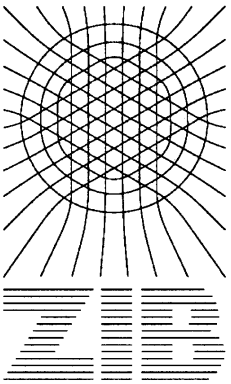
Konrad-Zuse-Zentrum für Informationstechnik Berlin
Heilbronner Str. 10, D-1000 Berlin 31



Gerhard Maierhöfer Georg Skorobohatyj

Implementierung von parallelen Versionen der Gleichungslöser EULEX und EULSIM auf Transputern

Technical Report TR 90–2 (März 1990)

Herausgegeben vom
Konrad-Zuse-Zentrum für Informationstechnik Berlin
Heilbronner Strasse 10
1000 Berlin 31
Verantwortlich: Dr. Klaus André
Umschlagsatz und Druck: Rabe KG Buch- und Offsetdruck Berlin

ISSN 0933-789X

Inhalt

Einleitung 1
1. Mathematischer Hintergrund 1
2. Ansätze zur Parallelisierung des Extrapolationsschemas 4
3. Synchronisation 17
4. Meßergebnisse 22
5. Parallelisierung der JACOBI–Matrix–Berechnung 37
6. Anwendung auf partielle Differentialgleichungen 39
Literatur 49

Zusammenfassung

Im vorliegenden Bericht wird die Parallelisierung zweier numerischer Algorithmen zur Lösung gewöhnlicher Differentialgleichungssysteme 1.Ordnung (explizite und semi–implizite Euler–Diskretisierung und h–Extrapolation) beschrieben.

Implementiert wurden die Algorithmen mit OCCAM2 unter TDS (Transputer Development System) mit bis zu 4 Transputern T800.

Meßwerte für die erreichten Beschleunigungen werden anhand mehrerer Beispiele von Differentialgleichungssystemen angegeben.

Schlüsselwörter: Adaptive, parallele Systeme; OCCAM2; Transputer; numerische Gleichungslöser; Euler–Diskretisierung; h–Extrapolation

Einleitung

In diesem Bericht werden Parallelisierungsmöglichkeiten für die Differentialgleichungslöser EULEX und EULSIM – implizite und semi–implizite Euler–Verfahren – untersucht, mögliche Ansätze diskutiert und für die implementierten Versionen die erzielten Werte für die Beschleunigungen dargestellt. Als Implementierungssprache wurde OCCAM2 verwendet. Es standen 8 Transputer T800 in einem SUN3–Hostsystem zur Verfügung. Für die Parallelisierung wurden bis zu 4 Transputer eingesetzt.

Ausgehend von den sequentiell implementierten adaptiven Algorithmen wurden der Zeitbedarf von Subalgorithmen – Berechnung der Jakobi–Matrix, Diskretisierung, Extrapolation – ermittelt, um adäquate Aufwandsabschätzungen für mögliche Beschleunigungswerte zu erhalten.

1. Mathematischer Hintergrund

Mit den als Unterprogramme implementierten Routinen EULSIM und EULEX können die numerischen Lösungen eines Systems steifer bzw. nicht steifer Differentialgleichungen erster Ordnung ermittelt werden:

$$\begin{aligned}
\dot{y}_1 &= f(y_1, y_2, \ldots, y_n, t) \\
\dot{y}_2 &= f(y_1, y_2, \ldots, y_n, t) \\
&\vdots \\
\dot{y}_n &= f(y_1, y_2, \ldots, y_n, t) \; .
\end{aligned}$$

Als Eingabe–Parameter erhalten EULEX und EULSIM eine Funktion $F = (f_1, f_2, \ldots, f_n)$, einen Anfangspunkt t, Anfangswerte $y_1(t), y_2(t), \ldots, y_n(t)$, einen Endpunkt t_{end}, eine relative Genauigkeit *eps* und eine initiale und maximale Schrittweite h und $h_{\max}$. Als Ergebnis werden Lösungswerte $y_1(t_{\text{end}})$, $y_2(t_{\text{end}}), \ldots, y_n(t_{\text{end}})$ des Differentialgleichungssystems im Endpunkt t_{end} geliefert mit der vorgegebenen relativen Genauigkeit oder eine Fehleranzeige, falls das nicht möglich ist. In beiden Fällen wird die numerische Lösung schrittweise ermittelt derart, daß auch Lösungen an Zwischenpunkten $t_1 < t_2 < \ldots < t_m$ $(t < t_1, t_m < t_{\text{end}})$ entstehen, wobei die Schrittweiten $h_i = t_{i+1} - t_i$, $h_1 = h$, gemäß der in [1] beschriebenen Ordnungs- und Schrittweitensteuerung bestimmt werden; auf jedem Teilintervall $[t_i, t_{i+1}]$ wird ein Extrapolationsverfahren durchgeführt mit Näherungswerten, die bei EULEX durch explizite und bei EULSIM durch semi–implizite EULER–Diskretisierung erhalten werden. Im ersten Fall ergibt sich bei Vorgabe von $y(t) = (y_1(t), y_2(t), \ldots y_n(t))$ für ein

kleines t eine Näherung an der Stelle $t + \Delta t$ zu

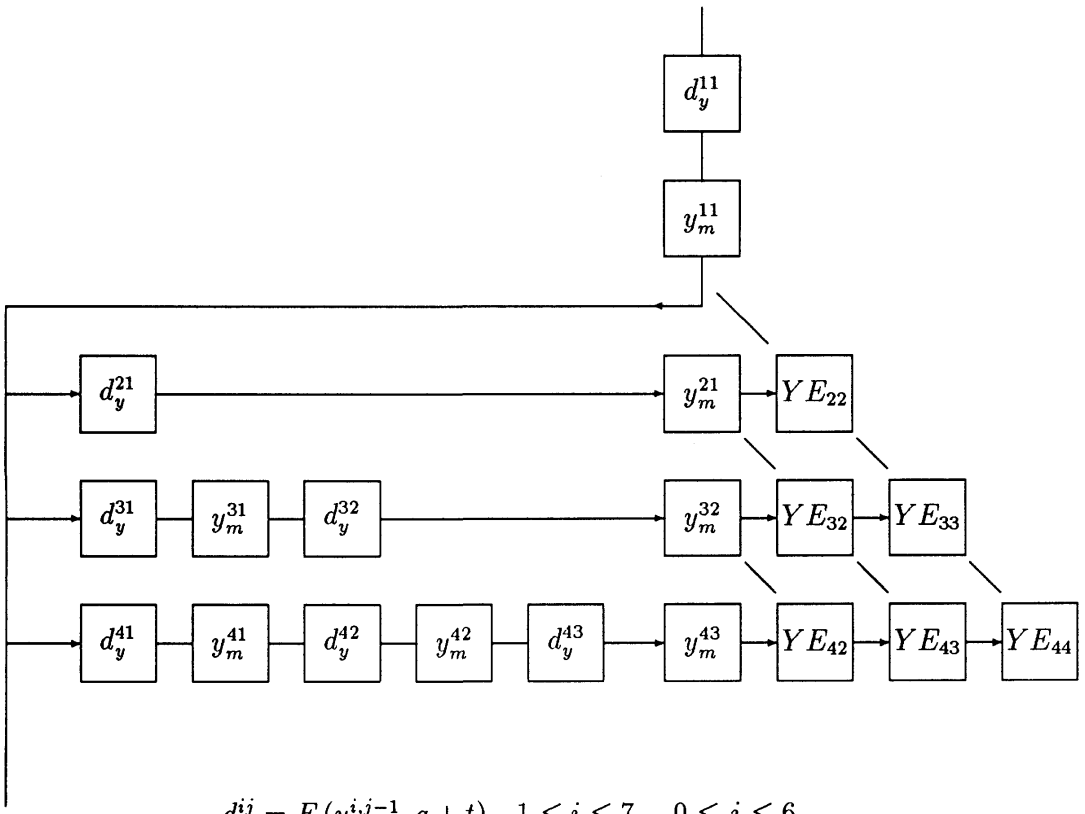
$$y(t + \Delta t) \cong y(t) + \Delta t * F(y, t)$$

und im zweiten Fall aus der Lösung des nicht-linearen Gleichungssystems

$$(I - \Delta t * A) * \Delta y = \Delta t * F(y, t) \ ,$$

mit $\Delta y = y(t + \Delta t) - y(t)$, wobei I die $(n \times n)$-Einheitsmatrix ist und A die JACOBI-Matrix $(\partial F(t)/\partial y)$, die in EULSIM näherungsweise durch Differenzen-Approximationen gewonnen wird. Für die Extrapolation über einem Teilintervall $[t_i, t_{i+1}]$ werden bezüglich einer Ordnung j jeweils auf die beschriebene Weise Schätzwerte an den Stützstellen $t_i + (k/n_j) * h_i$, $0 \leq k \leq j-1$, bestimmt, dabei ist $(n_j = j)$, $1 \leq j \leq j_{\max}$, die harmonische Folge. Da in eine F-Auswertung auch ein zu einer Stützstelle t gehöriger Wert y eingeht, müssen die bei einer Extrapolationsordnung nötigen F-Auswertungen sequentiell vorgenommen werden, und auch die einzelnen "basic steps" müssen sequentiell erfolgen. Die Datenabhängigkeiten und der Ablauf von Diskretisierung und Extrapolation sind in Bild [1] für EULEX dargestellt. In einem (Polynomial-) Extrapolationsschritt wird ein Zwischenwert $YE_{j,k}$ berechnet zu

$$\begin{aligned} YE_{j,1} &= y_m^{j1} \\ YE_{j,k} &= YE_{j,k-1} + \frac{YE_{j,k-1} - YE_{j-1,k-1}}{\left(n_j / n_{j-k+1}\right) - 1} \ . \end{aligned}$$



$$d_y^{ij} = F\,(y_m^{i,j-1}, g+t) \quad 1 \le i \le 7, \quad 0 \le j \le 6$$

mit:

$$y_m^{10} = y_0 \quad \text{und} \quad g_{ij} = \tfrac{j}{i} * h_{11}$$

Bild 1

Nach Berechnung eines $YE_{j,j}$ findet für $(koh-1) \le j \le (koh+1)$, $j \ge 2$, eine Überprüfung auf Konvergenz - Erreichen der geforderten relativen Genauigkeit - statt mit *koh* als optimaler Ordnung aus der Ordnungs- und Schrittweitensteuerung (siehe [1]). Im Falle der Konvergenz wird $YE_{j,j}$ als numerische Lösung des Differentialgleichungssystem mit der vorgegebenen Genauigkeit an der Stelle $t_{i+1} = t_i + h_i$ betrachtet und das Verfahren wiederholt mit einem Intervall $[t_{i+1}, t_{i+2}]$. Falls keine Konvergenz vorliegt, wird spätestens bei $j = koh+1$ das Verfahren mit einer kleineren Schrittweite h_i wiederholt unter Beibehaltung des linken Intervall-Endpunktes t_i. Letzteres geschieht aber nicht beliebig oft, sondern nur bis zu einer Grenze, bei deren Überschreiten sich beide Programme mit einer Fehlermeldung beenden.

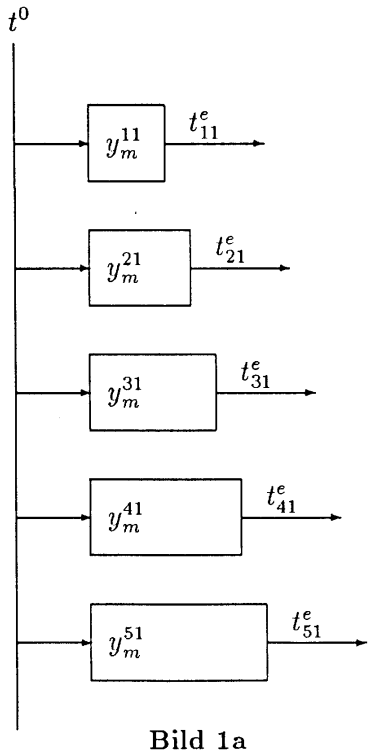


Bild 1a

2. Ansätze zur Parallelisierung des Extrapolationsschemas

Wie aus Bild [1] ersichtlich, können die bei den einzelnen Etrapolationsordnungen nötigen Berechnungen parallel zueinander ablaufen, so daß dafür mehrere Prozessoren einsetzbar sind; bei der Extrapolation benötigte Daten sind dann von einem Prozessor zum nächsten zu senden. Würde jedoch jedem Prozessor genau ein solcher "Berechnungsstrang" (task) zugeordnet, so wären diese stark unterschiedlich ausgelastet (Bild [1a]), da der für $j = j_{\max} = 10$ am meisten zu tun hätte und der für $j = 1$ sehr wenig; daher sollten nur wenige Prozessoren verwendet werden derart, daß die Extrapolationszeilen optimal auf diese aufgeteilt sind. Zu beachten ist weiter, daß die Ordnung $koh + 1$, aus der die Anzahl der einzusetzenden Prozessoren zu bestimmen ist, über die "basic steps" variiert und auch, daß Konvergenz schon bei einer niedrigeren Ordnung $(koh-1)$ auftreten kann. Stellt sich z.B. schon bei der Ordnung zwei Konvergenz ein, werden Ergebnisse aus höheren Ordnungen nicht mehr gebraucht; die betreffenden Prozessoren sollten dann laufende Berechnungen abbrechen. Es ist aber

in OCCAM und auch in anderen Programmiersprachen nicht möglich, einem Prozessor eine asynchrone Abbruch–Nachricht zuzustellen, d.h. die aktive Task kann nicht zu einem beliebigen Zeitpunkt unterbrochen werden und daraufhin an einer definierten Stelle fortfahren, wie es in diesem Falle erforderlich ist, da ja im allgemeinen weitere "basic steps" folgen. Als Ersatz muß an geeigneten Stellen im Programm abgefragt werden, ob eine solche Nachricht eingetroffen ist, und dasselbe so aufgebaut sein, daß es darauf entsprechend reagiert. Soll z.B. eine Wiederholung einer Folge von Anweisungen $A_1, A_2, \ldots, A_n$ auf diese Weise unterbrechbar sein, so kann das betreffende Programmstück in OCCAM folgendermaßen formuliert werden:

```
CHAN OF BOOL in :
BOOL loop, break :
loop := TRUE
break := FALSE
WHILE (not break) AND loop
   SEQ
     PRI ALT
         in ? break
            SKIP
         TRUE & SKIP
            SEQ
              A1
               PRI ALT
                 in ? break
                   SKIP
                 TRUE & SKIP
                   SEQ
                     A2
                       ...
                          PRI ALT
                             in ? break
                               SKIP
                             TRUE & SKIP
                                SEQ
                                   An
```

Die `PRI ALT`–Konstruktion bewirkt, daß jeweils ein boolescher Wert auf dem Kanal "in" eingelesen wird, falls ein solcher aktuell bereit steht, d.h. von einem anderem Prozessor gerade gesendet wird, ansonsten aber der Zweig durchlaufen wird, der zur Eingabebedingung `TRUE & SKIP` gehört, die stets erfüllt ist.

Weitere Synchronisationsmaßnahmen sind erforderlich für das Übermitteln von Teilergebnissen $YE_{j,k}$ von einem Prozessor zu einem anderen. Ersichtlich steht

z.B. $YE_{3,1}$ auf Prozessor A früher für den die Ordnung vier bearbeitenden Prozessor (B) bereit, als dieser ihn abholen kann; Prozessor A sollte aber nicht auf Prozessor B warten, wie es bei Absetzen einer Maschineninstruktion "Output Message" über ein Transputer–Link der Fall ist, sondern mit der Berechnung von $YE_{3,2}$ fortfahren. Derartige Synchronisationsprobleme sind durch Einführung zusätzlicher paralleler Prozesse lösbar; die zeitliche Koordinierung erfolgt durch den Multi–Tasking–Scheduler auf Hardware–Ebene, d.h. während ein Prozeß auf Datentransfer wartet, kann ein anderer weiterarbeiten. Für die Übergabe der $YE_{j,k}$ ist daher auf jedem Transputer ein parallel zur Berechnung derselben laufender Prozeß vorzusehen, der diese auch puffert und erst auf Anforderung durch einen anderen Prozessor absendet; eine solche Vorgehensweise wird auch durch das Programmiermodell von OCCAM nahe gelegt. Ein Prozeß, der das Beschriebene leistet, kann in OCCAM – in etwas vereinfachter Notation – folgendermaßen formuliert werden:

```
PROTOCOL in IS
   tag_result ; INT::[]REAL64
   tag_end
CHAN OF in from_comp :
CHAN OF INT::[]REAL64 to_next_ord :
CHAN OF BOOL result_request :
PROC comm ()
   [n]REAL64 Y :
   INT k :
   BOOL busy, result_request :
   BEGIN
      result_request := FALSE
      busy := TRUE
     WHILE busy
         ALT
            from_comp ? CASE
               tag_res ; k ; Y
                  IF (result_request AND buffer_empty())
                  THEN
                     to_next_ord ! k ; Y
                     result_request := FALSE
                  ELSE put_buffer (k, Y)
            from_next_ord ? result_request
                  IF (NOT buffer_empty())
                       get_buffer (Y, k)
                       to_next_ord ! k ; Y
                       result_request := FALSE
                  END IF
   END.
```

Über den internen Kanal "from_comp" treffen die $Y_{j,k}$ des – hier nicht dargestellten – Rechenprozesses auf demselben Prozessor ein und über den Link–Kanal "from_next_ord" Anforderungen auf Übermittlung derselben durch den die nächste Ordnung bearbeitenden Rechenprozeß auf einem anderen Prozessor. Die ALT–Konstruktion bewirkt, daß der Kommunikationsprozeß auf beiden Kanälen gleichzeitig empfangsbereit ist und auf die jeweils zuerst eintreffende Nachricht reagiert. Treffen beide Nachrichten gleichzeitig ein, so ist die Auswahl, welche Nachricht zuerst behandelt wird, beliebig (implementierungsabhängig). Zum Speichern der $Y_{j,k}$ wird eine Pufferverwaltung gemäß FIFO benötigt, die Prozeduren "put_buffer", "get_buffer" und "buffer_empty" zur Verfügung stellt. Der Kommunikationsprozeß vermittelt also eine asynchrone Kommunikation zwischen den Rechenprozessen; bei Multiprozessor–Systemen, die eine asynchrone Kommunikation zwischen den Prozessoren gestatten, entfällt die Notwendigkeit der Programmierung solcher Kommunikationsprozesse. Bei solchen ist aber für das message passing auch eine Warteschlangenverwaltung nötig, die im allgemeinen durch ein zugrunde gelegtes Betriebssystem vorgenommen wird. Es ist daher für die obige Lösung im Vergleich zu Systemen mit synchroner Kommunikation kein Effizienzverlust zu erwarten. Ein Effizienzgewinn ist eher dadurch zu erzielen, daß das sequentielle Programm derart in parallel ausführbare Moduln zerlegt wird, daß die Kommunikation zwischen denselben minimal wird. In diesem Sinne erscheint es vorteilhaft, die Extrapolation doch auf einem Prozessor zu belassen und nur die Diskretisierung auf verschiedene Prozessoren zu verteilen. Das vereinfacht jedenfalls die Synchronisation erheblich, insbesondere die Behandlung der erwähnten Abbruch–Nachrichten. Zur Bestimmung der Anzahl der sinnvollerweise einzusetzenden Prozessoren sowie zur Verteilung der Ordnungen auf dieselben wurden einige Zeitmessungen durchgeführt, und zwar mit den Beispielen FE1 bis FE5 aus D36.DETEST für EULEX und mit den Beispielen A1 bis A4 aus H83.ALL.STITEST. In den folgenden Tabellen sind jeweils die Ausführungszeiten der Diskretisierung und der Extrapolation je Ordnung und Beispiel in μs angegeben, wobei durch die Hardware bedingte Meßungenauigkeiten vernachlässigt wurden.

Eulex

	Beispiel FE1		Beispiel FE2		Beispiel FE3	
j	Diskr.	Extr.	Diskr.	Extr.	Diskr.	Extr.
1	12	30	27	30	27	30
2	71	419	66	419	123	417
3	115	478	103	478	219	475
4	158	536	141	535	328	534
5	201	593	178	593	482	593
6	245	651	216	650	596	651
7	288	708	253	708	709	708
8	331	766	291	766	821	766

	Beispiel FE4		Beispiel FE5	
j	Diskr.	Extr.	Diskr.	Extr.
1	28	30	27	30
2	66	417	338	416
3	105	473	649	474
4	143	531	965	532
5	182	590	1276	590
6	220	648	(1589)	(648)
7	258	704	(1901)	(704)
8	297	761	(2212)	(764)

EULSIM

	Beispiel FA2		Beispiel FA2	
j	Diskr.	Extr.	Diskr.	Extr.
1	701	55	3368	115
2	989	554	4378	889
3	1275	666	5393	1139
4	1560	780	6399	1394
5	1846	893	7411	1647
6	2132	1006	8430	1901
7	2418	1119	9446	2154

	Beispiel FA3		Beispiel FA4	
j	Diskr.	Extr.	Diskr.	Extr.
1	706	55	3229	127
2	1012	552	4482	957
3	1315	665	5732	1239
4	1619	778	6983	1521
5	1923	891	8234	1802
6	2227	1003	9484	2081
7	2531	1116	—	—

Die angegebenen Ausführungszeiten sind Mittelwerte über alle für die jeweilige Ordnung vorkommenden Berechnungen. Die als höchste vorkommende Ordnung hängt ab von der vorzugebenden relativen Genauigkeit und vom Beispiel; in EULEX ist dafür als Maximum zehn vorgesehen und in EULSIM sieben. Bei Vorgabe von $eps = 10^{-4}$, was für technische Genauigkeit ausreicht, wird in EULEX für die herangezogenen Test-Beispiele jedoch nicht die Ordnung zehn erreicht, sondern neun für FE1 und FE3, acht für FE2 und sechs für FE4 und FE5. Es erscheinen daher drei oder vier Prozessoren für die Diskretisierung ausreichend, wenn ein Prozessor mehrere Ordnungen bearbeiten soll. Zur Erzielung einer günstigen Auslastung können jedem Prozessor genau zwei Ordnungen zugewiesen werden, indem das Dreiecks-Schema gemäß Bild [1] durch horizontale Halbierung in der Mitte und Umklappen in ein Rechteck verwandelt wird, falls die maximale Ordnung n gerade ist. Das Resultat ist in Bild [2] angegeben: Prozessor j bearbeitet die Ordnungen j und $n-(j-1)$, $j = 1,\ldots(n/2)$. Die Summen der betreffenden Ausführungszeiten sind für EULEX annähernd konstant, wie aus den Meßwerten hervorgeht. Für die Extrapolation ist dann ein weiterer Prozessor einzusetzen, der von den anderen die Eingabewerte $ym_{j,1}$ erhält. Dieses Parallelisierungsschema hat jedoch den Nachteil, daß die günstige Auslastung der Prozessoren nur für solche "basic steps" erzielt wird, bei denen tatsächlich bis zu einer vorgegeben Ordnung gerechnet werden muß, d.h., bis $j = joh + 1$. Ist z.B. $n = 8$ und $joh = 6$ für einen "basic step" vorgegeben, und stellt sich schon bei $j = joh - 1$ Konvergenz ein, so hat Prozessor vier einen höheren Anteil an der Diskretisierung als die Prozessoren eins bis drei zusammen, wie aus den Tabellen für EULEX ersichtlich. Nachteilig ist außerdem, daß in diesem Falle die anderen Prozessoren zu einem früheren Zeitpunkt mit Berechnungen beginnen, die dann wieder abgebrochen werden müssen wie oben beschrieben, als Prozessor vier mit der für die Ordnung fünf, die tatsächlich gebraucht wird. Günstiger wäre es, wenn Prozessor eins mit der Ordnung fünf fortführe an Stelle von Prozessor vier. Da die Extrapolation die Vektoren ym aus den Diskretisierungen in aufsteigender Reihenfolge sequentiell

verarbeitet, sollten jene möglichst auch in dieser Reihenfolge bereitgestellt werden, und zwar zum frühest möglichen Zeitpunkt. Diese Forderung führt zu einem Parallelisierungsschema gemäß Bild [3], in dem der Prozessor, der die Diskretisierungsordnung zwei behandelt, auch die Extrapolation durchführt. In den folgenden Diagrammen ist der daraus entstehende zeitliche Ablauf für die Beispiele $FA1$ und $FA2$ (EULSIM) angegeben unter Vernachlässigung der Datenübertragungszeiten und auch der Zeit, die der Konvergenz–Monitor braucht.

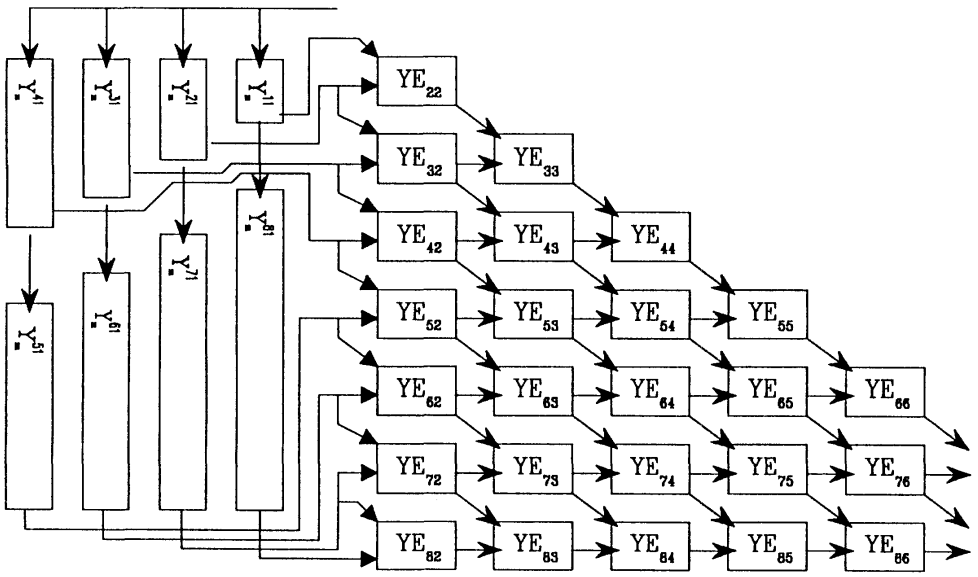


Bild 2. Rechenschema für geradzahlige i (z.B. $i = 8$)
bei Verwendung von i/2 Prozessoren.

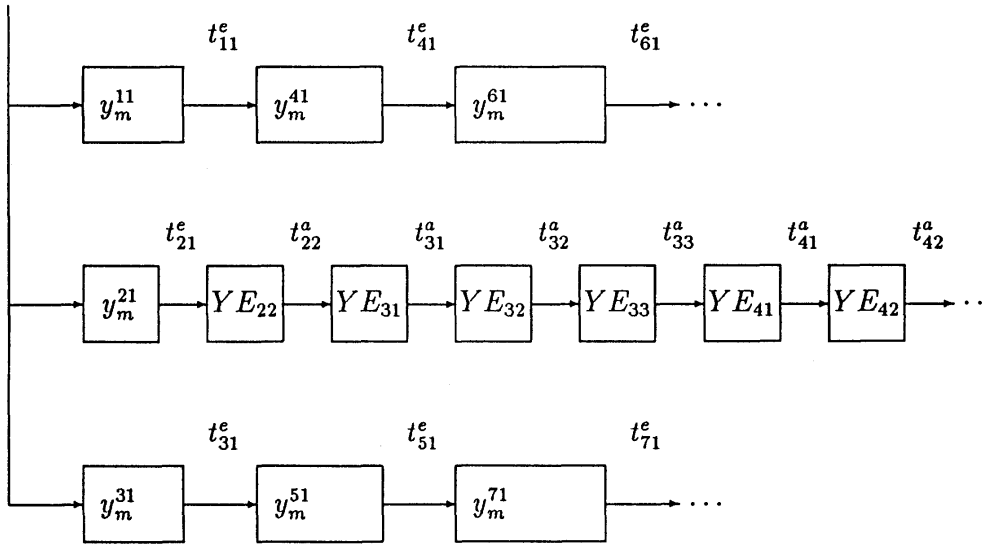
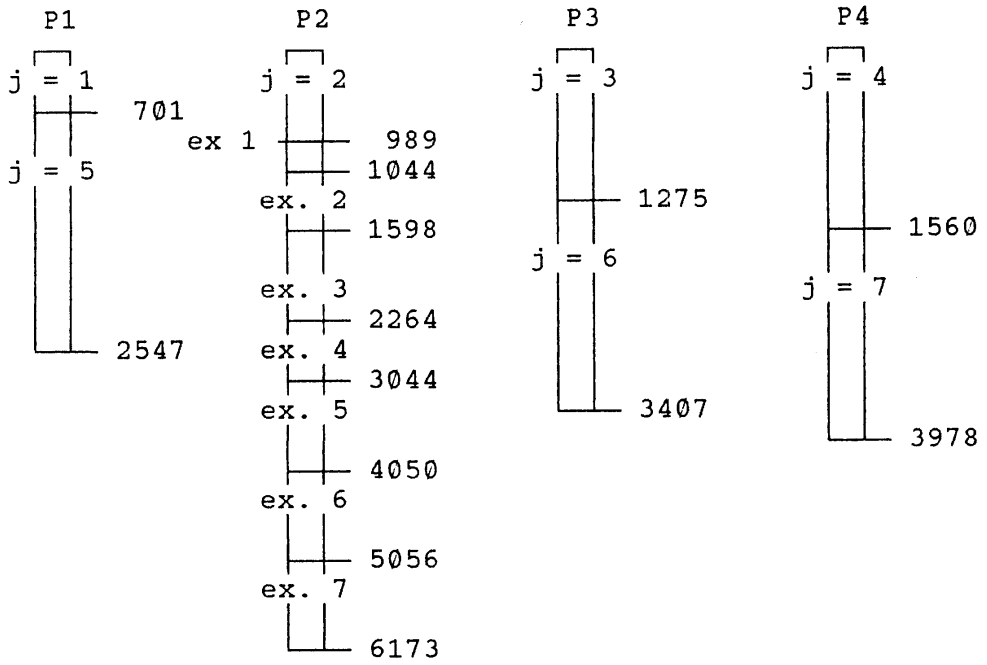
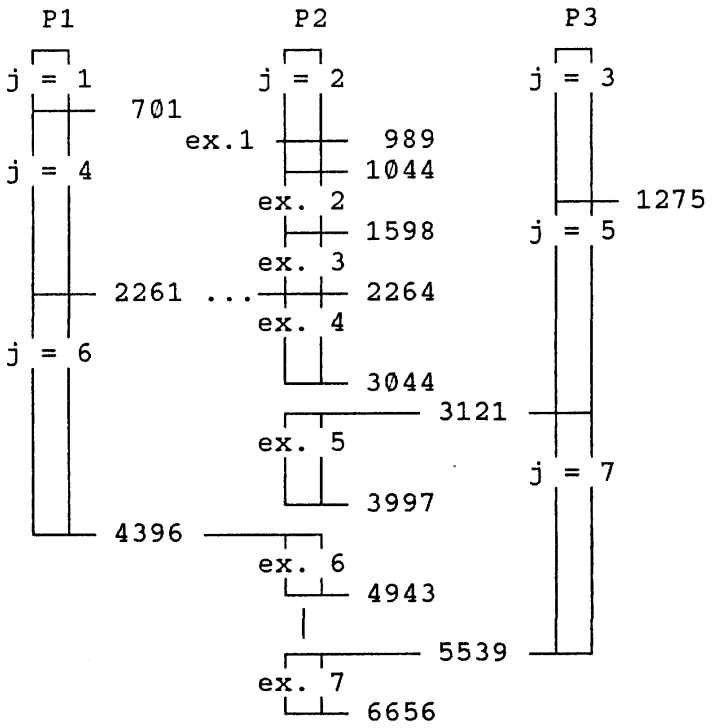

t^e_{11}
t^e_{41}
t^e_{61}
y^{11}_m
y^{41}_m
y^{61}_m
t^e_{21}
t^a_{22}
t^a_{31}
t^a_{32}
t^a_{33}
t^a_{41}
t^a_{42}
y^{21}_m
YE_{22}
YE_{31}
YE_{32}
YE_{33}
YE_{41}
YE_{42}
t^e_{31}
t^e_{51}
t^e_{71}
y^{31}_m
y^{51}_m
y^{71}_m

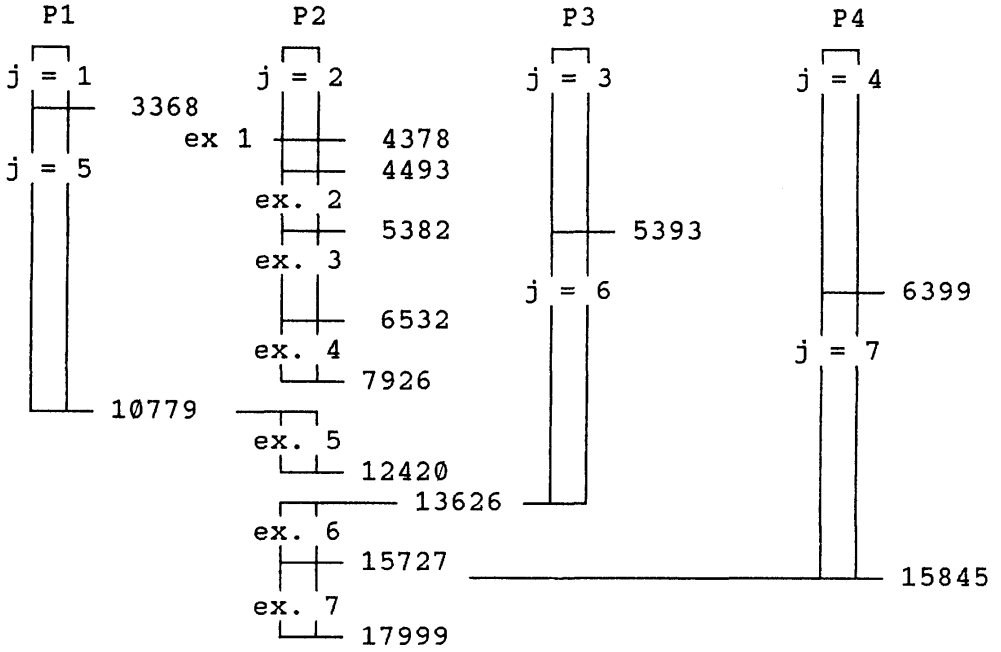
Bild 3

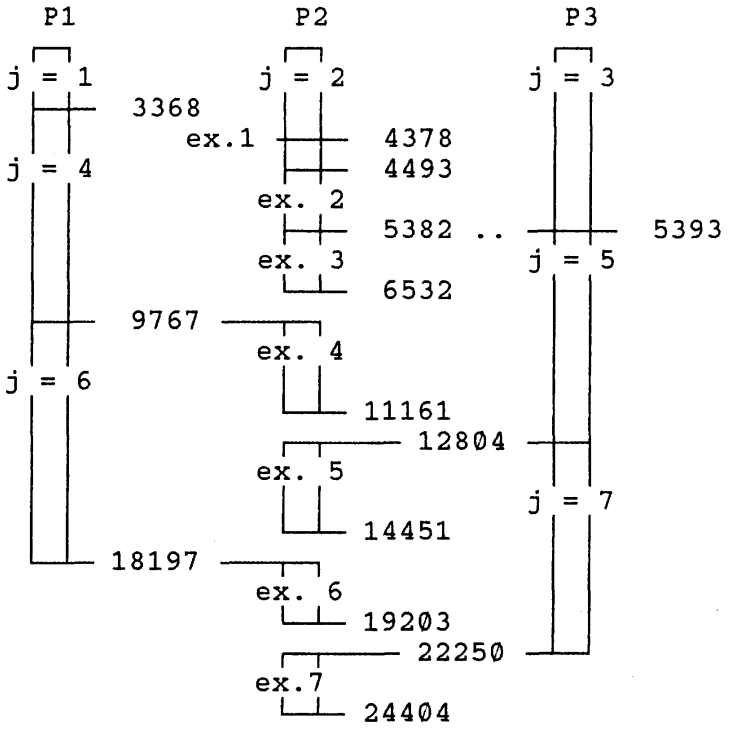
Beispiel FA1:





Beispiel FA2:

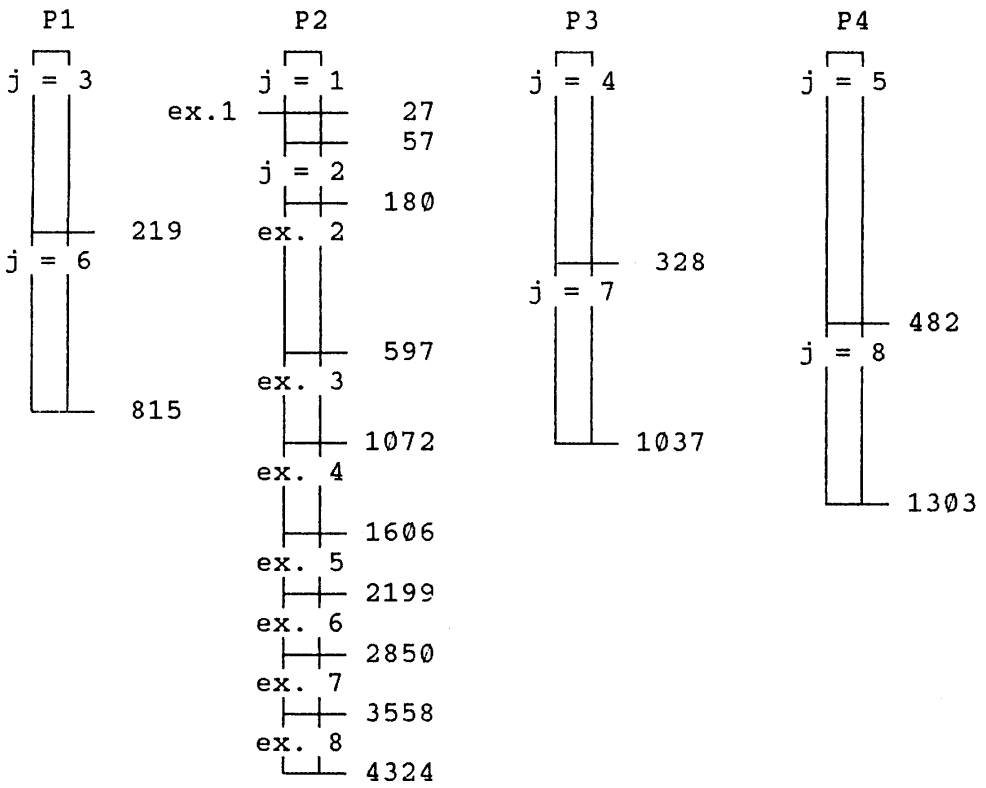




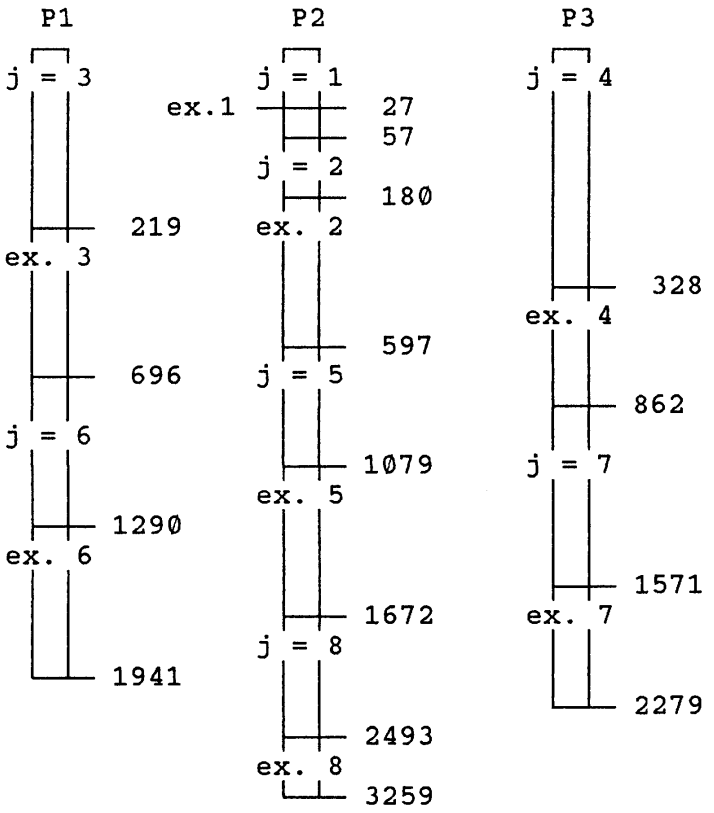
Wie die Beispiele zeigen, lohnt sich der Einsatz von vier CPUs nur dann, wenn oft bis zur maximalen Ordnung gerechnet werden muß; ansonsten sind drei CPUs ausreichend. Die Tatsache, daß der die Extrapolation durchführende Prozessor länger beschäftigt ist als die anderen und teilweise auf Eingaben warten muß, scheint es nahezulegen, die Diskretisierung und Extrapolation nach Ordnungen zusammengefaßt gemäß dem gleichen Schema auf die einzelnen Prozessoren zu verteilen, d.h. eine Parallelisierung gemäß Bild [1] vorzunehmen, bei der jeder Prozessor mehrere Ordnungen bearbeitet. Ein Vergleich der sich daraus ergebenden Ausführungszeiten mit den obigen zeigt für die beiden Beispiele jedoch nur geringe Unterschiede, und im allgemeinen ist in einer solchen Variante der Synchronisationsaufwand höher, der für diese Betrachtungen vernachlässigt wurde. Der Vorteil des obigen Schemas ist, daß eine "master/slave"-Beziehung besteht zwischen dem Prozessor, der die Extrapolation durchführt, und den übrigen Prozessoren, was die Kommunikation vereinfacht, insbesondere die Behandlung von Abbruch-Nachrichten.

Aus Bild [1] ist ersichtlich, daß in EULEX bei der Ordnung eins nur eine Initialisierung stattfindet und die eigentliche Extrapolation erst mit der Ordnung zwei beginnt. Da diese Initialisierung im allgemeinen weniger Zeit in Anspruch nimmt als die Übertragung eines Vektors $ym_{1,1}$, die etwa 54 μs dauert, sollte sie auf demselben Prozessor ablaufen, der die Ordnung zwei behandelt. Daraus ergibt sich für die Beispiele FE3 und FE5 folgender zeitlicher Ablauf mit denselben Vereinfachungen wie zuvor:

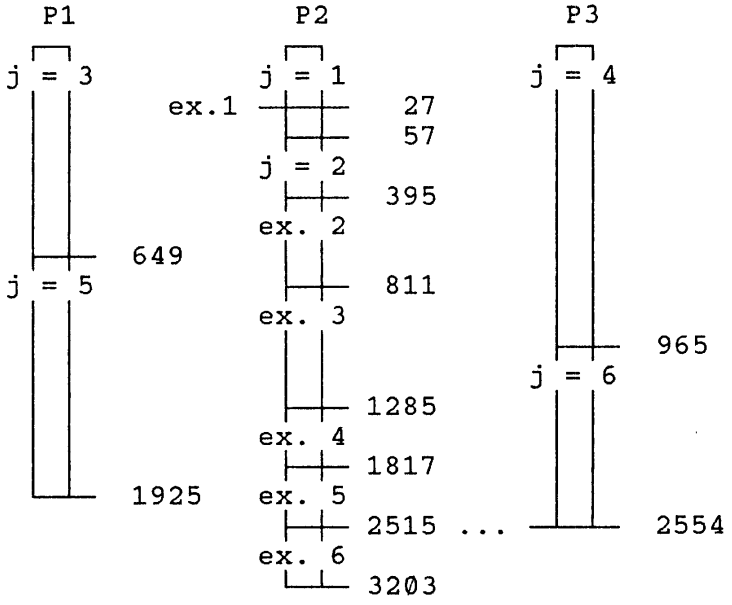
Beispiel FE3:



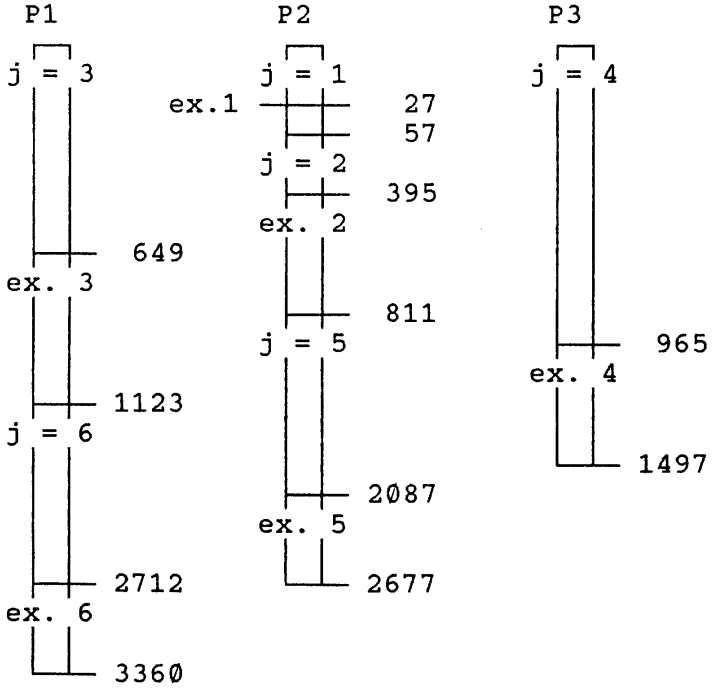
Werden nur drei Prozessoren benutzt, von denen $P1$ die Diskretisierungen für $j = 3$, $j = 5$ und $j = 7$ vornimmt und $P3$ die für $j = 4$, $j = 6$ und $j = 8$, so ändern sich die Zeiten, zu denen ein Extrapolationsergebnis auf Prozessor $P2$ anfällt, nicht, wie leicht nachprüfbar. Werden jedoch auch die Extrapolationen auf die einzelnen Prozessoren verteilt, so ergeben sich rechnerisch kürzere Ausführungszeiten, wie im folgenden Diagramm angegeben.



Beispiel FE5:



Das folgende Diagramm zeigt für dieses Beispiel die rechnerischen Ausführungszeiten bezüglich des alternativen Parallelisierungsschema:



Die beiden Beispiele unterscheiden sich dadurch, daß bei FE5 die Diskretisierung aufwendiger ist als bei FE1; dadurch ist auch das Verhältnis des Aufwandes zwischen Diskretisierung und Extrapolation bei FE5 größer als bei FE1. Das "Master–Slave"–Parallelisierungsschema ist daher auch für EULEX nicht ungünstiger als das andere, falls die F–Auswertungen aufwendiger sind als bei den Test–Beispielen.

3. Synchronisation

Für die Synchronisation der Diskretisierung mit der Extrapolation gibt es beim "Masters–Slave"–Schema zwei Möglichkeiten: sie kann auf dem Master erfolgen oder durch zusätzliche Prozesse auf den Slave–Prozessoren. Im ersten Falle sendet ein Slave ein Ergebnis sofort nach Errechnung desselben an den Master und fährt mit der nächsten Ordnung fort, wie es auch den dargestellten zeitlichen Abläufen entspricht. Auf dem Master–Prozessor ist dafür ein Empfangsprozeß vorzusehen, der die Ergebnisse solange puffert, bis der Extrapolationsprozeß sie braucht. Es kann jedoch vorkommen, daß Ergebnisse nicht mehr gebraucht werden, und zwar dann, wenn Abbruch–Nachrichten des Master sich zeitlich überschneiden mit Ergebnissen von Slave–Prozessoren; in einem solchen Falle kann ein Ergebnis noch in den Puffer des Empfangsprozesses gelangen, das vom Extrapolationsprozeß dann nicht mehr abgeholt wird. Damit solche Ergebnisse abgefangen werden können, müssen sie von gültigen unterschieden werden. Implementiert wurde ein sogenanntes Token–Passing–Verfahren, welches darin besteht, daß der Master jeden Auftrag an die Slave–Knoten, d.h. "basic step" oder Reduktion, mit einer laufenden Nummer versieht und diese mit den Ergebnissen an den Eingabeprozeß zurückgeliefert wird. Der Master setzt außerdem parallel zur Abbruch–Nachricht an die Slave–Knoten eine entsprechende Nachricht an den Empfangsprozeß ab unter Angabe der Nummer, bis zu der er keine Ergebnisse mehr erwartet, woraufhin dieser eventuell im Puffer vorhandene überflüssige Nachrichten entfernt. Der Nachteil dieses Verfahrens ist, daß zusätzliche Kontrolldaten gesendet werden müssen und daß die Pufferverwaltung etwas aufwendiger ist. Bei der zweiten Synchronisationsmethode ist auf dem Master nur ein Prozeß vorhanden und auf den Slave–Prozessoren jeweils drei Prozesse COMP, COMM und STOP. COMM ist ein Kommunikationsprozeß, der als Puffer zwischen dem Master und dem Rechenprozeß COMM wirkt wie oben beschrieben, d.h, er nimmt die Ergebnisse des Rechenprozesses entgegen und Anforderungen des Master auf Absenden derselben. Dieser Prozeß ist damit demjenigen Transputer–Link, über das Nachrichten des Master eintreffen, fest zugeordnet und muß deshalb auch alle anderen Daten, die der Mas-

ter an den Slave sendet, empfangen und an den Rechenprozeß weiterleiten. Eine Abbruch–Nachricht darf aber nicht direkt an den Rechenprozeß weitergeleitet werden, weil sonst die Möglichkeit eines "deadlock" besteht. Es kann nämlich passieren, daß zur selben Zeit eine Abbruch–Nachricht vom Kommunikationsprozeß zum Rechenprozeß und ein Ergebnis vom Rechenprozeß zum Kommunikationsprozeß zur Übertragung anstehen; in diesem Falle würde der Multi–Tasking–Scheduler endlos zwischen beiden Prozessen hin und her schalten. Wird die Abbruch–Nachricht dagegen einem anderen Prozeß STOP zugestellt, der ständig empfangsbereit ist, so kann in der beschriebenen Situation die Abbruch–Nachricht gesendet werden, danach das Ergebnis und schließlich das Stop–Signal vom STOP–Prozeß an den Rechenprozeß. Wegen der dabei auftretenden Verzögerung kann nun außerdem ein neuer Auftrag des Master beim Kommunikationsprozeß eintreffen, bevor der Rechenprozeß auf das Stop–Signal reagiert hat, d.h. zur Bearbeitung des nächsten "basic step" bereit ist. Daher ist auch eine Quittung des Rechenprozesses für das Stop–Signal nötig, nach deren Erhalt der Kommunikationsprozeß den nächsten Auftrag absenden kann. Es sind sind also einige Zustände zu unterscheiden, in denen der Kommunikationsprozeß bei Empfang einer Nachricht unterschiedlich reagiert, so daß dieser insgesamt durch ein Zustand–Übergangs–Diagramm beschrieben werden kann, wie in Bild [4] angegeben. Ein Vergleich der Ausführungszeiten bezüglich beider Synchronisationsmethoden ergab einen leichten Vorteil für die zweite Variante. Das Diagramm repräsentiert einen endlichen Automaten, der aus einer Menge von Zuständen $Z = \{Z1, Z21, Z22, Z23, Z3, Z41, Z42, Z5\}$, einer Menge von Eingaben $E = \{E1, E2, E3, E4, E5, E6\}$, einer Menge von Ausgaben $A = \{A1, A2, A3, A4, A5, NIL\}$ und einer Abbildung $\delta : D \rightarrow Z \times A$, $D \subset Z \times E$, besteht, wobei der Zustand $Z1$ als initial ausgezeichnet ist. Bezüglich der Ausführung des Kommunikationsprozesses kommt den Zuständen und den Ein– und Ausgabemengen folgende Bedeutung zu:

Z1: idle – Grundzustand,

Z2: comp – ein Auftrag wird bearbeitet,

Z21: buffer empty = TRUE & reqres = FALSE,

Z22: buffer empty = TRUE & reqres= TRUE,

Z23: buffer empty = FALSE

Z3: wait stopack – Warten auf Bestätigung eines Stop-Signals,

Z4: wait next – Puffern des nächsten Auftrages bis der laufende beendet ist,

Z41: wait next & reqres = FALSE,

Z42: wait next & reqres = TRUE,

Z5: wait term – Terminationszustand, in dem noch auf die Bestätigung eines Stop–Signals gewartet werden muß,

E1: next task main – neuer Auftrag des MAIN–Prozesses ("next basic step" oder Reduktion) eingetroffen,

E2: result COMP – Ergebnis vom Rechenprozeß COMP eingetroffen,

E3: result request – Anforderung eines Ergebnisses durch den MAIN–Prozeß,

E4: stop main – Stop–Signal vom MAIN–Prozeß eingetroffen,

E5: stop ack – Bestätigung des STOP–Signals durch den Rechenprozeß,

E6: term main – Terminationssignal vom MAIN–Prozeß eingetroffen,

A1: next task comp – Übergabe eines neuen Auftrages an den Rechenprozeß COMP,

A2: result main – Senden eines Ergebnisses an den MAIN–Prozeß,

A3: stop STOP – Senden eines Stop-Signals an den STOP–Prozeß,

A4: skip result – Entfernen eines im Puffer vorhandenen Ergebnisses,

A5: term comp stop – Senden eines Terminationssignals an den Rechen– und an den STOP–Prozeß,

NIL: leere Ausgabe.

Die Zustände Z21, Z22, Z23 sind im ersten Diagramm der Übersichtlichkeit halber in dem Zustand Z2 zusammengefaßt, ebenso die Zustände Z41 und Z42 in Z4. Die Numerierung der Pfeile mit runden Klammern gibt die jeweilige

Übergangsregel an; das ist eine der folgenden:

1)	$\delta(Z1, E1) = (Z21, A1)$,	12)	$\delta(Z3, E1) = (Z41, NIL)$
2)	$\delta(Z21, E2) = (Z23, NIL)$	13)	$\delta(Z3, E6) = (Z5, NIL)$
3)	$\delta(Z22, E2) = (Z21, A2)$	14)	$\delta(Z41, E2) = (Z41, A4)$
4)	$\delta(Z23, E2) = (Z23, NIL)$	15)	$\delta(Z41, E3) = (Z42, NIL)$
5)	$\delta(Z21, E3) = (Z22, NIL)$	16)	$\delta(Z41, E5) = (Z21, A1)$
6)	$\delta(Z23, E3) = (Z21, A2)$	17)	$\delta(Z42, E5) = (Z22, A1)$
7)	$\delta(Z21, E4) = (Z3, A3)$	18)	$\delta(Z5, E2) = (Z5, A4)$
8)	$\delta(Z22, E4) = (Z3, A3)$	19)	$\delta(Z5, E5) = (Z1, A5)$
9)	$\delta(Z23, E4) = (Z3, A3)$	20)	$\delta(Z41, E4) = (Z3, A3)$
10)	$\delta(Z3, E5) = (Z1, NIL)$	21)	$\delta(Z1, E5) = (Z1, A5)$
11)	$\delta(Z3, E2) = (Z3, A4)$.		

Die Numerierung der Pfeile mit eckigen Klammern im ersten Diagramm bezieht sich auf eine Regelmenge, nach denen die Zustandsübergänge bezüglich der Zustände Z2 und Z4 zusammengefaßt sind. Die Regelmengen sind definiert wie folgt:

$$\langle 2\rangle = \{(2), (3), (4), (5), (6)\}, \quad \langle 4\rangle = \{(14), (15)\},$$
$$\langle 3\rangle = \{(7), (8), (9)\}, \quad \langle 5\rangle = \{(16), (17)\}.$$

In den beiden unteren Diagrammen ist auch die Zugehörigkeit eines Zustandsüberganges zu einer Regelmengegekennzeichnet.

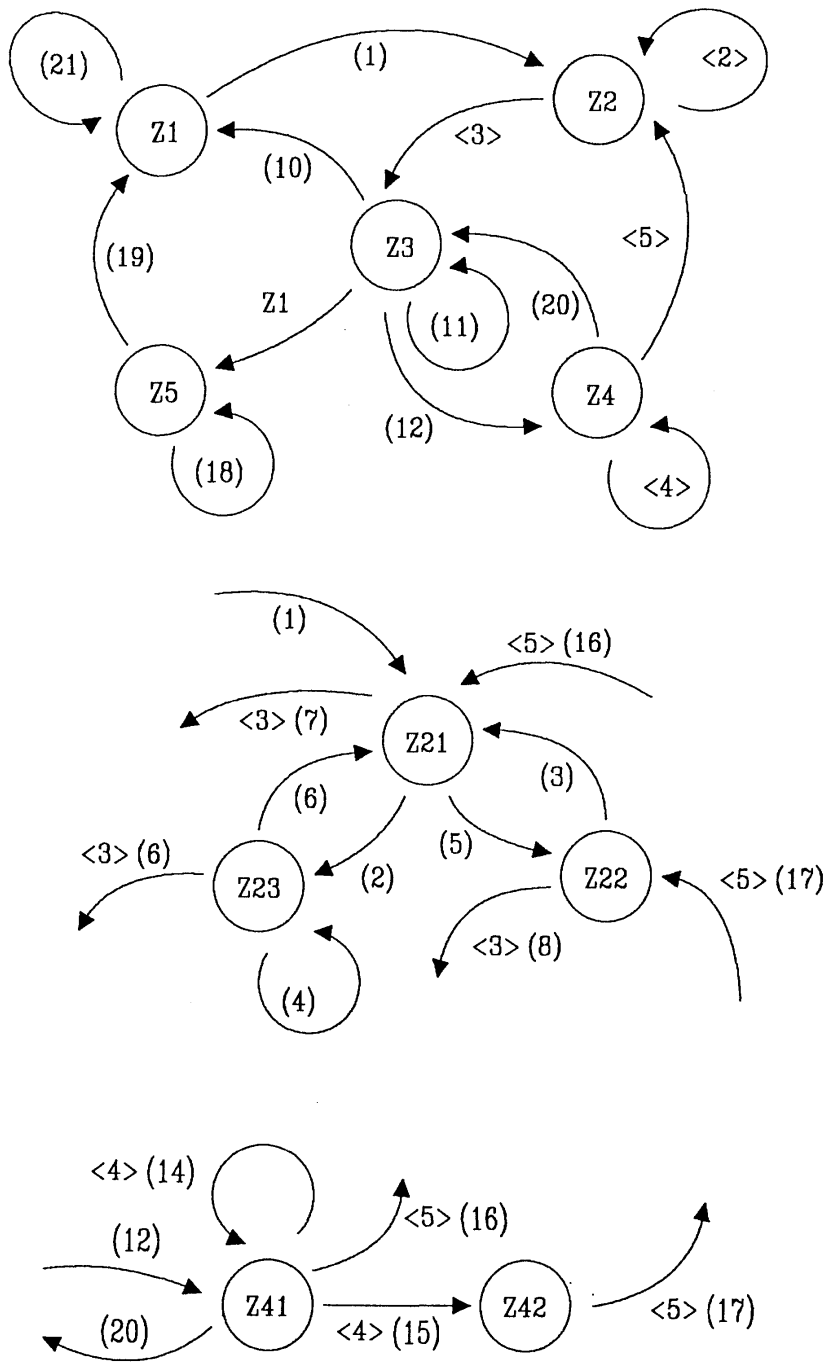


Bild 4. Zustands–Übergangsdiagramme für den Kommunikationsprozeß

4. Meßergebnisse

In den nachfolgenden Tabellen sind Ausführungszeiten in μs und erreichte Beschleunigungen angegeben, die gemessen wurden mit einer Implementierung des "Master–Slave"–Parallelisierungsschema und Synchronisation gemäß Bild [4] bei EULEX bzw. einer etwas vereinfachten Synchronisation bei EULSIM, die auch den Abbruch einer Diskretisierung im Falle, daß keine Lösung des Gleichungssystems ermittelt werden kann, berücksichtigt. Für die Messungen sind eine relative Genauigkeit *eps*, eine Anfangsschrittweite h_{ini} und eine maximale Schrittweite h_{max} vorzugeben; gewählt wurden $eps = 10^{-4}$, $h_{\text{ini}} = eps$ und $h_{\text{max}} = \text{tend} - t$.

EULEX $(eps = 10^{-4}, h_{\text{max}} = \text{tend} - t, h_{\text{ini}} = eps)$

Bsp.	SEQ	RED	3 CPUs	ACCs	ACCr	4 CPUs
FE1	123.4	108.2	90.2	1.37	1.44	90.1
FE2	356.0	331.5	268.3	1.33	1.36	270.4
FE3	427.7	394.8	292.5	1.46	1.52	291.9
FE4	33.9	21.3	26.4	1.28	1.54	26.6
FE5	70.5	52.5	48.9	1.44	1.92	50.5

EULSIM $(eps = 10^{-4}, h_{\text{max}} = \text{tend} - t, h_{\text{ini}} = eps)$

Bsp.	SEQ	3 CPUs	ACCs	ACCr	4 CPUs	ACCs	ACCr
FA1	141.3	93.8	1.51	2.01	92.7	1.52	2.06 3
FA2	438.2	295.4	1.48	1.93	243.0	1.80	2.94
FA3	182.6	112.2	1.63	2.16	109.6	1.67	2.25
FA4	560.6	374.0	1.50	2.00	304.1	1.84	3.18
FB1	740.5	437.6	1.69	2.15	406.6	1.82	2.43
FB2	212.9	130.9	1.63	2.21	126.2	1.69	2.38
FB3	228.2	148.8	1.53	2.06	137.9	1.66	2.42
FB4	362.7	227.0	1.60	2.07	202.4	1.79	2.57
FB5	1041.1	617.1	1.69	2.16	548.6	1.90	2.66
C1	155.1	99.3	1.56	2.09	97.3	1.59	2.18
C2	169.7	110.4	1.54	2.03	104.9	1.62	2.24
C3	165.6	104.7	1.58	2.10	100.4	1.65	2.27
C4	223.0	141.8	1.62	2.06	130.1	1.77	2.40
C5	247.9	149.9	1.65	2.08	135.2	1.83	2.48

Bsp.	SEQ	3 CPUs	ACCs	ACCr	4 CPUs	ACCs	ACCr
D1	230.7	135.7	1.70	2.09	132.1	1.75	2.18
D2	105.4	74.0	1.42	1.83	74.0	1.42	1.83
D3	170.6	109.6	1.56	2.06	104.8	1.63	2.25
D4	33.8	29.1	1.16	1.38	29.0	1.16	1.38
D5	27.2	26.0	1.04	1.11	26.0	1.04	1.11
D6	43.0	35.1	1.22	1.52	35.1	1.22	1.52
E1	32.3	29.6	1.09	1.29	29.6	1.09	1.29
E2	300.3	201.5	1.49	1.73	201.8	1.49	1.73
E3	124.3	81.9	1.52	1.94	80.7	1.54	1.99
E4	418.7	252.7	1.66	2.12	224.6	1.86	2.61
E5	49.0	40.6	1.21	1.54	40.6	1.21	1.54
S1	701.7	503.5	1.39	1.79	485.8	1.44	1.93
S2	602.7	360.3	1.67	2.03	317.7	1.90	2.49
S3	354.9	211.4	1.68	2.05	185.6	1.91	2.52
S4	180.9	112.4	1.61	2.00	97.5	1.85	2.56
S5	741.4	438.6	1.69	2.30	411.6	1.80	2.59
S6	1126.2	659.2	1.71	2.11	594.0	1.90	2.50
S7	1224.5	788.7	1.68	2.11	721.6	1.84	2.45
LSa	455.0	366.4	1.24	1.48	364.7	1.25	1.49
LS	69.0	52.2	1.32	1.67	52.2	1.32	1.67

Bsp.	RED	%SEQ	JAC	%SEQ
FA1	94.4	66.8	38.7	27.4
FA2	295.6	67.5	132.6	30.3
FA3	131.3	71.9	42.7	23.4
FA4	374.1	66.7	175.0	31.2
FB1	566.9	76.6	156.5	21.1
FB2	149.7	70.3	55.6	26.1
FB3	154.1	67.5	65.8	28.8
FB4	262.7	72.3	80.4	24.9
FB5	788.6	75.8	236.2	22.7

Bsp.	RED	%SEQ	JAC	%SEQ
C1	106.7	68.8	39.9	25.7
C2	117.1	69.0	43.7	32.0
C3	116.4	70.3	40.7	24.6
C4	179.9	74.3	49.8	21.7
C5	188.8	76.2	49.7	20.1
D1	182.4	79.1	39.4	17.1
D2	69.1	65.6	27.9	26.5
D3	118.7	69.5	43.3	25.4
D4	18.9	50.1	10.6	31.3
D5	12.1	44.5	8.6	31.8
D6	22.8	53.0	13.3	31.0
E1	11.7	36.3	14.0	43.2
E2	233.9	77.9	53.2	17.7
E3	87.5	70.4	28.5	23.0
E4	314.5	75.1	90.9	21.7
E5	23.9	48.8	18.3	37.3
S1	448.9	64.0	231.9	33.0
S2	476.7	79.1	113.8	18.9
S3	280.6	79.1	65.0	18.3
S4	136.9	75.7	36.6	20.2
S5	536.6	72.4	162.2	21.9
S6	887.8	78.8	219.6	19.5
S7	1019.6	77.0	280.6	21.2
LSa	274.2	60.3	148.0	32.5
LS	42.0	60.8	19.7	28.6

Bei den EULEX–Beispielen FE1 bis FE5 ist die Effizienz der Parallelisierung (= erreichte Beschleunigung/CPU–Anzahl) kleiner als 0.5, was als kritische Grenze angesehen werden kann. Es ist jedoch möglich, daß sich für andere Beispiele, bei denen die Funktionsauswertungen aufwendiger sind, günstigere Werte ergeben, so daß diese Parallelisierung nicht unbedingt zu verwerfen ist. Zu beachten ist auch, daß nicht der gesamte Code durch die Parallelisierung erfaßt wird; insbesondere gilt dies für die Initialisierung, die bei den Beispielen FE1 bis FE5 11.2 μs beträgt und bei den für EULSIM herangezogenen 5.6 μs. Die sequentielle Ausführungszeit des parallelisierten Code–Anteil wurde auch ermittelt und ist für EULSIM unter RED angegeben; die darauf bezogenene relative Beschleunigung ist unter ACCr angegeben. Für EULSIM ist zusätzlich die Zeit angegeben, die zur näherungsweisen Berechnung aller JACOBI–Matrizen

gebraucht wird, was auch in der ersten parallelisierten Version sequentiell geschieht. Zu berücksichtigen ist ferner, daß die Meßergebnisse auch von den fest vorgegebenen Parametern *eps*, h und h abhängen, die direkt oder indirekt den Rechenaufwand beeinflussen. So ist etwa bei Vorgabe einer geringen relativen Genauigkeit zu erwarten, daß öfter schon mit niedrigen Extrapolationsordnungen Konvergenz erreicht wird als bei einer höheren relativen Genauigkeit. Es wurde daher auch eine Meßreihe mit einer relativen Genauigkeit $eps = 10^{-7}$ vorgenommen unter Beibehaltung der Anfangsschrittweite $h = 10^{-4}$ und auch die Häufigkeitsverteilung der Ordnungen, bei denen Konvergenz auftritt, über alle "basic steps" bzw. Reduktionen ermittelt. Die Ergebnisse sind in den folgenden Tabellen angegeben.

EULEX $(eps = 10^{-7}, h_{\max} = \text{tend} - t, h_{\text{ini}} = 10^{-4})$

Bsp.	SEQ	RED	3 CPUs	ACCs	ACCr	4 CPUs
FE1	253.2	236.6	175.2	1.34	1.49	174.8
FE2	726.6	699.4	518.5	1.40	1.42	519.7
FE3	1172.2	1104.7	799.7	1.47	1.51	790.5
FE4	96.3	81.3	73.2	1.31	1.40	73.9
FE5	1605.3	1377.9	1207.4	1.33	1.41	1336.6

EULSIM $(eps = 10^{-7}, h_{\max} = \text{tend} - t, h_{\text{ini}} = 10^{-4})$

Bsp.	SEQ	3 CPUs	ACCs	ACCr	4 CPUs	ACCs	ACCr
FA1	400.9	226.3	1.77	2.21	210.7	1.90	2.48
FA2	1280.2	722.6	1.77	2.14	586.4	2.18	2.96
FA3	533.6	294.2	1.81	2.17	268.1	1.99	2.58
FA4	2144.4	1119.9	1.91	2.40	965.8	2.22	3.04
FB1	3189.4	1683.3	1.89	2.30	1507.7	2.12	2.71
FB2	601.2	335.3	1.79	2.38	285.0	2.11	3.42
FB3	691.8	384.3	1.80	2.21	326.0	2.12	2.89
FB4	1317.9	713.5	1.85	2.27	606.1	2.17	2.93
FB5	5011.8	2717.5	1.84	2.25	2229.9	2.25	3.07

Bsp.	SEQ	3 CPUs	ACCs	ACCr	4 CPUs	ACCs	ACCr
C1	471.9	266.7	1.77	2.21	243.9	1.93	2.56
C2	488.3	273.6	1.78	2.18	237.9	2.05	2.72
C3	525.0	296.7	1.77	2.17	259.6	2.02	2.68
C4	850.1	463.6	1.83	2.22	400.7	2.12	2.76
C5	1126.4	605.2	1.86	2.23	522.0	2.16	2.78
D1	724.5	392.6	1.85	2.22	379.7	1.91	2.33
D2	280.1	161.4	1.74	2.15	156.1	1.79	2.26
D3	416.8	238.4	1.75	2.16	210.2	1.98	2.65
D4	60.6	42.5	1.43	1.78	42.1	1.44	1.82
D5	393.9	258.4	1.52	1.75	258.7	1.52	1.75
D6	101.0	61.9	1.63	2.06	60.2	1.68	2.16
E1	171.7	107.3	1.60	2.08	95.6	1.80	2.58
E2	634.8	377.5	1.68	1.92	377.3	1.68	1.92
E3	320.7	181.1	1.77	2.18	174.9	1.83	2.30
E4	1297.5	705.0	1.84	2.23	593.7	2.19	2.89
E5	139.2	87.4	1.59	1.99	79.7	1.75	2.34
S1	617.7	339.1	1.82	2.17	283.6	2.18	2.82
S2	773.6	414.5	1.89	2.24	350.0	2.21	2.89
S3	797.2	445.0	1.79	2.11	369.5	2.16	2.76
S4	465.4	255.5	1.82	2.22	218.4	2.13	2.82
S5	1573.3	876.1	1.80	2.10	734.7	2.14	2.70
S6	4390.2	2553.8	1.72	2.11	2227.2	1.97	2.64
S7	2713.6	1484.6	1.83	2.16	1262.3	2.15	2.73
LSa	456.5	244.9	1.86	2.25	240.0	1.90	2.32
LS	175.5	108.2	1.63	2.01	105.4	1.67	2.10

Bezeichnungen in den folgenden Tabellen:

ns: Anzahl der "basic steps",

fanz: Anzahl der Funktionsauswertungen,

red: Anzahl der Reduktionsschritte,

$o2, \ldots, o10$: Anzahl der "basic steps" oder Reduktionsschritte, bei denen $2, \ldots, 9$ bzw 10 als Endordnung auftritt.

EULEX $(eps = 10^{-4}, h_{\max} = \text{tend} - t, h_{\text{ini}} = eps)$

Bsp.	FE1	FE2	FE3	FE4	FE5
ns	21	71	83	7	15
fanz	414	1153	1040	72	93
red	0	9	29	0	2
o2	2	1	4	2	8
o3	1	1	28	0	4
o4	0	7	46	2	2
o5	1	31	11	0	1
o6	5	26	11	2	1
o7	7	8	6	1	1
o8	3	6	3	0	0
o9	2	0	3	0	0
o10	0	0	0	0	0

EULEX $(eps = 10^{-7}, h_{\max} = \text{tend} - t, h_{\text{ini}} = 10^{-4})$

Bsp.	FE1	FE2	FE3	FE4	FE5
ns	27	83	216	19	454
fanz	1062	3018	3039	276	2124
red	2	17	96	3	224
o2	1	1	4	1	225
o3	1	0	89	1	449
o4	0	5	179	6	02
o5	1	1	0	5	1
o6	0	4	1	6	1
o7	1	14	0	1	0
o8	1	29	5	2	1
o9	12	8	11	0	1
o10	12	8	11	0	1

EULSIM $(eps = 10^{-4}, h_{\max} = \text{tend} - t, h_{\text{ini}} = eps)$

Bsp.	ns	fanz	red	o2	o3	o4	o5	o6	o7
FA1	13	155	0	2	2	2	7	0	0
FA2	15	227	0	2	1	12	0	0	0
FA3	14	202	0	0	0	2	12	0	0
FA4	16	275	0	0	2	13	0	1	0
FB1	52	832	0	3	4	10	14	14	7
FB2	11	167	0	1	1	1	8	0	9
FB3	13	177	0	1	1	7	4	0	0
FB4	18	287	0	1	0	9	4	3	1
FB5	47	848	0	1	4	13	11	10	8
C1	13	168	0	1	2	1	9	0	0
C2	14	163	0	3	0	5	6	0	0
C3	13	159	0	2	2	1	8	0	0
C4	16	223	0	2	1	3	6	4	0
C5	16	240	0	1	0	4	6	5	0
D1	18	323	3	3	0	2	2	13	1
D2	13	123	0	2	3	5	3	0	0
D3	14	169	0	2	1	4	7	0	0
D4	5	35	0	3	0	2	0	0	0
D5	6	30	0	3	3	0	0	0	0
D6	6	45	0	2	3	0	1	0	0
E1	4	24	0	4	0	0	0	0	0
E2	38	506	11	2	17	7	12	9	2
E3	13	157	0	2	1	5	1	4	0
E4	26	384	1	4	2	7	4	8	2
E5	6	42	0	3	3	0	0	0	0
S1	57	516	19	45	19	12	0	0	0
S2	28	459	11	1	4	22	8	3	1
S3	16	282	4	1	1	7	5	6	0
S4	9	151	0	1	1	2	0	4	1
S5	40	582	11	1	31	3	8	3	5
S6	54	903	18	10	17	15	14	8	8
S7	69	1044	21	5	38	17	13	13	4
LSA	96	542	29	86	30	3	6	0	0
LS	9	76	1	4	0	6	0	0	0

EULSIM ($eps = 10^{-7}, h_{\max} = \text{tend} - t, h_{\text{ini}} = 10^{-4}$)

Bsp.	ns	fanz	red	o2	o3	o4	o5	o6	o7
FA1	24	511	0	0	1	2	1	9	11
FA2	25	651	0	0	1	0	2	14	8
FA3	29	671	1	0	0	1	1	13	15
FA4	34	1005	3	0	0	0	4	17	16
FB1	161	3947	4	2	2	4	11	11	135
FB2	21	511	0	0	1	0	1	8	11
FB3	24	589	0	0	1	0	1	9	13
FB4	44	1127	0	0	1	0	3	9	31
FB5	168	4242	14	0	1	36	2	22	117
C1	28	586	0	0	3	0	2	11	12
C2	26	544	0	1	1	0	2	12	10
C3	29	574	0	2	1	0	4	13	9
C4	42	924	1	1	1	0	5	16	20
C5	53	1235	3	1	1	0	6	17	31
D1	49	1099	5	1	0	5	2	19	27
D2	22	401	0	0	1	4	1	10	6
D3	25	470	0	2	0	1	3	11	7
D4	6	75	1	2	1	1	1	2	0
D5	46	670	17	2	15	16	17	9	4
D6	8	141	0	1	0	1	0	4	2
E1	11	163	0	2	1	2	1	4	1
E2	58	1255	13	1	0	2	6	49	13
E3	24	467	0	0	2	1	2	10	9
E4	57	1289	4	1	1	1	4	30	24
E5	9	139	2	2	2	1	2	4	0
S1	22	541	3	0	1	1	1	12	10
S2	28	697	1	0	1	1	1	6	20
S3	28	680	6	0	1	0	2	26	5
S4	18	416	0	0	1	0	2	2	10
S5	55	1329	17	0	1	0	15	55	1
S6	208	3527	71	0	78	99	47	29	26
S7	97	2320	30	0	1	19	28	53	26
LSA	39	781	1	0	0	1	7	14	18
LS	15	238	3	4	3	2	0	6	3

Das Parallelisierungsschema scheint demnach für höhere Extrapolationordnungen günstiger zu sein als für niedrige, wie auch auf Grund der für die Beispiele FE1 bis FE5 und FA1 bis FA5 oben angegeben Einzelzeiten für Diskretisierung und Extrapolation zu erwarten ist. Als nächstes wurde daher untersucht, ob eine Verbesserung der Beschleunigung erzielt werden kann durch Änderung der Ordnungs- und Schrittweitensteuerung derart, daß der Aufwand für höhere Extrapolationsordnungen geringer bewertet wird, was auch der parallelen Ausführung angemessen ist. Die optimale Ordnung und die Schrittweite werden nämlich auf Grund einer Aufwandsabschätzung festgelegt, die eine sequentielle Ausführung der Diskretisierung voraussetzt, wie aus der näherungsweisen Festlegung des Rechenaufwandes a_j für eine Ordnung$j$ durch

$$a_j = a_{j-1} + (n_j - 1), \quad a_1 = 1 \quad \text{(EULEX)},$$

bzw.

$$a_j = a_{j-1} + n_j, \quad a = 1 + n_1 \quad \text{(EULSIM)}$$

herrvorgeht. Bei EULEX ist a_j die Anzahl der bis zur Ordnung j nötigen Funktionsauswertungen; bei EULSIM geht in a_j außerdem der Aufwand zur Lösung des linearen Gleichungssystem ein. Auf Grund des Parallelisierungsschemas ist nun der Aufwand bis zur Ordnung j zu definieren als Summe über alle Ordnungen, die auf demselben Prozessor bearbeitet werden wie Ordnung j:

- EULEX

$$\begin{aligned}
&a_1 = 1, a_2 = a_1 + (n_2 - 1), \\
&a_j = n_j - 1 && \text{falls } 3 \le j \le 4 \text{ (3 Prozessoren)} \\
& && \text{bzw. } 3 \le j \le 5 \text{ (4 Prozessoren)}, \\
&a_j = a_{j-2} + (n_j - 1) && \text{falls } j \ge 5 \text{ (3 Prozessoren) bzw.} \\
&a_j = a_{j-3} + (n_j - 1) && \text{falls } j \ge 6 \text{ (4 Prozessoren)},
\end{aligned}$$

- EULSIM

$$\begin{aligned}
&a_j = 1 + n_j && \text{falls } j \le 3 \text{ (3 Prozessoren)} \\
& && \text{bzw. } j \le 4 \text{ (4 Prozessoren)}, \\
&a_4 = a_1 + n_4 && \text{(3 Prozessoren) bzw.} \\
&a_5 = a_1 + n_5 && \text{(4 Prozessoren)}, \\
&a_j = a_{j-2} + n_j && \text{falls} j \ge 5 \text{ (3 Prozessoren) bzw.} \\
&a_j = a_{j-3} + n_j && \text{falls} j \ge 6 \text{ (4 Prozessoren).}
\end{aligned}$$

In EULEX versagt mit der geänderten Folge a_j aber die Ordnungs- und Schrittweitensteuerung, weil jene für $j \geq 2$ nicht streng monoton steigend ist. Abhilfe ist möglich durch Verwendung einer nochmals geringfügig geänderten Folge, für die die Monotonie–Bedingung erfüllt ist, oder dadurch, daß die geänderte Folge nur lokal zur Aufwandsabschätzung herangezogen wird, d.h. daß die in beiden Algorithmen vorhandenen und von der Aufwandsfolge a_j sowie der relativen Genauigkeit abhängigen globalen Fehlergrenzen $al_{j,k}$ beibehalten werden, wie es auch in EULSIM vorgesehen ist für den Fall, daß der Aufwand durch verschiedene Kostenfaktoren für die Lösung des Gleichungssysytems, die Diskretisierung und die Funktionsauswertungen erfaßt werden soll. Die globalen Fehlergrenzen werden u.a. verwendet bei der Überprüfung, ob die optimale Ordnung für den nächsten "basic step" erhöht werden kann. Diese Überprüfung geschieht, sobald Konvergenz erreicht ist; eine optimale Ordnung für den nächsten "basic step" wird zuvor auf Grund einer lokalen Aufwandsabschätzung mit dem bis zu der betreffenden Ordnung aufgetretenen Fehler ermittelt. Eine parallele EULEX–Variante mit verschiedenen streng monoton steigenden Aufwandsfolgen wurde auch getestet; es ergaben sich aber zumeist längere Ausführungszeiten. In den folgenden Tabellen sind die gemessenen Ausführungszeiten und erzielten Beschleunigungen angegeben, und zwar sowohl für parallele als auch für sequentielle Ausführung, damit auch der Einfluß der Schrittweiten–Steuerung auf die Ausführungszeiten erfaßt werden kann.

Bezeichnungen

SEQ*:	Ausführungszeit des sequentiellen Algorithmus mit geänderter Aufwandsfolge a_j,
ACCa:	auf den ungeänderten sequentiellen Algorithmus bezogene absolute Beschleunigung,
ACCs:	auf SEQ* bezogene Beschleunigung,
ACCr:	auf SEQ* bezogene relative Beschleunigung.

EULEX ($eps = 10^{-4}, h_{\max} = \text{tend} - t, h_{\text{ini}} = eps$,
a_j nochmals geändert unter Einbeziehung von al_{jk})
mit 3 Prozessoren:

$$a_1 = 1,\ a_j = n_j - 1 \quad \text{für } 2 \le j \le 4,$$
$$a_j = a_{j-2} + (n_j - 1) \quad \text{für } j > 5\ .$$

Bsp.	SEQ*	3 CPUs	ACCa	ACCs	ACCr
FE1	136.2	99.5	1.24	1.37	1.44
FE2	484.1	378.1	0.94	1.28	1.31
FE3	411.7	279.1	1.53	1.47	1.54
FE4	38.5	29.7	1.14	1.30	1.54
FE5	71.0	48.9	1.45	1.45	1.73

mit 4 Prozessoren:

$$a_1 = 1,\ a = n_j - 1 \quad \text{für } 2 \le j \le 5,$$
$$a_j = a_{j-3} + (n_j - 1) \quad \text{für } j \ge 6\ .$$

Bsp.	SEQ*	4 CPUs	ACCa	ACCs	ACCr
FE1	153.1	114.4	1.07	1.34	1.40
FE2	358.8	271.2	1.31	1.32	1.36
FE3	741.2	544.6	0.79	1.36	1.40
FE4	43.0	33.5	1.01	1.29	1.45
FE5	95.1	68.8	1.03	1.38	1.53

EULEX ($eps = 10^{-4}, h_{\max} = \text{tend} - t, h_{\text{ini}} = eps$,
a_j geändert, al_{jk} unverändert)

Bsp.	SEQ*	3 CPUs	ACCa	ACCs	ACCr	4 CPUs
FE1	131.2	97.3	1.27	1.35	1.42	97.2
FE2	359.6	272.8	1.31	1.32	1.35	275.3
FE3	405.0	277.0	1.545	1.46	1.52	276.7
FE4	33.9	25.7	1.32	1.32	1.64	25.9
FE5	70.8	48.8	1.45	1.45	1.73	50.5

EULSIM ($eps = 10^{-4}, h_{\max} = \text{tend} - t, h_{\text{ini}} = eps$, a_j und al_{jk} geändert)

Bsp.	SEQ*	3 CPUs	ACCa	ACCs	ACCr
FA1	143.4	95.0	1.49	1.51	2.01
FA2	447.9	286.6	1.63	1.56	2.14
FA3	193.3	122.2	1.49	1.58	2.11
FA4	520.3	338.1	1.66	1.54	2.07
FB1	716.1	416.7	1.78	1.72	2.16
FB2	213.2	131.5	1.62	1.62	2.21
FB3	242.1	152.5	1.50	1.59	2.16
FB4	346.9	212.8	1.70	1.63	2.54
FB5	1053.8	619.3	1.68	1.70	2.16
C1	169.0	110.5	1.40	1.53	2.06
C2	171.9	111.4	1.52	1.44	2.03
C3	177.0	113.1	1.46	1.57	2.05
C4	236.6	147.7	1.56	1.60	2.05
C5	257.7	159.5	1.55	1.62	2.06
D1	201.0	120.4	1.92	1.67	2.09
D2	115.8	77.9	1.35	1.49	1.91
D3	174.0	113.8	1.50	1.53	2.03
D4	29.9	27.7	1.22	1.08	1.20
D5	31.4	29.5	0.92	1.06	1.15
D6	49.0	39.8	1.08	1.23	1.53
E1	55.0	44.0	0.73	1.27	1.77
E2	298.5	203.4	1.48	1.46	1.70
E3	130.7	88.1	1.41	1.48	1.92
E4	420.3	253.4	1.65	1.66	2.12
E5	49.0	40.7	1.20	1.20	1.53
S1	738.0	513.3	1.37	1.44	1.89
S2	644.1	402.8	1.50	1.60	2.05
S3	332.5	207.5	1.71	1.60	2.01
S4	181.4	111.0	1.54	1.63	2.06
S5	922.8	570.1	1.30	1.62	2.06
S6	1538.3	919.1	1.28	1.67	2.10
S7	1405.1	833.7	1.59	1.69	2.10
LSa	108.2	71.4	6.37	1.51	1.90
LS	73.1	56.3	1.23	1.30	1.63

Bsp.	SEQ*	4 CPUs	ACCa	ACCs	ACCr
FA1	149.0	98.2	1.44	1.52	2.07
FA2	463.1	257.6	1.82	1.80	2.99
FA3	177.1	110.9	1.65	1.60	2.20
FA4	543.8	288.4	1.94	1.89	3.32
FB1	718.4	392.2	1.89	1.83	2.46
FB2	208.7	124.4	1.71	1.68	2.40
FB3	230.0	135.9	1.68	1.69	2.42
FB4	383.6	219.4	1.65	1.75	2.53
FB5	1041.1	547.3	1.90	1.90	2.70
C1	168.9	107.7	1.44	1.57	2.17
C2	161.9	99.1	1.71	1.63	2.27
C3	170.6	105.0	1.58	1.62	2.25
C4	236.7	135.4	1.70	1.75	2.40
C5	264.0	142.9	1.73	1.85	2.52
D1	224.8	128.8	1.79	1.75	2.20
D2	138.3	98.2	1.07	1.41	1.81
D3	171.0	108.0	1.58	1.58	2.21
D4	35.9	32.1	1.05	1.12	1.29
D5	70.3	59.1	0.46	1.19	1.36
D6	49.0	39.6	1.08	1.24	1.55
E1	60.2	48.6	0.66	1.24	1.71
E2	253.6	168.6	1.78	1.50	1.75
E3	134.3	86.8	1.43	1.55	2.01
E4	475.6	258.3	1.62	1.84	2.58
E5	55.3	45.8	1.07	1.21	1.55
S1	570.7	391.8	1.79	1.46	1.95
S2	758.9	477.5	1.26	1.59	2.13
S3	308.2	170.1	2.09	1.80	2.52
S4	195.1	109.8	1.65	1.78	2.51
S5	784.3	394.8	1.88	1.99	2.53
S6	1176.8	628.2	1.79	1.87	2.47
S7	1149.7	593.5	2.23	1.94	2.56
LSa	114.6	74.9	6.08	1.53	1.94
LS	75.1	56.8	1.22	1.32	1.68

EULSIM ($eps = 10^{-4}, h_{\max} = \text{tend} - t, h_{\text{ini}} = eps$,
al_{jk} unverändert, a_j geändert mit Kostenfaktoren
$COSTLR = 4, COSTS = 1, COSTF = 1$)

mit 3 Prozessoren:

$$
\begin{aligned}
a_1 &= (COSTF + COSTS) * (n_j + 1) + COSTLR \\
a_j &= (COSTF + COSTS) * n_j + (COSTS + COSTLR) \text{ für } 2 \leq j \leq 3 \\
a_4 &= a_1 + (COSTF + COSTS) * n_4 + (COSTS + COSTLR) \\
a_j &= a_{j-2} + (COSTF + COSTS) * n_j + (COSTS + COSTLR) \text{ für } j \geq 5
\end{aligned}
$$

Bsp.	SEQ*	3 CPUs	ACCa	ACCs
FA1	147.8	95.9	1.47	1.48
FA2	439.4	296.1	1.50	1.48
FA3	173.2	108.7	1.68	1.59
FA4	562.3	374.9	1.50	1.50
FB1	761.7	447.6	1.65	1.71
FB2	213.5	131.3	1.62	1.63
FB3	230.3	142.7	1.60	1.61
FB4	378.1	240.1	1.51	1.57
FB5	1093.5	646.3	1.61	1.69
C1	155.4	99.5	1.56	1.56
C2	170.0	110.6	1.53	1.54
C3	169.6	106.5	1.56	1.59
C4	230.4	142.1	1.62	1.62
C5	248.3	150.1	1.65	1.65
D1	226.8	132.6	1.74	1.71
D2	104.0	71.1	1.48	1.46
D3	170.9	109.8	1.55	1.56
D4	38.0	33.0	1.01	1.15
D5	27.3	26.1	1.04	1.05
D6	45.2	38.3	1.12	1.18

Bsp.	SEQ*	3 CPUs	ACCa	ACCs
E1	32.4	29.7	1.08	1.09
E2	267.3	178.2	1.68	1.50
E3	124.6	82.0	1.52	1.52
E4	455.9	272.5	1.54	1.67
E5	49.2	40.7	1.20	1.21
S1	824.5	589.8	1.19	1.40
S2	830.7	593.2	1.02	1.40
S3	355.6	211.7	1.68	1.68
S4	172.4	108.6	1.66	1.59
S5	738.1	440.3	1.68	1.68
S6	1124.3	652.9	1.72	1.72
S7	1131.3	652.4	2.03	1.73
LSa	106.9	70.7	6.44	1.51
LS	69.2	52.4	1.32	1.32

mit 4 Prozessoren:

$$
\begin{aligned}
a_1 &= (COSTF + COSTS) * (n_j + 1) + COSTLR \\
a_j &= (COSTF + COSTS) * n + (COSTS + COSTLR) \text{ für } 2 \leq j \leq 4 \\
a_5 &= a_1 + (COSTF + COSTS) * n_5 + (COSTS + COSTLR) \\
a_j &= a_{j-3} + (COSTF + COSTS) * n_j + (COSTS + COSTLR) \text{ für } j \geq 6
\end{aligned}
$$

Bsp.	SEQ*	4 CPUs	ACCa	ACCs
FA1	147.8	98.2	1.43	1.50
FA2	466.6	250.8	1.75	1.86
FA3	209.8	131.4	1.39	1.60
FA4	540.2	285.0	1.97	1.90
FB1	708.8	386.5	1.92	1.83
FB2	218.9	133.7	1.59	1.64
FB3	235.6	143.5	1.59	1.64
FB4	353.5	198.3	1.83	1.78
FB5	1065.6	552.0	1.89	1.93
C1	187.8	123.9	1.25	1.52
C2	171.7	109.1	1.56	1.57
C3	186.5	117.0	1.42	1.59
C4	264.6	154.7	1.49	1.71
C5	247.8	135.2	1.83	1.83

Bsp.	SEQ*	4 CPUs	ACCa	ACCs
D1	223.1	127.0	1.82	1.76
D2	102.8	70.9	1.49	1.45
D3	186.2	116.5	1.46	1.60
D4	33.8	29.1	1.16	1.16
D5	27.2	26.1	1.04	1.04
D6	42.7	35.1	1.22	1.21
E1	32.3	29.6	1.09	1.09
E2	283.5	197.1	1.52	1.44
E3	125.9	81.6	1.52	1.54
E4	529.7	277.6	1.51	1.91
E5	49.1	40.6	1.21	1.21
S1	530.9	345.0	2.03	1.54
S2	599.8	350.9	1.72	1.71
S3	45.5	193.0	1.84	1.79
S4	177.2	96.7	1.87	1.83
S5	769.6	421.0	1.76	1.83
S6	886.1	440.9	2.55	2.01
S7	1018.2	515.7	2.57	1.97
LSa	108.2	70.7	6.43	1.53
LS	69.0	52.3	1.32	1.32

Aus den Ergebnissen ist ersichtlich, daß gegenüber der ersten parallelen Version höhere oder geringere Ausführungszeiten bedingt sind durch den Eingriff in die Ordnungs- und Schrittweiten-Steuerung als Folge der Änderung der Aufwandskoeffizienten. Daß der Parallelisierungseffekt im wesentlichen unverändert ist, zeigt sich an den Werten für die (relative) Beschleunigung ACCs bzw. ACCr.

5. Parallelisierung der JACOBI–Matrix–Berechnung

In EULSIM kann zusätzlich zu Diskretisierung und Extrapolation auch die Berechnung der JACOBI-Matrix $\partial F/\partial y$ parallelisiert werden, die bis zu einem Drittel der Ausführungszeit beansprucht, wie aus den unter %JAC angegebenen Werten in den obigen Tabellen ersichtlich. Eine JACOBI-Matrix wird pro "basic step" einmal berechnet; zwischen dieser Prozedur und dem Teil "Diskretisierung und Extrapolation" bestehen Datenabhängigkeiten, da einer-

seits die Elemente der JACOBI–Matrix in die Diskretisierung eingehen und andererseits das Resultat der Extrapolation im nächsten "basic step" zur Berechnung der JACOBI–Matrix gebraucht wird; dieser sequentielle Ablauf muß daher beibehalten werden. Die JACOBI–Matrix wird durch den sequentiellen Code in einer äußeren Schleife spaltenweise aufgebaut, ohne daß zwischen den einzelnen Durchläufen Datenabängigkeiten bestehen; diese können daher auf mehrere Prozessoren verteilt werden unter Einbeziehung derjenigen, die die Diskretisierung und Extrapolation durchführen. Aus Effizienzgründen sollten auch nicht mehr Prozessoren verwendet werden, so daß vier Prozessoren insgesamt ausreichen. Ein Prozessor hat dann im allgemeinen mehrere – in unserer Implementierung aufeinander folgende – Spalten der JACOBI–Matrix zu berechnen, verwendet dazu aber stets dieselben Eingangsdaten, nämlich die Nummer der Start–Spalte und Vektoren *y*, *dz*, *sm* und *eta*, bestehend aus Näherungen der Lösung bzw. deren Ableitung an der Stelle t und zugehöriger Skalierungen; diese werden daher pro "basic step" nur einmal an alle Prozessoren übermittelt. Als Ergebnis liefert jeder Prozessor eine Teilmatrix; die Teile werden auf dem Master zur Gesamt–Matrix zusammengesetzt, was einen in OCCAM nicht vermeidbaren overhead ergibt, da Matrizen elementweise, zeilenweise oder als ganze über Kanäle übertragen werden können, nicht aber spaltenweise, wie es hier am günstigsten wäre. Zur Implementierung wurde die erste parallelisierte Version erweitert, d.h. die Aufwandsfaktoren wurden aus der sequentiellen Version übernommen und es wurden keine speziellen Kostenfaktoren herangezogen. In der folgenden Tabelle sind die gemessenen Ausführungszeiten und Beschleunigungen angegeben, insbesondere auch die nur auf die JACOBI–Matrix–Berechnungen bezogene relative Beschleunigung ACCr, sowie für jedes Beispiel die Dimension n des Differentialgleichungssystems.

EULSIM $(eps = 10^{-4}, h_{\max} = \text{tend} - t, h_{\text{ini}} = eps)$

Bsp.	n	3 CPUs	ACC	ACCr	4 CPUs	ACC	ACCr
FA1	4	86.2	1.64	1.79	81.2	1.74	1.98
FA2	9	259.9	1.71	2.12	192.9	2.27	2.61
FA3	4	103.6	1.76	1.80	96.5	1.89	2.24
FA4	10	316.6	1.77	2.33	244.3	2.30	2.40
FB1	4	406.0	1.82	2.05	358.8	2.06	2.23
FB2	6	117.0	1.82	2.05	111.7	1.91	2.07
FB3	6	132.3	1.72	2.03	120.7	1.89	2.07
FB4	6	204.1	1.78	2.04	178.7	2.03	2.05
FB5	6	557.0	1.87	1.81	486.0	2.14	2.06

C1	4	91.2	1.70	1.81	85.1	1.82	2.25
C2	4	101.4	1.67	1.82	91.1	1.86	2.27
C3	4	96.3	1.72	1.82	87.6	1.89	2.27
C4	4	131.6	1.75	1.81	114.4	2.01	2.26
C5	4	139.7	1.77	1.87	119.4	2.08	2.26
D1	3	127.8	1.80	1.82	124.5	1.85	1.85
D2	3	68.4	1.54	1.81	68.8	1.53	1.80
D3	4	100.7	1.69	1.79	91.4	1.87	2.25
D4	3	27.4	1.23	1.38	27.5	1.23	1.77
D5	2	26.2	1.03	1.86	26.3	1.03	1.37
D6	3	32.5	1.32	1.89	32.6	1.32	1.85
E1	4	26.6	1.21	1.35	25.1	1.28	2.37
E2	2	201.3	1.49	1.86	202.6	1.48	1.35
E3	3	75.9	1.64	1.89	75.0	1.66	1.84
E4	4	232.3	1.80	1.85	194.3	2.15	2.38
E5	4	36.9	1.33	1.78	35.3	1.39	2.21
S1	4	460.0	1.53	1.78	417.4	1.68	2.09
S2	4	339.3	1.78	1.78	284.2	2.12	2.09
S3	4	199.5	1.78	1.78	166.5	2.13	2.10
S4	4	105.7	1.71	1.80	88.9	2.08	2.12
S5	4	408.2	1.82	1.78	363.8	2.04	2.09
S6	4	617.9	1.82	1.78	529.1	2.13	2.09
S7	4	735.9	1.80	1.78	639.1	2.07	2.09
LSa	2	362.4	1.26	1.38	361.6	1.26	1.38
LS	3	48.4	1.43	1.84	48.9	1.42	1.82

6. Anwendung auf partielle Differentialgleichungen

Nach den Beispielen des Test–Set aus H83.ALL.STITEST wurden zum Testen von EULSIM noch zwei etwas umfangreichere Beispiele (Beispiel 1 aus HINDMARSH [2] und Beispiel F aus SINOVEC/MADSEN [3]) herangezogen, bei denen das System gewöhnlicher Differentialgleichungen durch Anwendung der Linienmethode auf ein System partieller Differentialgleichungen hervorgeht, das die Produktion von Ozon in der Stratosphäre beschreiben soll bzw. den Einfluß auf dieselbe durch den NO– und NO_2–Ausstoß von Überschallflugzeugen. Mit den

Bezeichnungen

$z_1 = z_1(t,x)$ für die O_1–Konzentration (in mol/cm)
$z_2 = z_2(t,x)$ für die O_3–Konzentration,
$z_3 = z_3(t,x)$ für die NO–Konzentration,
$z_4 = z_4(t,x)$ für die NO_2–Konzentration,
t für die Zeit(in $s, t \geq 0$, $t \leq 24h$) und
x für die Höhe (in km bei Beispiel OZ1 mit $30 \leq x \leq 50$,
bei Beispiel 2 normalisiert mit $0 \leq x \leq 1$)

sind die Systeme partieller Differentialgleichungen gegeben durch

OZ1:

$$\partial z_1/\partial t = D\partial^2 z_1/\partial x^2 + f_1(z_1, z_2) ,$$
$$f_1(z_1, z_2) = k_0 - k_1 z_1 - k_2 z_1 z_2 + k_3 z_2,$$
$$\partial z_2/\partial t = D\partial^2 z_2/\partial x^2 + f_2(z_1, z_2) ,$$
$$f_2(z_1, z_2) = k_1 z_1 - k_2 z_1 z_2 - k_3 z_2 ,$$
$$D = 3 * 10^{-5} , \quad k_0 = 1.48 * 10^7 , \quad k_1 = 7.4 ,$$
$$k_2 = 5 * 10^{-16} , \quad k_3 = 5 * 10^{-4}$$

mit den Randbedingungen

$$\partial z_1/\partial x(t,30) = 0 , \quad \partial z_1/\partial x(t,50) = 0 ,$$
$$\partial z_2/\partial x(t,30) = 0 , \quad \partial z_2/\partial x(t,50) = 0$$

und den Anfangsbedingungen

$$\begin{aligned} 10^{-6} z_1(0,x) &= 10^{-12} z_2(0,x) \\ &= 1 - (x/10 - 4)^2 + (x/10 - 4)^4/2 \end{aligned}$$

OZ2:

$$\begin{aligned} \partial z_1/\partial t &= D\partial^2 z_1/\partial x^2 + f_1(z_1, z_2, z_3, z_4) , \\ f &= a_1 - a_2 z_1 + a_3 z_2 + a_3 z_2 + a_4 z_4 - a_5 z_1 z_2 - a_6 z_1 z_4 , \\ \partial z_2/\partial t &= D\partial^2 z_2/\partial x^2 + f_2(z_1, z_2, z_3, z_4) , \\ f_2 &= b_1 z_1 - b_2 z_2 + b_3 z_1 z_2 - b_4 z_2 z_3 \\ \partial z_3/\partial t &= D\partial^2 z_3/\partial x^2 + f_3(z_1, z_2, z_3, z_4) , \\ f_3 &= -c_1 z_3 + c_2 z_4 + c_3 z_1 z_4 - c_4 z_3 + 800 + SST1 \\ \partial z_4/\partial t &= D\partial^2 z_4/\partial x^2 + f_4(z_1, z_2, z_3, z_4) , \\ f_4 &= -d_1 z_4 + d_2 z_2 z_3 - d_3 z_1 z_4 + 800 , \end{aligned}$$

$$D = 10^{-9},\ a_1 = 4 * 10^5,\ a_2 = 272.443800016,\ a_3 = 10^{-4},$$
$$a_4 = 0.007,\ a_5 = 3.67 * 10^{-16},\ a_6 = 4.13 * 10^{-12},$$
$$b_1 = 272.4438,\ b_2 = 1.00016 * 10^{-4},\ b_3 = 3.67 * 10^{-16},$$
$$b_4 = 3.57 * 10^{-15},\ c_1 = 1.6 * 10^{-8},\ c_2 = 0.007,$$
$$c_3 = 4.1283 * 10^{-12},\ c_4 = 3.57 * 10^{-15},\ d_1 = 7.000016 * 10^{-3},$$
$$d_2 = 3.57 * 10^{-15},\ d_3 = 4.1283 * 10^{-12},$$
$$SST1 \begin{cases} 3250 & \text{falls } 0.475 \leq x \leq 0.575 \\ 360 & \text{sonst} \end{cases}$$

mit den Randbedingungen

$$\partial z_i/\partial x(t,0) = 0,\ \partial z_i/x(t,1) = 0,\ \ 1 \leq i \leq 4$$

und den Anfangsbedingungen

$$z_1(0,x) = 1.306028 * 10^6, \quad z_2(0,x) = 1.076508 * 10^{12},$$
$$z_3(0,x) = 6.457715 * 10^{10}, \quad z_4(0,x) = 3.542285 * 10^{10}.$$

In beiden Fällen wird die Höhe x unterteilt in äquidistante Gitterpunkte x_j, $j = 1, \ldots m$, wobei

$$x_j = 30 + (j-1)\Delta x,\ \ \Delta x = 20/50,\ \ m = 51 \text{ für } OZ1 \text{ bzw.}$$
$$x_j = (j-1)\Delta x,\ \ \Delta x = 1/30,\ \ m = 31 \text{ für } OZ2,$$

und an diesen Stellen zum Einsetzen in das gegebene System eine Approximation der Ableitungen durch Differenzenquotienten vorgenommen derart, daß die erste Ableitung an der Stelle x ersetzt wird durch den Differenzenquotienten bezüglich der Nachbarpunkte x_{j-1} und x_{j+1} und die zweite Ableitung an der Stelle x_j durch einen Differenzenquotienten aus Näherungen für die erste Ableitung an den Halbierungspunkten bezüglich des rechten und linken Nachbarpunktes, als Näherungen werden dabei wiederum Differenzenquotienten bezüglich x_j und des rechten bzw. linken Nachbarpunktes herangezogen. Die Randbedingungen werden durch zusätzliche Punkte x_0 und x_{m+1} erfaßt, für die $z_i(t,x_0) = z_i(t,x_2)$ und $z_i(t,x_{m+1}) = z_i(t,x_{m-1})$, $i = 1,\ldots,n$ ($n = 2$ für $OZ1$, $n = 4$ für $OZ2$) angesetzt wird. Man erhält so das folgende System

gewöhnlicher Differentialgleichungen:

$$\begin{aligned}
\partial z_{ij}/\partial t &= D/(\Delta x)^2(z_{i,j+1} - 2z_{ij} + z_{i,j-1}) \\
&\quad + f_i(z_1 = z_{1j}, z_2 = z_{2j}, \ldots z_n = z_{nj}) \text{ für } 2 \leq j \leq m-1 \,, \\
\partial z_{i1}/\partial t &= -2D/(\Delta x)^2(z_{i,1} - z_{i2}) \\
&\quad + f_i(z_1 = z_{11}, z_2 = z_{21}, \ldots z_n = z_{n1}) \text{ für } j = 1 \,, \\
\partial z_{im}/\partial t &= 2D/(\Delta x)^2(-z_{i,m-1} + z_{im}) \\
&\quad + f_i(z_1 = z_{1m}, z_2 = z_{2m}, \ldots z_n = z_{nm}) \text{ für } j = m \,,
\end{aligned}$$

wobei $z_{ij} = z_i(t, x = x_j)$, $i = 1, \ldots, n$, $j = 1, \ldots, m$.
Mit den Bezeichnungen

$$\begin{aligned}
Y = (y_1, y_2, \ldots, y_{n \cdot m})^T &= (z_{11}, z_{21}, \ldots, z_{n1}, z_{12}, z_{22}, \ldots z_{n2} \,, \\
&\qquad \ldots z_{1m}, z_{2m}, \ldots, z_{nm})^T \,,
\end{aligned}$$

$$G = (g_1, g_2, \ldots, g_{n \cdot m})^T = \begin{bmatrix}
f_1(z_1 = z_{11}, z_2 = z_{21}, \ldots, z_n = z_{n1}) \\
f_2(z_1 = z_{11}, z_2 = z_{21}, \ldots, z_n = z_{n1}) \\
\ldots \\
f_n(z_1 = z_{11}, z_2 = z_{21}, \ldots, z_n = z_{n1}) \\
f_1(z_1 = z_{12}, z_2 = z_{22}, \ldots, z_n = z_{n2}) \\
f_2(z_1 = z_{12}, z_2 = z_{22}, \ldots, z_n = z_{n2}) \\
\ldots \\
f_n(z_1 = z_{12}, z_2 = z_{22}, \ldots, z_n = z_{n2}) \\
\ldots \\
\ldots \\
f_1(z_1 = z_{1m}, z_2 = z_{2m}, \ldots, z_n = z_{nm}) \\
f_2(z_1 = z_{1m}, z_2 = z_{2m}, \ldots, z_n = z_{nm}) \\
\ldots \\
f_n(z_1 = z_{1m}, z_2 = z_{2m}, \ldots, z_n = z_{nm})
\end{bmatrix}$$

sieht das Gleichungssystem in Matrixschreibweise folgendermaßen aus:

$$\partial Y/\partial t = A * Y + G \,.$$

Die Matrix A ist eine Block–Band–Matrix mit dem Aufbau

$$A = \begin{bmatrix} B & C & 0 & & & & 0 \\ C & B & C & 0 & & & 0 \\ & & + & * & + & & \\ & & & + & * & + & \\ 0 & & & 0 & C & B & C & 0 \\ 0 & & & & 0 & C & B & C \\ 0 & & & & & 0 & C & B \end{bmatrix}_{(m \times m)},$$

$$B = -2C,$$

$$C = \begin{bmatrix} D/\Delta x^2 & & 0 & & & & 0 \\ & D/\Delta x^2 & 0 & & & & 0 \\ & & * & & & & \\ & & & * & & & \\ 0 & & 0 & & D/\Delta x^2 & & 0 \\ 0 & & & & 0 & & D/\Delta x^2 \end{bmatrix}_{(n \times n)}.$$

In dieser Form können die beiden Beispiele ebenso in das Programm eingefügt werden wie die Beispiele des Test–Set H83.ALL.STITST. Wegen der Band–Form der Eingangsmatrix und der daraus resultierenden ebensolchen der JACOBI–Matrix wurden die Prozeduren zur Lösung des linearen Gleichungssystems (LINGL, SUBST) und zur Berechnung der JACOBI–Matrix ersetzt durch solche für Band–Matrizen (DGBFA, DGBSL), die aus dem in der ZIB–Sammlung H83.ALL.PROG1 vorhandenem Programm LIMEX übernommen wurden; dazu gehört insbesondere die dort beschriebene Form der Abspeicherung von Band–Matrizen durch mb Zeilen und n Spalten mit $mb = 2 * ml + mu + 1$, wobei ml die untere ($ml = 2$ für $OZ1$, $ml = 4$ für $OZ2$) und mu die obere Bandbreite ($mu = 2$ für $OZ1$, $mu = 4$ für $OZ2$) ist. Im übrigen sind Diskretisierung und Extrapolation ebenso parallelisiert wie oben beschrieben. In den folgenden Tabellen sind Meßergebnisse angegeben; in der betreffenden Implementierung wird die JACOBI–Matrix sequentiell berechnet.

Beispiel $OZ1$ ($eps = 10^{-4}, h = eps, hmax = \text{tend} - t$, al_{jk} und a_j unverändert)

tend	3 h	6 h	9 h	12 h
SEQ	4910	4910	5013	5283
nstep	13	13	13	14
fanz	163	163	166	175
RED	3939	3939	4043	4239
% SEQ	80.2	80.2	80.6	80.2
JAC	900	900	900	969
% SEQ	18.3	18.3	18.0	18.3
3 CPUs	2691	2691	2786	2907
ACCs	1.82	1.82	1.81	1.82
ACCr	2.29	2.29	2.23	2.28
4 CPUs	2178	2178	2191	2322
ACCa	2.25	2.25	2.29	2.28
ACCr	3.26	3.26	3.31	3.32

Beispiel $OZ2$ ($eps = 10^{-4}, h = 10^{-10}, hmax = \text{tend} - t$, al_{jk} und a_j unverändert)

tend	3 h	6 h	9 h	12 h
SEQ	3809	8973	15770	153686
nstep	9	18	29	198
fanz	104	217	364	3047
RED	2530	6434	11694	125981
% SEQ	66.4	71.7	74.1	82.0
JAC	1208	2405	3869	26361
% SEQ	31.7	26.8	24.5	17.2
3 CPUs	2381	5330	9094	82025
ACCa	1.60	1.68	1.73	1.87
ACCr	2.30	2.31	2.33	2.32
4 CPUs	2252	5047	8330	72275
ACCa	1.69	1.78	1.89	2.13
ACCr	2.60	2.57	2.75	2.83

Beispiel $OZ1$ ($eps = 10^{-4}, h = eps, h\max = \text{tend} - t$, al_{jk} und a_j ebenso geändert wie für die Beispiele des Test–Set H83.ALL.STITST)

tend	3 h	6 h	9 h	12 h
SEQ*3	4910	4910	5014	5014
3 CPUs	2694	2694	2789	2789
ACCa	1.82	1.82	1.80	1.89
ACCs	1.82	1.82	1.80	1.80
SEQ*4	5432	5432	5702	5703
4 CPUs	2412	2412	2412	2543
ACCa	2.04	2.04	2.08	2.08
ACCs	2.25	2.25	2.36	2.24

Beispiel $OZ2$ ($eps = 10^{-4}, h = 10^{-10}, h\max = \text{tend} - t$, al_{jk} und a_j ebenso geändert wie für die Beispiele des Test–Set H83.ALL.STITST)

tend	3 h	6 h	9 h	12 h
SEQ*3	3993	6358	9278	18311
3 CPUs	2503	3820	5373	10182
ACCa	1.52	2.35	2.93	15.1
ACCs	1.60	1.66	1.73	1.80
SEQ*4	3996	6362	8143	34443
4 CPUs	2503	3692	4452	16798
ACCa	1.52	2.43	3.54	9.15
ACCs	1.52	2.28	2.79	2.05

Beispiel $OZ1$ ($eps = 10^{-4}, h = eps, h\max = \text{tend} - t$, nur a_j geändert mit Kostenfaktoren wie für die Beispiele des Test-Set H83.ALL.STITST)

tend	3 h	6 h	9 h	12 h
SEQ*3	4910	4910	4910	5284
3 CPUs	2693	2693	2789	2910
ACCa	1.82	1.82	1.80	1.82
ACCs	1.82	1.82	1.80	1.82
SEQ*4	4910	4910	5014	5284
4 CPUs	2180	2180	2192	2324
ACCa	2.23	2.25	2.29	2.27
ACCs	2.25	2.25	2.29	2.27

Beispiel $OZ2$ ($eps = 10^{-4}, h = 10^{-10}, h\max = \text{tend} - t$, nur a_j geändert mit Kostenfaktoren wie für die Beispiele des Test-Set H83.ALL.STITST)

tend	3 h	6 h	9 h	12 h
SEQ*3	3810	8801	11936	191361
3 CPUs	2382	5297	7058	101058
ACCa	1.60	1.69	2.23	1.52
ACCs	1.60	1.66	1.69	1.89
SEQ*4	3810	8433	12415	18128
4 CPUs	2254	4717	6757	9274
ACCa	1.69	1.90	2.33	16.6
ACCs	1.69	1.79	1.84	1.95

Die für Beispiel $OZ2$ mit vier Prozessoren nach Änderung der Aufwandsfaktoren erzielten Beschleunigungen um den Faktor 9 bzw. 16 sind offenbar bedingt durch den Eingriff in die Ordnungs- und Schrittweitensteuerung. Eine genauere Untersuchung ergab, daß im Algorithmus ohne Änderung der Aufwandsfaktoren 198 "basic steps" mit 3047 Funktionsauswertungen durchgeführt werden, wobei insgesamt 95 mal die Schrittweite reduziert wird. In der Version mit Kostenfaktoren für die parallele Bearbeitung durch vier Prozessoren werden jedoch nur 29 "basic steps" mit 395 Funktionsauswertungen durchgeführt, wobei nur 7 mal die Schrittweite reduziert wird. Da in die Bestimmung der nächsten Schrittweite außer der Aufwandsabschätzung auch das Verhältnis $h/h\max$ eingeht, wurden mit der Version ohne Änderung der Aufwandsfaktoren weitere Messungen

vorgenommen mit verschiedenen Werten für h max. Als Ergebnis stellte sich heraus, daß es eine optimale Schrittweite h max gibt, für die die Ausführungszeit minimal ist, und zwar gegenüber dem Standard-Fall h max = tend $-t$ um einen Faktor, der wesentlich günstiger ist, als durch Parallelisierung erreichbar. Die Meßergebnisse sind in der folgenden Tabelle enthalten.

tend [h]	h max [min]	SEQ [ms]	3 CPUs	ACC [ms]	4 CPUs	ACC
3	180	3809	2381	1.60	2252	1.69
3	150	4351	2688	1.61	2525	1.72
3	120	4374	2702	1.58	2539	1.72
3	90	4168	2515	1.66	2516	1.66
3	60	5180	2938	1.62	2940	1.62
6	360	8973	5330	1.68	5047	1.78
6	180	7759	4572	1.70	4135	1.88
6	150	5947	3420	1.74	3275	1.82
6	120	5709	3407	1.68	3080	1.85
6	90	5892	3410	1.73	3248	1.81
6	60	6849	3962	1.73	3963	1.73
9	540	15770	9094	1.73	8330	1.89
9	180	12017	6878	1.74	6088	1.97
9	150	7488	3879	1.93	3753	2.00
9	120	7488	4254	1.76	3817	1.96
9	90	7402	4184	1.77	3857	1.92
9	60	12387	6783	1.83	6650	1.86
12	720	153686	82025	1.87	72275	2.13
12	180	144035	77514	1.87	67899	2.13
12	150	64131	36095	1.78	31485	2.04
12	140	42612	24490	1.74	21170	2.01
12	135	8497	4712	1.80	4294	1.98
12	130	12438	10520	1.75	9293	1.98
12	120	33621	19229	1.75	16933	1.99
12	90	127257	67701	1.88	59420	2.14
12	60	155202	82020	1.89	72879	2.13

Die Berechnung der JACOBI-Matrix läßt sich auch für die vorliegende Form der Band-Matrix-Abspeicherung derart parallelisieren, daß die Erzeugung der Spalten auf mehrere Prozessoren verteilt wird. Jeder Prozessor erhält dazu die Nummer einer Start-Spalte $scol$ und einen Wiederholungsfaktor r und liefert

Serien von Spalten als Ergebnis, deren erste mit der Start–Spalte $scol$ beginnt und weiter die Spalten $scol + mm$, $scol + 2 * mm, \ldots, scol + k * m$ enthält, wobei k bestimmt ist durch $scol + k * mm \leq n$ und $mm = ml + mu + 1$ die Bandbreite ist. Der Wiederholungsfaktor r gibt an, wie viele Serien von Spalten zu berechnen sind, beginnend jeweils mit $scol, scol + 1, \ldots, scol + (r - 1)$, und ist bestimmt durch $r \leq [mm/p] + 1$, wobei p die Anzahl der Prozessoren ist. In den folgenden Tabellen sind die mit einer solchen Implementierung gemessenen Ergebnisse angegeben mit den Standard–Parametern $eps = 10^{-4}$ und $h = eps$ für $OZ1$, $eps = 10^{-4}$ und $h = 10^{-10}$ für $OZ2$, $h\max = \text{tend} - t$, a_{jk} und a_j unverändert (Ausführungszeiten in μs).

Bsp.	tend	3 h	6 h	9 h	12 h
OZ1	3 CPUs	2463	2463	2558	2661
	ACC	1.99	1.99	1.96	1.99
	4 CPUs	1917	1917	1929	2040
	ACC	2.56	2.56	2.60	2.59
OZ2	3 CPUs	1963	4495	7771	72915
	ACC	1.94	2.00	2.03	2.11
	4 CPUs	1788	4126	6865	62254
	ACC	2.13	2.17	2.30	2.47
OZ2	3 CPUs	($h\max = 135$ min)			4050
	ACC				2.10
	4 CPUs	($h\max = 135$ min)			3534
	ACC				2.40

Literatur:

[1] P. Deuflhard: *Order and Stepsize Control in Extrapolation Methods.* Preprint Nr. 93, Universität Heidelberg (1982).

[2] A.C. Hindmarsh: *ODE Solvers for Use with the Method of Lines.* Preprint UCRL–85293, Rev.1, Lawrence Livermore National Laboratory, CA 94550 (1981).

[3] R.F. Sinovec/N.K. Madsen: *Software for Nonlinear Partial Differential Equations.* ACM Transactions on Mathematical Software, Vol.1, New York (1975).

Veröffentlichungen des Konrad-Zuse-Zentrum für Informationstechnik Berlin
Preprints März 1990

SC 86-1. P. Deuflhard; U. Nowak. *Efficient Numerical Simulation and Identification of Large Chemical Reaction Systems.* (vergriffen) In: Ber. Bunsenges. Phys. Chem., vol. 90, 1986, 940-946
SC 86-2. H. Melenk; W. Neun. *Portable Standard LISP for CRAY X-MP Computers.*

SC 87-1. J. Anderson; W. Galway; R. Kessler; H. Melenk; W. Neun. *The Implementation and Optimization of Portable Standard LISP for the CRAY.*
SC 87-2. Randolph E. Bank; Todd F. Dupont; Harry Yserentant. *The Hierarchical Basis Multigrid Method.* (vergriffen) In: Numerische Mathematik, 52, 1988, 427-458.
SC 87-3. Peter Deuflhard. *Uniqueness Theorems for Stiff ODE Initial Value Problems.*
SC 87-4. Rainer Buhtz. *CGM-Concepts and their Realizations.*
SC 87-5. P. Deuflhard. *A Note on Extrapolation Methods for Second Order ODE Systems.*
SC 87-6. Harry Yserentant. *Preconditioning Indefinite Discretization Matrices.*

SC 88-1. Winfried Neun; Herbert Melenk. *Implementation of the LISP-Arbitrary Precision Arithmetic for a Vector Processor.*
SC 88-2. H. Melenk; H. M. Möller; W. Neun. *On Gröbner Bases Computation on a Supercomputer Using REDUCE.* (vergriffen)
SC 88-3. J. C. Alexander; B. Fiedler. *Global Decoupling of Coupled Symmetric Oscillators.*
SC 88-4. Herbert Melenk; Winfried Neun. *Parallel Polynomial Operations in the Buchberger Algorithm.*
SC 88-5. P. Deuflhard; P. Leinen; H. Yserentant. *Concepts of an Adaptive Hierarchical Finite Element Code.*
SC 88-6. P. Deuflhard; M. Wulkow. *Computational Treatment of Polyreaction Kinetics by Orthogonal Polynomials of a Discrete Variable.* (vergriffen)
SC 88-7. H. Melenk; H. M. Möller; W. Neun. *Symbolic Solution of Large Stationary Chemical Kinetics Problems. I*
SC 88-8. Ronald H. W. Hoppe; Ralf Kornhuber. *Multi-Grid Solution of Two Coupled Stefan Equations Arising in Induction Heating of Large Steel Slabs.*
SC 88-9. Ralf Kornhuber; Rainer Roitzsch. *Adaptive Finite-Element-Methoden für konvektionsdominierte Randwertprobleme bei partiellen Differentialgleichungen.*
SC 88-10. S -N. Chow; B. Deng; B. Fiedler. *Homoclinic Bifurcation at Resonant Eigenvalues.*

SC 89-1. Hongyuan Zha. *A Numerical Algorithm for Computing the Restricted Singular Value Decomposition of Matrix Triplets.*
SC 89-2. Hongyuan Zha. *Restricted Singular Value Decomposition of Matrix Triplets.*
SC 89-3. Wu Huamo. *On the Possible Accuracy of TVD Schemes.*
SC 89-4. H. Michael Möller. *Multivariate Rational Interpolation: Reconstruction of Rational Functions.*
SC 89-5. Ralf Kornhuber; Rainer Roitzsch. *On Adaptive Grid Refinement in the Presence of Internal or Boundary Layers.*
SC 89-6. Wu Huamo; Yang Shuli. *MmB-A New Class of Accurate High Resolution Schemes for Conservation Laws in Two Dimensions.*
SC 89-7. U. Budde; M. Wulkow. *Computation of Molecular Weight Distributions for Free Radical Polymerization Systems.*
SC 89-8. Gerhard Maierhöfer. *Ein paralleler adaptiver Algorithmus für die numerische Integration.*
SC 89-9. Harry Yserentant. *Two Preconditioners Based on the Multi-Level Splitting of Finite Element Spaces.*
SC 89-10. Ronald H. W. Hoppe. *Numerical Solution of Multicomponent Alloy Solidification by Multi-Grid Techniques.*

SC 90-1. M. Wulkow; P. Deuflhard. *Towards an Efficient Computational Treatment of Heterogeneous Polymer Reactions.*
SC 90-2. Peter Deuflhard. *Global Inexact Newton Methods for Very Large Scale Nonlinear Problems.*
SC 90-3. Karin Gatermann. *Symbolic solution of polynomial equation systems with symmetry.*

Veröffentlichungen des Konrad-Zuse-Zentrum für Informationstechnik Berlin
Technical Reports März 1990

TR 86-1. H. J. Schuster. *Tätigkeitsbericht (vergriffen)*

TR 87-1. Hubert Busch; Uwe Pöhle; Wolfgang Stech. *CRAY-Handbuch. - Einführung in die Benutzung der CRAY.*
TR 87-2. Herbert Melenk; Winfried Neun. *Portable Standard LISP Implementation for CRAY X-MP Computers. Release of PSL 3.4 for COS.*
TR 87-3. Herbert Melenk; Winfried Neun. *Portable Common LISP Subset Implementation for CRAY X-MP Computers.*
TR 87-4. Herbert Melenk; Winfried Neun. *REDUCE Installation Guide for CRAY 1 / X-MP Systems Running COS Version 3.3*
TR 87-5. Herbert Melenk; Winfried Neun. *REDUCE Users Guide for the CRAY 1 / X-MP Series Running COS. Version 3.3*
TR 87-6. Rainer Buhtz; Jens Langendorf; Olaf Paetsch; Danuta Anna Buhtz. *ZUGRIFF - Eine vereinheitlichte Datenspezifikation für graphische Darstellungen und ihre graphische Aufbereitung.*
TR 87-7. J. Langendorf; O. Paetsch. *GRAZIL (Graphical ZIB Language).*

TR 88-1. Rainer Buhtz; Danuta Anna Buhtz. *TDLG 3.1 - Ein interaktives Programm zur Darstellung dreidimensionaler Modelle auf Rastergraphikgeräten.*
TR 88-2. Herbert Melenk; Winfried Neun. *REDUCE User's Guide for the CRAY 1 / CRAY X-MP Series Running UNICOS. Version 3.3.*
TR 88-3. Herbert Melenk; Winfried Neun. *REDUCE Installation Guide for CRAY 1 / CRAY X-MP Systems Running UNICOS. Version 3.3.*
TR 88-4. Danuta Anna Buhtz; Jens Langendorf; Olaf Paetsch. *GRAZIL-3D. Ein graphisches Anwendungsprogramm zur Darstellung von Kurven- und Funktionsverläufen im räumlichen Koordinatensystem.*
TR 88-5. Gerhard Maierhöfer; Georg Skorobohatyj. *Parallel-TRAPEX. Ein paralleler, adaptiver Algorithmus zur numerischen Integration ; seine Implementierung für SUPRENUM-artige Architekturen mit SUSI.*

TR 89-1. *CRAY-HANDBUCH. Einführung in die Benutzung der CRAY X-MP unter UNICOS.*
TR 89-2. Peter Deuflhard. *Numerik von Anfangswertmethoden für gewöhnliche Differential-*gleichungen.
TR 89-3. Artur Rudolf Walter. *Ein Finite-Element-Verfahren zur numerischen Lösung von Erhaltungsgleichungen.*
TR 89-4. Rainer Roitzsch. *KASKADE User's Manual.*
TR 89-5. Rainer Roitzsch. *KASKADE Programmer's Manual.*
TR 89-6. Herbert Melenk; Winfried Neun. *Implementation of Portable Standard LISP for the SPARC Processor.*
TR 89-7. Folkmar A. Bornemann. *Adaptive multilevel discretization in time and space for parabolic partial differential equations.*
TR 89-8. Gerhard Maierhöfer; Georg Skorobohatyj. *Implementierung des parallelen TRAPEX auf Transputern.*

TR 90-1. Karin Gatermann. *Gruppentheoretische Konstruktion von symmetrischen Kubaturformeln.*
TR 90-2. Gerhard Maierhöfer; Georg Skorobohatyj. *Implementierung von parallelen Versionen der Gleichungslöser EULEX und EULSIM auf Transputern.*
TR 90-3. *CRAY-Handbuch. Einführung in die Benutzung der CRAY X-MP unter UNICOS 5.1*
TR 90-4. Hans-Christian Hege. *Datenabhängigkeitsanalyse und Programmtransformationen auf CRAY-Rechnern mit dem Fortran-Präprozessor* ***fpp.***