KfK 4688
Februar 1990

Parallele Berechnung elektrostatischer Potentiale und Felder in technischen Geometrien auf SUPRENUM

Benutzerhandbuch EPOTZR-P und EFLDZR-P

M. Alef
Institut für Datenverarbeitung in der Technik

Kernforschungszentrum Karlsruhe

KERNFORSCHUNGSZENTRUM KARLSRUHE
Institut für Datenverarbeitung in der Technik

KfK 4688

Parallele Berechnung elektrostatischer Potentiale und Felder in technischen Geometrien auf SUPRENUM – Benutzerhandbuch EPOTZR-P und EFLDZR-P –

Manfred Alef

Kernforschungszentrum Karlsruhe GmbH, Karlsruhe

Das diesem Bericht zugrundeliegende Vorhaben wurde mit Mitteln des Bundesministers für Forschung und Technologie unter dem Förderkennzeichen ITR8502K/4 gefördert. Die Verantwortung für den Inhalt dieser Veröffentlichung liegt beim Autor.

Kernforschungszentrum Karlsruhe GmbH
Postfach 3640, 7500 Karlsruhe 1

ISSN 0303-4003

Parallele Berechnung elektrostatischer Potentiale und Felder in technischen Geometrien auf SUPRENUM – Benutzerhandbuch EPOTZR-P und EFLDZR-P –

Zusammenfassung:

Die Programme EPOTZR und EFLDZR gestatten eine effiziente Berechnung elektrostatischer Potentiale bzw. Felder in rotationssymmetrischen, technischen Geometrien. Dazu wird die Poissongleichung in einem zweidimensionalen, randangepaßten Gitter in der (r,z)-Ebene (Zylinderkoordinaten) diskretisiert und das so definierte Gleichungssystem mittels Mehrgittermethoden gelöst. Aus diesem Potential werden dann die z- und r-Komponente des elektrostatischen Felds durch numerische Differentiation berechnet.

Dieser Bericht enthält die Programmdokumentation der Programmversionen EPOTZR-P und EFLDZR-P für den Parallelrechner SUPRENUM.

Parallel Computation of Electrostatic Potentials and Fields in Technical Geometries on SUPRENUM – EPOTZR-P and EFLDZR-P User's Guide –

Summary:

The programs EPOTZR and EFLDZR have been developed in order to compute electrostatic potentials and the corresponding fields in technical geometries. The Poisson equation is discretized in a two-dimensional boundary-fitted grid in the (r,z)-plane and solved using multigrid methods. The z- and r-components of the field are determined by numerical differentiation of the potential.

This report contains the user's guide of the SUPRENUM versions EPOTZR-P and EFLDZR-P.

Inhalt:

1. Einführung

In vielen technischen und wissenschaftlichen Bereichen erlangt die numerische Simulation eine wachsende Bedeutung. Ein Beispiel ist die Entwicklung gepulster Hochstrom-Ionendioden, welche fokussierte Ionenstrahlung mit hoher Teilchendichte liefern sollen. Das physikalische Problem ist es, eine Diodengeometrie zu finden, in der die Ionenstrahlen optimal fokussiert werden.

Um die zugrundeliegenden physikalischen Phänomene besser verstehen und um die Geometrie rotationssymmetrischer Dioden optimieren zu können, wurde das zweidimensionale, quasistationäre Programmpaket BFCPIC [1,2] auf der Grundlage der „Particle-in-Cell"-Methode (PIC) in randangepaßten Gittern (boundary-fitted coordinates - BFC) [3,4] entwickelt.

Da diese Simulationen extrem hohe Rechenzeiten erfordern, wurden die rechenintensivsten Module auf SUPRENUM implementiert (Programmpaket BFCPIC-P) [5,6].

Eine der rechenintensivsten Komponenten von BFCPIC ist die Berechnung der elektrostatischen Felder bzw. deren Potentiale. Diesem Zweck dienen die Programme EFLDZR bzw. EPOTZR, die ebenfalls als SUPRENUM-Version in BFCPIC-P enthalten sind.

Diese Potentiale sind durch die Poisson-Gleichung mit geeigneten Randbedingungen bestimmt. Bei der numerischen Berechnung werden diese zunächst im vorgegebenen randangepaßten Gitter in der (r,z)-Ebene diskretisiert und das so definierte Gleichungssystem mit einem Mehrgitteralgorithmus gelöst [4,7-12]. Die z- und die r-Komponente des Feldes werden anschließend durch numerische Differentiation des Potentials bestimmt.

Die Parallelisierung für SUPRENUM basiert auf einer Aufteilung des Gebiets in Streifen, die gleichzeitig von jeweils einem SUPRENUM-Knotenrechner bearbeitet werden [13].

2. Programmbeschreibungen

2.1 Allgemeines

Programme zur Potentialberechnung (EPOTZR-P)

Die in Abschnitt 2.2 vorgestellten Programme

EPOT1H, -2H, -3S, -3A, -3B, -1K, -2K und -3K

(Aufstellung über die unterschiedlichen Anwendungsbereiche dieser Programme s.u.) dienen der Berechnung des Potentials Φ eines elektrostatischen Feldes in einer rotationssymmetrischen Geometrie mit der Raumladungsdichte ρ und mit gebietsweise konstanter Dielektrizitätskonstante ε. Dazu wird die Poisson-Gleichung in (z,r)-Zylinderkoordinaten

$$\Phi_{zz} + \Phi_{rr} + 1/r\,\Phi_r = -\rho/\varepsilon$$

auf einem randangepaßten Gitter mittels einer Mehrstellendiskretisierung zweiter Ordnung diskretisiert und mit Hilfe von Mehrgittermethoden gelöst [12].

Die z- und r-Koordinaten der Punkte dieses Gitters sind vom Benutzer vorzugeben. Das Gitter muß verschiedenen Anforderungen genügen (s. Abschnitt 3), um einerseits eine ausreichend große Genauigkeit, andererseits die effiziente Anwendbarkeit der Mehrgittermethoden und schließlich einen vorteilhaften Einsatz von Vektor- und Parallelrechnern zu erzielen.

Die Indices der Koordinaten der Gitterpunkte bilden das sog. „logische Gitter". Sie werden im folgenden mit x und y bezeichnet; „x-Richtung" („y-Richtung") bedeutet, daß im logischen Gitter die x-(y-)Koordinate variiert wird, während die y-(x-)Koordinate fest ist. Das Gitter muß eine logische Rechteckstruktur aufweisen, d.h. alle Zellen des randangepaßten Gitters müssen viereckig sein (im Gegensatz beispielsweise zu einer Triangulierung [16]).

Der Umfang der vorgebbaren Randbedingungen entspricht den technischen Gegebenheiten [12]. Möglich sind Dirichlet-Randpunkte und Randpunkte mit verschwindender Normalableitung in x- oder y-Richtung sowie Punkte mit Sprung der Normalableitung in x- oder y-Richtung. Außerdem gibt es eine Bedingung für die Rotationsachse (stetige Fortsetzung der Poisson-Gleichung unter Ausnutzung der Rotationssymmetrie).

Das Prozeß-Konzept des SUPRENUM-Rechners (s. Abschnitt 4.3.1) bedingt eine Aufspaltung der Algorithmen in ein Steuerprogramm, das im Haupt- oder einem speziellen Steuerprozeß aufgerufen wird (s. Abschnitt 2.2.1), sowie dem eigentlichen Berechnungsprogramm, welches parallel in den Knotenprozessen aufgerufen wird (s. Abschnitt 2.2.2) [14].

Um unterschiedliche Aufgabenstellungen behandeln zu können, sind die folgenden Programm-Kombinationen möglich:

1. Bei den Programmen EPOT1H und EPOT1K werden alle Ein- und Ausgabedaten vom Haupt- an die Knotenprozesse bzw. zurück gesendet. Das Programm EPOT1H **muß** vom Hauptprozeß, EPOT1K von den Knotenprozessen aus aufgerufen werden.

   Dieses Programmpaar dient hauptsächlich Testzwecken.

2. Die Programme EPOT2H und EPOT2K entsprechen im wesentlichen den Programmen EPOT1H und EPOT1K, s.o. Im Unterschied dazu werden hierbei jedoch nur die Verfahrensparameter beim ersten Aufruf an die Knoten gesandt. Umgekehrt erhält der Hauptprozeß nur Informationen über den Konvergenzverlauf von den Knoten.

   Für Produktionsläufe sollten nur diese beiden Programme verwendet werden.

   Ein Austausch speziell der Matrizen RHO (Raumladungsdichten) und PHI (Potential) ist mittels der Programme VERT2H und VERT2K (Aufteilung einer Matrix vom Hauptprozeß auf die Knotenprozesse) bzw. SAML2H und SAML2K (Einsammeln der Ergebnismatrix im Hauptprozeß) möglich, s. Abschnitt 2.3.

3. Die Programme EPOT3S, oder EPOT3A zusammen mit EPOT3B, und EPOT3K entsprechen den unter 1. genannten. Der Unterschied besteht darin, daß die zentrale Steuerung (EPOT3S oder EPOT3A mit EPOT3B) statt im Hauptprozeß auch auf einem Knotenrechner ablaufen kann. Da in SUPRENUM-Knotenprozessen keine Ein-/ Ausgabe vorgesehen ist, bieten sich jedoch nur sehr eingeschränkte Diagnostikmöglichkeiten: Fehlermeldungen werden im Gegensatz zu den detailierten Hinweisen der unter 1. und 2. genannten Programmen nur in einer Kurzform ausgegeben; auf Warnungen wird ganz verzichtet. Außerdem werden keine Übersichtstabellen zur Kontrolle des Konvergenzverlaufs ausgegeben. (Die Erzeugung des Steuerprozesses sowie die Kommunikation zwischen Haupt- und Steuerprozeß erfolgen außerhalb dieses Programmpakets.)

Dieser steuernde Knotenprozeß wird im folgenden Steuerprozeß genannt. Ansonsten sind mit Knotenprozessen stets nur die eigentlichen parallelen Rechenprozesse zur Potential- bzw. Feldberechnung gemeint.

Für den Steuerprozeß gibt es dabei zwei Möglichkeiten. Entweder wird dort das Programm EPOT3S aufgerufen, welches die parallelen Prozesse erzeugt, diesen die jeweils benötigten Eingabedaten sendet und anschließend auf das Ergebnis wartet. Während dieser Wartezeit ist der Steuerprozeß blockiert, d.h. er kann keine anderen Tätigkeiten (Berechnungen, Kommunikation mit anderen Prozessen) ausführen. Als Alternative kann zuerst das Programm EPOT3A aufgerufen werden, welches nur die für die parallele Potentialberechnung benötigten Knotenprozesse erzeugt und mit den Eingabedaten versorgt. Anschließend können andere Anweisungen durchgeführt werden, bis schließlich mittels des Programms EPOT3B die Ergebnisse angenommen werden.

Bei dieser zweiten Vorgehensweise hat der Benutzer jedoch selber darauf zu achten, daß die zwischen EPOT3A und EPOT3B stehenden Anweisungen nicht mehr Zeit in Anspruch nehmen, als die parallele Potentialberechnung, da ansonsten die Knotenprozesse während der restlichen Zeit stillstehen.

Die Erzeugung der Knotenprozesse für die parallele Potentialberechnung sowie die Berechnung der Diskretisierungskoeffizienten kosten Zeit. Damit dieser Aufwand bei mehreren aufeinanderfolgenden Berechnungen nicht nicht jedesmal wiederholt werden muß, besteht die Möglichkeit zu Wiederholungsaufrufen: Ab dem zweiten Aufruf müssen keine Knotenprozesse erzeugt werden, und es sind nur neue Diskretisierungskoeffizienten zu berechnen, wenn mit einem anderen Gitter weitergearbeitet werden soll. Dies kann mittels des Parameters IAUFR gesteuert werden (s. Abschnitt 2.2.1.3).

Feldberechnung (EFLDZR-P)

Das elektrostatische Feld E wird aus seinem Potential Φ (s.o.) berechnet:

$$E = (E_z, E_r) = -\nabla\Phi = -(\partial / \partial z\, \Phi, \partial / \partial r\, \Phi).$$

Dazu werden Differenzenquotienten zweiter, auf Leiteroberflächen erster Ordnung angewendet. Die Diskretisierung nur erster Ordnung auf Leiterflächen wurde deshalb gewählt, weil für die weitere Bearbeitung innerhalb des Programmpakets BFCPIC die elektrischen Feldstärken nicht unmittelbar auf der Oberfläche der Elektrode, sondern in der Mitte der angrenzenden Gittermaschen benötigt

werden. Genau diese Zwischenwerte werden durch die Differenzenquotienten erster Ordnung berechnet.

Zur Berechnung elektrostatischer Felder dienen die ebenfalls in Abschnitt 2.2 näher beschriebenen Programme

EFLD1H, -2H, -3S, -3A, -3B, -1K, -2K und -3K

(Aufstellung s.u.), die zunächst das Potential Φ und daraus die z- und r-Komponenten des Feldes berechnen.

Für die unterschiedlichen Aufgabenstellungen stehen die folgenden Programm-Kombinationen zur Verfügung:

1. Bei den Programmen EFLD1H und EFLD1K werden alle Ein- und Ausgabedaten vom Haupt- an die Knotenprozesse bzw. zurück gesendet. Das Programm EFLD1H **muß** vom Hauptprozeß, EFLD1K von den Knotenprozessen aus aufgerufen werden.

   Dieses Programmpaar, welches im wesentlichen den Programmen EPOT1H und EPOT1K entspricht, dient hauptsächlich Testzwecken.

2. Die Programme EFLD2H und EFLD2K entsprechen im wesentlichen den Programmen EPOT2H und EPOT2K zur Potentialberechnung: Es werden hierbei nur die Verfahrensparameter beim ersten Aufruf an die Knoten gesandt. Umgekehrt erhält der Hauptprozeß nur Informationen über den Konvergenzverlauf von den Knoten.

   Für Produktionsläufe sollten nur diese beiden Programme verwendet werden.

   Ein Austausch speziell der Matrizen RHO (Raumladungsdichten) und EZ, ER (Feldstärken) ist im Zusammenhang mit diesem Programmpaar mittels der Programme VERT2H und VERT2K (Aufteilung einer Matrix vom Hauptprozeß auf die Knotenprozesse) bzw. SAML2H und SAML2K (Einsammeln der Ergebnismatrix im Hauptprozeß) möglich, s. Abschnitt 2.3.

3. Die Programme EFLD3S, oder EFLD3A zusammen mit EFLD3B, und EFLD3K entsprechen den unter 1. genannten. Der Unterschied besteht darin, daß die zentrale Steuerung (EFLD3S oder EFLD3A mit EFLD3B) statt im Hauptprozeß auf einem Knotenrechner ablaufen kann. Da in SUPRENUM-Knotenprozessen keine Ein-/ Ausgabe vorgesehen ist, bieten sich jedoch nur sehr eingeschränkte Diagnostikmöglichkeiten. (Die Erzeugung des Steuerprozesses sowie die Kommunikation zwischen Haupt- und Steuerprozeß erfolgen außerhalb dieses Programmpakets.)

Diese Programme entsprechen im wesentlichen den Programmen EPOT3S, EPOT3A mit EPOT3B sowie EPOT3K (s.o.), die anstelle der Feldstärken das Potential an den Steuerprozeß senden.

Die Erzeugung der Knotenprozesse für die parallele Potentialberechnung sowie die Berechnung der Diskretisierungskoeffizienten kosten Zeit. Damit dieser Aufwand bei mehreren aufeinanderfolgenden Berechnungen nicht nicht jedesmal wiederholt werden muß, besteht die Möglichkeit zu Wiederholungsaufrufen: Ab dem zweiten Aufruf müssen keine Knotenprozesse erzeugt werden, und es sind nur dann neue Diskretisierungskoeffizienten zu berechnen, wenn mit einem anderen Gitter weitergearbeitet werden soll. Dies kann mittels des Parameters IAUFR gesteuert werden (s. Abschnitt 2.2.1.3).

Expliziter Datenaustausch zwischen Hauptprozeß und Knotenprozessen

Nachdem die Knotenprozesse erzeugt sind (s. Programmpaare EPOT2H/-2K und EFLD2H/-2K zur Potential- bzw. zur Feldberechnung), können explizit Daten vom Hauptprozeß an die Knotenprozesse (VERT2H und VERT2K) und umgekehrt (SAML2H und SAML2K) gesendet werden, s. Abschnitt 2.3.

2.2 Potential- und Feldberechnung

Wie bereits geschildert, erfordert das SUPRENUM-Prozeß-Konzept eine Aufspaltung der Algorithmen in zwei Teile (vgl. Abbildungen 2.2.0.1 und 2.2.0.2, Abschnitt 4.3.1 sowie Tabelle 2.2.0.3). Dazu benötigt man:

1. Eines der Programme EPOT1H/-2H bzw. EFLD1H/-2H, welches vom Hauptprozeß aufgerufen wird, oder EPOT3S bzw. EFLD3S (bzw. die aufgespaltenen Versionen -3A/-3B), die auch von einem Knotenprozeß aus aufgerufen werden können.

   Dieser Steuerprogramm-Aufruf hat nur die Aufgabe, die parallelen Prozesse auf den Knotenrechnern zu starten, diesen die erforderlichen Eingabedaten zu senden, ggf. das Endergebnis von den Knoten einzusammeln, sowie Informationen über etwaige Fehler sowie über den Konvergenzverlauf entgegenzunehmen und auszudrucken.

2. Eines der Programme EPOT1K, -2K, -3K, EFLD1K, -2K oder -3K, welches vom Knoten-Hauptprogramm aus aufgerufen wird und die eigentlichen Berechnungen in einem bestimmten Teil des Berechnungsgebiets durchführt. Die Steuerung dieser Berechnungen (Information über das zu bearbeitende Teilgebiet, Bereitstellung der erforderlichen Eingabedaten wie Raumladungsdichte oder Gitterkoordinaten des Teilgebiets, Überwachung des Konvergenzverlaufs, ggf. Entgegennahme des Endergebnisses u.s.w.) erfolgt vom Haupt- bzw. dem steuernden Knotenprozeß aus in korrespondierenden Aufrufen eines der unter 1. genannten Programme.

Weitere Details zum Parallelisierungskonzept s. Kapitel 4.

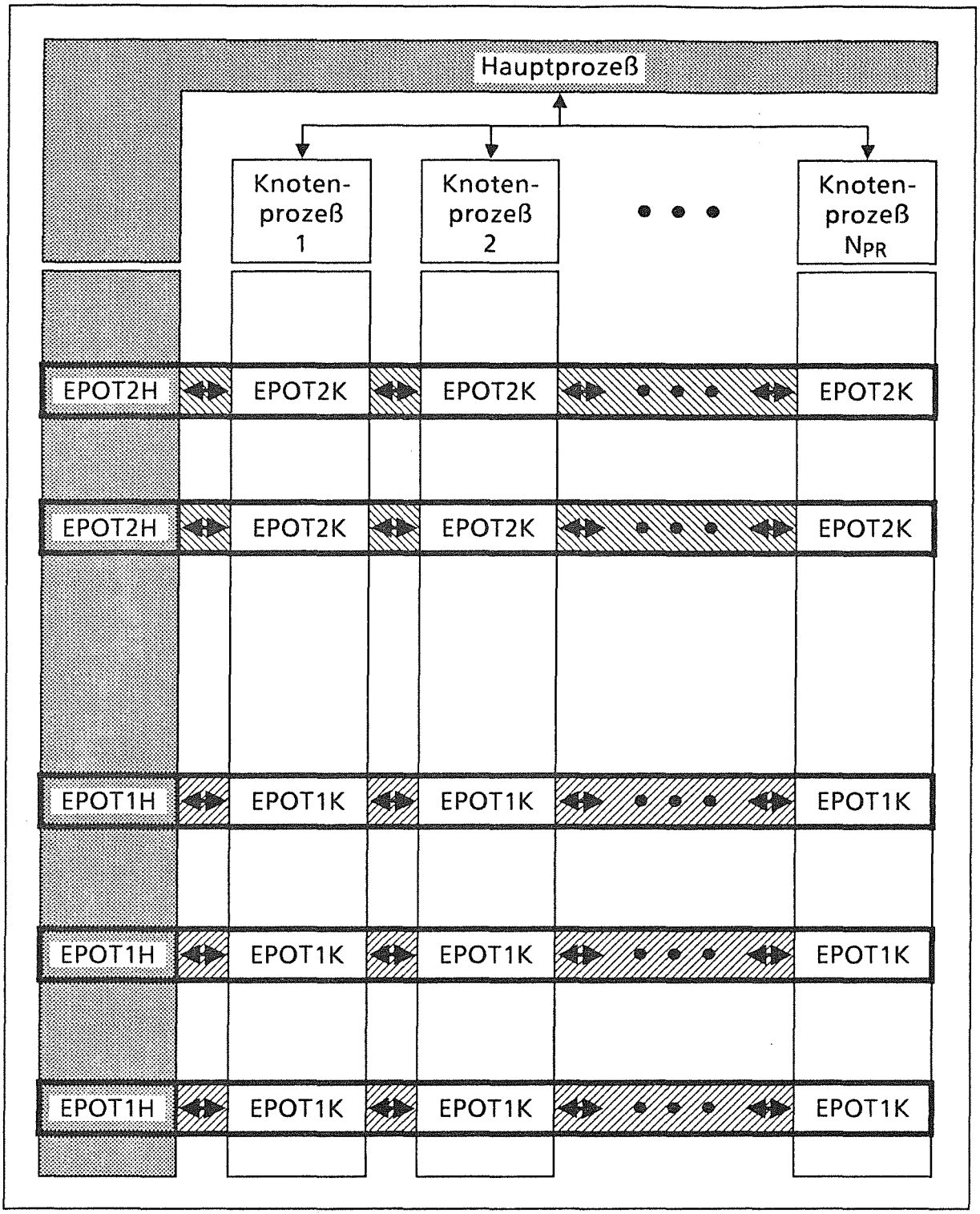


Abb. 2.2.0.1: Prozeß-Konzept:
Korrespondierende Aufrufe der Programme EPOT1H und EPOT2H im Hauptprozeß und EPOT1K bzw. EPOT2K in den Knotenprozessen zur Potentialberechnung.
(Zur Feldberechnung sind die Programme EFLDxx anstelle von EPOTxx entsprechend aufzurufen.)

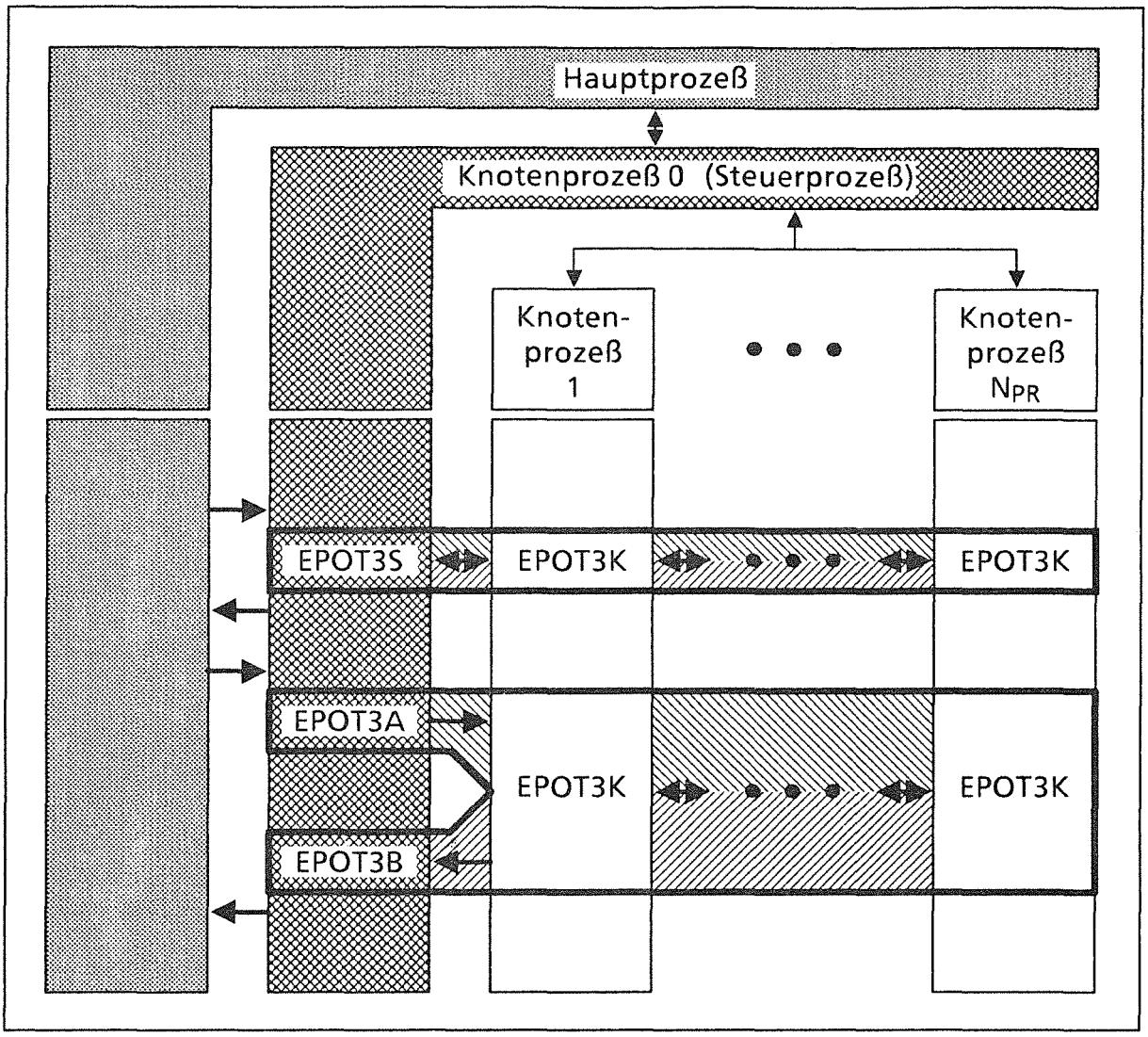


Abb. 2.2.0.2: Prozeß-Konzept:
Korrespondierende Aufrufe der Programme EPOT3S, bzw. EPOT3A zusammen mit EPOT3B, in einem Steuerprozeß und EPOT3K in den Knotenprozessen zur Potentialberechnung. - Zur Feldberechnung sind entsprechend im Steuerprozeß die Programme EFLD3S bzw. -3A und -3B statt EPOT3S/-3A/-3B und in den Knotenprozessen EFLD3K statt EPOT3K aufzurufen.
(Die Erzeugung dieses Steuerprozesses sowie die Kommunikation zwischen Haupt- und Steuerprozeß erfolgen außerhalb dieses Programmpakets.)

Hauptprozeß (auf Vorrechner)	Steuerprozeß (auf Knotenrechner)	Knotenprozesse (auf Knotenrechnern)
EPOT1H Steuerung der Potentialberechnung mit Bereitstellung im Hauptprozeß		**EPOT1K** Berechnung des Potentials und Übersendung an den Hauptprozeß
EPOT2H Steuerung der Potentialberechnung, ohne Bereitstellung im Hauptprozeß		**EPOT2K** Berechnung des Potentials ohne Übersendung an den Hauptprozeß
	EPOT3S Steuerung der Potentialberechnung mit Bereitstellung im Steuerprozeß	**EPOT3K** Berechnung des Potentials und Übersendung an den Steuerprozeß
	Steuerung der Potentialberechnung: **EPOT3A** Übersendung der Raumladungsdichte an die Knotenprozesse **EPOT3B** Annahme des Potentials von d. Knotenprozessen	**EPOT3K** Berechnung des Potentials und Übersendung an den Steuerprozeß
EFLD1H Steuerung der Feldberechnung mit Bereitstellung im Hauptprozeß		**EFLD1K** Berechnung der Felder mit Übersendung an den Hauptprozeß
EFLD2H Steuerung der Feldberechnung ohne Bereitstellung im Hauptprozeß		**EFLD2K** Berechnung der Felder ohne Übersendung an den Hauptprozeß
	EFLD3S Steuerung der Feldberechnung mit Bereitstellung im Steuerprozeß	**EFLD3K** Berechnung der Felder und Übersendung an den Steuerprozeß
	Steuerung der Feldberechnung: **EFLD3A** Übersendung der Raumladungsdichte an die Knotenprozesse **EFLD3B** Annahme der Felder von den Knotenprozessen	**EFLD3K** Berechnung der Felder und Übersendung an den Steuerprozeß

Tab. 2.2.0.3: Korrespondierende Programmaufrufe zur Potential- bzw. Feldberechnung in Haupt- oder Steuerprozeß einerseits und in den Knotenprozessen andererseits.

2.2.1 Haupt- bzw. Steuerprozeß

Die Steuerung der parallelen Berechnung kann entweder unmittelbar durch den auf dem Vorrechner (Host) ablaufenden initialen Prozeß (Hauptprozeß) erfolgen, oder durch einen speziellen, vom Benutzer zu erzeugenden und mit den erforderlichen Eingabedaten zu versorgenden Knotenprozeß (Steuerprozeß).

Dabei wird im Hauptprozeß eines der Programme EPOT1H, EPOT2H, EFLD1H oder EFLD2H (s. Abschnitt 2.2.1.1), im Steuerprozeß EPOT3S bzw. EFLD3S aufgerufen (s. Abschnitt 2.2.1.2).

Anstelle des Aufrufs des Programms EPOT3S können die Programme EPOT3A und EPOT3B aufgerufen werden. Dies hat den Vorteil, daß im Steuerprozeß ebenfalls gewisse Berechnungen durchgeführt werden können, während die Knotenprozesse mit der Potentialberechnung beschäftigt sind (s. Beispiel 3 im Abschnitt 5.3). Entsprechend kann das Programm EFLD3S durch die Teilprogramme EFLD3A und EFLD3B ersetzt werden.

2.2.1.1 Hauptprozeß (EPOT1H/-2H, EFLD1H/-2H)

Aufrufe zur Potentialberechnung

Mit Bereitstellung des Potentials im Hauptprozeß:

```
      CALL EPOT1H (PHI, RHO, ZK, RK, PPAR, IFALL, IPAT,
     *             NX, NY, NDIM1, NDIM2, NPAT, IAUFR,
     *             NPRMX, NPR, NZKLMN, NZKLMX,
     *             AFLRMX, RFLRMX, KNOTEN, IDRUCK, IFLR)
```

Ohne Bereitstellung des Potentials im Hauptprozeß:

```
      CALL EPOT2H (ZK, RK, PPAR, IFALL, IPAT,
     *             NX, NY, NDIM1, NDIM2, NPAT, IAUFR,
     *             NPRMX, NPR, NZKLMN, NZKLMX,
     *             AFLRMX, RFLRMX, KNOTEN, IDRUCK, IFLR)
```

Aufrufe zur Feldberechnung

Mit Bereitstellung des Felds im Hauptprozeß:

```
      CALL EFLD1H (EZ, ER, RHO, ZK, RK, PPAR, IFALL, IPAT,
     *             NX, NY, NDIM1, NDIM2, NPAT, IAUFR,
     *             NPRMX, NPR, NZKLMN, NZKLMX,
     *             AFLRMX, RFLRMX, KNOTEN, IDRUCK, IFLR)
```

Ohne Bereitstellung des Felds im Hauptprozeß:

```
      CALL EFLD2H (ZK, RK, PPAR, IFALL, IPAT,
     *             NX, NY, NDIM1, NDIM2, NPAT, IAUFR,
     *             NPRMX, NPR, NZKLMN, NZKLMX,
     *             AFLRMX, RFLRMX, KNOTEN, IDRUCK, IFLR)
```

Parameter

S. Abschnitt 2.2.1.3.

2.2.1.2 Steuerprozeß (EPOT3S/-A /-B, EFLD3S/-A /-B)

Aufrufe zur Potentialberechnung

Entweder:

```
      CALL EPOT3S (PHI, RHO, ZK, RK, PPAR, IFALL, IPAT,
     *             NX, NY, NDIM1, NDIM2, NPAT, IAUFR,
     *             NPRMX, NPR, NZKLMN, NZKLMX,
     *             AFLRMX, RFLRMX, KNOTEN, IDRUCK, IFLR)
```

Oder:

```
      CALL EPOT3A (RHO, ZK, RK, PPAR, IFALL, IPAT,
     *             NX, NY, NDIM1, NDIM2, NPAT, IAUFR,
     *             NPRMX, NPR, NZKLMN, NZKLMX,
     *             AFLRMX, RFLRMX, KNOTEN, IDRUCK, IFLR)
               .
               .
               .
      CALL EPOT3B (PHI, NX, NY, NDIM1, NDIM2, IAUFR,
     *             NPRMX, NPR, IDRUCK, IFLR)
```

(Diese Programmkombination leistet genau dasselbe wie das Programm EPOT3S. Zwischen dem Verteilen der Eingabedaten auf die parallelen Knotenprozesse und dem Empfangen der Ergebnisse können jedoch noch andere Berechnungen etc. durchgeführt werden, während der Steuerprozeß bei EPOT3S während dieser Wartezeit stillsteht. Dabei sollte jedoch darauf geachtet werden, daß diese Berechnungen nicht länger dauern als die Potentialberechnung, da sonst die Knotenprozesse während der restlichen Zeit stillständen.)

Aufrufe zur Feldberechnung

Entweder:

```
      CALL EFLD3S (EZ, ER, RHO, ZK, RK, PPAR, IFALL, IPAT,
     *             NX, NY, NDIM1, NDIM2, NPAT, IAUFR,
     *             NPRMX, NPR, NZKLMN, NZKLMX,
     *             AFLRMX, RFLRMX, KNOTEN, IDRUCK, IFLR)
```

Oder (vgl. die Bemerkungen zu den Programmen EPOT3A und -3B):

```
      CALL EFLD3A (RHO, ZK, RK, PPAR, IFALL, IPAT,
     *             NX, NY, NDIM1, NDIM2, NPAT, IAUFR,
     *             NPRMX, NPR, NZKLMN, NZKLMX,
     *             AFLRMX, RFLRMX, KNOTEN, IDRUCK, IFLR)
                .
                .
                .
      CALL EFLD3B (EZ, ER, NX, NY, NDIM1, NDIM2, IAUFR,
     *             NPRMX, NPR, IDRUCK, IFLR)
```

Parameter

S. Abschnitt 2.2.1.3.

2.2.1.3 Parameterbeschreibung der im Haupt- oder Steuerprozeß aufzurufenden Programme zur Potential- bzw. Feldberechnung

Die folgende Tabelle gibt Auskunft über Funktion, Typ und Deklaration der einzelnen Programmparameter.

Der Einfachheit halber werden alle in mindestens einem der o.g. Programme vorkommenden Parameter fortlaufend beschrieben. Welche davon für den jeweiligen Programmaufruf infrage kommen, ist den in den Abschnitten 2.2.1.1 und 2.2.1.2 beschriebenen Parameterlisten dieser Programme zu entnehmen.

Erläuterungen zur Tabelle

Name: Variablenname, wie er in den Programmen benutzt wird.

Funktion: Aufgabe des Parameters (Ein-, Ausgabe oder Hilfsvektor).

Typ: Datentyp der Variablen, wie er im Unterprogramm benötigt wird (INTEGER, REAL oder TASK EXTERNAL {=SUPRENUM-Erweiterung von FORTRAN}).

Dimension: Dimensionierung von Vektoren bzw. Matrizen im Unterprogramm, die der jeweiligen Dimension im aufrufenden Programm entsprechen muß. Ist in dieser Spalte nichts angegeben, so handelt es sich um eine skalare Variable.

Name	Funktion	Typ	Dimension
PHI	***Ausgabe***	**REAL*8**	**(NDIM1,NDIM2)**

Gesuchte Näherungslösung für das elektrostatische Potential Φ.

EZ, ER	***Ausgabe***	**REAL*8**	**(NDIM1,NDIM2)**

Gesuchte Näherungslösung für das elektrostatische Feld E
(z- und r-Komponente).

RHO	***Eingabe***	**REAL*8**	**(NDIM1,NDIM2)**

In Feldpunkten (im Inneren des Gebiets oder als Feldpunkte auf der Rotationsachse) sowie in Punkten mit verschwindender Normalableitung außerhalb der Rotationsachse ist

RHO : = $+\rho/\varepsilon$

die rechte Seite der Poisson-Gleichung. Dabei ist ρ die Raumladungsdichte und $\varepsilon = \varepsilon_0\varepsilon_*$ die Dielektrizitätskonstante in dem Gebietsteil, dem der jeweilige Gitterpunkt angehört (ε_0 absolute, ε_* materialabhängige relative Dielektrizitätskonstante im jeweiligen Teilgebiet).
In Punkten mit Sprung der Normalableitung ist

RHO : = $2\rho/(\varepsilon_0(\varepsilon_+ + \varepsilon_-))$

vorzugeben, wobei ε_- die Permittivität im links bzw. unten und ε_+ die im rechts oder oben angrenzenden Gebietsteil ist, bezogen auf das logische Gitter.
In Dirichlet-Randpunkten braucht RHO nicht vorgegeben zu werden.

ZK, RK	***Eingabe***	**REAL*8**	**(NDIM1, NDIM2)**

z- bzw. r-Koordinaten der Punkte des randangepaßten Gitters, in dem das Potential berechnet werden soll. Die r-Koordinaten dürfen nicht negativ sein, und es muß gelten:

RK(IX,IY) = 0 $\Rightarrow$ IY = 1.

Die Anforderungen, die das Gitter erfüllen muß, sind in Kapitel 3 näher erläutert.

Name	Funktion	Typ	Dimension
PPAR	*Eingabe*	REAL*8	(-NPAT:NPAT,2)

Punktparameter lt. folgender Vorschrift:

IFALL(IPAT(IX,IY))		PPAR(IPAT(IX,IY),i)
0		Ohne Bedeutung
1	Feldpunkt außerhalb der Symmetrieachse	Ohne Bedeutung
2	Dirichlet-Randpunkt [1]	Falls PPAR(... ,2) ≥ 0, dann enthält PPAR(... ,1) den Dirichlet-Randwert für Φ an Anode oder Kathode; sonst ohne Bedeutung [1]
3,4	Randpunkt mit verschwindender Normalableitung in x- bzw. in y-Richtung	Ohne Bedeutung
5,6	Innerer Randpunkt mit Normalableitungssprung in x- bzw. in y-Richtung	PPAR(... ,1) enthält die relative Dielektrizitätskonstante des links bzw. unten, PPAR(... ,2) die des rechts bzw. oben angrenzenden Teilgebiets
8	Feldpunkt auf der Symmetrieachse	Ohne Bedeutung

1) Die Fallunterscheidung in Dirichlet-Punkten ermöglicht es, das Potential in diesen Punkten wahlweise über PPAR anzugeben oder aber bereits selber in PHI einzutragen; in diesem Fall ist PPAR(IPAT(IX,IY),2) mit einem negativen Wert (z. B. mit -1.0) zu belegen. Diese zweite Möglichkeit ist jedoch nur anwendbar bei Aufruf der Programme EPOT1H bzw. EFLD1H, oder bei Verwendung von EPOT2K bzw. EFLD2K in den Knotenprozessen (s. Abschnitt 2.2.2).
PPAR wird in Dirichlet-Randpunkten auch bei Wiederholungsaufrufen (IAUFR = 2) ausgewertet, so daß nacheinander mehrere Berechnungen mit unterschiedlichen Dirichlet-Randbedingungen möglich sind.

Name	Funktion	Typ	Dimension

IFALL *Eingabe* **INTEGER** **(-NPAT:NPAT)**

Tabelle, die den in IPAT für jeden Gitterpunkt enthaltenen Punktattributen die zugehörige Punktart zuordnet:

Mögliche Punktarten:

0	Das entsprechende Punktattribut tritt in IPAT nicht auf
1	Feldpunkt außerhalb der Symmetrieachse
2	Dirichlet-Randpunkt
3	Randpunkt mit Normalableitung = 0 in x-Richtung
4	Randpunkt mit Normalableitung = 0 in y-Richtung
5	Innerer Randpunkt mit Normalableitungssprung in x-Richtung
6	Innerer Randpunkt mit Normalableitungssprung in y-Richtung
8	Feldpunkt auf der Symmetrieachse

IPAT *Eingabe* **INTEGER** **(NDIM1,NDIM2)**

Matrix der Punktattribute, die jedem Punkt des (feinsten) Gitters zuzuweisen sind. Diese Werte müssen zwischen -NPAT und NPAT liegen. Die Falltabelle IFALL gibt zu jedem in IPAT auftretenden Punktattribut die zugehörige Punktart an, s. oben.

NX, NY *Eingabe* **INTEGER**

Anzahl der Punkte des (feinsten) Gitters in x- bzw. in y-Richtung. Aus Effizienzgründen sollten NX und NY wie folgt gewählt werden:

$$NX = 2^m + 1,$$
$$NY = 2^n + 1,$$

mit natürlichen Zahlen m und n.

NDIM1, NDIM2 *Eingabe* **INTEGER**

Dimensionierung von PHI, RHO, IPAT, ZK und RK im aufrufenden Programm, mit $NX \leq NDIM1$ und $NY \leq NDIM2$.

NPAT *Eingabe* **INTEGER**

Begrenzung der möglichen Punktattribute, s. IPAT. Es muß gelten:

$$NPAT \leq 2^{31} - 1.$$

Name	Funktion	Typ	Dimension

IAUFR *Eingabe* **INTEGER**

Gibt an, ob das Programm für eine bestimmte Aufgabe zum ersten Mal aufgerufen wird, oder ob bereits vorher ein Aufruf für das gleiche Problem (mit dem selben Wert für NPRMX) erfolgte. IFALL und IPAT dürfen seit diesem Aufruf nicht verändert worden sein; jedoch dürfen mittels PPAR (s.o.) in jedem Folge-Aufruf andere Dirichlet-Randwerte vorgegeben werden (s. auch Abschnitt 2.4):

IAUFR =	-1	0	1	2	9
Erzeugung der parallelen Knotenprozesse		x	x		
Initialisierung (Berechnung der Diskretisierungskoeffizienten)	x	x	x		
Berechnung des Potentials			x	x	x
Setzen eines Codes, mit dem in den parallelen Knotenprozessen eine Bedingung getestet werden kann (s. Abschnitt 2.2.2)					x

NPRMX *Eingabe* **INTEGER**

Maximale Anzahl der für die parallele Ausführung der Potential- bzw. Feldberechnung zur Verfügung stehenden Knotenrechner. Diese muß mindestens 1 und möglichst nicht größer als die Anzahl der auf dem SUPRENUM-Rechners vorhandenen Knotenrechner sein.
Intern wird jedoch möglicherweise eine geringer Anzahl von Knotenrechnern verwendet, wenn dies im Einzelfall effizienter ist; vgl. den Parameter NPR.
Am günstigsten ist es, wenn NPRMX wie folgt gewählt wird:

$$NPRMX = (NY - 1) / 2^{m}$$

mit einer natürlichen Zahl m.

NPR *Ausgabe* **INTEGER**

Tatsächliche Anzahl der für die parallele Berechnung erzeugten Knotenprozesse. Diese Anzahl kann u.U. kleiner als die vorgegebene Anzahl NPRMX sein, wenn dies im Einzelfall effizienter ist.

Name	Funktion	Typ	Dimension

NZKLMN *Eingabe* **INTEGER**

Minimale Anzahl der auf dem feinsten Gitter durchzuführenden Mehrgitterzyklen, mit NZKLMN > 0.
Wenn kein Fehler (mit Fehlerindikator IFLR > 0) auftritt, wird diese Anzahl unter keinen Umständen unterschritten. Falls nach NZKLMN Zyklen die vorgegebenen Fehlerschranken (AFLRMX, RFLRMX) noch nicht erreicht sind und NZKLMX größer als NZKLMN ist, dann werden noch weitere Zyklen durchgeführt, vgl. den Parameter NZKLMX.
Aus Effizienzgründen sollte NZKLMN nicht oder zumindest nicht wesentlich kleiner sein als NZKLMX, s.u.

NZKLMX *Eingabe* **INTEGER**

Maximale Anzahl der auf dem feinsten Gitter durchzuführenden Mehrgitterzyklen, mit NZKLMX ≥ NZKLMN.
Falls die vorgegebenen Fehlerschranken (AFLRMX, RFLRMX) schon früher erfüllt werden, werden weniger als NZKLMX Zyklen durchgeführt, mindestens jedoch NZKLMN Zyklen.

AFLRMX *Eingabe* **REAL*8**

Maximal zulässiger Wert des absoluten Fehlers des Potentials PHI nach Beendigung des Programms, jedoch ohne Berücksichtigung von Diskretisierungsfehlern.
Die Fehlerbedingungen AFLRMX und RFLRMX (s.u.) sind logisch durch „oder" verknüpft.
In der Regel sollte AFLRMX = 0.0 vorgegeben werden, damit eine relative Genauigkeit gewährleistet ist.

RFLRMX *Eingabe* **REAL*8**

Maximal zulässiger Wert des relativen Fehlers des Potentials PHI nach Beendigung des Programms, jedoch ohne Berücksichtigung von Diskretisierungsfehlern. Sollte RFLRMX in Abhängigkeit von der verfügbaren Maschinengenauigkeit EPS zu gering gewählt worden sein, so wird intern ein größerer Wert verwendet.

Name	Funktion	Typ	Dimension

KNOTEN *Eingabe* **TASK EXTERNAL**

Vom Benutzer definierter Name des Hauptprogramms der Knotenprozesse. In diesem Knoten-Hauptprogramm „KNOTEN" ist eines der Programme EPOTxK bzw. EFLDxK aufzurufen, das laut Tabelle 2.2.0.3 zum entsprechenden Aufruf im Haupt- oder Steuerprozeß „passen" muß.

IDRUCK *Eingabe* **INTEGER**

Umfang der vom Hauptprozeß auszudruckenden Informationen:

IDRUCK =	0	1	2	3	4
Fehlermeldungen		x	x	x	x
Konvergenzraten			x	x	x
Warnungen (ungünstige Parameterkombinationen, etc.)				x	x
Verfahrenshinweise					x

Die Ausgabe erfolgt bei EPOT1H, -2H, EFLD1H und -2H über den Standard-Ausgabekanal, bei EPOT3S bzw. -3A/-3B oder EFLD3S bzw. -3A/-3B über die Konsole (dabei ist nur IDRUCK $\leqq$ 1 möglich).

Bei der Wahl dieses Parameters ist auf den unterschiedlichen Kommunikationsaufwand zu achten: Die Berechnung der Konvergenzraten erfordert die Ermittlung der Betragsmaxima des absoluten Fehlers und der Lösung, d.h. diese sind zunächst in allen Prozessen zu berechnen und über alle Prozesse zu vergleichen. Dies ist mit einem hohen Kommunikationsaufwand verbunden, der relativ viel Zeit kostet und deshalb nur in den beiden folgenden Fällen durchgeführt wird:

1. es sollen Konvergenzraten ausgedruckt werden (IDRUCK $\geqq$ 2), oder
2. falls NZKLMX > NZKLMN, nach jedem ab dem NZKLMN-ten Mehrgitterzyklus.

Deshalb:

- Konvergenztabellen nur zu Testzwecken ausdrucken, und
- die Anzahl der Mehrgitterzyklen möglichst genau abschätzen und möglichst NZKLMX nicht (wesentlich) größer als NZKLMN wählen.

Name	Funktion	Typ	Dimension
IFLR	***Ausgabe***	**INTEGER**	

Fehlerindikator:

= 0 Es wurden keine Fehler festgestellt.

> 0 Es wurde ein so schwerwiegender Fehler festgestellt, daß die Berechnung nicht durchgeführt werden konnte.

< 0 Warnung (z. B. ungünstige Wahl der Eingabeparameter), die Berechnung wurde jedoch beendet.

Über den Parameter IDRUCK (s.o.) kann gewählt werden, ob Fehlernachrichten bzw. Warnhinweise ausgedruckt werden sollen oder nicht.

Bedeutung der Fehlercodes:

Fehler in den Eingabedaten:

= 104 Es wurde NX > NDIM1 angegeben.

= 105 Es wurde NY > NDIM2 angegeben.

= 111 In einem Gitterpunkt (IX,IY) wurde ein fehlerhaftes Punktattribut festgestellt:
IPAT (IX,IY) < –NPAT oder IPAT (IX,IY) > NPAT.

= 112 In einem Gitterpunkt (IX,IY) trat eine fehlerhafte Fallzuordnung auf, d. h. IFALL(IPAT(IX,IY)) $\notin \{1, 2, \ldots, 6, 8\}$.

= 113 Ein als Feldpunkt deklarierter Punkt (IFALL(IPAT(IX,IY)) = 1) liegt auf dem Rand. Feldpunkte dürfen nur im Inneren des Gebiets vorgegeben werden.

= 114 Einem Gitterpunkt wurde ein Punktattribut zugewiesen, welches dort nicht vorgebbar ist.

= -127 Der angegebene Minimalwert für die Anzahl der durchzuführenden Mehrgitterzyklen ist fehlerhaft. Es muß gelten:
1 ≤ NZKLMN.
Intern wurde daher NZKLMN : = 1 verwendet.

= -128 Der angegebene Maximalwert für die Anzahl der durchzuführenden Mehrgitterzyklen ist fehlerhaft. Es muß gelten:
NZKLMN ≤ NZKLMX.
Deshalb wurde intern mit NZKLMX : = NZKLMN gearbeitet.

= 131 Der angegebene Wert für IAUFR ist kleiner als –1, erlaubt sind jedoch nur Werte ≥ –1, s. Parameterbeschreibung IAUFR.

= -141 Die Berechnungen können nur auf einem Gitter durchgeführt werden, da z. B. die Anzahl der Teilintervalle zwischen zwei parallelen Randlinien ungerade ist. Dies reduziert die Effi-

zienz der Berechnung äußerst stark. Bitte das Gitter so verändern, wie in Kapitel 3 beschrieben.

= -142 Die Berechnungen können nur auf zwei Gittern durchgeführt werden, da z. B. die Anzahl der Teilintervalle zwischen zwei parallelen Randlinien zwar gerade, aber nicht durch vier teilbar ist. Dies reduziert stark die Effizienz der Berechnung. Bitte das Gitter so verändern, wie in Kapitel 3 beschrieben.

= -143 Die Anzahl der Punkte des gröbsten Gitters ist sehr groß. Dadurch verringert sich die Effizienz des Mehrgitterverfahrens. Bitte das Gitter so verändern, wie in Kapitel 3 beschrieben.

Rechengenauigkeit:

= -301 Die vorgegebene Schranke RFLRMX für den relativen Fehler von PHI ist kleiner als die verfügbare Maschinengenauigkeit. Intern wurde daher mit einem größeren Wert gearbeitet.

= -302 Die vorgegebene Schranke RFLRMX für den relativen Fehler von PHI wurde nach Durchführung der höchstzulässigen Anzahl (NZKLMX) von Mehrgitterzyklen noch nicht unterschritten. Eine weitere Fehlerverkleinerung ist möglich, wenn weitere Mehrgitterzyklen durchgeführt werden. Dazu müssen die Parameter AFLRMX und IAUFR wie folgt gesetzt sein:
AFLRMX = 0.0, IAUFR > 1.

= -303 Die vorgegebene Schranke AFLRMX für den absoluten Fehler von PHI wurde noch nicht unterschritten. Eine weitere Fehlerverkleinerung ist möglich, wenn weitere Mehrgitterzyklen durchgeführt werden. Dazu müssen die Parameter RFLRMX und IAUFR wie folgt gesetzt sein:
RFLRMX = 0.0, IAUFR > 1.

Fehler in der Parallelisierung:

= 400 Ein anderer Prozeß dieser Anwendung endete unnormal. Deshalb wurden auch die übrigen Prozesse abgebrochen.

= 411 Fehler bei der Erzeugung der Knotenprozesse *).

= 412 Fehler beim Versenden der Eingabedaten an die Knotenprozesse *).

= 413 Fehler beim Versenden der Aufforderung an die Knotenprozesse zur Rücksendung der Teilergebnisse *).

= 415 Die Dimensionierungen der Ergebnismatrizen in Haupt- bzw. Steuerprozeß einerseits und in den Knotenprozessen andererseits stimmen nicht überein *).

= 416 Fehler beim Empfangen der Betragsmaxima der Lösung und des absoluten Fehlers *).

Tritt einer dieser Fehler auf, so enthält die im Falle IDRUCK>0 ausgedruckte Fehlernachricht einen Code der Form

pppppp-mmmmmm-fff.

Dabei ist

pppppp: die logische Nummer des Prozesses, in dem dieser Fehler festgestellt wurde
(=0 oder H: Haupt-, =S: Steuerprozeß, =X: Nummer ist unbekannt),

mmmmmm: Name des internen Programm-Moduls, in dem der Fehler aufgetreten ist (Aufstellung s.u.),

fff: einer der oben bzw. in Abschnitt 2.2.2 beschriebenen Fehlercodes.

Ein Zusatz der Form

K=kkkk

bedeutet, daß der Fehler zuerst in einem Modul der Kommunikationsbibliothek [15] festgestellt wurde; kkkk ist der entsprechende Fehlercode dieses Moduls.

*) Interne Fehler:

Die Fehlercodes 411, 412, 413, 415 u. 416, sowie einige in dieser Tabelle nicht aufgeführte Fehler, können nur unter besonderen Voraussetzungen auftreten, z.B. wenn eine Eingabematrix im aufrufenden Programm zu klein deklariert wurde, oder wenn die in korrespondierenden Programmaufrufen in Haupt- bzw. Steuerprozeß einerseits und in den Knotenprozessen andererseits übergebenen Werte für IAUFR nicht übereinstimmen. In diesem Fall sollten Eingabeparameter, Deklarationen etc. gründlich überprüft werden.

Wichtiger Hinweis:

Tritt einer der Fehler 411, 412, 413, 415 oder 416 auf, so wird die parallele Berechnung umgehend terminiert. Andere Prozesse, in denen kein Fehler festgestellt wird, können in diesem Fall mit Fehlercode 400 enden.

Verzeichnis der Unterprogramme

EPOT1H Steuerprogramm für die Potentialberechnung (nur vom Hauptprozeß aus aufzurufen), mit Bereitstellung des Potentials im Hauptprozeß;

EPOT2H Steuerprogramm für die Potentialberechnung (nur vom Hauptprozeß aus aufzurufen), ohne Bereitstellung des Potentials im Hauptprozeß;

EPOT3S Steuerprogramm für die Potentialberechnung (kann vom Haupt- oder von einem speziellen Steuerprozeß aufgerufen werden), mit Bereitstellung des Potentials im Steuerprozeß;

EPOT3A Steuerprogramm zum Verteilen der Eingabedaten an die parallelen Knotenprozesse zur Potentialberechnung (kann vom Haupt- oder von einem speziellen Steuerprozeß aufgerufen werden, Aufruf nur in Verbindung mit EPOT3B);

EPOT3B Steuerprogramm zum Empfangen des Potentials von den parallelen Knotenprozessen (kann vom Haupt- oder von einem speziellen Steuerprozeß aufgerufen werden, Aufruf nur in Verbindung mit EPOT3A);

EFLD1H Steuerprogramm für die Feldberechnung (nur vom Hauptprozeß aus aufzurufen), mit Bereitstellung des elektrischen Felds im Hauptprozeß;

EFLD2H Steuerprogramm für die Feldberechnung (nur vom Hauptprozeß aus aufzurufen), ohne Bereitstellung des elektrischen Felds im Hauptprozeß;

EFLD3S Steuerprogramm für die Feldberechnung (kann vom Haupt- oder von einem speziellen Steuerprozeß aufgerufen werden), mit Bereitstellung des elektrischen Felds im Steuerprozeß;

EFLD3A Steuerprogramm zum Verteilen der Eingabedaten an die parallelen Knotenprozesse zur Feldberechnung (kann vom Haupt- oder von einem speziellen Steuerprozeß aufgerufen werden, Aufruf nur in Verbindung mit EFLD3B);

EFLD3B Steuerprogramm zum Empfangen des elektrostatischen Felds von den parallelen Knotenprozessen (kann vom Haupt- oder von einem speziellen Steuerprozeß aufgerufen werden, Aufruf nur in Verbindung mit EFLD3A);

MPE092 Überprüfung der Eingabedaten und Ausdruck von Fehlermeldungen und Warnungen (Aufruf durch EPOT1H, EPOT2H, EFLD1H oder EFLD2H);

MPE093 Berechnung der maximal möglichen Vergröberung bei Durchführung von Mehrgittermethoden;

MPE095 Ermittlung der Maschinengenauigkeit;

MPE096 Berechnung der Gitterstufe, auf der ein gegebener Index noch „sichtbar" ist;

MPE099 Tabellarischer Ausdruck von Fehlern und Konvergenzraten;

MPE410 Erzeugung der parallelen Knotenprozesse auf der Basis einer Aufteilung des Gebiets in wagerechte „Streifen";

MPE411 Versand der Eingabedaten an die parallelen Knotenprozesse;

MPE412 Versand der Eingabedaten an die parallelen Knotenprozesse;

MPE413 Versand der Eingabedaten an die parallelen Knotenprozesse;

MPE416 Empfang der Endergebnisse (z- und r-Komponenten EZ und ER des elektrostatischen Felds) von den parallelen Knotenprozessen;

MPE417 Empfang der Endergebnisse (elektrostatisches Potential PHI) von den parallelen Knotenprozessen;

MPE418 Empfang des globalen absoluten Fehlers bzw. des Lösungsmaximums;

MPE419 Berechnung der maximalen Anzahl der zur Durchführung von Mehrgittermethoden geeigneten Gitter und von Steuerungsintormationen;

MPE449 Erwarten und Empfangen der Fertig-Meldungen der Knotenprozesse;

MPE480 Erzeugung der parallelen Knotenprozesse auf der Basis einer Aufteilung des Gebiets in wagerechte „Streifen";

MPE483 Versand der Eingabedaten an die parallelen Knotenprozesse;

MPE486 Empfang der Endergebnisse (z- und r-Komponenten des elektrostatischen Felds) von den parallelen Knotenprozessen;

MPE487 Empfang der Endergebnisse (elektrostatisches Potential) von den parallelen Knotenprozessen;

MPE492 Überprüfung der Eingabedaten (Aufruf durch EPOT3S, EPOT3A, EFLD3S oder EFLD3A); wird ein Fehler festgestellt, so wird die parallele Berechnung mit einer entsprechenden Fehlernachricht abgebrochen. Soweit dort kein anderer Fehler entdeckt wird, enden alle anderen Prozesse dann mit dem Fehlercode 400. Im Gegensatz zum Programm MPE092, das in EPOT1H, EPOT2H, EFLD1H oder EFLD2H aufgerufen wird, werden von MPE492 **keine Warnungen** ausgegeben.

CRGR2D, GLOPH, INTRPT, RECRA, SENDIA, SENDRA
Module der Kommunikationsbibliothek [15].

Verzeichnis der verwendeten Nachrichtenkennungen

Alle Nachrichten, die innerhalb dieses Programmpakets zwischen Prozessen ausgetauscht werden, enthalten eine Nachrichtenkennung („Tag"), die dem Empfängerprozeß das Erkennen und damit das gezielte Empfangen der erwarteten Nachricht gestattet.

Es ist jeweils die verwendete Nachrichtenkennung sowie der Inhalt der Nachricht angegeben. „V" bedeutet, daß die Nachricht vom Haupt- bzw. Steuerprozeß an die Knotenprozesse geschickt, „E", daß sie von dort empfangen wird.

Prozeß-Erzeugung und Versand der Eingabedaten an die Knoten

25000 / 25050 / 25100 / 25200 / 25300 / 25330 / 25380
V Versand von PHI / RHO / ZK / RK / PPAR / IPAT / IFALL

25001 V Parameter-Vektor des Typs „REAL"

25002 V Parameter-Vektor des Typs „INTEGER"

Empfangen des Endergebnisses von den Knotenprozessen

25400 V Aufforderung zur Rücksendung des jeweiligen Teilergebnisses

25400 E Empfang der jeweiligen Teilergebnisse (Potential PHI)

25435 E Empfang der jeweiligen Teilergebnisse (Feldkomponente EZ)

25465 E Empfang der jeweiligen Teilergebnisse (Feldkomponente ER)

Empfangen der Betragsmaxima der Lösung und des absoluten Fehlers

25999 E Betragsmaxima der Lösung und des absoluten Fehlers zur Berechnung von Konvergenzraten und zum Ausdruck entsprechender Tabellen

Programmtechnische Daten

Programmiersprache: SUPRENUM-FORTRAN
Zur Steuerung der parallelen Prozesse (Erzeugung, Nachrichtenaustausch) werden ausschließlich die jeweiligen Module der für SUPRENUM entwickelten Kommunikationsbibliothek [15] verwendet, so daß auf maschinenspezifische Sprachelemente insoweit verzichtet werden konnte.

Typ: SUBROUTINE

COMMON-Bereiche: Keine

Dateizugriffe: EPOT1H, EPOT2H, EFLD1H und EFLD2H:
Standard-Ausgabekanal:
Ausgabe von Fehlermeldungen, Warnungen und Hinweisen, siehe die Parameterbeschreibung IDRUCK.
EPOT3S, -3A, -3B, EFLD3S, -3A und -3B:
Keine reguläre Ein-/Ausgabe von Daten. Ausgabe kurzer Fehlernachrichten über die Konsole.

Stand: 12.12.1989

Autor: Manfred Alef
Kernforschungszentrum Karlsruhe GmbH
Institut für Datenverarbeitung in der Technik
Postfach 36 40, D-7500 Karlsruhe 1

2.2.2 Knotenprozesse (EPOT1K/-2K/-3K, EFLD1K/-2K,/3K)

Diese Programme bearbeiten jeweils einen waagerechten Streifen des Gesamtgitters, der ihnen vom Hauptprozeß zugewiesen wird.

Für Berechnungen in der Nachbarschaft innerer Ränder (Grenzbereiche zu von Nachbarprozessen bearbeiteten Teilgebieten) werden ständig Zwischenergebnisse dieser Nachbarn benötigt. Zu deren Abspeicherung sind einige Matrizen etwas größer deklariert als es dem behandelten Gebietsteil entspricht. Dieser Überlappungsbereich umfaßt jeweils die beiden nächstgelegenen Gitterlinien des oberen und des unteren Nachbarprozesses. Lediglich die Prozesse, die den logisch untersten bzw. obersten Streifen des Gitters bearbeiten, enthalten an der Gebietsaußenseite keinen Überlappungsbereich.

Aufrufe zur Potentialberechnung

Gegenstück zum Aufruf des Programms EPOT1H im Hauptprozeß
(s. Abschnitt 2.2.1):

```
      CALL EPOT1K (A, NDIMA, I, NDIMI, IAUFR, IFLR)
```

Gegenstück zum Aufruf des Programms EPOT2H im Hauptprozeß
(s. Abschnitt 2.2.1):

```
      CALL EPOT2K (A, NDIMA, I, NDIMI,
     *             IVPHI, IVRHO, IVZK, IVRK,
     *             IVFALL, IVPAT, IVHOM,
     *             NX, NY, IYBU, IYAU, IYAO, IYBO, NPAT,
     *             IAUFR, IFLR)
```

Gegenstück zum Aufruf entweder des Programms EPOT3S oder der Programmkombination EPOT3A und EPOT3B im Steuerprozeß
(s. Abschnitt 2.2.1):

```
      CALL EPOT3K (A, NDIMA, I, NDIMI, IAUFR, IFLR)
```

Aufrufe zur Feldberechnung

Gegenstück zum Aufruf des Programms EFLD1H im Hauptprozeß
(s. Abschnitt 2.2.1):

```
      CALL EFLD1K (A, NDIMA, I, NDIMI, IAUFR, IFLR)
```

Gegenstück zum Aufruf des Programms EFLD2H im Hauptprozeß
(s. Abschnitt 2.2.1):

```
      CALL EFLD2K (A, NDIMA, I, NDIMI,
     *             IVEZ, EVER, IVPHI, IVRHO, IVZK, IVRK,
     *             IVFALL, IVPAT, IVHOM,
     *             NX, NY, IYBU, IYAU, IYAO, IYBO, NPAT,
     *             IAUFR, IFLR)
```

Gegenstück zum Aufruf entweder des Programms EFLD3S oder der
Programmkombination EFLD3A und EFLD3B im Steuerprozeß
(s. Abschnitt 2.2.1):

```
      CALL EFLD3K (A, NDIMA, I, NDIMI, IAUFR, IFLR)
```

Parameter

Die folgende Tabelle gibt Auskunft über Funktion, Typ und Deklaration der einzelnen Parameter (s. Erläuterungen in Abschnitt 2.2.1).

Name	Funktion	Typ	Dimension
A	*Hilfsspeicher, Ein-/Ausgabe*	REAL*8	(NDIMA)

Hilfsvektor, in dem das Potential PHI, die z- und r-Komponenten EZ und ER des elektrostatischen Feldes, die Raumladungsdichte RHO, die z- und r-Koordinaten, die Diskretisierungskoeffizienten sowie verschiedene andere Parameter abgelegt werden. Als Eingabe dient nur die Raumladungsdichte RHO (Anfangsadresse in A: IVRHO), als Ausgabe nur das Potential PHI (Anfangsadresse: IVPHI) und die Feldstärken EZ und ER (Anfangsadressen: IVEZ und IVER). Auch auf die Gitterkoordinaten (Anfangsadressen: IVZK und IVRK) kann von außen zugegriffen werden, dies sollte jedoch nur in Ausnahmefällen geschehen. Der Zugriff auf diese Daten ist jedoch nur bei Verwendung eines der Programmpaare EPOT2H / -2K zur Potential- oder EFLD2H / -2K zur Feldberechnung möglich, da die Ergebnisse ansonsten nur im Haupt- bzw. Steuerprozeß ausgewertet werden.
(Dimensionierungen von PHI, EZ, ER, RHO etc.: s.u.)
Bei der Vorgabe von RHO sind die in Abschnitt 2.2.1.3 genannten Regeln zu beachten.
Beispiel 2 in Kapitel 5 (Abschnitt 5.2.2) demonstriert den Zugriff auf den Vektor A in den Knotenprozessen.

NDIMA	*Eingabe*	INTEGER

Dimension des Vektors A, wie sie im aufrufenden Programm erfolgte. Reicht dieser Wert nicht aus, dann wird der Lauf abgebrochen.
Der erforderliche Mindestwert beträgt ungefähr

14*NX*(IYBO-IYBU) bei der Potential- und
16*NX*(IYBO-IYBU) bei der Feldberechnung.

I	*Ein-/Ausgabe*	INTEGER	(NDIMI)

Hilfsvektor zur Abspeicherung der Falltabelle IFALL (Anfangsadresse in I: IVFALL), der Punktattributmatrix IPAT (Anfangsadresse: IVPAT), dem Linienhomogenitätsvektor IHOM (Anfangsadresse: IVHOM) sowie einiger anderer interner Daten.

NDIMI	*Eingabe*	INTEGER

Dimension des Vektors I, wie sie im aufrufenden Programm erfolgte. Reicht dieser Wert nicht aus, dann wird der Lauf abgebrochen.
Der erforderliche Mindestwert beträgt ungefähr 2*NX*(IYBO–IYBU).

Name	Funktion	Typ	Dimension

IVPHI, ***Ausgabe (nur bei EPOT2K u. EFLD2K)*** **INTEGER**
IVEZ, Anfangsadressen von PHI, EZ, ER, RHO, ZK bzw. RK im Vektor A, s.o.
IVER,
IVRHO,
IVZK,
IVRK

IVFALL, ***Ausgabe (nur bei EPOT2K u. EFLD2K)*** **INTEGER**
IVPAT, Anfangsadressen von IFALL, IPAT bzw. IHOM im Vektor I, s.o.
IVHOM

NX, NY ***Ausgabe (nur bei EPOT2K u. EFLD2K)*** **INTEGER**
Anzahl der Gitterpunkte in x- bzw. y-Richtung.

IYAU, ***Ausgabe (nur bei EPOT2K u. EFLD2K)*** **INTEGER**
IYAO Unterer bzw. oberer y-Index des vom jeweiligen Prozeß bearbeiteten Gitterstreifens ohne Berücksichtigung des Überlappungsbereichs.
War der Prozeß inaktiv, dann ist IYAU = IYAO = 0. Dies ist immer dann für einige Knotenprozesse der Fall, wenn intern weniger Knotenprozesse als vorgegeben verwendet werden, d.h. falls im Hauptprozeß NPR kleiner als NPRMX ist, s. Abschnitt 2.2.1.3.

IYBU, ***Ausgabe (nur bei EPOT2K u. EFLD2K)*** **INTEGER**
IYBO Unterer bzw. oberer y-Index des vom jeweiligen Prozeß bearbeiteten Gitterstreifens mit Berücksichtigung des Überlappungsbereichs, d.h. IYBU $\leqq$ IYAU, IYBO $\geqq$ IYAO.
Bei einem inaktiven Prozeß wird IYBU = IYBO = 0 gesetzt.

NPAT ***Ausgabe (nur bei EPOT2K u. EFLD2K)*** **INTEGER**
Begrenzung der Anzahl der Punktattribute (IPAT = I (IVPAT)); für alle (IX,IY) mit $1 \leqq IX \leqq NX$, $1 \leqq IY \leqq NY$ muß gelten:

$$-NPAT \leqq IPAT(IX,IY) \leqq +NPAT.$$

Name	Funktion	Typ	Dimension

IAUFR *Ein-/Ausgabe* INTEGER

Aufruf-Parameter, s. Abschnitt 2.2.1.3.

Es ist sehr sorgfältig darauf zu achten, daß die Angaben für IAUFR in den korrespondierenden Programmaufrufen in Haupt- bzw. Steuerprozeß einerseits und Knotenprozessen andererseits übereinstimmen. Da es leider nicht möglich ist, alle denkbaren, fehlerhaften Kombinationen zu erkennen und den Programmlauf mit einer entsprechenden Fahlermeldung abzubrechen, können abweichende Angaben für IAUFR zu unkontrollierbaren Fehlersituationen führen.

Abweichend davon darf allerdings im Haupt- bzw. Steuerprozeß IAUFR = 9 und in den Knotenprozessen IAUFR = 2 angegeben werden. In diesem Fall wird in den Knotenprozessen von EPOTxK bzw. EFLDxK ebenfalls IAUFR: = 9 (sowie ferner IFLR: = -999, falls kein Fehler mit Fehlercode IFLR>0 erkannt wurde) zurückgegeben. Damit ist es möglich, z.B. die Knotenprozesse vom Hauptprozeß aus zu terminieren, indem im Knoten-Hauptprogramm diese Parameter abgefragt werden.

IFLR *Ausgabe* INTEGER

Fehlerindikator:

= 0 Es wurden keine Fehler festgestellt.

> 0 Es wurde ein so schwerwiegender Fehler festgestellt, daß die Berechnung nicht durchgeführt werden konnte.

< 0 Warnung (z. B. ungünstige Wahl der Eingabeparameter), die Berechnung wurde jedoch vollendet.

Der Parameter IDRUCK (s. Abschnitt 2.2.1) gestattet es, anzugeben, ob Fehlernachrichten bzw. Warnhinweise ausgedruckt werden sollen oder nicht.

Die Fehlermeldungen im einzelnen:

Fehler in den Eingabedaten:

= 198 Die angegebene Dimension von A (NDIMA) reicht nicht aus.
Ungefähr erforderlicher Mindestwert:

NDIMA = 14 * NX * (IYBO–IYBU) (Potential-),

NDIMA = 16 * NX * (IYBO–IYBU) (Feldberechnung).

= 199 Die angegebene Dimension von I (NDIMI) reicht nicht aus.
Ungefähr erforderlicher Mindestwert:
NDIMI = 2 * NX * (IYBO–IYBU).

Laufzeitfehler, -warnungen:

= 201 Das Gitter besitzt in einem Punkt eine Singularität, d.h. zwei parallele Gitterlinien haben einen zu geringen Abstand.

= 202 Punktart 4 oder 8 wurde in einem Punkt vorgegeben, wo diese Punktarten nicht vorgebbar sind.

= -203 Die Punktart 4 wurde auf der Rotationsachse vorgegeben. Stattdessen wurde hier Punktart 8 angenommen.

= 204 Punktart 3 oder 4 wurde in einem Punkt vorgegeben, in denen diese Punktarten nicht vorgebbar sind.

Fehler in der Parallelisierung (vgl. Abschnitt 2.2.1.3):

= 400 Ein anderer Partnerprozeß endete unnormal.

= 401 Nicht harmonierende Programmkombination:
Es sind nur die in Tabelle 2.2.0.3 genannten Programmkombinationen in Haupt- bzw. Steuerprozeß und in den Knotenprozessen erlaubt.

= 402 Nicht harmonierende Programmkombination:
Im Haupt- oder Steuerprozeß wurde ein Programm zur Potential- (EPOTxx), in den Knotenprozessen dagegen eines zur Feldberechnung (EFLDxK) aufgerufen.

= 403 Nicht harmonierende Programmkombination:
Im Haupt- oder Steuerprozeß wurde ein Programm zur Feldberechnung (EFLDxx), in den Knotenprozessen dagegen eines zur Potentialberechnung (EPOTxK) aufgerufen.

= 409 Fehlerhafter Parameter beim Aufruf eines Kommunikationsprogramms *).

= 420 Die Angaben für IAUFR in den korrespondierenden Aufrufen im Haupt- bzw. Steuerprozeß und in den Knotenprozessen stimmen nicht überein *).

= 421 Fehler bei der Initialisierung der Prozesse *).

= 422 Fehler beim Empfangen von Eingabedaten *).

= 423 Fehler beim Empfangen der Aufforderung zur Ergebnisrücksendung *).

= 424 Fehler bei der Ergebnisrücksendung *).

= 426 Fehler beim Berechnen der Betragsmaxima der Lösung und des absoluten Fehlers *).

= 431 Fehler beim Datentausch (Versand) *).
= 432 Fehler beim Datentausch (Versand) *).
= 433 Fehler beim Datentausch (Versand) *).
= 439 Fehler bei der Aufforderung zur Zusendung von Daten *).
= 441 Fehler beim Datentausch (Empfang) *).
= 442 Fehler beim Datentausch (Empfang) *).
= 443 Fehler beim Datentausch (Empfang) *).
= 449 Fehler bei der Aufforderung zur Zusendung von Daten *).

Achtungshinweis seitens des Hauptprozesses:

= -999 Im Haupt- oder Steuerprozeß wurde IAUFR = 9 gesetzt. Daraufhin wird in den Knotenprozessen ebenfalls IAUFR: = 9 zurückgegeben, und dieser Fehlercode wird gesetzt (jedoch nur, falls kein Fehlerzustand mit Fehlerindikator IFLR > 0 bemerkt wurde).

*) Interne Fehler:

Die Fehlercodes 409, 421, 422, 423, 424, 426, 431, 432, 433, 439, 441, 442, 443 und 449, sowie einige in dieser Tabelle nicht aufgeführte Fehler, weisen auf einen Fehler im Programmaufruf hin, z.B. wenn die Dimensionierung von Vektoren im aufrufenden Programm zu klein gewählt wurde, wenn die nur für die interne Zwischenspeicherung von Daten vorgesehenen Bereiche in A oder in I zwischen zwei aufeinanderfolgenden Aufrufen verändert werden, oder wenn die in korrespondierenden Programmaufrufen in Haupt- bzw. Steuerprozeß einerseits und in den Knotenprozessen andererseits übergebenen Werte für IAUFR nicht übereinstimmen. In diesem Fall sollten alle Eingabeparameter, Deklarationen etc. gründlich überprüft werden.

Wichtiger Hinweis:

Tritt einer der Fehler 409, 421, 422, 423, 424, 426, 431, 432, 433, 439, 441, 442, 443 oder 449 auf, so wird die parallele Berechnung umgehend terminiert. Andere Prozesse, in denen kein Fehler festgestellt wird, können in diesem Fall mit Fehlercode 400 enden.

Interne Belegung des Hilfsvektors A

In den Knotenprozessen ist beim Aufruf der Programme EPOTxK bzw. EFLDxK ein REAL-Vektor der Länge NDIMA bereitzustellen. Darin werden intern die folgenden, im Falle der Programme EPOT2K und EFLD2K auch für den Benutzer interessanten Daten in der in Abb. 2.2.2.1 dargestellten Weise abgespeichert:

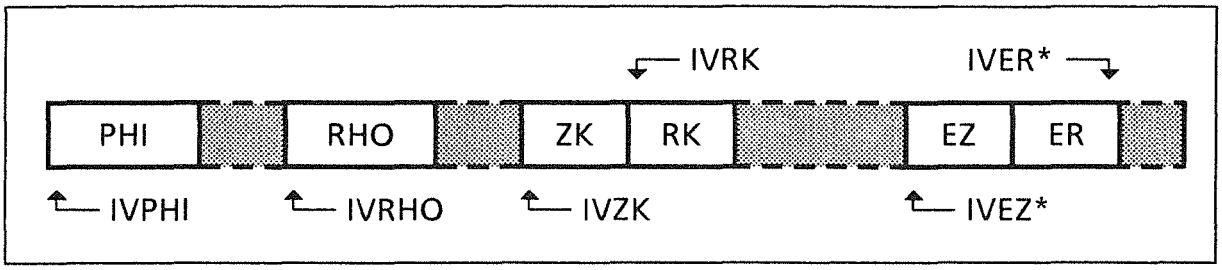


Abb. 2.2.2.1: Anordnung der für den Benutzer interessanten Daten im Hilfsvektor A. (: nur bei EFLDxK)*

Der Teilvektor PHI enthält, beginnend mit dem Element A(IVPHI), die folgenden Werte in dieser Reihenfolge:

```
PHI(1,IYBU), PHI(2,IYBU), ... , PHI(NX,IYBU), PHI(1,IYBU+1), ... ,
PHI(NX,IYBU+1), PHI(1,IYBU+2), ... , PHI(NX,IYBO-1), PHI(1,IYBO),
PHI(2,IYBO), ... , PHI(NX,IYBO).
```

Der Teilvektor RHO enthält, beginnend mit dem Element A(IVRHO), die folgenden Werte in dieser Reihenfolge:

```
RHO(1,IYAU), RHO(2,IYAU), ... , RHO(NX,IYAU), RHO(1,IYAU+1), ... ,
RHO(NX,IYAU+1), RHO(1,IYAU+2), ... , RHO(NX,IYAO-1), RHO(1,IYAO),
RHO(2,IYAO), ... , RHO(NX,IYAO).
```

Der Teilvektor ZK enthält, beginnend mit dem Element A(IVZK), die folgenden Werte in dieser Reihenfolge:

```
ZK(1,IYBU), ZK(2,IYBU), ... , ZK(NX,IYBU), ZK(1,IYBU+1), ... ,
ZK(NX,IYBU+1), ZK(1,IYBU+2), ... , ZK(NX,IYBO-1), ZK(1,IYBO),
ZK(2,IYBO), ... , ZK(NX,IYBO).
```

Der Teilvektor RK ist, beginnend mit dem Element A(IVRK), in der gleichen Weise wie ZK angeordnet.

Bei Aufruf von EFLD2K enthält der Teilvektor EZ, beginnend mit dem Element A(IVEZ), die folgenden Werte in dieser Reihenfolge:

```
EZ(1,IYAU), EZ(2,IYAU), ... , EZ(NX,IYAU), EZ(1,IYAU+1), ... ,
EZ(NX,IYAU+1), EZ(1,IYAU+2), ... , EZ(NX,IYAO-1), EZ(1,IYAO),
EZ(2,IYAO), ... , EZ(NX,IYAO).
```

Der Teilvektor ER ist, beginnend mit dem Element A(IVRK), in der gleichen Weise wie EZ angeordnet.

Die anderen, in Abb. 2.2.2.1 grau gerasterten Bereiche von A enthalten interne Zwischenergebnisse, die für den Benutzer nicht von Interesse sind und vor allem auch nicht in irgendeiner Weise zwischen aufeinanderfolgenden Aufrufen der Programme EPOTxK oder EFLDxK verändert werden dürfen.

Beim Zugriff auf PHI, EZ, ER, RHO, ZK und RK achte man unbedingt darauf, daß PHI, ZK und RK einen Überlappungsbereich enthalten (der y-Index IY variiert zwischen IYBU und IYBO), EZ, ER und RHO dagegen nicht (y-Index zwischen IYAU und IYAO).

An dieser Stelle sei darauf hingewiesen, das sich die Adressen IVPHI, IVEZ, EVER, IVRHO, IVZK und IVRK selbst innerhalb desselben Laufs zwischen den verschiedenen Knotenprozessen, bedingt durch eine nicht 100 %-ig gleichmäßige Aufteilbarkeit des Gebiets, in der Regel unterscheiden.

Interne Belegung des Hilfsvektors I

In den Knotenprozessen ist beim Aufruf der Programme EPOTxK bzw. EFLDxK ein INTEGER-Vektor der Länge NDIMI bereitzustellen. Darin werden intern die folgenden, im Falle der Programme EPOT2K und EFLD2K auch für den Benutzer interessanten Daten in der in Abb. 2.2.2.2 dargestellten Weise abgespeichert:

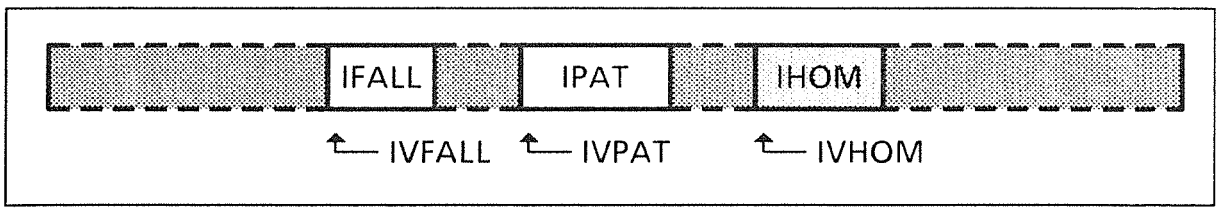


Abb. 2.2.2.2: Anordnung der für den Benutzer interessanten Daten im Hilfsvektor I.

Der Teilvektor IFALL enthält, beginnend mit dem Element I(IVFALL), die folgenden Werte in dieser Reihenfolge:

```
IFALL(-NPAT), IFALL(-NPAT+1), ... , IFALL(-1), IFALL(0),
IFALL(1), ... , IFALL(+NPAT-1), IFALL(+NPAT).
```

Der Teilvektor IPAT enthält, beginnend mit dem Element I(IVPAT), die folgenden Werte in dieser Reihenfolge:

```
IPAT(1,IYBU), IPAT(2,IYBU), ... , IPAT(NX,IYBU),
IPAT(1,IYBU+1), ... , IPAT(NX,IYBU+1), IPAT(1,IYBU+2), ... ,
IPAT(NX,IYBO-1), IPAT(1,IYBO), IPAT(2,IYBO), ... , IPAT(NX,IYBO).
```

Die in Abbildung 2.2.2.2 grau gerasterten Bereiche von I enthalten interne Zwischenergebnisse, die für den Benutzer nicht von Interesse sind und insbesondere auch nicht in irgendeiner Weise zwischen aufeinanderfolgenden Aufrufen von EPOTxK oder EFLDxK verändert werden dürfen.

Beim Zugriff auf IPAT ist der Überlappungsbereich zu beachten (der y-Index IY variiert zwischen IYBU und IYBO).

Auch die Ausgabedaten IVFALL, IVPAT und IVHOM unterscheiden sich in der Regel zwischen den verschiedenen Knotenprozessen eines Laufs.

Verzeichnis der Unterprogramme

EPOT1K Programmsteuerung (Potentialberechnung);
EPOT2K Programmsteuerung (Potentialberechnung);
EPOT3K Programmsteuerung (Potentialberechnung);
EFLD1K Programmsteuerung (Feldberechnung);
EFLD2K Programmsteuerung (Feldberechnung);
EFLD2K Programmsteuerung (Feldberechnung);
MPE010 Steuerprogramm zur Vorbereitung des Mehrgitterprozesses: Berechnung der Diskretisierungskoeffizienten auf allen Gittern, Übertragung der erforderlichen Daten vom feinsten auf die groben Gitter;
MPE011 Übertragung von auf dem feinsten Gitter gegebenen Daten auf die groben Gitter;
MPE012 Übertragung von auf dem feinsten Gitter gegebenen Daten auf die groben Gitter (Vorbereitung der Anfangsschätzung mittels der „Full Multigrid"-Technik);
MPE013 Übertragung der Punktattribute auf das nächstgröbere Gitter;

MPE017 Untersuchung der „Homogenität" der einzelnen waagerechten Gitterlinien und Berechnung von „Linienattributen" zur Vermeidung von Abfragen in später aufgerufenen Programm-Modulen;

MPE020 Berechnung der Diskretisierungskoeffizienten im feinsten Gitter (Steuerprogramm);

MPE021, MPE022, MPE023, MPE024
Berechnung der Diskretisierungskoeffizienten im feinsten Gitter;

MPE030 Berechnung der Diskretisierungskoeffizienten auf den groben Gittern (Steuerprogramm);

MPE031, MPE032, MPE033, MPE034
Berechnung der Diskretisierungskoeffizienten auf den groben Gittern;

MPE100 Berechnung einer Anfangsschätzung für das Potential PHI mittels der „Full Multigrid"-Technik;

MPE110 Steuerprogramm zur Durchführung eines Mehrgitterzyklus;

MPE111 Übergang vom feinen zum nächstgröberen Gitter (Relaxation, Berechnung des Residuums und Restriktion);

MPE112 Übergang vom groben zum nächstfeineren Gitter (Interpolation und Addition der Korrekturen, Relaxation);

MPE120 Steuerprogramm zur Durchführung einer vorgegebenen Anzahl von Glättungsschritten („Vier-Farb-Schachbrett"-artige Gauß-Seidel-Relaxation);

MPE121 Relaxation auf der „untersten" Gitterlinie;

MPE122 Relaxation zwischen zweiter Gitterlinie von unten bzw. oben;

MPE123 Relaxation auf der „obersten" Gitterlinie;

MPE130 Steuerprogramm zur Berechnung des Residuums und dessen Übertragung auf das nächstgröbere Gitter mittels „Full Weighting";

MPE131 Residuumsberechnung auf der „untersten" Gitterlinie;

MPE132 Residuumsberechnung zwischen zweiter Gitterlinie von unten bzw. oben, Inkrement 1;

MPE133 Residuumsberechnung zwischen zweiter Gitterlinie von unten bzw. oben, Inkrement 2 (Berücksichtigung der Tatsache, daß beim hier gewählten Relaxationsverfahren der Wert des Residuums in den Punkten des vierten Relaxations-Teilschritts verschwindet);

MPE134 Residuumsberechnung auf der „obersten" Gitterlinie;

MPE140 Steuerprogramm zur linearen Interpolation und Addition der Korrekturen (dabei werden diejenigen Punkte ausgeklammert, in denen im ersten Relaxationsteilschritt neue Werte berechnet werden);

MPE141 Waagerechte lineare Interpolation;

MPE142 Senkrechte lineare Interpolation;
MPE150 Steuerprogramm zur Berechnung einer Schätzung für den absoluten und den relativen Fehler;
MPE151 Fehlerschätzung auf der „untersten" Gitterlinie;
MPE152 Fehlerschätzung zwischen zweiter Gitterlinie von unten bzw. oben;
MPE153 Fehlerschätzung auf der „obersten" Gitterlinie;
MPE158 Steuerprogramm zur Fehlerschätzung auf dem gröbsten Gitter (Verfahren: Lösung der Fehler-Defekt-Gleichung durch einen Jacobi-Relaxationsschritt, ausgehend von 0);
MPE159 Steuerprogramm zur Fehlerschätzung auf den feineren Gittern (Verfahren: Berechnung des Maximums der im Mehrgitterprozeß berechneten Korrektur);
MPE160 Kubische Interpolation der Näherungslösung bei der „Full Multigrid"-Technik;
MPE161 Waagerechte kubische Interpolation;
MPE171 bis MPE179
Senkrechte kubische Interpoletion in Abhängigkeit von der Lage der Gitterlinie im (Teil-)Rechengebiet;
MPE420 Empfang der Lage eines Prozesses im logischen Prozeßgitter und einiger globaler Daten;
MPE421 Empfang der Eingabedaten durch die parallelen Prozesse (Aufruf durch EPOT1K);
MPE422 Empfang der Eingabedaten durch die parallelen Prozesse (Aufruf durch EPOT2K und EFLD2K);
MPE423 Empfang der Eingabedaten durch die parallelen Prozesse (Aufruf durch EPOT3K, EFLD1K und EFLD3K);
MPE426 Versand der Endergebnisse (Feld) an den Haupt- bzw. Steuerprozeß;
MPE427 Versand der Endergebnisse (Potential) an Haupt- bzw. Steuerprozeß;
MPE428 Versand des lokalen und Empfang des globalen absoluten Fehlers und Lösungsmaximums;
MPE429 Berechnung der Anfangsadressen der verschiedenen Gitter innerhalb der lokalen Vektoren in A und I;
MPE431 Austausch von Daten zwischen Nachbarprozessen;
MPE432 Versand eines einzelnen REAL-Vektors an einen Nachbarprozeß;
MPE433 Versand eines einzelnen INTEGER-Vektors an einen Nachbarprozeß;
MPE434 Versand eines einzelnen REAL-Vektors an einen Nachbarprozeß;
MPE435 Versand eines einzelnen INTEGER-Vektors an einen Nachbarprozeß;
MPE436 Versand eines einzelnen REAL-Vektors an einen Nachbarprozeß;

MPE439 Versand einer Fertig-Nachricht an den Hauptprozeß;
MPE441 Austausch von Daten zwischen Nachbarprozessen;
MPE442 Empfang eines einzelnen REAL-Vektors von einem Nachbarprozeß;
MPE443 Empfang eines einzelnen INTEGER-Vektors von einem Nachbarprozeß;
MPE444 Empfang eines einzelnen REAL-Vektors von einem Nachbarprozeß;
MPE445 Empfang eines einzelnen INTEGER-Vektors von einem Nachbarprozeß;
MPE446 Empfang eines einzelnen REAL-Vektors von einem Nachbarprozeß;
GLOPS, GRID2D, INTRPT, RECIA, RECRA, SENDIA, SENDRA
Module der Kommunikationsbibliothek [15].

Verzeichnis der verwendeten Nachrichtenkennungen

Es ist jeweils die verwendete Nachrichtenkennung sowie der Inhalt der Nachricht angegeben. „H" bedeutet, daß die Nachricht an den Hauptprozeß geschickt, „E", daß sie von dort empfangen, „K", daß sie zwischen Knotenprozessen ausgetauscht wird.

Prozeß-Erzeugung und Annahme der Eingabedaten von den Knoten

25001 E Parameter-Vektor des Typs „REAL"
25002 E Parameter-Vektor des Typs „INTEGER"
25010 E Empfang von PHI
25020 E Empfang von RHO
25030 E Empfang von ZK
25040 E Empfang von RK
25050 E Empfang von PPAR
25060 E Empfang von IPAT
25070 E Empfang von IFALL

Initialisierung (Berechnung der Diskretisierungskoeffizienten u.s.w.)

25499 K Aufforderung zum Versand von Daten
25500, 25510, 25520
K Austausch von IPAT
25540, 25560, 25580
K Austausch von C (Matrix der Diskretisierungskoeffizienten)

Zusammenlegung von Prozessen während der Behandlung grober Gitter (Agglomeration)

25600, 25610, 25620, 25650, 25660, 25670

K Austausch von RHO

25700, 25710, 25720, 25750, 25760, 25770

K Austausch von PHI

Relaxation

25100, 25150

K Versand von PHI im Überlappungsbereich an den unteren bzw. Empfang vom oberen Nachbarn

25120, 25170

K Dto., jedoch in umgekehrter Richtung

Restriktion

25850 K Versand des Residuums im Grenzbereich an den unteren bzw. Empfang vom oberen Nachbarn

25870 K Dto., jedoch in umgekehrter Richtung

Kubische Interpolation

25800 K Versand von PHI im Grenzbereich des feineren Gitters an den unteren bzw. Empfang vom oberen Nachbarn

25820 K Dto., jedoch in umgekehrter Richtung

25900 K Versand von PHI im Grenzbereich des gröberen Gitters an den unteren bzw. Empfang vom oberen Nachbarn

25920 K Dto., jedoch in umgekehrter Richtung

Feldberechnung

25000 K Versand von PHI im Überlappungsbereich an den unteren bzw. Empfang vom oberen Nachbarn

25020 K Dto., jedoch in umgekehrter Richtung

Versand des Endergebnisses an die Knoten

25400 E Aufforderung zur Rücksendung des jeweiligen Teilergebnisses

25400 H Versand der jeweiligen Teilergebnisse (Potential PHI)

25435 H Versand der jeweiligen Teilergebnisse (Feldkomponente EZ)

25465 H Versand der jeweiligen Teilergebnisse (Feldkomponente ER)

Berechnung der Betragsmaxima der Lösung und des absoluten Fehlers

25998 K Betragsmaxima der Lösung und des absoluten Fehlers im Teilgebiet zur Ermittlung der Gesamtmaxima

25999 K/H Betragsmaxima der Lösung und des absoluten Fehlers im Gesamtgebiet

Programmtechnische Daten

Programmiersprache:	SUPRENUM-FORTRAN Zur Steuerung der parallelen Prozesse (Erzeugung, Nachrichtenaustausch) werden ausschließlich die entsprechenden Module der für SUPRENUM entwickelten Kommunikationsbibliothek [15] verwendet, maschinenspezifische Sprachelemente werden insoweit nicht benutzt.
Typ:	SUBROUTINE
COMMON-Bereiche:	Keine
Dateizugriffe:	Keine
Stand:	12.12.1989
Autor:	Manfred Alef Kernforschungszentrum Karlsruhe GmbH Institut für Datenverarbeitung in der Technik Postfach 36 40, D-7500 Karlsruhe 1

2.2.3 Beschreibung der verfügbaren Punktarten

Die Programme unterstützen die folgenden Punktarten (vgl. Parameterbeschreibung zu PPAR) [12]:

Punktart 1 (Feldpunkt):

Zur Berechnung des Potentials in Feldpunkten wird die Poisson-Gleichung gelöst. Die Diskretisierung erfolgt mittels einer Mehrstellendiskretisierung zweiter Ordnung.

Die Vorgabe von Feldpunkten ist in der Regel nur in inneren Gitterpunkten zulässig, d. h. es muß gelten:

$(IX = 1 \vee IX = NX \vee IY = 1 \vee IY = NY) \Rightarrow IFALL(IPAT(IX,IY)) \neq 1.$

RHO muß mit $+\rho/\varepsilon$ initialisiert sein.

Punktart 2 (Dirichlet-Randpunkt):

Direkte Vorgabe eines Sollwerts für das Potential an Anode oder Kathode entweder direkt über einen entsprechenden Eintrag in PHI, oder indirekt über die Parametertabelle PPAR.

Dirichlet-Randpunkte können sowohl auf äußeren als auch auf inneren Rändern vorgegeben werden. Im Inneren des Gebiets, d. h. in Punkten mit $1 < IX < NX$ und $1 < IY < NY$, dürfen Dirichlet-Bedingungen jedoch nur auf Flächen und nicht auf einzelnen Linien vorgegeben werden, wobei die **Ränder** dieser „Dirichlet-Flächen" die in Abschnitt 2.2.4.2 genannten Bedingungen erfüllen müssen.
RHO wird in Dirichlet-Randpunkten nicht ausgewertet, ρ/ε braucht dort also nicht vorgegeben zu werden.

Punktart 3 (Randpunkt mit verschwindender Normalableitung in x-Richtung):

In diesen Randpunkten wird die Bedingung gestellt, daß die Ableitung des Potentials in Richtung der Normale an die Gitterlinie IX = const. verschwinden soll. Zur Berechnung des Potentials werden Diskretisierungen der Poisson-Gleichung und der Normalableitung geeignet kombiniert. Diese Vorgehensweise hat verschiedene numerische und programmtechnische Vorteile.

Diese Punktart ist auf dem (im logischen Gitter) linken oder rechten Rand des Gebiets, jedoch nicht in den vier Eckpunkten, vorgebbar.

Punktart 4 (Randpunkt mit verschwindender Normalableitung in y-Richtung):

Diese Punktart ist auf dem (im logischen Gitter) oberen oder unteren Rand des Gebiets, jedoch nicht in den vier Eckpunkten, vorgebbar.

In Punkten 1<IX<NX, IY = 1 darf diese Punktart nur dann vorgegeben werden, wenn für diese Punkte RK(IX,1) >0 ist. Auf der Symmetrieachse ist stattdessen die Punktart 8 zu wählen.

Punktart 5 (Punkt mit Sprung der Normalableitung in x-Richtung):

Diese Punktart auf Grenzlinien zwischen Teilgebieten mit unterschiedlichen Dielektrizitätskonstanten vorgegeben.

Als rechte Seite ist RHO = + 2 ρ / (ε_0 ($\varepsilon_+ + \varepsilon_-$)) zu setzen, wobei ε_0 die absolute, ε_- die relative Permittivität im links und ε_+ die im rechts angrenzenden Gebietsteil, bezogen auf das logische Gitter, ist.

Die Vorgabe dieser Punktart ist nur gestattet, wenn
2 < IX < NX–1.

Punktart 6 (Punkt mit Sprung der Normalableitung in y-Richtung):

Diese Punktart wird auf Grenzlinien zwischen Teilgebieten mit unterschiedlichen Dielektrizitätskonstanten vorgegeben.

Als rechte Seite ist RHO = + 2 ρ / (ε_0 ($\varepsilon_+ + \varepsilon_-$)) zu setzen, wobei ε_0 die absolute, ε_- die relative Permittivität im unten und ε_+ die im oben angrenzenden Gebietsteil, bezogen auf das logische Gitter, ist.

Die Vorgabe dieser Punktart ist nur gestattet, wenn
2 < IY < NY–1.

Punktart 8 (Feldpunkt auf der Symmetrieachse):

Zur Bestimmung des Potentials wird hier wie in normalen Feldpunkten die Poisson-Gleichung gelöst, wobei die Rotationssymmetrie ausgenutzt wird. Dazu wird zur Elimination des 1/r-Terms in der Poisson-Gleichung zunächst ein Grenzübergang $r \rightarrow 0$ durchgeführt. Dies geschieht mittels der Regel von de l' Hospital unter Ausnutzung der Tatsache, daß wegen der Rotationssymmetrie die partielle Ableitung Φ_r auf der Symmetrieachse verschwindet.

Die Vorgabe dieser Punktart ist nur in Punkten (IX,IY) mit $1 < IX < NX$ und $IY = 1$ möglich, wenn dort $RK(IX,1) = 0$ für alle IX mit $1 \leq IX \leq NX$ gilt.

Ergänzung zu den Punktarten 3, 4 und 8:

RHO ist wie in Feldpunkten zu initialisieren ($RHO = +\rho/\varepsilon$).

Wichtig:
In Punkten $1 < IX < NX$, $IY = 1$ mit $RK(IX,1) = 0.0$ (Rotationsachse) dürfen grundsätzlich nur die Punktarten 2, 5 oder 8 vorgegeben werden.

2.2.4 Hinweise und Einschränkungen

2.2.4.1 Fehlerschranken

Die Parameter **RFLRMX** und **AFRLMX** beziehen sich nur auf den Fehler

$$\Phi_h^{(n)} - \Phi_h^*,$$

jedoch nicht auf den **Diskretisierungsfehler**

$$\Phi_h^* - \Phi^*,$$

der vor allem bei einem zu grobmaschigen Gitter relativ groß sein kann. Dabei bezeichnet $\Phi_h^{(n)}$ die nach n Mehrgitterzyklen erhaltene Näherung, Φ_h^* die **exakte diskrete Lösung (d.h. die fehlerfreie Lösung der diskretisierten Aufgabe)** und Φ^* die **exakte kontinuierliche Lösung,** die sich in der Regel von Φ_h^* unterscheidet.

2.2.4.2 Effizienzbedingung

Um einen effizienten Einsatz der Mehrgittermethoden zu gewährleisten, sind die im Kapitel 3 genannten Regeln für Gittererzeugung und Lage innerer Ränder unbedingt zu beachten. Insbesondere sind NX und NY so zu wählen, daß (NX–1) und (NY–1) durch eine möglichst hohe Zweierpotenz teilbar sind.

2.2.4.3 Prozeß-Erzeugung

Um eine parallele Berechnung durchführen zu können, muß zuerst die entsprechende Anzahl von Prozessen erzeugt worden sein. Dies geschieht durch Angabe des Parameters IAUFR = 0 oder = + 1. Werden die hier beschriebenen Programme mehrmals nacheinander aufgerufen (z.B. bei Berechnung des Potentials in einer Schleife mit jeweils anderen RHO-Werten), so ist beim ersten Aufruf IAUFR = 0 oder = 1, bei späteren Aufrufen dagegen IAUFR = –1 oder $\geqq$ + 2 anzugeben; dabei muß der gleiche Wert für NPRMX (s. Abschnitt 2.2.1.3) angegeben werden, wie bei der vorherigen Prozeßerzeugung.

2.2.4.4 Initialisierungs- / Wiederholungsaufruf

Zur Durchführung der Berechnung in einem randangepaßten Gitter müssen vorher die jeweiligen Diskretisierungskoeffizienten berechnet worden sein, die intern im Hilfsvektor A abgelegt werden. Dies geschieht immer bei Angabe des Parameters IAUFR = 0 bzw. = ± 1: bei IAUFR = 0 oder = – 1 werden nur die Diskretisierungskoeffizienten berechnet (die Parameter PHI, RHO, NZKLMN, NZKLMX, AFLRMX und RFLRMX bleiben dann unberücksichtigt), bei IAUFR = 1 wird zusätzlich das Potential PHI durch Lösung der Poisson-Gleichung sowie ggf. die z- und r-Komponenten EZ und ER des elektrostatischen Feldes ermittelt. Bei IAUFR = 0 oder = 1 werden zusätzlich die parallelen Prozesse erzeugt. Daneben ist es möglich, die Diskretisierungskoeffizienten als Eingabe vorzugeben, so daß diese nicht mehr neu berechnet zu werden brauchen. In diesem Fall ist IAUFR $\geqq$ 2 zu setzen. Dadurch ist es z. B. möglich, die Berechnungen nacheinander für unterschiedliche Raumladungsdichten durchzuführen, wobei zwischen zwei Aufrufen immer nur RHO (sowie PHI in Dirichlet-Randpunkten) modifiziert werden darf, während alle anderen Parameter (insbesondere auch solche zur Steuerung des Verfahrensablaufs, wie z.B. NPRMX, s. Abschnitt 2.2.1.3) unverändert sein müssen. Zusätzlich dürfen mittels der Punktparameter PPAR die Dirichlet-Randbedingungen verändert werden.

2.3 Nachrichtenaustausch zwischen Hauptprozeß und Knotenprozessen

Zum Versand von Daten vom Hauptprozeß an alle Knotenprozesse und umgekehrt sind die Programme VERT2H, VERT2K, SAML2H und SAML2K vorgesehen. (Die Verwendung dieser Programme ist nur in Verbindung mit EPOT2H / -K bzw. EFLD2H / -K sinnvoll, da in EPOT1H, -3S, -3A / -3B, EFLD1H, -3S und -3A / -3B sowie in EPOT1K, -3K, EFLD1K und -3K bereits entsprechende Funktionen integriert sind.)

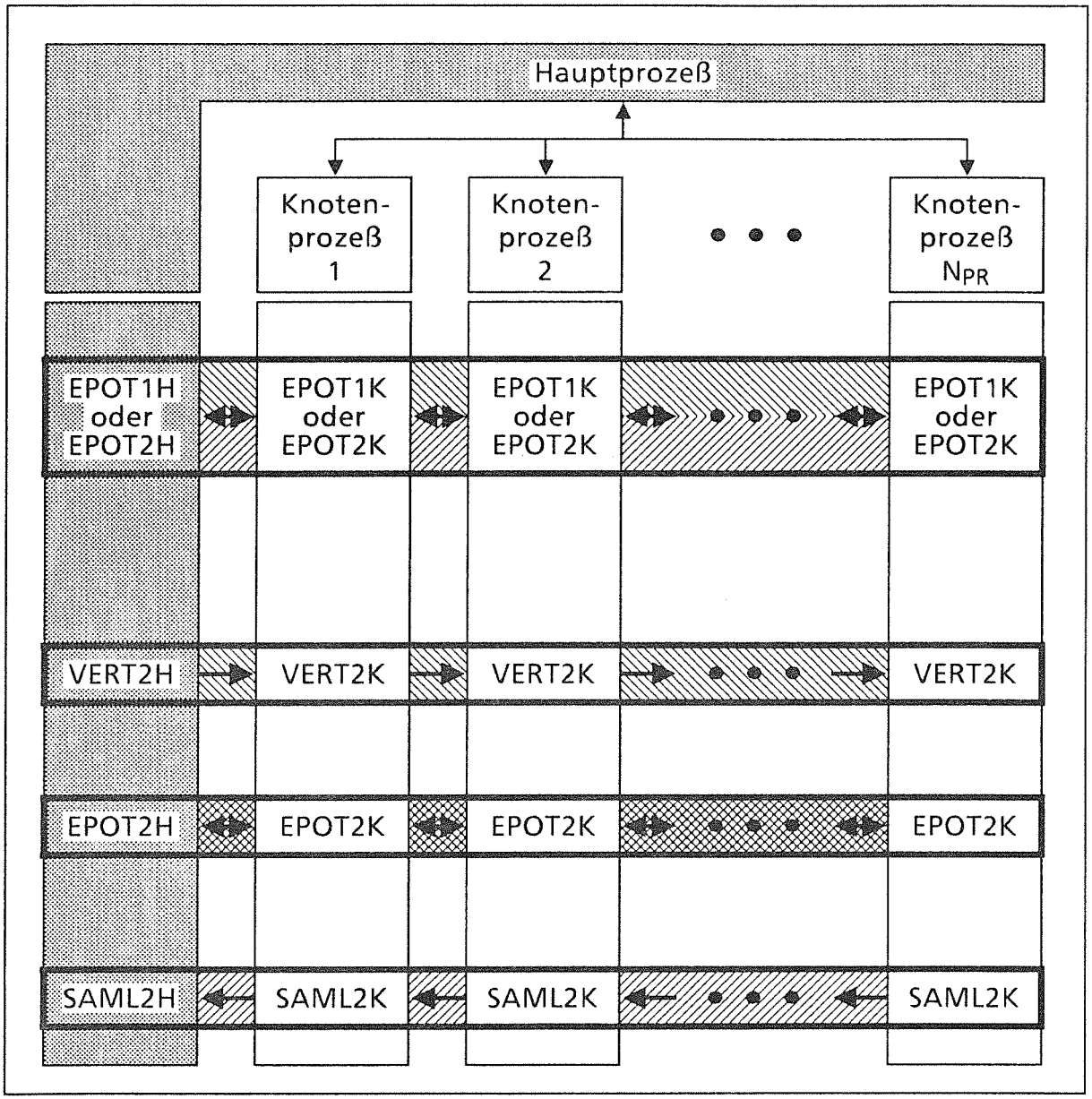


Abb. 2.3.0.1: Prozeß-Konzept: Korrespondierende Aufrufe der Programme VERT2H und SAML2H im Haupt- und VERT2K bzw. SAML2K in den Knotenprozessen.

2.3.1 Hauptprozeß (VERT2H und SAML2H)

Aufrufe

- **Verteilung einer Matrix auf die Knotenprozesse:**

```
      CALL VERT2H (A, NX, NY, NDIM1, NDIM2, NPRMX, NPR, IRAND,
     *             IDRUCK, IFLR)
```

- **Sammeln einer Matrix im Hauptprozeß:**

```
      CALL SAML2H (A, NX, NY, NDIM1, NDIM2, NPR,
     *             IDRUCK, IFLR)
```

Parameter

Die folgende Tabelle gibt Auskunft über Funktion, Typ und Deklaration der einzelnen Parameter.

Erläuterungen zur Tabelle

Name: Variablenname, wie er in den Programmen benutzt wird.

Funktion: Aufgabe des Parameters (Ein-, Ausgabe).

Typ: Typ der Variablen, wie er im Unterprogramm benötigt wird.

Dimension: Dimensionierung von Vektoren und Matrizen im Unterprogramm, die kleiner oder gleich der entsprechenden Dimension im aufrufenden Programm sein muß. Ist hier nichts angegeben, so handelt es sich um eine skalare Variable.

Name	Funktion	Typ	Dimension
A	*Eingabe (nur bei VERT2H),* *Ausgabe (nur bei SAML2H)*	REAL*8	(NDIM1,NDIM2)

Matrix, die auf NPR Knotenprozesse aufgeteilt (VERT2H) bzw. von diesen angenommen und zusammengefaßt (SAML2H) werden soll.

NX, NY *Eingabe* INTEGER
Anzahl der Gitterpunkte in x- bzw. y-Richtung.

NDIM1, NDIM2 *Eingabe* INTEGER
Dimensionierung der Matrix A im aufrufenden Programm.

NPRMX *Eingabe (nur bei VERT2H)* INTEGER
Maximale Anzahl der Knotenprozesse, vgl. die Beschreibung der Programme EPOT2H bzw. EFLD2H in Abschnitt 2.2.1.3.

NPR *Eingabe* INTEGER
Tatsächlich verwendete Anzahl der Knotenprozesse, auf die A aufgeteilt bzw. von denen A gesammelt werden soll. Dabei wird A in NPR–1 gleichbreite, waagerechte Streifen aufgeteilt; der oberste (NPR-te) Streifen kann jedoch weniger oder auch bis zu einer Gitterlinie mehr enthalten.

(Es ist der von EPOT2H bzw. EFLD2H zurückgegebene Wert anzugeben.)

IRAND *Eingabe (nur bei VERT2H)* INTEGER
IRAND gibt an, ob die an die Prozesse zu verteilenden Teilmatrizen von A einen Überlappungsbereich erhalten sollen (IRAND = 1) oder nicht (IRAND = 0).

IDRUCK *Eingabe* INTEGER
Steuerung der Druckausgabe (über den Standard-Ausgabekanal) im Fehlerfall:
Falls IDRUCK $\geqq$ 1, werden Fehlernachrichten gedruckt, sonst nicht.

Name	Funktion	Typ	Dimension
IFLR	*Ausgabe* Fehlerindikator:	INTEGER	

Fehler in den Eingabedaten:

= 104 Es wurde NX > NDIM1 angegeben.

= 105 Es wurde NY > NDIM2 angegeben.

= 135 Es wurde ein undefinierter Wert für IRAND angegeben.

Fehler in der Parallelisierung (vgl. Abschnitt 2.2.1.3):

= 400 Ein anderer Prozeß endete unnormal. Deshalb wurden auch die übrigen Prozesse abgebrochen.

= 406 Fehler beim Empfang der Annahmebestätigung von einem Knotenprozeß.

= 407 Fehler bei der Matrixaufteilung auf die Knotenprozesse.

= 413 Fehler beim Versenden der Aufforderung an die Knotenprozesse, ihre Teilmatrix zurückzusenden.

= 414 Fehler bei der Matrixzusammenfassung von den Knotenprozessen.

= 415 Die übersandte Matrix weist falsche Dimensionsgrenzen auf.

(Die Fehlercodes 406, 407, 413, 414 und 415, sowie einige weitere, nicht in der obigen Tabelle aufgeführten Codes, können nur unter ganz bestimmten Umständen auftreten, vgl. die entsprechenden Hinweise in Abschnitt 2.2.1.)

Wichtiger Hinweis:

Tritt einer der Fehler 406, 407, 413, 414 oder 415 auf, so wird die parallele Berechnung umgehend terminiert. Andere Prozesse, in denen kein Fehler festgestellt wird, können in diesem Fall mit Fehlercode 400 enden.

Verzeichnis der verwendeten Nachrichtenkennungen

Programm VERT2H

25900 Versand der Teilmatrizen an die zugehörigen Knotenprozesse

Programm SAML2H

25950 Versand einer Aufforderung an die Knotenprozesse, ihre zugehörige Teilmatrix zu senden

25950 Empfang der Teilmatrizen von den Knotenprozessen

Programmtechnische Daten

Programmiersprache:	SUPRENUM-FORTRAN
Typ:	SUBROUTINE
COMMON-Bereiche:	Keine
Dateizugriffe:	Standard-Ausgabekanal: Ausgabe von Fehlermeldungen
Stand:	04.12.1989
Autor:	Manfred Alef Kernforschungszentrum Karlsruhe GmbH Institut für Datenverarbeitung in der Technik Postfach 36 40, D-7500 Karlsruhe 1

2.3.2 Knotenprozesse (VERT2K und SAML2K)

Aufrufe

- **Annehmen einer Teilmatrix vom Hauptprozeß:**

```
CALL VERT2K (A, NX, IYU, IYO, IFLR)
```

- **Zurücksenden einer Teilmatrix an den Hauptprozeß:**

```
CALL SAML2K (A, NX, IYU, IYO, IYAU, IYAO, IFLR)
```

Parameter

Die folgende Tabelle gibt Auskunft über Funktion, Typ und Deklaration der einzelnen Parameter.

(Die Hinweise in Abschnitt 2.3.1 zur Interpretation der Tabelle sowie zu den programmtechnischen Fragen an Ende dieses Abschnitts gelten hier sinngemäß.)

Name	Funktion	Typ	Dimension
A	*VERT2K: Ausgabe*	REAL	(NX,IYU:IYO)
	Teilmatrix, die vom Hauptprozeß angenommen werden soll.		
	SAML2K: Eingabe	REAL	(NX,IYU:IYO)
	Teilmatrix, die an den Hauptprozeß gesendet werden soll.		
NX	*Eingabe*	INTEGER	
	Anzahl der Gitterpunkte in x-Richtung.		
IYU, IYO	*Eingabe*	INTEGER	
	Untere bzw. obere Dimensionsgrenze der Matrix A im aktuellen Knotenprozeß. Hierfür sind folgende Angaben erforderlich: - wenn A = PHI, dann ist IYU = IYBU und IYO = IYBO anzugeben, - wenn A = EZ, ER oder RHO, dann IYU = IYAU und IYO = IYAO (s. Parameterliste der Programme EPOT2K / EFLD2K in Abschnitt 2.2.2).		
IYAU, IYAO	*Eingabe (nur bei SAML2K)*	INTEGER	
	Unterer und oberer y-Index des zu übersendenden Bereichs. Dabei sind die gleichen Werte anzugeben, die vom Programm EPOT2K bzw. EFLD2K (s. Abschnitt 2.2.2) zuvor berechnet wurden.		
IFLR	*Ausgabe*	INTEGER	
	Fehlerindikator: Fehler in der Parallelisierung (vgl. Abschnitt 2.2.1.3): = 400 Ein anderer Prozeß endete unnormal. Deshalb wurden auch die übrigen Prozesse abgebrochen. = 406 Fehler beim Versand der Annahmebestätigung an den Hauptprozeß. = 408 Fehler bei der Annahme der Matrix vom Hauptprozeß. = 423 Fehler beim Empfangen der Aufforderung zur Rücksendung der Teilmatrix. = 424 Fehler bei Rücksendung der Teilmatrix an den Hauptprozeß. (Die Fehlercodes 406, 408, 423 und 424, sowie einige weitere, nicht in der obigen Tabelle aufgeführten Codes, können nur unter ganz bestimmten Umständen auftreten, vgl. die entsprechenden Hinweise in Abschnitt 2.2.2.)		

Wichtiger Hinweis:
Tritt einer der Fehler 406, 408, 423 oder 424 auf, so wird die parallele Berechnung umgehend terminiert. Andere Prozesse, in denen kein Fehler festgestellt wird, können in diesem Fall mit Fehlercode 400 enden.

Verzeichnis der verwendeten Nachrichtenkennungen

Programm VERT2K:

25900 Empfang der dem jeweiligen Knotenprozeß zugewiesenen Teilmatrix

Programm SAML2K:

25950 Empfang einer Aufforderung vom Hauptprozeß, die dem jeweiligen Knotenprozeß zugewiesene Teilmatrix zu senden

25950 Versand der Teilmatrizen an den Hauptprozeß

Programmtechnische Daten

Programmiersprache:	SUPRENUM-FORTRAN
Typ:	SUBROUTINE
COMMON-Bereiche:	Keine
Dateizugriffe:	Keine
Stand:	04.12.1989
Autor:	Manfred Alef Kernforschungszentrum Karlsruhe GmbH Institut für Datenverarbeitung in der Technik Postfach 36 40, D-7500 Karlsruhe 1

2.4 Anpassung von Eingabeparametern (SORPAH)

Die in diesem Bericht beschriebenen Potentialberechnungsrogramme sind eine Weiterentwicklung des Programms PSOR, welches auf dem SOR-Verfahren basiert [7,20]. Darüberhinaus unterscheiden sich diese Programme auch in der Formulierung der Benutzerschnittstelle, insbesondere in der Beschreibung der verschiedenen Punktarten und -parameter. Während die Punktarten in EPOTxx (bzw. entsprechend in EFLDxx) in Form einer Tabelle (IFALL, s. Abschnitt 2.1) beschrieben werden, die jedem möglichen Punktattribut (IPAT, s. Abschnitt 2.1) die zugehörige Punktart (Feldpunkt oder einer der verschiedenen Randpunktarten) unmittelbar in numerischer Form zuordnet, werden in PSOR mehrere Listen (IPATTL, PARA1L, PARA2L, PUBEZL) übergeben, die in PSOR intern ebenfalls in die bei EPOTxx angewendete Form überführt werden.

Um den Aufruf etwas übersichtlicher zu gestalten und dem Benutzer beide Möglichkeiten zur Dateneingabe zu bieten, wurde diese Transformation nicht in EPOTxx übernommen, sondern als eigenständiges Programm formuliert.

Dieses Programm SORPAH erhält als Eingabe die Liste IPATTL der vorkommenden Punktattribute, die zugehörige Punktbezeichner-Liste PUBEZL sowie die Punktparameterlisten PARA1L und PARA2L, und liefert als Ausgabe die für EPOTxx benötigten Punktart- und -parametertabellen IFALL und PPAR.

Dabei wird jedoch nicht überprüft, ob die vergebenen Punktarten in dieser Form auch sinnvoll sind, s. hierzu die entsprechenden Hinweise in der Beschreibung von EPOTxx.

Aufruf:

```
      CALL SORPAH (PPAR, PARA1L, PARA2L, PUBEZL, LILAE,
     *             IFALL, IPATTL, NPAT, IFLR)
```

Parameter:

Die folgende Tabelle gibt Auskunft über Funktion, Typ und Deklaration der einzelnen Parameter (Interpretation der Tabelle: vgl. Abschnitt 2.2.1.3).

Name	Funktion	Typ	Dimension

PPAR ***Ausgabe*** **REAL*8** **(-NPAT:NPAT,2)**

Punktparameter lt. folgender Definition:

PPAR(IPATTL(i),1) : = PARA1L(i), i = 1, ... , LILAE,

PPAR(IPATTL(i),2) : = PARA2L(i), i = 1, ... , LILAE

PARA1L, PARA2L ***Eingabe*** **REAL*8** **(LILAE)**

Punktparameter, s. Programmbeschreibung **PSOR** [20].

PUBEZL ***Eingabe (Ausgabe *)*** **CHARACTER*4** **(LILAE)**

Mnemotechnische Punktbezeichnungen, s. Programmbeschreibung zu **PSOR** [20] und Aufstellung unter IPATTL.

*) Die Punktbezeichnung 'ACHS' wird stets in 'RPN2' geändert, d.h. falls die Eingabe PUBEZL(i) = 'ACHS' lautet, wird PUBEZL(i) = 'RPN2' zurückgegeben. Damit ist es möglich, anschließend zu Kontrollzwecken Potentialberechnungen mittels PSOR durchzuführen.

LILAE ***Eingabe*** **INTEGER**

Listenlänge, d.h. Anzahl der Punktparameter und Punktbezeichnungen, s. Programmbeschreibung **PSOR** [20].

IFALL ***Ausgabe*** **INTEGER** **(-NPAT:NPAT)**

Tabelle, die den in IPAT für jeden Gitterpunkt enthaltenen Punktattributen die zugehörige Punktart zuordnet, vgl. Parameterbeschreibung IPATTL:

Mögliche Punktarten:

0	Punktattribut tritt in IPAT nicht auf
1	Feldpunkt im Inneren des Gebiets
2	Dirichlet-Randpunkt
3	Randpunkt mit Normalableitung = 0 in x-Richtung
4	Randpunkt mit Normalableitung = 0 in y-Richtung
5	Innerer Randpunkt mit Normalableitungssprung in x-Richtung
6	Innerer Randpunkt mit Normalableitungssprung in y-Richtung
8	Feldpunkt auf der Rotationsachse mit Symmetriebedingung

Name	Funktion	Typ	Dimension

IPATTL *Eingabe* **INTEGER (LILAE)**

Liste der vorkommenden Punktattribute. Zu jedem Element IPATTL(i) dieser Liste enthalten PARA1L(i) und PARA2L(i) die zugehörigen Punktparameter, wie z. B. Dirichlet-Randvorgaben, und PUBEZL(i) die Punktbezeichnung wie z. B. 'FELD' für Feld-, 'RPDI' für Dirichlet-Randpunkte, s. Parameterbeschreibung zum Programm **PSOR** [20].
Die Ausgabetabelle IFALL ordnet jedem Element IPATTL(i) die o.g. Punktartnummer zu:

```
PUBEZL(i) = 'FELD'      ⇒   IFALL(IPATTL(i)) = 1,
PUBEZL(i) = 'RPDI'      ⇒   IFALL(IPATTL(i)) = 2,
PUBEZL(i) = 'RPN1'      ⇒   IFALL(IPATTL(i)) = 3,
PUBEZL(i) = 'RPN2'      ⇒   IFALL(IPATTL(i)) = 4,
PUBEZL(i) = 'NAS1'      ⇒   IFALL(IPATTL(i)) = 5,
PUBEZL(i) = 'NAS2'      ⇒   IFALL(IPATTL(i)) = 6,
PUBEZL(i) = 'ACHS' 1)   ⇒   IFALL(IPATTL(i)) = 8,
i = 1, ... , LILAE.
```

1) Die Punktartbezeichnung 'ACHS' ist in PSOR nicht enthalten.

NPAT *Eingabe* **INTEGER**

Begrenzung der Anzahl der möglichen Punktattribute, vgl. Abschnitt 2.2.1.3.

IFLR *Ausgabe* **INTEGER**

Fehlerindikator:

= 0 Es wurden keine Fehler festgestellt.

= 190 Es wurde eine fehlerhafte Punktbezeichnung PUBEZL festgestellt.

Hinweise und Einschränkungen

- Die Programme PSOR und EPOTxx / EFLDxx sind bezüglich des Umfangs und der Verteilung der vorgebbaren Punktattribute nicht 100%ig kompatibel. Zum Beispiel ist in EPOTxx / EFLDxx die Vorgabe von Punktart 4 ('RPN2') auf der Symmetrieachse unzulässig und durch Punktart 8 ('ACHS') zu ersetzen, die wiederum in PSOR nicht vorgesehen ist. Weitere Unterschiede sind aus den Programmbeschreibungen ersichtlich. Diese Differenzen werden von SORPAH **nicht** berücksichtigt.

- SORPAH wird gewöhnlich unmittelbar nach dem Einlesen der Eingabedaten IPATTL, PUBEZL, PARA1L, PARA2L und LILAE aufgerufen. Wegen der Änderung der Punktbezeichnung 'ACHS' in 'RPN2' sind weitere Aufrufe, z.B. innerhalb der BFCPIC-Zeitschleife, nicht möglich.

Programmtechnische Daten

Programmiersprache:	FORTRAN 77
Typ:	SUBROUTINE
COMMON-Bereiche:	Keine
Dateizugriffe:	Standard-Ausgabekanal: Ausgabe von Fehlermeldungen
Stand:	14.12.1989
Autor:	Manfred Alef Kernforschungszentrum Karlsruhe GmbH Institut für Datenverarbeitung in der Technik Postfach 36 40, D-7500 Karlsruhe 1

3. Gitterkonzept und zugrundeliegender Algorithmus

Die Feldberechnungen erfolgen in einem randangepaßten Gitter (dem sogenannten „physikalischen Gitter"), dessen z- und r-Koordinaten vom Benutzer beim Aufruf zur Verfügung zu stellen sind. Wie dieses Gitter erzeugt wurde, ist für das Funktionieren des Programms nicht wesentlich, da zur Berechnung der Diskretisierungskoeffizienten nur die Koordinaten der Gitterpunkte benötigt werden.

Das Gitter muß eine „logische Rechteckstruktur" besitzen, d.h. es muß eine eineindeutige Zuordnung zwischen diesem Berechnungsgitter und einem sogenannten „logischen Gitter", einem äquidistanten Gitter in einem Rechteck, geben. Dies ist ein wesentlicher Unterschied z.B. zu einer Triangulierung (Unterteilung des Gebiets in Dreiecke) [16].

In diesem Bericht ist das logische Gitter der Indexraum der z- und r-Koordinaten des physikalischen Gitters, also die Menge

$$\{(IX,IY) \mid IX = 1, 2, \ldots, NX;\ IY = 1, \ldots, NY\}$$

mit natürlichen Zahlen NX und NY.

Auch die Gitterzellen des physikalischen Gitters sollten „möglichst quadratisch" sein, da stark verformte Zellen die erreichbare Genauigkeit (Diskretisierungsfehler) und das Konvergenzverhalten negativ beeinflussen können. **Dies gilt insbesondere für alle randnahen Maschen des Gitters.** „Möglichst quadratisch" heißt, daß sich die Gitterlinien möglichst rechtwinklig schneiden und daß alle vier Seiten einer Gitterzelle nahezu gleich lang sind; ein Seitenverhältnis zwischen kürzester und längster Seite von etwa 1:5 sollte möglichst nicht überschritten werden.

Andererseits sollte das Gitter speziell in der Nähe stark unregelmäßiger Randabschnitte, vor allem im Bereich einspringender Ecken, möglichst fein sein, um eine ausreichend hohe Genauigkeit zu erhalten. Auch in Bereichen, in denen die Raumladungsdichte RHO stark schwankt, soll das Gitter besonders fein sein.

Innere Ränder müssen mit Gitterlinien zusammenfallen.

Schließlich erfordert der Einsatz von Mehrgittermethoden, daß sämtliche (äußeren und inneren) Ränder, ferner möglichst alle „kritischen" Gitterpunkte, in denen eine Gitterlinie stark gekrümmt ist oder gar einen Knick aufweist, auf allen durchlaufenen Gittern „sichtbar" sind. Aus diesem Grund ist bei der Gittererzeugung darauf zu achten, daß sich zwischen allen parallelen Randlinien eine durch

eine möglichst hohe Zweierpotenz teilbare Anzahl von Zellen befindet. Dies gilt vor allem auch für den Abstand zwischen allen kritischen Stellen untereinander und zu Randlinien sowie zwischen den parallelen äußeren Rändern, d.h. für die Gesamtgröße des Gitters.

Gegebenenfalls sind noch weitere Gitterlinien einzufügen, um diese Bedingung zu erfüllen. Das ist in der Regel vorteilhafter, als wenn wegen einer ungünstig gewählten Randverteilung nur auf einem oder zwei Gittern gerechnet werden könnte.

Grundsätzlich ist es möglich, größere zusammenhängende Bereiche des Gitters als Dirichlet-Randbedingung zu kennzeichnen (Vergabe von Punktattributen IPAT(IX,IY) mit IFALL(IPAT(IX,IY)) = 2). Besser ist es jedoch, das Gitter so umzustrukturieren, daß solche „Dirichlet-Flächen" nicht vorkommen oder zumindest möglichst klein gehalten werden.

Als Beispiel wird nachfolgend ein Gitter in einer Pinchdiode beschrieben. Für weitere Hinweise zur Gittererzeugung am Beispiel einer selbstmagnetisch isolierten B_θ-Diode sei auf [17] verwiesen.

Um nun die elektrostatischen Potentiale numerisch berechnen zu können, sind zunächst alle Ableitungen sowohl der zugrundeliegenden Poisson-Gleichung als auch in den Randbedingungen im vorgegebenen Gitter geeignet zu diskretisieren. Durch diese Diskretisierungen ist ein lineares Gleichungssystem definiert, das mittels eines Mehrgitterverfahrens gelöst wird.

Diskretisierungen und Mehrgitteransatz sind in [12] detailliert beschrieben.

Beispiel: Gitter in einer Pinchdiode

In der Pinchdiode (s. Abb. 3.1) wird die Symmetrieachse als untere Begrenzung des Berechnungsgebiets gewählt. Der linke Rand ist durch die Anode mit einem vorgegebenen Potential Φ von z.B. 1,5 MV und der rechte durch die Kathode ($\Phi = 0$ V) gegeben. Auf diesen Rändern werden die genannten Potentialwerte als Dirichlet-Randbedingungen vorgegeben.

In dieser Diode treten im oberen Bereich kaum noch Ladungsträger auf, so daß die Raumladungsdichte ρ dort verschwindet und die Äquipotentiallinien fächerförmig, nahezu parallel verlaufen (Abb. 3.4). Aus diesem Grund wurde der obere Rand abgerundet, so daß er Anode und Kathode, aber auch alle anderen

Äquipotentiallinien rechtwinklig schneidet. Daher kann auf dieser Kurve die Neumann-Randbedingung $\partial\Phi / \partial n = 0$ gestellt werden.

Im Bereich der einspringenden Ecken der Kathode wurde das Gitter wegen der hier auftretenden extrem hohen elektrischen Feldstärken wesentlich verfeinert.

Das randangepaßte Gitter ist in Abb. 3.2, das zugehörige logische Gitter mit der Zuordnung von Punktattributen (IPAT) zu den Gitterpunkten sowie von Punktarten (IFALL) und -parametern (PPAR) zu diesen Punktattributen in Abb. 3.3 wiedergegeben. Abb. 3.4 zeigt die Äquipotentiallinien zu 0, 0,1, 0,2, . . . , 1,5 MV in der leeren Diode ($\rho = 0$).

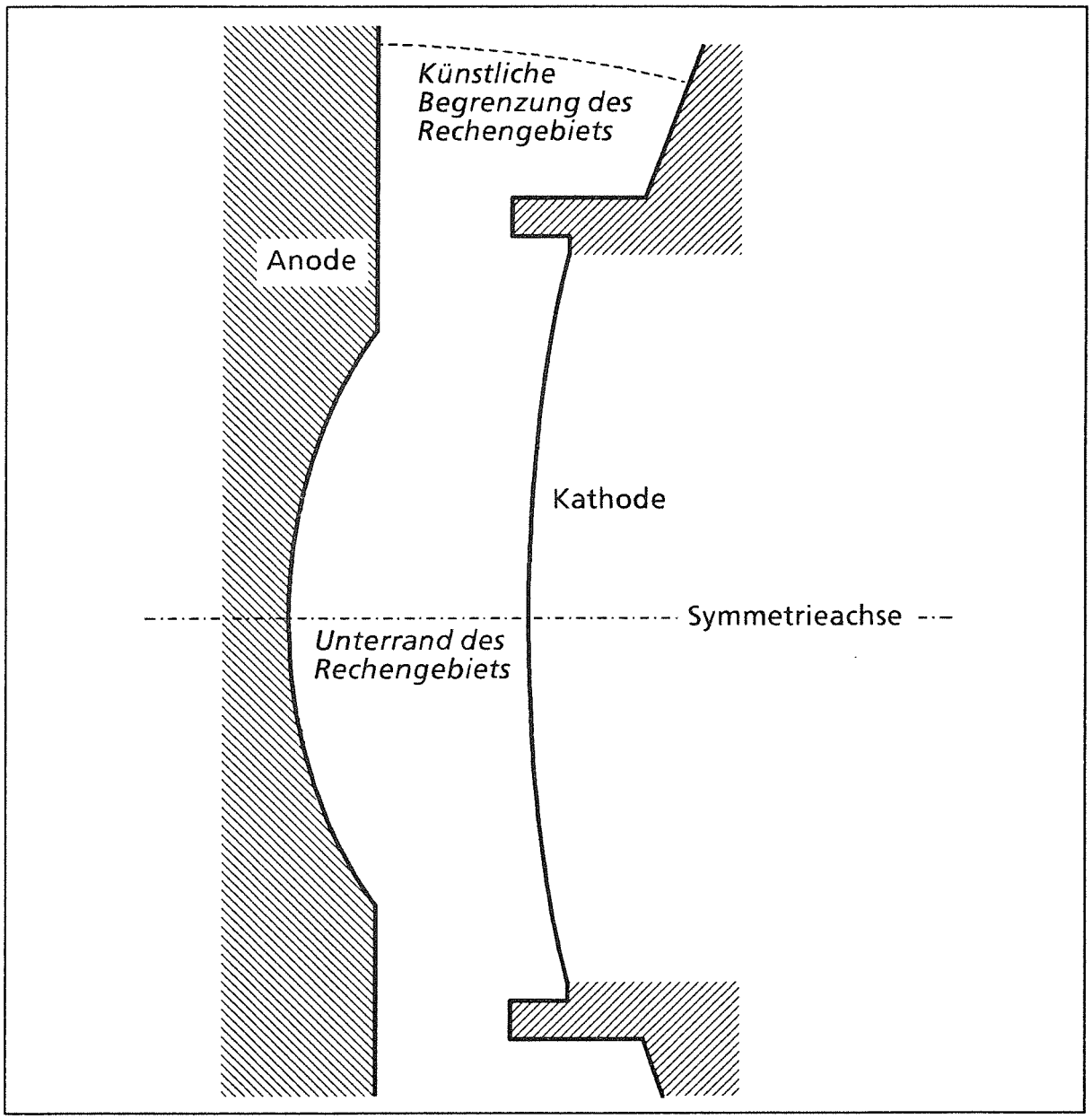


Abb. *3.1: Schematischer Querschnitt einer Pinchdiode (ohne Maßstab).*

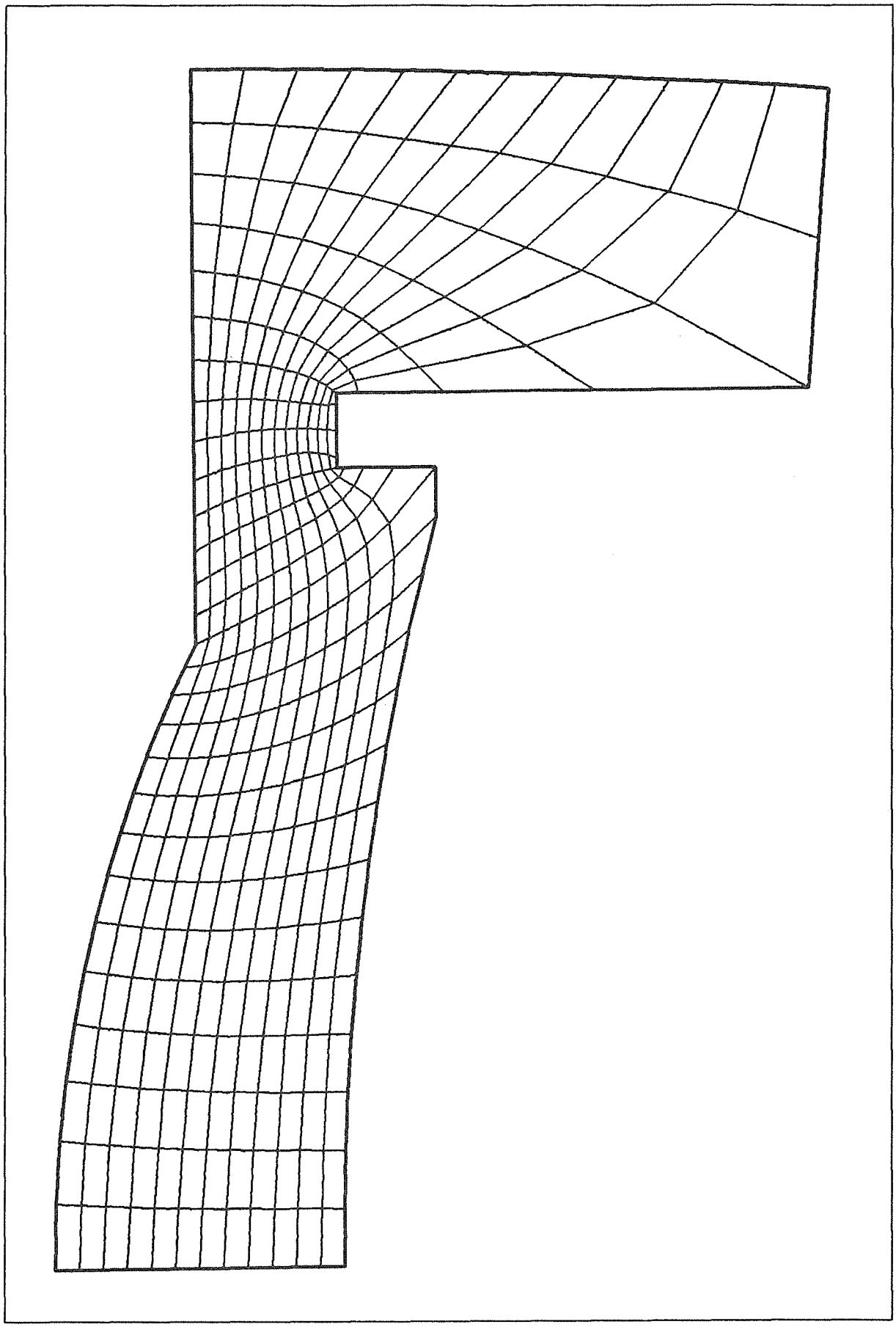
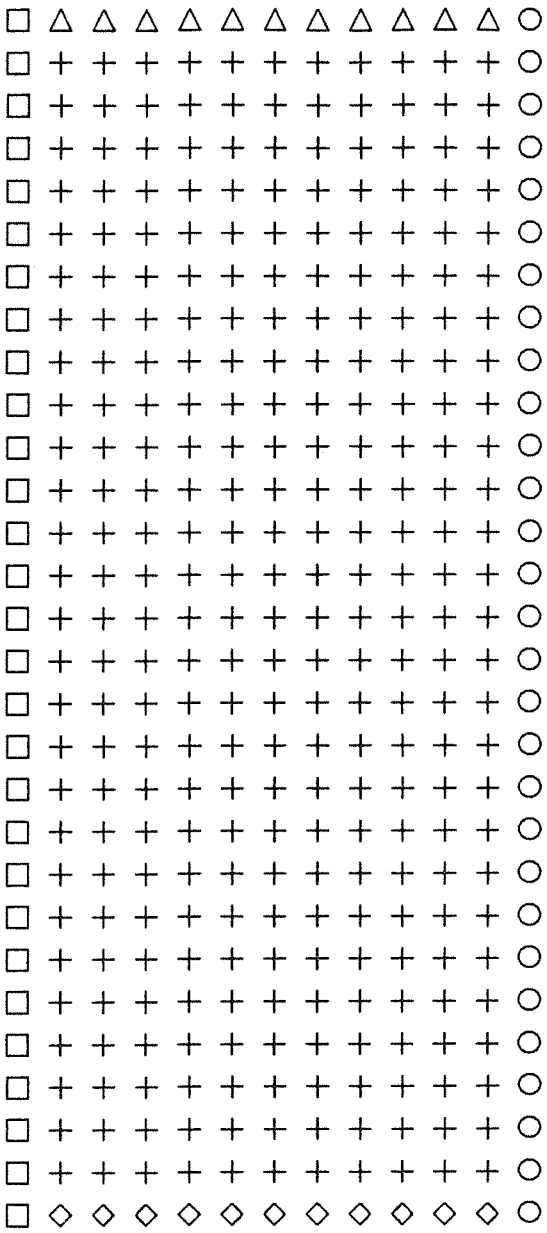


Abb. 3.2: Beispiel eines randangepaßten Gitters (Pinchdiode).



Symbol	Punktattribut (IPAT)	Punktart (IFALL)	Punktparameter (PPAR(*,1), PPAR(*,2)	
+	0	1	0.0	0.0
□	1	2	$1.5*10^6$	1.0
○	2	2	0.0	0.0
◇	3	8	0.0	0.0
△	4	4	0.0	0.0

Abb. 3.3: Logisches Gitter zum randangepaßten Gitter aus Abb. 3.2.

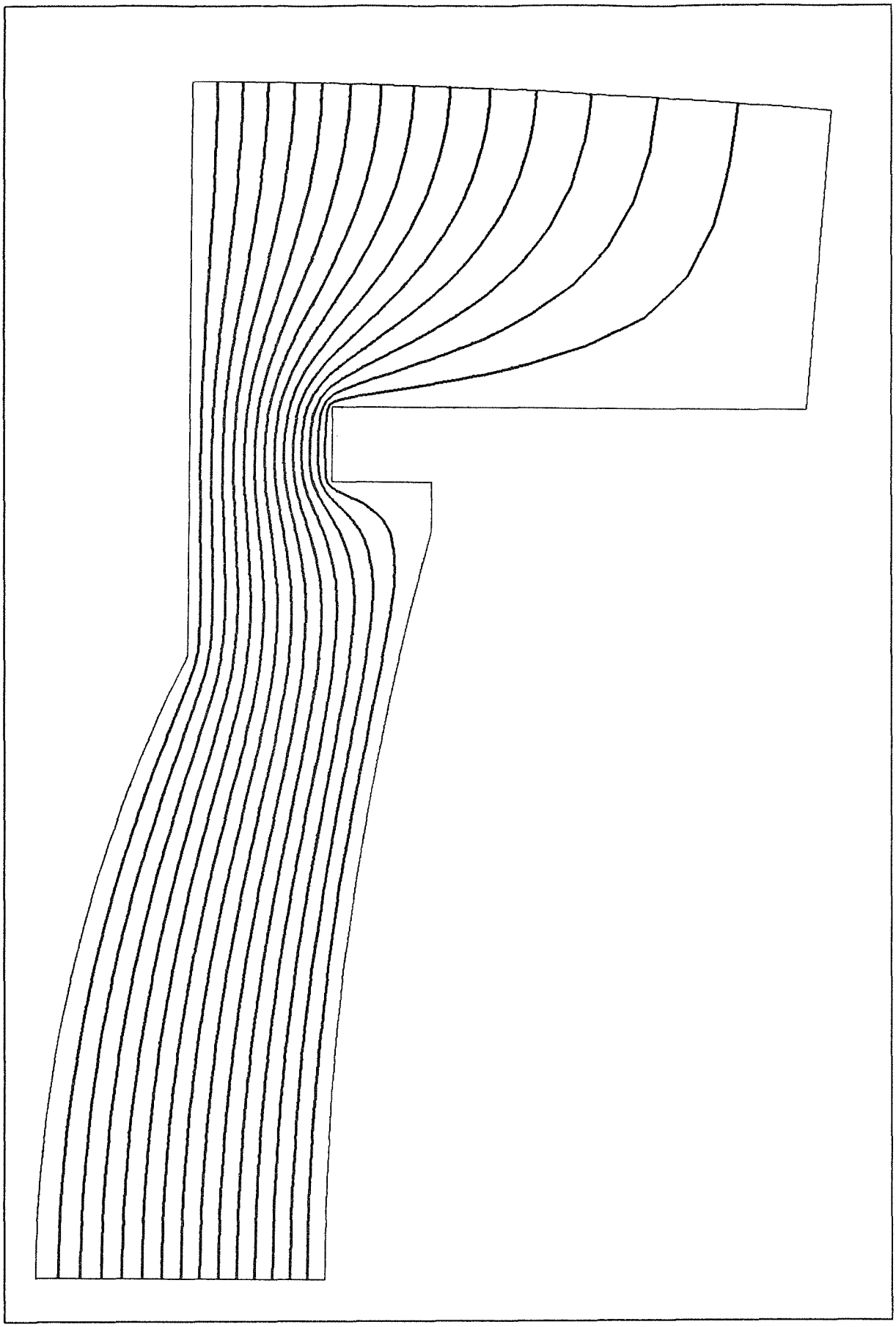


Abb. 3.4: Äquipotantiallinien in einer Pinchdiode.

4. Implementierung auf SUPRENUM

4.1 Beschreibung der SUPRENUM-Architektur

Der SUPRENUM-Rechner ist als Hochleistungsrechner speziell für rechenintensive Aufgaben vor allem in der numerischen Simulation konzipiert und basiert auf den folgenden Ideen [18]:

- Hochgradige Parallelverarbeitung.
- Ausnutzung der „lokalen Struktur" einer sehr großen, sehr wichtigen Klasse von Aufgaben (Gitterproblemen) in der Numerik, zu der auch die hier beschriebene Potentialberechnung mittels eines Differenzenverfahrens gehört.
- Unterstützung des Mehrgitterprinzips, womit sich herkömmliche Gitterprobleme drastisch beschleunigen lassen und welches ebenfalls bei der hier geschilderten Aufgabenstellung ausgenutzt wurde.

Die wichtigsten Hardwareeigenschaften sind [19]:

- Parallelrechner mit (zunächst) maximal 256 Knotenrechnern, wobei jeweils bis zu 16 Knoten zu einem „Cluster" zusammengefaßt sind.
- Die Knoten besitzen Skalar- und Vektorprozessoren sowie lokale Speicher von jeweils 8 MByte.
- Der Datenaustausch (Synchronisation) zwischen den Knoten erfolgt über den Austausch von Nachrichten.
- Die Knoten innerhalb der Cluster einerseits sowie die Cluster untereinander andererseits sind durch ein zweistufiges, schnelles Bussystem verbunden.
- Die Spitzenleistung eines Systems mit 256 Knoten wird 5 GFLOPS (= 5 Milliarden Gleitkommaoperationen pro Sekunde) betragen.

4.2 Parallelisierung der Programme

4.2.1 Prozeßkonzept

Ein Anwendungsprogramm, das auf SUPRENUM ablaufen soll, ist analog der Hardwarestruktur grundsätzlich wir folgt zu gliedern [10]:

Gliederung in Prozesse

Zentrales Element der Anwendung ist der Hauptprozeß („initialer Prozeß"). Dieser hat hauptsächlich die Aufgabe, den Ablauf der parallelen Berechnung zu koordinieren (Ein-/Ausgabe, Prozeßerzeugung etc.), führt selber jedoch normalerweise keine umfangreichen Berechnungen durch.

Die eigentlichen Berechnungen finden in den Knotenprozessen statt, die vom Hauptprozeß gestartet, mit den erforderlichen Eingabedaten versorgt und überwacht werden. Am Ende des Laufs werden die Ergebnisse vom Hauptprozeß „eingesammelt" und ausgegeben. Daneben können jederzeit Zwischenergebnisse dorthin gesandt werden. Eine direkte Daten-Ein-/Ausgabe durch die Knotenprozesse, außer der kurzzeitigen Zwischenspeicherung von Daten, ist dagegen nicht vorgesehen.

Jeder Knotenprozeß bearbeitet jeweils einen kleinen Teil der Gesamtaufgabe, dessen Umfang ihm vom Hauptprozeß mitgeteilt wird. Bei der in diesem Bericht vorgestellten Potential- und Feldberechnung geschieht dies durch eine geeignete Aufteilung des Gebiets (s. Abbildung 4.2.1 und Abschnitt 4.2.2) auf die Knotenprozesse.

Die Knotenprozesse können ihrerseits weitere Knotenprozesse erzeugen.

Aufbau der Prozesse

Die Prozesse sind grundsätzlich genauso aufgebaut und in Hauptprogramm und Unterprogramme gegliedert, wie Anwendungen für herkömmliche Rechner. Die Programme sind jetzt lediglich in korrespondierende Anteile aufgespalten, die vom Hauptprozeß bzw. von den Knotenprozessen aufgerufen werden.

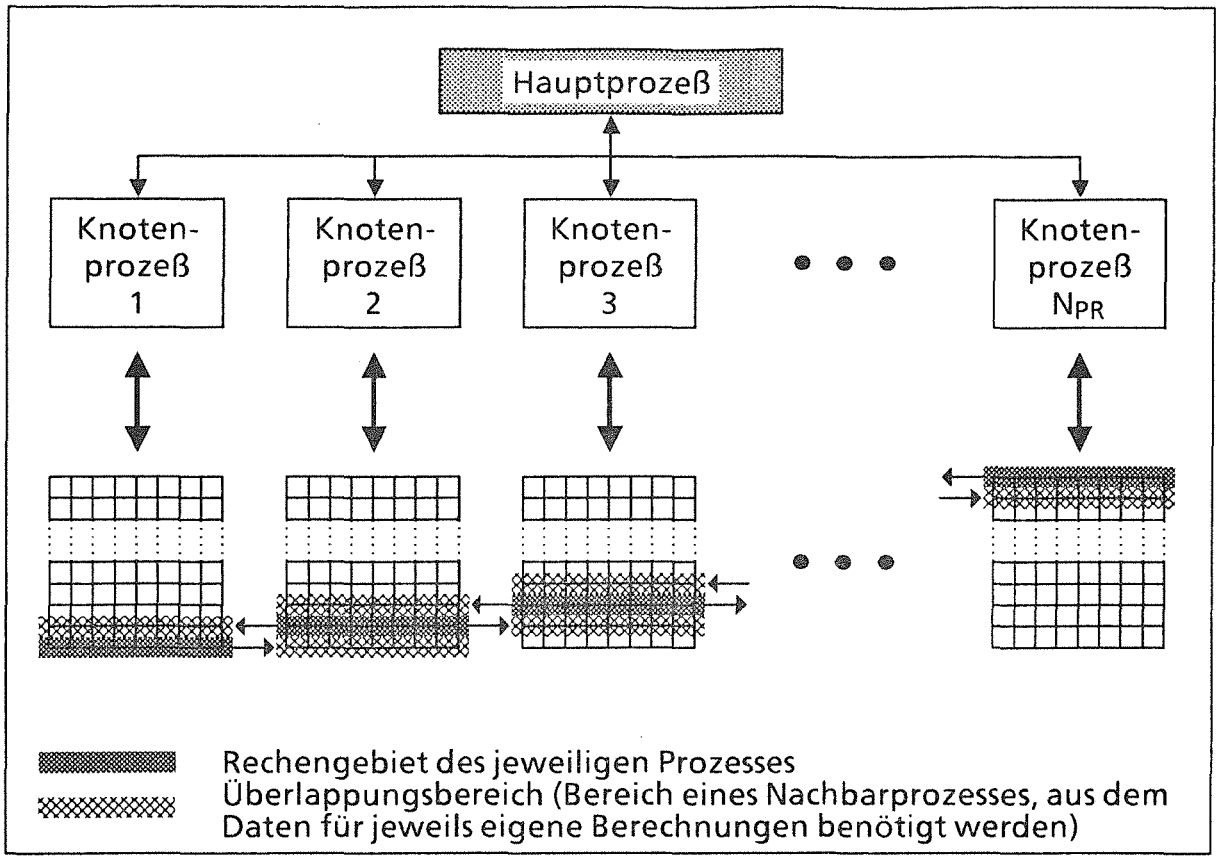


Abb. 4.2.1: Prozeß-Konzept: Aufteilung des Rechengitters auf die Knotenprozesse.

Daneben treten Module, die den Nachrichtenaustausch zwischen den Prozessen durchführen (Austausch von Zwischenergebnissen, Berechnung globaler Größen wie Fehler-Maxima, etc.). Aus Portabilitätsgründen bauen die hier beschriebenen Programme auf der für den SUPRENUM-Rechner verfügbaren Kommunikationsbibliothek [15] auf, so daß auf die Verwendung maschinenspezifischer Sprachelemente (für die es noch keinen Sprach-Standard gibt) weitestgehend verzichtet werden konnte. Dadurch ist es im Prinzip möglich, die Programme auch auf anderen Rechnern zu verwenden, auf denen diese Bibliothek ebenfalls vorhanden ist.

Ablauf einer parallelen Potential- oder Feldberechnung

Im Hauptprogramm des Hauptprozesses (oder ggf. in einem besonderen Steuerprozeß) wird eines der in Abschnitt 2.2.1 beschriebenen Programme aufgerufen, wobei die folgenden Parameterwerte anzugeben sind (s. Abschnitt 2.2.1.3):

- Prozeßerzeugung und Initialisierung:
  Für IAUFR ist entweder 0 oder 1 vorzugeben.
  Wiederholungsaufruf (Potentialberechnung):
  Für IAUFR ist 2 (oder 9) vorzugeben.
  Wiederholungsaufruf (Neuberechnung der Diskretisierungskoeffizienten):
  Für IAUFR ist -1 vorzugeben.
- Anzahl der maximal verfügbaren Knotenrechner:
  Diese ist mittels des Parameters NPRMX vorzugeben. Programmintern kann jedoch auch eine kleinere Anzahl verwendet werden, falls dies effizienter ist (s. Ausgabeparameter NPR).
- Name des Hauptprogramms der Knotenprozesse.

Mit diesen Angaben werden automatisch die Knotenprozesse gestartet und mit den erforderlichen Eingabedaten versorgt. Der Hauptprozeß wartet dann auf die Ende-Meldungen sowie ggf. die Ergebnisse der Knotenprozesse. Anschließend können weitere Potential- oder Feldberechnungen durchgeführt werden; dabei darf IAUFR nicht mehr die Werte 0 oder 1 annehmen, und NPRMX darf nicht verändert werden.

Das Hauptprogramm der Knotenprozesse beginnt mit dem Aufruf eines der in Abschnitt 2.2.2 beschriebenen Programme. Dabei muß für IAUFR unbedingt der selbe Wert angegeben werden wie im korrespondierenden Programmaufruf im Haupt- oder Steuerprozeß (Ausnahme s. Abschnitte 2.1 und 2.2). Nach Abschluß der Berechnungen senden die Programme EPOT1K und -3K (Potential-) bzw. EFLD1K und -3K (Feldberechnung) ihre Teilergebnis (Potential bzw. z- und r-Komponenten des elektrischen Feldes) nach Empfang einer entsprechenden Aufforderung an an den Hauptprozeß; EPOT2K und EFLD2K senden dagegen lediglich eine Fertig-Meldung.

Analog zu diesem Prozeßerzeugungs-Aufruf können weitere Wiederholungen erfolgen, wobei IAUFR weiterhin zwischen Haupt- und Knotenprozessen übereinstimmen muß ist.

Beispiele für verschiedene Anwendungsfälle sind in Kapitel 5 wiedergegeben.

4.2.2 Parallelisierungskonzept

Die Parallelisierung der Potential- und Feldberechnung basiert auf einer geeigneten Aufteilung des Gebiets in Teilgebiete (Streifen) mit möglichst gleichgroßem Rechenaufwand (Abb. 4.2.2.1). Jedem der parallel ablaufenden Knotenprozesse wird ein solcher Streifen vom Hauptprozeß zur Bearbeitung zugewiesen.

Die Ermittlung der Streifenbreite geschieht wie folgt: Zunächst wird die Anzahl der Gitterlinien in y-Richtung (NY) durch die Zahl der insgesamt verfügbaren Prozesse (NPRMX) dividiert und das Ergebnis auf die nächsthöhere gerade Zahl aufgerundet. Alle Streifen erhalten diese Breite, lediglich der (logisch) oberste darf weniger oder (maximal eine Linie) mehr erhalten.

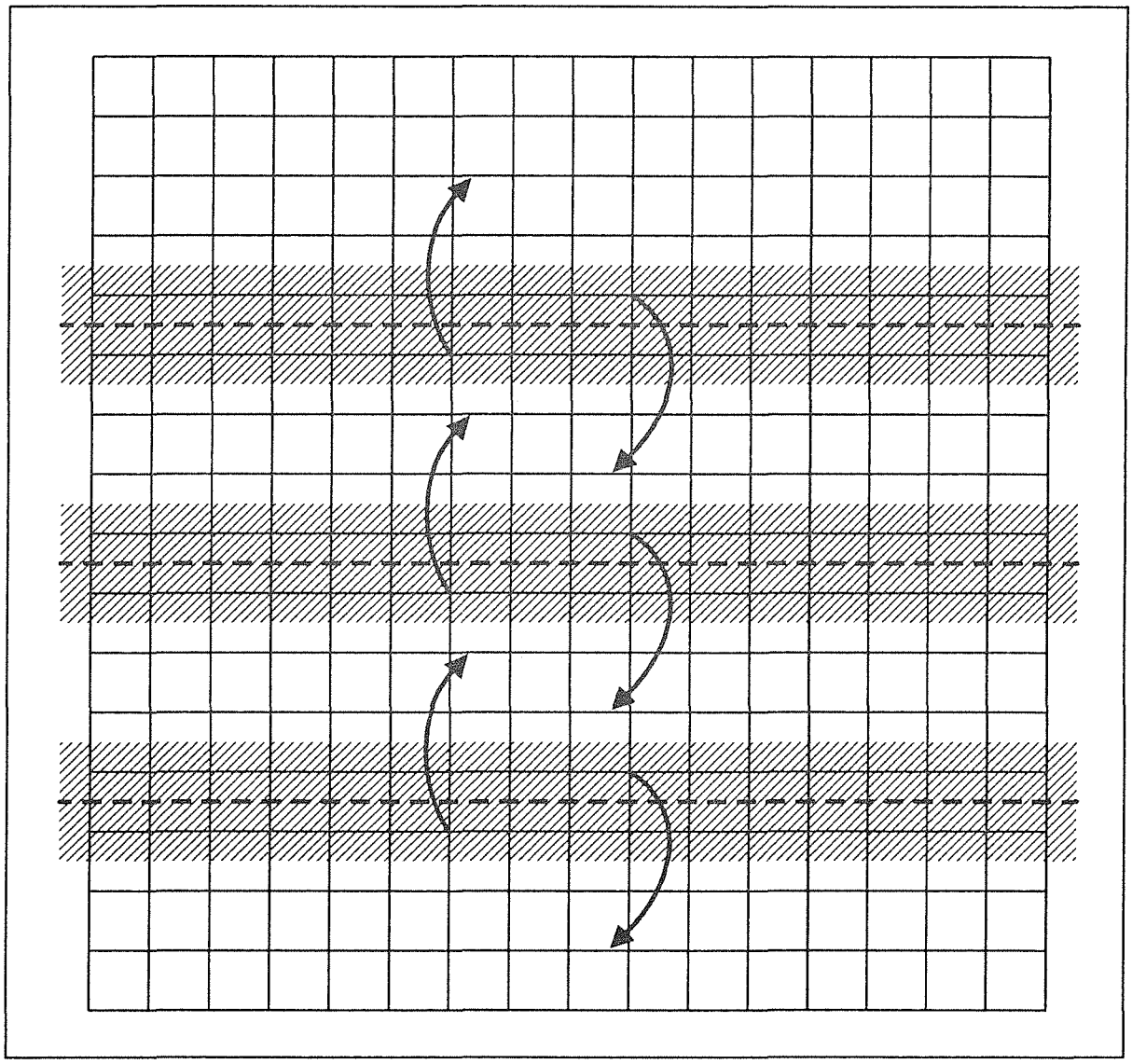


Abb. 4.2.2.1: Einteilung des Rechengebiets in „Streifen".

Die Forderung nach einer geraden Streifenbreite ist wegen des speziellen „4-Farb"-Relaxationsschemas [13] aus Effizienzgründen optimal. Es reichen damit jedoch unter Umständen weniger als NPRMX Prozesse aus, s. Abschnitt 2.2.1.3 (Ausgabeparameter NPR).

Im Verlauf seiner Berechnungen benötigt jeder Prozeß Zwischenergebnisse von sämtlichen Prozessen, die benachbarte Teilgebiete bearbeiten. Diese Daten werden per Austausch entsprechender Nachrichten nach jedem Relaxationsteilschritt sowie innerhalb der Restriktionsphase aktualisiert; bei der Interpolation ist dagegen kein Nachrichtenaustausch notwendig. Wegen der nahezu ausschließlichen Verwendung von Neun-Punkte-Operatoren genügt der Austausch von Daten im Bereich der unmittelbar den Grenzen zwischen Teilgebieten benachbarten Gitterlinien (s. Pfeile und schraffierte Gebiete in Abbildung 4.2.2.1).

Da die Kommunikation im Vergleich zur Rechnung sehr viel Zeit kosten kann, wurde bei der Programmierung darauf geachtet, daß der Kommunikationsaufwand möglichst gering ist. Deshalb wurden die folgenden Parallelisierungskonzepte entwickelt, deren Ziel die Minimierung der Anzahl der Nachrichten ist [13]:

- optimierte „Vier-Farb"-Relaxation;
- Zusammenfassung von jeweils mehreren benachbarten Teilgebieten in nur einem Prozeß während der Behandlung grober Gitter (Agglomeration).

5. Beispiele

Wie bereits erwähnt, bestehen Anwendungsprogramme, die auf SUPRENUM gerechnet werden sollen, aus je einem Hauptprogramm (mit den zugehörigen Unterprogrammen) für den Hauptprozeß (der hauptsächlich die Ein- und Ausgabe durchführt und die parallelen Prozesse koordiniert) sowie für die auf den Knotenrechnern parallel ablaufenden Knotenprozesse, die vor allem für die eigentliche Rechenarbeit zuständig sind, aber normalerweise keine Ein-/Ausgabeoperationen durchführen. Abb. 5.0.1 zeigt dies schematisch. (Abweichend können die Steuerprogramme EPOT3S, EPOT3A mit EPOT3B, EFLD3S oder EFLD3A mit EFLD3B auch in einem steuernden Knotenprozeß aufgerufen werden.)

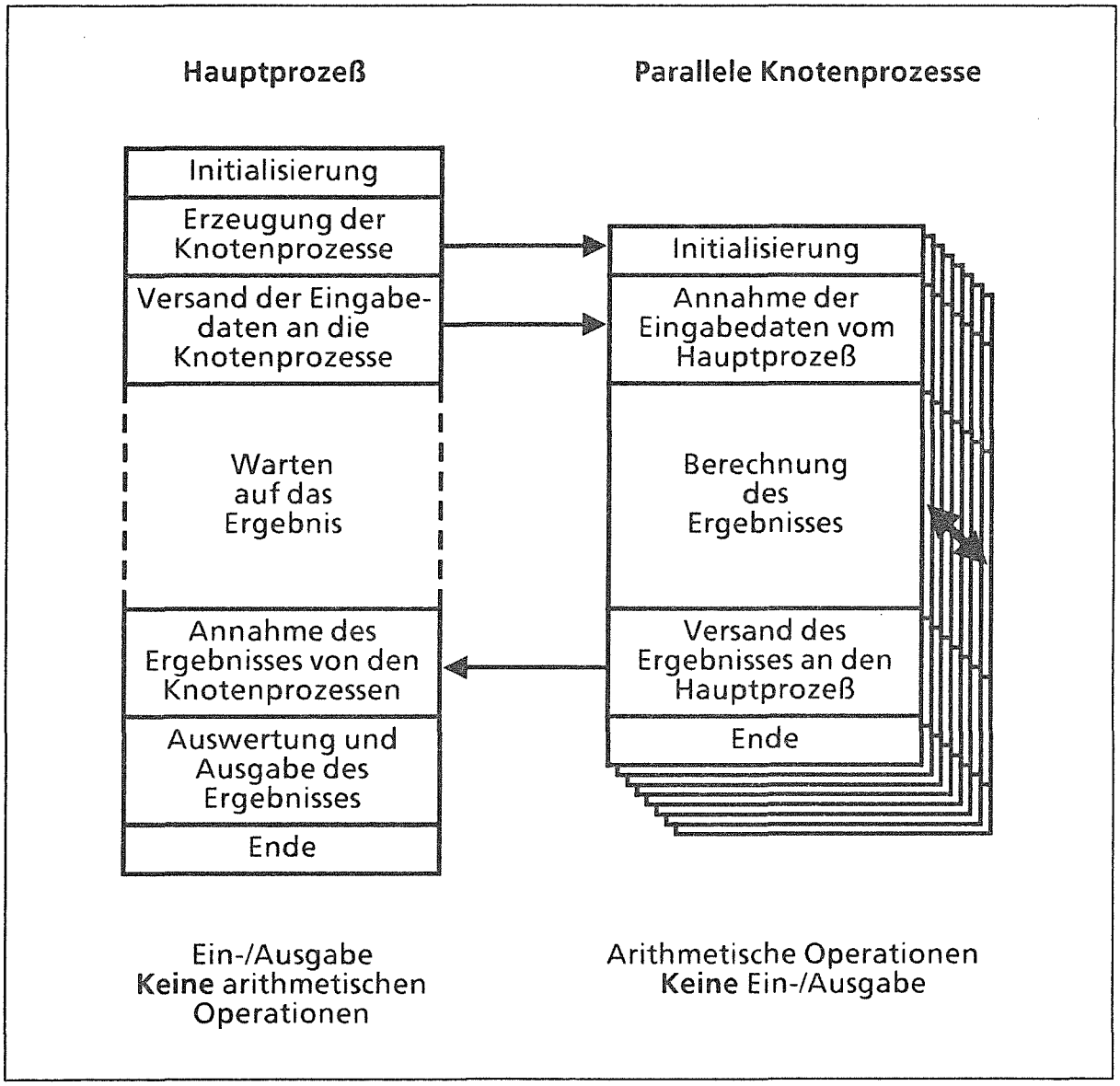


Abb. 5.0.1: Prinzip der Aufteilung in Hauptprozeß und Knotenprozesse.

Es werden die folgenden Beispiele behandelt:

1. Parallele Berechnung eines elektrostatischen Potentials zu einer vorgegebenen Raumladungsdichte, die zunächst im Hauptprozeß eingelesen wird; das berechnete Potential wird von den parallelen Knotenprozessen an den Hauptprozeß gesendet und dort in eine Datei geschrieben (Abschnitt 5.1).

   Für diese Berechnungen werden die Programme EPOT1H (Haupt-) und EPOT1K (Knotenprozesse) verwendet.

2. Im zweiten Beispiel (Abschnitt 5.2) soll das elektrostatische Feld zu einer im Hauptprozeß vorliegenden Raumladungsdichte berechnet werden. Es wird angenommen, daß die Feldstärken nur in den Knotenprozessen benötigt werden. Das berechnete Potential soll dagegen an den Hauptprozeß gesendet und dort in eine Datei geschrieben werden.

   Da nicht alle berechneten Daten im Hauptprozeß benötigt werden, ist es sinnvoll, die Programme EFLD2H (Haupt-) und EFLD2K (Knotenprozesse) zu benutzen und die Raumladungsdichte RHO und das Potential PHI mittels der Kommunikationsprogramme VERT2H / VERT2K und SAML2H / SAML2K zu versenden.

3. Hier wird angenommen, daß in einer Schleife jeweils das Potential zu einer anderen Raumladungsdichte RHO berechnet werden soll. Die Steuerung der Programme soll nicht vom Haupt-, sondern von einem besonderen Steuerprozeß aus erfolgen, der während der parallelen Potentialberechnung ebenfalls gewisse Arbeiten ausführen soll.

   Dazu wird im Steuerprozeß im Initialisierungsaufruf das Programm EPOT3S, in den späteren Schleifendurchläufen die Programme EPOT3A und EPOT3B aufgerufen. Zwischen EPOT3A und EPOT3B sind weitere Anweisungen eingefügt (hier: Aufruf eines Beispiel-Unterprogramms „ZZZZZZ").

   In den Knotenprozessen wird das Programm EPOT3K aufgerufen. Zur Beendigung dieser Schleife wird mittels des Parameters IAUFR eine entsprechende Aufforderung vom Steuerprozeß übermittelt (Abschnitt 5.3).

5.1 Beispiel 1

5.1.1 Hauptprozeß

Das hier wiedergegebene Beispielprogramm liest zunächst ein Gitter von einer externen Datei, ruft das Potentialberechnungsprogramm EPOT1H auf, wartet auf das Ergebnis und gibt dieses abschließend aus:

```
************************************************************************
*                                                                      *
*     Demonstrationsprogramm 1:                                        *
*     Hauptprogramm des Hauptprozesses:                                *
*     ================================                                 *
*                                                                      *
*     Es werden die Gitterkoordinaten mit den zugehörigen Punktattri-  *
*     buten, Startwerte für das Potential PHI, die Ladungsdichte RHO   *
*     sowie die Anzahl der verfügbaren Knotenrechner eingelesen.       *
*                                                                      *
*     Anschließend werden die Eingabedaten an die Knotenprozesse       *
*     geschickt, die das elektrostatische Potential durch Lösung der   *
*     Poisson-Gleichung berechnen und schließlich an den Hauptprozeß   *
*     senden, der das Ergebnis auswertet und ausgibt.                  *
*                                                                      *
*     Der Name des Hauptprogramms der parallelen Knotenprozesse        *
*     (hier: KP1, s. Abschnitt 5.1.2) muß als TASK EXTERNAL            *
*     deklariert und dem Programm EPOT1H mitgeteilt werden.            *
*                                                                      *
************************************************************************
*      4.12.1989                                                       *
************************************************************************


      INITIAL TASK PROGRAM HP1


      INTEGER   NDIM1, NDIM2, NPAT

      PARAMETER (NDIM1=1025, NDIM2=1025, NPAT=5)

      REAL*8    PHI(NDIM1,NDIM2), RHO(NDIM1,NDIM2),
     *          ZK (NDIM1,NDIM2), RK (NDIM1,NDIM2), PPAR(-NPAT:NPAT,2),
     *          AFLRMX, RFLRMX

      INTEGER   IFALL(-NPAT:NPAT), IPAT(NDIM1,NDIM2),
     *          NX, NY, NPRMX, NPR, NZKLMN, NZKLMX, IFLR

      TASK EXTERNAL KP1

      PARAMETER (AFLRMX=0.0, RFLRMX=0.5E-10)
      PARAMETER (NZKLMN=1, NZKLMX=10)
```

```
*  Einlesen eines Gitters aus einer externen Datei:

      READ  (1,101) NX, NY
      IF (NX .GT. NDIM1) THEN
        WRITE (6,601) NX, NDIM1
        STOP
      ENDIF
      IF (NY .GT. NDIM2) THEN
        WRITE (6,602) NY, NDIM2
        STOP
      ENDIF
      READ  (1,102)
     *      ((ZK(IX,IY), RK(IX,IY), IPAT(IX,IY), IX=1,NX), IY=1,NY)

*  Einlesen der Ladungsdichte RHO sowie einer Startnäherung für das
*  Potential PHI aus einer externen Datei:

      READ  (2,201)
      READ  (2,202) ((RHO(IX,IY), IX=1,NX), IY=1,NY)

      READ  (3,201)
      READ  (3,202) ((PHI(IX,IY), IX=1,NX), IY=1,NY)

*  Initialisierung von IFALL und PPAR:

      IFALL(-NPAT:NPAT) = 0
      PPAR(-NPAT:NPAT,1:2) = 0.0D+00

*  Zuordnung der Punktarten und -parameter zu den beispielhaften
*  Punktattributen (0 in Feldpunkten, -1 für die Anode, -2 für die
*  Kathode, -3 für die Symmetrieachse und -4 für den oberen Rand):
*
*  - Oberer Rand (bezogen auf das logische Gitter):
      IFALL(-4) = 4
*
*  - Symmetrieachse:
      IFALL(-3) = 8
*
*  - Kathode:
      IFALL(-2) = 2
*
*  - Anode:
      IFALL(-1) = 2
      PPAR(-1,1) = 1.5D+06
*
*  - Feldpunkte (im Inneren der Diode, außerhalb der Achse):
      IFALL( 0) = 1

*  Einlesen der Anzahl der zur Verfügung stehenden Prozessoren:

      READ  (5,501) NPRMX
```

```
*  Initialisierung der parallelen Prozesse, Veranlassung der Berechnung
*  der Diskretisierungskoeffizienten und parallele Potentialberechnung:

      CALL EPOT1H (PHI, RHO, ZK, RK, PPAR, IFALL, IPAT,
     *             NX, NY, NDIM1, NDIM2, NPAT, 1, NPRMX, NPR,
     *             NZKLMN, NZKLMX, AFLRMX, RFLRMX, KP1, 4, IFLR)
      IF (IFLR .GT. 0) THEN
        WRITE (6,603) IFLR
        STOP
      ENDIF

*  Ausgeben des Ergebnisses:

      WRITE (4,203) NX, NY, ((PHI(IX,IY),IX=1,NX),IY=1,NY)


      STOP

101   FORMAT (//2I5)
102   FORMAT (10X,2E15.6,I5)
201   FORMAT (//)
202   FORMAT (5E15.7)
203   FORMAT (2I5/(5E15.7))
501   FORMAT (I5)
502   FORMAT (2I5)
601   FORMAT (' +++  Fehler in den Eingabedaten:  +++'/
     *        ' Die Dimension in der ersten Koordinatenrichtung',
     *        ' des eingelesenen Gitters NX =',I5/
     *        ' ist groesser als die im Programm vorgesehene',
     *        ' Hoechstgroesse NDIM1 =',I5,'.'//)
602   FORMAT (' +++  Fehler in den Eingabedaten:  +++'/
     *        ' Die Dimension in der ersten Koordinatenrichtung',
     *        ' des eingelesenen Gitters NY =',I5/
     *        ' ist groesser als die im Programm vorgesehene',
     *        ' Hoechstgroesse NDIM2 =',I5,'.'//)
603   FORMAT (' +++  Fehler im Programm EPOT1H/EPOT1K:  +++'/
     *        ' Es ist der Fehler Nr.',I4,' aufgetreten.'//)

      END
```

5.1.2 Knotenprozesse

Nachdem die Knotenprozesse vom Hauptprozeß gestartet wurden, empfangen sie die Eingabedaten, führen ihre Berechnungen durch und senden schließlich das von ihnen berechnete Teilergebnis an den Hauptprozeß:

```
************************************************************************
*                                                                      *
*     Demonstrationsprogramm 1:                                        *
*     Hauptprogramm der Knotenprozesse:                                *
*     ================================                                 *
*                                                                      *
*     Dieses Programm ist das Gegenstück zum Hauptprogramm HP1,        *
*     (Hauptprozeß, s. Abschnitt 5.1.1).                               *
*                                                                      *
************************************************************************
*      4.12.1989                                                       *
************************************************************************

      TASK PROGRAM KP1

      INTEGER   NDIMA, NDIMI, IFLR

      PARAMETER (NDIMA=50000, NDIMI=5000)

      REAL*8    A(NDIMA)

      INTEGER   I(NDIMI)

*  Bestätigung der Prozeß-Erzeugung, Berechnung der
*  Diskretisierungskoeffizienten und des Potentials PHI:

      CALL EPOT1K (A, NDIMA, I, NDIMI, 1, IFLR)

      RETURN
      END
```

5.2 Beispiel 2

5.2.1 Hauptprozeß

Das hier wiedergegebene Beispielprogramm liest zunächst Eingabedaten ein und ruft das Programm EFLD2H auf, welches zunächst die Diskretisierungskoeffizienten sowie das Potential PHI sowie die z- und r-Komponenten des elektrischen Feldes berechnet. Im Gegensatz zum Beispielprogramm in Abschnitt 5.1 müssen die Matrizen RHO und PHI explizit versendet werden.

```
************************************************************************
*                                                                      *
*     Demonstrationsprogramm 2:                                        *
*     Hauptprogramm des Hauptprozesses:                                *
*     ================================                                 *
*                                                                      *
*     Zuerst werden die Gitterkoordinaten, die Ladungsdichte und die   *
*     Anzahl der zur Verfügung stehenden Knotenrechner eingelesen.     *
*     Anschließend werden die se Daten an die Knotenprozesse  gesen-   *
*     det, die das elektrostatische Potential durch Lösung der         *
*     Poisson-Gleichung und daraus die z- und r-Komponenten des Feldes *
*     durch numerische Differentiation berechnen. Zum Schluß wird das  *
*     Potential PHI an den Hauptprozeß gesendet und dort ausgegeben.   *
*                                                                      *
*     Der Name des Hauptprogramms der parallelen Knotenprozesse        *
*     (hier: KP2, s. Abschnitt 5.2.2) muß als TASK EXTERNAL            *
*     deklariert und dem Programm EFLD2H mitgeteilt werden.            *
*                                                                      *
************************************************************************
*      4.12.1989                                                       *
************************************************************************

      INITIAL TASK PROGRAM HP2


      INTEGER   NDIM1, NDIM2, NPAT

      PARAMETER (NDIM1=1025, NDIM2=1025, NPAT=5)

      REAL*8    PHI(NDIM1,NDIM2), RHO(NDIM1,NDIM2),
     *          ZK (NDIM1,NDIM2), RK (NDIM1,NDIM2), PPAR(-NPAT:NPAT,2),
     *          AFLRMX, RFLRMX

      INTEGER   IFALL(-NPAT:NPAT), IPAT(NDIM1,NDIM2),
     *          NX, NY, NPRMX, NPR, NZKLMN, NZKLMX, IFLR

      TASK EXTERNAL KP2

      PARAMETER (NZKLMN=1, NZKLMX=10, AFLRMX=0.0, RFLRMX=0.5D-10)
```

```
*  Einlesen eines Gitters aus einer externen Datei:

      READ  (1,101) NX, NY
      IF (NX .GT. NDIM1) THEN
        WRITE (6,601) NX, NDIM1
        STOP
      ENDIF
      IF (NY .GT. NDIM2) THEN
        WRITE (6,602) NY, NDIM2
        STOP
      ENDIF

      READ  (1,102)
     *      ((ZK(IX,IY), RK(IX,IY), IPAT(IX,IY), IX=1,NX), IY=1,NY)

*  Einlesen der Ladungsdichte aus einer externen Datei:

      READ  (2,201) ((RHO(IX,IY), IX=1,NX), IY=1,NY)

*  Initialisierung von IFALL und PPAR:

      IFALL(-NPAT:NPAT) = 0
      PPAR(-NPAT:NPAT,1:2) = 0.0D+00

*  Zuordnung der Punktarten und -parameter zu den beispielhaften
*  Punktattributen (0 in Feldpunkten, -1 für die Anode, -2 für die
*  Kathode, -3 für die Symmetrieachse und -4 für den oberen Rand):
*
*  - Oberer Rand (bezogen auf das logische Gitter):
      IFALL(-4) = 4
*
*  - Symmetrieachse:
      IFALL(-3) = 8
*
*  - Kathode:
      IFALL(-2) = 2
*
*  - Anode:
      IFALL(-1) = 2
      PPAR(-1,1) = 1.5D+06
*
*  - Feldpunkte (im Inneren der Diode, außerhalb der Achse):
      IFALL( 0) = 1

*  Anzahl der zur Verfügung stehenden Prozessoren:

      READ  (5,501) NPRMX
```

```
*  Initialisierung der parallelen Prozesse und
*  Veranlassung der Berechnung der Diskretisierungskoeffizienten:

      CALL EFLD2H (ZK, RK, PPAR, IFALL, IPAT,
     *             NX, NY, NDIM1, NDIM2, NPAT, 0, NPRMX, NPR,
     *             NZKLMN, NZKLMX, AFLRMX, RFLRMX, KP2, 1, IFLR)
      IF (IFLR .GT. 0) THEN
        WRITE (6,604) IFLR
        STOP
      ENDIF

*  Versand der Ladungsdichte an die Knotenprozesse; dies kann erst nach
*  dem Initialisierungsaufruf des Programms EFLD2H geschehen, bei dem
*  bereits die Gitterdaten versendet werden:

      CALL VERT2H (RHO, NX, NY, NDIM1, NDIM2, NPRMX, NPR, 0, 1, IFLR)
      IF (IFLR .GT. 0) THEN
        WRITE (6,603) IFLR
        STOP
      ENDIF

*  Parallele Potentialberechnung:

      CALL EFLD2H (ZK, RK, PPAR, IFALL, IPAT,
     *             NX, NY, NDIM1, NDIM2, NPAT, 2, NPRMX, NPR,
     *             NZKLMN, NZKLMX, AFLRMX, RFLRMX, KP2, 4, IFLR)
      IF (IFLR .GT. 0) THEN
        WRITE (6,604) IFLR
        STOP
      ENDIF

*  Empfangen und Ausgeben des Ergebnisses:

      CALL SAML2H (PHI, NX, NY, NDIM1, NDIM2, NPR, 1, IFLR)
      IF (IFLR .GT. 0) THEN
        WRITE (6,605) IFLR
        STOP
      ENDIF

      WRITE (3,202) NX, NY, ((PHI(IX,IY),IX=1,NX),IY=1,NY)
```

```
      STOP

101   FORMAT (//2I5)
102   FORMAT (10X,2E15.6,I5)
201   FORMAT (///(5E15.7))
202   FORMAT (//2I5/(5E15.7))
501   FORMAT (I5)
601   FORMAT (' +++  Fehler in den Eingabedaten:  +++'/
     *        ' Die Dimension in der ersten Koordinatenrichtung',
     *        ' des eingelesenen Gitters NX =',I5/
     *        ' ist groesser als die im Programm vorgesehene',
     *        ' Hoechstgroesse NDIM1 =',I5,'.'//)
602   FORMAT (' +++  Fehler in den Eingabedaten:  +++'/
     *        ' Die Dimension in der ersten Koordinatenrichtung',
     *        ' des eingelesenen Gitters NY =',I5/
     *        ' ist groesser als die im Programm vorgesehene',
     *        ' Hoechstgroesse NDIM2 =',I5,'.'//)
603   FORMAT (' +++  Fehler im Programm VERT2H/VERT2K:  +++'/
     *        ' Es ist der Fehler Nr.',I4,' aufgetreten.'//)
604   FORMAT (' +++  Fehler im Programm EFLD2H/EFLD2K:  +++'/
     *        ' Es ist der Fehler Nr.',I4,' aufgetreten.'//)
605   FORMAT (' +++  Fehler im Programm SAML2H/SAML2K:  +++'/
     *        ' Es ist der Fehler Nr.',I4,' aufgetreten.'//)

      END
```

5.2.2 Knotenprozesse

Nachdem die Knotenprozesse vom Hauptprozeß gestartet wurden, empfangen sie die Eingabedaten, führen ihre Berechnungen durch und senden schließlich das von ihnen berechnete elektrostatische Potential an den Hauptprozeß:

```
************************************************************************
*                                                                      *
*     Demonstrationsprogramm 2:                                        *
*     Hauptprogramm der Knotenprozesse:                                *
*     ================================                                 *
*                                                                      *
*     Dieses Programm ist das Gegenstück zum Hauptprogramm HP2,        *
*     (Hauptprozeß, s. Abschnitt 5.2.1).                               *
*                                                                      *
************************************************************************
*      4.12.1989                                                       *
************************************************************************

      TASK PROGRAM KP2


      INTEGER   NDIMA, NDIMI, NX, NY, IYBU, IYAU, IYAO, IYBO, IFLR,
     *          IVEZ, IVER, IVPHI, IVRHO, IVZK, IVRK,
     *          IVFALL, IVPAT, IVHOM

      PARAMETER (NDIMA=50000, NDIMI=5000)

      REAL*8    A(NDIMA)

      INTEGER   I(NDIMI)


*  Bestätigung der Prozeß-Erzeugung und Berechnung der
*  Diskretisierungskoeffizienten:

      CALL EFLD2K (A, NDIMA, I, NDIMI,
     *             IVEZ, IVER, IVPHI, IVRHO, IVZK, IVRK,
     *             IVFALL, IVPAT, IVHOM,
     *             NX, NY, IYBU, IYAU, IYAO, IYBO, 0, IFLR)
      IF (IFLR .GT. 0) GOTO 90

*  Annahme der Ladungsdichte RHO im Rechengebiet vom Hauptprozeß
*  (die Startadresse von RHO innerhalb des Hilfsvektors A, IVRHO,
*  wird im obigen Aufruf des Programms EFLD2K berechnet):

      CALL VERT2K (A(IVRHO), NX, IYAU, IYAO, IFLR)
      IF (IFLR .GT. 0) GOTO 90
```

```
*  Initialisierung des Lösungsvektors mit 0:

      A(IVPHI:IVPHI+NX*(IYBO-IYBU+1)-1) = 0.0D+00

*  Parallele Potentialberechnung:

      CALL EFLD2K (A, NDIMA, I, NDIMI,
     *             IVEZ, IVER, IVPHI, IVRHO, IVZK, IVRK,
     *             IVFALL, IVPAT, IVHOM,
     *             NX, NY, IYBU, IYAU, IYAO, IYBO, 2, IFLR)
      IF (IFLR .GT. 0) GOTO 90

*  Senden des Potentials an den Hauptprozeß zur Ausgabe auf eine Datei:

      CALL SAML2K (A(IVPHI), NX, IYBU, IYBO, IYAU, IYAO, IFLR)
      IF (IFLR .GT. 0) GOTO 90

*  Auswertung der Feldstärken in den Knotenprozessen:

      CALL FELDER (A(IVEZ), A(IVER), NX, IYAU, IYAO, ... )

   90 CONTINUE


      RETURN
      END
```

5.3 Beispiel 3

5.3.1 Hauptprozeß

Bei Anwendung der Programme EPOT3S, -3A / -3B und -3K zur Potentialberechnung (bzw. den entsprechenden Feldprogrammen EFLD3S, -3A / -3B und -3K) werden im Hauptprozeß selbst keinerlei Programme des Programmpakets zur Potentialberechnung aufgerufen. Die Steuerung der parallelen Berechnung von PHI erfolgt stattdessen von einem eigenen Steuerprozeß aus, der auf einem SUPRENUM-Knoten abläuft.

5.3.2 Steuerprozeß

Das Hauptprogramm des Steuerprozesses entspricht in seinem grundsätzlichen Aufbau dem Programm HP1 (s. Abschnitt 5.1.1). Der Unterschied besteht darin, daß in diesem Hauptprogramm weder Ein- noch Ausgabe möglich sind. Stattdessen müssen die Eingabedaten im Hauptprozeß eingelesen und an den Steuerprozeß gesendet werden. Umgekehrt sendet dieser die von ihm zuvor von den Knotenprozessen eingesammelten Ausgabedaten an den Hauptprozeß, der die abschließende Auswertung durchführt.

Dieser Steuerprozeß ist vom Benutzer zu erzeugen und mit den benötigten Eingabedaten zu versorgen. Auch für das Versenden der (Zwischen-) Ergebnisse an den Hauptprozeß ist der Benutzer zuständig. Entsprechende Module sind jedoch nicht Gegenstand dieses Programmpakets.

```
************************************************************************
*                                                                      *
*     Demonstrationsprogramm:                                          *
*     Hauptprogramm des   S t e u e r prozesses:                       *
*     ==========================================                       *
*                                                                      *
*     Es werden zunächst die Gitterkoordinaten mit den zugehörigen     *
*     Punktattributen, die Ladungsdichte RHO sowie die Anzahl der      *
*     zur Verfügung stehenden Knotenrechner vom Hauptprozeß empfangen  *
*     und an die Knotenprozesse gesendet, die das elektrostatische     *
*     Potential berechnen und schließlich an den Steuerprozeß senden.  *
*     Dort wird das Ergebnis auswertet und ggf. an den Hauptprozeß     *
*     weitergesendet.                                                  *
*                                                                      *
*     Im Gegensatz zum Hauptprozeß (vgl. Beispiele 1 und 2 in Ab-      *
*     schnitten 5.1 und 5.2) kann der Steuerprozeß ebenfalls gewisse   *
*     Berechnungen etc. durchführen. In diesem Beispiel soll während   *
*     der parallelen Potentialberechnungen im Steuerprozeß jeweils     *
*     das Programm ZZZZZZ aufgerufen werden. Zu diesem Zweck wird       *
*     innerhalb der Berechnungsschleife nicht das Programm EPOT3S,     *
*     sondern die beiden Teile EPOT3A (Übersendung der Eingabedaten)   *
*     und EPOT3B (Empfangen des Potentials).                           *
*                                                                      *
*     Der Name des Hauptprogramms der parallelen Knotenprozesse        *
*     (hier: KP3, s. Abschnitt 5.3.2) muß als TASK EXTERNAL dekla-     *
*     riert und den Programmen EPOT3S bzw. EPOT3A mitgeteilt werden.   *
*                                                                      *
************************************************************************
*      4.12.1989                                                       *
************************************************************************


      INITIAL TASK PROGRAM HP3


      INTEGER   NDIM1, NDIM2, NPAT

      PARAMETER (NDIM1=1025, NDIM2=1025, NPAT=5)

      REAL*8    PHI(NDIM1,NDIM2), RHO(NDIM1,NDIM2),
     *          ZK (NDIM1,NDIM2), RK (NDIM1,NDIM2), PPAR(-NPAT:NPAT,2),
     *          AFLRMX, RFLRMX

      INTEGER   IFALL(-NPAT:NPAT), IPAT(NDIM1,NDIM2),
     *          NX, NY, IAUFR, NPRMX, NPR, NZKLMN, NZKLMX, IFLR

      TASK EXTERNAL KP3

      PARAMETER (NZKLMN=1, NZKLMX=10, AFLRMX=0.0, RFLRMX=0.5E-10)


*  Empfangen der Eingabedaten vom Hauptprozeß:

      CALL EINGAB (RHO, ZK, RK, IPAT, IFALL, PPAR,
     *             NX, NY, NDIM1, NDIM2, NPRMX, IFLR)
      IF (IFLR .GT. 0) RETURN
```

```
*  Parallele Potentialberechnung mit dem ersten Datensatz für die
*  Ladungsdichte RHO:

      IAUFR = 1

      CALL EPOT3S (PHI, RHO, ZK, RK, PPAR, IFALL, IPAT,
     *             NX, NY, NDIM1, NDIM2, NPAT, IAUFR, NPRMX, NPR,
     *             NZKLMN, NZKLMX, AFLRMX, RFLRMX, KP3, 1, IFLR)
      IF (IFLR .GT. 0) RETURN

*  Auswertung des Ergebnisses, Berechnung einer neuen Ladungsdichte RHO:

      CALL XXXXXX (PHI, ... )
      CALL YYYYYY (RHO, ... )

*  Parallele Potentialberechnung für neun weitere Ladungsdichten RHO;
*  im letzten Aufruf wird IAUFR = 9 gesetzt und damit die Knotenprozesse,
*  in denen der korrespondierende Programmaufruf EPOT3K in einer Endlos-
*  schleife erfolgt:

      IAUFR = 2

      DO 10 JAUFR = 1, 10

        IF (JAUFR .EQ. 10) IAUFR = 9

        CALL EPOT3A (RHO, ZK, RK, PPAR, IFALL, IPAT,
     *               NX, NY, NDIM1, NDIM2, NPAT, IAUFR, NPRMX, NPR,
     *               NZKLMN, NZKLMX, AFLRMX, RFLRMX, KP3, 1, IFLR)
        IF (IFLR .GT. 0) RETURN

        CALL ZZZZZZ ( ... )

        CALL EPOT3B (PHI, NX, NY, NDIM1, NDIM2, IAUFR, NPRMX, NPR,
     *               1, IFLR)
        IF (IFLR .GT. 0) RETURN

   10 CONTINUE

*  Auswertung des Ergebnisses:

      CALL XXXXXX (PHI, ... )

*  Versenden des Ergebnisses an den Hauptprozeß zur abschließenden
*  Auswertung bzw. Ausgabe:

      CALL AUSGAB (PHI, NX, NY, NDIM1, NDIM2, NPRMX, IFLR)


      STOP
      END
```

5.3.3 Knotenprozesse

Das Hauptprogramm der Knotenprozesse, welche die parallele Berechnung des Potentials PHI durchführen, entspricht dem Programm KP1, s. Abschnitt 5.1.2. Abweichend davon wird jetzt das Potential PHI mehrfach berechnet. Dazu wird EPOT3K in einer Endlosschleife aufgerufen. Die Berechnungen werden beendet, falls IAUFR = 9 zurückgegeben wird (vgl. Steuerprozeß, Abschnitt 5.3.2).

```
************************************************************************
*                                                                      *
*     Demonstrationsprogramm:                                          *
*     Hauptprogramm der Knotenprozesse:                                *
*     ================================                                 *
*                                                                      *
*     Dieses Programm ist das Gegenstück zum Hauptprogramm HP3,        *
*     (Steuerprozeß, s. Abschnitt 5.3.2).                              *
*                                                                      *
************************************************************************
*      4.12.1989                                                       *
************************************************************************


      TASK PROGRAM KP3


      INTEGER   NDIMA, NDIMI, IAUFR, IFLR
      PARAMETER (NDIMA=50000, NDIMI=5000)

      REAL*8    A(NDIMA)

      INTEGER   I(NDIMI)


*  Berechnung dees Potentials PHI mit dem ersten Datensatz für die
*  Raumladungsdichte RHO:

      IAUFR = 0

      CALL EPOT3K (A, NDIMA, I, NDIMI, IAUFR, IFLR)
      IF (IFLR .GT. 0) RETURN

*  Berechnung des Potentials PHI für weitere Ladungsdichten RHO:

      IAUFR = 2

   10 CALL EPOT3K (A, NDIMA, I, NDIMI, IAUFR, IFLR)
      IF (IAUFR .LT. 9 .AND. IFLR .LE. 0) GOTO 10


      RETURN
      END
```

6. Literatur

[1] T. Westermann:
A Particle-in-Cell Method as a Tool for Diode Simulations.
Nucl. Instr. Meth. A263 (1988), S. 271-279

[2] T. Westermann:
Numerische Simulationen von technisch relevanten Ionen-Dioden mit der Particle-in-Cell Methode.
Kernforschungszentrum Karlsruhe, KfK 4510, Januar 1989

[3] J.F. Thompson, Z.U.A. Warsi, C.W. Mastin:
Boundary-Fitted Coordinate Systems for Numerical Solution of Partial Differential Equations - A Review.
J. Comp. Phys. 47, 1982, S. 1-108

[4] S. Ohring:
Application of the Multigrid Method to Poisson's Equation in Boundary-Fitted Coordinates.
J. Comp. Phys. 50, S. 307-315

[5] M. Alef, D. Seldner, T. Westermann:
Numerische Algorithmen für elektrodynamische Modelle und ihre Implementierung auf Supercomputern.
Informatik-Fachberichte 150 (J. Halin, Hrsg.), Springer 1987, S. 298-305

[6] D. Seldner, M. Alef, T. Westermann, E. Halter:
Parallel Particle Simulation in High Voltage Diodes
(Algorithms and Concepts for Implementation on SUPRENUM).
In [18]

[7] E. Halter:
Die Berechnung elektrostatischer Felder in Pulsleistungsanlagen.
Kernforschungszentrum Karlsruhe, KfK 4072, April 1986

[8] A. Brandt:
Multi-Level Adaptive Solutions to Boundary-Value Problems.
Mathematics of Computation Vol. 31 Nr. 138, 1977, S. 333-390

[9] U. Trottenberg:
Schnelle Lösung elliptischer Differentialgleichungen nach dem Mehrgitterprinzip.
Arbeitspapiere der GMD 40, St. Augustin, Juli 1983

[10] W. Hackbusch, U. Trottenberg (Hrsg.):
Multigrid Methods.
Lecture Notes in Math. 960, Springer-Verlag, 1982, S. 614-630

[11] W. Hackbusch:
Multi-Grid Methods and Applications.
Springer Series in Computational Mathematics Vol. 4, Springer-Verlag, 1985

[12] M. Alef:
Effiziente Berechnung elektrostatischer Potentiale mit Mehrgittermethoden in technischen Geometrien.
Kernforschungszentrum Karlsruhe, KfK 4613, Oktober 1989

[13] M. Alef:
Concepts for Efficient Implementation of Multigrid Methods
on SUPRENUM-like Architectures.
Kernforschungszentrum Karlsruhe, KfK 4614, Oktober 1989

[14] K. Solchenbach, C.-A. Thole, U. Trottenberg:
Parallel Multigrid Methods: Implementation on SUPRENUM-Like
Architectures and Applications.
SUPRENUM Report Nr. 4, SUPRENUM GmbH, Bonn, August 1987

[15] R. Hempel, A. Schüller:
Vereinheitlichung und Portabilität paralleler Anwendersoftware durch
Verwendung einer Kommunikationsbibliothek.
Arbeitspapiere der GMD 234, St. Augustin, November 1986

[16] A.M. Winslow:
Numerical Solution of the Quasilinear Poisson Equation in a Nonuniform
Triangle Mesh.
J. Comp. Phys. 2, 1967, S. 149-172

[17] M. Alef, T. Westermann:
Unveröffentlichter Bericht, KfK, Juni 1989

[18] U. Trottenberg (Hrsg.):
2nd International SUPRENUM Colloquium
"Supercomputing Based on Parallel Computer Architectures".
Parallel Computing 7, 1988

[19] W.K. Giloi:
SUPRENUM: A trendsetter in modern supercomputer development.
In [18]

[20] E. Halter:
Unveröffentlichter Bericht, KfK, Juli 1985

Danksagung

An dieser Stelle möchte ich mich herzlich bei Prof. Dr. Eberhard Halter für die geleisteten Vorarbeiten sowie die zahlreichen Diskussionen über die Weiterentwicklung bedanken.

Mein Dank gilt ebenso den Mitarbeitern im SUPRENUM-Projekt für ihre freundliche Unterstützung, sei es durch Diskussionen oder die Hilfe bei Implementierung und Tests der Programme auf dem SUPRENUM-Prototyp, sowie meinen Kollegen David Seldner und Dr. Thomas Westermann für Anregungen und Verbesserungsvorschläge.