Fachhochschule Aachen
Campus Jülich

Fachbereich: Medizintechnik und Technomathematik
Studiengang: Scientific Programming

Algorithmische Bestimmung von Teilchenflugbahnen durch inhomogene Magnetfelder

Eine Seminararbeit von Christian Peters

Jülich, den 5. Dezember 2017

Eigenständigkeitserklärung

Hiermit versichere ich, dass ich diese Seminararbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe, alle Ausführungen, die anderen Schriften wörtlich oder sinngemäß entnommen wurden, kenntlich gemacht sind und die Arbeit in gleicher oder ähnlicher Fassung noch nicht Bestandteil einer Studien- oder Prüfungsleistung war. Ich verpflichte mich, ein Exemplar der Seminararbeit fünf Jahre aufzubewahren und auf Verlangen dem Prüfungsamt des Fachbereiches Medizintechnik und Technomathematik auszuhändigen.

Christian Peters
Jülich, 5. Dezember 2017

Diese Arbeit wurde betreut von:
1. Prüfer: Prof. Dr. Andreas Terstegge
2. Prüfer: Günter Sterzenbach

Inhaltsverzeichnis

1 Einleitung

Dies ist die Einleitung zu meiner Seminararbeit.

2 Physikalische Grundlagen

Um ein Fundament zu schaffen, auf welchem diese Arbeit im weiteren Velauf aufbauen kann, werden an dieser Stelle zunächst die physikalischen Grundlagen dargelegt, auf denen das später beschriebene Verfahren basiert. Es werden vornehmlich die Wechselwirkungen von bewegten Teilchen in homogenen Magnetfeldern beschrieben, ein Überblick dieser Zusammenhänge ist für das weitere Verständnis dieser Arbeit unbedingt notwendig. Die hier dargelegten Ausführungen berufen sich im Wesentlichen auf [Vog99] [1].

2.1 Die Lorentz-Kraft und ihre Eigenschaften

Bewegt sich ein geladenes Teilchen mit der Geschwindigkeit v durch ein homogenes magnetisches Feld der Feldstärke B, so erfährt es die Lorentz-Kraft F_L. Diese Kraft wirkt nur, wenn sich das geladene Teilchen *bewegt*, weiterhin ist der Betrag dieser Kraft sowohl proportional zur Ladung Q des Teilchens, als auch zu seiner Geschwindigkeit. Die Lorentz-Kraft wirkt stets sowohl *senkrecht* zur Bewegungsrichtung des Teilchens, als auch *senkrecht* zur Richtung des Magnetfeldes. Das Teilchen wird in seiner Flugbahn also *seitlich* abgelenkt. Der Betrag der Kraft hängt außerdem von der Richtung der Teilchenbewegung ab: Verläuft diese parallel zur Richtung des magnetischen Feldes, so gilt $F_L = 0$. Für beliebige Winkel α verändert sich der Betrag der Lorentz-Kraft wie $v \cdot \sin\alpha$. Fasst man alle diese Eigenschaften zusammen, so erhält man den folgenden Ausdruck:

$$F_L = Q \cdot v \cdot B \cdot \sin\alpha \tag{2.1}$$

Die konkrete Wirkungsrichtung der Lorentz-Kraft lässt sich leicht anhand der *Linke-Hand-Regel* ermitteln: Zeigt der Daumen der linken Hand in Richtung der Ursache für die Lorentz-Kraft, also der Bewegung eines geladenen Teilchens, und zeigt ferner der Zeigefinger der linken Hand in Richtung des magnetischen Feldes, so ergibt sich die Richtung der Lorentz-Kraft aus der Richtung des Mittelfingers der linken Hand. Fasst man die bisherigen Größen als die Vektoren $\vec{v}$, $\vec{B}$ und $\vec{F_L}$ auf, so bilden diese ein *Linkssystem*.

[1] Vgl. S.354-355 zur Lorentz-Kraft und S.451-453 zu Elektronen in homogenen Magnetfeldern.

Hierbei einigt man sich auf die sogenannte *physikalische Stromrichtung*: Negative Ladungen (beispielsweise freie Elektronen) bewegen sich per Definition vom negativen hin zum positiven Pol einer Spannungsquelle. Im Falle der *Linke-Hand-Regel* zeigt der Daumen also immer hin zur *physikalischen Stromrichtung*. Würde man unter sonst gleichen Bedingungen die Richtung des Stromes andersherum definieren (*technische Stromrichtung*), so käme anstelle der linken Hand nun die rechte Hand zur Anwendung. Alle Größen blieben aber unverändert, somit bleibt es jedem selbst überlassen, welche der Definitionen er bevorzugt.

2.2 Bewegte Ladungsträger im homogenen Magnetfeld

Bewegt sich nun ein Ladungsträger durch ein homogenes Magnetfeld, so stellt sich die Frage, durch welche Eigenschaften sich die Flugbahn beschreiben lässt, auf welcher sich dieser aufgrund der Lorentz-Kraft bewegt. Beachtet man die Tatsache, dass die Lorentz-Kraft stets senkrecht zur Bewegungsrichtung wirkt, so wird man unweigerlich zu der Schlussfolgerung gelangen, dass es sich hierbei um eine gleichförmige Kreisbewegung handeln muss. Um dies nachvollziehen zu können, werden die allgemeinen Eigenschaften einer gleichförmigen Kreisbewegung und die Eigenschaften der Flugbahn des Ladungsträgers im Speziellen im Folgenden näher beschrieben.

2.2.1 Eigenschaften der gleichförmigen Kreisbewegung

Bewegt sich ein Körper der Masse m mit konstanter Geschwindigkeit v auf einer kreisförmigen Bahn mit dem Radius r um den Kreismittelpunkt M, so spricht man von einer gleichförmigen Kreisbewegung. Der Begriff „gleichförmig“ rührt daher, dass sich der Betrag der *Bahngeschwindigkeit* v des Körpers nicht ändert, wohl aber die Richtung, welche stets tangential zur Kreisbahn verläuft. Während die *Bahngeschwindigkeit* v angibt, welche Bogenlänge des Kreises in einer bestimmten Zeit t durchlaufen wird, so beschreibt die *Winkelgeschwindigkeit* ω, welcher Winkel (man verwendet hier das *Bogenmaß*) in t zurückgelegt wird.

Damit sich ein Körper überhaupt auf einer Kreisbahn bewegen kann, muss auf ihn eine Kraft F wirken, die ihn auf dieser Bahn hält. Diese Kraft, die stets radial zum Mittelpunkte des Kreises hin gerichtet ist, nennt man *Zentripetalkraft*. Es gilt der folgende Zusammenhang für die Größe der Zentripetalkraft:

$$F = \frac{mv^2}{r} = m\omega^2 r \tag{2.2}$$

2.2.2 Beschreibung der Flugbahn

Betrachtet man vor diesem Hintergrund erneut die Lorentz-Kraft F_L, so wird deutlich, dass diese alle Eigenschaften der in Abschnitt 2.2.1 auf der vorherigen Seite beschriebenen Zentripetalkraft erfüllt. Dies liegt daran, dass die Lorentz-Kraft zum einen den Betrag der Teilchengeschwindigkeit nicht ändert und zum anderen immer senkrecht auf der Bewegungsrichtung des Teilchens steht.

Da sich der Ladungsträger jedoch im Allgemeinen nicht vollständig senkrecht zum Magnetfeld bewegt, gibt es auch eine Komponente der Geschwindigkeit des Teilchens (im Folgenden $v_{\parallel}$), welche von der Lorentz-Kraft unbeeinflusst bleibt, da sie parallel zum Magnetfeld verläuft. Um diesem Sachverhalt gerecht zu werden, betrachtet man diese unterschiedlichen Komponenten getrennt voneinander. Dies hat zur Folge, dass der Anteil der Bewegungsrichtung, welcher senkrecht zum magnetischen Feld verläuft ($v_{\perp}$), eine Kreisbewegung beschreibt, der Anteil $v_{\parallel}$ wird jedoch nicht von der Lorentz-Kraft beeinflusst. Die Summe dieser beiden Bewegungen ergibt eine Schraubenlinie, welche durch einen Zylinder mit dem Radius r der Kreisbahn begrenzt ist. Die orthogonale Projektion dieser Schraubenbewegung vollzieht genau die Kreisbewegung, welche von der Lorentz-Kraft verursacht wird. Um die konkreten Eigenschaften der Flugbahn zu quantifizieren, reicht es also aus, beide Teile der Bewegung für sich genommen zu beschreiben.

Betrachtet man die zum Magnetfeld senkrechte Komponente $v_{\perp}$ der Teilchenbewegung, so lässt sich die resultierende Kreisbewegung eindeutig durch die Angabe des Radius r, sowie durch die Winkelgeschwindigkeit ω beschreiben. Den Radius r erhält man aus dem Zusammenhang, dass die Lorentz-Kraft die Rolle der Zentripetalkraft einnimmt. Setzt man Gleichung (2.1) auf Seite 5 unter der Bedingung $\alpha = \frac{\pi}{2}$, die sich aus der Konstruktion ergibt ($v_{\perp}$ steht *senkrecht* zum Magnetfeld), mit Gleichung (2.2) auf der vorherigen Seite, der Gleichung der Zentripetalkraft, gleich, so erhält man:

$$\frac{m \cdot v_{\perp}^2}{r} = Q \cdot v_{\perp} \cdot B$$

$$r = \frac{m \cdot v_{\perp}}{Q \cdot B} \tag{2.3}$$

Ein ähnlicher Ansatz wird verwendet, um die Winkelgeschwindigkeit ω der Kreisbewegung zu berechnen. Der einzige Unterschied ist, dass die Gleichung der Zentripetalkraft verwendet wird, welche die Winkelgeschwindigkeit enthält. Gleichsetzen ergibt den folgenden Ausdruck:

$$m \cdot \omega^2 \cdot r = Q \cdot v_{\perp} \cdot B$$

$$\omega^2 = \frac{Q \cdot v_\perp \cdot B}{m \cdot r}$$

Setzen wir nun für r das Ergebnis aus Gleichung (2.3) auf der vorherigen Seite ein, so ergibt sich folgender Zusammenhang für die Winkelgeschwindigkeit der Bewegung:

$$\omega^2 = \frac{Q^2 \cdot v_\perp \cdot B^2}{m^2 \cdot v_\perp} = \frac{Q^2 \cdot B^2}{m^2}$$

$$\omega = \frac{Q \cdot B}{m} \tag{2.4}$$

Um den Bewegungsanteil $v_\parallel$ zu beschreiben, welcher entlang des Magnetfeldes verläuft, genügen die Gesetze einer gleichförmigen Bewegung konstanter Geschwindigkeit. Ist man daran interessiert, welche Strecke S das Teilchen innerhalb einer Zeit t zurücklegt, so gilt der folgende allgemein bekannte Zusammenhang:

$$S = v_\parallel \cdot t \tag{2.5}$$

Addiert man diese beiden unabhängigen Teile der Bewegung, so erhält man insgesamt die oben beschriebenen Schraubenlinien, welche das Teilchen auf seinem Weg durch das homogene Magnetfeld verfolgt.

3 Aufbau des Verfahrens

Um nun aufbauend auf diesen Erkenntnissen die allgemeinen Flugbahnen beliebiger Ladungsträger durch beliebige, unter Umständen auch *inhomogene*, Magnetfelder simulieren zu können, sind mehrere Zwischenschritte notwendig. Unter Verwendung der Ergebnisse aus Abschnitt 2.2.2 auf Seite 7 wird zunächst ein Verfahren hergeleitet, welches es ermöglicht, die Teilchenflugbahnen in *homogenen* Magnetfeldern zu simulieren. Dieses Verfahren kann anschließend auf beliebige Magnetfelder ausgeweitet werden, indem man diese abschnittsweise durch homogene Teilfelder approximiert, auf welchen dann das hergeleitete Verfahren zur Anwendung kommt.

3.1 Simulation der Flugbahn im homogenen Magnetfeld

Bewegen sich Teilchen im Raum, so kann dies in allen möglichen Orientierungen erfolgen. Um diesem Sachverhalt Rechnung zu tragen, werden zunächst sämtliche Schritte im lokalen Bezugssystem des Ladungsträgers durchgeführt. Unabhängig von der speziellen Orientierung des Teilchens gewährleistet dies einheitliche Rechenvorschriften, welche im Anschluss durch einfache Basistransformationen in das globale Koordinatensystem überführt werden können. Die größen v und B, welche in Abschnitt 2.2.2 auf Seite 7 noch als Skalare betrachtet wurden, müssen nun *vektoriell* als $\vec{v}$ und $\vec{B}$ aufgefasst werden. Es wird sich aber zeigen, dass dies durch geeignete Wahl der lokalen Basisvektoren nur eine untergeordnete Rolle spielt.

3.1.1 Die Flugbahn im lokalen Bezugssystem des Ladungsträgers

Das lokale Koordinatensystem wird nun so gewählt, dass sich die Position des Teilchens genau im Ursprung befindet. Weiterhin zeigen die magnetischen Feldlinien genau in Richtung der *z-Achse*, die orthogonale Projektion der Teilchenbewegung auf die xy-Ebene zeigt genau entlang der *y-Achse*, wie in Abbildung 3.1.1 auf der nächsten Seite dargelegt. Auf diese Weise lassen sich die beiden Anteile der Bewegung, welche in Abschnitt 2.2.2 auf Seite 7 beschrieben wurden, optimal trennen: Der gleichförmige Anteil der Bewegung, welcher auf $v_{\parallel}$ zurückzuführen ist, verläuft genau entlang der *z-Achse*, während $v_{\perp}$ eine

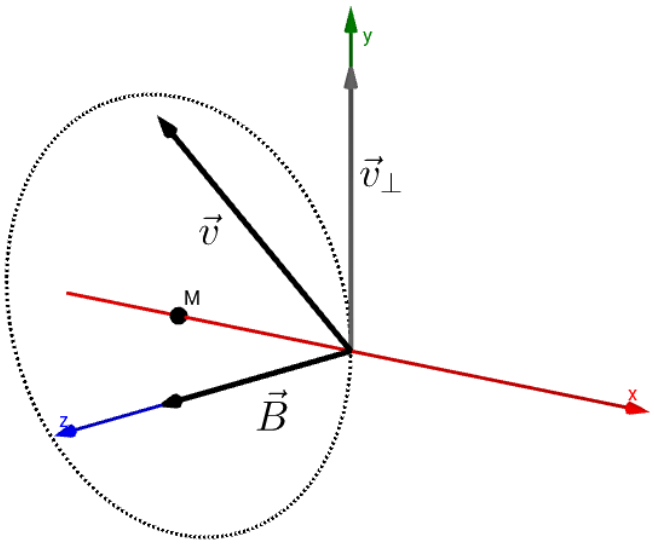

Abbildung 3.1: Das lokale Koordinatensystem des im Ursprung befindlichen Ladungsträgers mit Andeutung der Kreisbahn in der xy-Ebene.

Kreisbewegung in der *xy-Ebene* verursacht. $v_\parallel$ beeinflusst also nur die *z-Koordinate* des Ladungsträgers, während $v_\perp$ ausschließlich die x-, sowie die *y-Koordinate* beeinflusst.

Um nun den Mittelpunkt M des Kreises zu ermitteln, welchem die Bewegung in der *xy-Ebene* folgt, kann zunächst der Radius r dieses Kreises bestimmt werden. Gemäß der Konstruktion und unter Berücksichtigung der *Linke-Hand-Regel* liegt M dann genau auf der *x-Achse* und hat die xy-Koordinaten $(-r|0)$. Der Radius lässt sich nun, wie in Abschnitt 2.2.2 auf Seite 7 mit Gleichung (2.3) auf Seite 7 beschrieben, ohne weiteres berechnen. Man beachte, dass aufgrund der Konstruktion nur die *z-Koordinate* des Magnetfeldes $\vec{B}$, sowie die *y-Koordinate* der Teilchengeschwindigkeit $\vec{v}$ relevant ist. Weiterhin ist ebenfalls konstruktionsbedingt $\alpha = \frac{\pi}{2}$, weshalb der Term $\sin\alpha$ entfällt. Benutzen wir also $v_\perp = \vec{v}_y$ und $B = \vec{B}_z$, so erhalten wir folgenden Zusammenhang für den Radius r im lokalen Bezugssystem:

$$r = \frac{m \cdot \vec{v}_y}{Q \cdot \vec{B}_z} \tag{3.1}$$

Damit der exakte Punkt berechnet werden kann, an dem sich der Ladungsträger nach einer gewissen Zeitspanne t befindet, wird weiterhin die Winkelgeschwindigkeit ω der Kreisbewegung benötigt. Wenden wir die Überlegung $B = \vec{B}_z$ auf Gleichung (2.4) auf Seite 8 an, so ergibt sich für die Winkelgeschwindigkeit:

$$\omega = \frac{Q \cdot \vec{B}_z}{m} \tag{3.2}$$

Aus der Winkelgeschwindigkeit ω, dem Radius r und der gegebenen Simulationszeit t lässt sich mithilfe der Bewegungsgleichungen der gleichförmigen Kreisbewegung die exakte Position des Teilchens in der xy-Ebene ermitteln. Hierzu wird der folgende allgemeine Zusammenhang angewandt, der für gleichförmige Kreisbewegungen in der Ebene gültig ist:

$$\begin{pmatrix} x(t) \\ y(t) \end{pmatrix} = \begin{pmatrix} r \cdot \sin(\omega \cdot t) + M_x \\ r \cdot \cos(\omega \cdot t) + M_y \end{pmatrix}$$

Die Koordinaten von M_x und M_y erhält man direkt aus den obigen Überlegungen.

Die Veränderung der z-Koordinate, also der Anteil der Bewegung, welcher durch $\vec{v_{\parallel}}$ verursacht wird, lässt sich direkt mithilfe Gleichung (2.5) auf Seite 8 bestimmen. Auch hier lässt sich aufgrund der Konstruktion die Vereinfachung $v_{\parallel} = \vec{v}_z$ durchführen, da $v_{\parallel}$ lediglich durch die Projektion von $\vec{v}$ auf die z-Achse gegeben ist. Insgesamt erhalten wir folgenden Zusammenhang über die neue Position des Ladungsträgers nach der Zeit t, bezogen auf sein lokales Koordinatensystem:

$$\begin{pmatrix} x(t) \\ y(t) \\ z(t) \end{pmatrix} = \begin{pmatrix} r \cdot \sin(\omega \cdot t) - r \\ r \cdot \cos(\omega \cdot t) \\ \vec{v}_z \cdot t \end{pmatrix} \tag{3.3}$$

Weiterhin geht aus diesem Zusammenhang der Drehwinkel φ unmittelbar hervor, der die Drehung der Richtung von $v_{\perp}$ beschreibt. Die Drehung von $v_{\perp}$ um diesen Winkel muss erfolgen, um im nächsten Schritt auf dieser neuen Richtung aufbauen zu können. Der zugehörige Winkel kann wie folgt berechnet werden:

$$\varphi = \omega \cdot t \tag{3.4}$$

3.1.2 Übertragung auf das globale Koordinatensystem

Um die neue Position des Ladungsträgers bezogen auf das globale Koordinatensystem zu erhalten, ist ein Wechsel der Basis erforderlich. Zunächst müssen die Größen $\vec{v}$ und $\vec{B}$ in das lokale Bezugssystem des Ladungsträgers überführt werden, damit in dieser neuen Basis die Zusammenhänge aus Abschnitt 3.1.1 auf Seite 9 angewandt werden können. Anschließend werden die Ergebnisse, also die berechnete neue Position des Ladungsträgers sowie dessen Geschwindigkeit, wieder zurück in die ursprüngliche Basis transformiert.

3.1.2.1 Eulersche Winkel

Betrachtet man das vorliegende Modell genauer, so fällt auf, dass sich alle notwendigen Transformationen auf die Hintereinanderausführung von Drehungen zurückführen lassen. Da es sich bei Drehungen um sogenannte orthogonale Transformationen handelt, bleiben Abstände und Normen nach der Transformation erhalten. Diese Eigenschaft ist besonders wichtig, da sich sonst die Erkenntnisse aus Abschnitt 3.1.1 auf Seite 9 nicht ohne weiteres auf die transformierten Größen anwenden ließen.

Die Hintereinanderausführung von Drehungen wird durch sogenannte *Eulersche Winkel* beschrieben. Hierbei handelt es sich um die Angabe von Winkeln, die jeweils die Rotation um eine Achse beschreiben. Die Besonderheit hierbei ist, dass nur die erste Rotation um eine *raumfeste* Achse erfolgt. Die weiteren Drehungen beziehen sich stets auf die mitgedrehten lokalen Achsen des zu drehenden Körpers.

Seien $R_x(\alpha)$, $R_y(\beta)$, $R_z(\gamma)$ die Drehmatrizen zur Drehung um die *globalen* Achsen x, y und z. Nach [Fis12][1] ergibt sich die Gesamtdrehmatrix einer eulerschen Drehung durch das Produkt der einzelnen Drehmatrizen in der gewünschten Reihenfolge. Möchte man beispielweise zunächst um die *globale* x-Achse, dann um die mitgedrehte *lokale* y-Achse und anschließend um die resultierende *lokale* z-Achse drehen, ergibt sich die Gesamtdrehmatrix R zu $R = R_x(\alpha) \cdot R_y(\beta) \cdot R_z(\gamma)$, wenn jeweils um die zugehörigen Winkel α, β und γ gedreht wird, die man in diesem Kontext als *Eulersche Winkel* bezeichnet.

Möchte man die Drehung rückgängig machen, so stellt sich die Frage nach der *inversen* zu R, also R^{-1}. Da für die Matrizen der Einzeldrehungen jeweils der Zusammenhang $R_g(\alpha)^{-1} = R_g(-\alpha)$ gilt, wobei g eine beliebige Achse beschreibt, gilt für die Inverse der Gesamtdrehmatrix R:

$$\begin{aligned} R^{-1} = (R_x(\alpha) \cdot R_y(\beta) \cdot R_z(\gamma))^{-1} &= R_z(\gamma)^{-1} \cdot R_y(\beta)^{-1} \cdot R_x(\alpha)^{-1} \\ &= R_z(-\gamma) \cdot R_y(-\beta) \cdot R_x(-\alpha) \end{aligned} \tag{3.5}$$

3.1.2.2 Bestimmung der Drehwinkel

Wie in Abschnitt 3.1.1 auf Seite 9 beschrieben, zeigt die z-Achse im lokalen Bezugssystem des Ladungsträgers genau in die Richtung des magnetischen Feldes $\vec{B}$. Um $\vec{B}$ nun durch Drehungen so zu transformieren, dass die Richtung der z-Achse entspricht, können Eulersche Winkel eingesetzt werden. Da die Drehung von $\vec{B}$ auf die z-Achse genau der entgegengesetzten, also der *inversen* Drehung der z-Achse auf $\vec{B}$ entspricht, die Winkel zu dieser inversen Drehung allerdings leichter zu bestimmen sind, wird zunächst die z-Achse

[1] Vgl. hierzu Abschnitt 5.3.6 für einen ausführlichen Beweis.

auf $\vec{B}$ gedreht. Diese Transformation kann dann unter Verwendung von Gleichung (3.5) auf der vorherigen Seite im Umkehrschluss auf $\vec{B}$ angewendet werden, um die eigentlich relevante Drehmatrix zu erhalten.

Um die z-Achse, im Folgenden mit der Bezeichnung des Einheitsvektors $\vec{z}$ beschrieben, auf $\vec{B}$ zu drehen, sind zwei Drehungen erforderlich: Zunächst muss eine Drehung um die y-Achse um den Winkel α, anschließend eine Drehung um die lokal mitgedrehte x-Achse um den Winkel β erfolgen.

Der Winkel α lässt sich bestimmen, indem $\vec{B}$ zunächst orthogonal auf die xz-Ebene projiziert wird. α entspricht dann genau dem Winkel zwischen dem aus der Projektion resultierenden Vektor $\vec{p}_B$ und $\vec{z}$. Um $\vec{B}$ auf die xz-Ebene zu projizieren, kann einfach $\vec{B}_y = 0$ gesetzt werden. Zur Berechnung des Winkels zwischen $\vec{p}_B$ und $\vec{z}$ wird folgender Zusammenhang verwendet, der für den Winkel zwischen zwei beliebigen Vektoren $\vec{a}$ und $\vec{b}$ gilt:

$$\sphericalangle\left(\vec{a}, \vec{b}\right) = \arccos\left(\frac{\vec{a} \cdot \vec{b}}{\|a\|_2 \cdot \|b\|_2}\right) \tag{3.6}$$

wobei $\|\cdot\|_2$ der euklidischen Norm entspricht. Da $\vec{z}$ Einheitsvektor ist, gilt $\|\vec{z}\|_2 = 1$, weiterhin ist $\vec{z}_x = \vec{z}_y = 0$, weshalb $\vec{z} \cdot \vec{p}_B = \vec{p}_{B_z} = \vec{B}_z$. Insgesamt ergibt sich also für den Winkel α:

$$\alpha = \arccos\left(\frac{\vec{B}_z}{\|\vec{p}_B\|_2}\right) \tag{3.7}$$

Da man durch Anwendung dieser Formel ausschließlich Winkel im Bereich von $0 \leq \alpha \leq \pi$ erhält, ist noch eine kleine Anpassung nötig, damit auch im Falle eines überstumpfen Winkels korrekt gedreht werden kann. Dieser Fall lässt sich erkennen, indem die x-Koordinate von $\vec{p}_B$ überprüft wird. Ist diese positiv, sind keine weiteren Anpassungen notwendig, ist sie negativ, kann einfach in die andere Richtung, also um den Winkel $-\alpha$ gedreht werden, es ergibt sich also:

$$\alpha = -\arccos\left(\frac{\vec{B}_z}{\|\vec{p}_B\|_2}\right) \tag{3.8}$$

Für den Fall, dass $\vec{p}_B$ gerade dem Nullvektor entspricht, also dass $\vec{B}$ genau in Richtung der y-Achse zeigt, entfällt die Notwendigkeit der Drehung. Hier kann formal $\alpha = 0$ gesetzt werden.

Um nun den Drehwinkel β zu bestimmen, der die Drehung um die *lokale* mitgedrehte x-Achse beschreibt, wird zunächst der Winkel δ berechnet, welcher den Winkel zwischen der y-Achse, im Folgenden mit $\vec{y}$ bezeichnet, und $\vec{B}$ angibt. Der Winkel β ergibt sich dann

anschaulich aus dem Zusammenhang $\beta = -\left(\frac{\pi}{2} - \delta\right) = \delta - \frac{\pi}{2}$. Das $-$ rührt daher, dass in diesem Fall entgegen der mathematisch positiven Drehrichtung gedreht wird. Auch hier muss der Fall $\vec{B}_y < 0$ um der Drehrichtung willen gesondert betrachtet werden: Da hier genau entgegengesetzt gedreht werden muss, entfällt das $-$ und es gilt $\beta = \frac{\pi}{2} - \delta$. Zeigt $\vec{z}$ analog zum vorherigen Fall bereits in Richtung $\vec{B}$, so entfällt auch hier die Notwendigkeit der Drehung, und es kann $\beta = 0$ gesetzt werden. Analog zu den obigen Überlegungen berechnet sich δ für $\vec{B}_y \geq 0$ zu

$$\delta = \arccos\left(\frac{\vec{B}_y}{\left\|\vec{B}\right\|_2}\right) \tag{3.9}$$

und für $\vec{B}_y < 0$ zu

$$\delta = \arccos\left(\frac{-\vec{B}_y}{\left\|\vec{B}\right\|_2}\right) \tag{3.10}$$

Da die z-Achse nun durch diese beiden Drehungen erfolgreich auf $\vec{B}$ gedreht wurde, fehlt im nächsten Schritt nur noch die Drehung von $\vec{v}$, so dass die orthogonale Projektion von $\vec{v}$ auf die xy-Ebene genau auf der y-Achse liegt. Der Winkel γ, welcher zu diesem Zweck benötigt wird, kann leicht bestimmt werden, indem der Winkel zwischen der Projektion $\vec{p}_v$ und $\vec{y}$ bestimmt wird. Analog zu den obigen Ergebnissen berechnet sich dieser für $\vec{p}_{v_x} \geq 0$ zu

$$\gamma = \arccos\left(\frac{\vec{v}_y}{\|\vec{p}_v\|_2}\right) \tag{3.11}$$

aufgrund der umgekehrten Drehrichtung im Falle von $\vec{p}_{v_x} < 0$ ergibt sich hier

$$\gamma = -\arccos\left(\frac{\vec{v}_y}{\|\vec{p}_v\|_2}\right) \tag{3.12}$$

3.1.2.3 Zusammenfassung der Transformationsschritte

Unter Verwendung der in Abschnitt 3.1.2.2 auf Seite 12 bestimmten Drehwinkel, kann nun die vollständige Transformationsvorschrift angegeben werden, die den Übergang vom *globalen* ins *lokale* Bezugssystem des Ladungsträgers beschreibt. Im ersten Schritt muss hierzu die Position des Teilchens in den Ursprung verschoben werden. Für die relevanten Vektoren $\vec{B}$ und $\vec{v}$ hat dies keine Auswirkungen, sie bleiben davon unbeeinflusst. Im nächsten Schritt müssen die beschriebenen Drehungen durchgeführt werden. Da sich α und β darauf beziehen, wie die z-Achse auf $\vec{B}$ gedreht wird, wir aber $\vec{B}$ auf die z-Achse drehen wollen, erhalten wir aus obigen Überlegungen die *inverse* der gewünschten

Transformation T_B, die sich aus den *Eulerschen Winkeln* α und β ergibt:

$$T_B^{-1} = R_y(\alpha) \cdot R_x(\beta) \tag{3.13}$$

Nach Gleichung (3.5) auf Seite 12 lässt sich hieraus die Drehung von $\vec{B}$ auf die z-Achse wie folgt berechnen:

$$T_B = R_x(-\beta) \cdot R_y(-\alpha) \tag{3.14}$$

Die anschließende Transformation von $\vec{v}$ ergibt sich dann zu

$$T_v = R_z(\gamma) \tag{3.15}$$

Aus diesen beiden Matrizen lässt sich dann die Gesamttransformation wie folgt berechnen:

$$T = T_v \cdot T_B = R_z(\gamma) \cdot R_x(-\beta) \cdot R_y(-\alpha) \tag{3.16}$$

Durch diese Matrix T lässt sich also jeder beliebige Vektor in das lokale Bezugssystem des Ladungsträgers überführen. Nachdem die in Abschnitt 3.1.1 auf Seite 9 beschriebenen Berechnungen in dieser Basis durchgeführt wurden, müssen die Größen dann im Anschluss wieder in die alte Basis zurücktransformiert werden. Zu diesem Zweck kann die *inverse* Matrix von T wie folgt bestimmt werden:

$$T^{-1} = T_B^{-1} \cdot T_v^{-1} = R_y(\alpha) \cdot R_x(\beta) \cdot R_z(-\gamma) \tag{3.17}$$

Die Verschiebung der Teilchenposition muss nach Anwendung der lokalen Berechnungsvorschriften ebenfalls rückgängig gemacht werden.

3.2 Ausweitung auf inhomogene Magnetfelder

Nachdem die Flugbahnen bewegter Ladungsträger in *homogenen* Magnetfeldern nun ohne weiteres bestimmt werden können, kann das Verfahren im nächsten Schritt auf *inhomogene* Magnetfelder ausgeweitet werden. Zu diesem Zweck wird das Verfahren auf mehrere Iterationen heruntergebrochen. In jedem Iterationsschritt wird das magnetische Feld in der lokalen Umgebung des Teilchens durch ein *homogenes* Magnetfeld approximiert. In dieser Approximation wird anschließend die neue Position des Ladungsträgers bestimmt, ausgehend von dieser neuen Position kann dann wieder ein lokales homogenes Magnetfeld zur Näherung verwendet werden. Im Rahmen dieser Vorgehensweise wird also eine Diskretisierung des inhomogenen Feldes vorgenommen. Das inhomogene Feld wird dabei

in lokal homogene Teilfelder zerteilt, auf denen die bekannten Berechnungsvorschriften zur Anwendung kommen.

3.2.1 Diskretisierung inhomogener Magnetfelder

Um nun inhomogene Magnetfelder diskretisieren zu können, muss im ersten Schritt die lokale Feldstärke $\vec{B}$ an der Position des Teilchens bestimmt werden. Aufbauend auf $\vec{B}$ und der Teilchengeschwindigkeit $\vec{v}$ können anschließend nach der notwendigen Transformation die Erkenntnisse aus Abschnitt 3.1.1 auf Seite 9 angewendet werden. Zur Diskretisierung stellt sich nun noch die Frage, wie fein bzw. grob das inhomogene Feld unterteilt werden soll. Diese Unterteilung lässt sich leicht über die in Abschnitt 3.1.1 auf Seite 9 beschriebene Simulationszeit t steuern. Beschränkt man die Simulation der Flugbahn unter Verwendung des lokalen homogenen Modells nur auf einen kurzen Zeitabschnitt t, so ist die Unterteilung fein. Erhöht man t, so erhält man eine gröbere Diskretisierung. Die Zeit t wird also zum Diskretisierungsparameter des Verfahrens. Je kleiner die einzelnen Zeitabschnitte sind, in die die Gesamtsimulation unterteilt wird, desto feiner wird diskretisiert und desto geringer sind die Abweichungen von den tatsächlichen Größen.

3.2.1.1 TODO Fehler

3.3 Abbruchkriterien

Nachdem beschrieben wurde, welche einzelnen Schritte in jeder Iteration durchzuführen sind, muss nun noch entschieden werden, wann die Iteration abgebrochen werden soll. Abhängig vom tatsächlichen Einsatzgebiet des Verfahrens sind hierzu verschiedene Abbruchkriterien denkbar. Die hier aufgelisteten möglichen Bedingungen zur Beendigung der Iteration werden im Folgenden kurz erläutert.

- Schnittpunkt der Flugbahn mit einer Ebene
- Zurücklegen einer maximalen Distanz
- Wechsel der Detektorgeometrie
- Passieren eines bestimmten Ortes

Schnittpunkt der Flugbahn mit einer Ebene Oft können bestimmte räumliche Bereiche, in denen die Simulation der Teilchenflugbahn nicht weiter gewünscht ist, durch

Ebenen abgetrennt werden. Durchquert das Teilchen auf seinem Weg eine solche Ebene, die den Übergang in einen derartigen Bereich markiert, kann die Simulation somit abgebrochen werden.

Zurücklegen einer maximalen Distanz Möchte man die Ausdehnung der Flugbahn über die zurückgelegte Distanz beschränken, so kann dieses Kriterium zur Anwendung kommen. Unter Verwendung eines Bezugspunktes kann in jedem Iterationsschritt die aktuelle Distanz zu diesem Punkt berechnet werden, überschreitet diese eine gegebene Schwelle, so kann die Iteration abgebrochen werden.

Wechsel der Detektorgeometrie Liegen Informationen über den Aufbau des verwendeten Teilchendetektors vor, so kann mithilfe dieser Bedingung der Übergang in irrelevante Bereiche des Detektors erfragt werden. Gelangt das Teilchen auf seinem Weg in solche Detektorabschnitte, so kann die Simulation beendet werden.

Passieren eines bestimmten Ortes Unter Umständen ist die Flugbahn des Teilchens nur solange von Interesse, bis dieses einen bestimmten Ort passiert hat. Hier kann in jedem Iterationsschritt der aktuelle Abstand zwischen der Teilchenposition und dem gegebenen Ort berechnet werden. Erreicht diese Distanz in einem Schritt ihr Minimum und wächst im nächsten Iterationsschritt wieder an, so wird die Simulation abgebrochen.

Je nach Anwendungsfall ist es also unterschiedlich, welche der Bedingungen sinnvollerweise eingesetzt werden. Natürlich sind auch diverse Kombinationen und Verknüpfungen der verschiedenen Abbruchkriterien denkbar. Beispielsweise könnte man in einem speziellen Fall die Simulation beenden wollen, wenn das Teilchen entweder eine bestimmte Ebene schneidet, die beispielsweise eine Seitenwand des Detektors markiert, oder wenn es eine maximale Distanz zurücklegt.

Wichtig ist es, an dieser Stelle zu betonen, dass die aufgeführte Liste bei Weitem keine Anspruch auf Vollständigkeit erhebt, das Verfahren lässt sich in dieser Hinsicht also beliebig erweitern und konfigurieren.

3.4 Der resultierende Algorithmus

Um abschließend alle Schritte zusammenzufassen, welche das Verfahren beschreiben, wird der resultierende Algorithmus im Folgenden in Pseudocode dargestellt. Das erzielte Ergebnis der Simulation, also die Flugbahn des Ladungsträgers, wird hierbei durch eine

Aneinanderreihung von Teilchenzuständen, repräsentiert, die sich jeweils auf einen konkreten Zeitpunkt beziehen. Diese Information wird in der Variable *track* abgelegt.

Der Algorithmus in Pseudocode

```
1:  // Init variables
2:  track, timeStep, particleState
3:  track.add(particleState)
4:  while !stoppingCondition do
5:      // Get quantities
6:      magneticField ← getMagneticField(particleState.location)
7:      velocity ← particleState.velocity
8:      oldLocation ← particleState.location
9:
10:     // Get angles for transformation
11:     α ← getAlpha()
12:     β ← getBeta()
13:     γ ← getGamma()
14:
15:     // Do the transformation
16:     magneticField.rotateXY(−β, −α)
17:     velocity.rotateXY(−β, −α)
18:     velocity.rotateZ(γ)
19:     newLocation ← (0, 0, 0)
20:
21:     // Calculate the characteristics of the local motion
22:     r ← calculateRadius()
23:     ω ← calculateAngularVelocity()
24:     φ ← ω· timeStep
25:
26:     // Update the position in the local coordinate system
27:     newLocation.x ← r · sin(φ) − r
28:     newLocation.y ← r · cos(φ)
29:     newLocation.z ← velocity.z · timeStep
30:
31:     // Rotate the velocity as it follows the circular motion
32:     velocity.rotateZ(φ)
```

```
33:
34:     // Transform back into the original system
35:     velocity.rotateZ(−γ)
36:     velocity.rotateYX(α, β)
37:     particleState.velocity ← velocity
38:     particleState.location ← oldLocation + newLocation
39:
40:     // Set a timestamp for each particle state in the track
41:     particleState.time ← particleState.time + timeStep
42:
43:     // Add the new state to the track
44:     track.add(particleState)
45: end while
46:
47: return track
```

4 Implementierung

Im Folgenden wird vorgestellt, wie das beschriebene Verfahren, welches wir auf Basis der vorigen Überlegungen erhalten haben, konkret in Programmcode umgesetzt wird. Aufgrund des hohen Maßes an Kontrolle über den Einsatz der Hardware, des damit einhergehenden Geschwindigkeitsvorteils, sowie der erleichterten Anbindung an bestehende Frameworks, wurde zu diesem Zweck die Programmiersprache C++ gewählt. Der funktionale Kern des Programms wird hierbei als *statische Bibliothek* realisiert, diese kann dann zum konkreten Einsatz bequem eingebunden werden.

4.1 Aufbau der Software

Das Gesamtgerüst des Programms wurde auf dem Paradigma der *objektorientierten Programmierung* aufgebaut. Dies bedeutet, dass der Algorithmus und die zugrundeliegende Funktionalität in Klassen gekapselt und dem Benutzer durch klar definierte Schnittstellen zugänglich gemacht wird.

4.1.1 Klassendesign

Abbildung 4.1.1 auf der nächsten Seite gibt einen Überblick über die Struktur der wichtigsten Klassen und Interfaces in Form eines UML-Klassendiagramms. Weniger wichtige Beziehungen und Operationen, sowie Klassen, die nur der reinen Repräsentation von Daten dienen, wurden bewusst aus Gründen der Übersichtlichkeit ausgespart. Die konkreten Aufgaben aller Klassen werden im Folgenden näher beschrieben.

Particle Enthält Informationen über den Zustand eines Teilchens. Hierzu zählen die Eigenschaften *Zeit, Energie, Masse, Ladung, Ort* und *Impuls*. Die Geschwindigkeit des Teilchens kann unter Kenntnis der Masse mithilfe des Impulses bestimmt werden.

Vector3D Wird verwendet, um Vektoren im dreidimensionalen Raum darzustellen. Implementiert außerdem diverse Vektoroperationen wie *Skalarmultiplikation, Vektoradditi-*

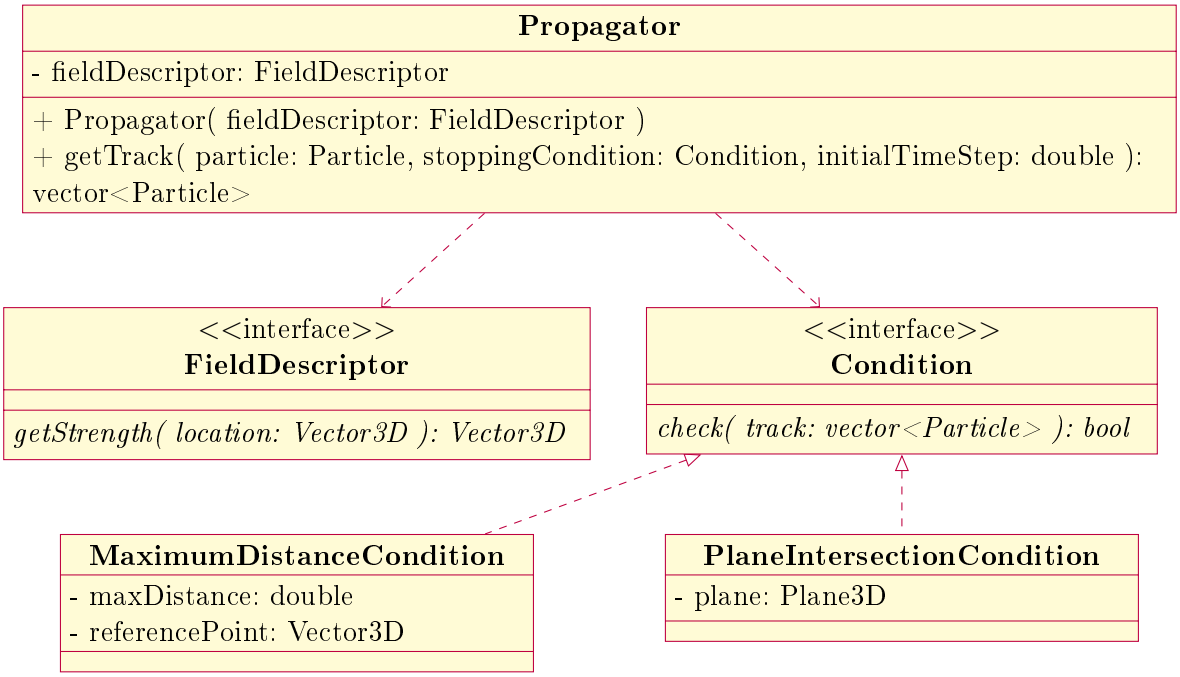

Abbildung 4.1: Die strukturelle Anordnung der wichtigsten Klassen.

on oder das *Skalarprodukt*. Die Rotation eines Vektors, welche in der Transformation zur Anwendung kommt, ist ebenfalls innerhalb dieser Klasse realisiert worden.

Plane3D Repräsentiert eine Ebene im dreidimensionalen Raum. Hierzu wird der Normalenvektor, sowie ein beliebiger Punkt in der Ebene gespeichert.

FieldDescriptor Definiert eine Schnittstelle, welche spezifiziert, wie auf die Informationen eines magnetischen Feldes zugegriffen werden kann. Durch Angabe des gewünschten Ortes, an welchem die Feldstärke bestimmt werden soll, kann diese erfragt werden.

Condition Liefert eine Schnittstelle für Abbruchbedingungen. Die implementierenden Klassen erhalten in der Methode *check* eine Referenz auf die bis dahin vorliegende Flugbahn. Soll das Verfahren abgebrochen werden, wird der Wert *true* zurückgegeben.

MaximumDistanceCondition Konkrete Implementierung der *Condition*-Schnittstelle. Gibt den Wahrheitswert *true* zurück, sobald eine maximale Distanz zu einem Referenzpunkt überbrückt wurde.

PlaneIntersectionCondition Diese Implementierung der *Condition*-Schnittstelle gibt den Wert *true* zurück, sobald die Flugbahn eine gegebene Ebene schneidet.

Propagator Kapselt den beschriebenen Algorithmus zur Bestimmung der Teilchenflugbahn. Bei der Erzeugung des Propagators muss ein *FieldDescriptor* angegeben werden, welcher das magnetische Feld beschreibt, auf welchem die Simulation durchgeführt werden soll. Die Methode *getTrack* enthält im Wesentlichen den bereits vorgestellten Pseudocode aus Abschnitt 3.4 auf Seite 18 und gibt die Flugbahn als Aneinanderreihung von Teilchenzuständen zurück.

4.1.2 Möglichkeit der Einbindung

Um die Software nun einbinden und nutzen zu können, muss zunächst die Datei *libpropagator.a* dem *Linking-Prozess* hinzugefügt werden. Die Bibliotheksdatei erhält man direkt aus der Übersetzung, welche wie in Abschnitt 4.2.1 auf der nächsten Seite beschrieben durch den Aufruf der Programme *CMake* und *Make* (Auf UNIX-Betriebssystemen) erfolgt. Weiterhin müssen die mitgelieferten *Header-Dateien* in einem Ordner liegen, der dem Compiler bekannt ist.

Bei der Nutzung des Verfahrens muss zunächst das magnetische Feld zugreifbar gemacht werden, auf welchem Simuliert werden soll. Hierzu kann bequem das Interface *FieldDescriptor* erweitert werden. Im nächsten Schritt lässt sich unter Zuhilfenahme dieser neu erstellten Klasse ein Objekt vom Typ *Propagator* instanziieren. Um die Methode *getTrack* ausführen zu können, welche die gewünschten Ergebnisse liefert, muss zunächst ein Objekt vom Typ *Particle* erzeugt werden. Dieses repräsentiert den Startzustand des Teilchens in der Simulation. Anschließend muss man eine Abbruchbedingung auswählen, ist man mit den beiden bestehenden Bedingungen noch nicht zufrieden, lässt sich das Interface *Condition* nach beliebigen Wünschen erweitern. Je nach erforderlicher Genauigkeit muss nun der Parameter *initialTimeStep* angepasst werden, schon lässt sich die Simulation starten.

4.2 Verwendete Entwicklerwerkzeuge

Im Zuge der Erstellung der Simulationssoftware wurden einige Werkzeuge eingesetzt, die den Entwicklungsprozess besonders im Hinblick auf das Übersetzen und Testen der Programmteile wesentlich unterstützt und vereinfacht haben. Diese werden im Folgenden kurz beschrieben.

4.2.1 CMake

Das plattformunabhängige Programmierwerkzeug CMake *(cross platform make)* wurde verwendet, um einen komplikationslosen Build-Prozess zu gewährleisten. CMake verfügt über die Fähigkeit, automatisch Abhängigkeiten zwischen den verschiedenen Quelldateien zu erkennen. Zusammen mit den weiteren Instruktionen, wie das gegebene Projekt zu übersetzen ist, werden diese Abhängigkeiten in ein resultierendes *Makefile* überführt, welches die Basis für das unter UNIX verfügbare Build-Management-Tool *Make* bildet, mit dessen Hilfe das Programm anschließend reibungslos übersetzt werden kann. Arbeitet man auf einer anderen Plattform als UNIX, beispielsweise auf Windows, lässt sich mit CMake auch ohne Probleme ein *Visual Studio*-Projekt erzeugen. Somit kann das Programm plattformunabhängig übersetzt und eingesetzt werden.

Ein weiterer Vorteil von CMake besteht darin, einzelne Komponenten der Software unabhängig zu verarbeiten. In diesem Fall wurde es durch CMake beispielsweise möglich, die Test- und Demonstrationsfälle als eigenständige ausführbare Programme zu übersetzen. Diese Programme betten allesamt die Kernbibliothek *libpropagator.a* ein, was die lose und bequeme Kopplung von funktionalem Programmkern und speziellem Anwendungsfall verdeutlicht.

4.2.2 Google Test

Um die Korrektheit der einzelnen Programmteile zu gewährleisten, mussten zahlreiche Testfälle kodiert werden. Hierbei wurde nach dem Unittest-Prinzip vorgegangen, welches es ermöglicht, Testfälle für unabhängige Komponenten unabhängig voneinander zu betrachten. Auf diese Weise konnte zum Beispiel das Berechnen der Drehwinkel getrennt von der tatsächlichen Transformation untersucht werden. Durch diese feingranulare Spezifikation von Testfällen können Fehlerfälle schnell identifiziert, zugeordnet und damit auch schnell behoben werden.

Eine Programmbibliothek, welche diese Art der Arbeitsweise bestens unterstützt, wird von der Firma Google unter dem Namen *Google Test* oder auch *GTest* bereitgestellt. GTest ermöglicht es, eine Reihe einzelner Unit-Tests zu kodieren und diese komfortabel hintereinander auszuführen. Dieses Verhalten erleichtert die Durchführung von sogenannten *Regressionstests* ungemein: Sobald eine Änderung am Programm vorgenommen wurde, können im Anschluss alle Testfälle bequem durch einen Aufruf von GTest ausgeführt werden. So lässt sich leicht erkennen, ob die eingeführte Änderung die Korrektheit des Programms weiterhin gewährleistet.

4.3 Visualisierung und Demonstration

5 Ausblick

Was noch alles gemacht werden kann.

Literatur

[Fis12] Gerd Fischer. *Lernbuch Lineare Algebra und Analytische Geometrie*. 2. Aufl. Springer Vieweg, 2012. ISBN: 978-3-8348-2379-3.

[Vog99] Helmut Vogel. *Gerthsen Physik*. 20. Aufl. Springer-Verlag, 1999. ISBN: 3-540-65479-8.

Abbildungsverzeichnis