Institut für Angewandte Analysis
und Stochastik

im Forschungsverbund Berlin e.V.

Zur Analyse großer strukturierter chemischer Reaktionssysteme mit Waveform–Iterationsverfahren

Jürgen Borchardt und Ingo Bremer

submitted: 8th October 1993

Institut für Angewandte Analysis
und Stochastik
Hausvogteiplatz 5 – 7
D – 10117 Berlin
Germany
email:borchardt/bremer@iaas–berlin.d400.de

Preprint No. 65
Berlin 1993

1991 Mathematics Subject Classification. 65 L 05, 65 Y 05, 80 A 30.
Key words and phrases. Kinetik chemischer Reaktionssysteme, Anfangswertproblem für gewöhnliche Differentialgleichungen, Waveform–Iteration, strukturierte Analyse, parallele Algorithmen.

Herausgegeben vom
Institut für Angewandte Analysis und Stochastik
Mohrenstraße 39
D – 10117 Berlin

Fax: + 49 30 2004975
e–mail (X.400): c=de;a=d400;p=iaas-berlin;s=preprint
e–mail (Internet): preprint@iaas-berlin.d400.de

1 Einleitung

Bei der Analyse chemischer Reaktionssysteme (chemische Kinetik) geht es darum, die zeitliche Veränderung der Massenverhältnisse der beteiligten chemischen Spezies zu bestimmen. Ausgangspunkt sind dafür die Anfangskonzentrationen der Spezies eines gegebenen Reaktionsgemisches. Zur Modellierung dieses Problems ist es erforderlich, daß man für die relevanten Teilreaktionen des Systems jeweils die relativen molaren Verhältnisse der beteiligten Spezies und eine Abschätzung für die Geschwindigkeitskonstante der Reaktion kennt. Unter vereinfachenden Annahmen erhält man als mathematisches Modell für dieses Problem eine Anfangswertaufgabe für ein autonomes System gewöhnlicher Differentialgleichungen. Diese Differentialgleichungssysteme sind in der Regel numerisch steif und haben unregelmäßig, aber sparsam besetzte Jacobimatrizen. Aus diesem Grund gelangen bei ihrer numerischen Lösung vorrangig implizite (insbesondere BDF–Methoden) [22] [23] bzw. semiimplizite [24] Integrationsverfahren zum Einsatz. Das Vorgehen ist dabei klassisch zeitinkrementell. Das System gewöhnlicher Differentialgleichungen wird in der Zeit diskretisiert und dann sequentiell für fortschreitende Zeitpunkte in seiner Gesamtheit gelöst. Die Lösung der dabei für implizite Integrationsverfahren entstehen nichtlinearen Gleichungssysteme wird über Newton–artige Verfahren auf die Lösung linearer Systeme zurückgeführt. Eine derartige Behandlungsweise führt bei großen Systemen insbesondere zu einem erheblichen Rechenzeitaufwand.
In der vorliegenden Arbeit wird eine andere Vorgehensweise untersucht, die insbesondere bei großen Systemen zu Anwendung kommen soll. Dabei wird das Gesamtsystem in Teilsysteme zerlegt und dann das so strukturierte Problem mit angepaßten Methoden iterativ entkoppelt. Hierfür gibt es verschiedene Ansätze auf unterschiedlichen Ebenen des numerischen Lösungsprozesses [18], [17]. Eine Methode die sich schon bei der Simulation elektrischer Schaltungen bewährt hat, ist die Waveform-Iteration [5]. Hierbei wird das Gesamtsystem auf dem Niveau der Differentialgleichungen entkoppelt und die Teilsysteme werden getrennt voneinander über Zeitfenster gelöst, wobei u.a. mit verschiedener Zeitschrittweitenwahl und lokaler Konvergenz der Teilsysteme gearbeitet wird. Diese Vorgehensweise bietet günstige Voraussetzungen für eine parallele Implementierung des Lösungsalgorithmus. Von entscheidender Bedeutung ist dabei die Wahl einer geeigneten Strukturierung des Gesamtsystems in möglichst nur schwach wechselwirkende Teilsysteme. Dieser Umstand macht klar, daß ein solches Vorgehen nur auf geeignete Aufgabenklassen sinnvoll anwendbar ist.
Die vorliegende Arbeit behandelt das Problem der Partitionierung chemischer Reaktionssysteme in Teilsysteme und die Möglichkeiten der Implementierung entsprechender Waveform–Iterationsverfahren auf Transputersystemen.
Im folgenden Abschnitt wird kurz auf die Modellierung chemischer Reaktionssysteme eingegangen, um dann im 3. Abschnitt die topologischen (strukturellen) und numerischen Eigenschaften dieser Systeme hinsichtlich ihrer Strukturierbarkeit zu untersuchen. Abschnitt 4 beschreibt den Algorithmus zur Strukturierung des Differentialgleichungssystems, der neben der Auswertung der stöchiometrischen Matrix des Reaktionssystems insbesondere auf Informationen aus der Jacobimatrix des Differentialgleichungssystems zurückgreift. Wir gehen kurz auf die Realisierung des Algorithmus und auf ein illustrierendes Beispiel ein.
Abschnitt 5 enthält eine kurze Beschreibung der Waveformiteration. Besonders wird auf die Methode der Verwendung variabler Zeitfenster (Zeitsegmente) aus [25] eingegangen. Bei der Diskussion der numerischen Resultate zu einem Beispiel aus der chemischen Reaktionskinetik

in Abschnitt 6 stehen die Zusammenhänge von „starken“ Kopplungen und dem Grenzschichtverhalten von Lösungskomponenten zur Effiziens des Waveformalgorithmus im Vordergrund. Im Abschnitt 7 wird die parallele Variante der Waveform-Iteration und ihre Implementierung vorgestellt. Die bisher erfolgten Testrechnungen zeigen die prinzipielle Durchführbarkeit des Algorithmus auf Architekturen mit verteiltem Speicher (distributed memory).
Für Vergleichsrechnungen wurde der Simulator LARKIN [8] verwendet, der uns freundlicherweise von seinen Autoren zur Verfügung gestellt wurde. Er hat sich bei allen Experimenten als sehr zuverlässig erwiesen.
Unsere Arbeiten wurden im Rahmen des SFB 123 von der Deutschen Forschungsgemeinschaft gefördert.

2 Modellierung

Die r Reaktionen, die in einem Reaktionsgemisch aus n Spezies S_i ablaufen, lassen sich formal in der Form

$$\sum_{i=1}^{n} \nu_{ij} S_i = 0, \quad j = 1(1)r \tag{2.1}$$

schreiben. Dabei sind mit ν_{ij} die stöchiometischen Koeffizienten bezeichnet, die angeben, in welchen relativen molaren Verhältnissen die Spezies S_i an den Reaktionen beteiligt sind. Für sie gilt:

$\nu_{ij} > 0$ wenn die Spezies S_i Produkt der j–ten Reaktion ist,

$\nu_{ij} < 0$ wenn die Spezies S_i Reaktant der j–ten Reaktion ist,

$\nu_{ij} = 0$ wenn die Spezies S_i nicht an der j–ten Reaktion beteiligt ist.

Unter der Annahme, daß Druck, Temperatur und Volumen im Reaktionsgemisch konstant bleiben und alle Reaktionen räumlich homogen ablaufen, sind die Konzentrationen c_i der Spezies S_i reine Zeitfunktionen. Der Zustand eines solchen Systems wird durch den Vektor der molaren Konzentrationen $c = (c_1, \ldots, c_n)^T$ mit $c_i \in \overline{\mathbb{R}}^+$, d.h. $c_i \geq 0$, charakterisiert und sein Zeitverhalten läßt sich durch ein System gewöhnlicher Differentialgleichungen

$$\frac{dc}{dt} = \nu R(c), \tag{2.2}$$

mit der Anfangsbedingung

$$c(0) = c^0,$$

für $t \in [0, t_{end}]$. beschreiben.
Dabei bezeichnet die $n \times r$–dimensionale Matrix $\nu = (\nu_{ij})$ die Stöchiometrische Matrix und die Komponenten $R_j(c)$ von $R(c)$ sind die Raten mit denen die einzelnen Reaktionen ablaufen.
Für diese Raten gilt:

$$R_j(c) = r_j \prod_{k \in I_j} c_k^{-\nu_{kj}}, \qquad j = 1(1)r, \tag{2.3}$$

wobei r_j die konstante Reaktionsgeschwindigkeit (Ratenkonstante) der j-ten Reaktion bezeichnet und die Indexmenge I_j genau die Indizes der Spezies enthält, die Reaktant der j-ten Reaktionen sind. Man beachte dabei, daß in (2.3) für die stöchiometrischen Koeffizienten stets $\nu_{kj} < 0$ gilt, da das Produkt nur über Spezies gebildet wird, die Reaktant der j-ten Reaktion sind. Damit sind die rechten Seiten von (2.2) Polynome in den c_i und unter der Voraussetzung, daß diese hinreichend glatt sind, existiert eine eindeutig bestimmte Lösung $c(t)$ lokal in t, wobei $c_i(t) \geq 0$ für $c_i(0) \geq 0$ für alle $i = 1(1)n$ gilt.
Wir betrachten zur Illustration folgendes Beispiel eines Reaktionssystems mit $r = 6$ Reak-

tionen zwischen $n = 7$ Spezies und den Ratenkoeffizienten $r_1, r_2, \ldots, r_7$ (siehe [8]).

$$\begin{aligned} H_2 + O_2 &\xrightarrow{r_1} H^* + HO_2 \\ H_2 + OH^* &\xrightarrow{r_2} H^* + H_2O \\ O_2 + H^* &\xrightarrow{r_3} OH^* + O^* \\ H_2 + O^* &\xrightarrow{r_4} H^* + OH^* \\ H^* &\xrightarrow{r_5} \\ O^* &\xrightarrow{r_6} \end{aligned} \tag{2.4}$$

Die Terme rechts und links von den Reaktionspfeilen werden als Komplexe bezeichnet. Die Spezies auf der linken Seite sind Reaktanten der Reaktion und bilden den Reaktionskomplex. Die Spezies auf der rechten Seite sind Produkte der Reaktion und bilden den Produktkomplex.
Es handelt sich hierbei um ein offenes System, da die Abbruchreaktionen 5 und 6 (sogenannte Pseudoreaktionen) die Abführung der Spezies H^* und O^* aus dem Reaktionssystem beschreiben. Für diese Reaktionen ist der Produktkomplex der sogenannte "Nullkomplex".
Die Stöchiometrische Matrix des Systems hat die Gestalt:

$$\nu = \begin{bmatrix} -1 & -1 & 0 & -1 & 0 & 0 \\ -1 & 0 & -1 & 0 & 0 & 0 \\ 0 & +1 & 0 & 0 & 0 & 0 \\ +1 & +1 & -1 & -1 & -1 & 0 \\ 0 & -1 & +1 & +1 & 0 & 0 \\ 0 & 0 & +1 & -1 & 0 & -1 \\ +1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{array}{l} : H_2 \\ : O_2 \\ : H_2O \\ : H^* \\ : OH^* \\ : O^* \\ : HO_2 \end{array} \tag{2.5}$$

In diesem Fall ist der Betrag sämtlicher stöchiometrischer Koeffizienten gleich 1.
Bezeichnet c_1 die Konzentration von H_2, c_2 die von O_2, c_3 die von H_2O usw., so gilt für die Ratenfunktionen R_i:

$$\begin{aligned} R_1(c) &= r_1 * c_1 * c_2 \\ R_2(c) &= r_2 * c_2 c_5 \\ R_3(c) &= r_3 * c_2 * c_4 \\ R_4(c) &= r_4 * c_1 * c_6 \\ R_5(c) &= r_5 * c_4 \\ R_6(c) &= r_6 * c_6 \end{aligned} \tag{2.6}$$

und man erhält für (2.2) das Differentialgleichungssystem:

$$\begin{aligned} \dot{c}_1 &= -R_1(c) - R_2(c) - R_3(c) \qquad (2.7) \\ \dot{c}_2 &= -R_1(c) - R_3(c) \\ \dot{c}_3 &= +R_2(c) \\ \dot{c}_4 &= +R_1(c) + R_2(c) - R_3(c) + R_4(c) - R_5(c) \\ \dot{c}_5 &= -R_2(c) + R_3(c) + R_4(c) \qquad (2.8) \\ \dot{c}_6 &= +R_3(c) - R_4(c) - R_6(c) \\ \dot{c}_7 &= +R_1(c) \end{aligned}$$

3 Topologische und numerische Eigenschaften chemischer Reaktionssysteme

Identifiziert man gleiche Komplexe, so lassen sich chemische Reaktionssysteme als Netzwerke auffassen. Wenn man dabei nicht den konkreten Wert der Ratenkonstanten betrachtet, sondern nur ob eine Reaktion stattfindet ($r_j \neq 0$) oder nicht, so beschreibt ein solches Reaktionsnetzwerk die topologische Struktur einer Klasse von Reaktionssystemen und läßt sich als Graph mit sogenannten Verbindungsklassen darstellen.
Für Beispiel (2.4) hat das Reaktionsnetzwerk die Gestalt:

$$\begin{aligned} S_1 + S_2 &\longrightarrow S_4 + S_7 \qquad (3.1)\\ S_1 + S_5 &\longrightarrow S_4 + S_3 \\ S_2 + S_4 &\longrightarrow S_5 + S_6 \qquad (3.2)\\ S_1 + S_6 &\longrightarrow S_4 + S_5 \end{aligned}$$

$$\begin{matrix} S_4 \\ S_6 \end{matrix} \searrow K_\emptyset$$

Knoten des zugehörigen Reaktions–Graphen sind die Komplexe $K_\emptyset$, $K_1 = S_1 + S_2$, $K_2 = S_4 + S_7, \ldots, K_{10} = S_6$, wobei $K_\emptyset$ den "Null"–Komplex bezeichnet, einen Komplex der keine Spezies enthält. Das Reaktionsnetzwerk beschreibt in diesem Fall 6 Reaktionen zwischen 11 Komplexen mit 7 Spezies und besitzt 5 Verbindungsklassen.
Die folgenden Notationen sind Grundlage für die Charakterisierung chemischer Reaktionsnetzwerke.
Ordnet man jeder Spezies S_i eines Reaktionsnetzwerkes den Standard–Basisvektor e_i des $\mathbb{R}^n$ zu, so kann man für die j-te Reaktion mittels

$$y_{r_j} := \sum_{S_i \in K_{r_j}} \nu_{ij} e_i \quad \text{und} \quad y_{p_j} := \sum_{S_i \in K_{p_j}} \nu_{ij} e_i \qquad (3.3)$$

den Komplexvektor y_{r_j} des Reaktionskomplexes K_{r_j} und den Komplexvektor y_{p_j} des Produktkomplexes K_{p_j} definieren.
Der Reaktionsvektor w_j der j-ten Reaktion ist dann die Differenz der beiden Komplexvektoren

$$w_j := y_{p_j} - y_{r_j} \qquad (3.4)$$

Im Beispiel (3.3) ist etwa für die 1. Reaktion:

$$\begin{aligned} K_{r_j} &= K_1 = S_1 + S_2 \\ K_{p_j} &= K_2 = S_1 + S_2 \\ y_{r_j} &= y_1 = e_1 + e_2 = (1,1,0,0,0,0,0)^T \\ y_{p_j} &= y_2 = e_4 + e_7 = (0,0,0,1,0,0,1)^T \end{aligned}$$

und folglich

$$w_1 \;=\; y_2 - y_1 = (-1,-1,0,1,0,0,1)^T$$

Die Reaktionsvektoren w_j sind identisch mit den Spalten der stöchiometrischen Matrix. Im Beispiel ist w_1 die erste Spalte in (2.5). Der von den Reaktionsvektoren w_j , $j = 1(1)r$, aufgespannte Unterraum

$$\mathbb{S} \;:= span_{j=1(1)r}(w_j) \subset \mathbb{R}^n \tag{3.5}$$

heißt stöchiometrischer Unterraum des Reaktionsnetzwerkes und für seine Dimension s gilt:

$$s := \;\dim(\mathbb{S}) = rank(\nu) \leq n \tag{3.6}$$

Damit läßt sich (2.2) in der Form

$$\dot{c} \;=\sum_{j=1}^{r} \left[r_j \prod_{k\in I_j} c_k^{-\nu_{kj}}\right] w_j \tag{3.7}$$

schreiben. Die rechte Seite von (3.7) ist eine Linearkombination der Reaktionsvektoren w_j und folglich gilt $\dot{c} \in \mathbb{S}$, d. h. der stöchiometrische Unterraum schränkt die möglichen Richtungen für die Lösung c im $\mathbb{R}^n$ ein.
Es kann gezeigt werden, daß wenn zwei Konzentrationsvektoren c und c' die Gleichungen (2.2) erfüllen, so liegt $c - c'$ in $\mathbb{S}$. Man nennt c und c' dann stöchiometrisch kompatibel bzw. zu der selben stöchiometrischen Kompatibilitätsklasse des $\mathbb{R}^n_+$ gehörend.
Mit Hilfe des stöchiometrischen Unterraums lassen sich Reaktionsnetzwerke klassifizieren und es können qualitative Aussagen über das Verhalten der Lösungstrajektorien der entsprechenden chemischen Reaktionssysteme gewonnen werden. Eine wichtige Größe ist dabei der sogenannte "Defekt" δ eines Reaktionsnetzwerkes, der sich nach

$$\delta \;:= N - l - s \tag{3.8}$$

berechnet. Dabei bezeichnet N die Anzahl der Komplexe, l die Anzahl der Verbindungsklassen des Netzwerkes und s ist die Dimension des stöchiometrischen Unterraumes. Der Defekt δ ist eine nichtnegative ganze Zahl und kann als Differenz der Anzahl der aufgrund der Graphenstruktur maximal möglichen unabhängigen Reaktionen und der Anzahl der tatsächlich unabhängigen Reaktionen des Netzwerkes interpretiert werden. Für Beispiel (2.4) ist $N = 11$, $l = 5$, $s = b$ und somit $\delta = 0$.
Für Reaktionsnetzwerke mit $\delta = 0$ und $\delta = 1$ haben Othmer [9], Feinberg [10] und andere Aussagen über das qualitative Verhalten der Lösung dieser Systeme angegeben. Wir wollen hier nur die Hauptsätze für schwach reversible Netzwerke mit Defekt $\delta = 0/1$ zitieren.
In diesem Zusammenhang wird ein Reaktionsnetzwerk reversibel genannt, wenn für jede chemische Reaktion auch ihre Gegenreaktion existiert, d. h. alle Reaktionen reversibel sind. Zwischen den Knoten (Komplexen) des Reaktionsgraphen bestehen dann nur bidirektionale Kanten. Verallgemeinernd heißt ein Netzwerk schwach reversibel, wenn jede gerichtete Kante des Reaktionsgraphen in einem gerichteten Zyklus enthalten ist. Reversible Netzwerke sind somit stets auch schwach reversibel.
Beispiel (2.4) ist nicht reversibel.

Satz 3.1 *Für alle Reaktionsnetzwerke mit $\delta = 0$ gilt unabhängig von den (positiven) Werten der Ratenkonstanten:*

1. *Wenn das Netzwerk nicht schwach reversibel ist, läßt $\dot{c} = \nu R(c)$ keinen positiven Gleichgewichtszustand und keine periodische Lösung zu.*

2. *Wenn das Netzwerk schwach reversibel ist, hat $\dot{c} = \nu R(c)$ genau einen positiven Gleichgewichtszustand in jeder stöchiometrischen Kompatibilitätsklasse. Der Gleichgewichtszustand ist (hinsichtlich c^0) asymptotisch stabil und es gibt keine periodischen Lösungen.*

Satz 3.2 *Für ein schwach reversibles Reaktionsnetzwerk mit l Verbindungsklassen und*

a) $\delta_i \leq 1, \quad i = 1(1)l,$

b) $\sum_{i=1}^{l} \delta_i = \delta$

läßt $\dot{c} = \nu R(c)$ genau einen positiven Gleichgewichtszustand in jeder stöchiometrischen Kompatibilitätsklasse zu.

Satz 3.2 beinhaltet den Spezialfall $l = 1$, $\delta = 1$. Man beachte, daß die asymptotische Stabilität für Netzwerke mit $\delta = 1$ nicht mehr gesichert werden kann.
Als eine wesentliche Aussage erhält man aus Satz 3.1, daß Reaktionssysteme, deren Netzwerke den Defekt $\delta = 0$ haben, keine periodischen Lösungen erzeugen, ob sie nun (schwach) reversibel sind oder nicht. Reaktionssysteme, deren Netzwerke nicht schwach reversibel sind, können nur Gleichgewichtszustände annehmen, bei denen die Konzentration mindestens einer Spezies den Wert 0 annimmt. Offene Reaktionssysteme wie (2.4) sind in der Regel nicht schwach reversibel.
Nicht reversible Reaktionsnetzwerke lassen sich zusätzlich über die Topologie der stöchiometrischen Matrix wie folgt charakterisieren:

1. Gibt es für einen Index $i \leq n$ einen Index $j \leq r$, so daß $v_{ij} > 0$ ($v_{ij} < 0$) und $v_{kj} = 0 \; \forall k \neq i$ gilt, so ist die Spezies S_i Produkt (Reaktant) der j-ten Reaktion, deren Reaktant (Produkt) der Nullkomplex ist. Das Reaktionsnetzwerk beschreibt somit ein offenes System, dem die Spezies S_i permanent zugeführt (entzogen) wird und daher externer INPUT (OUTPUT) genannt werden soll.

2. Gibt es für einen Index $i \leq n$ mit $v_{ij} \leq 0$ ($v_{ij} \geq 0$) $\forall j = 1(1)r$, so tritt die Spezie S_i im Reaktionsnetzwerk nur als Reaktant (Produkt) von Reaktionen auf, sie wird also nicht aufgebaut (abgebaut) und soll daher als interner INPUT (OUTPUT) bezeichnet werden.

Die INPUT–/OUTPUT–Eigenschaften nicht reversibler Reaktionsnetzwerke lassen sich zur Charakterisierung des dynamischen Verhaltens der entsprechenden chemischen Reaktionssysteme und auch zu ihrer Strukturierung heranziehen. So gelten für das Zeitverhalten der Konzentrationen folgende Monotonieaussagen:

- Die Konzentrationen interner OUTPUT–Spezies sind monoton nicht fallend und ohne Einfluß auf die Konzentration anderer Spezies.
- Die Konzentrationen interner INPUT–Spezies sind monoton nicht wachsend.
- Für ein Reaktionssystem ohne externen INPUT ist die Konzentration externer OUTPUT–Spezies ab einem endlichen Zeitpunkt t^* monoton fallend und das System nimmt seinen Gleichgewichtszustand frühestens dann ein, wenn die Konzentration aller OUTPUT–Spezies verschwindet.

Die INPUT–/OUTPUT–Eigenschaften lassen sich, wie alle topologischen Eigenschaften chemischer Reaktionssysteme, algorithmisch vollständig aus der stöchiometrischen Matrix gewinnen. Die stöchiometrische Matrix ermöglicht eine globale qualitative Charakterisierung des dynamischen Verhaltens dieser Systeme. Diese Informationen allein sind für eine Strukturierung des Systems (2.2) durch Zerlegung in schwach wechselwirkende Teilsysteme jedoch nicht ausreichend. Wenn hierfür keine apriori–Kenntnisse vorliegen, so kommt man nicht umhin, eine quantitative Bewertung der Kopplung zwischen den einzelnen Spezies vorzunehmen. Macht man eine solche Bewertung nicht nur von den konstanten Ratenkoeffizienten, sondern auch von den konkreten Konzentrationen der Spezies abhängig, so haben die daraus gewonnenen Aussagen nur noch lokale Gültigkeit. Bei der praktischen Realisierung ist man dann darauf angewiesen, daß die gewonnenen Bewertungen etwa durch Beschränktheits– und Monotonieeigenschaften für größere Zeitintervalle sinnvoll sind.
Als geeigneter Gegenstand für die numerische Bewertung der Kopplung zwischen einzelnen Spezies bietet sich die Jacobimatrix des durch Zeitdiskretisierung aus dem Differentialgleichungssystem (2.2) gewonnenen nichtlinearen Systems

$$F(x_k) = 0 \tag{3.9}$$

an. Dabei bezeichnet x_k die letzte für den aktuellen Zeitpunkt $t = t_k$ ermittelte Näherung für den Vektor der Konzentrationen $c(t_k)$, und die Jacobimatrix

$$A := \frac{\partial F(c)}{\partial c} \mid_{c=x_k}, \qquad A = (a_{ij}) : n \times n, a_{ij} \in \mathbb{R}^n \tag{3.10}$$

beschreibt in ihrer Topologie welche Spezies miteinander gekoppelt sind und enthält in den numerischen Werten ihrer Elemente eine lokal für den Zeitpunkt $t = t_k$ gültige quantitative Wertung dieser Wechselwirkung. Für Beispiel (2.4) hat die Jacobinmatrix A nach der Vertauschung der Zeilen und Spalten 2 und 6 die Struktur

$$\nu = \begin{bmatrix} \bullet & * & * & & * & & \\ * & \bullet & & * & & & \\ * & * & \bullet & * & & & \\ * & * & * & \bullet & & & \\ * & * & * & * & \bullet & & \\ * & & & & * & \bullet & \\ * & * & & & & & \bullet \end{bmatrix} \begin{matrix} : H_2 \\ : O_2 \\ : O^* \\ : H^* \\ : OH^* \\ : H_2O \\ : HO_2 \end{matrix} \tag{3.11}$$

Dabei erkennt man noch, daß die Spezies H_2O und HO_2 interne OUTPUT–Größen sind. Einige andere topologischen Informationen der stöchiometrischen Matrix sind dagegen an der Jacobimatrix nicht mehr ablesbar.
Die Jacobimatrizen großer chemischer Reaktionssysteme lassen sich allgemein wie folgt charakterisieren. Sie sind

1. unregelmäßig sparsam besetzt,
2. in der Regel nicht symmetrisch und
3. im allgem. nicht diagonaldominant.

Darüber hinaus gibt es Reaktionssysteme, bei denen einige weinige Spezies an relativ vielen Reaktionen beteiligt sind, so daß es Zeilen und Spalten in A gibt, die überdurchschnittlich stark besetzt sind.
Die Jacobimatrixen reversibler Netzwerke sind in der Regel nur "struktursymmetrisch", d.h. aus $a_{ij} \neq 0$ folgt $a_{ji} \neq 0$, es ist aber im allgemeinen $a_{ij} \neq a_{ji}$. Wegen der speziellen autonomen Gestalt von (2.2) enthalten die Diagonalelemente a_{ij} von A jeweils einen Summanden mit dem Faktor $\frac{1}{h}$, wobei $h = t_k - t_{k-1}$ die aktuelle Zeitschrittweite bezeichnet. Da mit den Ratenkoeffizienten und den Konzentrationen auch die Jacobmatrixelemente in ihrem Betrag nach oben beschränkt sind, kann man somit theoretisch für $h \longrightarrow 0$ die Diagonaldominanz von A erzwingen. Praktisch ist dies jedoch in den meisten Fällen sinnlos, da man hierfür h viel kleiner als die minimale Schrittweite des Integrationsverfahrens wählen müßte.
Experimente mit dem Programmsystem LARKIN haben gezeigt, daß für Reaktionssysteme mit Defekt $\delta = 0$ sowohl die Konzentrationen c_i der Spezies S_i, als auch die Jacobimatrixelemente a_{ij} weitgehend ein monotones Verhalten zeigen. In vielen Zeilen von A bleiben die selben Elemente über das gesamte Zeitintervall dominant, in anderen finden in der Regel nur weinige Wechsel statt, bei denen ein bislang dominiertes Matrixelement selbst dominant wird. Für solche Beispiele kann man erwarten, daß die lokal vorgenommenen Bewertungen der Wechselwirkungen zwischen den Spezies für hinreichend große Zeitintervalle ihren Sinn behalten.
Ein Problem, das sich bei der quantitiven Bewertung der Kopplung zwischen Spezies bzw. Gruppen von Spezies auf jeden Fall stellt, ist das Problem der Skalierung. Für chemische Reaktionssysteme, deren Ratenkonstanten von erheblich verschiedener Größenordnung sind (Verhältnisse zwischen den Ratenkonstanten einzelner Reaktionen des selben Systems von $1 : 10^p$ mit $p > 10$ sind keine Seltenheit), variieren auch die Jacobimatrixelemente in Bereichen sehr unterschiedlicher Größenordnung. Somit ist eine Bewertung der Kopplung über einen absoluten Vergleich der Jacobimatrixelemente wenig sinnvoll. Vielmehr muß eine Skalierung sowohl innerhalb der Zeilen der Jacobimatrix als auch hinsichtlich der Größenordnungen, in denen sich die Konzentrationen der einzelnen Spezies bewegen, erfolgen.

4 Algorithmen zur Strukturierung

Im weiteren wollen wir unsere Betrachtungen auf Reaktionssysteme beschränken, deren Netzwerk den Defekt $\delta = 0$ hat. Dabei beziehen wir nicht (schwach) reversible Netzwerke ebenso ein, wie offene Systeme.
Ausgehend von den beschreibenen topologischen und numerischen Eigenschaften solcher chemischer Reaktionssysteme wird von uns folgende Strategie zur Zerlegung des Differentialgleichungssystems (2.2) in Teilsysteme (disjunkte Blöcke von Gleichungen zwischen dennen möglichst nur eine "schwache" Wechselwirkung besteht) vorgeschlagen.

1. Durch Auswertung der stöchiometrischen Matrix erfolgt eine Kennzeichnung der INPUT– und OUTPUT–Variablen, und es wird eine Separierung von Gleichungen für interne OUTPUT–Variablen vorgenommen.

2. Ausgehend von der Jacobimatrix (3.10) wird ein parametrisierter gerichteter / ungerichteter Graph mit Knoten, die den Spezies S_i, $i = 1(1)n$, sowie Zweigen, die den Jacobimatrixelementen a_{ij}, $i \neq j$, entsprechend erzeugt, wobei eine numerische Bewertung für Knoten und Zweige erfolgt.

3. Zerlegung des Graphen entsprechend der vorgenommen Bewertungen in Teilgraphen.

4. Zerlegung größerer Teilgraphen durch

   4.1. Erzeugung und Zerlegung einer Levelstruktur des Teilgraphen

   4.2. Separierung von Knoten mit maximaler Anzahl abgehender Zweige und Zerlegung des verbleibenden Rests des Teilgraphens gemäß 3.

Der 4. Schritt läßt sich rekursiv auf Teilgraphen anwenden. Wenn Schritt 3. keine oder nur eine unzureichende Zerlegung liefert, so wird sukzessive eine Zerlegung durch 4. bestimmt.
Nachfolgend wollen wir näher auf Einzelheiten des Zerlegungsalgorithmus eingehen, wobei der Fall betrachtet werden soll, daß aus der Jacobimatrix A des Systems (3.9) ein ungerichteter parametrisierter Graph

$$G(V, inf(V), E, inf(E)) \tag{4.1}$$

erzeugt wird. Die Knotenmenge $V = \{v_i\}$, $i = 1(1)n$, entspricht den Zeilen (bzw. Spalten) der Jacobimatrix und somit repräsentiert jeder Knoten v_i eine Spezies S_i des Reaktionssystems. Die Zweigmenge $E = \{e_l\} \in V \times V$, $l = 1(1)m$, entspricht den nicht verschwindenden Jacobimatrixelementen a_{ij}, $i \neq j$, in der Weise, daß es genau einen Zweig e_l gibt, der die Knoten v_i und v_j verbindet, wenn $a_{ij} \neq 0$, oder $a_{ij} \neq 0$, gilt. Damit repräsentiert ein Zweig e_l die Wechselwirkung zwischen den Konzentrationen der Spezies S_i und S_j.
Eine Parametrisierung (Wichtung) der Knoten– und Zweigmenge wird über $\inf(\nu)$ und $\inf(E)$ wie folgt vorgenommen:
Den Knoten v_i wird

$$inf(v_i) = \sum_{j=1}^{n} \mid a_{ij} x_k(j) \mid / p_1(i) \tag{4.2}$$

und den Zweigen $e_l = (v_i, v_j)$

$$inf(e_l) = \max(\mid a_{ij}x_k(j) \mid, \mid a_{ji}x_k(i) \mid) \tag{4.3}$$

zugeordnet. Dabei bezeichnet p_1 in (4.2) einen Parameter des Partitionierungsalgorithmus, der im Standardfall mit

$$p_1(i) = |a_{ii}|, \quad i = 1(1)n, \tag{4.4}$$

belegt wird.
Man beachte, daß in (4.2), (4.3) nicht nur die Approximationen $x_k(i) \sim c_i(t_k)$, sondern auch die Jacobimatrixelemente a_{ij} vom Zeitpunkt t_k abhängen. Die Wichtung der Knoten und Zweige des Graphen (4.1) ist also zeitabhängig.
Im folgenden wollen wir auf die Techniken eingehen, die unabhängig von dieser Wichtung, allein basierend auf der Topologie des Graphen, seine Partitionierung ermöglichen. Dazu führen wir in Analogie zu [20] über dem Graphen G, bzw. einem Teilgraphen, eine sogenannte Levelstruktur ein. Dabei wird jedem Knoten v_i in der Weise eine Levelnummer $l_i \in \{0, 1, 2, \ldots\}$ zugeordnet, so daß – wenn man jeweils die Knoten mit gleicher Levelnummer j zu einem Level L_j zusammenfaßt – gilt, daß von den Knoten des Levels L_j nur Zweige zu den Knoten benachbarter Level L_{j-1} und L_{j+1} sowie zu den Knoten von L_j selbst führen.
Über jedem Graphen lassen sich eine Vielzahl von Levelstrukturen bilden, die diese Bedingungen erfüllen. In diesem Zusammenhang ist es unser Ziel, eine Levelstruktur mit möglichst vielen Leveln zu finden. Wir betrachten aus diesem Grund nur Levelstrukturen, die in Level L_0 nur 1 Knoten enthalten. Für einen Graphen mit n Knoten gibt es genau n verschiedene solcher Strukturen.
Über Knotenmengen $\hat{V} \subset V$ und Zweigmengen $\hat{E} \subset E$ des Graphen G betrachten wir für fixierte "Startknoten" $v_s \in \hat{V}$ Levelstrukturen

$$\mathcal{L}(v_s) = \{L_0(v_s), L_1(v_s), ..., L_{l(v_s)}\}, \tag{4.5}$$

die man, bezeichnet man mit $Adj(K)$ die Menge aller benachbarten Knoten der Knotenmenge $K \subset \hat{V}$, algorithmisch nach der Vorschrift

$$\begin{aligned} L_0(v_s) &= \{v_s\} \\ L_1(v_s) &= Adj(L_0(v_s)) \\ &\vdots \\ L_i(v_s) &= Adj(L_{i-1}(v_s)) - L_{i-2}(v_s)) \quad , \end{aligned} \tag{4.6}$$

die man fortsetzt bis $L_{l(v_s)+1} = \emptyset$ ist, bestimmen.
Um einen Startknoten v_s zu finden, dessen Levelstruktur $\mathcal{L}(v_s)$ möglichst viele Level L_i enthält, verwenden wir den pseudo–optimalen Algorithmus:

(1) wähle $v_s^* = v_i \in \hat{V}$ beliebig
(2) bilde $\mathcal{L}(v_s^*)$
(3) wähle $v_s \in L_{l(v_s^*)}$ mit $degree(v_s)$ minimal in $L_{l(v_s^*)}$
(4) bilde $\mathcal{L}(v_s)$
(5) wenn $l(v_s) > l(v_s^*)$ so setze $v_s^* = v_s$ und gehe zu (3)
(6) setze $v_s^* = v_s$ und stopp. (4.7)

Dabei bezeichnet $degree(v)$ die Anzahl der Zweige aus $\hat{E}$ die zum Knoten v führen.
Die Zerlegung einer Levelstruktur läuft prinzipiell nach folgendem Schema ab:

(1) bestimme einen pseudo–optimalen Startknoten $v_s^* \in \hat{V}$
(2) bilde $\mathcal{L}(v_s^*) = \{L_0, L_1, ..., L_l\}$
(3) setze die Menge der „separierenden" Knoten $K = \emptyset$ $i = 0$
und die Anzahl der Blöcke $n_B = 1$
(4) wähle ein Separator-Level L_j mit $i < j < l$
(5) ergänze $K = K \cup \{v \in L_j \mid Adj(v) \cap L_{j+1} \neq \emptyset\}$
setze $i = j + 1$ $n_B = n_B + 1$ und gehe zu (4) (4.8)

Der Algorithmus bricht ab, wenn $i > l - 2$ ist, beziehungsweise in (4) kein Separator–Level mehr gefunden werden kann. Eine solche Zerlegung der Levelstruktur entspricht der Strukturierung der zugrundeliegenden Teilmatrix in eine geränderte Blockdiagonalgestalt.
Wir können nun die Strategie für den Algorithmus zu Strukturierung des Differentialgleichungssystems (2.2) zum Zeitpunkt $t = t_k$ notieren.

(1) generiere den Graphen $G = G(V, inf(V), E, inf(E))$ aus der Jacobimatrix A aus (3.10) und bewerte die Knoten und Zweige gemäß (4.2), (4.3)

(2) separiere alle Knoten v_i deren Spezies S_i nur als Produkt von Reaktionen auftreten, in der Knotenmenge $\bar{K}$ und reduziere die Knotenmenge von G auf $\bar{V} = V - \bar{K}$ und die Zweigmenge auf $\bar{E} = \{e_k = (v_i, v_j) \in E \mid v_i \notin \bar{K} \text{ und } v_j \notin \bar{K}\}$

(3) bestimme den stark gekoppelten Teilgraphen $G^*(\bar{V}, \bar{E}^*)$ von $G(\bar{V}, \bar{E})$ mit $\bar{E}^* \subset \bar{E}$ gemäß $\bar{E}^* = \{e_k = (v_i, v_j) \in \bar{E} \mid inf(e_k) > min(inf(v_i), inf(v_j))/p_2\}$

(4) bestimme die zusammenhängenden Komponenten $G_i(V_i, E_i)$ von $G^*(\bar{V}, \bar{E}^*)$ so, daß für $i \neq j$ sowohl $V_i \cap V_j = \emptyset$, als auch $E_i \cap E_j = \emptyset$ gilt und es zu je 2 Knoten aus V_i stets eine Kette von Zweigen aus E_i gibt, die diese Knoten miteinander verbindet.

(5) für alle Teilgraphen G_i :

(5-0) setze $\hat{V}_i = V_i$, $\hat{E}_i = E_i$ und $\hat{K}_i = \emptyset$

(5-1) finde pseudo–optimalen Startknoten v_s in $\hat{V}_i$ gemäß (4)

(5-2) generiere Levelstruktur $\mathcal{L}_i$ gemäß (4) über $G_i(\hat{V}_i, \hat{E}_i)$

(5-3) Zerlege die Levelstruktur $\mathcal{L}_i$ durch Separierung der Knotenmenge $K_i \subset \hat{V}_i$ gemäß (4)

(5-4) bewerte die durch $K_i + \hat{K}_i$ über $\hat{V}_i$ beschriebene Blockstruktur mit dem Wert $w_i \in \{-1, 0, +1\}$

(5-5) falls $w_i > 0$: bilde $W = \{v \in \hat{V}_i \mid degree(v)$ maximal in $\hat{V}_i\}$, $\hat{V}_i = \hat{V}_i - W$, $\hat{K}_i = \hat{K}_i + W$ und gehe zu (5-1)

(6) beschreibe Gesamtstruktur und stopp.

Über den Parameter p_2 in (3) läßt sich der Algorithmus hinsichtlich der Bewertung, was unter einer „starken“ Kopplung verstanden werden soll, steuern. Die Standardbelegung ist $p_2 = 10$.
Um diesen Algorithmus wohldefiniert anwenden zu können, müssen noch zwei Teilschritte näher spezifiziert werden. Es ist dies zum einen die Wahl des nächsten Separatorlevels in Schritt (4) von (4) und zum anderen die Bestimmung der Bewertung w_i der gewonnenen Blockstruktur in Schritt (5-4).
Zu diesem Zweck führen wir mit der Bezeichnung $card(M)$ für die Anzahl der Elemente der Menge M folgende Größen zur Beschreibung einer Block-Zerlegung der Levelstruktur $\mathcal{L} = \{L_0, L_1, L_2, ..., L_l\}$ eines Teilgraphen $\hat{G}(\hat{V}, \hat{E})$ ein:

$$
\begin{aligned}
n_0(j) &= card\{v \in L_j \mid Adj(v) \cap L_{j+1} \neq \emptyset\} \\
n_1(j) &= card\{e = (v, w) \in \hat{E} \mid v \in L_j \quad \text{und} w \in Lj + 1\} \\
n_2(j) &= card\{v \in L_k \ , \ k = i(1)j - 1\} \\
n_3(j) &= card\{v \in L_k \ , \ k = j + 1(1)l\} \\
N &= card\{v \in L_k \ , \ k = 0(1)l\} = card\{\hat{V}\} \\
N_0 &= card\{K\} + card\{\hat{K}\}
\end{aligned}
$$

Dabei ist n_0 die Anzahl der Knoten von L_j die über Zeige aus $\hat{E}$ mit den Knoten des nächsten Levels L_{j+1} verbunden sind und n_1 ist die Anzahl eben dieser Zweige. Die Größe n_2 ist die Anzahl der Knoten, die in den Leveln zwischen den beiden letzten Separatorleveln liegen und als ein Block zusammengefaßt werden sollen. Während n_4 die Anzahl der Knoten in den bislang noch nicht behandelten Leveln bezeichnet, ist N die Anzahl aller Knoten der Levelstruktur $\mathcal{L}$.
Für die Wahl des nächsten Separatorlevels L_j in Schritt (4) von (4) stellen wir folgende Randbedingungen:

1. $\frac{n_2}{n_0} \geq p_3$
2. $n_2 \leq \frac{N}{p_4}$
3. $n_3 > \frac{n_0}{2} + 1$

Dabei haben die Parameter p_1 und p_2 des Algorithmus die Standardbelegungen $p_3 = p_4 = 2$. Bedingung 1 zielt darauf, mit möglichst wenigen Knoten in L_j möglichst viele Knoten zu separieren, während Bedingung 2 als Gegenläufer dazu verhindern soll, daß zu große Blöcke gebildet werden. Die Bedingung 3 stellt ein Abbruchkriterium für den Algorithmus (3.7) zur Wahl der Separatormenge K dar.
Für die Bestimmung der Bewertung w_i der gebildeten Blockstruktur setzen wir folgende Kriterien an:

1. $w_i = -1$: wenn keine weitere Zerlegung sinnvoll ist, da:
   (a) $N < p_5$ oder
   (b) $N_0 < \frac{N}{p_6}$
2. $w_i = 0$: wenn die Blockstruktur zufriedenstellend ist, da:
   (c) $n_B > \frac{N}{p_7}$
3. $w_i = +1$: sonst.

Bedingung (a) ist eine untere Schranke für die Größe der zu behandelnden Levelstruktur. Bedingung (b) beschränkt die Gesamtanzahl der Knoten in der Separatormenge und Bedingung (c) verhindert eine zu feine Strukturierung. Für $w_i = +1$ wird durch Erweiterung der gesetzten Separatormenge $\hat{K}_i$ versucht, eine feinere Blockstruktur zu bestimmen.
Als Standardwerte für die Parameter werden $p_5 = 10$, $p_6 = 4$, $p_7 = 5$ benutz. Es ist klar, daß einige der Parameter p_i sehr stark von der Anzahl N der Knoten in der Levelstruktur abhängen und für entsprechende Beispiele angepaßt werden müssen. Die vorgestellte Strategie zur Strukturierung des Differentialgleichungssystems (2.2) wurde mit Hilfe der Programmbibliothek LEDA [21] in C++ programmiert. Von LEDA werden u.a. Datentypen und Basisalgorithmen bereitgestellt mit denen man in einfacher Weise Operationen über Graphen beschreiben kann. Die Algorithmen wurden von uns an Beispielen getestet, die uns zusammen mit dem Programmsystem LARKIN [8] zur Verfügung standen.

Im folgenden wollen wir für ein überschaubares Beispiel einige mögliche Zerlegungen angeben. Wir verwenden hierfür einen Glycose–Reaktionsprozeß der Biochemie. Das geschlossene Reaktionssystem läßt sich durch ein nicht schwach reversibles Reaktionsnetzwerk von 89 Reaktionen über 65 Spezies beschreiben. Die zugehörige Jacobimatrix hat die in Abb. 1 angegebene Struktur.
Die Struktur der Matrix in Abb. 2 ist Ergebnis des Partitionierungsalgorithmus für den Standardfall ($p_2 = 10$).
Für eine „abgeschwächte" Kopplungsbedingung ergibt sich die Struktur aus Abb. 3. Diese Partitionierung basiert auf einer relativ feinen Levelzerlegung (Abb. 4).
Setzt man auf diese Zerlegung eine Variante des Partitionierungsalgorithmus an, in der einzelne Gleichungen mit relativ dicht besetzter Zeile in der Jacobimatrix vom Gesamtsystem separiert werden, so gelangt man zu einer geränderten Blockdiagonalgestalt wie in Abb. 5.

Dabei kennzeichnet:

- • ein Matrixelement, das (im jeweiligen Zusammenhang) einer "starken" Kopplung und
- ∘ komplementär dazu ein Matrixelement, das einer "schwachen" Kopplung entspricht.

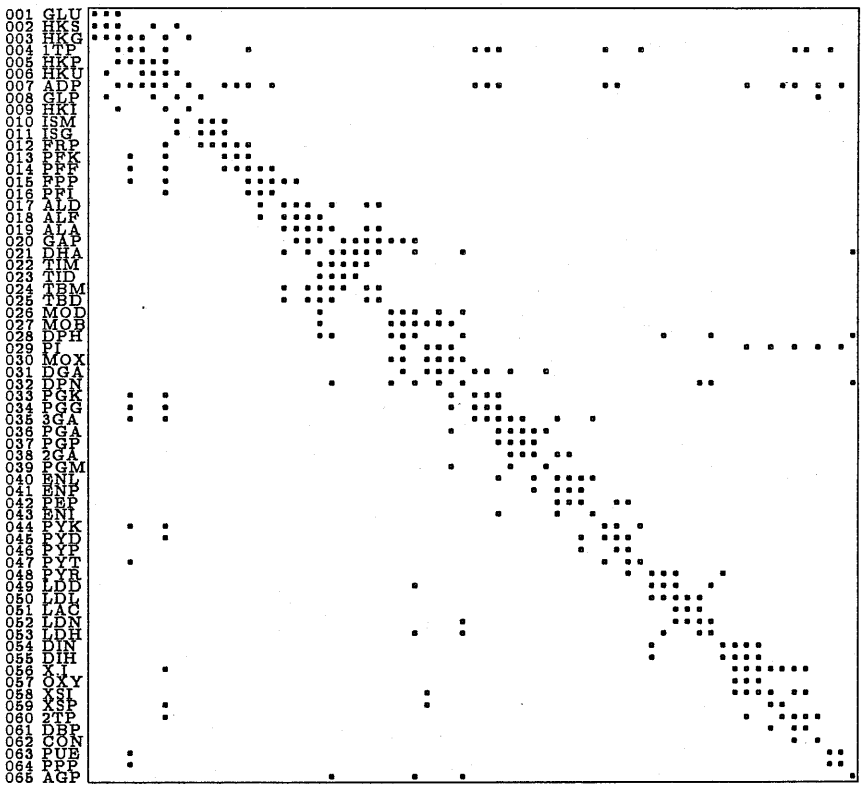

Abbildung 1: GLYCO1 : Struktur der Jacobimatrix

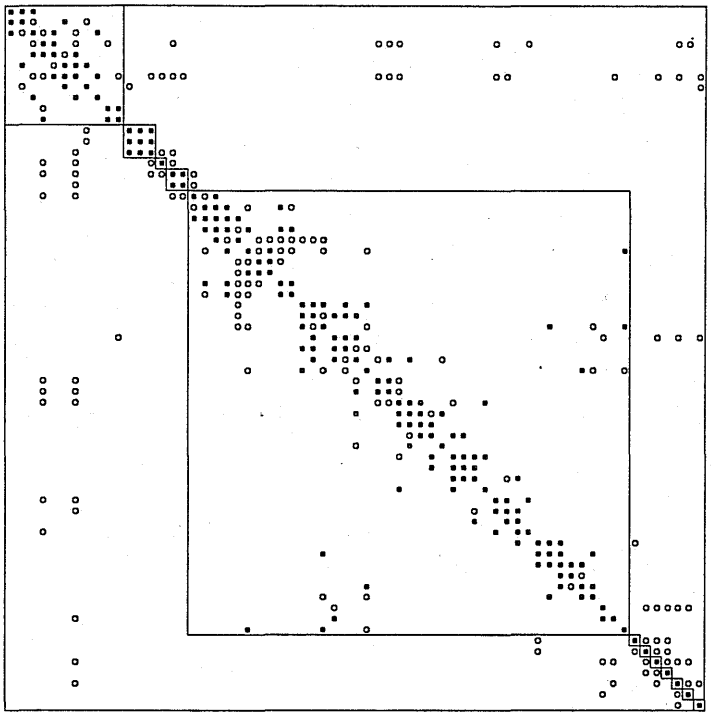

Abbildung 2: GLYCO1 : Zerlegung in "stark" zusammenhängende Komponenten, $p_2 = 10$

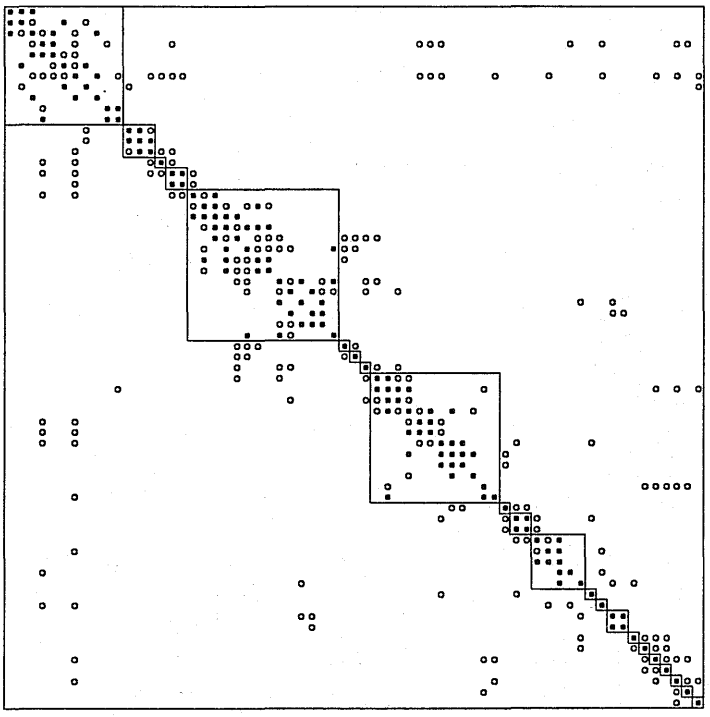

Abbildung 3: GLYCO1 : Zerlegung in "stark" zusammenhängende Komponenten, $p_2 = 2.5$

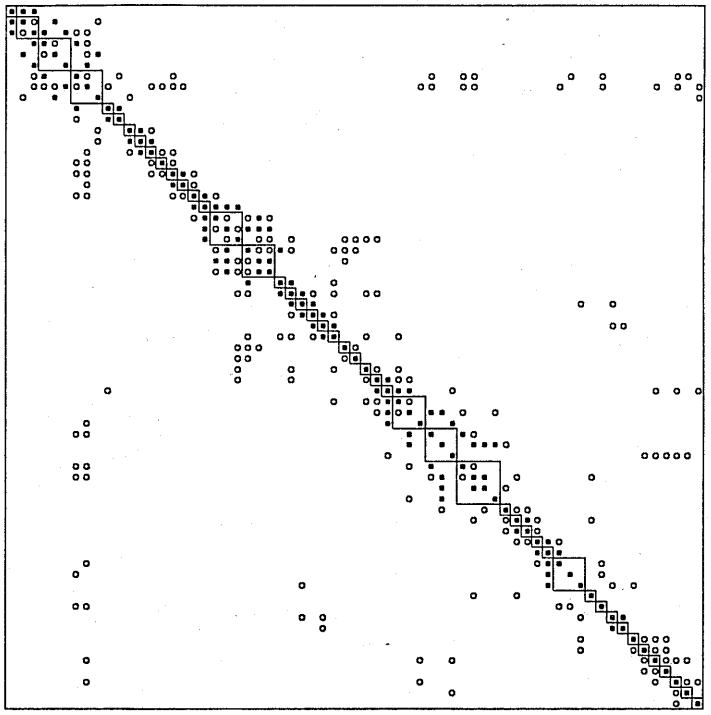

Abbildung 4: GLYCO1 : Levelstruktur, $p_2 = 2.5$

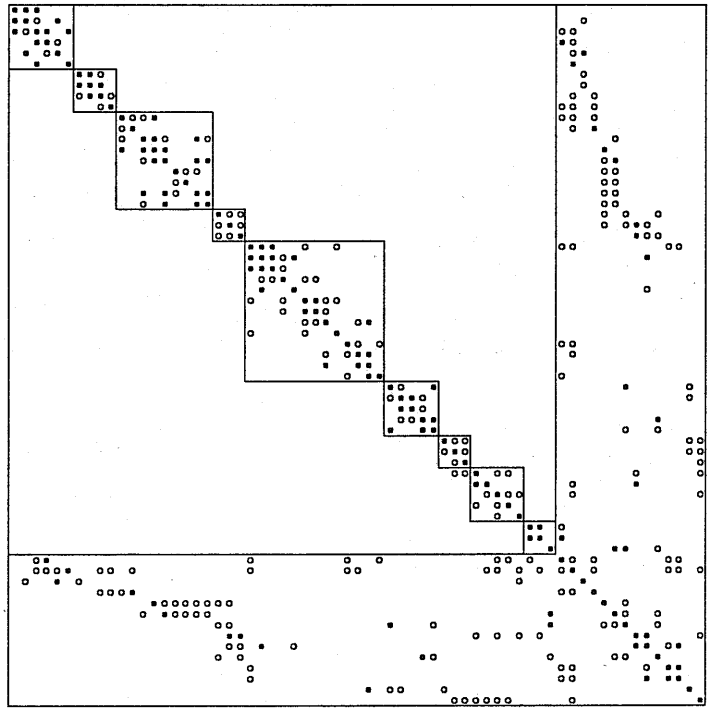

Abbildung 5: GLYCO1 : Zerlegung in eine geränderte Blockstruktur

5 Das Waveform-Iterationsverfahren

Wir nehmen an, daß das zu behandelnde Gleichungssytem (2.2) in einer Blockstruktur vorliegt, wie sie durch den obigen Partitionierungsalgorithmus geliefert wird.
Sei also $\mathbb{R}^n = \mathbb{R}^{n_1} \times \mathbb{R}^{n_2} \times \cdots \times \mathbb{R}^{n_r}$, $x = (x_1, \ldots, x_r)$, $y = (y_1, \ldots, y_r)$ und $g = (g_1, \ldots, g_r)^T$ mit $x_i, y_i \in \mathbb{R}^{n_i}$ und $g_i : \mathbb{R}^n \times [t_a, t_z] \to \mathbb{R}^{n_i}$, dann erhalten wir ein Gleichungssystem der Form

$$\left.\begin{array}{rcl} \dot{x}_i & = & g_i(x_1, \ldots, x_r, t) \\ x_i(t_a) & = & x_{i,0} \end{array}\right\}, i = 1, \ldots, r, \tag{5.1}$$

wobei x_i der Vektor ist, der aus den Komponenten des i-ten Blocks (Teilsystems) gebildet wird.
Wir führen die Bezeichnung $f_i(x, y, t) := g_i(x_1, \ldots, x_i, y_{i+1}, \ldots, y_n, t)$ ein.
Unter Verwendung des Gauß-Seidel-Ansatzes lautet der allgemeine Ansatz für die Waveform-Iteration für (5.1) dann

$$\begin{array}{rcl} \dot{x}^k & = & f(x^k, x^{k-1}, t), \\ x(t_a) & = & x_0. \end{array} \tag{5.2}$$

Die Konvergenzeigenschaften des Verfahrens lassen sich unter der Verwendung von exponentiell gewichteten Normen herleiten [5, 25]. Beschränkt man sich auf das implizite Eulerverfahren bzw. auf die Trapezregel, dann liegen ähnliche Resultate auch für Diskretisierungen mit unterschiedlichen Schrittweiten in jedem Teilsystem und jeder Iteration vor. Man erhält gleichzeitig obere Schranken für die Schrittweite, bei deren Einhaltung die Konvergenz des Verfahrens noch gesichert ist. Die Schranke für die Schrittweite hängt von der exponentiellen Wichtung der Norm und von der Konvergenzrate ab, diese wiederum lassen sich mit Hilfe der Lipschitzkonstanten von f und dem Zeitintervall $[t_a, t_z]$ angeben.
Die so ermittelte Schranke für die Konvergenzrate ist um so kleiner, je kürzer das betrachtete Zeitintervall ist. Deshalb ist es günstiger, wenn die Simulation in sogenannten Zeitfenstern bzw. Zeitsegmenten erfolgt.
Zu jeder Komponente i des Gleichungssystems (5.1) seien zwei Folgen $(\bar{t}_i^k)_{k=0}^{\infty}$ und $(\bar{T}_i^k)_{k=0}^{\infty}$ mit $t_a \leq \bar{t}_i^k < \bar{T}_i^k \leq t_z$ $(\forall i, \forall k > 0)$ und $\bar{t}_i^0 = \bar{T}_i^0 = t_a$ $(\forall i)$ gegeben. Das Intervall $[\bar{t}_i^k, \bar{T}_i^k] \subseteq [t_a, t_z]$ bildet dann das Zeitfenster für die Komponente i in der Iteration k, über dem die aktuelle Simulation erfolgt.
Mit $\Xi_i^k(z, \cdot) : [\bar{T}_i^k, T] \to \mathbb{R}^{n_i}$ sei eine stetig differenzierbare Funktion bezeichnet, für die für alle $z \in \mathbb{R}^{n_i}$ $\Xi_i^k(z, \bar{T}_i^k) = z$ ist. Für eine in $[t_a, \bar{T}_i^k]$ definierte Funktion $z(t)$ ist $\Xi_i^k(z(\bar{T}_i^k), t)$ eine Extrapolation im Intervall $[\bar{T}_i^k, t_z]$.
Wir definieren mit $(x^k)_{k=0}^{\infty}$, $x^k = (x_1^k, \ldots, x_r^k)$ eine neue Folge von Funktionen (Waveform-Iterierten) über $\bar{J} := [t_a, t_z]$:

$$x_i^0(t) := \Xi_i^0(x_{i,0}, t),$$

$$x_i^k(t) = \begin{cases} x_i^{k-1}(t), & t \le \bar{t}_i^k, \\ x_i^{k-1}(\bar{t}_i^k) + \int_{\bar{t}_i^k}^t f_i(x^k(s), x^{k-1}(s), s)ds, & \bar{t}_i^k < t \le \bar{T}_i^k, \\ \Xi_i^k(x_i^k, t), & \bar{T}_i^k < t. \end{cases} \tag{5.3}$$

Im folgenden wollen wir ein ein Verfahren zur Wahl der Zeitfenster $[\bar{t}_i^k, \bar{T}_i^k]$ angegeben, das in [25] ausführlich beschrieben wird. Wir bezeichnen dafür diejenigen Funktionen x_j^k, die explizit Eingang in die rechte Seite von (5.3) zur Bestimmung von x_i^k finden, als Input der i-ten Gleichungskomponente. Die Menge der Inputkomponenten zur i-ten Komponente des Systems (5.1) ist dementsprechend die Menge derjenigen Komponenten, die auf der rechten Seite der Gleichung für die i-te Komponente nicht konstant verschwinden. Diese Menge sei mit $\mathcal{N}_{\text{in}}(i)$ abgekürzt.
Mit $k_i(t,k)$ sei diejenige Iteration innerhalb der ersten k Iterationen bezeichnet, die den aktuellen Wert von $x_i^k(t)$ durch Berechnung oder Extrapolation liefert, d.h. es ist

$$x_i^k = x_i^{k_i(t,k)}.$$

Definiert man

$$k_{i,j}(k) := \begin{cases} k, & \text{für } j < i, \\ k-1, & \text{für } j \ge i, \end{cases}$$

so stammen die Werte, die für die Komponente x_j auf der rechten Seite der Gleichung (5.3) für x_i^k zum Zeitpunkt t zur Verfügung stehen, für alle $j \in \mathcal{N}_{\text{in}}(i)$ aus der Iteration $k_{i,j}(k_i(t,k))$.
Zu zwei gegebenen Abbruchschranken $\epsilon_1 > 0$ und $\epsilon_2 > 0$ und einem Parameter $\tilde{h} > 0$ definieren wir nun die Zeitfenster für Iteration (5.3) wie folgt:
Für $k = 0, 1$ sei $\bar{t}_i^0 = \bar{t}_i^1 = \bar{T}_i^0 := t_a$.
Im übrigen gelte

$$\begin{aligned} \bar{t}_i^k := \sup \Big(& t \in [t_a, \bar{t}_{\min}^{k-1}] \cup [\min(\bar{t}_{\min}^{k-1} + \tilde{h}, t_z), t_z] : \\ & |x_j^{k_{i,j}(k)}(t) - x_j^{k_{i,j}(k_i(t,k-1))}(t)| \le \epsilon_1, \\ & \forall j \in \mathcal{N}_{\text{in}}(i) \Big), \end{aligned} \tag{5.4}$$

$$\bar{T}_i^k := \sup \left(t \in [\bar{t}_i^k, \bar{t}_i^k + w_{\max}^k] : |\bar{x}_i^k(t) - x_i^{k-1}(t)| \le \epsilon_2 \right),$$

wobei $\bar{x}_i^k$ die Lösung von

$$\begin{aligned} \dot{y} &= g_i(x_1^k, \ldots, x_{i-1}^k, y, x_{i+1}^{k-1}, \ldots, x_n^{k-1}) \\ y(\bar{t}_i^k) &= x_i^{k-1}(\bar{t}_i^k) \end{aligned}$$

im Intervall $(\bar{t}_i^k, t_z]$ sei und $\bar{t}_{\min}^k := \min_i(\bar{t}_i^k)$, $w_{\max}^k := c_w w_{\min}^k$ und $w_{\min}^k := \min_i(\bar{T}_i^{k-1} - \bar{t}_i^{k-1})$ bezeichnen. Der Faktor c_w ist die Größe, um die die Fenster von Iteration zu Iteration höchstens vergrößert werden dürfen. Der Wert $c_w = 3$ hat sich als eine gute Wahl erwiesen.

Da bei diesem Vorgehen nicht immer wieder vom Intervallanfang an gerechnet wird, tritt ein Fehler durch vorzeitigen Iterationsabbruch auf, der in [25] abgeschätzt wurde.
Für die durch (5.4) definierten Fenster sei zusätzlich $\bar{t}_i^k \leq \bar{T}_i^{k-1}$ erfüllt. Dann gilt für die C- bzw. C^1-Norm der Abweichung der letzten Iterierten von der Lösung des AWP's (5.1):

$$|x^k - x^*|_\iota \leq K_\iota \exp\left(2\frac{h_{\max}}{h_{\min}}\alpha(t_z - t_a)\right)\epsilon_1,\ \iota = 0, 1,$$

wobei K_ι jeweils eine Konstante und $h_{\min}/h_{\max}$ die minimale/maximale Fensterlänge ist. Sortiert man die Fensterenden der Größe nach und bezeichnet diese mit $s_0, s_1, s_2 \ldots$ dann erhalt man für die Konstanten

$$K_\iota = \frac{1}{2}\max_j\left(1 + (\iota+1)\frac{\alpha}{1 - e^{-\alpha(s_j - s_{j-1})}}\right)e^{\alpha(s_j - s_{j-1})}.$$

α ist der Parameter für die exponentielle Wichtung der Norm im zugrundeliegenden Banachraum. Da α in Abhängigkeit von der Lipschitzkonstanten von g zu bestimmen ist, kommt man bei steifen Systemen auf sehr große Werte von α. In diesem Fall ist die obige Abschätzung nur von theoretischer Bedeutung. Im realisierten numerischen Verfahren wird für ϵ_1 ein heuristischer Wert angenommen, der $0.1 * \epsilon/\tilde{\kappa}$ beträgt. Wobei $\tilde{\kappa}$ das Maximum aus geschätzter Konverenzrate und einem Hundertstel der theoretischen Konvergenzrate ist.

6 Reaktionssystem GLYCO1

Wir wollen im folgenden einige Probleme bei der Realisierung des Waveform-Iterationsverfahrens mit variablen Zeitfenstern an einem konkreten Beispiel diskutieren. Wir verwenden dazu das schon in Abschnitt 4 betrachtete Glykose-Reaktionssystem.
Bei den vorliegenden chemischen Systemen kann aus der Struktur der Jacobimatrix keine Zerlegung abgeleitet werden, die für alle Zeiten optimal an den Algorithmus angepaßt ist. Es muß eine Wichtung der Jacobimatrixelemente eingeführt werden, damit überhaupt eine Zerlegung vorgenommen werden kann. Dabei können auch Kopplungen aufgetrennt werden, die sich später als wesentlich erweisen. Daher steht die Frage, in welchem Maße durch das Auftrennen einer starken Kopplung der Aufwand und das Ergebnis der Waveform-Iteration beeinflußt werden. Der Algorithmus sollte so einstellbar sein, daß die iterative Auftrennung von starken Kopplungen nicht zu einer Verschlechterung der Qualität der Lösung sonder höchstens zu einer Erhöhung des Rechenzeitaufwandes führt.
Bei strenger Beachtung der theoretischen Ergebnisse für die Fehlerschranken ist zwar die Korrektheit des Ergebnisses gesichert, aber meistens der Aufwand wegen der starken Beschränkung der Schrittweite unvertretbar hoch. Deswegen sind im implementierten Algorithmus über verschiedene Heuristiken Abweichungen zur Theorie zugelassen [25]. Solange keine großen Anforderungen an die Genauigkeit gestellt werden und das Gleichungssystem „normale“ Lipschitzkonstanten aufweist, kommt man mit Erfahrungswerten bei den eingebauten Heuristiken aus. Anders ist das bei Systemen mit extrem unterschiedlichen Reaktionskonstanten, wie in dem hier verwendeten Beispiel GLYCO1. Um die Richtigkeit des Simulationsergebnisses sicherzustellen, wurde deshalb erst einmal mit dem „worst case“,

der trivialen Zerlegung in eindimensionale Blöcke gearbeitet. Dadurch lassen sich auch die Kriterien für schwache/starke Kopplungen überprüfen und so die Parameter für den Partitionierungsalgorithmus ableiten.

6.1 Numerische Konstanten für GLYCO1

Die Reaktionskonstanten beim Beispiel GLYCO1 liegen im Bereich von $3 \cdot 10^{-3}$ bis $1.62 \cdot 10^{10}$. Das führt bei entsprechenden Konzentrationen der beteiligten Spezies auf Lipschitzkonstanten $L_x = 6.5 \cdot 10^6$ und $L_y = 6.52 \cdot 10^6$ beim Gauß-Jacobi-Ansatz. Da die Reaktionstherme, aus denen sich die Lipschitzkonstanten ableiten, in den wesentlichen Bestandteilen linear sind, sind diese Werte für das gesamte Zeitintervall gültig.
Verlangt man eine theoretische Konverganzrate von 0.5 unabhängig von der Länge des Zeitintervalls, so ist für die exponentielle Wichtung der Norm $\alpha = 1.954 \cdot 10^7$ zu setzen. Das Schrittweitenmaximum für die Trapezregel liegt dann bei $9.29 \cdot 10^{-8}$. Bei Einhaltung dieser Schranke hätte man im Intervall $[0, 4.0]$ mindestens $4.3 \cdot 10^7$ Schritte pro Komponente zu rechnen. Bei 2 – 3 Funktionsauswertungen pro Schritt kommt man auf $8.6 \cdot 10^7$ – $12.9 \cdot 10^7$ Funktionsauswertungen pro Komponente.
Um die Ergebnisse einordnen zu können, wurde zum Vergleich mit dem System LARKIN gerechnet. Dabei wurde in LARKIN für die Extrapolation nur die Berechnungen zu zwei verschiedenen Schrittweiten verwendet, da der implementierte Waveformalgorithmus höchstens die Ordnung 2 (Trapezregel) besitzt. Unter Beachtung dieser Beschränkung benötigt LARKIN 595 Funktionsauswertungen pro Komponente.
Es ist offensichtlich, daß unter diesen Voraussetzungen das Festhalten an den theoretischen Werten zur Sicherung der Konvergenz nicht zu einem konkurenzfähigen Verfahren führt.
Bei weiterer Analyse zeigt sich, daß ungefähr bei $t = 3.7 \cdot 10^{-6}$ für alle Komponenten die obere Schrittweitenschranke erreicht ist. Damit ist das Ende einer Grenzschicht erreicht, auf der eine relativ schnelle Dynamik abläuft. Danach ändern sich die Werte der einzelnen Konzentrationen nur noch langsam.
Die geschätzte Konvergenzrate liegt in diesem Bereich weit unter der eingestellten theoretischen Konvergenzrate von 0.5. Dadurch scheint das Beachten der Schrittweitenschranke aus Konvergenzgründen nicht mehr gerechtfertigt zu sein. Das Schrittweitenmaximum wird in diesem Fall solange heraufgesetzt, bis die Konvergenzrate sich wieder verschlechtert.
Durch das Verlasssen des Bereichs, in dem durch theoretische Abschätzungen das Ende der Iteration bestimmbar ist, ist das Einhalten der geforderten Fehlertoleranz nicht mehr apriori gesichert. Das Heraufsetzen des Schrittweitenmaximums macht sich nach einiger Zeit infolge eines größeren Fortpflanzungsfehlers bemerkbar. Das Verhalten der Lösungen zu den einzelnen Iterationen kann man sich so vorstellen, daß die Lösung langsam von der Mannigfaltigkeit mit langsamer Dynamik abdriftet und wieder in die Transientphase gerät. Das System befindet sich praktisch wieder auf einer neuen Grenzschicht. Das Verfahren reagiert mit einer entsprechend starken Schrittweitenreduzierung, bis die Konvergenz des Verfahrens wieder gesichert ist.
Das Verhalten des numerischen Algorithmus ist auch vergleichbar mit dem eines nichtsteifen Solvers bei steifen Problemen.
Wie sich bei den Testrechnungen zum Beispiel GLYCO1 herausgestellt hat, sind Aufwand und Genauigkeit wesentlich durch die Parameter ϵ_1 und $\tilde{h}$ der Fensterdefinition beeinflußbar.

Mit einem globalen Herabsetzen von ϵ_1 kann zwar die Genauigkeit positiv beeinflußt werden, dadurch erhöht sich aber der Aufwand in allen Komponenten. Besser ist ein selektives Vorgehen über eine Skalierung der geforderten Fehlertoleranz bei Berücksichtigung der „Stärke" der jeweiligen Kopplungen.

6.2 Testergebnisse für GLYCO1

Zunächst wurde das Beispiel GLYCO1 innerhalb der sogenannten Grenzschicht mit dem Waveformalgorithmus berechnet. Bei $t_a = 0$ wurde für t_z mit $4 \cdot 10^{-6}$ ein Zeitpunkt gewählt, bei dem die Schrittweite nur noch durch die theoretische Schranke, die sich aus Konvergenzbedingungen ableitet, beschränkt wird.
Der Simulator LARKIN benötigt in diesem Intervall 17 Schritte und 154 Funktionsauswertungen, d.h. bei 65 Spezies 10010 Funktionsauswertungen insgesamt. Mit dem Waveformalgorithmus werden für die entgültig akzeptierten Lösungen 29 – 396 Diskretisierungsschritte benötigt, wofür insgesamt 26510 Funktionsauswertungen notwendig sind. Davon sind 7866 Funktionsauswertungen nur für die Bestimmung der Fensteranfänge erforderlich. Für die reine Berechnung der Lösung ergibt das ungefähr den zweifachen Aufwand. Bei Beachtung der unterschiedlichen Diskretisierungsverfahren und der Tatsache, daß die ungünstigste Form der Zerlegung in Einzelgleichungen verwendet wurde, ist das ein recht gutes Ergebnis.
Die Tabelle 1 auf Seite 23 zeigt das Listing der statistischen Auswertung für die Simulation von GLYCO1 in der Grenzschicht. Dabei steht fac für die Anzahl der Funktionsauswertungen pro Spezie, stc für die Zahl der Diskretisierungsschritte über alle Fenster und ssc für die Zahl der Diskretisierungspunkte in der akzeptierten Lösung (der letzten Iterierten). Die Größe rate ist der Quotient von stc und ssc und gibt die aufwandsmäßige Anzahl von Iterationen wieder. Die letzte Spalte der Tabelle enthält mit itc das Verhältnis der Summe aller Fensterlängen zum Analyseintervall und somit die durchschittliche Anzahl von Iterationen (die Anzahl der Waveform-Iterationen), wenn man den zugrundeliegenden Funktionenraum betrachtet.
Bei näherer Untersuchung stellt sich heraus, daß ein relativ hoher Aufwand aus den Spezies 0 (GLU), 2 (HKG) und 4 (HKP) resultiert. Diese kommen in den Reaktionsgleichungen 1 – 3 vor. Die Reaktion 2 enthält dabei die Reaktionskonstante, die die großen Lipschitzkonstanten bewirkt. Nimmt man Reaktion 4 noch hinzu, erhält man ein „relativ geschlossenes" Teilsystem aus den ersten 8 Komponenten, das wir im weiteren mit T8 bezeichnen wollen. Die Lipschitzkonstanten in dem reduzierten System sind die gleichen wie im vollständigen System. Die Grenzschicht liegt im Intervall $[0, 1.36 \cdot 10^{-5}]$, wenn man wieder das Überschreiten des Schrittweitenmaximums als Kriterium verwendet.
Für den Vergleich mit dem vollständigen System zeigen wir zunächst die Auswertung der Simulation im Intervall $[0, 4 \cdot 10^{-6}]$. Die Tabelle 2 auf Seite 24 zeigt eine ähnliche Aufwandsverteilung der entsprechenden Spezies wie im Gesamtsystem. Der Aufwand liegt bei mehr als einem Drittel des Aufwandes für das Gesamtsystem. LARKIN benötigt hier 235 Schritte also insgesamt 1880 Funktionsauswertungen.
Betrachtet man nur die Reaktion 2, erhält man ein Teilsystem von 3 gewöhnlichen Differentialgleichungen zwischen den Konzentrationen der Spezies 2 (HKG), 3 (1TP) und 4 (HKP). Die Grenzschicht für dieses System (im weiteren mit T3 bezeichnet) liegt im Intervall $[0, 5.0 \cdot 10^{-6}]$ und die Lösungen streben schnell auf eine Ruhelage zu. In diesem Fall ist der Waveformalgorithmus auch mit großen Schrittweiten durchführbar. Man muß bei der Implemen-

```
 0: fac =   1947 stc =   814 ssc =   388 rate = 2.098 itc = 3.818
 1: fac =    644 stc =   251 ssc =    46 rate = 5.457 itc = 6.395
 2: fac =   1488 stc =   609 ssc =   158 rate = 3.854 itc = 4.881
 3: fac =    702 stc =   266 ssc =    44 rate = 6.045 itc = 7.968
 4: fac =   2313 stc =  1001 ssc =   396 rate = 2.528 itc = 4.217
 5: fac =    473 stc =   173 ssc =    37 rate = 4.676 itc = 6.413
 6: fac =    728 stc =   258 ssc =    40 rate = 6.450 itc = 8.543
 7: fac =    350 stc =   119 ssc =    32 rate = 3.719 itc = 4.329
 8: fac =    504 stc =   192 ssc =    45 rate = 4.267 itc = 5.209
 9: fac =    276 stc =    84 ssc =    35 rate = 2.400 itc = 2.888
10: fac =    303 stc =    97 ssc =    43 rate = 2.256 itc = 3.223
11: fac =    323 stc =   111 ssc =    32 rate = 3.469 itc = 4.493
12: fac =    371 stc =   118 ssc =    38 rate = 3.105 itc = 4.197
13: fac =    373 stc =   119 ssc =    31 rate = 3.839 itc = 5.147
14: fac =    368 stc =   132 ssc =    31 rate = 4.258 itc = 5.570
15: fac =    344 stc =   120 ssc =    51 rate = 2.353 itc = 3.677
16: fac =    280 stc =    85 ssc =    28 rate = 3.036 itc = 4.442
17: fac =    287 stc =    87 ssc =    35 rate = 2.486 itc = 3.045
18: fac =    298 stc =    94 ssc =    38 rate = 2.474 itc = 3.385
19: fac =    285 stc =    89 ssc =    29 rate = 3.069 itc = 4.115
20: fac =    320 stc =   108 ssc =    39 rate = 2.769 itc = 3.674
21: fac =    253 stc =    77 ssc =    32 rate = 2.406 itc = 3.489
22: fac =    239 stc =    71 ssc =    33 rate = 2.152 itc = 3.191
23: fac =    270 stc =    83 ssc =    30 rate = 2.767 itc = 4.117
24: fac =    268 stc =    80 ssc =    40 rate = 2.000 itc = 2.650
25: fac =    250 stc =    75 ssc =    33 rate = 2.273 itc = 2.914
26: fac =    247 stc =    75 ssc =    27 rate = 2.778 itc = 4.057
27: fac =    323 stc =   104 ssc =    42 rate = 2.476 itc = 3.273
28: fac =    316 stc =   108 ssc =    35 rate = 3.086 itc = 3.794
29: fac =    270 stc =    84 ssc =    33 rate = 2.545 itc = 4.201
30: fac =    321 stc =   108 ssc =    33 rate = 3.273 itc = 4.203
31: fac =    317 stc =   101 ssc =    38 rate = 2.658 itc = 3.893
32: fac =    434 stc =   140 ssc =    41 rate = 3.415 itc = 4.400
33: fac =    394 stc =   131 ssc =    33 rate = 3.970 itc = 4.994
34: fac =    404 stc =   138 ssc =    32 rate = 4.312 itc = 5.738
35: fac =    365 stc =   121 ssc =    42 rate = 2.881 itc = 4.476
36: fac =    283 stc =    86 ssc =    45 rate = 1.911 itc = 2.343
37: fac =    268 stc =    85 ssc =    35 rate = 2.429 itc = 3.351
38: fac =    235 stc =    67 ssc =    36 rate = 1.861 itc = 3.044
39: fac =    349 stc =   121 ssc =    41 rate = 2.951 itc = 4.906
40: fac =    352 stc =   114 ssc =    55 rate = 2.073 itc = 3.405
41: fac =    311 stc =    94 ssc =    35 rate = 2.686 itc = 3.420
42: fac =    267 stc =    72 ssc =    43 rate = 1.674 itc = 1.664
43: fac =    397 stc =   126 ssc =    37 rate = 3.405 itc = 4.831
44: fac =    365 stc =   123 ssc =    34 rate = 3.618 itc = 4.899
45: fac =    278 stc =    82 ssc =    41 rate = 2.000 itc = 2.632
46: fac =    353 stc =   109 ssc =    35 rate = 3.114 itc = 4.626
47: fac =    278 stc =    90 ssc =    39 rate = 2.308 itc = 3.290
48: fac =    326 stc =   104 ssc =    41 rate = 2.537 itc = 3.310
49: fac =    357 stc =   117 ssc =    46 rate = 2.543 itc = 3.551
50: fac =    302 stc =    91 ssc =    56 rate = 1.625 itc = 2.539
51: fac =    306 stc =    96 ssc =    42 rate = 2.286 itc = 2.992
52: fac =    360 stc =   113 ssc =    51 rate = 2.216 itc = 3.093
53: fac =    283 stc =   104 ssc =    44 rate = 2.364 itc = 3.517
54: fac =    287 stc =    98 ssc =    49 rate = 2.000 itc = 2.427
55: fac =    307 stc =   113 ssc =    32 rate = 3.531 itc = 4.681
56: fac =    277 stc =    86 ssc =    48 rate = 1.792 itc = 2.591
57: fac =    275 stc =   100 ssc =    37 rate = 2.703 itc = 3.782
58: fac =    297 stc =    96 ssc =    43 rate = 2.233 itc = 3.224
59: fac =    317 stc =   117 ssc =    33 rate = 3.545 itc = 4.696
60: fac =    184 stc =    69 ssc =    43 rate = 1.605 itc = 1.950
61: fac =    237 stc =    65 ssc =    49 rate = 1.327 itc = 1.298
62: fac =    351 stc =   107 ssc =    36 rate = 2.972 itc = 3.569
63: fac =    352 stc =   110 ssc =    36 rate = 3.056 itc = 3.985
64: fac =    308 stc =    95 ssc =    51 rate = 1.863 itc = 2.324

Summe fac = 26510.
```

Tabelle 1: Statistik der Waveform-Iteration für Beispiel GLYCO1 im Intervall $[0, 4 \cdot 10^{-6}]$.

```
0: fac =   3178 stc =  1368 ssc =   680 rate = 2.012 itc = 4.514
1: fac =    774 stc =   327 ssc =    68 rate = 4.809 itc = 7.263
2: fac =   1010 stc =   407 ssc =   129 rate = 3.155 itc = 4.049
3: fac =    503 stc =   198 ssc =    54 rate = 3.667 itc = 4.920
4: fac =   2720 stc =  1195 ssc =   515 rate = 2.320 itc = 3.715
5: fac =    556 stc =   220 ssc =    53 rate = 4.151 itc = 5.739
6: fac =    394 stc =   163 ssc =    48 rate = 3.396 itc = 4.679
7: fac =    368 stc =   137 ssc =    51 rate = 2.686 itc = 3.840

Summe fac = 9503
```

Tabelle 2: Statistik der Waveform-Iteration für das Teilsystem T8 im Intervall $[0, 4 \cdot 10^{-6}]$.

```
0: fac = 1223 stc =  532 ssc = 275 rate = 1.935 itc = 2.151 err = 1.577E-4
1: fac = 1230 stc =  632 ssc = 195 rate = 3.241 itc = 5.608 err < 1E-10
2: fac = 3313 stc = 1535 ssc = 767 rate = 2.001 itc = 2.203 err = 1.581E-4

Summe fac = 5766.
```

Tabelle 3: Statistik der Waveform-Iteration für das Teilsystem T3 im Intervall $[0, 4]$, $\tilde{h} = 0$.

tierung allerdings einige Sorgfalt walten lassen, da durch den Einfluß von Rundungsfehlern das Verhalten ähnlich sein kann, wie im oben beschriebenen Fall des ständigen Abgleitens in die Transientphase.
Vor der Berechnung eines neuen Fensters kann analysiert werden, ob sich die jeweilige Komponente quasi in einer Ruhelage befindet. Ist das der Fall, dann muß dafür gesorgt werden, das der erste Schritt im neuen Fenster vom Solver akzeptiert wird, damit es nicht zu unnötig starken Schrittweitenreduzierungen kommt.
Die Tabellen 3 und 4 zeigen Aufwand und Genauigkeit für $\tilde{h} = 0$ bzw. $\tilde{h} = 0.5 \cdot w_{\min}$. Als exakte Lösung wurde das Ergebnis der Simulation mit LARKIN angesehen. LARKIN benötigt bei 25 Schritten $226 \cdot 3 = 678$ Funktionsauswertungen. Beachtet man die Unterschiede in den verwendeten Verfahren und den Mehraufwand durch Iteration, dann ist das Ergebnis noch zufriedenstellend. Bei Einhaltung der geforderten Genauigkeit erhöht sich der Aufwand auf ungefähr das 10-fache.

```
0: fac = 1362 stc =  607 ssc =  264 rate = 2.299 itc = 2.077 err = 1.527E-4
1: fac = 1130 stc =  515 ssc =  198 rate = 2.601 itc = 1.966 err < 1E-10
2: fac = 4645 stc = 2068 ssc = 1107 rate = 1.868 itc = 4.818 err = 1.532E-4

Summe fac = 7137.
```

Tabelle 4: Statistik der Waveform-Iteration für Teilsystem T3 GLYCO1 im Intervall $[0, 4]$, $\tilde{h} = 0.5 \cdot w_{\min}$.

```
0: fac =  56530 stc =  24453 ssc =  1721 rate = 14.209 itc =  99.390
1: fac =  80068 stc =  34348 ssc =  1827 rate = 18.800 itc =  88.370
2: fac = 275261 stc =  98287 ssc = 11055 rate =  8.891 itc = 123.459
3: fac =  69384 stc =  27248 ssc =  1668 rate = 16.336 itc = 108.007
4: fac = 367920 stc = 136264 ssc = 23732 rate =  5.742 itc = 145.420
5: fac =  51121 stc =  19792 ssc =  1298 rate = 15.248 itc =  48.517
6: fac =  23590 stc =   9221 ssc =   925 rate =  9.969 itc =  52.804
7: fac =  27290 stc =  10853 ssc =   979 rate = 11.086 itc =  56.476

Summe fac = 951164

LARKIN: 68 Schritte 613*8 = 4904 Funktionsauswertungen
```

Tabelle 5: Teilsystem T8 im Intervall $[0,4]$, für $\tilde{h} = 0$ ohne Toleranzskalierung

Spezies	Waveform-Iteration	LARKIN	Relativer Fehler
GLU	6.4028180891360690E-05	.662754E-04	0,03390729
HKS	8.0894458108394218E-07	.958760E-06	0,15625956
HKG	1.3286980072353675E-07	.162498E-06	0,18232962
1TP	5.0640489580573868E-03	.506629E-02	0,00044234
HKP	4.1525967495533082E-10	.507955E-09	0,18248728
HKU	4.1402589465511283E-09	.463355E-08	0,10646072
ADP	9.3628713245427928E-04	.934186E-03	0,00224916
GLP	3.6750431000279282E-05	.346529E-04	0,06052974

Tabelle 6: Beispiel wie Tabelle 5, Werte für $t = 4$.

Während sich beim Teilsystem T3 die Auftrennung der starken Kopplungen zwischen den Gleichungen nicht so gravierend auf Aufwand und Genauigkeit auswirken, ist dies beim System T8 im starken Maße der Fall. Als Ursache dafür ist zu sehen, daß die langsame Dynamik noch weit ab von der steady state Lösung stattfindet. Die Iteration führt das System immer wieder in die Transientphase, wo ein entsprechend hoher Aufwand notwendig ist, um die geforderte Genauigkeit zu erreichen. Der globale Fehler ist dabei stark vom Parameter $\tilde{h}$ in der Fensterdefinition abhängig, der wiederum den Gesamtaufwand stark beeinflußt. Tabellen 5 und 6 zeigen das Ergebnis der Simulation, wenn man wie bei den obigen Rechnungen $\tilde{h} = 0$ setzt. Trotz hohen Aufwands ist am Ende des Intervalls bereits ein Fehler zu verzeichnen, der die geforderte Genauigkeit von 0.1% stark überschreitet.
In weiteren Simulationen wurde $\tilde{h}$ in der Form $c \cdot w_{\min}$ angesetzt. Durch Variation von $\tilde{h}$ allein, konnte allerdings keine zufriedenstellende Genauigkeit erzielt werden. Zusätzlich ist noch die Verwendung der Skalierung in der Fehlertoleranz notwendig. Die Skalierung wird so bestimmt, daß die Komponente, die andere stark beeinflußt, mit größerer Genauigkeit berechnet wird, als die Komponente, die andere nur schwach oder gar nicht beeinflußt. Die für die Wichtung verwendete Formel entspricht der Wichtung der Jacobimatrixelemente im Partitionierungsalgorithmus aus Abschnitt 4. Die Produkte aus Jacobimatrixelement und Betrag der jeweiligen Komponente werden in Relation zueinander gesetzt. Der kleinste dieser Werte wird auf 1.0 und der größte auf eine vorgegebene Skalierungsgrenze abgebildet.
Die Tabelle 7 gibt die Werte für $\tilde{h}$, die Skalierungsgrenze und die erzielte Genauigkeit im Vergleich zur Simulation mit LARKIN wieder.
Die Testrechnungen haben gezeigt, daß man die vorgstellte Variante der Waveform-Iteration über Parameter in der Fensterdefinition so einstellen kann, daß auch bei schlechter Partitionierung das richtige Ergebnis (allerdings unter erhöhtem Aufwand) geliefert wird. Damit ist die Voraussetzung geschaffen, um verschiedene Partitionierungen hinsichtlich ihrer Effiziens zu vergleichen.
Weiterhin hat sich gezeigt, daß man eine starke Kopplung nicht Auftrennen sollte, wenn diese zu einem Teilsystem gehört, das die wesentliche Dynamik des Gesamtsystems ausmacht und wenn diese keine quasi konstante Dynamik aufweist.
Beachtet man die „Stärke“ einer aufgetrennten Kopplung bei der Skalierung der Fehlertoleranz, so kann die Genauigkeit der Gesamtsimulation postiv beeinflußt werden.
Die behandelten Systeme wiesen jeweils eine Grenzschicht auf, in der mit der Waveform-Iteration schon bei der Zerlegung in einzelne Gleichungen ein brauchbares Ergebnis erzielt wurde, das bei einer geeigneten Zerlegung in Teilsysteme nur besser werden kann.

6.3 Vergleich verschiedener Partionierungen für das Beispiel GLYCO1

In Abbildung 6 und Tabelle 8 werden die Anzahl der Funktionsauswertungen für das Beispiel GLYCO1 bei verschiedenen Partitionierungen und verschiedenen Simulationsintervallen dargestellt. Mit *glyco1_1, glyco1_2* und *glyco1_3* sind die Zerlegungen entsprechend den Abbildungen 2,3 und 5 auf Seite 16 ff. bezeichnet. Während *glyco1_4* eine weitere Zerlegung in eine geränderte Blockstruktur mit Konzentrationen der Spezies zu Anfang des Analyseintervalls bezeichnet, ist *glyco1_5* eine Zerlegung mit den entsprechenden Werten am Ende des Analyseintervalls ($[0.0, 4.0]$).

fac	tolscal		
$\tilde{h}$	0.1	0.01	0.001
$0.05 \cdot w_{\min}$	1080737	1197873	1489407
$0.25 \cdot w_{\min}$	1362448	1384956	1629552
$0.5 \cdot w_{\min}$	1600534	1452828	1777382
$0.75 \cdot w_{\min}$	1501466	1531340	1771979
$1.0 \cdot w_{\min}$		1676311	1814487

GLU	tolscal		
$\tilde{h}$	0.1	0.01	0.001
$0.05 \cdot w_{\min}$	$2.86 \cdot 10^{-2}$	$2.44 \cdot 10^{-2}$	$2.26 \cdot 10^{-3}$
$0.25 \cdot w_{\min}$	$2.75 \cdot 10^{-2}$	$3.60 \cdot 10^{-3}$	$3.96 \cdot 10^{-4}$
$0.5 \cdot w_{\min}$	$1.26 \cdot 10^{-2}$	$3.88 \cdot 10^{-4}$	$2.13 \cdot 10^{-4}$
$0.75 \cdot w_{\min}$	$2.66 \cdot 10^{-3}$	$9.01 \cdot 10^{-4}$	$1.43 \cdot 10^{-4}$
$1.0 \cdot w_{\min}$		$3.60 \cdot 10^{-4}$	$5.33 \cdot 10^{-4}$

HKS	tolscal		
$\tilde{h}$	0.1	0.01	0.001
$0.05 \cdot w_{\min}$	$1.20 \cdot 10^{-1}$	$1.07 \cdot 10^{-1}$	$2.44 \cdot 10^{-2}$
$0.25 \cdot w_{\min}$	$1.23 \cdot 10^{-1}$	$4.45 \cdot 10^{-2}$	$1.64 \cdot 10^{-2}$
$0.5 \cdot w_{\min}$	$6.84 \cdot 10^{-2}$	$6.75 \cdot 10^{-3}$	$9.94 \cdot 10^{-4}$
$0.75 \cdot w_{\min}$	$1.85 \cdot 10^{-2}$	$6.52 \cdot 10^{-4}$	$1.02 \cdot 10^{-3}$
$1.0 \cdot w_{\min}$		$2.02 \cdot 10^{-3}$	$1.36 \cdot 10^{-3}$

Tabelle 7: Anzahl der Funktionsauswertungen und Genauigkeit für zwei ausgewählte Spezies bei verschiedenen Toleranzskalierungen und Werten für $\tilde{h}$

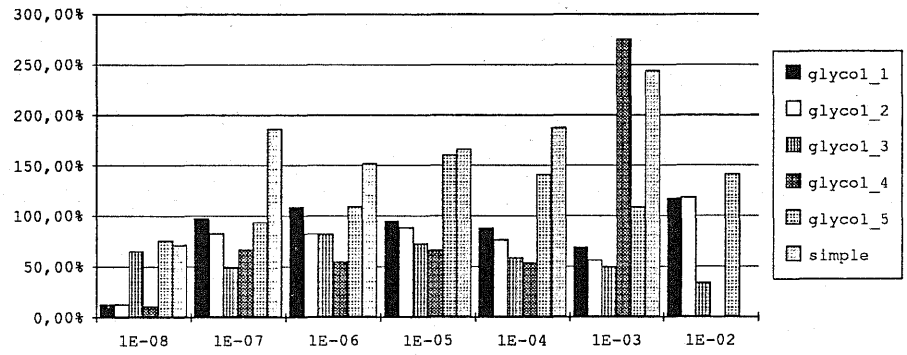

Abbildung 6: Vergleich unterschiedlicher Partitionen, Beispiel GLYCO1

Anzahl der Funktionsauswertungen

	1E-08	1E-07	1E-06	1E-05	1E-04	1E-03	1E-02
glyco1_0	8905	18980	27690	36920	52715	91390	143390
glyco1_1	1133	18494	30139	34949	46142	62475	438606
glyco1_2	1133	15698	22791	32621	40095	51337	178308
glyco1_3	5813	9264	22824	26721	30925	45003	61495
glyco1_4	963	12652	15053	24329	28174	251665	1464228
glyco1_5	6718	17798	30217	59185	74143	99206	131709
simple	6344	35326	42056	61409	98819	222812	813652

Prozentualer Vergleich mit glyco1_0

	1E-08	1E-07	1E-06	1E-05	1E-04	1E-03	1E-02
glyco1_1	12,72%	97,44%	108,84%	94,66%	87,53%	68,36%	305,88%
glyco1_2	12,72%	82,71%	82,31%	88,36%	76,06%	56,17%	124,35%
glyco1_3	65,28%	48,81%	82,43%	72,38%	58,66%	49,24%	42,89%
glyco1_4	10,81%	66,66%	54,36%	65,90%	53,45%	275,37%	1021,15%
glyco1_5	75,44%	93,77%	109,13%	160,31%	140,65%	108,55%	91,85%
simple	71,24%	186,12%	151,88%	166,33%	187,46%	243,80%	567,44%

Tabelle 8: Vergleich unterschiedlicher Partitionen, Beispiel GLYCO1

Mit *simple* wurde die Zerlegung in einzelne Gleichungen bezeichnet. Verglichen wurde die Anzahl der Funktionsauswertungen jeweils mit den Ergebnissen, die der Waveformalgorithmus für das gesamte System als ein Teilsystem (= 100%) liefert. Dabei bleibt von dem Algorithmus im wesentlichen die Unterteilung des Analyseintervalls in Fenster übrig. Die Werte für die Zerlegungen *glyco1_4* und *simple* im Intervall $[0.0, 0.01]$ wurden weggelassen, da sie über 500% liegen.
Es zeigt sich, daß in einer kleinen Umgebung des Anfangspunktes des Analyseintervalls das iterative Vorgehen bei allen gewählten Zerlegungen von Vorteil ist.
Begibt man sich etwas weiter weg vom Startpunkt, ist die Zerlegung 4 zunächst sehr günstig, bis sie dann, ähnlich wie die triviale Zerlegung, sehr schlechte Resultate liefert. Die Zerlegung 5 hingegen wird zum Ende der Analyse hin immer günstiger, was auch den Erwartungen entspricht. Der Vergleich der Zerlegungen 1 und 2 verdeutlicht, daß eine feinere Aufteilung über die Wahl des Parameters p_2 bessere Ergebnisse liefern kann. Schließlich hat sich bei diesem Beispiel die Zerlegung 3 als besonders günstig (für ein längeres Zeitintervall) erwiesen. Hier bieten sich weitere Untersuchungen im Zusammenhang mit dynamischer Partitionierung an.

7 Parallelisierung

Bedingt durch die Beschränktheit von Chache- bzw. Hauptspeicher, nimmt die Leistung schneller Einprozessor-Maschinen mit der Größe des zu lösenden Problems ab. Die volle Ausnutzung der Prozessorleistung verlangt bei großen Problemen entsprechend viel Hauptspeicher. Praktisch ist dieser zur Zeit nur auf Architekturen mit verteiltem Speicher (distributed memory), d.h. Parallelrechnern, vorhanden. Um Parallelrechner einsetzen zu können, ist es gleichzeitig erforderlich, daß die zur Verfügung stehenden Algorithmen für den effizienten Einsatz auf Mehrprozessor-Maschinen geeignet sind.
Numerische Verfahren, die auf der Waveform-Iteration basieren, weisen neben der Möglichkeit, die Schrittweiten an die einzelnen Komponenten anzupassen, zusätzlich den Vorteil auf, daß sie auf einfache Weise parallelisierbar sind. Dabei sind für konkrete Implementierungen verschiedene Ansätze möglich.
Der einfachste Weg ist die Verwendung des Gauß-Jacobi-Ansatzes. Bei diesem Ansatz ist die Möglichkeit der parallelen Simulation der einzelnen Teilsysteme offensichtlich.
Für die praktische Realisierung einer parallelen Waveformiteration stand den Autoren bisher ein Transputersystem, d.h. ein Parallelrechner mit verteiltem Speicher zur Verfügung, bei dem der Datenaustausch auf dem sogenannten message passing basiert.
Bei Verwendung des Gauß-Jacobi-Ansatzes in der ursprünglichen Form ist auf dieser Hardware nach jeder Iteration mit einer relativ großen Pause für den Datenaustausch zu rechnen. Verschiedene Tests (z.B. [26]) haben ergeben, daß ein einmaliges Versenden eines Datenpakets ungefähr soviel Zeit wie 5000 Gleitkommaoperationen in Anspruch nimmt. Je nach Größe des Problems kann dadurch der Gesamtaufwand auf mehreren Prozessoren größer sein, als auf einem.
Die Leistungsfähigkeit der Transputer kann nur ausgenutzt werden, wenn die Kommunikation im wesentlichen parallel zur Simulation verläuft. Das heißt, der Gauß-Jacobi-Algorithmus muß derart abgewandelt werden, daß nach jeder Iteration die Lösungen mittels ansynchroner Kommunikation im Hintergrund ausgetauscht werden und gleichzeitig die Simulation fortgesetzt wird. Da im allgemeinen mit unterschiedlichem Aufwand für die Simulation der verschiedenen Teilsysteme zu rechenen ist und der Zeitaufwand für die Kommunikation von der Datenmenge und der Lokalisierung der Transputer im Prozessornetz abhängt, kann man nicht davon ausgehen, daß zur k-ten Iteration alle benötigen Daten aus der $(k-1)$-ten Iteration bereits verfügbar sind. Es stehen die Eingangsdaten von verschiedenen weiter zurückliegenden Iterationen zur Verfügung.
Das resultierende Verfahren läßt sich durch den folgenden asynchronen Gauß-Jacobi-Ansatz beschreiben.

$$
\begin{aligned}
\dot{x}_1^k &= g_1(x_1^k, \quad x_2^{k-d_{1,2}^k}, \quad \cdots, \quad x_r^{k-d_{1,r}^k}, t), \\
&\vdots \quad \vdots \\
\dot{x}_i^k &= g_i(x_1^{k-d_{i,1}^k}, \quad \cdots, \quad x_{i-1}^{k-d_{i,i-1}^k}, x_i^k, x_{i+1}^{k-d_{i,i+1}^k}, \cdots, \quad x_r^{k-d_{i,r}^k}, t), \\
&\vdots \quad \vdots \\
\dot{x}_r^k &= g_r(x_1^{k-d_{r,1}^k}, \quad \cdots, \quad x_r^k, \quad t), \\
x_i^k(t_a) &= x_{0,i}, \; i = 1, \cdots, n.
\end{aligned}
$$

Hierbei ist $d_{i,j}^k$ die Differenz zwischen dem Iterationszähler des verwendeten Inputs und dem Iterationszähler der aktuellen Iteration. Für das normale Gauß-Jacobi-Verfahren ist $d_{i,j}^k \equiv 1$. Für das Gauß-Seidel-Verfahren ist

$$d_{i,j}^k = \begin{cases} 1, & i < j, \\ 0, & i > j. \end{cases}$$

Für die Konvergenz dieses Verfahrens ist außer den üblichen Voraussetzungen die Beschränktheit der Verzögerungen $d_{i,j}^k$ hinreichend. Sei $d := \max_{i,j,k} d_{i,j}^k$. Dann läßt sich das asynchrone Gauß-Jacobi-Verfahren in der allgemeineren Form

$$\begin{aligned} \dot{x}^k(t) &= f(x^k, x^{k-1}(t), x^{k-2}(t), \ldots, x^{k-d}(t), t), \\ x^k(t_a) &= x_0 \end{aligned}$$

schreiben.

Eine andere Möglichkeit der Parallelisierung bietet das Gauß-Seidel-Verfahren in Verbindung mit dem sogenannten Time-Point-Pipelining [27]. Bei diesem Verfahren findet die Kommunikation nicht nach jeder Iteration sondern nach jedem Zeitschritt statt. Der Prozessor für das zweite Teilsystem kann mit der Berechnung des ersten Zeischrittes beginnen, nachdem der erste Prozessor die Lösung für den ersten Zeitschritt des ersten Teilsystems geliefert hat. Gleichzeitig wird auf dem ersten Prozessor der zweite Zeitschritt für das erste Teilsystem berechnet usw. . Die Kommunikation ist bei Transputersystemen mit T800 Prozessoren bei kleinne Datenmengen relativ teuer. Dadurch ist der Zeitgewinn durch Parallelisierung bei diesem Vorgehen nicht besonders hoch [27].
Da der Zeitaufwand für den Austausch von einigen Kbyte Daten im wesentlichen so groß ist wie für ein Byte, lassen sich die Kosten für die Kommunikation wesentlich verringern, wenn man das Pipeline-Prinzip nicht auf die einzelnen Zeitschritte, sondern auf die oben definierten Zeitfenster anwendet.
Bei unabhängiger Zeitdiskretisierung der einzelnen Teilsysteme ist der Simulationsaufwand für die einzelnen Zeitfenster unterschiedlich groß. Ein synchrones Vorgehen wie beim Time-Point-Pipelining würde auf den Prozessoren, die weniger Zeitschritte zu berechnen haben, zu unnötigen Wartezeiten führen. Deswegen scheint es angebracht, das Verfahren zusätzlich mit dem asynchronen Vorgehen zu kombinieren. Das Resultat ist ein asynchrones Gauß-Jacobi-Verfahren mit variablen Fenstern. Unter Verwendung der Bezeichnungen von (5.3) können wir es in der Form

$$\begin{aligned} x_i^0(t) &:= \Xi_i^0(x_{i,0}, t), \\ x_i^k(t) &= \begin{cases} x_i^{k-1}(t), & t \le \bar{t}_i^k, \\ x_i^{k-1}(\bar{t}_i^k) + \int_{\bar{t}_i^k}^t f_i(x^k(s), x^{k-1}(s), \ldots, x^{k-d}(s), s)ds, & \bar{t}_i^k < t \le \bar{T}_i^k, \\ \Xi_i^k(x_i^k, t), & \bar{T}_i^k < t \end{cases} \end{aligned} \tag{7.1}$$

schreiben.
Die Fenster werden nicht entsprechend der bereits global berechneten Iterierten berechnet, sondern nach den lokal auf jedem Prozessor verfügbaren Informationen. Dabei spielt es keine Rolle, aus welcher Iteration die Lösungen für die einzelnen Inputvariablen stammen. Wichtig ist nur die Abweichung zu den Werten, die für die Berechnung der vorherigen Iteration auf diesem Prozessor benutzt wurden.
Für den Fall, daß mehr Teilsysteme als Prozessoren vorhanden sind, können auf einem Prozessor mehrere Teilsysteme sequientiell entsprechend der „normalen" Waveformiteration nacheinander abgearbeitet werden. In (7.1) sind dann die entsprechenden Verzögerungen 0 bzw. 1, da unterhalb der Teilsysteme auf einem Prozessor auf keine kommunikationsbedingte Zeitverzögerung auftritt.

Die Effiziens des parallelen Algorithmus hängt stark vom Zeitaufwand für die Kommunikation und von der kommunikationsbedingten Verzögerung bei der Bereitstellung der Inputvariabeln ab.
Im weiteren wollen wir eine Lösung dieses Transportproblems mittels einer Paketvermittlung, die asynchron im Hintergrund und im wesentlichen zeitlich parallel zur eigentlichen Simulation erfolgt, vorstellen.
Es wird vorausgesetzt, daß auf jedem Prozessor die Information über die lokale Hardwaretopology vorhanden ist. Das Minimum an Information besteht in der Prozessornummer zur Identifikation und der Anzahl der Eingangs- und Ausgangsdatenkanäle. Die Datenkanäle entsprechen den physikalischen oder virtuellen Links auf dem Transputernetz. Bei Prozessornetzen ohne Beschränkung der Anzahl der Kommunikationskanäle, kann man als Datenkanäle alle paarweise möglichen Kommunikationsverbindungen ansehen.
Ausgehend von dieser Information wird in einem Initialisierungsschritt die Hardwaretopology auf den Datenfluß des Gleichungssystems entsprechend der gewählten Dekomposition abgebildet. Zunächst wird ermittelt, welcher Prozessor über welchen Ausgangskanal erreichbar ist (Abbildung 7). Dann verschickt der Masterprozessor die Partitionierungsinformation an alle Prozessoren. Diese besteht aus einer Tabelle mit einem Eintrag der Prozessornummer pro Gleichung. Für den Fall, daß ein Prozessor nicht direkt über einen Ausgangskanal erreichbar ist, werden die Daten über mehrere Prozessoren weitergeleitet. Dazu wird ein Datenpaket aufgebaut, daß aus zwei Teilen besteht, einem Kopf mit der Information über Größe des Pakets, Art des Inhalts und Liste der Prozessoren, an die es verschickt werden soll, und dem Rumpf mit den eigentlichen Daten.
Bei Paketen, die die Lösung für ein Fenster enthalten, wird die Liste der Prozessoren aus der Struktur der Jacobimatrix und der Partitionierungsinformation ermittelt. Bei der Kommunikation von Nachbar zu Nachbar wird dann entsprechend der Empfängerliste und der Information, über welchen Ausgabekanal die einzelnen Prozessoren erreichbar sind, das Datenpaket in die jeweils notwendigen Richtungen mit einer entsprechend korrigierten Empfängerliste verschickt. Der Empfänger untersucht, ob die Daten lokal weiterverarbeitet werden sollen und liest die Daten entsprechend ein. Danach wird das Paket an die verbliebenen Adressaten weitergeleitet.
Unter Verwendung der ansynchronen Kommunikationsroutinen kann der Datentransfer so in den sequentiellen Algorithmus eingebettet werden, daß er zeitlich parallel zur eigentlichen Simulation erfolgt, ohne den CPU-Zeitaufwand wesentlich zu erhöhen (Abbildung 8).

```
Aktuelle Prozessornummer              : pr_id
Anzahl der Prozessoren                : pn
Anzahl der ermittelten Ausgabekanaele : ipn
Ausgabekanal fuer Prozessor pr_id = -1;
if (Anzahl der Ausgabekanaele > 1)
        {
        Schicke Packet f"ur Eintrag von Prozessornummern an
            alle Ausgabekanaele; /* Code MSG_PROC_GRAF */
        ipn=1;
        }
        else
        {
        fuer alle Prozessoren != pr_id Ausgabekanal = 0;
        ipn=pn+1;
        }
fuer alle Eingabekanaele inp_ready = false;

for (ready=0;(ipn<pn)||(!ready);)
        {
        warten auf Eingangsmeldung Kanal msg_stream, Code msg_code,
                Laenge msg_len;
        if (msg_stream<0)
                break; /* keine ungelesenen Meldungen mehr vorhanden*/
        if (msg_code==MSG_PROC_GRAF)
                        {
                        Meldung lesen;
                        msg_stream++;
                           /* naechste Lesen beginnt beim naechsten Kanal */
                        if (Absenderprozessor==pr_id)
                                {
                                Kanaele fuer Prozessore laut Paket eintragen;
                                ipn weiterzaehlen;
                                }
                                else
                                if (pr_id nicht im Paket gekennzeichnet oder
                                        Anzahl der Ausgabekanaele >1)
                                   {
                                   pr_id eintragen und Paket an alle
                                        Ausgabekanaele  weiterschicken;
                                   }
                        }
                        else
                        if (msg_code==MSG_FINISH) /* fertig */
                                {
                                Meldung auf Kanal msg_stream loeschen;
                                Fuer Kanal msg_stream inp_ready = true
                                }
        for (i=0,ready=1;i<nr_in_stream;i++)
                if (inp_ready == false fuer Kanal i)
                        {
                        ready=0;
                        break;
                        }
        if (ipn==pn)
                {
                Meldung MSG_FINISH an alle Ausgabekanaele;
                ipn++;
                }
        }
if (ipn<pn)
        Fehler;
```

Abbildung 7: Aufbau des Prozessorgraphen

```
Initialisierung (Prozessorgraph usw.)
ready := false
while not ready do
        begin
        if (Daten im Eingangskanal vorhanden)
                begin
                Lese Daten
                Wenn Daten fuer Prozessor pr_id bestimmt
                        {
                        igl = Nummer der Gleichung der Eingangsdaten
                        pr_id aus Liste der Adressaten loeschen
                        }
                end
                else
                begin
                wenn fuer alle Gleichungen
                        Konvergenzintervall > Analyseintervall
                        dann ready = true;
                igl = naechste Gleichung fuer Prozessor pr_id;
                Test, ob Gleichung simuliert werden kann und
                        Festlegung des Fensteranfangs
                wenn Gleichung igl behandelt werden soll
                        begin
                        wenn Eingangsdaten vorhanden
                                starte asynchrones lesen
                        Berechne Loesung im neuen Fenster
    fuer GLeichung igl
                        Paket zur Weitergabe der Loesung aufbauen
                        end
                end
        Weitergabe der Loesung fuer die Gleichung igl an alle noch
                ausstehenden Adressaten
        Wenn Loesung fuer Prozessor pr_id bestimmt ist
                begin
                Werte fuer Gleichung igl in Speicher fuer spaetere
                        Interpolation eintragen
                Konvergenzrate, Schranke fuer Fensterlaenge,
                        Schrittweitenmaximum usw. korrigieren
                Bei Bedarf Jacobimatrix neu auswerten,
                        Lipschitzkonstanten aktualisieren
                end
        end
```

Abbildung 8: Simulationsablauf mit eingebetteter Kommunikation

Zusätzlicher Aufwand entsteht dadurch, daß die Kommunikation wesentlich langsamer ist als eine Rechenoperation. Der Empfang eines Datenpakets erfolgt mit relativ großer Zeitverzögerung, d.h. die jeweilige Iterierte ist erst viel später auf anderen Prozessoren verfügbar (große Verzögerung d). Dadurch kann, vor allem in der Startphase und in Abhängigkeit vom verfügbaren Speicherplatz, Leerlauf entstehen, bis wieder neue Eingangsdaten verfügbar sind. Dieser Leerlauf wird umso geringer, je mehr Diskretisierungspunkte das neu zu berechnende Fenster enthält.
Wie in [25] gezeigt, ist der Aufwand für die Kommunikation an sich bei entsprechend langen Zeitintervallen zu vernachlässigen, d.h. die Kommunikation findet im wesentlichen parallel zur Simulation statt.
Der Mehraufwand resultiert vor allem durch die kommunikationsbedingte Verzögerung in der Iterierten, die eine Verschlechterung der Konvergenzeigenschaften über eine große Verzögerung d in (7.1) verursacht. Diese Größe d hängt unter anderem auch von der Zuordnung der Gleichungen zu den Prozessoren ab und sollte daher bei der Partitionierung mit berücksichtig werden.
Im Zusammenhang mit einer schlechten Verteilung der Prozessorlast kann es noch zu einem weiteren Problem kommen. Wenn ein Prozessor schneller mit der Verarbeitung der lokal vorhandenen Daten fertig ist, als neue Daten bereitliegen, kommt es auf diesem Prozessor zu einer Leerlaufphase. In dieser Zeit werden von ihm auch keine neuen Lösungen produziert, die in anderen Prozessoren benötigt werden. Es kommt zu einer Lücke im Datenstrom. Die Lücken im Datenstrom setzen sich von Prozessor zu Prozessor über die gesamte Simulation fort. Die Frage ist, wie die Prozessorlast so verteilt werden kann, daß die Lücken im Datenstrom klein bleiben.
Das folgende „akademische" Beispiel zeigt, daß der Aufwand für die Kommunikation bei entsprechender Problemgröße und geeigneter Lastverteilung zu vernachlässigen ist.
Wir betrachten das chemisches Rekationssystem

$$\begin{aligned} X0 &\longleftrightarrow X1 \\ X1 &\longleftrightarrow X2 \\ &\vdots \\ X(n-1) &\longleftrightarrow Xn. \end{aligned}$$

Die mathematische Modellierung führt auf das Gleichungssystem (5.1) mit der rechten Seite

$$\begin{aligned} g_1(x,t) &:= -2x_1 + x_2 + x_0(t) \\ &\vdots \\ g_i(x,t) &:= x_{i-1} - 2x_i + x_{i+1} \\ &\vdots \\ g_n(x,t) &:= x_{n-1} - x_n. \end{aligned}$$

Für die Reaktionsgeschwindigkeiten wurde jeweils der Wert 1 angenommen. Für X0 wird eine gegebene Konzentration $x_0(t)$ gesetzt. Bei den unten vorgestellten Testrechnungen ist $x_0(t) := 2(1 + \sin(20t)$. Die Anfangswerte zum Zeitpunkt $t_a = 0$ sind jeweils 0.

```
0: fac =   2035 stc =   847 ssc =   291 rate = 2.911 itc = 3.586 cput =  2.18
1: fac =   1374 stc =   577 ssc =   188 rate = 3.069 itc = 4.352 cput =  1.74
2: fac =    876 stc =   373 ssc =    94 rate = 3.968 itc = 5.001 cput =  1.22
3: fac =    708 stc =   300 ssc =    83 rate = 3.614 itc = 5.134 cput =  0.96
4: fac =    657 stc =   274 ssc =    71 rate = 3.859 itc = 5.340 cput =  0.84
5: fac =    584 stc =   239 ssc =    62 rate = 3.855 itc = 5.453 cput =  0.79
6: fac =    553 stc =   222 ssc =    61 rate = 3.639 itc = 5.107 cput =  0.78
7: fac =    554 stc =   224 ssc =    63 rate = 3.556 itc = 5.047 cput =  0.75
8: fac =    476 stc =   193 ssc =    54 rate = 3.574 itc = 4.453 cput =  0.66
9: fac =    363 stc =   149 ssc =    57 rate = 2.614 itc = 3.240 cput =  0.52

Anzahl der Funktionsaufrufe = 8180
Prozessorzahl: 1,
cpu-time: i     0.22 w     0.15 g     10.4 s 10.86
```

Tabelle 9: 10 Gleichungen, $t_z = 10$, sequentiell.

```
0: fac =   7436 stc =  3576 ssc =   518 rate = 6.903 itc = 16.203 cput =  0.32
1: fac =   3856 stc =  1701 ssc =   421 rate = 4.040 itc =  7.233 cput =  4.59
2: fac =   1865 stc =   813 ssc =   133 rate = 6.113 itc =  7.534 cput =  2.52
3: fac =   1209 stc =   519 ssc =    93 rate = 5.581 itc =  7.455 cput =  1.63
4: fac =    940 stc =   397 ssc =    77 rate = 5.156 itc =  6.836 cput =  1.35
5: fac =    856 stc =   360 ssc =    67 rate = 5.373 itc =  7.229 cput =  1.24
6: fac =    773 stc =   318 ssc =    68 rate = 4.676 itc =  6.590 cput =  1.22
7: fac =    756 stc =   307 ssc =    59 rate = 5.203 itc =  7.042 cput =  0.98
8: fac =    745 stc =   305 ssc =    56 rate = 5.446 itc =  6.748 cput =  1.14
9: fac =    465 stc =   192 ssc =    47 rate = 4.085 itc =  5.110 cput =  0.77

Anzahl der Funktionsaufrufe = 18901
Prozessorzahl: 2,
cpu-time: i     0.56 w     0.73 g     15.8 s 17.16
```

Tabelle 10: 10 Gleichungen, $t_z = 10$, parallel.

In [25] wurde bereits gezeigt, daß bei hinreichend großer Dimension n die sequentielle Waveform-Iteration gegenüber klassischen Verfahren im Vorteil ist.
Die Tabellen 9 – 12 zeigen den zusätzlichen Zeitgewinn durch den Einsatz von Parallelrechnern mit verteiltem Speicher in Abhängigkeit von der Länge des Lösungsintervalls bzw. der Problemgröße.
Dabei ist zu beachten, daß das Verfahren des message passing bei Transputersystemen zu den zeitaufwendigen Kommunikationsmethoden gehört und außerdem unter dem Betriebssystem Helios gearbeitet wurde, das in Bezug auf die Kommunikationsgeschwindigkeit in der einschlägigen Literatur keine besonders guten Kritiken bekommen hat.
Die Tabellen 9 und 10 zeigen die statistische Auswertung für eine sequentielle und eine parallele Simulation für $n = 10$ und $t_z = 10$. Der höchste Aufwand liegt jeweils in der 1. Gleichung. In der parallelen Variante wurde die 1. Gleichung auf dem 1. Prozessor, die restlichen 9 Gleichungen auf dem 2. Prozessor gerechnet. Die Zeitmessungen erfolgen jeweils auf

Intervall	fc[0]	fc	cput	fc	cput	i	w	theoret	faktor
10.0	2035	8180	10.86	18901	0.56	0.73	17.16	0.75	1.58
20.0	2604	10876	14.41	28824	0.56	1.45	27.50	0.76	1.91
50.0	4051	18181	23.99	37566	0.55	0.60	37.17	0.78	1.55
100.0	6405	29714	39.09	47886	0.55	0.35	46.53	0.78	1.19
200.0	12170	52985	69.34	72416	0.55	0.71	71.05	0.77	1.02
500.0	26307	117493	159.89	147979	0.55	1.73	152.09	0.78	0.95
1000.0	50757	225297	319.83	272024	0.56	1.38	289.43	0.77	0.90

Tabelle 11: Vergleich sequentiell/parallel zu unterschiedlichen Lösungsintervallen $[t_a, t_z]$.

Dim.	fc[0]	fc	cput	fc	cput	i	w	theoret	faktor
10	50757	225297	319.83	272024	0.56	1.38	289.43	0.77	0.90
20	51647	323784	456.82	329390	0.58	3.99	382.16	0.84	0.83
40	52356	521960	761.97	507488	0.61	6.67	657.84	0.88	0.86

Tabelle 12: Vergleich sequentiell/parallel zu unterschiedlichen Problemgrößen.

dem 2. Prozessor. Dadurch erhält man erwartungsgemäß für die 1. Gleichung einen geringen Zeitaufwand. Die höhere Anzahl von Iterationen und Diskretisierungsschritten führen allerdings insgesamt zu einem höhrem Aufwand in der parallelen Rechnung.
Tabelle 11 zeigt den Vergleich von sequentieller und paralleler Simulation bei unterschiedlichen Lösungsintervallen und Tabelle 12 bei unterschiedlichen Problemgrößen. Die letzten beiden Spalten zeigen jeweils den theoretisch möglichen bzw. den praktisch erzielten Zeitgewinn . Die letzte Spalte (faktor) enthält den Quotienten der CPU-Zeiten von paralleler und sequientieller Rechnung.
Abschließend stellen wir Resultate für das Teilsystem T3 des Beispiels GLYCO1 aus Abschnitt 6 unter Verwendung von 3 Prozessoren vor. Hier ist besonders interessant, wie sich die Anzahl der Iterationen erhöht, da jetzt Rückkopplungen jeweils über zwei Prozessoren verzögert werden und die Jacobimatrix voll besetzt ist.
Es zeigt sich, daß sich die Anzahl der Iterationen in der parallelen Variante etwas erhöht hat (Tabelle 13). Gleichzeitig ist die kommunikationsbedingte Wartezeit wesentlich höher als die Zeit, die zum Lösen der Gleichungen benötigt wird. Verteuert man die Newtoniteration künstlich, dann verbessert sich das Verhältnis von Simulation zu Kommunikation (Tabelle 14).
Die folgenden zwei Zeitmessungen sind mit dem gleichen Beispiel erfolgt. In der Newtoniteration wurde die Funktionsauswertung für die Gleichung i jeweils DELAY[i] mal wiederholt. Um eine annähernd gleiche Prozessorlast zu erhalten wurde DELAY[0] = DELAY[1] = 3*DELAY[2] gewählt. Bei DELAY[2] = 300 ist der Zeitaufwand für eine Komponente in der sequentiellen Rechnung vergleichbar mit der Wartezeit der parallelen Version und der Zeitaufwand ehöht sich in der letzten Komponente durch die größere Anzahl von Iterationen auf das 1.5 fache. Insgesamt ist die parallele Simulation etwas schneller (Faktor 0.94) als die sequentielle. Verteuert man den Aufwand weiter, so verbessert sich das Verhältnis leicht. Bei DELAY[2]=600 erhält man z.B. ein Verhältnis von 1:0.89. Insgesamt gesehen zeigt dieses Beispiel wieder, daß ab einer gewissenen Problemgröße das parallele Verfahren gegenüber dem sequentiellen im Vorteil ist. Bei zu niedrigen Dimensionen, wie in dem hier diskutierten

sequentiell:

```
0: fac =   1767 stc =   769 ssc =   294 rate = 2.616 itc = 2.039 cput =  3.66
1: fac =   1692 stc =   728 ssc =   252 rate = 2.889 itc = 1.388 cput =  3.89
2: fac =   4654 stc =  2104 ssc =   724 rate = 2.906 itc = 2.446 cput =  5.74

Anzahl der Funktionsaufrufe = 8113
Prozessorzahl: 1,
cpu-time: i     0.23 w     2.19 g     13.3 s 15.8, eps = 0.001
```

parallel:

```
0: fac =   3069 stc =  1308 ssc =   244 rate = 5.361 itc =  1.895 cput =  0.97
1: fac =   2018 stc =   952 ssc =   309 rate = 3.081 itc =  4.073 cput =  0.87
2: fac =   8432 stc =  3848 ssc =   750 rate = 5.131 itc = 11.461 cput = 11.0

Anzahl der Funktionsaufrufe = 13519
Prozessorzahl: 3,
cpu-time: i     0.93 w     39.3 g     12.8 s 53.26, eps = 0.001
```

Tabelle 13: Teilsystem T3 von Beispiel GLYCO1, Vergleich sequentiell/parallel

sequentiell:

```
0: fac =   1767 stc =   769 ssc =   294 rate = 2.616 itc = 2.039 cput =  19.3
1: fac =   1692 stc =   728 ssc =   252 rate = 2.889 itc = 1.388 cput =  19.3
2: fac =   4654 stc =  2104 ssc =   724 rate = 2.906 itc = 2.446 cput =  24.3

Anzahl der Funktionsaufrufe = 8113, Org. 1164
Prozessorzahl: 1,
cpu-time: i     0.22 w     2.09 g     62.9 s 65.23, eps = 0.001
```

parallel:

```
0: fac =   1699 stc =  1055 ssc =   297 rate = 3.552 itc = 3.141 cput =   0.85
1: fac =   1898 stc =  1065 ssc =   231 rate = 4.610 itc = 7.254 cput =   0.91
2: fac =   7056 stc =  3133 ssc =   694 rate = 4.514 itc = 4.402 cput =  36.4

Anzahl der Funktionsaufrufe = 10653
Prozessorzahl: 3,
cpu-time: i     0.91 w     20.9 g     38.2 s 61.6, eps = 0.001
```

Tabelle 14: Teilsystem T3 bei „künstlich verteuerter“ Newtoniteration

Beispiel, lohnt sich der Einsatz von Parallelrechnern hingegen noch nicht.

Literatur

[1] Saeks, R. (Ed.) Large-Scale Dynamical Systems. North Hollywood. California 1976.

[2] Large Scale Scientific Computing, Vol. 1 und folgende.

[3] Sangiovanni-Vincentelli, A.L., White, J.: Relaxation techniques for the simulation of VLSI circuits. Boston 1987.

[4] Miščenko, E.F., Rozov, N.Ch.: Differentialgleichungen mit einem kleinen Parameter und Relaxationsschwingungen (russ.), Moskau 1975.

[5] Lelarasmee, E., Ruehli, A.E., Sangiovanni-Vincentelli, A.L.: The waveform relaxation method for time domain analysis of large scale integrated circuits. IEEE Trans. on CAD of IC and Syst. vol 1, 131-145 (1982).

[6] Miekkala, U., Nevanlinna, O.: Convergence of dynamic iteration methods for initial value problems. SIAM J. Sci. Stat. Comput., (1987), 459-482.

[7] Nevanlinna, O.: Remarks on Picard-Lindelöf Iteration. Report-Mat-A254, Inst. of Math., Helsinki Univ. of Techn (1987).

[8] Deuflhard, P., Bader, G., Nowak, U.: LARKIN - A Software Package for the Simulation of Large Systems arising in Chemical Reaction Kinetics. (Univ. Heidelberg, SFB 123: Techn. Rep. 100 (1980))
in: K.H. Ebert, P. Deuflhard, W. Jaeger (ed.): Modelling of Chemical Reaction Systems. Springer Series Chem. Phys. 18 (1981).

[9] Othmer, H.G.: The Interaction of Structure and Dynamics in Chemical Reaction Networks. Springer Series Chem. Phys. 18 (1981).

[10] Feinberg, M.: Reaction Network Structure, Multiple Steady States, and Sustained Composition Oscillations: A Review of Some Results. Springer Series Chem. Phys. 18 (1981).

[11] Chen, C.-C., Hu, Y.H.: Parallel LU Factorization for Circuit Simulation on MIMD Computer. Proc. ICCD 1988, 129-132.

[12] Smart, D.; White, J.: Reducing the Parrallel Solution Time of Sparse Circuit Matrices Using Recordered Gaussian Elimination and Relaxation. Proc. of Int. Symposium on Circuit and Systems, 627-630, 1988.

[13] Schneider, K.R.: Singularly perturbed autonomous differential systems. In "Dynamical systems and environmental models", ed. by Bothe, H.G. et al., Akademie-Verlag, Berlin 1987, 32-39.

[14] Schneider, K.R.: Relaxation oscillations and singularly perturbed systems. Proc. V. Conf. Diff. Equs., Rousse 1989.

[15] Schneider, K.R., Bremer, I.: Contraction and decomposition. Preprint P-Math-33/88 des KWI der AdW, 9 S. (1988).

[16] Bremer, I., Schneider, K.R.: A remark on solving large systems in function spaces. Aplikace Mathematiky 35, 494-498 (1990).

[17] Borchardt, J.; Bremer, I.; Grund, F.; Horn, D.; Uhle, M.: MAGNUS - Mehrstufige Analyse großer Netzwerke auf der Basis von Untersystemen. Tagungsmaterial zur "3. Tagung Schaltkreisentwurf Dresden", 1989, 28.-30.3.1989, S. 148-152.

[18] Borchardt, J.; Bremer, I.; Grund, F.; Horn, D.; Uhle, M.: Erweiterung des Simulators MAGNUS zum Mixed-Mode-Elektriksimulator. A2-Bericht zum Thema Netzwerk- und Timinganalyse", Karl-Weierstraß-Institut für Mathematik, AdW der DDR, 37 S., 1989.

[19] Schneider, K.R.: A remark on the wave-form relaxation method. Int. J. Circuit Theory and Appl. 19, 101-104 (1991).

[20] George,A.; Lin,J.W.: Computer Solution of Large Sparse Positive Definite Systems New York, Prentice Hall (1991).

[21] Naeher, St.: LEDA: Library of Efficient Data Types and Algorithms – user manual Saarbrücken (1992).

[22] Gear, C.W.: Numerical Initial Value Problems in Ordinary Differential Equations New York, Prentice Hall (1971).

[23] Hindmarsh, A.C.: GEAR – Ordinary Differential Equation System Solver Lawrence Livermore Laboratory, Techn. Report UCID–30001, Rev.3 (1974).

[24] Bader, G.; Deuflhard, P.: A Semiimplicit Mid–Point Rule for Stiff Systems of Ordinary Differential Equations Numerische Mathematik 41, 373-398 (1983).

[25] Bremer, I.: Kurveniteration auf diskreten Zeitskalen IWR Universitaet Heidelberg Preprint 93 - 06 Januar 1993.

[26] Bader, G., Gehrke, E.: On the Performance of Transputer Networks for Solving Linear Systems of Equations. Parallel Computing, Vol. 17 (1991), pp 1397 - 1407,

[27] W. Rissiek and J. Stroop.: Ein paralleles waveform-Relaxationsverfahren für die Simulation von VLSI-Schaltungen. Cadlab Report 18/90, November 1990.

Veröffentlichungen des Instituts für Angewandte Analysis und Stochastik

Preprints 1992

1. D.A. Dawson, J. Gärtner: Multilevel large deviations.
2. H. Gajewski: On uniqueness of solutions to the drift–diffusion–model of semiconductor devices.
3. J. Fuhrmann: On the convergence of algebraically defined multigrid methods.
4. A. Bovier, J.-M. Ghez: Spectral properties of one–dimensional Schrödinger operators with potentials generated by substitutions.
5. D.A. Dawson, K. Fleischmann: A super–Brownian motion with a single point catalyst.
6. A. Bovier, V. Gayrard: The thermodynamics of the Curie–Weiss model with random couplings.
7. W. Dahmen, S. Prößdorf, R. Schneider: Wavelet approximation methods for pseudodifferential equations I: stability and convergence.
8. A. Rathsfeld: Piecewise polynomial collocation for the double layer potential equation over polyhedral boundaries. Part I: The wedge, Part II: The cube.
9. G. Schmidt: Boundary element discretization of Poincaré–Steklov operators.
10. K. Fleischmann, I. Kaj: Large deviation probability for some rescaled superprocesses.
11. P. Mathé: Random approximation of finite sums.
12. C.J. van Duijn, P. Knabner: Flow and reactive transport in porous media induced by well injection: similarity solution.
13. G.B. Di Masi, E. Platen, W.J. Runggaldier: Hedging of options under discrete observation on assets with stochastic volatility.
14. J. Schmeling, R. Siegmund–Schultze: The singularity spectrum of self-affine fractals with a Bernoulli measure.
15. A. Koshelev: About some coercive inequalities for elementary elliptic and parabolic operators.
16. P.E. Kloeden, E. Platen, H. Schurz: Higher order approximate Markov chain filters.

17. H.M. Dietz, Y. Kutoyants: A minimum–distance estimator for diffusion processes with ergodic properties.

18. I. Schmelzer: Quantization and measurability in gauge theory and gravity.

19. A. Bovier, V. Gayrard: Rigorous results on the thermodynamics of the dilute Hopfield model.

20. K. Gröger: Free energy estimates and asymptotic behaviour of reaction–diffusion processes.

21. E. Platen (ed.): Proceedings of the 1^{st} workshop on stochastic numerics.

22. S. Prößdorf (ed.): International Symposium "Operator Equations and Numerical Analysis" September 28 – October 2, 1992 Gosen (nearby Berlin).

23. K. Fleischmann, A. Greven: Diffusive clustering in an infinite system of hierarchically interacting diffusions.

24. P. Knabner, I. Kögel–Knabner, K.U. Totsche: The modeling of reactive solute transport with sorption to mobile and immobile sorbents.

25. S. Seifarth: The discrete spectrum of the Dirac operators on certain symmetric spaces.

26. J. Schmeling: Hölder continuity of the holonomy maps for hyperbolic basic sets II.

27. P. Mathé: On optimal random nets.

28. W. Wagner: Stochastic systems of particles with weights and approximation of the Boltzmann equation. The Markov process in the spatially homogeneous case.

29. A. Glitzky, K. Gröger, R. Hünlich: Existence and uniqueness results for equations modelling transport of dopants in semiconductors.

30. J. Elschner: The *h-p*–version of spline approximation methods for Mellin convolution equations.

31. R. Schlundt: Iterative Verfahren für lineare Gleichungssysteme mit schwach besetzten Koeffizientenmatrizen.

32. G. Hebermehl: Zur direkten Lösung linearer Gleichungssysteme auf Shared und Distributed Memory Systemen.

33. G.N. Milstein, E. Platen, H. Schurz: Balanced implicit methods for stiff stochastic systems: An introduction and numerical experiments.

34. M.H. Neumann: Pointwise confidence intervals in nonparametric regression with heteroscedastic error structure.

35. M. Nussbaum: Asymptotic equivalence of density estimation and white noise.

Preprints 1993

36. B. Kleemann, A. Rathsfeld: Nyström's method and iterative solvers for the solution of the double layer potential equation over polyhedral boundaries.

37. W. Dahmen, S. Prössdorf, R. Schneider: Wavelet approximation methods for pseudodifferential equations II: matrix compression and fast solution.

38. N. Hofmann, E. Platen, M. Schweizer: Option pricing under incompleteness and stochastic volatility.

39. N. Hofmann: Stability of numerical schemes for stochastic differential equations with multiplicative noise.

40. E. Platen, R. Rebolledo: On bond price dynamics.

41. E. Platen: An approach to bond pricing.

42. E. Platen, R. Rebolledo: Pricing via anticipative stochastic calculus.

43. P.E. Kloeden, E. Platen: Numerical methods for stochastic differential equations.

44. L. Brehmer, A. Liemant, I. Müller: Ladungstransport und Oberflächenpotentialkinetik in ungeordneten dünnen Schichten.

45. A. Bovier, C. Külske: A rigorous renormalization group method for interfaces in random media.

46. G. Bruckner: On the regularization of the ill–posed logarithmic kernel integral equation of the first kind.

47. H. Schurz: Asymptotical mean stability of numerical solutions with multiplicative noise.

48. J.W. Barrett, P. Knabner: Finite element approximation of transport of reactive solutes in porous media. Part I: Error estimates for non–equilibrium adsorption processes.

49. M. Pulvirenti, W. Wagner, M.B. Zavelani Rossi: Convergence of particle schemes for the Boltzmann equation.

50. J. Schmeling: Most β shifts have bad ergodic properties.

51. J. Schmeling: Self normal numbers.

52. D.A. Dawson, K. Fleischmann: Super–Brownian motions in higher dimensions with absolutely continuous measure states.

53. A. Koshelev: Regularity of solutions for some problems of mathematical physics.

54. J. Elschner, I.G. Graham: An optimal order collocation method for first kind boundary integral equations on polygons.

55. R. Schlundt: Iterative Verfahren für lineare Gleichungssysteme auf Distributed Memory Systemen.

56. D.A. Dawson, K. Fleischmann, Y. Li, C. Müller: Singularity of super–Brownian local time at a point catalyst.

57. N. Hofmann, E. Platen: Stability of weak numerical schemes for stochastic differential equations.

58. H.G. Bothe: The Hausdorff dimension of certain attractors.

59. I.P. Ivanova, G.A. Kamenskij: On the smoothness of the solution to a boundary value problem for a differential–difference equation.

60. A. Bovier, V. Gayrard: Rigorous results on the Hopfield model of neural networks.

61. M.H. Neumann: Automatic bandwidth choice and confidence intervals in nonparametric regression.

62. C.J. van Duijn, P. Knabner: Travelling wave behaviour of crystal dissolution in porous media flow.

63. J. Förste: Zur mathematischen Modellierung eines Halbleiterinjektionslasers mit Hilfe der Maxwellschen Gleichungen bei gegebener Stromverteilung.

64. A. Juhl: On the functional equations of dynamical theta functions I.