Forschungszentrum Karlsruhe

Technik und Umwelt

Wissenschaftliche Berichte

FZKA 6274

PARTICLE-IN-CELL-SIMULATION MIT AutoCAD, InGrid, BFCPIC UND ANSYS

Walter Bauer

Institut für Neutronenphysik und Reaktortechnik

Forschungszentrum Karlsruhe GmbH, Karlsruhe
1999

Zusammenfassung

Stationäre Particle-in-Cell Simulationen mit dem am damaligen Kernforschungszentrum Karlsruhe entwickelten Programmsystem BFCPIC erfordern ein in einer Vorbereitungsphase erzeugtes randangepaßtes Gitter und in vielen Fällen ein extern angelegtes Magnetfeld. In der vorliegenden Arbeit werden die Schritte im Einzelnen beschrieben, die zur Gittererzeugung mit Hilfe des Programmsystems AutoCAD und InGrid nötig sind. Die Magnetfeldberechnung mit dem kommerziellen Programm ANSYS wird detailliert erläutert. Schließlich werden einige noch in der Diskussion befindliche Probleme dargestellt. Im Anhang findet sich eine Liste der gebräuchlichsten UNIX-Befehle, so weit sie für die beschriebene Aufgabe benötigt werden.

PARTICLE-IN-CELL-SIMULATIONS USING AutoCAD, InGrid, BFCPIC and ANSYS

Abstract

Stationary particle-in-cell simulations using the code system BFCPIC that was developed at the then so called Nuclear Research Center Karlsruhe need a boundary fitted mesh and, very often an externally applied magnetic field. This report describes the detailed steps necessary to generate the grid using AutoCAD and InGrid as well as the computation of the magnetic field using the commercially available code ANSYS. Finally some problems are outlined that are still under discussion. UNIX - commands necessary to operate the programs described are collected in an addendum.

Inhaltsverzeichnis

I. Einleitung: Rechner- und Dateienstruktur

Seit1988 wurden im Institut für Neutronenphysik und Reaktortechnik Simulationsrechnungen mit dem zeitlich stationären Particle-in-Cell-Code „**BFCPIC**“ [1] durchgeführt. Die Ergebnisse dieser Rechnungen sind in [2] - [11] dokumentiert. Selbst wenn in Zukunft der Schwerpunkt der Simulationsrechnungen bei der Anwendung des zeitabhängigen PIC-Codes **KADI2D** [12] liegen sollte, wird man einen Teil der dabei gemachten Erfahrungen weiter benötigen, z. B. die Tricks, die bei der Erzeugung des Gitters eine Rolle spielen. Es ist auch zu vermuten, daß es weiterhin Aufgabenstellungen geben kann, auf die der stationäre Code eine vielleicht vorläufige, aber schnellere Antwort finden wird. Die Berechnung des Magnetfeldes wird bei fremdmagnetisch isolierten Dioden immer nötig sein, so daß die Arbeit mit **ANSYS** [13] auf jeden Fall weitergeht, so lange man keine einfachere Methode zur Berechnung der Magnetfelder gefunden hat.

Im folgenden wird versucht die Schritte, die zu einer Simulation führen so zu beschreiben, daß ein Anfänger sie nachvollziehen kann. Wenn auch wesentliche Teile des Programmsystems heute mit einer komfortablen Benutzeroberfläche zu bedienen sind, ist es unerläßlich, daß der Benutzer dieser Programme Grundkenntnisse in UNIX hat. Im Anhang ist eine Liste der wichtigsten UNIX-Befehle angefügt. Im Text sollen folgende Konventionen eingehalten werden:

- **Programmnamen** sind **fett** gedruckt
- *Befehle*, die man eintippen oder anklicken muß, sind *kursiv* gedruckt
- `Dateien` und `Verzeichnisse` sind durch die Schrift „`Courier New`“ kenntlich gemacht

Ein Grundkurs zur Bedienung von **AutoCAD** [14] und zwei Kurse zu **ANSYS** (Einführung und EMAG) sind zu empfehlen. Sie sparen Monate des Herumprobierens.

Es ist unvermeidbar, daß sich alle Beschreibungen auf die Rechner des Instituts für Neutronenphysik und Reaktortechnik des Forschungszentrums Karlsruhe im Jahre 1999 beziehen. Zunächst ist es also wichtig, zu wissen, auf welchem Rechner die verschiedenen Programme liegen, damit man sich dort Zugangsberechtigungen (accounts) besorgen kann. Die Systemadministratoren der verschiedenen Rechner helfen bei diesen allgemeinen Organisationsfragen.

AutoCAD und **InGrid** [15], die Programme zur Erzeugung der Gitter für den PIC-Code, werden von der inrrisc8 aus gestartet. Es ist zweckmäßig, ein Verzeichnis `InGrid` anzulegen, in dem die eigentliche Gittererzeugung stattfindet, und in dem man danach die Dateien findet, die das Gitter beschreiben. Auch die Dateien `acad.cfg` und `acad.ini`, die für die Funktion von **AutoCAD** wichtig sind, befinden sich hier. Daneben sollte es ein Verzeichnis, z, B. `bfcpic` geben, in dem jeder Diodentyp ein eigenes Unterverzeichnis hat. **ANSYS** wird aus diesem Verzeichnis auf dem Rechner der Universität Karlsruhe aufgerufen. Der PIC-Code wird von der inrrisc6 aus aufgerufen. [Anmerkung: Das war samt allen Dateien bis vor kurzem die inrrisc5. Man muß immer damit rechnen, daß sich ähnliche Umorganisationen auch in Zukunft wiederholen, so daß man darauf gefaßt sein muß, daß jedesmal, wenn man etwas versucht, was vor vier Wochen noch vorhanden war und gegangen ist, wieder neu gesucht und geübt werden muß]. Auf der inrrisc8 wird das Auswerte-

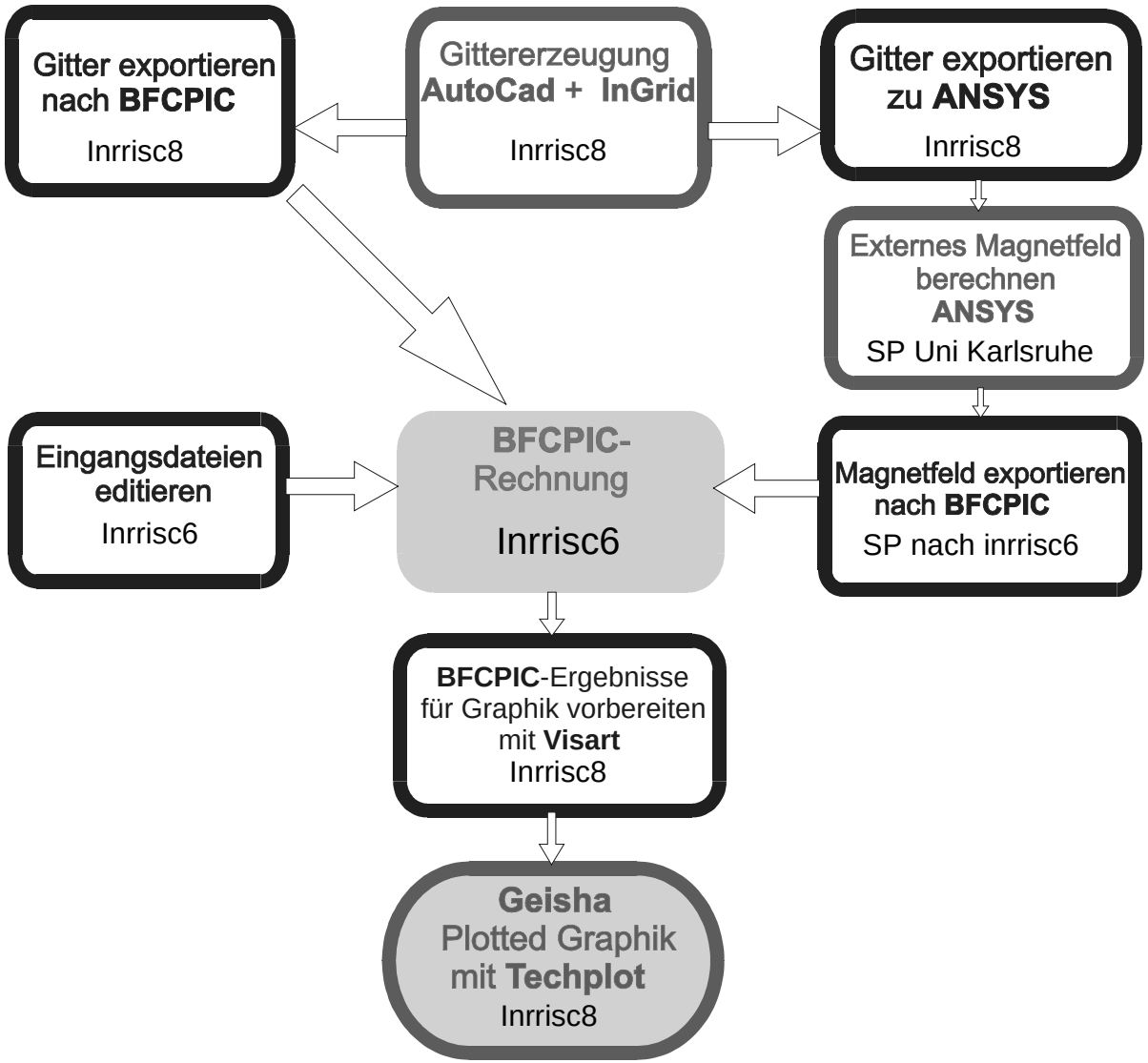

Fig. 1: Programm- und Rechnersystem für PIC-Simulationsrechnungen

programm „**geisha**“ aufgerufen, das nach der Gittererzeugung und vor allem nach der Simulation gute Dienste tut. Die jeweils nötigen Dateien werden mit **ftp** von einem Rechner zum anderen transferiert. Auf dem Rechner der Universität hat man automatisch dieselben Verzeichnisse und Dateien wie auf der inrrisc8. Auch auf der inrrisc6 sollte je ein Verzeichnis für jeden Diodentyp angelegt werden, (das natürlich denselben Namen hat wie auf der inrrisc8). Das ganze Programmsystem ist in Fig. 1 dargestellt. Man kann sich z. B. angewöhnen, den Namen einer Diode z. B. so zu bezeichnen:

`bap98a1`

Dabei bedeuten die ersten Buchstaben, daß es sich um eine B_{app} - Diode handelt (eine Diode mit extern angelegtem Magnetfeld), 98 ist die Jahreszahl der Gittererzeugung, a1 sind zwei Angaben, mit denen man noch verschiedene Untertypen unterscheiden kann. So gibt es viele Dioden mit dem Namen `bth96xy` (B_Θ - Dioden verschiedenen Typs), oder `eldio` (Elektronendioden) usw. Es hat sich gezeigt, daß man bei der Erfindung dieser Namen sehr systematisch vorgehen muß, weil man nach kurzer Zeit so viele Dateien hat, daß man sich nur noch schwer hindurch findet.

II. Das Erfassen des Modells.

1. Festlegung des Rechengebietes und der Randbedingungen.

Das zu simulierende Objekt kann in Form einer Skizze oder einer detaillierten **AutoCAD**-Zeichnung vorliegen. In jedem Fall muß man sich zuerst darüber klar werden, was der Umfang des Gebietes sein soll, in dem gerechnet werden soll. Die Regel heißt hier, daß das Rechengebiet alle Teile des Objekts umfassen soll, in dem etwas für die Funktion Signifikantes passiert: Wo sich elektrische und magnetische Felder ändern, wo Teilchen zu erwarten sind, das muß innerhalb des Rechengebietes liegen. Wo hingegen Felder gegen Null gehen oder wo man sich für das weitere Schicksal der Teilchen nicht mehr interessiert, kann das Rechengebiet abgebrochen werden. Symmetrieüberlegungen müssen hier angestellt werden. Wenn ein Gebiet in kleinere Teilgebiete zerlegt werden kann, in denen jeweils dasselbe passiert, genügt es oft, nur das Teilgebiet als Rechengebiet zu verwenden. Im allgemeinen braucht man für den PIC-Code und für **ANSYS** verschiedene Rechengebiete, weil ersterer sich für die Vorgänge im Anoden-Kathoden-Spalt interessiert, während die Spulen und sonstigen Strukturen, die das Magnetfeld erzeugen und beeinflussen, außerhalb dieser Region liegen. Das PIC-Gitter umfaßt also ein kleineres Gebiet innerhalb des Berechnungsgebiets von **ANSYS**. Hier kommt nun eine Besonderheit der beiden Programme zum Tragen, die viel Kopfzerbrechen gemacht hat:

ANSYS arbeitet mit einem unstrukturierten Gitter; das heißt, die Gitterzellen liegen wild durcheinander und haben sehr unregelmäßige Formen. **BFCPIC** (und auch **KADI2D**) benutzen ein strukturiertes Gitter, bei dem regelmäßig geformte Gitterzellen in schön durchnummerierter Reihenfolge auftreten. Der Übergang zwischen diesen Programmen wurde so gelöst, daß zuerst ein PIC-Gitter erzeugt und dieses in **ANSYS** eingelesen wurde. Dabei muß es gedreht und gespiegelt werden, weil **ANSYS** nur eine Rotationsachse in y-Richtung versteht. Dann wird das **ANSYS**-Modell außen

herum gebaut. Das Ergebnis der **ANSYS**-Rechnung wird schließlich in den PIC-Code als Eingabegröße eingelesen.

Nach der Festlegung des Rechengebietes muß man sich darüber klar werden, welche Teile des so erhaltenen Umfanges zur Anode und welche zur Kathode gehören. Diese Teile erhalten später als Randbedingung die Dirichlet-Randbedingung mit einer Angabe über das auf dieser Fläche herrschende Potential (Äquipotentiallinien verlaufen vorwiegend parallel zu solchen Flächen). Es kann auch Anoden- oder Kathodengebiete geben, die entweder als getrennte Flächen irgendwo im Rechengebiet „schweben", oder um ungünstige Ecken auszufüllen, an einem Rand außen eingefügt sind. Stellen, an denen das Modell offen gegen die Außenwelt ist, erhalten die Neumann-Randbedingung (Äquipotentiallinien verlaufen senkrecht zu diesen Flächen).

Über diese Definitionen muß man sich von vornherein klar sein, weil ein wesentlicher Bestandteil der Gittererzeugung die Festlegung der sogenannten „Punkt-Attribute" und die Definition der „Elektrodenzellen" ist. Dies wird später noch genauer erklärt. Hier nur so viel: Jeder Gitterpunkt hat nicht nur eine R- und eine Z-Koordinate, sondern auch noch ein Attribut. An dem Attribut erkennt das Programm, ob es sich um einen Anoden-, Kathoden- oder inneren Punkt des Rechengebietes handelt. Anode und Kathode haben an ihrem Rand „Elektrodenzellen", die von verschiedenem Typ sein können. An ihnen erkennt das Programm z. B., ob an diesen Flächen Teilchen entstehen können. Diese beiden Definitionen bringt man leicht durcheinander.

> Attribute sind jedem Gitter***punkt*** zugeordnet, Elektrodenzellen sind Gitter***zellen*** am Rand der Elektroden.

2. Festlegung des Umrisses.

Wer einige Übung mit **AutoCAD** hat, kann aus einer **AutoCAD**-Zeichnung den Umriß eines Rechengebietes samt den Elektroden herausziehen. Man definiert einen neuen Layer und zeichnet die für das Rechenmodell signifikanten Linien wie auf einem Transparentpapier ab. Diese Methode ist nur zu empfehlen, wenn man **AutoCAD** schon ziemlich gut beherrscht.

Einfacher für den Anfänger ist es, mit der Erstellung einer Liste zu beginnen, in der die Koordinaten der einzelnen Punkte des Rechengebietes aufgelistet sind. Notfalls kann man sich diese Zahlen aus der **AutoCAD**-Zeichnung mit Hilfe der Funktion „*Assist - Inquiry - locate point*" holen. Das sind schlimmstenfalls einige -zig Zahlenpaare, die schnell eingetippt sind. Bögen (*arcs*) macht **AutoCAD** sehr schön automatisch, wenn man sich mit einigen der zahlreichen Methoden vertraut gemacht hat, wie sie eingegeben werden. [Wenn nach einiger Zeit die Bögen eckig geworden sind, hilft der Befehl „*regen*"]

Die anfängliche Prozedur ist in einer sehr ausführlichen Beschreibung von E. Stein zusammengestellt [16], die bei jedem neuen Modell wieder sehr nützlich ist.

Es ist sehr wichtig, daß die obere, untere, rechte und linke Berandung des Rechengebietes 4 Polylinien werden, die nicht unterbrochen sein dürfen und die in den Ekken miteinander verbunden sind. Wenn man versehentlich in einer Ecke zwei Linien hat, die sich nicht treffen, kann man große Wunder erleben. Ob eine Linie, die man

gezeichnet hat, eine durchgehende Polylinie ist, kann man durch Anklicken feststellen: Soweit sie blau wird, ist sie durchgehend. Ist das nicht der Fall, muß man sie mit Hilfe des Menüs „*Modify - Edit - Polyline - join*“ verbinden.

Den fertigen Rand sollte man speichern und sich ausdrucken lassen. Das geht bisher noch sehr holprig. Man muß die **AutoCAD**-Zeichnung als postscript-Datei exportieren (wobei man einige Fragen gestellt bekommt, die man einigermaßen sinngemäß beantworten kann) und diese dann mit „*sprint*“ drucken. Man erhält dann ein winziges Bildchen, das man auf dem Kopierer beliebig vergrößern kann. In ein solches Bild zeichnet man sich mit der Hand etwa ein, wie man sich das zu erstellende Gitter vorstellt. Von dieser „künstlerischen“ Zeichnung hängt nämlich ab, wie man die Gittererzeugung anfängt.

3. Erzeugung des Gitters.

Den Umriß, der z. B. in einer Datei `bap98a0.dwg` gespeichert ist, speichert man nun mit „*safe as*“ unter dem Namen `bap98a1.dwg` ab. [Es sollte natürlich ein Name sein, den es noch nicht gibt!] Unter den tools von **AutoCAD** findet man eines namens „**InGrid**“. Das wird nun für die nächsten Tage oder Wochen das wichtigste Hilfsmittel sein. Es ist in [15] recht gut beschrieben, hat aber viele Tücken und Fallstricke, die man nur mit viel Erfahrung heil überstehen kann.

Unter „Gitter erzeugen“ wird man zuerst nach zwei Ecken gefragt. Man klickt ein Rechteck an, das etwas größer als der gezeigte Umriß ist. Je nach der Kompliziertheit des Umrisses wählt man dann unter „Gitter erzeugen“ auf die Frage, wie viele Gitterpunkte man in x und y-Richtung haben will eine Minimalzahl von Gitterpunkten. Sind es zu wenige, kann man den Umriß nicht richtig erfassen: Die Regel heißt, daß man zunächst mindestens auf jede wichtige Ecke einen Gitterpunkt schieben muß. Sind es aber zu viele, wird die Geschichte von Anfang an zu unüberschaubar und auch zu langsam. Je später man zu großen Gitterdimensionen kommt, desto besser! Man sollte zuerst mit möglichst wenigen Gitterpunkten ein einigermaßen überzeugendes Gitter erzielen. Das heißt: in wichtigen Gebieten kleinere Gitterzellen, die möglichst rechteckig und nicht schräg sein sollten und, wo sich Randgebiete und damit auch Felder stark ändern, engere Maschen. Eine Regel, die in [15] erwähnt wird:

„Ästhetisch schöne Gitter geben bessere Rechenergebnisse“

kann man gar nicht ernst genug nehmen. Nach der Angabe, wie viele Gitterpunkte in x- und y-Richtung man zu Beginn wünscht, erhält man ein Rechteck-Gitter. Jeder Gitterpunkt besteht aus dem eigentlichen Punkt, einem (roten) Index für die x-Richtung, einem (grünen) Index für die y-Richtung und einem Attribut, das im Moment entweder K (für den Rand) oder 0 (für die Innenpunkte) heißt.

Dann muß man die Eckpunkte auf die Ecken des Modells schieben: Anklicken einer der Zahlen (dadurch wird der ganze Gitterpunkt blau) und nochmal Anklicken des Zentrums. Wenn das Zentrum sich in einen roten Punkt verwandelt hat, kann man den ganzen Gitterpunkt bei gedrückter linker Taste mit der Maus verschieben. (Das ist sehr wichtig: Macht man hier etwas falsch, hat man womöglich nur den Punkt verschoben und die weiteren Angaben sind noch dort, wo sie vorher waren!!) Man schiebt den Punkt aber nicht ungefähr dahin, wohin man ihn haben will, sondern man

klickt jetzt mit der rechten Maustaste (dabei muß man vorübergehend die linke Taste loslassen): Das erzeugt ein Menü, in dem man z. B. „*endpoint*“ oder „*intersection*“ oder „*midpoint*“ anklickt. Dann verschwindet das Menü wieder und man kann nun mit der linken Maustaste in der Nähe des Punktes klicken, wo man den Gitterpunkt hinschieben möchte: Also an den Endpunkt einer Linie, oder an den Schnittpunkt zweier Linien oder auch in die Mitte einer Linie. Das geht ganz schnell, wenn man etwas Übung hat. Später wird auch öfters vorkommen, daß man Punkte einfach irgendwohin auf eine Polylinie schieben will. Man setzt ihn dann auf die Linie so gut es geht, und klickt mit der linken Maustaste. Dann sitzt der Punkt in der Nähe der Linie, aber nicht exakt auf ihr! Für diese Fälle gibt es in **InGrid** den Menüpunkt „*nahe Punkte pinnen*“. Man geht schrittweise vor, wie das Menü es angibt, und muß sehr aufpassen, daß nichts unerwünschtes passiert. Vorher auf jeden Fall „*save*“!

Jetzt sind zunächst einmal die Eckpunkte dort, wo sie sein sollen. Die Punkte dazwischen bringt man unter **InGrid** mit „*äquidistantes Füllen*“ auf die Linie, die zwischen diesen Eckpunkten liegt. Wenn das auf allen vier Teilen des Randes erfolgreich war, kann *man „Gitterlinien zeichnen*“ verlangen. Das Ergebnis sieht seltsam aus, weil ja bisher nur der Rand einigermaßen richtig mit Gitterpunkten belegt ist. Es ist jetzt nötig „*Gitter glätten*“ zu verlangen. Als Anzahl der Glättungen gibt man 5 oder 10 ein. Es zeigt sich jetzt die Bedeutung des Attributes K oder 0: Die Punkte mit dem Attribut K bleiben, wo sie sind, während die mit dem Attribut 0 verschoben werden. [Allgemein gilt: nur Punkte mit dem Attribut 0 werden verschoben, alle mit einem Attribut $\neq 0$ bleiben fest]. Nochmaliges Zeichnen der Gitterlinien kann schon ein kleines Erfolgserlebnis sein. Wenn das Ergebnis völlig unverständlich ist, muß man überlegen, was schief gelaufen ist. Meist sieht man aber an dieser Stelle, wie man weiter verfahren muß. Vielleicht muß ein Punkt, der in der Nähe einer signifikanten Ecke des Randes liegt, genau auf diese Ecke geschoben werden. Vielleicht merkt man jetzt, daß für eine einigermaßen randgetreue Erfassung doch noch die eine oder andere zusätzliche Gitterlinie eingefügt werden muß. Am unangenehmsten ist, wenn in irgendwelchen Ecken die Gitterlinien außerhalb des Rechengebietes geraten sind. Um das zu beheben, muß man eventuell viele Gitterpunkte mit Gefühl von Hand nach innen verschieben.

Sehr wichtig ist, daß man nach jedem einigermaßen als Erfolg erkennbaren Schritt das Ergebnis speichert („*save*“). Wenn nämlich der nächste Schritt eine Katastrophe erzeugt, kann man dann leicht wieder zu dem noch plausiblen vorherigen Zustand zurückkehren. Mit der „*undo*“-Funktion von **AutoCAD** muß man sehr vorsichtig umgehen. Sie zeitigt manchmal seltsame Ergebnisse, weil ein **InGrid**-Vorgang oft aus mehreren **AutoCAD**-Schritten besteht, die man alle rückgängig machen muß.

Ein irregulärer Ausstieg aus **AutoCad**, bei dem die augenblickliche Zeichnung verloren geht, und nur die zuletzt ge“*save*“-te Fassung erhalten bleibt, muß mit der in den Unix-Rezepten (Anhang) beschriebenen Prozedur repariert werden („*kill* usw.). Außerdem ist bei dem Ausstieg eine Datei `bap98a.dw`**`k`** entstanden. Die muß man löschen, sonst geht es nicht mehr weiter.

Es kann passieren, daß man viele Gitterpunkte so verschoben hat, wie man es sich gewünscht hat, und bei der nächsten Glättung rutschen sie alle wieder an eine Stelle, wo man sie nicht haben will. Um das zu verhindern, muß man einigen Punkten, die man künstlich an eine von **InGrid** nicht gerne gesehenen Stelle geschoben hat, mit

Hilfe von „*Punktattribute ändern*“ vorübergehend ein von 0 verschiedenes Attribut geben. Dann werden diese Punkte beim Glätten oder Relaxieren nicht verschoben.

Glätten ist ein einfacher algebraischer Vorgang, bei dem das Programm versucht, Gitterpunkte mit zueinander passenden Indizes (Koordinaten) in etwa gleichem Abstand aufzureihen. *Relaxieren* ist fast dasselbe, nur mathematisch wesentlich komplizierter. Hier versucht das Programm, sich die Gitterlinien als Gummifäden vorzustellen, die alle mit einer Kraft aneinander ziehen, sodaß sich in einem Gleichgewicht ein harmonisches Gitternetz bildet. In der Praxis versucht man mal das eine mal das andere. *Glätten* geht fast immer, *Relaxieren* konvergiert manchmal erst, wenn das Gitter schon einigermaßen fertig ist. Beim *Relaxieren* wird man gefragt, ob das homogen oder inhomogen gewünscht wird. Diese Feinheit wird später erklärt. Zunächst wählt man immer „*homogen*“. So erhält man Schritt für Schritt ein immer besseres randangepaßtes Gitter.

In diesem ersten Zustand sind Krümmungen des Randes meist nicht gut im Gitter wiedergegeben. Man entschließt sich jetzt - zögernd, weil das Gitter nicht schon so bald zu viele Punkte erhalten soll, im Bereich einer Krümmung eine zusätzliche Gitterlinie einzufügen. Nach der Beantwortung sinnvoller Fragen, ob die neue Linie waagrecht oder senkrecht sein soll, und an welcher Adresse sie gewünscht wird, folgt eine Frage, welches Attribut die neuen Gitterpunkte bekommen sollen. Es gibt dazu eine sogenannte „Prioritäten-Liste“, die in [15] beschrieben wird, jedoch trotzdem schwer zu durchschauen ist und deshalb zunächst vermieden werden sollte. Man verlangt, daß alle neuen Punkte das Attribut 0 bekommen sollen, und ändert danach von Hand die wenigen Punktattribute, die von 0 verschieden werden müssen (nämlich die auf dem Rand). Das wird später sehr arbeits- und zeitintensiv, weshalb man sich mit der Prioritätenliste später doch irgend einmal auseinandersetzen sollte.

Inzwischen sind vielleicht so viele Gitterpunkte entstanden, daß das Bild auf dem Bildschirm unübersichtlich wird, und man auch die Adressen bei den Punkten nicht mehr lesen kann. Hier hilft ein Knopf von **AutoCAD**, der mit einem kleinen Flugzeug gekennzeichnet ist. Klickt man das an, erscheint ein Fenster, in dem ein verkleinertes Abbild des ganzen Gitters und ein violettes Rechteck zu sehen ist. Klickt man in diesem Rechteck, kann man es durch Verschieben der Maus größer und kleiner machen. Klickt man nochmal, kann man das nun in seiner Größe feste Rechteck mit der Maus an jede beliebige Stelle des Gitters schieben. Ein Klick auf die mittlere Maustaste vergrößert das so ausgewählte Gebiet auf Bildschirmgröße. Mit diesem Hilfsmittel (rückgängig zu machen mit dem Befehl „*view - zoom - all*“ - oder „*limits*“) kann man überall in dem Gitter hinkommen und schwierige Gebiete genau betrachten.

Die Punkte von nachträglich eingefügten Gitterlinien, die auf einen Rand gehören, liegen noch nicht alle auf diesem Rand. Man schiebt sie mit der Maus dorthin und darf „*nahe Punkte pinnen*“ nicht vergessen. Es ist jetzt auch nötig, nachzusehen, wo auf einem anderen Teil des Randes durch das Einfügen einer Linie die Punkte sinnlos ungleichmäßig verteilt sind. Das korrigiert man mit „*äquidistantes Füllen*“. Wenn dabei Katastrophen passieren, liegt es meist daran, daß aus irgendeinem Grund dieser Rand jetzt nicht mehr eine durchgehende Polylinie ist. Man muß dann erst *Gitterlinien - löschen*, *redraw* und anschließend die Polylinien wieder verbinden. Es kommt vor, daß nach einigen Manipulationen plötzlich Ecken nicht mehr zusammenstoßen, die vorher einwandfrei verbunden waren!

Bei einem einigermaßen komplizierten Rand ist es meist nicht zu vermeiden, daß einige Gitterzellen in ihrer Form noch sehr weit von einem Rechteck oder von dem als Ideal anzustrebenden Quadrat abweichen. Je „trapezförmiger“ sie sind, desto schlechter für die spätere Rechnung. Man muß abwägen, ob die schrägen Zellen in einem für die Rechnung wichtigen Gebiet sind, oder nicht. Ganz schlecht sind schräge Zellen in der Nähe eines Randes, auf dem später die Neumann-Randbedingung gelten soll! Das muß unbedingt vermieden werden, weil sonst im Unterprogramm **SOR** des PIC-Code keine Konvergenz erreicht wird. In diesem Falle hilft meist nur, das Modell durch einen außerhalb des Rechengebietes liegenden Bereich zu ergänzen, sodaß die Gitterlinien senkrecht durch den Rand gehen können. Das kostet zusätzlichen Speicherplatz und Rechenzeit, ist aber manchmal nicht zu vermeiden. Unbrauchbar sind auch Zellen, bei denen ein Punkt nach innen geht, oder solche, die wie ein Dreieck aussehen.

Manchmal genügt es, die Gitterlinien künstlich in eine bestimmte Richtung zu „zerren“. Man muß dann gegen die beim Relaxieren entstehende gleichmäßige Anspannung der „Gummilinien“ arbeiten. Dazu gibt es verschiedene Methoden, die sehr viel Übung und Erfahrung verlangen. Unter „*InGrid -Inhomogen - aus Gitter*“ werden vom Programm die „Zugkräfte“ an den Gitterpunkten berechnet. Man bekommt dann unter „*ändern-einzeln*“ ein Fenster, in dem für einen angeklickten Punkt diese Zahlen angegeben sind. Da man sich anschaulich nichts unter diesen Zahlen vorstellen kann, bleibt einem nichts anderes übrig, als eine willkürlich zu ändern. (Nicht zimperlich: Faktoren 10 oder 100 sind nicht ungewöhnlich, oder Vorzeichen wechseln). Danach macht man „*relaxieren - inhomogen“* und sieht nach, wohin der Punkt gewandert ist. Entsprechend kann man bei der nächsten Änderung schon ein bißchen besser abschätzen, wieviel man ihn ändern muß.

Das ist so viel Empirie, daß es nach Möglichkeit vermieden werden sollte. Ähnliches gilt für eine andere Methode: Man fügt in der kritischen Gegend eine Gitterlinie ein, relaxiert homogen und entfernt danach die Gitterlinie wieder. Dann sind sozusagen die „Zugkräfte“ der nicht mehr vorhandenen Linie noch wirksam.

Wenn dies alles zu einem zwar noch groben, aber in seiner Gesamtanlage zufriedenstellend aussehendem Gitter geführt hat, muß es verfeinert werden. Das ist ein Menüpunkt in **InGrid**, bei dem man nur gefragt wird, um welchen Faktor verfeinert werden soll, und welches Attribut die neuen Punkte erhalten sollen. Wie oben verlangt man das Attribut 0, solange man die Prioritätenliste noch nicht verstanden hat.

Wie fein muß das Gitter sein? Es kann hier nur auf Gitter verwiesen werden, mit denen schon gerechnet worden ist und empfohlen werden, die Maschengröße etwa genauso zu machen. Das ist eine sehr ungenaue Angabe, aber im Moment die einzige mögliche. Hinterher kann man einen Versuch machen und die Maschenweite weiter verfeinern. Wenn die Rechenergebnisse mit grobem und feineren Gitter übereinstimmen, war die Gitterweite richtig gewählt.

Von jetzt an werden **AutoCAD** und **InGrid** sehr langsam. Jeder Schritt muß sehr gut überlegt werden, weil man minutenlang warten muß, bis man das Ergebnis sieht. Auch hier empfiehlt sich wieder das häufige Speichern, obwohl auch dies viel Zeit kostet.

Es sind jetzt viele neue Gitterlinien entstanden. Auf allen Krümmungen liegen die neuen Punkte etwas neben der Randkurve. Um sie mit „*nahe Punkte pinnen*“ auf die Kurve zu ziehen, tut man gut daran, ihnen allen zuerst das Attribut K zu geben. Hilfreich ist hier der Menüpunkt "*Punktattribute ändern - Bereich*“. Damit kann man einen Adressenbereich angeben, in dem alle Attribute geändert werden (vorher unbedingt „save“!)

Ist das alles richtig gelungen, speichert man das ganze Gitter nochmal unter einem anderen Namen. Denn jetzt werden die Punktattribute endgültig so geändert, wie sie für den PIC-Code gebraucht werden. Alle Punkte, die zur Anode gehören, erhalten z. B. das Attribut 1. Alle Punkte, die zur Kathode gehören, erhalten das Attribut 2. Alle Feldpunkte, also alle Punkte innerhalb des Rechengebietes behalten das Attribut 0. Die Punkte auf der Achse und die am oberen Rand gehören weder zur Anode noch zur Kathode. Sie erhalten die Attribute 3 und 4. [Diese Zahlen kann man auch anders wählen. Aus historischen Gründen sind leider die Attribute bei den vorhandenen Gittern immer wieder anders bezeichnet].

Wenn eine Elektrode nicht einfach und durchgehend ist, sondern aus Teilstücken besteht, kann es nützlich sein, den Teilen jeweils ein anderes Attribut zu geben. Alle Attribute müssen zuletzt Zahlen sein! Buchstaben als Attribut versteht der PIC-Code nicht. Wenn das geschehen und richtig ist, kann man das Gitter exportieren. **InGrid** fragt dann, ob es nach **ANSYS** oder nach **KADI2D** exportiert werden soll. Man gibt die entsprechenden Namen an und wartet, bis **AutoCAD** wieder befehlsbereit ist. Dann kann man **AutoCAD** verlassen. Es gibt jetzt eine Datei mit dem Namen z. B. „`bap98a.kdi`“ und eine mit dem Namen `bap98a.ans`.

4. Export zu **ANSYS** und nötige Ergänzungen

Es muß nun noch an das obere Ende der Datei `bap98a1.ans` eine Datei kopiert werden, die „`kopf`“ genannt wurde. Der Kopf enthält die Angaben über den Element-Typ, die **ANSYS** für die Berechnung benötigt. Nachdem dieser Kopf oben an die Datei kopiert wurde, muß man die entsprechenden Zeilen, die **InGrid** erzeugt hat, löschen.

Nach dem Aufruf von **ANSYS** muß vor dem Einlesen noch das Makro „`ROTATION`“ ablaufen. Das sorgt dafür, daß **ANSYS** das Gitter in einer um 90° gedrehten und gespiegelten Form einliest, damit für die **ANSYS** - Rechnung die Rotationsachse senkrecht steht, also Z nach oben und R nach rechts zeigt.

Jetzt kann die Datei in **ANSYS** eingelesen werden (*read input from*). Danach beginnt der Aufbau des **ANSYS**-Modells, der weiter unten beschrieben wird.

5. Export zum PIC-Code und nötige Ergänzungen.

Vor die Datei `bap98a1.kdi` muß eine kurze Datei kopiert werden, die „`head`“ genannt wurde. Was in diesen `head` hineingeschrieben werden muß, ist sehr trickreich und wird im Prinzip in [17] und [18] genau beschrieben. Es handelt sich um Angaben, die der PIC-Code für die Berechnung benötigt. Nur wenn man diesen `head` schon einigermaßen richtig ausgefüllt hat, kann man mit dem Auswerteprogramm „**geisha**“ das Gitter betrachten und danach eventuelle Fehler korrigieren. **Geisha** erzeugt nicht nur Bilder, aus denen man viel lernen kann, sondern auch eine Liste von Elektrodenzellen, die nach einiger Einarbeitung aufschlußreich und nützlich zur Fehlerbehebung ist.

Beschreibung von „head“:

1. Zeile: ein Text zur näheren Beschreibung des Gitters, z. B.
Applied - B - Diode bap98a

2. Zeile: die beiden Zahlen I1MX und I2MX bedeuten die Maximalzahlen der Gitteradressen in X-Richtung (I1) und in y-Richtung (I2). Diese Zahlen stehen in dem von InGrid erzeugten file unter NGRIDP. Also z. B.
I1MX = 25 I2MX =151

NGRIDP (NUMBER GRID POINTS)
hier werden diese beiden Zahlen noch einmal wiederholt

ELECAT (ELECTRODE ATTRIBUTES)

erste Zeile: Anzahl der Anodenattribute Anzahl der Kathodenattribute, also meistens:
1 1

zweite Zeile: Das Anodenattribut ist z. B. 1, das Kathodenattribut ist 2, also:
1 2

STARTP (START-POINTS FOR ELECTRODE DETERMINATION)

erste Zeile: meist gibt es einen Startpunkt für die Anode und einen für die Kathode:
1 1

darunter stehen zwei Zeilen, die die Adresse angeben, wo die Anode beginnt und wo die Kathode beginnt. Wenn die Anode bei der Adresse 1, 1 beginnt, steht hier also zuerst:

1, 1, und dahinter die Anzahl der Anodenzellen. Die kann man noch einigermaßen raten, aber es ist auch nicht schlimm, wenn man dabei einen Fehler macht. Man sieht es nachher. Also

1, 1, 150

Ein ganz großes Geheimnis ist, daß in den meisten Fällen vor der ersten 1 ein Minuszeichen stehen muß. Das hat etwas damit zu tun, in welcher Weise die Ströme in den Anodenzellen aufaddiert werden. Oft funktioniert gar nichts, wenn man das - wegläßt. Genauere Hinweise hierzu finden sich in [1] und [18] Also heißt diese Zeile

-1, 1, 150

Darunter steht eine Zeile für den Beginn der Kathode. Die beginnt z. B. bei der Adresse 25, 1. Das Programm versteht das nur, wenn man als Anfangsadresse die **linke untere Ecke** der ersten Kathodenzelle angibt, also nicht 25, 1 sondern:

24, 1, und danach wieder eine geschätze Anzahl von Kathodenzellen, also z. B.:

20, 1, 166

hier ist noch nie ein Minuszeichen benötigt worden.

Es kommt vor, daß es mehrere Anoden oder Kathoden gibt. Dann gibt es eben nicht einen Startpunkt für Anode und Kathode, sondern es steht unter STARTP

2 3

und darunter die zwei Zeilen für die Anfänge der zwei Anoden und 3 Zeilen für die Anfänge der drei Kathoden, ... etc.

Nun fehlt noch die letzte und schwierigste Angabe:

ETYPES (ELECTRODE TYPES)

Es gibt nämlich verschiedene Arten von Elektrodenzellen. Bei der Anode z. B. gibt es Zellen, die bei Vorhandensein einer gewissen Feldstärke Ionen emittieren, und andere, die keine Ionen emittieren. Diese Elektrodenzellen unterscheidet man durch die Elektrodentypen 1 und 2. Wenn z. B. die Anodenzellen von Nummer 1 bis 64 nicht emittieren, die von Nummer 65 bis 87 emittieren, und die von 88 bis 150 wieder nicht, so beginnt die Angabe unter ETYPES so:

```
3   _     _

   1,  64   2
  65,  87   1
  88,150   2
```

Diese 150 muß mit der Zahl übereinstimmen, die man oben unter STARTP für die Anzahl der Anodenzellen angegeben hat. Sie ist vielleicht noch falsch, weil sie nur geschätzt wurde.

Für die Kathodenzellen geht es analog weiter. Nur muß man wissen, daß die erste Kathodenzelle die Nummer 151 hat; es wird also einfach weitergezählt. Wenn man bisher alles richtig gemacht hat, geht das auch gut. Sind aber in den bisherigen Angaben schon Fehler, dann wird weder das Bild, das **geisha** zeichnet, noch die Liste der Elektrodenzellen so aussehen, wie man es erwartet. Das kann so schlimm sein, daß man aus beidem überhaupt nichts sehen kann, also auch zunächst nicht weiß,

wie man den Fehler beheben soll. Es hilft deshalb am meisten, jedes neue Gitter in seinem logischen Aufbau und damit mit seinen Einträgen im head möglichst einem schon vorhandenen nachzuempfinden und sich langsam von dem alten zu dem neuen Gitter hinarbeiten.

Die nächste Zahl in der Zeile unter ETYPES ist die Anzahl der Kathodenzellen-Typen, also z. B. 4, die tragen die Bezeichnung -1, -2, -3, wobei die -3 zweimal vorkommt. Es wird dann also so weitergehen:

3 4 7 (die 7 ist die Gesamtsumme 3 + 4)

```
  1,  64   2
 65,  87   1
 88,150   2
151,166  -3
167,244  -1
245,270  -2
271,316, -3
```

die letzte Kathodenzelle , Nr. 316, muß die 166. Kathodenzelle sein - diese Angabe muß mit der oben unter STARTP gemachten Zahl übereinstimmen. Der PIC-Code verlangt, daß den Anodenzellen vom Typ 1 (die Ionen-emittierend sind) Kathodenzellen vom Typ -1 gegenüberstehen.

Es folgt danach nun nur noch das Wort

GRIDP

und danach die Gitterpunkte, wie sie von **InGrid** geliefert worden sind.

Die Zahlen, die man in diesem head, und insbesondere unter ETYPES geschrieben hat, können noch viele Fehler enthalten. Man kann sie nach und nach richtig machen, wenn man erst einmal ein Bild von dem Gitter hat, und anfangen kann, die Elektrodenzellen zu zählen. Dieses Bild erzeugt man durch den Aufruf

geisha

Es kommt dann ein Menü, in dem man „*convert grid file in a visart file*“ anklicken muß. Dahinter verbirgt sich ein Programmsystem „**Visart** [19], das von **geisha** aufgerufen wird, ohne daß der Benutzer etwas davon wissen muß. Man erhält ein Fenster, in dem man angibt wie der Gitterfile heißt, und wie die zu erzeugenden Visart-files heißen sollen. Dann drückt man Ausführung und erhält nach einer Weile eine Meldung daß man *upper window* drücken soll. In diesem Fenster verlangt man „*create Tecplot files“* und gibt noch einige Dateinamen an. Wenn alles gut geht, meldet sich **Tecplot** [20]. Darin erscheint ein Menü, was man alles ansehen kann. Man drückt auf „*next*“ so oft, bis man „*Gitter*“ angezeigt bekommt. Das Bild kann man mit *file print* sofort z. B. auf dem HP-Tintenstrahldrucker im Bau 630 drucken. Unter „*view*“ gibt es die Möglichkeit „*zoom*“, mit der man auch deutlich überblickbare Ausschnitte ansehen und drucken kann.

In demselben Kästchen, in dem man „*Gitter*“ angeklickt hat kann man auch „*Elektrodentypen*“ verlangen. Man bekommt buntes Bild, aus dem man erkennen kann, ob man die Elektrodentypen richtig angegeben hat. Zu jedem Elektrodentyp gehört eine andere Farbe. Jetzt beginnt ein großes Zählen und ein schrittweises Annähern an die korrekten Elektrodentypen. Dabei muß man leider immer wieder **geisha** verlassen und **tspful** (oder einen anderen Editor) aufrufen, die Zahlen im `head` ändern, **geisha** wieder aufrufen, die entsprechenden files wie oben erzeugen lassen und die Bilder wieder anschauen, bis man mit allem zufrieden ist.

Auch „*Punktattribute*“ ergibt ein schönes buntes Bild, in dem man eventuelle Fehler entdecken kann. Auch hier ergibt „*Zoom*“ und „*redraw*“ ein klareres Bild von Ausschnitten.

In diesem Stadium ist eventuell die Liste der Elektrodenzellen hilfreich, die von **geisha** mit dem suffix `.list` erstellt worden ist. Es folgt hier eine verkürzte Version:

Liste der Elektrodenzellen

```
E    Nr     I1     I2   Fall  Bez    Typ    Mat
```

Die erste Zeile enthält eine Überschrift, die man so verstehen soll:
E: Elektrode, also A für Anode und K für Kathode. Nr: Die Nummer der Elektrodenzelle, beginnt also mit 1 und steigt bis 316, der letzten Elektrodenzelle. I1 und I2 sind die Adressen dieser Zellen. Die Anode fängt also mit der Adresse 1, 1 an und geht hoch bis 1, 150. Die Angaben unter Fall, Bez und Typ kann man ignorieren, unter Mat ist der Elektrodentyp aufgeführt, der im head angegeben wurde. Dann beginnt die Liste so:

```
A     1      1      1      4   W       2      2
A     2      1      2      4   W       2      2
A     3      1      3      4   W       2      2
A     4      1      4      4   W       2      2
A     5      1      5      4   W       2      2
A     6      1      6      4   W       2      2
A     7      1      7      4   W       2      2
A     8      1      8      4   W       2      2
A     9      1      9      4   W       2      2
A    10      1     10      4   W       2      2
```

und so weiter, bis 64, wo sich der Elektrodentyp von 2 nach 1 ändert, wie es oben angegeben wurde:

```
A    63      1     63      4   W       2      2
A    64      1     64      4   W       2      2
A    65      1     65      4   W       2      1
A    66      1     66      4   W       2      1
A    67      1     67      4   W       2      1
A    68      1     68      4   W       2      1
```

und so weiter, bis 87, wo sich der Elektrodentyp von 1 nach 3 ändert, wie es oben verlangt wurde:

```
A    85      1     85      4   W       2      1
```

```
A    86      1     86      4   W       2      1
A    87      1     87      4   W       2      1
A    88      1     88      4   W       2      3
A    89      1     89      4   W       2      3
A    90      1     90      4   W       2      3
A    91      1     91      4   W       2      3
```

und so weiter, bis 150, der letzten Anodenzelle. Danach beginnt mit Zelle 151 die Kathode bei der Adresse 24, 1. Die ersten Kathodenzellen haben den Elektrodentyp -3.

```
A   147      1    147      4   W       2      3
A   148      1    148      4   W       2      3
A   149      1    149      4   W       2      3
A   150      1    150      4   W       2      3
K   151     24      1      2   O       2     -3
K   152     24      2      2   O       2     -3
K   153     24      3      2   O       2     -3
```

und so weiter, entsprechend für alle vorkommenden Kathodentypen.

```
K   165     24     15      2   O       2     -3
K   166     24     16      2   O       2     -3
K   167     24     17      2   O       2     -1
K   168     24     18      2   O       2     -1
K   169     24     19      2   O       2     -1
K   170     24     20      2   O       2     -1
K   171     24     21      2   O       2     -1
K   172     24     22      2   O       2     -1
```

und so weiter

```
K   240     24     90      2   O       2     -1
K   241     24     91      2   O       2     -1
K   242     24     92      2   O       2     -1
K   243     24     93      2   O       2     -1
K   244     24     94      5  NO       3     -1
K   245     23     94      1   N       2     -2
K   246     22     94      1   N       2     -2
K   247     21     94      1   N       2     -2
K   248     20     94      1   N       2     -2
K   249     19     94      1   N       2     -2
K   250     18     94      1   N       2     -2
```

und so weiter

```
K   266     24    100      2   O       2     -2
K   267     24    101      2   O       2     -2
K   268     24    102      2   O       2     -2
K   269     24    103      2   O       2     -2
K   270     24    104      2   O       2     -2
K   271     24    105      2   O       2     -3
K   272     24    106      2   O       2     -3
K   273     24    107      2   O       2     -3
```

und so weiter bis zur letzten Kathodenzelle, die die Nummer 316, die Adresse 24, 150 und den Typ -3 hat.

```
K   311     24    145      2   O       2     -3
K   312     24    146      2   O       2     -3
K   313     24    147      2   O       2     -3
```

```
 K   314     24    148      2   0       2     -3
 K   315     24    149      2   0       2     -3
 K   316     24    150      2   0       2     -3
```

Wenn in dieser Liste Ungereimtheiten sichtbar sind, z. B. die Adressen nicht wie erwartet monoton ansteigen, oder zwischen den einzelnen Zeilen öfters die Adresse 0 oder negative Adressen auftauchen, dann ist im head noch etwas falsch und muß aufgrund dessen, was die Liste und die farbigen Bilder von geisha aussagen, korrigiert werden.

Die fertige Gitterdatei könnte dann so aussehen:

```
Applied-B-Diode bap98a
I1MX =25   I2MX =151
NGRIDP (NUMBER GRID POINTS)
 25 151
ELECAT (ELECTRODE ATTRIBUTES)
 1 1
 1 2
STARTP (START-POINTS FOR ELECTRODE DETERMINATION)
1 1
-1,01,150
24,01,166
ETYPES (ELECTRODE TYPES)
3 4 7
  1, 64  2
 65, 87  1
 88,150  3
151,166 -3
167,244 -1
245,270 -2
271,316 -3
GRIDPT
    1    1  0.0000000E+00  0.0000000E+00    1
    2    1  3.0583333E-03  0.0000000E+00    4
    3    1  6.1166667E-03  0.0000000E+00    4
    .
```

hier sind alle weiteren Gitterpunkte ausgelassen !

```
    .
    .
   23  151  9.7958333E-02  1.5000000E-01    3
   24  151  1.0067917E-01  1.5000000E-01    3
   25  151  1.0340000E-01  1.5000000E-01    2
```

Die Gitterpunkte sind also durch ihre Adressen I1 und I2, ihre geometrischen Koordinaten in Metern (Z, R) und hinten durch ihr Attribut charakterisiert.

Bei einer Diode ohne extern angelegtes Magnetfeld könnte jetzt mit der PIC-Rechnung begonnen werden. Für die hier als Beispiel gewählte Applied-B-Diode wird jedoch zunächst das Magnetfeld benötigt.

III. Berechnung externer Magnetfelder

1. Kurze Gebrauchsanweisung für **ANSYS**.

ANSYS, z. Zt. in der Version 5.3 steht auf dem Rechner der Universität Karlsruhe zur Verfügung. Die Prozedur, dorthin zu kommen, ist einfach:

Von der inrrisc8 aus schreibt man:

splogin

dann wird man nach seinem Paßwort gefragt, es muß das für die inrrisc8 gültige Paßwort eingegeben werden.

Damit ist die Verbindung mit der Universität Karlsruhe hergestellt. Man stellt fest, daß die eigenen Verzeichnisse von der inrrisc8 dort genauso wie hier vorhanden sind.

Wenn jetzt „standard input“ dasteht, muß man so oft die Leertaste drücken, bis ein prompt mit dem eigenen Namen erscheint.

Dann geht man in das richtige Verzeichnis und schreibt

export DISPLAY=inrxterm28.fzk.de:0 (bzw. entsprechend den Namen des eigenen Displays)

xansys53 &

ANSYS meldet sich dann mit einem kleinem grünen Fenster, in dem man z. B. das **ANSYS**-Hilfeprogramm aufrufen kann. Da dieses zwar weitgehend selbst erklärend ist, aber im jetzigen Stadium dem Anfänger nicht viel nützt, wird hier auf eine Einführung in das „help“-Programm verzichtet.

a) Wie betrachtet man ein bereits von ANSYS gerechnetes Problem?

Man wählt „*Interactive*“. Dann erscheint ein Fenster, in dem man den Verzeichnispfad und den Namen des Problems, das man bearbeiten oder ansehen möchte, angeben muß. [Im vorliegenden Beispiel wählt man also als Pfad `/fzk/inr/home/bauer/bfcpic/bap98a` und `bap98a4` als file-Name]. In diese Fenster kann man nur schreiben, wenn sie mit der Maus hervorgehoben sind. Nach „OK“ erscheint zuerst das **ANSYS**-Output-Fenster, in dem viele uninteressante Informationen stehen; zuletzt wird man aufgefordert, mit "Enter“ fortzufahren, und dann erscheint nach einiger Zeit das „Graphische Benutzer-Interface GUI“

Das ist die Arbeitsoberfläche von **ANSYS**, mit der man sich vertraut machen muß. Sie ist in Fig. 2 abgebildet. Es gibt blaue, graue und grüne Felder. Große **ANSYS**-Experten benutzen nur das graue Feld „**ANSYS** Input“ und schreiben darin kauderwelsch-artige **ANSYS**-Befehle („commands“). Diese commands findet man unter „*Hilfe, Help*“ in dem Abschnitt „*Commands manual*“, aber nur wenn man schon weiß, wie sie heißen. Dies wird hier nur deshalb erwähnt, weil die Ratschläge, die man z. B. durch Anrufen der hotline der Firma CADFEM, die **ANSYS** vertreibt, bekommt, meist aus Hinweisen auf solche commands bestehen. Sie sind also für den Anfänger fast nutzlos. Im „Commands manual“ des Hilfeprogramms sind sie alle beschrieben. Am Ende jeder Beschreibung steht der Weg, wie man diesen command durch Klikken im GUI erreichen kann.

Anfänger benutzen vorläufig nur die graphische Oberfläche. Es wird empfohlen, als erstes ein schon vorhandenes und fertig gelöstes Problem anzuschauen. Als Beispiel könnte das in dem Verzeichnis
`/fzk/inr/home/bauer/bfcpic/bap98a4` befindliche Problem `bap98a4 die-`

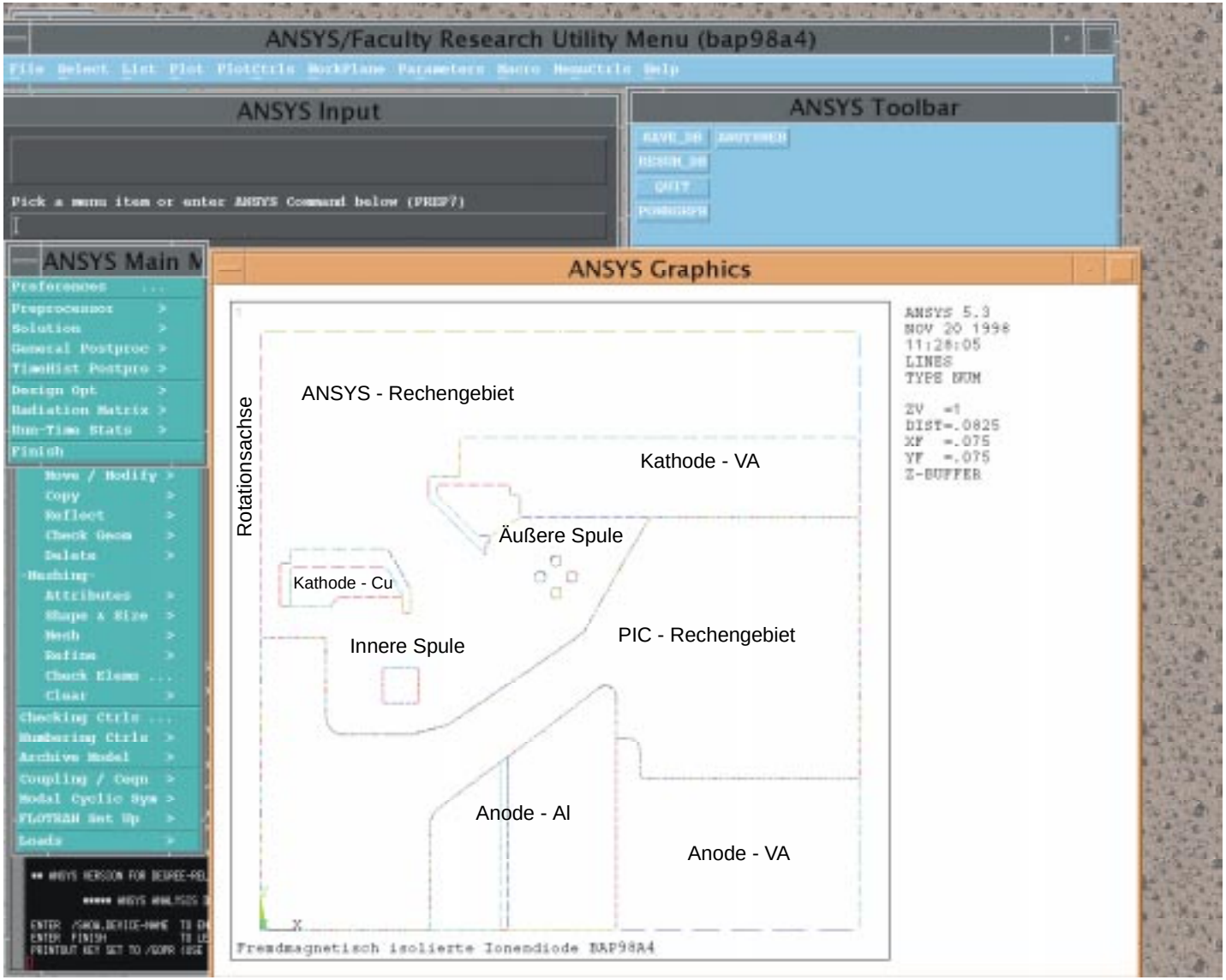

Fig. 2: ANSYS-Benutzeroberfläche. Im Graphikfenster sind „Lines“ der Fremdmagnetisch isolierten Ionendiode dargestellt.

nen. In der blauen Tool-bar muß man *RESUM_DD* anklicken. [Bitte nicht versehentlich auf „*SAVE*“ drücken! Sonst ist das Beispiel weg!!] Mit „*RESUM*“ holt man das schon vorhandene Problem (nämlich das, was man beim Einstieg ausgewählt hat) in den Arbeitsbereich von **ANSYS** zurück.

Ganz oben in der blauen Zeile steht der Befehl „*Plot*“. Wenn man den anklickt, kommt ein Menü, in dem man auswählt, was man in dem Graphik-Fenster sehen will:

Z. B. „*lines*“, oder „*areas*“, oder „*elements*“ oder „*nodes*“. Das ist alles schon recht erleuchtend und selbsterklärend. Für Fig. 2 ist z. B. „*plot lines*“ verlangt worden.

Als nächstes macht man sich mit dem Menüpunkt „*plotctrls*“ vertraut. Vorsicht! Das kann ungeahnte Wirkungen erzielen! *Pan-Zoom-Rotate* erzeugt ein blaues Fenster, in dem man z. B. „*box zoom*“ anklicken kann. Man bekommt dann einen Cursor, mit dem man einen Ausschnitt wählen kann. „*Numbering*“ bringt ein Fenster hervor, in dem man angeben kann, ob man die Nummern der lines, areas, elements, etc. sehen will. Man wird auch gefragt, ob die gewählten Nummern noch mit Farben verschönt werden sollen. Das kann man alles ausprobieren und wird es später sehr gut nutzen können.

Weiter unten unter „*plot controls*“ gibt es einen Befehl „*hard copy*“, mit dem man z. B. das auf dem Bildschirm befindliche Bild oder auch nur das Graphik-Fenster auf einen Drucker schicken kann. Achtung! Das muß man sich gut überlegen, weil es den Drucker sehr lange beschäftigt. Auch muß man unmittelbar vor dem Abschicken des Plot-Befehls das Graphik-Fenster etwas nach rechts verschieben und nochmal anklicken, weil sonst darüber liegende andere Fenster mitgedruckt werden. Stattdessen kann man diese Bilder auch als file.ps in eine Datei schreiben und diese dann später mit *sprint* auf einen Postscript-Drucker schicken.

An dieser Stelle sei eine Anmerkung über den Umgang mit Bildern eingefügt: Fig. 2 wurde wie beschrieben als *file.ps* von **ANSYS** ausgegeben. Mit **FTP** wurde dieser file aus der UNIX-Welt in den PC geholt und in **CorelDraw8** als Postscript - Datei importiert.

In **CorelDraw** kann man das Bild etwas bearbeiten, z. B. wie hier erläuternde Texte hineinschreiben und kann es auch gut drucken und als `.cdr`-Datei speichern. Dann wird es als `.wmf`-Datei exportiert. Wie sie mittels „*Einfügen Graphik*“ in **word97** angekommen ist, zeigt das obige Bild.

In der schmalen blauen Befehlsleiste oben steht ganz links der Befehl „*File*“ Darunter verbirgt sich ein großes Menü, das weitgehend von anderen Programmen bekannt ist. Es gibt aber einen Unterpunkt, auf den hier hingewiesen werden muß: Mit „*Change Title*“ kann man den Titel ändern, der im Graphik-Fenster unter der Zeichnung steht. Es trägt sehr zur Ordnung bei, wenn man hier dafür sorgt, daß der Titel im Bild identisch ist mit dem Namen des Problems, das man gerade bearbeitet. Das kommt leider nicht von selbst. Also, wenn das Problem `bap98a4` beim Einstieg in **ANSYS** ausgewählt wurde, sollte hier auch dieser Name als Titel stehen.

Da dieses Modell schon gerechnet ist, kann man in dem kleinen grünen Fenster „**ANSYS** *Main Menü*“ den „*General Postprocessor*“ aufrufen. In dem dann erscheinenden Fenster verlangt man „*plot results*“ und danach *2d-flux-lines*. Das darauf er-

scheinende Bild ist etwas unbefriedigend. Es wird empfohlen, unter „*Plot Ctrls*“ den Punkt „*erase options*“ aufzurufen und darin das Schalterchen „*erase between plots*“ zu deaktivieren. Dann bleibt das Bild erhalten, wenn man nun zusätzlich noch „*plot lines*“ macht. Man sieht dann das Modell mit seinen Linien und dazu die rA_z - Äquipotentiallinien.

„*Erase between plots*“ muß man schnell wieder aktivieren, sonst bekommt man ein großes Durcheinander bei den nächsten Aktionen im Graphik-Fenster.

Im „*General Postprocessor*“ kann man auch einen Befehl: „*Path operations*“ aufrufen. In dem dann erscheinenden Menü steht als oberster Befehl „*define path*“. Jetzt wird man nach den Knoten („nodes“) gefragt, von wo bis wo ein Pfad gewünscht wird. Man muß also erst mal „*plot nodes*“ machen, dann kann man zwei oder mehrere Knoten anklicken, die den Pfad definieren. Danach verlangt man „*map onto path*“ und gibt an, daß man Bx und By sehen will. „*Plot path items*“ ergibt dann die Kurven von Bx und By als Funktion des Ortes auf dem Pfad. Falls vorher gezoomt wurde, muß man dazu mit „*reset*“ das volle Bild herstellen. Wenn man nun auch noch sehen will, wo der Pfad verläuft, geht man so vor: Unter *Plot Ctrls* verlangt man „*Symbols*“ und in dem dann erscheinenden Fenster „individual“ und ok. Dann erhält man eine lange Liste, in der ganz unten „*postprocessor path geometry*“ aktiviert werden muß. Wenn man danach *plot nodes* macht, sieht man den Pfad! Man kann einen neuen Pfad definieren. Dann ist der alte Pfad und alles was damit geplottet wurde, verloren.

In dieser Weise kann man jeden Tag neue Möglichkeiten von **ANSYS** entdecken. Es soll an dieser Stelle diese ausführliche Darstellung verlassen werden; es folgt eine Beschreibung, wie man ein ganz neues Problem angeht, das noch nicht vorhanden ist.

b) Schrittweises Entwickeln eines neuen ANSYS-Problems

Man verläßt jetzt **ANSYS** mit „*Quit*“, wobei „*no save*“ gewählt werden sollte, weil wir nichts von dem Beispielfall verändern werden soll.

Für ein neues Problem haben wurde ja vorher mit **InGrid** ein Gitter für den PIC-Code erzeugt und in eine Datei, z. B. `reo3.ans` exportiert. (Es wird jetzt ein einfacheres Beispiel gewählt, um nicht zuviel schreiben zu müssen) Es ist jetzt günstig, zunächst ein Verzeichnis anzulegen (*mkdir reo3*), in dem alle dazu gehörigen Dateien landen sollen. In dieses Verzeichnis müssen folgende Dateien kopiert werden:

```
ROTATION.MAC
BAUS.MAC
reo3.ans
```

`ROTATION` ist das Makro, das die Rotation des von **InGrid** erzeugten Gitters bewirkt, `BAUS` ist ein Makro, das später beim Export der **ANSYS**-Ergebnisse für den PIC-Code gebraucht wird. `reo3.ans` ist das Gitter, das vorher mit **InGrid** erzeugt und mit dem richtigen „`kopf`“ versehen wurde. (Den richtigen Kopf erkennt man daran, daß dabei das Element plane13 vorkommt; mehr über Elemente siehe weiter unten.)

Wenn diese Dateien vorhanden sind, kann **ANSYS** aufgerufen und der Pfad dorthin im Eingangsmenü angegeben werden. Jetzt wird in das graue Command-Feld

„*ROTATION*“ geschrieben und „*Enter*“ gedrückt. Danach kann man in dem blauen Feld ganz oben links „*file*“ anklicken und dann „*read input from“*. Man wird gefragt, was eingelesen werden soll, und klickt „*reo3.ans*“ an.

Dann tut sich etwas, was eine Weile dauert und mit der Darstellung von bunten Elementen in einem Rechteck endet. Wenn in diesem Stadium gelbe Warnungen und unverständliche Meldungen erscheinen, kann man das meist ignorieren und später durch „ok“ zum Verschwinden bringen. Das Beispiel sieht anders aus als das vorige, man kann aber bei genauem Hinsehen die Kontur einer Elektronendiode erkennen. Es ist aber nur das InGrid-Gitter für den PIC-Code, Magnetspulen und Magnetberechnungsgebiet fehlen noch.

Aus gutem Grunde gewöhnt man sich an, in das graue Command-Feld gelegentlich einen Kommentar zu schreiben, damit man später zurückverfolgen kann, was man gemacht hat.

Deshalb schreiben wir hier jetzt !**********Eingabe des **ANSYS**-Modells**********
Das Ausrufezeichen am Beginn der Zeile interpretiert **ANSYS** als Kommentarzeile.

Danach drückt man *Preprocessor* in dem grünen *Main-Menü*. Man sieht dann ein Menü, in dem man der Reihe nach durch alle vorbereitenden Schritte geführt wird. Die ersten drei sind schon erledigt, nämlich die Wahl des Elements und seiner Materialkonstanten, sowie die Wahl des MKS-Systems. Dies steckt nämlich schon im Kopf der eben eingelesenen Datei. Man kann also gleich zu „*Create*“ gehen und muß zunächst „*Keypoints*“ erzeugen. Das sind Punkte, die bei der jetzt zu definierenden Geometrie eine Rolle spielen. Wenn alles richtig angeklickt ist (*Create, Keypoints, „on working plane*“ erscheinen zwei hellgrünes Fenster, in denen man aufgefordert wird, die Koordinaten der Keypoints einzugeben. Man gibt z. B. nacheinander in dem oberen Fenster folgendes ein:

0,-0.275,0	*enter*	die letzte 0 ist die Z-Koordinate, man kann sie bei ebenen Problemen auch weglassen.
0.85,-0.275,0	*enter*	
0.85,0.575,0	*enter*	
0,0.575,0	*enter*	

und zuletzt „*ok*“ in dem grünen Fenster unten links. Wenn jetzt *plot Keypoints* verlangt wird, sieht man die 4 neu erzeugten Keypoints (ganz klein und schwach in einem gewissen Abstand außerhalb des **InGrid**-Modells).

Jetzt soll durch die Keypoints eine Fläche definiert werden, nämlich das Rechengebiet für **ANSYS**. Dazu wird unter „*Create“ „Areas“* und „*arbitrary*“ angeklickt und danach „*Through Keypoints*“. Der Mauszeiger hat sich jetzt in einen nach oben gerichteten Pfeil verwandelt, mit dem man die neuen Keypoints der Reihe nach anklicken kann. Es kommt vor, daß dabei in einem gelben Fenster die Warnung erscheint: “There are two keypoints..“ Man wählt dann den mit der höheren Nummer aus. Nach dem letzten drückt man „*ok“* links unten und man sieht die neue Fläche. Wenn man etwas falsch geklickt hat, kann man in dem grünen Fenster unten links „*unpick*“ anklicken, dann zeigt der Pfeil nach unten und man kann das falsche Klicken ungeschehen machen. „*Unpick“* erreicht man auch durch Drücken der rechten Maustaste. Auch eine falsch erzeugte Fläche kann man mit „*delete*“ und Anklicken wieder löschen. Wenn es dabei Schwierigkeiten gibt, muß man die Flächen mit ihren Num-

mern darstellen (*PlotCtrl, numbering* ...) und dann genau auf die Nummer der Fläche klicken, die man vernichten will.

In derselben Weise werden die beiden kleinen Flächen angeben, die die Magnetspulen darstellen sollen: „*Create" „Keypoints*":

0.5,-0.135,0
0.55,-0.135,0
0.5,-0.055,0
0.55,-0.055

0.5,0.33,0
0.5,0.25,0
0.55,0.25,0
0.55,0.33,0

Man sollte jetzt zwei kleine Flächen innerhalb des oben erzeugten Rechengebietes und außerhalb des **InGrid**-Gebiets sehen.

Es ist fast unvermeidlich, daß man an irgendeiner Stelle während der bis jetzt beschriebenen Prozedur entdeckt, daß man einen Fehler gemacht hat. Wenn der Fehler beim Erzeugen des **ANSYS**-Modells passiert ist, muß man die falsch erzeugten Flächen oder Linien oder Keypoints mit „*delete*" entfernen. Dabei meldet **ANSYS** häufig, daß das zu entfernende Element Teil eines anderen sei und nicht entfernt werden kann. Man muß sich dann vorsichtig weitertasten, um genau das Element zu finden, daß entfernt werden kann, ohne zuviel von dem schon erzeugten Modell zu zerstören.

Schlimmer ist, wenn man feststellt, daß ein Fehler im **InGrid** - Gitter bzw. in dessen Umrandung ist. Um nicht ganz von vorne anfangen zu müssen, geht man zurück nach AutoCAD und ruft zunächst den Umriß auf. Zum Üben bringt man darin die gewünschte Korrektur an. Man zeichnet am besten die neuen gewünschten Polylinien und schneidet dann mit „*modify, trim*" die falschen Linien weg. Dann übt man auch gleich, die neuen Linien mit den alten zu einer einzigen Poyline wieder zu verbinden mit „*modify, edit polyline, join*". Wenn man das im Griff hat, kann man das fertige Gitter aufrufen (das schon die Zahlen-Attribute hat) und dieselben Schritte wiederholen. Danach müssen die Gitterpunkte auf die neuen Linien verschoben werden, also wie oben, die Eckpunkte auf die neuen Ecken, dann mit „*äquidistant verteilen*" die Zwischenpunkte auf die neuen Linien verlegen, dann mit „*relaxieren"* oder „*glätten*" alle Punkte mit dem Attribut 0 an die richtigen Stellen zaubern. Dabei kann sich das Gitter so verzerren, daß man vielleicht noch weitere Reparaturen vornehmen muß. Dies alles ist sehr zeitraubend, weil das Gitter schon so dicht ist. Aber es geht schneller als wenn man vom Umriß ausgehend das Gitter noch einmal ganz neu machen muß. Genauso geht man vor, wenn im Verlauf der Rechnungen der Wunsch entsteht, kleine Geometrieänderungen vorzunehmen.

Danach wiederholt sich das Exportieren und das Kopieren von `head` und `kopf`. Da sich ja an der Logik des Gitters durch die Verschiebung nichts geändert hat, kann man den alten `head` ohne Änderung wieder verwenden.

Nach dieser Zwischenbemerkung folgt nun eines der tieferen Geheimnisse von **ANSYS**: Unter „*Boolean operations*“ muß man jetzt „*overlap*“ „*areas*“ verlangen und die vier vorhandenen areas anklicken: Das **InGrid**-Gebiet, das Rechengebiet und die beiden Spulen. (Es geht auch einfacher mit „*pick all*“). Danach sieht man die Flächen in neuen Farben und macht unter allen Umständen zunächst einmal „*save*“ in dem blauen Tool-Kasten rechts oben.

An dieser Stelle sei ein weiteres **ANSYS**-Geheimnis verraten: In der blauen Befehlsliste oben kann man „*List*“ anklicken und bekommt eine Auswahl. Verlangt man hier „*logfile*“, bekommt man in einem blauen Fenster einen Text zu sehen, der in der **ANSYS**-Kommando-Sprache alles wiedergibt, was man bis jetzt gemacht hat. Es ist zu empfehlen, dieses Kauderwelsch verstehen zu lernen (eventuell mit Hilfe von „*help*“ - ganz rechts oben im blauen Befehlsfeld), denn man kann später daraus oft ersehen, was man falsch gemacht hat. Auch kann man den logfile unter einem anderen Namen kopieren und diese Kopie dann editieren, sie z. B. von Irrwegen und falschen Eingaben befreien und später in einem anderen Fall als Eingabe verwenden („*read input* from...“). Das spart manchmal viel Tipp-Arbeit.

In diesem logfile stehen natürlich auch die Kommentare, die, beginnend mit einem ! geschrieben wurden. Man sollte jetzt einen neuen schreiben, nämlich:

!**********Beginn des Vernetzens**********

Vorher müssen den einzelnen Flächen „*Attributes*“ zugeteilt werden. Das ist in diesem Fall einfach: Die beiden Spulen erhalten das Material 2, alle anderen Flächen sind aus Material 1 gefertigt, nämlich Vakuum. Diese beiden Materialien sind bereits ganz oben im `kopf` definiert worden. Wenn man noch andere Materialien hat, muß man im *Preprocessor* „Material *constants*“, dann „*isotropic*“ aufrufen. Dann bekommt das neue Material erst eine Nummer, dann folgt eine lange Liste mit Materialeigenschaften. Hier endlich sollte der Leser einmal das Hilfeprogram aufzurufen, und zwar das *„Element manual*“, darin das Element plane 13 suchen (das im kopf ausgewählt wurde), und dort zu finden, daß man in diesem Falle nur unter *MURX* und *RSVX* etwas eintragen muß. Von jetzt an sollen die Schritte nicht mehr so detailliert beschrieben werden, weil anzunehmen ist, daß der Leser inzwischen schon einige Grundbegriffe der Bedienung von **ANSYS** gelernt hat.

Es folgt jetzt die Wahl der Maschengröße. Gebiete, wo das Magnetfeld wichtig erscheint, erhalten kleinere Maschen als andere, wo im Magnetfeld nichts passiert. Es ist jetzt gut, „*plot*“ *„lines*“ zu verlangen. Man sieht dann die durchgezogenen Begrenzungslinien der eben erzeugten Flächen. Dazu viele Linien am Rande des **InGrid**-Modells. Mit „*Shape and size*“ bekommt man eine Möglichkeit, die Linien anzuklicken und ihnen entweder eine Elementgröße, oder eine Anzahl der Unterteilungen zuzuordnen. Erfahrungsgemäß ist das Letztere einfacher. In diesem Menü gibt es auch eine Möglichkeit, die Maschengröße kontinuierlich nach einer Richtung hin wachsen oder schrumpfen zu lassen. Das muß jeder selbst ausprobieren. Nach „*ok*“ sieht man die unterteilte Linie. Wenn die Teilung noch nicht richtig ist, kann man einfach dieselben Schritte noch einmal mit anderen Angaben machen.

Wenn alle Linien unterteilt sind, kann man „*mesh*“ und „*areas*“ verlangen. Man klickt immer zuerst die kleineren Flächen an, weil das schnell geht, und zuletzt das Rechengebiet. Wenn alles gut geht, hat man dann ein Maschennetz über das ganze

Gebiet, das sich an das **InGrid**-Gitter exakt anschließt. Man schaut es sich an mit „*plot*“ „*Elements*“. (Vorher unter *pltctrls, numbering, elements only colors* einstellen!) Es kommt häufig vor, daß das Vernetzen nicht glatt geht. Aus den Fehlermeldungen lernt man so gut wie nichts. Man muß sich vor allem die Linienunterteilungen noch einmal genau ansehen. Große Unterschiede in der Dimensionierung - also etwa µm in einer und mm in einer anderen Richtung können viel Kopfzerbrechen verursachen.

Es ist jetzt noch ein Schritt nötig, der sehr viel Programmier-Zeit gekostet hat. Verlangt man nämlich jetzt einmal unter „*Plot Ctrl*“ „*numbering*“ die Knoten (*node*)-Nummern und macht *plot nodes*, dann kann man, eventuell unter Zuhilfenahme von *Zoom* erkennen, daß an der Grenze zwischen dem **InGrid** -Modell und dem **ANSYS**-Rechengebiet die Knotennummern seltsam schlecht zu lesen sind. Das liegt daran, daß hier aus unerfindlichen Gründen überall zwei nodes übereinander liegen. Das führt zu Verwicklungen bei der späteren Rechnung. Deshalb muß man unter „*numbering controls* (im *Preprocessor*)

„*merge items*“ und „*nodes*“ verlangen.

Dann sieht man die doppelten nodes verschwinden und man kann auch an dieser Grenze die Knoten-Nummern gut lesen. Jetzt kann eine **ANSYS**-Rechnung begonnen werden; man macht „*save*“ und verläßt den Preprocessor, indem man nun auf „*Solution*“ klickt.

Hier werden wir einiges gefragt, das meist leicht zu beantworten ist (es gibt für alles default-Werte, wenn man nicht weiß, was man wählen soll). Wichtig ist nur, daß es eine neue Rechnung werden soll, und zwar eine statische. Die anderen Berechnungsarten (harmonisch und transient) sind in den Diplomarbeiten von Häfner [21] und Baus [22] recht gut dargestellt, was deshalb hier nicht mehr wiederholt werden muß.

Für die Rechnung müssen jetzt einige Randbedingungen eingegeben werden. Das geschieht mit „*Apply*“ „*Potential*“ „*on nodes*“. Man muß die nodes auf dem Bildschrim haben und klickt jetzt in mühsamer Arbeit alle Knoten auf dem Rand (außer auf der senkrechten Rotationsachse links) an. Nur reifere **ANSYS**-Benutzer können mit dem Befehl „*Select*“ (oben fast links im blauen Feld) umgehen. Wenn diese Rand-Knoten alle gelb unterlegt sind, sagt man „*ok*“ und wird gefragt, welches Vektorpotential dieser Rand haben soll: Natürlich gibt man hier *0.0* ein.

Ebenfalls für später sei hier verraten, daß es sogenannte „Infinite Elemente“ gibt, mit denen man den Rand umgeben kann, und die dann der Rechnung einen ins Unendliche gehenden Rand vorzaubern. Der Umgang mit diesen Elementen ist schwierig und im jetzigen Stadium ist es im Zweifelsfalle immer noch einfacher, das Rechengebiet zu vergrößern. Das Thema „Elemente“ verdiente eigentlich ein eigenes Kapitel. Da man aber bei den vorliegenden Rechnungen fast immer mit dem Element „*plane13*“ auskommt, erscheint dies überflüssig. In den Elementen und ihrer richtigen Wahl steckt ein Großteil der Wirkungsweise von **ANSYS**. Es ist nämlich die Physik, die man berechnen will, einfach in den Element-Typ gesteckt worden, und alles andere ist dann im Programm identisch. Das heißt, ob Magnetfelder, Temperaturfelder, mechanische Spannungen oder vieles andere mehr berechnet werden soll, es kommt nur auf die Wahl des Elements an, alles andere wird identisch gehandhabt. Im „*elements manual*“ unter „*help*“, kann man das alles lernen. So gibt es z. B. auch

ein Element „*plane53*“, mit dem man Magnetfelder rechnen kann. Das ist für das Zusammenspiel mit **InGrid** ungeeignet, weil jedes Element 8 Knotenpunkte hat und somit nicht an die **InGrid-**Zellen paßt. Für eigenständige Magnetfeldrechnungen ist *plane53* genauer als *plane13.*

Die Mittelachse bekommt die Randbedingung „*Flux parallel - on lines*“. Man muß dabei nicht nur die lines anklicken, die man selbst erzeugt hat, sondern auch die vielen kleinen, die von **InGrid** stammen.

Zuletzt muß man noch unter „*Apply*“ „*current density*“ den Elementen der beiden Spulen eine Stromdichte zuordnen. Dazu gibt es zwei Möglichkeiten. Entweder man überlegt sich an dieser Stelle, welche Stromdichte man hier wirklich haben will und welche Zahlen in A/m^2 realistisch sind. Dann bekommt man die Felder in MKS-Einheiten richtig ausgerechnet. Oder man gibt als Stromdichte einfach 1.0 an und erhält winzig kleine Zahlen für die Magnetfelder. Der PIC-Code erlaubt die Eingabe eines Faktors, mit dem die eingelesenen Magnetfelder multipliziert werden. Da der PIC-Code die Felder so betrachtet, als seien sie mit kA erzeugt worden, muß man wahnsinnig aufpassen, daß man sich bei dieser ganzen Angelegenheit nicht um Faktoren 1000 irrt. Es wird dringend empfohlen, diesen verschlungenen Pfaden, die in Jahren historisch gewachsen sind, sorgfältig nachzugehen. Es ist auch bei den gerechneten Beispielen nicht einheitlich.

Damit sind die Randbedingungen eingegeben und man kann es wagen auf „*solve*“ zu klicken. Da wird man noch einiges gefragt, was man einfach wegklickt, und dann dauert es einige Minuten, bis man die Meldung „Done“ oder etwas Unerfreulicheres erhält. Im zweiten Fall geht ein großes Rätselraten an, das hier nicht wiedergegeben werden kann, im ersten verläßt man das Menü „Solution“, indem man den „*General Postprocessor*“ aufruft.

Hier wiederholt man nun die Schritte, die oben beschrieben wurden und erhält hoffentlich ähnliche Resultate.

2. Erzeugung der BFELD-Datei für den PIC-Code

Es bleibt jetzt nur noch, das Macro „BAUS“ aufzurufen. Das geschieht aber im Gegensatz zu „*ROTATION*“ von vorhin mit dem Befehlspunkt „*Macro, execute*“ oben in der blauen Zeile etwas weiter rechts. Man muß nämlich als Parameter die Dimensionen des **InGrid**-Gitters angeben, die man am Anfang der Datei `reo3.kdi` finden kann. (Man kann sich bekanntlich mit „*List - files - others*“ jede Datei auf den Bildschirm holen, um etwas darin nachzusehen). Als Parameter 1 ist die Zahl der InGrid-Gitterpunkte in z-Richtung, als Parameter 2 die Zahl in r-Richtung anzugeben.

Wenn BAUS richtig gelaufen ist, verläßt man **ANSYS** unter Speicherung von allem und stellt fest, daß man eine Datei `reo3.bfeld` erzeugt hat. Die wird für den PIC-Code benötigt.

IV. Simulation

1. Eingabedateien

Die Simulation findet auf der inrrisc6 statt (früher auf der inrrisc5). Es gibt verschiedene Varianten des PIC-Codes, die alle etwas verschieden zu handhaben sind. Es soll hier mit der einfachsten begonnen werden, nämlich einer Simulation einer B_Θ - Diode mit dem Code **bfcpic2d**, der kein externes Magnetfeld benötigt. Man wechselt in das Verzeichnis der Diode, die berechnet werden soll, also z. B. `bth96g1` und braucht zunächst nur die beiden Dateien `grid` und `input`. `Grid` ist das oben z. B. `bth96g1.kdi` genannte Gitter, das man sich mit **FTP** hierhin geholt hat und ihm dabei den einfachen Namen `grid` gegeben hat. (Die Eindeutigkeit entsteht dadurch, daß man sich jetzt im Verzeichnis `bth96g1` befindet, und alles, was in diesem Verzeichnis steht, sich auf diese Diode bezieht.)

Die Datei „`input`“ muß jetzt im Einzelnen besprochen werden. Sie sieht etwa so aus:

```
&NAMEL1 IFILEG=21,IFILEB=00,IFILEI=36,LOGD=08,  LOGP=07        /
&NAMEL2                                                        /
&NAMEL3                                                        /
&NAMEL4 DT=1.90E-12,NGEW=4500,NDT=2000,NDTE=16,NNDTE=5          /
&NAMEL5 NPSORT=2,NPMAXN=01,QMIN=1.E-10,ITIMX=240,IBILD=0       /
&NAMEL6 PANT2=1.,PANT3=00.,PANT4=00.,PANT5=00.,PANT6=00.       /
&NAMEL7
        BOUND(-3)=1.E20,
        BOUND(-2)=1.E7,
        BOUND(-1)=1.E7,
        BOUND(01)=0.0,
        BOUND(02)=1.E20                                      /
  -3   'RPN2'   0.0                      /
  -2   'RPDI'   0.0                      /
  -1   'RPDI'   0.0                      /
   0   'FELD'                            /
   1   'RPDI'   1.3E6                    /
```

Genauso war nämlich die Fortran-Eingabe bei der alten MVS-Maschine. Nach all den Umwandlungen und Umorganisationen wird auf den modernen Maschinen nun dieses alte Kauderwelsch immer noch verstanden.

&NAMEL1, die erste Zeile interessiert nur an der Stelle IFILEI = . Steht dort 36, handelt es sich um einen Fortsetzungslauf, der weiter unten besprochen wird. Da aber zunächst von vorne angefangen wird, muß die 36 durch 00 ersetzt werden.

`LOGD` wird nach dem Rechenlauf eine Datei heißen, in der auf zwei Seiten einige grundsätzliche Angaben zur Rechnung und zum Ergebnis stehen. `LOGP` wird nach der Rechnung die Datei sein, in der alle berechneten Zahlen stehen, und die man später mit **geisha** ansehen wird. Später, d. h. bevor ein neuer PIC-Lauf gestartet wird, tauft man diese Dateien um, z. B. in `LOGD1` und `LOGP1`, usw, das trägt bei vielen Rechnungen der gleichen Diode sehr zur Ordnung bei.

In der vierten Zeile stehen Angaben für die PIC-Rechnung: DT ist die Länge des Zeitschrittes in Sekunden. NGEW ist eine Angabe, die wir ganz schnell 0000 setzen, sie wird später erklärt. NDT ist die Zahl der Zeitschritte, die man rechnen will. Die

Angabe 2000 ist für einen Anfangslauf bei dieser Diode gut. Sie muß so gewählt werden, daß innerhalb der Zeit dieser Anzahl von Zeitschritten die Ionen den Spalt zwischen Kathode und Anode durchqueren können. NDTE und NNDTE kontrollieren die Unterteilung der Ionenzeitschritte in kürzere Abschnitte für die Elektronen. Normalerweise steht da immer 4 und 5.

Die fünfte Zeile enthält nichts Aufregendes, außer der geschätzten Rechenzeit ITIMX. Wenn diese Zeit (in Minuten) fast abgelaufen ist, wird der Rechenlauf vom Programm abgebrochen. Dabei kann passieren, daß nicht alle Ergebnisse, die bis zu dieser Zeit berechnet worden sind, in der Datei `LOGP` gelandet sind. Es ist besser, eine solche Situation zu vermeiden und ITIMX groß genug zu wählen, weil **geisha** sonst Schwierigkeiten machen kann.

In der 6. Zeile steht gar nichts interessantes, dafür ist &NAMEL7 äußerst wichtig:

Die Zahlen in den Klammern bei BOUND(...) beziehen sich auf die **Elektrodenzellen**, denen ja im `head` der Gitterdatei verschiedene Bezeichnungen gegeben wurden. Es zeigt sich jetzt, daß Elektrodenzellen der Sorte -3 eine Schwelle für die Emission von Teilchen von 1.E20 hat. Das heißt, da eine so hohe Feldstärke nirgends auftritt, emittiert diese Elektrodenfläche keine Teilchen.

Dagegen emittiert die Fläche mit den Elektrodenzellen der Sorte -2 ab einer Feldstärke von 1.E7 [V/m] Teilchen. Dasselbe tut auch die Elektrode der Sorte -1. Die Anode, deren emittierenden Zellen die Bezeichnung 01 tragen, emittiert sogar bei der Feldstärke 0. (Eine Besonderheit: Der Platz in diesen Klammern muß ganz ausgefüllt sein, also (0) geht nicht).

Nun folgen noch Zahlenangaben, die sich auf die **Attribute** von Gitterpunkten beziehen. Hier ist für die Punkte mit dem Attribut -3 durch die Angabe ‘RPN2‘ 0.0 eine Neumann-Randbedingung in die Koordinatenrichtung 2, also Potentiallinien in Richtung der y-Achse gefordert. Entsprechend heißt ‘RPDI‘, daß für die Punkte mit den Attributen -1 und -2 eine Dirichlet-Randbedingung mit dem Potentialwert 0.0 oder beim Attribut 1 mit dem Potential 1.3E6 [V/m] verlangt wird. Hier verlaufen die Potentiallinen etwa parallel zu den Elektroden. Das Attribut 0 gehört zu einem ‘FELD‘ -Punkt, also zu einem inneren Punkt des Berechnungsgebietes.

2. Rechnung der ersten 2000 Zeitschritte.

Wenn diese Angaben alle richtig sind, ruft man den PIC-Code mit dem Befehl

kadi

auf. Es erscheint ein Fenster, in dem man *bfcpic2d* anklickt und danach „*Ausführen*“. Im nächsten Fenster werden die beschriebenen Dateien angezeigt, die für Ein- und Ausgabe eine Rolle spielen. Noch nicht erwähnt wurde die Datei „`output`“, die am Ende des Rechenlaufs vielfältige Auskünfte über den Lauf enthält. Früher hat man diese Datei nach dem Lauf auf einem für Kindergärten und Schulen sehr brauchbaren Papier ausgedruckt bekommen. Heute schaut man sie sich nur noch am Bildschirm an, wenn man anders nicht weiterkommt. Wenn man „Prüfe Filenamen“ drückt, erhält man entweder eine Nachricht, daß noch etwas fehlt, oder der Rechenlauf beginnt.

Wenn man die Nachricht erhalten hat, daß die Rechnung begonnen hat, kann man sich aus **kadi** verabschieden. Es gibt zwei UNIX-Befehle, mit denen man den Fortgang der Rechnung beobachten kann:

top

bringt eine Liste, in der die laufenden Jobs angezeigt werden. Wenn man seinen eigenen ("bfcpic") darin nicht findet, ist er schon fertig und man muß in `logd` oder `output` nach einem Hinweis suchen, warum er so schnell herausgeflogen ist. Wenn man ihn dagegen findet, wird angezeigt, wieviele Prozent des Rechners ihm zugewiesen worden sind und wieviel CPU-Zeit schon verflossen ist. Man kriegt dann schnell ein Gefühl dafür, ob man heimgehen kann, oder nur schnell einen Kaffee trinken soll. Aus dieser Liste kommt man mit *ctrl-c* wieder heraus. Der zweite Befehl:

tail logp

bringt das Ende des files `logp` auf den Bildschirm, der gerade eben geschrieben wird. Man kann hier an der ersten Zahl, die geschrieben wird, sehen, wie viele Zeitschritte schon gerechnet sind. (*tail -20 -f logp* bringt kontinuierlich die letzten 20 Zeilen).

Wenn der Lauf ordnungsgemäß zu Ende gegangen ist, kann man sich das Ergebnis mit **geisha** ansehen. Dazu muß man leider `logp` mit Hilfe von **FTP** in ein Verzeichnis auf die Maschine inrrisc8 verschieben. Deshalb sind auf der inrrisc8 im Verzeichnis *bfcpic* alle Unterverzeichnisse, die den Namen einer Diode tragen, noch einmal angelegt. Wenn die Übertragung gelungen ist, verabschiedet man sich von der inrrisc6 und meldet sich bei der inrrisc8 an, geht in das entsprechende Verzeichnis und verlangt:

geisha

Das wurde oben bei der Gittererzeugung schon einmal benutzt. Man wiederholt die erwähnten Schritte, muß aber jetzt eben auch die Konvertierung von `logp` verlangen. Wenn das gelungen ist, erscheint **Tecplot** auf dem Bildschirm und fragt, was man sehen will.

Es gibt nun eine Fülle von Auswertungen, die man sich am besten anhand eines schon gerechneten Beispiels anschaut. **Geisha** und **Tecplot** sind so weitgehend selbsterklärend, daß hier auf eine detaillierte Beschreibung verzichtet werden kann.

Nun wurden erst 2000 Zeitschritte gerechnet. Der PIC-Code verlangt, daß dies zunächst so geschehen muß. Für die nächsten - etwa 4000 - Zeitschritte muß das Ergebnis des Anfangslaufs eingelesen werden. Der PIC-Code weiß dann, daß es sich um einen Fortsetzungslauf handelt und beginnt eine Mittelung der Ergebnisse, die in [1] näher erläutert ist. Für den Benutzer sind folgende Schritte nötig:

3. Endgültige Rechnung.

Es muß jetzt die Datei „`input`" ein bißchen redigiert werden. Bei IFILEI muß jetzt 36 geschrieben werden. Bei NDT ändert man die 2000 in 4000. Vielleicht muß man bei

ITIMX eine höhere Zahl von Minuten eingeben (Erfahrung, wie lange die 2000 Zeitschritte gedauert haben). Die vorhin erhaltene Datei `logp` muß umgetauft werden in `logpi .` Dann kann man wieder *kadi* aufrufen und alles wiederholen, was oben beschrieben wurde. Ob man eine genügende Anzahl von Zeitschritten gewählt hat, zeigt das Bild, das man in **Tecplot** unter „*Teilchen*“ erhält: Wenn die Kurven der Teilchenzahl und der Ladung nicht mehr stark schwanken, ist genügend lange gerechnet worden. Man sieht es auch an den Potentiallinien: Wenn sie glatt und ohne Ecken sind, kann man der Rechnung vertrauen. Gewinnt man hier den Eindruck, daß noch mehr Zeitschritte nötig wären, geht man so vor: Das eben entstandene `logp` wird in `logpi` umgetauft. IFILEI bleibt =36, **und bei NGEW muß die Anzahl der beim vorherigen Fortsetzungslauf gerechneten Zeitschritte, also im beschriebenen Fall 4000 eingesetzt werden.** Das hängt wieder mit der oben erwähnten Mittelung der Ströme zusammen.

4. Varianten.

Den folgenden Abschnitt kann der Anfänger überspringen. Er muß hier nur der Vollständigkeit wegen angefügt werden. Bei **kadi** wird man an einer Stelle gefragt, ob man mit der Standardversion rechnen will, oder mit einer eigenen. Klickt man die eigene Version an, dann muß vorher folgendes passiert sein:

Es gibt in dem Verzeichnis `/home18/inrrisc6/bauer` ein Verzeichnis `BFCPIC`, unter dem die Verzeichnisse `BFCPIC2D` und `BFCPIC2H` versteckt sind. In dem ersteren finden sich einige Dateien, die mit dem suffix .f bezeichnet sind. Das sind Fortran-Dateien des PIC-Codes. Wenn man das Bedürfnis hat, in dem zu benutzenden PIC-Code etwas zu ändern, kann man das hier tun. (Vorsicht!! Man sollte unbedingt alle solche Änderungen sorgfältig mit Kommentaren kennzeichnen!) Das geänderte Programm muß durch den Befehl

translate

neu übersetzt werden. Das dauert einige Minuten. Danach kann man **kadi** dazu bringen, mit dieser Version des Programmes zu rechnen. In der letzten Zeit wurde immer mit dieser Version gerechnet, weil darin die Rechengenauigkeit erhöht und die Abfrage bei dem Potentialberechnungsprogram SOR verfeinert wurde.

5. Rechnung mit Magnetfeld.

Hier muß nun neben den bereits erläuterten Dateien auch noch die Datei „`diode.bfeld`“ vorhanden sein, die als letzter Schritt von **ANSYS** mit dem Makro BAUS erzeugt wurde. Sie wird wieder mit **FTP** zur inrrisc6 in das richtige Verzeichnis kopiert und dabei in „`bfeld`“ umgetauft.

Die input-Datei sieht für diesen Fall etwas anders aus:

```
&NAMEL1 IFILEG=20,IFILEB=21,IFILEI=10,LOGD=08,LOGP=09          /
&NAMEL2 DELTAZ=0.0,F1=1.E-3                                    /
&NAMEL3 /
&NAMEL4 DT=5.E-12,NGEW=0000,NDT=4000,NDTE=4,NNDTE=5            /
&NAMEL5 NPSORT=2,NPMAXN=1,ITIMX=240                            /
&NAMEL6 /
```

```
 &NAMEL7 BOUND(+3)=1.E20,
         BOUND(+2)=1.E20,
         BOUND(+1)=0.,
         BOUND(-1)=1.E07,
         BOUND(-2)=1.E07,
         BOUND(-3)=1.E20                                        /
ELEKTRISCHE REFERENZLISTE
5
    4   'RPN2'   0.0                        /
    3   'RPN2'   0.0                        /
    2   'RPDI'   0.0                        /
    1   'RPDI'   1.7E6                      /
    0   'FELD'                              /
MAGNETISCHE REFERENZLISTE
5
    4   'RPN2'   0.0                        /
    3   'RPN2'   0.0                        /
    2   'RPDI'   0.0                        /
    1   'RPDI'   0.0                        /
    0   'FELD'                              /
```

In der ersten Zeile steht bei IFILEG = 20, bei IFILEB =21, und der Fortsetzungslauf wird mit IFILEI = 10 angefordert.

In der zweiten Zeile gibt es zwei historische Zahlen, die man angeben muß, aber eigentlich nicht mehr braucht. DELTAZ setzt man immer gleich 0. Man könnte hier eine Versetzung des geometrischen 0-Punktes zwischen der **ANSYS**-Rechnung und der PIC-Rechnung angeben. Für F1 ist hier 1.E-3 angegeben. Das ist richtig, wenn die **ANSYS**-Rechnung in MKS-Einheiten gemacht wurde, also die Stromdichte in A/m^2 eingegeben wurde. Da der PIC-Code noch an Kiloampere denkt, müssen hier die B-Feld-Werte durch 1000 dividiert werden. Dieser Faktor bewährt sich, wenn man eine Rechnung mit variablem Magnetfeld machen will. Man muß dann nicht immer wieder **ANSYS** laufen lassen.

Die übrigen Angaben sind etwa so wie oben beschrieben, wenn man darauf achtet, daß die Elektrodenzellen und die Punktattribute in diesem Gitter aus historischen Gründen andere Bezeichnungen tragen.

Es kommt aber noch ein „Magnetische Referenzliste“ hinzu. Es funktioniert, wenn man die elektrische Referenzliste einfach abschreibt und die angelegte Spannung, oben = 1.3E6 hier =0.0 setzt.

Nun könnte die Rechnung genauso verlaufen, wie oben angegeben. Leider ist diese Rechnung aber noch nicht in der Benutzeroberfläche **kadi** integriert, sodaß man an dieser Stelle einen etwas rudimentären Weg gehen muß.

Es gibt in den Verzeichnissen, die zu einer Diode mit externem Magnetfeld gehören, immer mindestens eine Datei, die entweder `run` oder `run1` oder so ähnlich heißt. Die sieht so (oder so ähnlich) aus:

```
home5/inrrisc5/bauer/BFCPIC/BFCPIC2H//bfcPIC2h 0<input
1>druckausgabe grid bfeld DUMMY logd logp
```

Das ist alles in einer Zeile hintereinander und es wird dringend empfohlen, vor der Anwendung, zum Systemadministrator zu gehen. Die Angaben im ersten Teil haben sich wahrscheinlich geändert und müssen auf neuesten Stand gebracht werden. Die

Angaben nach der Stelle 0< erkennt der Leser wahrscheinlich wieder: `input` ist die eben beschriebene Datei `input`, `druckausgabe` ist das, was oben `output` genannt wurde, `grid`, `bfeld`, `logd` und `logp` sind wie oben. `DUMMY` steht da, wenn in `input` IFILEI=0 steht, also ein Anfangslauf. Anstelle von `DUMMY` schreibt man `logpi`, wenn in `input` IFILEI=10 steht.

Wenn man das alles richtig gemacht hat und alle Dateien vorhanden sind, ruft man den PIC-Lauf mit dem Befehl

run &

auf und wartet wie oben beschrieben auf das Ergebnis.

V. Ungelöste Probleme

Wenn alle Dateien richtig sind und alles beachtet wurde, was oben beschrieben ist, sollte der PIC-Code ein Ergebnis liefern. Wenn er schon nach weniger als einer Minute fertig ist, muß man den `output` anschauen und sehen, wie weit er gekommen ist. Ist er schon beim Einlesen des Gitters gestolpert, muß man im Gitter nach einem Fehler suchen. Das kommt relativ selten vor, denn wenn man das Gitter mit **geisha** richtig ansehen konnte und dort optisch nichts Schlechtes entdecken konnte, müßte das Gitter in Ordnung sein.

Bei einem neuen Gitter kommt es vor, daß irgendwelche Dimensionierungsangaben im Hauptprogramm nicht ausreichend sind. Das wird in `output` ziemlich deutlich gesagt und man muß dann die Änderungen durchführen, wie sie oben unter „4. Varianten“ erläutert wurden.

Ist der Code bei der Potentialberechnung hängen geblieben, dann stimmt oft etwas nicht mit den Randbedingungen. So wurde z. B. einmal erst nach sehr zeitraubender Fehlersuche entdeckt, daß bei einer Neumann-Randbedingung in Richtung 1, das heißt in einer nach der Seite hin offenen Struktur (Potentiallinien waagerecht) die Gitterzellen unbedingt streng rechteckig sein müssen.

Wenn der Code eine angemessene Zeit lang gelaufen und ordentlich zu Ende gekommen ist (oft mehrere Stunden), sieht man das am `output` daran, daß als letztes eine Tabelle über die verbrauchten cpu-Zeiten gedruckt wird. Dann schiebt man `logp` mit Hilfe von **FTP** zur inrrisc8 und schaut sich das Ergebnis mit **geisha** an. Hier gewinnt man mit der Zeit Erfahrung, ob das Ergebnis plausibel ist, oder grobe Fehler enthält. Meist kann man sich das Potential aufgrund der Geometrie vorstellen. Weicht das Ergebnis weit von dieser Vorstellung ab, muß man nachdenken.

Seit etwa zwei Jahren wurde z. B. die Frage untersucht, warum im Potential oft Stellen vorkommen, wo die Potentiallinien zu negativen Potentialwerten gehören und ***in sich geschlossen*** sind. (Fig. 3). Diese Graphik wurde von **Tecplot** erstellt und als Postscript-Datei exportiert. Die Weiterverarbeitung funktioniert genauso wie oben bei Fig. 2. beschrieben. Solche Beulen widersprechen der physikalischen Vorstellung; leider konnten sie bis heute nicht erklärt oder beseitigt werden. T. Westermann und der Verfasser haben alles versucht, was diesen Effekt beeinflussen könnte, und im-

mer nur Teilerfolge gehabt. Je nach Einzelheiten der Einstellung tritt in der Potentialberechnung nach einer bestimmten Anzahl von Zeitschritten zum ersten Mal ein negativer Wert auf, der sich dann bei weiteren Zeitschritten zu einem ganzen negativen Gebiet auswächst. Es sieht aber so aus, als ob dieses erste Auftreten eines negativen Potentialwertes nicht das erste Auftreten des Problems sei. Denn schon lange vorher, wenn noch alles positiv ist, beginnt das Potential eine eigenartige Beule zu bekommen, in der dann erst später der negative Wert erscheint.

a) Z. B. wurde ein neues Gitter gemacht. BTH98A1 (21 x 213) Es war zwar nicht so schön wie das alte (70 x 90), weil keine Dummy-Zellen außen sind. Es schmiegte sich aber dem Rand besser an und ähnelte ein bißchen den erwarteten Potential-Linien. Die Beule war davon nicht zu beeindrucken.

b) Die Spannung wird bei **bfcpic** nicht schlagartig, sondern in einer Rampe von 1500 Zeitschritten angelegt. Eine Erhöhung dieser Rampe auf 3000 Zeitschritte zeigte nur eine Andeutung von Erfolg.

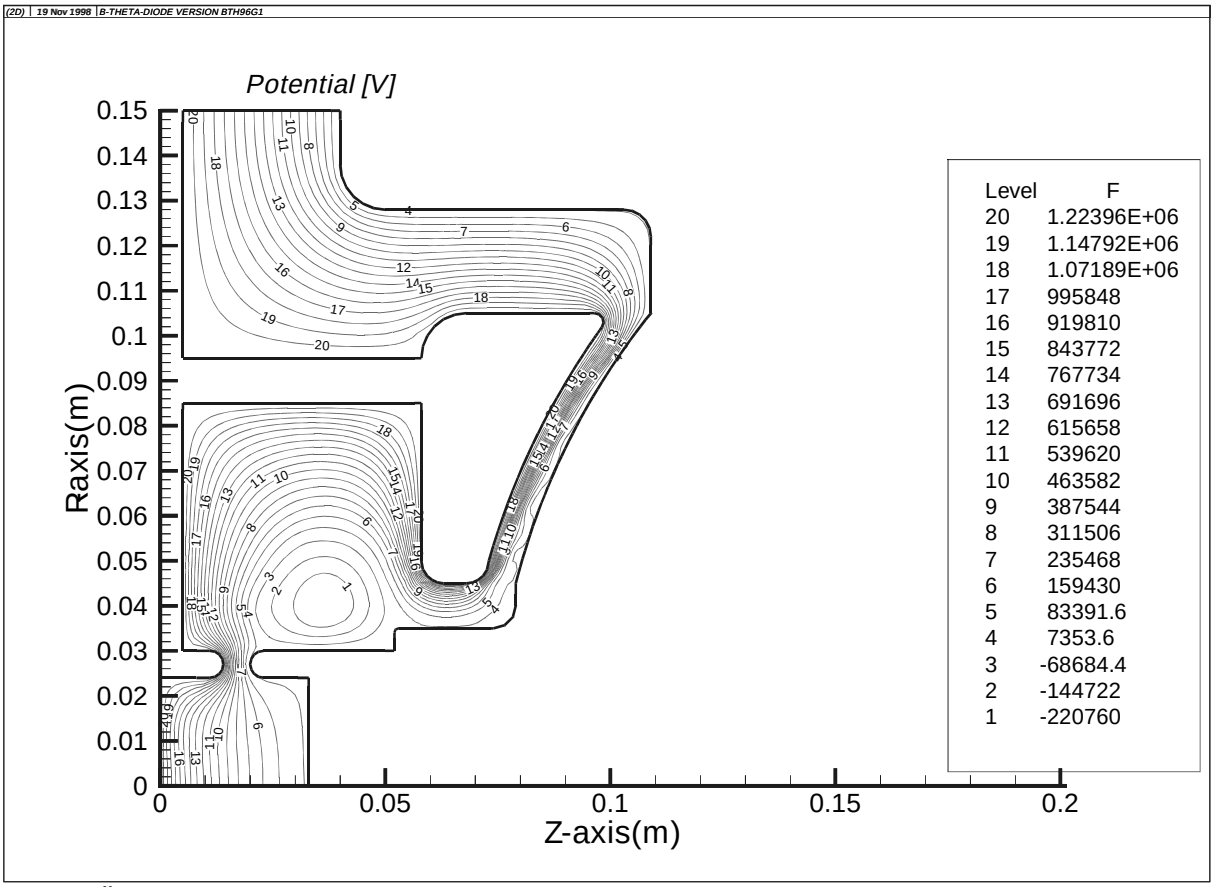

Fig. 3: Äquipotentiallinien, in der modifizierten B_Θ-Diode, von **bfcpic2d** berechnet. Das Bild zeigt die im Text beschriebene, nicht mit der physikalischen Vorstellung übereinstimmende „Beule“. Außerdem illustriert die Zeichnung die weiter unten beschriebene Modifikation der B_Θ-Diode, bei der das isolierende B_Θ-Feld durch eine Elektronendiode (links unten) erzeugt werden soll, während die Ionen sich wie gewohnt im Beschleunigungsspalt (rechts) bewegen

c) Weiterhin wurde das ganze Programm mit doppelter Genauigkeit übersetzt und im Unterprogramm **SOR** mit hoher Genauigkeit gerechnet: $8e^{-8}$. Mit Hilfe einer Abfrage nach negativem F wurde der erste negative Wert bei Zeitschritt 560 gefunden. Da diese Zahl vorher um 1015 lag, war zuerst an einen starken gitterab-

hängigen Effekt zu denken. Aber eine Wiederholung mit dem alten Gitter BTH96G1 (70 x 90) ergab 581. Also mußte es an dem Spannungsanstieg liegen. Mit dem altem Spannungsanstieg kam dann auch mit dem neuen Gitter der erste negative Wert bei Zeitschritt 1013. Die Adresse des negativen Wertes ist jeweils genau da, wo auch die Beule im Potential auftritt.

d) Man kann daraus schließen, daß die Form des Gitters für dieses Problem nicht so wichtig ist. Also wurde ein doppelt so feines Gitter gemacht: BTH98B1 (41 x 425) und weiterhin damit gerechnet. Es geht sehr langsam, aber nach drei Stunden 55 Minuten waren auch hier die 1050 Zeitschritte erreicht. Erstaunliches Ergebnis: Wieder sehr früh, sogar schon beim 489. Zeitschritt, kommt der erste negative Wert, wieder genau in der Ecke, in der die Beule schon vorher entstanden war.

e) Da das Gitter 41 x 425 zuerst große Probleme wegen verschiedener Dimensionierungs-Angaben gemacht hat, wurde inzwischen eine Gewaltkur versucht. Genau nach der Stelle, wo die Abfrage auf negatives F und der Druckbefehl steht, wurde versuchsweise F = 0. gesetzt. Das hat schön gerechnet und das Ergebnis war verblüffend: Es sind keine negativen Potentialwerte mehr da, aber die Beule ist geblieben wie immer! In der Mitte der Beule ist einfach ein weißes Loch und keine Potentiallinien. Aber außen, wo die Potentiallinien positiv werden, haben sie die gute alte Beulenform. Das bestätigt eine frühere Beobachtung, daß die Beule in den Potentiallinien schon vor dem ersten negativen Wert auftritt. Was hier besonders schwer zu verstehen ist, daß der Ausdruck „erstes Auftreten eines negativen Potentialwertes“, der ja vor dem Nullsetzen kommt, erst bei Zeitschritt 660 auftritt, während er ohne die Nullsetzung – bei Zeitschritt 560 auftrat. Natürlich wurde danach schnellstens die F = 0. – Setzung wieder entfernt.

f) Da das mehr rechteckige Gitter ein kleines bißchen besser erschien, wurde ein neues, ganz ähnliches mit **InGrid** gemacht (die früheren Gitter waren noch nach der alten Methode erzeugt worden) BTH98C1 (49 x 101), das - wieder ohne deutliches Ergebnis schrittweise verfeinert wurde.

An dieser Stelle müßte also ein späterer Benutzer des Codes mit neuen Ideen weiter fortfahren. Eine weitere Rechnung, die nicht zu Ende geführt werden konnte, weil eigentlich vorher das Problem der Beule im Potential zu lösen wäre, ist die Berechnung der von P. Hoppé vorgeschlagenen modifizierten B_Θ-Diode. Es wird hier versucht, [10], die beiden Dioden, die hier eine Rolle spielen, nämlich die das Magnetfeld erzeugende Elektronendiode und die eigentliche Ionendiode räumlich voneinander zu trennen. Die Ergebnisse lassen sich in folgender Weise zusammenfassen, unter der Voraussetzung, daß die Beule keinen allzu großen Einfluß auf das generelle Ergebnis hat:

a) Bei konstantem Spaltabstand an der Elektronendiode verhalten sich die Gesamtströme I_e, I_i und I_{ges} wie gewohnt. Es müßte allerdings diskutiert werden, was bis heute ungeklärt ist, warum der PIC-Code nur Wirkungsgrade von etwa 20% ergibt, während bei allen Messungen eher 50% erreicht werden.

b) Das maximale B_Θ-Feld tritt an der unteren Kante der Ionen-emittierenden Anode auf und hat etwa denselben Verlauf wie der Gesamtstrom. Nach der Überlegung, die zu dieser Form der B_Θ-Diode geführt haben, sollte das B_Θ-Feld aber von der Elektronendiode bestimmt werden. **Das ist nicht der Fall!**

c) bei konstantem Spaltabstand an der Ionendiode (4.5 mm, entsprechend etwa 2 Ohm) lassen sich die Gesamtströme kaum beeinflussen von einer Variation des Spaltabstands der Elektronendiode.

d) Die Ströme an der Elektronendiode - allein betrachtet - zeigen einen Verlauf, der dem Child-Langmuir-Gesetz recht genau entspricht. Allerdings wird B_Θ-(max) weiterhin von der Ionendiode dominiert und liegt am unteren Rand der Ionen-Anode. **Erst beim Spaltabstand g = 2 mm (`BTH96J1`) springt B_Θ-(max) auf den Rand der Anode der Elektronendiode und steigt stark an.** Trotzdem ist noch kein Einfluß auf den Gesamtstrom oder den Ionenstrom zu bemerken.

e) Eine Erhöhung der Zahl der Zeitschritte und eine Verfeinerung des Gitters im Bereich der Elektronendiode ergeben kleine zahlenmäßige Änderungen (10%), aber keine prinzipiellen Unterschiede im Verhalten.

f) **Schlußfolgerung:** Die Elektronendiode verhält sich nicht so, wie erwartet. Sie spielt keine dominierende Rolle bei der Erzeugung von B_Θ. Erst bei extrem kleinem Spalt (2 mm) der Elektronendiode verlagert sich die Stelle des höchsten B_Θ-Feldes an die Anode der Elektronendiode und nimmt einen wesentlich höheren Wert an. Allerdings hat auch dies noch keinen entscheidenden Einfluß auf die Gesamtströme. Man könnte deshalb versuchen, die Wirkung der Elektronendiode noch weiter zu verstärken, indem man ihre Elektrodenflächen vergrößert.

Entscheidendes Ergebnis: Eine Erhöhung von B_Θ sollte die Elektronen im Ionenspalt besser isolieren. Das heißt, die Elektronenschicht entfernt sich von der Anode, der effektive Spalt wird größer und der Ionenstrom sinkt. Es muß also nicht einfach B_Θ vergrößert werden, sondern ein Optimum gesucht werden, bei dem der (Gesamt)-Elektronenstrom am kleinsten und der Ionenstrom am größten wird.

In den nachfolgenden Tabellen soll versucht werden, eine Zusammenfassung der erhaltenen Zahlen zu geben.

Danksagung

Ich habe in der ganzen Zeit, in der ich mich mit den beschriebenen Programmen beschäftigt habe, ganz ausgezeichnet mit Herrn Prof. Dr. Thomas Westermann zusammengearbeitet, zuerst als er noch Kollege am Forschungszentrum Karlsruhe war, und auch später, nachdem er zur Fachhochschule Karlsruhe gewechselt war. Für die freundschaftliche und kompetente Art, in der er mir immer wesentliche Ratschläge und Tips gegeben und die Arbeit weitergeführt hat, danke ich ihm sehr herzlich.

Herr Eckardt Stein war eine unverzichtbare Hilfe bei allen Fragen des Computer-Systems. Ohne ihn wäre die Umstellung der Programme auf moderne Rechner überhaupt nicht möglich gewesen. Ich danke ihm vor allem für die stets bereitwillige und unverzügliche Unterstützung, die er mir in der täglichen Arbeit bei allen erdenklichen Problemen gegeben hat.

Darüber hinaus bedanke ich mich bei den vielen Kollegen, die mir für eine gewisse Zeit oder bei bestimmten Problemen geholfen haben: C. Broeders, E. Halter, W. Höbel, D. Rusch, B. Schrempp, und bei denen, deren Interesse an den Ergebnissen die Arbeit vorangetrieben hat: H.J. Bluhm, P. Hoppé, G. Keßler.

Tabelle I Überblick über die gerechneten Fälle.

Membername	gap-Ionendiode [mm]	gap-Elektronendiode [mm]	I_{el} [kA]	I_{ion} [kA]	I_{tot} [kA]	Impedanz [Ω]	B_{max} [Vsm^{-2}]	η(%)
BT94A [1]	8.5	-	468.7	143.7	612.3	2.173	3.1	23.2
BT94D [2]	8.5	-	536.3	145.5	681.9	1.907	2.6	21.5
BTH96A1 [3]	8.5	5	339.0	62.1	401.1	3.241	1.4	13.8
BTH96B1	3.5	5	617.1	204.1	821.2	1.583	2.9	25.0
BTH96C1	6.5	5	437.1	92.1	529.3	2.456	1.9	15.9
BTH96D1	5.5	5	438.7	107.6	546.3	2.380	2.0	18.8
BTH96E1	4.5	5	533	138	671	1.938	2.41	19.8
BTH96F1	7.5	5	372	68.3	440.4	2.959	1.565	14.0
BTH96G1	4.5	6	480	136.4	616.4	2.109	2.21	21.7
BTH96G2 [4]	4.5	6	487.2	136.9	624.1	2.083	2.25	21.5
BTH96H1	4.5	4	514.3	139.9	654.1	1.987	2.3	20.7
BTH96H2 [5]	4.5	4	477.8	133.8	611.6	2.126	2.2	19.0
BTH96H3 [6]	4.5	4	467.8	124.82	592.6	2.194	2.15	21.1
BTH96H4 [7]	4.5	4	465.1	135.2	600.3	2.166	2.14	22.5
BTH96H5 [8]	4.5	4	487.7	139.3	627.0	2.073	2.2	21.7
BTH96I1	4.5	3	490.7	125.4	616.1	2.11	2.1	19.9
BTH96J1	4.5	2	508.9	109.7	618.6	2.102	2.9	17.9
BTH96J2 [8]								

Anmerkungen:

[1] Alte B_Θ-Diode mit Feldemissionskante

[2] Alte B_Θ-Diode ohne Feldemissionskante

[3] Neue B_Θ-Diode, mit separater Elektronendiode

[4] 8000 Zeitschritte statt 6000

[5] feineres Gitter im Bereich der Elektronendiode (73x98 statt 70x90)

[6] noch feineres Gitter: 73x117

[7] noch feineres Gitter: 80x117

[8] 9000 Zeitschritte statt 6000

Tabelle II. Abhängigkeit einzelner Teilströme von der gap-Weite der Elektronendiode

gap der El.-Diode [mm]	Child-Langm. [kA]	von der Kathode der Elektronen-Diode [kA]	auf die Anode der Elektronendiode [kA]	auf große Anode [kA]	$B_{\Theta\text{-max}}$ [Vsm^{-2}]
2	410	271	252	150	2.9
2 [1]	410	251	234	166	2.7
3	182	163	145	209	2.1
4	102	104	109	267	2.3
5	65	73	87	321	2.4
6	45	60	84	288	2.2

Tabelle III. Weitere Teilströme als Funktion der gap-Weite der Elektronendiode

gap der El.Diode [mm]	vanes unten [kA]	vanes oben [kA]	Strom von vanes [kA]	Anode unten [kA]	Anode oben [kA]	Strom auf Anode vorne [kA]	„Vakuumstrom“ [kA]
2	273.2	618.6	345.4	333.5	445.9	112.4	233
2 [1]	253.9	624.0	370.1	278.0	444.4	166.4	203
3	163.3	616.1	452.7	279.0	411.1	132.0	320
4	107.6	627.0	519.4	269.5	418.4	148.9	370
5	72.3	670.9	598.6	279.3	442.0	162.7	436
6	61.7	624.1	562.4	257.6	413.5	155.9	406

[1] 9000 Zeitschritte anstelle der sonst immer 6000.

Literatur

[1] T. Westermann, „A Particle-in-Cell Method as a Tool for Diode simulations“, Nuclear Instr. Meth. **A 263** (1988), p. 271

[2] W. Bauer, unveröffentlichter Bericht, Kernforschungszentrum Karlsruhe 14.04.01P58H, 1989

[3] W. Bauer, E. Stein, T. Westermann, unveröffentlichter Bericht, Kernforschungszentrum Karlsruhe 14.04.01P70B, 1989

[4] W. Bauer, H. Bachmann, H. Bluhm, P. Hoppé, H. U. Karow, D. Rusch, Ch. Schultheiss, E.. Stein, O. Stoltz, T. Westermann „Computations and Experiments on the „Small B_Ω-Diode“, Proc. 8th Int. Conf. High-Power Particle BEAMS, Novosibirsk, (1990), 457-462

[5] W. Bauer, H. Bachmann, H. Bluhm, L. Buth, P. Hoppé, H. U. Karow, H. Lotz, D. Rusch, Ch. Schultheiss, E.. Stein, O. Stoltz, T. Westermann, „New Results from Experimental and Numerical Investigations of the Self-Magnetically B_Ω-insulated Ion Diode“, Proc. 9th Int. Conf. High-Power Particle Beams, Washington, DC, (1992), 735-740

[6] T. Westermann, W. Bauer, „Treatment of edge beams in a focusing self-magnetically B_Ω-insulated Ion Diode“,Laser and Particle Beams **10 no. 3.**, 539-574 (1992)

[7] W. Bauer, W. Höbel, A. Ludmirsky, E. Stein, T. Westermann, „A Contribution to the magnetic focusing in an Applied-B-Extractor Ion Diode by a Laser Pulse Driven Solenoid“, Proc. 10th Int. Conf. High-Power Particle Beams, San Diego, (1994), 83-86

[8] W. Bauer, W. Höbel, A. Ludmirsky, E. Stein, T. Westermann, „A Contribution to the magnetic focusing in an Applied-B-Extractor Ion Diode“, Forschungszentrum Karlsruhe, FZKA 5840, (1995), 58-62

[9] W. Bauer, R. Häfner, „Magnetic Field Calculations“ Forschungszentrum Karlsruhe, FZKA 5840, (1996), 65-68

[10] W. Bauer, P. Hoppé, H. Bachmann, H. Bluhm, L. Buth, H. Massier, D. Rusch, E. Stein, O. Stoltz, W. Väth, T. Westermann, „Recent Results from Experimental and Numerical Investigations of the self-magnetically B_Ω-insulated Ion Diode“, Proc. 11th Int. Conf. High-Power Particle Beams, Prag, (1996), 1123-1126

[11] W. Bauer, E. Stein, B. Schrempp, T. Westermann, „Stationary Particle-inCell Simulations on Electron and Ion Diodes“, Proc. 12th Int. Conf. High-Power Particle Beams, Haifa, (1998)

[12] C.-D. Munz, P. Omnes, R. Schneider, E. Sonnendrücker, E. Stein, U. Voss, T. Westermann, “Diode Simulations with KADI2D” Proc. 12th Int. Conf. High-Power Particle Beams, Haifa, (1998)

[13] **ANSYS® ANSYS**, Inc. 201 Johnson Road,, Houston, PA, 15342-1300 von Fa. CADFEM, Grafing

[14] AutoCAD® Autodesc Inc. von Fa. Mann, Datentechnik

[15] E. Halter, B, Schrempp, Fachhochschule Karlsruhe, private Mitteilung

[16] E. Halter, B. Schrempp, E. Stein, private Mitteilung („Einführung in die Benutzung der Unix-Installation des interaktiven Programms InGrid zur Erzeugung von randangepaßten Rechengittern"), 1997

[17] E. Halter, unveröffentlichter Bericht, Kernforschungszentrum Karlsruhe, 14.04.01P36F, 1985

[18] D. Grether, M. Sararu, B. Haas, „LIDIS, ein Programm zur numerischen Simulation gepulster Leichtionendioden, Kernforschungszentrum Karlsruhe KfK 4578, 1989

[19] S. Kleinheinz, „Das VISART-Konzept einer standardisierten Schnittstelle zwischen Codes und Auswerteprogrammen", Forschungszentrum Karlsruhe FZKA 5995 and 5596, 1997,

[20] Tecplot® Amtec Engineering, Inc. Bellevue, Washington

[21] R. Häfner, „Finite Elemente-Rechnung von Magnetfeldstrukturen in technischen Anwendungen", Diplomarbeit Fachhochschule Karlsruhe und Forschungszentrum Karlsruhe (1995)

[22] M. Baus, „Zeitabhängige Analyse von Magnetfeldstrukturen in Hochstrom-Ionendioden mit Hilfe der Finite-Element-Methode", Diplomarbeit Fachhochschule Karlsruhe und Forschungszentrum Karlsruhe (1998)

Anhang: Rezepte für den Umgang mit Unix, X-Terminal, etc..

A. Allgemeine UNIX-Befehle:

mkdir erzeugt ein neues Verzeichnis - mkdir name

rmdir löscht ein leeres Verzeichnis

cd Name des Directorys: in ein anderes Directory wechseln

cp zum Kopieren *cp* `datei1 datei2`

rm zum Löschen *rm* `datei`, *rm* `da*` etc.

cd .. eine Stufe zurück im Verzeichnisbaum

cd ganz zurück in das eigene home-Verzeichnis

ls ist ähnlich wie dir bei DOS, es wird eine Liste der in dem aktuellen Verzeichnis vorhandenen `files` angezeigt.

ls -a auch „hidden files“ werden angezeigt - das sind solche mit einem Punkt davor

ls -l es werden auch die Zugriffsberechtigungen und das Datum angezeigt.

lpg gibt es nur auf der inrrisc5 und zeigt auch mehr von den Dateien an

pg wird zum Anschauen eines `files` benutzt. Man kommt mit *ctr-c* wieder aus einem solchen File heraus. Weiterblättern geht mit *Enter*. Besser geeignet sind *spfux* oder *tspful*, weil man damit nach gewohnter Manier suchen und hin und her springen kann.

more macht dasselbe, seitenweise. Weiterblättern mit *enter* um 1 Zeile, mit *space* um 1 Seite

spfux **`filename`** ruft einen wohlbekannten Editor auf, um `filename` zu editieren.

tspful ohne `filename` ist etwas komfortabler, ansonsten wie *spfux*
hier muß im zweiten panel unter „UNIX FILENAMES“ %PWD% stehen

pwd zeigt das aktuelle Verzeichnis an, in dem man sich gerade befindet

vi filename Dies ist ein besonders trickreicher Editor für files in der unix-Welt. Man kann ihn ohne spezielle Hilfstabellen kaum benutzen, Wenn man fertig ist: **esc w q** dann wird die geänderte Version gespeichert. (oder **ZZ**)

man vi Damit ruft man Kapitel über den Befehl vi im manual auf. Mit ctrl-C kann man das manual wieder verlassen.

Wenn man etwas mit der linken Maustaste markiert, ist es sozusagen in der Maus gespeichert, und kann mit der rechten Maustaste dorthin übertragen werden, wo man das Einfügezeichen hingestellt hat.

B. Befehle zum Übertragen von Dateien von einem Rechner zu einem anderen:

In einem Verzeichnis, auf dem Rechner inrrisc6:

ftp inrrisc8

Dann wird nach Namen und Passwort auf der inrrisc8 gefragt:

Der Befehl zum Holen von der Datei `bap98a1` im Verzeichnis `bap98` heißt:

get /fzk/inr/home/bauer/bap98/bap98a1 bap98a1
Der Name des files auf der inrrisc8 so soll er bei der inrrisc6 heißen

Der Befehl zum Hinschicken von Dateien heißt entsprechend:

put bap98a1 /home18/inrrisc6/bauer/bap98a/bap98a1
die Datai, die weggeschickt werden soll der Verzeichnisbaum bis zum Namen der Zieldatei

Wenn alles herübergeholt oder hinüber geschafft ist, was benötigt wird:

bye damit verläßt man den Befehl *ftp* wieder**,** die Verbindung zur anderen Maschine ist gelöst und das alte prompt ist wieder da.

C. Befehle zum Ausdrucken

Auf den Rechnern inrrisc5 und inrrisc6 und inrrisc8 gibt es ein Programm, das man mit

sprint

aufruft. Es erscheint dann ein Menü, das die verschiedenen möglichen Drucker und Plotter anbietet. Die Geräte befinden sich an verschiedenen Stellen, hier eine Liste der in der vorliegenden Arbeit am meisten benutzten:

Für ASCII - Texte:

LJ4_630, der Laserdrucker im Bau 630

hdi4hd1, doppelseitiger Druck beim HIK

Für Postscript-files:

djPCL630, der Farbplotter im Bau 630

hdi4ps, doppelseitiger Druck beim HIK

Es gibt noch viele andere, neuere Drucker.

XV ist ein Graphik-Programm in unix, mit dem man aus postscript dateien .gif dateien machen kann, die word97 einlesen kann. Wird aber nicht sehr schön.

D. Sonstiges

Wenn man aus einem Anwendungsprogramm irregulär ausgestiegen ist, kann es sein, daß noch Prozesse ewig weiterlaufen. Die muß man killen. Das geht so:

ps -ef | grep name *des Benutzers*

In der dann erscheinenden Anzeige ist die erste Nummer jeweils die Nummer des angezeigten Prozesses. Prozesse, aus deren Namen und Uhrzeit man schließen kann, daß sie zu dem Unglücksausstieg gehören, werden so gekillt:

kill -9 Nummer

Danach muß noch aus dem Verzeichnis bin ab geschrieben werden:

CleanIPC.tcl name *des Benutzers*

Wenn man wissen will, wieviele Megabyte man belegt hat:

du -ks *

top zeigt laufende jobs an

tail logp zeigt das Ende der Datei `logp` an, die gerade geschrieben wird

tail -20 -f logp zeigt die letzten 20 Zeilen der Datei `logp` fortlaufend an

fts lsquota zeigt den Speicherplatz, den man belegt hat, und wieviel dem Benutzer zusteht (nur auf der inrrisc8)

Wenn sich herausstellt, daß man fast 100% des einem zustehenden Speicherplatzes verbraucht hat, muß man entweder Dateien löschen (Vorsicht!!) oder sie in den Massenspeicher schieben.

dsm2 ruft den Massenspeicher ins Leben, mit dem man nicht so oft gebrauchte Dateien auslagern und wieder holen kann. Braucht einige Übung und Geduld.

adsmCall_new ist der modernste Massenspeicher, der von der inrrisc8 und ähnlichen Rechnern aus angesteuert werden kann. Man muß im ersten Fenster „archive“ anklicken und im nächsten ... network ..., dann erscheinen die eigenen Verzeichnisse und man kann entweder ein ganzes Verzeichnis oder einzelne Dateien archivieren. Ebenso verläuft das Zurückholen mit „retrieve“.

E. Einloggen bei der Universität Karlsruhe und Rechnen mit ANSYS53 dort:

Man muß auf der inrrisc8 sein und schreibt:

telnet hiksp01 die letzte 1 kann auch 2, 3, oder 4 heißen

kommt: sp2login: dahinter schreibt man seinen Namen, dann wird man nach dem Paßwort gefragt, man gibt das Paßwort von der inrrisc8 an: dann kommt:

hiksp01 $ dahinter schreibt man: ***splogin*** dann muß man nochmal das gleiche Paßwort angeben.

Damit bin ist man im Rechner der Uni und stellt fest, daß die eigenen Verzeichnisse von der inrrisc8 dort auch sind.

Wenn jetzt „standard input“ dasteht, muß man zweimal die Leertaste drücken. Dann:

geht man in das richtige Verzeichnis und schreibt

export DISPLAY=inrxterm28.fzk.de:0
hier natürlich den Namen des eigenen Terminals
xANSYS53 &

Diese Prozedur geht immer noch, es ist aber inzwischen eine Vereinfachung eingetreten: Es genügt, von der inrrisc8 aus ***splogin*** zu schreiben. Ansonsten geht es wie oben beschrieben weiter.

Übrigens:
Das exortieren des DISPLAY muß immer gemacht, wenn man sich auf einem anderen Rechner mit *rlogin* einloggen will! Wenn man dabei innerhalb eines Instituts INR bleibt, genügt:

export DISPLAY=inrxterm28:0.0