Diplomarbeit

Berechnung dreidimensionaler Potenziale, Trajektorien und Raumladungen zur Simulation von Ionenstrahlen in elektrischen und magnetischen Feldern

vorgelegt von

Jan Dietrich

Institut für Angewandte Physik

der Johann Wolfgang Goethe Universität

Frankfurt am Main

Januar 2004

Inhaltsverzeichnis

1. Einleitung

1.1. Motivation

Für den Entwurf ionenoptischer Geräte ist heutzutage die vorangegangene numerische Simulation der elektrischen Potenzial- und Feldverteilungen [1], der Ionentrajektorien sowie der durch sie verursachten Raumladungseffekte, eine grundlegende Voraussetzung.

In der Vergangenheit führten die mangelnden Speicherkapazitäten sowie Rechenleistungen der verwendeten Computer dazu, dass die meisten einschlägig verwendeten Programme nur für die Berechnung zweidimensionaler Probleme ausgelegt sind. Zu dieser Kategorie von Programmen gehört *IGUN* [2][3]. Durch diese Beschränkung musste eine dreidimensionale Problemstellung bisher auf eine zweidimensionale reduziert werden, bevor sie berechnet werden konnte.

Obwohl dies in vielen Fällen durchaus möglich ist, gibt es auch Probleme bei denen eine Dimensionsreduzierung nicht durchführbar ist. Als Beispiel sei hier die Extraktion negativer Wasserstoff-Ionen genannt: Bei den bisher verwendeten Verfahren zur H^--Extraktion wird ein starker H^--Ionenstrahl direkt aus einer hochfrequenz-induzierten Multicusp-Ionenquelle entnommen. Dabei wird gleichzeitig ein unerwünschter Elektronenstrahl extrahiert, der durch einen magnetischen Filter vom Ionenstrahl getrennt werden muss. Die dabei verwendeten Magnetfelder separieren Elektronen und Ionen, lenken aber auch die H^--Ionen ab. Dadurch wird die Zylindersymmetrie des Ionenstrahls verletzt, weshalb dieser Vorgang nur bei dreidimensionaler Beschreibung vollständig simuliert werden kann.

1.2. Ziel der Arbeit

Im Rahmen dieser Arbeit wurde ein Programm entwickelt, dass speziell die dreidimensionalen elektrischen Potenziale und Feldverteilungen sowie die daraus resultierenden Partikeltrajektorien berechnet und grafisch darstellt. Gemessene oder mit anderen Programmen berechnete Magnetfelder können in die Bahnberechnung einbezogen werden. Auftretende Raumladungseffekte werden berücksichtigt und fließen in die Berechnung der Trajektorien ein. Die grafischen Darstellungen können dabei vom Nutzer interaktiv durch Vergrößerung, Drehung oder Verschiebung angepasst werden.

1.3. Mathematische Grundlagen

Zur numerischen Berechnung der Potenziale wird das Problem auf Gittermaschen diskretisiert und die entsprechenden Differentialgleichungen (im statischen elektrischen Fall also die Poissongleichung) mit den Randbedingungen in ein lineares Gleichungssystem endlichen Ranges umgeformt. Bei dem hier verwendeten finiten Differenzenverfahren kommt ein orthogonales regelmäßiges Maschennetz zum Einsatz, bei dem zur Bestimmung der Potenzialwerte in den Maschenknoten die sechs Nachbarknoten (rechts, links, über, unter, vor und hinter dem Ursprungsknoten) herangezogen werden.

1.4. Auswahl der Programmiersprache

Für die Umsetzung wurde die Programmiersprache Java verwendet, die sich aufgrund ihrer Eigenschaften besonders für die Lösung solcher Problemstellungen eignet:

- Zum Einen können die mit den jeweiligen Problemfeldern verbundenen Algorithmen und Datenstrukturen besonders leicht in einer einfach zu formulierenden Java-Klasse gekapselt werden,

- zum Anderen bietet sich Java durch seine Skalierbarkeit und Platformunabhängigkeit auch für die Lösung umfangreicher Problemstellungen an.

Die Programme können kostengünstig auf Einzelplatz-PCs entwickelt und getestet werden, um später auf Multiprozessor-Großrechnersysteme portiert und für die Lösung rechenintensiver komplexerer Probleme genutzt zu werden.

Java ist lizenzkostenfrei und in der jeweils aktuellen Fassung im Internet unter http://java.sun.com erhältlich.

Für die Darstellung dreidimensionaler Grafiken bietet Java eine umfangreiche Programmbibliothek an, die sowohl die OpenGL-, als auch die DirectX-Grafikschnittstelle moderner Mikrocomputersysteme effizient unterstützt.

1.5. Erstellte Java-Klassen

Eine Java-Applikation setzt sich aus Programmmodulen (im Java-Terminus *Klasse* genannt) zusammen, die jeweils Themengebiete voneinander abgrenzen (*kapseln*). Eine Klasse besteht dabei einerseits aus speicherbaren Daten (*Eigenschaften*) und andererseits aus Funktionen und Prozeduren (*Methoden*), die diese Daten manipulieren.

Im folgenden Kapitel werden die wichtigsten der im Rahmen dieser Arbeit erstellten Java-Klassen in ihrer Funktion und mit ihren Schnittstellen beschrieben.

Die Geometrie eines Problems wird in der Klasse *Geometry3D* gespeichert und verarbeitet. Sie interpretiert nicht nur die für die geometrische Anordnung relevanten Daten, sondern stellt auch Methoden zur grafischen Ausgabe zur Verfügung. Ist die Geometrie eines Problems bekannt, so können mit Hilfe der Klasse *Boundary3D* die Randwertpunkte bestimmt werden, die für die Berechnung des Potenzialfeldes notwendig sind. Auf die Geometrieeingabe und Randwertpunktbestimmung wird im Kapitel 2 näher eingegangen.

Sind die geometrischen Anordnungen und die Randwertpunkte bestimmt, kann das Potenzialfeld in der Klasse *Potential* berechnet und gespeichert werden. Die verwendeten Verfahren werden in Kapitel 3 beschrieben.

Auf die Berechnung der elektrischen Feldstärken aus dem Potenzialfeld mittels der Klasse *ElectricField* wird in Kapitel 4 eingegangen.

Um magnetische Felder in die Trajektorienberechnung einfliessen zu lassen, wird über die hierzu definierte Java-Klasse *MagneticFieldFromAmpere* eine Schnittstelle zum Programm *Ampere* der Firma *Integrated Engineering Software* [7] bereitgestellt. Die von *Ampere* generierten magnetischen Feldstärken können mit dieser Schnittstelle bequem eingelesen werden.

Nach erfolgter Berechnung der elektrischen und magnetischen Felder bietet das Programm die Möglichkeit, Trajektorien von Teilchen mit beliebig wählbarer Ladung und Masse zu berechnen. Diese Trajektorien werden in einer Klasse *Trajectory3D* zusammengefasst, die sowohl die nötigen Datenstrukturen als auch die Algorithmen zur Bahnberechnung umfassen. Kapitel 5 beschreibt die dazu verwendeten mathematischen Grundlagen und Verfahren.

2. Geometrieeingabe und Randwertbestimmung

Um die elektrostatischen Potenziale berechnen zu können, ist es notwendig die Versuchsanordnung und insbesondere die exakte Positionierung der auf bestimmten Potenzialen befindlichen Elektroden in dem zu simulierenden Raumbereich dem Programm übermitteln zu können. Dabei müssen die korrekten Abstände zwischen den Gitterpunkten und der Elektrodenbegrenzung definiert werden.

Die Erfassung von Gitterpunkten ist unproblematisch zu lösen. Die Beschreibung von Randwertpunkten erfordert dagegen ein besonderes Vorgehen, für das verschiedene Alternativen bestehen.

Eine Möglichkeit die Randwertpunkte zu bestimmen wäre, sie gemeinsam mit den Geometriepunkten einzulesen. Diese Möglichkeit wurde nicht weiter verfolgt, da sie gleichzeitig sehr zeitaufwendig und fehleranfällig ist. Stattdessen werden nur die Schnittpunkte der Elektrodenoberfläche mit den x-y-Ebenen in das Programm eingegeben, aus welchen dann alle fehlenden Randwertpunkte errechnet werden.

Um diese Eingabe weiter zu vereinfachen (und damit auch die Fehlerquelle der Eingabe zu minimieren), wurde zusätzlich ein von *R.Jürgens* [8] entwickeltes Verfahren übernommen, dass es erlaubt, geometrische Elektrodenformen analytisch zu definieren. So wird zum Beispiel eine die z-Achse kreisförmig schneidende Elektrode durch Angabe des Elektrodenmittelpunktes und des Radius vollständig beschrieben.

Die Geometriedaten werden in einer Textdatei abgespeichert und von der im Rahmen der Diplomarbeit erstellten Java-Klasse *Geometry3D* eingelesen und dann in entsprechende Geometriepunkte umgewandelt. Dabei werden die Elektroden in einfacher Weise durch xy-Ebenen geschnitten, so dass die Oberflächen in einer xy-Ebene als gerade oder gekrümmte Linien erscheinen. Für Geraden ist nur die Angabe des Anfangs- sowie des Endpunktes erforderlich, wobei innere Gitterpunkte in

Blickrichtung zum Endpunkt rechts liegen. Ist die Linie gekrümmt, ist außer dem Anfangs- und Endpunkt noch ein Kreis mit der entsprechenden Krümmung anzugeben.

Für die Definition der Randwertpunkte wird in der Java-Klasse *Boundary3D* über den zu untersuchenden Bereich ein rechtwinkliges Gitternetz gelegt, dessen Gitterkonstanten in x-, y- und z-Richtung frei wählbar sind. Alle Gitterpunkte, die innerhalb des zu berechnenden Bereiches liegen („innere Gitterpunkte“) und deren Abstand zu einer Elektrodenoberfläche kleiner als die Gitterkonstante ist, werden als Randwertpunkte abgespeichert.

Zu jedem so ermittelten Wert wird neben den Koordinaten des Punktes auch der Abstand zur Oberfläche und eine Potenzialnummer abgespeichert. Die Potenzialnummer gibt an, welches Dirichletpotenzial zur Berechnung verwendet werden soll.

Abstände in z-Richtung werden durch lineare Interpolation zwischen zwei benachbarten xy-Schnitten definiert. Ändert sich die geometrische Anordnung in z-Richtung nicht, so genügt die Angabe einer Länge in z-Richtung, um die sich wiederholende Geometrieanordnung auf die weiteren Ebenen zu kopieren.

Die bei der Geometrieeingabe erstellten Objekte vom Typ *Geometry3D* und *Boundary3D* dienen im weiteren Programmablauf als Grundlage für die Berechnung des Potenzialfeldes durch die Java-Klasse *Potential*.

3. Berechnung des elektrostatischen Potenzials

Im folgenden werden zunächst die mathematischen Grundlagen zur Berechnung des elektrostatischen Potenzials beschrieben. Im Anschluß wird die Umsetzung in geeignete Datenstrukturen und Algorithmen erläutert.

3.1. Grundlagen

Die elektrische Feldstärke $\vec{E}$ bildet ein wirbelfreies Quellfeld

$$\nabla \times \vec{E}(\vec{r}, t) = 0\,, \tag{3.1}$$

dessen Ergiebigkeit durch die Raumladung ρ und die umgebende Oberflächenladung bestimmt ist. Es kann in konservativen Feldern durch den negativen Gradienten

$$\vec{E}(\vec{r}, t) = -\nabla\varphi \tag{3.2}$$

unter Verwendung des Potenzials $\varphi = \varphi(\vec{r})$ ausgedrückt werden. Wird nun die Flussdichte $\vec{D} = \varepsilon\vec{E}$ über die Feldstärke durch das Potenzial ausgedrückt, so gelangt man zur Feldgleichung der Elektrostatik, die eine inhomogene partielle Differentialgleichung für das Potenzial φ darstellt:

$$\rho = \mathrm{div}\vec{D} = \mathrm{div}(\varepsilon\vec{E}) = \varepsilon\mathrm{div}\vec{E} = \varepsilon\mathrm{div}(-\mathrm{grad}\varphi) = -\varepsilon\mathrm{divgrad}\varphi = -\varepsilon\Delta\varphi\ . \tag{3.3}$$

Die Beschreibung eines räumlich ausgedehnten Feldes, erzeugt durch eine Ladungsdichteverteilung $\rho(\vec{r})$, geschieht durch die Poissongleichung

$$\Delta\varphi = -\frac{\rho}{\varepsilon}\ . \tag{3.4}$$

Wenn keine Ladungen vorhanden sind ($\rho = 0$), geht die Poissongleichung in eine homogene partielle Differentialgleichung über

$$\Delta\varphi = 0 \text{ (Laplacegleichung)} . \tag{3.5}$$

Die Laplacegleichung bildet die Ausgangsbeziehung für die Potenzialtheorie der Elektrostatik.

3.2. Die Feldberechnung

3.2.1. Ziel der numerischen Feldberechnung

Die Anwendung analytischer Lösungsverfahren (konforme Abbildungen, Produktansatz, Reihenentwicklung) zur Lösung der Laplaceschen bzw. Poissonschen Differentialgleichung ist an bestimmte Bedingungen geknüpft, die in vielen praktischen Fällen nicht erfüllt sind. So können zum Beispiel mit Hilfe konformer Abbildungen nur zweidimensionale Felder berechnet werden. Reihenentwicklungen gestatten die analytische Feldberechnung mit vertretbarem Aufwand nur dann, wenn bestimmte Symmetrieeigenschaften des Feldes vorliegen.

Wenn die Materialien, in denen sich das Feld ausbreitet, inhomogen, anisotrop oder nichtlinear sind, versagen analytische Verfahren. Numerische Näherungsverfahren werden auch dann angewandt, wenn analytische Verfahren existieren, deren Anwendung aber zu aufwändig ist.

Für eine numerische Feldberechnung muss ein Rechenmodell erstellt werden, dass ein reales System in der geforderten Genauigkeit nachbildet. Das Rechenmodell muss unter anderem die Geometrie der zu simulierenden Anordnung, die Eigenschaften der Materialien sowie Umgebungseinflüsse beinhalten.

Je besser das reale System numerisch modelliert werden kann, umso genauer sind die Ergebnisse. Meist steigt aber auch der Aufwand an Vorbereitung und Rechenzeit sowie die Auftrittswahrscheinlichkeit von systematischen Fehlern erheblich.

3.2.2. Lösungsmöglichkeiten des Feldproblems

Bei der numerischen Feldberechnung können unterschiedliche Ansätze zur Lösung verwendet werden. Diese sind zum Beispiel:

Finite-Differenzen-Methode (FDM):

Ein geschlossenes, durch einen Rand begrenztes Feldgebiet, wird mit einem Gitter überzogen. An den Knoten (Schnittpunkte der Gitterlinien) wird eine Differenzengleichung gelöst.

Finite-Elemente-Methode (FEM):

Das Feldgebiet wird in Grundelemente (z.B. Dreiecke, Vierecke, Tetraeder, etc.) zerlegt, an dessen Knoten die Lösung ermittelt wird. Im Gegensatz zur FDM wird ein Funktional aller Funktionswerte extremal gestaltet, z.B. als Minimierung der Feldenergien, die eine Influenzladung auf der Oberfläche umgeben.

Integralgleichungsmethode (IGM):

Bei der IGM werden Integralgleichungen gelöst. Sonderformen sind die Boundary Element Methode BEM und die Volumenintegralmethode .

Diese Lösungsansätze sind für verschiedene Probleme unterschiedlich gut geeignet. In einem Feldberechnungsprogramm können ein oder mehrere der genannten Lösungsansätze implementiert sein. Das im Rahmen dieser Arbeit entwickelte Programm wendet zur Berechnung der elektrostatischen Felder die Finite-Differenzen-Methode an, bei der die Potenziale auf den Schnittpunkten der Gitterlinien diskretisiert werden.

3.3. Diskretisierung der Poissongleichung für finite Differenzen im ebenen kartesischen System

Bei der Methode der finiten Differenzen werden die Ableitungen der zu lösenden Diffentialgleichung durch Differenzenquotienten ersetzt. Gesucht ist die Lösung der Poissongleichung

$$\Delta U = \frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} + \frac{\partial^2 U}{\partial z^2} = -\frac{\rho}{\varepsilon_0} \tag{3.6}$$

3.3.1. Allgemeine Lösung der Poissongleichung

Gegeben sei ein beliebiger innerer Maschenknoten mit Potenzial U. Nennen wir die Position dieses Knotens x_0, y_0, z_0, so können wir die 6 Nachbarknoten (einen rechts, einen links, einen über, einen unter, einen vor und einen hinter dem Ursprungsknoten) bezeichnen als

1. $U(x_0 + h_x, y_0, z_0)$,
2. $U(x_0 - h_x, y_0, z_0)$,
3. $U(x_0, y_0 + h_y, z_0)$,
4. $U(x_0, y_0 - h_y, z_0)$,
5. $U(x_0, y_0, z_0 + h_z)$
6. $U(x_0, y_0, z_0 - h_z)$.

Die Abstände der Nachbarknoten sind dann h_x auf der x-Achse, h_y auf der y-Achse, und h_z auf der z-Achse (Siehe auch Abbildung 3-1).

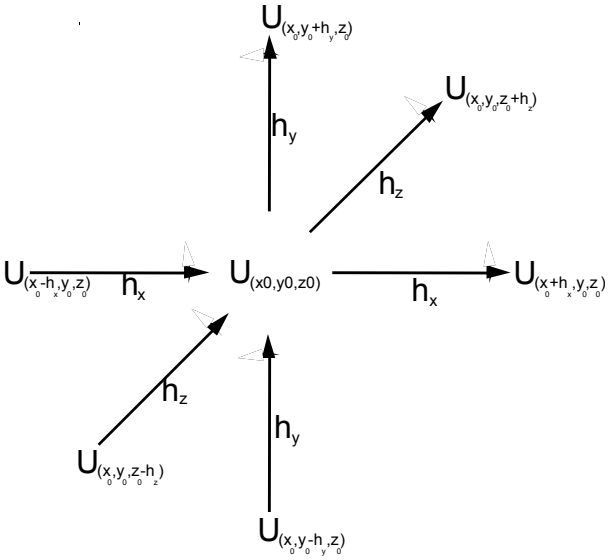

Abbildung 3-1: Nachbarknoten eines zu berechnenden Potenzials

Das Potenzial des Nachbarknotens mit Abstand h_x wird mit einer Taylor-Reihe approximiert:

$$f(x) = f(x_0) + \frac{f'(x_0)}{1!}(x - x0) + \frac{f''(x_0)}{2!}(x - x_0)^2 + \frac{f'''(x_0)}{3!}(x - x_0)^3 + O(h^4) \tag{3.7}$$

Durch Einsetzen von $f(x_0) = U(x_0, y_0, z_0)$, $x = x_0 + h_x$ und $f(x) = U(x_0 + h_x, y_0, z_0)$ folgt daraus

$$\begin{gathered} U(x_0 + h_x, y_0, z_0) = \\ U(x_0, y_0, z_0) + \frac{\partial U}{\partial x}(x_0, y_0, z_0)(x_0 + h_x - x_0) + \frac{1}{2!}\frac{\partial^2 U}{\partial x^2}(x_0, y_0, z_0)(x_0 + h_x - x_0)^2 + \ldots \end{gathered} \tag{3.8}$$

und nach Abbruch der Entwicklung zur 4. Ordnung folgt daraus als Näherung

$$\begin{aligned} U(x_0 + h_x, y_0, z_0) \approx U(x_0, y_0, z_0) + h_x \frac{\partial U}{\partial x}(x_0, y_0, z_0) + \\ + \frac{h_x^2}{2}\frac{\partial^2 U}{\partial x^2}(x_0, y_0, z_0) + \frac{h_x^3}{6}\frac{\partial^2 U}{\partial x^3}(x_0, y_0, z_0) + \sigma(h^4) \end{aligned} \tag{3.9}$$

mit dem Fehler der Ordnung $\sigma(h^4)$.

Für das Potenzial des Nachbarmaschenknotens mit Abstand $-h_x$ in Gegenrichtung folgt analog

$$U(x_0 - h_x, y_0, z_0) \approx U(x_0, y_0, z_0) - h_x \frac{\partial U}{\partial x}(x_0, y_0, z_0) + \\ + \frac{h_x^2}{2} \frac{\partial^2 U}{\partial x^2}(x_0, y_0, z_0) - \frac{h_x^3}{6} \frac{\partial^2 U}{\partial x^3}(x_0, y_0, z_0) \tag{3.10}$$

Bei der Addition fallen die Ableitungen der ersten und dritten Ordnung heraus:

$$U(x_0 + h_x, y_0, z_0) + U(x_0 - h_x, y_0, z_0) = 2U(x_0, y_0, z_0) + h_x \frac{\partial^2 U}{\partial x^2} \tag{3.11}$$

und es folgt durch einfache Termumformung

$$\frac{\partial^2 U}{\partial x^2} = \frac{U(x_0 + h_x, y_0, z_0) + U(x_0 - h_x, y_0, z_0) - 2U(x_0, y_0, z_0)}{h_x^2} \tag{3.12}$$

Analog kann für die y- Richtung

$$\frac{\partial^2 U}{\partial y^2} = \frac{U(x_0, y_0 + h_y, z_0) + U(x_0, y_0 - h_y, z_0) - 2U(x_0, y_0, z_0)}{h_y^2} \tag{3.13}$$

und für die z-Richtung

$$\frac{\partial^2 U}{\partial z^2} = \frac{U(x_0, y_0, z_0 + h_z) + U(x_0, y_0, z_0 - h_z) - 2U(x_0, y_0, z_0)}{h_z^2} \tag{3.14}$$

hergeleitet werden.

Einsetzen der Differenzenquotienten (3.12), (3.13) und (3.14) in die Poissongleichung ergibt

$$\begin{aligned}&\frac{U(x_0+h_x,y_0,z_0)+U(x_0-h_x,y_0,z_0)-2U(x_0,y_0,z_0)}{h_x^{\,2}}\\&+\frac{U(x_0,y_0+h_y,z_0)+U(x_0,y_0-h_y,z_0)-2U(x_0,y_0,z_0)}{h_y^{\,2}}\\&+\frac{U(x_0,y_0,z_0+h_z)+U(x_0,y_0,z_0-h_z)-2U(x_0,y_0,z_0)}{h_z^{\,2}}=-\frac{\rho}{\varepsilon_0}\end{aligned} \tag{3.15}$$

Nach Isolierung aller $U(x_0,y_0,z_0)$-Terme auf der linken Seite folgt daraus

$$\begin{aligned}&\left(\frac{2}{h_x^{\,2}}+\frac{2}{h_y^{\,2}}+\frac{2}{h_z^{\,2}}\right)U(x_0,y_0,z_0)=\\&\frac{U(x_0+h_x,y_0,z_0)}{h_x^{\,2}}+\frac{U(x_0-h_x,y_0,z_0)}{h_x^{\,2}}+\frac{U(x_0,y_0+h_y,z_0)}{h_y^{\,2}}+\\&+\frac{U(x_0,y_0-h_y,z_0)}{h_y^{\,2}}+\frac{U(x_0,y_0,z_0+h_z)}{h_z^{\,2}}+\frac{U(x_0,y_0,z_0-h_z)}{h_z^{\,2}}+\frac{\rho}{\varepsilon_0}\end{aligned} \tag{3.16}$$

und nach elementarer Umformung

$$\begin{aligned}&U(x_0,y_0,z_0)=\\&\frac{1}{2(h_x^{\,2}h_y^{\,2}+h_x^{\,2}h_z^{\,2}+h_y^{\,2}h_z^{\,2})}\left(\begin{array}{c}h_y^{\,2}h_z^{\,2}\left[U(x_0+h_x,y_0,z_0)+U(x_0-h_x,y_0,z_0)\right]+\\+h_x^{\,2}h_z^{\,2}\left[U(x_0,y_0+h_y,z_0)+U(x_0,y_0-h_y,z_0)\right]+\\+h_x^{\,2}h_y^{\,2}\left[U(x_0,y_0,z_0+h_z)+U(x_0,y_0,z_0-h_z)\right]+\\+h_x^{\,2}h_y^{\,2}h_z^{\,2}\frac{\rho}{\varepsilon_0}\end{array}\right)\end{aligned} \tag{3.17}$$

Die Gleichung wird in dieser Form in der Java-Klasse *Potential* zur Berechnung von inneren (nicht angeschnittenen) Maschenpunkten verwendet.

Sind die Maschen isotrop, also die Abstände in den drei Koordinatenrichtung x, y und z äquidistant zueinander ($h_x=h_y=h_z\equiv h$), vereinfacht sich die Gleichung weiter zu

$$U(x_0,y_0,z_0)=\frac{1}{6}\begin{pmatrix} U(x_0+h,y_0,z_0)+U(x_0-h,y_0,z_0) \\ +U(x_0,y_0+h,z_0)+U(x_0,y_0-h,z_0) \\ +U(x_0,y_0,z_0+h)+U(x_0,y_0,z_0-h) \\ +h^2\frac{\rho}{\varepsilon_0} \end{pmatrix} \tag{3.18}$$

3.3.2. Lösung der Poissongleichung für ungleiche Maschenabstände

In allgemeineren Fall, nämlich wenn die Abstände zu den sechs Nachbarmaschenknoten unterschiedlich sind (z.B. an den Rändern oder in unmittelbarer Nachbarschaft zu einer Elektrode), bezeichnen wir die Potenziale der Nachbarknoten als

1. $U(x_0+h_x,y_0,z_0)$,
2. $U(x_0-g_x,y_0,z_0)$,
3. $U(x_0,y_0+h_y,z_0)$,
4. $U(x_0,y_0-g_y,z_0)$,
5. $U(x_0,y_0,z_0+h_z)$
6. $U(x_0,y_0,z_0-g_z)$.

Wie in Abbildung 3-2 zu erkennen, sind die Abstände der Knoten zueinander dann h_x und g_x auf der x-Achse, h_y und g_y auf der y-Achse, und h_z und g_z auf der z-Achse.

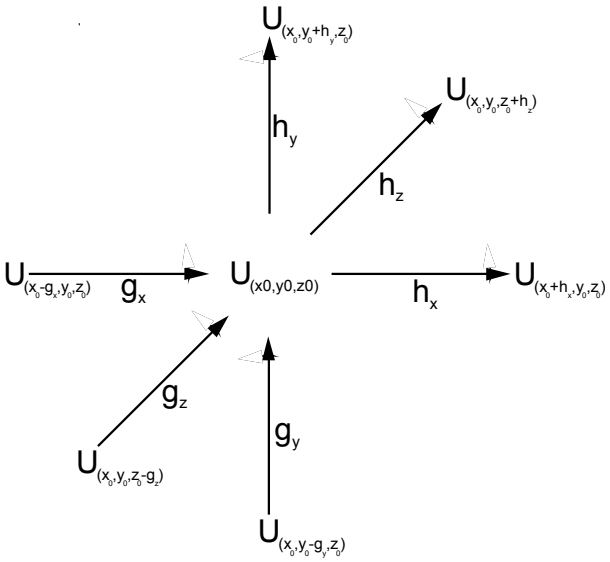

Abbildung 3-2: verwendete Potenziale bei ungleichen Maschenabständen

Eine Herleitung kann analog zum Verfahren bei isotropen Maschen durchgeführt werden. Da allerdings bei ungleichen Maschenabständen die Ableitungen der 1. und 3. Ordnung nicht wegfallen, muss die Taylor-Näherung bereits nach der 2. Ordnung abgebrochen werden. Eine Herleitung mit Hilfe einer Taylor-Näherung führt zu

$$U(x_0 + h_x, y_0, z_0) \approx U(x_0, y_0, z_0) + h_x \frac{\partial U}{\partial x}(x_0, y_0, z_0) + \frac{h_x^2}{2} \frac{\partial^2 U}{\partial x^2}(x_0, y_0, z_0) + O(h^3) \qquad (3.19)$$

und für das Potenzial des Nachbarmaschenknotens mit Abstand g_x in Gegenrichtung ergibt sich

$$U(x_0 - g_x, y_0, z_0) \approx U(x_0, y_0, z_0) - g_x \frac{\partial U}{\partial x}(x_0, y_0, z_0) + \frac{g_x^2}{2} \frac{\partial^2 U}{\partial x^2}(x_0, y_0, z_0) \ . \qquad (3.20)$$

Um die Gleichungen in die Poissongleichung einsetzen zu können, muss erst der unbekannte Term $\frac{\partial U}{\partial x}(x_0, y_0, z_0)$ eliminiert werden. Dazu werden beide Gleichungen nach $\frac{\partial U}{\partial x}(x_0, y_0, z_0)$ aufgelöst und gleichgesetzt:

$$U(x_0 + h_x, y_0, z_0) \approx U(x_0, y_0, z_0) + h_x \frac{\partial U}{\partial x}(x_0, y_0, z_0) + \frac{h_x^2}{2} \frac{\partial^2 U}{\partial x^2}(x_0, y_0, z_0) \qquad (3.21)$$

wird zu

$$\frac{\partial U}{\partial x}(x_0, y_0, z_0) \approx \frac{1}{h_x}\left[U(x_0 + h_x, y_0, z_0) - U(x_0, y_0, z_0) - \frac{h_x^2}{2} \frac{\partial^2 U}{\partial x^2}(x_0, y_0, z_0) \right] \qquad (3.22)$$

und aus

$$U(x_0 - g_x, y_0, z_0) \approx U(x_0, y_0, z_0) + g_x \frac{\partial U}{\partial x}(x_0, y_0, z_0) + \frac{g_x^2}{2} \frac{\partial^2 U}{\partial x^2}(x_0, y_0, z_0) \qquad (3.23)$$

wird

$$\frac{\partial U}{\partial x}(x_0, y_0, z_0) \approx \frac{1}{g_x}\left[U(x_0 - g_x, y_0, z_0) - U(x_0, y_0, z_0) - \frac{g_x^2}{2} \frac{\partial^2 U}{\partial x^2}(x_0, y_0, z_0) \right] . \qquad (3.24)$$

Durch Gleichsetzen erhält man

$$\begin{aligned} &\frac{1}{h_x}\left[U(x_0 + h_x, y_0, z_0) - U(x_0, y_0, z_0) - \frac{h_x^2}{2} \frac{\partial^2 U}{\partial x^2}(x_0, y_0, z_0) \right] \\ &\approx \frac{1}{g_x}\left[U(x_0 - g_x, y_0, z_0) - U(x_0, y_0, z_0) - \frac{g_x^2}{2} \frac{\partial^2 U}{\partial x^2}(x_0, y_0, z_0) \right] \end{aligned} \qquad (3.25)$$

und daraus folgt durch einfache Termumformung

$$\frac{\partial^2 U}{\partial x^2} = \frac{2}{h_x g_x}\left[\frac{g_x U(x_0 + h_x, y_0, z_0)}{h_x + g_x} - \frac{h_x U(x_0 - g_x, y_0, z_0)}{h_x + g_x} - U(x_0, y_0, z_0)\right]. \qquad (3.26).$$

Analog kann für die y- Richtung hergeleitet werden

$$\frac{\partial^2 U}{\partial y^2} = \frac{2}{h_y g_y}\left[\frac{g_y U(x_0, y_0 + h_y, z_0)}{h_y + g_y} - \frac{h_y U(x_0, y_0 - g_y, z_0)}{h_y + g_y} - U(x_0, y_0, z_0)\right] \qquad (3.27)$$

und für die z-Richtung ergibt sich

$$\frac{\partial^2 U}{\partial z^2} = \frac{2}{h_z g_z}\left[\frac{g_z U(x_0, y_0, z_0 + h_z)}{h_z + g_z} - \frac{h_z U(x_0, y_0, z_0 - g_z)}{h_z + g_z} - U(x_0, y_0, z_0)\right]. \qquad (3.28)$$

Einsetzen der Differenzenquotienten in die Poissongleichung ergibt:

$$\begin{aligned}
&\frac{2}{h_x g_x}\left[\frac{g_x U(x_0 + h_x, y_0, z_0)}{h_x + g_x} - \frac{h_x U(x_0 - g_x, y_0, z_0)}{h_x + g_x} - U(x_0, y_0, z_0)\right] \\
&+\frac{2}{h_y g_y}\left[\frac{g_y U(x_0, y_0 + h_y, z_0)}{h_y + g_y} - \frac{h_y U(x_0, y_0 - g_y, z_0)}{h_y + g_y} - U(x_0, y_0, z_0)\right] \qquad (3.29) \\
&+\frac{2}{h_z g_z}\left[\frac{g_z U(x_0, y_0, z_0 + h_z)}{h_z + g_z} - \frac{h_z U(x_0, y_0, z_0 - g_z)}{h_z + g_z} - U(x_0, y_0, z_0)\right] = -\frac{\rho}{\varepsilon_0}
\end{aligned}$$

Und nach Isolation aller $U(x_0, y_0, z_0)$ -Terme auf der linken Seite:

$$\begin{aligned}
&\left(\frac{2}{h_x g_x} + \frac{2}{h_y g_y} + \frac{2}{h_z g_z}\right) U(x_0, y_0, z_0) = 2\frac{U(x_0 + h_x, y_0, z_0)}{h_x (h_x + g_x)} + \\
&+2\frac{U(x_0 - g_x, y_0, z_0)}{g_x (h_x + g_x)} + 2\frac{U(x_0, y_0 + h_y, z_0)}{h_y (h_y + g_y)} + 2\frac{U(x_0, y_0 - g_y, z_0)}{g_y (h_y + g_y)} + \quad . \qquad (3.30) \\
&+2\frac{U(x_0, y_0, z_0 + h_z)}{h_z (h_z + g_z)} + 2\frac{U(x_0, y_0, z_0 - g_z)}{g_z (h_z + g_z)} + \frac{\rho}{\varepsilon_0}
\end{aligned}$$

Gleichung (3.30) ist die allgemeine Lösung der Poissongleichung für beliebige unterschiedliche Abstände zu den sechs Nachbarpunkten. Sie wird in dieser Form zur Berechnung des Potenzials von äußeren angeschnittenen Maschenpunkten verwendet.

Der durch das Vernachlässigen des Restgliedes entstandene Fehler liegt in der Ordnung $\sigma(h_x^4 + h_y^4 + h_z^4)$ und kann durch geringe Schrittweiten klein gehalten werden. Im anisotropen Fall vergrößert sich der Fehler zu $\sigma(h_x^2 + h_y^2 + h_z^2)$, zum Beispiel an den Elektrodenrändern.

3.3.3. Interpolationsverfahren für Potenziale zwischen den Stützstellen

Ist das Potenzial auf allen Maschenpunkten hinreichend genau bekannt, kann zwischen den Maschen interpoliert werden. Um den Interpolationsfehler gering zu halten, werden nicht nur die sechs nächstliegenden Maschenknoten (in der Abbildung 3-3 blau eingezeichnet) zur Berechnung herangezogen, sondern zusätzlich noch die zwölf (in der Abbildung grünen) Eckmaschenknoten verwendet.

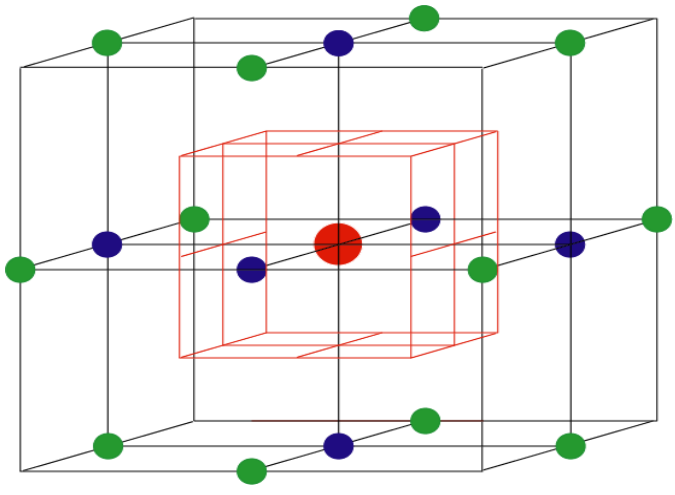

Abbildung 3-3: Verwendete Stützstellen bei der Potenzialinterpolation

Dabei gilt für das Potenzial U eines Punktes $(x_0 + x, y_0 + y, z_0 + z)$ nahe des Maschenknotens (x_0, y_0, z_0), dessen Potenzial U_0 bekannt ist, nach dem Taylor-Ansatz:

$$\begin{aligned} U \equiv U(x_0 + x, y_0 + y, z_0 + z) = U_0 + x\frac{\partial U_0}{\partial x} + y\frac{\partial U_0}{\partial y} + z\frac{\partial U_0}{\partial z} \\ + \frac{x^2}{2}\frac{\partial^2 U_0}{\partial x^2} + \frac{y^2}{2}\frac{\partial^2 U_0}{\partial y^2} + \frac{z^2}{2}\frac{\partial^2 U_0}{\partial z^2} + xy\frac{\partial^2 U_0}{\partial x \partial y} + xz\frac{\partial^2 U_0}{\partial x \partial z} + yz\frac{\partial^2 U_0}{\partial y \partial z} + \sigma(R^3) \end{aligned} \tag{3.31}$$

Ersetzt man den Gradienten $\frac{\partial U}{\partial x}$ (und analog auch $\frac{\partial U}{\partial y}$ und $\frac{\partial U}{\partial z}$) durch seine diskretisierte Form

$$\frac{U(x_0 + h_x) - U(x_0 - h_x)}{2h_x} \tag{3.32}$$

und ersetzt man weiterhin $\frac{\partial^2 U}{\partial x^2}$ durch

$$\frac{U(x_0 + h_x, y_0 + h_y) + U(x_0 - h_x, y_0 - h_y) - U_0}{h_x^{\,2}} \tag{3.33}$$

und $\frac{\partial^2 U}{\partial x \partial y}$ durch

$$\frac{U(x_0 + h_x, y_0 + h_y) + U(x_0 - h_x, y_0 - h_y) - U(x_0 - h_x, y_0 + h_y) - U(x_0 + h_x, y_0 - h_y)}{4h_x h_y}\,, \tag{3.34}$$

so folgt daraus

$$
\begin{aligned}
&U(x_0+x, y_0+y, z_0+z) = \\
&U_0 \\
&+x\frac{U(x_0+h_x)-U(x_0+h_x)}{2h_x} \\
&+y\frac{U(y_0+h_y)-U(y_0+h_y)}{2h_y} \\
&+z\frac{U(z_0+h_z)-U(z_0+h_z)}{2h_z} \\
&+x^2\frac{U(x_0+h_x)+U(x_0-h_x)-2U_0}{2h_x^{\,2}} \\
&+y^2\frac{U(y_0+h_y)+U(y_0-h_y)-2U_0}{2h_y^{\,2}} \\
&+z^2\frac{U(z_0+h_z)+U(z_0-h_z)-2U_0}{2h_z^{\,2}} \\
&+xy\frac{U(x_0+h_x,y_0+h_y)+U(x_0-h_x,y_0-h_y)-U(x_0+h_x,y_0-h_y)-U(x_0-h_x,y_0+h_y)}{4h_xh_y} \\
&+xz\frac{U(x_0+h_x,z_0+h_z)+U(x_0-h_x,z_0-h_z)-U(x_0+h_x,z_0-h_z)-U(x_0-h_z,y_0+h_z)}{4h_xh_z} \\
&+yz\frac{U(y_0+h_y,z_0+h_z)+U(y_0-h_y,z_0-h_z)-U(y_0+h_y,z_0-h_z)-U(y_0-h_y,z_0+h_z)}{4h_yh_z}
\end{aligned}
\tag{3.35}
$$

In der Java-Klasse *Potential* wird diese Formel zur Interpolation eines Potenzials innerhalb eines Maschenvolumens $V_{mesh} = h_x h_y h_z$ um einen Knoten mit bekanntem Potenzialwert verwendet. Mit Hilfe einer virtuellen Sonde, die vom Anwender im Koordinatensystem bewegt wird, kann damit für jede Ortskoordinate das entsprechende elektrostatische Potenzial angezeigt werden.

3.4. Relaxationsverfahren

Um den Iterationsprozess bei der Berechnung zu beschleunigen, wird die bei jedem Iterationsschritt ermittelte Potenzialdifferenz zum vorherigen Schritt mit einem

Relaxationsfaktor ϖ multipliziert. Der Potenzialwert an einem Maschenpunkt x,y,z während des Iterationsschrittes k ändert sich somit von

$$U_k(x,y,z) = U_{calc}(x,y,z) \tag{3.36}$$

zu

$$U_k(x,y,z) = U_{k-1}(x,y,z) + \omega\left[U_{calc}(x,y,z) - U_{k-1}(x,y,z)\right] \tag{3.37}.$$

3.4.1. Unterrelaxation , Überrelaxation, Sukzessive Überrelaxation

Bei einem Relaxationsfaktor von $0 < \varpi < 1$ spricht man von Unterrelaxation, bei einem Wert von $1 < \varpi \leq 2$ spricht man von Überrelaxation.

Werden die überrelaxierten Werte direkt wieder in das Potenzialwertefeld zurückgeschrieben und bei der nächsten Iteration bereits verwendet, so spricht man von einer sukzessiven Überrelaxation (SOR). Dieses Verfahren wird hier eingesetzt, da sich Potenzialänderungen schneller im Feld auswirken und dadurch den Berechnungsprozess wesentlich verkürzen.

Entscheidet man sich, die Methode der sukzessiven Überrelaxation einzusetzen, bilden sich gewöhnlich zwei Schwierigkeiten heraus, die gelöst werden müssen:

- Zum Einen hängt die Effizienz der Methode stark vom Relaxationsfaktor ab, dessen optimaler Wert erst bei Lösung des Problems bekannt wird,
- und zum Anderen von der Schwierigkeit, die Genauigkeit der Lösung während der einzelnen Iterationsschritte zu bestimmen.

3.4.2. Bestimmung des optimalen Relaxationsfaktors

Der Überrelaxationsfaktor ϖ wird im Rahmen dieser Arbeit durch ein Verfahren von Carré [4] optimiert. Während der ersten Iteration ist ϖ gleich 1, danach wird ϖ für 15 Iterationsschritte auf den Wert 1,6 gesetzt. Anschließend berechnet das Programm ϖ alle 15 Schritte neu aus

$$\omega=\omega_0-\frac{2-\omega_0}{4} \tag{3.38}$$

mit

$$\omega_0=\frac{2}{1+\sqrt{1-\frac{\left(\lambda_{max}+\omega-1\right)^2}{\lambda_{max}\omega^2}}} \tag{3.39}$$

und

$$\lambda_{max}=\frac{\left|\max(U_{calc}-U_k)\right|_{alleMaschen}}{\left|\max(U_{calc}-U_{k-1})\right|_{alleMaschen}} \tag{3.40}$$

solange, bis

$$\frac{\Delta\omega_0}{2-\omega_0}<0,05\ . \tag{3.41}.$$

3.5. Grafische Darstellungsmethoden von Potenzialfeldern

3.5.1. Äquipotenzialflächen

Für die Berechnung wird ein Verfahren eingesetzt, bei dem der dreidimensionale Raum in zweidimensionale Schnitte unterteilt wird. Für jeden Schnitt wird unabhängig voneinander der Potenziallinienverlauf berechnet, der sich dann sukkzessiv zu einer Äquipotenzialfläche zusammensetzen lässt.

Für die Äquipotenziallinienberechnung im zweidimensionalen Raum wird das von Lorensen [10] entwickelte *Marching Squares* Verfahren eingesetzt, welches, basierend auf den berechneten Potenzialwerten der Gitterpunkte, mittels linearer Interpolation die Äquipotenttiallinen für ein gegebenes Potenzial errechnet. Dabei werden jeweils vier benachbarte Gitterpunkte als eine Zelle betrachtet, deren Gitterpunkt-Potenzialwerte auf den Liniensegmenten paarweise verglichen werden können. Ist einer der beiden Potenzialwerte über und der andere Potenzialwert unter dem gesuchten Isopotenzial, wird auf dem Segment ein Punkt der Äquipotenziallinie berechnet.

Durch Ausführung dieser Operation in jeder Zelle der Ebene und anschließender Verbindung aller Punkte kann eine Äquipotenziallinie mit zur Gitterkonstanten umgekehrt proportionalen Genauigkeit gezeichnet werden. Für eine Ausgabe von Feldstärkenlinien werden die Trajektorien masseloser Teilchen durch Bahnintegration berechnet und dreidimensional dargestellt, eine schrittweise Berechnung entfällt.

Ein Vorteil dieses Verfahrens gegenüber dem Suchlinien und dem *Predictor-Corrector*-Verfahren besteht darin, dass jede Äquipotenziallinie gefunden wird, auch bei mehrfachem Auftreten.

Durch ein Übereinanderlegen der Äquipotenziallinien der jeweiligen aneinandergrenzenden Schnitte entsteht eine anschauliche grafische Darstellung der

Potenzialverteilung. Durch Drehung, Verschiebung und Vergrößerung/Verkleinerung können Teilbereiche, die den Nutzer besonders interessieren, hervorgehoben werden.

Ein Beispiel einer dreidimensionalen Feldverteilung ist in Abbildung 3-4 und Abbildung 3-5 gezeigt. Die Geometrie entspricht der eines dreidimensionalen elektrolytischen Trogs, bei der zwischen zwei auf 0V und 6V geladenen Platten eine dritte auf 10 V geladene Platte mit mittiger Durchlassöffnung eingestellt wird.

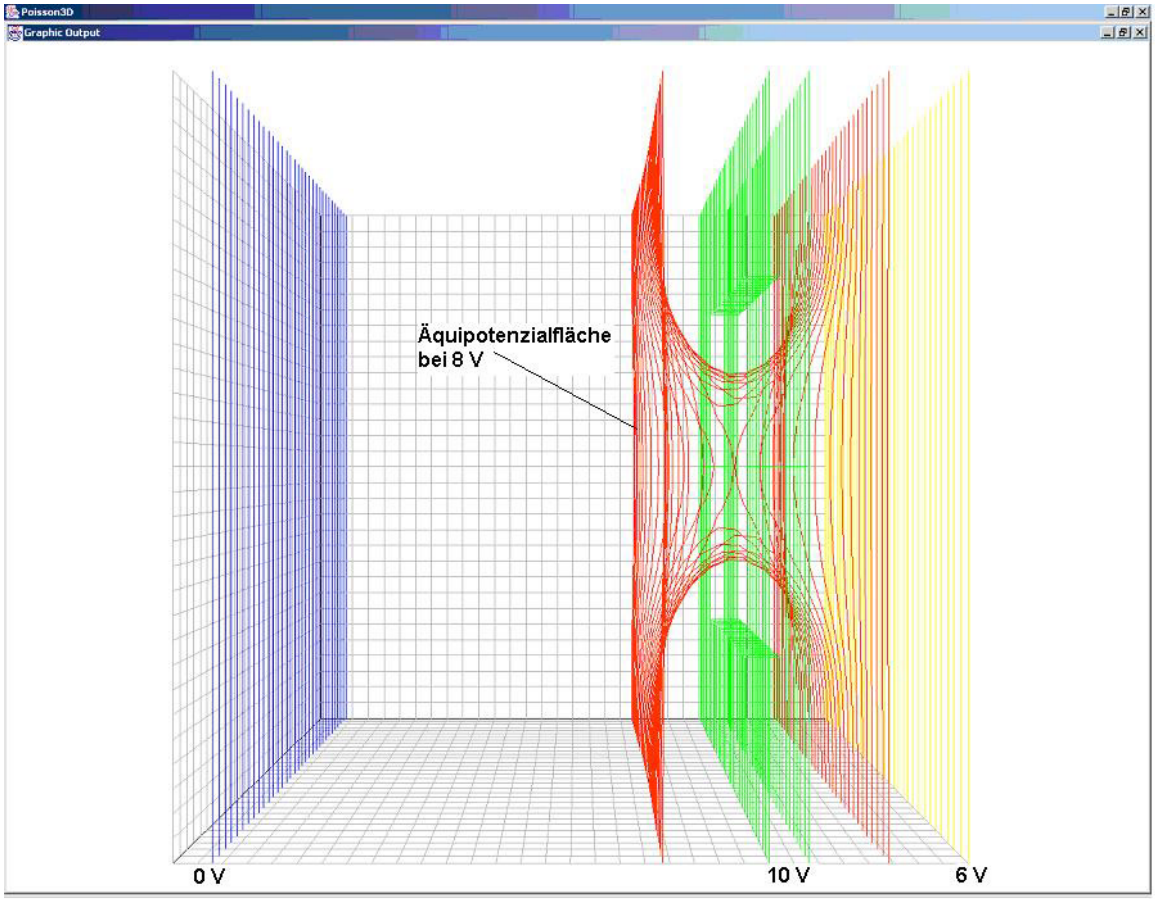

Abbildung 3-4: Grafische Ausgabe eines dreidimensionalen Äquipotenzialflächenverlaufs
Die geladenen Platten sind blau (0V), grün (10V) und gelb (6V) eingezeichnet, die Äquipotenzialfläche (8 V) ist rot dargestellt.

Dreht man die grafische Repräsentation der Geometrie und der Äquipotenzialfläche um die y-Achse, so werden weitere Charakteristika sichtbar:

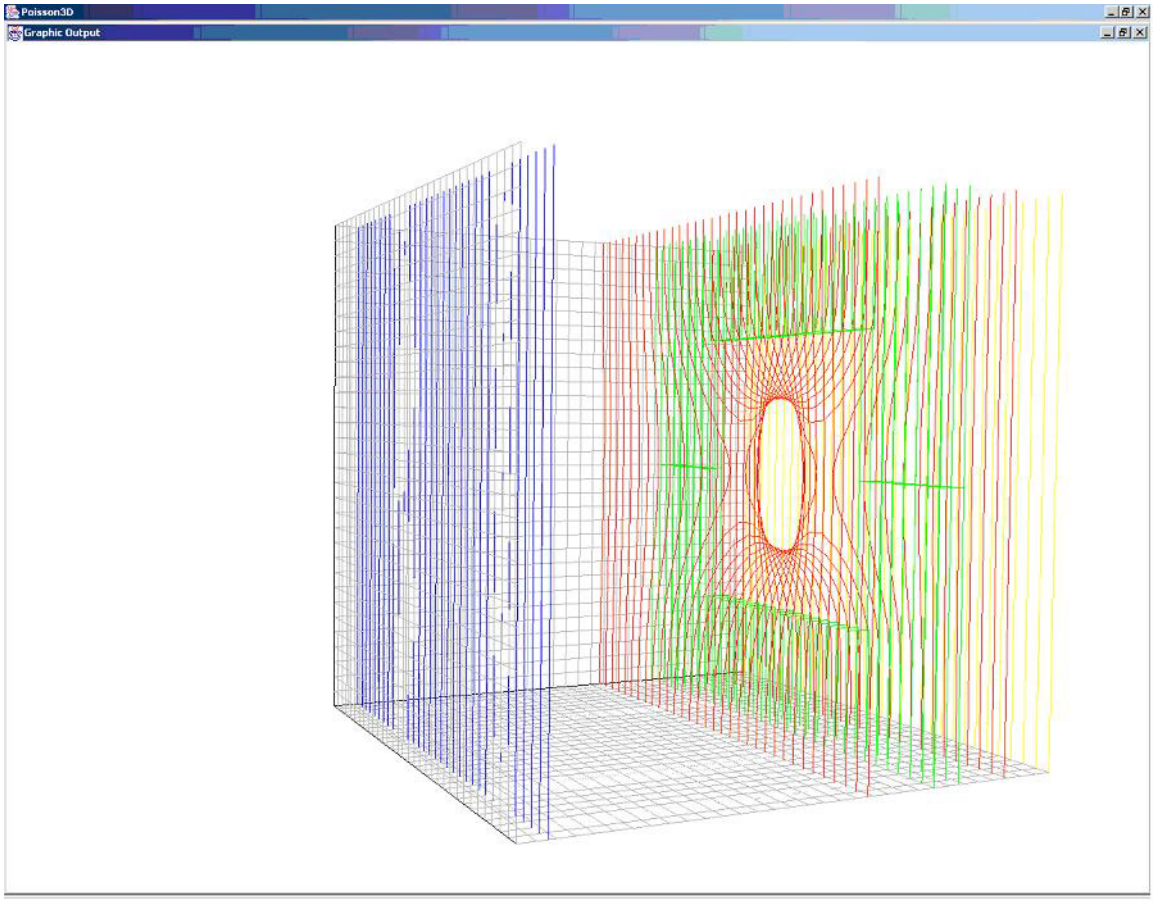

Abbildung 3-5: Der gleiche Äquipotenzialflächenverlauf bei Drehung der Darstellung um 30° nach rechts

Deutlich zu erkennen ist hier der Verlauf der Potenzialfläche durch die Durchlassöffnung der mittleren geladenen Platte.

4. Berechnung elektrischer Feldstärken

Eine Berechnung der elektrischen Feldstärken ist unter Anderem für die Bahnberechnung von geladenen Partikeln notwendig. Normalerweise wird die Feldstärke direkt aus den errechneten Potenzialwerten abgeleitet.

4.1. Homogene und analytisch definierte elektrische Felder

Die Berechnungen aus den Potenzialwerten sind in der Java-Klasse *ElectricFieldOfPotential* enthalten. Bei der Berechnung werden die elektrischen Feldstärken direkt aus den bereits errechneten diskreten Potenzialwerten interpoliert. Ist die Feldstärke jedoch im gesamten Feld homogen, kann sie auch direkt als fester Wert in der Eingabedatei definiert werden. Sind die Feldstärken stationär, aber nicht homogen, können sie im Bedarfsfall auch in analytischer Form definiert werden: Dafür wird die Java-Klasse *AnalyticalElectricField* mit der Formel für die Berechnung der Feldstärke erweitert. Das folgende Kapitel beschreibt die Feldstärkenberechnung aus einem vorher errechneten Potenzialfeld.

4.2. Feldstärkeninterpolation aus diskreten Potenzialwerten

Die elektrische Feldstärke $\vec{E}$ ermittelt sich direkt aus dem Gradienten des Potenzialfeldes gemäß

$$\vec{E} = -\text{grad}U \tag{4.1}$$

und damit im Fall diskretisierter Potenzialwerte analog zur Potenzialfeldinterpolation zwischen den Maschenknoten für die x-Komponente zu

$$E_x(x_0+x, y_0+y, z_0+z) = -\frac{\partial}{\partial x} U(x_0+x, y_0+y, z_0+z) \ . \tag{4.2}$$

Mit dem Einsetzen von Gleichung (3.30) und einer Taylor-Näherung bis zur zweiten Ordnung folgt für

$$\begin{aligned} E_x(x_0+x, y_0+y, z_0+z) = {} & \frac{U(x_0+h_x)-U(x_0-h_x)}{2h_x} + x\frac{U(x_0+h_x)+U(x_0-h_x)-2U_0}{h_x^{\,2}} + \\ & + y\frac{U(x_0+h_x, y_0+h_y)+U(x_0-h_x, y_0-h_y)-U(x_0+h_x, y_0-h_y)-U(x_0-h_x, y_0+h_y)}{4h_x h_y} + \\ & + z\frac{U(x_0+h_x, z_0+h_z)+U(x_0-h_x, z_0-h_z)-U(x_0+h_x, z_0-h_z)-U(x_0-h_x, z_0+h_z)}{4h_x h_z} \end{aligned} \tag{4.3}$$

Analog erhält man die y- und z-Komponente der Feldstärke durch zyklisches Vertauschen der Komponenten bei partieller Differentiation nach y bzw. z:

$$\begin{aligned} E_y(x_0+x, y_0+y, z_0+z) = {} & \frac{U(y_0+h_y)-U(y_0-h_y)}{2h_y} + y\frac{U(y_0+h_y)+U(y_0-h_y)-2U_0}{h_y^{\,2}} + \\ & + x\frac{U(x_0+h_x, y_0+h_y)+U(x_0-h_x, y_0-h_y)-U(x_0+h_x, y_0-h_y)-U(x_0-h_x, y_0+h_y)}{4h_x h_y} + \\ & + z\frac{U(y_0+h_y, z_0+h_z)+U(y_0-h_y, z_0-h_z)-U(y_0+h_y, z_0-h_z)-U(y_0-h_y, z_0+h_z)}{4h_y h_z} \end{aligned} \tag{4.4}$$

und

$$\begin{aligned} E_z(x_0+x, y_0+y, z_0+z) = {} & \frac{U(z_0+h_z)-U(z_0-h_z)}{2h_z} + z\frac{U(z_0+h_z)+U(z_0-h_z)-2U_0}{h_z^{\,2}} + \\ & + x\frac{U(x_0+h_x, z_0+h_z)+U(x_0-h_x, z_0-h_z)-U(x_0+h_x, z_0-h_z)-U(x_0-h_x, z_0+h_z)}{4h_x h_z} + \\ & + y\frac{U(y_0+h_y, z_0+h_z)+U(y_0-h_y, z_0-h_z)-U(y_0+h_y, z_0-h_z)-U(y_0-h_y, z_0+h_z)}{4h_y h_z} \end{aligned} \ . \tag{4.5}$$

5. Trajektorienberechnung

5.1. Bewegungsgleichungen

Bewegt sich ein Partikel mit der Masse m und der Ladung q auf einer Trajektorie r(t) mit einer Geschwindigkeit v(t) durch eine Region mit elektrostatischem Feld E(r) und magnetischem Feld B(r), so gilt die Lorentzkraft

$$F = q\left[\vec{E} + \left(\vec{v} \times \vec{B}\right)\right] . \tag{5.1}$$

Durch Gleichsetzen kann die Bewegungsgleichung ausgedrückt werden durch

$$q\left[\vec{E} + \left(\frac{d}{dt}\vec{r} \times \vec{B}\right)\right] = m\frac{d^2}{dt^2}\vec{r} . \tag{5.2}$$

Dies stellt eine gewöhnliche Differentialgleichung (DGL) zweiter Ordnung dar, die durch Einsetzen der Impulsgleichung p=mv umgewandelt werden kann in zwei gekoppelte Differentialgleichungen erster Ordnung

$$\frac{d}{dt}\vec{p}(t) = q\left[\vec{E} + \left(\vec{v}(t) \times \vec{B}\right)\right] \tag{5.3},$$

und

$$\vec{p}(t) = m\vec{v}(t) . \tag{5.4}.$$

Im dreidimensionalen Raum sind die sechs Koordinaten des entstehenden Vektors die Komponenten des Orts- und Impulsvektors r_x, r_y, r_z, p_x, p_y, und p_z.

Dabei gilt für die Masse im relativistischen Fall

$$m = \frac{m_0}{\sqrt{1 - \frac{v^2}{c^2}}} \quad . \tag{5.5}$$

5.2. Approximationsverfahren zur Lösung von gewöhnlichen Differentialgleichungen

Möchte man bei bekanntem Ortsvektor $\vec{r_n}$ zum Zeitpunkt t_n die Position des Partikels zum Zeitpunkt $t_{n+1} = t_n + h$ ermitteln, kann mittels einer Taylor-Näherung der nächste Punkt beliebig genau approximiert werden:

$$\vec{r}(t_{n+1}) = \lim_{m\to\infty} \sum_{k=0}^{m} \frac{h^k}{k!} \left(\frac{d}{dt}\right)^k \vec{r}(t_n) = \vec{r}(t_n) + h\frac{\vec{r}'(t_n)}{1!} + h^2\frac{\vec{r}''(t_n)}{2!} + \ldots \tag{5.6}$$

Bei der Diskretisierung kann Eulers Näherungsformel erster Ordnung herangezogen werden, bei der die Taylor-Entwicklung nach dem zweiten Term (also der ersten Ableitung) abgebrochen wird:

$$\overrightarrow{r_{n+1}} \approx \vec{r_n} + hf(t_n, \vec{r_n}) \tag{5.7}$$

mit

$$f = \frac{d}{dt}\vec{r_n} \tag{5.8}$$

Allerdings spricht ein hoher Fehler der Ordnung $\sigma(h^2)$ gegen die Nutzung der Euler'schen Methode.

In der Praxis hat sich eine von Runge und Kutta entwickelte Methode durchgesetzt, bei der der zu approximierende Schritt in mehrere Teilschritte zerlegt wird und die Teilschritte mit entsprechender Gewichtung in die Ganzschrittberechnung einfließen.

Das hier vorgestellte Programm verwendet unter Anderem eine Runge-Kutta Methode vierter Ordnung, bei der bei jedem Schritt viermal die Ableitung berechnet wird:

- Einmal am Startpunkt,
- zweimal an den approximierten Mittelpunkten und
- einmal am approximierten Endpunkt.

Der endgültige Funktionswert wird dann von diesen (verschieden gewichteten) Ableitungen berechnet. Ausgehend von

$$\vec{k_1} = hf(t_n, \vec{r_n}) \text{ (Steigung am Startpunkt)} \tag{5.9}$$

$$\vec{k_2} = hf(t_n + \frac{h}{2}, \vec{r_n} + \frac{\vec{k_1}}{2}) \text{ (Steigung an einem geschätzten Mittelpunkt)} \tag{5.10}$$

$$\vec{k_3} = hf(t_n + \frac{h}{2}, \vec{r_n} + \frac{\vec{k_2}}{2}) \text{ (Steigung an einem weiteren geschätzten Mittelpunkt)} \tag{5.11}$$

$$\vec{k_4} = hf(t_n + h, \vec{r_n} + \vec{k_3}) \text{ (Steigung am geschätzten Endpunkt)} \tag{5.12}$$

setzt sich der Endpunkt aus den vier Vektoren $\vec{k_1}$, $\vec{k_2}$, $\vec{k_3}$ und $\vec{k_4}$ zusammen zu

$$\vec{r_{n+1}} = \vec{r_n} + \frac{\vec{k_1}}{6} + \frac{\vec{k_2}}{3} + \frac{\vec{k_3}}{3} + \frac{\vec{k_4}}{6} + \sigma(h^5) \ . \tag{5.13}$$

Das hier beschriebene Programm bietet vier verschiedene Integrationsmethoden an, die über das Setzen eines einfachen Schlüsselwortes in der Eingabedatei ausgewählt werden können. Neben dem Integrationsverfahren nach Euler (Schlüsselwort *EULER*) und

Runge-Kutta (Schlüsselwort *RK4*), dessen feste Integrationsschrittweite in der Eingabedatei definiert werden, stehen zusätzlich noch zwei Verfahren mit variabler Integrationsschrittweite zur Auswahl: zum Einen ein eingebettetes Runge-Kutta-Verfahren achter Ordnung nach Dormand-Prince (Schlüsselwort *DP8*) [12] und zum Anderen ein Verfahren nach Gregg-Bulirsch-Stoer (Schlüsselwort *GBS*) [13]. Die in der Eingabedatei definierte Schrittweite wird hier als Startschrittweite verwendet, dann von den Algorithmen aber entsprechend zur gewünschten Genauigkeit modifiziert.

Um mit Hilfe der Java-Klasse *Trajector3D* eine Partikelbahn zu berechnen, muss in der Eingabedatei ein entsprechender Abschnitt mit dem Schlüsselwort *TRAJECTORY* eingeleitet werden. In diesem Abschnitt können dann Ladung, Masse, Startposition und Anfangsenergie des zu simulierenden Partikelclusters festgelegt werden. Weitere Parameter bezüglich der Trajektorienberechnung (wie zum Beispiel die Anzahl der Rechnungsschritte innerhalb einer Masche) oder der Grafikausgabe können zusätzlich eingefügt werden. Ein Beispiel hierzu ist im Anhang 9.1.3 angeführt.

Die berechneten Trajektorien können numerisch oder grafisch ausgegeben werden. Das Programm fügt je simuliertem Partikel genau eine Linie ein. Ein Beispiel findet sich hierzu im Kapitel 6 „Anwendungen von Poisson3D“.

5.3. Raumladungseffekte von Trajektorien

Bewegen sich geladene Teilchen durch einen Raum, wird dort eine sogenannte Raumladung induziert (Q: Ladung des Teilchens, V: Volumen des umgebenden Raums).

$$\rho = \frac{Q}{V} \tag{5.14}$$

Ordnet man diesem Teilchen einen Strahlstrom I zu und betrachtet man die Raumladung in einer Masche des Gitters mit Volumen $V_{mesh} = h_x \cdot h_y \cdot h_z$, so gilt für jeden Iterationsschritt der Bahnberechnung

$$\rho_{mesh} = \frac{I \cdot \Delta t}{V_{mesh}} \tag{5.15}$$

mit Δt als der Zeitdifferenz zum letzten Iterationsschritt.

Die durch die bewegten Ladungen induzierte Raumladung wird in der Java-Klasse *SpaceCharges* berechnet. Dabei werden die Raumladungsbeträge der Maschenvolumen gewichtet auf die benachbarten Maschenknoten aufgerechnet. Der Gewichtungsfaktor für jeden Maschenknoten entspricht dabei dem jeweiligen Volumen, dass die Position des Ladungsträgers mit dem gegenüberliegenden Maschenknoten aufspannt.

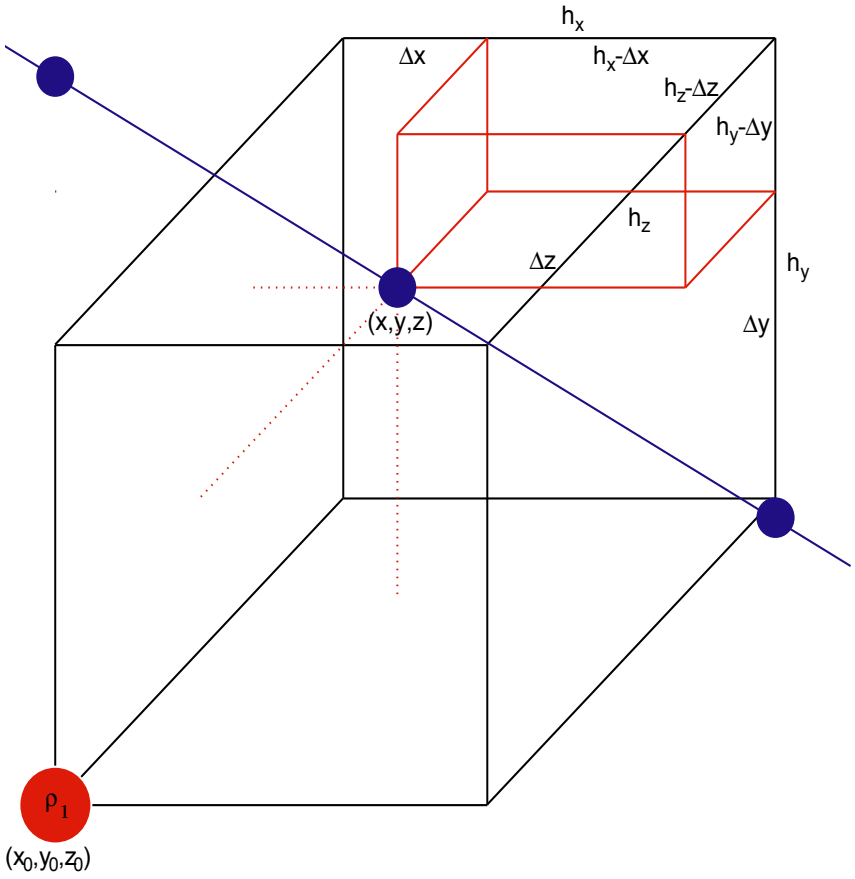

Abbildung 5-1: Raumladungsinterpolation in Maschen

Betrachtet man die Abbildung 5-1, so bezeichnet (x, y, z) die Position des Ladungsträgers innerhalb einer Masche und es gilt $x = x_0 + \Delta x$, $y = y_0 + \Delta y$ und $z = z_0 + \Delta z$ mit (x_0, y_0, z_0) als Position eines Maschenknotens und (Δx, Δy, Δz) als Abstand des Knotens zur Ladung. Die Masche wird begrenzt durch die Knoten an den Positionen (x_0,y_0,z_0), (x_0+h_x,y_0,z_0), (x_0,y_0+h_y,z_0), (x_0+h_x,y_0+h_y,z_0), (x_0,y_0,z_0+h_z), (x_0+h_x,y_0,z_0+h_z), (x_0,y_0+h_y,z_0+h_z) und $(x_0+h_x,y_0+h_y,z_0+h_z)$.

Der Raumladungsanteil für den ersten Knoten („vorne unten links“), in der Abbildung 5-1 rot eingezeichnet, lässt sich also bestimmen durch

$$\rho(x_0, y_0, z_0) = \rho \cdot \frac{(h_x - \Delta x)(h_y - \Delta y)(h_z - \Delta z)}{h_x h_y h_z} \tag{5.16}$$

Für die anderen Knoten folgt analog durch Vertauschen der Abstandsvektoren für „vorne unten rechts“

$$\rho(x_0 + h_x, y_0, z_0) = \rho \cdot \frac{(\Delta x)(h_y - \Delta y)(h_z - \Delta z)}{h_x h_y h_z} \quad , \tag{5.17}$$

für „vorne oben links“

$$\rho(x_0, y_0 + h_y, z_0) = \rho \cdot \frac{(h_x - \Delta x)(\Delta y)(h_z - \Delta z)}{h_x h_y h_z} \quad , \tag{5.18}$$

für „vorne oben rechts“

$$\rho(x_0 + h_x, y_0 + h_y, z_0) = \rho \cdot \frac{(\Delta x)(\Delta y)(h_z - \Delta z)}{h_x h_y h_z} \quad , \tag{5.19}$$

für „hinten unten links“

$$\rho(x_0, y_0, z_0 + h_z) = \rho \cdot \frac{(h_x - \Delta x)(h_y - \Delta y)(\Delta z)}{h_x h_y h_z} \quad , \tag{5.20}$$

für „hinten unten rechts“

$$\rho(x_0 + h_x, y_0, z_0 + h_z) = \rho \cdot \frac{(\Delta x)(h_y - \Delta y)(\Delta z)}{h_x h_y h_z} \quad , \tag{5.21}$$

für „hinten oben links“

$$\rho(x_0, y_0 + h_y, z_0 + h_z) = \rho \cdot \frac{(h_x - \Delta x)(\Delta y)(\Delta z)}{h_x h_y h_z} \quad , \tag{5.22}$$

und für „hinten oben rechts“

$$\rho(x_0 + h_x, y_0 + h_y, z_0 + h_z) = \rho \cdot \frac{(\Delta x)(\Delta y)(\Delta z)}{h_x h_y h_z} \quad . \tag{5.23}$$

6. Anwendungen von Poisson3D

6.1. Trajektorien in rotationssymmetrischen Feldern

6.1.1. Untersuchung des Einflusses des Interpolationsfehlers auf die Bahnberechnung

Um den Einfluss des Interpolationsfehlers bei der Feldstärkenberechnung auf die Trajektorie eines Partikels untersuchen zu können, wurde die Geometrie eines rotationssymetrischen Zylinderkondensators gewählt, bei der die elektrostatische Potenzialverteilung auch analytisch bestimmt werden kann. Die Bahnkurve wurde einmal unter Verwendung von elektrischen Feldstärken, basierend auf die numerisch ermittelten Potenziale, und einmal unter Verwendung von analytisch bestimmten elektrostatischen Feldern gerechnet und miteinander verglichen. Die Berechnung der Bahnkurven wurde dabei jeweils mehrmals mit verschiedenen in Kapitel 3 beschriebenen Integrationsmethoden wiederholt, um auch deren Einfluss auf die Bahnbestimmung untersuchen zu können.

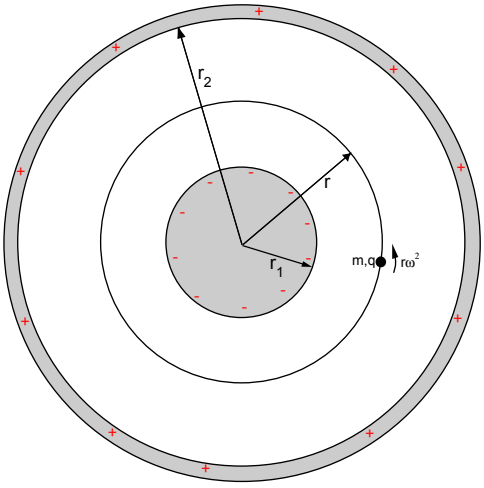

Abbildung 6-1: Zylinderkondensator

Betrachtet man Abbildung 6-1, so folgt für einen Zylinderkondensator mit der Ladung Q und der Länge l aus der Maxwell-Gleichung

$$\oint \mathrm{E}\,\mathrm{dA} = \frac{\mathrm{Q}}{\varepsilon_0} \tag{6.1}$$

und mit $A = 2\pi rl$ für die umschließende Fläche:

$$\mathrm{E_r}(\mathrm{r}) = \frac{\mathrm{Q}}{\varepsilon_0 2\pi \mathrm{l}} \cdot \frac{1}{\mathrm{r}} \ . \tag{6.2}$$

Die Potenzialdifferenz zwischen der inneren Platte U_1 (Radius $r_{1)}$ und der äußeren Platte U_2 (Radius r_2) beträgt damit

$$\mathrm{U_2} - \mathrm{U_1} = \int_{\mathrm{r_1}}^{\mathrm{r_2}} \mathrm{E}\,\mathrm{dr} = \int_{\mathrm{r_1}}^{\mathrm{r_2}} \frac{\mathrm{Q}}{2\pi\varepsilon_0 \mathrm{l}} \cdot \frac{1}{\mathrm{r}}\,\mathrm{dr} = \frac{\mathrm{Q}}{2\pi\varepsilon_0 \mathrm{l}} \ln\left(\frac{\mathrm{r_2}}{\mathrm{r_1}}\right) \tag{6.3}$$

und für ein beliebiges r zwischen r_1 und r_2

$$\mathrm{U(r)} = \mathrm{U(r_1)} + \frac{\mathrm{Q}}{2\pi\varepsilon_0 \mathrm{l}} \ln\left(\frac{\mathrm{r}}{\mathrm{r_1}}\right) = \mathrm{U_1} + \frac{\mathrm{U_2} - \mathrm{U_1}}{\ln \mathrm{r_2} - \ln \mathrm{r_1}} \ln\left(\frac{\mathrm{r}}{\mathrm{r_1}}\right) \ . \tag{6.4}$$

Für die Feldstärke $E_r(r)$ gilt damit

$$\mathrm{E_r}(\mathrm{r}) = -\frac{\partial}{\partial \mathrm{r}} \mathrm{U(r)} = -\frac{\mathrm{U_2} - \mathrm{U_1}}{\ln \mathrm{r_2} - \ln \mathrm{r_1}} \cdot \frac{1}{\mathrm{r}} \tag{6.5}$$

Für ein Partikel mit der Ladung q und der Masse m auf einer kreisförmigen Umlaufbahn um die Zentralladung ist die Coulomb-Kraft gleich der Zentrifugalkraft

$$\mathrm{qE_r} = \mathrm{F_r} = \mathrm{mr}\omega^2 = \mathrm{m}\frac{\mathrm{v_{tan}}^2}{\mathrm{r}} \ . \tag{6.6}$$

Für die Tangentialgeschwindigkeit des umlaufenden Partikels gilt somit

$$v_{tan} = \pm\sqrt{r\frac{q}{m}E_r} = \pm\sqrt{r\frac{q}{m}\cdot\frac{-(U_2-U_1)}{\ln r_2-\ln r_1}\cdot\frac{1}{r}} = \pm\sqrt{-\frac{q}{m}\cdot\frac{U_2-U_1}{\ln r_2-\ln r_1}} \quad . \tag{6.7}$$

Abweichungen eines sich mit v_0 um die Zylinderachse rotierenden Partikels von der Kreisbahn schließen im Falle eines analytisch gerechneten Feldes auf einen Fehler im Trajektorienalgorithmus. Bei Verwendung eines numerisch interpolierten Feldes kann dieser Fehler zusätzlich vom Interpolationsverfahren abhängen.

Die folgende Abbildung 6-2 zeigt die Geometrie und die gerechneten Partikeltrajektorien bei Programmdurchlauf. Die Eingabedatei „Zylinder.txt“, die zur Definition der Geometrie, der Feldstärken und des Strahls verwendet wurde, als auch die Konsolenausgabe, finden sich als Ausdrucke im Anhang 9.1.1 und 9.1.2.

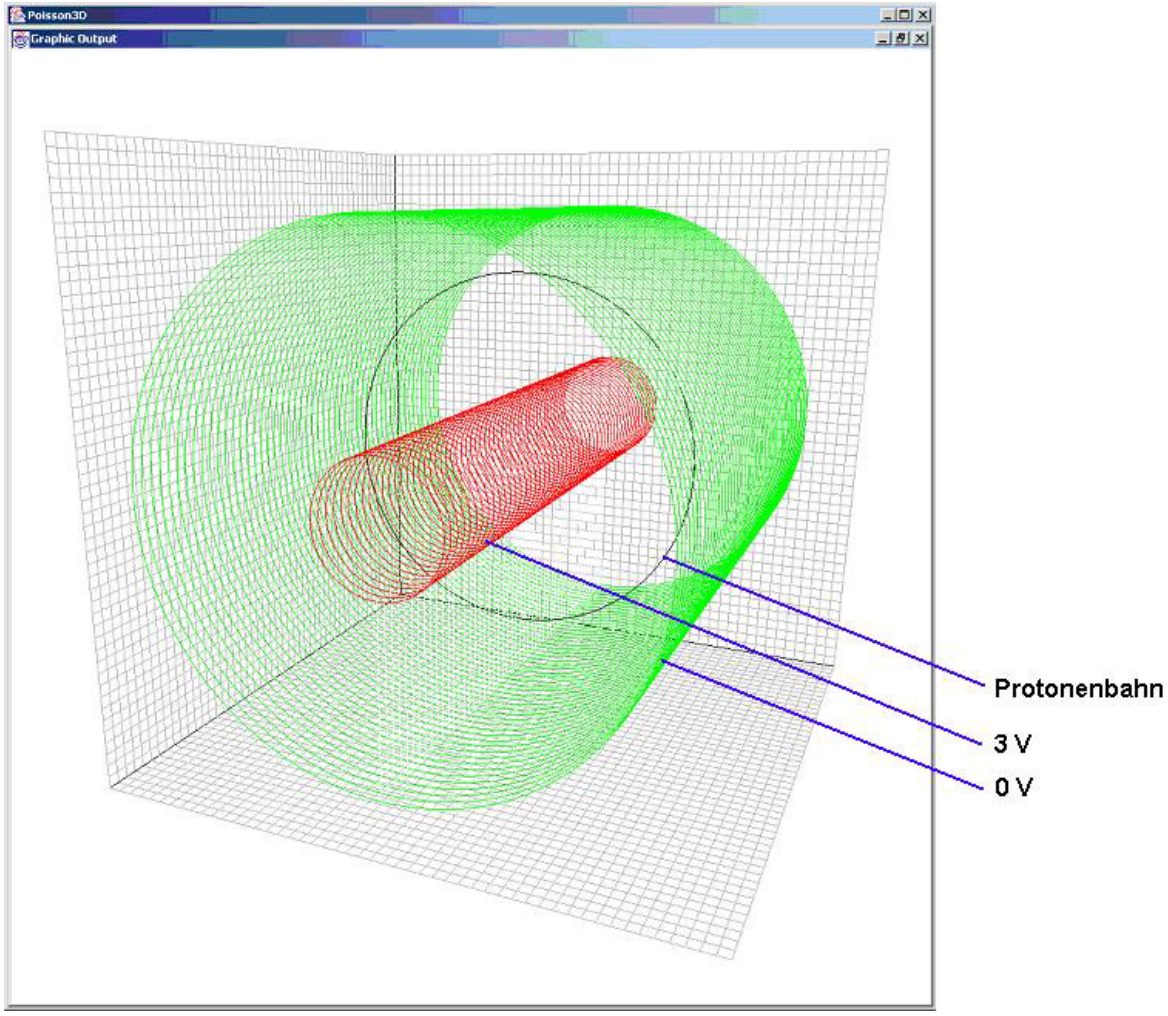

Abbildung 6-2: Grafische Ausgabe der Zylindergeometrie und der Trajektorie (Schrägansicht)
Die Abbildung zeigt den Zylinderkondensator mit einer auf 3V negativ geladenen Innenschale (rot) und einer auf 0V geerdeten Außenschale (grün). Darin bewegt sich ein Proton (schwarz) auf einer Kreisbahn um die Zylinderachse mit einer Geschwindigkeit von 14.397 $\frac{m}{s}$.

6.1.2. Vergleich der Integrationsmethoden

In der folgenden Abbildung 6-3 sind die errechneten Kreisbahnen für zwei Vollumdrehungen bei Verwendung verschiedener Integrationsverfahren eingezeichnet. Dabei sind Trajektorien nach Euler in violett, nach Runge-Kutta(4) in rot, nach Dormand-Prince(8) in blau und nach Gregg-Bulirsch-Stoer in blaugrün eingezeichnet. Zum Vergleich ist zusätzlich die Sollumlaufbahn in schwarz eingezeichnet.

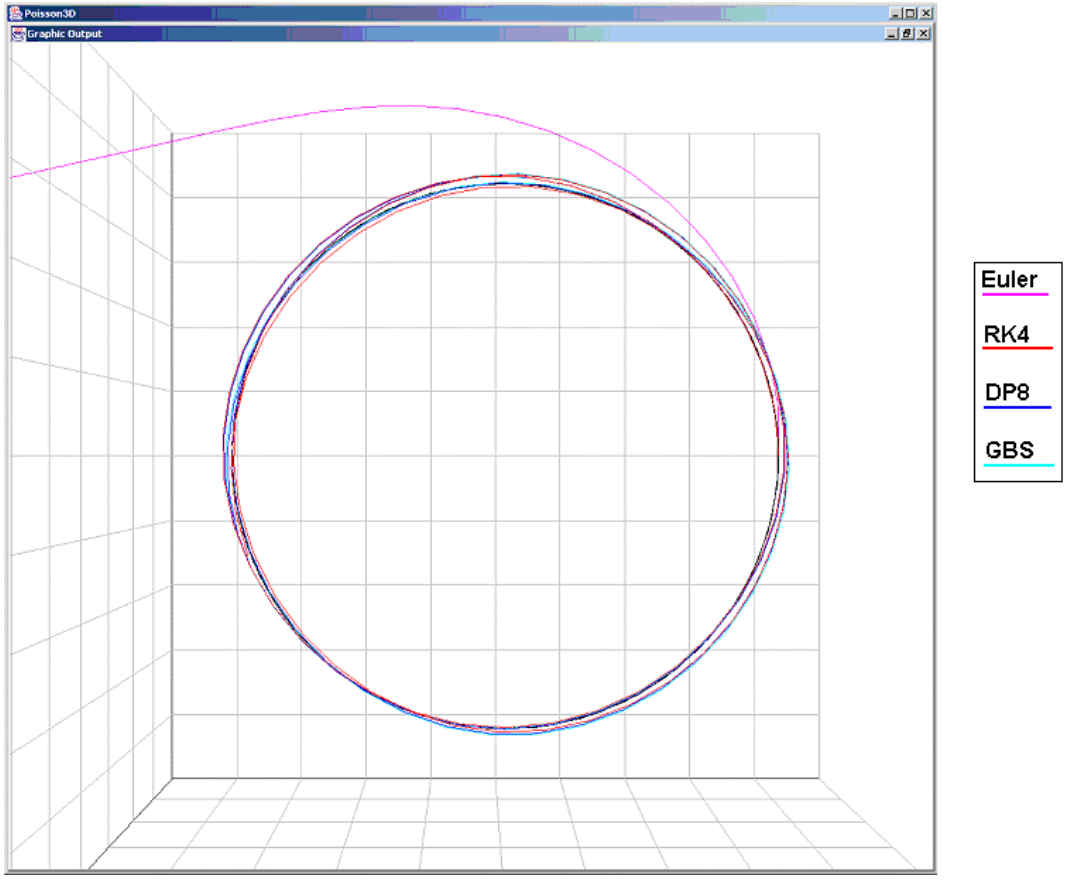

Abbildung 6-3: Grafische Ausgabe von Trajektorien aus verschiedenen Integrationsmethoden

Da eine quantitative Abschätzung aufgrund der grafischen Ausgabe hier nicht möglich ist, geben die folgenden Diagramme die normierten Abweichungen $\frac{\Delta r}{r}$ von der Sollumlaufbahn gegen die Anzahl der Umläufe wieder.

Bei einer Schrittweite von 0,5 Maschen entspricht dabei ein Umlauf 190 Iterationsschritten, bei einer Schrittweite von 0,25 Maschen entsprechend 380 Iterationsschritten.

Jede Umlaufbahn wurde mit jeweils 4 verschiedenen Integratoren berechnet und in das selbe Diagramm eingetragen:

- Diagramm 6-1 und Diagramm 6-2 vergleichen die zwei gewählten Schrittweiten bei der Verwendung eines analytisch gerechneten elektrischen Feldes nach Gleichung (6.5),
- Diagramm 6-3 und Diagramm 6-4 für ein numerisches Feld mit 50x50x10 Gitterknoten, dessen E-Felder direkt aus dem Potenzialfeld interpoliert werden.
- Diagramm 6-5 und Diagramm 6-6 schließlich tragen die Abweichungen bei einer verbesserten Feldstärkeninterpolation nach Gleichung (4.3) auf.

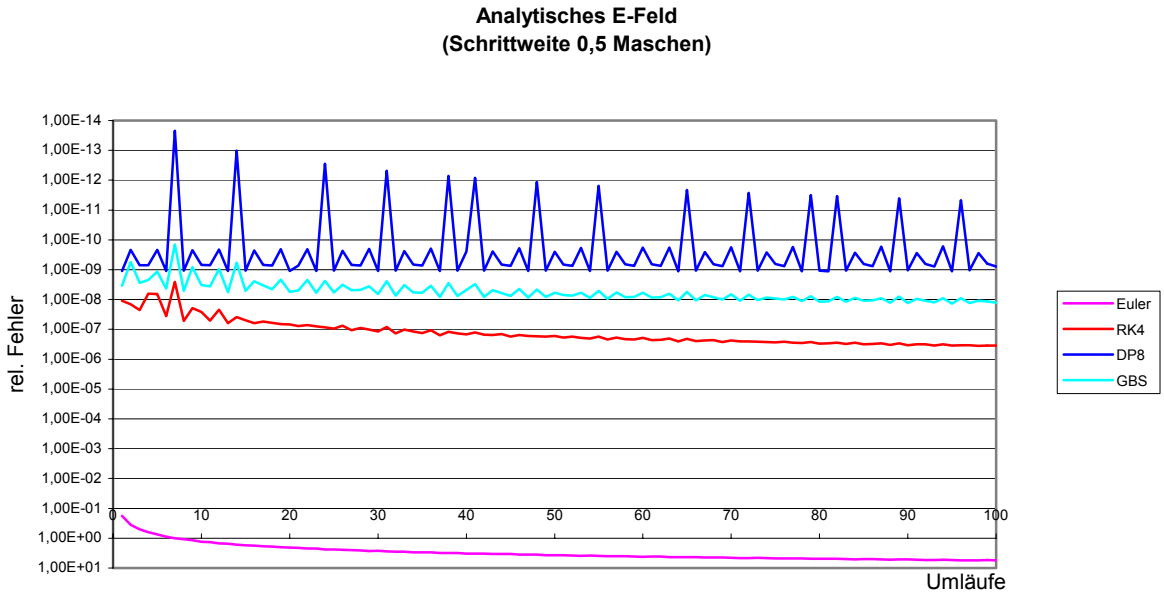

Diagramm 6-1: Abweichung von der Sollumlaufbahn, normiert auf Einheitskreis

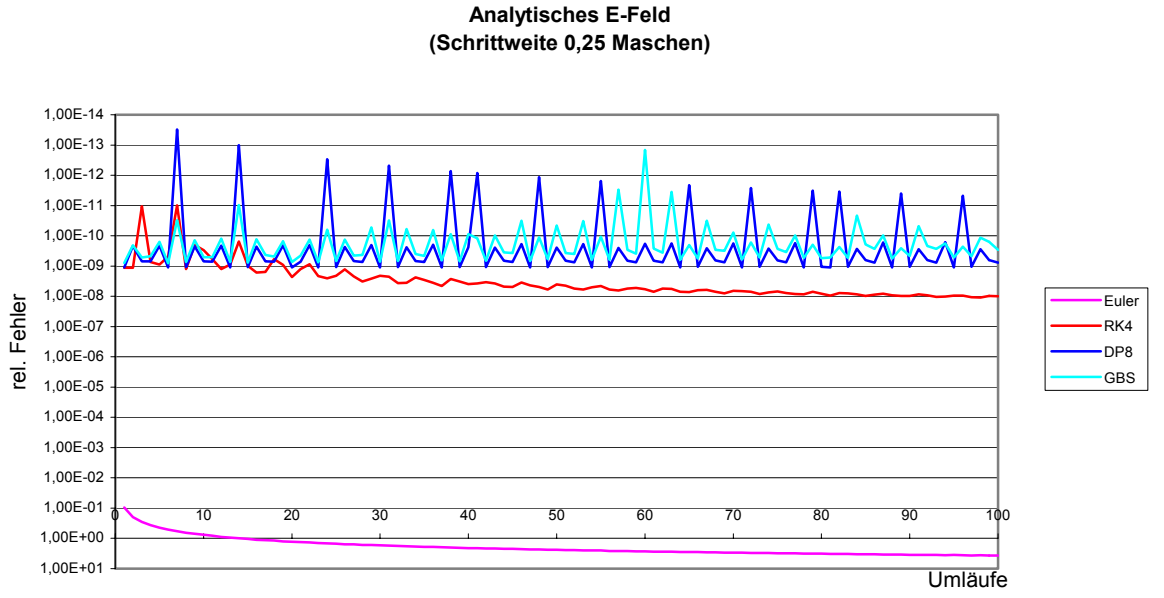

Diagramm 6-2: wie oben, jedoch bei Halbierung der Integrationsschrittweite

Die verschiedenen Integrationsalgorithmen verhalten sich im analytischen Feld wie erwartet: Die Bahnabweichung ist umgekehrt proportional zur Ordnung der Integrationsmethode. Bei Methoden mit fester Schrittweite ist der Fehler abhängig von der gewählten Schrittweite, während er bei einer Integration nach Dormand-Prince keine Auswirkung auf die Genauigkeit hat. (Die beobachtete Verringerung des relativen Fehlers bei GBS durch Verringerung der Schrittweite ist unerwartet, aber nicht Gegenstand dieser Untersuchung.)

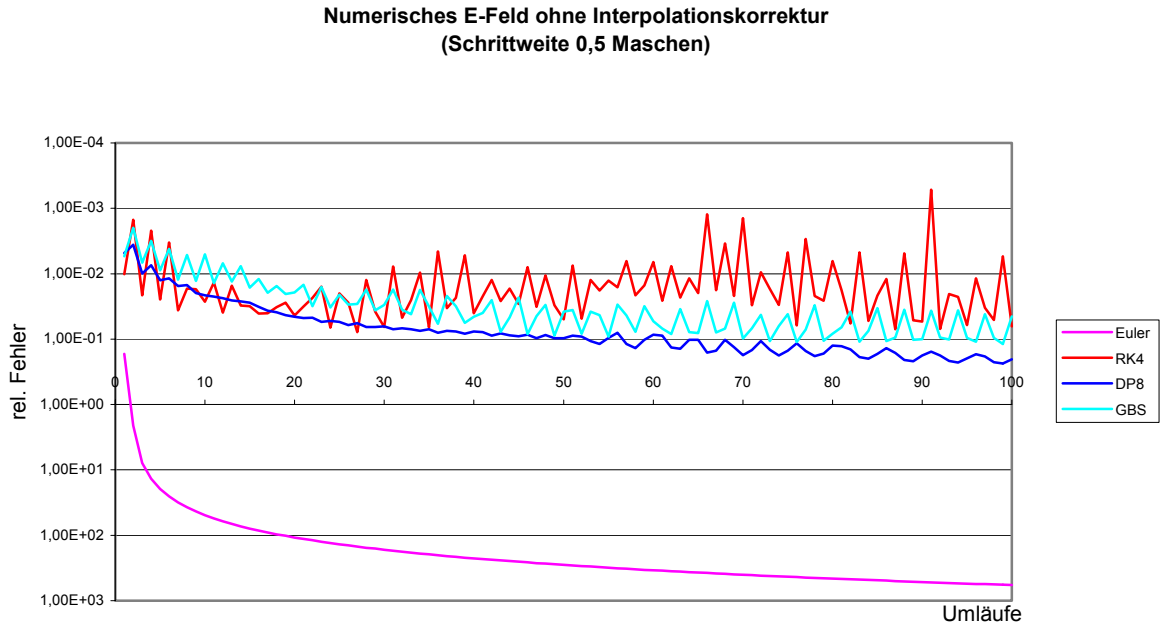

Diagramm 6-3: Abweichung von der Sollbahn in numerischen E-Feldern, ohne Interpolation

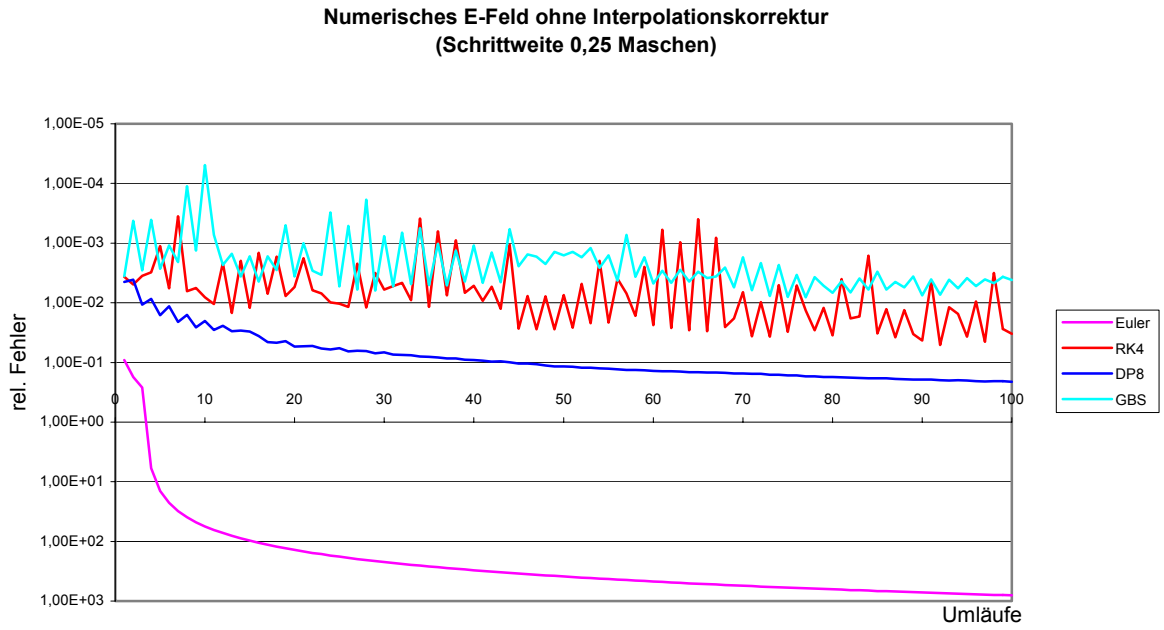

Diagramm 6-4: wie oben, bei halber Schrittweite

Zieht man als Grundlage für die Bahnberechnung ein Potenzialfeld heran, welches durch Lösung der Poisson- bzw. Laplacegleichung berechnet wurde, so entspricht die beobachtete Abweichung bei der Bahnsimulation nicht mehr den Erwartungen. Durch Wahl einer geeigneten Schrittweite ist eine Integration nach Runge-Kutta genauer als die durch Dormand-Prince.

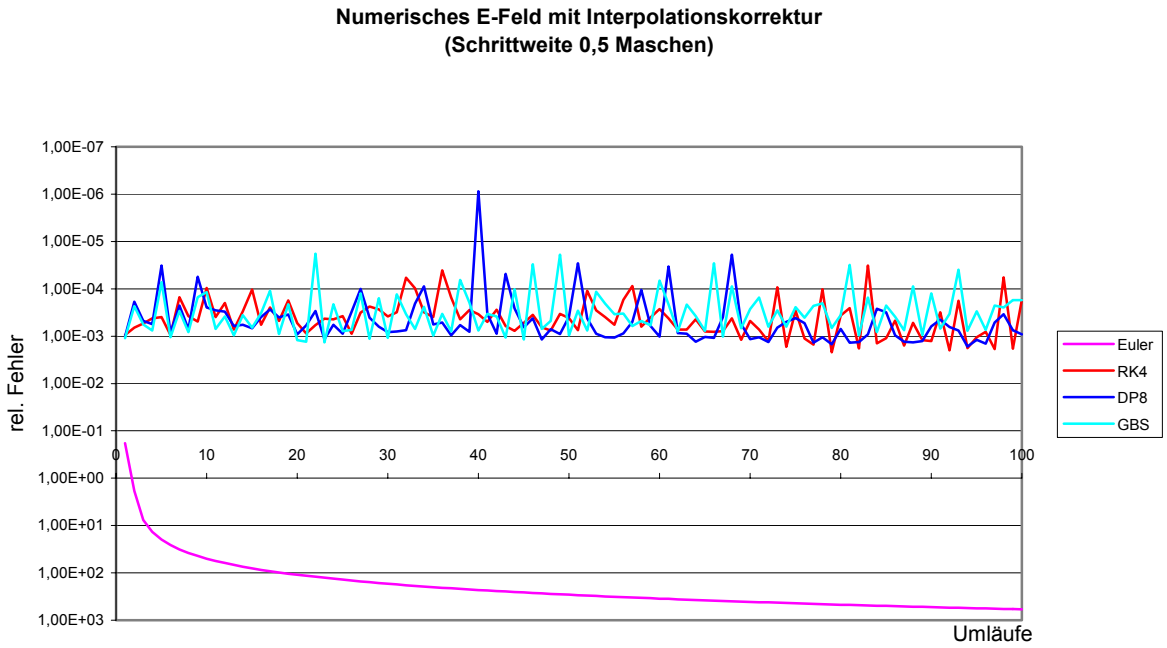

Diagramm 6-5: Abweichung bei verbesserter Feldinterpolation

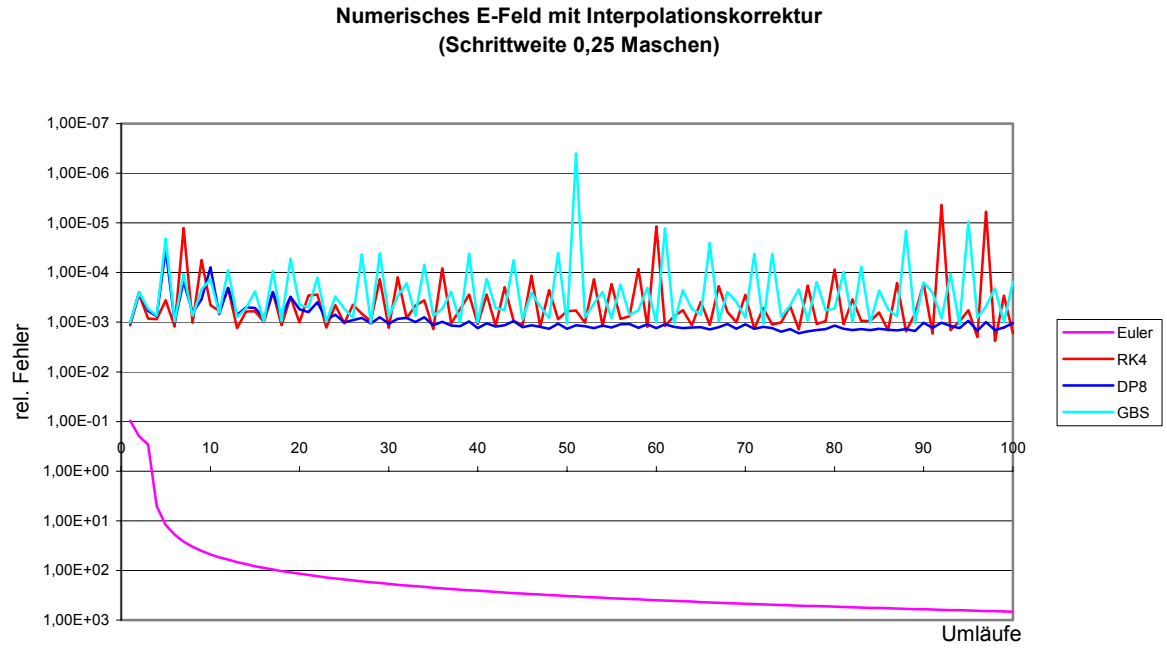

Diagramm 6-6:Abweichung bei verbesserter Feldinterpolation und halber Schrittweite

Durch die Einführung einer Interpolationskorrektur zweiter Ordnung lässt sich die Genauigkeit der Bahnberechnung unabhängig von der gewählten Integrationsmethode um zwei Größenordungen erhöhen. Auch der Einfluss der Schrittweite wird vernachlässigbar.

6.2. Separation von koextrahierten Elektronen und H^--Ionen

Um aus einer Ionenquelle gemeinsam extrahierte Elektronen und H^--Ionen im beschleunigten Strahl voneinander zu trennen, wird dieser durch zwei senkrecht zur Strahlachse stehende stationäre magnetische Felder geführt. Die Felder sind dabei dem Betrag nach gleich, haben jedoch umgekehrte Vorzeichen. Ist die magnetische Feldstärke entsprechend gewählt, werden die Elektronen bereits im ersten Feld durch die Lorentzkraft $F_L = q(\vec{v} \times \vec{B})$ senkrecht zur Strahlachse und zum Magnetfeld abgelenkt, während die ebenfalls negativ geladenen Ionen aufgrund ihrer um Faktor 10^3 größeren Masse nur eine leichte Ablenkung erfahren. Nach der Separation wird der Elektronenstrahl in einen negativ geladenen Elektronenkollektor geführt, um dort durch die Potenzialdifferenz zur Strahlkammer abgebremst zu werden. Die Ionen dagegen durchqueren das zweite Magnetfeld, das ihre Abweichung kompensiert. Die folgende Abbildung 6-4 zeigt schematisch die Separation; das Magnetfeld steht hier senkrecht zur Blattebene.

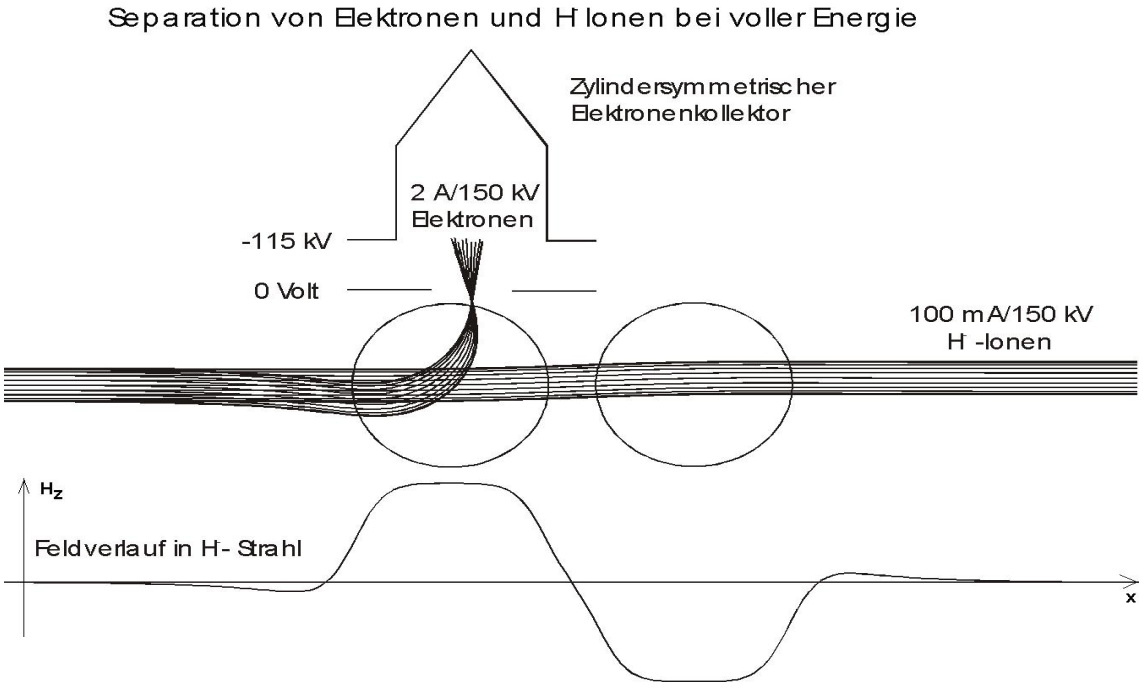

Abbildung 6-4: Schematischer Verlauf der Ionenseparation

Die folgenden Abbildungen zeigen die Simulation einer solchen Elektronen/Ionenseparation. Der auf 150 kV beschleunigte Strahl wird dabei von negativer x-Richtung („links“) kommend in die auf 0 V geerdete Strahlkammer mit 10 cm x 10 cm Grundfläche eingeschossen und dann mittig durch ein in z-Richtung (auf der Blattebene) stehendes 140 mT starkes Magnetfeld in y-Richtung („nach oben“) ausgelenkt. Dabei werden die um 90° abgelenkten Elektronen (in den Abbildungen rot dargestellt) durch eine kreisförmige Öffnung (schwarz) mit 2,2 cm Durchmesser in den oberen Teil der Geometrie geführt, wo sie vor der Ableitung durch den auf 115 kV geladenen zylindersymmetrischen Elektronenkollektor (grün) auf 35 kV abgebremst werden. Die Ionenkomponente des Strahls (blau) erfährt nur eine leichte Auslenkung, die beim Durchlauf des hinter dem Ablenkfeld liegenden zweiten Magnetfeldes gleicher Ausrichtung aber umgedrehten Vorzeichens kompensiert wird. Sie tritt gegenüber der Eintrittöffnung am positiven x-Ende ohne Energieverlust aus der Strahlkammer heraus. Sowohl die Eingabedatei „Extraktion.txt“, die zur Definition der Geometrie, der magnetischen Feldstärken und des Strahls verwendet wurde, als auch die Konsolenausgabe bei Programmdurchlauf sind als Ausdrucke im Anhang 9.1.3 und 9.1.4 aufgeführt.

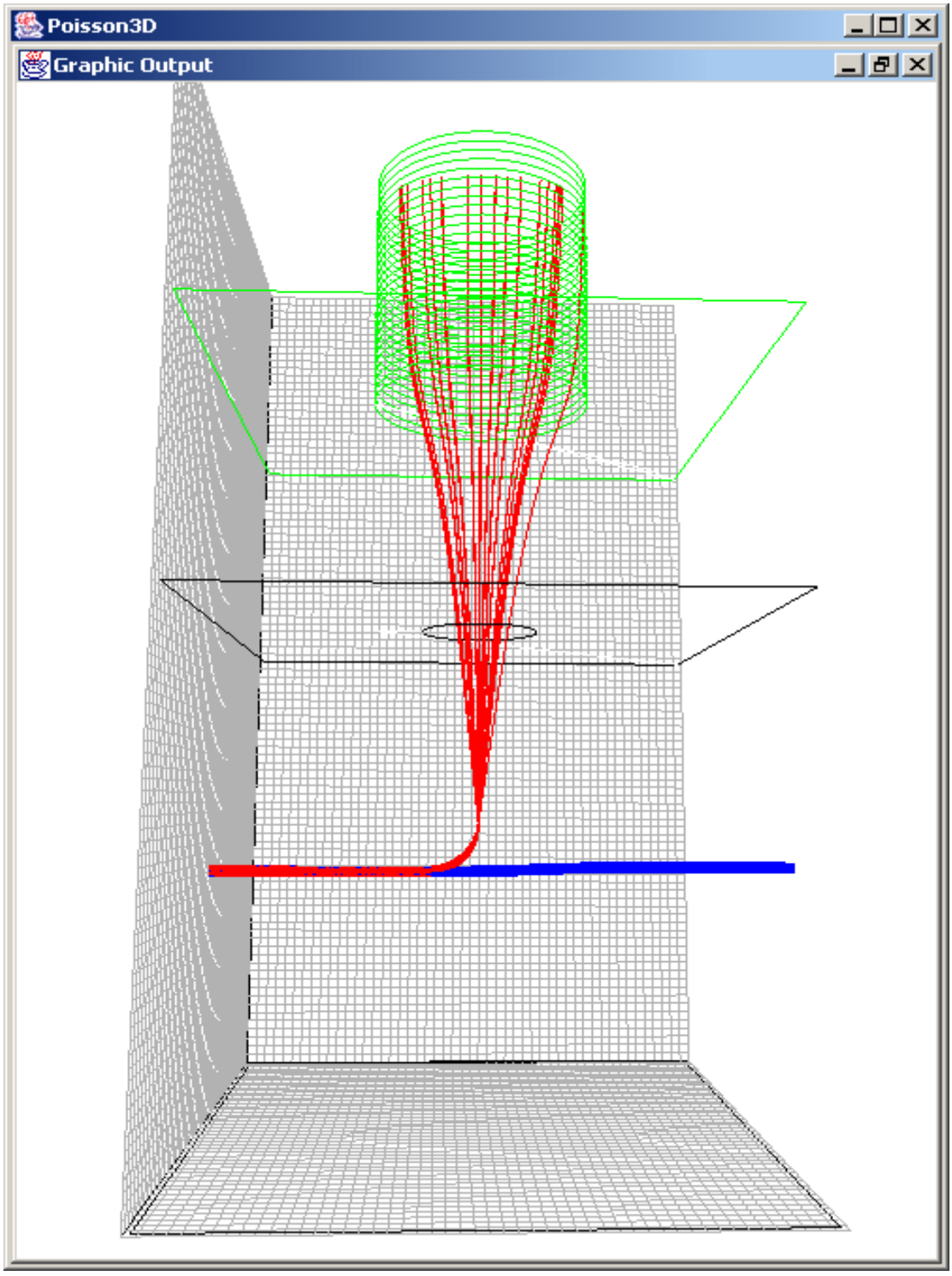

Abbildung 6-5: Separation der Elektronen vom Ionenstrahl, Ansicht von vorne
Die Strahlkammerbegrenzung ist durch zwei schwarzumrandete Platten (oben und unten) gekennzeichnet, der Elektronenkollektor grün.

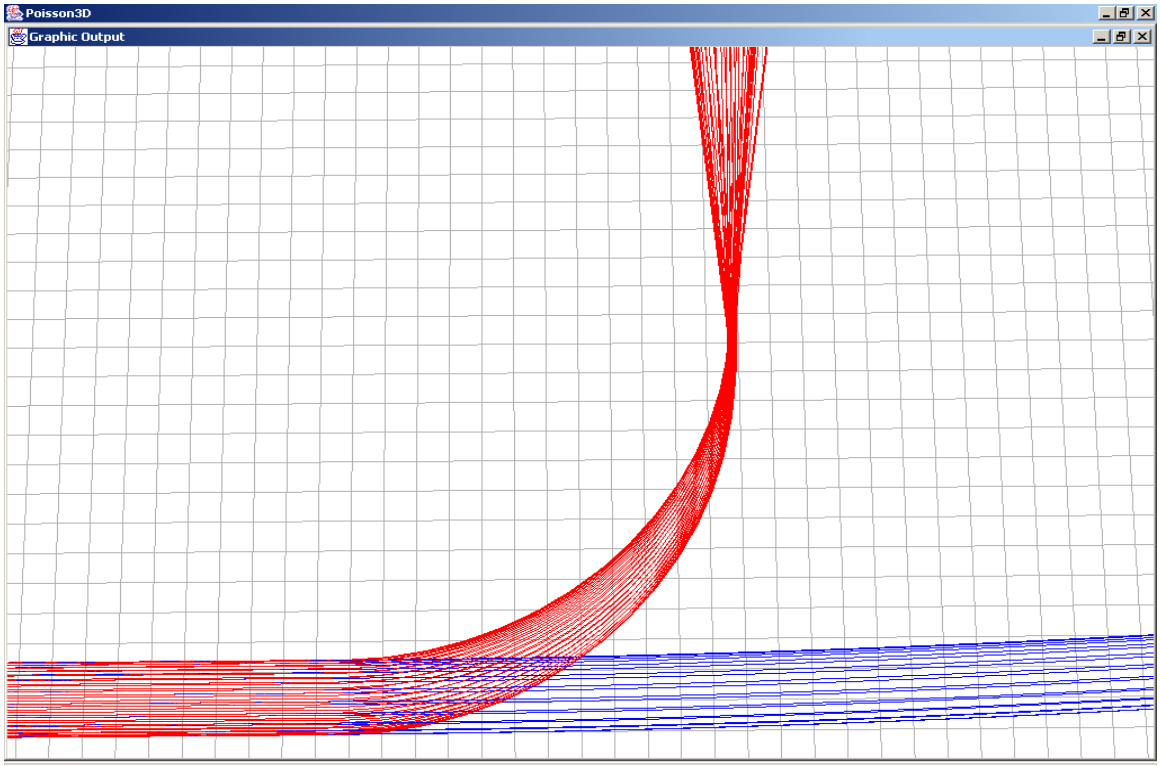

Abbildung 6-6: Vergrößerung des Separationspunktes. Der Elektronenstrahl ist rot, der Ionenstrahl blau eingezeichnet.

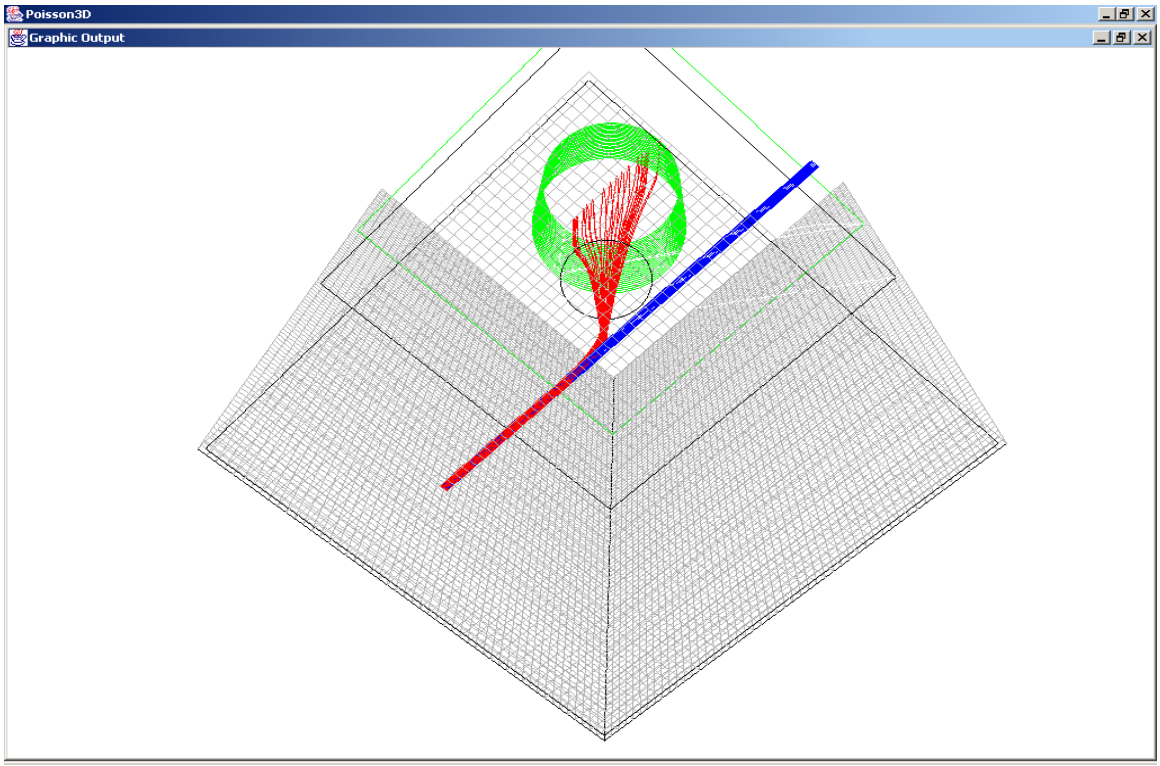

Abbildung 6-7: Ansicht von unten durch die gekippte Grundfläche hindurch.

Abbildung 6-8: Separation der Ionenbahnen aus einer anderen Perspektive.
Die leichte Krümmung des Ionenstrahls und dessen Kompensation durch das zweite Magnetfeld ist hier gut zu erkennen. Am oberen Bildrand ist als kreisförmige Öffnung (schwarz) der Durchlass zum Elektronenkollektor (grün) zu sehen.

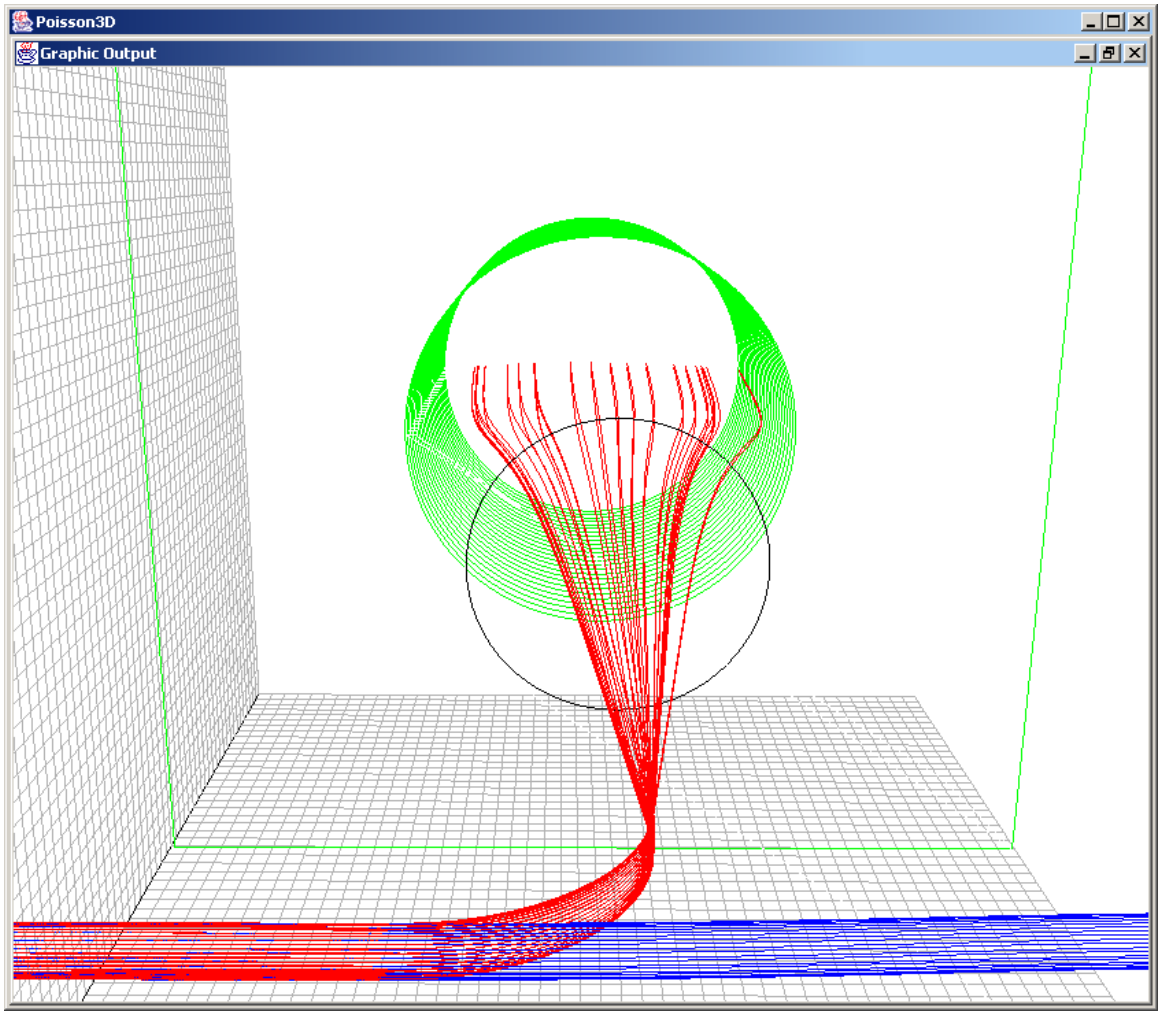

Abbildung 6-9: Ansicht in Richtung y-Achse.
Gut zu erkennen ist die Auffächerung des Elektronenstrahls in x-Richtung (links nach rechts) bei Eintritt in den Kollektor. In z-Richtung erfährt der Elektronenstrahl keine Ablenkung.

7. Zusammenfassung

Im Rahmen dieser Diplomarbeit wurde eine Methode entwickelt,im dreidimensionalen Raum den Strahlverlauf geladener Teilchen durch ein stationäres elektromagnetisches Feld unter Berücksichtigung der elektrischen Eigenfelder zu bestimmen.

Weiterhin wurde gezeigt, wie sich die Genauigkeit der Bahnintegration durch Einführung verbesserter Feldinterpolationsmethoden und verschiedener Integrationsmethoden erhöhen läßt. Dabei zeigte sich, dass die Wahl einer geeigneten Methode zur Feldinterpolation wichtiger ist als die zur Bahnintegration.

Die entwickelten Methoden wurden in einer (in der Sprache Java programmierten) Applikation umgesetzt und verwendet, um die Separation von Elektronen aus einem H^--Ionenstrahl dreidimensional zu simulieren.

8. Danksagung

Ich danke herzlich Herrn Prof. Dr. Ratzinger für die Aufnahme in das Institut und Herrn Prof. Dr. Becker für die Aufnahme in die Arbeitsgruppe und für seine ständige Diskussionsbereitschaft und Hilfe. Meiner Arbeitsgruppe danke ich für ihr Entgegenkommen, Verständnis und Toleranz.

Herrn Christoph Gabor und Herrn Holger Höltermann danke ich für die vielen wertvollen Anregungen und Hinweise bei Erstellung dieser Arbeit.

Herrn Eiko Ermold und Herrn Helmut Dietrich schließlich danke ich für das Gegenlesen in letzter Minute.

Außerdem möchte ich mich noch bei allen nicht namentlich genannten Mitarbeitern des Instituts bedanken, die mich durch ihr freundliches Entgegenkommen und ihrer Hilfe bei dieser Arbeit unterstützt haben.

9. Anhang

9.1. Beispieldateien

9.1.1. Eingabedatei “Zylinder.txt“

```
TITLE Condensator (No fields)

// Dimensionierung des Problems
SPACE
      SIZEX 10  // 10 Meter breit
      SIZEY 10
      SIZEZ 10
      MESHESX 50 // 50 Maschen in X-Richtung
      MESHESY 50
      MESHESZ 50
      SHOW GRID
      SHOW AXIS

// Definition für Geometrie
GEOMETRY
      // Verwendete Einheiten in der Datei
      UNITXY 1 M // setze Einheit in XY-Richtung auf 1 Meter
      UNITZ 1  M // setze Einheit in Z-Richtung auf 1 Meter
      MAXXY 10 // Ausdehnung in X- und Y-Richtung
      MAXZ 10 // Ausdehnung in Z-Richtung
      POTN 2 // zwei verschiedene Potenzialwerte
      POT 0 3 // Potenziale in Volt auf POT1 und POT2
      FORMAT X Y R // Format in SECTION
      SECTION 0 10 // fur Z=0..10 gilt:
            0 5 3
            1 5 4
            1 5 5 1
            1 5 4
            2 5 1
            2 5 5 -4
            2 5 1
            0 5 3
      SHOW
      COLORS RED GREEN

// Definition für Potenzialfeld
POTENTIAL_FIELD
      LAPLACE 1E-12 // Abbruchbedingung fuer Feldabgleichung nach
Laplace
      BRINKS NEUMANN NEUMANN NEUMANN NEUMANN NEUMANN NEUMANN //
Raender x x y y z z
```

```
// Definition für E-Feld
ELECTRIC_FIELD

// Definition für Äquipotenziallinien
EQUIPOT 1 2

// Definition B-Feld
MAGNETIC_FIELD HOMOGEN
      B_X 0
      B_Y 0
      B_Z 0

// Definition für Beamlet
BEAMLET
      PARTICLES 1 PROTONS
      POSITION 8 5 5 // Anfangsposition (3m rechts vom Mittelpunkt)
      SPREAD 0.5 0 0.5 RADIUS 0.5 // Verteilung bei mehr als 1
partikel
      VELOCITY 0 14397.568219700852 0 // Anfangsgeschwindigkeit für
Proton, Abstand beliebig
//    ENERGY 0 1e9 0 eV // kann alternativ zur Geschw. verw. werden)
      INTEGRATOR DP8 // Integrationsmethode Euler, RK4, DP5, DP8, GBS
      STEPSIZE 0.25 // Integrationsschrittweite relativ zur
Maschengroesse
      MAXSTEPS 380 // 18,86m Umfang => ca. 95 Maschen => 380 Schritte
      CURRENT 0 // fuer Raumladung
      POINTS 380 // Anzahl Punkte aus denen eine Trajektorie bestehen
soll
      SHOW BEAM // Grafik einschalten
SHOW BOUNDARIES NO
```

9.1.2. Konsolenausgabe bei Verarbeitung von „Zylinder.txt“

```
D:\Jan's Documents\Physik\java\nigun3d>run Zylinder
D:\Jan's Documents\Physik\java\nigun3d>java -Xms64M -Xmx128M -
Djava.util.logging.config.file=output.cfg de.iap.particleoptics.Nigun3d
Zylinder.TXT
SPACE : Dimension 10m x 10m x 10m with 50 x 50 x 50 meshes at 0.2m x 0.2m x
0.2m
GEOMETRY : Dimension 10m x 10m x 10m with Potentials of 0V, 3V
POTENTIAL FIELD : de.iap.particleoptics.Potential@b5f53a
MAGNETIC FIELD : homogen stationary B-field with Bx=0.0T , By=0.0T , Bz=0.0T
BEAMLET : Beam with 1 Protons at v=14397.568219700852 m/s (-
1.0820212806667222eV )
Initial Laplace solving (without space charge)
Ratio of Z meshes to XY meshes: 1.0.
Problem size 51x51x51=132651 points ; Break at Delta < 1.0E-12
Cycle #     1 , OMEGA = 1.635 , DMAX =   0.0463503861385
Cycle #     2 , OMEGA = 1.829 , DMAX =   0.0215557173384
Cycle #     3 , OMEGA = 1.863 , DMAX =   0.0278238926520
Cycle #     4 , OMEGA = 1.879 , DMAX =   0.0128666406649
Cycle #     5 , OMEGA = 1.879 , DMAX =   0.0029366361267
Cycle #     6 , OMEGA = 1.879 , DMAX =   0.0002990032120
Cycle #     7 , OMEGA = 1.879 , DMAX =   0.0000499861364
Cycle #     8 , OMEGA = 1.879 , DMAX =   0.0000135572569
Cycle #     9 , OMEGA = 1.879 , DMAX =   0.0000016372970
Cycle #    10 , OMEGA = 1.879 , DMAX =   0.0000008240761
Cycle #    11 , OMEGA = 1.879 , DMAX =   0.0000001447893
Cycle #    12 , OMEGA = 1.879 , DMAX =   0.0000000239371
Cycle #    13 , OMEGA = 1.879 , DMAX =   0.0000000026012
Cycle #    14 , OMEGA = 1.879 , DMAX =   0.0000000005084
Cycle #    15 , OMEGA = 1.879 , DMAX =   0.0000000000628
Cycle #    16 , OMEGA = 1.879 , DMAX =   0.0000000000093
Cycle #    17 , OMEGA = 1.879 , DMAX =   0.0000000000023
Converging ( DMAX = 6.914468997365475E-13 < 1.0E-12 ) after 260 iterations
Computing trajectory run, cycle #1
Rendering of 3D graphics
System up&running
total computing time: 85513 ms
D:\Jan's Documents\Physik\java\nigun3d>
```

9.1.3. Eingabedatei „Extraktion.txt“

```
TITLE H-Extraktion

SPACE // Dimensionierung des Problems
      SIZEX 0.10  // 0.1 m lang
      SIZEY 0.10  // 0.1 m breit
      SIZEZ 0.20  // 0.2 m hoch
      MESHESX 50 // 50 Maschen in X-Richtung
      MESHESY 50 // 50 Maschen in Y-Richtung
      MESHESZ 100 // 100 Maschen in Z-Richtung
      SHOW GRID
      SHOW AXIS

GEOMETRY // Definition für Geometrie
      UNITXY 1 CM // verwendete Einheiten: cm
      UNITZ 1 CM
      MAXXY 10 // Ausdehnung in X- und Y-Richtung: je 10cm
      MAXZ 20 // Ausdehnung in Z-Richtung: 20cm
      POTN 2 // zwei verschiedene Potenzialwerte
      POT 0 -115000 // Pot1=0V, Pot2=-115kV (Elektronenkollektor)
      COLORS BLACK GREEN // Pot1 ist schwarz, Pot2 grün
      FORMAT X Y R // Format in SECTION
      SECTION 9.9 10 // es folgt die Definition der Geometrie
            1 0.1 0.1
            1 9.9 0.1
            1 9.9 9.9
            1 0.1 9.9
            0 5 3
            1 5 3.9
            1 5 5 -1.1
            1 5 3.9
            0 5 3
            1 0.1 9.9
            1 0.1 0.1
      SECTION 14.9 15
            2 0.1 0.1
            2 9.9 0.1
            2 9.9 9.9
            2 0.1 9.9
            0 5 3
            2 5 2.9
            2 5 5 -2.1
            2 5 2.9
            0 1 9
            2 0.1 9.9
            2 0.1 0.1
      SECTION 15 19.9
            0 5 3
            2 5 2.9
            2 5 5 -2.1
            2 5 2.9
```

```
            0 5 3
      SECTION 19.9 20
            0 5 3
            2 5 2.9
            2 5 5 2.1
            2 5 2.9
            0 5 3
      SECTION 0.0 0.1 // für Z=0.0 - 0.1 gilt:
            1 0.1 0.1
            1 9.9 0.1
            1 9.9 9.9
            1 0.1 9.9
            1 0.1 0.1
      SHOW YES // Grafische Anzeige der Geometrie erwünscht

POTENTIAL_FIELD // Definition für Potenzialfeld
      LAPLACE 3E-10 // Abbruchbed.für Feldabgleichung nach Laplace
      BRINKS NEUMANN NEUMANN 0 0 0 NEUMANN // Raender x x y y z z

MAGNETIC_FIELD FOR H- // Definition B-Feld
      B_X_MAX 0.14 // max Bx = 0.14 T
      B_Y_MAX 0
      B_Z_MAX 0

BEAMLET ELECTRONBEAM // Definition für Beamlet
      PARTICLES 40 ELECTRONS
      POSITION 5 0 5 // Anfangsposition (in cm)
      SPREAD 0.1 0 0.1 // Strahldurchmesser 0.2cm
      ENERGY 0 -150e3 0 eV // Anfangsenergie (x z y Komponenten)
      INTEGRATOR DP8 // Integrationsmethode Dormand-Prince8
      STEPSIZE 0.05 // Integrationsschrittweite relativ zur
Maschengroesse
      MAXSTEPS 3000 // Anzahl Schritte mit Schrittweite STEPSIZE
      CURRENT 0.1 // Strahlstärke I [A] (fuer Raumladung)
      POINTS 300 // Anzahl Punkte für grafische Bahnausgabe
      SHOW BEAM // Grafik einschalten
      COLOR RED // Trajektorie in rot

BEAMLET H-BEAM
      PARTICLES 20 PARTICLES
      CHARGE -1 e // Ladung: 1 negative Elementarladung
      MASS 1.002 u // Ruhemasse: 1 Proton + 2 Elektronen
      POSITION 5 0 5
      SPREAD 0.1 0 0.1
      ENERGY 0 -150e3 0 eV
      INTEGRATOR DP8
      STEPSIZE 0.05
      MAXSTEPS 1100
      CURRENT 0.1
      POINTS 300
      SHOW BEAM
      COLOR BLUE

SHOW BOUNDARIES NO // Grafikausgabe der Randwertpunkte nicht erwünscht
```

9.1.4. Konsolenausgabe bei Verarbeitung von „Extraktion.txt“

```
D:\Jan\Eigene Dateien\Physik\java\nigun3d>run Extraktion

D:\Jan\Eigene Dateien\Physik\java\nigun3d>java -Xms64M -Xmx128M -
Djava.util.logging.config.file=output.cfg de.iap.particleoptics.Nigun3d
Extraktion.TXT
SPACE : Dimension 0.1m x 0.1m x 0.2m with 50 x 50 x 100 meshes at 0.0020m x
0.0020m x 0.0020m
GEOMETRY : Dimension 0.1m x 0.1m x 0.2m with Potentials of 0V, -115000V
POTENTIAL FIELD : de.iap.particleoptics.Potential@199f91c
MAGNETIC FIELD : stationary B-field designed for H- with maximal values
Bx=0.14T , By=0.0T , Bz=0.0T
BEAMLET : ELECTRONBEAM with 40 Electrons at v=1.901642814745074E8 m/s (-
149999.9999999999eV )
BEAMLET : H-BEAM with 20 Particles at v=5354649.5063069025 m/s (-
149999.9999999103eV )
Initial Laplace solving (without space charge)
Ratio of Z meshes to XY meshes: 1.0.
Problem size 51x51x101=262701 points ; Break at Delta < 3.0E-10
Cycle #     1 , OMEGA = 1.676 , DMAX =6141.2103687024610
Cycle #     2 , OMEGA = 1.790 , DMAX =1930.9358327415578
Cycle #     3 , OMEGA = 1.856 , DMAX =1700.8529914913998
Cycle #     4 , OMEGA = 1.881 , DMAX =1087.4335106072395
Cycle #     5 , OMEGA = 1.881 , DMAX = 436.8598211691206
Cycle #     6 , OMEGA = 1.881 , DMAX = 270.2955698413789
Cycle #     7 , OMEGA = 1.881 , DMAX = 157.7491581510694
Cycle #     8 , OMEGA = 1.881 , DMAX =  87.0399884015951
Cycle #     9 , OMEGA = 1.881 , DMAX =  45.8825122722387
Cycle #    10 , OMEGA = 1.881 , DMAX =  22.9142694097706
Cycle #    11 , OMEGA = 1.881 , DMAX =  10.9594914069140
Cycle #    12 , OMEGA = 1.881 , DMAX =   5.1693160297555
Cycle #    13 , OMEGA = 1.881 , DMAX =   2.4262500041068
Cycle #    14 , OMEGA = 1.881 , DMAX =   1.1339699763940
Cycle #    15 , OMEGA = 1.881 , DMAX =   0.5291235483180
Cycle #    16 , OMEGA = 1.881 , DMAX =   0.2467381558068
Cycle #    17 , OMEGA = 1.881 , DMAX =   0.1150220526412
Cycle #    18 , OMEGA = 1.881 , DMAX =   0.0536119133873
Cycle #    19 , OMEGA = 1.881 , DMAX =   0.0249869997504
Cycle #    20 , OMEGA = 1.881 , DMAX =   0.0116454194145
Cycle #    21 , OMEGA = 1.881 , DMAX =   0.0054273854017
Cycle #    22 , OMEGA = 1.881 , DMAX =   0.0025294362654
Cycle #    23 , OMEGA = 1.881 , DMAX =   0.0011788423590
Cycle #    24 , OMEGA = 1.881 , DMAX =   0.0005493982020
Cycle #    25 , OMEGA = 1.881 , DMAX =   0.0002560463072
Cycle #    26 , OMEGA = 1.881 , DMAX =   0.0001193300168
Cycle #    27 , OMEGA = 1.881 , DMAX =   0.0000556135788
Cycle #    28 , OMEGA = 1.881 , DMAX =   0.0000259186254
Cycle #    29 , OMEGA = 1.881 , DMAX =   0.0000120793365
Cycle #    30 , OMEGA = 1.881 , DMAX =   0.0000056295566
Cycle #    31 , OMEGA = 1.881 , DMAX =   0.0000026236464
Cycle #    32 , OMEGA = 1.881 , DMAX =   0.0000012227464
Cycle #    33 , OMEGA = 1.881 , DMAX =   0.0000005698591
Cycle #    34 , OMEGA = 1.881 , DMAX =   0.0000002655820
Cycle #    35 , OMEGA = 1.881 , DMAX =   0.0000001237741
Cycle #    36 , OMEGA = 1.881 , DMAX =   0.0000000576847
```

```
Cycle #    37 , OMEGA = 1.881 , DMAX =   0.0000000268839
Cycle #    38 , OMEGA = 1.881 , DMAX =   0.0000000125292
Cycle #    39 , OMEGA = 1.881 , DMAX =   0.0000000058392
Cycle #    40 , OMEGA = 1.881 , DMAX =   0.0000000027213
Cycle #    41 , OMEGA = 1.881 , DMAX =   0.0000000012683
Cycle #    42 , OMEGA = 1.881 , DMAX =   0.0000000005911
Converging ( DMAX = 2.8983038191654487E-10 < 3.0E-10 ) after 644 iterations
Computing trajectory run, cycle #1
Rendering of 3D graphics
System up&running
total computing time: 297708 ms
D:\Jan\Eigene Dateien\Physik\java\nigun3d>
```

9.2. Klassenübersicht

amperedatfile.class	AnalyticalEField.class	AnalyticElectricField.class
AnalyticPotential.class	Beamlet.class	Boundarypoints3D.class
Console.class	ElectricField.class	ElectricFieldOfPotential.class
Electron.class	Equipotential.class	Geometry3D.class
GeometryData.class	GraphicOutput.class	HomogenMagneticField.class
InputData.class	InputPoint.class	InputSegment.class
MagneticField.class	MagneticFieldForHminus.class	MagneticFieldFromAmpere.class
Particle.class	Point.class	Poisson3d.class
Polygon.class	Potential.class	Proton.class
Space.class	SpaceCharges.class	Trajectory3D.class
TrajectoryInEMfield.class		

Tabelle 9-1: Alle Java Klassen im Überblick

9.3. Klassenhierarchie

- class java.lang.**Object**
  - class java.util.**AbstractMap** (implements java.util.Map)
    - class java.util.**HashMap** (implements java.lang.Cloneable, java.util.Map, java.io.Serializable)
      - class de.iap.particleoptics.**SpaceChargeHashMap**
  - class de.iap.particleoptics.**Beamlet** (implements de.iap.NigunConstants, de.iap.particleoptics.renderable)
  - class de.iap.particleoptics.**Boundary3D** (implements de.iap.particleoptics.BoundaryConstants, de.iap.particleoptics.renderable)
  - class de.iap.particleoptics.**Boundarypoints3D** (implements de.iap.particleoptics.BoundaryConstants)
  - class de.jdietrich.**Color2**
  - class java.awt.**Component** (implements java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable)
    - class java.awt.**Container**
      - class javax.swing.**JComponent** (implements java.io.Serializable)
        - class javax.swing.**JPanel** (implements javax.accessibility.Accessible)
          - class de.iap.particleoptics.**Rendering**
  - class de.iap.particleoptics.**Coordinate**
  - class de.iap.particleoptics.ElectricField
    - class de.iap.particleoptics.**ElectricFieldOfPotential**
  - class java.util.logging.**Formatter**
    - class java.util.logging.**SimpleFormatter**
      - class de.jdietrich.**LongFormatter**
      - class de.jdietrich.**MsgOnlyFormatter**

        - class de.jdietrich.**SmallFormatter**
- class de.iap.particleoptics.**Geometry3D** (implements de.iap.particleoptics.renderable)
- class de.iap.particleoptics.**GeometryData**
- class de.iap.particleoptics.**GraphicOutput**
- class java.util.logging.**Handler**
    - class de.jdietrich.**GuiOutHandler**
    - class java.util.logging.**StreamHandler**
        - class de.jdietrich.**ConsoleOutHandler**
- class de.iap.particleoptics.**InputData** (implements de.iap.NigunConstants)
- class de.iap.particleoptics.**Integratortest**
- class de.iap.particleoptics.**IntegratortestAnalyticalFields**
- class de.iap.particleoptics.**IntegratortestNumericFields**
- class de.iap.particleoptics.**MagneticField**
    - class de.iap.particleoptics.**HomogenMagneticField**
    - class de.iap.particleoptics.**MagneticFieldForHminus**
    - class de.iap.particleoptics.**MagneticFieldFromAmpere**
- class de.jdietrich.**Mathe**
- class de.iap.particleoptics.**Nigun3d** (implements de.iap.NigunConstants)
- class de.iap.particleoptics.**Particle** (implements de.iap.NaturalConstants)
    - class de.iap.particleoptics.**Electron** (implements de.iap.NigunConstants)
    - class de.iap.particleoptics.**Proton** (implements de.iap.NigunConstants)
- class de.iap.particleoptics.**Poisson3d**
- class de.iap.particleoptics.**Potential**
- class de.iap.particleoptics.**Space** (implements java.lang.Cloneable, de.iap.particleoptics.renderable)

    - class de.iap.particleoptics.**SpaceCharges** (implements de.iap.NaturalConstants)
    - class de.jdietrich.**StringTool**
    - class de.jdietrich.**Timer2**
    - class de.iap.particleoptics.**Zylinderkondensator** (implements de.iap.particleoptics.AnalyticalEField)

- interface de.iap.particleoptics.**AnalyticalEField**
- interface de.iap.particleoptics.**BoundaryConstants**

9.4. Grafikverzeichnis

9.5. Diagrammverzeichnis

9.6. Literaturverzeichnis

[1] *Becker, R.; Numerical Simulation of ion-beam formation*
Rev. Sci. Instrum. 67(3), March 1996

[2] *Hermannsfeldt, W.B.; SLAC Electron Optics Program*
SLAC-166 (1973)

[3] *Becker,R. and Herrmannsfeldt, W.B.;*
Rev. Sci. Instrum. 63, 2756 (1992)

[4] *Carré, B.A.; The Determination of the Optimum Accelerating Factor for Successive Over-relaxation*
Nelson Research Laboratories, Stafford, UK, April 1961

[5] *Ehrenberg, Uwe; Optimierung von Elektrodenanordungen kapazitiver Sensoren mittels numerischer Feldberechnung*
Technische Universität Ilmenau, Dezember 1998

[6] *Hoffmann, Oliver; Lösung des Poisson-Problems durch Finite-Differenzen-Gleichungssysteme mittels der Mehrgitter-Methode*
Universität Stuttgart, Dezember 1999

[7] *Integrated Engineering Software Inc.*
220-1821 Wellington Avenue, Winnipeg, Manitoba, Canada R3H 0G4
Phone: (204) 632-5636 Fax: (204) 633-7780
E-mail:info@integratedsoft.com

[8] *Jürgens, R.; A Computer Program to solve the Laplace equation in three dimensions using a powerful boundary input method*
Institut für Angewandte Physik, Universität Frankfurt am Main, 1981

[9] *Leitner, M.A. and Leung, K.N.; Optomization of the volume H- ion source extraction system for the Spallation Neutron Source accelerator utilizing 3D magnet and ion optics codes*
Nucl. Inst. and Meth. A 427 (1999) 250-254

[10] *Lorensen, W.E., Cline, H.E.; Marching cubes: A high resolution 3d surface construction algorithm*
Computer Graphics, 21(4):163-169, 1987

[11] *Mayer, Norbert; Verbesserung der Potentialberechnung und Feldinterpolation bei der Diskretisierung durch finite Differenzen*
Institut für Angewandte Physik, Universität Frankfurt am Main, Mai 1996

[12] *Prince, P.J., Dormand, J.R.; High order embedded Runge-Kutta formulae*
J. Comput. Appl. Maths. 17 (1981) 67

[13] *Bulirsch, R., Oetti, W., Stoer, J.; Optimization and Optimal Control*
Lecture Notes in Mathematics 477 Springer Verlag, 1975, pp. 95-101