INSTITUT FÜR PHYSIK

JOHANNES GUTENBERG UNIVERSITÄT

MAINZ

Entwicklung von Methoden zur Untergrundreduzierung am Mainzer Tritium–β–Spektrometer

Diplomarbeit

von

Thomas Thümmler

Mainz, Januar 2002

Johannes Gutenberg (Kupferstich von Theret)

Inhaltsverzeichnis

1 Einführung **1**

2 Das Mainzer Tritium-β-Experiment **7**

2.1 Das Tritium-β-Spektrum 7

2.2 Das Experiment in Mainz 11

2.2.1 Funktionsweise des MAC-E-Filters 11

2.2.1.1 Parameter des Mainzer MAC–E–Filters 15

2.2.2 Adiabatische Näherung 1. Ordnung 17

2.2.3 Magnetisches Moment als adiabatische Invariante 20

2.2.4 Adiabatische Energietransformation in relativistischer Form . 22

2.2.5 Energiekorrektur als Folge der Driftbewegung 23

3 Werkzeuge zur Teilchenbahnverfolgung **25**

3.1 SIMION 3D Version 7.0 25

3.2 Das Simulationsprogramm ADIPARK 27

3.2.1 Motivation für ein neues Simulationsprogramm 27

3.2.2 Modularisierung und Funktionsweise 28

3.2.3 Teilchenbahnverfolgung mit ADIPARK 32

3.2.4 Prinzipielle und numerische Grenzen 34

3.2.4.1 Diskussion der technischen Grenzen 35

3.2.4.2 Diskussion der mathematischen Grenzen 35

3.2.4.3 Tests der numerischen Genauigkeit 40

3.3 Vergleich von ADIPARK mit SIMION 3D 44

4 Gespeicherte Teilchen im MAC–E–Filter 49

4.1 Speicherbedingungen für geladene Teilchen 49

4.2 Speicherung von Elektronen 53

4.3 Speicherung von Ionen 59

4.4 Speichervolumenkarten des Mainzer Spektrometers 61

4.5 Untergrundprozesse 67

5 Aufheben der Speicherbedingungen 71

5.1 Diskussion einer elektrischen Dipolelektrode 71

5.1.1 Motivation einer elektrischen Dipolelektrode 71

5.1.2 Wirkung eines idealisierten elektrischen Dipolfeldes 72

5.1.3 Stabilitätskriterien gespeicherter Teilchen 73

5.2 Simulationen mit Dipolelektrode 78

5.2.1 Simulation einer realen Dipolelektrode 78

5.2.2 Speichervolumenkarten mit Dipolelektrode 80

5.3 Betrachtung eines gekrümmten Magnetfeldes 84

5.3.1 Motivation für ein gekrümmtes Magnetfeld 84

5.3.2 Simulation mit gekrümmten Magnetfeldlinien 84

6 Zusammenfassung und Ausblick 89

A Quelltexte 93

A.1 ADIPARK Hauptprogramm: adipark.c 93

A.2 Bahnverfolgung: tracking.c 95

A.3 Modul für Speichervolumenkarten: trapping.c 110

A.4 Elektrische Feldkartenverwaltung: efield.c 112

A.5 Magnetische Feldkartenverwaltung: mag_pa_reader.c 119

A.6 Mathematische Bibliothek: math_vector.c 123

B Parameterdateien 131

B.1 Globale Konstanten: global.h 131

B.2 Simulationparameter: .run Datei 132

B.3 Magnetfeldparameter: .par Datei 133

Abbildungsverzeichnis

2.1 Signatur einer endlichen Neutrinomasse 9
2.2 MAC-E-Filter 12
2.3 Transmissionsfunktion des MAC-E-Filters 14
2.4 Elektrodenkonfiguration des Mainzer MAC–E–Filters 15
2.5 Spulenkonfiguration des Mainzer MAC–E–Filters 16
2.6 Nicht adiabatischer Energiegewinn eines Elektrons 24

3.1 Relaxation in SIMION 3D 26
3.2 Modulare Struktur von ADIPARK 29
3.3 Struktogramm des Teilchenverfolgungsalgorithmus 30
3.4 Adiabasieabweichung in SIMION 3D 37
3.5 Flugbahnabweichung durch adiabatische Näherung 37
3.6 Vergleich von Flugbahn- und Adiabasieabweichung 38
3.7 Schrittweitenabhängigkeit des Reflektionspunktes 41
3.8 Schrittweitenabhängigkeit der Berechnungszeit 42
3.9 Energieabhängigkeit des Reflektionspunktes 43
3.10 Vergleich einer Elektronentrajektorie in x-y-Projektion 45
3.11 Vergleich einer Elektronentrajektorie in x-z-Projektion 46
3.12 Vergleich der Abbildungseigenschaften 47
3.13 Energieerhaltung eines gespeicherten Elektrons 48

4.1 Teilchenfalle im Elektrodenzwischenraum 50
4.2 Potentialmaximum im Elektrodenzwischenraum 50
4.3 Teilchenfalle mit magnetischen Reflektionspunkten 52
4.4 Flugbahn eines gespeicherten Elektrons in der Spektrometermitte . . 55

4.5 Flugbahn eines gespeicherten Elektrons im gesamten Volumen 56
4.6 Flugbahn eines gespeicherten Elektrons im linken Halbraum 57
4.7 Flugbahn eines gespeicherten Elektrons in einer Penningfalle 58
4.8 Flugbahn eines gespeicherten Protons 60
4.9 Speichervolumenkarte für Elektronen mit $E_{kin} = 512\ eV$ 64
4.10 Speichervolumenkarte für Elektronen mit $E_{kin} = 256\ eV$ 64
4.11 Speichervolumenkarte für Elektronen mit $E_{kin} = 128\ eV$ 65
4.12 Speichervolumenkarte für Elektronen mit $E_{kin} = 32\ eV$ 65
4.13 Speichervolumenkarte für Elektronen mit $E_{kin} = 16\ eV$ 66
4.14 Speichervolumenkarte für Elektronen mit $E_{kin} = 8\ eV$ 66
4.15 Speichervolumenkarte der kleinen Penningfallen 67

5.1 Wirkung eines elektrischen Dipolfeldes 72
5.2 Gespeichertes Teilchen trotz elektrischem Dipolfeld 74
5.3 Radiale Abhängigkeit der Driftgeschwindigkeit 74
5.4 Teilchenbahnen von ADIPARK mit variiertem Dipolfeld 75
5.5 Radiale Abhängigkeit der Driftgeschwindigkeit 75
5.6 Dipolhalbschale im Mainzer Spektrometer 78
5.7 ADIPARK-Simulation der Wirkung verschiedener Dipolfelder 79
5.8 Speichervolumenkarte für Elektronen im Dipolfeld mit $E_{kin} = 512\ eV$ 81
5.9 Speichervolumenkarte für Elektronen im Dipolfeld mit $E_{kin} = 256\ eV$ 82
5.10 Speichervolumenkarte für Elektronen im Dipolfeld mit $E_{kin} = 128\ eV$ 82
5.11 Speichervolumenkarte für Elektronen im Dipolfeld mit $E_{kin} = 32\ eV$ 83
5.12 Speichervolumenkarte für Elektronen im Dipolfeld mit $E_{kin} = 16\ eV$ 83
5.13 Feldlinien des gekrümmten Magnetfeldes 85
5.14 Vergleich von gekrümmtem Magnetfeld und Dipolfeld 86

Tabellenverzeichnis

2.1 Parameter der Elektrodenkonfiguration des Mainzer MAC–E–Filters 17

3.1 Aufbau einer '.run'-Datei 32

3.2 Aufbau einer '.track<X>'-Datei 33

3.3 Eingabeparameter für numerische Genauigkeit 44

Kapitel 1

Einführung

Im Jahre 1914 beobachtete Chadwick das Energiespektrum des β-Zerfalls. Er ging von einem Zwei-Körper-Zerfall aus, d.h. er erwartete eine monoenergetische Linie im Energiespektrum. Zur Überraschung der Physiker ergab sich aber ein kontinuierliches Energiespektrum. Der β-Zerfall stand im Widerspruch zu den Erhaltungssätzen für Energie, Drehimpuls und der quantenmechanischen Spinstatistik.
W. Pauli [Pau30] postulierte 1930 in einem Brief an H. Geiger und L. Meitner ein hypothetisches neutrales Teilchen mit Spin $\frac{1}{2}$, das er vorläufig Neutron nannte. Damit ließ sich der β-Zerfall als Drei-Körper-Zerfall beschreiben und die Widersprüche waren entkräftet. Nachdem Chadwick 1932 das neutrale Nukleon, das Neutron, entdeckt hatte, war durch die sehr viel höhere Masse klar, daß es sich nicht um das von Pauli postulierte Teilchen handeln konnte.
Zwei Jahre nach der Entdeckung des Neutrons wurde der Name 'Neutrino' gebräuchlich, den E. Fermi [Fer34] einführte, als ihm 1934 die theoretische Beschreibung des β-Zerfalls gelang. Er betrachtete das Neutrino als masseloses Teilchen.

Es dauerte weitere 22 Jahre, bis 1956 E. Reines (Nobelpreis 1995) und C. L. Cowan das Neutrino experimentell nachweisen konnten. Mit den jetzt vorhandenen Kernreaktoren standen sehr starke Antineutrinoquellen zur Verfügung, die es ermöglichten, den inversen β-Zerfall

$$\overline{\nu}_{\mathrm{e}} + \mathrm{p} \longrightarrow \mathrm{n} + \mathrm{e}^{+} \tag{1.1}$$

in einer wässrigen Kadmiumchlorid-Lösung zu beobachten. Hierbei ergab sich ein mittlerer Wirkungsquerschnitt von $\sigma = (1.1 \pm 0.3) \cdot 10^{-43}\mathrm{cm}^2$. Dieses Ergebnis bestätigt die um V-A erweiterte Fermitheorie für Neutrinoenergien von $\mathrm{E}_\nu \leq 8$ MeV [Rei59]. Die Entdeckung des Myonneutrinos 1962 durch Ledermann, Steinberger und Schwartz wurde 1988 mit dem Nobelpreis gewürdigt. Das Tauneutrino wurde schließlich im Jahr 2000 durch das DONUT-Experiment am Fermilab nachgewiesen [Don00].

Laut Standardmodell der Teilchenphysik wird die gesamte Materie aus zwölf Fermionen aufgebaut. Neutrinos $(\nu_e, \nu_\mu, \nu_\tau)$ sind die schwachen Isospinpartner der Leptonen: Elektron, Myon und Tau. Sie ergeben zusammen mit den ebenfalls in Dubletts auftretenden sechs Quarks ((u,d) , (c,s) , (b,t)) die zwölf Fermionen. Die Quark- und

Leptondubletts ergeben, wenn man sie nach ansteigender Masse ordnet, die drei Fermionfamilien. Das Standardmodell enthält jedoch keine Information über die Massen der Fermionen, daher müssen diese durch Experimente bestimmt und dem Modell hinzugefügt werden. Erst kürzlich wurden experimentell Beweise für eine endliche Neutrinomasse gefunden; das Standardmodell geht noch von masselosen Neutrinos aus. In Theorien, die über das Standardmodell hinausgehen (sog. neue Physik, z.B. Supersymmetrie, Grand Unified Theorie(GUT)), werden Neutrinos im allgemeinen als massiv betrachtet.

Die Kenntnis der Neutrinoeigenschaften hat Auswirkungen auf Astrophysik, Kosmologie und Teilchenphysik. Da Neutrinomassen sehr viel kleiner sind, als die der anderen Fermionen, würde die Kenntnis einer absoluten Massenskala für Neutrinos das Verständnis aller Fermionmassen stark beeinflussen. Neutrinos spielen eine entscheidende Rolle in der Urknalltheorie, dem frühen Universum und seiner Strukturbildung. Nach Standard-Urknall-Modell entkoppelten die Neutrinos und bilden heute, analog zur Mikrowellenhintergrundstrahlung, den Neutrinohintergrund. Zur Zeit der Entkopplung der Neutrinos war das Baryonen zu Neutrino Verhältnis $1 : 10^9$, d.h. es existieren sehr viel mehr sog. Relic Neutrinos, die zur Dunklen Materie beitragen können, wenn sie denn eine endliche Masse besitzen. Dies würde sich auf das Verständnis der Gesamtstruktur des Universums und der Dynamik von Galaxien und Galaxienhaufen auswirken. In den Modellen mit Dunkler Materie wird zwischen Kalter Dunkler Materie und Heißer Dunkler Materie unterschieden, für letztere sind Neutrinos die einzigen bekannten Kandidaten.

Eine indirekte Methode zur Suche nach der Neutrinomasse sind Oszillationsexperimente. Bei Neutrinooszillationen nimmt man an, daß Übergänge zwischen den einzelnen Neutrinosorten stattfinden. Man geht davon aus, daß die Masseneigenzustände (ν_1, ν_2, ν_3) nicht identisch mit den Flavoureigenzuständen $(\nu_e, \nu_\mu, \nu_\tau)$ sind, es also Übergänge zwischen Neutrinoflavours gibt. Vorraussetzung dafür ist aber, daß mindestens ein Masseneigenzustand ungleich Null ist, d.h. Neutrinos besitzen endliche Massen. Oszillationsexperimente können jedoch nur Massenquadratdifferenzen und Mischungswinkel messen und sind daher nicht in der Lage eine absolute Massenskala aufzustellen. Sonnenneutrinoexperimente (Gallex, Sage,...) messen einen niedrigeren Fluß von Sonnenelektronneutrinos als erwartet, das sog. solare Neutrinoproblem. Man geht davon aus, daß dies eine Konsequenz der Oszillation ist.

Das Super–Kamiokande–Experiment ist ein Oszillationsexperiment und hat 1998 erste Evidenzen für Neutrinooszillationen gesehen [Fuk98]. In diesem Experiment werden atmosphärische und solare Neutrinos, die in einen unterirdischen Wassertank fliegen und dort wechselwirken, durch die Čerenkow-Strahlung ihrer Sekundärteilchen nachgewiesen. Die veröffentlichten Daten zeigen Massenquadratdifferenzen im Bereich von $3 \cdot 10^{-3}\ eV^2/c^4$ für den Übergang von atmosphärischen $\nu_\mu \rightarrow \nu_\tau$.

Eine weitere Evidenz für Neutrinooszillation zeigt LSND. Dieses Beschleunigerexperiment hat Massenquadratdifferenzen von $1\ eV^2/c^4$ für den Übergang $\bar{\nu}_\mu \rightarrow \bar{\nu}_e$

gefunden [Ath95]. Die Evidenz wurde nicht durch das KARMEN-Experiment bestätigt, sondern es zeigten sich zum Teil sogar Widersprüche [Eit00].

Im Juni 2001 wurden vom Sudbury Neutrino Observatory (SNO) weitere Evidenzen für eine endliche Neutrinomasse gefunden. Auch bei diesem Experiment handelt es sich um ein Oszillationsexperiment, es wird ein Wasser Čerenkov Detektor mit 1000 Tonnen ultrareinem schweren Wasser D_2O in 6000 Meter Wasseräquivalent Tiefe verwendet. Das entstehende Čerenkovlicht wird durch 9456 Photomultiplier registriert. Die im Juni 2001 veröffentlichten Ergebnisse zeigen eine Obergrenze für die Massenquadratdifferenz der Neutrinos von $\Delta m^2 \leq 10^{-3}\ eV^2$. Es wurde der ν_e–Fluß über die Reaktion $\nu_e + d \rightarrow e^- + p + p$ gemessen. Super–Kamiokande mißt den ν_x–Fluß über die Reaktion $\nu_x + e^- \rightarrow \nu_x + e^-$. Es ergab sich $\Phi_{SK} > \Phi_{SNO}$, womit gezeigt wurde, daß das solare Neutrinoproblem eine Konsequenz der Neutrinooszillation ist [SNO01].

Eine Methode zur Neutrinomassenbestimmung stellen die Experimente zum neutrinolosen doppelten β–Zerfall dar, hierbei finden zwei β-Zerfälle simultan statt und es werden zwei ν_e oder $\bar{\nu}_e$ emittiert. Der doppelte β–Zerfall kann bei Kernen auftreten, für die der einfache β–Zerfall verboten ist. Der Nachweis gelang erstmals 1987 [Ell87]. Ein Spezialfall ist der neutrinolose doppelte β–Zerfall, bei dem die Neutrinos nicht emittiert, sondern als virtuelle Teilchen zwischen den beiden Zerfallsvertices ausgetauscht werden. In diesem Fall müssen die Neutrinos Majorana-Teilchen sein, d.h. Teilchen und Antiteilchen sind identisch. Damit das Neutrino als linkshändiges Teilchen emittiert werden kann und am zweiten Vertex als rechtshändiges absorbiert wird, ist ein Zustand gemischter Helizität notwendig, den es wiederum nur geben kann, wenn das Neutrino eine Masse hat. Die Experimente sind hierbei sensitiv auf die effektive Neutrinomasse $m_{ee}(\nu)$. Sie ist die kohärente Summe aller Neutrinomasseneigenzustände ν_i, die aufgrund der Mischungsmatrix U_{ei} zum Elektronneutrino ν_e beitragen.

$$m_{\mathrm{ee}} = |\sum_{i=1}^{3} U_{ei}^2 \cdot m(\nu_i)| \tag{1.2}$$

Die Matrixelemente sind im allgemeinen komplex, d.h. es kann zu Auslöschungen kommen und $m_{ee}(\nu)$ kann verschwinden, obwohl die Masseneigenzustände $m(\nu_i)$ endliche Massen tragen.

Ein Experiment zur Suche nach dem neutrinolosen doppelten β–Zerfall ist das Heidelberg–Moskau–Experiment an ^{76}Ge [Die00]. Das, obwohl es das zur Zeit sensitivste ist, noch keinen neutrinolosen doppelten β–Zerfall beobachtet hat. Aus dieser Tatsache läßt sich eine Massenobergrenze für Neutrinos als Majoranateilchen von $m_{ee}c^2 \leq 0.34\ eV$ (90% CL) ableiten.

Eine andere Möglichkeit der Neutrinomassenbestimmung bieten Experimente zur direkten Massenbestimmung. Hier sind keine weiteren Modellannahmen nötig, nur die Gültigkeit der relativistischen Energie-Impuls-Beziehung wird vorrausgesetzt. Ergebnis dieser Untersuchungen ist das Quadrat der Neutrinomasse.

Eine direkte Methode stellt die Flugzeitmessung dar. Hier wird die Flugzeit der Neutrinos gemessen, die sie zum Zurücklegen einer definierten Strecke benötigen. Da sehr lange Flugstrecken und starke Neutrinoquellen nötig sind, kann diese Methode nicht auf der Erde angewendet werden. Erst die Supernova SN1987A, eine Supernova vom Typ II, die Neutrinos abstrahlte, ermöglichte die Anwendung dieser Methode. Aus dem Nachweis von insgesamt 19 Elektronneutrinos konnte auf eine obere Massengrenze von $m_{\nu_e} \leq 23\ eV/c^2$ geschlossen werden [Par98].

Die bisher erfolgreichste direkte Massenbestimmung findet über die Untersuchung der Kinematik schwacher Zerfälle statt. Bevorzugt wird hier der β-Zerfall von Tritium, mit einer Endpunktsenergie des β–Spektrums von $18,6\ keV$. Untersuchungen des β-Spektrums im Endpunktsbereich bieten hier die Möglichkeit auf die Neutrinomasse zu schließen. Die Meßgröße solcher Experimente ist wieder das Massenquadrat.

$$m^2(\nu_e) = \sum_{i=1}^{3} |U_{ei}|^2 \cdot m^2(\nu_i) \tag{1.3}$$

Ein wichtiger Unterschied ist, daß in diesen Experimenten die Betragsquadrate der Mischungsmatrix auftreten und keine Auslöschungen mehr möglich sind, d.h. die Neutrinomasse kann bestimmt werden. In Mainz und Troitsk werden Experimente nach dieser Methode durchgeführt. Beide Experimente haben Neutrinomassenobergrenzen veröffentlicht, die bei

- $m_{\nu_e} \leq 2.2\ eV/c^2$ in Mainz (95% CL) [Bon01]
- $m_{\nu_e} \leq 2.5\ eV/c^2$ in Troitsk (95% CL) [Lob99]

liegen. Die Experimente in Mainz und Troitsk haben mit diesen Werten ihr Sensitivitätslimit erreicht. An Verbesserungen des experimentellen Aufbaus wird jedoch weiterhin gearbeitet, um die Untergrundzählrate noch stärker zu reduzieren. Diese Modifikationen sind, wenn sie Erfolge zeigen, sehr wichtig für das in Entwicklung befindliche Tritium–Experiment der nächsten Generation: KATRIN (KArlsruhe TRItium Neutrinoexperiment). Dieses neue Experiment wird am Forschungszentrum Karlsruhe gebaut und wird in einen Neutrinomassenbereich im sub–eV–Bereich vorstoßen können. Damit wird es möglich werden, den kosmologisch relevanten Neutrinomassenbereich und eine absolute Massenskala für Neutrinos bis in den sub–eV–Bereich zu untersuchen.

Ziel dieser Arbeit ist es, Methoden zur Reduzierung von Untergrundereignissen am Mainzer Tritium-β-Spektrometer zu entwickeln und zu testen. Zu diesem Zweck wurden neue Werkzeuge zur Computersimulation in beliebigen dreidimensionalen elektromagnetischen Umgebungen entwickelt und angewendet, um gespeicherte Teilchen zu untersuchen und zu entfernen. Die Arbeit ist wie folgt gegliedert:

- Im zweiten Kapitel wird das Tritium-β-Spektrum als physikalische Grundlage erläutert, bevor dann der experimentelle Aufbau und seine Eigenschaften vorgestellt werden. Daran anschließend wird die adiabatische Bewegung der geladenen Teilchen erläutert.
- Im dritten Kapitel werden die Werkzeuge zur Computersimulation, die Entwicklung des neuen Programms und seine Tests vorgestellt.
- Das vierte Kapitel befaßt sich mit den Eigenschaften von verschiedenen gespeicherten Teilchen und denen im Spektrometer vorhandenen Speicherregionen. Es werden Ursachen und Auswirkungen von Untergrundprozessen besprochen.
- In Kapitel fünf werden dann die verschiedenen Ansätze zur Reduzierung der Anzahl gespeicherter Teilchen vorgestellt, durch Simulationen getestet und miteinander verglichen.
- Abschließend enthält das sechste Kapitel eine Zusammenfassung der erarbeiteten Ergebnisse und einen Ausblick auf noch offene Fragestellungen und Ideen.

Kapitel 2

Das Mainzer Tritium-β-Experiment

In diesem Kapitel wird das Mainzer Tritium-β-Experiment, die physikalische Idee und der experimentelle Aufbau vorgestellt. Es werden die Eigenschaften des Spektrometers und seine prinzipielle Funktionsweise erklärt. Zum Verständnis der Teilchenbewegung im Spektrometer wird die adiabatische Näherung in erster Ordnung betrachtet, wodurch die Teilchenbewegung in verschiedene Bewegungsanteile zerlegt werden kann.

2.1 Das Tritium-β-Spektrum

Wenn sich in einem nackten Atomkern ein Neutron in ein Proton umwandelt, wird dabei ein Elektron und ein Elektronantineutrino abgestrahlt. Dieser Prozeß, in dem sich die Kernladung Z um eins erhöht, die Massenzahl A aber konstant bleibt, nennt man β^--Zerfall.

$$(\mathrm{Z}, \mathrm{A}) \longrightarrow (\mathrm{Z}+1, \mathrm{A}) \; + \; e^- \; + \; \bar{\nu}_\mathrm{e} \tag{2.1}$$

Man spricht hierbei von einem Drei-Körper-Zerfall, die Zerfallsenergie verteilt sich auf das Elektron und das Antineutrino. Der zurückgelassene Kern nimmt nur Rückstoß–Impuls auf, aber keine Energie, da er als unendlich schwer angenommen wird. Man erwartet daher ein kontinuierliches Energiespektrum der emittierten Elektronen. Da die Energie erhalten ist, ist es möglich die Ruhemasse des Neutrinos aus der Form des Energiespektrums der Elektronen abzuleiten.

Die Form des Energiespektrums wird durch Fermis Goldene Regel beschrieben. Unter Verwendung der Phasenraumdichte der möglichen Endzustände von Elektron und Antineutrino und des Quadrats des Kernmatrixelementes $|\mathrm{M}|^2$, liefert sie die theoretische Beschreibung. Hierbei ergibt sich die Anzahl der Elektronen dN, die im Energieintervall dE emittiert werden, zu:

$$\frac{dN}{dE} = \frac{G_f^2 m_e^5 c^4}{2\pi^3 \hbar^7} \cos^2(\Theta_c) \, |M|^2 F(Z,E) \, p \left(E + m_e c^2\right) \left(E_0 - E\right) \cdot \sqrt{(E_0 - E)^2 - m_\nu^2 c^4} \cdot \theta\left(E_0 - E - m_\nu c^2\right) \quad (2.2)$$

mit : G_f: Fermi–Kopplungskonstante
Θ_c: Cabbibo–Winkel
M: Kernmatrixelement
F: Fermi–Funktion
E: kinetische Energie des e^-
p: Impuls des e^-
E_0: β–Endpunkt
m_e, m_ν: Ruhemasse des Elektrons und des Neutrinos.

Bis auf die Abbruchbedingung kommt die Masse des Neutrinos m_ν in dieser Gleichung nur quadratisch vor, d.h. bei Untersuchungen eines β-Spektrums ist die Observable das Neutrinomassenquadrat ${m_\nu}^2 c^4$.

Das Mainzer Experiment verwendet das neutronenreiche Wasserstoffisotop Tritium wegen seiner niedrigen Endpunktsenergie von $E_0 = 18.6\ keV$. Tritium ist β^--instabil und zerfällt durch einen supererlaubten Übergang in seinen Spiegelkern Helium3 ($^3_2\mathrm{He}^+$).

$$^3_1\mathrm{H} \longrightarrow ^3_2\mathrm{He}^+ + \mathrm{e}^- + \bar{\nu}_e \quad (2.3)$$

Die theoretische Beschreibung (Gl. 2.2) geht von einem nackten Atomkern aus, verwendet wird aber eine Quelle aus molekularem Tritium, d.h. das Spektrum muß modifiziert werden, um den Einfluß von Hüllenelektronen zu berücksichtigen. Diese können nach einem Zerfall als angeregte Zustände des Tochteratoms verbleiben.

Um die Besetzungswahrscheinlichkeit dieser Zustände zu berechnen, wird die 'sudden approximation'-Methode angewendet. Hierbei geht man davon aus, daß der Zerfallsprozeß im Kern sehr viel schneller abläuft, als die Prozesse in der Elektronenhülle. Für die Hüllenelektronen ändert sich die Kernladung plötzlich ('sudden'). Die Übergangswahrscheinlichkeit von $\mathrm{i} \to \mathrm{f_n}$ wird durch das Überlappintegral der Wellenfunktion des Anfangszustandes φ_i und der möglichen Endzustände $\varphi_{\mathrm{f_n}}$ angegeben.

$$\mathrm{W}_{\mathrm{i}\to\mathrm{f_n}} = | < \varphi_{\mathrm{f_n}} | \varphi_\mathrm{i} > |^2 \quad (2.4)$$

Hiermit ergibt sich das Energiespektrum der Zerfallselektronen des Tritiummoleküls aus der Summe der mit $\mathrm{W}_{\mathrm{i}\to\mathrm{f_n}}$ gewichteten Einzelspektren der Zerfälle, die um die Anregungsenergien der Elektronenhülle V_n verschoben sind, in die jeweiligen Endzustände. Das modifizierte β-Spektrum ergibt sich dann zu:

$$\frac{dN}{dE} = const \cdot |M|^2 \; F(Z,E) \; p \; (E + m_e c^2) \; \sum_\mathrm{n} W_{\mathrm{i}\to\mathrm{f_n}} \varepsilon_\mathrm{n} \sqrt{\varepsilon_\mathrm{n}^2 - m_\nu^2 c^4} \quad (2.5)$$

mit $\varepsilon_\mathrm{n} = E_0 - V_\mathrm{n} - E$.

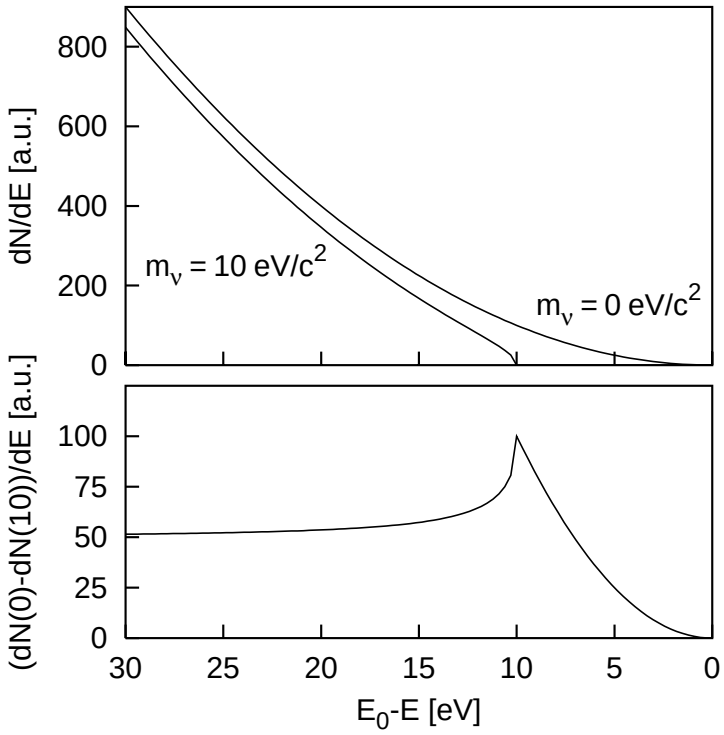

Abbildung 2.1: Signatur einer endlichen Neutrinomasse. Oben dargestellt ist der Endpunktsbereich des Tritium-β–Spektrums für zwei angenommene Neutrinomassen (0 eV und 10 eV) beim Übergang in den Grundzustand des Tochterions. Unten ist die Differenz der beiden Spektren aufgetragen, hierbei sieht man die Signatur einer endlichen ν-Ruhemasse. Sie ist in der Nähe des β–Endpunktes maximal, verschwindet jedoch nirgendwo ganz.

Eine endliche Neutrinomasse führt zu einer Absenkung und einem früheren Abknicken des β–Spektrums. Die Maximalenergie des Elektrons wird mindestens um die Ruhemasse $m_\nu c^2$ des Neutrinos reduziert, das Energiespektrum bricht also vor dem Endpunkt E_0 ab. Diesen Effekt kann man in Abbildung 2.1 beim Vergleich zweier Beispielrechnungen erkennen. Da die Signatur einer endlichen Neutrinomasse nirgendwo ganz verschwindet, könnte man auch einen großen Teil des Spektrums analysieren, um auf die Neutrinomasse zu schließen. Problematisch ist hierbei, daß dann weitere systematische Effekte, wie Energieverluste durch inelastische Streuung in der Elektronenquelle, immer mehr an Einfluß gewinnen. Diese sind in der Regel nicht einfach zu bestimmen, daher ist die Analyse des Elektronenspektrums auf einen kleinen Bereich in der Nähe der Endpunktsenergie beschränkt.

Zur Bestimmung der Neutrinomasse muß man also das β–Spektrum nahe dem Endpunkt vermessen. Man möchte einen Bereich untersuchen, der sehr geringe Zählraten $\frac{dN}{dE}$ aufweist. In diesem Zusammenhang ergeben sich dann auch die Vorteile, die Tritium als Quellmedium bietet:

- Die Endpunktsenergie liegt bei $E_0 = 18.6\ keV$ sehr niedrig, d.h. der relative

Anteil des auf die Neutrinomasse sensitiven Bereichs ist hier groß.

- Der β–Zerfall von Tritium ist ein supererlaubter Übergang in den Spiegelkern, dies führt zu einem sehr großen, elektronenenergieunabhängigen Kernmatrixelement $|\mathrm{M}|^2 = 5.5$.
- Tritium ist ein Kern mit einfach aufgebauter Elektronenhülle. Die Endzustände (V_n, E_n) lassen sich daher berechnen, was für das theoretische Modell des Spektrums (Gl. 2.5) wichtig ist.
- Tritium hat eine kurze Halbwertszeit von 12,3 Jahren, d.h. auch bei geringen Quelldicken können relativ hohe Zählraten erreicht werden. Dies ist wichtig, um Verluste durch inelastische Streuung im Quellmaterial zu vermeiden, die die Form des Spektrums verändern würden.

Auf dieser Basis lassen sich Kriterien aufstellen, die ein Experiment zur Messung der Neutrinomasse erfüllen muß:

- Der akzeptierte Raumwinkel sollte aufgrund der niedrigen Zählraten im Endpunktsbereich des Spektrums möglichst groß sein.
- Die Energieauflösung sollte im Bereich der statistischen Empfindlichkeitsgrenze für die erwartete Neutrinomasse liegen. Eine Empfindlichkeit auf $m_{\bar{\nu}_e} = 2\ eV/c^2$ erfordert in etwa eine relative Auflösung von $\frac{\Delta E}{E}$ von 10^{-4}.
- Die Untergrundzählrate im Betrieb soll niedrig sein, ein gutes Signal–zu–Rausch–Verhältnis ist wichtig. Eine zu hohe Untergrundzählrate würde die sehr niedrige Zählrate im Endpunktsbereich überdecken und damit die Sensitivität auf die Neutrinomasse zerstören.

Diese Anforderungen erfüllt das in Mainz entwickelte Spektrometer, dessen experimenteller Aufbau anschließend vorgestellt wird.

2.2 Das Experiment in Mainz

Das in Mainz betriebende Experiment basiert auf einem dort entwickelten neuen Spektrometertypen, dem Solenoid-Retardierungs-Spektrometer (SRS). Das Spektrometer wird heute MAC–E–Filter (Magnetic Adiabatic Collimation with Electrostatic Filter) genannt und im folgenden Abschnitt ausführlich vorgestellt.

Die β–Elektronenquelle ist ein schockkondensierter Film aus molekularem Tritium, der auf unter 1.9 K gekühlt wird. Oberhalb von 2 K würde sich die Filmoberfläche aufrauhen, dadurch würden zusätzliche unberechenbare Energieverluste auftreten. Der Film ist etwa 480 dick und hat einen Durchmesser von 17 mm, die Aktivität beträgt etwa 20 mCi.

Als Detektor wird ein ionenimplantierter Silizium-Halbleiterdetektor verwendet. Der Detektor ist in fünf konzentrische Segmente gleicher Fläche (1 cm^2) aufgeteilt, seine Energieauflösung beträgt 1.4 keV (FWHM) für Energien in der Umgebung von 18 keV.

Dieses Kapitel wird sich hauptsächlich mit den Eigenschaften des Spektrometers befassen, alle Details zu Quelle und Detektor findet man in [Bor00].

2.2.1 Funktionsweise des MAC-E-Filters

Zur Bestimmung der Ruhemasse des Neutrinos wird das β–Spektrum der Zerfallselektronen gemessen, um dann über die Energieerhaltung Rückschlüsse auf die Neutrinomasse machen zu können. Zentrales Instrument des Mainzer Experimentes ist daher das MAC-E-Filter, ein integrierendes Spektrometer, das mit einem elektrostatischen Gegenfeld die Energie der Elektronen analysiert. Um dies möglichst effizient zu tun, wird das Prinzip der magnetischen adiabatischen Kollimation genutzt.

Anhand von Abbildung 2.2a soll nun die Funktionsweise des MAC-E-Filters erläutert werden. Zwei supraleitende Solenoide erzeugen im Spektrometer, zwischen den Solenoiden, ein inhomogenes Magnetfeld, das symmetrisch zur Mittelebene, der sog. Analysierebene, als Führungsfeld dient. Im Spektrometer zwischen den Solenoiden wird der Magnetische Flußschlauch von einer mehrteiligen zylindersymmetrischen Elektrodenkonstruktion umschlossen. Diese Elektroden erzeugen das zur Mittelebene symmetrische elektrostatische Gegenfeld. In der Mittelebene entsteht somit das maximale elektrische Potential – daher die Bezeichnung Analysierebene.

Im linken Solenoiden ist die β–Elektronen Quelle positioniert, hier herrscht das Magnetfeld B_s. Die Quelle strahlt isotrop in den vorderen Halbraum ab. Alle Elektronen werden durch das starke Magnetfeld auf Zyklotronbahnen gezwungen, jedes Elektron beschreibt eine Spirale um eine Magnetfeldlinie, die sog. Führungsfeldlinie. Es ist sinnvoll in dieser Situation die gesamte kinetische Energie der Elektronen in zwei Komponenten aufzuspalten, eine longitudinale kinetische Energie $E_{\|} = E_{kin}\left(\frac{\vec{B}\cdot\vec{v}}{|B|\cdot|v|}\right)^2$ entlang der Führungsfeldlinie und eine transversale kinetische

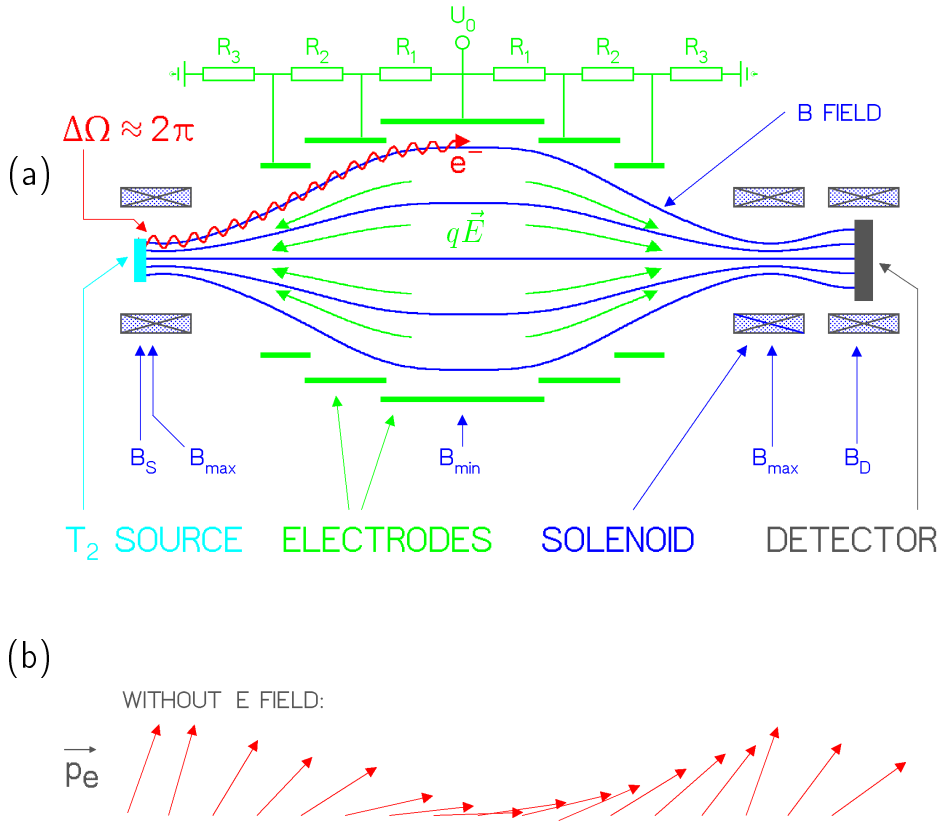

Abbildung 2.2: MAC-E-Filter In (a) ist der schematische Aufbau eines MAC-E-Filters dargestellt. Die Solenoide erzeugen den von Elektroden umgebenen magnetischen Flußschlauch, der die Elektronen von Quelle zu Detektor führt.
Teil (b) stellt die Transformation der Transversalenergie in die Longitudinalenergie graphisch am Beispiel des Impulsvektors dar. Dies ist die Folge eines konstanten magnetischen Moments μ in einem inhomogenen statischen Magnetfeld.

Energie $E_\perp = E_{kin} - E_{||}$ in der Zyklotronbewegung. Während die Elektronen nun den Magnetfeldlinien in Richtung Analysierebene folgen, nimmt das Magnetfeld um den Faktor 4000 ab, d.h. es wirkt die Gradientenkraft

$$\vec{F}_\nabla = \vec{\nabla}\left(\vec{\mu}\cdot\vec{B}\right) \tag{2.6}$$

$$\text{mit} \quad \mu = |\vec{\mu}| = \frac{e}{2m_e}|\vec{l}| = \frac{E_\perp}{B}. \tag{2.7}$$

Diese Kraft beschleunigt die Elektronen in longitudinaler Richtung. Da die Gesamtenergie im statischen Magnetfeld erhalten ist, kann dies nur auf Kosten der Transversalenergie geschehen. Durch diese Transformation der Transversal- in die Longitudinalenergie wird praktisch die gesamte kinetische Energie der Elektronen elektrostatisch analysierbar, denn das elektrostatische Gegenfeld wirkt nur auf die longitudinale Komponente der Bewegung.

In Abschnitt 2.2.4 wird die Energietransformation der adiabatischen Näherung im

relativistischen Fall beschrieben. In nichtrelativistischer Näherung gilt, daß die transversale kinetische Energie des Elektrons in gleichem Maße abnimmt, wie die magnetische Feldstärke. Das Teilchen am Ort i im Magnetfeld B_i mit transversaler Energie $E_{\perp i}$ wird am Ort f im Magnetfeld B_i die Energie

$$E_{\perp f} = E_{\perp i} \cdot \frac{B_f}{B_i} \tag{2.8}$$

haben.[1] Im Feldmaximum B_{max} startende Elektronen werden in der Analysierebene, wo das Magnetfeld minimal ist ($B_{min} \approx B_{max}/4000$), fast ihre gesamte transversale kinetische Energie in die Longitudinalbewegung transferiert haben, die vom elektrostatischen Gegenfeld analysiert werden kann. Mit Hilfe dieser Gleichung läßt sich ein Wert für das Auflösungsvermögen des Spektrometers angeben. Mit Auflösung ist in diesem Zusammenhang das Energieintervall ΔE gemeint, in welchem die Transmission von 0% auf 100% ansteigt. Wenn man von der maximal möglichen Energie in der Zyklotronbewegung ausgeht, der Startenergie E_S, ergibt sich:

$$\Delta E = E_{\perp \mathrm{f,max}} = E_S \cdot \frac{B_{\mathrm{min}}}{B_{\mathrm{max}}} \approx 4.8 \text{ eV} \tag{2.9}$$

Das parallel zum Magnetfeld ausgerichtete elektrostatische Gegenfeld erfüllt zwei Aufgaben. Eine Hauptaufgabe des elektrostatischen Gegenfeldes ist seine Funktion als Energiefilter. Nur Elektronen, deren longitudinale Energie größer als das Potential in der Analysierebene ($-eU_0$) ist, können diese überwinden und zum Detektor gelangen.

$$E_{\|\mathrm{f}} = E_S - E_{\perp \mathrm{f}} > -e \cdot U_0 \tag{2.10}$$

Nicht alle Elektronen, die die Quelle isotrop in den vorderen Halbraum emittiert, können das Analysierpotential $-eU_0$ überwinden. Es gilt, daß das Überwinden des Analysierpotentials nur dann möglich ist, wenn zwischen Quelle und Analysierebene die longitudinale kinetische Energie $E_{\|}$ nicht verschwindet. Die Transformation von transversaler kinetischer Energie zu longitudinaler kinetischer Energie muß schneller sein, als die Retardierung der longitudinalen kinetischen Energie durch das Gegenpotential. Alle diese Kriterien fließen in die sog. Transmissionsfunktion, die weiter unten beschrieben wird, ein.

Eine andere Aufgabe des Gegenfeldes ist, die Elektronen auf ihrem Weg zur Analysierebene abzubremsen, indem longitudinale kinetische Energie in potentielle Energie umgewandelt wird. Dieser Vorgang ist wichtig, damit sich die Elektronen weiterhin adiabatisch bewegen. Adiabatisch bedeutet in diesem Zusammenhang, daß die Energietransformation gemäß Gleichung (2.8) stattfindet, was durch eine über eine Zyklotronschrittlänge hinreichend kleine relative Magnetfeldänderung $\frac{\Delta B}{B}$ erreicht werden kann. Die hier geltenden Zusammenhänge werden ausführlich in den folgenden Abschnitten beschrieben. An dieser Stelle ist wichtig zu wissen, daß das magnetische Bahnmoment der Elektronen $\vec{\mu}$ eine Erhaltungsgröße ist.

[1] Dies ist der Umgekehrte Fall im Vergleich zum magnetischen Spiegel, bei dem ein Teilchen in das höhere Magnefeld fliegt. Nach 2.8 nimmt dann die Transversalenergie zu und die Longitudinalenergie nimmt bis zum Erreichen des Reflektionspunktes ab.

Ein weiterer limitierender Faktor für die Transmission ist die Position der Tritiumquelle. Sie sitzt nicht genau im Magnetfeldmaximum B_{max} des Solenoiden, sondern weiter links in Abbildung 2.2 im niedrigeren Feld B_S. Dies führt dazu, daß der maximal akzeptierte Raumwinkel kleiner als 90° ist und alle Elektronen, die mit einem Winkel größer θ_{max} starten, vor dem Feldmaximum magnetisch reflektiert werden. Der maximal akzeptierte Startwinkel θ_{max} ergibt sich aus dem Verhältnis der Magnetfelder am Startort B_S und im Feldmaximum B_{max} zu

$$\theta_{\max} = \arcsin\left(\sqrt{\frac{B_S}{B_{\max}}}\right). \tag{2.11}$$

Die Quelle wird bewußt in einem schwächeren Magnetfeld $B_S < B_{max}$ positioniert, um Elektronen mit hohem Startwinkel zu entfernen. Elektronen mit großen Startwinkeln haben eine, durch die Zyklotronbewegung, sehr viel längere Flugbahn im Quellmaterial und daher ist die Wahrscheinlichkeit für Energieverluste durch inelastische Stöße mit Tritiummolekülen viel größer.

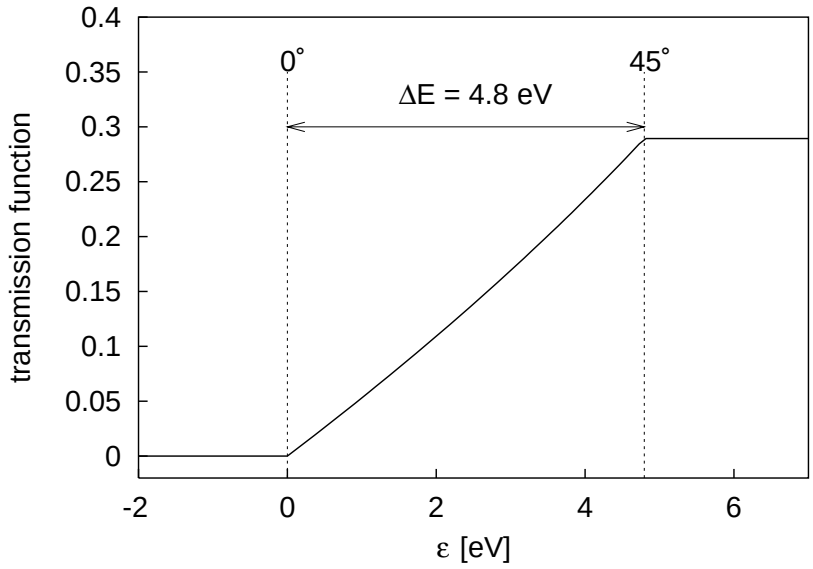

Abbildung 2.3: Die Transmissionsfunktion des MAC-E-Filters. Zu sehen ist die Form der Transmissionsfunktion. Für die jeweilige Überschußenergie in der Analysierebene ε beschreibt sie, wieviele Elektronen das Analysierpotential überwinden, im Verhältnis zur Gesamtzahl der in den vorderen Halbraum emittierten Elektronen. Die Breite des Anstiegs ΔE wird durch die Spektrometerauflösung definiert.

Die Transmissionsfunktion

$$T_H^0(\varepsilon) = \begin{cases} 0 & \text{für} \quad \varepsilon < 0 \\ 1 - \sqrt{1 - \frac{\varepsilon}{E_S}\frac{B_S}{B_A}} & \text{für} \quad 0 \le \varepsilon \le \Delta E \quad \text{mit} \quad \varepsilon = E_S - eU_0 \\ 1 - \sqrt{1 - \frac{\Delta E}{E_S}\frac{B_S}{B_A}} & \text{für} \quad \varepsilon > \Delta E \end{cases} \tag{2.12}$$

läßt nur Winkel bis θ_{max} zu und hängt von der Startenergie der Elektronen E_S, dem Verhältnis der Magnetfelder vom Quellort zur Analysierebene $\frac{B_S}{B_A}$, dem Retardierungspotential U_0 und der Spektrometerauflösung ΔE (s. Gl. (2.9)) ab. Der Parameter ε beschreibt die Überschußenergie in der Analysierebene.

Das so aufgebaute Spektrometer integriert die Zählrate der Elektronen über ihre Energie, es arbeitet im Prinzip wie ein Hochpassfilter. Es werden alle Elektronen nachgewiesen, deren Energie ausreicht, um die Analysierebene zu erreichen. Das Spektrometer kann auch in einem nicht integrierenden Flugzeitmodus (TOF) betrieben werden, wie ausführlich in [Bon99] beschrieben wird.

2.2.1.1 Parameter des Mainzer MAC–E–Filters

Hier sollen kurz die technischen Parameter des in Mainz verwendeten MAC–E–Filters beschrieben werden. In Mainz wird ein etwa 4 Meter langer zylindrischer Vakuumtank verwendet, der eine Konstruktion von 27 Elektroden beinhaltet, wie in Abbildung 2.4 zu sehen ist. Der Vakuumtank selbst liegt auf Erdpotential, die Daten

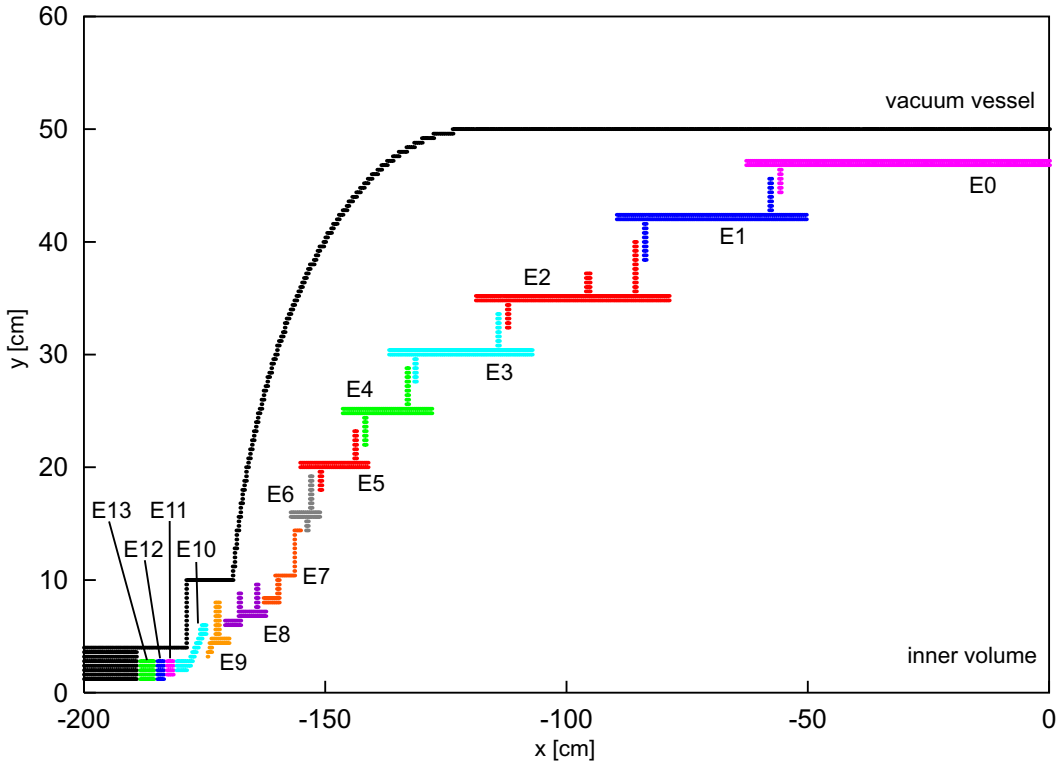

Abbildung 2.4: Elektrodenkonfiguration des Mainzer MAC–E–Filters. Gezeigt ist die Mainzer Elektrodenkonfiguration mit Numerierung der einzelnen Elektroden. Diese Elektrodenkonfiguration ist im Spektrometer spiegel- und zylindersymmetisch realisiert.

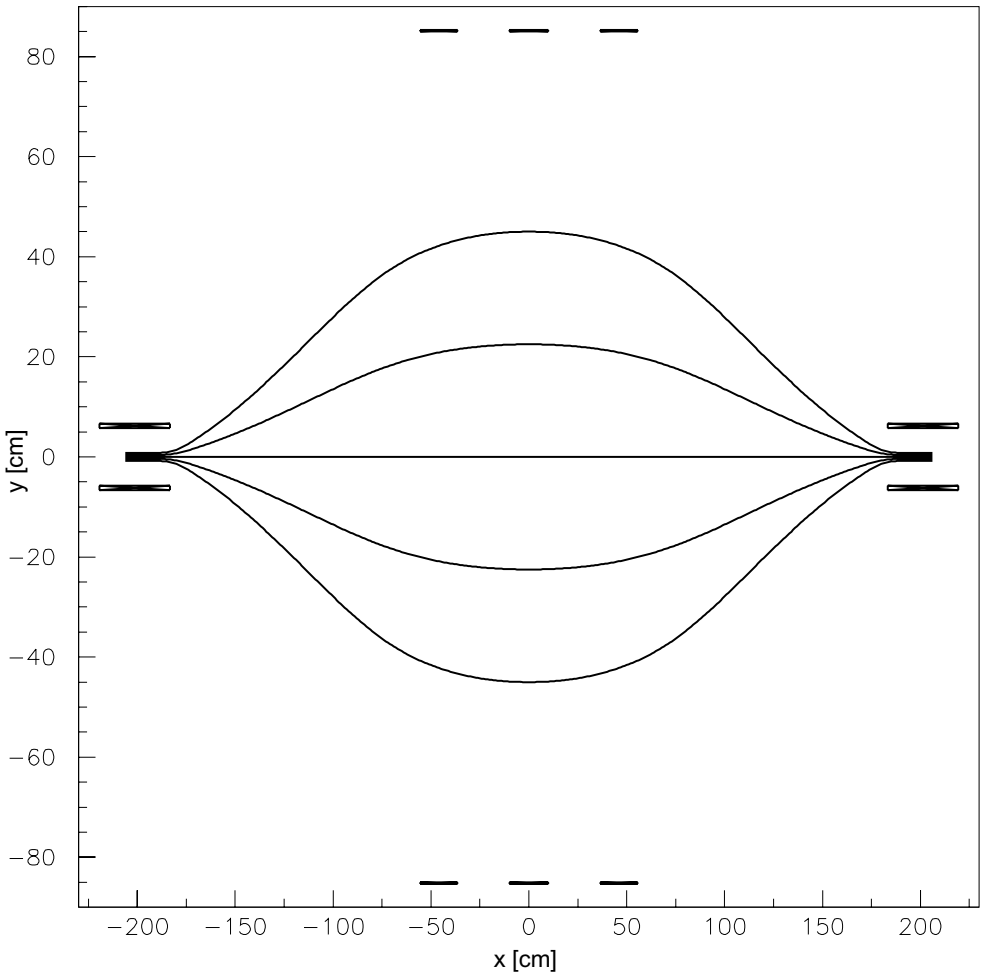

Abbildung 2.5: Spulenkonfiguration des Mainzer MAC–E–Filters. In Mainz werden zwei Solenoide verwendet, um das Magnetfeld des MAC–E–Filters zu erzeugen. Zur Korrektur des Magnetfeldes werden drei Luftspulen im zentralen Bereich verwendet. Eingezeichnet sind die Magnetfeldlinien mit Radien von $R = \pm 45\ cm$, $R = \pm 22.5\ cm$ und $R = 0\ cm$ in der Analysierebene ($x = 0\ cm$). Die eingezeichneten Magnetfeldlinien zeigen den magnetischen Flußschlauch, der Elektronen von Quelle zu Detektor führt.

der Elektroden sind ausführlich in [Bor00] nachzulesen und in Tabelle 2.1 kurz zusammengefaßt. Durch diese Elektrodenkonfiguration wird das zur Energiefilterung nötige, $U_0 = -18690\ V$ starke, Analysierpotential erzeugt. Die Elektroden umschließen spiegel- und zylindersymmetrisch den magnetischen Flußschlauch, der von zwei Solenoiden und drei Luftspulen erzeugt wird. Die Spulenkonfiguration ist in Abbildung 2.5 mit Magnetfeldlinien zu sehen. Die Solenoide erzeugen ein Magnetfeld von $B_{max} \approx 1.8\ Tesla$, das in der Analysierebene auf $B_{min} \approx 5\ Gauss$ abfällt. Die Zerfallselektronen des Tritium–β–Zerfalls werden von diesem Magnetfeld von der Quelle bis zum Detektor geführt, ohne dabei gegen eine Elektrode zu stoßen. Die Parameter der Spulenkonfiguration sind im Anhang B.3 am Beispiel einer '.par' Datei abgedruckt.

Tabelle 2.1: Parameter der Elektrodenkonfiguration des Mainzer MAC–E–Filters. Hier sind die Potentiale und die inneren Radien der Mainzer Elektrodenkonfiguration nochmals zusammengefaßt. Bei Elektroden mit radialer Ausdehnung ist der mittlere Radius angegeben. Eine ausführliche Beschreibung ist in [Bor00] nachzulesen.

Elektrode NR	innerer Radius	Potential
E0	$R = 47\ cm$	$U = -18690\ V$
E1	$R = 42\ cm$	$U = -18684\ V$
E2	$R = 35\ cm$	$U = -18672\ V$
E3	$R = 30\ cm$	$U = -18645\ V$
E4	$R = 25\ cm$	$U = -18597\ V$
E5	$R = 20\ cm$	$U = -18535\ V$
E6	$R = 15.5\ cm$	$U = -18351\ V$
E7	$R = 9.1\ cm$	$U = -17650\ V$
E8	$R = 7.5\ cm$	$U = -15119\ V$
E9	$R = 4.6\ cm$	$U = -14180\ V$
E10	$R = 2.9\ cm$	$U = -7602\ V$
E11	$R = 1.7\ cm$	$U = -5723\ V$
E12	$R = 1.3\ cm$	$U = -3843\ V$
E13	$R = 1.2\ cm$	$U = -1964\ V$

2.2.2 Adiabatische Näherung 1. Ordnung

In einem orts- und zeitunabhängigen magnetischen Feld bewegen sich geladene Teilchen auf Zyklotronbahnen. Diese Flugbahn kann man durch eine Kreisbewegung beschreiben, deren Zentrum sich entlang der Magnetfeldlinie bewegt. Wenn das Magnetfeld nicht mehr gleichmäßig und zeitunabhängig ist, erwartet man, daß diese ideale Bewegung nicht mehr stattfindet. Man kann aber andererseits davon ausgehen, daß weiterhin eine annähernd zyklotronartige Bewegung vorhanden ist. Diese Bewegung sollte sich daher durch eine Näherungsrechnung aus der idealen Bewegung mit dazu senkrechten Korrekturen beschreiben lassen. Genau dies ist die Idee hinter der adiabatischen Näherung, einer Bewegung entlang des Führungszentrums der Zyklotronbewegung eines geladenen Teilchens.

Zur Bahnberechnung eines geladenen Teilchens (Ladung q und Masse m) in elektromagnetischen Umgebungen ist es in einfachen Fällen möglich, die Bewegungsglei-

chung

$$m\ddot{\vec{r}} = \frac{q\dot{\vec{r}}}{c} \times \vec{B}(\vec{r},t) + q\vec{E}(\vec{r},t) \tag{2.13}$$

direkt zu intergrieren. Ist dies nicht möglich, muß auf Näherungsrechnungen zurückgegriffen werden. Unter der Annahme einer Kreisbewegung entlang des Führungszentrums zerlegt man die Ortskoordinate in $\vec{r} = \vec{R} + \vec{\varrho}$, mit dem Ortsvektor $\vec{R}$ auf das Führungszentrum und den Zyklotronvektor $\vec{\varrho}$ vom Führungszentrum zum Zyklotronorbit.

Der Zyklotronvektor habe nun die Form $\vec{\varrho} = \varrho\ (\hat{e}_2\ sin\,\omega t + \hat{e}_3\ cos\,\omega t)$, mit $\omega = \frac{qB(\vec{R})}{mc}$, der Zyklotronfrequenz im Magnetfeld am Punkt $\vec{R}$. Die Einheitsvektoren $\hat{e}_2$ und $\hat{e}_3$ sind senkrecht zur Richtung von $\vec{B}(\vec{R})$ und zueinander. Substituiert man damit die Bewegungsgleichung (2.13) und mittelt über einen Zyklotronumlauf, erhält man nach einigen Umformungen [Nor61] mit den Einheitsvektoren die Gleichung

$$\ddot{\vec{R}} = \frac{q}{m}\left[\vec{E}(\vec{R}) + \frac{1}{c}\dot{\vec{R}} \times \vec{B}(\vec{R})\right] - \frac{\mu}{m}\nabla\vec{B}(\vec{R}) + \mathcal{O}\left(\frac{m}{q}\right) \tag{2.14}$$

mit dem magnetischen Moment $\mu = \frac{q\varrho^2\omega}{2c} = \frac{mv_\perp^2}{2B}$. In Gleichung (2.14) werden nur Terme nullter Ordnung in $\frac{m}{q}$ berücksichtigt.

Es stellt sich jetzt die Frage, warum man rein technisch als Entwicklungsparameter $\frac{m}{q}$ verwendet. Diese Frage läßt sich nur sehr knapp beantworten und zur genauen Begründung muß auf [Nor61] verwiesen werden. In [Nor61] wird die Bewegungsgleichung in dimensionsloser Form aufgestellt. Der Zyklotronradius ϱ wird mit der charakteristischen Größe L des Systems dimensionslos. Die charakteristische Größe L des Systems ist die Länge, über die sich das Magnetfeld so wenig ändert, daß die adiabatische Näherung (2.14) gut erfüllt ist. Es ergibt sich dann

$$\frac{\varrho}{L} = \frac{m}{q} \cdot \frac{cv_\perp}{B_0 L}, \tag{2.15}$$

wobei in der dimensionslosen Formulierung $\frac{\varrho}{L}$ der Entwicklungsparameter ist. In [Nor61] wird jetzt Gleichung (2.15) und deren Proportionalität zu $\frac{m}{q}$ ausgenutzt, um $\frac{\varrho}{L}$ klein zu machen. Natürlich gilt dies nur rein mathematisch, d.h. um physikalische Ergebnisse zu erhalten, muß man die anderen Parameter in Gleichung (2.15) in dem Maße verkleinern, wie es für $\frac{m}{q}$ erforderlich wäre. Der Vorteil von $\frac{m}{q}$ als Entwicklungsparameter liegt nun darin, daß er in der normalen Bewegungsgleichung automatisch auftritt und man nicht den Umweg über die dimensionslose Formulierung machen muß.

Die senkrechte Komponente von $\dot{\vec{R}}$ gibt die senkrechte Abweichung der Bewegung entlang des Führungszentrums an. Man erhält diese senkrechte Komponente durch das Vektorprodukt von $\ddot{\vec{R}}$ mit $\vec{B}(\vec{R})$.

$$\vec{B}(\vec{R}) \times \ddot{\vec{R}} = \frac{q}{m}\left[\vec{B}(\vec{R}) \times \vec{E}(\vec{R}) + \frac{1}{c}\vec{B}(\vec{R}) \times \dot{\vec{R}} \times \vec{B}(\vec{R})\right] - \frac{\mu}{m}\vec{B}(\vec{R}) \times \nabla\vec{B}(\vec{R}) + \mathcal{O}\left(\frac{m}{q}\right) \tag{2.16}$$

Nach wenigen Umformungen, der Beschränkung auf die senkrechte Komponente und Auflösen nach $\dot{\vec{R}}_\perp$ ergibt sich eine Gleichung für die senkrechte Driftgeschwindigkeit.

$$\dot{\vec{R}}_\perp = c\frac{\vec{E}\times\vec{B}}{B^2} + \frac{\mu c}{q}\frac{\vec{B}\times\nabla\vec{B}}{B^2} + \frac{mc}{q}\frac{\vec{B}\times\ddot{\vec{R}}}{B^2} + \mathcal{O}\left(\frac{m^2}{q^2}\right) \tag{2.17}$$

Schon an dieser Gleichung kann man die ersten beiden Terme als $\vec{E}\times\vec{B}$–Drift und magnetische Gradientendrift klar identifizieren. Der dritte Term in dieser Gleichung enthält noch sechs weitere Driftterme der ersten Ordnung in der zweiten Ableitung $\ddot{\vec{R}}$ des Ortsvektors. Analog zu [Kun66] wird nun mit Hilfe von Gleichung (2.17), die darin enthaltene zweite Ableitung $\ddot{\vec{R}}$ im senkrechten Fall gebildet. Man muß hier nur Terme nullter Ordnung berücksichtigen, da der dritte Term in Gleichung (2.17) ohnehin schon mit $\frac{m}{q}$ multipliziert wird und damit der ersten Ordnung angehört.

Schließlich ergibt sich eine Gleichung für die senkrechte Driftgeschindigkeit, die alle acht Driftanteile bis einschließlich erster Ordnung in adiabatischer Näherung enthält.

$$\begin{aligned}
\dot{\vec{R}}_\perp \;=\;& c\frac{\vec{E}\times\vec{B}}{B^2} + \frac{\mu c}{q}\frac{\vec{B}\times\nabla\vec{B}}{B^2} + \frac{mc}{q}\frac{|\dot{\vec{R}}_\parallel|}{B^2}\vec{B}\times\frac{\partial\hat{e}_1}{\partial t} \\
&+\frac{mc}{q}\frac{|\dot{\vec{R}}_\parallel|}{B^2}\vec{B}\times\left(\frac{c\vec{E}\times\vec{B}}{B^2}\nabla\hat{e}_1\right) + \frac{mc}{q}\frac{|\dot{\vec{R}}_\parallel|^2}{B^2}\vec{B}\times\frac{\partial\hat{e}_1}{\partial s} \\
&+\frac{mc}{q}\frac{1}{B^2}\vec{B}\times\frac{\partial}{\partial t}\left(\frac{c\vec{E}\times\vec{B}}{B^2}\right) + \frac{mc}{q}\frac{1}{B^2}\vec{B}\times\left(\frac{c\vec{E}\times\vec{B}}{B^2}\nabla\left(\frac{c\vec{E}\times\vec{B}}{B^2}\right)\right) \\
&+\frac{mc}{q}\frac{|\dot{\vec{R}}_\parallel|}{B^2}\vec{B}\times\frac{\partial}{\partial s}\left(\frac{c\vec{E}\times\vec{B}}{B^2}\right) + \mathcal{O}\left(\frac{m^2}{q^2}\right)
\end{aligned} \tag{2.18}$$

Der in dieser Gleichung interessante Term, neben den schon bekannten, ist Term fünf, der die Krümmungsdrift beschreibt. In der Ableitung wird mit s ein Wegelement entlang der Führungsfeldlinie bezeichnet. Die Terme mit Ableitungen von $\frac{c\vec{E}\times\vec{B}}{B^2}$ sind vernachlässigbar klein für schwache elektrische Felder. Wenn die magnetischen Feldlinien nur langsam gekrümmt werden, sind auch Terme proportional zu $\frac{\partial\hat{e}_1}{\partial t}$ vernachlässigbar. In der Plasmaphysik beschreibt der Term proportional zu $\nabla\frac{c\vec{E}\times\vec{B}}{B^2}$ die Scherdrift oder sog. Helmholtz-Instabilität[2].

Von all diesen Drifttermen erster Ordnung werden im folgenden nur die $\vec{E}\times\vec{B}$–Drift, die Gradientendrift und die Krümmungsdrift zur Beschreibung der Teilchenbewegung verwendet, alle anderen Terme werden vernachlässigt. Nach weiteren Umformungen kann man die drei Korrekturterme der Bewegung des Führungszentrums in Abhängigkeit der kinetische Energie und für Elektronen mit $q = -e$ in folgender Weise schreiben.

[2] In [Nor61] wird die Helmholtz-Instabilität am Beispiel des Sonnenwindes, der um das Erdmagnetfeld herum fließt, erklärt.

$$\vec{E} \times \vec{B}\text{–Drift} \qquad \vec{u}_E = \frac{c}{B^2}\vec{E} \times \vec{B} \tag{2.19}$$

$$\text{Gradientendrift} \qquad \vec{u}_G = -\frac{cE_\perp}{eB^3}\vec{B} \times \nabla_\perp \vec{B} \tag{2.20}$$

$$\text{Krümmungsdrift} \qquad \vec{u}_C = -\frac{2cE_\parallel}{eB^3}\vec{B} \times \nabla_\perp \vec{B} \tag{2.21}$$

Die $\vec{E} \times \vec{B}$–Drift (Gl. (2.19)) resultiert aus der nicht überall erfüllten Parallelität von $\vec{E}$ und $\vec{B}$. Die zum Magnetfeld $\vec{B}$ senkrechte Komponente von $\vec{E}$ beschleunigt und bremst das Teilchen abwechselnd, d.h. die Zyklotronbahn wird verformt. Diese Verformung hat eine azimutale Driftbewegung zur Folge, die von der Teilchenladung unabhängig ist. Die kinetische Energie für diese Bewegung wird aus dem elektrischen Potential bezogen; über einen Zyklotronumlauf bleibt die kinetische Energie im Mittel gleich.

Die Gradientendrift (Gl. (2.20)) resultiert, wie der Name schon sagt, aus dem Gradienten des Magnetfeldes. Ändert sich das Magnetfeld während eines Zyklotronumlaufes, wirkt sich das direkt auf den zum Magnetfeld antiproportionalen Zyklotronradius aus. Es kommt hier zu einer Driftbewegung senkrecht zum Magnetfeld und der senkrechten Komponente des Magnetfeldgradienten. Die Zyklotronbewegung bleibt erhalten, nur ändert sich der Zyklotronradius während eines Umlaufes und es ergibt sich die Driftbewegung. Der Term ist proportional zur transversalen kinetischen Energie, steigt diese an, vergrößert sich proportional dazu der Zyklotronradius und die vom Teilchen durchquerten Magnetfeldgebiete weisen größere Differenzen auf. Der Driftterm beschreibt den Anteil der Zyklotronenergie, der für die senkrechte Driftbewegung erforderlich ist. Werden positiv geladene Teilchen betrachtet, ändert sich das Vorzeichen der Bewegung.

Schließlich beschreibt die Krümmungsdrift (Gl. (2.21)) eine Flugbahnkorrektur, die durch gebogene Magnetfelder verursacht wird. Dieser Driftterm ist nun proportional zur longitudinalen kinetischen Energie, anschaulich gesehen, will das Teilchen geradeaus weiter fliegen, wird jedoch von der gekrümmten führenden Magnetfeldlinie auf eine gekrümmte Bahn gezwungen. Durch diese Bewegung wird kinetische Energie aus der longitudinalen Komponente in die Zyklotronbewegung übertragen. Es ergibt sich eine Driftbewegung senkrecht zum Magnetfeld und dem Krümmungsradius, der proportional zum Magnetfeldgradienten ist. Auch diese Bewegung ändert ihre Richtung für positiv geladene Teilchen.

2.2.3 Magnetisches Moment als adiabatische Invariante

In [Jac82] wird gezeigt, daß in Gebieten mit hinreichend orts- und zeitunabhängigen magnetischen Feldern die adiabatische Invarianz der magnetischen Wirkung

$J = \frac{q}{c} B\pi r_c^2$, einen konstanten, von der Zyklotronbewegung eingeschlossenen, magnetischen Fluß zur Folge hat. Der magnetische Fluß durch die Bahn des Teilchens ist $B\pi r_c^2$. Bei wachsendem Magnetfeld B, nimmt der Zyklotronradius r_c gerade so ab, daß $B\pi r_c^2 = const.$ erfüllt ist.

Die Konstanz des von der Teilchenbahn eingeschlossenen magnetischen Flusses kann man auf unterschiedlichen Wegen formulieren, entweder über den Zyklotronradius der Teilchenbahn, den transversalen Teilchenimpuls oder über das magnetische Moment μ des Teilchens.

$$\left.\begin{array}{c} B r_c^2 \\ p_\perp^2 / B \\ \gamma\mu \end{array}\right\} \text{ sind adiabatische Invarianten} \tag{2.22}$$

In orts- und zeitunabhängigen magnetischen Feldern ist die Invarianz des magnetischen Moments erfüllt. Werden die Magnetfelder ortsabhängig, ist die Invarianz nur noch im nicht relativistischen Grenzfall $\gamma \to 1$ erfüllt.

Im folgenden soll das magnetische Moment als Invariante der adiabatischen Näherung im nicht relativistischen Fall betrachtet werden. Für das magnetische Moment wird eine asymptotische Reihe aufgestellt, die die Invariante in der adiabatischen Näherung darstellt.

$$\mu = \mu_0 + \xi\mu_1 + \xi^2\mu_2 + \mathcal{O}\left(\xi^3\right) \tag{2.23}$$

Für diese Reihe gilt im nicht relativistischen Fall für konstante Magnetfelder und verschwindende elektrische Felder $\mu = \mu_0 = \frac{mv_\perp^2}{2B}$, wobei $v_\perp$ die Geschwindigkeit senkrecht zur führenden Feldlinie ist. Als Entwicklungsparameter dient hier wieder $\xi = \frac{m}{q} \propto \frac{\varrho}{L}$, wie schon im vorherigen Abschnitt. Adiabatische Invariante ist nun μ, d.h. die vollständige Reihe ist die zu betrachtende Invariante.

Für kleine $\frac{\vec{E}\times\vec{B}}{B}$ ist schon μ_0 eine gute Näherung. Wird $\frac{\vec{E}\times\vec{B}}{B}$ nun größer, kann man die Driftterme nicht mehr vernachlässigen. Das Teilchen beschreibt jetzt keine exakt kreisförmige Zyklotronbewegung mehr, sondern die Azimutalbewegung kommt als senkrechter Bewegungsanteil hinzu und verzerrt die Zyklotronbahn (s. Gl. (2.18)). Betrachtet man jedoch weiterhin den mit der Azimutalbewegung mitbewegten Koordinatenrahmen (s. Gl. (2.19)), findet man wieder die bekannte Zyklotronbewegung.

Wenn das elektrische Feld nicht mehr vernachlässigbar ist, muß es in der Reihe für μ berücksichtigt werden. Die senkrechte Geschwindigkeitskomponente $v_\perp$ beinhaltet nun auch die Anteile für $\vec{E}\times\vec{B}$–Drift, Gradientendrift und Krümmungsdrift. Da die $\vec{E}\times\vec{B}$–Drift unabhängig vom magnetischen Moment ist, darf sie hier nicht beachtet werden [Nor63]. Man schreibt daher

$$\mu = \mu_0 = \frac{mv_\perp^2}{2B} \longrightarrow \mu = \frac{m\left(\vec{v}_\perp - \vec{u}_E\right)^2}{2B} \tag{2.24}$$

$$\vec{v}_\perp - \vec{u}_E = \vec{v}_{Cycl} + \vec{u}_C + \vec{u}_G \tag{2.25}$$

und verwendet für die Driftgeschwindigkeiten $\vec{u}_C$ und $\vec{u}_G$ Gleichungen (2.20) und (2.21). $\vec{v}_{Cycl}$ beschreibt den Geschwindigkeitsanteil der ursprünglichen Zyklotronbewegung. Die restlichen Terme aus Gleichung (2.18) werden hier vernachlässigt. Durch Sortieren nach Potenzen von $\frac{m}{q}$ und Vernachlässigung höherer Terme erhält man eine Reihe der Form

$$\mu = \mu_0 + \frac{m}{q}\mu_1 + \mathcal{O}\left(\frac{m^2}{q^2}\right) \tag{2.26}$$

mit den Komponenten

$$\mu_0 = \frac{m}{2B}|\vec{v}_{Cycl}|^2 \tag{2.27}$$

$$\mu_1 = \frac{c}{B^4}\left(E_\perp + 2E_{||}\right)\vec{v}_{Cycl}\cdot\left(\vec{B}\times\nabla\vec{B}\right). \tag{2.28}$$

An diesen beiden Reihentermen kann man sehr schön den Einfluß der Driftbewegung erkennen, in nullter Ordnung bleibt nur der ursprüngliche Zyklotronterm erhalten, in der ersten Ordnung sind die Korrekturen durch die Driftterme enthalten. Anschaulich kann man den zur transversalen kinetischen Energie proportionalen Term als Faktor verstehen, der beschreibt, welcher Anteil der Zyklotronenergie in der Azimutalbewegung steckt. Der zur longitudinalen kinetischen Energie proportionale Term beschreibt die Erhöhung der transversalen Energie auf Kosten der longitudinalen Energie.

Abschließend kann man festhalten, daß jedes Verhalten eines Teilchens, welches im Widerspruch zu dieser Reihenentwicklung steht, als nicht adiabatisch einzustufen ist. Gerade bei Näherungen mit niedriger Ordnung muß man davon ausgehen, daß es immer Teilchen geben wird, die sich nicht adiabatisch beschreiben lassen.

2.2.4 Adiabatische Energietransformation in relativistischer Form

Wie schon in Abschnitt 2.2.1 angesprochen, wird im MAC–E–Filter der Effekt der adiabatischen Energietransformation von transversaler zu longitudinaler kinetischer Energie ausgenutzt. Die adiabatische Energietransformation wird unter Verwendung der Erhaltungsgrößen von Gleichung (2.22) formuliert.

Um die Energietransformation relativistisch korrekt zu berechnen, geht man von der Impulserhaltung $\frac{p_\perp^2}{B} = const$ aus. Für die kinetische Energie ergibt sich

$$E_{kin} = \frac{1}{\gamma+1}\frac{p^2}{m} \tag{2.29}$$

als relativistischer Zusammenhang mit dem Impuls. Analog dazu läßt sich dann ein Zusammenhang zwischen Energie und Geschwindigkeit in relativistischer Form aufstellen.

$$|v| = \sqrt{\frac{\gamma+1}{m\gamma^2}E_{kin}} \tag{2.30}$$

Mit diesen Gleichungen kann man nun die Transformation der kinetischen Energie formulieren. Aus der Erhaltungsgleichung (2.22) ergibt sich für die transversale Komponente des Teilchenimpulses

$$\frac{p_\perp^2}{B} = const \Longrightarrow \frac{p_{\perp i}^2}{B_i} = \frac{p_{\perp f}^2}{B_f} \tag{2.31}$$

Unter Verwendung der obigen Zusammenhänge ergibt sich für die transversale kinetische Energie die Transformationsgleichung

$$E_\perp(\vec{x}) = \frac{\gamma(\vec{s})+1}{\gamma(\vec{x})+1} \cdot \frac{B(\vec{x})}{B(\vec{s})} E_\perp(\vec{s}), \tag{2.32}$$

wobei hier $\vec{x}$ die aktuelle Teilchenposition darstellt und $\vec{s}$ der Startort des Teilchens war. Analog zu dieser Transformationsgleichung ergibt sich die Transformation der transversalen Geschwindigkeit zu

$$|v_\perp(\vec{x})| = \frac{\gamma(\vec{x})}{\gamma(\vec{s})} \sqrt{\frac{B(\vec{x})}{B(\vec{s})}} |v_\perp(\vec{s})|. \tag{2.33}$$

Auf diesen beiden Transformationsgleichungen beruht auch das Programm ADIPARK, welches im dritten Kapitel ausführlich vorgestellt werden wird. Wenn man die Energietransformationsgleichung im nicht relativistischen Grenzfall betrachtet, erhält man gerade die Gleichung, die in Abschnitt 2.2.1 zur Energietransformation angegeben wurde.

2.2.5 Energiekorrektur als Folge der Driftbewegung

Wenn man die Erfahrungen aus dem letztem Abschnitt umsetzen will, muß man zur Energietransformation noch einige Anmerkungen machen. Die Erhaltung der Gesamtenergie bedeutet, daß die Summe der kinetischen Energieanteile und der potentiellen Energie konstant ist.

$$E_\parallel + E_\perp + E_{pot} = const. \tag{2.34}$$

Die transversale kinetische Energie setzt sich zusammen aus den Energieanteilen, die auf die Zyklotronbewegung E_{Cycl}, die Krümmungsdrift E_C, die Gradientendrift E_G und die $\vec{E} \times \vec{B}$–Drift E_E verteilt sind.

$$E_\perp = E_{Cycl} + E_C + E_K + E_E \tag{2.35}$$

Die adiabatische Energietransformation durch die Magnetfeldänderung wirkt sich in nullter Ordnung nur auf die reine Zyklotronbewegung aus, hier ändert sich E_{Cycl}. Der Energieanteil der Gradientendrift E_G wird gleichzeitig mittransformiert, da der Driftterm zur Zyklotronenergie E_{Cycl} proportional ist. Die Gradientendrift führt zu einer Verminderung der Zyklotronenergie, d.h. der Zyklotronradius wird kleiner. Andererseits ist in erster Ordnung auch diese Driftbewegung Teil des magnetisches Momentes, so daß dieses weiterhin erhalten bleibt.

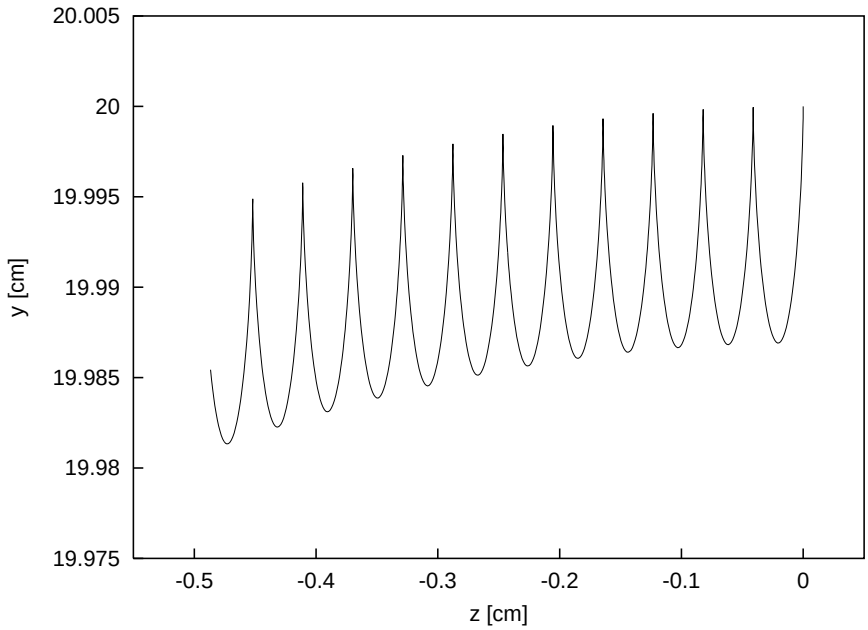

Abbildung 2.6: Nicht adiabatischer Energiegewinn eines Elektrons. Hier ist die von SIMION berechnete Flugbahn eines ruhend startenden Elektrons $E_{kin}^{start} = 0\ eV$ gezeigt. Das Elektron startet bei $(x, y, z) = (0, 20, 0)\ cm$. Das elektrische Feld führt dem Elektron Energie zu, in diesem Moment setzt dann die Driftbewegung ein.

Die Krümmungsdrift ist proportional zur longitudinalen kinetischen Energie, sie transformiert daher longitudinale kinetische Energie in Zyklotronenergie, d.h. Zyklotronenergie und Zyklotronradius nehmen zu, während die longitudinale kinetische Energie kleiner wird.

Die $\vec{E} \times \vec{B}$–Drift bezieht Energie aus dem Potential, d.h. bei Betrachtung des Führungszentrums der Bewegung wird dieser Effekt über einen Zyklotronumlauf nicht sichtbar. Jede Änderung des Ortes aufgrund der Driftbewegung ist gleichbedeutend mit einer Änderung der potentiellen Energie. Dieser Effekt wird sehr deutlich, wenn ein ruhendes Elektron in die Analysierebene des Spektrometers gesetzt wird (Abb. 2.6). Das Elektron wird durch den Potentialdurchgriff zum Zentrum des Spektrometers beschleunigt, ihm wird Energie aus dem elektrischen Potential zugeführt. Sobald sich das Teilchen bewegt, wirken wieder die bekannten Driftanteile und zwingen es auf eine azimutale Bahn. Die Gesamtenergie des Teilchens bleibt erhalten, aber die mittlere kinetische Energie steigt an. Ist dies der Fall, ist die Adiabasie verletzt. Ein solches Verhalten ist mit adiabatischer Näherung nicht mehr zu beschreiben. Eine Lösung dieses Problems wäre eine vom Algorithmus generierte Verschiebung des Führungszentrums, um den Effekt des elektrischen Feldes auszugleichen. Ein anderer Ansatz wäre, zu untersuchen, ob ein Term höherer Ordnung in der adiabatischen Näherung dieses Verhalten beschreibt.

Abschließend muß noch betont werden, daß die Energieänderungen durch Driftbewegungen sehr klein sind und im allgemeinen im Bereich von etwa $10^{-3}\ eV$ bis $10^{-5}\ eV$ liegen.

Kapitel 3

Werkzeuge zur Teilchenbahnverfolgung

Im Rahmen dieser Arbeit sollten Methoden zur Reduzierung von Untergrundereignissen am Mainzer Tritium-β-Spektrometer entwickelt und durch Simulationen getestet werden. Diese Simulationen wurden mittels kommerzieller und selbst entwickelter Software durchgeführt. Im ersten Teil dieses Kapitels wird das kommerzielle Simulationsprogramm SIMION 3D vorgestellt, dessen elektrische Potentialfeldkarten auch von dem selbst entwickelten Programm benutzt werden. Der zweite Teil dieses Kapitels widmet sich dem neu entwickelten Simulationsprogramm ADIPARK. Die Idee zu ADIPARK, sein Aufbau und seine Eigenschaften werden hier detailliert vorgestellt. Schließlich sollen beide Simulationsprogramme miteinander verglichen werden.

3.1 SIMION 3D Version 7.0

Die Simulationssoftware SIMION 3D [SIM] wurde in der Version 7.0 unter Microsoft Windows NT 4.0 [MS] verwendet. SIMION 3D bietet die Möglichkeit geladene Teilchen relativistisch korrekt in beliebigen elektromagnetischen Umgebungen, die im Programm entworfen werden können, zu verfolgen. Hier wird es angewendet, zum Erstellen der elektrischen Potentialfeldkarten, zum Verifizieren der ADIPARK-Ergebnisse und zur Berechnung der Bahnen von geladenen Teilchen, auf die die adiabatische Näherung nicht mehr angewendet werden kann.

Zum Erstellen der Elektrodengeometrie wird der in SIMION 3D integrierte graphische Editor verwendet. Auf Basis der erstellten Geometrie berechnet SIMION 3D die elektrische Potentialkarte durch Relaxation. Ausgehend vom Potentialwert am Ort der definierten Elektroden wird durch diese Relaxations-Methode für jeden Gitterpunkt im Raumbereich das Potential als Funktion der Nachbarpunkte bestimmt. Dabei wird iterativ so lange verfeinert, bis ein anzugebender Grenzwert in der Potentialdifferenz unterschritten wird.

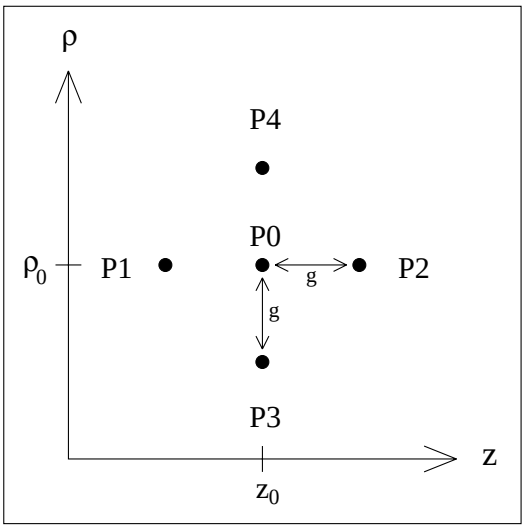

Abbildung 3.1: Relaxation in SIMION 3D. Schema zur Relaxation bei Gitter mit Gitterkonstante g im zweidimensionalen Fall. Das Potential am Punkt P0 wird solange iterativ verfeinert, bis die Differenz zu den Potentialwerten der Nachbargitterpunkte (P1 - P4) einen Grenzwert unterschreitet.

Die von SIMION 3D verwendeten Relaxations-Methoden unterscheiden sich geringfügig für zweidimensionale und dreidimensionale Probleme. Im zweidimensionalen, und damit auch im zylindersymmetrischen, Fall kommt Gl.(3.1) zum Einsatz. Beim Übergang zu drei Dimensionen werden die Punkte P5 und P6 vor und hinter Punkt P0 im Raum hinzugefügt, es ergibt sich dann Gleichung (3.2).

$$P0 = \frac{1}{4}(P1 + P2 + P3 + P4) \tag{3.1}$$

$$P0 = \frac{1}{6}(P1 + P2 + P3 + P4 + P5 + P6) \tag{3.2}$$

Zur Berechnung des elektrischen Potentials wird die Additivität der Lösungen der Laplace-Gleichung ausgenutzt. Es wird eine Potentialkarte für jede Elektrode separat berechnet und abgespeichert. Das Gesamtpotential ergibt sich dann als Superposition aller Potentialkarten der Einzelelektroden.

SIMION 3D verarbeitet Magnetfelder auf Grundlage von Feldkarten, die skalare magnetische Potentialwerte enthalten. In SIMION 3D gibt es nur die Möglichkeit auf Basis von nur mathematisch definierten magnetischen Monopolen ein Magnetfeld zu erzeugen. Die Problematik, für eine reale Spulenkonfiguration diese magnetische Monopolverteilung zu finden, wird in [Deg99] erläutert und kann auf folgendem Wege umgangen werden. Die magnetischen Potentialkarten werden mit dem von B. Flatt [Fla01] entwickelten Programm Bfield_3D auf Basis von Spulengeometrien erstellt. Bfield_3D verwendet die Informationen der Spulenparameter wie Position, Kippwinkel, Größe, Stromstärke und Windungszahl, um zylinderförmige dicke Spulen in einer Ebene im Raum zu positionieren, was es ermöglicht, Spulenanordungen zu berechnen, die in einer Ebene gekippt sind. Aus diesen Parametern kann Bfield_3D

außerhalb der Spulen zwei- und dreidimensionale Karten eines skalaren magnetischen Potentials für SIMION 3D erstellen, sowie Magnetfeldlinien und Magnetfelder berechnen.

Die elektrischen und magnetischen Kräfte werden von SIMION 3D getrennt berechnet. Die Bewegungsgleichung wird durch das Runge-Kutta-Verfahren in 4. Ordnung numerisch integriert. Um numerische Fehler zu minimieren, verwendet SIMION 3D eine dynamische Schrittweitensteuerung, die vom Benutzer über den sog. 'computation quality factor' Q bestimmt wird. Ein Kompromiss zwischen Rechengeschwindigkeit und ausreichender numerischer Genauigkeit ist nach [Fic00] ein Wert von $Q = 110$. Diese Einstellung wird auch in den hier durchgeführten Simulationen beibehalten.

Wie in [Fic00] diskutiert wurde, tritt in SIMION 3D eine Energieverletzung in Bereichen mit hohen elektrischen Feldgradienten auf. In Relation zur Spektrometerauflösung wurde diese Energieverletzung damals als vernachlässigbar klein angenommen. Die Frage ist, wie wirkt sich diese Energieverletzung auf ein gespeichertes Teilchen aus, wenn es sich sehr häufig in Bereichen hoher Feldgradienten aufhält. Zur Klärung dieser Frage sei auf Kapitel 3.3 verwiesen.

3.2 Das Simulationsprogramm ADIPARK

Um Methoden zur Untergrundreduzierung zu entwickeln, zu testen und Modifikationen am experimentellen Aufbau zu studieren, ist es wichtig Simulationen durchzuführen. Dafür wurde ein vollkommen neues Programm, das Simulationsprogramm ADIPARK, ADIabatic PARticle tracKing, zur Verfolgung von geladenen Teilchen in beliebigen elektromagnetischen Umgebungen, unter Verwendung der adiabatischen Näherung erster Ordnung, entwickelt. Es sollen Aussagen gemacht werden können, ob ein geladenes Teilchen gegen eine Elektrode fliegt oder gespeichert wird. Das Programm berechnet die Flugbahn anhand des Führungszentrums (engl. 'guiding center'), d.h. die Zyklotronbewegung wird vernachlässigt, dadurch reduziert man den Berechnungsaufwand und gleichzeitig die numerischen Fehler, da viel weniger Berechnungsschritte notwendig sind (siehe Abbildung 3.10).

3.2.1 Motivation für ein neues Simulationsprogramm

Vor der Entwicklung von ADIPARK war es nur möglich, im zylindersymmetrischen Spezialfall die Transmission von Teilchen entlang von Magnetfeldlinien (adiabatische Näherung nullter Ordnung) zu simulieren [Fic00], d.h. keine adiabatischen Korrekturen flossen in die Teilchenbewegung ein. Die zylindersymmetrische Einschränkung hat schon dazu geführt, daß in der Diplomarbeit von B. Flatt [Fla01] das Magnetfeldberechnungsprogramm Bfield_3D entwickelt wurde, um dreidimensionale Magnetfeldkarten zu erzeugen und mit der Möglichkeit, Magnete in einer Ebene gegeneinander

zu kippen, wie es für das KATRIN–Experiment erforderlich ist.

Teilchen in elektromagnetischen Umgebungen bewegen sich auf Zyklotronbahnen, d.h. sie beschreiben eine Spirale um die führende Magnetfeldlinie. In der adiabatischen Näherung wird die Teilchenbahn unter Vernachlässigung der Zyklotronbewegung beschrieben, man verfolgt das Teilchen dann entlang seines Führungszentrums. Bei dieser Methode reduzieren sich die Berechnungsschritte erheblich, was einerseits eine kürzere Berechnungszeit zur Folge hat und andererseits weniger anfällig für numerische Fehler ist. Das Programm ADIPARK wurde daher nach diesem Gesichtspunkt entwickelt.

Das Führungszentrum der Teilchenbahn bewegt sich entlang der Magnetfeldlinien, aber es gibt zusätzlich die adiabatischen Korrekturen erster Ordnung, die die Abweichungen senkrecht zur Teilchenbahn beschreiben. Auch diese Korrekturen wurden in ADIPARK eingebaut.

Um entscheiden zu können, wann ein Teilchen gespeichert wird und wann nicht, ist es notwendig die Reflektionspunkte der Teilchenbahn zu finden und das Teilchen in die entgegengesetzte Richtung weiter fliegen zu lassen. Unter Verwendung der adiabatischen Näherung ist die Gesamtenergie automatisch erhalten. Der Fehler in der Energieerhaltung, den die von SIMION 3D verwendete Methode erzeugt und der in [Fic00] beschrieben wird, kann auf diesem Weg umgangen werden. Obwohl die adiabatischen Korrekturen erster Ordnung nur nichtrelativistisch eingebaut wurden, wurde die Energietransformation in relativistisch korrekter Weise implementiert.

Die Einbeziehung von adiabatischen Korrekturen erster Ordnung mit der Möglichkeit gespeicherte Teilchen verfolgen zu können, ermöglicht es, die Ergebnisse von ADIPARK direkt mit den Ergebnissen von SIMION 3D zu vergleichen. Der Geschwindigkeitsvorteil gegenüber SIMION 3D, durch die Berechnung entlang des Führungszentrums bei ADIPARK, schlägt sich in deutlich kürzeren Berechnungszeiten nieder.

ADIPARK sollte vom Aufbau her so modular konzipiert werden, daß spätere Ergänzung oder alternative numerische Verfahren einfach in die Struktur eingebunden werden können. Ein selbst entwickeltes Simulationsprogramm bietet natürlich uneingeschränkten Einblick in den Programmquelltext, dieser bleibt dem Anwender unter SIMION 3D verwehrt. Mögliche Erweiterungen für ADIPARK sind:

- Abstrahlung von Synchrotronstrahlung.
- Wechselwirkungen mit Restgasmolekülen.
- Verfolgung der echten Teilchenbahn durch numerische Integration der dreidimensionalen Bewegungsgleichung (2.13).

3.2.2 Modularisierung und Funktionsweise

In diesem Abschnitt soll die Struktur und Modularisierung von ADIPARK vorgestellt und erläutert werden. Bei der Entwicklung von ADIPARK wurde großen Wert auf modulare Programmierung gelegt, d.h. die Problemstellung wurde in mehrere kleine Probleme zerlegt, die dann in einzelnen Modulen verkapselt gelöst wurden. Ziel war es, die Wiederverwendbarkeit der Programmkomponenten für zukünftige Projekte sicherzustellen. Die Struktur von ADIPARK ist in Abb.(3.2) dargestellt.

Hauptprogramm und Steuerzentrale ist das Modul **adipark**, welches die Routinen zum Bereitstellen der Potentialkarten initialisiert und auf Benutzereingaben wartet, um dann die entsprechenden Funktionen und Simulationsmodule wie

- Teilchenbahnverfolgung zu starten
- Speichervolumen zu berechnen
- Transmission eines MAC–E–Filters zu testen
- Anzeigen der Spulenkonfiguration
- Programm beenden

auszuführen.

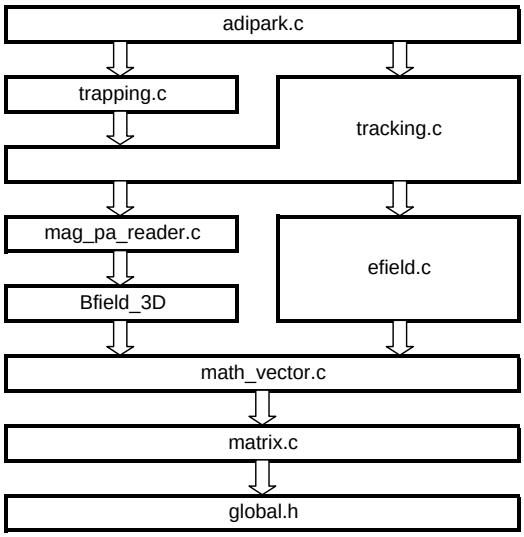

Abbildung 3.2: Modulare Struktur von ADIPARK. Darstellung der Hierarchie der in ADIPARK verwendeten Module. Jedem Modul stehen die darunter angeordneten Teilmodule zur Verfügung.

Kern des Simulationsprogrammes ist das Modul **tracking**, es liest die nötigen Startparameter ein und enthält den Hauptalgorithmus zur Teilchenbahnverfolgung (s.

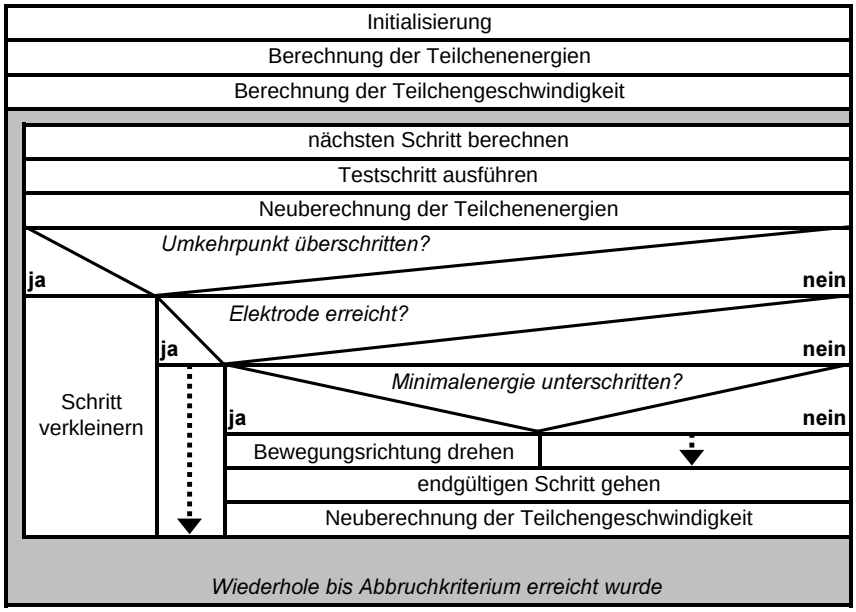

Abbildung 3.3: Struktogramm des Teilchenverfolgungsalgorithmus. Das Struktogramm zeigt die Hauptschleife zur Teilchenbahnverfolgung mit Verarbeitung von Reflektionspunkten, Kollisionen mit Elektroden und Schrittweitensteuerung.

Abb.(3.3)). Nachdem alle erforderlichen Parameter aus den Parameterdateien geladen und initialisiert wurden, werden Energien und Geschwindigkeiten des startenden Teilchens berechnet. Energien sind in diesem Zusammenhang die gesamte kinetische Energie, sowie die zum Magnetfeldvektor longitudinale $E_{||} = E_{kin} \left(\frac{\vec{B}\cdot\vec{v}}{|B|\cdot|v|}\right)^2$ und transversale $E_{\perp} = E_{kin} - E_{||}$ Komponente. Mit Kenntnis dieser Energien können dann auch die longitudinale und transversale Geschwindigkeit berechnet werden. In der Simulationsschleife wird dann der nächste Schritt entlang der Magnetfeldlinie durch Runge-Kutta 2. Ordnung [Num92] berechnet und die transversalen adiabatischen Korrekturen erster Ordnung (s. Gl. (2.19) bis(2.21)) hinzugefügt.

Die wesentlichen, bei der Entwicklung des Programms, auftretenden Probleme waren:

- **Eine korrekte Kollisionserkennung.** Fliegt ein Teilchen dicht an eine Elektrode, stellt sich die Frage, ob es reflektiert wird oder kollidiert. In diesem Fall muß der Algorithmus gewährleisten, daß das Teilchen nicht durch die Elektrode hindurchfliegt. Aufgrund des Interpolationsalgorithmus kann dies bei einer groben Gitterkonstante der Potentialfeldkarte und dünnen Elektroden eintreten.

- **Die korrekte Erkennung von Reflektionspunkten.** Verschwindet die longitudinale kinetische Energie, hat des Teilchen einen Reflektionspunkt erreicht. Ein Schritt über einen solchen Punkt würde eine negative longitudinale kinetische Energie ergeben. Um diesen Energienullpunkt möglichst gut approximieren zu können, wurde der Parameter *minimale longitudinale kinetische Energie*

eingebaut, der angibt wie weit die Energie bei Auftreten eines Reflektionspunktes absinken muß, bevor die Bewegungsrichtung gedreht wird.

All diese Probleme konnten durch die komplexe Schleifenform mit Schrittweitenverkleinerung erfolgreich bewältigt werden. Entscheidend war, daß jeder neue Schritt zuerst getestet wird, bevor er tatsächlich getan wird (s. Abb. 3.3).

Nachdem dem Schrittvektor die adiabatischen Korrekturen hinzugefügt wurden, wird dieser Schritt nun als Testschritt verwendet, um die Teilchenenergien an der neuen Position zu berechnen. Wird nun ein Energienullpunkt, d.h. Reflektionspunkt für das Teilchen, überschritten, wird die Schrittweite halbiert und der Testschritt erneut berechnet. Diese Schrittweitenverkleinerung wird dann solange wiederholt, bis der Reflektionspunkt ausreichend gut approximiert wurde. Wurde kein Reflektionspunkt überschritten, wird getestet, ob sich das Teilchen im Volumen einer Elektrode befindet, wobei die Schleife mit einer Kollisionsmeldung abgebrochen werden würde. War dies nicht der Fall, wird geprüft, ob das Minimallimit für die longitudinale kinetische Energiekomponente unterschritten wurde. Wenn ja, wird die Bewegungsrichtung des Teilchens für den nächsten Schritt umgekehrt. Anschließend wird der getestete Schritt endgültig gegangen, die Teilchenenergien werden neu berechnet und die Schleife beginnt erneut. Abbruchbedingungen sind ein Überschreiten der maximalen vorgegebenen Flugzeit, der maximalen Anzahl von Reflektionspunkten, Kollisionen mit Elektroden oder das Verlassen des Spektrometers durch eine der Öffnungen.

Das Modul **trapping** dient als Steuerroutine zur Berechnung der Speicherphasendichte. Dazu folgt es einem vorzugebenden Gitter und startet an jedem Gitterpunkt eine Reihe von Teilchen mit verschiedenen Startwinkeln und Energien. Für jedes Teilchen wird dann geprüft, ob es gespeichert wurde. Für jeden Gitterpunkt und jede Energie erhält man so die Anzahl der gespeicherten Teilchen, die dort gestartet wurden. Dieses Modul arbeitet eng mit **tracking** zusammen und dient quasi als Steuerroutine.

Im Modul **mag_pa_reader** wird die magnetische Potentialkarte verwaltet, es sei denn man entschließt sich dafür, das Magnetfeld an jedem Punkt direkt zu berechnen. In diesem Fall gibt das Modul die Aufrufe an **Bfield_3D** von B. Flatt [Fla01] weiter.

Analog findet in **efield** die Verwaltung der elektrischen Potentialkarten statt.

Die mathematischen Module **math_vector** und **matrix** kapseln Funktionen zur Vektor- und Matrizenverarbeitung. In **math_vector** sind auch die Routinen zur Krümmungsvektorberechnung, Gradientenberechnung und die Interpolationsalgorithmen für Feldkarten abgelegt. Abschließend bildet **global** die unterste Instanz und enthält alle benötigten globalen Datenstrukturen, Variablen und Konstanten.

Die Quelltexte der ADIPARK-Module sind im Anhang A abgedruckt. Die Konfiguration und Anwendung von ADIPARK wird in Abschnitt 3.2.3 ausführlich demonstriert.

3.2.3 Teilchenbahnverfolgung mit ADIPARK

Hier soll nun gezeigt werden, welche Möglichkeiten die Teilchenbahnverfolgung mit ADIPARK bietet, welche Parameter das Programm erwartet und wie sich diese auswirken. Weiterhin wird der Inhalt, bzw. das Format von Parameterdateien und der erzeugten Datendateien erläutert.

Tabelle 3.1: Aufbau einer '.run'-Datei. Erläuterung der in '.run'-Dateien nötigen Start- und Konfigurationsparameter. Eine Beispieldatei ist im Anhang abgedruckt.

Parameter	Beschreibung
display	Steuert die Bildschirmausgabe: 0=aus, 1=zeilenweise, 2=ausführlich
calc-order	Schalter für Korrekturterme: 0=alle, 1=nur $E \times B$-Drift, 2=nur Krümmungs- und Gradientendrift, 3=keine
retard-onoff	Retardierungspotential: 0=aus, 1=an
dipol	Stärke der Dipolfeldes in y-Richtung in [V/m].
max-mirrors	Abbruchkriterium, maximale Anzahl von Reflektionspunkten.
max-tof-in-sec	Abbruchkriterium, maximale Flugzeit in [m/s].
e-para-min	Minimallimit für longitudinale kinetische Energie.
max-step-length	Gibt maximale Schrittweite vor.
x,y,z	Startkoordinaten, Ursprung liegt in Spektrometerzentrum.
ekin	kinetische Startenergie.
theta	Zenitwinkel zwischen Teilchenrichtung und x-Achse.
phi	Azimuthwinkel für Drehung um x-Achse.
mass	Teilchenmasse in Einheiten der Elektronenmasse.
charge	Ladung in Einheiten der Elektronenladung.

Zuerst soll hier der Programmstart an der Linux-Kommandozeile demonstriert werden, der Befehl lautet:

adipark <*Sammelname der Parameterdateien*>

Der Sammelname der Parameterdateien darf keine Dateinamenerweiterung enthalten, er dient vielmehr dazu zusammengehörige Konfigurationsdateien zu laden. ADIPARK fügt entsprechende Dateinamenerweiterungen hinzu und lädt die benötigten Dateien. Beim Programmstart wird zuerst aus Sammelname + '.pa0' die elektrische Potentialkarte eingelesen, anschließend aus Sammelname + '.pa' die magnetische. In Sammelname + '.par' werden die Parameter der magnetischen Spulenkonfiguration erwartet, in Sammelname + '.run' müssen die Start- und Konfigurationsparameter der einzelnen Simulationsdurchläufe abgelegt sein. Die durch die Simulation erhaltenen Daten werden nach dieser Methode in Sammelname + '.track<X>' gespeichert,

Tabelle 3.2: Aufbau einer '.track<X>'-Datei. Kommentierte Auflistung der in '.track<X>'-Dateien abgelegten Informationen.

Spalte	Beschreibung	Einheit
1, 2, 3	x, y, z Position	cm
4	Zyklotronradius	cm
5	elektrisches Potential	V
6	kinetische Energie	eV
7	longitudinale kinetische Energiekomponente	eV
8	transversale kinetische Energiekomponente	eV
9	magnetische Feldstärke	$Tesla$
10	Flugzeit	μs
11, 12, 13	x, y, z Komponenten der longitudinalen Geschwindigkeit	m/s
14	longitudinaler Geschwindigkeitsbetrag	m/s
15	transversaler Geschwindigkeitsbetrag	m/s
16, 17, 18	x, y, z Komp. der $E \times B$-Driftgeschwindigkeit	m/s
19, 20, 21	x, y, z Komp. der Krümmungsdriftgeschwindigkeit	m/s
22, 23, 24	x, y, z Komp. der Gradientendriftgeschwindigkeit	m/s
25	Vorzeichen der Teilchenbewegung rel. zu x-Richtung	
26	Schrittweiten-Verkleinerungsfaktor	

<X> enthält hierbei die Nummer des Simulationsdurchlaufes aus den '.run'-Dateien.

In den '.run'-Dateien sind alle Parameter der Teilchenbahnverfolgung abgelegt (s. Tab.(3.1)). Die Datei enthält Kommentarzeilen, die die einzugebenden Parameter erläutern sollen, eine Beispieldatei ist im Anhang abgedruckt. In einer '.run'-Datei werden zuerst einige numerische Parameter festgelegt, bevor dann die Parameter der zu verfolgenden Teilchen angegeben werden. Werden Parameter für mehrere Teilchen, in Zeilen untereinander, angegeben, so startet das Programm für jedes Teilchen einen separaten Simulationsdurchlauf und nummeriert die Ergebnisdateien fortlaufend durch.

Die Ergebnisse eines Simulationsdurchlaufes werden in den '.track<X>'-Dateien abgelegt. Ihre Struktur wird in Tab. (3.2) beschrieben. Zum Inhalt der '.par'-Dateien sei auf [Fla01] verwiesen. Die für das Mainzer Spektrometer verwendete '.par'-Datei ist im Anhang B.3 abgedruckt.

Eine weitere Möglichkeit der Konfiguration findet sich in der Datei 'global.h'. In ihr werden neben global notwendigen Konstanten auch die Parameter der Potentialfeldkarten festgelegt. Es handelt sich dabei im wesentlichen um die Gitterkonstante und die Koordinatenursprungsverschiebung (engl. 'offset'). Solange mit denselben Potentialkarten gearbeitet wird, ist hier keine Änderung nötig, es ist jedoch sehr wichtig, hier die zu den Potentialkarten passenden Werte einzutragen. Zu beachten ist, daß ADIPARK neu übersetzt (engl. 'compiled') werden muß, damit Änderungen wirksam werden. In einer zukünftigen Version ist das Einlesen einer Textdatei mit Parameterdaten vorgesehen.

Wenn alle Konfigurationen und Parameter eingestellt wurden, kann ADIPARK gestartet werden. Es werden dann elektrische und magnetische Potentialkarten in den Speicher geladen und das Auswahlmenü wird angezeigt. Wählt man hier *'start tracking loop'* aus, wird die Teilchenbahnverfolgung für die in der '.run'-Datei definierten Teilchen gestartet. Unabhängig von der Einstellung des *display*-Parameters wird auf dem Bildschirm ausgegeben, wenn ein Reflektionspunkt passiert wurde. Am Ende eines jeden Durchlaufes wird die Abbruchursache, z.B. Kollision mit Elektrode, Verlassen des Spektrometers oder Eintreten einer definierten Abbruchbedingung, ausgegeben.

3.2.4 Prinzipielle und numerische Grenzen

Im folgenden soll diskutiert werden, inwieweit die verwendeten numerischen Methoden sinnvoll sind und wo die prinzipiellen Grenzen der verwendeten Näherung liegen, bevor dann die Grenzen der numerischen Genauigkeit abgesteckt werden. Als prinzipielle Grenzen des Programmes ADIPARK werden hier technische und mathematische Beschränkungen betrachtet. Die technischen Beschränkungen ergeben sich zum einen aus der zur Verfügung stehenden Hardware und zum anderen aus dem

verwendeten Algorithmus. Die mathematischen Beschränkungen sind Konsequenzen der verwendeten Näherungsmethode für die Teilchenbewegung.

3.2.4.1 Diskussion der technischen Grenzen

Eine grundlegende Tatsache ist, daß sich die elektrischen und magnetischen Potentialfeldkarten den vorhandenen Speicher teilen müssen, dies sollte daher bei der Potentialkartenerstellung beachtet werden. Im wesentlichen wird die Gitterkonstante, mit der die Potentialwerte abgelegt werden, den Ausschlag geben. Eine um Faktor zwei verkleinerte Gitterkonstante einer dreidimensionalen Potentialkarte erfordert um den Faktor 2^3 mehr Speicherplatz. Die verwendeten dreidimensionalen elektrischen Feldkarten haben daher eine Gitterkonstante von $g = 4\ mm$, was einem Speicherplatzbedarf von etwa 200 MB entspricht, wenn durch Nutzung von Symmetrien nur ein Viertel des Gesamtvolumens gespeichert werden muß.

3.2.4.2 Diskussion der mathematischen Grenzen

Der verwendete Algorithmus basiert auf der Bewegung des Führungszentrums, d.h. die Zyklotronbewegung wird vernachlässigt. ADIPARK kann daher keine Zyklotronbewegung berechnen, es ist jedoch möglich, an jedem Punkt der Teilchenbahn den Betrag des Zyklotronradius, unter der Annahme einer kreisförmigen Zyklotronbewegung, anzugeben. Die Verformung der Zyklotronbewegung durch stark inhomogene elektrische Felder, wird von ADIPARK z. Z. weder berechnet noch erkannt (in späteren Versionen durchaus möglich). In solchen Fällen muß auf andere Simulationsprogramme zurückgegriffen werden, z.B. SIMION 3D.

Die in der adiabatischen Näherung verwendeten Driftgleichungen (s. Gl. (2.19) bis (2.21)) sind allesamt in nichtrelativistischer Näherung formuliert. Bewegt sich ein Teilchen in Feldbereiche, die zu relativistischen Driftgeschwindigkeiten führen, kann ADIPARK keine verläßliche Aussage mehr über die Teilchenbahn machen. Die Gültigkeit der nichtrelativistischen Driftterme ist dann nicht mehr gegeben. Feldbereiche mit dieser Wirkung existieren in den Elektrodenzwischenräumen, besonders beim Einsatz von nicht zylindrischen Zusatzelektroden. Man kann aber davon ausgehen, daß Teilchen, die in diese Bereiche fliegen, so starke Kräfte erfahren, daß sie mit einer Elektrode kollidieren werden.

Die größten prinzipiellen Einschränkungen sind Konsequenzen der adiabatischen Näherung für die Teilchenbewegung. Diese Näherung, wie in Abschnitt 2.2.2 behandelt, steht und fällt mit der Homogenität des Magnetfeldes und der Größe der kinetischen Energie. Für eine gute Näherung ist es wichtig, daß die relative Magnetfeldänderung $\frac{\Delta B}{B}$ pro Zyklotronumlauf sehr klein ist. Die charakterische Länge L ist als Strecke zu verstehen, auf der die Magnetfeldänderung vernachlässigbar klein ist.

Man kann eine Abschätzung für die Gültigkeit der adiabatischen Näherung formulieren, indem man davon ausgeht, daß der Zyklotronradius ϱ des Teilchens klein sein muß gegenüber L [Del65]. Es müssen dann folgende Voraussetzungen gelten:

$$\frac{\varrho}{L} \ll 1 \quad \text{mit} \quad \varrho = \frac{mcv_\perp}{qB} = \frac{c}{qB}\sqrt{2mE_\perp} \tag{3.3}$$

Unter Verwendung dieser Voraussetzungen kann man schreiben.

$$qBL \gg qB\varrho = mcv_\perp \tag{3.4}$$

Nach einigen Umformungen läßt sich dies in Abhängigkeit der transversalen kinetischen Energie $E_\perp$ schreiben.

$$BL \gg \frac{1}{q}\sqrt{2mE_\perp c^2} \tag{3.5}$$

Für Elektronen und Protonen kann man daraus leicht eine Grenzbedingung in gebräuchlichen Dimensionen aufstellen [Del65].

$$B_{[Gauss]}L_{[cm]} \gg \begin{cases} 3\ \sqrt{E_{\perp[eV]}} & \text{, für Elektronen} \\ 135\ \sqrt{E_{\perp[eV]}} & \text{, für Protonen} \end{cases} \tag{3.6}$$

Hierbei soll noch angemerkt werden, daß sich aufgrund des Wurzel Masse zu Ladungsverhältnisses für α–Teilchen ebenfalls der für Protonen geltende Zusammenhang ergibt. Interessant und wichtig ist nun der Faktor $\sqrt{\frac{m_P}{m_e}} \approx \sqrt{2000} \approx 45$ zwischen leichten Teilchen, wie Elektronen, und schweren Teilchen, wie Protonen und α–Teilchen. Anhand dieses Faktors kann man eine Abschätzung der adiabatischen Näherung für Ionen aufstellen. Soll ein Ion, einem Elektron vergleichbar, in guter adiabatischer Näherung verfolgt werden, muß entweder das Magnetfeld um Faktor 45 ansteigen oder die kinetische Energie um den Faktor 2000 kleiner werden. Diese Betrachtung zeigt, daß es prinzipiell möglich ist Ionen zu verfolgen, allerdings ist man in adiabatischer Näherung auf sehr niederenergetische Ionen beschränkt oder muß in sehr starken Magnetfeldern arbeiten. Aufgrund der Proportionalität des Zyklotronradius zur Wurzel der Teilchenmasse bei identischer kinetischer Energie, ist dieser für Ionen um etwa $\sqrt{\frac{m_P}{m_e}} \approx 45$ größer. Daher ist es problematisch in einem Volumen der Größe des Mainzer Spektrometers Ionen zu verfolgen, wenn diese nicht sehr geringe transversale kinetische Energie besitzen. Die Ionen mit hoher Energie werden, aufgrund des großen Zyklotronradius, gegen eine Elektrode fliegen. Dies zeigt, daß es sehr viel schwerer ist, Ionen im Mainzer Spektrometer zu speichern, was aus den Gründen, die in Abschnitt 4.5 besprochen werden, von Vorteil ist.

Um den Gültigkeitsbereich der adiabatischen Näherung festzulegen, werden Vergleichssimulationen mit SIMION 3D durchgeführt. Weichen die ADIPARK–Ergebnisse von den SIMION–Ergebnissen für dasselbe Teilchen ab, so reicht die adiabatische Näherung erster Ordnung nicht mehr aus, um die Flugbahn zu beschreiben.

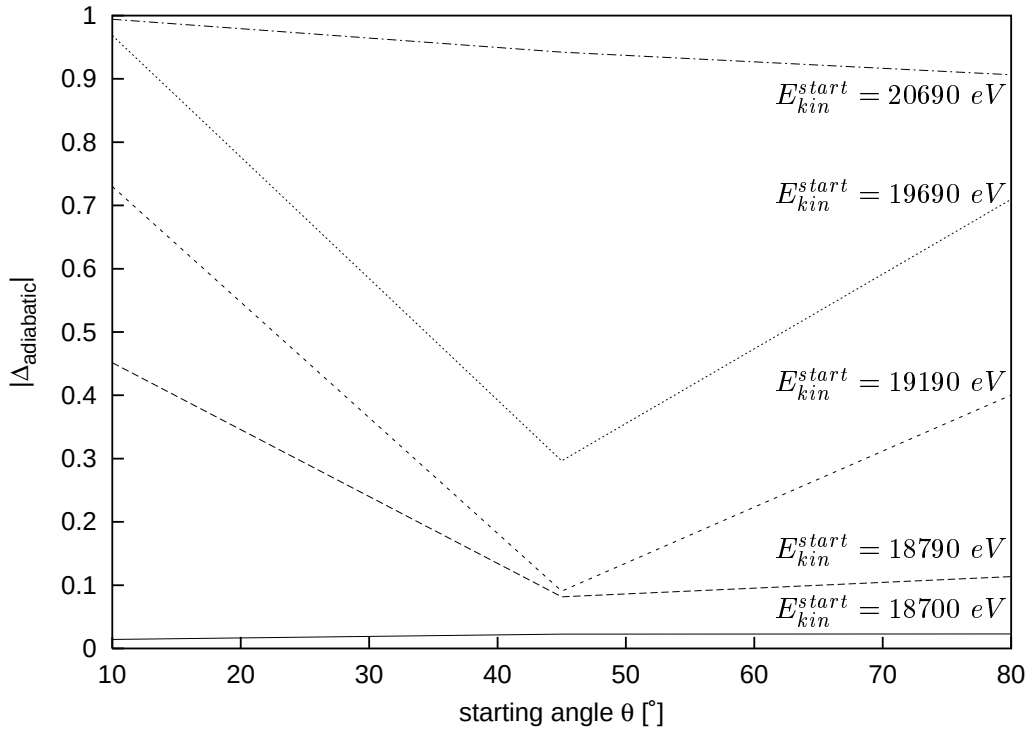

Abbildung 3.4: Adiabasieabweichung in SIMION 3D. Dargestellt ist der Wert der Adiabasieabweichung $\Delta_{adiabatic}$ als Funktion des Startwinkels für verschiedene Startenergien $E_{kin}^{start} = 18700\ eV,\ 18790\ eV,\ 19190\ eV,\ 19690\ eV$ und $20690\ eV$. Größere Werte bedeuten größere Abweichungen in der adiabatischen Energietransformation.

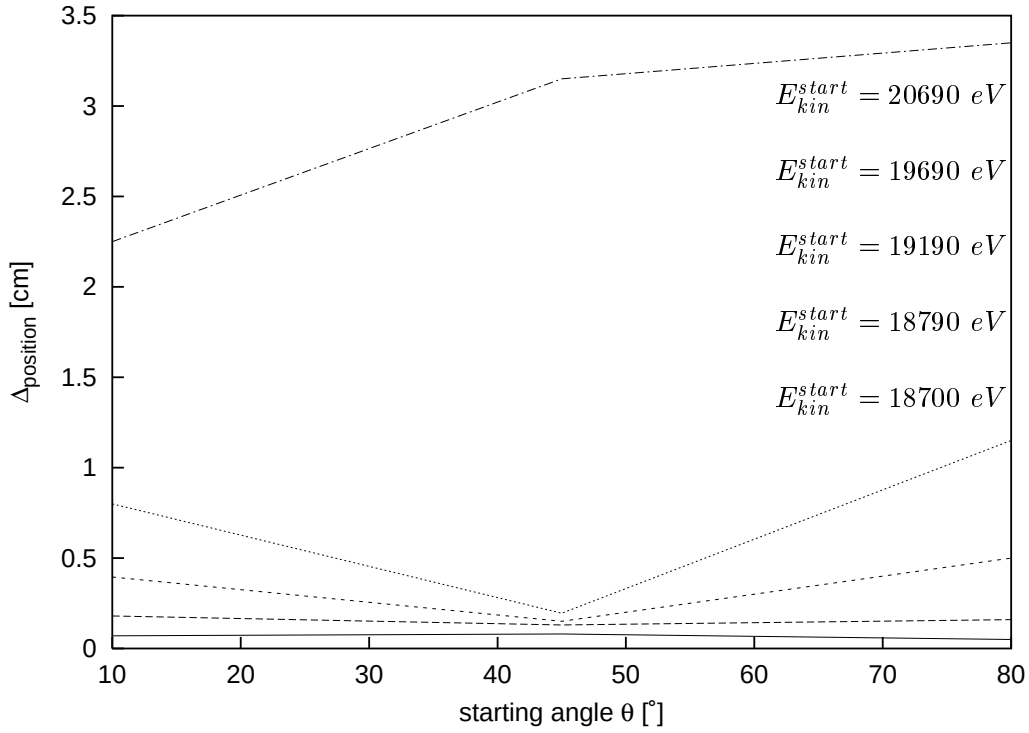

Abbildung 3.5: Flugbahnabweichung durch adiabatische Näherung. Dieses Bild zeigt die Flugbahndifferenz $\Delta_{position}$ als Funktion des Startwinkels zwischen berechneten Bahnen von SIMION 3D und ADIPARK. Es wurde eine Serie von Elektronen mit Startenergien von $E_{kin}^{start} = 18700\ eV,\ 18790\ eV,\ 19190\ eV,\ 19690\ eV$ und $20690\ eV$ berechnet. Es ist eine Verschlechterung der adiabatischen Näherung bei großen Überschußenergien zu sehen.

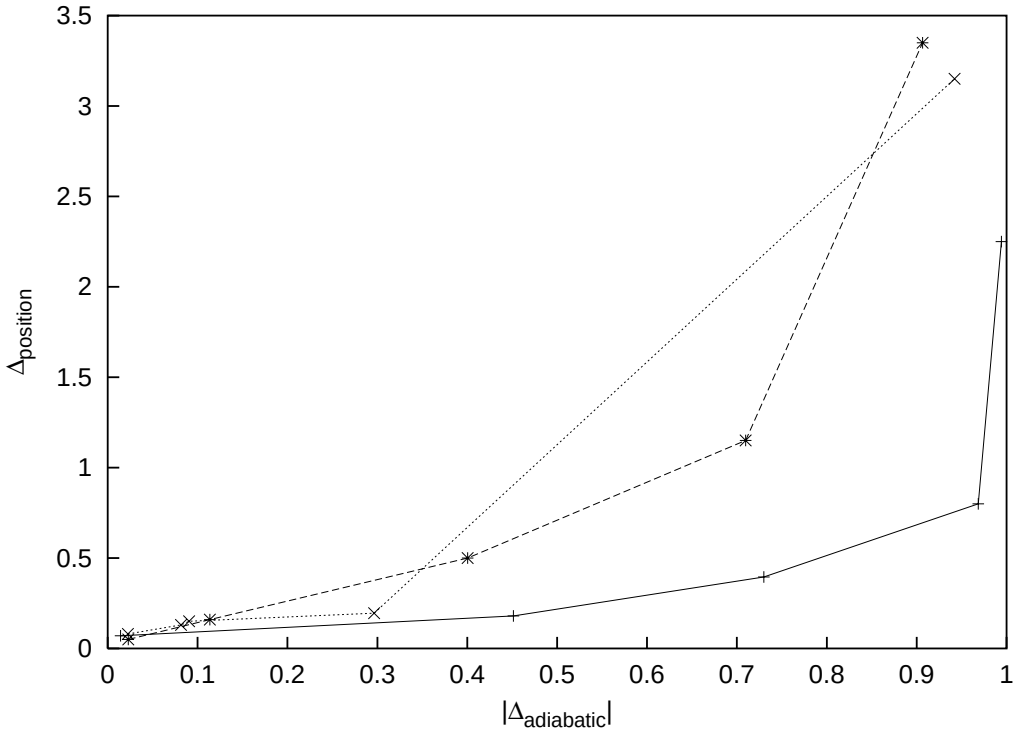

Abbildung 3.6: Vergleich von Flugbahn- und Adiabasieabweichung. Dieses Bild zeigt die Flugbahndifferenz $\Delta_{position}$ zwischen SIMION 3D und ADIPARK als Funktion des Adiabasieabweichung $\Delta_{adiabatic}$ bei verschiedenen Startwinkeln mit ansteigender Überschußenergie. Es wurden Elektronen mit Startenergien von $E_{kin}^{start} = 18700\,eV$, $18790\,eV$, $19190\,eV$, $19690\,eV$ und $20690\,eV$ berechnet. Es sind die Startwinkel $\theta = 10°$ (durchgezogen), $\theta = 45°$ (gepunktet) und $\theta = 80°$ (gestrichelt) dargestellt.

Wenn bei der adiabatischen Energietransformation eine Energieverletzung auftritt, wird von Adiabasieverletzung gesprochen. Für die SIMION–Ergebnisse läßt sich die Adiabasieverletzung durch die Größe $\Delta_{adiabatic}$ beschreiben. Diese Größe gibt in nullter Ordnung an, wie weit das magnetische Moment als adiabatische Invariante (Gl. (2.22)) vom Anfangswert abweicht. Der in ADIPARK verwendete Algorithmus basiert auf der Energietransformation durch diese adiabatische Invariante, so daß hier die Adiabasie per Definition erhalten ist.

$$\Delta_{adiabatic} = \frac{P_{\perp i}^2}{B_i} \cdot \frac{B_f}{P_{\perp f}^2} - 1 \tag{3.7}$$

Im adiabatischen Idealfall ist $\frac{P_\perp^2}{B} = const.$ und man erwartet, daß sich $\Delta_{adiabatic} = 0$ ergibt. Sollte nun $\Delta_{adiabatic} \neq 0$ sein, gibt es Verluste bei der Energietransformation von transversaler zu longitudinaler Komponente der Bewegung.

Bei ansteigendem $\Delta_{adiabatic}$ erwartet man eine Abweichung der SIMION–Ergebnisse von der ADIPARK–Flugbahn, die durch die adiabatische Näherung erster Ordnung berechnet wurde. Man muß dann davon ausgehen, daß die adiabatische Näherung

erster Ordnung nicht mehr ausreicht, um das Führungszentrum der wirklichen Flugbahn gut zu beschreiben.

Für den Vergleich wurden fünfzehn Simulationen mit gleichen Startparametern in SIMION 3D und ADIPARK durchgeführt. Es werden Elektronen mit $E_{kin} = 10\ eV$, 100 eV, 500 eV, 1000 eV und 2000 eV Überschußenergie im Spektrometereingang bei $x = -201.2\ cm$ ($y = 0.5\ cm,\ z = 0\ cm$) gestartet. Überschußenergie bezeichnet den Energieanteil, der über dem im Spektrometer verwendeten Analysierpotential $U_0 = 18690\ eV$ liegt. Für jede Startenergie werden drei Simulationsrechnungen mit unterschiedlichen Startwinkeln (10°, 45° und 80°) in beiden Programmen durchgeführt.

In Abbildung 3.4 sind die Adiabasieabweichungen der SIMION–Berechnungen über den entsprechenden Startwinkel für die verschiedenen Startenergien aufgetragen. Die Abbildung 3.5 zeigt analog dazu die Flugbahnabweichung der SIMION-Bahn zu den ADIPARK–Berechnungen. Flugbahnabweichungen von $\Delta_{position} < 5\ mm$ werden als noch vertretbare Abweichung angenommen. Betrachtet man diese beiden Darstellungen, fällt sofort auf, daß eine Überschußenergie von 2000 eV die Adiabasieerhaltung nicht mehr erfüllt und auch die Flugbahnabweichung sehr groß ist, man erhält dann mit ADIPARK kein vertrauenswürdiges Ergebnis mehr. Bei einer Überschußenergie von 1000 eV ist die Flugbahnabweichung bei mittlerem Startwinkel von $\theta = 45°$ noch vertretbar, nur die sehr großen und sehr kleinen Startwinkel weichen deutlich ab. Ein vergleichbares Verhalten ist analog dazu auch bei der Adiabasieabweichung zu erkennen. Bei Überschußenergien $\leq 500\ eV$ werden die Flugbahnabweichungen sehr gering und auch die Adiabasieabweichungen nähern sich immer kleineren Werten an. Für die Adiabasieabweichung kann man festhalten, daß sich relativ niedrige Flugbahnabweichungen ergeben, wenn man $\Delta_{adiabatic} < 0.5$ fordert. Darüber hinaus kann man nicht mehr davon ausgehen, daß die adiabatische Näherung erster Ordnung die Flugbahn erfolgreich beschreiben kann. Abbildung 3.6 zeigt die Flugbahnabweichung gegen die Adiabasieabweichung aufgetragen. In dieser Darstellung kann man erkennen, daß die großen Flugbahnabweichungen mit einem Anstieg des Adiabasiefaktors korrespondieren.

Abschließend kann man festhalten, daß die adiabatische Näherung erster Ordnung von ADIPARK die Flugbahn bei Überschußenergien von $E_{kin} \leq 100\ eV$ sehr gut beschreibt. Bei Überschußenergien von $500\ eV \leq E_{kin} \leq 1000\ eV$ wird die Genauigkeit bei großen und kleinen Startwinkel zunehmend schlechter, mittlere Startwinkel werden noch gut beschrieben. Teilchen mit Überschußenergien $E_{kin} > 1000\ eV$ können nicht mehr adiabatisch beschrieben werden. Dies ist der Rahmen, in dem ADIPARK als verläßliches Simulationsprogramm eingesetzt werden kann. Weitere Einschränkungen ergeben sich noch aus der numerischen Genauigkeit des verwendeten Algorithmus, dieses Problem wird im nächsten Abschnitt ausführlich behandelt.

3.2.4.3 Tests der numerischen Genauigkeit

Es stellt sich die Frage, wie stark die numerische Genauigkeit von Variationen in Eingabeparametern wie Schrittweite und Minimalenergie abhängt. Darüber hinaus ist es noch wichtig zu wissen, welche Abweichungen sich ergeben, wenn die Gitterkonstanten der magnetischen oder elektrischen Potentialkarten verändert werden. Die nun folgenden Testläufe wurden mit den Potentialkarten des aktuellen Mainzer Spektrometers erstellt.

Für die Untersuchung der Abhängigkeit von der eingestellten Schrittweite wurde ein Elektron magnetisch gespeichert. Das Teilchen startet im Mainzer Spektrometer mit einem Radius von 20 cm und einem Abstand von 30 cm links der Analysierebene. Daraus ergeben sich die Startkoordinaten $(x, y, z) = (-30, 20, 0)$. Die kinetischen Startenergie beträgt $E_{\text{kin}} = 1000\ eV$ und ein Startwinkel von $\theta = 50°$ relativ zur Spektrometerachse wurde vorgegeben. Dieses Elektron ist im Bereich von $-86,1\ cm \leq x \leq 86,1\ cm$ im zentralen Bereich des Spektrometers gespeichert. Man erwartet, daß die x-Koordinate des Reflektionspunktes konstant bleibt; ist dies nicht der Fall, kommt es zu Fehlern in der Gesamtenergie. Um die numerischen Ungenauigkeiten möglichst deutlich sehen zu können, fliegt das Elektron 1000 Speicherzyklen lang, d.h. für das Elektron vergehen etwa 362 μs und es wird 2000 mal reflektiert. Die gleiche Simulation wurde für Schrittweiten von $\Delta s = 10\ mm$, $\Delta s = 0,5\ mm$, $\Delta s = 1\ mm$, $\Delta s = 0,5\ mm$ und $\Delta s = 0,1\ mm$ durchgeführt.

In Abb. (3.7) kann man sehr gut erkennen, wie stark die Änderung des Reflektionspunktes einer Teilchenbahn von der vorgegebenen Schrittweite abhängt. Diese Tatsache ist auch einzusehen, da mit feinerer Schrittweitenvorgabe der Runge-Kutta-Algorithmus zu Berechnung der Teilchenbahn besser arbeitet. Um vertrauenswürdige Daten zu erzeugen, sollte man mit einer Schrittweite von $\Delta s \leq 1\ mm$ arbeiten. Natürlich kann man auch auf gröbere Schrittweiten zurückgreifen, um die Berechnungszeit zu verkürzen. Man sollte sich dann allerdings auf kurze Flugzeiten oder wenige Speicherzyklen beschränken, um nicht zu große Ungenauigkeiten zu erzeugen.

Eine Konsequenz der feineren Schrittweitenvorgabe ist natürlich eine höhere Berechnungszeit. In Abb. (3.8) ist die Berechnungszeit als Funktion der Schrittweiteneinstellung aufgetragen. Man sieht hier sehr deutlich, daß eine feinere Schrittweite eine höhere Berechnungszeit zur Folge hat, d.h. eine um den Faktor zehn feinere Schrittweitenvorgabe erfordert eine um den Faktor zehn höhere Berechnungszeit.

Aufgrund dieser Ergebnisse werden alle nun folgenden Simulationen mit einer Schrittweitenvorgabe von $\Delta s = 0,1\ mm$ durchgeführt.

Neben der Schrittweite kann sich auch noch die Einstellung der minimalen longitudinalen kinetischen Energie, auf die numerische Genauigkeit auswirken. Bei auftreten eines Reflektionspunktes legt die minimale kinetische Energie (s. Abs. 3.2.2) das Energielimit fest, unter das die kinetische Energie des Teilchens absinken muß, damit das Programm die Reflektion ausführt. Auf diese Weise sollen die Reflektionspunkte möglichst gut approximiert werden. Um die Auswirkungen dieses Parameters auf die Genauigkeit zu testen, wird die oben durchgeführte Simulation für verschiedene Minimalenergien wiederholt. In Abb. (3.9) ist die Minimalenergie gegen die

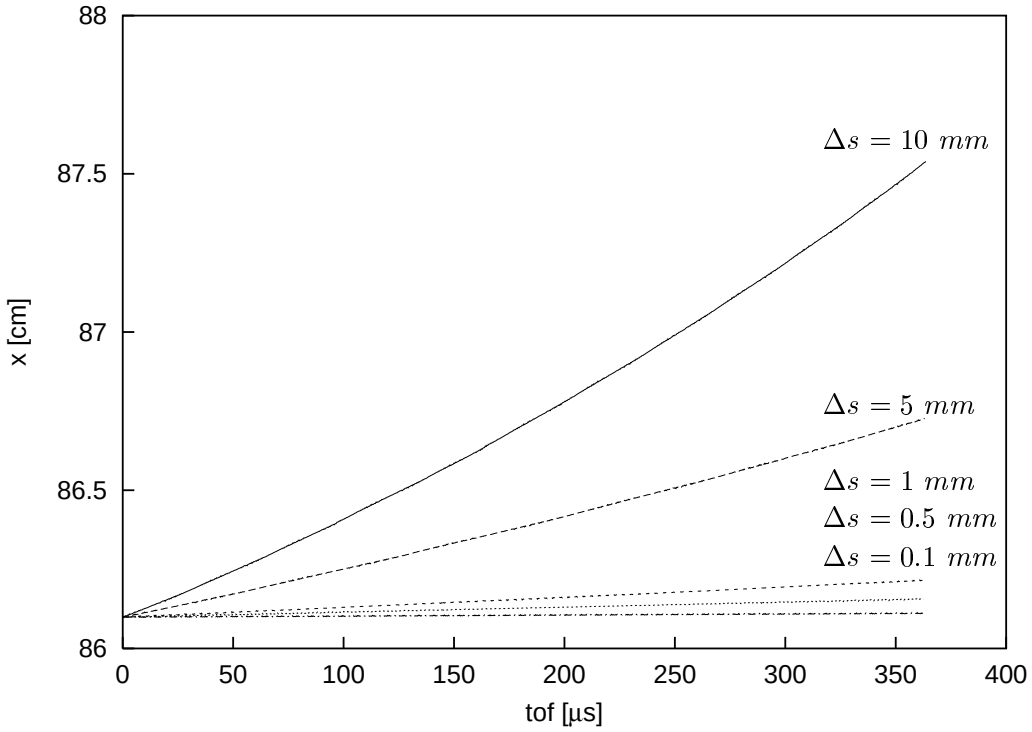

Abbildung 3.7: Schrittweitenabhängigkeit des Reflektionspunktes. Dargestellt ist der detektorseitige Reflektionspunkt des gespeicherten Teilchens als Funktion der Flugzeit. Für gröbere Schrittweiteneinstellungen ist deutlich zu sehen, wie sehr der Reflektionspunkt zu größeren Werten hin abweicht.

Position des detektorseitigen Reflektionspunktes des gespeicherten Teilchens aufgetragen. Es sind die Ergebnisse für die Minimalenergien $E_{||}^{min} = 1\,eV$, $E_{||}^{min} = 0,5\,eV$, $E_{||}^{min} = 0,2\,eV$, $E_{||}^{min} = 100\,meV$, $E_{||}^{min} = 10\,meV$ und $E_{||}^{min} = 1\,meV$ dargestellt. Man sieht hier, daß für $E_{||}^{min} \leq 0,1\,eV$ die Position des Reflektionspunktes stabil bleibt.

Ein weiterer Effekt, der in dieser Darstellung auffällt, ist der Einfluß des Schrittweitenverkleinerungsalgorithmus. Bei den drei oberen Energien reicht eine Halbierung der Schrittweite aus, um unter das Limit der minimalen longitudinalen kinetischen Energie zu kommen, daher die Streifenform. Bei den niedrigeren Minimalenergieeinstellungen sind mehrere und verschiedene noch feinere Verkleinerungsfaktoren nötig, was dazu führt, daß hier ein Bereich mit gleichmäßiger Verteilung entsteht. Die Verbreiterung in x ist die Konsequenz der numerischen Ungenauigkeit, die schon in Abb. (3.7) als Anstieg der Geraden zu sehen war.

Es bleibt festzuhalten, daß alle nun folgenden Berechnungen mit einer minimalen Longitudinalenergie von $E_{||}^{min} = 1\,meV$ durchgeführt werden.

Ein abschließender Test für die numerische Genauigkeit ist der Vergleich von Teilchenbahndaten, die unter Verwendung verschieden feiner Potentialkarten erzeugt wurden. Für diesen Test wurde ein Elektron vom selben Startpunkt am Spektrome-

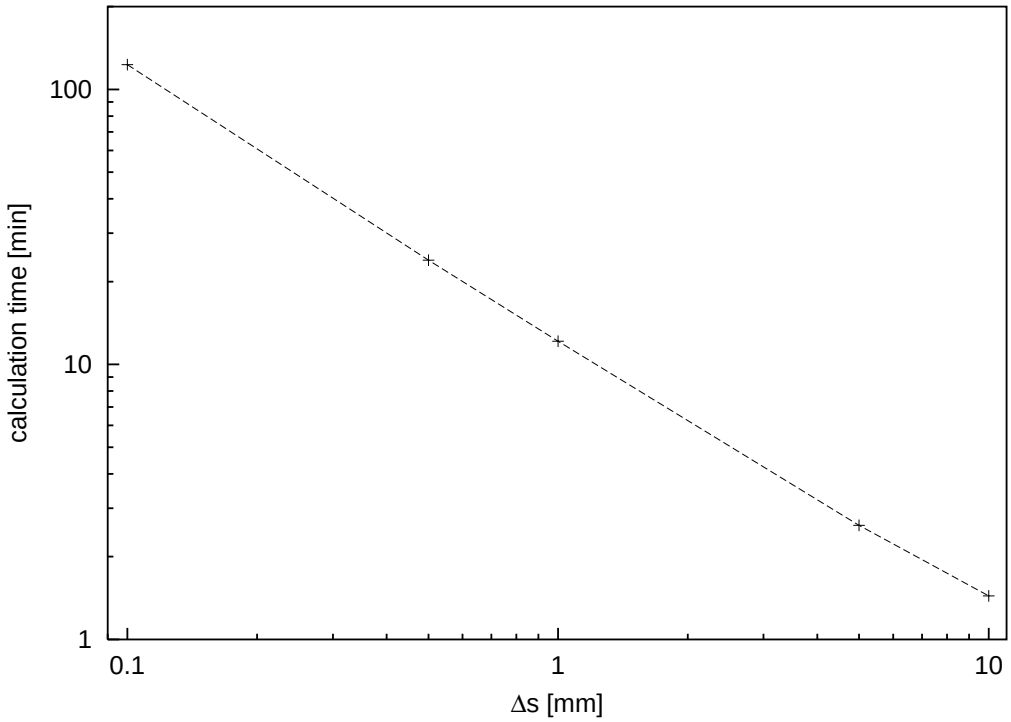

Abbildung 3.8: Schrittweitenabhängigkeit der Berechnungszeit. Die Berechnungszeit als Funktion der Schrittweiteneinstellung, gemessen auf einem Rechner mit AMD Athlon Prozessor mit 800 MHz Taktfrequenz. Für jeden Punkt wurde ein Teilchen 1000 Speicherzyklen (2000 Reflektionen) lang verfolgt, dies entspricht einer Flugzeit von je 362 μs. Wie zu erwarten verzehnfacht sich die Berechnungszeit bei einer um Faktor zehn kleineren Schrittweite.

tereingang durch das Spektrometer verfolgt, bis es den Spektrometerausgang erreicht hat. Dies wurde dann für verschieden grobe magnetische und elektrische Potentialkarten durchgeführt. Im Idealfall sollten die Elektronen am Spektrometerausgang wieder den selben radialen Abstand zur Spektrometerachse haben wie am Startpunkt. Beim Übergang von einer elektrischen Potentialkarte mit $g = 4\ mm$ Gitterkonstante zu einer mit $g = 2\ mm$ ergab sich eine radiale Abweichung am Spektrometerausgang von $\Delta R_{el} = 7,9\ nm$. Ein Wechsel der magnetischen Potentialkarte, deren Gitterabstände dem elektrischen Fall entsprechen, ergibt eine radiale Abweichung am Spektrometerausgang von $\Delta R_{mag} = 2,3\ \mu m$. Hier zeigt sich, daß sich der Gitterabstand der magnetischen Potentialkarte mit einem Faktor von etwa 300 stärker auf die Flugbahn auswirkt als der Gitterabstand der elektrischen Potentialkarte. Da ADIPARK für die Bahnverfolgung primär den Magnetfeldlinien folgt, ist klar, daß sich die Änderungen in den magnetischen Potentialkarten am stärksten auf die Flugbahn auswirken. Jede Ungenauigkeit durch die Interpolation von groben magnetischen Potentialkarten wirkt sich so direkt auf die Flugbahn des simulierten Teilchens aus. Eine hohe Genauigkeit, d.h. kleine Gitterkonstante, in den magnetischen Potentialkarten ist daher sehr wichtig.

Es gilt hier immer einen Kompromiß zwischen Speicherplatzverbrauch und Gitterkon-

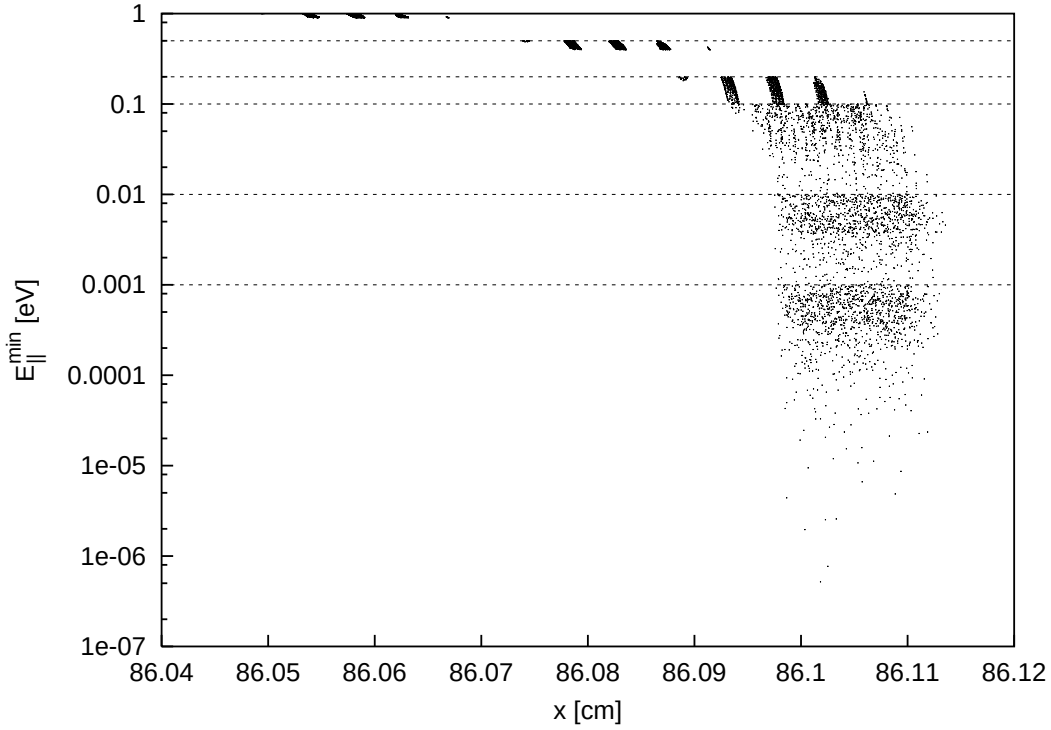

Abbildung 3.9: Energieabhängigkeit des Reflektionspunktes. Minimale longitudinale kinetische Energie am detektorseitigen Reflektionspunkt des gespeicherten Teilchens. Der Reflektionspunkt wird mit abnehmendem Energielimit zunehmend besser approximiert, für $E_{\parallel}^{min} \leq 0,1\ eV$ bleibt der Reflektionspunkt ausreichend stabil. Jeder Punkt beschreibt die Energie eines Reflektionspunktes, es wurden je Energielimiteinstellung 2000 Reflektionen berechnet.

stante zu finden, um möglichst genaue Daten zu erzeugen. Die zweidimensionalen zylindersymmetrischen Potentialkarten verursachen keine Speicherplatzprobleme, erst die dreidimensionalen Potentialkarten sind problematisch. Für das Mainzer Spektrometer benötigt eine dreidimensionale elektrische Potentialkarte mit $g = 4\ mm$ schon etwa 100 MB an Speicherplatz, wenn Spiegelsymmetrien ausgenutzt werden können und nur 1/8 des Gesamtvolumens enthalten sein muß. Soll dieselbe Potentialkarte mit halbierter Gitterkonstante ($g = 2\ mm$) abgespeichert werden, wird $2^3 = 8$ mal soviel Speicher benötigt. Fallen Spiegelsymmetrien (z.B. beim Einsetzen von Zusatzelektroden) weg, vergrößert sich der Speicherbedarf nochmals.

Im folgenden werden magnetische Potentialkarten mit $g = 2\ mm$ und elektrische Potentialkarten mit $g = 4\ mm$ verwendet. Die elektrischen Potentialkarten sind gröber, da SIMION 3D an sein Speicherverwaltungslimit stößt, wenn man versucht, feinere dreidimensionale Potentialkarten zu erzeugen oder zu verwenden. Im allgemeinen ist es ausreichend, mit zweidimensionalen zylindersymmetrischen magnetischen Potentialkarten zu arbeiten, solange keine gekrümmten Magnetfelder gebraucht werden. Daher stößt man hierbei auch nicht so schnell an die Speichergrenze.

In Tabelle 3.3 sind die bei Berechnungen mit dem Mainzer Spektrometer verwendeten Parameter aufgelistet. Man muß hier aus Speicherplatzgründen anmerken, daß

Tabelle 3.3: Eingabeparameter für numerische Genauigkeit. Hier sind nochmals alle Eingabeparameter und Faktoren zusammengefaßt, die die numerische Genauigkeit positiv beeinflussen. Es ist natürlich immer möglich auf Kosten der Genauigkeit die Berechnungszeit zu verkürzen, indem gröbere Parameter verwendet werden.

Parameter	Wert
maximale Schrittweite	$\Delta s \leq 1\ mm$
minimale longitudinale kinetische Energie	$E_{\parallel}^{min} \leq 0,1\ eV$
Gitterkonstante für magnetische Potentialkarten	$g = 2\ mm$
Gitterkonstante für elektrische Potentialkarten	$g = 4\ mm$

bei Berechnungen mit größeren Spektrometern, wie sie bei KATRIN zum Einsatz kommen werden, zwangsläufig gröbere Gitterkonstanten zu wählen sind.

3.3 Vergleich von ADIPARK mit SIMION 3D

Abschließend sollen in diesem Abschnitt die Simulationsprogramme ADIPARK und SIMION 3D miteinander verglichen werden. Der Vergleich soll anhand der Flugbahndaten, der Abbildungseigenschaft und der Energieerhaltung zeigen, wie groß die Zuverlässigkeit von ADIPARK im Rahmen seiner numerischen und prinzipiellen Grenzen ist und wie gut die Ergebnisse mit denen von SIMION 3D korrespondieren.

Den Anfang soll ein einfacher Vergleich der Flugbahndaten eines Elektrons, jeweils von SIMION und ADIPARK berechnet, machen. Das Elektron startet im Spektrometereingang bei $x = -201.2\ cm$ mit einer kinetischen Energie von $E_{kin} = 18700\ eV$ und einem Startwinkel von $\theta = 45°$ relativ zur führenden Magnetfeldlinie. In Abbildung 3.10 ist die Übereinstimmung der Ergebnisse der beiden Simulationsprogramme sehr schön zu sehen. Die rote Linie stellt die durch SIMION berechnete Flugbahn dar, hier ist auch die Zyklotronbewegung sichtbar. Von ADIPARK wird nur das Führungszentrum der Zyklotronbewegung und damit der Teilchenbewegung berechnet. Die Ergebnisse der ADIPARK–Berechnung sind an der grünen Linie im Zentrum der von SIMION berechneten Zyklotronbewegung zu erkennen. Durch die Darstellung der Projektion der Teilchenbahn auf die x-z-Ebene, sind die Auswirkungen der Driftbewegung zu erkennen. Die Abbildung 3.11 zeigt die Teilchenbahn aus Abbildung 3.10 in der x-z-Darstellung. Hier ist wieder das von ADIPARK berechnete Führungszentrum und die Zyklotronbewegung der SIMION–Berechnung zu sehen. Die gestrichelte Linie bezeichnet die Ausdehnung des von ADIPARK berechneten mittleren Zyklotronradius. Zu sehen ist eine Abweichung im Bereich des niedrigen Magnetfeldes im Spektrometerzentrum, die darauf hinweist, daß hier die adiabatische Näherung an ihre Grenze stößt. In den Bereichen des starken Magnetfeldes ist die Übereinstimmung sehr gut. Die Abweichung des von ADIPARK berechne-

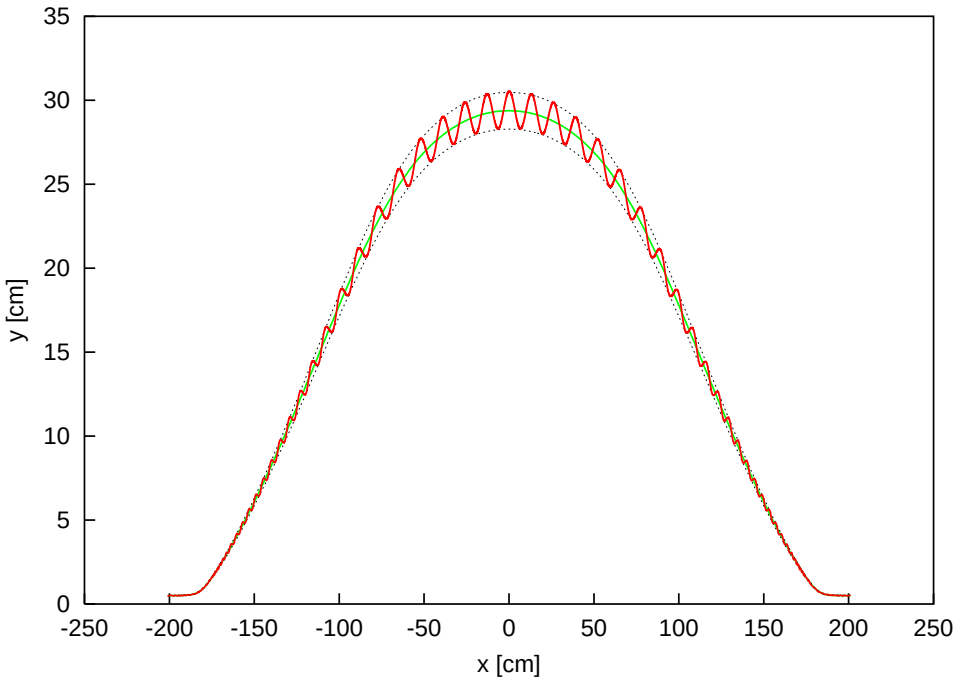

Abbildung 3.10: Vergleich einer Elektronentrajektorie in x-y-Projektion. Projektion der Trajektorie eines Elektrons ($E_{kin} = 18700\ eV$, $\theta = 45°$) auf die x-y-Ebene, das im Spektrometereingang ($x = -201.2\ cm$) startet und bis zum Spektrometerausgang ($x = -201.2\ cm$) verfolgt wird. Das Analysierpotential beträgt $U_0 = 18690\ V$. In rot ist das Ergebnis der SIMION–Berechnung mit Zyklotronbewegung zu sehen. Im Zentrum der Zykklotronbewegung ist das durch ADIPARK berechnete Führungszentrum (grün) der Elektronenbewegung zu sehen. Die gestrichelten Linien zeigen den in ADIPARK berechneten Zyklotronradius.

ten Zyklotronradius von der SIMION-Bahn ist zu verstehen, da es sich um einen mittleren Zyklotronradius handelt. Die SIMION-Bahn zeigt in diesem Bereich starke Verformungen der Zyklotronbahn. Die adiabatische Näherung erster Ordnung der Bewegung des Führungszentrums der Teilchenbahn beschreibt die wirkliche Teilchenbewegung der numerisch integrierten Bewegungsgleichung relativ gut. Man muß aber einschränken, daß dies nur im Rahmen der prinzipiellen und numerischen Grenzen von ADIPARK gilt.

Als Nächstes soll an mehreren Elektronenbahnen genauer überprüft werden, ob beide Programme identische Ergebnisse im Hinblick auf die Abbildung der simulierten Teilchen liefern. Abbildungserhaltung soll in diesem Zusammenhang bedeuten, daß geladene Teilchen die von Spektrometereingang bis Spektrometerausgang verfolgt werden, wieder mit dem ursprünglich vorhandenen Radius ihr Ziel erreichen. Neben dem Radius ist es auch wichtig, die azimutale Verschiebung durch die Driftkorrekturen zu vergleichen. Für diesen Vergleich werden Elektronen, die sich gut adiabatisch beschreiben lassen, simuliert. Die Elektronen starten mit $E_{kin} = 18790\ eV$ und $\theta = 45°$ relativ zur Magnetfeldlinie im Spektrometereingang bei $x = -201.2\ cm$ und radialen Abständen von $y = 0.0\ cm,\ 0.1\ cm,\ \ldots\ , 0.7\ cm$, was dem Quellradius der Tritiumquelle in Mainz entspricht. In Abbildung 3.12 sind nun die Ergebnisse die-

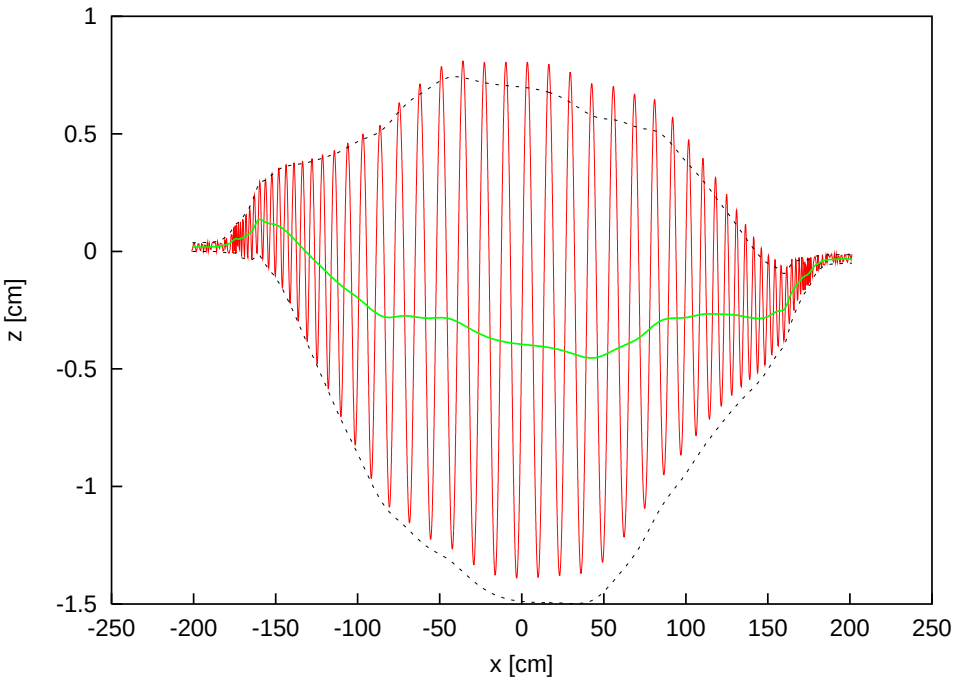

Abbildung 3.11: Vergleich einer Elektronentrajektorie in x-z-Projektion. Analog zu Abbildung 3.10 ist hier die zugehörige Projektion der Flugbahn auf die x-z-Ebene zu sehen. In dieser Darstellung ist die mehr oder weniger gute Übereinstimmung der Driftbewegung sehr deutlich zu erkennen.

ses Vergleichs zu sehen. Aufgetragen ist die z-y-Ebene im Spektrometerausgang bei $x = 201.2\ cm$. Die Kreise markieren die Ergebnisse der SIMION-Berechnungen, analog stehen die Kreuze für die ADIPARK-Ergebnisse. Die eingezeichneten Fehlerbalken an den ADIPARK-Datenpunkten sind durch den Zyklotronradius gegeben und sollen die Ausdehnung der Zyklotronbewegung beschreiben. Da ADIPARK nur das Führungszentrum der Zyklotronbewegung berechnet, sind die Fehlerbalken wichtig, um die Daten vergleichen zu können. Die Ergebnisse von SIMION, die die Zyklotronbewegung beinhalten, sollten allesamt innerhalb dieser Fehlerbalken liegen, was gut erfüllt ist. Als Ergebnis kann man hier festhalten, daß in beiden Programmen die Abbildungsradien mit azimutaler Verschiebung im Bereich eines Zyklotronradius übereinstimmen.

Als abschließender Vergleich soll die Erhaltung der Gesamtenergie eines simulierten Elektrons überprüft werden. Mit den Simulationsprogrammen sollen gespeicherte Teilchen über viele Speicherzyklen verfolgt werden, d.h. während diesen Langzeitverfolgungen muß die Gesamtenergie erhalten sein. Um dies zu überprüfen wird ein Elektron im Spektrometer durch zwei magnetische Spiegel so gespeichert, daß es nach jeder Reflektion das gesamte Spektrometer durchquert, bevor es den nächsten magnetischen Spiegel erreicht. Das verfolgte Elektron startet $x = -100\ cm$ von der Analysierebene entfernt in einem radialen Abstand von $y = 20\ cm$ mit $E_{kin} = 100\ eV$ und einem Winkel von $\theta = 35°$ relativ zur führenden Magnetfeldlinie. Die beiden magnetischen Spiegel liegen für dieses Elektron bei $x = \pm 180\ cm$. Das Elektron wird verfolgt, bis es 15 mal reflektiert wurde, was etwa einer Flugzeit von $tof = 10\ \mu s$

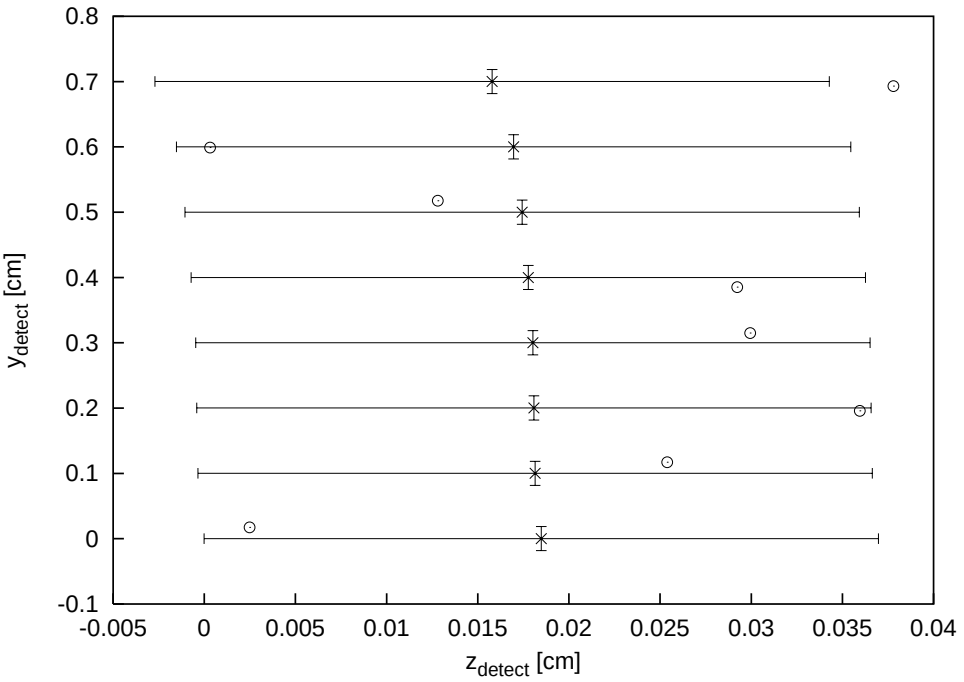

Abbildung 3.12: Vergleich der Abbildungseigenschaften. Hier wurden in beiden Programmen acht Elektronen bei $x = -201.2\ cm$ gestartet, mit Startradien von $y = 0.0\ cm,\ 0.1\ cm,\ \ldots\ , 0.7\ cm$. Für alle Elektronen gilt $E_{kin} = 18790\ eV$ und $\theta = 45°$ relativ zur führenden Magnetfeldlinie. Aufgetragen ist die Position in der z-y-Ebene des Spektrometerausgangs bei $x = 201.2\ cm$. Die ADIPARK-Ergebnisse (Kreuze) stimmen sehr gut mit den SIMION-Ergebnissen (Kreise) überein. Die Fehlerbalken an den ADIPARK-Datenpunkten sind durch den Zyklotronradius gegeben.

entspricht. In Abbildung 3.13 ist das Ergebnis dieser Berechnung zu sehen, aufgetragen ist die Gesamtenergie des Elektrons als Funktion seiner Flugzeit. Die gestrichelte Linie stellt die Berechnungen von ADIPARK dar, hier ist eindeutig zu sehen, daß die Gesamtenergie erhalten ist. Dies entspricht den Erwartungen, denn die in ADIPARK verwendete Berechnungsmethode basiert auf Energieerhaltung, so daß hier keine Abweichung, außer einer numerischen, auftreten kann. Die durchgezogene Linie zeigt das Ergebnis der SIMION-Berechnungen; hier sind starke Energieschwankungen zu sehen. Diese Energieschwankungen treten jedesmal im Bereich des Reflektionspunktes auf, der in einem Bereich des Spektrometers liegt, in dem die elektrischen Feldgradienten sehr hoch sind. Man erkennt, daß die Gesamtenergie konstant bleibt, bis das Elektron wieder in den Bereich der starken elektrischen Feldgradienten fliegt und dort reflektiert wird, hier tritt eine Energieoszillation auf, die zu dem Energiesprung führt. Diese Energiesprünge sind ein Artefakt der Potentialkarteninterpolation von SIMION. Wählt man sehr feine Potentialkarten, wird dieser Fehler vernachlässigbar klein. SIMION kann aber aufgrund der Begrenzung der Potentialkartenverwaltung auf eine maximale Anzahl von Gitterpunkten keine beliebig feinen dreidimensionalen Potentialkarten verwalten. Das Problem läßt sich daher in der Praxis nicht durch eine beliebige Verfeinerung der Potentialkarte beheben, will man die geplanten Untersuchungen mit dreidimensionalen Potentialkarten durchführen.

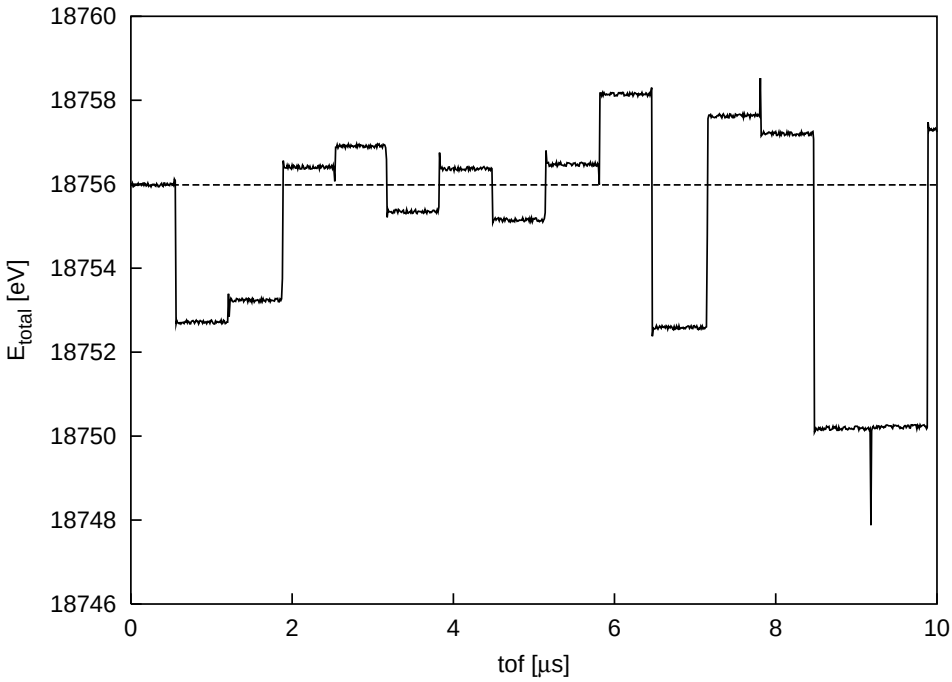

Abbildung 3.13: Energieerhaltung eines gespeicherten Elektrons. Dargestellt ist die Gesamtenergie, kinetische und potentielle, eines über 15 Reflektionen gespeicherten Elektrons über seine Flugzeit (tof). Das Elektron startet bei $x = -100\ cm$ und $y = 20\ cm$ mit $E_{kin} = 100\ eV$ und einem Winkel von $\theta = 35°$ zur führenden Magnetfeldlinie. Die beiden magnetischen Spiegel liegen bei $x = \pm 180\ cm$. Die durchgezogene Linie zeigt das Ergebnis der SIMION-Berechnung, gestrichelt ist das ADIPARK-Ergebnis eingezeichnet.

Parallel zur Langzeitmessung für die Energieerhaltung, wurde die Berechnungszeit der beiden Programme auf identischen Computern festgehalten. Um das Elektron $tof = 10\ \mu s$ lang zu verfolgen, benötigt SIMION $t^{SIMION} \approx 190\ s$. ADIPARK benötigt für die gleiche Berechnung mit sehr feiner Schrittweiteneinstellung von $\Delta s_1 = 0.1\ mm$ nur $t_1^{ADIPARK} \approx 90\ s$, mit noch ausreichend feiner Einstellung $\Delta s_2 = 1\ mm$ nur noch $t_2^{ADIPARK} \approx 9\ s$. Es ergibt sich daher ein minimaler Geschwindigkeitsgewinn um Faktor 2 bis 20 für ADIPARK, wenn man mit sehr hoher Genauigkeit rechnet. Für gewöhnliche Kurzzeitmessungen sind auch Schrittweiteneinstellungen von $\Delta s \approx 5\ mm$ noch ausreichend, so daß sich ein Geschwindigkeitsgewinn von bis zu Faktor 100 ergibt.

Als Ergebnis all dieser Vergleiche kann man festhalten, daß man mit ADIPARK sehr vertrauenswürdige und schnelle Berechnungen durchführen kann. Zu beachten ist jedoch, daß man nur mit Teilchen arbeiten kann, die sich durch die adiabatische Näherung erster Ordnung ausreichend gut beschreiben lassen. Für nicht adiabatisch zu beschreibende Teilchen muß auf SIMION zurückgegriffen werden, allerdings sollte man hier dann die Gesamtenergie beobachten. Es ist klar, daß SIMION nicht durch ADIPARK ersetzt werden kann, beide Programme können sich aber zu einem sehr wertvollen Werkzeug ergänzen.

Kapitel 4

Gespeicherte Teilchen im MAC–E–Filter

In diesem Kapitel soll nun erläutert werden, daß in einem Experiment mit der in Kapitel 2.2 beschriebenen elektromagnetischen Konfiguration geladene Teilchen gespeichert werden können. Die beschriebenen Speichermechanismen sollen durch Simulationen der Flugbahnen der geladenen Teilchen in einem MAC–E–Filter belegt werden. Im Hinblick auf folgende Kapitel, soll die Darstellungsweise der Flugbahndaten eingeführt werden. Eine besondere Art der Darstellung wird durch die Möglichkeit der Berechnung von Speichervolumenkarten vorgestellt. Schließlich wird noch auf potentielle Quellen für Untergrundereignisse eingegangen.

4.1 Speicherbedingungen für geladene Teilchen

Gespeicherte Teilchen bewegen sich auf Zyklotronbahnen um eine führende Magnetfeldlinie herum, d.h. die Flugbahn der Teilchen wird durch das axiale Magnetfeld definiert. Daraus folgt, daß sich gespeicherte Teilchen in axialer Richtung bewegen und die radiale Ausdehnung der Flugbahn durch den jeweiligen Zyklotronradius gegeben ist. Dies gilt, solange die magnetische Führung stark genug ist. Teilchen mit hoher Masse, z.B. Protonen oder Ionen, und Teilchen mit sehr hoher transversaler kinetischer Energie erfahren aufgrund der kleineren Zyklotronfrequenz und damit der größeren Zyklotronschrittlänge eine geringere Führung. In Abschnitt 3.2.4.2 wird gezeigt, unter welchen Bedingungen die magnetische Führung eines Ions mit der eines Elektrons vergleichbar ist. Es treten jetzt verschiedene Arten von Reflektionspunkten auf. Wird ein Teilchen durch das elektrische Potential reflektiert, gilt das Prinzip der Penningfalle. Im Gegensatz dazu kann ein Teilchen auch unabhängig vom elektrischen Feld durch das Prinzip der magnetischen Flasche (s. Gl. (2.8)) gespeichert werden. Diese Tatsachen werfen weitere Fragen nach Speicherbedingungen für verschiedene Teilchen mit unterschiedlichen Ladungen auf.

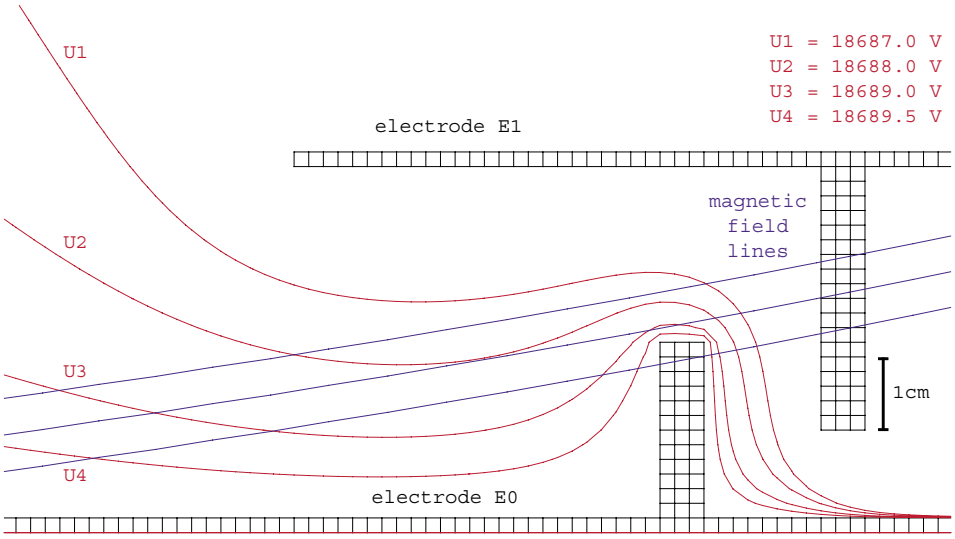

Abbildung 4.1: Teilchenfalle im Elektrodenzwischenraum. Hier ist der Zwischenraum zwischen Elektrode E0 und E1 dargestellt. Die mittlere Magnetfeldlinie schneidet die Äquipotentiallinie U3 = $-18689.0\ V$ zwei Mal, d.h. hier entsteht ein Potentialmaximum (s. Abb. 4.2) und ein Elektron kann gespeichert werden.

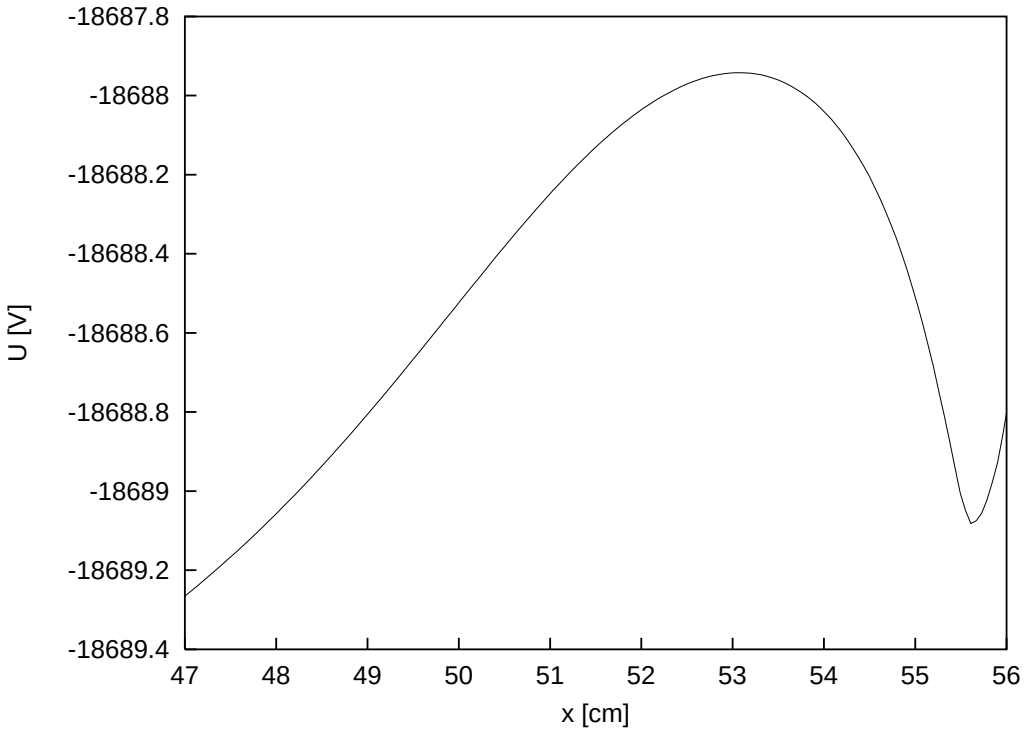

Abbildung 4.2: Potentialmaximum im Elektrodenzwischenraum. Gezeigt ist das Potential im Zwischenraum der Elektroden E0 und E1 entlang der mittleren Magnetfeldlinie von Abbildung 4.1. Der Potentialtopf hat eine Tiefe von etwa 1 eV.

Zuerst zu gespeicherten Teilchen mit elektrischen Reflektionspunkten:

Der einfachste Speicherfall tritt für positiv geladene Teilchen auf. Diese Teilchen besitzen aufgrund des negativen Analysierpotentials symmetrisch zur Analysierebene rein elektrische Reflektionspunkte. Das Analysierpotential bildet eine Art Potentialtopf, in dem die positiv geladenen Teilchen gefangen sind. Die Bewegung im Potentialtopf wird weiterhin vom Magnetfeld vorgegeben, während das elektrische Potential nur die Reflektionspunkte definiert. Negative Ionen werden genauso vom Analysierpotential abgestoßen wie Elektronen. Es besteht natürlich die Möglichkeit Positronen zu speichern, jedoch ist kein ernstzunehmender Prozeß bekannt, durch den diese in das Spektrometervolumen gelangen oder darin entstehen können.

Die einzige Möglichkeit durch elektrische Reflektionsbedingungen Elektronen zu speichern besteht in der Existenz von kleinen Penningfallen, die in den Elektrodenzwischenräumen vorhanden sind. Kreuzt eine magnetische Feldlinie eine elektrische Äquipotentiallinie zweimal in der Art, daß sie senkrecht über eine Sattelfläche des elektrischen Potentials verläuft, entsteht in diesem Bereich ein elektrisches Potentialmaximum entlang einer magnetischen Feldlinie (s. Abb. 4.1, 4.2). Elektronen, die entlang einer solchen magnetischen Feldlinie geführt werden und deren Energie nicht ausreicht, das elektrische Potential zu überwinden, werden gespeichert.

Wie werden nun geladene Teilchen durch magnetische Reflektionen gespeichert?

Das Prinzip der magnetischen Speicherung beruht auf der in Abschnitt 2.2.1 beschriebenen Transformation von transversaler kinetischer Energie in longitudinale kinetische Energie und umgekehrt (magn. Spiegeleffekt). Dieser Prozeß soll hier am Beispiel eines Elektrons erläutert werden. Ein Elektron starte an einer Stelle mit Magnetfeld B_i und elektrischem Potential U_i. Das negative elektrische Potential der Analysierebene beschleunigt das Elektron von der Mittelebene weg in Richtung des positiveren elektrischen Potentials U_f. Während dieser Bewegung nimmt die magnetische Feldstärke auf B_f zu, die Gradientenkraft (s. Gl. (2.6)) wirkt in diesem Fall abbremsend auf das Teilchen. Es findet hier eine Transformation von longitudinaler zu transversaler kinetischer Energie statt und mit den Gleichungen (2.8) und (2.10) ergibt sich die neue longitudinale kinetische Energie $E_{\|f}$ zu

$$E_{\|f} = E_{\|i} + \left(1 - \frac{B_f}{B_i}\right) E_{\perp i} + q\,(U_i - U_f)\,. \tag{4.1}$$

Beim Flug in Richtung des starken Magnetfeldes ist das Verhältnis $\frac{B_f}{B_i} > 1$ und es wird fortschreitend mehr Energie der longitudinalen Komponente entzogen. Während dieses Prozesses wird das Elektron durch die Potentialdifferenz $\Delta U = U_i - U_f$ beschleunigt, d.h. die longitudinale kinetische Energie steigt an. Allgemein kann man festhalten, daß ein Elektron dann magnetisch gespeichert wird, wenn die longitudinale kinetische Startenergie und die Beschleunigung durch die Potentialdifferenz kleiner sind, als der bis zum Erreichen des magnetischen Feldmaximums benötigte

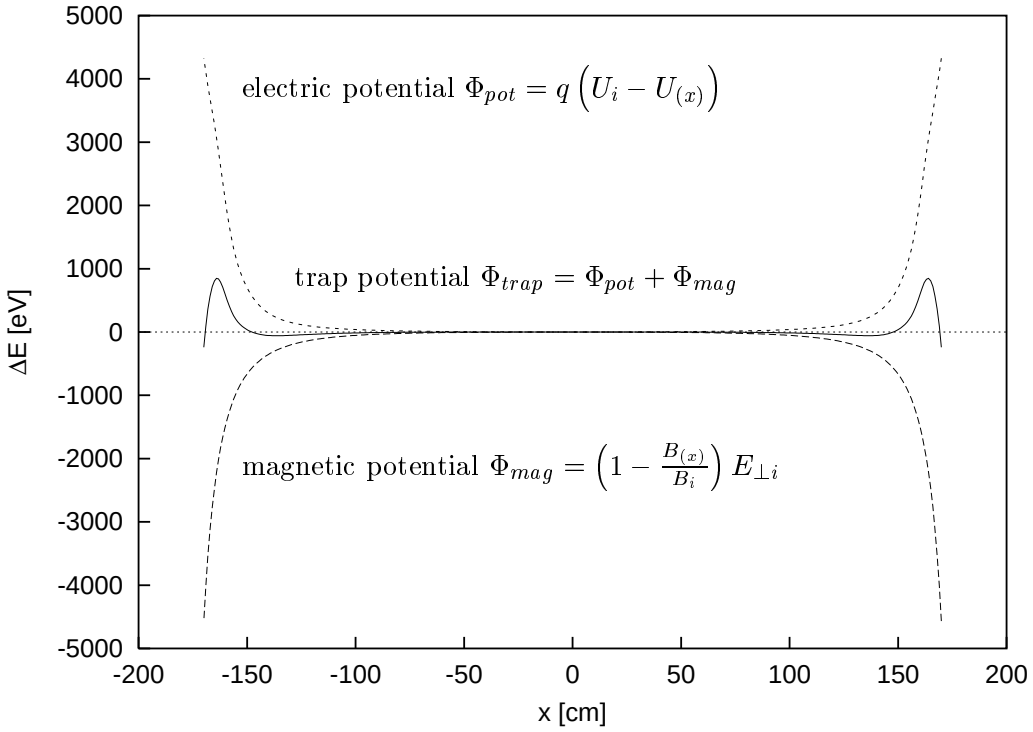

Abbildung 4.3: Teilchenfalle mit magnetischen Reflektionspunkten. Dieses Bild zeigt die Potentialänderungen eines gespeicherten Elektrons, relativ zu seiner Startposition. Das Elektron startet im Spektrometerzentrum mit $E_{kin} = 250\ eV$ und $\theta = 20°$ und wird im Bereich $-170\ cm \leq x \leq 170\ cm$ durch magnetische Reflektion gespeichert. Das für dieses Elektron formulierbare magnetische Potential $\Phi_{mag} = \left(1 - \frac{B_{(x)}}{B_i}\right) E_{\perp i}$ bildet mit dem elektrischen Potential Φ_{pot} einen Potentialtopf Φ_{trap}, der das Elektron gefangen hält.

transversale Energieanstieg (s. Abb. 4.3).

$$E_{\parallel i} + q\,(U_i - U_f) \leq E_{\perp i}\left(1 - \frac{B_f}{B_i}\right) \tag{4.2}$$

Die Speicherbedingung lautet also: Ein Elektron wird magnetisch gespeichert, wenn die magnetische Energietransformation schneller kinetische Energie aus der longitudinalen Komponente entzieht, als das elektrische Potential longitudinale kinetische Energie hinzufügen kann. Zu beachten ist, daß die Energietransformation stark von den Startparametern $E_{\parallel i}$, $E_{\perp i}$ und den Feldern U_i, B_i abhängt.

Betrachtet man nun ein Elektron, welches durch einen Stoßprozeß mit Restgasmolekülen in der Analysierebene startet und ausschließlich transversale kinetische Energie besitzt, so wird dieses durch die magnetischen Spiegel gespeichert, falls seine kinetische Energie größer als die Spektrometerauflösung (s. Gl. (2.9)) ist. In diesem Fall wäre der erforderliche transversale kinetische Energieanstieg größer, als der durch das elektrische Potential mögliche Gewinn an kinetischer Energie, d.h. das Elektron

wird gespeichert.

Jetzt soll noch ein Fall gemischter Speicherbedingungen angesprochen werden.

Wenn ein negatives Teilchen die Speicherbedingungen für einen magnetischen Reflektionspunkt erfüllt, aber nicht genügend kinetische Energie besitzt, um das Analysierpotential zu überwinden, wird es nur auf einer Spektrometerseite gespeichert. In Abbildung 4.3 kann man diesen Effekt am lokalen Minimum von Φ_{trap} bei $x \approx \pm 130\ cm$ erkennen. Zur Spektrometermitte hin wird das negative Teilchen durch das starke elektrische Potential reflektiert. Auf seinem Weg zu den Spektrometeröffnungen überwiegt die magnetische Energietransformation und ein magnetischer Reflektionspunkt tritt auf. Dieses Teilchen besitzt dann einen magnetischen und einen elektrischen Reflektionspunkt, daher die Bezeichnung als Speicherung mit gemischten Speicherbedingungen. Auch in diesem Fall wird das Teilchen weiterhin vom Magnetfeld geführt.

Abschließend kann man klar bestätigen, daß es sehr viele Möglichkeiten zur Speicherung geladener Teilchen in einem MAC–E–Filter gibt. Es können sowohl Ionen als auch Elektronen gespeichert werden und sind daher mögliche Ursachen für Erhöhungen der Untergrundzählrate, wie Abschnitt 4.5 zeigt.

4.2 Speicherung von Elektronen

In Abschnitt 4.1 wurde beschrieben, daß Elektronen im Flußschlauch nur durch die Möglichkeit der magnetischen Reflektion gespeichert werden können. Als Spezialfall wurde beschrieben, daß Elektronen zur Analysierebene hin durch das elektrische Potential reflektiert werden und zum Spektrometerausgang wieder die magnetische Flasche die Reflektion verursacht. Weiterhin wurde abseits des Flußschlauchs die Existenz von kleinen Penningfallen in den Elektrodenzwischenräumen gezeigt. Alle Flugbahnsimulationen mit Elektronen wurden mit ADIPARK erstellt.

Wie stellt sich nun das Simulationsergebnis solcher Elektronen dar? In Abbildung 4.4 ist die Flugbahn eines Elektrons mit einem Startwinkel von $\theta = 70°$ relativ zur führenden Magnetfeldlinie dargestellt. Bei diesem Elektron wirkt die magnetische Energietransformation aufgrund des großen Startwinkels sehr früh ausreichend stark, um bei $x = \pm 50\ cm$ zu einer Reflektion zu führen. Das Elektron wird in einem ein Meter breiten Volumen um die Analysierebene gespeichert. Zu erkennen ist auch die azimutale Bewegung des Teilchens, d.h. die Teilchenbahn rotiert um die Spektrometerachse. Die radiale Ausdehnung der Flugbahn beträgt nur einige Zentimeter, wobei man anmerken muß, daß hier keine Zyklotronbewegung sichtbar ist, sie aber in Realität dennoch vorhanden ist und einige Zentimeter ausmacht.

In Abbildung 4.5 ist jetzt ein Elektron mit identischen Startparametern zu sehen, jedoch wurde hier der Startwinkel auf $\theta = 30^{\circ}$ reduziert. Mit diesem geringen Startwinkel wirkt die magnetische Energietransformation erst viel später, was dazu führt, daß das Elektron jedesmal fast das gesamte Spektrometervolumen durchläuft, bis es den nächsten Reflektionspunkt erreicht. Das Teilchen wird jetzt in einem 3.4 m breiten Volumen gespeichert, die radiale Ausdehnung der Flugbahn beträgt mehr als 10 cm.

Den oben erwähnten Spezialfall, daß ein Elektron zur Analysierebene hin durch das elektrische Potential reflektiert wird, kann man in Abbildung 4.6 sehen. Das Elektron besitzt nicht genug kinetische Energie um das Analysierpotential zu überwinden, wird zum Spektrometereingang hin durch die magnetische Flasche reflektiert und bleibt so auf einer Spektrometerseite gespeichert.

Ein Beispiel für die abseits des Flußschlauchs liegenden kleinen Penningfallen in den Elektrodenzwischenräumen ist in Abbildung 4.7 zu sehen. Das Elektron wird hier in dem Potentialtopf zwischen den beiden Elektroden E0 und E1, der entlang der Magnetfeldlinie vorhanden ist, gefangen. Es handelt sich hierbei um ein niederenergetisches Elektron mit nur $E_{kin} = 0.25\ eV$, es wird sehr lokal zwischen den beiden Elektroden gespeichert. Auch hier ist wieder die Azimutalbewegung vorhanden; das Elektron kann aber niemals den magnetischen Flußschlauch, der auf den Detektor trifft, erreichen.

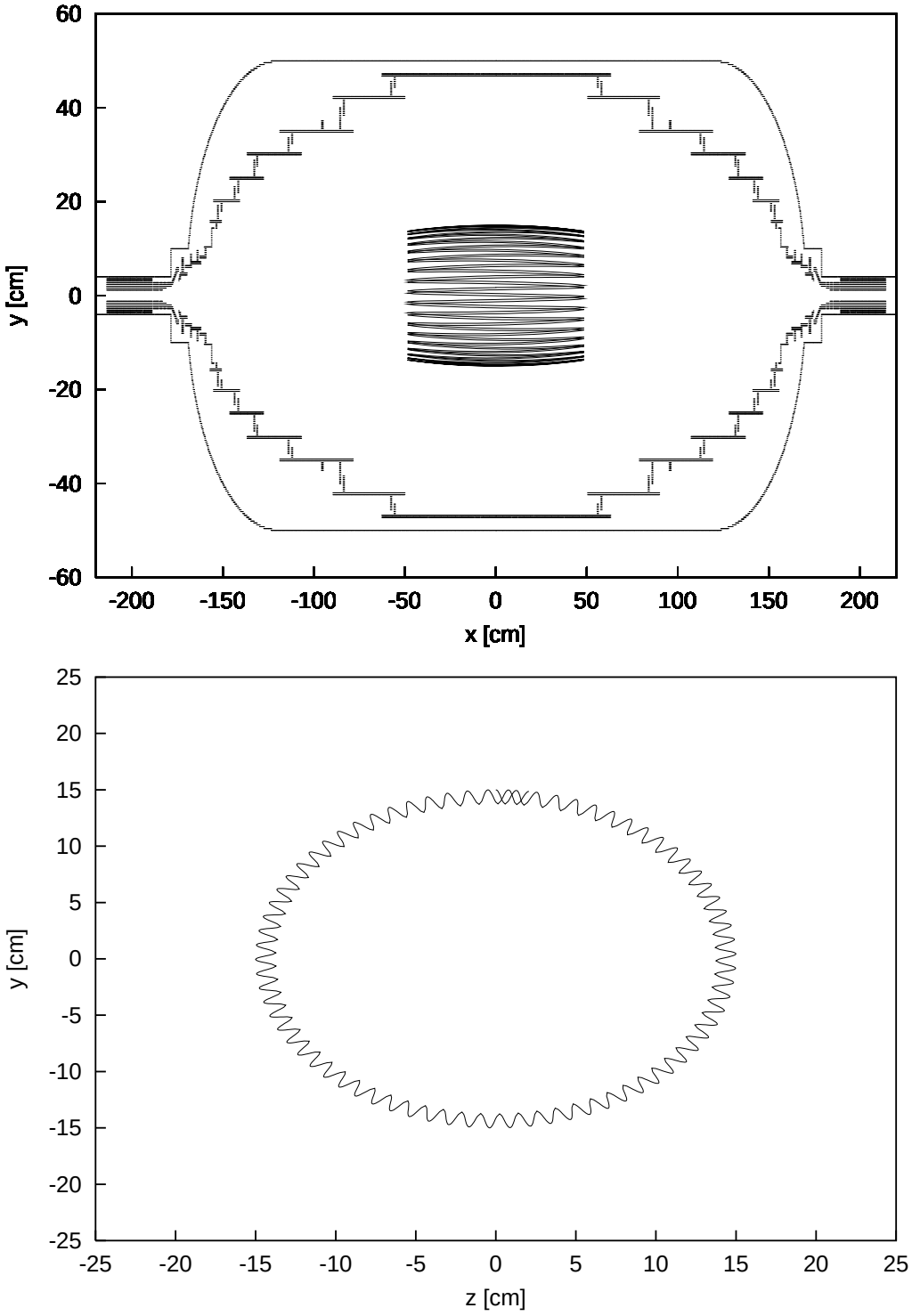

Abbildung 4.4: Flugbahn eines gespeicherten Elektrons in der Spektrometermitte. Dargestellt ist die Flugbahn eines Elektrons, das in der Analysierebene mit einem radialen Abstand zur Spektrometerachse von $y = 15\ cm$ mit $E_{kin} = 100\ eV$ und einem Winkel von $\theta = 70°$ relativ zur führenden Magnetfeldlinie gestartet wurde. Das Elektron wird $tof \approx 53\ \mu s$ lang verfolgt und in dieser Zeit 75 mal reflektiert. Im oberen Bild ist die Seitenansicht (x-y-Ebene) des Spektrometers mit der Teilchenbahn zu sehen. Das Teilchen wird in einem ein Meter breiten Volumen um die Analysierebene gespeichert. Im unteren Bild ist der zugehörige Schnitt durch die Analysierebene (z-y-Ebene) und die Projektion der Teilchenbahn zu sehen. Hier ist sehr schön zu erkennen, wie das Elektron die azimutale Bewegung beschreibt.

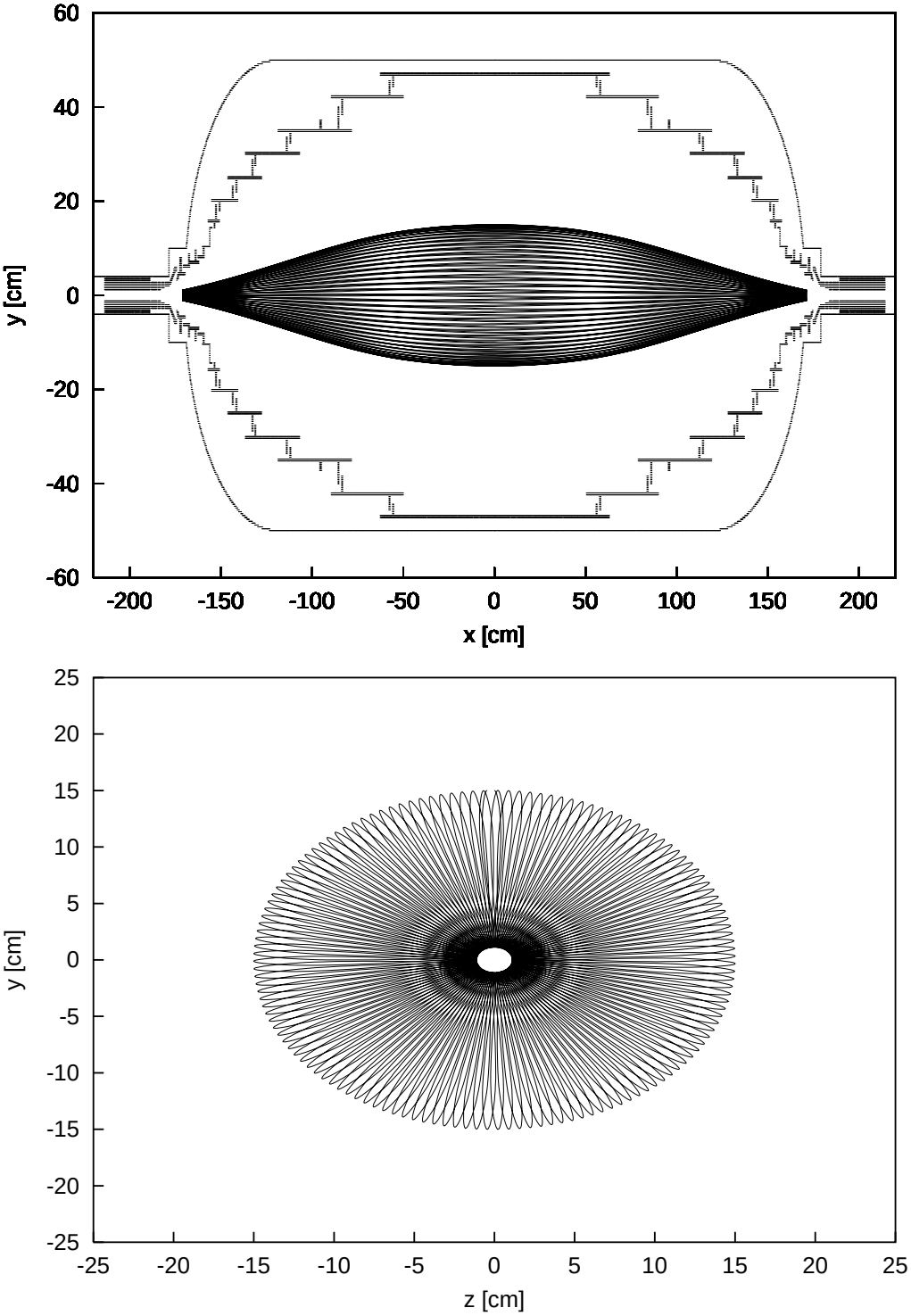

Abbildung 4.5: Flugbahn eines gespeicherten Elektrons im gesamten Volumen. Dargestellt ist die Flugbahn eines Elektrons, das in der Analysierebene mit einem radialen Abstand zur Spektrometerachse von $y = 15\ cm$ mit $E_{kin} = 100\ eV$ und einem Winkel von $\theta = 30°$ relativ zur führenden Magnetfeldlinie gestartet wurde. Das Elektron wird $tof \approx 90\ \mu s$ lang verfolgt und in dieser Zeit 140 mal reflektiert. Im oberen Bild ist wieder die Seitenansicht (x-y-Ebene) des Spektrometers mit der Teilchenbahn zu sehen. Das Elektron durchläuft jedesmal annähernd das gesamte Spektrometervolumen, bevor es reflektiert wird. Im unteren Bild ist der zugehörige Schnitt durch die Analysierebene (z-y-Ebene) und die Projektion der Teilchenbahn zu sehen.

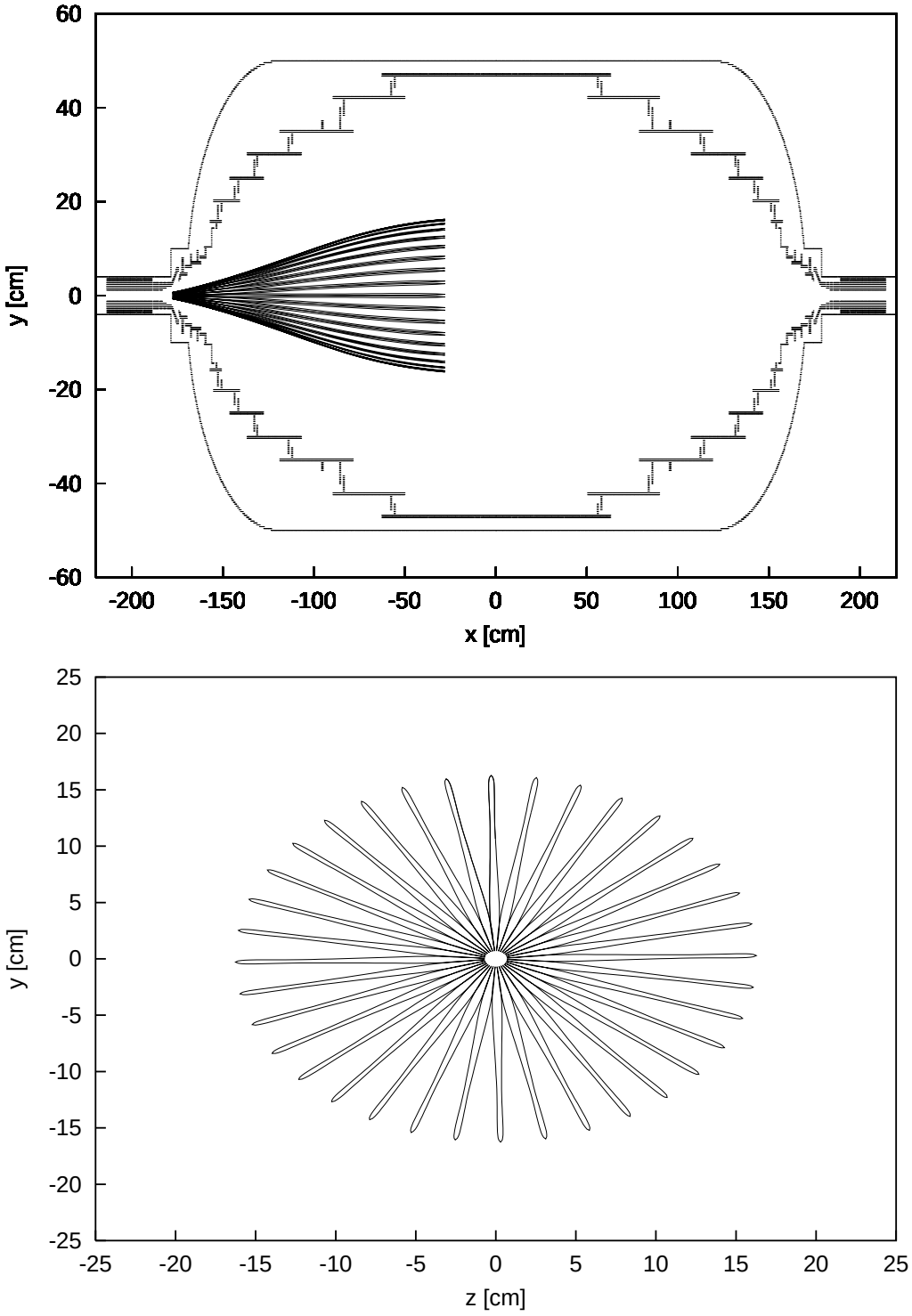

Abbildung 4.6: Flugbahn eines gespeicherten Elektrons im linken Halbraum. Dargestellt ist die Flugbahn eines Elektrons, das bei $(x, y, z) = (-100, 10, 0)\ cm$ mit $E_{kin} = 55\ eV$ und einem Winkel von $\theta = 80°$ relativ zur Spektrometerachse gestartet wurde. Das Elektron wird $tof \approx 110\ \mu s$ lang verfolgt und in dieser Zeit 75 mal reflektiert. Im oberen Bild ist die Seitenansicht (x-y-Ebene) des Spektrometers mit der Teilchenbahn zu sehen. Das Elektron wird links der Analysierebene gespeichert, bei $x \approx -25\ cm$ reicht seine kinetische Energie nicht mehr aus, um das Analysierpotential zu überwinden. Im unteren Bild ist der zugehörige Schnitt durch die Analysierebene (z-y-Ebene) und die Projektion der Teilchenbahn zu sehen.

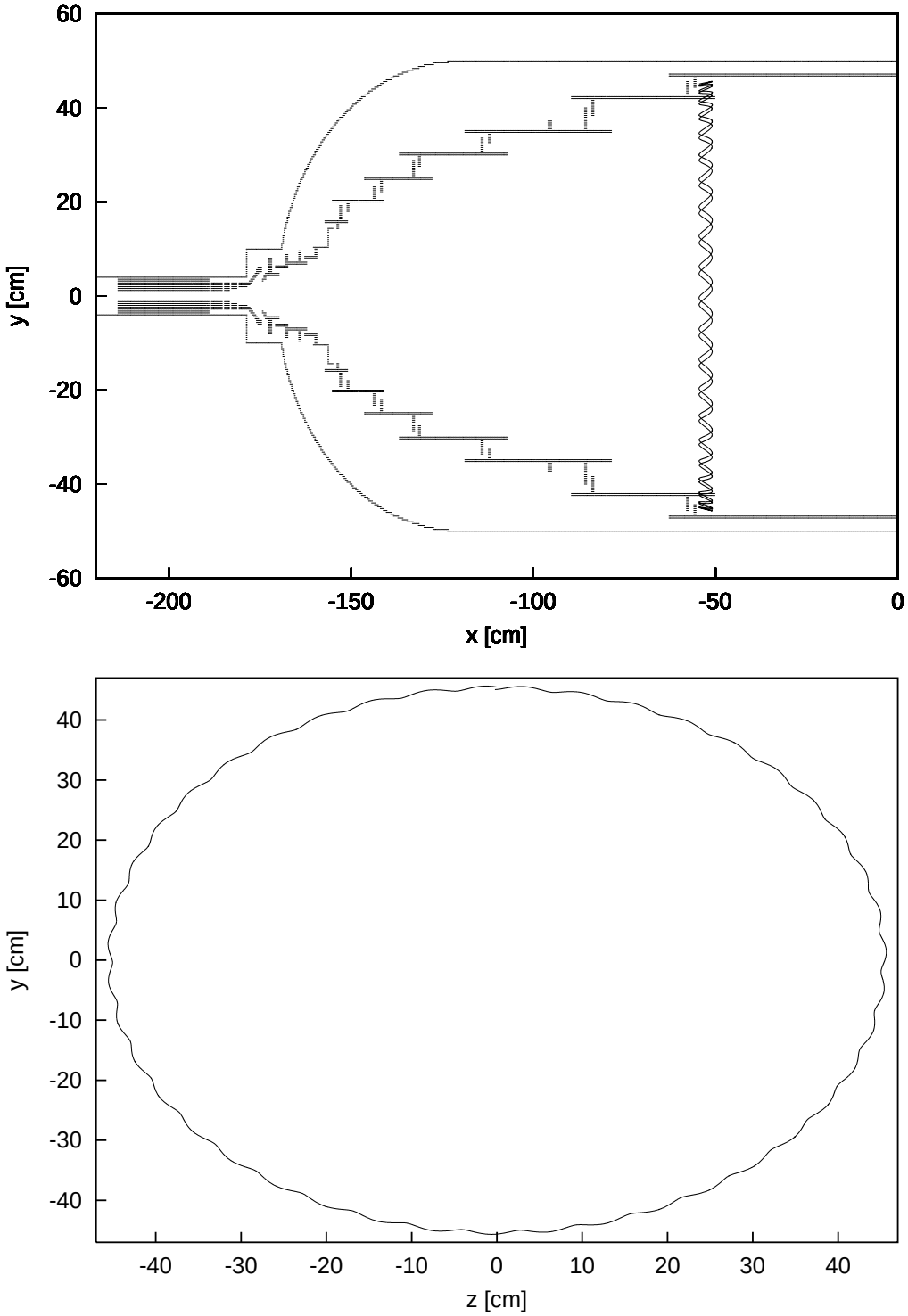

Abbildung 4.7: Flugbahn eines gespeicherten Elektrons in einer Penningfalle. Dargestellt ist die Flugbahn eines Elektrons, das bei $(x, y, z) = (-52.0, 45.5, 0.0)\ cm$ mit $E_{kin} = 0.25\ eV$ und in Richtung des Magnetfeldes gestartet wurde. An dieser Position liegt eine kleine Penningfalle im Elektrodenzwischenraum. Das Elektron wird $tof \approx 14\ \mu s$ lang verfolgt und in dieser Zeit 85 mal reflektiert. Im oberen Bild ist die Seitenansicht (x-y-Ebene) der Hälfte des Spektrometers mit der Teilchenbahn im Zwischenraum zwischen Elektrode E0 und E1 zu sehen. Im unteren Bild ist der zugehörige Schnitt durch die Analysierebene (z-y-Ebene) und die Projektion der Teilchenbahn zu sehen.

4.3 Speicherung von Ionen

Der Versuch, ein Ion im Mainzer MAC–E–Filter zu speichern, hat in keinem der vorhandenen Simulationsprogramme zum Erfolg geführt. Dies ist aus verschiedenen Gründen verständlich.

Um negative Ionen speichern zu können, wird der magnetische Spiegeleffekt benötigt. Die Auflösung des Mainzer Spektrometers beträgt $\Delta E = 4.8\ eV$, daher benötigt ein Ion eine transversale kinetische Energie von $E_\perp \geq 4.8\ eV$, um den magnetischen Speigeleffekt zu nutzen. Ein Ion der Masse eines Protons führt in der Analysierebene des Mainzer Spektrometers mit $B_{min} \approx 5\ Gauss$ zu einem Zyklotronradius von etwa $r_{cycl}^{Proton} \approx 60\ cm$. Da der maximale Radius der Elektrodenkonstruktion $r_{max} = 47\ cm$ ist, ist es nicht möglich im Mainzer Spektrometer ein Ion magnetisch zu speichern.

Positive Ionen können aufgrund des Potentialtopfes des Analysierpotentials in einer Penningfalle gespeichert werden. Hier sind dann auch Energien $E_\perp < 4.8\ eV$ möglich, so daß die Zyklotronbahn nicht zu groß ist. Allerdings werden, durch das schwache Magnetfeld, schon Protonen nur schlecht magnetisch geführt. Der vorhandene Potentialdurchgriff führt dann dazu, daß die positiven Ionen sich direkt in Richtung der Elektroden bewegen und nicht gespeichert werden können.

Aufgrund der schwachen magnetischen Führung und der prinzipiellen Grenzen der adiabatischen Näherung aus Abschnitt 3.2.4, können im Mainzer Spektrometer keine Ionen mit ADIPARK verfolgt werden und es muß auf SIMION zurückgegriffen werden. Die SIMION-Simulationen zeigen, daß Ionen auf den Detektor geführt werden können, falls sie nicht gegen eine Wand stoßen.

Um dennoch Ionen simulieren zu können, wird hier der Entwurf des Vorspektrometers des geplanten KATRIN-Experimentes verwendet. Dieser Entwurf stammt von B. Flatt und wird in [Fla01a] beschrieben. In diesem Vorspektrometer herrscht ein minimales Magnetfeld von $B_{min} \approx 250\ Gauss$, was um den Faktor 50 stärker ist, als im Mainzer Spektrometer. Die Zyklotronradien sind dann um Faktor 50 kleiner und ein Ion stößt nicht unweigerlich gegen eine Elektrode. Der Zyklotronradius eines Protons im Vorspektrometer ist dann in etwa dem eines Elektrons im Mainzer Spektrometer vergleichbar.

In Abbildung 4.8 ist nun die SIMION-Simulation und die ADIPARK-Simulation eines Protons im KATRIN-Vorspektrometer gezeigt. Das Proton verhält sich ähnlich einem Elektron, nur daß es sehr viel langsamer fliegt. Es benötigt $tof \approx 170\ \mu s$ um 19 mal reflektiert zu werden. Die ADIPARK-Bahn zeigt das Führungszentrum der Bewegung, bei den SIMION-Daten ist zusätzlich die Zyklotronbewegung zu sehen.

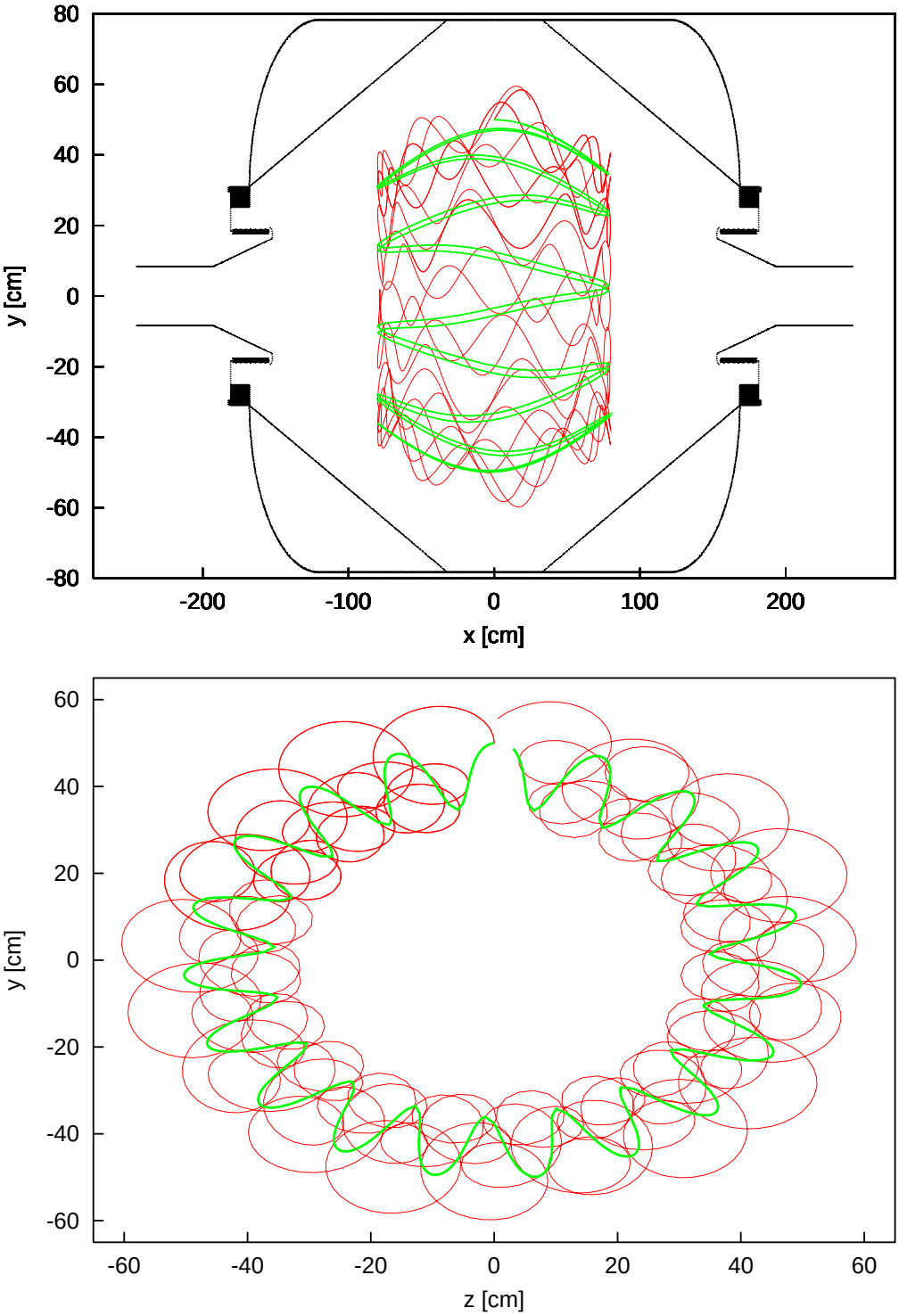

Abbildung 4.8: Flugbahn eines gespeicherten Protons. Dargestellt ist die Flugbahn eines Protons im Vorspektrometer-Entwurf von B. Flatt für das KATRIN-Experiment [Fla01a]. Das Analysierpotential des Vorspektrometers beträgt $U_0 = -18000\ V$. Das Proton startet bei $(x, y, z) = (0, 50, 0)\ cm$ mit $E_{kin} = 500\ eV$ und mit einem Winkel von $\theta = 35°$ relativ zur führenden Magnetfeldlinie. Das Proton wird $tof \approx 170\ \mu s$ lang verfolgt und in dieser Zeit 19 mal reflektiert. Oben ist die Seitenansicht (x-y-Ebene) des Katrin-Vorspektrometer-Entwurfs zu sehen, das Proton wird in einem 1.5 m breiten Bereich um die Analysierebene gespeichert. Unten ist der Schnitt durch die Analysierebene (z-y-Ebene) und die Projektion der Teilchenbahn zu sehen. Zum Vergleich ist das Ergebnis der SIMION-Simulation (rot), mit Zyklotronbewegung, und das ADIPARK-Ergebnis (grün), nur Führungszentrum, dargestellt.

4.4 Speichervolumenkarten des Mainzer Spektrometers

Nachdem jetzt die Flugbahnen gespeicherter Teilchen bekannt sind, stellt sich die Frage nach den Startparametern, die ein gespeichertes von einem nicht gespeicherten Teilchen unterscheiden. Man könnte jetzt mühsam eine Vielzahl Teilchen mit unterschiedlichen Startparametern von Hand simulieren, um diese Frage beantworten zu können. Ein viel einfacherer Weg besteht darin, die modulare Struktur des Simulationsprogramms ADIPARK auszunutzen und ein zusätzliches Modul einzubinden. Dieses Modul soll dann die Suche nach den Startparametern der gespeicherten Teilchen durchführen. Auf Basis dieser Idee und auf Grundlage der Vorarbeit von [Con01] konnte ein solches Modul, im folgenden Trapping–Modul genannt, entwickelt werden.

Das Trapping–Modul arbeitet auf Basis eines zweidimensionalen Koordinatengitters, das über das Spektrometervolumen gelegt wird. Ein dreidimensionales Gitter ist prinzipiell möglich, erfordert dann aber einen extrem großen Rechenaufwand. Dem Modul werden dann ein Startwinkelbereich und ein Startenergiebereich übergeben, innerhalb derer dann die Teilchenverfolgung durchgeführt werden soll. Der Algorithmus arbeitet Schritt für Schritt jeden Gitterpunkt ab und startet jeweils alle Teilchen des vorgegebenen Energie- und Winkelbereiches. Es werden der Reihe nach von großen nach kleinen Startwinkeln Teilchen gestartet. Wird ein nicht gespeichertes Teilchen gefunden, werden alle niedrigeren Startwinkel verworfen und als nicht gespeichert gewertet. Im linearen zylindersymmetrischen Mainzer Spektrometer wird ein Elektron als gespeichert betrachtet, wenn es mindestens zwei mal reflektiert wurde. Kommt das Führungszentrum näher als einen Zyklotronradius an eine Elektrode heran, wird das Teilchen als kollidiert betrachtet. Für die Berechnung der in diesem Abschnitt gezeigten Bilder mußte je Bild eine Anzahl von etwa 50000 Elektronen simuliert werden, was eine Berechnungszeit von unter einer Stunde je Bild erfordert hat. Dies zeigt den Fortschritt, den das Trapping–Modul durch Automatisierung der Bahnverfolgung darstellt. Ergebnis einer solchen Berechnung ist dann eine Datei, die für jeden Gitterpunkt des gewählten Koordinatengitters Informationen enthält, welche Teilchen mit welchen Startparametern gespeichert werden, falls sie an genau dieser Stelle starten, bzw. erzeugt werden. Man erhält hier keine weitere Informationen über die reale Teilchenbahn, es gibt nur ein Ja/Nein Kriterium, das angibt, ob das Teilchen gespeichert wird oder nicht.

Zur Darstellung der so gewonnenen Daten bietet sich die Konturdarstellung an, in der Gebiete gleicher Höhe mit einer Konturlinie umrandet gezeichnet werden. Grundlage für die Konturdarstellung ist das Koordinatengitter der Trapping-Berechnung, wobei sich die Höhe aus dem zur Speicherung nötigen minimalen Startwinkel der Teilchen an jeder Stelle ergibt. Die so entstandenen Grafiken aus Konturlinien werden im folgenden Speichervolumenkarten genannt, da sie angeben, mit welchen Startparametern ein Teilchen, das im markierten Raumbereich startet, im Spektrometer gespeichert wird. Man erhält auf diese Weise eine Art Phasenraumdichte für die gespeicherten Teilchen.

Zur Untersuchung von Elektronenspeicherbedingungen im Mainzer Spektrometer, wird ein Koordinatengitter mit $g = 1\ cm$ Gitterabstand verwendet. Es werden Elektronen mit Energien von $2\ eV \leq E_{kin} \leq 512\ eV$ und Startwinkeln von $10° \leq \theta \leq 80°$ verwendet. Die Startwinkel werden immer relativ zur führenden Magnetfeldlinie am Startort eingestellt. Zur Berechnung wird nur die Hälfte der Querschnittsfläche des Spektrometers verwendet, da Spiegel- und Zylindersymmetrie besteht. Weiterhin soll schon hier die gleiche Darstellungsform wie in späteren Untersuchungen verwendet werden. In den Abbildungen 4.9 bis 4.14 sind die Resultate für $E_{kin} = 512\ eV$, $E_{kin} = 256\ eV$, $E_{kin} = 128\ eV$, $E_{kin} = 32\ eV$, $E_{kin} = 16\ eV$ und $E_{kin} = 8\ eV$ dargestellt. Zu sehen sind die Bereiche im Spektrometervolumen, in denen Elektronen starten müssen, um gespeichert zu werden. Die dargestellten Konturen haben folgende Bedeutung:

- Die innerste Kontur für einen niedrigen Startwinkel von $\theta = 10°$, zeigt den Bereich, in dem ein Elektron mit $\theta \geq 10°$ starten muß, damit es gespeichert wird.
- Die nächst äußere Kontur für $\theta = 20°$ zeigt, daß Elektronen mit $\theta \geq 20°$ gespeichert werden, falls sie hier starten. Im Winkelbereich $10° < \theta < 20°$ existiert also ein Grenzwinkel, unter welchem die startenden Elektronen nicht mehr gespeichert werden.
- Schließlich definiert die äußerste Kontur für Startwinkel $\theta \geq 80°$, daß die hier startenden Elektronen nur noch gespeichert werden, falls ihr Startwinkel $\theta \geq 80°$ ist.

Es soll nochmals betont werden, daß die Konturen nur die Startbereiche der Elektronen angeben; die wirkliche Flugbahn kann daraus nicht abgeleitet werden. Diese Art der Darstellung soll zeigen, inwieweit man mit gespeicherten Elektronen rechnen muß, wenn im Spektrometervolumen Tritium-Zerfall stattfindet oder Stöße mit Restgas auftreten, wobei Elektronen mit den simulierten Startbedingungen entstehen können.

Die Abbildungen 4.9 bis 4.14 zeigen sehr schön, wie die Startbereiche für gespeicherte Elektronen mit der Startenergie abnehmen. Wählt man eine Startenergie unterhalb der Spektrometerauflösung (s. Gl. (2.9)) werden keine Elektronen mehr gespeichert; es ist nicht mehr genug transversale kinetische Energie $E_\perp$ vorhanden, um am Spiegel der magnetischen Flasche reflektiert zu werden. Diesen Effekt sieht man auch bei den Elektronen höherer Energie, bei $E_{kin} = 16\ eV$ in Abbildung 4.13 werden nur noch Elektronen mit $\theta \geq 40°$ gespeichert, da bei kleineren Winkeln die transversale kinetische Energie für den magnetischen Spiegeleffekt nicht mehr ausreicht. Bei $E_{kin} = 128\ eV$ in Abbildung 4.11 werden immer noch keine Elektronen mit $\theta < 20°$ gespeichert, erst bei Energien von $E_{kin} \geq 256\ eV$ (Abbildung 4.10) werden die Elektronen mit den niedrigsten untersuchten Winkeln gespeichert. Der Abstand der Konturen zu den Elektroden läßt sich durch die Ausdehnung der Zyklotronbewegung erklären. Startet ein Elektron zu dicht an einer Elektrode, stößt es im Laufe

eines Zyklotronumlaufes gegen diese und kann daher nicht gespeichert werden.

Alle bis jetzt besprochenen Speichervolumenkarten zeigen nur magnetisch gespeicherte Elektronen, wobei das Trapping-Modul nicht zwischen rein magnetisch gespeicherten Teilchen und Teilchen mit elektrischem und magnetischem Reflektionspunkt unterscheiden kann. Nicht zu sehen sind die kleinen Penningfallen in den Elektrodenzwischenräumen. Diese kleinen Penningfallen werden nur sichtbar, wenn man sehr niedrige Startenergien wählt und mit einem sehr feinen Raster zwischen den Elektroden abtastet. Abbildung 4.15 zeigt die Speichervolumenkarte für die kleinen Penningfallen, es wurden Elektronen mit Energien $E_{kin} < 1\ eV$ simuliert. In den Penningfallen ist der Speicherprozeß unabhängig vom Startwinkel; es werden sehr kleine Startwinkel von $\theta < 3°$ simuliert, um Kollisionen mit den Elektroden aufgrund des Zyklotronradius zu vermeiden. Der Konturplot zeigt in allen Elektrodenzwischenräumen kleine Bereiche, in denen die dort startenden niederenergetischen Elektronen gespeichert werden. Die außerhalb des inneren Spektrometervolumens liegenden Bereiche sind für die Untergrundproblematik nicht von Belang, da sie nur über komplizierte Folgeprozesse in den Flußschlauch gelangen können.

Die Möglichkeit, die hier eingeführten Speichervolumenkarten zu erstellen, wird im folgenden Kapitel noch sehr wichtig. Mit Hilfe dieser Darstellungsweise können die Wirkungen und der Nutzen von elektrischen Dipolelektroden und gekrümmten Magnetfeldern sehr anschaulich beschrieben werden.

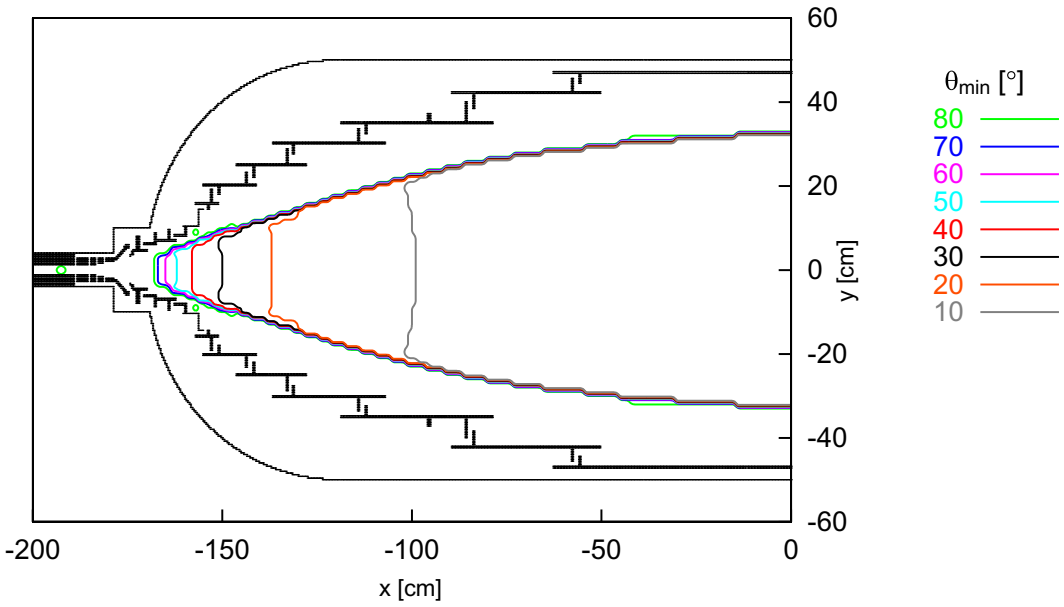

Abbildung 4.9: Speichervolumenkarte für Elektronen mit $E_{kin} = 512\ eV$. Dieses Konturbild zeigt das Ergebnis der Suche nach den magnetisch gespeicherten Elektronen im Mainzer Spektrometer. Es wurden Elektronen mit $E_{kin} = 512\ eV$ mit Startwinkeln zwischen $10° \leq \theta \leq 80°$ simuliert. Die Konturen zeigen die Bereiche, in denen ein Elektron mit entsprechendem minimalen Startwinkel θ_{min} starten muß, um gespeichert zu werden.

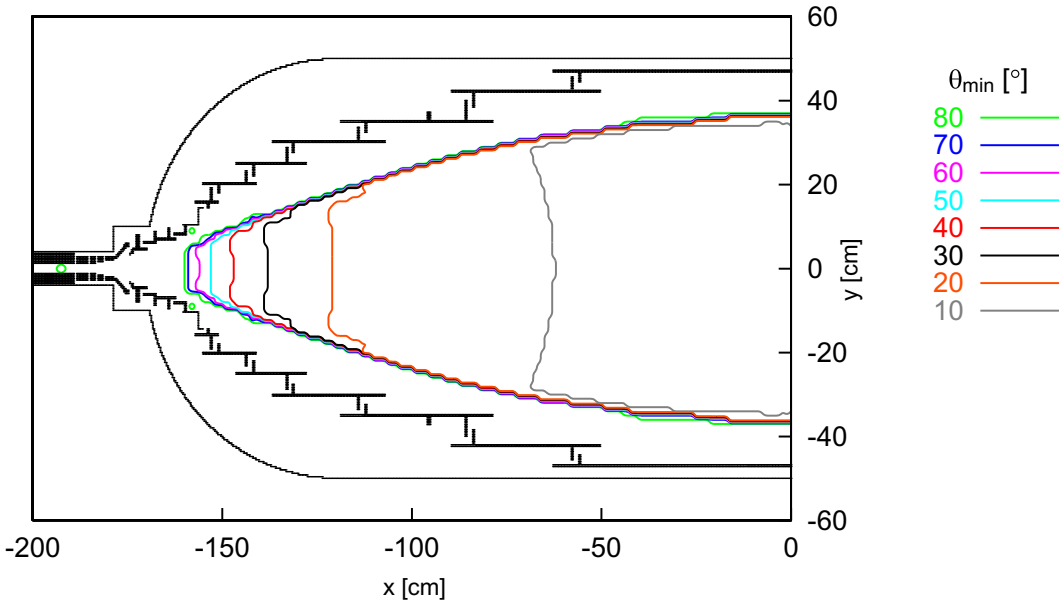

Abbildung 4.10: Speichervolumenkarte für Elektronen mit $E_{kin} = 256\ eV$. Dieses Konturbild zeigt das Ergebnis der Suche nach den magnetisch gespeicherten Elektronen im Mainzer Spektrometer. Es wurden Elektronen mit $E_{kin} = 256\ eV$ mit Startwinkeln zwischen $10° \leq \theta \leq 80°$ simuliert. Die Konturen zeigen die Bereiche, in denen ein Elektron mit entsprechendem minimalen Startwinkel θ_{min} starten muß, um gespeichert zu werden.

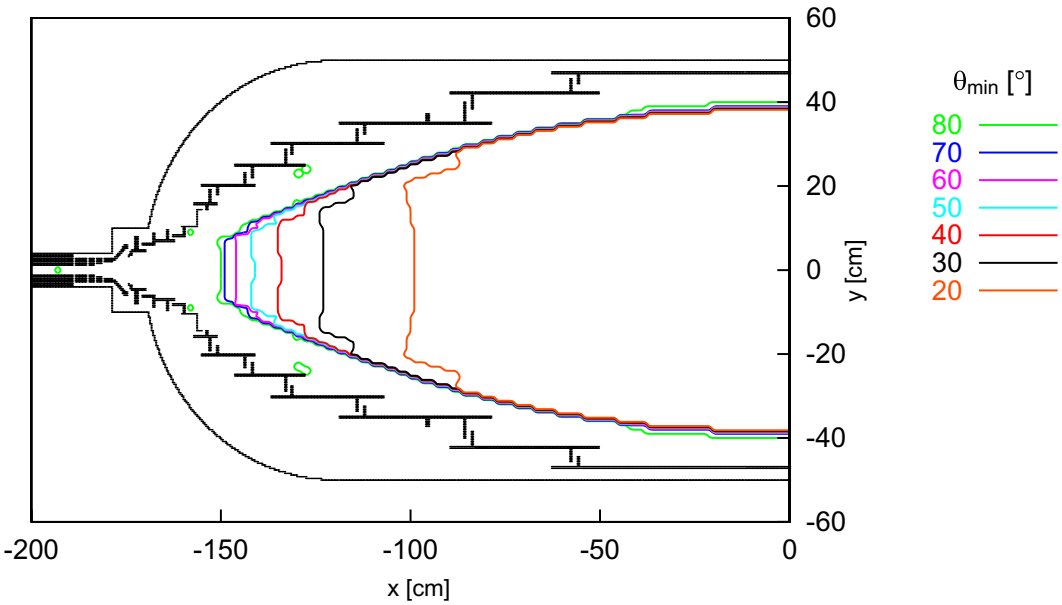

Abbildung 4.11: Speichervolumenkarte für Elektronen mit $E_{kin} = 128\ eV$. Dieses Konturbild zeigt das Ergebnis der Suche nach den magnetisch gespeicherten Elektronen im Mainzer Spektrometer. Es wurden Elektronen mit $E_{kin} = 128\ eV$ mit Startwinkeln zwischen $10° \leq \theta \leq 80°$ simuliert. Die Konturen zeigen die Bereiche, in denen ein Elektron mit entsprechendem minimalen Startwinkel θ_{min} starten muß, um gespeichert zu werden. Es konnten keine Elektronen mit $\theta < 20°$ gespeichert werden.

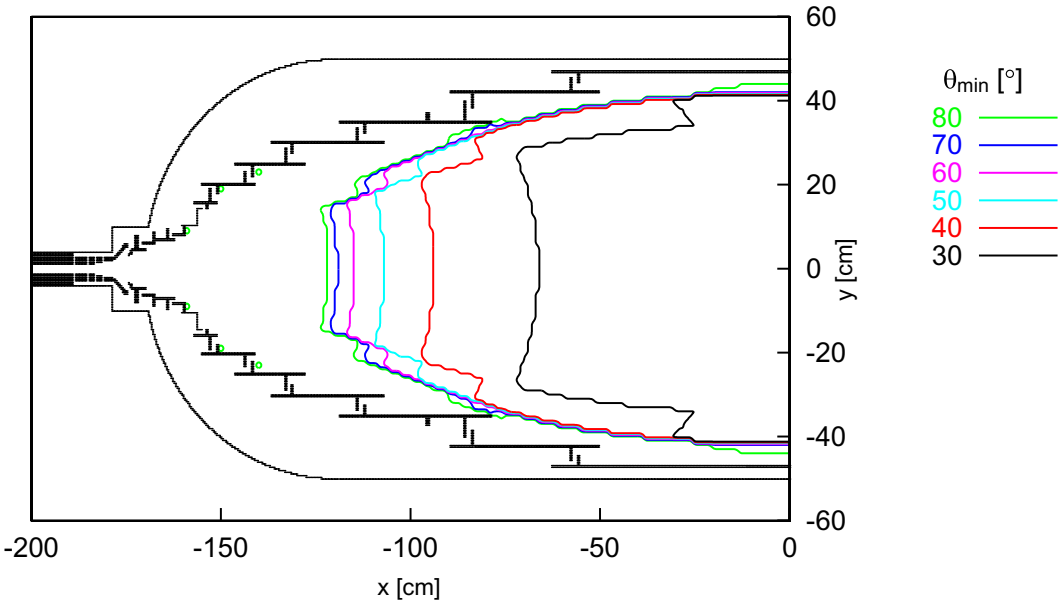

Abbildung 4.12: Speichervolumenkarte für Elektronen mit $E_{kin} = 32\ eV$. Dieses Konturbild zeigt das Ergebnis der Suche nach den magnetisch gespeicherten Elektronen im Mainzer Spektrometer. Es wurden Elektronen mit $E_{kin} = 32\ eV$ mit Startwinkeln zwischen $10° \leq \theta \leq 80°$ simuliert. Die Konturen zeigen die Bereiche, in denen ein Elektron mit entsprechendem minimalen Startwinkel θ_{min} starten muß, um gespeichert zu werden. Es konnten keine Elektronen mit $\theta < 30°$ gespeichert werden.

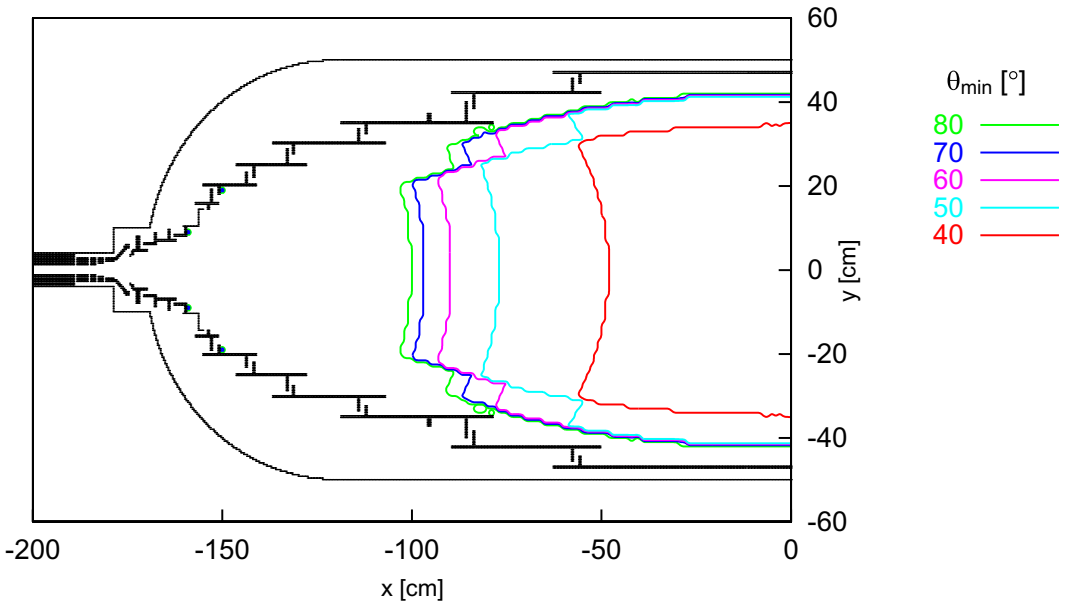

Abbildung 4.13: Speichervolumenkarte für Elektronen mit $E_{kin} = 16\ eV$. Dieses Konturbild zeigt das Ergebnis der Suche nach den magnetisch gespeicherten Elektronen im Mainzer Spektrometer. Es wurden Elektronen mit $E_{kin} = 16\ eV$ mit Startwinkeln zwischen $10° \leq \theta \leq 80°$ simuliert. Die Konturen zeigen die Bereiche, in denen ein Elektron mit entsprechendem minimalen Startwinkel θ_{min} starten muß, um gespeichert zu werden. Es konnten keine Elektronen mit $\theta < 40°$ gespeichert werden.

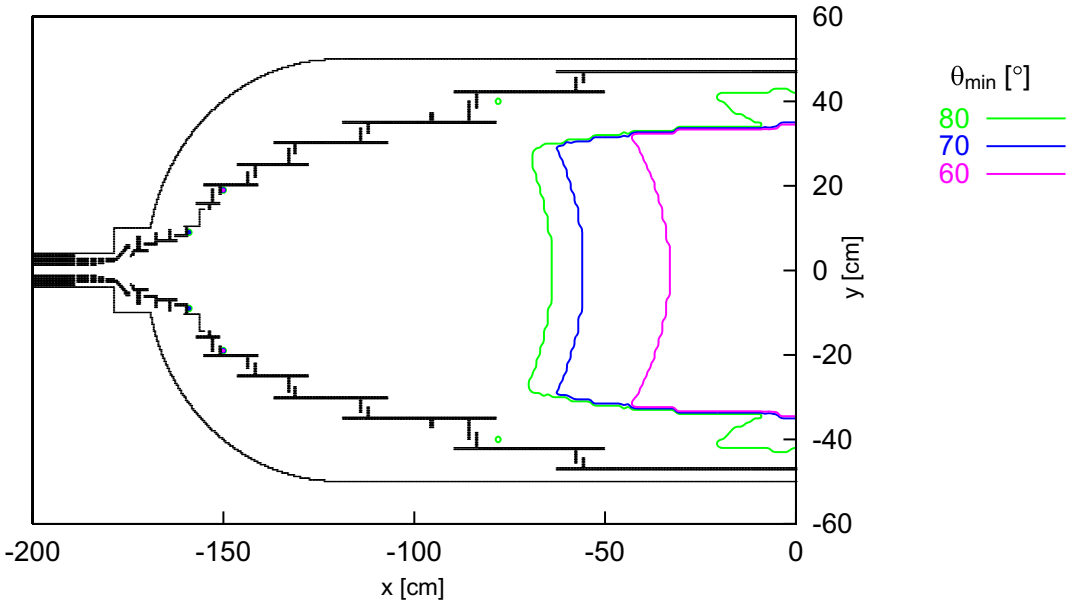

Abbildung 4.14: Speichervolumenkarte für Elektronen mit $E_{kin} = 8\ eV$. Dieses Konturbild zeigt das Ergebnis der Suche nach den magnetisch gespeicherten Elektronen im Mainzer Spektrometer. Es wurden Elektronen mit $E_{kin} = 8\ eV$ mit Startwinkeln zwischen $10° \leq \theta \leq 80°$ simuliert. Die Konturen zeigen die Bereiche, in denen ein Elektron mit entsprechendem minimalen Startwinkel θ_{min} starten muß, um gespeichert zu werden. Es konnten keine Elektronen mit $\theta < 60°$ gespeichert werden.

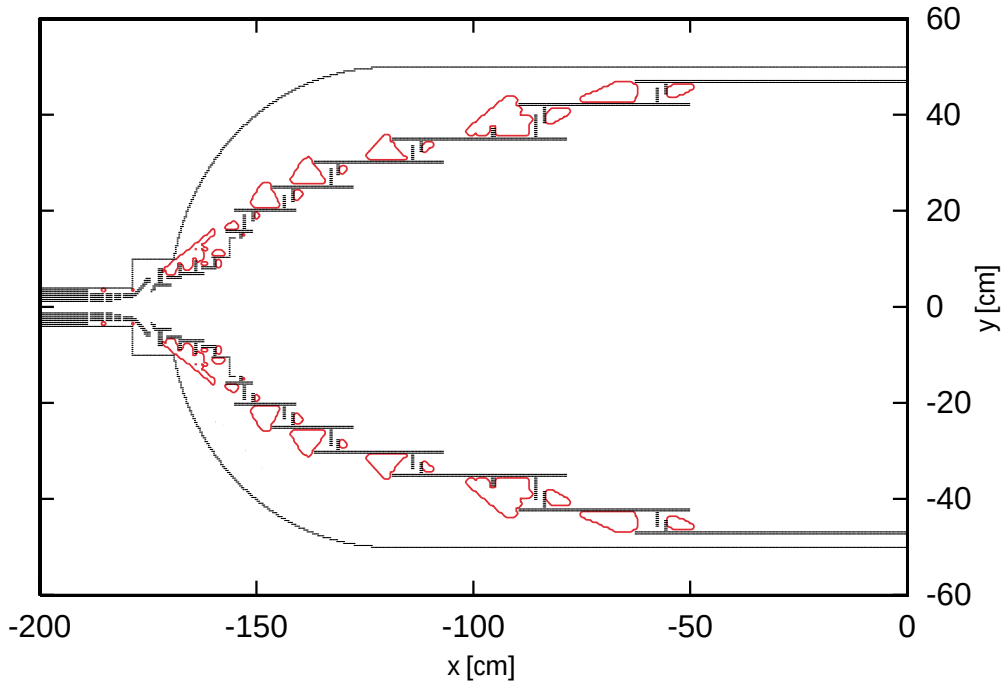

Abbildung 4.15: Speichervolumenkarte der kleinen Penningfallen. Gezeigt ist das Ergebnis der Suche nach den kleinen Penningfallen im Elektrodenzwischenraum. Es wurden niederenergetische Elektronen mit $E_{kin} < 1\ eV$ und einem Startwinkel $\theta < 3°$, der zur Berechnung verwendete Gitterabstand ist $g = 5\ mm$. Man erkennt sehr gut die kleinen Fallenbereiche zwischen den Elektroden. Die eingezeichneten Bereiche außerhalb der Elektroden, d.h. nicht im inneren Spektrometervolumen, sind nicht von Bedeutung, da dort entstehende Teilchen nur über Folgeprozesse zum Detektor gelangen können.

4.5 Untergrundprozesse

Abschließend soll noch auf die wirklichen Auswirkungen gespeicherter Teilchen eingegangen werden. Im allgemeinen geht man davon aus, daß sich die im Spektrometer gespeicherten Teilchen erhöhend auf die Untergrundzählrate auswirken. Es stellt sich hier die Frage, auf welche Weise wirken sich die unterschiedlichen gespeicherten Teilchen aus und welche Rolle spielen sie? Diese Frage soll hier nur qualitativ beantwortet werden, da die genaue Struktur des Untergrundspektrums und alle dazu beitragenden Effekte noch nicht vollständig verstanden sind und noch weitere Untersuchungen in diesem Zusammenhang notwendig und geplant sind.

Während der Diplomarbeit von J.-P. Schall [Sch01] wurden Messungen der Abhängigkeit der Untergrundzählrate vom herrschenden Spektrometerdruck durchgeführt. In der Analyse dieser Messungen wurde davon ausgegangen, daß sich die Form des Untergrundspektrums durch ein Polynom 2. Ordung beschreiben läßt. Unter den in [Sch01] gemachten Annahmen, könnte der gemessene Anstieg des Untergrundspektrums durch die Anwesenheit von gespeicherten Elektronen im magnetischen Flußschlauch, den der Detektor erfassen kann, erklärt werden. Es wurde festgestellt, daß der Anstieg schon von im zeitlichen Mittel nur einem gespeicherten Elektron

verursacht werden könnte.

Als maßgeblich verantwortlichen Prozeß, um gespeicherte Elektronen und Ionen zu erzeugen und in weiteren Prozessen einen Anstieg der Untergrundzählrate zu verursachen, betrachtet man die Bildung von Wasserstoffionen, die durch Stoß eines Elektrons mit einem Restgasmolekül entstehen können.

$$e_1^- + H_2 \longrightarrow e_1^- + H_2^+ + e_2^- \tag{4.3}$$

Der maximale Wirkungsquerschnitt für diesen Stoßprozeß liegt bei etwa 100 eV Elektronenenergie mit einer spezifischen Ionisation von 4 Ionisationen pro $cm \cdot mbar$. Durch die frühe Retardierung wird der größte Teil der niederenergetischen Zerfallselektronen $\bar{E}_{kin} \approx 6\ keV$ schon sehr früh auf der Quellseite des Spektrometers reflektiert. Die $2 \cdot 10^8$ Elektronen pro Sekunde, mit Energien zwischen 3 keV und 6 keV, können nur etwa 50 cm tief in das Spektrometervolumen eindringen. Man nimmt jetzt für die Elektronen eine mittlere Flugstrecke von 100 cm an und erhält bei einem Druck von 10^{-10} $mbar$ etwa 8 Ionisationen pro Sekunde [Sch01].

Elektronen mit Energien unterhalb des Analysierpotentials können nicht über die Analysierebene gelangen, hier werden Sekundärprozesse wichtig, die dazu führen, daß die Energie der Elektronen über die Analysierebene transportiert werden kann.

Elektronen, die durch β–Zerfall von Tritium im Spektrometervolumen entstehen, haben hohe kinetische Energien. Wie im letzten Abschnitt an den Speichervolumenkarten zu sehen war, können hochenergetische Zerfallselektronen in mehr als 50% des Spektrometervolumens permanent gespeichert werden.

Die positiven Ionen entstehen auf der Quellseite und können, aufgrund ihrer Ladung, den Detektor nicht direkt erreichen, es existieren aber indirekte Prozesse, die dies ermöglichen. Handelt es sich um negative Ionen, ist klar, daß diese das Analysierpotential nicht überwinden können und somit keine Rolle spielen. Wenn es sich aber um positive Ionen handelt, wie in Gleichung (4.3), werden diese zur Spektrometermitte hin beschleunigt und gelangen in den detektorseitigen Bereich. Auch wenn diese positiven gespeicherten Ionen den Detektor nicht erreichen, stellen sie dennoch eine Gefahr dar. Da die Ionen auf ihrem Weg ins Zentrum des Spektrometers durch das elektrische Potential beschleunigt werden, besitzen sie dort ausreichend kinetische Energie, um durch weitere Stoßprozesse ihre Energie an Elektronen abzugeben, die dann den Detektor erreichen können. Sollten die hierdurch entstandenen Elektronen den Detektor nicht erreichen, können sie magnetisch gespeichert werden und stehen für weitere Stoßprozesse zur Verfügung.

Elektronen, die in den kleinen Penningfallen (s. Abb. 4.1) zwischen den Elektroden gespeichert sind, können auf direktem Weg keinen Untergrund erzeugen. Jedes angestoßene Elektron kann nur maximal die Energie des stoßenden Elektrons haben und daher nicht aus der Falle entkommen. Wie [Sch01] beschreibt, kann ein gespeichertes

Elektron in einem Wasserstoffmolekül des Restgases einen Schwingungseigenzustand anregen.

$$e^- + H_2 \longrightarrow e^- + H_2^* \tag{4.4}$$

Dieses angeregte Molekül kann in einem zweiten Schritt ein Elektron einfangen.

$$e^- + H_2^* \longrightarrow H_2^{-*} \tag{4.5}$$

Durch einen weiteren Stoß mit einem Restgasmolekül kann das Elektron wieder freigesetzt werden.

$$H_2^{-*} \longrightarrow H_2 + e^- \tag{4.6}$$

Aufgrund seiner hohen Masse wird dieses Ion nicht mehr magnetisch geführt und folgt dem elektrischen Potential in die Spektrometermitte. Gibt es das Elektron hier ab, kann dieses zum Detektor gelangen und zur Untergrundzählrate beitragen. N. Titov [Tit01] konnte sogar zeigen, daß H^-–Ionen, die in diesen Fallen durch Elektroneneinfang erzeugt werden, direkt auf den Detektor gelangen können.

Werden geladene Teilchen im Spektrometervolumen permanent gespeichert, spielen Wirkungsquerschnitte von Stoßprozessen und Gasdruck praktisch keine Rolle mehr. In einem solchen Fall wird die Speicherzeit des geladenen Teilchens beliebig groß, d.h. es wird mit Sicherheit ein Stoßprozeß stattfinden. Die anfangs gestellte Frage, nach Wirkung und Rolle gespeicherter Teilchen, kann also beantwortet werden. Man kann davon ausgehen, daß gespeicherte Teilchen jeglicher Art zur Untergrundzählrate beitragen. Es sind aber nicht primär die gespeicherten Elektronen und Ionen, die die im Detektor registrierten Untergrundereignisse ausmachen, vielmehr spielen sie die Rolle eines Boten. Die durch retardierte Elektronen des β–Spektrums oder andere Prozesse entstandenen positiven Ionen erhalten Energie aus der Beschleunigung durch das Analysierpotential. Geben diese Ionen ihre Energie an gespeicherte Elektronen auf der Detektorseite ab, können diese auf den Detektor gelangen und führen zu Untergrundereignissen. Die Ursache für die primäre Erzeugung der H_2^- Ionen nach Gleichung (4.3) kann nur durch besseres Vakuum bekämpft werden, was durch großen Aufwand und auch dann nur um maximal eine Größenordnung möglich wäre. Es ist daher das primäre Ziel dieser Arbeit, mit Modifikationen am elektromagnetischen Aufbau des Spektrometers, die Zahl der gespeicherten Teilchen zu reduzieren.

Kapitel 5

Aufheben der Speicherbedingungen

Dieses Kapitel beschäftigt sich nun mit Methoden, die es ermöglichen, die Speicherbedingungen geladener Teilchen im Mainzer Spektrometer zu brechen. Dies soll durch Modifikationen am elektromagnetischen Aufbau des Spektrometers, in Form von Zusatzelektroden oder gekrümmten Magnetfeldern, realisiert werden. Ziel dieses Kapitels ist es, die Idee eines überlagerten elektrischen Dipolfeldes vorzustellen und seine Wirkung auf die gespeicherten Teilchen zu verstehen. Schließlich soll auch die Möglichkeit von gekrümmten Magnetfeldern betrachtet werden, um ihre Wirkung auf die Speicherbedingungen der geladenen Teilchen zu zeigen.

5.1 Diskussion einer elektrischen Dipolelektrode

Die nun folgenden Abschnitte werden sich der Idee eines elektrischen Dipolfeldes, sowie seiner Eigenschaften widmen. Die Zusammenhänge werden hier anhand eines idealisierten elektrischen Dipolfeldes, das dem Analysierpotential überlagert wird, diskutiert und sollen dann im darauf folgenden Abschnitt mit einer möglichen realen Dipolelektrode durch Simulationen überprüft werden.

5.1.1 Motivation einer elektrischen Dipolelektrode

Die Idee hinter einem elektrischen Dipolfeld und damit einer Dipolelektrode beruht auf dem Versuch, mit einem gleichmäßigen elektrischen Feld eine Vorzugsrichtung auszuzeichnen und damit die gespeicherten Teilchen abzulenken. Die stabilen Bahnen der gespeicherten Teilchen sollen auf diesem Weg zerstört werden, d.h. die Speicherbedingung wird gebrochen. Angriffspunkt des Dipolfeldes ist die transversale Driftbewegung des gespeicherten Teilchens, welche u. A. durch die $\vec{E} \times \vec{B}$–Drift (s. Gl. (2.19)) verursacht wird. Das überlagerte Dipolfeld soll nun einen konstanten Driftanteil in einer Richtung erzwingen. Mit dieser nicht azimutalen Driftbewegung

sollen die Teilchenbahnen so weit abgelenkt werden, daß sie in relativ kurzer Zeit, d.h. bevor sie mit dem Restgas wechselwirken können, aus dem Spektrometer entfernt werden.

5.1.2 Wirkung eines idealisierten elektrischen Dipolfeldes

Zuerst soll hier die Frage nach der Wirkung des elektrischen Dipolfeldes beantwortet werden. Im MAC–E–Filter sind elektrische und magnetische Felder vorhanden, die, wie schon in Abschnitt 2.2.2 beschrieben, zu einer azimutalen Driftbewegung führen, wenn beide Felder nicht parallel liegen. Die sog. $\vec{E} \times \vec{B}$–Drift (s. Gl. (2.19)) beschreibt genau diese Bewegung, sie ist azimutal, da das ganze System Zylindersymmetrie besitzt. Wenn nun das elektrische Dipolfeld überlagert wird, bekommt diese Driftbewegung eine neue Komponente in nur einer Richtung. Dieser Bewegungsanteil wird durch das Dipolfeld und die Richtung des Magnetfeldes definiert und wirkt senkrecht zu beiden. Mit einem Magnetfeld in x-Richtung und einem elektrischen Dipolfeld in z-Richtung ergibt sich eine Driftbewegung in y-Richtung. Die einfachste Annahme, daß das Teilchen zum Dipol hin oder davon weg beschleunigt wird trifft

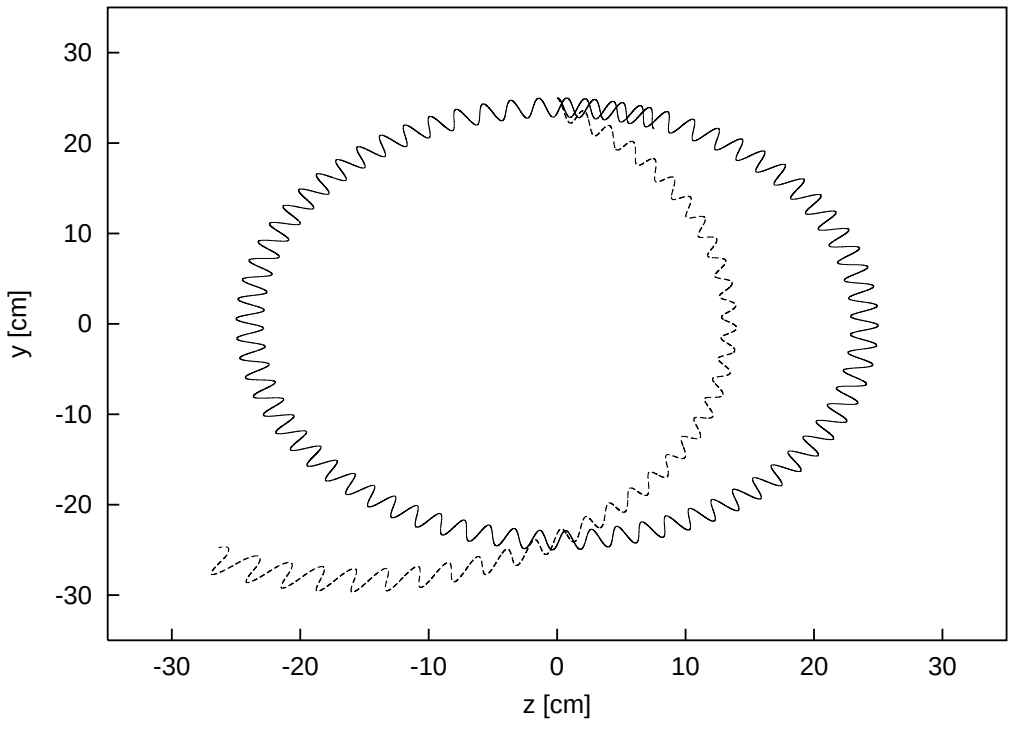

Abbildung 5.1: Wirkung eines elektrischen Dipolfeldes. Dargestellt ist die Projektion der von ADIPARK berechneten Flugbahn eines Elektrons auf die Analysierebene (z-y-Ebene), mit elektrischem Dipolfeld (gestrichelt) und ohne (durchgezogen). Das Elektron startet bei $(x, y, z) = (0, 25, 0)\ cm$ mit $E_{kin} = 100\ eV$ und $\theta = 70°$ relativ zur führenden Magnetfeldlinie. Das $-10\ V/m$ starke Dipolfeld wirkt in z-Richtung und das Elektron fliegt dadurch nach etwa $tof \approx 25\mu s$ gegen Elektrode E1 bei $R = 42\ cm$.

also nicht zu, da die Anwesenheit des Magnetfeldes das Teilchen auf eine dazu senkrechte Bewegung zwingt. Natürlich gilt das nur, solange das Dipolfeld nicht so stark wird, daß die Teilchen nicht mehr magnetisch geführt werden. Die Realisierung solch starker Dipolfelder stellt aber ein technisches Problem dar. Im Übrigen könnte dann auch nicht mehr die adiabatische Näherung angewendet werden.

In Abbildung 5.1 ist ein gespeichertes Elektron gezeigt, dessen Speicherbedingung durch ein Dipolfeld zerstört wird. Das Elektron startet in der Analysierebene mit einem radialen Abstand $y = 25\ cm$ zur Spektrometerachse mit $E_{kin} = 100\ eV$ und $\theta = 70°$ relativ zur führenden Magnetfeldlinie. Die geschlossene Flugbahn zeigt das gespeicherte Teilchen ohne elektrisches Dipolfeld. Wird ein $-10\ V/m$ starkes Dipolfeld in z-Richtung überlagert, so ergibt sich die gestrichelte Flugbahn und das Teilchen stößt nach $tof \approx 25\mu s$ gegen eine Elektrode. Der Effekt, daß das Teilchen durch ein in z-Richtung wirkendes elektrisches Dipolfeld durch die $\vec{E} \times \vec{B}$–Drift in y-Richtung abgelenkt wird, ist hier sehr schön zu sehen.

5.1.3 Stabilitätskriterien gespeicherter Teilchen

Obwohl die zusätzliche Dipoldrift eine in eine Richtung ausgezeichnete Drift bewirkt, wurde während der Untersuchungen der Wirkung des elektrischen Dipolfeldes auf ein gespeichertes Elektron festgestellt, daß gespeicherte Elektronen mit bestimmten Startparametern nicht einfach durch ein beliebiges Dipolfeld zu entfernen sind. Abbildung 5.2 zeigt ein Elektron, das trotz eingeschaltetem elektrischen Dipolfeld eine stabile, wenn auch verzerrte, Flugbahn hat. Die Vermutung liegt nahe, daß eine Komponente der Bewegung des gespeicherten Teilchens in dieser einfachen Überlegung nicht berücksichtigt ist. Diese Komponente scheint unabhängig vom elektrischen Dipolfeld zu sein und hält das Elektron auf der geschlossenen azimutalen Bahn, so daß es gespeichert bleibt.

Wir müssen in den Überlegungen zur Driftbewegung unter Anwesenheit eines elektrischen Dipolfeldes auch die magnetischen Driftterme: Gradienten und Krümmungsdrift miteinbeziehen. Diese, vom elektrischen Dipolfeld unabhängigen Flugbahnkorrekturen, führen dazu, daß das Teilchen gespeichert bleibt. Die magnetischen Driftterme sind stark genug, um dem nicht azimutalen Anteil der $\vec{E} \times \vec{B}$–Drift entgegenzuwirken. Einzig der nicht azimutale Anteil der $\vec{E} \times \vec{B}$–Drift lenkt das Elektron ab, der azimutale Anteil und die magnetische Gradienten- und Krümmungsdrift halten das Elektron auf der azimutalen Bahn. In Abbildung 5.3 sind die Driftgeschwindigkeiten des in Abbildung 5.2 dargestellten Elektrons als Funktion des Bahnradius aufgetragen, es sind die Driftanteile des nicht azimutalen Anteils der $\vec{E} \times \vec{B}$–Drift und der restlichen azimutalen Driftbewegungen gezeigt. In dieser Darstellung erkennt man, daß die azimutal wirkenden Driftterme (gestrichelt) sehr stark mit dem Radius ansteigen, während der nicht azimutale Anteil der $\vec{E} \times \vec{B}$–Drift konstant bleibt. Der starke Anstieg wird im wesentlichen von der magnetischen Gradienten- und Krümmungsdrift verursacht.

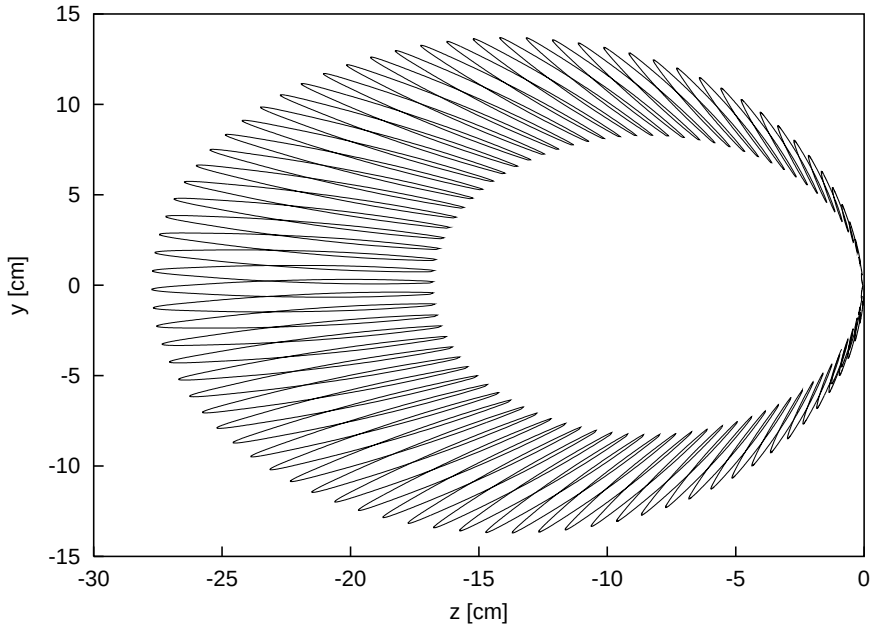

Abbildung 5.2: Gespeichertes Teilchen trotz elektrischem Dipolfeld. Hier ist die ADIPARK-Simulation eines Elektron zu sehen, das bei eingeschaltetem elektrischen Dipolfeld gespeichert bleibt. Das Elektron startet im Spektrometerzentrum mit $E_{kin} = 100\ eV$ und $\theta = 45°$ relativ zur führenden Magnetfeldlinie. Das Dipolfeld hat eine Stärke von $-10\ V/m$ in z-Richtung. Das Elektron wird vom Dipolfeld zwar sehr stark abgelenkt, es bleibt aber noch stabil gespeichert.

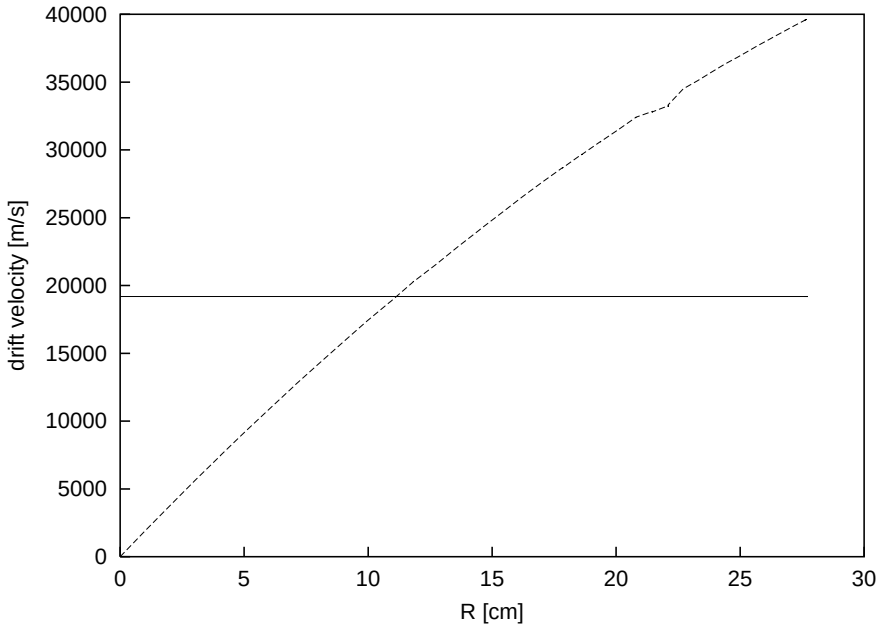

Abbildung 5.3: Radiale Abhängigkeit der Driftgeschwindigkeit. Aufgetragen sind die Driftgeschwindigkeiten des in Abbildung 5.2 gezeigten Elektrons als Funktion des Bahnradius. Die durchgezogene Linie zeigt die radiale Abhängigkeit des nicht azimutalen Anteils der $\vec{E} \times \vec{B}$–Drift (2.19). Gestrichelt ist die Summe aller azimutalen Driftbewegungen dargestellt. Die magnetische Driftgeschwindigkeit nimmt mit steigendem Radius sehr stark zu.

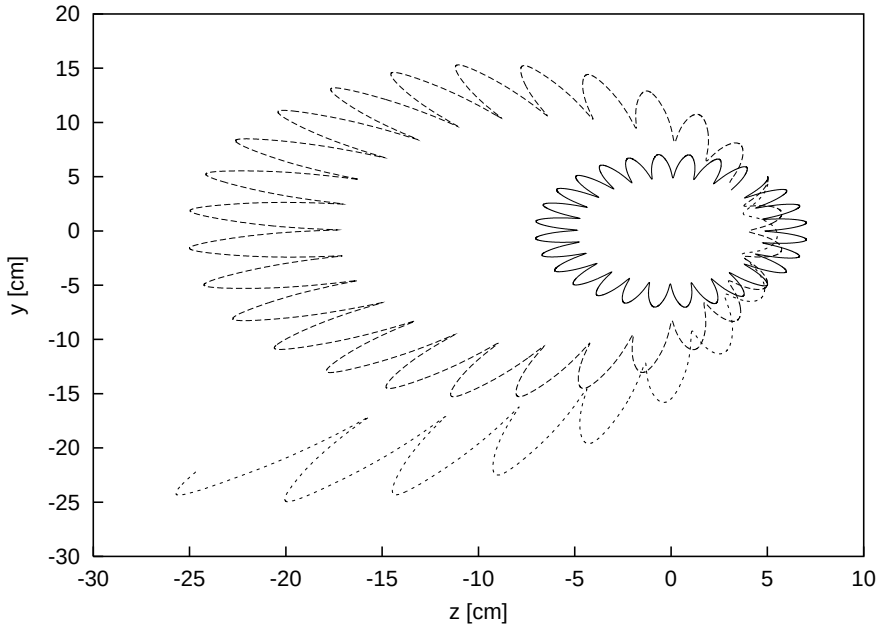

Abbildung 5.4: Teilchenbahnen von ADIPARK mit variiertem Dipolfeld. Dargestellt ist die Projektion der Flugbahn auf die Analysierebene (z-y-Ebene) eines Elektrons bei verschieden starken Dipolfeldern. Die Elektronen starten jeweils bei $(x, y, z) = (0, 5, 5)\ cm$ mit $E_{kin} = 500\ eV$ und $\theta = 45°$ zur führenden Magnetfeldlinie. Die durchgezogene Linie zeigt die Flugbahn ohne Dipolfeld, das Elektron ist gespeichert. Die gestrichelte Flugbahn zeigt die Verzerrung durch ein $-50\ V/m$ starkes Dipolfeld in z-Richtung, das Elektron bleibt gespeichert. Erst ein Dipolfeld von $-100\ V/m$ (grob gestrichelte Linie) läßt das Elektron nach $tof = 2.2\ \mu s$ gegen Elektrode E1 fliegen.

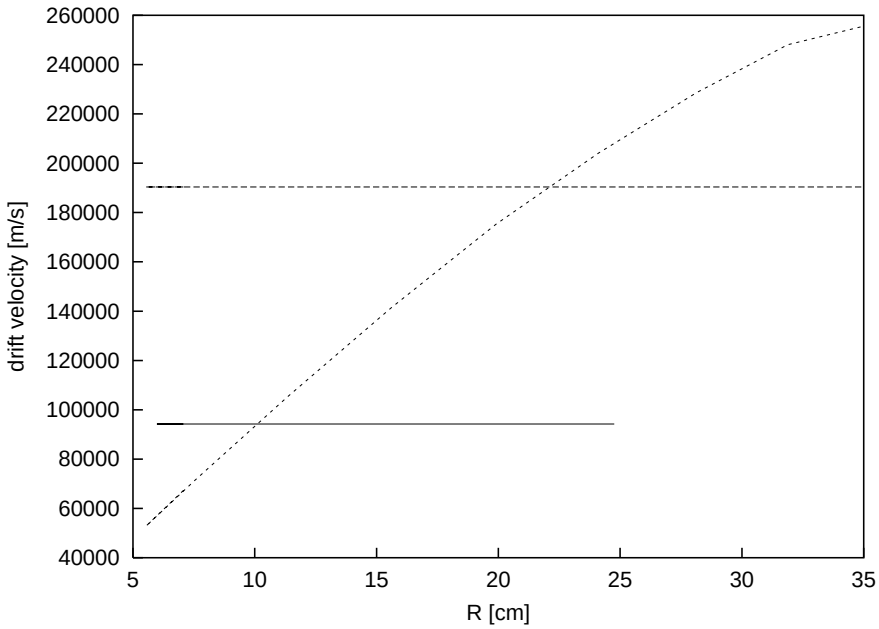

Abbildung 5.5: Radiale Abhängigkeit der Driftgeschwindigkeit. Analog zu Abbildung 5.3 sind hier die Driftgeschwindigkeiten der in Abbildung 5.4 dargestellten Elektronen als Funktion des Bahnradius zu sehen. Die durchgezogene Linie zeigt die radiale Abhängigkeit des nicht azimutalen Anteils der $\vec{E} \times \vec{B}$–Drift des Elektrons, das bei einem $-50\ V/m$ starken Dipolfeld gespeichert bleibt. Die gestrichelte Linie parallel dazu zeigt den nicht azimutalen Anteil der $\vec{E} \times \vec{B}$–Drift (2.19) des Elektrons, das durch das $-100\ V/m$ starke Dipolfeld gegen eine Elektrode fliegt. Für beide Elektronen ist die radiale Abhängigkeit der restlichen azimutalen Driftbewegungen (ansteigende Kurve) identisch.

Im Bereich $R < 11\ cm$ dominiert noch der nicht azimutale Anteil der $\vec{E} \times \vec{B}$–Drift, aber ab $R \approx 11\ cm$ sind es die magnetischen Driftterme, die die Teilchenbewegung am stärksten beeinflussen. Vereinfacht betrachtet kann man sagen, daß das gespeicherte Elektron vom nicht azimutalen Anteil der $\vec{E} \times \vec{B}$–Drift gegen eine Elektrode geführt werden muß solange dieser dominiert, da sonst die magnetischen Driftterme überwiegen und das Teilchen wieder auf eine geschlossene Bahn zwingen.

Die einzige Möglichkeit den nicht azimutalen Anteil der $\vec{E} \times \vec{B}$–Drift dominanter zu machen, ist ein stärkeres Dipolfeld[1]. Das Dipolfeld muß so stark werden, daß die azimutal wirkenden Driftanteile, im wesenlichen die magnetischen Driftterme, das Teilchen nicht mehr auf eine geschlossene Bahn führen können und es gegen eine Elektrode fliegt.

Um den steigenden Einfluß des nicht azimutalen Anteils der $\vec{E} \times \vec{B}$–Drift mit ansteigendem elektrischen Dipolfeld anschaulich zu zeigen, wird ein gespeichertes Elektron mit verschieden starken Dipolfeldern konfrontiert. Es wird ein Elektron verwendet, das sehr stabil gespeichert ist und bei schwachen Dipolfeldern gespeichert bleibt. Das Elektron startet bei $(x, y, z) = (0, 5, 5)\ cm$ mit $E_{kin} = 500\ eV$ und $\theta = 45°$ zur führenden Magnetfeldlinie. Es werden drei Simulationen durchgeführt, eine ohne Dipolfeld, eine mit $-50\ V/m$ starkem Dipolfeld und schließlich eine mit $-100\ V/m$ starkem Dipolfeld. Die Ergebnisse der Simulationen sind in Abbildung 5.4 zu sehen. Die durchgezogene Linie zeigt das gespeicherte Teilchen auf der gewohnten azimutalen Bahn ohne überlagertes elektrisches Dipolfeld. Das $-50\ V/m$ starke Dipolfeld führt zu einer stark verzerrten Flugbahn; das Elektron bleibt aber gespeichert. Erst das $-100\ V/m$ starke Dipolfeld reicht aus, um das Elektron innerhalb von $tof = 2.2\ \mu s$ gegen Elektrode E1 fliegen zu lassen.

Die Abbildung 5.5 zeigt die radiale Abhängigkeit der Driftgeschwindigkeiten für die in Abbildung 5.4 gezeigten Elektronen. Das Elektron ohne Dipolfeld ist nicht eingezeichnet, da hier der Radius immer konstant ist. Die stark ansteigende Kurve zeigt die radiale Abhängigkeit der azimutal wirkenden Driftanteile; sie ist unabhängig vom Dipolfeld für beide Elektronen identisch. Die durchgezogene Linie zeigt den nicht azimutalen Anteil der $\vec{E} \times \vec{B}$–Drift für das $-50\ V/m$ starke Dipolfeld: Hier erkennt man sehr schnell, warum das Elektron gespeichert bleibt, der nicht azimutale Anteil der $\vec{E} \times \vec{B}$–Drift überwiegt nur sehr kurz, bevor dann die magnetischen Driftterme die Bewegung bestimmen. Ganz anders sieht es bei einem $-100\ V/m$ starken Dipolfeld aus: Hier ist der nicht azimutale Anteil der $\vec{E} \times \vec{B}$–Drift stark genug, um die Teilchenbewegung lange zu bestimmen. Dies reicht aus, um das Elektron so nahe an eine Elektrode heran zu bringen, daß auch eine ansteigende Gradienten- oder Krümmungsdrift es nicht davon abhalten kann, gegen eine Elektrode zu fliegen.

Abschießend kann man festhalten, daß es ein elektrisches Dipolfeld ermöglicht, ge-

[1] Gleichung (2.19) skaliert mit $\frac{1}{B}$; der nicht azimutale Anteil der $\vec{E} \times \vec{B}$–Drift kann daher durch ein kleineres Magnetfeld verstärkt werden, allerdings skalieren die magnetische Krümmungs- und Gradientendrift auch mit $\frac{1}{B}$.

speicherte Teilchen aus dem Spektrometer zu entfernen. Es muß allerdings folgende Bedingung erfüllt sein: Das Dipolfeld muß so stark gewählt werden, daß der nicht azimutale Anteil der $\vec{E} \times \vec{B}$–Drift eines Teilchens ausreichend lange über die magnetischen Driftterme dominiert. Man kann hier keinen Grenzwert oder Sollwert angeben, da die relativen Driftanteile stark von der jeweiligen Startbedingung abhängen. Es bieten sich zur Untersuchung einer großen Anzahl von Startbedingungen die Speichervolumenkarten aus Abschnitt 4.3 an. Betont werden muß hier, daß bei starken elektrischen Dipolfeldern die Retardierungsenergie in der Analysierebene des MAC–E–Filters nicht mehr konstant ist, d.h. das Entfernen von gespeicherten Teilchen mittels elektrischem Dipolfeld kann nur in Meßpausen durchgeführt werden. Allerdings werden nur sehr kurze Meßpausen im Bereich von 10^{-3} s benötigt, um gespeicherte Teilchen zu entfernen.

5.2 Simulationen mit Dipolelektrode

Die Erkenntnisse über die Wirkung eines überlagerten idealisierten elektrischen Dipolfeldes sollen jetzt auf eine realisierbare Dipolelektrode übertragen werden. Diese Untersuchungen werden durch Simulation einer zusätzlichen Elektrode im Mainzer Spektrometer durchgeführt.

5.2.1 Simulation einer realen Dipolelektrode

Zur Realisierung eines elektrischen Dipolfeldes in der Spektrometermitte genügt es, eine zylindrische Halbschalenelektrode über der Zentralelektrode E0 einzufügen. In Abbildung 5.6 ist die Simulation dieser Dipolelektrode mit SIMION gezeigt.

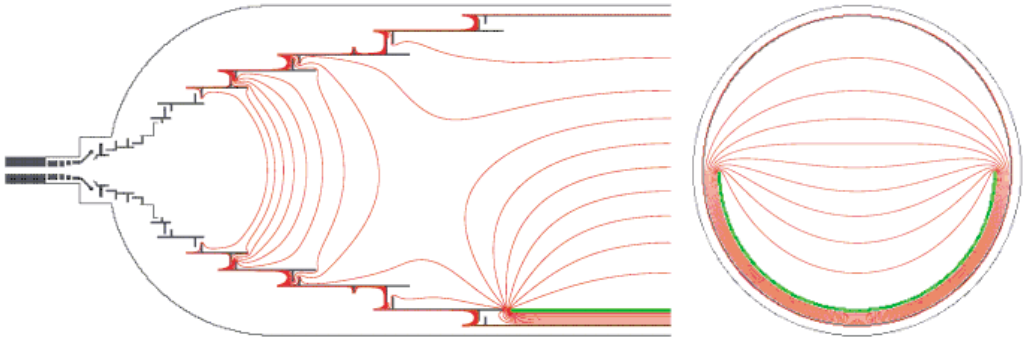

Abbildung 5.6: Dipolhalbschale im Mainzer Spektrometer. Hier ist die Seitenansicht einer Spektrometerhälfte und der Schnitt durch die Analysierebene der Mainzer Elektrodenkonfiguration mit Dipolhalbschale über der zentralen Elektrode E0 gezeigt. In grün ist die zylindrische Halbschalenelektrode eingezeichnet, die ein um $\Delta U = 100\,V$ positiveres Potential als die Elektrode E0 ($U_{E0} = -18690\,V$) hat. In rot sind die Äquipotentiallinien von $U = -18590\,V$ bis $U = -18690\,V$ in 10 V Schritten eingezeichnet.

Die zusätzliche Elektrode liegt konzentrisch zur Zentralelektrode E0. In dieser ersten Simulation wird eine halbschalenförmige Elektrode verwendet, weitere Elektrodenentwürfe werden in [Mue02] untersucht. Die Elektrode E0 hat ein Potential von $U_{E0} = -18690\,V$. Um ein Dipolfeld von etwa 100 V/m zu erzeugen, wird die Dipolelektrode auf ein Potential von $U_{Dipol} = -18590\,V$ gelegt, was um $\Delta U = 100\,V$ positiver ist als U_{E0}. In Abbildung 5.6 sind auch die Äquipotentiallinien von $U = -18590\,V$ bis $U = -18690\,V$ in 10 V Schritten eingezeichnet. Man erkennt, daß im Spektrometerzentrum ein ideales, in y-Richtung gerichtetes, Dipolfeld vorhanden ist, das an den Rändern verzerrt ist.

Jetzt soll untersucht werden, ob die hier entworfene Dipolelektrode wie erwartet auf gespeicherte Elektronen wirkt. Es wird ein Elektron bei Position $(x, y, z) =$

$(0, 0, 15)$ cm mit $E_{kin} = 500$ eV und einem Startwinkel $\theta = 55°$ relativ zur führenden Magnetfeldlinie gestartet. Im Mainzer Spektrometer ohne Dipolelektrode wird dieses Elektron gespeichert; mit dem idealisierten Dipolfeld und 100 V/m Feldstärke stößt das Elektron nach $tof \approx 2.7$ μs gegen Elektrode E0 bei $R = 47$ cm. Unter Einfluß der realen Dipolelektrode stößt das Elektron nach neun Reflektionen und $tof \approx 2.2$ μs gegen Elektrode E1 bei $R = 42$ cm. In Abbildung 5.7 sind die Flugbahndaten dieses Elektrons mit und ohne Dipolfelder gegenübergestellt. Das durch die reale Dipolelektrode erzeugte elektrische Dipolfeld wirkt wie erwartet und ist stark genug, die Speicherbedingung des Elektrons zu brechen. Vergleicht man das idealisierte Dipolfeld mit dem der realen Dipolelektrode, erkennt man sofort, daß beide eine vergleichbare Wirkung auf das Elektron haben. Es ergibt sich eine kleine Abweichung, die man auf die kompliziertere Form des elektrischen Feldes, besonders die Verzerrungen im Randbereich der Dipolhalbschale, zurückführen muß.

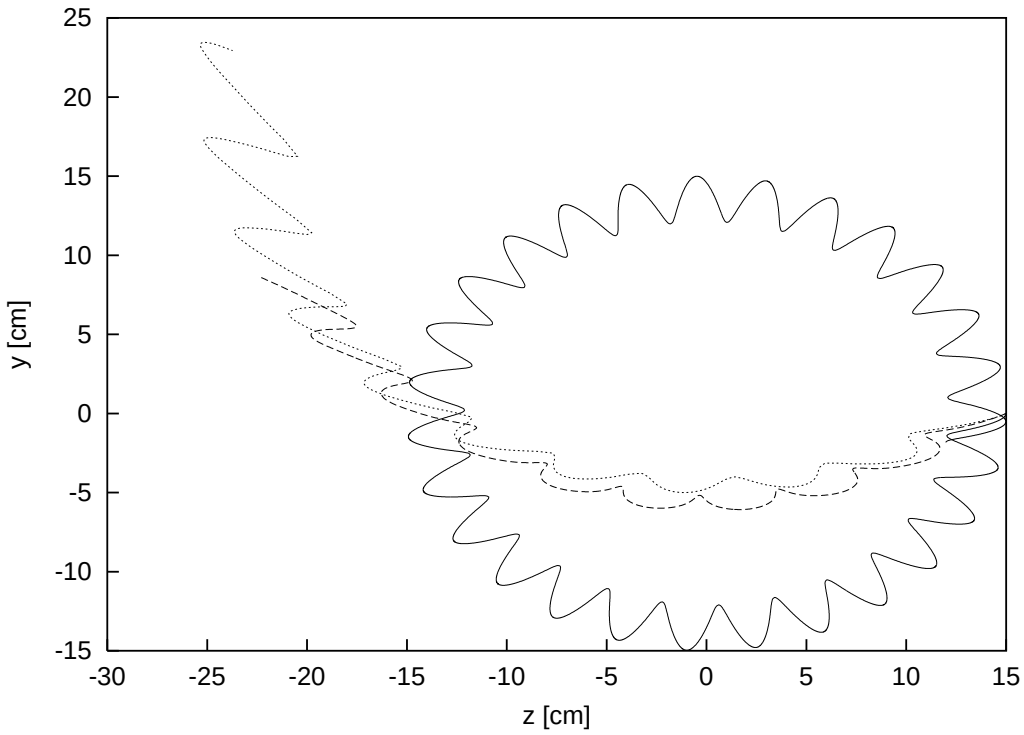

Abbildung 5.7: ADIPARK-Simulation der Wirkung verschiedener Dipolfelder. Projektion der Flugbahn eines Elektrons auf die z-y-Ebene, das bei Position $(x, y, z) = (0, 0, 15)$ cm mit $E_{kin} = 500eV$ und einem Startwinkel von $\theta = 55°$ relativ zur führenden Magnetfeldlinie startet. Die Flugbahn des Elektrons wurde ohne Dipolfeld (durchgezogene Linie), mit 100 V/m starkem idealisierten Dipolfeld (gepunktet) in y-Richtung und schließlich mit der realen Dipolelektrode (gestrichelt) simuliert.

5.2.2 Speichervolumenkarten mit Dipolelektrode

Im letzten Abschnitt wurde gezeigt, daß eine reale Dipolelektrode wie erwartet auf ein gespeichertes Elektron wirkt. Es stellt sich jetzt die Frage, ob es auch hier noch gespeicherte Elektronen mit bestimmten Startparametern gibt, die nicht durch das Dipolfeld entfernt werden.

Die in Abschnitt 4.3 vorgestellte Methode zur Berechnung von Speichervolumenkarten ist hier hilfreich, wenn man nach den weiterhin gespeicherten Elektronen suchen möchte. Zu diesem Zweck wird eine vergleichbare Berechnung wie in Abschnitt 4.3 nochmals durchgeführt, allerdings wird jetzt die elektrische Potentialkarte mit Dipolhalbschale verwendet. Da es sich jetzt nicht mehr um ein rein zylindersymmetrisches elektrisches Feld handelt, kann man nicht mehr einfach davon ausgehen, daß das Elektron gespeichert ist, wenn es mindestens zweimal reflektiert wurde. Das Trapping–Modul wird so konfiguriert, daß erst Elektronen, die mehr als 500 mal reflektiert werden oder deren Flugzeit $tof > 30\ \mu s$ ist, als gespeichert gewertet werden. Da die Berechnungszeit jetzt von Stunden auf Tage ansteigt, wird nur ein Gitter mit $g = 2\ cm$ Abstand berechnet. Es werden nur Untersuchungen in der x-y-Ebene durchgeführt, da die größten Effekte in Richtung des Dipolfeldes zu erwarten sind.

Das Ergebnis dieser Berechnung ist in den Abbildungen 5.8 bis 5.12 zu sehen. Zum Vergleich muß man die Ergebnisse aus Kapitel 4 betrachten (siehe Abbildungen 4.9 bis 4.14). Es wurde hier kein Bild mit $E_{kin} = 8\ eV$ abgedruckt, da dort kein Elektron mehr gespeichert werden konnte. Im Vergleich erkennt man, daß das Dipolfeld die Speicherbedingung für einen Großteil der gespeicherten Teilchen zerstört.

In Abbildung 5.8 erkennt man einen Bereich um die Dipolelektrode, der vollkommen von gespeicherten Teilchen gereinigt wurde. Das separate Gebiet im Spektrometereingang deutet darauf hin, daß die hier startenden Elektronen weiterhin gespeichert sind, diese besitzen einen magnetischen Reflektionspunkt und zur Spektrometermitte hin werden sie vom elektrischen Potential reflektiert. Die zackige, zerfranste Form der Kontur ist ein Hinweis, daß an diesen Stellen die oben vorgegebenen Parameter zur Unterscheidung zwischen gespeichert und nicht gespeichert noch nicht ausreichen. Die aufräumende Wirkung des Dipolfeldes wird bei den niedrigeren Energien von $E_{kin} = 256\ eV$ und $E_{kin} = 128\ eV$ (Abb. 5.9 und 5.10) noch deutlicher. Bei $E_{kin} = 128\ eV$ (Abb. 5.10) sind nahezu alle im mittleren Spektrometervolumen gespeicherten Elektronen entfernt worden. Der Bereich im Spektrometereingang bleibt unbeeinflußt.

Geht man zu noch niedrigeren Energien, verschwinden, wie auch im dipolfreien Fall (Abb. 4.12), die Speicherbereiche im Spektrometereingang. In den Abbildungen 5.11 und 5.12 erkennt man einen neuen Bereich, in dem gespeicherte Elektronen entstehen können. Dieser Bereich korrespondiert mit einer Sattelfläche des elektrischen Potentials, einem Potentialminimum, das durch die Dipolelektrode erzeugt wird und das in Abbildung 5.6 zu sehen ist. Die Elektronen sehen dieses Potentialmaximum als

Barriere und können daran reflektiert werden. Auch dieser Speicherbereich wird mit der Startenergie der Elektronen kleiner. Bei einer Energie von $E_{kin} = 8\ eV$ ist kein einziges gespeichertes Elektron mehr gefunden worden.

Man kann aus diesen Ergebnissen schlußfolgern, daß ein Dipolfeld einen sehr großen Teil der gespeicherten Elektronen aus dem Spektrometervolumen entfernt. Da hier der Effekt nur anhand einer Dipolkonfiguration und auch nur in einer Ebene gezeigt wurde, bleibt offen, wie die optimale Form einer Dipolelektrode aussehen muß und durch welche Feldstärke man die besten Ergebnisse erreicht. Ausführliche Untersuchungen dieser Problematik werden in [Mue02] durchgeführt und auch durch Testmessungen mit einem modifizierten Mainzer Spektrometer überprüft.

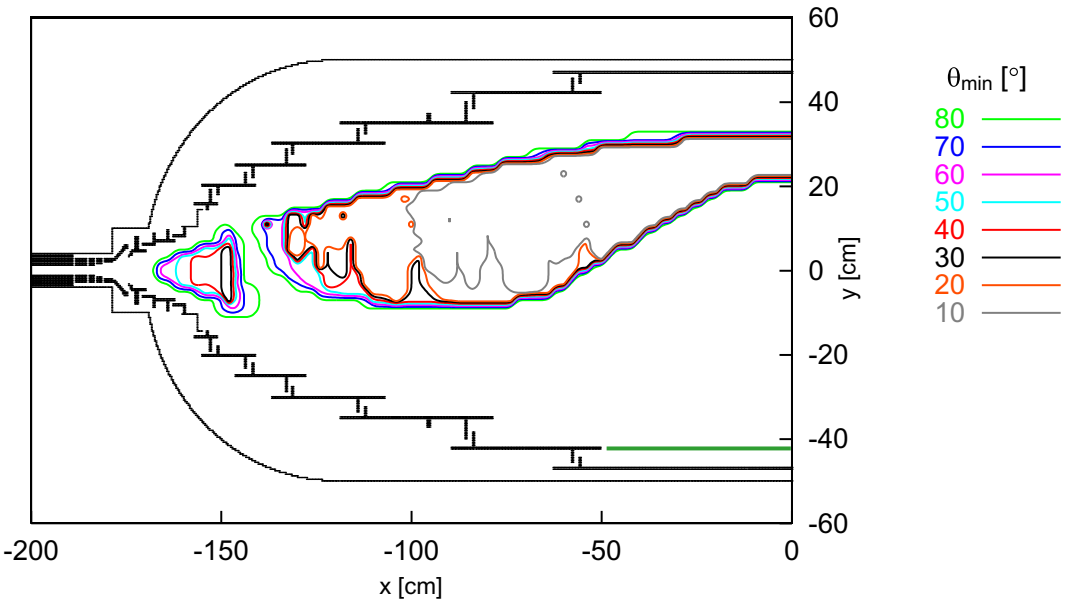

Abbildung 5.8: Speichervolumenkarte für Elektronen im Dipolfeld mit $E_{kin} = 512\ eV$. Ergebnis der Suche nach den magnetisch gespeicherten Elektronen im Mainzer Spektrometer mit Dipolhalbschale. Es wurden Elektronen mit $E_{kin} = 512\ eV$ mit Startwinkeln zwischen $10° \leq \theta \leq 80°$ simuliert. Die Konturen zeigen die Bereiche, in denen ein Elektron mit entsprechendem minimalen Startwinkel θ_{min} starten muß, um gespeichert zu werden. Die Dipolelektrode ist bei $y = -42\ cm$ in grün eingezeichnet.

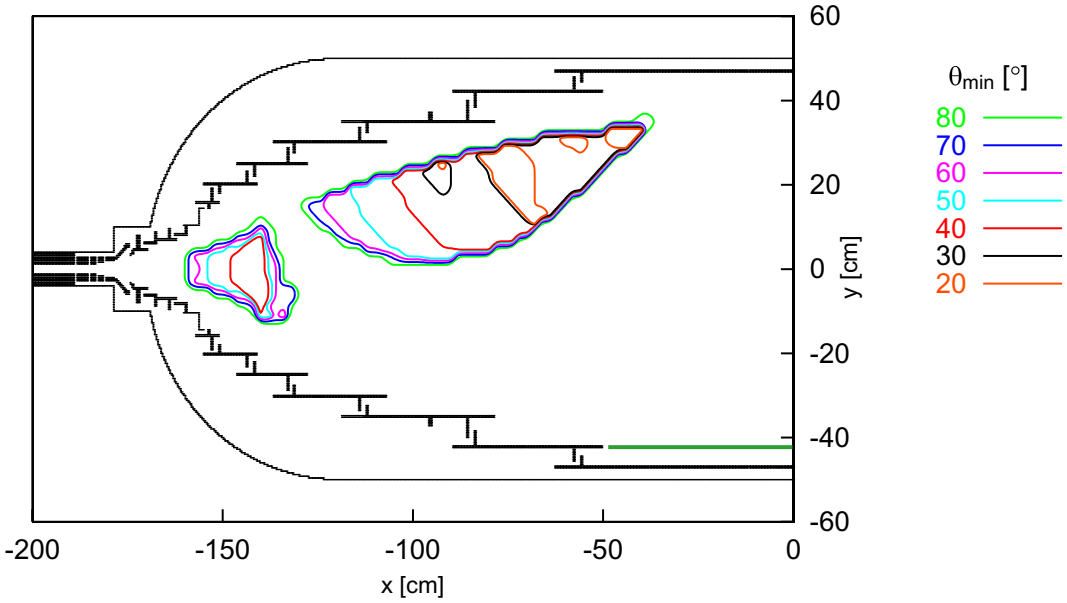

Abbildung 5.9: Speichervolumenkarte für Elektronen im Dipolfeld mit E_{kin} = 256 eV. Analog zu Abbildung 5.8 wurden hier Elektronen mit $E_{kin} = 256\ eV$ und Startwinkeln zwischen $10° \leq \theta \leq 80°$ simuliert. Die Konturen zeigen die Bereiche, in denen ein Elektron mit entsprechendem minimalen Startwinkel θ_{min} starten muß, um gespeichert zu werden. Es konnten keine Elektronen mit $\theta < 20°$ gespeichert werden.

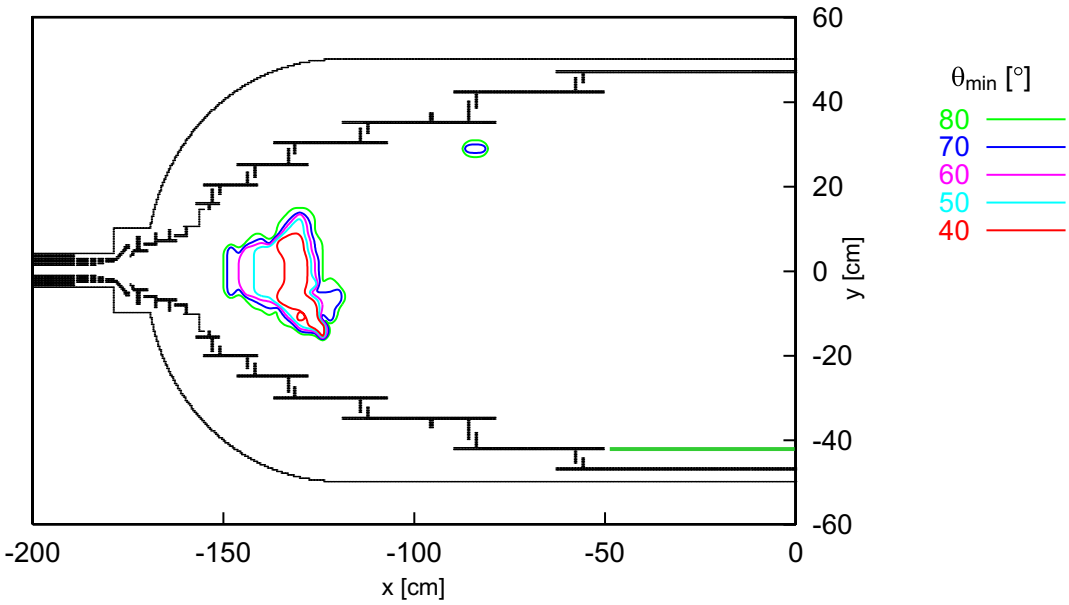

Abbildung 5.10: Speichervolumenkarte für Elektronen im Dipolfeld mit E_{kin} = 128 eV. Analog zu Abbildung 5.9 wurden hier Elektronen mit $E_{kin} = 128\ eV$ und Startwinkeln zwischen $10° \leq \theta \leq 80°$ simuliert. Die Konturen zeigen die Bereiche, in denen ein Elektron mit entsprechendem minimalen Startwinkel θ_{min} starten muß, um gespeichert zu werden. Es konnten keine Elektronen mit $\theta < 40°$ gespeichert werden.

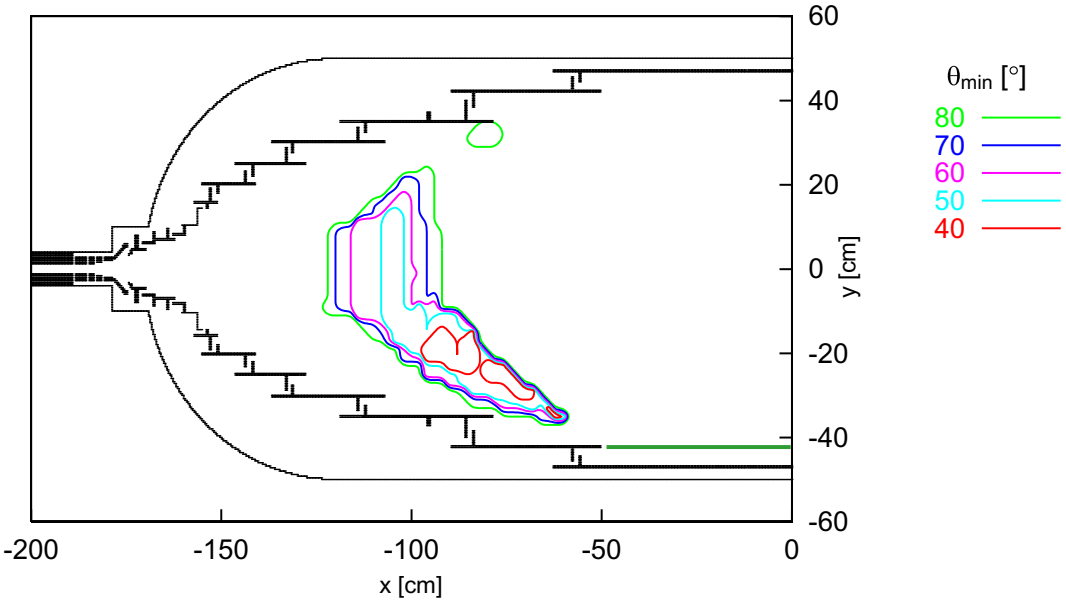

Abbildung 5.11: Speichervolumenkarte für Elektronen im Dipolfeld mit $E_{kin} = 32\ eV$. Analog zu Abbildung 5.10 wurden hier Elektronen mit $E_{kin} = 32\ eV$ und Startwinkeln zwischen $10° \leq \theta \leq 80°$ simuliert. Die Konturen zeigen die Bereiche, in denen ein Elektron mit entsprechendem minimalen Startwinkel θ_{min} starten muß, um gespeichert zu werden. Es konnten keine Elektronen mit $\theta < 40°$ gespeichert werden.

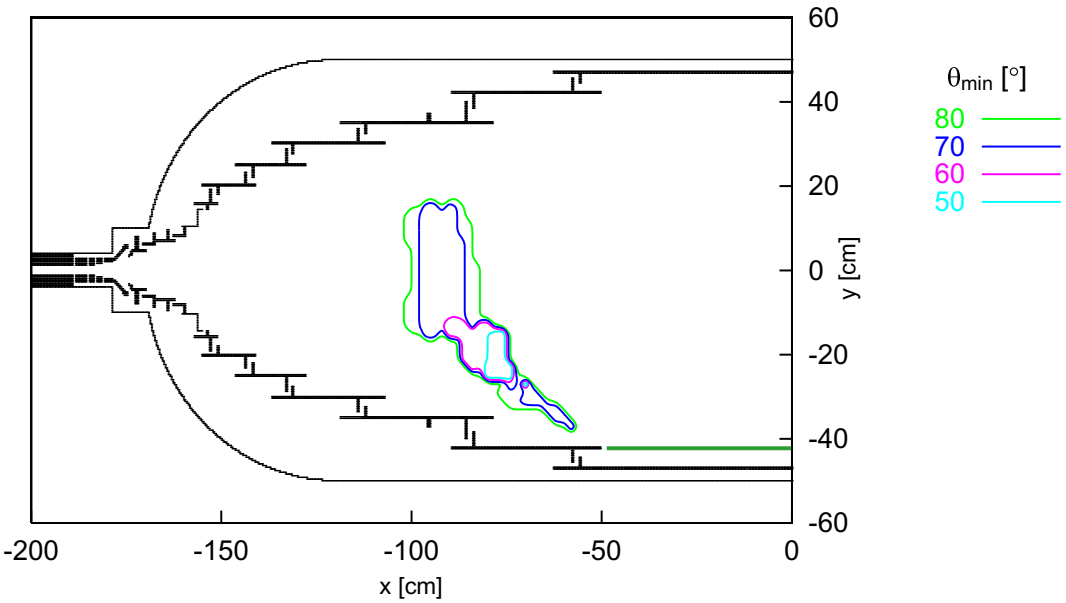

Abbildung 5.12: Speichervolumenkarte für Elektronen im Dipolfeld mit $E_{kin} = 16\ eV$. Analog zu Abbildung 5.11 wurden hier Elektronen mit $E_{kin} = 16\ eV$ und Startwinkeln zwischen $10° \leq \theta \leq 80°$ simuliert. Die Konturen zeigen die Bereiche, in denen ein Elektron mit entsprechendem minimalen Startwinkel θ_{min} starten muß, um gespeichert zu werden. Es konnten keine Elektronen mit $\theta < 50°$ gespeichert werden.

5.3 Betrachtung eines gekrümmten Magnetfeldes

In diesem Abschnitt soll die Idee, Speicherbedingungen gespeicherter Teilchen mit Hilfe von gekrümmten Magnetfeldern zu brechen, vorgestellt werden. Die Möglichkeiten, die gekrümmte Magnetfelder bieten, werden am Beispiel von gekippten Spulenkonfigurationen des Mainzer Spektrometers erläutert. Die Wirkung des gekrümmten Magnetfeldes auf gespeicherte Teilchen wird durch ADIPARK-Simulationen überprüft.

5.3.1 Motivation für ein gekrümmtes Magnetfeld

Mit gekrümmten Magnetfeldern versucht man, sich die magnetische Krümmungsdrift zu Nutze zu machen, um gespeicherte Teilchen aus dem Spektrometer zu entfernen. Analog zu der Wirkung eines elektrischen Dipolfeldes sollen die stabilen Bahnen gespeicherten Teilchen zerstört werden. Die magnetische Krümmungsdrift (s. Gl. (2.21)) verursacht eine Driftbewegung senkrecht zur Magnetfeldlinie und deren Krümmungsradius. Durch gekrümmte Magnetfelder soll eine definierte Krümmung in einer Richtung vorgegeben werden, die einen konstanten Anteil der Driftbewegung des gespeicherten Teilchens in einer Richtung verursacht. Wie im letzten Abschnitt soll dadurch die Bahn des gespeicherten Teilchens so weit ausgedehnt werden, bis dieses schließlich gegen eine Elektrode stößt. Auch hier ist es im Hinblick auf Untergrundereignisse wichtig, daß das Teilchen innerhalb kurzer Zeit gegen eine Elektrode fliegt, also bevor es mit Restgasmolekülen wechselwirken kann.

5.3.2 Simulation mit gekrümmten Magnetfeldlinien

Zur Simulation von gekrümmten Magnetfeldern, wird eine neue Magnetfeldkarte benötigt. Diese Magnetfeldkarte wird auf Basis der bestehenden Spulenkonfiguration für das Mainzer Spektrometer erstellt (s. Abb. 2.5). Zur Krümmung des Magnetfeldes werden die Spektrometersolenoide um jeweils 10° gegen die Spektrometerachse verkippt. Mit Hilfe des Programmes Bfield3D von B. Flatt kann daraus eine neue dreidimensionale Magnetfeldkarte für SIMION und ADIPARK erstellt werden. Die neue Spulenkonfiguration mit eingezeichneten Magnetfeldlinien ist in Abbildung 5.13 gezeigt. Auf die notwendigen Modifikationen an den Elektroden E8 bis E13 (vgl. Abb. 2.4) wird hier nicht eingegangen, da im wesentlichen der unveränderte zentrale Bereich des Spektrometers für die Simulationen wichtig ist.

Die zu erwartende Driftgeschwindigkeit im Vergleich zum Dipolfeld soll hier durch folgende grobe Abschätzung berechnet werden. Es werden die Beträge der Krümmungsdrift und der $\vec{E} \times \vec{B}$–Drift betrachtet.

$$|\vec{u}_c| = \frac{2E_{\|}}{eR^2B^2}|\vec{R} \times \vec{B}| = \frac{2E_{\|}}{eRB} \tag{5.1}$$

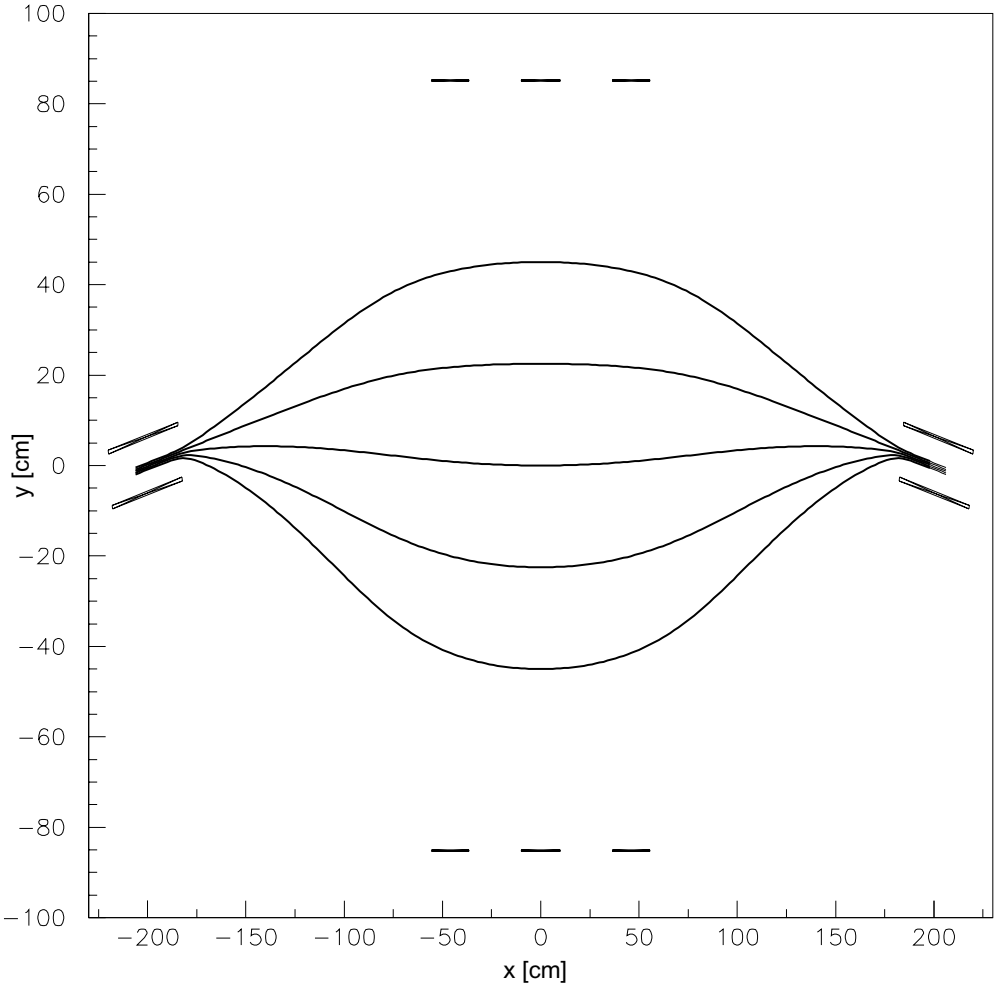

Abbildung 5.13: Feldlinien des gekrümmten Magnetfeldes. Dargestellt ist die Spulenkonfiguration des Mainzer Spektrometers. Links und rechts die, um je 10° gekippten Hauptsolenoide, in der Mitte drei Luftspulen zur Feldkorrektur. Eingezeichnet sind die Magnetfeldlinien mit Radien von $R = \pm 45\ cm$, $R = \pm 22.5\ cm$ und $R = 0\ cm$ in der Analysierebene ($x = 0\ cm$).

$$|\vec{u}_E| = \frac{|\vec{E} \times \vec{B}|}{B^2} = \frac{E}{B} \tag{5.2}$$

Der Krümmungsradius der Magnetfeldlinien, durch die um je 10° gekippten Spulen beträgt in etwa $R \approx 11\ m$. Ein Elektron mit $E_{kin} = 100\ eV$ und $\theta = 35°$ hat eine longitudinale kinetische Energie von $E_{||} \approx 67\ eV$. Der Vergleich wird anhand des Verhältnisses der beiden Komponenten durchgeführt.

$$1 = \frac{|\vec{u}_E|}{|\vec{u}_c|} = \frac{eRE}{2E_{||}} \Rightarrow E = \frac{2E_{||}}{eR} \approx 12\ V/m \tag{5.3}$$

Man erwartet also, daß ein Dipolfeld von $E \approx 12\ V/m$ äquvalent zu einer Gesamtkrümmung von 20° ist.

Zur Untersuchung der Auswirkungen auf ein gespeichertes Elektron wird es bei $(x, y, z) = (0, -5, 5)\ cm$ mit $E_{kin} = 100\ eV$ und einem Startwinkel von $\theta = 35°$

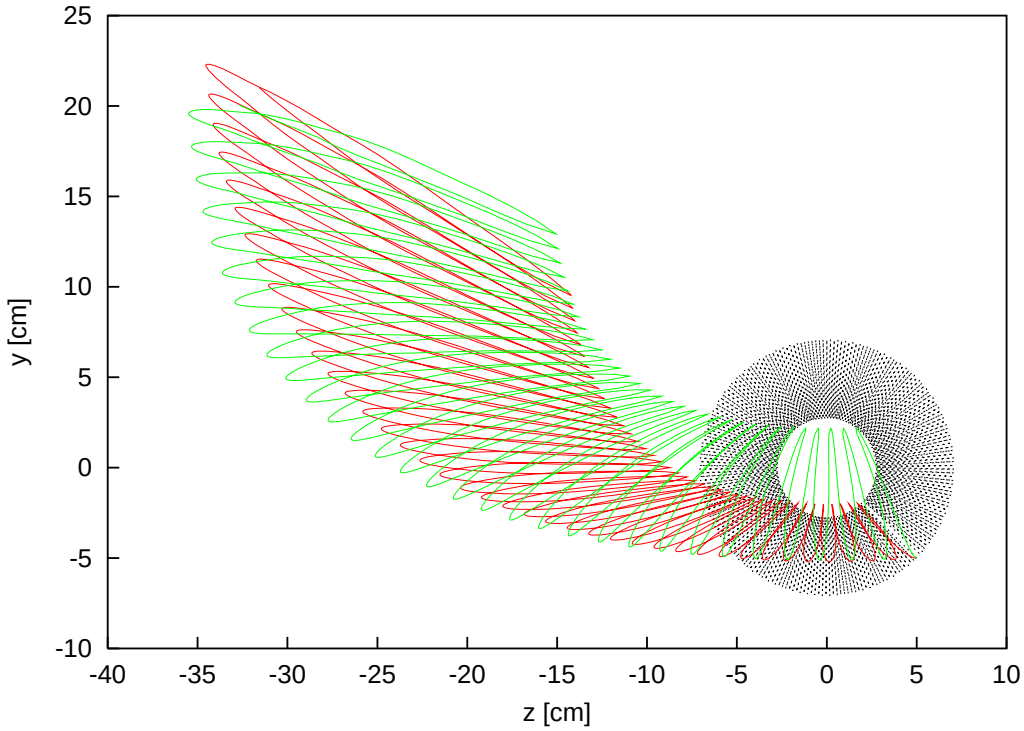

Abbildung 5.14: Vergleich von gekrümmtem Magnetfeld und Dipolfeld. Für diesen Vergleich wurde ein Elektron bei Position $(x, y, z) = (0, -5, 5)\ cm$ mit $E_{kin} = 100\ eV$ und einem Startwinkel von $\theta = 35°$ relativ zur führenden Magnetfeldlinie gestartet. Ohne Modifikationen am Spektrometer wird dieses Elektron gespeichert (schwarz). Durch das gekrümmte Magnetfeld (grün) stößt das Elektron nach $tof = 21\ \mu s$ gegen Elektrode E0. Zum Vergleich, durch ein nur 12 V/m starkes idealisiertes Dipolfeld in y-Richtung stößt das Elektron (rot) nach $tof = 27\ \mu s$ ebenfalls gegen Elektrode E0.

relativ zur führenden Magnetfeldlinie gestartet. In Abbildung 5.14 ist zu sehen, wie das Elektron vom gekrümmten Magnetfeld gegen eine Elektrode geführt wird. Hier ist auch ein Vergleich zwischen dem gekrümmten Magnetfeld und einem elektrischen Dipolfeld dargestellt. Der Vergleich wurde mit dem oben spezifizierten im unmodifizierten Spektrometer gespeicherten Elektron durchgeführt. Das um 20° gekrümmte Magnetfeld, 10° je Solenoid, verursacht, daß das Elektron nach $tof = 21\ \mu s$ gegen eine Elektrode fliegt. Durch ein vergleichbares Dipolfeld in y-Richtung, mit Feldstärke 12 V/m, fliegt das Elektron nach $tof = 27\ \mu s$ gegen eine Elektrode. Es ist also auch möglich, mit einem gekrümmten Magnetfeld gespeicherte Teilchen zu entfernen.

Der Vergleich zeigt jedoch auch, daß der Effekt eines elektrischen Dipolfeldes sehr viel größer ist. Wie erwartet erzeugt schon das 12 V/m starke Dipolfeld eine, dem gekrümmten Magnetfeld vergleichbare, Driftbewegung. Aus den Abschnitten 5.1 und 5.2 ist bekannt, daß selbst ein Dipolfeld mit 100 V/m noch nicht alle gespeicherten Elektronen aus dem Spektrometer entfernen kann. Übertragen auf ein gekrümmtes Magnetfeld bedeutet das, daß eine viel stärkere Krümmung, die aber nicht realisierbar ist, notwendig wäre, um ausreichend viele der gespeicherten Teilchen zu entfer-

nen. Der Vorteil eines gekrümmten Magnetfeldes ist die Möglichkeit permanent und ohne Pausen Meßdaten aufzunehmen. Ein Dipolfeld kann nur in Pausen, also bei unterbrochener Messung, aktiviert werden, da hierbei das elektrische Potential in der Analysierebene zu stark verformt wird. Ein Nachteil des gekrümmten Magnetfeldes ist, daß man die Krümmung der Spektrometersolenoide nur bei gleichzeitiger Veränderung des Elektrodensystems modifizieren kann. Ein einmal gekrümmtes Spektrometer muß und kann nur in der gegebenen Form betrieben werden. Anders die Dipolelektrode, die zwar nur in Meßpausen aktiviert werden kann, dafür aber variabel in der Feldstärke ist und insbesondere gilt, daß man sie abschalten kann.

Abschließend kann man festhalten, daß die Krümmung des Spektrometers im Vergleich zum Dipolfeld ungeeignet ist. Der Effekt von gekrümmten Magnetfeldern bei sehr hohen Feldstärken wird eventuell zwischen Vor- und Hauptspektrometer des KATRIN-Experiments ausgenutzt; es spielen dann auch Effekte wie Synchrotronabstrahlung eine Rolle. Hierfür sind noch gesonderte Untersuchungen notwendig, die nicht Bestandteil dieser Arbeit sind.

Kapitel 6

Zusammenfassung und Ausblick

Das Ziel dieser Arbeit war die Entwicklung von Methoden zur Untergrundreduzierung am Mainzer Tritium–β–Spektrometer, einem MAC–E–Filter, 'magnetic adiabatic collimation with electrostatic filter'. Für die Entwicklung solcher Methoden ist das Verständnis der Teilchenbewegung im MAC–E–Filter sehr wichtig. Der Versuch die Bewegung der geladenen Teilchen zu verstehen, hat zur adiabatische Näherung erster Ordnung geführt. In dieser Näherung wird die Zyklotronbewegung des Teilchens abseparіert und nicht mehr berücksichtigt. Es wird nur noch das Führungszentrum der Zyklotronbewegung betrachtet, mit dazu senkrechten Korrekturen. Man erhält so zwei zueinander senkrechte Bewegungsanteile. Als Hauptanteil der Bewegung wird die Bewegung entlang der Magnetfeldlinie als Führungszentrum betrachtet. Die dazu senkrechten Bewegungsanteile ergeben sich aus der Existenz von nicht parallelen elektrischen und magnetischen Feldern und aus Feldinhomogenitäten und der Krümmung des Magnetfeldes. Durch diese Vereinfachung kann die Teilchenbewegung sehr gut beschrieben und verstanden werden.

Aus der elektromagnetischen Konfiguration eines MAC–E–Filters lernt man, daß es im Inneren sehr viele Möglichkeiten zum permanenten Speichern von geladenen Teilchen gibt. Es existieren Penningfallen, magnetische Flaschen und natürlich das negative Analysierpotential. Diese Tatsachen führen zu verschiedenen Arten von Reflektionsbedingungen für geladene Teilchen. Mit Hilfe der Simulationsprogramme konnten die Teilchenfallen identifiziert werden. Untergrunduntersuchungen von J.-P. Schall [Sch01] haben ergeben, daß schon wenige gespeicherte Elektronen zu einer starken Erhöhung der Untergrundzählrate führen können (siehe Abschnitt 4.5). Als potentielle Quellen für die Erhöhung der Untergrundzählrate werden daher die gespeicherten Teilchen betrachtet. Um Methoden zu entwickeln, die die Speicherbedingungen der gespeicherten Teilchen brechen oder stören, müssen im wesentlichen Modifikationen am elektromagnetischen Aufbau des MAC–E–Filters untersucht werden.

Zur Simulation der Bewegung von geladenen Teilchen in der elektromagnetischen Umgebung des MAC–E–Filters gibt es das kommerzielle Programmpaket SIMION 3D, das die numerische Integration der Bewegungsgleichung verwendet. Die Langzeituntersuchungen von gespeicherten Teilchen waren mit SIMION schwierig, da die

numerische Integration zu viel Zeit in Anspruch nahm und das verwendete numerische Verfahren die Energieerhaltung nicht gewährleisten konnte.

Unter Anwendung der adiabatischen Näherung erster Ordnung wurde ein neues Simulationsprogramm entwickelt. Dieses Programm trägt den Namen ADIPARK, 'ADIabtic PARticle tracKing', und beschreibt die Teilchenbewegung durch das Führungszentrum mit dazu senkrechten Korrekturbewegungen. In diesem Programm ist die Energieerhaltung per Definition erhalten und es erlaubt um den Faktor 100 schnellere Berechnungen der Teilchenbahn. Es existiert aber die Einschränkung, daß ADIPARK Teilchenbewegungen nur dann richtig beschreibt, wenn die adiabatische Näherung erster Ordnung möglich ist. Verhält sich ein Teilchen nicht adiabatisch, kann die von ADIPARK berechnete Flugbahn sehr stark von der wirklichen Flugbahn abweichen. Neben den Tests der numerischen Genauigkeit, wurde daher auch durch einen Vergleich mit SIMION 3D der Gültigkeitsbereich der adiabatischen Näherung in der verwendeten Form festgelegt. Zur automatisierten Untersuchung von vielen Teilchen mit vielen Startparametern wurde ein Zusatzmodul, Trapping–Modul, erstellt, mit dessen Hilfe es möglich ist, Speichervolumenkarten zu erzeugen.

Zur Reduzierung von gespeicherten Teilchen gibt es mehrere Möglichkeiten. Das Überlagern eines elektrischen Dipolfeldes über das Analysierpotential ist eine Möglichkeit. Hierbei wird die $\vec{E} \times \vec{B}$–Drift der Teilchenbewegung ausgenutzt. Das Dipolfeld erzeugt eine Driftbewegung senkrecht zum Dipolfeld und dem Magnetfeld im Spektrometer. Wenn die Dipolfeldstärke so gewählt wird, daß die $\vec{E} \times \vec{B}$–Drift über alle anderen Driftbewegungen dominiert, wird das Teilchen aus dem im Detektor abgebildeten magnetischen Flußschlauch geführt und trifft auf eine Elektrode. Überwiegen die azimutal wirkenden Driftanteile, bleibt das Teilchen auf einer verzerrten Bahn gespeichert. Durch Simulationen einer realen Dipolhalbschale im Mainzer Spektrometer konnte gezeigt werden, daß diese Methode sehr erfolgreich die Speicherbedingungen der geladenen Teilchen stört. Es bleibt anzumerken, daß ein Dipolfeld nur in Meßpausen eingesetzt werden kann, da es das Analysierpotential zu stark verzerrt und die Funktion des MAC–E–Filters nicht mehr gewährleistet ist.

Eine zweite Möglichkeit den elektromagnetischen Aufbau zu modifizieren ist die Verwendung eines gekrümmten Magnetfeldes. In diesem Fall wird die magnetische Krümmungsdrift der Teilchenbewegung ausgenutzt. Ein Teilchen, das sich entlang einer gekrümmte Magnetfeldlinie bewegt, erfährt eine Driftbewegung senkrecht zur Magnetfeldlinie und dem Krümmungsradius derselben. Die vorgegebene Krümmung des Magnetfeldes muß so groß sein, daß die resultierende Krümmungsdrift über die restlichen azimutalen Driftbewegungen dominiert. In den Simulationen wurde das gekrümmte Magnetfeld durch Kippen der Spektrometersolenoide um je 10° realisiert. Auch in diesem Fall konnte erfolgreich gezeigt werden, daß Speicherbedingungen von geladenen Teilchen gestört werden können. Der Vorteil dieser Methode ist, daß gekrümmte Spektrometersolenoide die Funktion des MAC–E–Filters nicht beeinträchtigen und keine Meßpausen mehr nötig sind. Allerdings ist es nicht einfach möglich das einmal gekippte Spektrometerdesign zu verändern, denn für jede Änderung am

Kippwinkel muß das Elektrodensystem angepaßt werden.

Zusammenfassend kann man sagen, daß beide Methoden erfolgreich zum Ziel führen. Beide erfüllen die Anforderungen, daß die Speicherbedingungen der geladenen Teilchen zerstört werden. Unter dem Gesichtspunkt der Flexibilität muß man das elektrische Dipolfeld bevorzugen, da es variabel in der Feldstärke und der Feldkonfiguration ist und je nach Bedarf ein- und ausgeschaltet werden kann. Im Vergleich zum Dipolfeld ist das gekrümmte Magnetfeld ungeeignet, da schon relativ schwache Dipolfelder den Effekt einer realisierbaren Krümmung übertreffen.

Die ersten experimentellen Untersuchungen eines durch eine Dipolelektrode erzeugten elektrischen Dipolfeldes werden von B. Müller [Mue02] durchgeführt. Ziel dieser Arbeit wird sein, verschiedene Dipolkonfigurationen zu simulieren, um dann einen vielversprechenden Entwurf in das Mainzer Spektrometer zu integrieren. In den darauf folgenden Untergrundmessungen sollen die Auswirkungen des elektrischen Dipolfeldes experimentell überprüft werden und erste Aussagen über die erforderliche Größe der Dipolfeldstärke gemacht werden.

Das neu entwickelte Simulationsprogramm ADIPARK wird auch zur Entwicklung der MAC–E–Filter des KATRIN–Experimentes eingesetzt. Es dient dort zur frühzeitigen Erkennung von Teilchenfallen und zur Berechnung der Transmissionseigenschaften.

Da sich ADIPARK bewährt hat, sind für die Zukunft verschiedene Erweiterungen geplant, dies wird durch die modulare Struktur begünstigt. Mögliche Erweiterungen sind die Berechnung von Synchrotronabstrahlung, Wechselwirkungen mit Restgasmolekülen und die Möglichkeit die vollständige Teilchenbewegung durch numerische Integration der dreidimensionalen Bewegungsgleichung zu verfolgen.

Literaturverzeichnis

[Ath95] C. Athanassopoulos et al., Phys. Rev. Lett. **75** (1995) 2650

[Bon99] J. Bonn, L. Bornschein, B. Degen, E.W. Otten, Ch. Weinheimer, NIM **A421** (1999) 256

[Bon01] J. Bonn, et al. Nuclear Physics B 91 (2001) 273-276

[Bor00] B. Bornschein, Dissertation, Inst. f. Physik, Universität Mainz, 2000

[Con01] F. Galve Conda, Sommerstudent aus Valentia, Mainz 2001

[Deg99] R. Degenhardt, M. Berz, Nucl. Inst. and Meth., Proc. CPO5 (1999)

[Del65] J. L. Delcroix, Plasma Physics, Wiley, New York, 1965

[Die00] DARK 2000, Heidelberg, Vortrag von A. Dietz, Juli 2000

[Don00] Long Baseline News, Juli 2000, http://hepunx.rl.ac.uk/minos/longbnews/0007.html

[Eit00] K. Eitel, New Jour. Phys. **2** (2000) 1

[Ell87] S. R. Elliot et al., Phys. Rev. Lett. **59** (1987) 2020

[Fer34] E. Fermi, Z. Phys. **88** (1934) 161

[Fic00] L. Fickinger, Diplomarbeit, Inst. f. Physik, Universität Mainz, 2000

[Fla01] B. Flatt, Diplomarbeit, Inst. f. Physik, Universität Mainz, 2001

[Fla01a] B. Flatt und Ch. Weinheimer, Considerations on the electromagnetic design of the pre–spectrometer, KATRIN collaboration meeting, Karlsruhe, December 2001

[Fuk98] Y. Fukuda et al., Phys. Rev. Lett. **81** (1998) 1562

[Jac82] J.D. Jackson, Klassische Elektrodynamik, W. de Gruyter, Berlin 1982

[Kun66] W. B. Kunkel, 'Plasma Physics in Theory and Application', McGraw–Hill, New York, 1966

[Lob99] V.M. Lobashev et al., Phys. Lett. **B460** (1999) 227

[MS] Microsoft, Windows und Windows NT sind entweder eingetragene Warenzeichen oder Warenzeichen der Microsoft Corporation in den USA und/oder anderen Ländern.

[Mue02] B. Müller, Diplomarbeit, Inst. f. Physik, Universität Mainz, voraussichtliche Fertigstellung 2002

[Nor61] T. G. Northrop, Ann. Phys., **15**, 79 (1961)

[Nor63] T. G. Northrop, 'The Adiabatic Motion of Charged Particles', Wiley, New York, 1963

[Num92] W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery, 'Numerical Recipes in C', Cambridge University Press, 1992

[Par98] Particle Data Group, Eur. Phys. J. **C3** (1998) 1

[Pau30] W. Pauli, Brief an die Physikalische Gesellschaft in Tübingen vom 4.12.1930, nachzulesen in L.M. Brown, Phys. Today **31** (1978) 23

[Rei59] F. Reines, C.L. Cowan, Phys. Rev. **113** (1959) 273

[Sch01] J.-P. Schall, Diplomarbeit, Inst. f. Physik, Universität Mainz, 2001

[SIM] http://www.srv.net/~klack/simion.html

[SNO01] Sudbury Neutrino Observatory Collaboration, Measurement of charged current interactions produced by ^8B solar neutrinos at the Sudbury Neutrino Observatory, 18 June 2001

[Tit01] N. Titov, KATRIN Collaboration meeting, Mainz, March 2001

Danksagung

Diese Seite ist allen gewidmet, die zum Gelingen meiner Arbeit beigetragen haben. Ich möchte mich hiermit herzlich für all die Hilfe und Unterstützung bedanken, die ich erfahren habe.

Zuerst möchte ich mich bei Herrn Prof. Dr. Christian Weinheimer für die interessante Aufgabenstellung und die immer wieder motivierende Betreuung bedanken. Seine guten Ideen und das Interesse an neuen und exotischen Simulationsszenarien haben geholfen, sehr viele Fehler und Probleme in ADIPARK zu beheben.

Auch Herrn Prof. Dr. Ernst W. Otten gilt mein Dank. Die Diskussionen über das adiabatische Verhalten von geladenen Teilchen haben sehr zu meinem Verständnis der Zusammenhänge beigetragen. In jeder Diskussion mit ihm profitierte ich von seinem großen Wissen, das er jederzeit versucht weiterzugeben.

Dr. Jochen Bonn hatte immer ein offenes Ohr, wenn ich mal wieder Fragen zur Funktionsweise und zum Aufbau des Experimentes hatte. Es ist sehr hilfreich gewesen, ihn über Untergrundprozesse und Vakuum auszufragen. Von ihm habe ich viel über die Arbeit und den Umgang mit Laborgeräten gelernt.

Bei Frau Dr. Beate Bornschein bedanke ich mich für die guten Erklärungen, die Hilfsbereitschaft, nicht nur bei Linux-Problemen, und die Informationsflut, mit der sie ihre langjährigen Erfahrungen weitergibt. Die Laborarbeit mit ihr hat mein Verantwortungsbewußtsein am stärksten geprägt.

Der gute Lutz Bornschein hat die Arbeitsatmosphäre immer wieder positiv beeinflußt. Ob richtig harte Arbeit, Dartspiel oder die netten Diskussionen über alles außer Physik; alles hat zur guten Laune beigetragen. Ihm, als meinem einzigen Leidensgenossen beim Zusammenschreiben, möchte ich hier viel Erfolg und Glück wünschen.

Bei Christine Kraus möchte ich mich für das ununterbrochene Interesse an meiner Arbeit bedanken. Sie war mir eine große Hilfe beim Korrekturlesen und hat sich sehr viel Zeit für mich genommen.

Björn Flatt gilt mein besonderer Dank, erst sein Bfield3D und die Zusammenarbeit während der Entwicklung von ADIPARK haben das Programm in dieser Form möglich gemacht. Aber auch nicht physikalisch möchte ich mich bei ihm für diverse Freizeitaktivitäten bedanken.

Meinem Tischnachbarn, Jean-Pierre Schall, möchte ich für die Hilfe bei der Untergrundproblematik danken, die Diskussionen mit ihm waren immer sehr produktiv. An dieser Stelle möchte ich ihm viel Erfolg bei seinen Diplomprüfungen wünschen.

Meiner Nachfolgediplomandin, Beatrix Müller, danke ich für die amüsanten Unterhaltungen früh am Morgen. Es ist nun an ihr die Ergebnisse meiner Arbeit in die Tat umzusetzten und zu überprüfen. Ich wünsche ihr viel Erfolg.

Im Zusammenhang mit der adiabatischen Näherung möchte ich Dr. Ferenc Glück danken. Durch seine Erfahrung und Literatursammlung war es erst möglich die adiabatische Näherung in dieser Form zu verstehen und aufzuschreiben.

Für die Hilfsbereitschaft, die guten Plätzchen und die gute Laune möchte ich mich bei den Damen des Sekretariats, Christine Best und Elvira Stuck-Kerth, bedanken.

Auch dem Sommerstudenten Fernando Galve Conda möchte ich hier danken, er hat die ersten Schritte zur Entwicklung des Trapping-Moduls unternommen.

An dieser Stelle möchte ich mich auch bei allen ehemaligen Mitgliedern der Neutrinogruppe bedanken, die das Fundament für meine Arbeit gelegt haben.

Bei all meinen Freunden und Freundinnen, die immer wieder nach dem Stand meiner Arbeit gefragt haben und mich motiviert und unterstützt haben, möchte ich mich hier bedanken. Mein besonderer Dank gilt Alexander Veltzke, für das Interesse an und die interessanten Diskussionen über die Physik.

Schließlich möchte ich meinen Eltern und Großeltern ganz herzlich für die anhaltende Unterstützung und das Verständnis während meines Studiums und dieser Arbeit danken. Ohne euch hätte ich es nicht geschafft!

Anhang A

Quelltexte

A.1 ADIPARK Hauptprogramm: adipark.c

```
/*********************************************/
/*                  ADIPARK                  */
/*        Adiabatic Particle Tracking        */
/* July to September 2001, by Thomas Thümmler */
/*   Thanks for Support From Chr. Weinheimer  */
/*             and Björn Flatt               */
/*                                           */
/*Based on                                   */
/*               Transmission                */
/*           Adiabatic Ray Tracing           */
/*       May,June 2000, by Lars Fickinger    */
/*********************************************/
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <math.h>
#include <stdarg.h>
#include "global.h"
#include "efield.h"
#include "b_struct.h"
#include "b_util.h"
#include "tracking.h"
#include "trapping.h"
#include "mag_pa_reader.h"

int main(int argc, char *argv[]) {

  struct coils spul;
  struct bfield_geom geo;
  int choice;
  char* pa_in_name;
  char* par_filename;

    printf("\n\n");
    printf("\n*****************************************\n");
    printf("**                                     **\n");
```

```
    printf("**             A D I P A R K             **\n");
    printf("**                                       **\n");
    printf("**      ADIabatic PARticle tracKing      **\n");
    printf("**                                       **\n");
    printf("**    written by Th. Thuemmler in 2001   **\n");
    printf("**                                       **\n");
    printf("*******************************************\n");
    printf("**             Version 3.01              **\n");
    printf("*******************************************\n\n");
  if (argc<2) {
    puts("Usage: adipark <parameter filename without suffix>");
    printf("  use MM_PER_UNIT     %f\n", MM_PER_UNIT);
    printf("  use MAG_MM_PER_UNIT %f\n\n", MAG_MM_PER_UNIT);
  } else {
    alloc_electric_arrays();
    alloc_mag_arrays();
    par_filename = argv[1];
    read_epot(par_filename);
    read_mag_pa(par_filename);
    pot2efield(MM_PER_UNIT);
    do {
      choice = 0;
      printf("\nEnd:                                    0\n");
      printf("Show Magnetic Setup                     1\n");
      printf("Start Particle Tracking Loop            2\n");
      printf("Test Trapping Volume                    3\n\n");
      printf("Choose: ");
      scanf("%d", &choice);
      printf("\n");
      switch(choice){
      case 0 :
break;
      case 1 :
read_par(&spul, &geo, par_filename);
print_par(&spul, &geo);
break;
      case 2 :
read_par(&spul, &geo, par_filename);
print_par(&spul, &geo);
tracking_loop(&spul, &geo, par_filename);
break;
      case 3 :
read_par(&spul, &geo, par_filename);
print_par(&spul, &geo);
trapping(&spul, &geo);
break;
      default :
printf("--- Fehler ---\n");
      }
    }  while (choice != 0);
    free_electric_arrays();
    free_mag_arrays();
    printf("\n\n");
  }
  return 0;
}
```

A.2 Bahnverfolgung: tracking.c

```
#include <stdio.h>
#include <math.h>
#include "global.h"
#include "integral.h"
#include "matrix.h"
#include "efield.h"
#include "math_vector.h"
#include "b_struct.h"
#include "mag_pa_reader.h"
#include "tracking.h"


double dipol = 0.;              // factor to magnify dipol potential
int    retard_onoff = 1;        // switch on and off retarding potential
int    save_data_count = 0;     // counter to count between data to save
int    last_saved = -210.;

/******************* particle output ***********************/
void output(struct particle_data* particle)
{
  if (display == 2) // when set to 2, display data block with label
    {
      printf("Position     (x,y,z) %f %f %f\n",(*particle).position[0],
     (*particle).position[1],(*particle).position[2]);
      printf("B_START      (x,y,z) %f %f %f\n",(*particle).b_start[0],
     (*particle).b_start[1],(*particle).b_start[2]);
      printf("siný(alpha_start)    %f\n",(*particle).sin2_alpha_start);
      printf("V_PARA       (x,y,z) %f %f %f\n",(*particle).v_para[0],
     (*particle).v_para[1],(*particle).v_para[2]);
      printf("V_PERP-start (x,y,z) %f %f %f\n",(*particle).v_perp_start[0],
     (*particle).v_perp_start[1],(*particle).v_perp_start[2]);
      printf("|V_PARA|=%f, |V_PERP|=%f\n",(*particle).v_para_value,(*particle).v_perp_value);
      printf("Energy (para,perp,kin)  %f %f %f\n",(*particle).e_para,
     (*particle).e_perp,(*particle).e_kin);
      printf("STEP         (x,y,z) %f %f %f\n",(*particle).step_final[0],
     (*particle).step_final[1],(*particle).step_final[2]);
      printf("---\n");
    }
  if (display == 1) // when set to 1, display data in one line without label
    {
      printf("%f %f %f %f %f %f\n",(*particle).position[0],
     (*particle).position[1],(*particle).position[2],(*particle).e_kin,
     (*particle).time_of_flight*1000000.,(*particle).shrinkfactor);
    }

}


/******************* savedata ***********************/
void savedata(FILE* f_fly,struct particle_data* particle)
{ // save one datablock to harddisc
  //  save_data_count = 0; //only for the center points
  if (((*particle).position[0] >= -(*particle).max_step_length) &&
      ((*particle).position[0] <= (*particle).max_step_length))
    save_data_count = SAVE_EVERY; // force data saved around analysing plane
```

```
  if (((*particle).position[0] >= (SPEC_OUT-(*particle).max_step_length*10)) ||
      ((*particle).position[0] <= (SPEC_IN+(*particle).max_step_length*10)))
    save_data_count = SAVE_EVERY; // force data saved at source and detector
  if (save_data_count == SAVE_EVERY)
    {
      save_data_count = 0;
      fprintf(f_fly,"%1.9f %1.9f %1.9f ",(*particle).position[0],
      (*particle).position[1],
      (*particle).position[2]);
      fprintf(f_fly,"%1.9f ",(*particle).cyclrad);
      fprintf(f_fly,"%1.9f %1.9f ",(*particle).e_pot,(*particle).e_kin);
      fprintf(f_fly,"%1.9f %1.9f ",(*particle).e_para,(*particle).e_perp);
      fprintf(f_fly,"%1.9f %1.12f ",(*particle).b_value,(*particle).time_of_flight);
      fprintf(f_fly,"%1.9f %1.9f %1.9f ",(*particle).v_para[0],
      (*particle).v_para[1],
      (*particle).v_para[2]);
      fprintf(f_fly,"%1.9f %1.9f ",(*particle).v_para_value,(*particle).v_perp_value);
      fprintf(f_fly,"%1.9f %1.9f %1.9f ",(*particle).exb_vel[0],
      (*particle).exb_vel[1],
      (*particle).exb_vel[2]);
      fprintf(f_fly,"%1.9f %1.9f %1.9f ",(*particle).rxb_vel[0],
      (*particle).rxb_vel[1],
      (*particle).rxb_vel[2]);
      fprintf(f_fly,"%1.9f %1.9f %1.9f ",(*particle).dbxb_vel[0],
      (*particle).dbxb_vel[1],
      (*particle).dbxb_vel[2]);
      fprintf(f_fly,"%1.9f %1.9f ",(*particle).v_signum,(*particle).shrinkfactor);
      fprintf(f_fly,"\n");
    }
  else save_data_count++;
}


/******************** epot3d ***********************/
double epot3d(double *position, int *e_tag)
{ // this function gives 3 dim potentials out of cylindric values from efield module
  double cyl_pos[3]; // 2D coordinates
  double e_pot = 0.;    // electric potential
  int splat;

  if (retard_onoff == 1) // switch on and off retarding potential
    {
      if (get_symmetry() == 0)
{
  cyl_pos[0] = position[0];      // 2D z coordinate equal to 3D x coordinate
  cyl_pos[1] = sqrt(position[1]*position[1]+position[2]*position[2]);
  cyl_pos[2] = 0.;
  // read 2D flux radius by using distance between
  // x axis and position
  e_pot = get_epot2D(cyl_pos,&splat);
}
      else e_pot = get_epot3D(position,&splat);        // use 3D potential array
      if (splat == 1)                         // tag for electrode detection
{
  *e_tag = *e_tag + 1;               // use e_tag to inform parent routine
}
    }
```

```
  if (dipol != 0.)
    {
      e_pot = e_pot - dipol*(position[2]+50.)/100.;
    }
  // add a linear potential to force drift motion
  return e_pot;                          // return the value of epot
}


/******************** efield3d ***********************/
void efield3d(double *position, double *evec, int *e_tag)
{  // this function calculates the electric field vector E by gradient of potential
  double pos1[3],pos3[3];
  double delta = MM_PER_UNIT;
  int komp,splat_tag;


  delta = delta*MM2CM;   // delta in mm must be used in cm
  if (delta == 0.) printf("Error: MM_PER_UNIT constant not set!\n");
  for (komp = 0; komp <= 2; komp++)
    {
      vector_times_scalar(position,1.,pos1);
      vector_times_scalar(position,1.,pos3);
      pos1[komp] = pos1[komp] - delta;
      pos3[komp] = pos3[komp] + delta;
      evec[komp] = -(epot3d(pos3,&splat_tag)-epot3d(pos1,&splat_tag))
                   *M2CM/(2.*delta);
    }
  if (splat_tag > 0) *e_tag = *e_tag + 1;
}


/******************** get_gamma ***********************/
double get_gamma(struct particle_data* particle)
{ // this function calculates the relativistic gamma factor
  return ((*particle).e_kin + (*particle).mass)/(*particle).mass;
}


/******************** get_cyclrad ***********************/
double get_cyclrad(struct particle_data* particle)
{ // this function calculates the cyclotron radius of particle in cm
  return (sqrt( (*particle).e_cycl*(*particle).mass*(get_gamma(particle)+1.) )
         /(3000000.*(*particle).b_value));
}


/******************** go_step ***********************/
void go_step(struct particle_data* particle, struct particle_data* probe)
{ // this function adds to the position vector th estep vector
// in other words it goes one step forward
  (*probe) = (*particle);
  vector_sum((*probe).position,(*probe).step_final,(*probe).position);
}


/******************** shrink_step ***********************/
```

```
void shrink_step(struct particle_data* particle)
{ // this function halfs the shrinkfactor value each time it was called
  (*particle).shrinkfactor = (*particle).shrinkfactor * 0.5;
}

/******************** unshrink_step ***********************/
void unshrink_step(struct particle_data* particle)
{ // this function halfs the shrinkfactor value each time it was called
  (*particle).shrinkfactor = (*particle).shrinkfactor * 2.;
}

/******************** turn_direction ***********************/
void turn_direction(struct particle_data* particle)
{ // this function turns the direction of motion and counts the miror points
  (*particle).v_signum = -(*particle).v_signum;
  (*particle).mirrors++;
  printf("mirror=%d tof=%f\n",(*particle).mirrors,(*particle).time_of_flight); // put out mirror coun
}

/******************** test_electrode ***********************/
int test_electrode(struct particle_data* particle,struct particle_data* probe)
{
  double radius;
  double testvec[3];
  double testpos[3];
  double test_here[3];
  double origin[3];
  double gain;
  int e_tag;

  radius = sqrt(pow((*probe).position[1],2)+pow((*probe).position[2],2))
           + get_cyclrad(probe);

  vector_times_scalar((*probe).position,1.,testvec);
  vector_times_scalar((*probe).position,1.,origin);
  testvec[0] = 0.;
  origin[1] = 0.;
  origin[2] = 0.;

  if ((testvec[1] == 0.) && (testvec[2] == 0.)) testvec[1] = 1.;

  vector_times_scalar(testvec,MM_PER_UNIT/(2.*absvalue(testvec)),testvec);

  gain = 1.;
  e_tag = 0;

  do {
    vector_times_scalar(testvec,gain,testpos);
    vector_sum(testpos,origin,test_here);
    epot3d(test_here,&e_tag);
    gain++;
  } while ((e_tag == 0) && (absvalue(testpos) <= MAX_RADIUS));
  e_tag = 0;
  if (radius > sqrt(test_here[1]*test_here[1]+test_here[2]*test_here[2]))
    {
      printf("Particle at radius %5.2fcm splat on electrode at radius %5.2fcm\n",radius,sqrt(test_here
      e_tag = 1;
```

```
    }
  if (radius >= MAX_RADIUS) e_tag = 1;
  return e_tag;
}



/******************** Runge Kutta Midpoint Methode ***********************/
void runge_kutta_midpoint(struct particle_data* particle, struct coils* spulen,
  struct bfield_geom* geom)
// this routine calculates the next point on bfield line by using
// runge kutta midpoint method with second order.
{
  double pos1[3],b1[3],step1[3],step2[3];

  vector_times_scalar((*particle).step_final,0.5,step1);        // half step value
  vector_sum((*particle).position,step1,pos1);                  // go one step (of half value)
  get_bfield(pos1,b1,spulen,geom);                             // get bfield vector here
  vector_times_scalar(b1,(*particle).step_final[0]/b1[0],step2); // go one step (whole value)
                                                                // with bfield vector of middle point
  vector_times_scalar(step2,1.,(*particle).step_final);          // new step vector in v_step
}


/******************** ExB Drift Calculation ***********************/
void ecrossb_drift(struct particle_data* particle, struct coils* spulen,
   struct bfield_geom* geom)
{   // this function calculates the EcrossB drift velocity and uses
// the runge-kutta-method (2nd order) too
  double evec[3];
  int e_tag = 0;

  // starting point
  efield3d((*particle).position,evec,&e_tag);
  cross_prod(evec,(*particle).b_vec,(*particle).exb_vel);
  if ((*particle).b_value == 0.) printf("Error: B-Value zero in ExB-drift calculation!\n");
  vector_times_scalar((*particle).exb_vel,
      1./((*particle).b_value*(*particle).b_value),
      (*particle).exb_vel);
}

/******************** Drift relativity Check ***********************/
int test_drift(struct particle_data* particle)
{   // this function calculates the EcrossB drift velocity and uses
// the runge-kutta-method (2nd order) too
  int error = 0;

  if (absvalue((*particle).exb_vel) >= Clight/2.)
    {
      printf("WARNING: ExB Veclocity = %4.2f Clight\n",absvalue((*particle).exb_vel)/Clight);
      error=+1;
    }
  if (absvalue((*particle).dbxb_vel) >= Clight/2.)
    {
      printf("WARNING: Gradientdrift Veclocity = %4.2f Clight\n",absvalue((*particle).exb_vel)/Clight);
      error=+2;
    }
```

```
  if (absvalue((*particle).rxb_vel) >= Clight/2.)
    {
      printf("WARNING: Curcaturedrift Veclocity = %4.2f Clight\n",absvalue((*particle).exb_vel)/Clight
      error=+4;
    }
  return error;
}


/******************* magnetron drift **********************/
void mag_drift(struct particle_data* particle, struct coils* spulen, struct bfield_geom* geom)
{
  double pos1[3],pos3[3]; // positions on B field line
  double bvec1[3],bvec3[3];                  // B vector for gradient calculation
  double dbvec[3],db_perp_vec[3];            // vector of db and perpendicular db
  double delta = MAG_MM_PER_UNIT;            // delta value for gradient calculation
  double gdrift[3];                          // final gradient drift vector
  int komp;                                  // component counter

  // now  computing the gradient and curvature drift

  delta = delta*MM2CM;  // delta in mm must be used in cm
  if (delta == 0.) printf("Error: MM_PER_UNIT constant not set!\n");
  for (komp = 0; komp <= 2; komp++)
    { // this is a simple gradient calculation for Bfield value
      vector_times_scalar((*particle).position,1.,pos1);
      vector_times_scalar((*particle).position,1.,pos3);
      pos1[komp] = pos1[komp] - delta;
      pos3[komp] = pos3[komp] + delta;
      get_bfield(pos1,bvec1,spulen,geom);
      get_bfield(pos3,bvec3,spulen,geom);
      dbvec[komp] = (absvalue(bvec3)-absvalue(bvec1))*M2CM/(2.*delta);
    }

  gradB_perp((*particle).b_vec,dbvec,db_perp_vec); // use only perpendicular part of gradB

  cross_prod((*particle).b_vec,db_perp_vec,gdrift); // calc gradB_perp cross Bvector

  if ((*particle).b_value == 0.) printf("Error: B-Value zero in mag-drift calculation!\n");
  vector_times_scalar(gdrift,
      -(*particle).charge*(*particle).e_perp/
      ((*particle).b_value*(*particle).b_value*(*particle).b_value),
      (*particle).dbxb_vel);  // finishing gradient drift velocity
  vector_times_scalar(gdrift,
      -(*particle).charge*2.*(*particle).e_para/
      ((*particle).b_value*(*particle).b_value*(*particle).b_value),
      (*particle).rxb_vel);   // finishing curvature drift velocity
}



/******************* calc_drifts **********************/
void calc_drifts(struct particle_data* particle, struct coils* spulen, struct bfield_geom* geom)
{
  vector_times_scalar((*particle).exb_vel,0.,(*particle).exb_vel);
  vector_times_scalar((*particle).rxb_vel,0.,(*particle).rxb_vel);
  vector_times_scalar((*particle).dbxb_vel,0.,(*particle).dbxb_vel);
```

```
  if ((*particle).calc_order > 0)
    {
      if ((*particle).calc_order == 1)
{
  ecrossb_drift(particle,spulen,geom);                  // calculate ExB drift
}
      if ((*particle).calc_order == 2)
{
  mag_drift(particle,spulen,geom);           // calculate curvature and gradient drift
}
    }
  else
    {
      ecrossb_drift(particle,spulen,geom);                  // calculate ExB drift
      mag_drift(particle,spulen,geom);          // calculate curvature and gradient drift
    }
}


/******************* init_particle  ***********************/
void init_particle(struct particle_data* particle, struct coils* spulen, struct bfield_geom* geom)
{    // this routine is used to init all importent data before simulation begins
  double v_value;
  double v_vec[3];
  int e_tag = 0;

  // calculate start values now
  (*particle).u_start = epot3d((*particle).start_pos,&e_tag);// potential at start position
  (*particle).e_pot = (*particle).u_start;
  vector_times_scalar((*particle).start_pos,1.,(*particle).position);
                                                  // here starting position equal to position
  (*particle).e_kin = (*particle).e_start;                // e_kin equal to e_start
  (*particle).gamma_start = get_gamma(particle);          // calc rel gamma factor at start
  (*particle).v_signum = 1.;                              // start in positive direction
  (*particle).time_of_flight = 0.;                        // reset tof
  (*particle).delta_tof = 0.;                             // reset delta_tof
  (*particle).mirrors = 0;                                // reset mirror count
  (*particle).shrinkfactor = 1.;                          // reset shrinkfactor
  vector_times_scalar((*particle).step_final,0.,(*particle).step_final); // reset step_final

  v_value = sqrt((*particle).e_start*(*particle).e_start +              // velocity here
 2.*(*particle).e_start*(*particle).mass)*Clight/((*particle).e_start +
  (*particle).mass);

  spher2kart(v_vec,v_value,(*particle).starting_theta,(*particle).starting_phi);
                                                  // transform sherical vector in kartesian vector
  if (v_vec[0] < 0.) {(*particle).v_signum = -1.;}  // turn direction if velo_vec pointing to neg.

  get_bfield((*particle).start_pos, (*particle).b_start, spulen, geom);
                                                  // starting bvec vector at start position
  (*particle).b_start_value = absvalue((*particle).b_start);
                                                  // and its value

  (*particle).sin2_alpha_start = sin(angle_rad((*particle).b_start,v_vec))
                                *sin(angle_rad((*particle).b_start,v_vec));
                                          // angle between bfield vector and velocity vector
```

```
  vector_times_scalar((*particle).b_start,
      scalar_prod((*particle).b_start,v_vec)/
      ((*particle).b_start_value*(*particle).b_start_value),
      (*particle).v_para);
  vector_times_scalar((*particle).v_para,-1.,(*particle).v_para); // turn direction of v_para
  vector_sum(v_vec,(*particle).v_para,(*particle).v_perp_start);
                                        // to bfield perpendicular part of velocity vector
  vector_times_scalar((*particle).v_para,-1.,(*particle).v_para); // turn back direction of v_para

  (*particle).v_perp_value = absvalue((*particle).v_perp_start);
  (*particle).v_perp_start_value = (*particle).v_perp_value;    // value of perpendicular velocity
  (*particle).v_para_value = absvalue((*particle).v_para);      // value of parallel velocity
}


/******************* init_particle_se  **********************/
void init_particle_se(struct particle_data* particle, struct coils* spulen, struct bfield_geom* geom)
{    // this routine is used to init all importent data before simulation begins
  double v_value;
  double v_vec[3];
  int e_tag = 0;
  double beta;                    // angle between x-direction and B-vector
  double x_unit[3] = {1.,0.,0.}; // direction of x-axis

  // calculate start values now
  (*particle).u_start = epot3d((*particle).start_pos,&e_tag);// potential at start position
  (*particle).e_pot = (*particle).u_start;
  vector_times_scalar((*particle).start_pos,1.,(*particle).position);
                                                  // here starting position equal to position
  (*particle).e_kin = (*particle).e_start;                // e_kin equal to e_start
  (*particle).gamma_start = get_gamma(particle);          // calc rel gamma factor at start
  (*particle).v_signum = 1.;                              // start in positive direction
  (*particle).time_of_flight = 0.;                        // reset tof
  (*particle).delta_tof = 0.;                             // reset delta_tof
  (*particle).mirrors = 0;                                // reset mirror count
  (*particle).shrinkfactor = 1.;                          // reset shrinkfactor

  vector_times_scalar((*particle).step_final,0.,(*particle).step_final); // reset step_final

  v_value = sqrt((*particle).e_start*(*particle).e_start +              // velocity here
 2.*(*particle).e_start*(*particle).mass)*Clight/((*particle).e_start +
  (*particle).mass);
  get_bfield((*particle).start_pos, (*particle).b_start, spulen, geom);
                                          // starting bvec vector at start position

  beta = angle((*particle).b_start,x_unit);      // get angle of B-field-vector
  if ((*particle).b_start[1] < 0.) beta = -beta; // test B-Vector pointing down
  (*particle).starting_theta = beta + (*particle).starting_theta; // adjust starting_theta
  if (((*particle).starting_theta > 89.) && ((*particle).starting_theta < 91.))
    (*particle).starting_theta = 89.;

  spher2kart(v_vec,v_value,(*particle).starting_theta,(*particle).starting_phi);
                                          // transform sherical vector in kartesian vector
  if (v_vec[0] < 0.) {(*particle).v_signum = -1.;}  // turn direction if velo_vec pointing to neg.

  (*particle).b_start_value = absvalue((*particle).b_start);
                                          // and its value
```

```
  (*particle).sin2_alpha_start = sin(angle_rad((*particle).b_start,v_vec))
                                *sin(angle_rad((*particle).b_start,v_vec));
                                        // angle between bfield vector and velocity vector
  vector_times_scalar((*particle).b_start,
      scalar_prod((*particle).b_start,v_vec)/
      ((*particle).b_start_value*(*particle).b_start_value),
      (*particle).v_para);
  vector_times_scalar((*particle).v_para,-1.,(*particle).v_para); // turn direction of v_para
  vector_sum(v_vec,(*particle).v_para,(*particle).v_perp_start);
                                          // to bfield perpendicular part of velocity vector
  vector_times_scalar((*particle).v_para,-1.,(*particle).v_para); // turn back direction of v_para

  (*particle).v_perp_value = absvalue((*particle).v_perp_start);
  (*particle).v_perp_start_value = (*particle).v_perp_value;    // value of perpendicular velocity
  (*particle).v_para_value = absvalue((*particle).v_para);      // value of parallel velocity
}



/******************* calc_energies  ***********************/
void calc_energies(struct particle_data* particle, int* e_tag, struct coils* spulen, struct bfield_geom*
{ // this routine calculates the energies, perpendicular part and the parallel part
  double gamma;

  (*particle).e_pot = epot3d((*particle).position,e_tag);
  get_bfield((*particle).position,(*particle).b_vec,spulen,geom);
  (*particle).b_value = absvalue((*particle).b_vec);
  gamma = get_gamma(particle);

  (*particle).e_perp =
    (*particle).e_start*(*particle).sin2_alpha_start*(*particle).b_value*
    ((*particle).gamma_start + 1.)/((*particle).b_start_value*(gamma + 1.));

  (*particle).e_para = (*particle).e_start -
    (*particle).e_perp + (*particle).charge*((*particle).e_pot - (*particle).u_start);

  calc_drifts(particle,spulen,geom); // calculate all recommended drift velocities

  (*particle).e_curv = gamma*gamma*(*particle).mass*
                       absvalue((*particle).rxb_vel)*absvalue((*particle).rxb_vel)
                       /(Clight*Clight*(gamma+1));

  (*particle).e_grad = gamma*gamma*(*particle).mass*
                       absvalue((*particle).dbxb_vel)*absvalue((*particle).dbxb_vel)
                       /(Clight*Clight*(gamma+1));

  (*particle).e_ExB  = gamma*gamma*(*particle).mass*
                       absvalue((*particle).exb_vel)*absvalue((*particle).exb_vel)
                       /(Clight*Clight*(gamma+1));

  (*particle).e_perp = (*particle).e_perp + (*particle).e_curv;
  (*particle).e_para = (*particle).e_para - (*particle).e_curv;
  (*particle).e_cycl = (*particle).e_perp - (*particle).e_grad;
  (*particle).e_perp_all = (*particle).e_perp + (*particle).e_ExB;

  if ((*particle).e_para < (*particle).e_para_min)
```

```
    {
      if ((*particle).shrinkfactor > MIN_SHRINK_FACTOR)
{
  *e_tag = *e_tag + 2;
  // when parallel energy below zero
  // there has to be a mirror point, therefor e_tag is set to 2
}
      else
{
          (*particle).e_para = (*particle).e_para_min/2.;
  (*particle).shrinkfactor =  1.;
}
    }
  (*particle).e_kin = (*particle).e_para + (*particle).e_perp_all; // full kinetic energy
}



/******************* calc_velocities  **********************/
void calc_velocities(struct particle_data* particle, struct coils* spulen, struct bfield_geom* geom)
{
  double v_value;
  double v_trafo;

  if ((*particle).shrinkfactor < 1.)  // expand shrinkfactor if lower than 1
    unshrink_step(particle);
  else (*particle).shrinkfactor = 1.; // set default value for shrinkfactor

  (*particle).v_perp_value = (*particle).v_perp_start_value*
                             sqrt((*particle).b_value/(*particle).b_start_value)*
                             get_gamma(particle)/(*particle).gamma_start;
  // calculate perpendicular velocity value by squareroot of the fraction of B to Bstart

  v_value = sqrt((*particle).e_kin*(*particle).e_kin +               // velocity here
 2.*(*particle).e_kin*(*particle).mass)*Clight/((*particle).e_kin +
  (*particle).mass);
  // calculate value of velocity by using relativistic expression

  v_trafo = (get_gamma(particle) + 1.)*(*particle).e_para/
    ((*particle).mass*get_gamma(particle)*get_gamma(particle)); // new relativistic version

  vector_times_scalar((*particle).b_vec,
      (*particle).v_signum*
      sqrt(v_trafo)*Clight/(*particle).b_value,
      (*particle).v_para); // make parallel velocity vector

  (*particle).v_para_value = absvalue((*particle).v_para); // value of v_para
}

/******************* calc_step **********************/
void calc_step(struct particle_data* particle, struct coils* spulen, struct bfield_geom* geom)
{
  double all_drifts[3];

  // cut the steplength to value given by max_step_length
  vector_times_scalar((*particle).v_para,
      (*particle).max_step_length/(*particle).v_para_value,
```

```
      (*particle).step_final);

  do {
    vector_times_scalar((*particle).step_final,(*particle).shrinkfactor,(*particle).step_final);
    // this is the place where the shrinkfactor is finaly used to shorten the step
    runge_kutta_midpoint(particle,spulen,geom);  // use now the runge kutta approximation

    if ((*particle).v_para_value <= 0.)
      {
printf("ERROR: devide by zero in TOF calculation!\n");
      }
    (*particle).delta_tof = absvalue((*particle).step_final)*10000./(*particle).v_para_value;//in æs
    // tof for actual step in æs
    (*particle).delta_tof = (*particle).delta_tof/(1000000.); // in sec
    // and transform into sec

    vector_times_scalar(all_drifts,0.,all_drifts); // earase all_drift vector
    vector_sum(all_drifts,(*particle).exb_vel,all_drifts);// add to all_drift vector
    vector_sum(all_drifts,(*particle).rxb_vel,all_drifts); // add to all_drift vector
    vector_sum(all_drifts,(*particle).dbxb_vel,all_drifts); // add to all_drift vector

    vector_times_scalar(all_drifts,(*particle).delta_tof*M2CM,(*particle).v_drift);
    // transform all_drift velocity vector to v_drift distance vector with factor 100 for cm
    vector_sum((*particle).step_final,(*particle).v_drift,(*particle).step_final);
    // add v_drift to step_final vector

    if (absvalue((*particle).step_final) > (*particle).max_step_length) shrink_step(particle);
    // test max_step_length again
  }
  while (absvalue((*particle).step_final) > (*particle).max_step_length);
  // repeat this until step_final is shorter then max_step_length
}



/******************** single_track ***********************/
void single_track(FILE* f_fly,
  struct coils* spulen, struct bfield_geom* geom,
  struct particle_data* particle)
{
  // Routine for Particle Tracking, runs one single track

  int e_tag = 0;          // error tag
  int loop_counter;       // counter to prevent infinite loops
  double tof_backup = 0.; // time backup for pulsing control
  int got_turn = 0;       // stores turning point information of last move
  double dipol_value;     // backup for dipol value, used by pulsing control
  struct particle_data probe;

  init_particle(particle,spulen,geom);        // init particle data block
  calc_energies(particle,&e_tag,spulen,geom); // energies for the first time
  calc_velocities(particle,spulen,geom);      // velocities for the first time
  printf("\n");
  (*particle).cyclrad = get_cyclrad(particle);
  output(particle);
  savedata(f_fly,particle);
  time_reset_all();
```

```
  start_timer(2);
  dipol_value = dipol;
  loop_counter = 0;
  do {
    loop_counter++;
    calc_step(particle,spulen,geom);          // do the step calculation
    go_step(particle,&probe);                 // go forward
    e_tag = 0;                                // reset e_tag
    calc_energies(&probe,&e_tag,spulen,geom); // recalculate energies
    if ((got_turn > 0) && (probe.e_para > probe.e_para_min)) got_turn = 0;
    if (e_tag < 2)   // in case of no mirror point = energy minimum
      {
if (e_tag == 0)
  {
    if ((probe.e_para <= probe.e_para_min) && (got_turn == 0))
      // no collision and energy lower minimum energy
      {
turn_direction(&probe);    // turn direction of motion
got_turn = 1;
loop_counter = 0;
save_data_count = SAVE_EVERY; // force data to be saved
      }

    e_tag = test_electrode(particle,&probe);  // test collision with electrode
    if (test_drift(&probe) != 0) e_tag = e_tag + 10*test_drift(&probe);

    (*particle) = probe;          // store data in probe back in particle
    (*particle).time_of_flight = (*particle).time_of_flight + (*particle).delta_tof*1000000.; // add
    (*particle).cyclrad = get_cyclrad(particle);
    output(particle);// output if no error and not first point
    savedata(f_fly,particle);
  }
calc_velocities(particle,spulen,geom); // recalculate velocity
      }
    else   // in case of approaching mirror point shrink step length
      {
shrink_step(particle);
e_tag = e_tag - 2;  // remove error code from error detection tag
      }
  } while ((((*particle).position[0] < SPEC_OUT)  // repeat until particle out of range
    && ((*particle).position[0] > SPEC_IN))
   && ((*particle).mirrors <= (*particle).max_mirrors) // or mirror count reached
   && ((*particle).time_of_flight < (*particle).max_tof*1000000.)
   && (e_tag == 0)
   && (loop_counter <= MAX_LOOPS));           // or error happend
  if (loop_counter > MAX_LOOPS) printf("EXIT by loop count overflow!\n");
  if (((*particle).position[0] > SPEC_OUT)
      || ((*particle).position[0] < SPEC_IN)) printf("EXIT by Position! --> splat on detector\n");
  if ((*particle).mirrors > (*particle).max_mirrors) printf("EXIT by Mirrors! --> trapped?\n");
  if ((*particle).time_of_flight >= (*particle).max_tof*1000000.) printf("EXIT by TOF! --> trapped?\n
  if (e_tag != 0) printf("EXIT by Energy Tag! --> splat on Electrode\n");
  if (e_tag >= 10) printf("EXIT by relativistic Drift --> ADIPARK unable to calculate!\n");
  stop_timer(2);
  store_times();
}  // end of routine single_track
```

```
/******************* single_track_se  **********************/
int single_track_se(struct coils* spulen, struct bfield_geom* geom,
    struct particle_data* particle)
{
  // Routine for Particle Tracking, runs one single track
  // small version for special calculations

  int loop_counter;
  int e_tag = 0;     // error tag
  int exit_tag = 0;  // exit tag for trapping volume routine
  int got_turn = 0;  // detects a turn of direction
  struct particle_data probe;

  init_particle_se(particle,spulen,geom);        // init particle data block
  calc_energies(particle,&e_tag,spulen,geom); // energies for the first time
  calc_velocities(particle,spulen,geom);      // velocities for the first time
  output(particle);
  printf("%f %f %f %f %f\n",
 (*particle).position[0],(*particle).position[1],(*particle).position[2],
 (*particle).e_start,180*asin(sqrt((*particle).sin2_alpha_start))/M_PI);
  loop_counter = 0;
  do {
    loop_counter++;
    calc_step(particle,spulen,geom);          // do the step calculation
    go_step(particle,&probe);                 // go forward
    e_tag = 0;                                // reset e_tag
    calc_energies(&probe,&e_tag,spulen,geom); // recalculate energies

    if ((got_turn > 0) && (probe.e_para > probe.e_para_min)) got_turn = 0;
    if (e_tag < 2)   // in case of no mirror point = energy minimum
      {
if (e_tag == 0)
  {
    if ((probe.e_para <= probe.e_para_min) && (got_turn == 0))
      // no collision and energy lower minimum energy
      {
turn_direction(&probe);    // turn direction of motion
got_turn = 1;
loop_counter = 0;
      }

    e_tag = test_electrode(particle,&probe);  // test collision with electrode
    if (test_drift(&probe) != 0) e_tag = e_tag + 10*test_drift(&probe);

    (*particle) = probe;          // store data in probe back in particle
    (*particle).time_of_flight = (*particle).time_of_flight + (*particle).delta_tof*1000000.; // add to
    (*particle).cyclrad = get_cyclrad(particle);
  }
calc_velocities(particle,spulen,geom); // recalculate velocity
      }
    else   // in case of approaching mirror point shrink step length
      {
shrink_step(particle);
loop_counter++;
if (loop_counter <= MAX_LOOPS) e_tag = e_tag - 2;
                                  // remove error code from error detection tag
      }
```

```
  } while ((((*particle).position[0] < SPEC_OUT)  // repeat until particle out of range
    && ((*particle).position[0] > SPEC_IN))
   && ((*particle).mirrors <= (*particle).max_mirrors) // or mirror count reached
   && ((*particle).time_of_flight < (*particle).max_tof*1000000.)
   && (e_tag == 0)
   && (loop_counter <= MAX_LOOPS));           // or error happend

  exit_tag = 0; // not trapped
  if (e_tag >= 10)
    {
      printf("EXIT by relativistic Drift --> ADIPARK unable to calculate!\n");
      exit_tag = 3;
      e_tag = 0;
    }
  if (loop_counter > MAX_LOOPS)
    {
      printf("possible infinite loop -> force exit! \n");
      exit_tag = 2;
    }
  if (((*particle).position[0] > SPEC_OUT)
      || ((*particle).position[0] < SPEC_IN)) printf("Out of spectrometer!\n");
  if ((*particle).mirrors > (*particle).max_mirrors)
    {
      printf("Particle trapped! (mirrors)\n");
      exit_tag = 1;
    }
  if ((*particle).time_of_flight >= (*particle).max_tof*1000000.)
    {
      printf("Particle trapped! (tof)\n");
      exit_tag = 1;
    }
  if (e_tag != 0)
    {
      printf("Impact on electrode! (e-tag)\n");
      exit_tag = 0;
    }
  return exit_tag;
}  // end of routine single_track



/******************* tracking_loop ***********************/
void tracking_loop(struct coils* spulen, struct bfield_geom* geom, char* filename)
{
  // Loop for Particle Tracking of more than one Particle

  struct particle_data particle;    // structure containing particle data
  int particle_count = 0;           // counter for particles
  char* track_filename[30];         // filename for tracking data
  char* run_filename[30];         // filename for parameter list
  char* charline[100];              // commentline in parameter file
  FILE* f_run;                    // parameter input file
  FILE* f_track;                    // output file

  sprintf(run_filename,"%s.run",filename);// data input from file with extension .run
  f_run = fopen(run_filename, "r");
```

```
  printf("Particle Tracking parameter file: %s\n",run_filename);

  if (f_run == (FILE *)0)                // disk error handling
    {
      fprintf(stderr, "ERROR: Can't read parameter file %s\n",run_filename);
      exit(1);
    }
  else
    {
      fscanf(f_run,"%s\n",&charline);
      fscanf(f_run,"%s\n",&charline);
      fscanf(f_run,"%d %d %d %lf %d %lf %lf %lf\n",&display,&particle.calc_order,
     &retard_onoff,&dipol,
     &particle.max_mirrors,&particle.max_tof,&particle.e_para_min,
     &particle.max_step_length);
      fscanf(f_run,"%s\n",&charline);
      printf("Display Mode      = %d \n",display);
      printf("MAX_MIRRORS       = %d \n",particle.max_mirrors);
      printf("MAX_TOF[s]        = %4.2f \n",particle.max_tof);
      printf("E_PARA_MIN        = %11.9f \n",particle.e_para_min);
      printf("MAX_STEP_LENGTH   = %2.1f \n",particle.max_step_length);
      printf("Dipol Value       = %4.2f V/m\n",dipol);
      printf("calculate       : ");
      if (particle.calc_order == 0) printf("all drifts\n");
      if (particle.calc_order == 1) printf("only ExB drift\n");
      if (particle.calc_order == 2) printf("only curvature and gradient drift\n");
      if (particle.calc_order > 2) printf("no drift\n");
    }
  printf("\n----------------------------------------\n");

  while (fscanf(f_run,"%lf %lf %lf %lf %lf %lf %lf %lf\n",
&particle.start_pos[0],&particle.start_pos[1],&particle.start_pos[2],
&particle.e_start,&particle.starting_theta,&particle.starting_phi,
&particle.mass,&particle.charge)!=EOF)
    {// read data out of file until file ends
      particle_count++;
      particle.mass = particle.mass*M0Clight2;
      if (particle.e_start == 0.)     //< particle.e_para_min)
      {
particle.e_start = particle.e_para_min;
printf("warning: starting energie low, set to E_PARA_MIN for calculation!\n");
      }
      printf("starting run %d.\n\n",particle_count);

      sprintf(track_filename,
      "%s.track%d",
      filename,particle_count);        // data output to file with extension .track
      f_track = fopen(track_filename, "w");
      printf("Particle Tracking output file: %s\n",track_filename);
      if (f_track == (FILE *)0)                 // disk error handling
{
  fprintf(stderr, "ERROR: Can't write output file %s\n",track_filename);
  exit(1);
}

      single_track(f_track, spulen, geom, &particle); // start a single track simulation
```

```
      printf("\nfinished run %d.\n",particle_count);
      printf("----------------------------------------\n");

      fclose(f_track);
    }
  fclose(f_run);           // close all files
} // end of tracking_loop
```

A.3 Modul für Speichervolumenkarten: trapping.c

```
#include <stdio.h>
#include <math.h>
#include "global.h"
#include "b_struct.h"
#include "tracking.h"

#define theta_step 10.
#define start_e 32.
#define end_e 8.
#define start_theta 80.
#define start_phi 0.
#define include_neg_y 1
#define y_plane 1
#define z_plane 2

int trapping(struct coils* spulen, struct bfield_geom* geom)
                  //parameters of the stored electrons
{

  struct particle_data particle;    // structure containing particle data
  struct s_trap trap;
  double theta,energy;
  double position[3];
  double phi;
  double y_max;
  double y_limit;
  int e_tag =0;
  double trapped = 0.;
  int trap_tag = 0;
  FILE *log;
  FILE *f;

  f=fopen("trapping_vol.dat","w");
    if(!f)
      return 1;

  log=fopen("trapping_vol.log","w");
    if(!log)
      return 1;

  particle.max_mirrors = 250;      // tracking stops after this count of mirrors
  particle.mass = M0Clight2;      // electron mass
  particle.charge = 1.;           // electron charge
  particle.max_step_length = 0.1; // steplength of 1 mm
```

```
  particle.e_para_min = 0.01;     // minimum of parallel energy for mirror points
  particle.max_tof = 0.00003;       // break after this tof
  particle.calc_order = 0;        // order of drift calculations (0 all; 3 nothing)

    // x-direction loop
  y_limit = (*geom).y_max;

  if (include_neg_y == 1) printf("Remark: negative volume included!\n");
  else printf("Remark: only positive volume included!\n");
  if ((y_plane == 1) && (z_plane == 2)) printf("Remark: calculate in x-y-plane!\n");
  else printf("Remark: calculate in x-z-plane!\n");

  for (position[0]=(*geom).x_max; position[0]>=(*geom).x_min;position[0]-=(*geom).stepx)
    {
      position[z_plane]=0.;
      y_max = 0.;
      e_tag = 0;
      do
{
  position[y_plane] = y_max;
  y_max = y_max + MM_PER_UNIT/20.;
  epot3d(position,&e_tag);
}
      while ((y_max <= y_limit) && (e_tag == 0));
      y_limit = y_max;
      printf("x y_limit %f %f\n",position[0],y_max);
      position[y_plane] = (*geom).y_min;
      if (include_neg_y == 1) position[y_plane] = -(*geom).y_max;
      e_tag =0;
      epot3d(position,&e_tag);
      fprintf(f," \n");
      while (/*(e_tag == 0) &&*/ (position[y_plane] <= (*geom).y_max))        // y-direction loop
{
  for(energy = start_e;energy>= end_e;energy=energy*0.5)  // energy loop
    {
      fprintf(f,"%f %f %f ",position[0],position[y_plane],energy);
      theta=start_theta;
      phi=start_phi;
      particle.starting_theta = theta;
      particle.starting_phi = phi;
      particle.e_start = energy;
      if (particle.e_start == 0.) particle.e_start=energy+particle.e_para_min;
      particle.start_pos[0] = position[0];
      particle.start_pos[y_plane] = position[y_plane];
      particle.start_pos[z_plane] = position[z_plane];
      trapped = 0.;

      do                  //theta loop
{
  trap_tag == 0;
  if (sqrt(position[y_plane]*position[y_plane]) <= y_limit)
    trap_tag = single_track_se(spulen,geom,&particle);
  if (trap_tag == 2)
    {
      fprintf(log,"ADIPARK broken loop at (x,y,z,E,a) %f %f %f %f %f\n",
     position[0],position[1],position[2],energy,theta);
    }
```

```
  if (trap_tag == 3)
    {
      fprintf(log,"ADIPARK relativistic ERROR at (x,y,z,E,a) %f %f %f %f %f\n",
     position[0],position[1],position[2],energy,theta);
    }
  if (trap_tag == 1)
    {
      trap.trap_theta = theta;
      trap.trap_energy=energy;
      trap.trap_position[0]=position[0];
      trap.trap_position[y_plane]=position[y_plane];
      trap.trap_position[y_plane]=position[y_plane];
      trapped++;
      theta=theta-theta_step;
      particle.starting_theta = theta;
      if (theta <= 1.) theta = -1;
    }
  else
    {
      theta= -1;
    }
}
      while (theta> -1);
      if (sqrt(position[y_plane]*position[y_plane]) > y_limit) trapped = 0.;
      fprintf(f,"%f\n",trapped);
    }
  position[y_plane]=position[y_plane]+(*geom).stepy;
  e_tag = 0;
  epot3d(position,&e_tag);

}
    }
  fclose(f);
  fclose(log);
}
```

A.4 Elektrische Feldkartenverwaltung: efield.c

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <math.h>
#include "matrix.h"
#include "math_vector.h"
#include "global.h"


static int nx,ny,nz,sym;
static double* electric_potential_array;
static double* efield_z_array;
static double* efield_r_array;
static double maxvoltage;

double get_max_voltage()
```

```
{
  return maxvoltage;
}

int get_symmetry()
{
  return sym;
}

void alloc_electric_arrays(){
  electric_potential_array=malloc(N_ARRAY*sizeof(double));
  efield_z_array=malloc(N_ARRAY*sizeof(double));
  efield_r_array=malloc(N_ARRAY*sizeof(double));
  return;
}

void free_electric_arrays(){
  free(electric_potential_array);
  return;
}


void read_epot(char* el_filename) {

  int     pa_in;
  int     x,y,z;

  double  pot;
  char* pa_in_name[30];

  struct  header_3d {
    int mode;
    int symmetry;
    double max_voltage;
    int nx;
    int ny;
    int nz;
    int mirror;};

  struct  header_3d h;

  sprintf(pa_in_name,"%s.pa0",el_filename);// data input from file with extension .pa0
  printf("\nNow reading out the electric PA-File %s...", pa_in_name);
  pa_in  = open(pa_in_name, O_RDONLY );  /* open electric PA-File in read mode */
  read(pa_in,&h,sizeof(h)); /* read the header out of the file */

  maxvoltage = h.max_voltage;
  sym = h.symmetry;
  printf("\nInformation out of the file´s header:\n");
  printf("  mode %d\n",h.mode);
  printf("  symmetrie %d\n",sym);
  printf("  max_voltage %f\n",maxvoltage);
  printf("  Potentials at electrodes = potential + %f\n",2*h.max_voltage);
  printf("  mirror %d\n",h.mirror);
  printf("  nx, ny, nz : %d,%d,%d\n",  h.nx, h.ny, h.nz);
  nx = h.nx;
  ny = h.ny;
```

```
  nz = h.nz;
  for(z=0;z < h.nz; z++) {
    for(y=0;y < h.ny; y++) {
      for(x=0;x < h.nx; x++) {
        read(pa_in, &pot, sizeof(double));   /* read potential of an (x/y/z)-Koord. */
        if (pot>h.max_voltage)
  {
    pot = pot - h.max_voltage;
  }
write_matrix_value(electric_potential_array ,h.nx, h.ny, h.nz, 0, x, y, z, pot);
      }
    }
  }
  printf("Done.\n");
  close(pa_in);
}


double get_epot3D(double *position_cm,int *splat)
{
  double oktett[8];       // potential values at all 8 corners
  double pos_backup[3];   // real position without any mirroring
  double position[3];     // position used for epot calculation
  double epot_interpol;   // interpolated potential value
  int pos_left[3];     // position of lower left corner in front - position of oktett[0]
  int pos_right[3];    // position of upper right corner in back - position of oktett[5]
  int electrode_count; // counting points located in electrodes
  int comp;

  *splat = 0;
  vector_times_scalar(position_cm,1.,position);                // copy position vector
  position[0] = (position[0]-X_OFFSET_IN_CM)*CM2MM/MM_PER_UNIT; // transform to simion grid
  position[1] = (position[1]-Y_OFFSET_IN_CM)*CM2MM/MM_PER_UNIT;
  position[2] = (position[2]-Z_OFFSET_IN_CM)*CM2MM/MM_PER_UNIT;
  vector_times_scalar(position,1.,pos_backup);                  // backup position vector

  for (comp = 0; comp <=2; comp++)
    {
      pos_left[comp] = (int)(position[comp] + 5000.) - 5000;    // corner 0
      // 5000 shifts all to positive than truncate and shift back
      // this is only needed when negative position
      if (pos_left[comp] < 0.)
{
  pos_left[comp] = -pos_left[comp]; // use positive value
  pos_right[comp] = pos_left[comp] - 1;                      // corner 5
}
      else pos_right[comp] = pos_left[comp] + 1;                      // corner 5
      if (position[comp] < 0.) position[comp]= -position[comp]; // use positive values
    }
  if ((position[0]>nx)||(position[1]>ny)||(position[2]>nz))
    {
      return maxvoltage;                                        // watch for electrodes
    }
  else
    {                          // read potential in all 8 corners of cubus
      oktett[0] = read_matrix_value(electric_potential_array, nx, ny, nz, 0,
    pos_left[0], pos_left[1], pos_left[2]);
```

```
      oktett[1] = read_matrix_value(electric_potential_array, nx, ny, nz, 0,
    pos_right[0], pos_left[1], pos_left[2]);
      oktett[2] = read_matrix_value(electric_potential_array, nx, ny, nz, 0,
    pos_right[0], pos_right[1], pos_left[2]);
      oktett[3] = read_matrix_value(electric_potential_array, nx, ny, nz, 0,
    pos_left[0], pos_right[1], pos_left[2]);
      oktett[4] = read_matrix_value(electric_potential_array, nx, ny, nz, 0,
    pos_left[0], pos_right[1], pos_right[2]);
      oktett[5] = read_matrix_value(electric_potential_array, nx, ny, nz, 0,
    pos_right[0], pos_right[1], pos_right[2]);
      oktett[6] = read_matrix_value(electric_potential_array, nx, ny, nz, 0,
    pos_right[0], pos_left[1], pos_right[2]);
      oktett[7] = read_matrix_value(electric_potential_array, nx, ny, nz, 0,
    pos_left[0], pos_left[1], pos_right[2]);
      electrode_count = 0;
      for (comp = 0; comp <=7; comp++)             // patch epot when in electrode
{
  if (oktett[comp] > 0.)
    {
      oktett[comp] = oktett[comp] - maxvoltage;
      electrode_count++;
    }
}
      for (comp = 0; comp <=2; comp++)
{                                        // redo corner 0 and 5 for interpolation
  pos_left[comp] = (int)(pos_backup[comp]+5000.) - 5000;
                                                 // now positive and negative values needed
  pos_right[comp] = pos_left[comp] + 1;
}
      epot_interpol = interpol_3dim(pos_backup,pos_left,pos_right,oktett);
      // do interpolation
      if (electrode_count >= INTERPOL_SPLAT) *splat = 1;
      // if too many corners in electrode than interpolated point in electrode
      return epot_interpol; // return epot value
    }
}

double get_epot2D(double *position_cm,int *splat)
{
  double oktett[8];       // potential values at all 8 corners
  double pos_backup[3];   // real position without any mirroring
  double position[3];     // position used for epot calculation
  double epot_interpol;   // interpolated potential value
  int pos_left[3];     // position of lower left corner in front - position of oktett[0]
  int pos_right[3];    // position of upper right corner in back - position of oktett[5]
  int electrode_count; // counting points located in electrodes
  int comp;

  *splat = 0;
  vector_times_scalar(position_cm,1.,position);                // copy position vector
  position[0] = (position[0]-X_OFFSET_IN_CM)*CM2MM/MM_PER_UNIT; // transform to simion grid
  position[1] = (position[1]-Y_OFFSET_IN_CM)*CM2MM/MM_PER_UNIT;
  position[2] = 0.;
  vector_times_scalar(position,1.,pos_backup);                  // backup position vector

  for (comp = 0; comp <=2; comp++)
    {
```

```
      pos_left[comp] = (int)(position[comp] + 5000.) - 5000;    // corner 0
      // 5000 shifts all to positive than truncate and shift back
      // this is only needed when negative position
      if (pos_left[comp] < 0.)
{
  pos_left[comp] = -pos_left[comp]; // use positive value
  pos_right[comp] = pos_left[comp] - 1;                     // corner 5
}
      else pos_right[comp] = pos_left[comp] + 1;                      // corner 5
      if (position[comp] < 0.) position[comp]= -position[comp]; // use positive values
    }
  if ((position[0]>nx)||(position[1]>ny)||(position[2]>nz))
    {
      return maxvoltage;                                        // watch for electrodes
    }
  else
    {                          // read potential in all 8 corners of cubus
      oktett[0] = read_matrix_value(electric_potential_array, nx, ny, nz, 0,
    pos_left[0], pos_left[1], pos_left[2]);
      oktett[1] = read_matrix_value(electric_potential_array, nx, ny, nz, 0,
    pos_right[0], pos_left[1], pos_left[2]);
      oktett[2] = read_matrix_value(electric_potential_array, nx, ny, nz, 0,
    pos_right[0], pos_right[1], pos_left[2]);
      oktett[3] = read_matrix_value(electric_potential_array, nx, ny, nz, 0,
    pos_left[0], pos_right[1], pos_left[2]);
      oktett[4] = 0.;
      oktett[5] = 0.;
      oktett[6] = 0.;
      oktett[7] = 0.;

      electrode_count = 0;
      for (comp = 0; comp <=7; comp++)             // patch epot when in electrode
{
  if (oktett[comp] > 0.)
    {
      oktett[comp] = oktett[comp] - maxvoltage;
      electrode_count++;
    }
}
      for (comp = 0; comp <=2; comp++)
{                                      // redo corner 0 and 5 for interpolation
  pos_left[comp] = (int)(pos_backup[comp]+5000.) - 5000;
                                                 // now positive and negative values needed
  pos_right[comp] = pos_left[comp] + 1;
}
      epot_interpol = interpol_3dim(pos_backup,pos_left,pos_right,oktett);
      // do interpolation
      if (electrode_count >= INTERPOL_SPLAT) *splat = 1;
      // if too many corners in electrode than interpolated point in electrode
      return epot_interpol; // return epot value
    }
}



double epot(double z, double r, double mm_per_unit, double z_offset_in_cm) {
```

```
  double d_z, d_r, pot_left_below, pot_right_below, pot_right_above, pot_left_above;
  int i_z_left, i_z_right, i_r_below, i_r_above;

  if (z<0) z=-z;
  d_z=(z-z_offset_in_cm)*10./mm_per_unit;
  i_z_left  = (int)d_z;
  i_z_right = i_z_left+1;
  d_r=r*10./mm_per_unit;
  i_r_below = (int)d_r;
  i_r_above = i_r_below+1;
  if ((i_z_left<0)||(i_z_right>nx)||(i_r_below<0)||(i_r_above>ny))
    {
      return +1000000.;
    }
  else
    {
      pot_left_below  = read_matrix_value(electric_potential_array, nx, ny, nz, 0,
  i_z_left, i_r_below, 0);
      if (pot_left_below>=maxvoltage) pot_left_below = pot_left_below - maxvoltage;
      pot_right_below = read_matrix_value(electric_potential_array, nx, ny, nz, 0,
  i_z_right, i_r_below, 0);
      if (pot_right_below>=maxvoltage) pot_right_below = pot_right_below - maxvoltage;
      pot_right_above = read_matrix_value(electric_potential_array, nx, ny, nz, 0,
  i_z_right, i_r_above, 0);
      if (pot_right_above>=maxvoltage) pot_right_above = pot_right_above - maxvoltage;
      pot_left_above  = read_matrix_value(electric_potential_array, nx, ny, nz, 0,
  i_z_left, i_r_above, 0);
      if (pot_left_above>=maxvoltage) pot_left_above = pot_left_above - maxvoltage;
      return interpol_2dim(d_z, d_r,
   (float)i_z_left,(float)i_z_right,
   (float)i_r_below,(float)i_r_above,
   pot_left_below, pot_right_below,
   pot_right_above,pot_left_above);
    }
}


void pot2efield(double mm_per_unit) {

  int r,z;

  printf("\nNow making electric field out of electric potential...\n");

  for (z=1; z<=nx; z++) {
    for (r=1; r<=ny; r++) {

      write_matrix_value( efield_z_array, nx, ny, nz, 0, z, r, 0,
  ((read_matrix_value(electric_potential_array, nx, ny, nz, 0, z-1, r, 0))
   - (read_matrix_value(electric_potential_array, nx, ny, nz, 0, z+1, r, 0)))
  * (1000./(2*mm_per_unit)) );

      write_matrix_value( efield_r_array, nx, ny, nz, 0, z, r, 0,
  ((read_matrix_value(electric_potential_array, nx, ny, nz, 0, z, r+1, 0))
   - (read_matrix_value(electric_potential_array, nx, ny, nz, 0, z, r-1, 0)))
  * (1000./(2*mm_per_unit)) );
    }
  }
```

```
  z = 0;
  for (r=1; r<=ny; r++) {
    write_matrix_value( efield_z_array, nx, ny, nz, 0, z, r, 0,
((read_matrix_value(electric_potential_array, nx, ny, nz, 0, z, r, 0))
 - (read_matrix_value(electric_potential_array, nx, ny, nz, 0, z+1, r, 0)))
* (1000./(mm_per_unit)) );
    write_matrix_value( efield_r_array, nx, ny, nz, 0, z, r, 0,
((read_matrix_value(electric_potential_array, nx, ny, nz, 0, z, r+1, 0))
 - (read_matrix_value(electric_potential_array, nx, ny, nz, 0, z, r-1, 0)))
* (1000./(2*mm_per_unit)) );

  }

  r = 0;
  for (z=1; z<=nx; z++) {

    write_matrix_value( efield_z_array, nx, ny, nz, 0, z, r, 0,
((read_matrix_value(electric_potential_array, nx, ny, nz, 0, z-1, r, 0))
 - (read_matrix_value(electric_potential_array, nx, ny, nz, 0, z+1, r, 0)))
* (1000./(2.*mm_per_unit)) );
    write_matrix_value( efield_r_array, nx, ny, nz, 0, z, r, 0,
((read_matrix_value(electric_potential_array, nx, ny, nz, 0, z, r+1, 0))
 - (read_matrix_value(electric_potential_array, nx, ny, nz, 0, z, r, 0)))
* (1000./(mm_per_unit)) );
  }
  printf("Done.\n");
}


void efield(double z, double r, double* ez, double* er, double mm_per_unit, double z_offset_in_cm) {

  double d_z, d_r, ez_signum = -1.0;
  int i_z_left, i_z_right, i_r_below, i_r_above;
  double ez_left_below, ez_right_below, ez_right_above, ez_left_above;
  double er_left_below, er_right_below, er_right_above, er_left_above;

  if (z<0) { z=-z; ez_signum = -ez_signum;}
  d_z=(z-z_offset_in_cm)*10./mm_per_unit;
  i_z_left  = (int)d_z;
  i_z_right = i_z_left+1;
  d_r=r*10./mm_per_unit;
  i_r_below = (int)d_r;
  i_r_above = i_r_below+1;
  if ((i_z_left<0)||(i_z_right>nx)||(i_r_below<0)||(i_r_above>ny)){
    *er = 0.;
    *ez = 0.;
  }
  else
    {
      ez_left_below  = read_matrix_value(efield_z_array, nx, ny, nz, 0,
 i_z_left, i_r_below, 0);
      ez_right_below = read_matrix_value(efield_z_array, nx, ny, nz, 0,
 i_z_right, i_r_below, 0);
      ez_right_above = read_matrix_value(efield_z_array, nx, ny, nz, 0,
 i_z_right, i_r_above, 0);
      ez_left_above  = read_matrix_value(efield_z_array, nx, ny, nz, 0,
```

```
 i_z_left, i_r_above, 0);
      *ez = ez_signum * interpol_2dim(d_z, d_r,
      (float)i_z_left,(float)i_z_right,
      (float)i_r_below,(float)i_r_above,
      ez_left_below, ez_right_below,
      ez_right_above,ez_left_above);

      er_left_below  = read_matrix_value(efield_r_array, nx, ny, nz, 0,
 i_z_left, i_r_below, 0);
      er_right_below = read_matrix_value(efield_r_array, nx, ny, nz, 0,
 i_z_right, i_r_below, 0);
      er_right_above = read_matrix_value(efield_r_array, nx, ny, nz, 0,
 i_z_right, i_r_above, 0);
      er_left_above  = read_matrix_value(efield_r_array, nx, ny, nz, 0,
 i_z_left, i_r_above, 0);

      *er = interpol_2dim(d_z, d_r,
  (float)i_z_left,(float)i_z_right,
  (float)i_r_below,(float)i_r_above,
  er_left_below, er_right_below,
  er_right_above,er_left_above);
    }
}
```

A.5 Magnetische Feldkartenverwaltung: mag_pa_reader.c

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <math.h>
#include "matrix.h"
#include "math_vector.h"
#include "b_struct.h"
#include "mag_3d.h"
#include "global.h"


static int nx,ny,nz,sym;
static double* mag_potential_array;

int get_magpa_sym()
{
  return sym;
}

void alloc_mag_arrays(){
  mag_potential_array=malloc(N_ARRAY*sizeof(double));
  return;
}

void free_mag_arrays(){
  free(mag_potential_array);
  return;
}
```

```
void read_mag_pa(char* mag_filename) {

  int     pa_in;
  int     x,y,z;

  double  pot;
  char* pa_in_name[30];

  struct  header_3d {
    int mode;
    int symmetry;
    double max_voltage;
    int nx;
    int ny;
    int nz;
    int mirror;};

  struct  header_3d h;

  sprintf(pa_in_name,"%s.pa",mag_filename);// data input from file with extension .magpa
  printf("\nNow reading out the magnetic PA-File %s...", pa_in_name);
  pa_in  = open(pa_in_name, O_RDONLY );  /* open electric PA-File in read mode */
  read(pa_in,&h,sizeof(h)); /* read the header out of the file */

  sym = h.symmetry;
  printf("\nInformation out of the file´s header:\n");
  printf("  mode %d\n",h.mode);
  printf("  symmetrie %d\n",sym);
  printf("  max_voltage %f\n",h.max_voltage);
  printf("  Potentials at electrodes = potential + %f\n",2*h.max_voltage);
  printf("  mirror %d\n",h.mirror);
  printf("  nx, ny, nz : %d,%d,%d\n",  h.nx, h.ny, h.nz);
  nx = h.nx;
  ny = h.ny;
  nz = h.nz;
  for(z=0;z < h.nz; z++) {
    for(y=0;y < h.ny; y++) {
      for(x=0;x < h.nx; x++) {
        read(pa_in, &pot, sizeof(double));   /* read potential of an (x/y/z)-Koord. */
        if (pot>h.max_voltage)
  {
    pot = pot - h.max_voltage;
  }
write_matrix_value(mag_potential_array ,h.nx, h.ny, h.nz, 0, x, y, z, pot);
      }
    }
  }
  printf("Done.\n");
  close(pa_in);
}

double get_magpot3D(double *position_cm)
{
  double oktett[8];       // potential values at all 8 corners
  double pos_backup[3];   // real position without any mirroring
```

```
  double position[3];     // position used for epot calculation
  double magpot_interpol;   // interpolated potential value
  int pos_left[3];     // position of lower left corner in front - position of oktett[0]
  int pos_right[3];    // position of upper right corner in back - position of oktett[5]
  int comp;

  vector_times_scalar(position_cm,1.,position);                 // copy position vector
  position[0] = (position[0]-MAG_X_OFFSET_IN_CM)*CM2MM/MAG_MM_PER_UNIT; // transform to simion grid
  position[1] = (position[1]-MAG_Y_OFFSET_IN_CM)*CM2MM/MAG_MM_PER_UNIT;
  position[2] = (position[2]-MAG_Z_OFFSET_IN_CM)*CM2MM/MAG_MM_PER_UNIT;
  vector_times_scalar(position,1.,pos_backup);                   // backup position vector

  for (comp = 0; comp <=2; comp++)
    {
      pos_left[comp] = (int)(position[comp] + 5000.) - 5000;    // corner 0
      // 5000 shifts all to positive than truncate and shift back
      // this is only needed when negative position
      if (pos_left[comp] < 0.)
{
  pos_left[comp] = -pos_left[comp]; // use positive value
  pos_right[comp] = pos_left[comp] - 1;                      // corner 5
}
      else pos_right[comp] = pos_left[comp] + 1;                      // corner 5
      if (position[comp] < 0.) position[comp]= -position[comp];  // use positive values
    }
  if ((position[0]>nx)||(position[1]>ny)||(position[2]>nz))
    {
      printf("ERROR: in get_magpot3D: coordinates outside BFIELD_3D potential map!\n");
      return -1.;                                          // watch for electrodes
    }
  else
    {                           // read potential in all 8 corners of cubus
      oktett[0] = read_matrix_value(mag_potential_array, nx, ny, nz, 0,
    pos_left[0], pos_left[1], pos_left[2]);
      oktett[1] = read_matrix_value(mag_potential_array, nx, ny, nz, 0,
    pos_right[0], pos_left[1], pos_left[2]);
      oktett[2] = read_matrix_value(mag_potential_array, nx, ny, nz, 0,
    pos_right[0], pos_right[1], pos_left[2]);
      oktett[3] = read_matrix_value(mag_potential_array, nx, ny, nz, 0,
    pos_left[0], pos_right[1], pos_left[2]);
      oktett[4] = read_matrix_value(mag_potential_array, nx, ny, nz, 0,
    pos_left[0], pos_right[1], pos_right[2]);
      oktett[5] = read_matrix_value(mag_potential_array, nx, ny, nz, 0,
    pos_right[0], pos_right[1], pos_right[2]);
      oktett[6] = read_matrix_value(mag_potential_array, nx, ny, nz, 0,
    pos_right[0], pos_left[1], pos_right[2]);
      oktett[7] = read_matrix_value(mag_potential_array, nx, ny, nz, 0,
    pos_left[0], pos_left[1], pos_right[2]);

      for (comp = 0; comp <=2; comp++)
{                                       // redo corner 0 and 5 for interpolation
  pos_left[comp] = (int)(pos_backup[comp]+5000.) - 5000;
                                                // now positive and negative values needed
  pos_right[comp] = pos_left[comp] + 1;
}
      magpot_interpol = interpol_3dim(pos_backup,pos_left,pos_right,oktett);
      // do interpolation
```

```
      return magpot_interpol; // return epot value
    }
}

double get_magpot2D(double *position_cm)
{
  double oktett[8];       // potential values at all 8 corners
  double pos_backup[3];   // real position without any mirroring
  double position[3];     // position used for epot calculation
  double magpot_interpol;   // interpolated potential value
  int pos_left[3];     // position of lower left corner in front - position of oktett[0]
  int pos_right[3];    // position of upper right corner in back - position of oktett[5]
  int comp;

  vector_times_scalar(position_cm,1.,position);                // copy position vector
  position[0] = (position[0]-MAG_X_OFFSET_IN_CM)*CM2MM/MAG_MM_PER_UNIT; // transform to simion grid
  position[1] = (position[1]-MAG_Y_OFFSET_IN_CM)*CM2MM/MAG_MM_PER_UNIT;
  position[2] = (position[2]-MAG_Z_OFFSET_IN_CM)*CM2MM/MAG_MM_PER_UNIT;
  position[1] = sqrt(position[1]*position[1]+position[2]*position[2]);
  position[2] = 0.;
  vector_times_scalar(position,1.,pos_backup);                  // backup position vector

  for (comp = 0; comp <=2; comp++)
    {
      pos_left[comp] = (int)(position[comp] + 5000.) - 5000;    // corner 0
      // 5000 shifts all to positive than truncate and shift back
      // this is only needed when negative position
      if (pos_left[comp] < 0.)
{
  pos_left[comp] = -pos_left[comp]; // use positive value
  pos_right[comp] = pos_left[comp] - 1;                      // corner 5
}
      else pos_right[comp] = pos_left[comp] + 1;                      // corner 5
      if (position[comp] < 0.) position[comp]= -position[comp]; // use positive values
    }
  if ((position[0]>nx)||(position[1]>ny)||(position[2]>nz))
    {
      printf("ERROR: in get_magpot2D: coordinates outside BFIELD_3D potential map!\n");
      return -1.;                                          // watch for electrodes
    }
  else
    {                          // read potential in all 8 corners of cubus
      oktett[0] = read_matrix_value(mag_potential_array, nx, ny, nz, 0,
    pos_left[0], pos_left[1], pos_left[2]);
      oktett[1] = read_matrix_value(mag_potential_array, nx, ny, nz, 0,
    pos_right[0], pos_left[1], pos_left[2]);
      oktett[2] = read_matrix_value(mag_potential_array, nx, ny, nz, 0,
    pos_right[0], pos_right[1], pos_left[2]);
      oktett[3] = read_matrix_value(mag_potential_array, nx, ny, nz, 0,
    pos_left[0], pos_right[1], pos_left[2]);
      oktett[4] = 0.;
      oktett[5] = 0.;
      oktett[6] = 0.;
      oktett[7] = 0.;

      for (comp = 0; comp <=2; comp++)
{                                       // redo corner 0 and 5 for interpolation
```

```
  pos_left[comp] = (int)(pos_backup[comp]+5000.) - 5000;
                                                  // now positive and negative values needed
  pos_right[comp] = pos_left[comp] + 1;
}
      magpot_interpol = interpol_3dim(pos_backup,pos_left,pos_right,oktett);
      // do interpolation
      return magpot_interpol; // return epot value
    }
}


void get_bfield(double *position, double *bvec, struct coils *spulen, struct bfield_geom *geom)
{
  double pos1[3],pos3[3];
  double delta = MAG_MM_PER_UNIT;
  int komp;

  if (USE_MAG_PA == 0) bfield3d_all(position,bvec,spulen,geom);
  else {
    delta = delta*MM2CM;   // delta in mm must be used in cm
    if (delta == 0.) printf("Error: MAG_MM_PER_UNIT constant not set!\n");

    if (get_magpa_sym() == 1)
      {
for (komp = 0; komp <= 2; komp++)
  {
    vector_times_scalar(position,1.,pos1);
    vector_times_scalar(position,1.,pos3);
    pos1[komp] = pos1[komp] - delta;
    pos3[komp] = pos3[komp] + delta;
    bvec[komp] = (get_magpot3D(pos3)-get_magpot3D(pos1))
                 *M2CM*MAG_MM_PER_UNIT*MM2CM/(2.*TESLA2GAUSS*delta);
  }
      }
    else {
      for (komp = 0; komp <= 2; komp++)
{
  vector_times_scalar(position,1.,pos1);
  vector_times_scalar(position,1.,pos3);
  pos1[komp] = pos1[komp] - delta;
  pos3[komp] = pos3[komp] + delta;
  bvec[komp] = (get_magpot2D(pos3) - get_magpot2D(pos1))
                       *M2CM*MAG_MM_PER_UNIT*MM2CM/(2.*TESLA2GAUSS*delta);
}
    }
  }
}
```

A.6 Mathematische Bibliothek: math_vector.c

```
// Interface-Datei von:
// Bibliothekname: math_vector.c
// written by Thomas Thümmler in January 2001
// extended by B.Flatt on 02/13/2001
// new features by B.Flatt and T.Thümmler
```

```
// merged by T.Thümmler on 06/08/2001
// math_vector.c supplies math function to work with vectors

#include <math.h>
#include "math_vector.h"

/* ************************* signum ********************** */
int signum(double x){
  if (x>=0) {
    return +1;     // returns +1 if input >= zero
  } else {
    return -1;     // returns -1 if input < zero
  }
}

/* ************************ absvalue ********************** */
double absvalue(double *vector)
{
  double value;

  value = sqrt(vector[0]*vector[0] + vector[1]*vector[1] + vector[2]*vector[2]);
                        // returns the absolute value of input vector
  return value;
}

/* ************************ scalar_prod ********************** */
double scalar_prod(double *vec1, double *vec2)
{
  double value;

  value = vec1[0]*vec2[0]+vec1[1]*vec2[1]+vec1[2]*vec2[2];
                       // returns the scalar product of two vectors
  return value;
}

/* ************************ vector_times_scalar ********************** */
void vector_times_scalar(double *vec, double value, double *result)
{
  int index;

  for (index = 0; index <= 2; index++)  // multiply a vector with a scalar
  {
    result[index] = vec[index] * value;
  }
}

/* ************************ vector_sum ********************** */
void vector_sum(double *vec1, double *vec2, double *sum)
{
  int index;

  for (index = 0; index <= 2; index++)  // returns the summe of two vectors
  {
    sum[index] = vec1[index] + vec2[index];
  }
}
```

```
/* ********************** angle ******************** */
double angle(double *vec1, double *vec2)
{
  double value;

  value = (180./M_PI)*acos((scalar_prod(vec1,vec2))/(absvalue(vec1)*absvalue(vec2)));
                              // returns the angle between two vectors in degree
  return value;
}

/* ********************** cross_prod ******************** */
void cross_prod(double *vec1,double *vec2,double *vec3)
{
  vec3[0] = (vec1[1]*vec2[2] - vec1[2]*vec2[1]);  // calculate the vector product
  vec3[1] = (vec1[2]*vec2[0] - vec1[0]*vec2[2]);
  vec3[2] = (vec1[0]*vec2[1] - vec1[1]*vec2[0]);
}

/* ********************** vec_rotate ******************** */
void spher2kart(double *vec1,double value, double theta,double phi)
{
  theta = M_PI*(theta)/180.0;
  phi = M_PI*(phi)/180.0;
  vec1[0] = value*cos(theta);
  vec1[1] = value*sin(theta)*cos(phi);
  vec1[2] = value*cos(theta)*sin(phi);
}

/* ********************** point_curvation ******************** */
void point_curvation(double *x1,double *x2,double *x3,double *radius)
{
 double p12[3],p23[3];
 int komp;

  for (komp = 0; komp <= 2; komp++)
  {
    p12[komp] = (x2[komp] - x1[komp]);      // calculate point to point vectors
    p23[komp] = (x3[komp] - x2[komp]);
  }

  vector_curvation(p12,p23,radius);
}

/* ********************** vector_curvation ******************** */
void vector_curvation(double *p12,double *p23,double *radius)
{
 double k0[3],p13[3],n[3],n1[3],n2[3];
 double a;
 int komp;

  for (komp = 0; komp <= 2; komp++)
  {
    p12[komp] = 0.5 * p12[komp];
    p23[komp] = 0.5 * p23[komp];
                      // vectors to the centre
    p13[komp] = p12[komp]+p23[komp];
                      // vector from origin of p12 to destination of p23
```

```
  }

  cross_prod(p12,p23,n);           // calculate normal vector

  cross_prod(n,p12,n1);            // calculate perpendicular vector of
                                   // point to point vector 1 in normal plane

  cross_prod(n,p23,n2);            // calculate perpendicular vector of
                                   // point to point vector 2 in normal plane

  a = 0;
  if ((n1[0]*n2[1] - n1[1]*n2[0]) != 0)
  {
    a = (p13[0]*n2[1] - p13[1]*n2[0]) / (n1[0]*n2[1] - n1[1]*n2[0]);
  }
  else
  {
    if ((n1[0]*n2[2] - n1[2]*n2[0]) != 0)
    {
      a = (p13[0]*n2[2] - p13[2]*n2[0]) / (n1[0]*n2[2] - n1[2]*n2[0]);
    }
    else
    {
      if ((n1[1]*n2[2] - n1[2]*n2[1]) != 0)
                              a = (p13[1]*n2[2] - p13[2]*n2[1])
                                  / (n1[1]*n2[2] - n1[2]*n2[1]);
      }
    }
   // calculates the multiplication faktor of perpendicular vector 1 to
   // the crosspoint of both perpendicular vectors

  for (komp=0; komp<=2; komp++)
  {
  k0[komp] = p12[komp] + a*n1[komp];        // calculate origin of radius vector
  radius[komp] = 2.*p12[komp] - k0[komp];   // calculate radius vector
  }

  radius[3] = 1./absvalue(radius);          // store curvation value in fourth component
}


void gradB(double delta,double b,double *b_delta, double *grB)
{
  grB[1] = (b_delta[1]-b)/delta;
}



/* ********************** gradB_perp ********************** */
void gradB_perp(double *Bvec,double *grB,double *grB_perp)
{
  double grB_para[3];

  vector_times_scalar(Bvec,-scalar_prod(Bvec,grB)/(absvalue(Bvec)*absvalue(Bvec)),grB_para);
  vector_sum(grB_para,grB,grB_perp);
}
```

```
/* ********************** angle_rad ********************** */
double angle_rad(double *vec1, double *vec2)
{
  double value;

  value = acos((scalar_prod(vec1,vec2))/(absvalue(vec1)*absvalue(vec2)));

  return value;
}

/* ********************** rotate_vec ********************** */
void rotate_vec(double *vec1,double *omega, double *vec3)
// rotates vector vec1 (output vec3) around rotation axis omega
// by angle |omega|
// see Bronstein: section 2.6.5.2.3, page 215,216
{
  double theta, stheta, ctheta, one_ctheta, alpha, beta, gamma;

  theta = absvalue(omega);
  ctheta = cos(theta);
  one_ctheta = 1.0-cos(theta);
  stheta = sin(theta);
  alpha = omega[0]/absvalue(omega);
  beta  = omega[1]/absvalue(omega);
  gamma = omega[2]/absvalue(omega);
  vec3[0] = vec1[0]*(ctheta + alpha*alpha*one_ctheta) +
            vec1[1]*(gamma*stheta + alpha*beta*one_ctheta) +
            vec1[2]*(-beta*stheta + alpha*gamma*one_ctheta);
  vec3[1] = vec1[0]*(-gamma*stheta + alpha*beta*one_ctheta) +
            vec1[1]*(ctheta + beta*beta*one_ctheta) +
            vec1[2]*(alpha*stheta + beta*gamma*one_ctheta);
  vec3[2] = vec1[0]*(beta*stheta + gamma*alpha*one_ctheta) +
            vec1[1]*(-alpha*stheta + gamma*beta*one_ctheta) +
            vec1[2]*(ctheta + gamma*gamma*one_ctheta);
  return;
}


/* ********************** rotate_vec_on_z_axis ********************** */
void rotate_vec_on_z_axis(double *vec1,double *vec2, double *vec3)
// rotates vector vec1 (output vec3) in such a way,
// that vec2 points in z-direction
// see Bronstein: section 2.6.5.2.3, page 215,216
{
  double vecz[3],rot_axis[3];
  double theta, stheta, ctheta, one_ctheta, alpha, beta, gamma, abs_rot_axis;
  vecz[0] = 0.0;
  vecz[1] = 0.0;
  vecz[2] = 1.0;
  theta = angle_rad(vec2,vecz);
  ctheta = cos(theta);
  one_ctheta = 1.0-cos(theta);
  stheta = sin(theta);
  cross_prod(vecz,vec2,rot_axis);
  abs_rot_axis = absvalue(rot_axis);
  if (abs_rot_axis==0.0) {
    vec3[0] = vec1[0];
```

```
    vec3[1] = vec1[1];
    vec3[2] = vec1[2];
  } else {
    alpha = rot_axis[0]/abs_rot_axis;
    beta  = rot_axis[1]/abs_rot_axis;
    gamma = rot_axis[2]/abs_rot_axis;
    vec3[0] = vec2[0]*(ctheta + alpha*alpha*one_ctheta) +
              vec2[1]*(gamma*stheta + alpha*beta*one_ctheta) +
              vec2[2]*(-beta*stheta + alpha*gamma*one_ctheta);
    vec3[1] = vec2[0]*(-gamma*stheta + alpha*beta*one_ctheta) +
              vec2[1]*(ctheta + beta*beta*one_ctheta) +
              vec2[2]*(alpha*stheta + beta*gamma*one_ctheta);
    vec3[2] = vec2[0]*(beta*stheta + gamma*alpha*one_ctheta) +
              vec2[1]*(-alpha*stheta + gamma*beta*one_ctheta) +
              vec2[2]*(ctheta + gamma*gamma*one_ctheta);
    printf("rotated vec2: %f %f %f\n",vec3[0],vec3[1],vec3[2]);
    vec3[0] = vec1[0]*(ctheta + alpha*alpha*one_ctheta) +
              vec1[1]*(gamma*stheta + alpha*beta*one_ctheta) +
              vec1[2]*(-beta*stheta + alpha*gamma*one_ctheta);
    vec3[1] = vec1[0]*(-gamma*stheta + alpha*beta*one_ctheta) +
              vec1[1]*(ctheta + beta*beta*one_ctheta) +
              vec1[2]*(alpha*stheta + beta*gamma*one_ctheta);
    vec3[2] = vec1[0]*(beta*stheta + gamma*alpha*one_ctheta) +
              vec1[1]*(-alpha*stheta + gamma*beta*one_ctheta) +
              vec1[2]*(ctheta + gamma*gamma*one_ctheta);
  }
  return;
}

/* ********************** spherical_coordinate  ********************** */
void spherical_coordinate(double *vec,double* r, double* theta, double* phi)
// gives back from vector vec spherical coordinates
{

  double vecz[3];

  vecz[0] = 0.0;
  vecz[1] = 0.0;
  vecz[2] = 1.0;
  printf("vec %f %f %f theta %f\n",vec[0],vec[1],vec[2],angle(vec,vecz));

  *r = absvalue(vec);
  *theta = angle_rad(vec,vecz);
  if (vec[1]==0.0) {
    *phi=signum(vec[0])*M_PI/2.0;
  } else {
    if (vec[1]>0) {
      *phi = atan(vec[0]/vec[1]);
    } else {
      *phi = M_PI + atan(vec[0]/vec[1]);
    }
  }
  return;
}
```

```
/* ********************** interpol_2dim ******************** */
double interpol_2dim(double d_z, double d_r,
                     double z_left,double z_right, double r_below, double r_above,
                     double y_left_below,  double y_right_below,
                     double y_right_above, double y_left_above){
  double t,u; //see NUMERICAL RECIPES in C, chapter 3.6, page 123
  t = (d_z-z_left)/(z_right-z_left); //relative distance in z
  u = (d_r-r_below)/(r_above-r_below);
  return    (1-t)*(1-u)*y_left_below  +   t  *(1-u)*y_right_below
         +    t *   u  *y_right_above + (1-t)*  u  *y_left_above;
}


/* ********************** interpol_3dim ******************** */
double interpol_3dim(double *position,
     int *pos_left, int *pos_right,
     double *oktett)
{
  int comp;
  double rel_pos[3];  // relative position
                      // see Numerical Recipes in C, chapter 3.6, page 123
                      // and use 3 dimentions
  for (comp = 0; comp <=2; comp++)
    {
      rel_pos[comp] = (position[comp] - (double)pos_left[comp])/
                      ((double)pos_right[comp] - (double)pos_left[comp]);
    }
  return
    (1.-rel_pos[0])*(1.-rel_pos[1])*(1.-rel_pos[2])*oktett[0] +
    (rel_pos[0])*   (1.-rel_pos[1])*(1.-rel_pos[2])*oktett[1] +
    (rel_pos[0])*   (rel_pos[1])*   (1.-rel_pos[2])*oktett[2] +
    (1.-rel_pos[0])*(rel_pos[1])*   (1.-rel_pos[2])*oktett[3] +
    (1.-rel_pos[0])*(rel_pos[1])*   (rel_pos[2])   *oktett[4] +
    (rel_pos[0])*   (rel_pos[1])*   (rel_pos[2])   *oktett[5] +
    (rel_pos[0])*   (1.-rel_pos[1])*(rel_pos[2])   *oktett[6] +
    (1.-rel_pos[0])*(1.-rel_pos[1])*(rel_pos[2])   *oktett[7];
}

/* ********************** end of module math_vector ******************** */
```

Anhang B

Parameterdateien

B.1 Globale Konstanten: global.h

```
#define MM_PER_UNIT 4.0      // Mainz 97 Spectrometer
//#define MM_PER_UNIT 4.0      // Mainz 97 Spectrometer 3D
//#define MM_PER_UNIT 5.0      // Katrin Spectrometer
#define X_OFFSET_IN_CM 0.0     // for electric potential array
#define Y_OFFSET_IN_CM 0.0   // -60.0 for dipol array
#define Z_OFFSET_IN_CM 0.0
#define MAG_MM_PER_UNIT 4.0      // for magnetic potential array
#define MAG_X_OFFSET_IN_CM -206.0
#define MAG_Y_OFFSET_IN_CM -48.0
#define MAG_Z_OFFSET_IN_CM -48.0
#define SPEC_IN -201.2           // x boundaries of spectrometer
#define SPEC_OUT 201.2
#define N_ARRAY 60000000         // Maximum array points for potential arrays
#define MAX_RADIUS 47.       // maximum radius of spectrometer in cm (this one for mainz)
#define MAX_LOOPS  100000    // maximum iteration loops for one move
#define Clight 299792458.0   // PDG 98
#define MOClight2 510999.06  // PDG 98
#define MM2CM 0.1            // scale mm value to cm value
#define M2CM 100.            // scale m value to cm value
#define CM2MM 10.            // scale cm value to mm value
#define TESLA2GAUSS 10000.    // scale tesla to gauss
#define USE_MAG_PA 1         // switch betreen bfield potential array and calculation
#define SAVE_EVERY 100.     // save every xxth data block
#define PULS_TIME  1.        // time of pulsing on in usec
#define MIN_SHRINK_FACTOR 0.000000001 // minimum of shrinkfactor
#define INTERPOL_SPLAT 1     // corners of interpolated cube or rectangle to be in electrode

static int display = 0;         // output switch mode: 0=nothing, 1=some, 2=all

struct particle_data
{
  double start_pos[3];              // origin of particle
  double position[3];               // actual position
  double e_start;                   // kinetic energy at start_pos
  double b_start[3];                // magnetic vector bat start_pos
  double b_start_value;             // value of b_start
  double u_start;                   // el. potential at start_pos
```

```
  double sin2_alpha_start;          // siný of starting alpha
  double starting_theta;            // 0 to 89 degree
  double starting_phi;              // 0 to 359 degree
  double gamma_start;               // rel. gamma factor at start
  double b_vec[3];                  // actual magnetic vector
  double b_value;                   // value of b_vec
  double v_para[3];                 // actual parallel velocity vector
  double v_para_value;              // value of it
  double v_signum;                  // director relative to magnetic field line
  double v_perp_start[3];           // perpendicular velocity vector at start_pos
  double v_perp_start_value;        // and its value
  double v_perp_value;              // actual perp velocity value
  double v_drift[3];                // velocity vector for all drift parts
  double exb_vel[3];                // vector for exb drift velocity
  double dbxb_vel[3];               // vector for gradient dirft velocity
  double rxb_vel[3];                // vector for curvature dirft velocity
  double e_para;                    // parallel energy
  double e_para_min;                // lower energy limit
  double e_perp;                    // perpendicular energy only of magnetic motion, without ExB
  double e_kin;                     // kinetic energy
  double e_pot;                     // potential energy (el. field)
  double e_perp_all;                // kinetic energy in all perpendicular motions, with ExB
  double e_cycl;                    // kinetic energy only of cyclotron and gradientdrift motion
  double e_ExB;                     // kinetic energy in ExB motion
  double e_curv;                    // kinetic energy in curvation motion
  double e_grad;                    // kinetic energy in gradient drift motion
  double cyclrad;                   // cyclotron radius in cm
  double mass;                      // mass in eV
  double charge;                    // charge in e
  double time_of_flight;            // time of flight
  double delta_tof;                 // time of flight for single step
  double max_tof;                   // upper limit of flight time
  double step_final[3];             // final step to go
  double max_step_length;           // upper limit for step length
  double shrinkfactor;              // factor to shrink step length under max_step_length
  int    mirrors;                   // counting number of mirror
  int    max_mirrors;               // upper limit for mirrors
  int    calc_order;                // tag to switch on and off drift calculation
                                    // 0=alldrift 1=exb only 2=RxB and gradBxB 3=nothing
};


struct s_trap
{
  double trap_theta;       //theta for trapping
  double trap_energy;      //energy for trapping
  double trap_position[3];      //[x,y,z]
};
```

B.2 Simulationparameter: .run Datei

```
display_calc-order_retard-onoff_dipol_max-mirrors__max-tof-in-sec__e-para-min__max-step-length
0       0          1            0.    100          0.1             0.001       0.01
x______y______z______ekin_____theta___phi____mass___charge
0.     -5.     5.    100.     35.     0.     1.     1.
```

B.3 Magnetfeldparameter: .par Datei

```
5
-201.2  5.8  0.8  36. 20.0  2.0 540000.0  0.0 0.0
 -46.0  85.  0.3  19. 15.0  2.0    112.5  0.0 0.0
   0.0  85.  0.3  20. 15.0  2.0    112.5  0.0 0.0
  46.0  85.  0.3  19. 15.0  2.0    112.5  0.0 0.0
 201.2  5.8  0.8  36. 20.0  2.0 540000.0  0.0 0.0
-200.0  0.  2.
0.  47.  2.
0.  47.  2.
 0.0
```