KfK 4325
Januar 1988

Teilchenfortbewegung in elektro-magnetischen Feldern

T. Westermann
Institut für Datenverarbeitung in der Technik

Kernforschungszentrum Karlsruhe

KERNFORSCHUNGSZENTRUM KARLSRUHE

Institut für Datenverarbeitung in der Technik

KfK 4325

Teilchenfortbewegung in elektro-magnetischen Feldern

Thomas Westermann

Kernforschungszentrum Karlsruhe GmbH, Karlsruhe

Kernforschungszentrum Karlsruhe GmbH
Postfach 3640, 7500 Karlsruhe 1

ISSN 0303-4003

Zusammenfassung

Ein wichtiger Baustein der Particle-in-Cell-Codes zur selbstkonsistenten Simulation von Hochstromdioden ist der Particle Pusher, der die neuen Phasenraumkoordinaten elektrisch geladener Teilchen bei gegebenen elektromagnetischen Feldern über die Lorentzgleichung berechnet. Er gehört zu den rechenzeitintensivsten Teilen des PIC-Codes und bestimmt die Zeitskala des Codes. Ziel dieses Berichtes ist es, grundlegende Eigenschaften der Lorentzgleichung anzuführen, den sog. Cylrad-Algorithmus zur numerischen Lösung dieser Gleichung vorzustellen und Testergebnisse anzugeben. Bei vorgegebenen Genauigkeitsanforderungen an die numerische Rechnung ergibt sich eine entsprechende Zeitskala, die für Elektronen und Ionen gemäß ihrem Massenverhältnis unterschiedlich ist.
Vektorversionen an der Cyber 205 und Siemens VP50 wurden erstellt und Rechenzeiten verglichen.

Particle Movement in Electro-Magnetic Fields

Summary

An important module of electro-magnetic Particle-In-Cell codes is the particle pusher. This module computes the coordinates of electrically charged particles in electro-magnetic fields by solving the Lorentz equation. The time-scale of the PIC code is determined by the particle pusher and it is one of the most CPU-time intensive programs. The aim of this paper is to discuss basic properties of the Lorentz equation, to introduce the so-called Cylrad algorithm for solving this equation numerically and to present some numerical results.
Vector versions on a Cyber 205 and a Siemens VP50 have been implemented and CPU-times are given.

Inhaltsverzeichnis

1. Einleitung S. 1

2. Grundlegende Eigenschaften der Lorentzgleichung S. 2

3. Der Cylrad-Algorithmus S. 6

4. Spezialfall der Rotationssymmetrie S. 9

5. Nichtrelativistischer Grenzfall S. 13

6. Numerische Ergebnisse für den nichtrelativistischen Grenzfall S. 18

7. Graphische Darstellung der Ergebnisse S. 20

8. Numerische Ergebnisse für den relativistischen Fall S. 24

9. Vektorisierungsmöglichkeiten an dem Vektorrechner Cyber 205 S. 26

10. Numerische Ergebnisse an den Vektorrechnern Cyber 205 und Siemens VP50 S. 30

11. Konsequenzen für einen Particle-in-Cell Code S. 31

12. Literatur S. 32

13. Anhang S. 33

1. Einleitung

Um dynamische Vorgänge in elektro-magnetischen Systemen simulieren zu können, werden aufgrund von gegebenen elektro-magnetischen Feldern die Trajektorien von elektrisch geladenen Teilchen gemäß der Lorentzgleichung berechnet. In vielen Fällen genügt es jedoch nicht, nur die lineare, nichtrelativistische Gleichung zu lösen, sondern es muß die nichtlineare, relativistische Bewegungsgleichung zur Berechnung herangezogen werden.

Bei der numerischen Berechnung sollen dann möglichst die Eigenschaften der Lorentzgleichung (z.B. Zeitreversibilität) erhalten bleiben. Diskretisiert man die Bewegungsgleichung mit einem Leapfrog-Schema und berechnet die Lösung der diskretisierten Gleichung mit dem Cylrad-Algorithmus, so wird sowohl die Forderung nach Zeitreversibilität als auch nach Energieerhaltung erfüllt.

Nachdem einige grundlegende Eigenschaften der Lorentzgleichung in Kapitel 2 beschrieben werden, stellen wir den Cylrad-Algorithmus zur Lösung dieser Gleichung in Kapitel 3 vor. Anschließend wird in Kapitel 4 der Spezialfall der Rotationssymmetrie und in Kapitel 5 der nichtrelativistische Grenzfall diskutiert. In Kapitel 6 werden numerische Ergebnisse für den rotationssymmetrischen, nichtrelativistischen Grenzfall beschrieben und in Kapitel 7 graphisch dargestellt. Es schließt sich eine Diskussion der Ergebnisse für den relativistischen Fall in Kapitel 8 an.

Da der gewählte Algorithmus sehr zeitaufwendig ist, wurden Vektorversionen sowohl auf der Cyber 205 als auch auf der VP50 getestet. Da eine optimale Ausnutzung der Cyber 205 ohne spezielle Notation nicht möglich ist, wird in Kapitel 9 das zur Vektorisierung verwendete Cyber-spezifische FORTRAN kurz erläutert. In Kapitel 10 werden Rechenzeiten für die Cyber 205 und VP50 angegeben.

2. Grundlegende Eigenschaften der Lorentzgleichung

Die relativistische Bewegungsgleichung elektrisch geladener Teilchen mit Ladung q in elektro-magnetischen Feldern ist durch die Lorentzgleichung

$$\mathbf{F} = \frac{d(m\frac{d\mathbf{x}}{dt})}{dt} = q(\mathbf{E} + \frac{d\mathbf{x}}{dt} \times \mathbf{B}) \qquad (1a)$$

und den Anfangsbedingungen

$$\mathbf{x}(0) = \mathbf{x}_0 \qquad (1b)$$

$$\frac{d\mathbf{x}}{dt}(0) = \mathbf{v}_0 \qquad (1c)$$

gegeben, wobei

$$m = m_0 \gamma \quad , \qquad \gamma = \frac{1}{\sqrt{1 - \| \frac{d\mathbf{x}}{dt} \|^2 / c^2}} \qquad (1d)$$

und m_0 die Ruhemasse des Teilchens bedeuten. Die Summe aus Ruheenergie und kinetischer Energie ist durch $\varepsilon = m c^2$ bestimmt.

Die Lorentzgleichung ist eine gewöhnliche, nichtlineare Differentialgleichung zweiter Ordnung, die die Eigenschaft der Zeitreversibilität besitzt. D.h. ist eine Bewegung gemäß der Bewegungsgleichung erlaubt, so ist die inverse Bewegung, bei der das System die gleichen Zustände in zeitlich umgekehrter Reihenfolge durchläuft, ebenfalls zulässig. Besser als die Bezeichnung zeitreversibel wäre zeitisotrop, d.h. daß es keine mathematisch ausgezeichnete Zeitrichtung gibt. Da jedoch in der Literatur meist der Begriff zeitreversibel benutzt wird, wollen wir uns dem anschließen. Die inverse Bewegung erhält man, indem man t durch -t ersetzt und in Abweichung zur klassischen Mechanik auch das Vorzeichen von **B** invertiert.

Formal bedeutet dies: Ist **x**(t) Lösung von (1a) - (1c) und definieren wir **y**(t) : = **x**(-t) , so genügt **y**(t) der Bewegungsgleichung

$$\mathbf{F} = \frac{d(m\frac{d\mathbf{y}}{dt})}{dt} = q(\mathbf{E} + \frac{d\mathbf{y}}{dt} \times (-\mathbf{B}))$$

mit

$$\mathbf{y}(0) = \mathbf{x}_0$$

und

$$\frac{d\mathbf{y}}{dt}(0) = -\mathbf{v}_0 .$$

Wir betrachten zunächst den Spezialfall $\mathbf{E} = 0$. Wir führen

$$\mathbf{p}(t) := \gamma(t)\ \frac{d\mathbf{x}}{dt}(t)$$

als neue Variable ein und bestimmen $\mathbf{p}(t)$ aus dem Anfangswertproblem

$$\frac{d\mathbf{p}}{dt} = \frac{q}{m_0}\ (\frac{\mathbf{p}}{\gamma} \times \mathbf{B})$$

$$\mathbf{p}(0) = \gamma(0)\ \mathbf{v}_0 .$$

Ersetzen wir in Gleichung (1d) $\|\mathbf{v}\|^2$ gemäß obiger Definition von $\mathbf{p}(t)$ durch $\|\mathbf{p}\|^2 / \gamma^2$ und lösen

$$\gamma = \frac{1}{\sqrt{1 - \|\mathbf{p}\|^2 / (\gamma^2 c^2)}}$$

nach γ auf, ist γ durch

$$\gamma = \sqrt{1 + (\frac{\|\mathbf{p}\|^2}{c^2})}$$

gegeben. Die Ableitung berechnet sich aus

$$\frac{d\gamma}{dt} = \frac{d}{dt}\sqrt{1 + (\frac{\|\mathbf{p}\|^2}{c^2})} = \frac{1}{2}\ \frac{1}{c^2\,\gamma}\ \frac{d}{dt}\,(\mathbf{p}\ ,\ \mathbf{p}) = \frac{1}{c^2\,\gamma}\ (\frac{d\mathbf{p}}{dt}\ ,\ \mathbf{p})\quad ,$$

wenn wir die Regel

$$\frac{d}{dt}\,(\,\mathbf{p}\,,\,\mathbf{p}\,) \;=\; \frac{d}{dt}\sum_{i=1}^{3} p_i^2 \;=\; 2\sum_{i=1}^{3} p_i\,\frac{d}{dt}\,p_i \;=\; 2\,(\,\frac{d\,\mathbf{p}}{dt}\,,\,\mathbf{p}\,)$$

anwenden.

Aus der Differentialgleichung für **p**(t) folgt, daß

$$(\,\frac{d\,\mathbf{p}}{dt}\,,\;\mathbf{p}\,) \;=\; 0\quad .$$

Damit ist sowohl γ , $m = \gamma\, m_0$ als auch die Energie $\varepsilon = \gamma\, m_0\, c^2$ des Teilchens zeitlich konstant. Insbesondere ist in diesem Spezialfall (1a) eine lineare Differentialgleichung.

Setzt man nun noch **B** = (0, B_y, 0) als zeitlich konstant voraus und nimmt an, daß $v_y(0) = 0$ ist, erhält man als Lösung eine Gyration in der zu **B** senkrechten (x,z)-Ebene mit konstanter Gyrationsfrequenz $\omega = q/m * B_y / \gamma$ und konstanter Energie ε . Die zugehörigen Phasenraumkoordinaten sind:

$$p_x(t) = \cos(\omega * t) * p_x(0) \;-\; \sin(\omega * t) * p_z(0)$$
$$p_z(t) = \sin(\omega * t) * p_x(0) \;+\; \cos(\omega * t) * p_z(0)$$

und mit Division durch γ und anschließender Integration:

$$x(t) = x(0) \;+\; 1/\omega * \sin(\omega * t) * v_x(0) \;+\; 1/\omega * (\cos(\omega * t) - 1) * v_z(0)$$
$$z(t) = z(0) \;-\; 1/\omega * (\cos(\omega * t) - 1) * v_x(0) \;+\; 1/\omega * \sin(\omega * t) * v_z(0)\quad .$$

Um im allgemeinen Fall die Differentialgleichung besser behandeln zu können, transformieren wir die Differentialgleichung zweiter Ordnung in ein System von gekoppelten Differentialgleichungen erster Ordnung, indem wir

$$\mathbf{v}(t) := \frac{d\,\mathbf{x}}{dt}(t)$$

definieren.

Damit sind die Funktionen v(t) und x(t) gesucht, die Lösung von

$$\frac{d(m\mathbf{v})}{dt} = q\,(\mathbf{E} + \mathbf{v} \times \mathbf{B}), \quad \mathbf{v}(0) = \mathbf{v}_0 \qquad (2a)$$

mit

$$m = m_0\,\gamma\,, \qquad \gamma = \frac{1}{\sqrt{1 - \|\mathbf{v}\|^2 / c^2}} \qquad (2b)$$

und

$$\frac{d\mathbf{x}}{dt}(t) = \mathbf{v}(t)\,, \quad \mathbf{x}(0) = \mathbf{x}_0\,. \qquad (3)$$

Setzen wir auch hier zur Abkürzung

$$\mathbf{p}(t) := \gamma(t)\;\mathbf{v}(t)\,,$$

so erhalten wir aus (2a) und (2b)

$$\frac{d\mathbf{p}}{dt} = \frac{q}{m_0}\,(\mathbf{E} + \frac{\mathbf{p}}{\gamma} \times \mathbf{B}), \quad \mathbf{p}(0) = \mathbf{p}_0 = \gamma(0)\;\mathbf{v}(0)\,, \qquad (4)$$

mit

$$\gamma = \sqrt{1 + (\frac{\|\mathbf{p}\|^2}{c^2})}\,.$$

Es sind nun Algorithmen gesucht, die das System (3) und (4) lösen. Die Gleichungen (3) und (4) sind ebenfalls zeitreversibel. Ersetzt man die Funktionen x(t) und p(t) durch Werte an diskreten Zeitintervallen, so sind Approximations-Schemata erforderlich, die diese Eigenschaft beibehalten.

3. Der Cylrad-Algorithmus

Eine Möglichkeit dies zu realisieren bietet das zeitzentrierte Leapfrog-Schema:

$$\frac{\mathbf{p}^{n+1/2} - \mathbf{p}^{n-1/2}}{dt} = \frac{\mathbf{F}^n}{m_0} \qquad (5)$$

$$\frac{\mathbf{x}^{n+1} - \mathbf{x}^n}{dt} = \mathbf{v}^{n+1/2}, \qquad (6)$$

wobei dt der Zeitschritt und n das Zeitlevel ist. Es ist zu beachten, daß die Verwendung des Leapfrog-Algorithmus jedoch nicht die Vorgabe von **p** (n dt), sondern von **p** ((n- 1/2) dt) erfordert.

Im Falle einer elektro-magnetischen Kraft ist

$$\mathbf{F}^n = q\,(\,\mathbf{E}^n + \mathbf{v}^n \times \mathbf{B}^n\,) = q\,(\,\mathbf{E}^n + \mathbf{p}^n \times \frac{\mathbf{B}^n}{\gamma^n}\,) \quad .$$

Der Cylrad-Algorithmus ist ein Schema zur numerischen Lösung von Gleichung (5), das die charakteristischen Eigenschaften einer elektro-magnetischen Kraft berücksichtigt. Nach einer Halbbeschleunigung durch das elektrische Feld $\mathbf{E}^n$ schließt sich eine Drehung des Vektors $\mathbf{p}^n$ an der Achse $\mathbf{B}^n$ an. Eine zweite Halbbeschleunigung durch das elektrische Feld $\mathbf{E}^n$ liefert als Endresultat $\mathbf{p}^{n+1/2}$. Danach können über $\mathbf{p}^{n+1/2}$ die neuen Phasenraumkoordinaten berechnet werden.

Ist also **p** zur Zeit (n - 1/2) * dt und **x** zur Zeit n * dt vorgegeben, und sind die Felder **B** und **E** zur Zeit n * dt am Teilchenort bestimmt, so lautet der relativistische Cylrad-Algorithmus:

1. Halbbeschleunigung:

$$\mathbf{p}_1 = \mathbf{p}^{n-1/2} + \frac{q}{m_0}\,\frac{dt}{2}\,\mathbf{E}^n \qquad (C1)$$

$$\gamma = \sqrt{1 + \frac{\|\mathbf{p}_1\|^2}{c^2}}$$

Drehung:

$$\mathbf{p}_2 = \mathbf{p}_1 + f_1\,\mathbf{p}_1 \times \frac{\mathbf{B}^n}{\gamma} \qquad (C2)$$

$$\mathbf{p}_3 = \mathbf{p}_1 + f_2\,\mathbf{p}_2 \times \frac{\mathbf{B}^n}{\gamma} \qquad (C3)$$

2. Halbbeschleunigung:

$$\mathbf{p}^{n+1/2} = \mathbf{p}_3 + \frac{q}{m_0}\,\frac{dt}{2}\,\mathbf{E}^n \qquad (C4)$$

mit

$$\frac{\beta}{2} = \frac{q}{m_0}\,\frac{dt}{2}\,\frac{\|\mathbf{B}^n\|}{\gamma}$$

$$f_1 = \frac{tan\,(\,\beta/2\,)}{\|\mathbf{B}^n\|}$$

$$f_2 = \frac{sin\,(\,\beta\,)}{\|\mathbf{B}^n\|}$$

und anschließende Koordinatenbestimmung

$$\gamma = \sqrt{1 + \frac{\| \mathbf{p}^{n+1/2} \|^2}{c^2}}$$

$$\mathbf{v}^{n+1/2} = \frac{\mathbf{p}^{n+1/2}}{\gamma} \qquad (C5)$$

$$\mathbf{x}^{n+1} = \mathbf{x}^n + dt \quad \mathbf{v}^{n+1/2} \; . \qquad (C6)$$

Die Gleichungen (C2) und (C3) beschreiben eine Drehung des Vektors $\mathbf{p}_1$ an der Achse $\mathbf{B}$ um den Winkel β . Unter Benutzung von trigonometrischen Formeln gilt

$$f_2 = \frac{2 f_1}{1 + (f_1 \| \mathbf{B}^n \|)^2} \; ,$$

und für die numerische Rechnung entwickeln wir $\tan(\beta/2)$ für kleine Winkel β:

$$tan (\beta/2) \approx \beta/2 + 0.31755 (\beta/2)^3 + 0.2033 (\beta/2)^5 \; .$$

4. Spezialfall der Rotationssymmetrie

Da die meisten zu behandelnden physikalischen Problemstellungen rotationssymmetrisch bezüglich der z-Achse sind, führen wir Zylinderkoordinaten (r, Φ, z) ein. Man kann nun unter der Voraussetzung $\mathbf{B} = (0, B_\Phi, 0)$ bzw. $\mathbf{E} = (E_r, 0, E_z)$ und der Annahme, daß $v_\Phi(0) = 0$, zeigen, daß die Bewegung in der zu **B** senkrechten (r, z)-Ebene stattfindet und (4) sich reduziert zu

$$\frac{dp_r}{dt}(t) = \frac{q}{m_0}\left(E_r(t) - \frac{p_z(t)}{\gamma(t)} B_\Phi(t)\right) \quad , \qquad p_r(0) = p_{r0} \tag{7a}$$

$$\frac{dp_z}{dt}(t) = \frac{q}{m_0}\left(E_z(t) + \frac{p_r(t)}{\gamma(t)} B_\Phi(t)\right) \quad , \qquad p_z(0) = p_{z0} \tag{7b}$$

$$\frac{dr}{dt}(t) = v_r(t) \quad , \qquad r(0) = r_0 \tag{8a}$$

$$\frac{dz}{dt}(t) = v_z(t) \quad , \qquad z(0) = z_0 \quad . \tag{8b}$$

In diesem Spezialfall der Rotationssymmetrie vereinfacht sich der Cylrad-Algorithmus auf die folgenden Formeln, die entsprechend (7a), (7b) und (8a) (8b) in Komponentenschreibweise notiert wurden:

1. Halbbeschleunigung:

$$p_{r1} = p_r^{n-1/2} + \frac{q}{m_0}\,\frac{dt}{2}\,E_r^{\,n}$$

$$p_{z1} = p_z^{n-1/2} + \frac{q}{m_0}\,\frac{dt}{2}\,E_z^{\,n}$$

$$\gamma = \sqrt{1 + \left(\frac{p_{r1}^2 + p_{z1}^2}{c^2}\right)}$$

Drehung:

$$p_{r2} = p_{r1} - f_1 \; p_{z1} \; \frac{B_\Phi^n}{\gamma}$$

$$p_{z2} = p_{z1} + f_1 \; p_{r1} \; \frac{B_\Phi^n}{\gamma}$$

$$p_{r3} = p_{r1} - f_2 \; p_{z2} \; \frac{B_\Phi^n}{\gamma}$$

$$p_{z3} = p_{z1} + f_2 \; p_{r2} \; \frac{B_\Phi^n}{\gamma}$$

2. Halbbeschleunigung:

$$p_r^{n+1/2} = p_{r3} + \frac{q}{m_0} \; \frac{dt}{2} \; E_r^{\,n}$$

$$p_z^{n+1/2} = p_{z3} + \frac{q}{m_0} \; \frac{dt}{2} \; E_z^{\,n}$$

mit

$$\frac{\beta}{2} = \frac{q}{m_0} \; \frac{dt}{2} \; \frac{B_\Phi^n}{\gamma}$$

$$f_1 = \frac{tan\,(\,\beta/2\,)}{B_\Phi^n}$$

$$f_2 = \frac{2\,f_1}{1 + (f_1\,B_\Phi^n)^2}$$

und die anschließende Bestimmung der Koordinaten

$$\gamma = \sqrt{1 + \frac{(p_r^{n+1/2})^2 + (p_z^{n+1/2})^2}{c^2}}$$

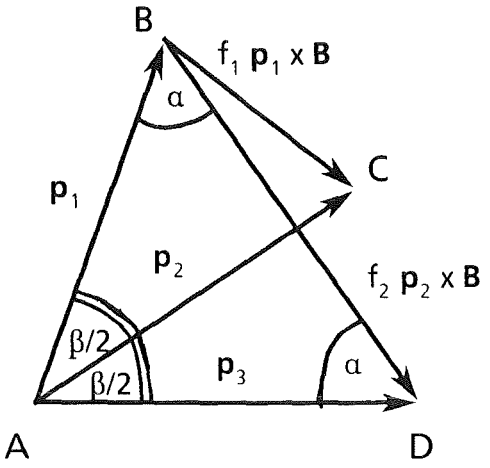
$$v_r^{n+1/2} = \frac{p_r^{n+1/2}}{\gamma}$$

$$v_z^{n+1/2} = \frac{p_z^{n+1/2}}{\gamma}$$

$$r^{n+1} = r^n + dt \quad v_r^{n+1/2}$$

$$z^{n+1} = z^n + dt \quad v_z^{n+1/2} \; .$$

In dem hier beschriebenen Spezialfall, daß **p** senkrecht auf **B** steht, läßt sich die Drehung des Vektors $\mathbf{p}_1$ an der Achse **B** um den Winkel β geometrisch einfach veranschaulichen:



Der erste Teil des Dreh-Algorithmus beschreibt die Drehung des Vektors $\mathbf{p}_1$ um den Winkel $\beta/2$ zum Vektor $\mathbf{p}_2$

$$\mathbf{p}_2 = \mathbf{p}_1 + f \frac{\mathbf{p}_1 \times \mathbf{B}^n}{\| \mathbf{p}_1 \| \| \mathbf{B}^n \|}$$

mit

$$f = BC = tan(\beta/2) \| \mathbf{p}_1 \| .$$

Insbesondere ist damit

$$f_1 = \frac{f}{\| \mathbf{p}_1 \| \| \mathbf{B}^n \|} = \frac{tan(\beta/2)}{\| \mathbf{B}^n \|} .$$

Der zweite Teil des Dreh-Algorithmus besteht aus der Vorschrift

$$\mathbf{p}_3 = \mathbf{p}_1 + f \frac{\mathbf{p}_2 \times \mathbf{B}^n}{\| \mathbf{p}_2 \| \| \mathbf{B}^n \|} .$$

f = BD bestimmt sich aus dem Sinussatz im Dreieck ABD

$$\frac{sin(\beta)}{sin(\alpha)} = \frac{f}{\| \mathbf{p}_1 \|} .$$

Aus dem Dreieck ABC folgt, daß

$$cos(\beta/2) = \frac{\| \mathbf{p}_1 \|}{\| \mathbf{p}_2 \|},$$

womit

$$f = \| \mathbf{p}_1 \| \frac{sin(\beta)}{sin(\alpha)} = \| \mathbf{p}_1 \| \frac{sin(\beta)}{cos(\beta/2)} = sin(\beta) \| \mathbf{p}_2 \|$$

und

$$f_2 = \frac{f}{\| \mathbf{p}_2 \| \| \mathbf{B}^n \|} = \frac{sin(\beta)}{\| \mathbf{B}^n \|} .$$

5. Nichtrelativistischer Grenzfall

Die Voraussetzung für die Gültigkeit des nichtrelativistischen Grenzfalles ist, daß $\mathbf{v}^2 \ll c^2$. Hierbei ist dann (in erster Näherung) $\gamma = 1$, $m = m_0$, $\mathbf{p} = \mathbf{v}$ und

$$\mathbf{F} = \frac{d(m\frac{d\mathbf{x}}{dt})}{dt} = m \frac{d^2\mathbf{x}}{dt^2}$$

zu setzen.

Das nichtlineare Anfangswertproblem (1a) - (1d) geht in ein lineares Anfangswertproblem zweiter Ordnung

$$\frac{d^2\mathbf{x}}{dt^2} = \frac{q}{m}(\mathbf{E} + \frac{d\mathbf{x}}{dt} \times \mathbf{B})$$

mit den Anfangsbedingungen

$$\mathbf{x}(0) = \mathbf{x}_0$$

$$\frac{d\mathbf{x}}{dt}(0) = \mathbf{v}_0 \ .$$

über. Dies ist durch die Definition von

$$\mathbf{v}(t) := \frac{d\mathbf{x}}{dt}(t)$$

äquivalent zum linearen Differentialgleichungssystem erster Ordnung

$$\frac{d\mathbf{v}}{dt} = \frac{q}{m}(\mathbf{E} + \mathbf{v} \times \mathbf{B}) \ , \quad \mathbf{v}(0) = \mathbf{v}_0$$

$$\frac{d\mathbf{x}}{dt}(t) = \mathbf{v}(t) \ , \quad \mathbf{x}(0) = \mathbf{x}_0 \ .$$

Das Leapfrog-Schema erhält die Form

$$\frac{\mathbf{v}^{n+1/2} - \mathbf{v}^{n-1/2}}{dt} = \frac{\mathbf{F}^n}{m} \qquad (9)$$

$$\frac{\mathbf{x}^{n+1} - \mathbf{x}^n}{dt} = \mathbf{v}^{n+1/2}, \qquad (10)$$

wobei dt der Zeitschritt und n das Zeitlevel ist.

Ersetzen wir in Gleichung (9) **v** durch den entsprechenden Ausdruck aus Gleichung (10), gilt

$$\frac{\mathbf{x}^{n+1} - 2\,\mathbf{x}^n + \mathbf{x}^{n-1}}{dt} = \frac{\mathbf{F}^n}{m} .$$

Das Leapfrog-Schema ist in diesem Fall also ein konsistentes Verfahren der Ordnung 2, das um n * dt zentriert ist.

Wir betrachten wieder den Spezialfall der Rotationssymmetrie unter den Voraussetzungen **B** = (0 , B_Φ, 0) bzw. **E** = (E_r , 0 , E_z) und der Annahme, daß $v_\Phi(0) = 0$. Die Bewegung findet unter diesen Bedingungen in der zu **B** senkrechten (r,z)-Ebene statt, und die Bewegungsgleichungen lauten in (r,z)-Koordinaten

$$\frac{dv_r}{dt}(t) = \frac{q}{m}\,(E_r(t) - v_z(t)\;\; B_\Phi(t))\,, \quad v_r(0) = v_{r0} \qquad (11a)$$

$$\frac{dv_z}{dt}(t) = \frac{q}{m}\,(E_z(t) + v_r(t)\;\; B_\Phi(t))\,, \quad v_z(0) = v_{z0} \qquad (11b)$$

zusammen mit den Gleichungen (8a) und (8b).

Wenn wir außerdem annehmen, daß die elektro-magnetischen Felder zeitunabhängig sind, können wir (11a) und (11b) exakt lösen durch

$$v_r(t) = \cos(\omega * t) * v_r(0) - \sin(\omega * t) * v_z(0) + (1./B_\Phi) * (\sin(\omega * t) * E_r + (\cos(\omega * t) - 1) * E_z) \qquad (12a)$$

$$v_z(t) = \sin(\omega * t) * v_r(0) + \cos(\omega * t) * v_z(0) + (1./B_\Phi) * (-(\cos(\omega * t) - 1) * E_r + \sin(\omega * t) * E_z) , \qquad (12b)$$

wobei $\omega = q/m * B_\Phi$ und $\mathbf{x}(t)$ sich aus

$$\mathbf{x}(t) = \mathbf{x}(0) + \int_0^t \mathbf{v}(\tau)\, d\tau \qquad (13)$$

ergibt.

Integriert man Gleichung (13), so ist die Lösung in (r, z)-Koordinaten gegeben durch

$$r(t) = r(0) + 1/\omega * \sin(\omega * t) * v_r(0) + 1/\omega * (\cos(\omega * t) - 1) * v_z(0) - q/m * 1/\omega^2 * (\cos(\omega * t) - 1) * E_r + q/m * 1/\omega^2 * (\sin(\omega * t) - \omega * t) * E_z \qquad (14a)$$

$$z(t) = z(0) - 1/\omega * (\cos(\omega * t) - 1) * v_r(0) + 1/\omega * \sin(\omega * t) * v_z(0) - q/m * 1/\omega^2 * (\sin(\omega * t) - \omega * t) * E_r - q/m * 1/\omega^2 * (\cos(\omega * t) - 1) * E_z . \qquad (14b)$$

Im Grenzfall $B_\Phi \to 0$ bleibt die Lösung (12a), (12b) und (14a), (14b) beschränkt, da für festes t die folgenden Formeln gelten:

$$\lim_{B_\Phi \to 0} \frac{1}{B_\Phi} \sin\left(\frac{q}{m} B_\Phi t\right) = \lim_{B_\Phi \to 0} \frac{q}{m} t \cos\left(\frac{q}{m} B_\Phi t\right) = \frac{q}{m} t ,$$

$$\lim_{B_\Phi \to 0} \frac{1}{B_\Phi} \left(\cos\left(\frac{q}{m} B_\Phi t\right) - 1\right) = - \lim_{B_\Phi \to 0} \frac{q}{m} t \sin\left(\frac{q}{m} B_\Phi t\right) = 0$$

und

$$\lim_{B_\Phi \to 0} \frac{1}{B_\Phi^2} \left(\cos\left(\frac{q}{m} B_\Phi t\right) - 1\right) = - \lim_{B_\Phi \to 0} \frac{1}{2 B_\Phi} \frac{q}{m} t \sin\left(\frac{q}{m} B_\Phi t\right) = - \left(\frac{q}{m}\right)^2 \frac{t^2}{2} ,$$

$$\lim_{B_\Phi \to 0} \frac{1}{B_\Phi^2} \left(\sin\left(\frac{q}{m} B_\Phi t\right) - \frac{q}{m} B_\Phi t\right) = \lim_{B_\Phi \to 0} \frac{1}{2 B_\Phi} \left(\frac{q}{m} t \cos\left(\frac{q}{m} B_\Phi t\right) - \frac{q}{m} t\right) = 0 .$$

Die Lösung reduziert sich im Grenzfall $B_\Phi \to 0$ zu

$$r(t) = r(0) + t * v_r(0) + t^2/2 * q/m * E_r$$

$$z(t) = z(0) + t * v_z(0) + t^2/2 * q/m * E_z \quad .$$

Um Aussagen über die Genauigkeit des Cylrad-Algorithmus zu gewinnen, vergleichen wir die über das Leapfrog-Schema numerisch berechneten Lösungen der nichtrelativistischen Gleichungen (11) und (8) mit den exakten Werten (12a), (12b) bzw. (14a), (14b). Hierbei ist allerdings zu beachten, daß zunächst **v**(0) aus **v**(- dt/2) berechnet werden muß. Dies kann durch Anwendung von Gleichungen (12a) und (12b) geschehen.

Seien $v_{num}^{n+1/2}$ bzw. x_{num}^{n+1} die zum Zeitpunkt (n + 1/2) * dt bzw. (n + 1) * dt über das Leapfrog-Schema berechneten numerischen Lösungen der Gleichungen (11) bzw. (8) und v_{ex} bzw. x_{ex} die exakten Lösungen zu den entsprechenden Zeiten.

Ist $\| \|_2$ die L_2-Norm, $\| \|_\infty$ die Maximumsnorm, und sind **v** (- dt/2) und **x**(0) die vorgegebenen Anfangsdaten, so führen wir den mittleren relativen Geschwindigkeitsfehler

$$\Delta v = \frac{1}{k} * \sum_{n=0}^{k} \frac{\| \mathbf{v}_{ex}((n+1/2)\,dt) - \mathbf{v}_{num}^{n+1/2} \|_2}{\| \mathbf{v}_{ex}((n+1/2)\,dt) - \mathbf{v}(-dt/2) \|_2}$$

und den mittleren relativen Ortsfehler

$$\Delta x = \frac{1}{k} * \sum_{n=0}^{k} \frac{\| \mathbf{x}_{ex}((n+1)\,dt) - \mathbf{x}_{num}^{n+1} \|_\infty}{\| \mathbf{x}_{ex}((n+1)\,dt) - \mathbf{x}(0) \|_\infty}$$

in Abhängigkeit der Zeitschrittweite dt und der Anzahl der Zeitschritte k ein.

6. Numerische Ergebnisse für den nichtrelativistischen Grenzfall

Gesucht ist ein Zeitschritt dt, bei dem die Fehler Δv und Δx einen vorgegebenen Toleranzwert nicht überschreiten.

Da eine charakteristische Größe für eine Zeitskala bei einer Teilchenbewegung in elektro-magnetischen Feldern durch die inverse Zyklotronfrequenz $1/\omega$ mit

$$\omega = \frac{q}{m} \| \mathbf{B} \| .$$

gegeben ist, führen wir den Parameter L durch

$$L := \omega \, dt$$

ein.

In unseren Testfällen wurde zur Berechnung der numerischen Lösungen der Gleichungen (11) und (8) der nichtrelativistische Cylrad-Algorithmus eingesetzt (vgl. Anhang A1). Es wurden die Felder als zeitlich konstant angenommen und E_r = 0 V/m, E_z = 20 MV/m, B_Φ = 2 T , die Anfangsdaten zu Null und q/m = 8.794 * 10^{-7} C/kg gesetzt. In Abhängigkeit von L ergibt sich ein Zeitschritt von dt = L * 1.13 * 10^{-9} s. Die Anzahl der Zeitschritte k wurde so gewählt, daß das Teilchen zweimal oszillierte. Mit diesen Daten erhielten wir die folgende Ergebnisse:

L	Δx in %	Δv in %	Schritte k
0.2	0.33	0.34	62
0.3	0.74	1.23	42
0.4	1.30	1.47	32
0.5	2.01	2.33	26
0.6	2.88	2.83	21
0.7	3.94	7.19	18

Die Rechnungen auf der Siemens M 7890 führten wir sowohl mit einfacher als auch mit doppelter Rechengenauigkeit durch, wobei die obige Tabelle in beiden Fällen gültig ist. Insbesondere ist es in Bezug auf den Fehler der numerischen Lösung, die über den Cylrad-Algorithmus berechnet wurde, verglichen mit der exakten Lösung gerechtfertigt, mit einfacher Genauigkeit zu rechnen.

In der graphischen Darstellung der Ergebnisse (Fig. 1 - 8) wurden für verschiedene Werte von L die Trajektorien der numerischen Lösung und der exakten Lösung aufgezeichnet. Die Abszisse entspricht der z-Komponente und die Ordinate der r-Komponente. Anhand dieser Darstellung erkennt man deutlich, daß mit wachsendem L die numerisch berechnete Lösung der exakten hinterherhinkt. Insbesondere gyrieren die Teilchen mit einer falschen Frequenz.

Für L = 0.2 erhält man die relativen Fehler für Δx von 0.33% und für Δv von 0.34%, was im Hinblick auf die erforderliche Genauigkeit akzeptabel erscheint. Für die numerische Rechnung bedeutet dies, daß ca. 31 Werte für eine Gyration benötigt werden. Mit einem B_Φ-Feld von 2T entspricht dies für nichtrelativistische Teilchen einer Zeitschrittweitenbeschränkung von $dt = 1.136 * 10^{-9}$ s.

In der folgenden Tabelle geben wir durchschnittliche CPU-Zeiten für den nicht-relativistischen Cylrad-Algorithmus an. Die für die Cyber 205 und Siemens VP50 bestimmten CPU-Zeiten gelten jeweils für die Skalarunit.

	REAL*4	REAL*8
Siemens M7890	2.8 µsec/Teilchen	3.3 µsec /Teilchen
IBM / 3090	4.3 µsec/Teilchen	4.6 µsec/Teilchen
Cyber 205	-	3.6 µsec/Teilchen
Siemens VP50	2.9 µsec/Teilchen	3.5 µsec/Teilchen

7. Graphische Darstellung der Ergebnisse

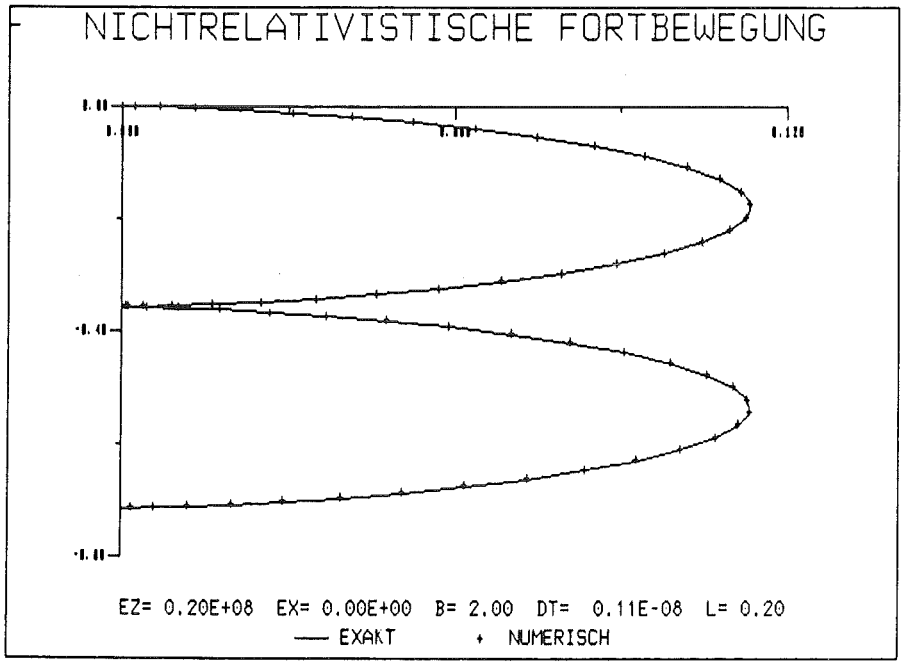


Fig. 1: Vergleich der mit dem Cylrad-Algorithmus numerisch berechneten Lösung mit der exakten Lösung der nichtrelativistischen Lorentzgleichung für L = 0.2.

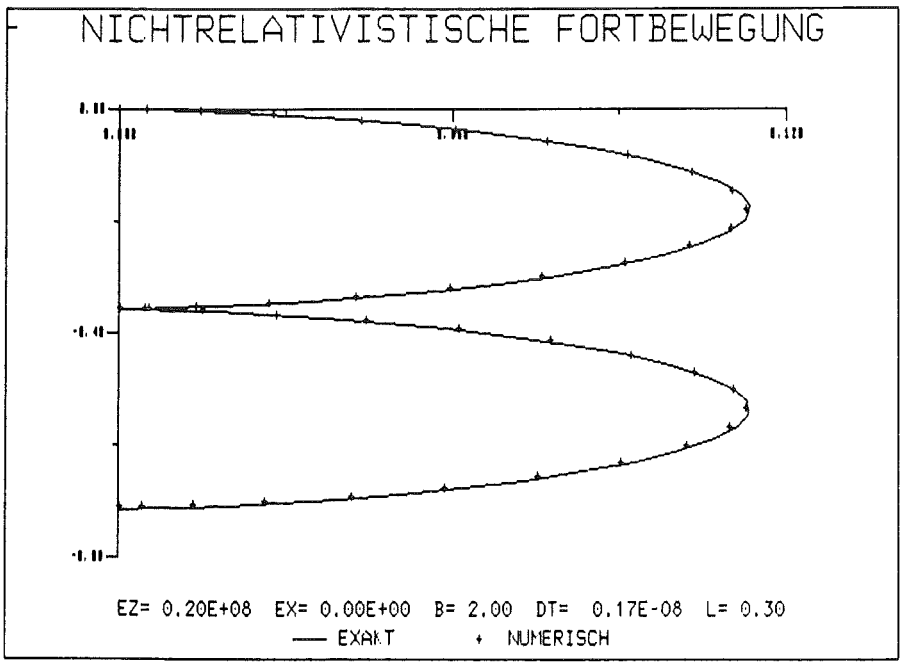


Fig. 2: Vergleich der mit dem Cylrad-Algorithmus numerisch berechneten Lösung mit der exakten Lösung der nichtrelativistischen Lorentzgleichung für L = 0.3.

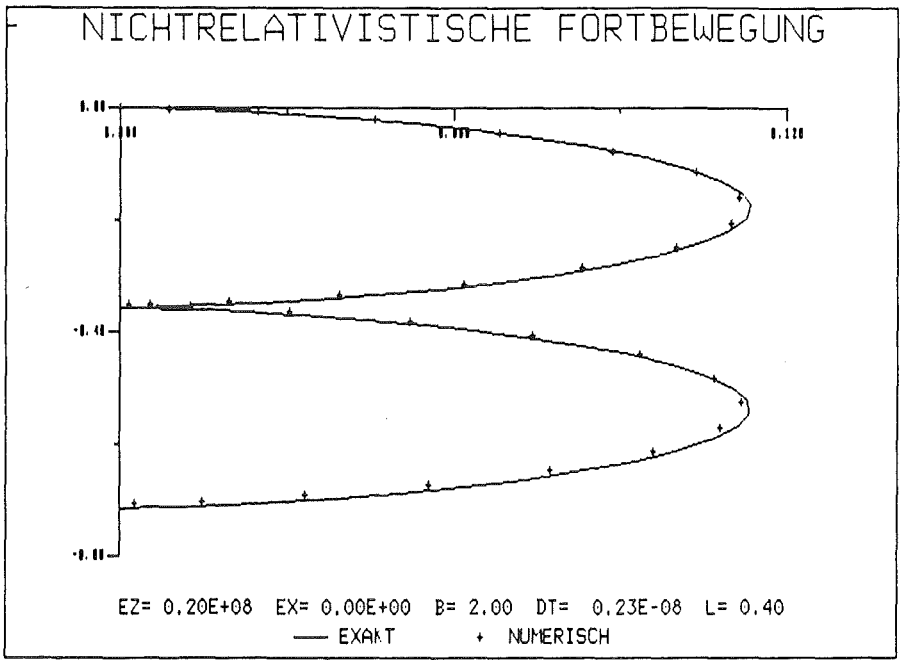


Fig. 3: Vergleich der mit dem Cylrad-Algorithmus numerisch berechneten Lösung mit der exakten Lösung der nichtrelativistischen Lorentzgleichung für L = 0.4.

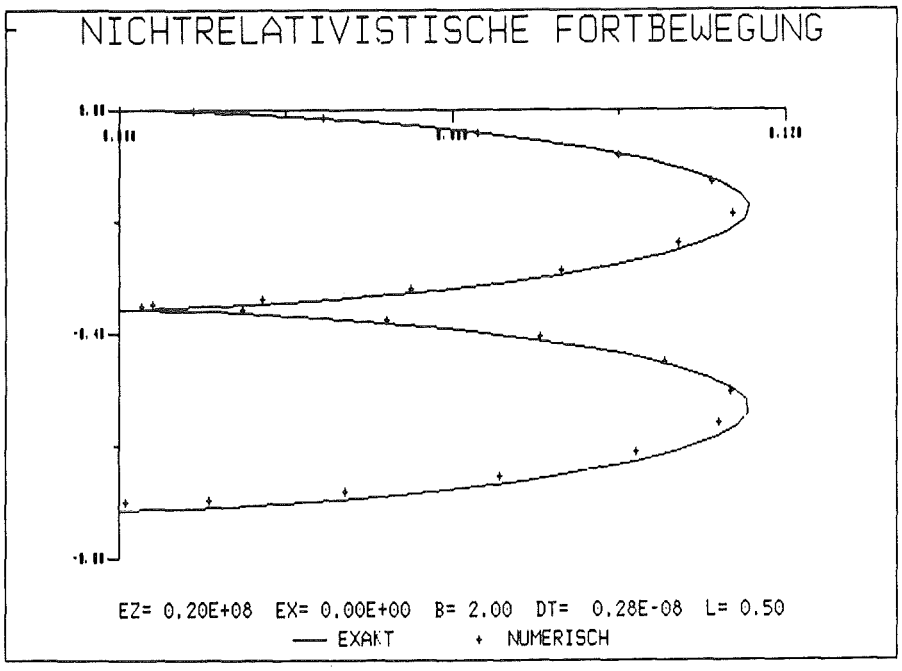


Fig. 4: Vergleich der mit dem Cylrad-Algorithmus numerisch berechneten Lösung mit der exakten Lösung der nichtrelativistischen Lorentzgleichung für L = 0.5.

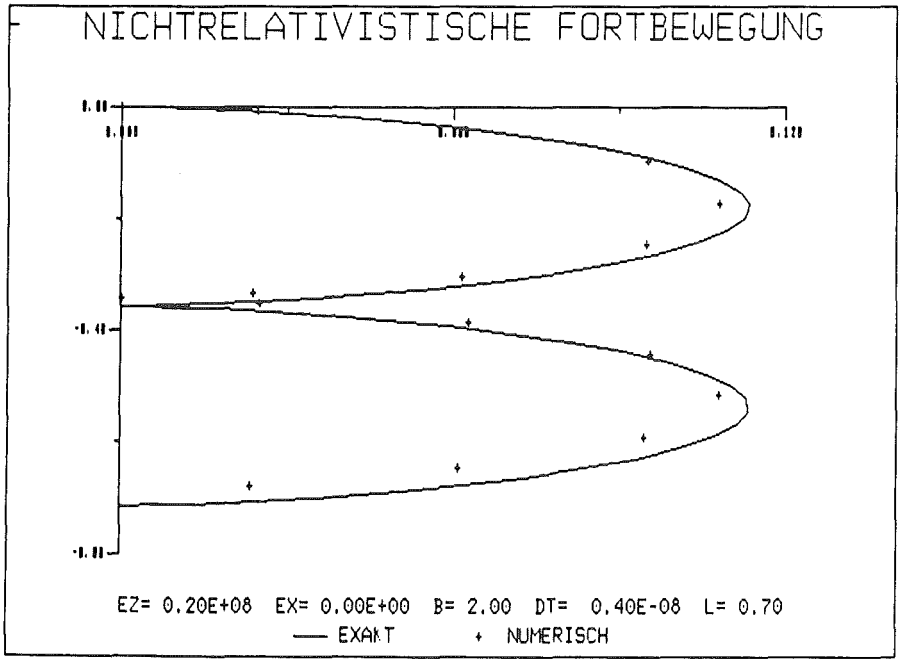


Fig. 5: Vergleich der mit dem Cylrad-Algorithmus numerisch berechneten Lösung mit der exakten Lösung der nichtrelativistischen Lorentzgleichung für L = 0.7.

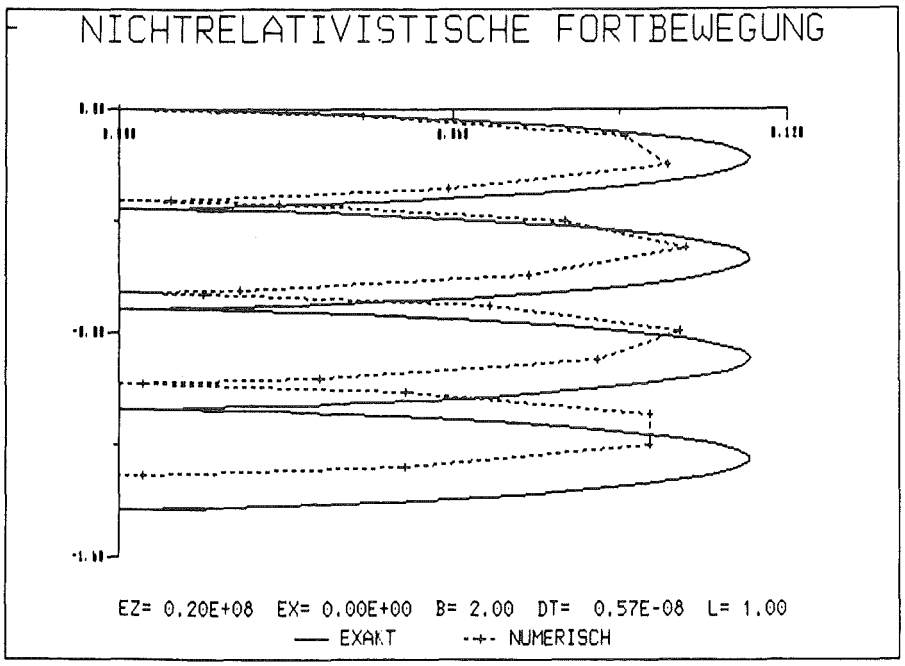


Fig. 6: Vergleich der mit dem Cylrad-Algorithmus numerisch berechneten Lösung mit der exakten Lösung der nichtrelativistischen Lorentzgleichung für L = 1.0.

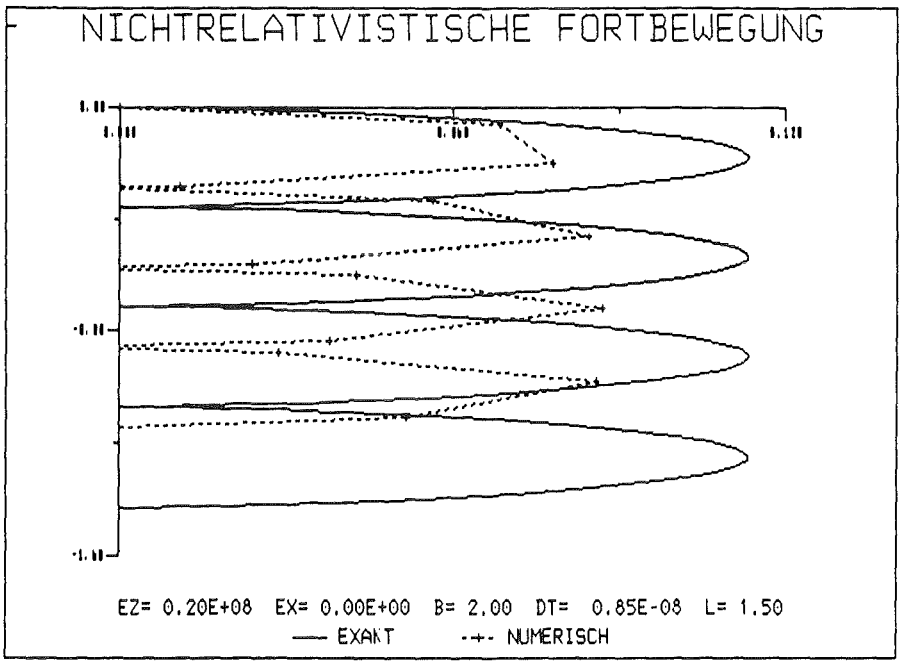


Fig. 7: Vergleich der mit dem Cylrad-Algorithmus numerisch berechneten Lösung mit der exakten Lösung der nichtrelativistischen Lorentzgleichung für L = 1.5.

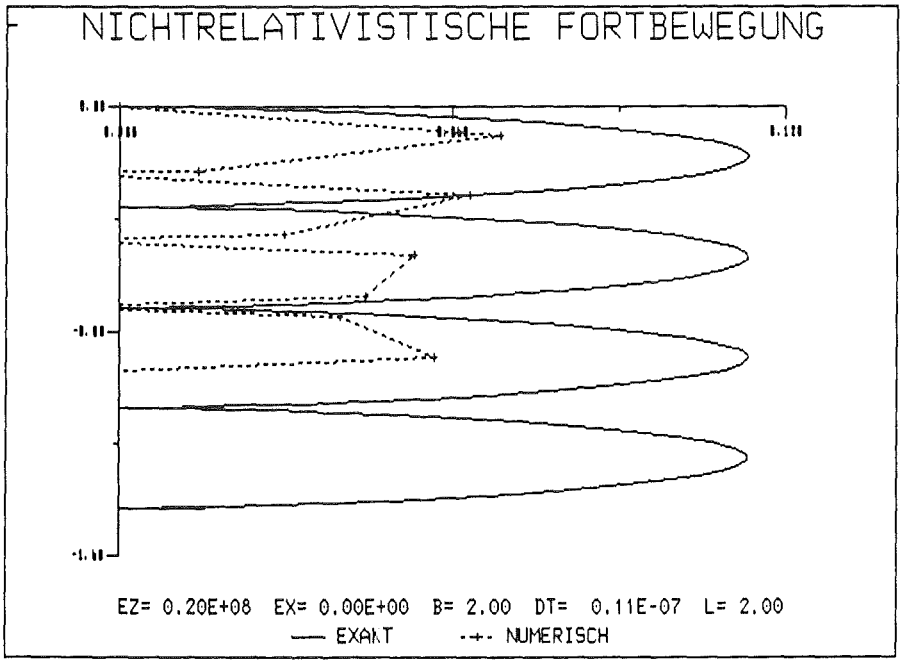


Fig. 8: Vergleich der mit dem Cylrad-Algorithmus numerisch berechneten Lösung mit der exakten Lösung der nichtrelativistischen Lorentzgleichung für L = 2.0.

8. Numerische Ergebnisse für den relativistischen Fall

In unseren Testfällen zur Berechnung der numerischen Lösungen der Gleichungen (7) und (8) wurde der relativistische Cylrad-Algorithmus gewählt.

Da für die relativistischen Teilchen keine analytische Lösung der relativistischen Lorentzgleichung existiert, sind wir nicht in der Lage, die numerisch ermittelten Lösungen mit exakten Lösungen direkt zu vergleichen, dennoch können wir in Spezialfällen Aussagen machen.

Für $B_\phi = 0$ T liefert der Cylrad-Algorithmus den richtigen elektrostatischen Teilchenpusher und für B_Φ = const. und $\mathbf{E} = 0$ V/m liefert er (mit L = 0.2) eine Zyklotronfrequenz, deren Abweichung von der exakten unterhalb des ‰-Bereichs liegt.

Der Algorithmus erhält bei der Drehung die kinetische Energie, und durch die Aufspaltung in eine erste Halbbeschleunigung vor der Drehung und einer zweiten nach der Drehung wird Gleichung (5) zeitreversibel ersetzt.

Würde man versuchen den Rechenaufwand des vorhandenen Cylrad-Algorithmus dadurch zu verringern, indem man die zwei Halbbeschleunigungen durch eine Vollbeschleunigung vor oder nach der Drehung ersetzt, müßte man auf die Eigenschaft der Zeitreversibilität verzichten.

Denn führen wir z.B. eine Vollbeschleunigung vor der Drehung durch ($\mathbf{p} \rightarrow \mathbf{p}_1$), so besitzt die anschließende Drehung ($\mathbf{p}_1 \rightarrow \mathbf{p}_2$) die Drehfrequenz

$$\omega_1 = q/m * B_\Phi / \gamma (\| \mathbf{p}_1 \|).$$

Insbesondere ist $\| \mathbf{p}_1 \| = \| \mathbf{p}_2 \|$. Invertieren wir nun den Prozeß (d.h. $dt \rightarrow -dt$, $\mathbf{B} \rightarrow -\mathbf{B}$, $\mathbf{p} \rightarrow -\mathbf{p}$), schließt sich der negativen Vollbeschleunigung ($-\mathbf{p}_2 \rightarrow \mathbf{p}_3$ mit $\| \mathbf{p}_3 \| \neq \| \mathbf{p}_2 \|$) eine Drehung ($\mathbf{p}_3 \rightarrow \mathbf{p}_4$) mit der Drehfrequenz

$$\omega_2 = q/m * B_\Phi / \gamma (\| \mathbf{p}_3 \|)$$

an. Hierbei ist aber $\omega_2 \neq \omega_1$, da $\| \mathbf{p}_3 \| \neq \| \mathbf{p}_1 \|$!

Im Grenzfall für nichtrelativistische Teilchen liefert der relativistische Algorithmus analoge Ergebnisse wie der nichtrelativistische. Prinzipiell könnte man damit sowohl für relativistische als auch für nichtrelativistische Teilchen den relativistischen Cylrad-Algorithmus wählen, was uns jedoch im Hinblick auf die recht unterschiedlichen CPU-Zeiten als nicht effektiv erscheint.

Übernehmen wir die Zeitschrittweitenrestriktion $dt = 0.2 / \omega$ aus dem nichtrelativistischen Fall, erhalten wir bei $B_\Phi = 2\ T$ und $q/m = 1.7588 * 10^{-11}$ C/kg eine obere Beschränkung des maximalen Zeitschrittes von $dt = 5.685 * 10^{-13}$ s.

In der folgenden Tabelle geben wir durchschnittliche CPU-Zeiten für den relativistischen Cylrad-Algorithmus an. Die für die Cyber 205 und Siemens VP50 bestimmten CPU-Zeiten gelten jeweils wieder für die Skalarunit.

	REAL*4	REAL*8
Siemens M7890	8.5 µsec/Teilchen	10.9 µsec /Teilchen
IBM / 3090	10.6 µsec/Teilchen	12.8 µsec/Teilchen
Cyber 205	-	10.2 µsec/Teilchen
Siemens VP50	8.3 µsec/Teilchen	10.5 µsec/Teilchen

9. Vektorisierungsmöglichkeiten an dem Vektorrechner Cyber 205

Eine prinzipielle Möglichkeit der Vektorisierung eines FORTRAN-Programmes an der Cyber 205 ist durch die automatische Vektorisierung gegeben. Durch die Compiler-Anweisung
FTN200,OPT = V.
wird der Compiler veranlaßt zu prüfen, ob DO-Schleifen des Programms vektorisierbar sind und führt dann gegebenenfalls die Vektorisierung durch.

Falls die zu vektorisierenden DO-Schleifen nur Wertzuweisungen enthalten, nur Daten vom Typ REAL und INTEGER benutzt werden, die Laufvariablen vom Typ INTEGER sind, keine Rekursionen vorliegen und keine IF-Abfragen auftreten, sind die wichtigsten Voraussetzungen für eine automatische Vektorisierung gegeben. Allerdings muß dabei die Anzahl der Schleifendurchläufe als Parameter im entsprechenden Programmteil auftreten und kleiner gleich der maximalen Vektorlänge (65536) sein.

Da es jedoch das Ziel ist, die Teilchenzahl als Parameter im Hauptprogramm festzulegen und den Particle Pusher als Subroutine aufzurufen, hat der vorhandene Compiler nicht die Möglichkeit zu prüfen, ob N kleiner der maximalen Vektorlänge ist. In diesem Fall vektorisiert er die Schleife nicht.

Einen Ausweg bietet die Erweiterung des Compiler-Aufrufes durch
UNS = 1.
Hierbei hat der Benutzer selbst darauf zu achten, daß die Zahl der Schleifendurchläufe nicht größer als 65536 wird. Falls sie dennoch diese Maximalgröße überschreiten, liefert die Cyber 205 fehlerhafte Ergebnisse.

Die automatische Vektorisierung hat den Vorteil, daß sich auch für Programme, die in Standard-FORTRAN geschrieben sind, trotzdem der Vorteil bietet, die Möglichkeiten des Vektorrechners auszunutzen.

Die wichtigste Voraussetzung, nämlich, daß keine IF-Statements in der DO-Schleife enthalten sind, ist in unseren Fällen i.a. jedoch nicht erfüllt. Deshalb reicht die automatische Vektorisierung nicht aus, um ein optimales Programm

für die Cyber 205 zu erstellen. Man benötigt dazu ein Cyber-spezifisches FORTRAN 8X, das erweiterte Sprachelemente für Vektoren besitzt. Im folgenden stellen wir kurz einige Erweiterungen vor, die in unseren Programmen benutzt werden.

Als erstes sei die explizite Vektornotation angeführt. Ist der Vektor $(v_j)_{j = 1,\ldots,100}$ durch die Vereinbarung
DIMENSION V(100)
festgelegt, so bezeichnet
V(I;K)
den Vektor, der bei v_i beginnt und k Elemente umfaßt ($1 \leqq i \leqq 100, k + i \leqq 101$).

Neben dieser expliziten Vektornotation lassen sich Vektoren auch durch sog. Deskriptoren beschreiben. Ein Deskriptor ist ein Zeiger auf einen Vektor, der die Anfangsadresse und die Länge des Vektors enthält. Der Vereinbarungsteil lautet:
DESCRIPTOR VD
DIMENSION VD(100)
Durch eine ASSIGN-Anweisung
ASSIGN VD , V(1;100)
wird dem Deskriptor während der Programmausführung ein Wert zugewiesen. Nach der ASSIGN-Anweisung zeigt der Deskriptor auf den Vektor V(1;100). Jedoch können mit einem Deskriptor nur Vektoroperationen durchgeführt werden, und der Zugriff auf einzelne Komponenten ist nicht ohne weitere Hardware-Anweisungen möglich.

Der Vorteil der Deskriptoren besteht darin, daß man sie durch
ASSIGN VD, .DYN. N
als temporäre Vektoren der Länge N anlegen kann.
Temporäre Vektoren sind notwendig, wenn z.B. Zwischenergebnisse noch einmal benutzt werden sollen. Nach Beendigung der Programmeinheit werden alle in dem Programmteil definierten temporären Vektoren wieder gelöscht.

Um DO-Loops, die IF-Statements enthalten, explizit zu vektorisieren, steht die WHERE-Anweisung zur Verfügung. Sie entspricht der Block-IF-Anweisung, z. B.:

```
WHERE(A(1;N).NE.0.)
   B(1;N) = 1./A(1;N)
END WHERE
```

oder

```
WHERE(A(1;N).NE.0.)
   B(1;N) = 1./A(1;N)
OTHERWISE
   B(1;N) = 0.
END WHERE
```

Dabei wird die Vektorisierung innerhalb des WHERE-Blocks für alle Vektorelemente berechnet, aber nur für die Vektorelemente, bei denen die WHERE-Bedingung erfüllt ist, wird das entsprechende Element im Ergebnisvektor überschrieben.

Als letzte vektorbezogene Spracherweiterung sollen hier die Standard-Vektorfunktionen genannt werden. Fast jede FORTRAN-Standard-Funktion besitzt auf der Cyber eine Vektorversion, die dadurch gekennzeichnet ist, daß vor den spezifischen Funktionsnamen ein 'V' gesetzt und die formale Parameterliste um ein Argument '*' ergänzt wird (Bsp: VTAN (; *)). Beim Aufruf einer Vektorfunktion wird der '*' in der Parameterliste durch den Ergebnisvektor, einen Hilfsvektor oder durch einen INTEGER-Ausdruck ersetzt. Im letzten Fall wird das Ergebnis dann in einen temporären Vektor der angegebenen Länge geschrieben und erst danach in den Ergebnisvektor umkopiert. Dazu das folgende Beispiel:

```
DIMENSION X(100)
DESCRIPTOR GD
ASSIGN GD, .DYN. 100
GD = VSIN( X(1;100)**2 ; GD )
```

Die einzige Möglichkeit, bei einer Cyber 205 die volle Leistung zu erreichen, besteht darin, den Algorithmus so aufzubauen, daß möglichst viele 'linked triads' benutzt werden.

Eine 'linked triad' kommt zustande, indem der Ausgangsvektor direkt als Eingangsvektor für die nächste Instruktion verwendet werden kann, ohne vorher in den Speicher zurückgeschrieben werden zu müssen. Voraussetzung hierzu ist jedoch, daß verschiedene Funktionseinheiten bei den Einzelinstruktionen angesprochen werden. Da an der Cyber 205 nur zwei Vektorströme vom Hauptspeicher in die Pipes erlaubt sind, muß bei einer solchen Operation immer eine skalare Größe beteiligt sein oder das Ergebnis der ersten Operation noch einmal verwendet werden, wie in den beiden folgenden Beispielen

```
A(1;N) = B(1;N) + C(1;N) * S
A(1;N) = ( B(1;N) + C(1;N) ) ** 2,
```

wobei A, B und C Vektoren und S eine skalare Größe ist.

Ist einer der Operanden ein Skalar, so wird er als Vektor mit gleichen Elementen interpretiert, dessen Länge sich aus der Anzahl der Elemente des Vektoroperanden ergibt (Broadcasting-Technik).

Als letztes sei der neue Datentyp
HALF PRECISION
vorgestellt. Durch ihn werden Zahlenwerte dargestellt, die nicht 64 Bit wie beim Zahlentyp REAL, sondern nur 32 Bit belegen. Der Vorteil dieses Datentyps besteht darin, daß arithmetische Operationen für Vektoren von diesem Typ nur etwa die Hälfte der Rechenzeit benötigen als die gleiche Operation mit dem Datentyp REAL. Der Vereinbarungsteil lautet zum Beispiel

```
HALF PRECISION X(N)
HALF PRECISION XD
```

Eine Konstante vom Typ HALF PRECISION wird statt einem 'E' bei REAL durch ein 'S' gekennzeichnet. Außerdem müssen Konstanten der Art 2. durch 2.S0 erkenntlich gemacht werden. Standard-FORTRAN-Funktionen können über den spezifischen Namen, der dann jedoch mit 'H' beginnt, aufgerufen werden.

10. Numerische Ergebnisse an den Vektorrechnern Cyber 205 und Siemens VP50

Da an der Siemens VP50 keine spezielle Vektornotation möglich ist, wurde der optimierte skalare Algorithmus implementiert. Auch besitzt die VP50 keine 32-Bit-Arithmetik. Die Vektorversion für die Cyber 205 enthält Cyber-spezifisches FORTRAN 8X (vgl. Anhang A2).

Es ist zu beachten, daß der Datentyp REAL an der Cyber 205 dem Datentyp REAL*8 an Siemens- oder IBM-Anlagen entspricht. Die Übereinstimmung in den Rechnungen besteht in 12 Mantissenstellen. Analog entspricht der Datentyp HALF PRECISION an der Cyber 205 dem Datentyp REAL*4 an Siemens- oder IBM-Anlagen. Hier beträgt die Übereinstimmung 6 Mantissenstellen.

Im folgenden geben wir sowohl für den nichtrelativistischen als auch für den relativistischen Particle Pusher CPU-Zeiten an, die wir an der Cyber 205 und Siemens VP50 für verschieden Fälle bestimmt haben. Zum Vergleich mit Skalarrechnern geben wir nochmals die Rechenzeiten an, die an Siemens M7890 und IBM/3090 gewonnen wurden. Es wurde dabei jedesmal derselbe Algorithmus mit einer IBM-REAL*8-Genauigkeit verwendet.

a) nichtrelativistischer Pusher

IBM/3090	4.6 µsec / Teilchen
Siemens M7890	3.3 µsec / Teilchen
Cyber 205	0.32 µsec / Teilchen
Siemens VP50	0.28 µsec / Teilchen
(Cyber 205, halb-genau	0.15 µsec / Teilchen)

b) relativistischer Pusher

IBM/3090	12.8 µsec / Teilchen
Siemens M7890	10.9 µsec / Teilchen
Cyber 205	0.93 µsec / Teilchen
Siemens VP50	1.65 µsec / Teilchen
(Cyber 205, halb-genau	0.44 µsec / Teilchen)

11. Konsequenzen für einen Particle-in-Cell Code

Da sich die Zeitskalen für leichte Ionen und Elektronen gemäß ihrem Massenverhältnis etwa um den Faktor 2000 unterscheiden, und die physikalisch relevanten Prozesse in der Ionenzeitskala stattfinden, empfiehlt es sich bei der Erstellung eines PIC-Codes mit zwei unterschiedlichen Zeitskalen zu rechnen. In der Ionenzeitskala sollten alle physikalischen Größen von neuem berechnet und in einer Subzeitskala lediglich die Elektronen fortbewegt werden. Die Subzeitskala der Elektronen ist durch die Zeitschrittweitenrestriktion festgelegt. Der sich über die Zeitschrittweitenrestriktion für die Ionen ergebende Ionenzeitschritt dürfte allerdings für den PIC-Code zu groß sein.
In Bezug auf Rechenzeitersparnis empfiehlt es sich, den Particle Pusher auf einem Vektorrechner zu betreiben, da er sehr effizient vektorisierbar ist.

Mein Dank gilt Frau Schmidt für die Mithilfe bei der Erstellung des Graphikprogramms.

12. Literatur

[1.] C. K. Birdsall, A. B. Langdon, Plasma Physics via Computer Simulation, McGraw-Hill, 1985

[2.] J. P. Boris, Relativistic Plasma Simulation, Fourth Conference on the Numerical Simulation of Plasma, Proceedings, p.3 ; Naval Research Laboratory, Washington, D.C., 2. - 3. November 1970

[3.] F. Chen, Plasma Physics and Controlled Fusion, Vol.1, Plenum Press, New York, 1984

[4.] H.-G. Fengler, N. Geers, K. F. Hanauer, P. Weber, CYBER 205 Benutzer-Handbuch, 2. Auflage, Universität Karlsruhe, 1985

[5.] FORTRAN-8X Standard, X3J3 S8, Version 99, 1986

[6.] B. Goplen, R.E. Clark and S.J. Flint, Geometrical Effectsin Magnetically Insulated Power Transmission Lines, MRC/WDC-R-001, 1979

[7.] R. W. Hockney, J.W. Eastwood, Computer Simulation Using Particles, McGraw-Hill, 1981

[8.] Landau, Lifshitz, The classical Theory of Fields, Vol. 2, Pergamon Press, 1962

[9.] Reference Manual, CDC VSOS Version 2, For Use with Cyber 200 Series Computer System, Control Data Corporation, 1984

[10.] Reference Manual, FORTRAN 200 Version 1, For Use with CDC Cyber 200 Virtual Storage Operating System, Control Data Corporation, 1984

[11.] Th. Westermann, D. Seldner, IUnveröffentlichter Bericht, Kernforschungszentrum Karlsruhe GmbH, März 1987

13. Anhang

ANHANG A1

NICHTRELATIVISTISCHER CYLRAD-ALGORITHMUS

1. HALBBESCHLEUNIGUNG

```
FAC = Q/M * DT * 0.5
VX1 = VR + FAC * ER
VZ1 = VZ + FAC * EZ
```

ROTATION

```
FACB = FAC * B
FACBQ = FACB * FACB
F1B = FACB*( 1. + (0.31755 + 0.2033 * FACBQ) * FACBQ )
VX2 = VX1 - F1B * VZ1
VZ2 = VZ1 + F1B * VX1
F2B = 2. * F1B/( 1. + F1B*F1B )
VX3 = VX1 - F2B * VZ2
VZ3 = VZ1 + F2B * VX2
```

2. HALBBESCHLEUNIGUNG

```
VR = VX3 + FAC * ER
VZ = VZ3 + FAC * EZ

R  = R + DT * VR
Z  = Z + DT * VZ
```

ANHANG A2

```
      SUBROUTINE PPUSHR(PA,NP,N,PIN,EDM,EZ,ER,B,DT)
*****************************************************************************************
C     SUBROUTINE PUSHER
C
C     AUTOR  : THOMAS WESTERMANN, KFK/IDT
C     DATUM  : OKTOBER 1986
C     SPRACHE: CYBER-SPEZIFISCHES-FORTRAN 8X
C
C
C     ARGUMENTLISTE
C     -------------
C
C
C     PA     (E/A):  TEILCHEN-KOORDINATEN-MATRIX
C                    PA(.,1) : Z-KOMPONENTE
C                    PA(.,2) : R-KOMPONENTE
C                    PA(.,3) : VZ-KOMPONENTE
C                    PA(.,4) : VR-KOMPONENTE
C     NP     (E)  :  MAXIMALE ANZAHL DER TEILCHEN
C     N      (E)  :  AKTUELLE ANZAHL DER TEILCHEN
C     EDM    (E)  :  E/M (LADUNGS-MASSE-VERHAELTNIS DER TEILCHEN)
C     PIN    (E)  :  PIN GROESSER 0, DANN TEILCHEN IM BERECHNUNGSGEBIET
C     EZ     (E)  :  ELEKTR. FELD IN Z-RICHTUNG AN DEN TEILCHENORTEN
C     ER     (E)  :  ELEKTR. FELD IN R-RICHTUNG AN DEN TEILCHENORTEN
C     B      (E)  :  MAGNET. FELD IN FI-RICHTUNG AN DEN TEILCHENORTEN
C     DT     (E)  :  ZEITSCHRITT
C
*****************************************************************************************
      HALF PRECISION PA(NP,4),PIN(NP),EZ(NP),ER(NP),B(NP)
      HALF PRECISION CQUAD,EM,DT,FAC

      DESCRIPTOR PR1,PR2,PR3,PR4,PZ1,PZ2,PZ3,PZ4
      DESCRIPTOR GAMMA,F1B,F2B,BETAB,BETABQ
      HALF PRECISION PR1,PR2,PR3,PR4,PZ1,PZ2,PZ3,PZ4
      HALF PRECISION GAMMA,F1B,F2B,BETAB,BETABQ

      DATA CQUAD / 9.S16 /

      ASSIGN PR1,.DYN.N
      ASSIGN PR2,.DYN.N
      ASSIGN PR3,.DYN.N
      ASSIGN PR4,.DYN.N
      ASSIGN PZ1,.DYN.N
      ASSIGN PZ2,.DYN.N
      ASSIGN PZ3,.DYN.N
      ASSIGN PZ4,.DYN.N
      ASSIGN GAMMA,.DYN.N
      ASSIGN F1B,.DYN.N
      ASSIGN F2B,.DYN.N
      ASSIGN BETAB,.DYN.N
      ASSIGN BETABQ,.DYN.N
```

```
      WHERE (PIN(1;N).GT.0.0)

C
C  CYLRAD - ALGORITHMUS
C  --------------------
C
C  UMRECHNUNG VON V = (VR,VZ)  AUF P = (PR,PZ) = GAMMA * V
C  UND 1. HALBBESCHLEUNIGUNG
C

      GAMMA=VHSQRT(CQUAD/(CQUAD-VR(1;N)*VR(1;N)-VZ(1;N)*VZ(1;N)); GAMMA)
      FAC = EM * DT * 5.S-1
      PR1 = VR(1;N) * GAMMA + FAC * ER(1;N)
      PZ1 = VZ(1;N) * GAMMA + FAC * EZ(1;N)
C
C  BERECHNUNG DER DREHUNG MITTELS DER HILFSGROESSEN
C  (PR2,PZ2) UND (PR3,PZ3)
C
      GAMMA = VHSQRT( (CQUAD / (PR1*PR1 + PZ1*PZ1 + CQUAD )); GAMMA)
      BETAB = FAC * GAMMA * B(1;N)
      BETABQ = BETAB * BETAB
      F1B = BETAB * ( 1.S0 + (0.31755S0 + 0.2033S0 * BETABQ) * BETABQ )
      PR2 = PR1 - F1B * PZ1
      PZ2 = PZ1 + F1B * PR1
      F2B = 2.S0 * F1B/( 1.S0 + F1B * F1B )
      PR3 = PR1 - F2B * PZ2
      PZ3 = PZ1 + F2B * PR2
C
C  2. HALBBESCHLEUNIGUNG
C
      PR4 = PR3 + FAC * ER(1;N)
      PZ4 = PZ3 + FAC * EZ(1;N)
C
C  UMRECHNUNG AUF GESCHWINDIGKEITEN (VR,VZ) = (VR(T+DT/2), VZ((T+DT/2))
C  UND KOORDINATEN (R,Z) = (R(T+DT) , Z(T+DT) )
C
      GAMMA = VHSQRT(CQUAD/( CQUAD + PR4*PR4 + PZ4*PZ4 ); GAMMA)

      VR(1;N) = PR4 * GAMMA
      R(1;N) = R(1;N) + DT * VR(1;N)
      VZ(1;N) = PZ4 * GAMMA
      Z(1;N) = Z(1;N) + DT * VZ(1;N)

      END WHERE
      END
```