Oliver Kayser-Herold, Andreas Keese, Markus Krosche,
Martin Krosche, Oliver Pajonk

Achterbahn-Editor/-Simulator

Dokumentation für das Softwareentwicklungspraktikum am
Institut für Wissenschaftliches Rechnen

Institut für Wissenschaftliches Rechnen
Carl-Friedrich-Gauss-Fakultät
Technische Universität Braunschweig
Version: 1.0.3 für das Sommersemester 2009
Revision: 2284
Erstellt am 30. März 2009

Braunschweig, 2009

Inhaltsverzeichnis

1 Einleitung

Das Institut für Wissenschaftliches Rechnen stellt zwei gekoppelte Projektthemen für das Softwareentwicklungspraktikum (SEP) 2009. Zum einen soll ein CAE-Simulator[1] entwickelt werden, der eine Achterbahn vereinfacht, aber mathematisch korrekt simuliert und dreidimensional visualisiert (Teilprojekt „Achterbahnsimulator“). Zum anderen soll ein CAE-Editor zur komfortablen, ingenieursmäßigen Konstruktion von Achterbahnkurven implementiert werden (Teilprojekt „Achterbahneditor“).

Dieses Dokument enthält die Projektbeschreibung, eine ausführliche Erklärung der mathematischen Grundlagen der Modellierung und der Simulation, eine kurze Beschreibung des Dokumentenaustauschformats und allgemeine Hinweise zum Projekt. Es bildet somit die Basis für das Gesamtprojekt; spezielle Hinweise zu den beiden Teilprojekten finden sich in gesonderten Dokumenten [4, 5]. Zudem ist es unerlässlich, dass sich beide Gruppen zusätzlich mit der jeweils anderen Aufgabenbeschreibung vertraut machen, um das Gesamtprojekt zu verstehen und auf ein gemeinsames Ziel hinarbeiten zu können.

Zusätzliche Informationen zur Organisation finden sich auf `http://www.wire.tu-bs.de` und der entsprechenden Informationsseite des Insituts für Software Systems Engineering (SSE), `http://www.sse-tubs.de/teaching/ss09/sep`.

2 Über das Dokument

Dieses Dokument enthält viele, im Normalfall blau gekennzeichnete, *Hyperlinks*. Benutzen Sie zuerst diese, um bei Fragen zusätzliche Hilfen oder einfach weiterführende und vertiefende Informationen zu erhalten.

3 Projektbeschreibung

Jeder kennt die klassische Achterbahn, die auf Jahrmärkten oder ähnlichen Veranstaltungen zu finden ist. Zumeist wird der Achterbahnwagen von einem Antrieb über eine Anfangssteigung auf eine Anfangshöhe gezogen. Von dort an läuft der Wagen unter dem Einfluss der Schwerkraft (Gravitation) selbstständig auf der vorgegebenen Bahn weiter. Physikalisch gesehen ist dies ein Wechselspiel zwischen potentieller Energie (an den höheren Stellen der Bahn) und kinetischer Energie (an den niedrigen Stellen der Bahn). Kontrolliert wird dieses Wechselspiel durch den Bahnverlauf.

Im unserem SEP sollen für diese Art von Jahrmarktattraktion zwei CAE-Werkzeuge entwickelt werden: ein Editor, mit dessen Hilfe man Streckenverläufe komfortabel designen kann, und ein Visualisierer, welcher diese Strecken dreidimensional darstellen und eine simulierte Achterbahnfahrt auf der Strecke durchführen kann.

Die Visualisierung und der Editor benötigen eine mathematische Repräsentation des Bahnverlaufs, also der **Geometrie** der Bahn. In diesem Praktikum wird der Bahnverlauf aus einzelnen Streckenelementen zusammengesetzt. Die einzelnen Streckenelemente werden dabei durch *kubische Polynome* beschrieben. Die so entstehende, stückweise aus Polynomen zusammengesetzte Raumkurve wird *Spline* genannt. Mit ihrer Hilfe lassen sich die typischen Elemente einer Achterbahn wie Loopings und Steilkurven problemlos umsetzen. Das genaue Vorgehen wird in Abschnitt 4 auf der nächsten Seite eingehend erläutert.

Die Lage und Form der einzelnen Streckenelemente wird im Editor vom Benutzer festgelegt und in einer **XML-Datei** abgelegt, welche der Simulator daraufhin einliest. Ein Vorschlag für das Format wird in Abschnitt 8 auf Seite 14 gegeben. Neben dem Streckenverlauf werden für das physikalische Modell weitere Parameter, wie die Anfangsposition und die Anfangsgeschwindigkeit

[1] CAE = *Computer Aided Engineering*

des Zuges und die Stärke der Gravitation benötigt. Diese und ggf. weitere Simulationsparameter sind aus einer weiteren XML–Datei einzulesen. Auch das Format dieser Datei wird in Abschnitt 8 auf Seite 14 kurz erläutert.

Abstrakt betrachtet handelt es sich bei der Achterbahn um ein *kontinuierliches dynamisches System*. Zu jedem Zeitpunkt ist der Zustand des Zuges durch seinen Ort im Raum – dem *Ortsvektor* – und seine Geschwindigkeit – dem *Geschwindigkeitsvektor* – bestimmt. Im Verlauf der Fahrt ändert sich dieser Zustand aufgrund der auf den Zug wirkenden Kräfte kontinuierlich. Im Rahmen des Praktikums wird nur die Gravitationskraft, die den Zug in Richtung Erde beschleunigt, berücksichtigt. Zusätzliche Kräfte, wie z.B. Luftwiderstand, Reibung auf der Schiene oder Energiezufuhr durch einen Antrieb, bleiben erstmal unberücksichtigt – diese erhöhen die Komplexität der Modellierung[2]. Mathematisch wird das dynamische Verhalten des Zuges durch eine **gewöhnliche Differentialgleichung** (DGL, im englischen ODE für „ordinary differential equation“) beschreiben, welche in Abschnitt 5 auf Seite 7 hergeleitet und eingehend erläutert wird.

Die Berechnung des tatsächlichen, durch die Geometrie und die physikalische Modellierung bedingten, zeitlichen Verhaltens der Bahn wird durch ein *numerisches* Lösungverfahren erledigt. Dieses übernimmt die sogenannte **Zeitintegration** der Differentialgleichungen und ist in Abschnitt 6 auf Seite 10 detailliert erläutert.

4 Modellierung des Streckenverlaufs

Als erste Zutat für die „Achterbahn im Computer“ benötigen wir die mathematische Beschreibung des Bahnverlaufs. Da wir CAE-Werkzeuge entwickeln wollen, soll dieses der Realität ausreichend genau entsprechen. Durch diese Vorgabe erhalten wir einige Anforderungen an die Geometrie.

Um die Kräfte, welche während einer Achterbahnfahrt auf die Passagiere wirken, in einem vertretbaren Rahmen zu halten, muss die Streckenführung einer Achterbahn gewisse Anforderungen erfüllen: da wir eine echte Achterbahnstrecke modellieren wollen muss die Stecke ohne „Löcher“ sein. Dies bedeutet, dass die Bahnkurve notwendigerweise geschlossen sein muss – insbesondere an den Stützstellen. Dies ist mathematisch gesehen eine sogenannte *Stetigkeitsanforderung*. Ebenso kann sich aus physikalischen Gründen[3] die *Geschwindigkeit* nur stetig ändern: die erste Ableitung der Streckenführung muss also notwendigerweise ebenfalls stetig sein.

Die ersten Achterbahnen erfüllten nur diese notwendigen Bedingungen. Eine Folge war, dass Passagiere nach einer Fahrt häufig Beschwerden in der Wirbelsäule hatten. Der Grund dafür war die Unstetigkeit in der zweiten Ableitung der Streckenführung, die physikalisch gesehen der *Beschleunigung* und damit den auf die Passagiere wirkenden Kräften entspricht. Durch plötzliche Änderungen dieser Kräfte kam es zu den Verletzungen. Auch die zweite Ableitung sollte also stetig sein.

Zusätzlich würden zu abrupte Richtungsänderungen, wie z.B. eine 90° Kurve mit extrem kleinem Radius, unweigerlich zu schweren Verletzungen führen, da die auf die Passagiere wirkenden Kräfte zu groß werden[4]. Wir müssen also darauf achten, dass die zweite Ableitung nicht nur stetig, sondern ihr Betrag auch *beschränkt* ist. Dies können wir allerdings nur während der Simulation verifizieren – direkt im Modell können wir es nicht erzwingen.

In diesem Praktikum wird für die Streckenführung eine Folge von kubischen Polynomen verwendet. Diese sind überall zweimal stetig differenzierbar und vermeiden so die oben beschriebenen Probleme. Sie werden daher auch im realen Achterbahnbau eingesetzt. Wir werden eine spezielle Art dieser Polynome einsetzen: *kubische Bézier-Kurven*. Benannt sind diese Kurven nach dem

[2] Es steht Ihnen natürlich frei, dies zu ändern und zusätzliche Kräfte einzuführen. Der Ansatzpunkt hierzu ist nicht allzu komplex und wird in Abschnitt 5 aufgezeigt.

[3] Das Stichwort hierzu ist „Trägheit der Masse“.

[4] Selbst mit einem speziellen Anzug halten Jetpiloten Kräfte nur bis maximal dem 9fachen der Erdbeschleunigung in vertikaler Richtung von oben nach unten aus – und dies auch nur für kurze Zeit. In Richtung von unten nach oben ist dieser Wert noch einmal deutlich geringer. In einer Achterbahn für „normale Menschen“ müssen diese Kräfte also viel kleiner sein!

französischen Ingenieur *Pierre Étienne Bézier*, welcher die Kurven im Automobilbau eingesetzt hat. Entwickelt wurden sie aber 1959 von *Paul de Casteljau*. Diese Kurven sind einfach zu implementieren und haben weitere Vorteile, wie wir sehen werden.

4.1 Grundlagen

Mathematisch gesehen ist der Streckenverlauf einer beliebigen dreidimensionalen Achterbahn als Satz von Funktionen gegeben, welche die Entwicklung der x,y und z-Koordinate der Strecke *in Abhängigkeit von einem skalaren Parameter*, hier s genannt, beschreiben (man erinnere sich an *Raumkurven* aus der Analysis):

$$\vec{\mathbf{x}}(s) = \begin{bmatrix} x(s) \\ y(s) \\ z(s) \end{bmatrix}, \quad s \in [a..b]. \tag{1}$$

Eine *geschlossene* Bahn muss zusätzlich noch die folgende Bedingung erfüllen:

$$\vec{\mathbf{x}}(a) = \vec{\mathbf{x}}(b).$$

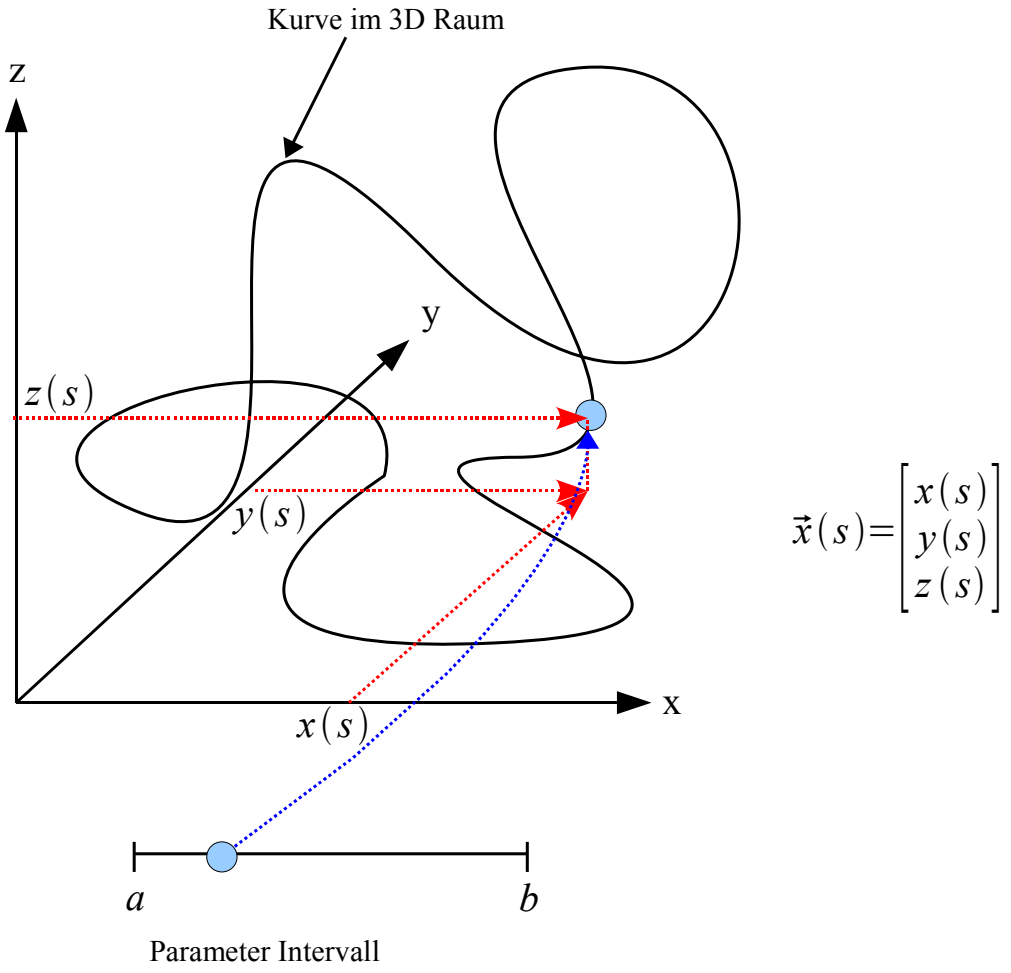

Abbildung 1: *Darstellung einer parametrisierten Kurve im Raum. Der Parameter s durchläuft das Intervall a..b und wird in den dreidimensionalen Raum abgebildet.*

Eine solche geschlossene Raumkurve ist in Abbildung 1 auf der vorherigen Seite dargestellt. Da generelle Funktionen x, y und z, welche eine Bahn mit Loopings und Kurven beschreiben, schwierig zu finden sind, bietet sich eine Approximation dieser ursprünglichen Funktionen durch einfachere Funktionen an. In diesem Praktikum wird die Funktion zwischen n *Stützstellen* im dreidimensionalen Raum $(\mathbf{x}_1, \ldots, \mathbf{x}_n)$ durch n kubische Bézier-Kurven approximiert[5], wobei eine Stützstelle gegeben ist durch

$$\mathbf{x}_i = \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix}. \tag{2}$$

Eine solche stückweise Approximation ist in Abbildung 2 dargestellt.

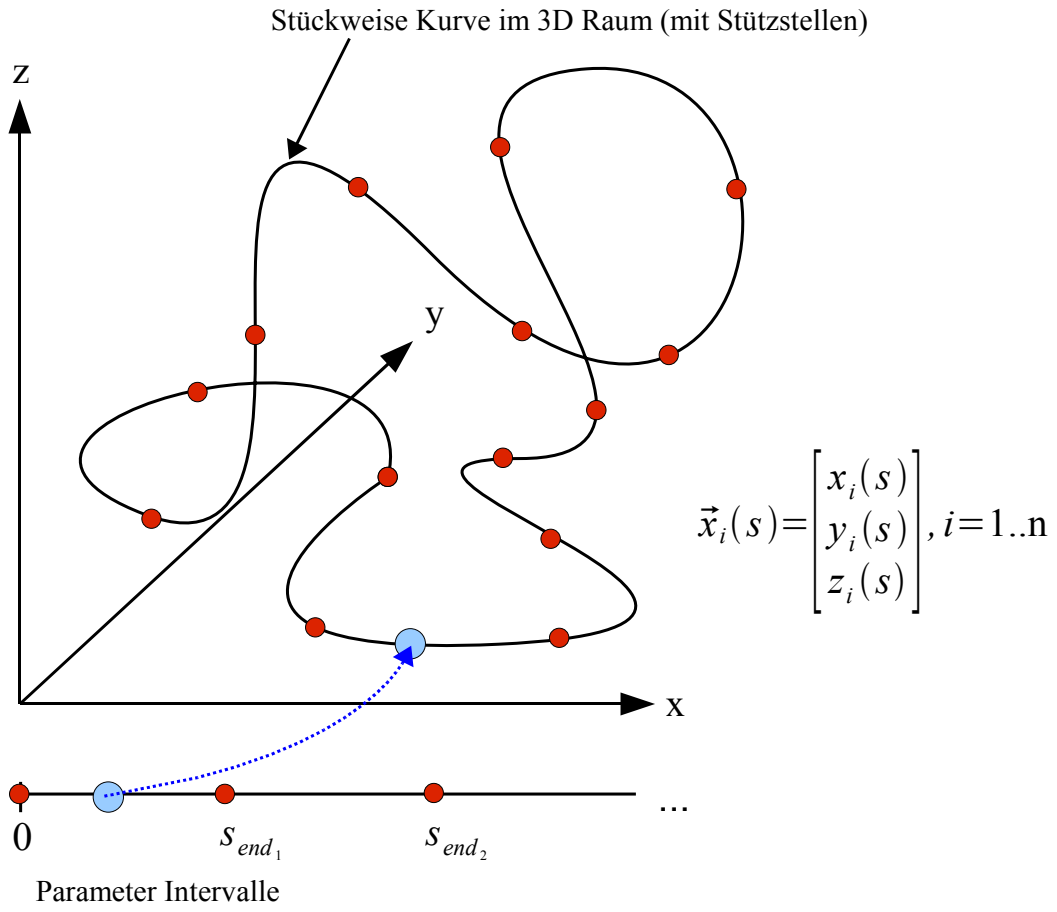

Abbildung 2: *Darstellung einer parametrisierten, stückweisen Kurve mit Stützstellen im Raum. Der Parameter s durchläuft das Intervall $0..s_{end_1}$ und wird in den dreidimensionalen Raum zwischen zwei Stützstellen abgebildet. An der Stützstelle wechselt die Funktion $\vec{\mathbf{x}}_1$ zu $\vec{\mathbf{x}}_2$ und der Parameter s durchläuft das Intervall $s_{end_1}..s_{end_2}$, usw.*

[5] Warum wir gerade kubische Funktionen benutzen ist in der Einführung von Abschnitt 4 auf Seite 2 beschrieben.

4.2 Kubische Bézier-Kurven

Eine d-dimensionale, kubische Bézier-Kurve hat die folgende Form:

$$\begin{aligned}\mathbf{B}(s) &= (1-s)^3 \begin{bmatrix} p_{0,1} \\ p_{0,2} \\ \vdots \\ p_{0,d} \end{bmatrix} + 3(1-s)^2 s \begin{bmatrix} p_{1,1} \\ p_{1,2} \\ \vdots \\ p_{1,d} \end{bmatrix} + 3(1-s)s^2 \begin{bmatrix} p_{2,1} \\ p_{2,2} \\ \vdots \\ p_{2,d} \end{bmatrix} + s^3 \begin{bmatrix} p_{3,1} \\ p_{3,2} \\ \vdots \\ p_{3,d} \end{bmatrix} \\ &= (1-s)^3 \mathbf{p}_0 + 3(1-s)^2 s \mathbf{p}_1 + 3(1-s)s^2 \mathbf{p}_2 + s^3 \mathbf{p}_3 \ , \ s \in [0,1], \end{aligned} \tag{3}$$

wobei $\mathbf{p}_0$ bis $\mathbf{p}_3$ Punkte im d-dimensionalen Raum[6] sind. Der skalare Wert s ist der Parameter. Weitere Informationen zu dieser Art von Kurven finden Sie z.B. in [6].

Bei genauerer Betrachtung stellt man fest, dass die Raumkurve für $s = 0$ genau bei $\mathbf{p}_0$ liegt und für $s = 1$ genau bei $\mathbf{p}_3$. Dazwischen haben wir eine kontinuierliche Funktion. Wenn wir uns jetzt $\mathbf{p}_0$ und $\mathbf{p}_3$ als zwei aufeinander folgende Stützstellen der Achterbahn, $\mathbf{x}_i$ und $\mathbf{x}_{i+1}$, vorstellen, haben wir eine einfach zu handhabende Beschreibung für die Achterbahn-Spline: wir benutzen für jeden der Abschnitte $\vec{\mathbf{x}}_i$, die in Abbildung 2 auf der vorherigen Seite angedeutet sind, eine solche Bézier-Kurve, im folgenden mit $\mathbf{B}_i(s)$ bezeichnet.

Für die fertige Streckenbeschreibung durch diese Folge von Bézier-Kurven fehlt jetzt also die Berechnung der „Zwischen-Stützstellen“ $\mathbf{p}_1$ und $\mathbf{p}_2$ für alle Teilabschnitte (diese werden oft „Kontrollpunkte“ genannt) – denn gegeben sind ja nur die n Stützstellen $\mathbf{x}_i$ (siehe Formel 2 auf der vorherigen Seite). Wie diese Kontrollpunkte aus den Stützstellen berechnet werden können wird im nächsten Abschnitt beschrieben.

4.3 Berechnung der Kontrollpunkte

Die Berechnung der Kontrollpunkte $\mathbf{p}_1$ und $\mathbf{p}_2$ aus den Stützstellen der Raumkurve $(\mathbf{x}_1, \ldots, \mathbf{x}_n)$ erfolgt über die Lösung eines *linearen Gleichungssystems* (LGS). Diese sind aus den Grundvorlesungen der Mathematik, insbesondere der linearen Algebra, bekannt. Eine kurze Erinnerung: ein solches System hat ganz allgemein die Form

$$\mathbf{A}\,\mathbf{x} = \mathbf{rhs},$$

wobei $\mathbf{A}$ eine Matrix ist, $\mathbf{rhs}$ ein Vektor[7] mit bekannten Werten und $\mathbf{x}$ ein Vektor mit unbekannten Werten. In unserem Fall sind die bekannten Werte die Stützstellen $\mathbf{x}_i$, während die Kontrollpunkte die unbekannten Werte sind.

Mit Hilfe der Formel für kubische Bézier-Kurven (Formel 3) und den im Text weiter oben beschriebenen Stetigkeitsanforderungen können wir unser LGS aufstellen. Zuerst definieren wir aber noch, wie oben gefordert, $\mathbf{p}_0 := \mathbf{x}_i$ und $\mathbf{p}_3 := \mathbf{x}_{i+1}$. Zusätzlich benennen wir $\mathbf{p}_1 := \mathbf{a}_i$ und $\mathbf{p}_2 := \mathbf{b}_i$, da wir ja für jeden Teilabschnitt des Splines verschiedene Werte für die Kontrollpunkte benötigen.

Für das weitere Vorgehen benötigen wir noch die ersten beiden Ableitungen von Gleichung 3:

$$\mathbf{B}_i'(s) = -3\mathbf{x}_i\,(1-s)^2 + 3\mathbf{a}_i\,(1-s)^2 - 6\mathbf{a}_i\,(1-s)\,s + 6\mathbf{b}_i\,(1-s)\,s - 3\mathbf{b}_i\,s^2 + 3\mathbf{x}_{i+1}\,s^2, \tag{4}$$

$$\mathbf{B}_i''(s) = 6\mathbf{x}_i\,(1-s) - 12\mathbf{a}_i\,(1-s) + 6\mathbf{a}_i\,s + 6\mathbf{b}_i\,(1-s) - 12\mathbf{b}_i\,s + 6\mathbf{x}_{i+1}\,s. \tag{5}$$

Damit erhält man für die Werte der Bézier-Kurven-Funktion und ihrer ersten beiden Ableitungen

[6] Dies wird bei unserer Achterbahn natürlich der 3-dimensionale Raum $\mathbb{R}^3$ sein – für die weiteren Ausführungen ist dies aber erst mal unerheblich.

[7] Der Vektor auf der rechten Seite der Gleichung (engl.: *right hand side*, daher **rhs**) enthält klassischerweise die bekannten Werte, während die unbekannten auf der linken Seite der Gleichung (engl.: *left hand side*, also **lhs**) stehen.

an den Enden des Intervalls $s \in [0..1]$:

$$\mathbf{B}_i(0) = \mathbf{x}_i \tag{6}$$
$$\mathbf{B}_i(1) = \mathbf{x}_{i+1} \tag{7}$$
$$\mathbf{B}'_i(0) = -3\mathbf{x}_i + 3\mathbf{a}_i \tag{8}$$
$$\mathbf{B}'_i(1) = 3\mathbf{x}_{i+1} - 3\mathbf{b}_i \tag{9}$$
$$\mathbf{B}''_i(0) = 6\mathbf{x}_i - 12\mathbf{a}_i + 6\mathbf{b}_i \tag{10}$$
$$\mathbf{B}''_i(1) = 6\mathbf{a}_i - 12\mathbf{b}_i + 6\mathbf{x}_{i+1}. \tag{11}$$

Die weiter oben beschriebenen Stetigkeitsforderungen lassen sich direkt in die folgenden Gleichungen übersetzen:

$$\mathbf{B}'_i(1) = \mathbf{B}'_{i+1}(0) \tag{12}$$
$$\mathbf{B}''_i(1) = \mathbf{B}''_{i+1}(0). \tag{13}$$

Da es sich um eine *geschlossene* Achterbahnkurve handelt, haben wir noch zwei weitere Gleichungen:

$$\mathbf{B}'_n(1) = \mathbf{B}'_1(0) \tag{14}$$
$$\mathbf{B}''_n(1) = \mathbf{B}''_1(0). \tag{15}$$

Wenn man jetzt die entsprechenden Gleichungen 8 bis 11 in die vier Bedingungen 12 bis 15 einsetzt (und die nötigen Indexverschiebungen macht), ergibt sich ein eindeutig lösbares System von $2n \cdot d$ linearen Gleichungen für die $2n \cdot d$ Unbekannten[8] $\mathbf{a}_i$ und $\mathbf{b}_i$. Dieses bringt man in eine Matrix-Vektor-Form und löst es mit Hilfe von Standardmethoden, wie z.B. dem *Gauß-Algorithmus*[9].

4.4 Gesamter Algorithmus

Zusammengefasst ergibt sich also folgender Algorithmus:

- Lese die x_i, y_i, z_i-Koordinaten der Stützstellen der Bahn ein.
- Bereite die Matrix A des LGS vor. In dieser werden die aus den Bedingungen 12 bis 15 resultierenden Gleichungen „codiert“.
- Bestimme aus den Koordinaten $\mathbf{x}_i$ die Kontrollpunkte $\mathbf{a}_i$ und $\mathbf{b}_i$ für die n verschiedenen Streckenabschnitte durch Lösung des LGS.

Danach hat man für jede Bahnkoordinate einen Satz von Polynomen, die den Bahnverlauf in einem bestimmten Intervall beschreiben. Diese Polynome werden sowohl zur 2D/3D-Visualisierung im Simulator bzw. Editor, als auch zur späteren physikalischen Simulation der Achterbahn selbst benötigt.

Bei [7] kann der Gauß-Algorithmus nachgeschlagen werden, falls er nicht mehr aus den grundlegenden Mathematik-Vorlesungen geläufig ist. Es wird allerdings dringlich empfohlen ein bereits fertig implementiertes Lösungsverfahren für das Lösen eines linearen Gleichungssystems zu verwenden.

[8] Diese Anzahl ergibt sich aus der Anzahl der Teilabschnitte der Spline, n, und der Dimension des Raumes, d – wobei die Dimension in unserem Fall natürlich $d = 3$ ist.

[9] Implementieren Sie diese Verfahren nicht selbst – in diesem Praktikum geht es primär um Softwaretechnik. Suchen Sie sich eine geeignete Bibliothek dafür.

5 Modellierung des dynamischen Verhaltens

Das physikalische Modell beschreibt den Achterbahnwagen als *einzelnen Massepunkt*. Normalerweise kann der Zustand eines Massepunkts im Raum durch sechs Werte vollständig beschrieben werden. Das sind die Position (in allen drei Raumachsen) sowie die Geschwindigkeit (in allen drei Raumachsen). Da es sich um ein dynamisches Verhalten handelt sind die Position und die Geschwindigkeit zeitabhängig, d.h. die Zeit (üblicherweise mit t bezeichnet) taucht als Parameter der sechs Werte auf.

Bei unserer Achterbahn wird sich durch die von der Spline vorgegebene Geometrie die Anzahl der Freiheitsgrade allerdings von sechs ganz natürlich auf zwei reduzieren: die momentane Position der Achterbahn *auf der Bahnkurve* und die momentane Geschwindigkeit der Achterbahn *in Richtung der Bahnkurve*.

5.1 Energieerhaltung

Zur Beschreibung des zeitlichen Verhaltens unserer Achterbahn muss die dazu passende *Differentialgleichung* (DGL) aufgestellt werden. Hierzu verwenden wir einen ganz einfachen, schon aus der Schulphysik bekannten *Erhaltungssatz*: den Energieerhaltungssatz der Newtonschen Mechanik:

$$E = T + V = const. \tag{16}$$

Er sagt aus, dass in diesem Modell die Gesamtenergie des Systems (E), welche die Summe aus *kinetischer Energie* (T) und *potentieller Energie* (V) ist, konstant bleibt. Dies passt zu unseren Modellanforderungen; wir wollen genau diese Energien berücksichtigen und, wie weiter oben schon erwähnt, alle weiteren Kräfte (Reibung etc.) vernachlässigen. Schreiben wir also die Formeln für potentielle und kinetische Energie hin:

$$E = \frac{1}{2} m \dot{p}^2 + m\, p\, g. \tag{17}$$

Hier bezeichnet m die Masse, $\dot{p}$ die momentane Geschwindigkeit dieser Masse, p die momentane Position der Masse und g das wirkende Kraftfeld, also in unserem Fall die Gravitationskraft. Der kleine Punkt über dem p ist die seit *Isaac Newton* übliche Notation für eine *Ableitung* nach der Zeit: die momentane Geschwindigkeit $\dot{p}$ ist also die Ableitung der momentanen Position p nach der Zeit. Nach etwas Nachdenken und ein paar Erinnerungsversuchen an den Physikunterricht wird man diese Aussage schnell verifizieren können.

Um jetzt von dieser abstrakten Gleichung 17 zu einer für uns nutzbaren Differentialgleichung zu kommen müssen wir uns überlegen, was in der Gleichung denn noch unklar ist. Dies sind die Geschwindigkeit $\dot{p}$ und die Position p, welche wir noch in unsere im Abschnitt 4 eingeführte Geometrie „übersetzen“ müssen: wir müssen einfließen lassen, was in unserer Geometrie „momentane Position“ und „momentane Geschwindigkeit“ konkret bedeuten. Dies beginnen wir mit simplem Einsetzen.

Der Verlauf der Bahnkurve im dreidimensionalen Raum wurde in Abschnitt 4.1 durch eine mit s parametrisierte, vektorwertige Funktion $\vec{\mathbf{x}}$ beschrieben[10]. Der Achterbahnzug befindet sich *zu jedem Zeitpunkt* auf eben genau dieser Bahnkurve, seine Position wird also durch einen bestimmten Wert von s für jeden Zeitpunkt t beschrieben. Mit anderen Worten: um die momentane Position des Zuges zu beschreiben wird s also *zeitabhängig*. Dies machen wir mit der üblichen Notation $s = s(t)$ kenntlich. Damit haben wir aber genau die Anforderung an die momentane Position p in unserem physikalischen Modell (Gleichung 17) erfüllt und schreiben:

$$p = \vec{\mathbf{x}}(s(t)) =: \vec{\mathbf{x}}. \tag{18}$$

Hierbei ist $\vec{\mathbf{x}}$ eine verkürzende Schreibweise für den Ortsvektor $\vec{\mathbf{x}}(s(t))$, um die folgenden Gleichungen übersichtlicher zu gestalten. Ebenfalls lassen wir die Notation der Zeitabhängigkeit von

[10] Das es sich bei dieser Funktion um eine stückweise Spline handelt ist jetzt nicht weiter wichtig.

s bei Bedarf weg. Diese beiden Vereinfachungen dürfen allerdings nicht vergessen werden, da sie sehr wichtig für die weiteren Berechnungen sind!

Um die Notation korrekt zu gestalten müssen wir g jetzt ebenfalls durch einen Vektor $\vec{\mathbf{g}}$ ersetzen, der wie folgt definiert ist:

$$\vec{\mathbf{g}} = \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} \tag{19}$$

(Sprich: die Gravitation wirkt ausschließlich in z-Richtung). Durch Einsetzen von Gleichungen 18 und 19 in Gleichung 17 erhalten wir folgendes:

$$E = \frac{1}{2} m \, \dot{\vec{\mathbf{x}}}^2 + m \, \vec{\mathbf{x}}^T \vec{\mathbf{g}}$$

(Achtung: Kettenregel beachten – erinnern wir uns an unsere Abkürzung: $\vec{\mathbf{x}} = \vec{\mathbf{x}}(s(t))$!)

$$\begin{aligned} &= \frac{1}{2} m \, (\dot{s} \, \vec{\mathbf{x}}_s)^2 + m \, \vec{\mathbf{x}}^T \vec{\mathbf{g}} \\ &= \frac{1}{2} m \, \dot{s}^2 \, \vec{\mathbf{x}}_s^2 + m \, \vec{\mathbf{x}}^T \vec{\mathbf{g}}. \end{aligned} \tag{20}$$

Hierbei bezeichnet $\vec{\mathbf{x}}_s$ die Ableitung von $\vec{\mathbf{x}}$ nach der Variablen s und $\vec{\mathbf{x}}^T$ die *Transponierung* des Vektors $\vec{\mathbf{x}}$. Die Transponierung ist notationstechnisch notwendig, da wir jetzt teilweise mit *Skalarprodukten* von Vektoren arbeiten.

Gleichung 20 kann man leicht umstellen und erhält eine einfache Differentialgleichung für die momentane Geschwindigkeit des Massepunktes (Achterbahnzuges) auf der Bahnkurve:

$$\dot{s} = \pm \sqrt{\frac{2(E - m \, \vec{\mathbf{x}}^T \vec{\mathbf{g}})}{m \, \vec{\mathbf{x}}_s^2}}. \tag{21}$$

Diese könnten wir eigentlich für unser weiteres Vorgehen nutzen, aber leider hat sie zwei mögliche Formen: eine positive und eine negative. Und es ist nicht klar zu Entscheiden, welche der beiden Lösungen wir benötigen. Wir müssen also etwas an der DGL ändern, damit sie eindeutig lösbar wird.

Durch einen Blick auf die Gleichung 20 sehen wir, dass unser „Problem“ durch das Quadrat der momentanen Geschwindigkeit verursacht wird. Leiten wir diese Gleichung nach der Zeit ab wird dieses Problem verschwinden. Also:

$$\dot{E} = \frac{1}{2} m \, \overbrace{\dot{s}^2 \, \vec{\mathbf{x}}_s^2}^{\cdot} + m \, \dot{\vec{\mathbf{x}}}^T \vec{\mathbf{g}}$$

(Kettenregel und Produktregel beachten)

$$= \frac{1}{2} m \left[\dot{s}^2 \, \dot{\vec{\mathbf{x}}}_s^2 + \overbrace{\dot{s}^2}^{\cdot} \, \vec{\mathbf{x}}_s^2 \right] + m \, \dot{s} \, \vec{\mathbf{x}}_s^T \vec{\mathbf{g}}$$

(Kettenregel beachten)

$$= \frac{1}{2} m \left[2 \, \dot{s}^3 \, \vec{\mathbf{x}}_s^T \vec{\mathbf{x}}_{s,s} + 2 \, \dot{s} \, \ddot{s} \, \vec{\mathbf{x}}_s^2 \right] + m \, \dot{s} \, \vec{\mathbf{x}}_s^T \vec{\mathbf{g}}.$$

Wieder stellen wir diese Gleichung um, diesmal nach $\ddot{s}$:

$$\ddot{s} = \frac{\dot{E} - m \, \dot{s} \, \vec{\mathbf{x}}_s^T \vec{\mathbf{g}} - m \, \dot{s}(t)^3 \, \vec{\mathbf{x}}_s^T \vec{\mathbf{x}}_{s,s}}{m \, \dot{s} \, \vec{\mathbf{x}}_s^2}. \tag{22}$$

Die Energie ist durch den Erhaltungssatz 16 eine Konstante. Die Ableitung einer Konstanten ist bekanntlich 0, womit wir erhalten:

$$\ddot{s} = -\frac{m\,\dot{s}\,\vec{\mathbf{x}}_s^T\vec{\mathbf{g}} + m\,\dot{s}^3\,\vec{\mathbf{x}}_s^T\vec{\mathbf{x}}_{s,s}}{m\,\dot{s}\,\vec{\mathbf{x}}_s^2}.$$

Durch Kürzen von $\dot{s}$ und der Masse[11] m erhalten wir die endgültige Form unserer DGL:

$$\ddot{s} = -\frac{\vec{\mathbf{x}}_s^T\vec{\mathbf{g}} + \dot{s}^2\,\vec{\mathbf{x}}_s^T\vec{\mathbf{x}}_{s,s}}{\vec{\mathbf{x}}_s^2}. \tag{23}$$

Diese hat im Gegensatz zu unserer ersten DGL (Gleichung 21) eine eindeutige, wenn auch leicht kompliziertere Form. Sie erlaubt es uns, aus der momentanen Position und Geschwindigkeit des Zuges (s bzw. $\dot{s}$), sowie den ersten und zweiten Ableitungen der Geometrie ($\vec{\mathbf{x}}_s$ bzw $\vec{\mathbf{x}}_{s,s}$) und der Gravitationskonstanten ($\vec{\mathbf{g}}$) die momentane Beschleunigung ($\ddot{s}$) zu berechnen.

5.2 Mögliche Modellerweiterungen

Das oben beschriebene Modell lässt sich recht einfach um sogenannte Quell- und Senkterme erweitern, welche dann zusätzliche Kräfte wie Antrieb und Reibung modellieren können.

Die zentrale Beobachtung, die eine solche Erweiterung erlaubt, ist, dass in diesem Fall die *Gesamtenergie des Systems keine Konstante mehr ist*. Wenn man also Gleichung 22 auf der vorherigen Seite hernimmt und dort $\dot{E}$ durch eine Funktion ersetzt, welche die gewünschte Änderung der Gesamtenergie beschreibt, hat man das Modell schon erweitert.

Beispielsweise würde die Gleichung

$$\dot{E} = -c \cdot |\dot{\vec{\mathbf{x}}}|^2 \tag{24}$$

so etwas wie eine *Luftreibung* modellieren: diese wächst ja bekanntlich mit dem Quadrat der Geschwindigkeit. In die positive Konstante c würde dann etwa der *Luftwiderstandsbeiwert*[12] der Achterbahn und die *Dichte* der Luft einfließen. Auf gleiche Art und Weise können weitere Erweiterungen erfolgen – diese tauchen dann in Gleichung 24 als zusätzliche Summanden auf.

5.3 Geometrieableitungen

Um Gleichung 23 berechnen zu können benötigen wir noch die erste und zweite Ableitung der Geometrie nach dem Parameter s. Dies sind aber genau die Gleichungen, welche wir schon zum Aufstellen des Gleichungssystems in Abschnitt 4.3 auf Seite 5 benötigt haben (Gleichungen 4 und 5).

5.4 Zeitintegration

Die DGL 23 gibt an, wie man aus gegebenen Werten (rechte Seite der Gleichung) die Beschleunigung $\ddot{s}$ berechnen kann.

Was wir jetzt brauchen ist ein Verfahren um aus dieser momentanen Beschleunigung die Position des Zuges auf der Achterbahnkurve zu einem bestimmten Zeitpunkt zu berechnen – denn das ist ja genau das, was wir für die Darstellung des Zuges benötigen. Solche Verfahren gibt es natürlich schon; es sind Integrationsverfahren für gewöhnliche Differentialgleichungen. Sie approximieren numerisch mit Hilfe eines Startwertes für $s(t_i)$ und einer Differentialgleichung $\dot{s}$ den Funktionswert zum Zeitpunkt $t_i + \delta t$. Durch wiederholtes Anwenden des Verfahrens kann man so eine Näherungslösung für $s(t)$ für jedes Vielfache t von δt, ausgehend von einem Startwert, berechnen.

[11] Die Masse des Zuges kürzt sich tatsächlich aus der DGL raus. Man denke an das Experiment mit Blei und Feder im Vakuum – da keine Luftreibung vorhanden ist fallen beide gleich schnell.

[12] Hierbei handelt es sich um den berühmten c_w-Wert.

Da unsere DGL allerdings eine zweifache Zeitableitung beschreibt – mathematisch benannt: sie ist eine „DGL zweiter Ordnung“ – benötigen wir einen kleinen Trick um ein solches Verfahren anwenden zu können. Wir schreiben unsere DGL zweiter Ordnung in ein System von zwei DGLs erster Ordnung um:

$$\begin{bmatrix} \dot{u}(t) \\ \dot{v}(t) \end{bmatrix} = \begin{bmatrix} v(t) \\ -\frac{\vec{\mathbf{x}}_s^T \vec{\mathbf{g}} + \dot{s}^2 \vec{\mathbf{x}}_s^T \vec{\mathbf{x}}_{s,s}}{\vec{\mathbf{x}}_s^2} \end{bmatrix}. \tag{25}$$

Dies bedeutet im Endeffekt nichts anderes als das Integrationsverfahren *zweimal* auf die ursprüngliche DGL für $\ddot{s}$ anzuwenden.

5.5 Startwerte für die Zeitintegration

Für die Zeitintegration des DGL-Systems 25 benötigen wir noch Startwerte $s(0) = s_0$ und $\dot{s}(0) = \dot{s}_0$. Erinnern wir uns an die Bedeutung von s: $s(t)$ ist die Position des Achterbahnzuges zum Zeitpunkt t und $\dot{s}(t)$ ist die Geschwindigkeit – aber *im Parameterraum*, nicht im 3D-Raum.

Die Startposition können wir auch am besten im Parameterraum angeben, da eine absolute Position im 3D-Raum nur Probleme verursacht: es ist nicht sofort klar, ob eine Position überhaupt „legal“ ist, also sich auch auf einem Splinestück befindet. Wir können also als Startposition z.B. wählen:

$$s_0 = 0. \tag{26}$$

Dies bedeutet dann, das wir am Anfang des ersten Splinestücks starten wollen. Genauso sind auch andere Werte möglich, z.B. würde $s_0 = 2,5$ bedeuten, dass wir auf der Hälfte des dritten Splinestücks starten wollen.

Anders sieht das bei der Anfangsgeschwindigkeit $\dot{s}_0$ aus. Diese ist nicht intuitiv im Parameterraum zu setzen, aber im 3D-Raum sehr wohl. Dies kann man sich am besten an den vorkommenden Einheiten veranschaulichen: im 3D-Raum ist die Einheit z.B. „Meter pro Sekunde“, wohingegen im Parameterraum die Einheit „Splinestücke pro Sekunde“ wäre und somit unabhängig von den konkreten Ausmaßen der Bahn selbst. Aus diesem Grund wollen wir als Anfangsgeschwindigkeit $|\dot{\mathbf{x}}_0|$ *in Richtung der Bahnkurve*[13] vorgeben.

Die DGL braucht aber weiterhin $\dot{s}_0$ als Startwert, welchen wir also ausrechnen müssen:

$$\begin{aligned} |\dot{\mathbf{x}}_0| &= |\mathbf{x}_{s,0}\, \dot{s}_0| \\ &= |\mathbf{x}_{s,0}|\, |\dot{s}_0| \\ \Rightarrow |\dot{s}_0| &= |\dot{\mathbf{x}}_0| / |\mathbf{x}_{s,0}|. \end{aligned} \tag{27}$$

Mit diesen Anfangswerten ($|\dot{\mathbf{x}}_0|$ und s_0) können wir die Zeitintegration starten.

6 Runge-Kutta Verfahren zur Zeitintegration

Wie oben schon erwähnt benötigt man zur Lösung der Differentialgleichungen ein numerisches Verfahren. Die Idee dieser Verfahren lässt sich am einfachsten am sogenannten *Euler Verfahren* verstehen, welches hier exemplarisch Beschrieben werden soll. Im Praktikum selbst findet es jedoch keine Verwendung; dort kommt ein *Runge-Kutta Verfahren* zum Einsatz (mehr dazu später).

[13] Normalerweise ist eine Geschwindigkeit im 3D-Raum ebenfalls ein 3D-Vektor. Da die Richtung aber durch die Bahnkurve vorgegeben ist reicht, es den Betrag des Vektors vorzugeben.

6.1 Das Eulersche Polygonzugverfahren

Die Aufgabe bei der Lösung einer DGL besteht darin, eine *unbekannte Funktion* $x(t)$, von der lediglich einige Eigenschaften (meistens die erste Ableitung, $\dot{x}(t)$) bekannt sind, zu finden. Dies ist selten analytisch geschlossen möglich, daher verwendet man numerische Verfahren. Das einfachste Verfahren ist, die unbekannte Funktion durch einen Linienzug zu approximieren. Am Startpunkt sind Funktionswert $x(0) = x_0$ (der sogenannte Anfangswert) und natürlich die Steigung der Funktion bekannt – diese ergibt sich ja gerade aus der Differentialgleichung. Jetzt geht man ein Stück entlang der Geraden, die dieselbe Steigung wie die Funktion hat. Dieser Punkt bildet den nächsten Anfangswert.

Gegeben sind:

$$\dot{x}(t) = f(t, x), \quad x(0) = x_0.$$

Dann lässt sich eine Approximation der gesuchten Funktion $x(t)$ inkrementell nach dem folgenden Algorithmus bestimmen:

$$x(\delta t \cdot (i+1)) = x(\delta t \cdot i) + \delta t f(i \cdot \delta t, x(i \cdot \delta t)).$$

Der Parameter δt wird gewöhnlich als *Zeitschrittweite* bezeichnet. Abbildung 3 zeigt, wie eine Lösung der DGL $\dot{f} = f$ mit Hilfe dieses Verfahrens aussieht. Abbildung 4 auf der nächsten Seite zeigt die Lösung mit kleinerer Zeitschrittweite δt. Man kann deutlich die Verbesserung sehen. Es lässt sich sogar zeigen, dass die Lösung dieses Algorithmus für $\delta t \to 0$ gegen die echte Lösung der Differentialgleichung konvergiert. Durch seiner Einfachheit hat dieser Algorithmus jedoch Nachteile bezüglich der Stabilität[14] und der Genauigkeit. Daher soll für das Praktikum ein anderer Algorithmus verwendet werden, der im Prinzip jedoch genauso funktioniert.

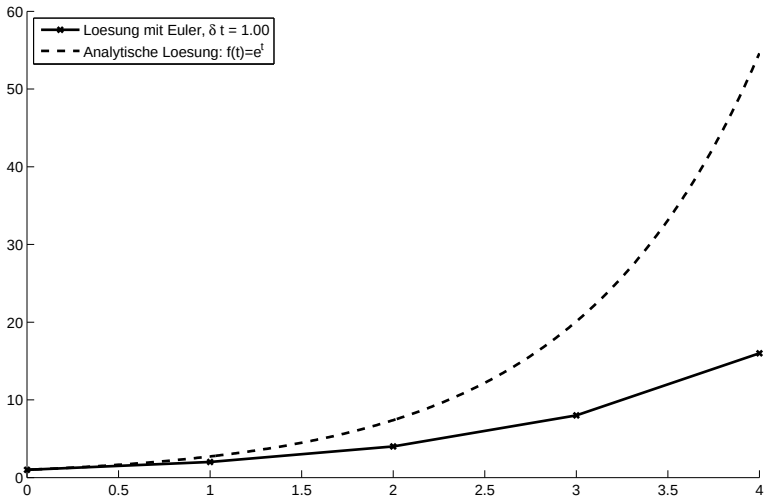

Abbildung 3: *Das Eulersche Polygonzugverfahren für die DGL* $\dot{f} = f$ *mit dem Startwert* $f_0 = 1$ *und* $\delta t = 1.0$. *Die analytische Lösung ist* $f(t) = e^t$.

6.2 Das Runge-Kutta-Verfahren vierter Ordnung

Das Runge-Kutta-Verfahren vierter Ordnung (siehe z.B. [7]) berechnet zunächst einige Zwischenwerte, aus denen dann der endgültige Wert für den nächsten Zeitschritt bestimmt wird. Dadurch erhöht sich die Stabilität und Genauigkeit. „Bezahlt“ wird dies mit einem erhöhten Rechenaufwand, da jetzt Funktionswerte der DGL viermal pro Zeitschritt berechnet werden müssen, anstatt

[14] Stabilität meint, dass die Lösung bei zu grosser Zeitschrittweite „explodieren“ kann

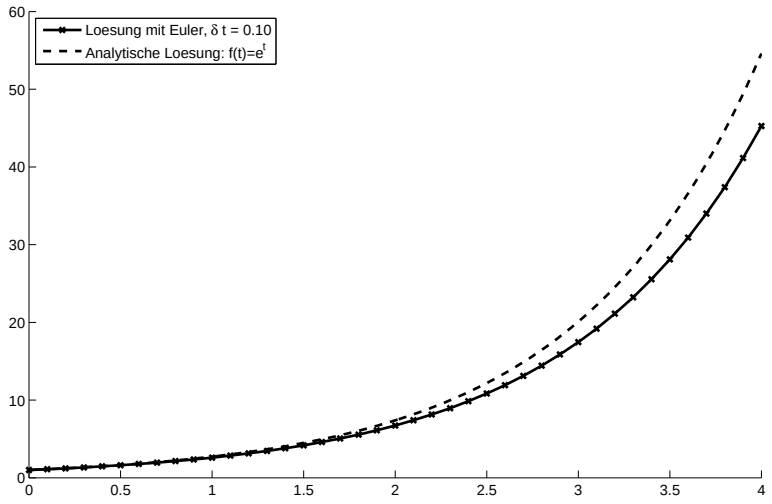

Abbildung 4: *Das Eulersche Polygonzugverfahren für die DGL* $\dot{f} = f$ *mit dem Startwert* $f_0 = 1$ *und* $\delta t = 0.1$*. Die analytische Lösung ist* $f(t) = e^t$*.*

nur einmal wie beim Euler-Verfahren. In Abbildung 5 auf der nächsten Seite wird die Lösung der DGL $\dot{f} = f$ mit dem Runge-Kutta-Verfahren 4. Ordnung und der gleichen Zeitschrittweite wie in Abbildung 4 gezeigt. Die numerische Lösung läßt sich von der analytischen in diesem Beispiel nicht mehr unterscheiden.

Gegeben ist das Runge-Kutta-Verfahren vierter Ordnung durch den folgenden Satz von einfachen Gleichungen[15]:

$$\begin{aligned} k_1 &= f(t_i, x_i) \\ k_2 &= f(t_i + (1/2)\delta t, x_i + (1/2)\delta t k_1) \\ k_3 &= f(t_i + (1/2)\delta t, x_i + (1/2)\delta t k_2) \\ k_4 &= f(t_i + \delta t, x_i + \delta t k_3) \\ x_{i+1} &= x_i + (\delta t/6)(k_1 + 2k_2 + 2k_3 + k_4). \end{aligned}$$

Hierbei ist f die DGL, welche integriert werden soll, und i wurde als Index für die Zeit eingeführt.

Hinweis: falls auch dieses Verfahren Stabilitätsprobleme zeigt wurde der Zeitschritt δt zu groß gewählt und muss reduziert werden.

6.3 Weiterführende Verfahren

Es gibt auch Verfahren, welche die Schrittweite während der Integration abhängig von einem geschätzten Integrationsfehler automatisch vergößern und verkleinern – dies sind sogenannte *adaptive* Verfahren, welche wir hier aber nicht weiter besprechen wollen. Es steht Ihnen natürlich frei ein solches Verfahren trotzdem zu verwenden; Stichwörter hierzu sind „Runge-Kutta-Fehlberg Verfahren“ [2] und „Dormand-Prince Verfahren“ [1] – beides sind sogenannte „eingebettete Runge-Kutta Verfahren“.

Da die Zeitschrittweite bei diesen Verfahren adaptiv gewählt wird, kann man sich vorstellen, dass dies zu Problemen bei der Visualisierung führt: diese benötigt idealerweise genauso viele Zeitschritte pro Sekunde wie sie Bilder pro Sekunde anzeigt. Um dies zu erreichen kann man ein Interpolationsverfahren einsetzen, welches zwischen den berechneten Zeitschritten in bestimmter

[15] Auf eine Herleitung der Gleichungen wird hier bewusst verzichtet, da sie für das Praktikum keinerlei Relevanz hat.

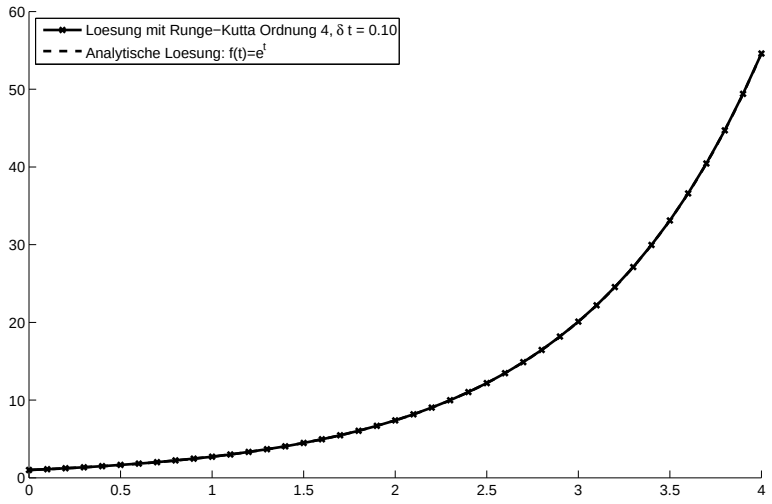

Abbildung 5: *Das Runge-Kutta-Verfahren vierter Ordnung für die DGL $\dot{f} = f$ mit dem Startwert $f_0 = 1$ und $\delta t = 0.1$. Die analytische Lösung ist $f(t) = e^t$. Es ist bei diesem Beispiel augenscheinlich kein Unterschied zwischen numerischer und analytischer Lösung feststellbar.*

Art und Weise interpoliert und es damit ermöglicht, *beliebige* Zeitschritte zu machen. Hierzu siehe zum Beispiel [3].

7 Der Simulationsalgorithmus

Mit der Geometrie aus Sektion 4, der Physik aus Sektion 5 und dem Lösungsverfahren aus Sektion 6 sind jetzt alle Zutaten für die „Achterbahn im Computer“ komplett. Nachdem die Streckenbeschreibung aus der XML-Datei eingelesen wurde, müssen daraus die Kontrollpunkte $\mathbf{a}_i$ und $\mathbf{b}_i$ für alle Streckenabschnitte bestimmt und in einer geeigneten Datenstruktur gespeichert werden.

Zur Beschreibung der Position des Zugs auf der Bahn soll lediglich die eine Variable $s(t)$ verwendet werden. Aus dieser Variablen muss daher der Streckenabschnitt und die Position auf diesem Streckenabschnitt bestimmt werden. Da jeder Streckenabschnitt genau einen Parameterbereich von $s \in [0..1]$ besitzt, ergibt sich der Abschnitt als der *ganzzahlige Anteil* von $s(t)$ modulo die Anzahl der Streckenabschnitte, n. Demzufolge ist der Parameter innerhalb des Streckenabschnitts genau der *Fließkommaanteil* von $s(t)$. Die Lösung der Differentialgleichungen fängt man also bei einer bestimmten Anfangsposition $s(0)$ mit einer Anfangsgeschwindigkeit $\dot{s}(0)$ an. Von dort aus benutzt man dann ein Runge-Kutta Verfahren zur Integration der Differentialgleichung 23 auf Seite 9. Die Parameter dieser Differentialgleichung ändern sich dann im Verlauf der Simulation, sobald $s(t)$ seinen ganzzahligen Wert ändert – also sobald der Streckenabschnitt wechselt.

Möglicherweise muss für die Zeitintegration ein relativ kleiner Zeitschritt gewählt werden. Daher empfiehlt es sich von vornherein das Programm so zu entwerfen, dass die Visualisierung von der Zeitintegration entkoppelt ist. Möglichkeiten sind hier zum Beispiel ein bestimmbarer Teiler zwischen den Integrations- und Visualisierungschritten oder die schon in Abschnitt 6.3 auf der vorherigen Seite erwähnten Interpolationsverfahren für Runge-Kutta Methoden.

7.1 Beispielimplementierung

Da die Beschreibungen in diesem Dokument trotz allem recht komplex und somit naturgemäß schwierig zu verstehen sind, steht den Teilnehmern ein vollwertiger, sehr gut kommentierter und strukturierter Prototyp der Achterbahnsimulation zur Verfügung. Er ist in MATLAB, einer Programmiersprache speziell für die numerische Mathematik, geschrieben. Diese Sprache ist allerdings

recht einfach zu lesen, daher kann der Prototyp sehr gut als Vorlage für die eigene Implementierung verwendet werden.

Der Prototyp enthält alle wichtigen in diesem Dokument beschriebenen Verfahren:

- Die Geometrie
- Berechnung der Bézier-Kontrollpunkte für alle Teilabschnitte aus den Stützstellen
- Die Differentialgleichung
- Das Zeitintegrationsverfahren
- Eine (sehr einfache) Visualisierung
- Mehrere Strecken zum Testen (allerdings nicht im XML Format)

Laden Sie sich also diesen Prototypen herunter und probieren Sie ihn aus. MATLAB steht auf den Terminalrechnern der Uni zur Verfügung, kann aber auch gegen eine geringe Schutzgebühr in der Rechenzentrumsberatung gekauft und auf dem eigenen Rechner installiert werden.

8 XML-Dateien

Während früher die Unterstützung von ASCII-Dateiformaten den Programmierer einiges an Zeit kostete kann man heutzutage durch den Einsatz von XML-Dateien XML-Parser aus der Schublade verwenden. Der Vorteil der menschlichen Lesbarkeit und einfachen Editierbarkeit von ASCII-Dateien bleibt trotzdem erhalten. Diese Vorteile wollen wir im SEP für uns nutzen und die Streckenbeschreibungen und die Simulationsparameter in einem XML-Format serialisieren.

Die XML-Dateien für die Streckenbeschreibung sollen in etwa folgendes Aussehen haben:

```
<?xml version="1.0"?>
<bahn>
    <stuetzpunkt>
        <nummer> ... <\nummer>
        <xcoord> ... <\xcoord>
        <ycoord> ... <\ycoord>
        <zcoord> ... <\zcoord>
    <\stuetzpunkt>
...
<\bahn>
```

Allerdings ist dieses XML Format noch nicht vollständig – durch Anforderungen an die Visualisierung (siehe [4]) werden hier weitere Daten benötigt (Stichwort: Giervektor). Hier herrscht also Diskussions- und Klärungsbedarf zwischen den Gruppen.

Weiterhin gibt es noch eine Datei, in der die Parameter für die Simulation stehen:

```
<?xml version="1.0"?>
<simparameter>
    <anfangsbedingungen>
        <s_null> ... <\s_null>
        <xdot_null> ... <\xdot_null>
    <\anfangsbedingungen>
    <physik>
        <grav> ... <\grav>
    <\physik>
<\simparameter>
```

Die exakte Spezifizierung des Dateiformats muss *durch alle Gruppen in Zusammenarbeit* erfolgen, da es die Schnittstelle zwischen den Projekten repräsentiert. Es bieten sich formale Beschreibungen durch *XML-Schemata* an, da diese automatisiert überprüfbar und somit auch testbar sind.

9 Allgemeine Hinweise

In diesem Praktikum liegt das Hauptaugenmerk klar auf der Softwaretechnik. In diesem Dokument wird zwar stark von Mathematik Gebrauch gemacht; der Grund hierfür ist jedoch, Ihnen die nötigen Grundlagen in möglichst verdaulicher Form zugänglich zu machen. Im Praktikum selbst kommt es dann eher auf gute Inter- und Intrateamarbeit, einen durchdachten Softwareentwurf, die strukturierte Umsetzung der Algorithmen und nicht zuletzt die Qualität der Dokumentation an.

Es wird empfohlen, dass die Gruppen sich frühzeitig um eine Unterteilung des Programms in Module kümmern. Ungeachtet der Komplexität der Aufgabe sind nach Absprache mit dem HiWi oder dem betreuenden Assistenten selbsterdachte Erweiterungen möglich und sogar wünschenswert, soweit diese die geforderte Funktionalität nicht einschränken. Bei Unklarheiten sollte nach Möglichkeit stets zuerst der HiWi gefragt werden!

Termine werden entweder per Mail oder auf der zugehörigen Institutswebseite bekannt gegeben. Weitere Informationen und Termine werden auf den Webseiten des Instituts für Software Systems Engineering angegeben.

Zum Abschluss wünschen wir allen Teilnehmern viel Spaß mit dem Praktikum und freuen uns auf viele rasante Achterbahnfahrten am Ende des Semesters.

Literatur

[1] Dormand, J. R. und P. J. Prince: *A family of embedded Runge-Kutta formulae.* Journal of Computational and Applied Mathematics, 6(1):19–26, März 1980.

[2] Fehlberg, E.: *Klassische Runge-Kutta-Formeln vierter und niedrigerer Ordnung mit Schrittweiten-Kontrolle und ihre Anwendung auf Wärmeleitungsprobleme.* Computing, 6(1):61–71, März 1970.

[3] Gladwell, I., L. F. Shampine, L. S. Baca und R. W. Brankin: *Practical Aspects of Interpolation in Runge-Kutta Codes.* SIAM Journal on Scientific and Statistical Computing, 8(3):322–341, 1987.

[4] Kayser-Herold, Oliver, Andreas Keese, Markus Krosche, Martin Krosche und Oliver Pajonk: *Teilprojekt Achterbahnsimulator -Aufgabenbeschreibung für das Softwareentwicklungspraktikum am Institut für Wissenschaftliches Rechnen*, März 2009.

[5] Krosche, Martin und Oliver Pajonk: *Teilprojekt Achterbahneditor - Aufgabenbeschreibung für das Softwareentwicklungspraktikum am Institut für Wissenschaftliches Rechnen*, März 2009.

[6] Prautzsch, Hartmut, Wolfgang Boehm und Marco Paluszny: *Bézier and B-spline techniques.* Springer Verlag, 2002.

[7] Schwarz, Hans Rudolf: *Numerische Mathematik.* B.G. Teubner, 1997.