,1} & I \end{bmatrix} \begin{bmatrix} U_{1,1} & \emptyset \\ \emptyset & I \end{bmatrix}$$

sind nur $L_{1,1}, L_{2,1}$ und $U_{1,1}$ zu berechnen.

Zur Reduzierung des Aufwandes bei der Lösung des Hauptsystems als wesentlichem Bestandteil des sequentiellen Anteils des Algorithmus kann man den Schritt **(ii)** im Wechsel mit einem Schritt ausführen, bei dem der den Teilsystemen entsprechende Anteil $C\Delta y^{k+1} = -\hat{F}_2$ durch

$$C_0\Delta y^{k+1} = -(\hat{F}_2 + C\Delta y^k - C_0\Delta y^k)$$

ersetzt wird, während der Anteil $\partial_y G\Delta y^{k+1} = -G$ unverändert bleibt, da G linear ist. Dabei bezeichnet C_0 die im letzten Schritt **(ii)** berechnete und bereits faktorisierte Matrix C. Somit kann in **(i)** die Berechnung der Matrix C durch die Berechnung des Vektors $C\Delta y^k$ mittels

$$\begin{bmatrix} \partial_x F_1 & \emptyset \\ \partial_x F_2 & I \end{bmatrix} \begin{pmatrix} B\Delta y^k \\ C\Delta y^k \end{pmatrix} = - \begin{pmatrix} \partial_y F_1 \Delta y^k \\ \partial_y F_2 \Delta y^k \end{pmatrix}$$

ersetzt werden. Ebenso wird der Vektor $B\Delta y^{k+1}$ für **(iii)** durch die Lösung des Gleichungssystems

$$[\partial_x F_1]\,(B\Delta y^{k+1}) = -\partial_y F_1 \Delta y^{k+1}$$

bestimmt.

Zur Erzeugung des erweiterten nichtlinearen Systems (6) für ein strukturiertes DAE–System (2) werden für die Teilsysteme F_i die inneren Variablen x_i, die nur in F_i vorkommen, und die äußeren Variablen y_i, die mindestens in zwei Teilsystemen vorkommen, bestimmt. Für die mehrfach auftretenden äußeren Variablen werden Hilfsvariablen eingefügt, so daß die Argumente verschiedener F_i disjunkt sind. Entsprechend diesen zusätzlichen Variablen werden Identitätsgleichungen $G(y) = 0$ hinzugefügt. Zur Steigerung der Effektivität der Parallelisierung können Teilsysteme so zu Blöcken in etwa gleicher Dimension zusammengefaßt (load balancing) werden, daß das Hauptsystem von möglichst geringer Dimension ist.

Ein zweistufiges Newton–Verfahren für nichtlineare Gleichungssysteme

$$\begin{aligned} F_{ij}(x_{ij}, y_{ij}, z_{ij}) &= 0 \\ G_i(y_i) &= 0, \qquad j = 1(1)m_i,\ i = 1(1)M \\ H(z) &= 0 \end{aligned} \tag{7}$$

mit $x_i = (x_{i1}, ..., x_{im_i})^T$, $x = (x_1, ..., x_M)$ und entsprechenden Vektoren y, z, F und G kann analog formuliert werden. Hierbei hat die Jacobi–Matrix die in Abbildung 2 gezeigte prinzipielle Struktur.

In diesem Fall können die auftretenden Gleichungssysteme jeweils in der F– und in der G–Ebene parallel gelöst werden, dadurch kann einerseits der sequentielle Anteil des Algorithmus reduziert werden und andererseits lassen sich mehr Prozessoren einsetzen.

Gegenwärtig ist ein einstufig strukturiertes Newton–Verfahren im DAE–Löser implementiert. Die Tabelle 1 zeigt die durch das strukturierte Newton–Verfahren erzielbaren Speedup–Faktoren bezogen auf die gesamte dynamische

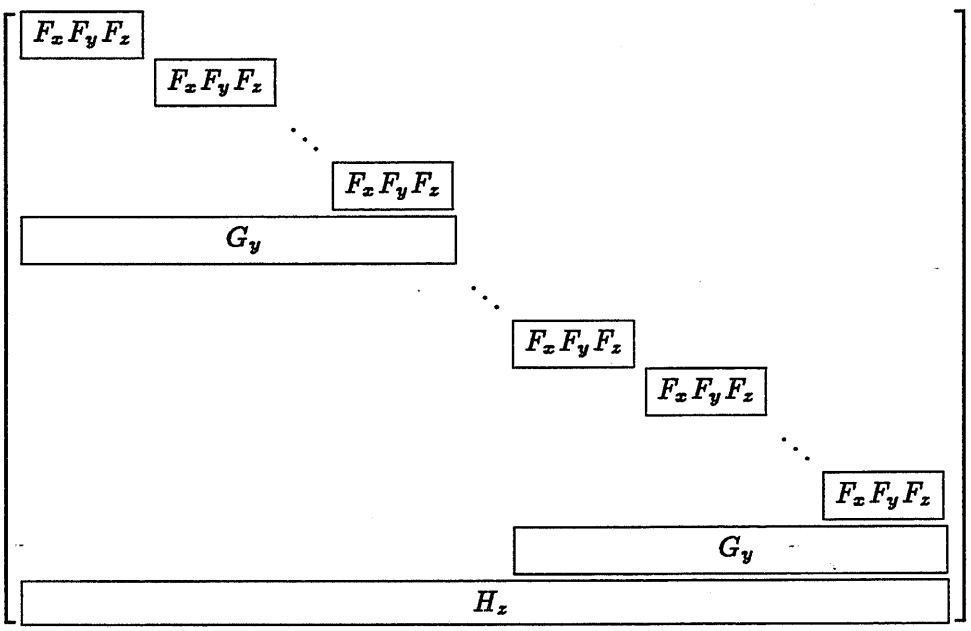


Abb. 2. Struktur der Jacobi–Matrix

Simulation der Anlage BTX (52 Teilsysteme, 1089 Gleichungen). Da mit der Anzahl der parallel behandelbaren Blöcke auch die Anzahl der äußeren Variablen und damit der Aufwand zur Lösung des Haupsystems (sequentieller Anteil des Algorithmus) steigt, ergibt sich eine maximale Beschleunigung bei 8 Blöcken und ebensovielen Prozessoren.

Tabelle 1. BTX: Beschleunigungsfaktoren auf CRAY J90

Blöcke = Prozessoren	**1**	**2**	**4**	**8**
äußere Variablen	0	12	37	73
innere Variablen	1089	1077	1052	1016
Speedup–Faktor*	**1**	**1.2**	**2.5**	**3.9**

* bestimmt mit atexpert

4 Bestimmung von Anfangswerten

Die Lösung von (1) erfordert die Bestimmung konsistenter Anfangswerte $y(t_0) = y_0$ [15]. Dabei ist zu gegebenen $\dot{y}_0 \equiv \dot{y}(t_0)$, $u_0 \equiv u(t_0)$ eine Lösung $x = y_0$ des nichtlinearen Gleichungssystems

$$F(t_0, x, \dot{y}_0, u_0) = 0 \tag{8}$$

zu bestimmen. Dieses Problem tritt auch bei der Bestimmung einer stationären Lösung bzw. bei einer Reinitialisierung während der dynamischen Simulation auf [16].

Für die Lösung des Systems (8), das in dieser allgemeinen Form numerisch schwierig zu behandeln ist, wird ausgenutzt, daß durch die Modellierung Wertebereiche für die Unbekannten, wie etwa positive Konzentrationen, Dichten oder Molenbrüche, gegeben sind. Somit kann von einem im Wertebereich liegenden x_{init} ausgegangen werden, um zunächst mit einem Suchverfahren (**S1**, **S2**) einen Startwert zu bestimmen, der im Einzugsbereich eines Iterationsverfahrens liegt. Dabei werden nacheinander für $i = 1(1)m$ Näherungslösungen für die entsprechend der Struktur (2) bestehenden nichtlinearen Teilsysteme bestimmt. Die unterbestimmten linearen Systeme

$$\partial_x F_i \Delta x = -F_i(t_0, x, \dot{y}_0, u_0)$$

werden unter Verwendung der Moore–Penrose–Pseudo–Inversen gelöst. Die Korrektur von x_{init} erfolgt in drei Schritten:

(S1) Block–Gauss–Seidel–Iteration:

$z_{00} := x_{init}$
do for $j = 0(1)j_{end}$
  do for $i = 1(1)m$

$$z_{ji} := z_{ji-1} - \partial_x F_i^T \left(\partial_x F_i (\partial_x F_i)^T\right)^{-1} F_i$$

  enddo
enddo

(S2) Block–Gauss–Seidel–Newton–Iteration:

$w_{00} := z_{j_{end}m}$
do for $j = 0(1)j_{end}$
  do for $i = 1(1)m$
    $w_{ji}^0 := w_{ji-1}$
    do for $k = 1, 2, ...$

$$w_{ji}^k := w_{ji}^{k-1} - \lambda \partial_x F_i^T \left(\partial_x F_i (\partial_x F_i)^T\right)^{-1} F_i$$

    until $\|w_{ji}^k - w_{ji}^{k-1}\| < \epsilon$
    $w_{ji} := w_{ji}^k$
  enddo
enddo

(S3) modifiziertes gedämpftes Newton–Verfahren:

$x^0 := w_{j_{end}m}$

do for $k = 1, 2, ...$

$$x^k = x^{k-1} - \lambda^k (\partial_x F)^{-1} F$$

until $\|x^k - x^{k-1}\| < \epsilon$

Der Dämpfungsparameter λ^k des Newton–Verfahrens wird wie in [19] beschrieben bestimmt. Zu Beginn, aber auch während der Iteration in **(S3)**, können numerisch singuläre Jacobi–Matrizen auftreten. Für diesen Fall wurde das Newton–Verfahren mit einer Matrix–Analyse versehen, in deren Ergebnis wenige Komponenten von x^k verändert werden. Wenn dann die Matrix regulär ist, wird mit **(S3)** fortgesetzt, sonst erfolgt mit einem neuen x_{init} ein Übergang zu **(S1)**.

Das skizzierte Verfahren wurde auf dem Parallelrechner CRAY T3D als skalierbarer synchroner Algorithmus implementiert. Dabei wird für die einzelnen Teilsysteme die Berechnung der jeweiligen Anteile der Funktion und der Jacobi–Matrix parallel ausgeführt. Die Programme wurden an den Beispielen DYNEVAP und BTX erprobt. Obwohl während der Iteration wiederholt singuläre Jacobi–Matrizen auftraten, konnten für beide Beispiele konsistente Anfangswerte berechnet werden. Für das Beispiel BTX wurden Beschleunigungsfaktoren zwischen vier und fünf erzielt.

5 Lineare Gleichungen

Das lineare Gleichungssystem

$$Ax = b, \quad A \in \mathrm{R}^{n \times n}, \quad x, b \in \mathrm{R}^n \tag{9}$$

mit einer nicht symmetrischen und im allgemeinen sehr schwach besetzten Matrix A wird mit dem Gaußschen Eliminationsverfahren

$$PAQ = LU, \qquad Ly = Pb, \qquad UQ^{-1}x = y$$

gelöst. Das Verfahren wird für die Entwicklung von Methoden sowohl bei Vektorrechnern als auch bei Parallelrechnern verwendet.

Von der Matrix A werden nur die Nichtnull-Elemente (NNE) gespeichert. Eines der kompliziertesten Probleme beim Gaußschen Eliminationsverfahren für schwach besetzte Matrizen ist die Auffindung einer Pivotreihenfolge, also die Bestimmung der Permutationsmatrizen P und Q. Hierbei sind verschiedene, sich teilweise widersprechende Bedingungen zu erfüllen. Die Pivotreihenfolge muß so sein, daß das Verfahren numerisch stabil und das Fill-in während der Elimination gering ist.

Bei der Auswahl der Pivotelemente werden nur Elemente zugelassen, die die sogenannte β-Bedingung erfüllen. Es sei $I = \{1, 2, ..., n\}$, und es bezeichne

$$\hat{a}_j = \max_{i \in I} |a_{ij}|, \qquad \forall j \in I.$$

Ein Nichtnull-Element erfüllt die β-Bedingung für ein $\beta \in [0, 1]$, wenn

$$\hat{a}_j \cdot \beta \leq |a_{ij}|$$

gilt. Bei praktischen Rechnungen wurden mit $\beta = 0.01$ bzw. $\beta = 0.001$ gute Resultate erzielt. Für die Bestimmung der Pivotelemente wird eine der nachfolgend genannten vier Strategien verwendet. Pivot wird ein Element mit minimalen Markowitz-Kosten

1. in dem noch nicht faktorisierten Teil der Matrix oder
2. in der ersten Zeile mit einer minimalen Anzahl von NNE oder
3. in der ersten Spalte mit einer minimalen Anzahl von NNE oder
4. in allen Spalten mit einer minimalen Anzahl von NNE.

Falls mehrere Elemente die Bedingungen erfüllen, wird als Pivot immer das betragsgrößte Element genommen. Die 1. Methode liefert im allgemeinen die besten Ergebnisse, sie ist aber sehr aufwendig und wird nur für kleinere Probleme (weniger als 1 000 Gleichungen) benutzt. Mit der 3. Methode wurden bei größeren Problemen (mehrere 10 000 Gleichungen) die günstigsten Resultate gewonnen.

Da bei den numerisch zu behandelnden Systemen von DAE's die Verteilung der NNE in der Jacobi-Matrix sich nicht ändert, bei den Faktorisierungen fast immer mit derselben Pivotreihenfolge gearbeitet wird und viele Systeme mit gleicher Jacobi-Matrix aber verschiedenen rechten Seiten zu lösen sind, wird mit einem Pseudo-Code gearbeitet. Dieser beschreibt die Operationen, die für die Faktorisierung von A bzw. für die Vor- und Rückwärtsrechnung notwendig sind. Er kann betriebssytemunabhängig formuliert werden und ist insbesondere auch für Vektor- und Parallelrechner geeignet[12].

Hierzu wird $LU = PAQ$ eine Matrix $M = (m_{i,j})$ zugeordnet. Die Elemente von M, sie beschreiben das Niveau der Unabhängigkeit der Elemente von A, werden mit dem Algorithmus von Yamamoto und Takahashi [22] bestimmt. Alle Matrixelemente von A mit gleichem Niveau der Unabhängigkeit in der Matrix M können unabhängig voneinander faktorisiert werden.

Der Pseudo-Code wird auf Vektorrechner und auf Parallelrechner mit shared bzw. distributed Memory angepaßt.

Bei einem Vektorrechner müssen Vektoranweisungen in jedem Niveau der Unabhängigkeit bestimmt werden. Es haben sich die folgenden Vektoroperationen mit indirekt adressierten Elementen bewährt:

Skalarprodukt
$A(K) = 1/A(K)$
$A(K) = A(K) * A(L)$
$A(K) = (A(I) * A(J) + A(L) * A(M)) * A(K).$

Für verschiedene Matrizen, die während der dynamischen Prozeßsimulation von großen Anlagen der Bayer AG mit SPEEDUP auftraten, wurde unser Algorithmus GSPAR mit FRONTAL, einem linearen Solver in SPEEDUP, der die Frontal-Methode benutzt, verglichen. Die Tabelle 2 zeigt die Ergebnisse für die Faktorisierung mit gegebener Pivot-Reihenfolge.

Tabelle 2. Rechenzeit in CPU-Sek., CRAY Y-MP

Beispiel	Gleichungen	NNE	Fill-in	Faktorisierung	
				FRONTAL	GSPAR
SPA	3 083	21 216	22 437	0.162	0.079
MPC	6 747	56 196	79 395	0.403	0.271
CSA	13 935	63 679	147 023	0.679	0.463
DEST	13 436	94 926	214 919	1.290	0.738
NIT	20 545	159 082	275 553	2.209	0.983

Bei den Parallelrechnern mit distributed Memory (z.B. CRAY T3D) wird der Pseudo-Code in jedem Niveau annähernd gleichmäßig auf den Speicher der Prozessoren verteilt und dann immer wieder ausgeführt, während bei Parallelrechnern mit shared Memory (z.B. CRAY J90) die Arbeit so organisiert wird, daß jeder Prozessor in etwa dieselbe Arbeit auf jedem Niveau leisten muß.

Für das Beispiel CSA wurden mit einer CRAY T3D die Ergebnisse in Tabelle 3 erzielt. Die Rechenzeit mit 4 Prozessoren wurde als Vergleichsgrundlage mit dem Speedup–Faktor 1.0 belegt.

Tabelle 3. Beispiel CSA: Rechenzeit in CPU-Sek., CRAY T3D

Prozessoren	Faktorisierung	Speedup–Faktor
4	1.57	1.00
8	0.99	1.58
16	0.60	2.61

Der Solver GSPAR wurde an großen Simulationsbeispielen der Bayer AG im Simulator SPEEDUP erprobt. Die Resultate für die Gesamtsimulation einer Destillationskolonne (DEST) mit 13 436 Gleichungen und eines Reaktormodells (REAK) mit 3 268 Gleichungen sind in Tabelle 4 dargestellt.

Die beachtliche Verringerung der gesamten Simulationszeit ist wie folgt zu erklären. GSPAR ist bei der Faktorisierung mit gegebener Pivotreihenfolge schneller als FRONTAL. Der Zeitanteil des linearen Solvers an der gesamten Simulationszeit wird mit etwa 50 % geschätzt, wodurch mit Tabelle 2 die star-

Tabelle 4. Gesamtsimulation, Rechenzeit in CPU-Sek., CRAY C90

Anlage	Zeitintervall in h	SPEEDUP mit FRONTAL	SPEEDUP mit GSPAR	in %
DEST	(0,2)	380.89	254.75	67
REAK	(0,10)	451.71	283.73	63

ke Verringerung der Rechenzeit nicht ausreichend zu erklären ist. Es kommt hinzu, daß GSPAR numerisch stabiler ist, weshalb weniger Faktorisierungen und Vor- und Rückwärtsrechnungen und damit weniger Newton-Schritte erforderlich sind.

Dieses Projekt wird in Kooperation mit der Bayer AG Leverkusen, der Cray Research GmbH München und AspenTech UK Cambridge durchgeführt und vom BMBF der Bundesrepublik Deutschland im Rahmen eines Programms auf ausgewählten Gebieten anwendungsorientierter Mathematik gefördert.

Literatur

1. Aspen Technology, SPEEDUP, User Manual, Library Manual, Aspen Technology, Inc., Cambridge, Massachusetts, USA, 1995
2. H. G. Bock, J. P. Schlöder, V. H. Schulz, Numerik großer Differentiell–Algebraischer Gleichungen – Simulation und Optimierung, in: Prozeßsimulation, herausg. von H. Schuler, VCH Verlagsgesellschaft, Weinheim, 1995, 35–80
3. J. Borchardt, I. Bremer, Zur Analyse großer strukturierter chemischer Reaktionssysteme mit Waveform–Iterationsverfahren, IAAS Berlin, Preprint Nr. 65, 1993
4. J. Borchardt , F. Grund, Parallelisierung numerischer Methoden zur Lösung großer Systeme von Algebro–Differentialgleichungen, Studie, WIAS, 1994
5. J. Borchardt , F. Grund , D. Horn , M. Uhle, MAGNUS–Mehrstufige Analyse großer Netzwerke und Systeme, WIAS, Report No. 9, 1994
6. J. Borchardt , F. Grund , D. Horn , T. Michael, H. Sandmann, Beschreibung der Schnittstelle eines Solvers für strukturierte DAE–Systeme auf MPP–Rechnern, WIAS, 1994
7. I. Bremer, Kurveniteration auf diskreten Zeitskalen, IWR Universität Heidelberg, Preprint 93 – 06, 1993
8. K.E. Brenan, S.L. Campbell, L.R. Petzold, Numerical Solution of Initial–Value Problem in Differential–Algebraic Equations, North–Holland, New York, 1989
9. L. Brüll, U. Pallaske, On differential algebraic equations with discontinuities, Z. Angew. Math. Phys. (ZAMP) 43, 1992, 319–327
10. P. Deuflhard, Global Inexact Newton Methods for Very Large Scale Nonlinear Problems, Konrad–Zuse–Zentrum für Informationstechnik

Berlin, Preprint SC 90–2, 1990
11. G. Elst, G. Pönisch, On two–level Newton methods for the analysis of large–scale nonlinear networks, Proc. ECCTD'85 Prag, 1985, 165–169
12. F. Grund, Numerische Lösung von hierarchisch strukturierten Systemen von Algebro–Differentialgleichungen, in Intern. Ser. of Num. Math., Vol. 117, Birkhäuser Verlag Basel, 1994, 17–31
13. W. Hoyer, J. W. Schmidt, Newton–Type Decomposition Methods for Equations Arising in Network Analysis, ZAMM 64 (1984) 9, S. 397–405
14. F. J. Keil, Application of Numerical Methods in Chemical Process Engineering, Proceedings Workshop Scientific Computing in der Verfahrenstechnik, Hamburg, Juni 1995
15. B. J. Leimkuhler, L. R. Petzold, C. W. Gear, On the consistent initialization of differential–algebraic systems of equations, SIAM J. Numeric. Anal., Vol. 28, No. 1, 205–226, 1991
16. C. Majer, W. Marquardt, and E. D. Gilles, Reinitialization of DAEs after Discontinuities, Fifth European Symposium on Computer Aided Process Engineering ESCAPE 5, 507 – 512, in Comp. & Chem. Engin. Suppl., 1995
17. R. März, E. Griepentrog, Differential–Algebraic Equations and Their Numerical Treatment, Teubner Leipzig, 1986
18. St. Näher, LEDA Manual 3.0, Max–Planck–Institut für Informatik, MPI–I–93–109, Saarbrücken, 1993
19. U. Nowak, L. Weimann, A Family of Newton Codes for Systems of Highly Nonlinear Equations, Technical Report TR 91–10 (December 1991), Konrad–Zuse–Zentrum für Informationstechnik Berlin
20. A. R. Secchi, M. Morari, E. C. Biscaia Jr., The Waveform Relaxation Method in the Concurrent Dynamic Process Simulation, Computers and Chemical Engineering, Vol. 17, No. 7, 1993, 683–704
21. G. Wozny, Simulation und Energetische Analyse thermischer Trennprozesse in Bodenkolonnen, Habilitationsschrift, Universität–Gesamthochschule Siegen, 1983
22. F. Yamamoto, S. Takahashi, Vectorized LU decomposition algorithms for large–scale circuit simulation, IEEE Trans. on Comp. Aided Des. CAD–4, 1985, 232–239

Recent publications of the Weierstraß–Institut für Angewandte Analysis und Stochastik

Preprints 1995

202. Sergei Leonov: On the solution of an optimal recovery problem and its applications in nonparametric statistics.

203. Jürgen Fuhrmann: A modular algebraic multilevel method.

204. Rolf Hünlich, Regine Model, Matthias Orlt, Monika Walzel: Inverse problems in optical tomography.

205. Michael H. Neumann: On the effect of estimating the error density in nonparametric deconvolution.

206. Wolfgang Dahmen, Angela Kunoth, Reinhold Schneider: Operator equations, multiscale concepts and complexity.

207. Annegret Glitzky, Konrad Gröger, Rolf Hünlich: Free energy and dissipation rate for reaction diffusion processes of electrically charged species.

208. Jörg Schmeling: A dimension formula for endomorphisms – The Belykh family.

209. Alfred Liemant: Leitfähigkeit eindimensionaler periodischer elektrischer Netze.

210. Günter Albinus: A thermodynamically motivated formulation of the energy model of semiconductor devices.

211. Dmitry Ioffe: Extremality of the disordered state for the Ising model on general trees.

212. Stefan Seelecke: Equilibrium thermodynamics of pseudoelasticity and quasiplasticity.

Preprints 1996

213. Björn Sandstede: Stability of N–fronts bifurcating from a twisted heteroclinic loop and an application to the FitzHugh–Nagumo equation.

214. Jürgen Sprekels, Songmu Zheng, Peicheng Zhu: Asymptotic behavior of the solutions to a Landau–Ginzburg system with viscosity for martensitic phase transitions in shape memory alloys.

215. Yuri I. Ingster: On some problems of hypothesis testing leading to infinitely divisible distributions.

216. Grigori N. Milstein: Evaluation of moment Lyapunov exponents for second order linear autonomous SDE.

217. Hans Günter Bothe: Shift spaces and attractors in non invertible horse shoes.

218. Gianfranco Chiocchia, Siegfried Prößdorf, Daniela Tordella: The lifting line equation for a curved wing in oscillatory motion.

219. Pavel Krejčí, Jürgen Sprekels: On a system of nonlinear PDE's with temperature–dependent hysteresis in one–dimensional thermoplasticity.

220. Boris N. Khoromskij, Siegfried Prößdorf: Fast computations with the harmonic Poincaré–Steklov operators on nested refined meshes.

221. Anton Bovier, Véronique Gayrard: Distribution of overlap profiles in the one–dimensional Kac–Hopfield model.

222. Jürgen Sprekels, Dan Tiba: A duality–type method for the design of beams.

223. Wolfgang Dahmen, Bernd Kleemann, Siegfried Prößdorf, Reinhold Schneider: Multiscale methods for the solution of the Helmholtz and Laplace equation.

224. Herbert Gajewski, Annegret Glitzky, Jens Griepentrog, Rolf Hünlich, Hans–Christoph Kaiser, Joachim Rehberg, Holger Stephan, Wilfried Röpke, Hans Wenzel: Modellierung und Simulation von Bauelementen der Nano– und Optoelektronik.

225. Andreas Rathsfeld: A wavelet algorithm for the boundary element solution of a geodetic boundary value problem.

226. Sergej Rjasanow, Wolfgang Wagner: Numerical study of a stochastic weighted particle method for a model kinetic equation.

227. Alexander A. Gushchin: On an information–type inequality for the Hellinger process.

228. Dietmar Horn: Entwicklung einer Schnittstelle für einen DAE–Solver in der chemischen Verfahrenstechnik.

229. Oleg V. Lepski, Vladimir G. Spokoiny: Optimal pointwise adaptive methods in nonparametric estimation.

230. Bernd Kleemann, Andreas Rathsfeld, Reinhold Schneider: Multiscale methods for boundary integral equations and their application to boundary value problems in scattering theory and geodesy.