

Simulation geladener Teilchen auf parallel geclusterten Rechnern

Diplomarbeit

angefertigt am Institut Theorie Elektromagnetischer Felder
der Technischen Universität Darmstadt

von

Felix Wolfheimer, geb. am 28.04.1978

Betreuer: Dr. rer. nat. Erion Gjonaj

Inhaltsverzeichnis

1 Aufbau und Ziele der Arbeit	7
1.1 Aufbau der Arbeit	7
1.2 Ziele der Arbeit	8
2 Physikalische Grundlagen	9
2.1 Einleitung	9
2.2 Kinetische Plasmatheorie	10
3 Der PIC Algorithmus	17
3.1 Einleitung	17
3.2 Zeitliche Diskretisierung der Newton-Lorenz Gleichung	19
3.3 Räumliche Diskretisierung der Newton-Lorenz Gleichung	21
4 Parallelisierung	33
4.1 Einleitung	33
4.2 Wichtige Begriffe	34
4.3 Nachrichtenorientierte Parallelisierung mit MPI	36
4.4 Effizienz einer Parallelisierung	38

INHALTSVERZEICHNIS

5 Parallelisierung des PIC Algorithmus	40
5.1 Einige Ansätze aus der Literatur	40
5.2 Verteilung des Gesamtproblems auf die Prozessoren	43
5.3 Datenstruktur zum Speichern der Teilchen	48
5.4 Interpretation der Gebietsteilung für die Rechengitter	49
5.5 Organisation der Kommunikation	50
5.6 Adaptives Verfahren zur Lastverteilung	53
5.7 Flussdiagramm für den parallelisierten PIC Algorithmus	54
6 Simulationsbeispiele	56
6.1 Ionenstrahl im konstanten Magnetfeld	56
6.2 Simulation einer Elektronenkanone	57
6.3 Simulationsergebnisse	59
7 Zusammenfassung und Ausblick	69
A Herleitung der Formeln des Leap-Frog Algorithmus	72

Abbildungsverzeichnis

3.1	Leap-Frog Algorithmus	20
3.2	Zur Numerischen Lösung der Poisson-Gleichung	24
3.3	Allokierung der Ladung bei PIC	27
3.4	Materialbehandlung in PIC	29
3.5	Interpolation der Felder bei PIC	31
3.6	Flussdiagramm zum PIC-Algorithmus	32
4.1	Möglichkeiten der Speicherorganisation	36
5.1	Möglichkeit zur Parallelisierung von PIC	41
5.2	Übersicht über Parallelisierungsverfahren für PIC	44
5.3	Die Familie der Orthogonalen Bisektionierungen (aus [25])	44
5.4	Binärbaum zur Gebietsaufteilung	46
5.5	Flussdiagramm zum Erstellen des Baumes	47
5.6	Details zur Gebietsaufteilung	50
5.7	Zuordnung der Geisterzellen	52
5.8	Flussdiagramm für den parallelisierten PIC Algorithmus	55
6.1	Phasenraumbilder des Ionenstrahls	58

ABBILDUNGSVERZEICHNIS

6.2	Elektronenstrahl	60
6.3	Skalierbarkeit des Pushers	62
6.4	Skalierbarkeit des Solvers	63
6.5	Skalierbarkeit der gesamten Zeitschleife	64
6.6	Vergleich zwischen transversaler und longitudinaler Gebietsteilung 1 .	66
6.7	Vergleich zwischen transversaler und longitudinaler Gebietsteilung 2 .	67
6.8	Laufzeitvergleich bei großen Simulationen	68

Kapitel 1

Aufbau und Ziele der Arbeit

1.1 Aufbau der Arbeit

In Kapitel 2 sollen zunächst einige Grundlagen aus der theoretischen Physik erläutert werden, soweit sie zum Verständnis der Arbeit erforderlich sind. Zudem soll dieser Abschnitt helfen, die durchgeführten Simulationen in ihren physikalischen Kontext einzubetten. Kapitel 3 stellt den PIC-Algorithmus vor, der für die Simulationen genutzt wurde. Bei der Darstellung wurde insbesondere darauf Wert gelegt, zu zeigen, wie die Gleichungen aus der kontinuierlichen Physik diskretisiert werden, um sie in einen Algorithmus übersetzen zu können, was die Voraussetzung ist, um eine Simulation an einem Rechner durchführen zu können. Kapitel 4 stellt einige wichtige Begriffe zur Parallelisierung bereit, die zum Verständnis der folgenden Abschnitte hilfreich sind. Dabei wird auch die Kommunikationsbibliothek MPICH kurz vorgestellt und einige wichtige Routinen dieser Bibliothek, die bei der Programmierung Verwendung fanden, kurz erläutert. Kapitel 5 zeigt nach einem kurzen Überblick über Parallelisierungsverfahren für PIC, die in der Literatur gefunden werden können, wie der PIC-Algorithmus im Rahmen dieser Arbeit parallelisiert wurde. Dabei wird auch auf einige interessante Implementierungsdetails eingegangen. In Kapitel 6 werden die Modelle vorgestellt, die zum Test der Software benutzt wurden, wobei es sich zum einen um ein Modell einer Elektronenkanone handelt und zum zweiten um das Modell eines Ionenstrahles in einem konstanten Magnetfeld. Außerdem werden die Simulationsergebnisse präsentiert, die in den Simulationen erhalten werden konnten.

In Kapitel 7 folgt schließlich eine kurze Zusammenfassung der Arbeit. Ein Ausblick, in dem aufgezeigt wird, wie die Software weiter entwickelt werden könnte, schließt die Arbeit ab.

1.2 Ziele der Arbeit

Das Ziel dieser Arbeit war es zum einen, eine Software zu entwickeln, die in der Lage ist, Probleme zu simulieren, die so groß sind, dass sie auf einem einzelnen Rechner unmöglich gelöst werden können. Dazu musste ein Verfahren gefunden werden, wie das Gesamtproblem parallelisiert werden kann, so dass die sich ergebenden kleineren Teilprobleme auf einzelnen Rechnern lösen lassen. Ein weiterer wichtiger Punkt in diesem Zusammenhang ist die Frage nach dem Zeitgewinn durch eine Parallelisierung. Da die Laufzeit eines parallelen Programmes immer von demjenigen Prozessor bestimmt wird, der für die Ausführung am längsten braucht, ist es wichtig, eine gute Lastbalancierung für die Prozessoren anzustreben. Das bedeutet, dass die Teilprobleme, die den einzelnen Prozessoren zugeordnet werden, möglichst den gleichen Rechenaufwand im Sinne gleicher Laufzeiten haben sollten. An zwei Problemen wird exemplarisch untersucht, wie sich die einzelnen Teile des Programmes skalieren lassen und ob die Parallelisierung im Sinne einer Verminderung der Gesamtlaufzeit des Programmes attraktiv ist. Dabei wird auch untersucht, wie sich unterschiedliche Partitionierungen des Rechengebietes auf die Laufzeit des Programmes auswirken.

Kapitel 2

Physikalische Grundlagen

In diesem Kapitel sollen die physikalischen Grundlagen vorgestellt werden, auf denen diese Arbeit basiert. Dabei wird die kinetische Plasmatheorie vorgestellt und die durchgeführte Simulation in ihren physikalischen Kontext eingebettet.

2.1 Einleitung

Der größte Teil der Materie im uns bekannten Universum befindet sich in einem hochionisierten Zustand. Das bedeutet, dass Elektronen nicht mehr an Atomkerne gebunden sind, sondern sich frei bewegen können. Materie, die sich in einem solchen Zustand befindet, bezeichnet man auch als **Plasma**. Die Materie, aus der sich Sterne zusammensetzen, sowie weite Bereiche der interstellaren Materie befinden sich in diesem Zustand [9]. Sogar in Erdnähe, in der so genannten **Ionosphäre**, befinden sich die Atome der Atmosphäre in diesem Zustand. Dies wird technisch im Rahmen des terrestrischen Funkverkehrs gezielt ausgenutzt. So lassen sich durch Reflektionen an der Ionosphäre Radiosignale über sehr weite Strecken übertragen.

Im Rahmen der Plasma-Physik wird das Verhalten von Materie, die sich im oben beschriebenen Zustand befindet, mathematisch modelliert und experimentell untersucht. Die grundsätzlichen Fragen und Probleme, die sich bei der Modellierung eines Plasmas¹ ergeben, können hier nicht in aller Ausführlichkeit diskutiert werden.

¹Gemeint ist hier das Aufstellen eines analytischen, mathematischen Modells, das das Verhalten eines Plasmas beschreibt

Beispielsweise ist schon die Frage, ob die Teilchen als Punktladungen oder als Ladungen mit räumlicher Ausdehnung modelliert werden sollen, nicht trivial. In [10] findet sich eine analytische Darstellung mehrerer Modelle, die unter anderem auch diesen Aspekt diskutiert und die mathematischen Konsequenzen der verschiedenen Modellierungsansätze beleuchtet. Es soll daher nur eine kurze Darstellung der kinetischen Plasmatheorie erfolgen, soweit sie hilfreich ist, die Simulationen zu verstehen und in ihren physikalischen Kontext einzuordnen. Die Darstellung des folgenden Abschnitts basiert großteils auf [20].

2.2 Kinetische Plasmatheorie

Wie bereits angedeutet handelt es sich bei einem Plasma um ein ionisiertes Gas, also um frei bewegliche Elektronen und Ionen. Die Bewegung dieser geladenen Teilchen unter dem Einfluss von äußerer elektrischen und magnetischen Feldern kann mathematisch modelliert werden. Es existieren dazu im wesentlichen zwei verschiedene Ansätze. Der erste beschreibt das Plasma mit Hilfe von Methoden der Fluidodynamik. Allerdings versagt dieser Ansatz unter extremen Bedingungen (z. B. bei der Wechselwirkung eines Plasmas mit einem hochenergetischen Laserstrahl [6]). Der Grund dafür liegt darin, dass die fluidynamischen Gleichungen nur dann gültig sind, falls sich das Plasma zumindest lokal in einem thermodynamischen Gleichgewicht befindet und die Geschwindigkeitsverteilung der Teilchen eine Maxwell-Boltzmann-Verteilung ist.

Die zweite Möglichkeit zur Modellierung eines Plasmas betrachtet direkt die einzelnen Teilchen aus denen sich das Plasma zusammensetzt. Das bedeutet, dass das Plasma nicht als Kontinuum sondern als eine Anhäufung sehr vieler Teilchen beschrieben wird, die sich unter dem Einfluss äußerer elektrischer und magnetischer Felder sowie unter dem Einfluss der Coulomb'schen Kräfte, die die Teilchen aufeinander ausüben, bewegen. Dieser Ansatz wird auch als kinetische Plasmatheorie bezeichnet. Aufgrund der Vielzahl der betrachteten Teilchen bietet sich eine statistische Modellierung an. Betrachtet man ein System von N_b gleichartigen Teilchen², so

²Zu Anfang dieses Abschnitts wurde ein Plasma als ein System bestehend aus verschiedenartigen geladenen Teilchen definiert (Ionen und Elektronen). Bei der Herleitung der folgenden Gleichungen wird nur von Teilchen einer Art ausgegangen. Die Erweiterung auf ein Mehrkomponentenplasma

lässt sich eine Verteilungsfunktion F_{N_b} definieren, die von den beiden Eigenschaften Ort \mathbf{r} und Impuls \mathbf{p} der Teilchen, sowie von der Zeit t abhängig ist. Es folgt:

$$F_{N_b} = F_{N_b}(\mathbf{r}_1, \dots, \mathbf{r}_{N_b}, \mathbf{p}_1, \dots, \mathbf{p}_{N_b}, t) \quad (2.1)$$

Die Werte für Ort und Impuls der einzelnen Teilchen werden durch tiefgestellte Indizes kenntlich gemacht. Der Definitionsbereich der $\mathbf{r}_1, \dots, \mathbf{r}_{N_b}$ und $\mathbf{p}_1, \dots, \mathbf{p}_{N_b}$ wird als Phasenraum $\Omega \subseteq \mathbb{R}^{6 \cdot N_b}$ des Systems bezeichnet. Der Phasenraum setzt sich zusammen aus den Definitionsbereichen Ω_i für die einzelnen Teilchen und lässt sich daher schreiben als $\Omega := \Omega_1 \times \dots \times \Omega_{N_b}$. Berücksichtigt man, dass die Orts- und Impulsvektoren im allgemeinen Fall eines dreidimensionalen Systems drei Komponenten besitzen, so hat die Funktion F_{N_b} $6 \cdot N_b$ unabhängige Veränderliche. Betrachtet man eine Teilmenge des Phasenraumes $G_\Omega \subseteq \Omega$, so gibt

$$\int_{G_\Omega} F_{N_b} dG_\Omega \quad (2.2)$$

die Wahrscheinlichkeit dafür an, dass sich zum Zeitpunkt t die Teilchen in dem durch G_Ω beschriebenen Teil des Phasenraumes befinden. Ist das $6 \cdot N_b$ -dimensionale Volumen des Phasenraumes zeitlich konstant, so handelt es sich um ein **konservatives** System und es gilt die Liouvillegleichung. Diese besagt, dass die zeitliche Ableitung der Funktion F_{N_b} verschwindet. Es gilt mit der Kettenregel der Differenzialrechnung:

$$\frac{\partial F_{N_b}}{\partial t} + \sum_{i=1}^{N_b} \left(\dot{\mathbf{r}}_i \cdot \frac{\partial}{\partial \mathbf{r}_i} F_{N_b} + \dot{\mathbf{p}}_i \cdot \frac{\partial}{\partial \mathbf{p}_i} F_{N_b} \right) = 0 \quad (\text{Liouville-Gleichung}) \quad (2.3)$$

Aufgrund der sehr großen Zahl von Unbekannten in physikalisch interessanten Systemen ist eine direkte Lösung der Liouville-Gleichung ausgeschlossen. Man interessiert sich daher mehr für Verteilungsfunktionen niedrigerer Ordnung, die zudem die einzelnen Teilchen nicht voneinander unterscheiden, sondern sie als ununterscheidbar ist jedoch anschließend einfach möglich und wird am Ende des Abschnitts gezeigt.

modellieren. Insbesondere ist die Einteilchenverteilungsfunktion F_1 von besonderem Interesse, wobei $F_1 \cdot d\Omega_1$ die Wahrscheinlichkeit dafür angibt, ein bestimmtes Teilchen im Phasenraumvolumen $d\Omega_1$ anzutreffen, sowie die Funktion f_1 , wobei $f_1 \cdot d\Omega_1$ die Wahrscheinlichkeit dafür angibt, ein beliebiges Teilchen im Phasenraumvolumen $d\Omega_1$ anzutreffen (siehe Abschnitt 2.2.1). Eine Verteilungsfunktion niedrigerer Ordnung lässt sich aus F_{N_b} durch Integration über die Ω_i der nicht näher betrachteten Teilchen gewinnen. Betrachtet man daher nur die ersten s Teilchen im Detail, so ergibt sich die Verteilungsfunktion F_s als

$$F_s(\mathbf{r}_1, \dots, \mathbf{r}_s, \mathbf{p}_1, \dots, \mathbf{p}_s, t) = \int_{\Omega_{s+1} \times \dots \times \Omega_{N_b}} F_{N_b} d(\Omega_{s+1} \times \dots \times \Omega_{N_b}) \quad (2.4)$$

Die Frage, die sich nun stellt, ist, wie sich Bestimmungsgleichungen für die Verteilungsfunktionen niedrigerer Ordnung aus der Liouville-Gleichung (2.3) herleiten lassen. Diese Frage wird durch die Hierarchiegleichungen beantwortet.

2.2.1 Die Hierarchiegleichungen

Um die Hierarchiegleichungen herzuleiten muss man den zweiten Summanden in der Liouville-Gleichung (2.3) näher betrachten. Der Vektor $\dot{\mathbf{p}}_i$ beschreibt die Impulsänderung von Teilchen i , also die Kraft auf dieses Teilchen. Diese Kraft lässt sich aufspalten in eine äußere Kraft \mathbf{K}_i^{ext} (hervorgerufen durch eingravierte EM-Felder z.B. in einem Teilchenbeschleuniger) und in die Kräfte die durch die Wechselwirkung mit anderen Teilchen hervorgerufen werden. Es gilt damit

$$\dot{\mathbf{p}}_i = \mathbf{K}_i^{ext} + \sum_{\substack{j=1 \\ j \neq i}}^{N_b} \mathbf{K}_{i,j} \quad (2.5)$$

Wobei $\mathbf{K}_{i,j}$ die Kraft ist, die Teilchen i durch Teilchen j erfährt. Setzt man (2.5) in (2.3) ein, so erhält man

$$\frac{\partial F_{N_b}}{\partial t} + \sum_{i=1}^{N_b} \dot{\mathbf{r}}_i \cdot \frac{\partial}{\partial \mathbf{r}_i} F_{N_p} + \sum_{i=1}^{N_b} \left(\mathbf{K}_i^{ext} + \sum_{\substack{j=1 \\ j \neq i}}^{N_b} \mathbf{K}_{i,j} \right) \cdot \frac{\partial}{\partial \mathbf{p}_i} F_{N_b} = 0. \quad (2.6)$$

Aufgrund der etwas umfangreichen Herleitung der Hierarchiegleichungen wird im folgenden nur der Weg zu ihrer Herleitung skizziert und die Resultate angegeben. Für die ausführliche Herleitung sei auf die Fachliteratur (z.B. [20]) verwiesen. Man geht nun so vor, dass man aus dem zweiten Summenterm die Summanden, für die i oder j gleich N_b sind, herauszieht. Dann wird die Gleichung über den Phasenraum für das N_b -te Teilchen integriert. Daraus ergibt sich die Liouville-Gleichung für die Verteilungsfunktion F_{N_b-1} . Allgemein ergibt sich die folgende Gleichung für die Verteilungsfunktion F_s :

$$\begin{aligned} \frac{\partial F_s}{\partial t} &+ \sum_{i=1}^s \dot{\mathbf{r}}_i \cdot \frac{\partial}{\partial \mathbf{r}_i} F_s + \sum_{i=1}^s \left(\mathbf{K}_i^{ext} + \sum_{\substack{j=1 \\ j \neq i}}^s \mathbf{K}_{i,j} \right) \cdot \frac{\partial}{\partial \mathbf{p}_i} F_s \\ &+ (N_b - s) \cdot \int_{\Omega_{s+1}} \sum_{i=1}^s \mathbf{K}_{i,s+1} \cdot \frac{\partial}{\partial \mathbf{p}_i} F_{s+1} d\Omega_{s+1} = 0. \end{aligned} \quad (2.7)$$

Insbesondere ergibt sich die Einteilchenverteilungsfunktion als

$$\begin{aligned} \frac{\partial F_1}{\partial t} &+ \dot{\mathbf{r}}_1 \cdot \frac{\partial}{\partial \mathbf{r}_1} F_1 + \mathbf{K}_1^{ext} \cdot \frac{\partial}{\partial \mathbf{p}_1} F_1 \\ &+ (N_b - 1) \cdot \int_{\Omega_2} \mathbf{K}_{1,2} \cdot \frac{\partial}{\partial \mathbf{p}_1} F_2 d\Omega_2 = 0. \end{aligned} \quad (2.8)$$

Bei den obigen Gleichung werden die betrachteten Teilchen als voneinander unterscheidbare Objekte angenommen. Im allgemeinen interessiert man sich jedoch nicht für bestimmte Teilchen. Daher wird die Verteilungsfunktion f_s definiert:

$$f_s := \frac{N_b!}{(N_b - s)!} \cdot F_s. \quad (2.9)$$

f_s ist die Verteilungsfunktion, die sich ergibt, wenn man Ensemble ununterscheidbarer Teilchen betrachtet.³ Mit (2.8) und (2.9) ergibt sich die Liouville-Gleichung für f_1 :

$$\frac{\partial f_1}{\partial t} + \dot{\mathbf{r}}_1 \cdot \frac{\partial}{\partial \mathbf{r}_1} f_1 + \mathbf{K}_1^{ext} \cdot \frac{\partial}{\partial \mathbf{p}_1} f_1 + \int_{\Omega_2} \mathbf{K}_{1,2} \cdot \frac{\partial}{\partial \mathbf{p}_1} f_2 d\Omega_2 = 0. \quad (2.10)$$

Man erkennt, dass in der vorgestellten Hierarchie von Gleichungen die Bestimmungs-gleichung für eine Verteilungsfunktion F_s bzw. f_s immer von der nächsthöheren Verteilungsfunktion F_{s+1} bzw. f_{s+1} abhängt. Um eine für praktische Zwecke lösbare Gleichung zu erhalten, muss man die Hierarchie abbrechen. Dazu betrachtet man den Term unter dem Integralzeichen, der von f_2 abhängt, näher. Die Funktion f_2 lässt sich schreiben als

$$f_2(\mathbf{r}_1, \mathbf{r}_2, \mathbf{p}_1, \mathbf{p}_2, t) = f_1(\mathbf{r}_1, \mathbf{p}_1, t) \cdot f_1(\mathbf{r}_2, \mathbf{p}_2, t) + g_{1,2}(\mathbf{r}_1, \mathbf{r}_2, \mathbf{p}_1, \mathbf{p}_2, t), \quad (2.11)$$

wobei $g_{1,2}$ die sogenannte Zweiteilchenkorrelationsfunktion darstellt. Sie gibt die statistischen Abhängigkeiten zwischen den Teilchen an. Sind die Teilchen vollkommen unabhängig voneinander, so ist $g_{1,2} \equiv 0$. Physikalisch bedeutet diese Annahme die Vernachlässigung von Stößen zwischen den Teilchen. Durch Einsetzen von (2.11) in (2.10) ergibt sich die **Vlassov-Gleichung**:

$$\frac{\partial f_1}{\partial t} + \dot{\mathbf{r}}_1 \cdot \frac{\partial}{\partial \mathbf{r}_1} f_1 + \mathbf{K}_1^{ext} \cdot \frac{\partial}{\partial \mathbf{p}_1} f_1 + \bar{\mathbf{K}}_{1,2} \cdot \frac{\partial}{\partial \mathbf{p}_1} f_1 = 0. \text{ (Vlassov-Gleichung)} \quad (2.12)$$

$\bar{\mathbf{K}}_{1,2}$ ist die mittlere Kraft zwischen zwei Teilchen im Sinne eines statistischen Erwartungswertes. Im Rahmen dieser Arbeit werden Systeme mit extern eingeprägten elektrischen \mathbf{E}^{ext} und magnetischen Feldern \mathbf{B}^{ext} betrachtet. Die Kräfte, die die Teilchen aufeinander ausüben, werden als elektrostatische Kräfte (Coulombkräfte)

³Der Zusammenhang (2.9) ergibt sich aus der Kombinatorik. Es gibt $\frac{N_b!}{(N_b-s)!}$ Möglichkeiten, s Teilchen aus einer Grundgesamtheit von N_b Teilchen anzurufen bzw. durchzumerken (siehe z.B.[15].)

angenommen. Diese Annahme ist dann gerechtfertigt, wenn das Magnetfeld, welches durch die Teilchenbewegungen erzeugt wird, gegenüber dem äußeren Magnetfeld vernachlässigt werden kann. Die Größe $\rho \cdot \mathbf{v}$, die ein Maß für die Ströme ist, die die Teilchenbewegungen darstellen, (und damit auch ein Maß für das dadurch erzeugte Magnetfeld) muss also klein ist. Damit folgt (siehe z.B. [7, 9]):

$$\mathbf{K}_1^{ext} + \bar{\mathbf{K}}_{1,2} = \underbrace{q \cdot (\mathbf{E}^{ext} + \mathbf{v}_1 \times \mathbf{B}^{ext})}_{= \mathbf{K}_1^{ext}} + \underbrace{q \cdot (\bar{\mathbf{E}}_{1,2})}_{= \bar{\mathbf{K}}_{1,2}} \quad (2.13)$$

Für ein Plasma, das aus Teilchen mit verschiedenen Eigenschaften besteht, wie z. B. Elektronen und Ionen, lässt sich für jede Teilchenart eine Vlassov-Gleichung angeben. Betrachtet man ein Plasma, das aus N_G verschiedenen Teilchenarten besteht, so erhält man ein System von N_G Vlassov-Gleichungen. Für eine ausführliche Formulierung siehe z. B. [20].

2.2.2 Die Newton-Lorenz Gleichung

In diesem Abschnitt soll der Zusammenhang zwischen den Trajektorien der einzelnen Teilchen im Phasenraum und der Vlassov-Gleichung (2.12) deutlich werden. Eine analytische Beschreibung der Bewegungsgleichung ist für jedes der einzelnen Teilchen möglich. Die Kraft auf ein geladenes Teilchen unter dem Einfluss elektrischer und magnetischer Felder kann mit Hilfe der **Newton-Lorenz-Gleichung** bestimmt werden, die auch bereits für (2.13) benutzt wurde (siehe z.B. [7, 9]):

$$\mathbf{F} = q \cdot (\mathbf{E} + \mathbf{v} \times \mathbf{B}) \quad (2.14)$$

Dabei ist q die Ladung und \mathbf{v} die Geschwindigkeit des Teilchens. \mathbf{E} bzw. \mathbf{B} sind die elektrischen und magnetischen Felder, die auf das Teilchen wirken. Das elektrische Feld \mathbf{E} setzt sich zusammen aus einem äußeren Anteil \mathbf{E}^{ext} , der aufgeprägt ist (z.B. in einem Teilchenbeschleuniger erzeugtes Feld) und einem Anteil \mathbf{E}^{int} , der durch die übrigen Teilchen erzeugt wird. Mit Hilfe der Beziehung

$$\mathbf{F} = \dot{\mathbf{p}} = \frac{d}{dt}(m \cdot \mathbf{v}), \quad (2.15)$$

wobei m die Masse des Teilchens und \mathbf{p} sein Impuls ist, lässt sich (2.14) schreiben als

$$\frac{d}{dt}(m \cdot \mathbf{v}) = q \cdot (\mathbf{E} + \mathbf{v} \times \mathbf{B}). \quad (2.16)$$

Mit dem Zusammenhang zwischen dem Ort des Teilchens \mathbf{r} und seiner Geschwindigkeit

$$\mathbf{v} = \dot{\mathbf{r}}, \quad (2.17)$$

lassen sich im Prinzip Ort und Geschwindigkeit eines Teilchens bestimmen. Die Trajektorie eines Teilchens im Phasenraum ist folglich bei vorgegebenen Anfangsbedingungen eindeutig bestimmt, und da dies für alle Teilchen des Plasmas gilt, ist das Verhalten des Plasmas als Ganzes nach diesem Modell vollständig determiniert. Der Zusammenhang der Trajektorien zu der im letzten Abschnitt angegebenen statistischen Beschreibung des Plasmas mit Hilfe der Vlassov-Gleichung liegt darin, dass die Trajektorien die Charakteristiken der Verteilungsfunktion f_1 im Phasenraum sind. Das bedeutet, dass f_1 entlang der Trajektorien seinen Wert nicht ändert. Eine Lösung von (2.12) kann also berechnet werden, indem eine unendliche Anzahl von „Testteilchen“ entlang der Charakteristiken-Lösung von (2.16) und (2.17) aufintegriert werden.

Allerdings ist es unmöglich (von trivialen Fällen einmal abgesehen) eine analytische Lösung von Gleichung (2.16) zu bestimmen. Um dennoch mit Hilfe des Modells Aussagen über das Verhalten eines Plasmas machen zu können, greift man zur Simulation mit Hilfe von Rechnern. Gemäß den Ausführungen im letzten Absatz ist das Ziel einer solchen Simulation folglich die Gleichungen (2.16) und (2.17) numerisch für eine große Anzahl von Testteilchen zu lösen und auf diese Weise eine Approximation für die Verteilungsfunktion f_1 zu erhalten. Dazu muss das Modell derart modifiziert werden, dass es sich auf einem Rechner in Form eines Algorithmus implementieren lässt.

Kapitel 3

Der PIC Algorithmus

In diesem Kapitel soll der PIC Algorithmus vorgestellt werden, wie er im Rahmen dieser Arbeit zur Simulation verwendet wurde. Dabei wird lediglich auf die Theorie eingegangen. Implementierungsdetails wie z.B. verwendete Datenstrukturen und ähnliches werden erst in Kapitel 5 angesprochen.

3.1 Einleitung

Die Anfänge der Simulation eines Plasmas mit Hilfe von Rechnern reichen bis in die fünfziger Jahre des zwanzigsten Jahrhunderts zurück [1]. Dabei war es zunächst gar nicht selbstverständlich, dass sich dadurch physikalisch verwertbare Ergebnisse ergeben würden, denn bis zu diesem Zeitpunkt waren lediglich Simulationen einfacher eindimensionaler Modelle mit Teilchen gleicher Ladung und gleicher Masse (z.B. Elektronenstrahlen) durchgeführt worden [1]. Ein Plasma setzt sich jedoch zusammen aus Teilchen mit stark unterschiedlichen Massen und Ladungen mit entgegengesetzten Vorzeichen, die dazu führen, dass das Plasma von außen betrachtet weitgehend elektrisch neutral ist. Zudem haben Plasmen oft eine hohe Temperatur, was bedeutet, dass die Teilchen eine hohe thermische Geschwindigkeit haben (ungeordnete Teilchenbewegung). Die sogenannte Debeye-Länge $\lambda_D := \frac{v_{thermisch}}{\omega_p}$ gibt eine Abschätzung an, ab welchen Größenordnungen mikroskopische Effekte (Teilchen-Teilchen Interaktionen) gegenüber dem kollektiven Verhalten des Plasmas vernachlässigt werden können. Dabei ist $v_{thermisch} = \|\mathbf{v}_{thermisch}\|_2$ die mittlere

thermische Geschwindigkeit der Teilchen und ω_P die Plasmafrequenz¹. Innerhalb eines Kubus, dessen Seiten die Länge λ_D haben, befinden sich bei einem realen Plasma etwa zwischen 10^2 und 10^6 Teilchen [1], wobei interessierende Probleme eine Ausdehnung haben, die in jeder Raumrichtung ein Vielfaches der Debeye-Länge ist. Dadurch sind zur Simulation eines Plasmas viel mehr Teilchen notwendig, als beispielsweise zur Simulation eines Elektronengases. Bei einem gängigen Plasmaproblem können durchaus 10^{14} bis 10^{24} Teilchen auftreten [1]. Bestünde die einzige Möglichkeit physikalisch interessante Ergebnisse mit Hilfe einer Simulation zu erhalten darin, die Trajektorien aller dieser Teilchen zu berechnen, so könnte man aufgeben, da dies in vertretbarer Zeit selbst mit den modernsten Rechnern nicht möglich ist. Oftmals interessiert man sich jedoch nur für ein kollektives Verhalten des Plasmas und nicht für die auf den kleinen Größenskalen auftretenden Teilchen-Teilchen Interaktionen. Insbesondere sind dies Stoßprozesse, die zum Austausch von kinetischer Energie zwischen den Teilchen führen. Zudem lässt sich zeigen, dass die Bedingung $N_D := n \cdot \lambda_D^3 \gg 1$ (Anzahl der Teilchen in einem Kubus mit der Kantenlänge λ_D) in gewisser Weise äquivalent zu der Forderung

$$\frac{\text{kinetische Energie}}{\text{mikroskopische,potentielle Energie}} \gg 1 \quad (3.1)$$

ist, die sich auch mit weniger Teilchen erfüllen lässt [1].

Daher ergibt sich die Idee, die Simulation mit weniger Teilchen durchzuführen, als in dem Plasma vorhanden sind. Diese sogenannten **Makroteilchen** modellieren damit das Verhalten mehrerer realer Teilchen. Wenn in den folgenden Teilen der Arbeit von Teilchen die Rede ist, so sind damit immer Makroteilchen gemeint.

Die Vernachlässigung der direkten Teilchen-Teilchen Interaktionen bedeutet physikalisch insbesondere, dass ein kollisionsfreies Plasma angenommen wird (siehe Abschnitt 2.2.1). Eine weitere Rechtfertigung der Vernachlässigung der mikroskopischen Plasmaeffekte ergibt sich aus der Betrachtung einer Punktladung q in einem quasineutralen Plasma. Es lässt sich zeigen, dass das elektrostatische Potenzial einer Punktladung q in einem quasineutralen Plasma in Abhängigkeit vom Abstand r von der Punktladung die Form

¹Unter der Plasmafrequenz versteht man die Frequenz, mit der Verschiebungen in der Ladungsverteilung auftreten und wieder verschwinden.

$$\phi(r) = \frac{q}{r} \cdot e^{-r/\lambda_D} \quad (3.2)$$

besitzt [20]. Aufgrund dieses exponentiellen Abfalls des Potenzials lässt sich physikalisch begründen, dass in Bereichen, die die Debeye-Länge deutlich überschreiten, die kollektiven Plasmaeffekte dominieren.

3.2 Zeitliche Diskretisierung der Newton-Lorenz Gleichung

Da ein Rechner nicht mit kontinuierlichen Größen umgehen kann, besteht ein erster Schritt in der Diskretisierung der Differenzialgleichungen (2.16) und (2.17). Zunächst soll die Diskretisierung in der Zeitvariablen t vorgestellt werden². Dazu wird häufig das Leap-Frog Verfahren benutzt [1, 11]. Das Verfahren ist vor allem wegen seiner Einfachheit attraktiv. Außerdem wurde in [1] gezeigt, dass das Verfahren trotz seiner Einfachheit sehr genaue Resultate liefert. Bei diesem Verfahren werden die Zeitableitungen in den Gleichungen (2.14) und (2.17) durch Differenzenquotienten ersetzt. Allerdings werden \mathbf{v} und \mathbf{x} nicht zu gleichen Zeiten ausgewertet, sondern um $\Delta t/2$ versetzt. Es folgt:

$$\mathbf{F}(t) = m \cdot \frac{\mathbf{v}(t + \frac{\Delta t}{2}) - \mathbf{v}(t - \frac{\Delta t}{2})}{\Delta t} + O(\Delta t^2) \quad (3.3)$$

$$\mathbf{v}\left(t + \frac{\Delta t}{2}\right) = \frac{\mathbf{x}(t + \Delta t) - \mathbf{x}(t)}{\Delta t} + O(\Delta t^2) \quad (3.4)$$

Ort und Geschwindigkeit eines Teilchens sind also niemals zur gleichen Zeit bekannt. In Abbildung 3.1 ist dargestellt, an welchen Stellen der Zeitachse der Ort und an welchen die Geschwindigkeit bekannt ist.

Die Herleitung ist im Anhang A kurz skizziert. Ein Problem, das sich bei der Anwendung der beiden Gleichungen ergibt ist, dass in Gleichung (3.3) die Kraft \mathbf{F} von

²Die Diskretisierung der Felder (\mathbf{E}, \mathbf{B}) wird in Abschnitt 3.3 dargestellt.

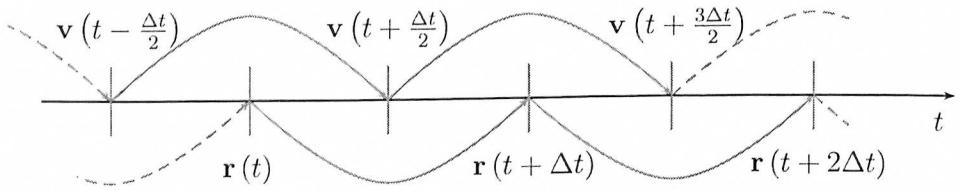


Abbildung 3.1: Zeitschritte beim Leap-Frog Algorithmus

\mathbf{v} abhängig ist. Die Gleichung lässt sich also nicht ohne weiteres zu einer expliziten Iterationsgleichung für \mathbf{v} umformulieren. Eine Möglichkeit, die auch bei der Software, die im Rahmen dieser Arbeit benutzt wurde, zum Einsatz kam, ist ein Verfahren, das von Boris [14] 1970 vorgestellt wurde. Boris schlägt vor, in Gleichung (3.3) für die Geschwindigkeit $\mathbf{v}(t)$, die in dem Ausdruck für $\mathbf{F}(t)$ auftritt, den Mittelwert der beiden Geschwindigkeiten $\mathbf{v}(t - \frac{\Delta t}{2})$ und $\mathbf{v}(t + \frac{\Delta t}{2})$ zu nehmen. Die Gleichung erhält dann folgende Form³:

$$\frac{q}{m} \cdot \left[\mathbf{E} + \frac{\mathbf{v}(t + \frac{\Delta t}{2}) + \mathbf{v}(t - \frac{\Delta t}{2})}{2} \times \mathbf{B} \right] = \frac{\mathbf{v}(t + \frac{\Delta t}{2}) - \mathbf{v}(t - \frac{\Delta t}{2})}{\Delta t} \quad (3.5)$$

Man substituiert nun $\mathbf{v}(t + \frac{\Delta t}{2})$ und $\mathbf{v}(t - \frac{\Delta t}{2})$:

$$\mathbf{v}\left(t - \frac{\Delta t}{2}\right) = \mathbf{v}^- - \frac{q}{m} \cdot \frac{\Delta t}{2} \cdot \mathbf{E} \quad (3.6)$$

$$\mathbf{v}\left(t + \frac{\Delta t}{2}\right) = \mathbf{v}^+ + \frac{q}{m} \cdot \frac{\Delta t}{2} \cdot \mathbf{E} \quad (3.7)$$

Damit verschwindet \mathbf{E} völlig aus Gleichung (3.5) und man erhält:

³Man beachte, dass die Situation für relativistische Betrachtung noch etwas komplizierter wird, da auch m von \mathbf{v} abhängt. Dieser Fall wird hier nicht betrachtet.

$$\frac{\mathbf{v}^+ - \mathbf{v}^-}{\Delta t} = \frac{q}{2m} \cdot (\mathbf{v}^+ + \mathbf{v}^-) \times \mathbf{B} \quad (3.8)$$

Das Vorgehen ist nun wie folgt. Man berechnet zunächst \mathbf{v}^- mit Hilfe von Gleichung (3.6), dann erhält man mit Hilfe von Gleichung (3.8) den Wert von \mathbf{v}^+ , worauf Gleichung (3.7) schließlich den Wert für $\mathbf{v}(t + \frac{\Delta t}{2})$ liefert. Gleichungen (3.6) bis (3.8) lassen sich auch wie folgt interpretieren. Gleichung (3.6) beschreibt zunächst die Beschleunigung des Teilchens im **E**-Feld mit dem halben Zeitschritt $\Delta t/2$. Gleichung (3.8) beschreibt eine Rotation des Vektors $\mathbf{v}^+ - \mathbf{v}^-$ im **B**-Feld, wobei sich der Betrag des Vektors $\mathbf{v}^+ - \mathbf{v}^-$ nicht ändert. Schließlich beschreibt Gleichung (3.7) wieder die Beschleunigung durch das **E**-Feld mit dem halben Zeitschritt $\Delta t/2$.

3.3 Räumliche Diskretisierung der Newton-Lorenz Gleichung

Nachdem in Abschnitt 3.2 gezeigt wurde, wie die Newton-Lorenz-Gleichung in der Zeitvariablen t diskretisiert werden kann, soll nun die räumliche Diskretisierung gezeigt werden. Dies betrifft die Felder **E** und **B**.

Um die Kraft auf ein Teilchen mit Hilfe von (2.14) auszurechnen ist es im Prinzip erforderlich **E**- und **B**-Feld am Ort des Teilchens zu kennen. Es ist jedoch prinzipiell nur möglich die Felder an bestimmten Punkten des Raumes - an den Punkten eines Gitters - zu berechnen. Um den Rechenaufwand vertretbar zu halten sind die Punkte dieses Gitters in jeder Raumrichtung deutlich weiter voneinander entfernt als es die Rechengenauigkeit des Rechners erfordert.

3.3.1 Lösung der Maxwell-Gleichungen

Der Anteil \mathbf{E}^{int} , der für die Wechselwirkungen der Teilchen untereinander verantwortlich ist, kann im Prinzip mit Hilfe der Formel für das **E**-Feld einer Punktladung

am Ort eines jeden Teilchens exakt (im Rahmen der Rechengenauigkeit des verwendeten Rechners) berechnet werden. Es ergibt sich für ein Plasma mit N_b Teilchen für das \mathbf{E}^{int} -Feld am Ort des i -ten Teilchens \mathbf{r}_i :⁴

$$\mathbf{E}^{int}(\mathbf{r}_i) = \frac{1}{4\pi\epsilon} \cdot \sum_{\substack{j=1 \\ j \neq i}}^{N_b} \frac{q_j}{\|\mathbf{r}_i - \mathbf{r}_j\|_2^3} \cdot (\mathbf{r}_i - \mathbf{r}_j) \quad (3.9)$$

Man beachte jedoch, dass dieses Vorgehen erfordert, für jedes der N_b Teilchen eine Summe über $N_b - 1$ Summanden zu berechnen. Das bedeutet, dass der Aufwand zur Berechnung von (3.9) quadratisch mit der Anzahl der Teilchen ansteigt ($O(N_b^2)$). Dieser Aufwand führt bereits für geringe Werte von N_b zu einem nicht mehr vertretbaren Rechenaufwand. Der Ansatz, der im PIC Algorithmus verfolgt wird, geht zur Berechnung der Felder den Umweg über die Ladungsdichte $\rho(\mathbf{r})$. Vereinfachend wird im Rahmen dieser Arbeit angenommen, dass das durch die Bewegung der Teilchen erzeugte Magnetfeld \mathbf{B}^{int} sehr viel schwächer ist, als das von außen angelegte \mathbf{B}^{ext} . Das ist insbesondere dann erfüllt, wenn die Größe $\rho(\mathbf{r}) \cdot \mathbf{v}(\mathbf{r})$, also das durch die Teilchen erzeugte Strömungsfeld, klein ist. Damit werden nur die Maxwell-Gleichungen für elektrostatische Probleme zur Feldberechnung benötigt. Die Maxwell-Gleichungen für ein elektrostatisches Problem lauten in differenzieller und integraler Form:

integrale Form

$$\oint \mathbf{E} \, ds = 0 \quad (3.10)$$

$$\iint_{\partial V} \mathbf{D} \, d\mathbf{A} = \iiint_V \rho \, dV \quad (3.11)$$

differenzielle Form

$$\nabla \times \mathbf{E} = 0 \quad (3.12)$$

$$\nabla \mathbf{D} = \rho \quad (3.13)$$

Zwischen \mathbf{E} - und \mathbf{D} -Feld besteht der Zusammenhang

$$\mathbf{D} = \epsilon \cdot \mathbf{E}, \quad (3.14)$$

⁴Es wurde ein homogenes, isotropes, lineares Material mit der Dielektrizitätskonstante ϵ angenommen, das das Rechengebiet ausfüllt. Falls ϵ nicht konstant ist, kann das Gesetz nicht mehr in der angegebenen Form geschrieben werden.

wobei ϵ die Dielektrizitätszahl an dem Punkt ist, an dem 3.14 ausgewertet wird. Aufgrund der Wirbelfreiheit des elektrischen Feldes gemäß Gleichung (3.10) kann ein Potenzialansatz für das elektrische Feld gemacht werden. Man setzt

$$\mathbf{E} = -\nabla\phi \quad (3.15)$$

Mit diesem Ansatz ist Gleichung (3.10) implizit erfüllt. Und man erhält mit Hilfe der Materialbeziehung aus (3.14) die **Poisson-Gleichung**

$$\nabla\epsilon\nabla\phi(\mathbf{r}) = -\rho(\mathbf{r}) \quad (\text{Poisson-Gleichung}) \quad (3.16)$$

Mit Hilfe der Poisson-Gleichung lässt sich eine Potenzialfunktion $\phi(\mathbf{r}) : \mathbb{R}^3 \rightarrow \mathbb{R}$ berechnen, aus der mit Gleichung (3.15) das \mathbf{E}^{int} -Feld berechnet werden kann. Um Gleichung (3.16) auf dem Rechengitter lösen zu können, ist auch hier eine Diskretisierung notwendig.

3.3.2 Diskretisierung der Poisson-Gleichung

Zur Diskretisierung der Gleichungen (3.15) und (3.16) auf einem Rechengitter für den PIC Algorithmus, müssen mehrere Probleme gelöst werden. Es muss entschieden werden, wie der Nabla-Operator zu diskretisieren ist und wie die Ladungsdichtefunktion ρ diskretisiert werden soll. Die zweite Frage wird im Abschnitt 3.3.3 behandelt. Zur Diskretisierung des Nabla-Operators wird das Rechengebiet mit einem Gitter überzogen. ϕ wird lediglich an den Punkten dieses Gitters ausgewertet. Dieses Rechengitter wird im folgenden als primäres Gitter bezeichnet. Um ein Gleichungssystem für die Werte des Potenzials an den Gitterpunkten des Rechengitters aufzustellen zu können, und gleichzeitig beliebige Materialfüllungen behandeln zu können, formuliert man mit Hilfe der integralen Gleichung (3.11) und mit dem Potenzialansatz aus (3.15) den Zusammenhang [11, 13]

$$-\iint_{\partial\tilde{V}_{i_u,i_v,i_w}} \epsilon(\nabla\phi) d\mathbf{A} = \iiint_{\tilde{V}_{i_u,i_v,i_w}} \rho dV. \quad (3.17)$$

Dabei ist $\tilde{V}_{i_u, i_v, i_w}$ das Volumen einer dualen Gitterzelle und $\partial\tilde{V}_{i_u, i_v, i_w}$ ihre Oberfläche. Das duale Gitter ist zu dem oben eingeführten primären Gitter um eine halbe Kantenlänge verschoben, so dass die Gitterpunkte des dualen Gitters in den Mittelpunkten der Gitterzellen des primären Gitters liegen und umgekehrt. Die rechte Seite von Gleichung (3.17) liefert die im Volumen der betrachteten dualen Gitterzelle eingeschlossene Ladung q_{i_u, i_v, i_w} . Das Integral auf der linken Seite wird nun aufgeteilt in Integrale über die einzelnen Flächen der dualen Gitterzelle. Zur Lösung der sich ergebenden Integrale wird zunächst der Nabla-Operator diskretisiert. Dies geschieht mit Hilfe einer Finiten-Differenzen-Näherung. So ergibt sich für die rechte Fläche der in Abbildung 3.2 gezeigten dualen Gitterzelle die folgende Näherung:

$$\iint_{\tilde{A}_u(i_u, i_v, i_w)} \epsilon(\nabla\phi) d\mathbf{A} \approx \frac{\Phi_{i_u, i_v, i_w} - \Phi_{i_u+1, i_v, i_w}}{h} \cdot \bar{\epsilon}_u(i_u, i_v, i_w) \cdot \tilde{A}_u(i_u, i_v, i_w). \quad (3.18)$$

Wobei h die Länge der Kante der primären Gitterzellen und $\tilde{A}_u(i_u, i_v, i_w)$ die Fläche der dualen Gitterzelle ist. Die mit $\bar{\epsilon}$ bezeichnete Größe ist der über die entsprechende Fläche der dualen Gitterzelle gemittelte Wert der Dielektrizitätszahl. In Abbildung 3.2 sind die Potenzialwerte eingetragen, die bei der Lösung der Poisson-Gleichung für die dargestellte duale Gitterzelle benötigt werden.

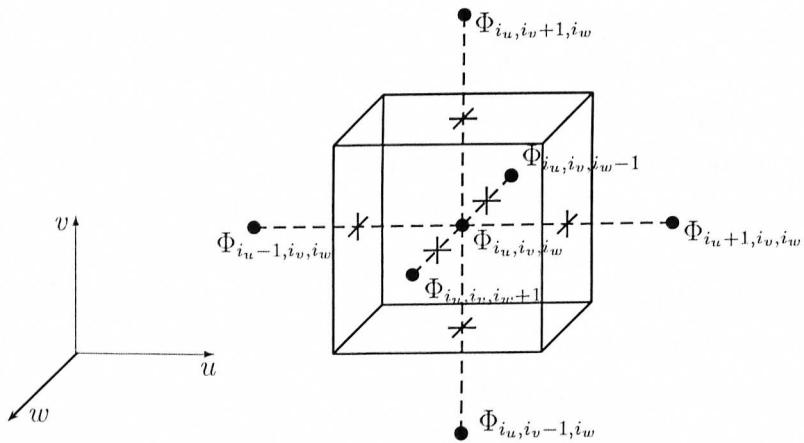


Abbildung 3.2: Zur Numerischen Lösung der Poisson-Gleichung

Insgesamt ergibt sich ein lineares Gleichungssystem für die Werte des Potenzials an den Gitterpunkten des primären Gitters. Dieses lässt sich mit den diskretisierten

Operatoren für die Divergenz und den Gradienten in der bekannten Formulierung [13] schreiben als

$$\tilde{\mathbf{S}}\mathbf{M}_\varepsilon \tilde{\mathbf{S}}^T \Phi = \mathbf{q} \quad (3.19)$$

Die Systemmatrix $\tilde{\mathbf{S}}\mathbf{M}_\varepsilon \tilde{\mathbf{S}}^T$ ist positiv definit [11], weshalb man eine Reihe von Standard-Verfahren zur Lösung einsetzen kann. In der verwendeten Software wurde das CG-Verfahren zur Lösung eingesetzt (siehe z. B. [12, 30]). Ist das Potenzial berechnet, so kann das elektrische Feld an den Gitterpunkten des primären Gitters durch den diskretisierten Gradientenoperator berechnet werden und es folgt:

$$\mathbf{e} = \tilde{\mathbf{S}}^T \Phi \quad (3.20)$$

Die Fehlerordnung der verwendeten Näherung hängt dabei von der Art des verwendeten Rechengitters ab [13]. Sie lässt sich mit Hilfe von Taylorentwicklungen von ϕ und \mathbf{E} ermitteln.

3.3.3 Allokierung der Ladung auf dem Gitter

Es stellt sich nun die Frage, wie die Ladung an den Gitterpunkten bestimmt werden soll, denn die Teilchen befinden sich im allgemeinen nicht an den Gitterpunkten des Rechengitters. Dazu wird ein Interpolationsverfahren eingesetzt. Jedem Teilchen wird dabei eine räumliche Ausdehnung in Form einer Dichtefunktion $S(\mathbf{r}) : \mathbb{R}^3 \rightarrow \mathbb{R}$ zugeordnet. Diese Funktion muss die Bedingung

$$\iiint_{\mathbb{R}^3} S(\mathbf{r}) d\mathbb{R}^3 = 1 \quad (3.21)$$

erfüllen. Damit ist die Ladungserhaltung im Rechengebiet gewährleistet. Unter der Annahme, dass das Rechengitter regelmäßig ist und die Gitterpunkte in u, v und w -Richtung um $\Delta u, \Delta v$ bzw. Δw auseinanderliegen, ergibt sich der Beitrag des

Teilchens i , das sich am Ort \mathbf{r}_i befindet und die Ladung q_i besitzt, zu der Ladung $q(i_u, i_v, i_w)$, die dem Gitterpunkt mit den Gitterkoordinaten (i_u, i_v, i_w) zugeordnet wird:

$$q_i(i_u, i_v, i_w) := q_i \cdot \int_{(i_u - \frac{3}{2}) \cdot \Delta u}^{(i_u - \frac{1}{2}) \cdot \Delta u} \int_{(i_v - \frac{3}{2}) \cdot \Delta v}^{(i_v - \frac{1}{2}) \cdot \Delta v} \int_{(i_w - \frac{3}{2}) \cdot \Delta w}^{(i_w - \frac{1}{2}) \cdot \Delta w} S(\mathbf{r} - \mathbf{r}_i) du dv dw \quad (3.22)$$

In Abbildung 3.3 (a) ist das Gebiet, über das $S(\mathbf{r} - \mathbf{r}_i)$ integriert wird, um den Beitrag der Ladung des Teilchens zur Ladung $q(i_u, i_v, i_w)$, die dem Gitterpunkt (i_u, i_v, i_w) zugerechnet wird, zu berechnen, mit einer gestrichelten Linie umrahmt. Aus Gründen der Übersichtlichkeit ist hier lediglich die Situation dargestellt, wie sie sich in einem zweidimensionalen Rechengebiet ergibt. Im folgenden wird das Integrationsgebiet für den Gitterpunkt (i_u, i_v, i_w) abkürzend mit G_{i_u, i_v, i_w} (bzw. G_{i_u, i_v} für den zweidimensionalen Fall) bezeichnet.

Die einfachste Interpolation (oft auch als Interpolation nullter Ordnung oder „Nearest-Grid-Point“ (NGP) bezeichnet [1, 4]) ordnet die Ladung dem Gitterpunkt zu, dem das Teilchen am nächsten liegt. Die Funktion $S(\mathbf{r})$ hat somit die Form

$$S(\mathbf{r}) = \delta(\mathbf{r}), \quad (3.23)$$

wobei $\delta(\cdot)$ die Dirac'sche Distribution bezeichnet. Diese Art der Interpolation ist in Abbildung 3.3 (b) dargestellt. Das in eingezeichnete Teilchen befindet sich innerhalb G_{i_u, i_v} und damit wird seine Ladung vollständig dem Gitterpunkt (i_u, i_v) zugerechnet. Für die NGP Interpolation folgt damit allgemein:

$$\iiint_{G_{i_u, i_v, i_w}} S(\mathbf{r} - \mathbf{r}_i) dG_{i_u, i_v, i_w} = \begin{cases} 1 & \text{falls } \mathbf{r}_i \in G_{i_u, i_v, i_w} \\ 0 & \text{falls } \mathbf{r}_i \notin G_{i_u, i_v, i_w} \end{cases} \quad (3.24)$$

Dieses Verfahren ist zwar sehr einfach, liefert aber numerisch schlechte Ergebnisse (siehe z.B. [1] oder [5]). Daher benutzt man für S eher Funktionen, die die Ladung

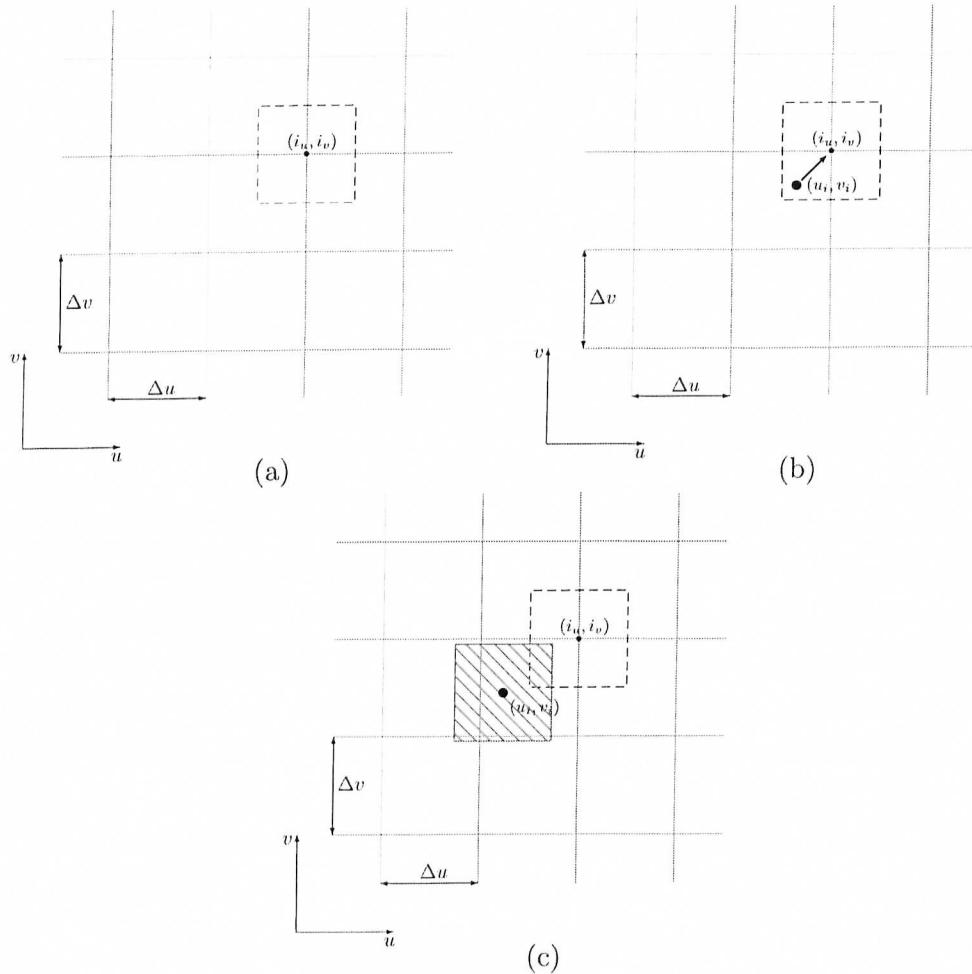


Abbildung 3.3: Die Zuordnung der Ladung eines Teilchens zu den Gitterpunkten.

(a) zeigt den Bereich über den S integriert wird, um den Anteil der Ladung des eingezeichneten Teilchens zu berechnen, der dem Gitterpunkt (i_u, i_v) zugerechnet wird.

(b) zeigt das Ergebnis der Interpolation nullter Ordnung: Die Ladung eines Teilchens, das sich im gestrichelt umrahmten Gebiet befindet, wird dem Gitterpunkt (i_u, i_v) zugerechnet.

(c) zeigt beispielhaft die Interpolation erster Ordnung. Der Anteil der schraffierten Fläche, der innerhalb des gestrichelt umrahmten Gebietes liegt, zur gesamten schraffierten Fläche entspricht dem Verhältnis der Gesamtladung des Teilchens zu der Ladung, die dem Gitterpunkt (i_u, i_v) zugerechnet wird.

nicht einem, sondern mehreren, dem Teilchen benachbarten Gitterpunkten, zurechnet (siehe z.B. [5, 4]). Eine Interpolation erster Ordnung führt z.B. auf eine Funktion S mit

$$\begin{aligned} S(\mathbf{r}) &= \frac{1}{\Delta u \cdot \Delta v \cdot \Delta w} \cdot \left(\sigma\left(r_u + \frac{\Delta u}{2}\right) - \sigma\left(r_u - \frac{\Delta u}{2}\right) \right) \cdot \dots \\ &\quad \dots \cdot \left(\sigma\left(r_v + \frac{\Delta v}{2}\right) - \sigma\left(r_v - \frac{\Delta v}{2}\right) \right) \cdot \left(\sigma\left(r_w + \frac{\Delta w}{2}\right) - \sigma\left(r_w - \frac{\Delta w}{2}\right) \right) \end{aligned} \quad (3.25)$$

wobei $\sigma(\cdot)$ die Einheitssprungfunktion beschreibt. In Abbildung 3.3 (c) ist dieser Fall dargestellt. Die schraffierte Fläche markiert den Bereich, in dem die Funktion S für das eingezeichnete Teilchen ungleich Null ist. Man erkennt, dass ein Teil der Ladung dem markierten Gitterpunkt zugerechnet wird, obwohl sich das Teilchen selbst außerhalb des umrahmten Gebietes befindet. Solche Interpolationsverfahren höherer Ordnung weisen im Allgemeinen ein besseres numerisches Verhalten auf (siehe z.B. [1]). Je glatter allerdings die Funktion S ist (je höher also die Ordnung des Interpolationverfahrens), desto aufwändiger wird auch die Berechnung, weshalb man selten ein Verfahren mit einer Ordnung größer eins wählt.

Schließlich sei noch eine kompakte, mathematische Formulierung für die Ladung an den einzelnen Gitterpunkten angegeben. Für ein Plasma mit N_b Makroteilchen folgt für die Gesamtladung, die einem Gitterpunkt zugerechnet wird

$$q(i_u, i_v, i_w) := \sum_{i=1}^{N_b} q_i \cdot \iiint_{G_{i_u, i_v, i_w}} S(\mathbf{r} - \mathbf{r}_i) \, dG_{i_u, i_v, i_w} \quad (3.26)$$

3.3.3.1 Behandlung von Materialfüllungen

Bei dem vorgestellten Allokierungsschema für die Teilchen wurde nicht berücksichtigt, dass das Rechengebiet mit unterschiedlichen Materialien gefüllt sein kann. Der im Rahmen dieser Arbeit verwendete Algorithmus kennt zwei Typen von Materialien. Der erste Typ wird als transparent für die Teilchen angenommen, d. h. die Ladung wird auf die Gitterpunkte allokiert wie im letzten Abschnitt gezeigt. Der zweite Typ ist absorbierend, d. h. ein Teilchen, das in dieses Material eindringt wird

gelöscht. Es stellt sich dabei die Frage, wie der Fall behandelt werden soll, dass sich ein Teilchen in einer Gitterzelle befindet, die teilweise mit solch absorbierendem Material gefüllt ist. Abbildung 3.4 veranschaulicht diesen Fall.

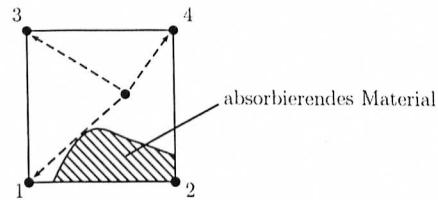


Abbildung 3.4: Behandlung von absorbierenden Materialien im PIC Algorithmus

Man geht hier so vor, dass man nur Ladung an den Punkten allokiert, die nicht im Material liegen und die Gewichte entsprechend modifiziert, so dass die Ladungserhaltung weiterhin gewährleistet ist. Die Bedingung für Ladungserhaltung ist, dass die Summe der Gewichte für jedes Teilchen 1 ergibt. Damit das weiterhin erfüllt ist, obwohl einzelne Gewichte zu Null gesetzt werden, müssen die verbliebenen mit einem Faktor normiert werden. Für das Beispiel aus Abbildung 3.4 würde sich ergeben:

$$\begin{aligned} \alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 &\stackrel{!}{=} 1 \\ \alpha_1 + \alpha_3 + \alpha_4 &= 1 - \alpha_2 \\ \underbrace{\frac{\alpha_1}{1 - \alpha_2}}_{=: \alpha'_1} + \underbrace{\frac{\alpha_3}{1 - \alpha_2}}_{=: \alpha'_3} + \underbrace{\frac{\alpha_4}{1 - \alpha_2}}_{=: \alpha'_4} &= 1 \end{aligned}$$

Wobei die α_i die Gewichte für die einzelnen Gitterpunkte für das betrachtete Teilchen sein sollen. Man erkennt, dass nach der Normierung für die neuen Gewichte α'_i die Ladungserhaltung noch immer gewährleistet ist.

3.3.4 Interpolation der Felder

Nach der Berechnung der Felder an den Gitterpunkten, stellt sich die Frage, welche Werte für \mathbf{E} und \mathbf{B} am Ort eines Teilchens angenommen werden sollen, das sich im allgemeinen nicht genau an einem Gitterpunkt aufhält. Dazu wird in [1, 5]

vorgeschlagen, das gleiche Interpolationsverfahren, das dazu benutzt wurde, um die Ladung eines Teilchens auf die Gitterpunkte zu verteilen, auch zur Berechnung der Felder zu benutzen. Es lässt sich zeigen, dass auf diese Weise vermieden wird, dass das Feld am Ort des Teilchen auf sich selbst (unphysikalische) Kräfte ausübt [4]. Das Feld am Ort des Teilchens wird aus den Feldwerten der Gitterpunkte berechnet, denen ein Teil der Ladung des Teilchens zugerechnet wurde. Diese Feldwerte werden gewichtet mit dem Verhältnis der Ladung die an dem entsprechenden Gitterpunkt allokiert wurde zu der Gesamtladung des Teilchens. Ein mathematischer Ausdruck für dieses Vorgehen lässt sich wiederum mit Hilfe der Funktion S angeben. Für das elektrische Feld $\mathbf{E}(\mathbf{r}_i)$ am Ort des Teilchens i ergibt sich auf diese Weise

$$\mathbf{E}(\mathbf{r}_i) := \sum_{i_u, i_v, i_w} \mathbf{E}_{i_u, i_v, i_w} \cdot \iiint_{G_{i_u, i_v, i_w}} S(\mathbf{r} - \mathbf{r}_i) dG_{i_u, i_v, i_w}. \quad (3.27)$$

Dabei steht $\mathbf{E}_{i_u, i_v, i_w}$ für das elektrische Feld am Gitterpunkt (i_u, i_v, i_w) , das durch die Lösung der Gitter-Poisson-Gleichung ergeben hat. Für die Beispiele aus Abschnitt 3.3.3 ist in Abbildung 3.5 gezeigt, welche Gitterpunkte zur Feldberechnung für das eingezeichnete Teilchen beitragen. Beim NGP-Verfahren wird das Feld an dem Gitterpunkt, der dem Teilchen am nächsten liegt, gleichzeitig als Feld am Ort des Teilchens angenommen. Bei dem vorgestellten Verfahren zur Interpolation erster Ordnung tragen die Feldwerte der nächsten vier Gitterpunkte (im zweidimensionalen Fall) zur Rechnung bei.

3.3.5 Flussdiagramm für den PIC-Algorithmus

Nachdem der PIC-Algorithmus in den vorangegangenen Abschnitten erläutert wurde, soll nun eine geschlossene Darstellung mit Hilfe eines Flussdiagrammes gezeigt werden. In Abbildung 3.6 sind die einzelnen Schritte in ihrer Reihenfolge dargestellt, wie sie bei einer Plasmasimulation mit Hilfe des PIC Algorithmus von einem Rechner abgearbeitet werden (ähnliche Darstellungen in [1] und [2]). Die gestrichelten Kästen stellen optionale Bestandteile des Codes dar. Die Emission der Teilchen kann durch ein iteratives Verfahren geschehen, das die Konsistenz sicherstellt (siehe [17]). Dieses Verfahren ist als mögliche Weiterentwicklung zu verstehen. Es wurde im Rahmen der vorliegenden Arbeit nicht benutzt.

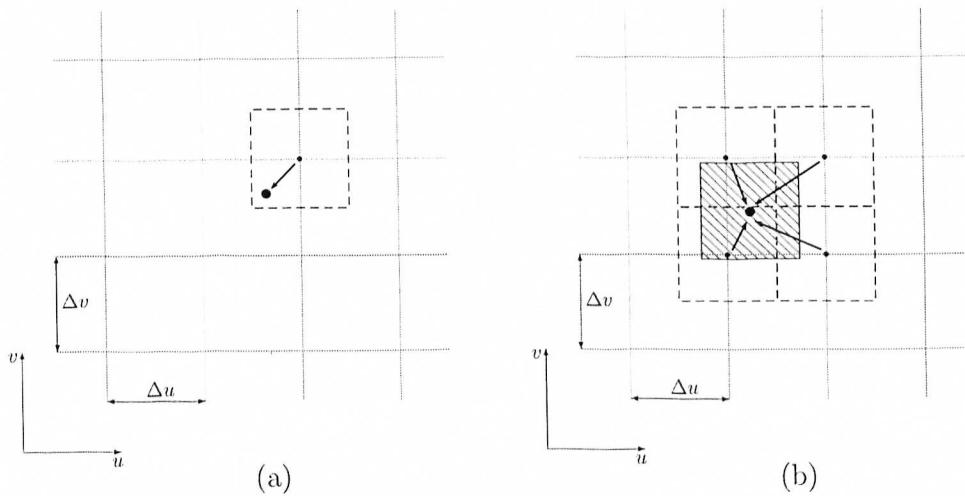


Abbildung 3.5: (a) Beim NGP Verfahren ist das Feld am Ort des Teilchens gleich dem Feld am Ort des nächsten Gitterpunktes. (b) Bei der Interpolation erster Ordnung tragen im allgemeinen die Feldwerte an den vier benachbarten Gitterpunkten zum Feld am Ort des Teilchens bei.

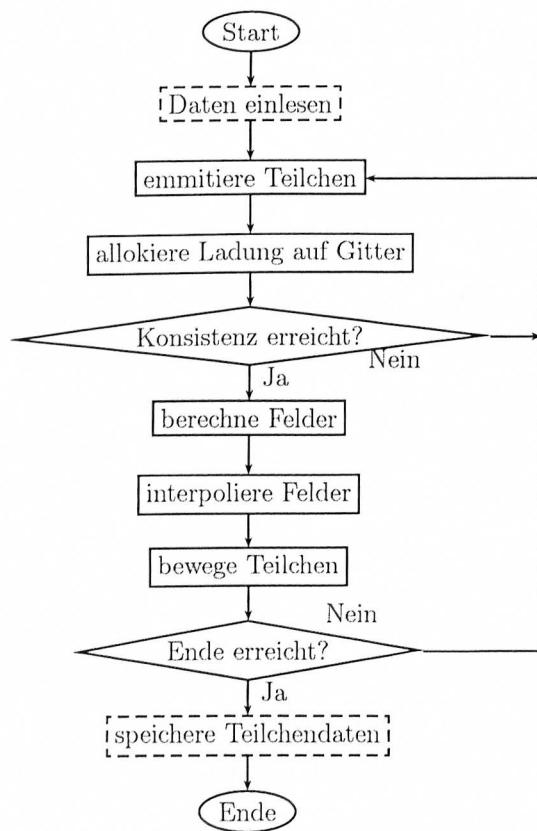


Abbildung 3.6: Flussdiagramm zum PIC-Algorithmus

Kapitel 4

Parallelisierung

In diesem Kapitel soll zunächst anhand einiger Beispiele die Motivation für eine Parallelisierung aufgezeigt werden. Danach werden einige wichtige Begriffe im Zusammenhang mit Parallelisierung allgemein erläutert. Die Informationen dazu stammen vor allem aus [18] und [28]. Im Anschluss werden einige wichtige Routinen aus dem MPI Standard vorgestellt, die im Rahmen dieser Arbeit genutzt wurde. Ist im folgenden Text von Prozessoren die Rede, so können diese Prozessoren sowohl Teile eines Mehrprozessorrechners sein, als auch auf mehrere Rechner verteilt sein, die über ein Netzwerk miteinander kommunizieren können.

4.1 Einleitung

In den Disziplinen der Ingenieurwissenschaften und der Physik ergeben sich häufig Problemstellungen, deren numerische Lösung auf einem einzigen Digitalrechner selbst mit den schnellsten verfügbaren Rechnern eine riesige Zeit in Anspruch nähme oder überhaupt nicht mehr möglich wäre. Zu denken ist hier z.B. an die numerische Wettersimulation, Fragestellungen aus der Astrophysik und ganz allgemein die Simulation von Vielteilchensystemen wie auch ein Plasma eines ist. Man versucht daher, das Gesamtproblem in kleinere Teilprobleme zu unterteilen (wie das im einzelnen geschieht wird später am Beispiel des PIC Algorithmus gezeigt) und diese Teilprobleme auf mehrere Rechner zu verteilen, die die Teilprobleme parallel lösen.

Die Bezeichnung „mehrere Rechner“ ist noch nicht sehr präzise. Was sich dahinter verbirgt, wird im nächsten Abschnitt erläutert.

4.2 Wichtige Begriffe

Zunächst sollen die verschiedenen Rechnerarchitekturen kurz vorgestellt werden, die im Rahmen der Parallelisierung auftreten. Die Unterteilung gehen auf M. J. Flynn zurück und haben eine weite Verbreitung gefunden [18].

- **SISD** - (Single Instruction Stream Single Data S- **SIMD** - (Single Instruction Stream Multiple Data SDatenparallelität bezeichnet
- **MIMD** - (Multiple Instruction Stream Multiple Data SInstruktionsparallelität.
- Ein wichtiger und in der Praxis häufig anzutreffender Sonderfall ist, dass alle Prozessoren eines MIMD Rechners das gleiche Programm ausführen, also datenparallel arbeiten. Dies wird als **SPMD** (Single Program Multiple Data) bezeichnet und wird auch in sehr vielen Plasmasimulationen verwendet. Dieser Ansatz vereint die relative Einfachheit der SIMD Architektur mit der Flexibilität der MIMD Architektur.

Im Rahmen dieser Arbeit wird der SPMD-Ansatz verfolgt. Große Parallelrechner mit MIMD Architektur haben bis zu einigen tausend Prozessoren. Für die effiziente Parallelisierung eines Problems auf solchen Rechnern (im Sinne einer möglichst

KAPITEL 4. PARALLELISIERUNG

kurzen Laufzeit) ist es daher erforderlich, nicht nur das Gesamtproblem in Teile mit möglichst gleichem Rechenaufwand zu teilen, sondern auch die Topologie der Prozessoren¹ zu berücksichtigen, um die zur Kommunikation zwischen den Prozessoren benötigte Zeit möglichst gering zu halten. Im Rahmen dieser Arbeit wird nur auf die Lastverteilung auf die Prozessoren eingegangen. Auf die Anordnung der Prozessoren wird nicht eingegangen und angenommen, dass alle Prozessoren in einem Stern geschaltet sind und über einen idealen Switch verbunden sind.

Eine weitere Klassifizierung von Parallelrechnern erfolgt durch die Art, wie der Arbeitsspeicher den Prozessoren zugeordnet ist. Man unterscheidet zwischen Rechnern mit gemeinsamem Speicher und solchen mit verteiltem Speicher. Bei einem Rechner mit gemeinsamem Speicher existiert ein Arbeitsspeicher für alle Prozessoren. Dabei können alle Prozessoren auf den gesamten Arbeitsspeicher zugreifen (siehe Abbildung 4.1 (b)). Bei einem Rechner mit verteiltem Speicher dagegen hat jeder Prozessor seinen eigenen Arbeitsspeicher, auf den nur er allein zugreifen kann (siehe Abbildung 4.1 (a)). Während bei einem Rechner mit verteiltem Speicher ein Datenaustausch zwischen den Prozessoren über den Arbeitsspeicher geschehen kann, ist bei Rechnern mit verteiltem Speicher eine explizite Kommunikation notwendig. Deshalb werden Rechner, die nach diesem Prinzip aufgebaut sind auch als „message passing machines“ bezeichnet. Der Nachteil von gemeinsam genutzttem Speicher ist jedoch, dass eine aufwändige Zugriffssteuerung für den Speicher notwendig ist, um gleichzeitiges Lesen und Schreiben verschiedener Prozessoren in ein und demselben Speicherbereich zu verhindern.

Unabhängig von der Architektur des Rechners unterscheidet man zwischen **expliziter** und **impliziter** Parallelisierung. Bei der impliziten Parallelisierung wird die Verteilung eines Problems auf die Prozessoren von speziellen Compilern festgelegt. Bei der expliziten Parallelisierung wird die Verteilung eines Problems auf die Prozessoren vom Anwender selbst, durch bestimmte Befehle, übernommen. Im Rahmen dieser Arbeit wird ausschließlich die explizite Parallelisierung betrachtet.

¹Anordnung der Prozessoren in dem Parallelrechner

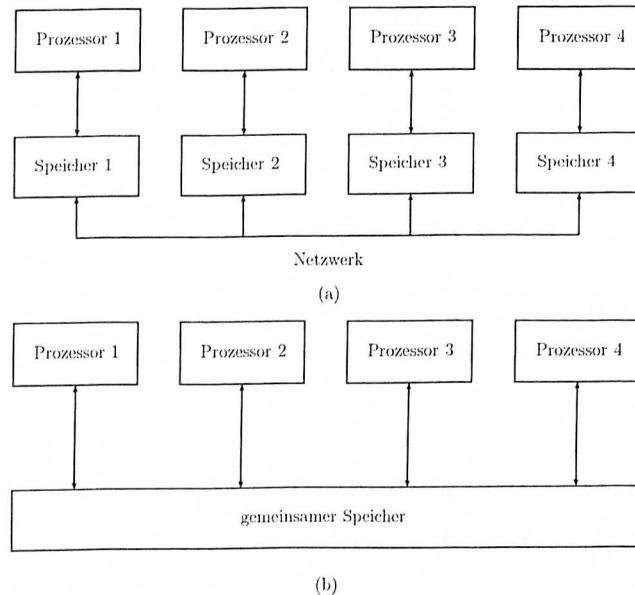


Abbildung 4.1: (a) ein Rechner mit verteiltem Speicher.(b) ein Rechner mit gemeinsamem Speicher.

4.3 Nachrichtenorientierte Parallelisierung mit MPI

Eine nachrichtenorientierte Parallelisierung lässt sich sowohl auf SIMD als auch auf MIMD Architekturen² anwenden. Bei diesem Ansatz ist der Programmierer dafür zuständig, die Kommunikation zwischen den Prozessen³ zu organisieren, falls ein Datenaustausch erforderlich ist, bzw. auch für die korrekte Synchronisation der Prozesse zu sorgen, soweit das für einen korrekten Programmablauf notwendig ist. Dies geschieht mit Hilfe von Kommunikationsbibliotheken, wie zum Beispiel **MPICH**. MPICH hat den Vorteil, dass es sich dabei um eine Bibliothek handelt, deren Benutzerschnittstelle standardisiert wurde. Es handelt sich dabei um eine Implementierung des MPI-Standards (Message Passing Interface) des Argonne National Laboratory. Inzwischen existieren Portierungen von MPI für alle großen Parallelrechner. Damit wird für ein Programm, das MPI-Routinen benutzt, eine sehr gute Plattformunabhängigkeit erreicht. Im Rahmen dieser Arbeit wurde MPICH in der Version 1.2.3

²im SPMD Modus

³Ein Prozess ist im allgemeinen nicht gleich einem Prozessor, jedoch kann man davon ausgehen, dass auf einem Rechner mit N_p Prozessoren auf dem N_p Prozesse gestartet werden, jedem Prozessor ein Prozess zugeordnet wird.

verwendet.

Ein Nachteil der nachrichtenorientierten Parallelisierung besteht darin, dass der Programmierer keinerlei Unterstützung von Seiten des Compilers erhält. Er muss sich um die effiziente Verteilung seines Problems auf die verfügbaren Prozessoren sowie um die richtige Kommunikation der Prozessoren selbst kümmern. Jedoch kann darin auch ein Vorteil liegen, denn neben allen Schwierigkeiten, die dadurch entstehen, hat der Programmierer die Möglichkeit spezielle Probleme hocheffizient zu parallelisieren.

4.3.1 Wichtige Befehle von MPI

Im folgenden soll eine kurze Vorstellung der wichtigsten MPI-Befehle erfolgen, so weit sie zum Verständnis der Arbeit unerlässlich sind. Der Teil des Programmcodes, der die Befehle der MPI-Bibliothek benutzt, wird in die beiden Befehle „`MPI_Init(ierr)`“ und „`MPI_Finalize(ierr)`“ eingeschlossen⁴. Bei der Initialisierung ordnet MPI jedem Prozess eine eindeutige Integer Zahl, seinen Rang, zu. Diese Zahl kann mittels der Funktion „`MPI_Comm_rank(MPI_Comm communicator, myrank, ierr)`“ von jedem Prozess abgefragt werden. In der Integer Variablen „`myrank`“ wird der Rang des aufrufenden Prozesses zurückgegeben. „`MPI_Communicator`“ ist ein sogenannter Kommunikator von MPI. Er beschreibt eine Gruppe von Prozessen (seine Mitglieder), die über den Kommunikator miteinander kommunizieren können. Im Rahmen dieser Arbeit wird nur der bereits vordefinierte Kommunikator „`MPI_Comm_World`“ benötigt, der alle Prozesse umfasst, die an der Ausführung des Programms beteiligt sind. Über die Befehle „`MPI_Send(buffer, count, MPI_Datatype, destination, tag, comm, ierr)`“ und „`MPI_Recv(buffer, count, MPI_Datatype, source, tag, comm, ierr)`“ lässt sich eine Punkt-zu-Punkt Kommunikation zweier Prozesse erreichen. „`buffer`“ gibt dabei beim sendenden Prozess den Teil des Speichers an, dessen Inhalte an den Empfänger verschickt werden sollen. Beim empfangenden Prozess gibt es den Teil des Speichers an, in den die empfangenen Daten geschrieben werden sollen. „`count`“ gibt an, wieviele Elemente des Datentyps „`MPI_Datatype`“ vom Sender zum Empfänger übertragen werden sollen. „`comm`“ ist

⁴Das Argument „`ierr`“ ist eine Integer Variable und dient zur Fehlerbehandlung. Sie kommt nur bei der MPI-Bibliothek für Fortran 95 vor.

der Kommunikator, der zur Kommunikation benutzt wird. Die Eindeutigkeit der Nachricht ist anhand der bisherigen Informationen noch nicht gegeben . Um es zu ermöglichen, dass mehrere Nachrichten gleichen Umfanges und Datentyps vom Sender zum Empfänger übertragen werden können, ohne dass das Ergebnis undefiniert ist, lässt sich mit „tag“ jede Nachricht mit einer Integer Zahl versehen, die sie eindeutig markiert. Damit eine Kommunikation mit den Befehlen „MPI_Send“ und „MPI_Recv“ funktioniert, muss sichergestellt sein, dass es zu jedem „Send“-Befehl auch einen entsprechenden „Recv“-Befehl gibt. Da die „Send“- und „Recv“-Befehle blockierend sind, wird ein Prozess, der den „Send“-Befehl erhält, so lange warten, bis ein anderer Prozess einen passenden „Recv“-Befehl aufgerufen hat. Wenn dies nicht erfolgt, wird er also nicht mehr mit der Ausführung des Programmes fortfahren. Über die Befehle „MPI_ISend(...“ und „MPI_IRecv(...“ mit den gleichen Argumenten wie „MPI_Send(...“ und „MPI_Recv(...“ wird dem aufrufenden Prozess erlaubt, mit der Ausführung des Programms fortzufahren, auch wenn der passende „ISend“- oder „IRecv“- Befehl von einem anderen Prozess noch nicht aufgerufen wurde.

Mit Hilfe der Routine „MPI_Barrier(MPI_comm, ierr)“ lassen sich die Prozesse synchronisieren. Alle Prozesse bleiben so lange an dieser Stelle des Programmes stehen, bis alle Prozesse, die den Kommunikator „MPI_comm“ benutzen, die Routine aufgerufen haben.

4.4 Effizienz einer Parallelisierung

Ein wichtiges Maß dafür, wie effizient eine Parallelisierung auf ein Problem angewandt wurde, ist der parallele Speedup S_{par} (siehe [18])

$$S_{par}(N_p) := \frac{T_{ges}(1)}{T_{ges}(N_p)}. \quad (4.1)$$

Hierbei bedeutet $T_{ges}(1)$ die Laufzeit, nach der ein Problem auf einem Einprozessorrechner gelöst wurde und $T_{ges}(N_p)$ die Laufzeit, die das gleiche Problem zur Lösung benötigt, wenn es auf N_p Prozessoren verteilt wird. Je größer $S_{par}(N_p)$ ist, desto effizienter ist die Parallelisierung.

Die Laufzeit, die zur Lösung eines Problems mittels einer Parallelisierung benötigt wird, wird von dem Prozessor bestimmt, der zur Lösung seines Teilproblems die längste Zeit benötigt. Sei $T_{ges}^{(i)}$ die Zeit, die Prozessor i zur Lösung seines Teilproblems benötigt, so gilt

$$T_{ges}(N_p) = \max_{i \in \{1, \dots, N_p\}} T_{ges}^{(i)}. \quad (4.2)$$

$T_{ges}(N_p)$ wird minimal, falls alle $T_{ges}^{(i)}$ gleich sind. Das folgt aus einer einfachen Überlegung. Benötigt ein Prozessor i länger als alle anderen zur Lösung seines Teilproblems, so könnte $T_{ges}(N_p)$ vermindert werden, wenn die anderen Prozessoren Last von i übernehmen würden. $T_{ges}(N_p)$ kann nur dann nicht mehr weiter gesenkt werden, wenn alle Prozessoren für ihre Teilprobleme die gleiche Zeit benötigen, denn dann kann $T_{ges}(N_p)$ durch Umverteilung der Last nicht mehr weiter gesenkt werden. Für die Effizienz einer Parallelisierung ist also die **Lastbalancierung** ein entscheidender Faktor.

Kapitel 5

Parallelisierung des PIC Algorithmus

In diesem Kapitel soll gezeigt werden, wie der PIC Algorithmus im Rahmen dieser Arbeit parallelisiert wurde. Dabei werden zunächst einige Ansätze zur Parallelisierung von PIC aus der Literatur vorgestellt. Das für die Parallelisierung ausgewählte Konzept wird dann in den Kontext dieser bekannten Verfahren eingeordnet. Es folgen einige Informationen zu den verwendeten Datenstrukturen und zu der Implementierung des Algorithmus. Dabei wird auch auf die Organisation der Kommunikation zwischen den Rechnern eingegangen.

5.1 Einige Ansätze aus der Literatur

In diesem Abschnitt sollen zunächst einige Verfahren zur Parallelisierung des PIC Algorithmus vorgestellt werden, wie sie in der Fachliteratur zu finden sind. Im nächsten Kapitel wird der Ansatz vorgestellt, der im Rahmen dieser Arbeit verfolgt wurde. Wie bereits in Abbildung 3.6 in Kapitel 3 gezeigt wurde, sind bei einer Plasmimulation mit Hilfe des PIC Algorithmus im wesentlichen zwei Probleme wiederholt zu lösen. Das erste Problem besteht in der Allokierung der Ladung auf dem Gitter sowie der Lösung der Poisson-Gleichung mit deren Hilfe das elektrische Feld an den Gitterpunkten berechnet werden kann. Das zweite Problem besteht in der Interpolation der Felder an den Ort der Teilchen sowie in der numerischen Lösung der

Newton-Lorenz-Gleichung (2.16), mit deren Hilfe die neue Geschwindigkeit und der neue Ort der Teilchen berechnet wird. Das Ziel besteht nun darin, diese beiden Probleme in Teilprobleme aufzuteilen, die von den einzelnen Prozessoren gelöst werden können. Aus den Überlegungen in Abschnitt 4.4 ist zudem klar, dass diese Aufteilung so erfolgen sollte, dass die entstehenden Teilprobleme möglichst die gleiche Zeit für ihre Lösung beanspruchen.

In der Literatur lassen sich einige Ansätze zu diesem Problem finden. In [22] wird ein Ansatz vorgestellt, bei dem das Rechengebiet in N_p Teilgebiete gleicher Größe aufgeteilt wird. Jedem Prozessor werden die Teilchen zugeordnet, die sich in seinem Rechengebiet befinden. Die Aufteilung des Gitters und der Teilchen sind also aneinander gekoppelt. In Abbildung 5.1 ist dieser Fall dargestellt.

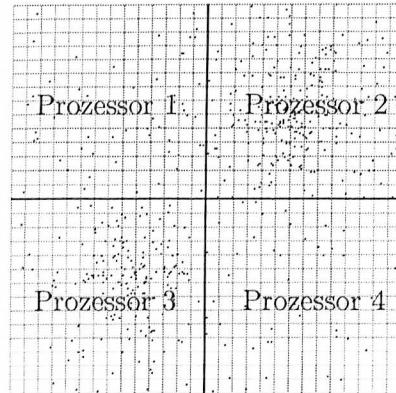


Abbildung 5.1: Gleichmäßige Aufteilung des Rechengebietes auf die Prozessoren.

Dieses Verfahren wurde zur Parallelisierung von XOOPICT, einem Programm zur Plasmasimulation mit graphischer Benutzerschnittstelle, verwendet [22]. Der Vorteil dieser Gebietsaufteilung ist sicherlich seine Einfachheit und seine Lokalität. Die Teilprobleme sind im wesentlichen unabhängig und es muss zwischen den Prozessoren lediglich eine Kommunikation erfolgen, wenn Teilchen von einem Rechengebiet in ein

KAPITEL 5. PARALLELISIERUNG DES PIC ALGORITHMUS

anderes eintreten und um die Randbedingungen für die Feldberechnung auszutauschen. Jedoch ist die Effizienz dieses Ansatzes umso geringer, je ungleichmäßiger die Teilchen auf die einzelnen Gebiete verteilt sind. Da die Gebiete alle die gleiche Größe besitzen, wird die Berechnung der Felder für alle Gebiete zumindest annähernd die gleiche Zeit benötigen. Die Lastbalancierung ist also für die Feldberechnung optimal gewährleistet. Für die Teilchen ist die Last jedoch nur dann balanciert, wenn sie gleichmäßig über das Rechengebiet verteilt sind, was im allgemeinen nicht der Fall ist. In Abbildung 5.1 ist beispielhaft eine Teilchenverteilung eingezeichnet. Man erkennt, dass Prozessor 1 praktisch keine Teilchen in seinem Rechengebiet hat, die Prozessoren 2 und 4 jedoch Anhäufungen von Teilchen in ihren Gebieten haben. Die Unterteilung wird dadurch ineffizient.

Ein weiterer Ansatz, der sich darum bemüht, die Last möglichst gleichmäßig auf die Prozessoren zu verteilen, entkoppelt die Gebietsaufteilung völlig von der Aufteilung der Teilchen. Er teilt die vorhandenen Teilchen so auf die Prozessoren auf, dass jeder Prozessor die gleiche Anzahl von Teilchen besitzt (siehe z.B. [24]). Der Ort, an dem sich die Teilchen befinden, spielt dabei keine Rolle. Der Vorteil dieses Ansatzes ist offenbar die optimale Lastbalancierung bezüglich der Feldberechnung und auch bezüglich der Berechnung der neuen Teilchenpositionen und Geschwindigkeiten. Ein entscheidender Nachteil, der sich vor allem bei großen Problemen, die über viele Prozessoren verteilt berechnet werden, ist, dass sowohl bei der Allokierung der Ladung auf dem Gitter, wie auch bei der Interpolation der Felder an die Teilchenpositionen, eine aufwändige Kommunikation notwendig ist. Im Gegensatz zum ersten Ansatz, wo üblicherweise nur geringe Datenmengen zwischen Prozessoren ausgetauscht werden müssen, die benachbarte Rechengebiete besitzen, muss nun eine Kommunikation zwischen allen Prozessoren stattfinden, da sich die Teilchen an beliebigen Orten des Rechengebietes befinden können. In [23] wird eine Untersuchung erwähnt, bei der sich herausgestellt hat, dass bei dieser Art der Lastbalancierung im ungünstigen Fall die Simulationszeit durch die Kommunikationszeit dominiert wird.

5.2 Verteilung des Gesamtproblems auf die Prozessoren

Wie im Abschnitt 5.1 gezeigt wurde, existieren grundsätzlich zwei Möglichkeiten, die Last auf die Prozessoren zu verteilen. Der erste Ansatz koppelt die Aufteilung der Teilchendaten und die der Rechengebiete für den Feldsolver aneinander, und der andere entkoppelt diese beiden Probleme. Im Rahmen dieser Arbeit wurde die erste Variante gewählt. Das bedeutet, dass die Teilchen dem Prozessor zugeordnet werden, in dessen Rechengebiet sie sich befinden. Dieses Verfahren hat den Vorteil, dass die Berechnungen in den einzelnen Gebieten weitgehend lokal erfolgen können. Eine Kommunikation der einzelnen Prozessoren ist lediglich dann notwendig, wenn ein Teilchen das Rechengebiet eines einen Prozessors verlässt und in das Rechengebiet eines anderen eintritt. Außerdem ist eine Kommunikation für die Ladungsallokierung an Randpunkten eines Rechengebietes notwendig, da die Ladung eines Randpunktes sowohl von Teilchen beeinflusst werden kann, die dem einen Prozessor zugeordnet sind, als auch von solchen, die dem Nachbarprozessor zugeordnet sind. Ansonsten können alle Rechnungen lokal erfolgen, insbesondere sind die für die Lösung der Newton-Lorenz-Gleichung erforderlichen Werte (**E**- und **B**-Feld) auf diese Weise für alle Teilchen lokal bekannt.

Diese Form Lastverteilung kann weiter unterteilt werden, je nachdem, wie strukturiert die entstehenden Teilprobleme sind. Die Varianten reichen hier von gleichmäßigen Gebietsaufteilungen (siehe z. B. [22]) bis hin zu nahezu völlig unstrukturierten Untergebieten wie sie sich durch die Lastverteilung mittels raumfüllender Kurven ergeben (siehe z. B. [25]). In Abbildung 5.2 werden die Verfahren zur Parallelisierung systematisch eingeordnet. Die hierarchischen Verfahren gehören zur Familie der Orthogonalen Bisektionierungen, von denen in Abbildung 5.3 einige Formen dargestellt sind. Die Form unter (a) wurde ausgewählt und implementiert. Diese Form wird als Orthogonale Rekursive Bisektionierung (ORB) bezeichnet. Neben dieser durch den Rechner vorgenommenen Verteilung des Gesamtproblems, lässt die programmierte Software auch die Möglichkeit einer benutzerdefinierten Gebietsaufteilung zu. Damit sind noch unstrukturiertere Aufteilungen möglich. Die einzige Einschränkung ist dann, dass die ausgewählten Gebiete kubisch sind.

Die Entscheidung, wie das globale Rechengebiet auf die Prozessoren aufgeteilt wer-

KAPITEL 5. PARALLELISIERUNG DES PIC ALGORITHMUS

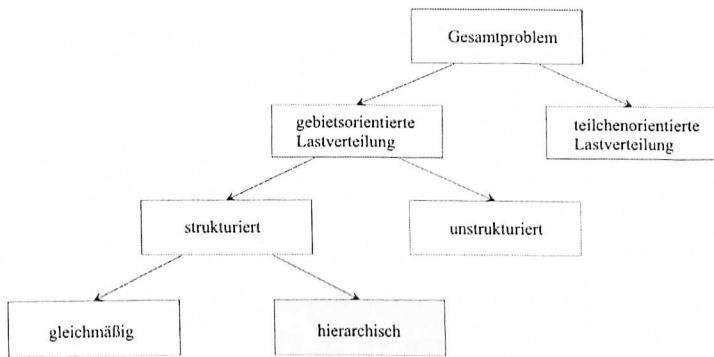


Abbildung 5.2: Übersicht über Parallelisierungsverfahren für PIC

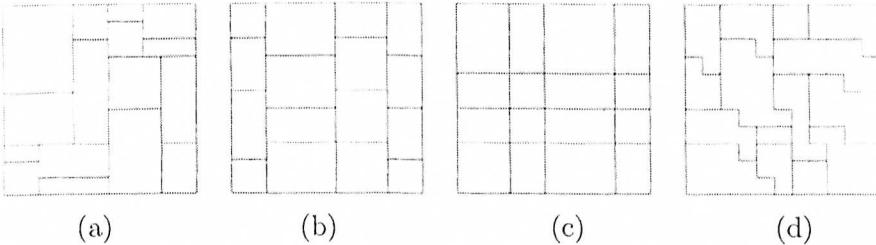


Abbildung 5.3: Die Familie der Orthogonalen Bisektionierungen (aus [25])

den soll, wird mit Hilfe eines rekursiven Algorithmus getroffen. Dazu wird jeder Gitterzelle (i_u, i_v, i_w) des primären Gitters ein Gewicht $w(i_u, i_v, i_w) \in \mathbb{R}$ zugeordnet. Dieses Gewicht setzt sich aus verschiedenen Größen zusammen, die die Rechenzeit beeinflussen, so zum Beispiel aus der Anzahl der Teilchen, die sich in dieser Zelle befinden, der gesamten Ladung in dieser Zelle und einem Mass für den Aufwand, die Felder für die Gitterpunkte der Zelle zu berechnen. Um nun das Gesamtgewicht $K_{\mathcal{G}}$ eines Gebietes \mathcal{G} zu berechnen, das sich aus bestimmten Gitterzellen $(i_u, i_v, i_w) \in \mathcal{G}$ zusammensetzt, addiert man die Gewichte der zu dem Gebiet gehörigen Gitterzellen. Man erhält:

$$K_{\mathcal{G}} = \sum_{(i_u, i_v, i_w) \in \mathcal{G}} w(i_u, i_v, i_w) \quad (5.1)$$

Gleichung (5.1) setzt voraus, dass das Gewicht eine lineare Funktion seiner Einflussgrößen ist. Ist dies nicht der Fall, so wird die Betrachtung komplizierter. Die Unter-

teilung wird nun so vorgenommen, dass N_P Untergebiete¹ $\mathcal{G}^{(1)}, \dots, \mathcal{G}^{(N_P)}$ entstehen, deren Gewichte möglichst gleich sind. Um das Problem weiter zu vereinfachen, werden für die $\mathcal{G}^{(i)}$ nur rechteckige Gebiete zugelassen. Die Unterteilung wird rekursiv vorgenommen. Das bedeutet, man beginnt mit dem globalen Rechengebiet und teilt dieses in zwei Teilgebiete, deren Gewichte im gleichen Verhältnis wie die Anzahl der Prozessoren stehen, die jedem der beiden Teilgebiete zugeordnet werden. Dabei wird die Anzahl der Prozessoren für jedes Teilgebiet möglichst so gewählt, dass sie gleich der Hälfte der verfügbaren Prozessoren ist. Es wird also so unterteilt, dass im i -ten Schritt möglichst

$$\frac{K_{Gebiet\ 1}}{K_{Gebiet\ 2}} = \frac{\left\lceil N_P^{(i)}/2 \right\rceil}{\left\lfloor N_P^{(i)}/2 \right\rfloor} \quad (5.2)$$

gilt. Dabei stehen die Bezeichnungen $K_{Gebiet\ 1}$ und $K_{Gebiet\ 2}$ für die Gewichte der beiden Untergebiete, die bei der Teilung entstehen, $N_P^{(i)}$ ist die Anzahl der Prozessoren, die im i -ten Schritt zur Aufteilung auf die Untergebiete zur Verfügung stehen. Die Klammern $\lceil \cdot \rceil$ und $\lfloor \cdot \rfloor$ bedeuten, dass der eingeschlossene Wert auf die nächstgrößere, ganze Zahl aufgerundet, beziehungsweise auf die nächstkleinere ganze Zahl abgerundet wird. Die beiden sich ergebenden Untergebiete, werden wiederum wie das Ausgangsproblem behandelt und weiter unterteilt. Dies wird so lange fortgesetzt, bis nur noch ein Prozessor pro Untergebiet zur Verfügung steht. Die Gebiete, die sich bis dahin ergeben haben, werden den einzelnen Prozessoren zugeordnet. Auf diese Weise ergibt sich eine hierarchische Aufteilung des Rechengebietes in rechteckige Untergebiete. Die Hierarchie, die sich durch dieses Verfahren ergibt, kann mit Hilfe eines Binärbaumes veranschaulicht werden. Bäume gehören zu gut untersuchten Datenstrukturen der Informatik [27]. In Abbildung 5.4 ist das Verfahren für vier Prozessoren veranschaulicht. Der schraffierte Teil dient jeweils als Ausgangsgebiet für die weitere Unterteilung bzw. ist in einem Blatt des Baumes das Gebiet, welches einem der Prozessoren zugeordnet wird.

Die Gebiete werden in Form von Gitterindizes gespeichert. Das bedeutet, dass von jedem Gebiet der Index zweier gegenüberliegender Eckzellen bekannt ist. Damit ist

¹ N_P sei die Anzahl der verfügbaren Prozessoren.

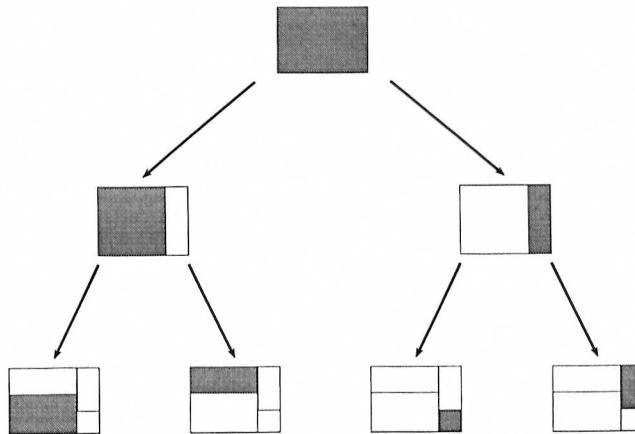


Abbildung 5.4: Veranschaulichung des Verfahrens zur Aufteilung des Rechengebietes auf die einzelnen Prozessoren. Jedem der Prozessoren wird ein Blatt des Baumes zugewiesen.

das Gebiet vollständig beschrieben. Jeder Knoten des Baumes enthält als Information das in ihm enthaltene Gebiet und Zeiger auf seine beiden Nachfolger in der Hierarchie. Die Blätter des Baumes enthalten zudem noch die Nummer des Prozessors, dem sie zugeordnet sind. Ein Knoten des Baumes lässt sich daher in Fortran 95 als folgender Datentyp angeben.

```

1 type type_node
2   type(type_bounding_box)::bbox
3   integer :: ProcessorNumber
4   type(type_node), pointer :: child1, child2
5 end type type_node
  
```

5.2.1 Erstellen des Baumes

Der Aufbau des Baumes erfolgt durch eine rekursive Routine. Dabei werden der Routine als Eingabewerte im ersten Schritt die Dimensionen des globalen Rechengebietes, die Zahl der für das Untergebiet verfügbaren Prozessoren $N_P^{(i)}$, sowie die Gewichte für jede der Zellen im globalen Rechengebiet übergeben. Die Routine bildet daraufhin zwei Teilgebiete, deren Gewichte möglichst genau die Bedingung aus Gleichung (5.2) erfüllen. Nun wird die Routine erneut aufgerufen und erhält als

Eingabewerte die Lage bezüglich des gesamten Rechengebietes und Größe des jeweiligen Untergebietes, das sie weiter unterteilen soll, die Gewichte für die Zellen dieses Untergebietes und die Anzahl der Prozessoren, die für dieses Untergebiet zur Verfügung stehen. Der Vorgang ist dann beendet, wenn nur noch ein Prozessor für ein Untergebiet übrig ist. Das Flussdiagramm in Abbildung 5.5 zeigt die Vorgehensweise.

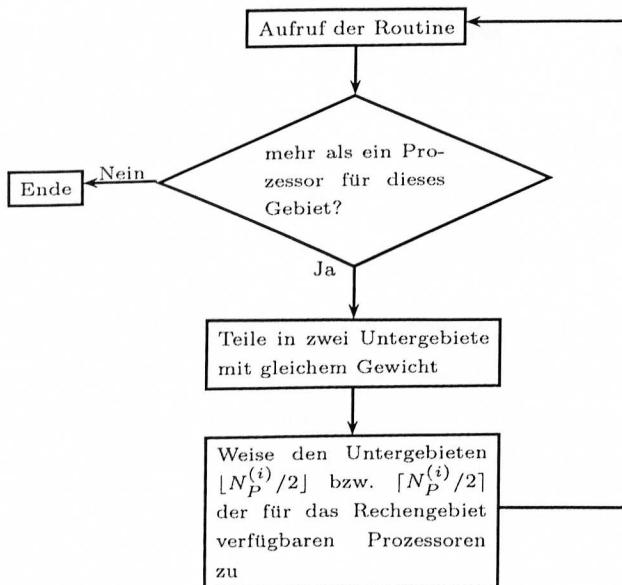


Abbildung 5.5: Flussdiagramm zum Erstellen des Baumes

5.2.2 Traversieren des Baumes

Nachdem der Baum aufgestellt wurde, ist es notwendig, dass zum einen jedem der verfügbaren Prozessoren eines der Rechengebiete eindeutig zugewiesen wird, und zum anderen, dass jeder Prozessor feststellt, welche Gebiete zu seinem benachbart sind. Von seinen Nachbarn kann er Teilchen geschickt bekommen und muss unter Umständen selbst Teilchen an sie verschicken.

Die Zuweisung der Prozessoren zu den Rechengebieten geschieht bereits während der Baum aufgebaut wird. Wird ein Blatt des Baumes erreicht, so wird diesem Blatt eine Prozessornummer zugewiesen. Auf diese Weise werden die Blätter des Baumes

fortlaufend nummeriert und jeder Prozessor erhält eindeutig ein Rechengebiet. Um Nachbarschaftsbeziehungen festzustellen muss der Baum traversiert werden. Das bedeutet, jeder Prozessor muss sein Gebiet mit denen der übrigen Prozessoren vergleichen und feststellen, ob sie gemeinsame Flächen, gemeinsame Kanten oder eine gemeinsame Ecke besitzen. Diese Traversierung geschieht wieder über eine rekursive Routine. Was im Falle einer Nachbarschaft mit einem anderen Rechengebiet getan wird, wird in Abschnitt 5.5 erläutert.

5.3 Datenstruktur zum Speichern der Teilchen

Zwei Verfahren erscheinen zur Speicherung der Teilchendaten sinnvoll. Bei dem ersten Ansatz werden die Teilchen in einer verketteten Liste gespeichert. Das bietet den Vorteil, dass die Speicherallokierung dynamisch erfolgen kann. Der Nachteil dieses Ansatzes ist jedoch, dass die Position der Teilchen in der Liste und auch im Speicher im Allgemeinen keine Aussagen über den Ort des Teilchens im Rechengebiet zulässt. Dadurch muss bei der Berechnung des neuen Ortes und der neuen Geschwindigkeit eines Teilchens zunächst festgestellt werden, in welcher Zelle sich das Teilchen befindet, um die entsprechenden Daten für das E- und B-Feld aus dem Speicher auszulesen. Das führt zu einer schlechten Nutzung des schnellen Cache-Speichers über den jeder Prozessor verfügt.

Bei einem anderen Ansatz werden die Teilchendaten auf einem Array gespeichert, dessen Größe vor der Laufzeit vorgegeben wird. Durch Sortieralgorithmen wie z. B. in [21] kann dafür gesorgt werden, dass die Teilchen, die sich in einer bestimmten Gitterzelle befinden, auch an lokal sehr begrenzten Stellen des Arrays gespeichert werden. Dieser Ansatz bietet den Vorteil, dass die Position auf dem Array recht gut mit der Position der Teilchen im Rechengebiet korrespondiert. Der Nachteil dieses Ansatzes sind die Rechenzeit verschlingenden Sortierprozesse, die verhindern sollen, dass die Ordnung der Teilchen auf dem Array verlorengeht.

In dieser Arbeit wurde ein Ansatz gewählt, der die Teilchen in Listen speichert. Dabei wird für jede der primären Gitterzellen eine Liste erzeugt, die die Teilchen enthält, die sich in jener Zelle befinden. Diese Listen werden im folgenden als Gruppen bezeichnet. Dieser Ansatz bietet den Vorteil, dass der schnelle Cache-Speicher der Rechner für die Berechnung der neuen Geschwindigkeit und Position für die

Feldwerte gut genutzt werden kann, da für alle Teilchen in einer Gruppe immer die gleichen Feldwerte benötigt werden. Außerdem eröffnet diese Form der Speicherung die Möglichkeit für eine sehr flexible Parallelisierung des Algorithmus, wie in Abschnitt 5.5.2 gezeigt wird. Der Nachteil dieses Ansatzes ist, dass die Position der Teilchendaten im Speicher im Allgemeinen nichts mehr mit der Position im Rechengebiet zu tun hat und daher für die Teilchendaten eine optimale Cachenutzung nicht garantiert werden kann.²

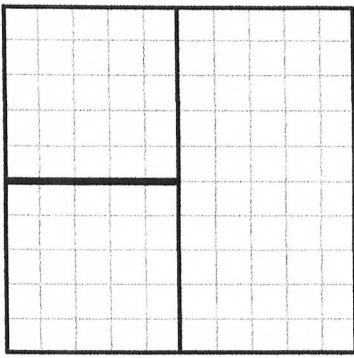
5.4 Interpretation der Gebietsteilung für die Rechengitter

In diesem Abschnitt soll gezeigt werden, wie die Teilung der Gebiete in Bezug auf das Rechengitter erfolgt. Die Zuordnung von Teilchen zu den primären Gitterzellen bedeutet, dass das Rechengebiet entlang der primären Gitterzellen geteilt wird. Das bedeutet, dass die dualen Gitterzellen von den Grenzen der lokalen Rechengebiete zerteilt werden. Abbildung 5.6(a) zeigt dies exemplarisch für ein zweidimensionales Gitter. Es wird deutlich, dass diese Art der Unterteilung es erforderlich macht, dass auf die Gitterpunkte, die an den Gebietsgrenzen liegen, für die Allokierung der Ladungen mehrere Prozessoren zugreifen können (bei einem dreidimensionalen Gitter sind das bis zu acht Prozessoren). Dieses Problem wurde so gelöst, dass man die rechte Seite des Gleichungssystems aus Gleichung (3.19) für alle Prozessoren zugänglich macht. Die Zuordnung der Gitterpunkte des primären Gitters ist nicht so offensichtlich, da es Gitterpunkte gibt, die direkt auf der Gebietsgrenze liegen und im Prinzip zu beiden Rechengebieten gehören³. Beim Aufbau der Matrix $\tilde{\mathbf{S}}\mathbf{M}_\epsilon\tilde{\mathbf{S}}^T$ aus Gleichung (3.19) verlangt die Bibliothek PETSc, die zur parallelen Lösung der Gitter-Poisson-Gleichung benutzt wurde (siehe z. B. [29]), jedoch die eindeutige Zuordnung der Gitterpunkte zu einzelnen Prozessoren. Daher muss festgelegt werden, welchem Prozessor die Gitterpunkte an den Gebietsgrenzen zugeordnet werden. Dies geschieht so, dass die Gitterpunkte immer demjenigen Prozessor zugeordnet werden, der in der betrachteten Raumrichtung die Gitterpunkte mit den höheren Indizes besitzt.

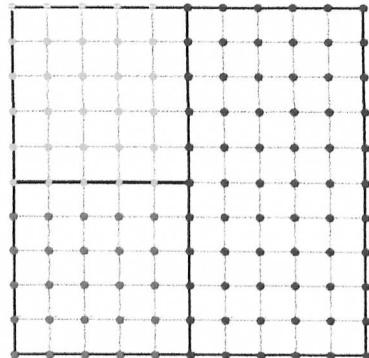
²Mit der „Position der Teilchendaten“ ist die physikalische Position der Daten des entsprechenden Teilchens im Speicher gemeint und **nicht** eine Gruppenzugehörigkeit.

³Beide Rechengebiete benötigen auch die Potenzialwerte an diesen Punkten.

Damit benutzt man für PIC und den Solver eine geringfügig andere Unterteilung. In Abbildung 5.6(b) ist die Zuordnung der Gitterpunkte für einen einfachen Fall mit drei Prozessoren farbig markiert. Nach dem Lösen der Gitter-Poisson-Gleichung mittels des iterativen CG-Verfahrens (siehe z.B. [12, 30]) wird die Lösung an alle Prozessoren verteilt.



(a)



(b)

Abbildung 5.6: (a) zeigt die Zugehörigkeit der Gruppen sowie der Zellen des primären Gitters zu den Rechengebieten. Die dick umrandeten Zellen gehören dabei jeweils zu einem Prozessor. (b) zeigt durch farbige Markierung die Zuordnung der Gitterpunkte zu den einzelnen Prozessoren. Man erkennt, dass die Unterteilung leicht unterschiedlich zu der aus (a) ist.

5.5 Organisation der Kommunikation

Aufgrund der Art der gewählten Aufteilung des Gesamtproblems in Teilprobleme, ist eine Kommunikation der Prozessoren (neben der Kommunikation zum Austausch der Werte für Ladung und Potenzial für die Gitterpunkte am Rand der Rechengebiete) dann erforderlich, wenn Teilchen von einem lokalen Rechengebiet in ein anderes wechseln. Um diese Kommunikation abzuwickeln wird wie folgt vorgegangen. Zunächst werden die Rechengebiete für PIC um die Zellen erweitert, die unmittelbar an das Rechengebiet angrenzen. Diese sogenannten „Geisterzellen“ existieren lediglich als Gruppen, in denen sich die Teilchen befinden, die während des aktuellen Zeitschrittes das Rechengebiet verlassen. In Abbildung 5.7(a) ist dies illustriert. Das

eingezeichnete Teilchen verlässt ein lokales Rechengebiet. Es wird zunächst einfach in die Gruppe eingetragen, die der grau eingezeichneten Geisterzelle zugeordnet ist. Nachdem für alle Teilchen die neuen Positionen und die neuen Geschwindigkeiten berechnet sind, müssen die Teilchen in den Geisterzellen zu den Prozessoren verschickt werden, in deren Rechengebiet sie sich befinden. Dies geschieht in mehreren einfachen Schritten, die in Abschnitt 5.5.2 erläutert werden. Zunächst soll jedoch gezeigt werden, wie die Geisterzellen genutzt werden können, um die Nachbarschaftsbeziehungen zwischen den Gebieten darzustellen.

5.5.1 Zuordnung der Geisterzellen

In Abschnitt 5.2.2 wurde gezeigt, wie die Nachbarschaft zwischen zwei Gebieten festgestellt wird. Sobald für ein Gebiet feststeht, dass es zu einem anderen Rechengebietes benachbart ist, so wird eine Liste mit Referenzen auf die Geisterzellen des lokalen Rechengebietes, die in dem benachbarten Rechengebiet liegen, erzeugt. Auf diese Weise reduziert sich die Suche nach Teilchen, die das eigene Rechengebiet verlassen haben und in das Gebiet eines bestimmten Nachbarprozessors gewechselt sind, auf die Traversierung der Liste von Geisterzellen, die für den entsprechenden Nachbarprozessor angelegt wurde. Alle Geisterzellen, die am Rande des globalen Rechengebietes liegen, werden einer speziellen Liste zugeordnet. Die Teilchen, die sich in diesen Zellen befinden, müssen nicht an benachbarte Prozessoren verschickt, sondern gelöscht werden.

Ein wesentlicher Punkt ist, dass bei der Traversierung des Baumes nicht nur festgestellt werden muss, zu welchem Nachbarprozessor die eigenen Geisterzellen gehören, sondern auch, von welchem der Prozessoren Teilchen empfangen werden müssen. Das heißt, es muss festgestellt werden, welchen Zellen im lokalen Rechengebiet die Geisterzellen der benachbarten Prozessoren zugeordnet sind. Dazu wird bei der Traversierung des Baumes, nachdem eine Nachbarschaftsbeziehung festgestellt wurde, neben der Liste, die Verweise auf die eigenen Geisterzellen enthält, eine zweite Liste angelegt, die Verweise auf die Zellen enthält, denen Geisterzellen anderer Prozessoren zugeordnet sind. In Abbildung 5.7 wird dies deutlich. Dies zeigt, dass zwar von mehreren Prozessoren Teilchen für ein und dieselbe Zelle empfangen werden können, aber eigene Geisterzellen steht nur einem Nachbarprozessor zugeordnet werden können.

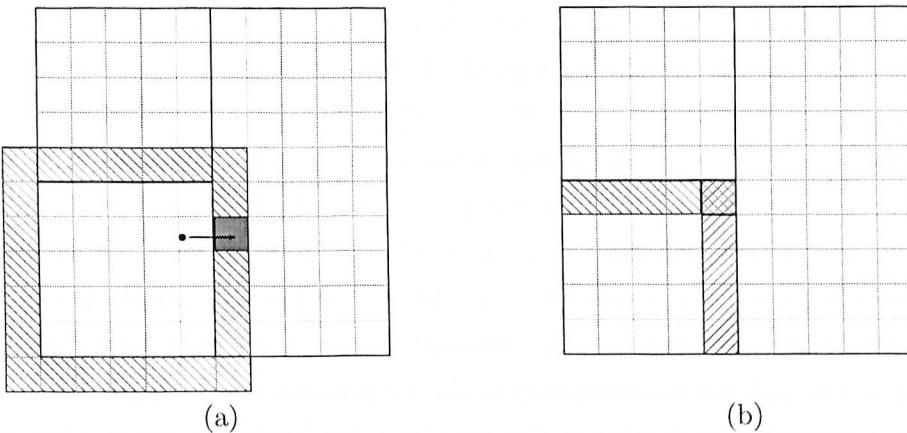


Abbildung 5.7: (a) zeigt die Geisterzellen des linken unteren Rechengebietes. (b) zeigt für welche Zellen der Prozessor, der das linke, untere Gebiet besitzt von seinen Nachbarn Teilchen empfängt. Man beachte, dass eine Zelle beiden Nachbargebieten zugeordnet ist.

5.5.2 Ablauf der Kommunikation

Nachdem für alle Teilchen der neue Ort und die neue Geschwindigkeit berechnet worden ist, müssen die Teilchen in den Geisterzellen verschickt werden und die Teilchen empfangen werden, die von benachbarten Rechengebieten in das eigene Gebiet gewechselt sind. Dies geschieht in mehreren Schritten. Zunächst werden die Listen mit Geisterzellen traversiert und alle Teilchen, die sich in einer Zelle befinden, entfernt und in einen Buffer kopiert. Damit liegen die Daten, die an einen Prozessor verschickt werden müssen, in einem kontinuierlichen Speicherbereich. Dies macht es möglich, alle Daten mit nur einer Nachricht an den Nachbarprozessor zu verschicken. Während die Teilchen in den Buffer kopiert werden, werden sie gleichzeitig gruppenweise gezählt. Dies ist aus zwei Gründen erforderlich. Zum einen muss der Nachbarprozessor, der die Teilchen empfangen soll wissen, wieviele Teilchen er erhält, da er ohne diese Information kein passendes „MPI_Recv(...“ (siehe Abschnitt 4.3.1) aufrufen kann, das die Angabe verlangt, wie groß die Datenmenge ist, die empfangen werden soll. Zum zweiten ist es dem Empfänger durch Angabe dieser Größen möglich, das Einsortieren der Teilchen in die entsprechenden Gruppen nach dem Ende der Kommunikation, extrem schnell und effizient vorzunehmen, wie später in diesem Abschnitt gezeigt wird.

Sobald alle Teilchen in den Buffer kopiert worden sind, wird an jeden Nachbarprozessor die Information verschickt, wieviele Teilchen er aus jeder Zelle erhält. Da die Länge dieser Nachricht nur von der Anzahl der Geisterzellen abhängt, die dem entsprechenden Nachbarprozessor zugeordnet sind, ist diese Länge dem Nachbarprozessor bekannt. Er kann also das entsprechende „`MPI_Recv(...)`“ aufrufen. Ist diese erste Kommunikation abgeschlossen, werden die Teilchen selbst verschickt. Ist auch dieser Teil der Kommunikation abgeschlossen, müssen die empfangenen Teilchen in die entsprechenden Gruppen eingesortiert werden. Dazu steht jedem Prozessor aus dem ersten Teil der Kommunikation, die Information zur Verfügung, wie viele Teilchen in jeder ihm zugeordneten Geisterzelle seines Nachbarn waren. Dies liefert ihm zusammen mit der Liste, in der er alle Zellen referenziert, für die er Teilchen von seinem Nachbarn erhält, alle Informationen, die er benötigt, um die Teilchendaten richtig einzusortieren ohne aus den empfangenen Teilchenpositionen wieder die richtigen Gruppen berechnen zu müssen oder ähnliches.

5.6 Adaptives Verfahren zur Lastverteilung

Einige Probleme, sind so beschaffen, dass sie nicht effizient mit Hilfe einer statischen Lastverteilung am Anfang der Laufzeit parallelisiert werden können. Dies betrifft auch die Simulationen, die im Rahmen dieser Arbeit durchgeführt wurden. Dabei handelte es sich um Simulationen, bei denen sich Teilchen nur in einem sehr begrenzten Teil des Rechengebietes aufhalten und bei denen das Rechengebiet zu Beginn der Simulation noch keine Teilchen enthält. Eine Entscheidung zur Aufteilung des Rechengebietes kann also nur aufgrund der Gebietsgröße getroffen werden. Damit ist eine effiziente Aufteilung des Rechengebietes nur für die Berechnung der Felder möglich, da der Aufwand hier direkt mit der Größe des Rechengebietes verbunden ist. Wenn sich das Rechengebiet nun mit Teilchen füllt, so kann sich herausstellen, dass eine ineffiziente Aufteilung des Gebietes gewählt wurde. Es besteht nun der Wunsch, die Last auf den einzelnen Rechnern zu beobachten und bei einem zu großen Ungleichgewicht, die Entscheidung, wie das Rechengebiet zerlegt werden soll, neu zu treffen. Um die Entscheidung treffen zu können, das Rechengebiet neu aufzuteilen, teilen sich die Rechner in jedem Zeitschritt ihre Gewichte $K^{(i)}$ mit. Aus den Gewichten kann ein mittleres Gewicht

$$\bar{K} := \frac{1}{N_P} \sum_{i=1}^{N_P} K^{(i)} \quad (5.3)$$

berechnet werden. Als Entscheidungsgröße wurde

$$\sigma = \max_i \left\{ \frac{K^{(i)} - \bar{K}}{\bar{K}} \right\} \quad (5.4)$$

gewählt. Man beachte, dass \bar{K} genau das Gewicht ist, dass sich für eine völlige Lastbalancierung als Gewicht der einzelnen Gebiete ergibt. Sobald die Last nicht mehr gleichmäßig verteilt ist, wird die Größe einen Wert größer Null annehmen. Da bei einem parallelen Programm außerdem der langsamste Prozessor die Laufzeit bestimmt, ist es sinnvoll als Entscheidungsgröße gerade die maximale Abweichung von dem balancierten Zustand zu benutzen.

Ist die Entscheidung getroffen, dass die Last neu verteilt werden soll, so ist es erforderlich, dass die Teilchen gemäß der neuen Gebietsaufteilung auf die Rechner verteilt werden. Dazu wird zunächst unter den Rechnern die Information ausgetauscht, wieviele Teilchen sich in jeder Zelle des Rechengebietes befinden. Aus dieser Information baut jeder der Prozessoren einen neuen Baum auf, der die Gebietszerlegung beschreibt. Der Baum, der die alte Gebietsaufteilung beschreibt, existiert parallel dazu weiter. Auf diese Weise kennt jeder der Prozessoren sein Rechengebiet in der alten und neuen Gebietsaufteilung und weiß automatisch, die Teilchen aus welchen Zellen er an welche Nachbarprozessoren verschicken muss, beziehungsweise auch, von welchen Prozessoren er Daten geschickt bekommt.

5.7 Flussdiagramm für den parallelisierten PIC Algorithmus

Zum Abschluss dieses Kapitels soll noch kurz das veränderte Flussdiagramm gezeigt werden, das sich nach der Parallelisierung ergibt. Die veränderten Bereiche sind durch Kästen mit schraffiertem Hintergrund gekennzeichnet. Wie bereits in Kapitel 3 gilt auch hier, dass die Konsistenzüberprüfung für die Emmission lediglich eine mögliche Weiterentwicklung des Codes ist. Sie wird im Rahmen dieser Arbeit nicht verwendet.

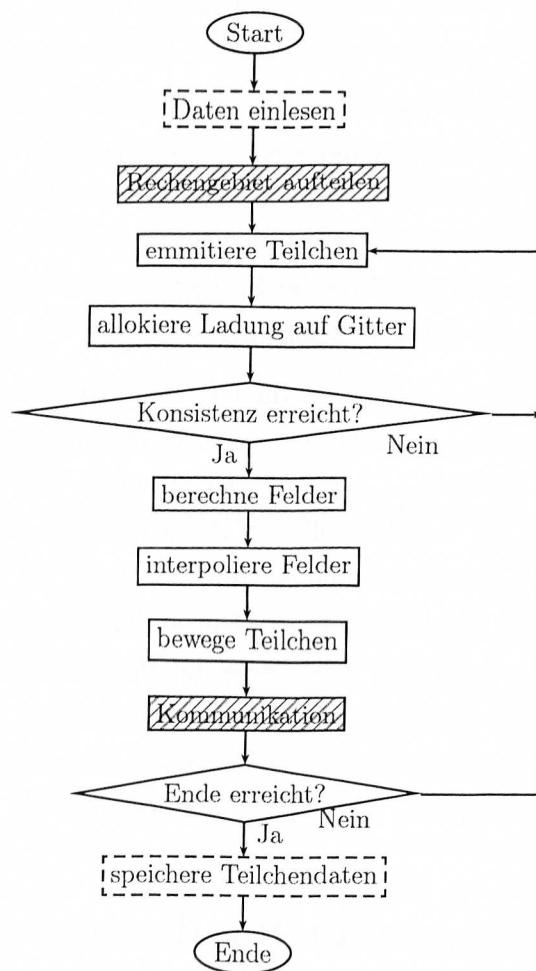


Abbildung 5.8: Flussdiagramm für den parallelisierten PIC Algorithmus

Kapitel 6

Simulationsbeispiele

In diesem Kapitel sollen die im Rahmen dieser Arbeit durchgeführten Simulationen vorgestellt werden. Das Ziel bei diesen Simulationen war es zum einen, die Skalierbarkeit der einzelnen Teile des implementierten Algorithmus zu testen und zum anderen zu zeigen, dass mit der entstandenen Software Probleme gelöst werden können, die auf einem einzelnen Rechner nicht mehr mit vertretbarem Aufwand simuliert werden können.

6.1 Ionenstrahl im konstanten Magnetfeld

Bei dem ersten simulierten Modell handelt es sich um einen Strahl bestehend aus schweren Ionen. Die emittierten Teilchen werden als einfach ionisiertes Potassium mit einer Masse $m = 39,1 \text{ amu}$ angenommen. In [35] wird dieses Modell simuliert, indem eine numerische Lösung der Vlassov-Gleichung berechnet wurde (2.12). Das Modell besteht aus einer runden Emissionsfläche mit einem Radius von $r = 0,02\text{m}$. Als Anfangsverteilung der Makroteilchen auf der Oberfläche wird eine Gleichverteilung angenommen. Weiter wird der Strahlstrom auf $0,2 \text{ A}$ festgesetzt. Mit Hilfe der kinetischen Energie des Strahls, die als $8 \cdot 10^4 \text{ eV}$ angenommen wird, lässt sich die Geschwindigkeit berechnen, mit der die Teilchen emittiert werden [35]. Um die Konvergenz des Solvers zu verbessern, wurde um den Ionenstrahl eine Röhre bestehend aus einem als ideal leitfähig angenommenen Material gelegt. Das Modell wurde

verwendet, um die Skalierbarkeit der einzelnen Codeteile zu analysieren. In den Abbildungen 6.1 sind drei verschiedene Phasenraumbilder für den Ionenstrahl gezeigt. Es ist die Position in x -Richtung gegen die Eigengeschwindigkeit u_x aufgetragen. Die Bilder veranschaulichen die Vorgänge in unterschiedlichen Abständen von der Kathode, wobei der Abstand zur Kathode vom linken zum rechten Bild zunimmt. Im ersten Bild ist der Zustand unmittelbar auf der Emissionsfläche ($u_x = 0$ und alle Teilchen gleichmäßig über das Intervall $[-0,01; 0,01m]$ verteilt) noch deutlich zu erkennen.

6.2 Simulation einer Elektronenkanone

Die von John R. Pierce in den 1940er Jahren in den Bell Laboratories entwickelte und später nach ihm benannte Elektronenkanone kommt noch heute in zahlreichen elektrischen Geräten zum Einsatz [31]. Unter anderem wird sie in den Wanderwellenröhren in Kommunikationssatelliten eingesetzt. Die Form der Kathode und der Anode sind so aufeinander abgestimmt, dass die Äquipotenzialflächen des elektrischen Feldes, orthogonal zu der Bewegung der emittierten Elektronen verläuft. Das elektrische Feld zwischen Anode und Kathode wird zur Beschleunigung der Elektronen verwendet. Dadurch, dass die Feldlinien parallel zu der Bewegung der emittierten Elektronen verlaufen, kann erreicht werden, dass das elektrische Feld über weite Strecken keine defokussierende Wirkung besitzt [32]. Allerdings gilt die oben beschriebene Orthogonalität von Äquipotenzialflächen und Elektronenbewegung für den von der Kathode weiter entfernten Bereich nicht mehr. Um die defokussierende Wirkung durch das elektrische Feld und auch durch die Raumladungskräfte zu verhindern, muss zusätzlich ein magnetisches Feld parallel zur Strahlachse angelegt werden. Da die Magnetfeldlinien die Tendenz haben, die Elektronen zu führen, kann der Strahl auf diese Weise in der Röhre gehalten werden. Das für die Simulation verwendete Modell ist in Abbildung 6.2 abgebildet.

6.2.1 Modellierung der Emission

Die Emission der Elektronen wird mit Hilfe der Child-Langmuir Gleichung modelliert [33, 34]:

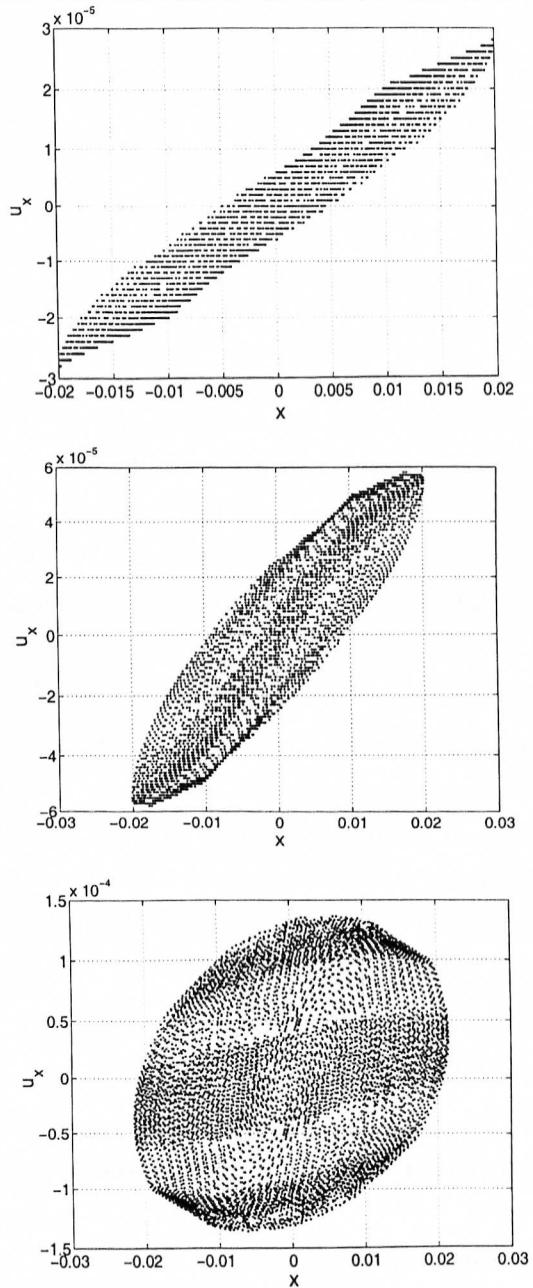


Abbildung 6.1: Phasenraumbilder des Ionenstrahls. Auf der horizontalen Achse ist die Position der Teilchen in x -Richtung aufgetragen und auf der vertikalen Achse ist die Eigengeschwindigkeit u_x der Teilchen in x -Richtung aufgetragen.

$$J_{CL} = \left(\frac{4\epsilon_0}{9} \right) \sqrt{\frac{2e}{m_e}} \frac{\delta\phi_b^{3/2}}{\delta d^2} \quad (6.1)$$

Dabei bedeutet J_{CL} die auf der Oberfläche der Kathode durch die emittierten Elektronen entstehende Stromdichte, e und m_e sind Elektronenladung und Elektronenmasse, δd beschreibt einen kleinen Abstand von der Emissionsfläche und $\delta\phi$ beschreibt die Potenzialdifferenz zwischen Kathode und einem Punkt im Abstand δd von der Kathode. Die Emission ist damit über das Potenzial und die Poisson-Gleichung (3.16) abhängig von der Raumladungsdichte in der Nähe der Kathode.

Mit Hilfe des lokal berechneten Wertes für die Stromdichte J_{CL} kann nun die Ladung eines Makroteilchens bestimmt werden. Aufgrund der Tatsache, dass die Ladung der Makroteilchen davon abhängig ist, wieviele Makroteilchen emittiert werden, verändern sich die einzelnen Trajektorien nicht, wenn die Zahl der Makroteilchen variiert wird.

Um eine bessere Anpassung an die tatsächliche Form der Kathode als Kugelsegment zu bekommen, werden die Makroteilchen nicht direkt auf der Kathode erzeugt, die durch die Triangulierung nicht mehr die Form eines idealen Kugelsegmentes hat, sondern auf einer separat erzeugten Emissionsfläche, die sich etwas vor der Kathode befindet. In Abbildung 6.2 ist ein Teil der Geometrie dargestellt. Man erkennt, dass die Teilchen ein Stück vor der Emissionsfläche erscheinen.

6.3 Simulationsergebnisse

6.3.1 Skalierbarkeit des PIC Algorithmus

Das Modell des Ionenstrahls, wurde dazu verwendet, die Skalierbarkeit der einzelnen Codeteile zu messen. Dazu wurden zunächst alle Routinen zur Berechnung der Felder ausgeschaltet und gemessen, wie sich die Berechnung der neuen Orte und Geschwindigkeiten der Teilchen skalieren lassen. Dazu gehört die Traversierung der Listen, in denen die Teilchendaten gespeichert sind, die Interpolation der externen Felder und schließlich die eigentliche Berechnung der numerischen Lösung der Newton-Lorenz-Gleichung wie in Abschnitt 2.2.2 gezeigt. Die Simulationen liefen so lange, bis die ersten Teilchen das Ende des Rechengebietes erreichten und dann

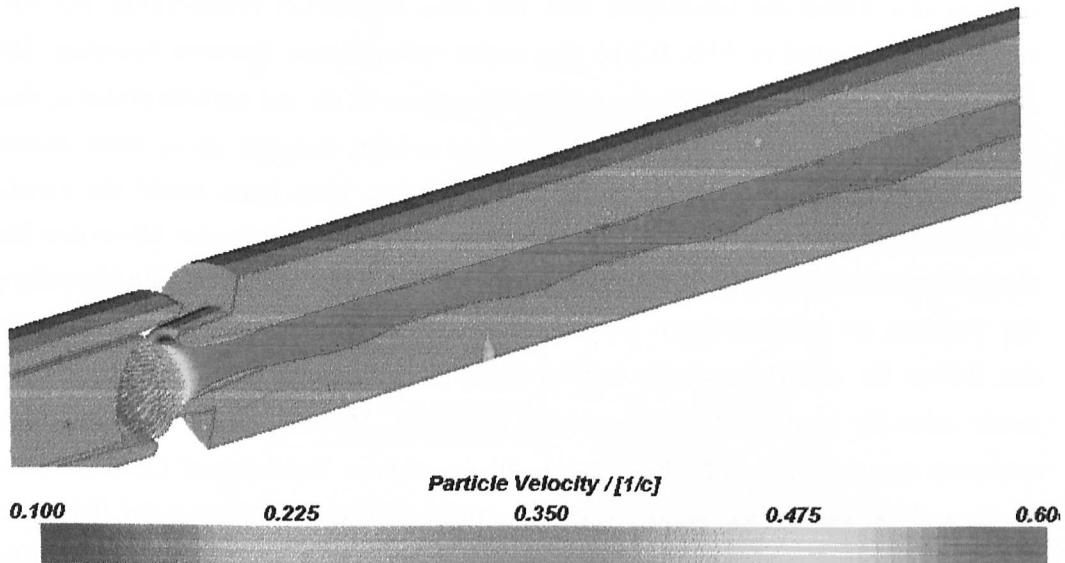
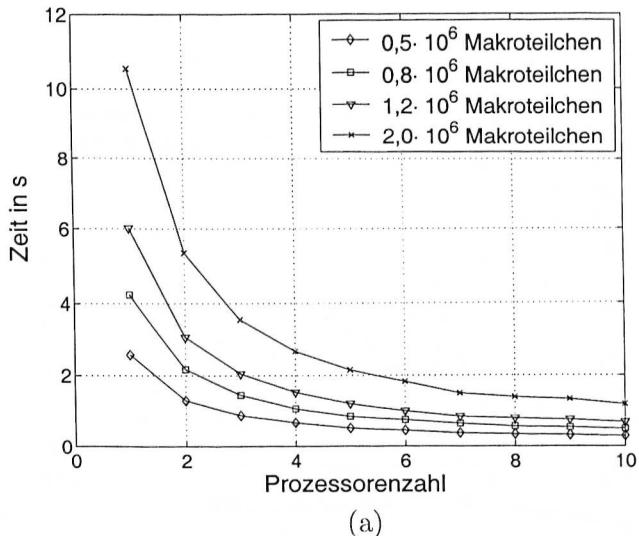
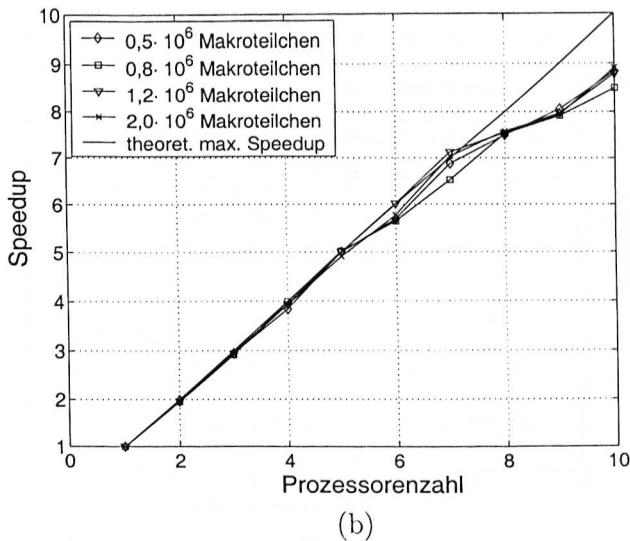


Abbildung 6.2: Der Elektronenstrahl in der Elektronenkanone. Entnommen aus einer der durchgeföhrten Simulationen. Man erkennt, dass die Teilchen nicht direkt auf der Kathode erzeugt werden, sondern ein Stück weiter vorn. Die Eigengeschwindigkeit der emittierten Makroteilchen ist durch unterschiedliche Farben gekennzeichnet.

noch einige Zeitschritte um zu untersuchen, wieviel Zeit für die Ausführung einzelner Programmteile und insbesondere für die Berechnung der neuen Orte und Geschwindigkeiten für die Teilchen benötigt wurde. Zu diesem Zeitpunkt geht der Strahl durch das komplette Rechengebiet. In Abb 6.3(a) sind die Zeiten als Funktion der Prozessorenanzahl aufgetragen. Jede Linie entspricht dabei einer unterschiedlichen Anzahl von Teilchen im gesamten Rechengebiet. Es wurde das Verhalten für eine Teilchenzahl zwischen 500.000 und 2.000.000 auf 1 bis 10 Prozessoren getestet. Man erkennt den Abfall der benötigten Zeit, der etwa umgekehrt proportional der Prozessorenanzahl ist und in Abb. 6.3(b) den damit verbundenen linearen Speedup. Das Abknicken der Kurve oberhalb des achten Prozessors ist darauf zurückzuführen, dass die Teilung der Gebiete immer an Zellgrenzen erfolgt. Folglich ist es nicht immer möglich die Teilchen so auf die Rechner zu verteilen, dass jeder exakt die gleiche Anzahl besitzt. Da sich die Teilchen zudem in wenigen Zellen in der Mitte des Rechengebietes konzentrieren, ist es nochmals schwieriger eine gleichmäßige Verteilung der Teilchen zu gewährleisten. Um den Speedup zu berechnen, wurde daher von den Zeiten für alle Prozessoren stets die größte benutzt, da der langsamste Prozessor stets die Laufzeit des Programmes bestimmt. Danach wurde die Simulation nochmals durchgeführt. Allerdings wurde die Anzahl der Teilchen auf 1.200.000 festgelegt und die Zeiten beobachtet, die der parallele Solver zur Lösung der diskreten Poisson-Gleichung benötigt. Das Bild, das sich dabei ergibt, ist in Abbildung 6.4(a) zu sehen. Es zeigt sich anhand dieses Bildes zwar eine Abhängigkeit des Solvers von der Anzahl der Gitterpunkte der Untergebiete, die Kurven werden aber insbesondere im Bereich weniger Prozessoren durch andere Effekte verfälscht. In Abbildung 6.4(b) ist der entsprechende Speedup eingetragen. Die Summe der Zeiten für die einzelnen Teile des Codes, der in der Zeitschleife durchlaufen wird, ergibt die Zeit, die für einen Schleifendurchlauf benötigt wird. In Abbildung 6.5(a) ist diese Zeit in Abhängigkeit von der Prozessorenanzahl angegeben. In Abbildung 6.5(b) ist der Speedup für diese Kurve aufgetragen. Man erkennt, dass zumindest für die Anzahl an Gitterpunkten in den gewählten Modellen die Laufzeit durch die Lösung der Newton-Lorenz-Gleichung bestimmt wird. Differiert die Anzahl der Gitterpunkten stärker, so ergibt sich ein anderes Bild wie in Abbildung 6.8 zu sehen ist.

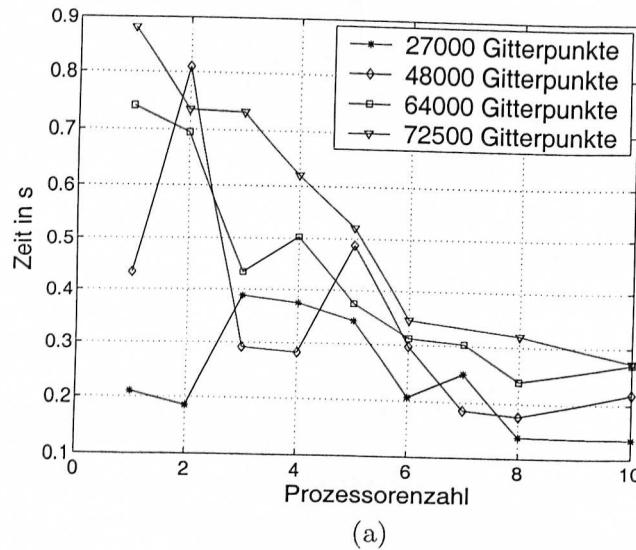


(a)

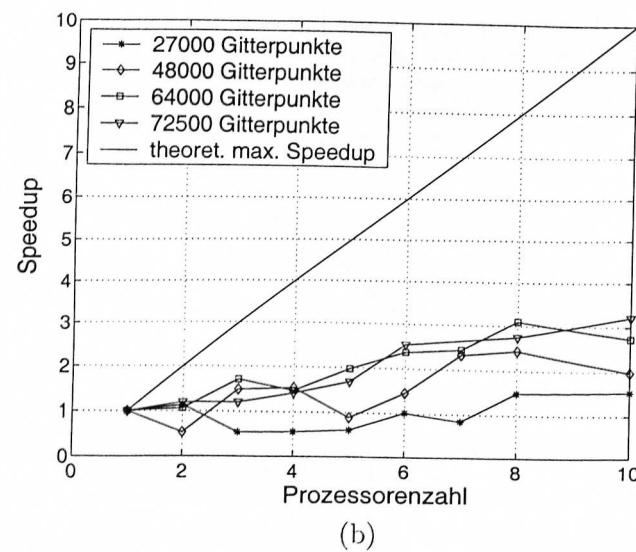


(b)

Abbildung 6.3: Skalierbarkeit der Lösung der Newton-Lorenz-Gleichung. (a) zeigt die Zeit, die für die Lösung der Newton-Lorenz-Gleichung in einem Zeitschritt bei (soweit möglich) vollständig balancierter Last benötigt wird, für Simulationen mit verschiedenen Teilchenzahlen als Funktion der verwendeten Prozessorenzahl. (b) zeigt den Speedup, der sich aus (a) ergibt. Es zeigt sich, eine fast optimale Skalierbarkeit. Der Knick in der Kurve oberhalb von 7 Prozessoren liegt an einer etwas ungleichmäßig aufgeteilten Last.

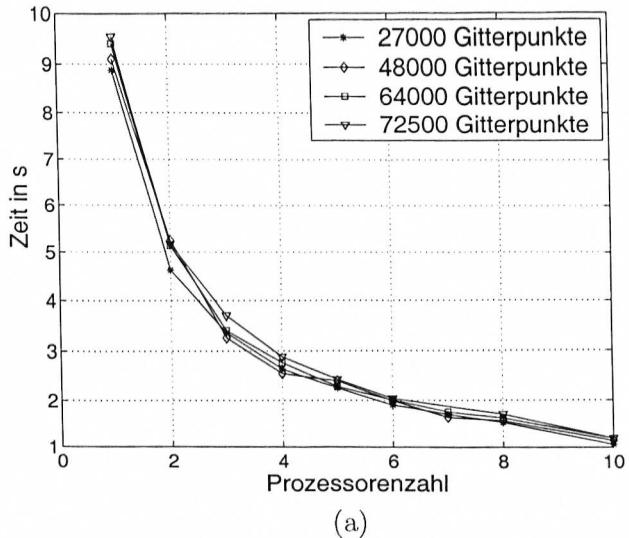


(a)

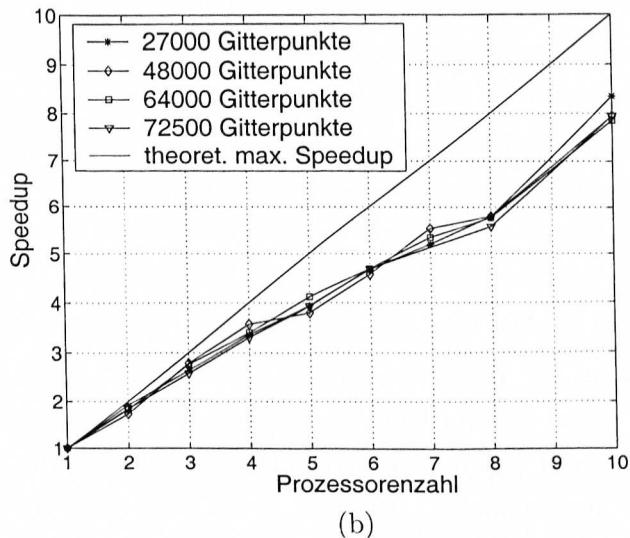


(b)

Abbildung 6.4: Skalierbarkeit des Solvers. (a) zeigt die Zeit, die zur numerischen Lösung der Poisson-Gleichung benötigt wird als Funktion der Prozessorenanzahl für verschiedene Modellgrößen. Man erkennt für geringe Prozessorenanzahl praktisch keine Regelmäßigkeit. Man erkennt jedoch tendenziell die Abnahme der benötigten Zeit. (b) zeigt den Speedup für (a). Trotz gleichgroßer Teilgebiete für jeden Prozessor, ergibt sich eine recht schlechte Skalierbarkeit für den Solver.



(a)



(b)

Abbildung 6.5: Skalierbarkeit der gesamten Zeitschleife für eine konstante Anzahl von einer Million Teilchen. Durch die schlechte Skalierbarkeit des Solvers sind die Kurven etwas vom maximal möglichen Speedup entfernt.

6.3.2 Untersuchung verschiedener Gebietsteilungsalgorithmen

Im Rahmen der Arbeit wurden zwei verschiedene Gebietsteilungsalgorithmen getestet und verglichen. Der erste Algorithmus trifft die Entscheidung, in welche Richtung ($u-, v-$ oder $w-$ Richtung) ein Rechengebiet in einem Schritt unterteilt werden soll, nach der Anzahl der Gitterpunkte in der jeweiligen Raumrichtung und entscheidet sich in der Weise, dass immer in die Richtung unterteilt wird, in der das Rechengebiet die größte Ausdehung besitzt. Das führte bei den durchgeföhrten Simulationen, die in einer Richtung eine sehr viel größere Ausdehnung besitzen als in die beiden anderen, weitgehend zu einer Unterteilung, die transversal zur Ausbreitungsrichtung des Elektronenstrahles beziehungsweise des Ionenstrahles verläuft. Der zweite Algorithmus teilt ein Rechengebiet nacheinander und abwechselnd in alle drei Raumrichtungen und beginnt mit der Unterteilung bei Richtung, in der das Rechengebiet die kleinste Ausdehnung aufweist. Die ersten Unterteilungen erfolgen somit in Ausbreitungsrichtung des Strahles. Um zu überprüfen, welche der beiden Unterteilungen zu bevorzugen ist, wurden mehrere Simulationen für die Elektronenkanone durchgeführt, die sich bis auf den unterschiedlichen Algorithmus für die Gebietsunterteilung nicht unterschieden. Die Simulationen liefen über 600 Zeitschritte und es wurde eine Gesamtzahl von 1,4 Millionen Makroteilchen benutzt. In Abbildung 6.6 ist die Laufzeit des Programmes als Funktion der Prozessorenanzahl aufgetragen. Man erkennt, dass die transversale Teilung für das betrachtete Beispiel klar überlegen ist. Das lässt sich anhand von Abbildung 6.7 plausibel erklären. Das obere Bild zeigt die Teilchenverteilung im Falle einer longitudinalen Gebietsteilung und das untere Bild im Falle der transversalen Teilung. Es zeigt sich, dass sich die Gebiete bei der transversalen Teilung sehr viel gleichmäßiger mit Teilchen füllen als, in der longitudinalen. Das liegt in der Natur des Problemes. Die Teilchen konzentrieren sich in wenigen Gitterzellen in der Mitte des Rechengebietes, da die Gebietsteilung aber immer an den Zellgrenzen erfolgen muss, lässt es sich nicht vermeiden, dass einer der Prozessoren einen „Streifen“ aus Gitterzellen mit einem hohen Gewicht erhält und die Last somit stark unbalanciert ist. Der Effekt wird also mit zunehmend feinerem Rechengitter abnehmen, da dadurch auch die Möglichkeit einer besseren Aufteilung der Teilchen geschaffen wird.

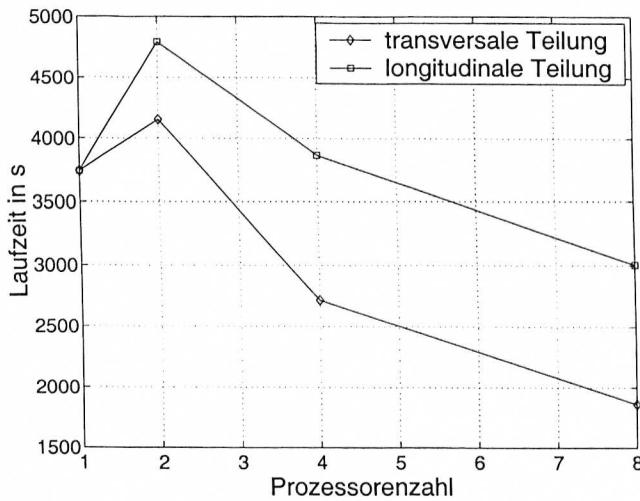


Abbildung 6.6: Laufzeitvergleich zwischen transversaler und longitudinaler Gebietsteilung. Es wurde die Gesamtlaufzeit über 600 Zeitschritte gemessen. Man erkennt in diesem Zusammenhang die Überlegenheit der transversalen Gebietsteilung, die auf einer besseren Lastbalancierung beruht.

6.3.3 Simulation großer Probleme

Ein wesentlicher Aspekt der Arbeit war es zu zeigen, dass mit Hilfe der Parallelisierung große Probleme gelöst werden können, die sich auf einem einzigen Rechner gar nicht mehr simulieren lassen. Um dies zu zeigen, wurden einige große Modelle der Elektronenkanone ausgewählt (500.000 bis 2.000.000 Gitterpunkte) und Simulationen mit 10 Millionen Teilchen durchgeführt. Es wurde über 2.000 Zeitschritte simuliert. In Abbildung 6.8 ist die Laufzeit für die Simulationen aufgetragen. Im Gegensatz zu den kleinen Modellen, die in Abschnitt 6.3.1 benutzt wurden, um die Skalierbarkeit zu messen, zeigt sich die Abhängigkeit der Laufzeit von der Modellgröße hier sehr deutlich.

Über die in Abbildung 6.8 gezeigten Simulationen hinaus, wurde eine weitere Simulation mit 12 Prozessoren und 20 Millionen Teilchen durchgeführt.

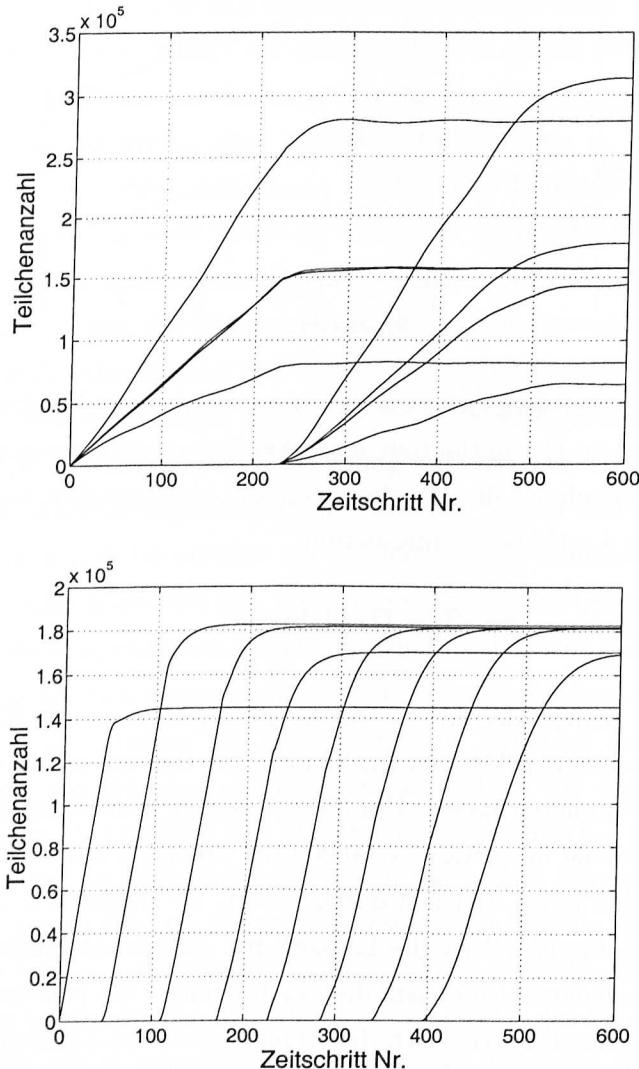


Abbildung 6.7: Vergleich zwischen der Anzahl der Teilchen, die zu den einzelnen Prozessoren gehören. Das obere Bild gehört zu einer longitudinalen Gebietsaufteilung und das untere gehört zu einer transversalen Gebietsaufteilung.

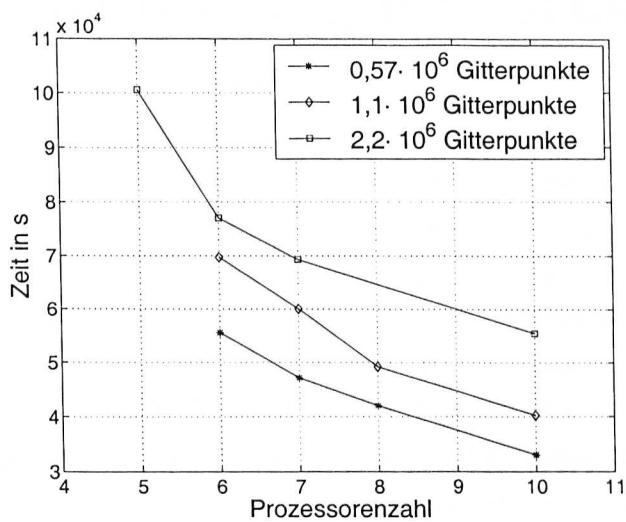


Abbildung 6.8: Laufzeitvergleich mehrerer großer Simulationen mit 10 Millionen Makroteilchen und verschiedener Anzahl von Gitterpunkten.



Kapitel 7

Zusammenfassung und Ausblick

In der vorliegenden Arbeit wurde gezeigt, wie der PIC Algorithmus zur Simulation von Vielteilchensystemen parallelisiert werden kann. Nachdem in Kapitel 2 in aller gebotenen Kürze die physikalischen Grundlagen vorgestellt wurden, die dem PIC Algorithmus zugrundeliegen, wurde in Kapitel 3 der PIC Algorithmus selbst vorgestellt. Anschließend wurden in Kapitel 4 wichtige Grundlagen zur Parallelisierung allgemein erläutert, die zum Verständnis der Parallelisierung für PIC hilfreich sind. In Kapitel 5 wurden schließlich einige Ansätze zur Parallelisierung von PIC aus der Literatur angesprochen. Daraus konnte eine Klassifizierung der einzelnen Ansätze in Form eines Baumdiagrammes abgeleitet, und das für die Parallelisierung im Rahmen dieser Arbeit gewählte Verfahren in den Kontext der übrigen Ansätze eingeordnet werden. Das gewählte Verfahren verbindet die Aufteilung der beiden Schritte des Algorithmus miteinander (Lösung der Newton-Lorenz-Gleichung für alle Teilchen einerseits und Lösung der Poisson-Gleichung zur Berechnung des elektrischen Feldes, das durch die Raumladungen entsteht, andererseits) miteinander. Das bedeutet, dass jedes Teilchen dem Prozessor zugeordnet wird, der auch die Felder für das Gebiet berechnet in dem sich das Teilchen befindet. Es besteht bei der entstandenen Software sowohl die Möglichkeit die Aufteilung des Rechengebietes manuell vorzugeben, oder sie dem Rechner zu überlassen. Wird die Gebietsunterteilung dem Rechner überlassen, so wird das Gebiet hierarchisch auf die zur Verfügung stehenden Prozessoren aufgeteilt und es ergibt sich eine Baumstruktur wie in Abbildung 5.4 gezeigt wurde. Die Unterteilung, die sich dabei ergibt, lässt sich der Familie der orthogonalen Bisektionierungen zurechnen. Durch diese einfache Struktur ist eine Erweiterung

zu einem adaptiven Algorithmus (im Sinne einer Neuverteilung der Last und einer damit verbundenen Neupartitionierung) recht einfach möglich. Einige Überlegungen dazu sind in Abschnitt 5.6 zu finden. Die Überlegungen wurden auch in den entstandenen Code eingearbeitet, konnten aber mangels Zeit nicht mehr ausführlich getestet werden. In Kapitel 6 wurde die entstandene Software anhand zweier Probleme getestet. Zunächst wurde untersucht, wie sich die beiden Teile des Algorithmus (Lösung der Newton-Lorenz-Gleichung für jedes Teilchen und Lösung der Poisson-Gleichung zur Feldberechnung) skalieren lassen, das heißt wie sich die Laufzeiten für einen Zeitschritt als Funktion der Prozessorenzahl bei, soweit wie möglich, vollständig balancierter Last verhält. Dabei zeigte sich, dass sich die Lösung der Newton-Lorenz-Gleichung praktisch ideal skalieren lässt, der theoretisch maximal mögliche Speedup also fast erreicht wurde. Für die Lösung der Poisson-Gleichung mit Hilfe eines parallelen Solvers ergab sich eine etwas schlechtere Skalierbarkeit, wobei bei einer kleiner Prozessorenzahl keine eindeutige Aussage möglich war. Dies relativierte sich etwas bei der Untersuchung großer Modelle in Abschnitt 6.3.3. Dort erkennt man deutlich die Abhängigkeit der Laufzeit als Funktion der Modellgröße bei konstanter Teilchenzahl. Dort zeigt sich auch, dass es mit der entstandenen Software möglich wird Probleme zu simulieren, die mit einem einzelnen Rechner nicht mehr behandelt werden können. Schließlich wurden zwei verschiedene Partitionierungen für das gleiche Problem miteinander verglichen. Dabei zeigte sich die klare Überlegenheit der einen Partitionierung, was mit Hilfe einer besseren Lastbalancierung begründet werden konnte.

Als mögliche Erweiterung der vorhandenen Software wurde bereits in den vorangegangenen Kapiteln eine konsistente Emission genannt. Dabei wird die Emission genau so vorgenommen, dass die emittierten Teilchen das Potenzial genau so modifizieren, dass das geänderte Potenzial genau die vorgenommene Emission bewirken würde. Das lässt sich durch ein iteratives Verfahren erreichen, wie es bereits in den Flussdiagrammen 3.6 und 5.8 angedeutet wurde. Eine weitere mögliche Untersuchung betrifft die adaptive Version der Software. Hier wäre zu prüfen, ob dadurch ein Gewinn bezüglich der Laufzeit zu erzielen ist, beziehungsweise bei welchen Problemen dies möglich ist. Die entwickelte adaptive Version der Software muss bei einer Neupartitionierung sehr große Datenmengen verschicken, da die Teilchen sämtlichst neu verteilt werden müssen. Dadurch sind sehr große Buffer notwendig, um dies zu vermeiden, wäre es wünschenswert, die Kommunikation zwischen den Prozessoren

KAPITEL 7. ZUSAMMENFASSUNG UND AUSBLICK

für den Fall der Neupartitionierung besser zu organisieren, so dass die Kommunikationsbuffer die Größe der maximal an einen Prozessor verschickten beziehungsweise der maximal von einem Prozessor zu empfangenen Teilchen haben können.

Anhang A

Herleitung der Formeln des Leap-Frog Algorithmus

Es soll exemplarisch gezeigt werden, wie man Gleichung (3.4) aus Gleichung (2.17) erhält. Zu Fragen der Konvergenz von Taylorreihen sei auf die Spezialliteratur verwiesen (z.B. [15]). Man geht von der Taylorentwicklung für \mathbf{x} aus¹:

$$\mathbf{x}(t_1) = \sum_{i=0}^{\infty} \frac{1}{i!} \left. \frac{d^i \mathbf{x}}{dt^i} \right|_{t=t_0} \cdot (t_1 - t_0)^i. \quad (\text{A.1})$$

Man drückt nun $\mathbf{x}(t_1)$ und $\mathbf{x}(t_1 + \Delta t)$ mit Hilfe der Taylorreihe um den Entwicklungspunkt $t_1 + \frac{\Delta t}{2}$ aus. Man erhält:

$$\mathbf{x}(t_1) = \sum_{i=0}^{\infty} \frac{1}{i!} \left. \frac{d^i \mathbf{x}}{dt^i} \right|_{t=t_1+\frac{\Delta t}{2}} \cdot \left(-\frac{\Delta t}{2} \right)^i \quad (\text{A.2})$$

und

$$\mathbf{x}(t_1 + \Delta t) = \sum_{i=0}^{\infty} \frac{1}{i!} \left. \frac{d^i \mathbf{x}}{dt^i} \right|_{t=t_1+\frac{\Delta t}{2}} \cdot \left(\frac{\Delta t}{2} \right)^i. \quad (\text{A.3})$$

Fasst man die Terme mit $i \geq 2$ zu einem Term zusammen, so erhält man:

¹ t_0 wird als Entwicklungspunkt der Taylorreihe bezeichnet

$$\mathbf{x}(t_1) = \mathbf{x} \left(t_1 + \frac{\Delta t}{2} \right) + \frac{d\mathbf{x}}{dt} \Big|_{t=t_1+\frac{\Delta t}{2}} \cdot \left(-\frac{\Delta t}{2} \right) + O(\Delta t^2) \quad (\text{A.4})$$

$$\mathbf{x}(t_1 + \Delta t) = \mathbf{x} \left(t_1 + \frac{\Delta t}{2} \right) + \frac{d\mathbf{x}}{dt} \Big|_{t=t_1+\frac{\Delta t}{2}} \cdot \left(\frac{\Delta t}{2} \right) + O(\Delta t^2) \quad (\text{A.5})$$

Subtrahiert man (A.4) von (A.5) so erhält man:

$$\mathbf{x}(t_1 + \Delta t) - \mathbf{x}(t_1) = \frac{d\mathbf{x}}{dt} \Big|_{t=t_1+\frac{\Delta t}{2}} \cdot \Delta t + O(\Delta t^3) \quad (\text{A.6})$$

Division durch Δt ergibt damit eine Abschätzung für die Ableitung von \mathbf{x} an der Stelle $t_1 + \frac{\Delta t}{2}$, die einen Fehlerterm enthält, der quadratisch gegen Null geht:

$$\frac{\mathbf{x}(t_1 + \Delta t) - \mathbf{x}(t_1)}{\Delta t} + O(\Delta t^2) = \frac{d\mathbf{x}}{dt} \Big|_{t=t_1+\frac{\Delta t}{2}} \quad (\text{A.7})$$

Wertet man Gleichung (2.17) an der Stelle $t_1 + \frac{\Delta t}{2}$ aus, so erhält man:

$$\mathbf{v} \left(t_1 + \frac{\Delta t}{2} \right) = \frac{d\mathbf{x}}{dt} \Big|_{t=t_1+\frac{\Delta t}{2}} \quad (\text{A.8})$$

Ersetzt man in (A.8) $\frac{d\mathbf{x}}{dt} \Big|_{t=t_1+\frac{\Delta t}{2}}$ durch den in (A.7) gefundenen Ausdruck, so ergibt sich:

$$\mathbf{v} \left(t_1 + \frac{\Delta t}{2} \right) = \frac{\mathbf{x}(t_1 + \Delta t) - \mathbf{x}(t_1)}{\Delta t} + O(\Delta t^2) \quad (\text{A.9})$$

Notation

- generell sind vektorielle Größen in Fettschrift gedruckt (z.B. \mathbf{x}) und skalare Größen in normalem Zeichensatz (z.B. x).
- Laplace Operator $\Delta := \frac{\partial^2}{\partial u^2} + \frac{\partial^2}{\partial v^2} + \frac{\partial^2}{\partial w^2}$
- Nabla Operator $\nabla := \left(\frac{\partial}{\partial u}, \frac{\partial}{\partial v}, \frac{\partial}{\partial w} \right)$
- $\frac{\partial}{\partial \mathbf{x}} F$, wobei $\mathbf{x} \in \mathbb{R}^N$ und $F : \mathbb{R}^N \rightarrow \mathbb{R}$ bedeutet, dass F nach den Komponenten von x differenziert wird: $\frac{\partial}{\partial \mathbf{x}} F := \left(\frac{\partial}{\partial x_1} F, \dots, \frac{\partial}{\partial x_N} F \right)$
- $\|\cdot\|_2$: euklidische Norm eines Vektors. Es gilt mit $\mathbf{x} \in \mathbb{R}^k$: $\|\mathbf{x}\|_2 := \sqrt{\sum_{i=1}^k x_i^2}$
- Landau Symbol $O(\cdot)$: Es bedeutet für $x \rightarrow a$ (siehe [15]):

$$f(x) = O(g(x)) : \quad \lim_{x \rightarrow a} \frac{f(x)}{g(x)} = A \neq 0.$$

wobei im Zusammenhang klar wird, welche Zahl a ist.

- \sum_i bedeutet, dass über alle zulässigen Werte von i summiert wird. Eine mehrfache Indizierung ist möglich $\sum_{i,j} := \sum_i \sum_j$.
- $\int_G F \, dG$ bedeutet, dass die Funktion F über die Menge G integriert wird. Je nachdem von welcher Dimension die Menge G ist, kann eine mehrfache Integration notwendig sein. Die Schreibweise mit nur einem Integralzeichen ist deshalb als symbolische Schreibweise zu verstehen.

Literaturverzeichnis

- [1] Birdsall, Charles K. *Plasma Physics via computer simulation*, Institute of Physics Publishing, London, 1991
- [2] V. Ivanov, G. Schussmann, M. Weiner *Particle Tracking Algorithms for 3D Parallel Codes in Frequency and Time Domains*,
- [3] Paulett C. Liewer, Viktor K. Decyk *A General Concurrent Algorithm for Plasma Particle-in-Cell Simulation Codes*, Journal of Computational Physics **85**, S.302-322, 1989
- [4] P. MacNeice *Particle-Mesh Techniques* Hughes STX, Goddard Space Flight Center, Greenbelt, 1996
- [5] Charles K. Birdsall, Dieter Fuss *Clouds-in-Clouds, Clouds-in-Cells Physics for Many-Body Plasma Simulation*, Journal of Computational Physics **135**, S. 141-148, 1997
- [6] Alexander Pukhov *High Performance 3D PIC Code VLPL: Virtual Laser Plasma Lab*, Max-Planck-Institut für Quantenoptik, Garching
- [7] Oliver Boine-Frankenheim *Introduction to the Physics of High Current Ion Beams in Accelerators and Storage Rings*, Gesellschaft für Schwerionenforschung GmbH, Darmstadt, 2001
- [8] David W. Walker *Characterizing the Parallel Performance of a Large-Scale, Particle-in-Cell Plasma Simulation Code*, Concurrency: Practice and Experience **2**, S. 257-288, 1990
- [9] H. Vogel *Gerthsen Physik*, Springer, Berlin, 19.Auflage, 1997

LITERATURVERZEICHNIS

- [10] H. Spohn *Dynamics of Charged Particles and Their Radiation Field*, Zentrum Mathematik und Physik Department, Technische Universität München, München, 1999
- [11] T. Weiland *Elektromagnetisches CAD - Numerische Methoden zur Feldberechnung*, Institut für Hochfrequenztechnik, Technische Universität Darmstadt, Darmstadt, 1998
- [12] P. Spelucci *Numerische Mathematik für Ingenieure und Physiker*, Skriptum zur Vorlesung, Technische Universität Darmstadt, Darmstadt, 2000
- [13] T. Weiland *Advances in FIT / FDTD Modeling* Proceedings of the 18th Annual Review of Progress, Monterey, USA, 2003
- [14] J. P. Boris *Relativistic plasma simulation-optimization of a hybrid code*, in Proc. Forth Conv. Num. Sim. Plasmas, S.3-67, Naval Research Lab., Washington D.C., 1970
- [15] Bronstein, Semendjajew, Musiol, Mühlig *Taschenbuch der Mathematik*, Verlag Harri Deutsch, Frankfurt, 4. Auflage, 1999
- [16] M. S. Warren, J. K. Salmon *Astrophysical N-body Simulations Using Hierarchical Tree Data Structures*, in Supercomputing '92, S.570-576, Los Alamitos, 1992. IEEE Comp. Soc.
- [17] E. Gjonaj, T. Lau und T. Weiland *Conformal Modeling of Space-Charge-Limited-Emission from Curved Boundaries in Particle Simulations* Technische Universität Darmstadt, TEMF, 2003
- [18] W. Huber *Paralleles Rechnen - Eine Einführung*, Oldenbourg, Wien, 1997
- [19] Message Passing Interface Forum *MPI: A Message-Passing Interface Standard* University of Tennessee, Knoxville, 1995
- [20] W. H. Kegel *Plasmaphysik - Eine Einführung* Springer, Berlin, 1998
- [21] K. J. Bowers *Accelerating a Particle-in-Cell Simulation Using a Hybrid Counting Sort* Journal of Computational Physics **173**, 393-411, 2001

LITERATURVERZEICHNIS

- [22] P. J. Mardahl, J. P. Verboncoeur *Progress in Parallelizing XOPIC* University of California, Berkeley, USA
- [23] D. W. Walker *Characterizing the Parallel-Performance of a Large-Scale, Particle-In-Cell Plasma Simulation Code* Concurrency: Practice and Experience 2, S.257-288, 1990
- [24] B. Di Martino, S. Briguglio, G. Vlad, P. Sguazzero *Parallel PIC plasma simulation through particle decomposition techniques* Parallel Computing 27, 2001, S.295-314
- [25] J. R. Pilkington, S. B. Baden *Dynamic Partitioning of Non-Uniform Structured Workloads with Spacefilling Curves* University of California, USA, 1995
- [26] R. W. Hockney, J. W. Eastwood *Computer Simulation using Particles* Institute of Physics Publishing, London, 1994
- [27] Robert Sedgewick *Algorithmen* Addison-Wesley, München, 1992
- [28] Claudia Leopold *Parallel and Distributed Computing* John Wiley & Sons Inc., Weinheim, 2001
- [29] PETSc *User's Manual* for Version 2.1.5, Argonne National Laboratory, 2003
- [30] Robert Plato *Numerische Mathematik kompakt* Vieweg, Braunschweig, 2000
- [31] John R. Pierce *My Work With Vacuum Tubes At Bell Laboratories* Vintage Electrics, Volume 3 issue 1 - 1991, sowie http://www.smecc.org/john_r_pierce___electron_tubes.htm
- [32] A. S. Gilmour, Jr. *Principles of Travelling Wave Tubes* Artech House, Boston, 1994
- [33] C. D. Child *Discharge from Hot CaO* Phys. Rev. **32**, 492, 1911
- [34] I. Langmuir *The Effect of Space Charge and Residual Gases on Thermionic Currents in High Vacuum* Phys. Rev. **2**, 450, 1913
- [35] F. Filbet, J.-L. Lemaire, E. Sonnendrücker *Direct axisymmetric Vlasov simulations of space charge dominated Beams*