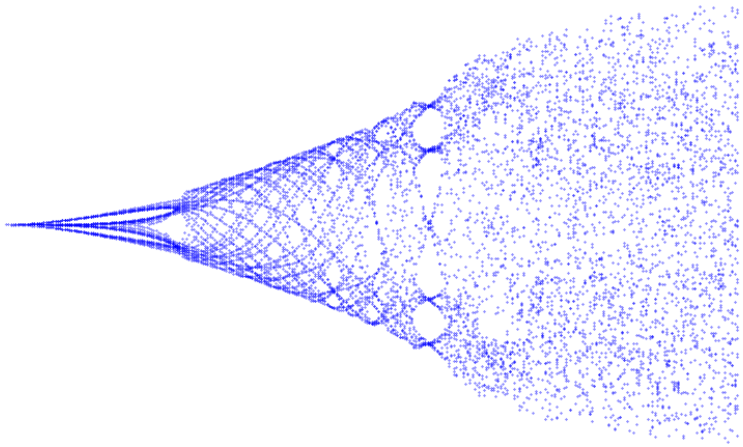
Kantonsschule Enge

Maturitätsarbeit



Theorie und Anwendung der Beschreibung dynamischer Systeme: Vom d'Alembertschen Prinzip bis zum Chaos

Betreuende Lehrperson:

Erich Schurtenberger

Verfasser der Arbeit:

Leo Horber, W4c

Abgabedatum: 17.12.2024

Selbständigkeitserklärung

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbstständig und ohne Benützung anderer als der angegebenen Quellen oder Hilfsmittel verfasst bzw. gestaltet habe. Datum:

Datum: _____ Unterschrift: _____

Inhaltsverzeichnis

Vorwort

Im Physikunterricht am Gymnasium und auch als ich im Rahmen des Schülerstudiums Physik I und II an der UZH belegte, wurde mir die Mechanik als eine sehr einfache Sache verkauft. Man bestimmt einfach die Summe der Kräfte auf einen Körper und löst die daraus resultierende Bewegungsgleichung analytisch. Als ich versuchte, dieses Verfahren auf eigene Systeme anzuwenden, merkte ich schnell, dass dieses Vorgehen fast ausschliesslich in den speziell gewählten Problemen des Unterrichts funktionierte. Beim Erlernen anderer Vorgehensweisen fiel mir auf, dass es keine kompakte Darstellung der wichtigsten Methoden zur Beschreibung komplexerer dynamischer Systeme gab. Das motivierte mich zu dieser Arbeit.

Diese Arbeit wäre nicht zustande gekommen ohne meinen Betreuer Erich Schurtenberger. Ihm gilt daher ein besonderer Dank für seine stete Hilfsbereitschaft.

Einleitung

Im Gymnasium wird zur Beschreibung eines dynamischen Systems von der Newtonschen Mechanik ausgegangen. Man bestimmt die Bewegungsgleichung eines Körpers also einfach, indem man die Summe der Kräfte, die auf ihn wirken, bestimmt und diese mit $m\ddot{\vec{x}}$ gleichsetzt. Diese Bewegungsgleichung wird dann analytisch gelöst und in einem einfachen Ort-Zeit-Diagramm dargestellt. Bei jedem dieser Schritte treten Probleme oder Unvollständigkeiten auf, wenn sie auf komplexere Systeme angewendet werden. Dort zeigt sich oftmals, dass die Vektoraddition der Kräfte sehr rechenintensiv ist. Ausserdem sind die Bewegungsgleichungen in vielen Fällen analytisch nicht lösbar. Zudem haben komplexere Systeme oft viele Variablen mit teilweise komplexem Verhalten, die nicht so einfach dargestellt werden können. Ohne passende Werkzeuge ist es schwierig, aussagekräftige Informationen über das System zu erhalten. Diese Arbeit beschäftigt sich deshalb mit der Frage:

Wie können komplexe, dynamische Systeme beschrieben werden?

Die Arbeit orientiert sich an drei Schwerpunkten, die sich aus den drei oben genannten Problemen ergeben. Also: Wie kann die Bewegungsgleichung effizient hergeleitet werden? Wie kann diese Bewegungsgleichung numerisch gelöst werden? Wie können die numerischen Resultate interpretiert werden? In dieser Arbeit wurde die nötige Literatur zusammengetragen, um diese Fragen zu beantworten. Das erste Kapitel beantwortet die erste, das zweite Kapitel die zweite und das dritte Kapitel die dritte Leitfrage. Zusätzlich wurden die Methoden simultan zu ihrer Erarbeitung auf ein Leitbeispiel angewendet. Als solches dient das planare, gekoppelte Federpendel.

Die Anwendungen wurden hauptsächlich mit Python implementiert. Der Code zu den damit erstellten Abbildungen finden Sie in dem Appendix.

Kapitel 1

Lagrange-Gleichung

In diesem Kapitel soll der erste Schwerpunkt beantwortet werden. Nämlich: Wie kann die Bewegungsgleichung effizient hergeleitet werden? In der klassischen Mechanik gibt es zwei Alternativen zur Newtonschen Mechanik. Der Lagrange-Formalismus und die Hamiltonsche Mechanik. Die zweitere ist eine stark verallgemeinerte Theorie. Für ein System wie das planare gekoppelte Federpendel ist der Lagrange-Formalismus und die darin enthaltene Lagrange-Gleichung die passende Wahl[1]. In diesem Kapitel soll daher die Lagrange-Gleichung hergeleitet und auf das Leitbeispiel angewendet werden.

1.1 Einige Definitionen

Für die Herleitung der Lagrange-Gleichung müssen zuerst einige Begriffe geklärt werden.

1.1.1 Zwangsbedingungen

In vielen der betrachteten Vorgänge ist die Bewegung des Körpers nicht völlig frei. Der Körper kann sich also nicht in allen Raumrichtungen uneingeschränkt bewegen. Denken wir uns als Beispiel eine schnurgerade Schiene. Eine Eisenbahn auf dieser Schiene kann sich nur entlang der Schienen, d.h. einer Dimension bewegen. Wenn man das kartesische Koordinatensystem (x, y, z) in dem Beispiel so orientiert, dass die x-Achse parallel zu der Schiene liegt,

[1] Weil das System holonom ist. Was das genau bedeutet, wird sich später klären.

dann könnte man die Zwangsbedingung folgendermassen ausdrücken:

$$y = 0$$
$$z = 0$$

Vorher hatte man also drei freie Variablen (x, y, z), die man berechnen musste. Mit den Zwangsbedingungen hat man nur noch eine Variable x, man könnte sie auch generalisiert Koordinate nennen. Die Anzahl an unbekannten Grössen hat von 3 auf 1 abgenommen. Ausserdem braucht es auch Kräfte, die sicherstellen, dass der Zug tatsächlich auf den Schienen bleibt, diese nennt man Zwangskräfte. Dies vereinfacht das Problem der Zugbeschreibung erheblich, da nur noch eine Koordinate, nämlich die x-Koordinate, berücksichtigt werden muss. Ausserdem sieht man hier schon, dass eine gute Wahl des Koordinatensystems wichtig ist, damit die Zwangsbedingungen ihre Wirkung entfalten können.

Allgemein sind Zwangsbedingungen Regeln für das System, welches es auf einen Teil aller möglichen Vorgänge beschränkt. Sie bedingen Zwangskräfte, das sind Kräfte, die sie in dem Teil der möglichen Vorgänge halten.[2]

Diese Regeln können verschiedene Formen annehmen. Denken Sie sich ein System aus N Teilchen in drei Raumdimensionen. Ohne Zwangsbedingungen bräuchte jeder Ortsvektor $\vec{r}$ drei Koordinaten, um ihn zu beschreiben. Also:

$$\begin{aligned}
\vec{r}_1 &= (x_1, x_2, x_3) \\
\vec{r}_2 &= (x_4, x_5, x_6) \\
&\vdots \\
\vec{r}_N &= (x_{3N-2}, x_{3N-1}, x_{3N})
\end{aligned}$$

Bedenken Sie, dass x_i hier alle Arten von Koordinaten sein können. Also z.B. die Zylinderkoordinaten, Kugelkoordinaten oder kartesischen Koordinaten. Holonom ist eine Zwangsbedingung, wenn sie in der Form:

$$f(\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_{3N}, t) = 0$$

geschrieben werden kann.[3] Die holonomen Zwangsbedingungen sagen also, wie die einzelnen Koordinaten voneinander abhängen. Wenn die Zwangsbedingungen nicht in der Form dargestellt werden können, z.B. bei Ungleichungen, dann wird sie nicht-holonom genannt.

[2]Serlo, o.J.
[3]Serlo, o.J.

1.1.2 Generalisierte Koordinaten

Wenn ein System holonomen Zwangsbedingungen unterliegt, könnte man die Zwangsbedingungs-Gleichung nach einer Koordinate auflösen und als Funktion der anderen Koordinaten darstellen. So kann eine Koordinate durch eine Funktion anderer Koordinaten ersetzt werden. Stellen Sie sich vor, wir haben s holonome Zwangsbedingungen:

$$f_n(\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_{3N}, t) = 0$$

mit $n = 1, 2, ..., s$. Dann könnte man pro Zwangsbedingung eine der Koordinaten x_i durch eine Funktion der anderen ersetzen, also eliminieren. Insgesamt kann die benötigte Zahl also von $3N$ Koordinaten $x_1, x_2, ..., x_{3N}$ auf $3N - s$ neue generalisierte Koordinaten $q_1, q_2, ..., q_{3N-s}$ reduziert werden. Diese generalisierten Koordinaten sind diejenigen Koordinaten, die das System beschreiben, ohne dabei voneinander abhängig zu sein. Die N Ortsvektoren nehmen dann die Form:

$$\begin{aligned} \vec{r}_1 &= \vec{r}_1(q_1, q_2, \ldots, q_{3N-s}, t) \\ \vec{r}_2 &= \vec{r}_2(q_1, q_2, \ldots, q_{3N-s}, t) \\ &\vdots \\ \vec{r}_N &= \vec{r}_N(q_1, q_2, \ldots, q_{3N-s}, t) \end{aligned} \tag{1.1}$$

an. Sie werden also zu Funktionen der generalisierten Koordinaten. Doch das kann am besten an einem Beispiel verdeutlicht werden.

Beispiel (Pendel) In diesem Beispiel betrachten wir ein einfaches Pendel, das in einer Ebene schwingt (siehe Abbildung 1.1). Ziel ist es, die Bewegung des Punktes P an der Spitze des Pendels mit Länge R auf möglichst einfache Weise zu beschreiben, indem wir die voneinander abhängigen Koordinaten eliminieren. Dazu verwenden wir das Konzept der Zwangsbedingungen und der generalisierten Koordinaten. Man hat $N = 1$ Teilchen und braucht somit einen Ortsvektor. Der Ortsvektor hat also in kartesischen Koordinaten die Form $\vec{r}_1 = (x_1, x_2, x_3)$. Nun kann man zwei holonome Zwangsbedingungen erkennen. Da sich der Punkt auf einer Ebene und einer Kreisbahn bewegt,

gilt:

$$x_3 = 0$$
$$x_1^2 + x_2^2 - R^2 = 0$$

Alle Faktoren, die mit x_3 multipliziert werden, fallen somit weg. Die zweite Gleichung kann man z.B. nach x_2 auflösen und den Ortsvektor nur mit einer Koordinate als Variable darstellen: $\vec{r}_1 = (x_1, \sqrt{R^2 - x_1^2}, 0)$. Durch die Einführung geeigneter Koordinaten, nämlich den Zylinderkoordinaten, lässt sich das Problem weiter vereinfachen. Wenn das Problem in Zylinderkoordinaten (r, α, x_3) transformiert wird, werden die Zwangsbedingungen zu:

$$x_3 = 0$$
$$r = R = konst.$$

x_3 fällt wie vorher weg. Da r aber konstant ist, ist es keine Variable mehr, sondern ein Parameter. Somit hat man keine Abhängigkeiten mehr zwischen den Koordinaten und α ist die generalisierte Koordinate q_1. Es reicht also die Koordinate α als Variable um die Position des Körpers eindeutig zu bestimmen. Das vereinfacht die Berechnungen erheblich.

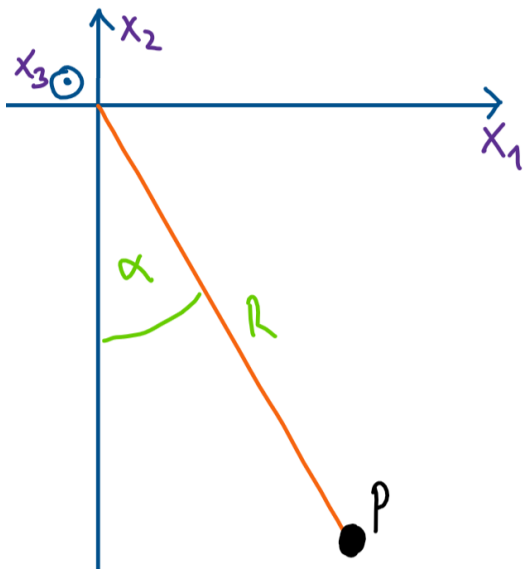


Abbildung 1.1: Pendel, mit kartesischen Koordinaten (x_1, x_2, x_3) und der generalisierten Koordinaten (α).

1.1.3 Virtuelle Verschiebung

Eine virtuelle Verschiebung $\delta\vec{r}$ ist eine mit den Zwangsbedingungen verträgliche, instantane und infinitesimale Koordinatenverschiebung[4]. Sie steht im Gegensatz zu der infinitesimalen Koordinatenverschiebung $d\vec{r}$, die in alle Richtungen gehen kann, also nicht zwingend mit den Zwangsbedingungen verträglich ist und während einer Zeit dt läuft. Der Nutzen der virtuellen Verschiebung besteht darin, dass sie die Zwangsbedingungen nicht verletzt, selbst wenn diese zeitabhängig sind, da sie den durch die Zwangsbedingungen definierten Raum nicht verlässt.

1.2 D'Alembertsches Prinzip

Das d'Alembertsche Prinzip ist die Grundlage der Lagrange-Mechanik. Man kann das d'Alembertsche Prinzip als Annahme verstehen, unter welchen die Lagrange-Mechanik funktioniert, d.h. die Lagrange-Mechanik beschränkt sich auf Systeme, in welchen das d'Alembertsche Prinzip gilt. Es kann aber gezeigt werden, dass das Prinzip allgemein für Systeme mit holonomen Zwangsbedingungen gilt, da im holonomen Fall die Zwangskräfte immer senkrecht auf den virtuellen Verschiebungen stehen[5]. Das Prinzip sagt, dass die Zwangskräfte keine virtuelle Arbeit verrichten. Das heisst:

$$\sum_{i=1}^{N} \vec{F}_i^z \cdot \delta\vec{r}_i = 0$$

Wobei $\vec{F}^z$ die Zwangskraft ist. Die Erfüllung der Gleichung kann zwei Gründe haben. Entweder steht die Verschiebung senkrecht auf der Zwangskraft (so im holonomen Fall), wodurch das Skalarprodukt verschwindet. Oder die Summe der Arbeiten verschwindet. Das ist das ganze Prinzip. Es ist aber, wie zuvor erwähnt, eine Annahme. Beispielsweise bei einem System mit Reibung stimmt die Gleichung nicht mehr[6].

Jetzt will ich noch eine nützlichere Form des d'Alembertschen Prinzips herleiten. Wir haben also verschiedene Kräfte. Einige davon sind Zwangskräfte, andere nicht. Wir können also die Kräfte $\vec{F}_i$ aufteilen in Zwangskräfte

[4] Spektrum, 2000b.
[5] Greiner, 2008, S. 249.
[6] Udwadia und Kalaba, 2001.

$\vec{F}_i^z$ und einen Rest $\vec{F}_i^a = \vec{F}_i - \vec{F}_i^z$. Diesen Rest $\vec{F}_i^a$ nennt man die angewandte Kraft. Somit gilt:

$$\sum_{i=1}^{N} \vec{F}_i \cdot \delta\vec{r}_i = \sum_{i=1}^{N} \left(\vec{F}_i^z + \vec{F}_i^a \right) \cdot \delta\vec{r}_i$$

In Kombination mit dem zweiten Newtonschen Axiom $\vec{F}_i = m_i \ddot{\vec{r}}_i$ erhält man:

$$\sum_{i=1}^{N} m_i \ddot{\vec{r}}_i \cdot \delta\vec{r}_i = \sum_{i=1}^{N} \vec{F}_i^a \cdot \delta\vec{r}_i + \sum_{i=1}^{N} \vec{F}_i^z \cdot \delta\vec{r}_i$$

$$\sum_{i=1}^{N} m_i \ddot{\vec{r}}_i \cdot \delta\vec{r}_i - \sum_{i=1}^{N} \vec{F}_i^a \cdot \delta\vec{r}_i = \sum_{i=1}^{N} \vec{F}_i^z \cdot \delta\vec{r}_i$$

Und wegen unseres Axioms des d'Alembertschen Prinzips ist die rechte Seite Null. Somit kann das d'Alembertsche Prinzip auch geschrieben werden als:

$$\sum_{i=1}^{N} \left(m_i \ddot{\vec{r}}_i - \vec{F}_i^a \right) \cdot \delta\vec{r}_i = 0 \tag{1.2}$$

Der Vorteil dieser Formulierung des d'Alembertschen Prinzips besteht darin, dass sie die Aufstellung der Bewegungsgleichungen ermöglicht, ohne dass die Zwangskräfte explizit berechnet werden müssen. Die Herausforderung liegt jedoch in der Bestimmung von $\delta\vec{r}_i$, da es sich dabei um eine virtuelle Verschiebung handelt. Es muss also mit den Zwangsbedingungen verträglich sein. Man muss das Koordinatensystem also so transformieren, dass $\delta\vec{r}_i$ nie der von den Zwangsbedingungen eingeschränkte Raum verlassen kann. Man sucht also ein Koordinatensystem, in welchem sich die variablen Koordinaten frei verändern können. Oder anders gesagt, das Koordinatensystem, in welchem die Koordinaten unabhängig voneinander sind. Die Lösung für dieses Problem sind die generalisierten Koordinaten. Denn sie erfüllen genau diese Anforderungen.

Beispiel (Verbundene Massen auf schiefer Ebene) In diesem Beispiel soll das d'Alembertsche Prinzip angewendet und damit die Bewegungsgleichung ermittelt werden. Das Beispiel wurde aus der Quelle[7] genommen.

[7] Greiner, 2008, S. 250-251.

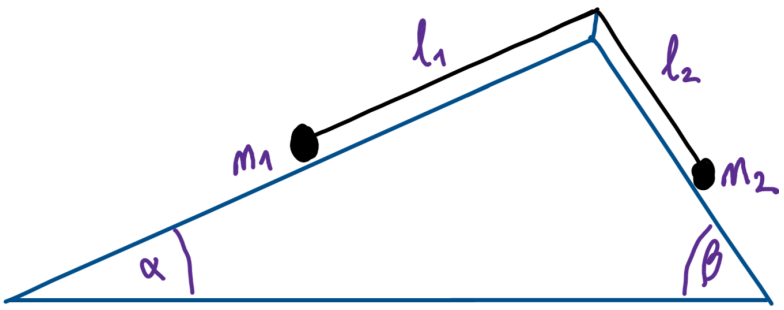


Abbildung 1.2: Zwei Massen auf schiefer Ebene, verbunden durch ein Seil

In dem zu betrachteten System liegen zwei Massen auf einer Ebene mit einem gewissen Steigungswinkel α und β. Ausserdem sind die beiden Massen durch ein Seil verbunden. Wir legen das Koordinatensystem entlang der jeweiligen Steigung mit dem Ursprung an der Spitze. Die beiden Massen werden also durch zwei Koordinaten l_1 und l_2 beschrieben, also $\vec{r}_1 = (l_1)$ und $\vec{r}_2 = (l_2)$. Wegen der Konstanz des Verbindungsseils gilt die Zwangsbedingung:

$$l_1 + l_2 = l$$

Und somit:

$$\delta l_1 = -\delta l_2$$

und

$$\ddot{\vec{l}}_1 = -\ddot{\vec{l}}_2$$

so können wir l_2 durch l_1 ersetzen. Wir brauchen also eine generalisierte Koordinate l_1 um das System vollständig zu beschreiben. Somit können auch die totalen Kräfte schreiben als:

$$m_1 \ddot{\vec{r}}_1 = m_1 \ddot{\vec{l}}_1$$

$$m_2 \ddot{\vec{r}}_2 = m_2 \ddot{\vec{l}}_2 = -m_2 \ddot{\vec{l}}_1$$

Das d'Alembertsche Prinzip lautet:

$$\left(m_1 \ddot{\vec{r}}_1 - \vec{F}_1^a\right) \cdot \delta\vec{l}_1 + \left(m_2 \ddot{\vec{r}}_2 - \vec{F}_2^a\right) \cdot \delta\vec{l}_2 = 0 \tag{1.3}$$

Die angewandten Kräfte $\vec{F}_1^a$ und $\vec{F}_2^a$ sind gegeben durch die zu der Ebene parallelen Anteile der Gewichtskraft. Also

$$\vec{F}_1^a = m_1 g \sin\alpha$$

$$\vec{F}_2^a = m_2 g \sin\beta$$

Setzen wir alles in diese Gleichung 1.3 ein, so folgt:

$$\left(m_1 \ddot{\vec{l}}_1 - m_1 g \sin\alpha\right) \cdot \delta l_1 + \left(m_2 \ddot{\vec{l}}_2 - m_2 g \sin\beta\right) \cdot \delta l_2 = 0$$

Bzw. mit den Zwangsbedingungen:

$$\left(m_1 \ddot{\vec{l}}_1 - m_1 g \sin\alpha + m_2 \ddot{\vec{l}}_1 + m_2 g \sin\beta\right) \delta l_1 = 0$$

oder:

$$\ddot{\vec{l}}_1 = \frac{m_1 g \sin\alpha - m_2 g \sin\beta}{m_1 + m_2}$$

Wir haben also ein Koordinatensystem gewählt, die Zwangsbedingungen bestimmt, damit die generalisierten Koordinaten entwickelt, um sie für die virtuelle Verrückung in dem d'Alembertschen Prinzip einzusetzen, das wir dann zu einer Bewegungsgleichung umformen.

1.3 Herleitung Lagrange-Gleichungen

Das Problem des d'Alembertschen Prinzips ist es also, $\delta\vec{r}_i$ durch generalisierte Koordinaten auszudrücken. Da $\vec{r}_i$ nach Gleichung 1.1 eine Funktion von q_α ist (mit $\alpha = 1, 2, ..., 3N - s$) kann $\delta\vec{r}_i$ mithilfe des totalen Differentials in Bezug auf die Variation δq_α ausgedrückt werden:

$$\delta\vec{r}_i = \sum_{\alpha=1}^{3N-s} \frac{\partial\vec{r}_i}{\partial q_\alpha}\, \delta q_\alpha$$

nach einsetzen in Gleichung 1.2 ergibt es:

$$\sum_{i=1}^{N} \left(m_i \ddot{\vec{r}}_i - \vec{F}_i^a\right) \cdot \sum_{\alpha=1}^{3N-s} \frac{\partial\vec{r}_i}{\partial q_\alpha}\, \delta q_\alpha = 0$$

Was umgeformt werden kann zu:

$$\sum_{\alpha=1}^{3N-s} \left(\sum_{i=1}^{N} \left(m_i \ddot{\vec{r}}_i - \vec{F}_i^a\right) \cdot \frac{\partial\vec{r}_i}{\partial q_\alpha}\right) \delta q_\alpha = 0$$

Da die einzelnen δq_α per Definition unabhängig voneinander sind, kann die Gesamtsumme nur dann für alle möglichen α Null sein, wenn jeder Term innerhalb der Klammern einzeln Null ist. Deshalb kann man schreiben:

$$\sum_{i=1}^{N} \left(m_i \ddot{\vec{r}}_i - \vec{F}_i^a \right) \cdot \frac{\partial \vec{r}_i}{\partial q_\alpha} = 0 \quad \text{für jedes } \alpha$$

Damit:

$$\sum_{i=1}^{N} \left(m_i \ddot{\vec{r}}_i \cdot \frac{\partial \vec{r}_i}{\partial q_\alpha} - \vec{F}_i^a \cdot \frac{\partial \vec{r}_i}{\partial q_\alpha} \right) = 0 \tag{1.4}$$

Nun wollen wir uns auf den ersten Term konzentrieren. Wir addieren und subtrahieren $\sum_{i=1}^{N} \left(m_i \dot{\vec{r}}_i \cdot \frac{d}{dt} \frac{\partial \vec{r}_i}{\partial q_\alpha} \right)$ gleichzeitig und wenden die Produktregel umgekehrt an:

$$\begin{aligned}
\sum_{i=1}^{N} m_i \ddot{\vec{r}}_i \frac{\partial \vec{r}_i}{\partial q_\alpha} &= \sum_{i=1}^{N} \frac{d}{dt} \left(m_i \dot{\vec{r}}_i \right) \frac{\partial \vec{r}_i}{\partial q_\alpha} + \sum_{i=1}^{N} \left(m_i \dot{\vec{r}}_i \cdot \frac{d}{dt} \frac{\partial \vec{r}_i}{\partial q_\alpha} \right) - \sum_{i=1}^{N} \left(m_i \dot{\vec{r}}_i \cdot \frac{d}{dt} \frac{\partial \vec{r}_i}{\partial q_\alpha} \right) \\
&= \sum_{i=1}^{N} \frac{d}{dt} \left(m_i \dot{\vec{r}}_i \frac{\partial \vec{r}_i}{\partial q_\alpha} \right) - \sum_{i=1}^{N} \left(m_i \dot{\vec{r}}_i \cdot \frac{d}{dt} \frac{\partial \vec{r}_i}{\partial q_\alpha} \right)
\end{aligned}$$

Wenn wir $\dot{\vec{r}}_i = \vec{v}_i$ definieren kann die Reihenfolge der Differentiation bezüglich t und q_α vertauscht werden[8]:

$$\frac{d}{dt} \left(\frac{\partial \vec{r}_i}{\partial q_\alpha} \right) = \frac{\partial}{\partial q_\alpha} \left(\frac{d \vec{r}_i}{dt} \right) = \frac{\partial \vec{v}_i}{\partial q_\alpha}$$

und mit der Identität[9]:

$$\frac{\partial \vec{r}_i}{\partial q_\alpha} = \frac{\partial \vec{v}_i}{\partial \dot{q}_\alpha}$$

Kann der obige Term umgeformt werden zu:

$$\begin{aligned}
\sum_{i=1}^{N} \left(m_i \ddot{\vec{r}}_i \cdot \frac{\partial \vec{r}_i}{\partial q_\alpha} \right) &= \sum_{i=1}^{N} \frac{d}{dt} \left(m_i \vec{v}_i \frac{\partial \vec{v}_i}{\partial \dot{q}_\alpha} \right) - \sum_{i=1}^{N} \left(m_i \vec{v}_i \cdot \frac{\partial \vec{v}_i}{\partial q_\alpha} \right) \\
&= \frac{d}{dt} \left(\frac{\partial}{\partial \dot{q}_\alpha} \sum_{i=1}^{N} \frac{1}{2} m_i v_i^2 \right) - \frac{\partial}{\partial q_\alpha} \left(\sum_{i=1}^{N} \frac{1}{2} m_i v_i^2 \right) \\
&= \frac{d}{dt} \left(\frac{\partial T}{\partial \dot{q}_\alpha} \right) - \frac{\partial T}{\partial q_\alpha}.
\end{aligned} \tag{1.5}$$

[8] Greiner, 2008, S. 253.

[9] Da $\dot{\vec{r}}_i = \sum_\alpha \frac{\partial \vec{r}_i}{\partial q_\alpha} \dot{q}_\alpha + \frac{\partial \vec{r}_i}{\partial t}$ und $\frac{\partial \dot{\vec{r}}_i}{\partial \dot{q}_\alpha} = 0$

Wobei T die kinetische Energie ist, mit:

$$\partial T = \partial \left(\sum_{i=1}^{N} \frac{1}{2} m_i v_i^2 \right) = \sum_{i=1}^{N} m_i \vec{v}_i \cdot \partial \vec{v}_i$$

Nun schauen wir uns den zweiten Term der Gleichung 1.4 an. Gehen wir also davon aus, dass $\vec{F}_i^a$ eine konservative Kraft ist. Somit kann $\vec{F}_i^a$ per Definition als negativer Gradient des Potentials $V(\vec{r}_1, \vec{r}_2, ..., \vec{r}_N)$ geschrieben werden[10]:

$$\vec{F}_i^a = -\vec{\nabla}_i(V) = -\frac{\partial V}{\partial \vec{r}_i}$$

Damit ist nach einsetzen in den zweiten Term aus Gleichung 1.4 und der umgekehrten Anwendung der Kettenregel:

$$\sum_{i}^{N} \vec{F}_i^a \cdot \frac{\partial \vec{r}_i}{\partial q_\alpha} = -\sum_{i}^{N} \frac{\partial V}{\partial \vec{r}_i} \cdot \frac{\partial \vec{r}_i}{\partial q_\alpha} = -\frac{\partial V}{\partial q_\alpha}$$

Wenn wir nun diesen Term und den Term aus Gleichung 1.5 in Gleichung 1.4 einsetzen ergibt es:

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{q}_\alpha}\right) - \frac{\partial T}{\partial q_\alpha} + \frac{\partial V}{\partial q_\alpha} = 0$$

Und da V nur von der Position abhängt, gilt:

$$\frac{\partial V}{\partial \dot{q}_\alpha} = 0 = \frac{d}{dt}\left(\frac{\partial V}{\partial \dot{q}_\alpha}\right)$$

Somit kann man schreiben:

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{q}_\alpha}\right) - \frac{d}{dt}\left(\frac{\partial V}{\partial \dot{q}_\alpha}\right) - \frac{\partial T}{\partial q_\alpha} + \frac{\partial V}{\partial q_\alpha} = 0$$

Mit der Summenregel gilt:

$$\frac{d}{dt}\left(\frac{\partial (T-V)}{\partial \dot{q}_\alpha}\right) - \frac{\partial (T-V)}{\partial q_\alpha} = 0$$

Wenn wir die sogenannte Lagrange-Funktion definieren als $L = T - V$ kann man die Lagrange-Gleichung schreiben:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_\alpha}\right) - \frac{\partial L}{\partial q_\alpha} = 0 \quad \text{für jedes } \alpha$$

[10] Universität Innsbruck, o.J. S.3.

1.4 Bewegungsgleichung des gekoppelten, planaren Federpendels

Im Folgenden soll mit der vorher hergeleiteten Lagrange-Gleichung die Bewegungsgleichungen eines gekoppelten planaren Federpendels (Abbildung 1.3) ermittelt werden.

In diesem System gibt es zwei um einen Abstand M voneinander getrennte Drehachsen. An diesen Achsen sind jeweils masselose, starre Pendel mit Länge r_1 und r_2 befestigt und am Ende dieser Pendel sind die Massen m_1 und m_2 befestigt. Die beiden Massen und damit auch die beiden Pendel sind mit einer Feder der Länge l, der Ruhelänge l_0 und der Federkonstante k verbunden.

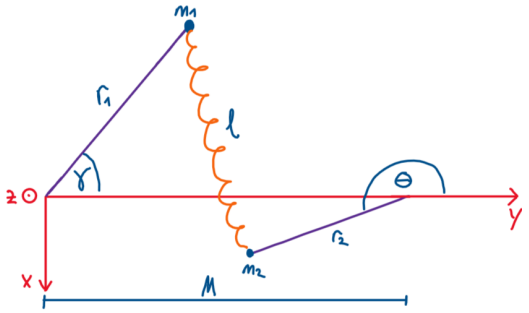


Abbildung 1.3: Schematische Darstellung eines planaren Federpendels

Unser Ziel ist es, die Bewegung der beiden Massen m_1 und m_2 zu ermitteln. Dafür folgen wir diesen drei Schritten:

1. System in generalisierten Koordinaten transformieren
2. Lagrange-Funktion $L = T - V$ in generalisierten Koordinaten ausdrücken
3. Lagrange-Funktion in die Lagrange-Gleichungen einsetzen und auflösen.

Für Schritt 1 müssen wir die Zwangsbedingungen bestimmen, um zu verstehen, wie die generalisierten Koordinaten aussehen müssen. Für dieses System lassen sich 4 holonome Zwangsbedingungen bestimmen:

$$z_{1,2} = 0$$

$$x_1^2 + y_1^2 - r_1^2 = 0$$
$$(x_2 - M)^2 + (y_2 - M)^2 - r_2^2 = 0$$

Sie beruhen darauf, dass das System einerseits planar ist und andererseits die Massen aufgrund der Natur eines starren Pendels eine konstante Länge zu ihrer Drehachse haben. Aufgrund dieser $s = 4$ holonomen Zwangsbedingungen sollte das System von $3N = 3 \cdot 2 = 6$ kartesischen auf $3N - s = 6 - 4 = 2$ generalisierte Koordinaten reduziert werden können. Wie fast immer im Pendelfall sind Polarkoordinaten die richtige Wahl für generalisierte Koordinaten. Das System braucht nur zwei Winkelangaben als Variablen, um es vollständig zu beschreiben. Damit kann man die Transformationsgleichungen von kartesischen zu Polarkoordinaten bestimmen:

$$x_1 = -\sin\gamma \cdot r_1$$
$$y_1 = \cos\gamma \cdot r_1$$
$$x_2 = -\sin\theta \cdot r_2$$
$$y_2 = M + \cos\theta \cdot r_2$$

Nun kommen wir zu Schritt 2. Wir müssen die Lagrange-Funktion bestimmen, wofür wir die Kinetische und potentielle Energie brauchen. Aus den Transformationsgleichungen kann man die Geschwindigkeiten bestimmen, die wir für die kinetische Energie der Lagrange-Funktion brauchen:

$$\dot{x}_1 = -\cos\gamma \cdot \dot{\gamma} \cdot r_1$$
$$\dot{y}_1 = -\sin\gamma \cdot \dot{\gamma} \cdot r_1$$
$$\dot{x}_2 = -\cos\theta \cdot \dot{\theta} \cdot r_2$$
$$\dot{y}_2 = -\sin\theta \cdot \dot{\theta} \cdot r_2$$

Somit ist die kinetische Energie:

$$\begin{aligned} T &= \frac{1}{2} m_1 (\dot{x}_1^2 + \dot{y}_1^2) + \frac{1}{2} m_2 (\dot{x}_2^2 + \dot{y}_2^2) \\ &= \frac{1}{2} m_1 \dot{\gamma}^2 r_1^2 + \frac{1}{2} m_2 \dot{\theta}^2 r_2^2 \end{aligned}$$

Nun wollen wir die potentielle Energie V in generalisierten Koordinaten bestimmen. V ist in diesem Fall ausschliesslich durch die potentielle Energie der

Feder gegeben. Wenn l den Abstand von den zwei Pendelspitzen bezeichnet, dann ist:

$$\begin{aligned} l = |\vec{r}_2 - \vec{r}_1| &= \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \\ &= \sqrt{\left(-\sin\theta \cdot r_2 + \sin\gamma \cdot r_1\right)^2 + \left(\cos\theta \cdot r_2 + M - \cos\gamma \cdot r_1\right)^2} \end{aligned}$$

Die Auslenkung Δl ist gegeben durch:

$$\Delta l = l - l_0$$

$$\Delta l = \sqrt{\left(-\sin\theta \cdot r_2 + \sin\gamma \cdot r_1\right)^2 + \left(\cos\theta \cdot r_2 + M - \cos\gamma \cdot r_1\right)^2} - l_0$$

Für die potentielle Energie gehen wir von einer linearen Abhängigkeit der Federkraft zur Auslenkung aus, d.h.:

$$V = \frac{1}{2} \cdot k \cdot \Delta x^2$$

$$= \frac{1}{2} \cdot k \cdot \left(\sqrt{\left(-\sin\theta \cdot r_2 + \sin\gamma \cdot r_1\right)^2 + \left(\cos\theta \cdot r_2 + M - \cos\gamma \cdot r_1\right)^2} - l_0 \right)^2$$

Mit der potentiellen und kinetischen Energie in den generalisierten Koordinaten ausgedrückt, kann man die Lagrange-Funktion L bestimmen:

$$L = T - V$$

$$\begin{aligned} =& \frac{1}{2} m_1 \dot{\gamma}^2 r_1^2 + \frac{1}{2} m_2 \dot{\theta}^2 r_2^2 \\ & - \frac{1}{2} k \left(\sqrt{\left(-\sin\theta \cdot r_2 + \sin\gamma \cdot r_1\right)^2 + \left(\cos\theta \cdot r_2 + M - \cos\gamma \cdot r_1\right)^2} - l_0 \right)^2 \end{aligned}$$

Nun können die Bewegungsgleichungen ermittelt werden, indem die hier entwickelte Lagrange-Funktion in die Lagrange-Gleichung einsetzt:

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{\theta}} - \frac{\partial L}{\partial \theta} = 0$$

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{\gamma}} - \frac{\partial L}{\partial \gamma} = 0$$

Mit Python berechnet[11] und von Hand vereinfacht ergibt das:

$$\ddot{\gamma} = \frac{k l_0 \left(M \sin(\gamma) + r_2 \sin(\gamma - \theta)\right)}{r_1 m_1 \sqrt{M^2 + 2M(r_2 \cos(\theta) - r_1 \cos(\gamma)) + r_1^2 + r_2^2 - 2 r_1 r_2 \cos(\gamma - \theta)}} - \frac{Mk \sin(\gamma)}{r_1 m_1} - \frac{k r_2 \sin(\gamma - \theta)}{r_1 m_1} \tag{1.6}$$

$$\ddot{\theta} = \frac{k l_0 \left(-M \sin(\theta) + r_1 \sin(\theta - \gamma)\right)}{r_2 m_2 \sqrt{M^2 + 2M(r_2 \cos(\theta) - r_1 \cos(\gamma)) + r_1^2 + r_2^2 - 2 r_1 r_2 \cos(\gamma - \theta)}} + \frac{Mk \sin(\theta)}{r_2 m_2} + \frac{k r_1 \sin(\gamma - \theta)}{r_2 m_2} \tag{1.7}$$

1.4.1 Anmerkung zu den Bewegungsgleichungen

Die hergeleiteten Bewegungsgleichungen sind gekoppelte, gewöhnliche Differentialgleichungen zweiter Ordnung[12]. Aufgrund der trigonometrischen Funktionen, der multiplikativen Zusammenhänge und der Wurzelfunktion weisen die Gleichungen starke Nichtlinearitäten auf[13].

Ich möchte jetzt kurz eine Intuition dafür geben, warum es nicht sinnvoll ist, nach einer analytischen Lösung zu suchen, sondern das System numerisch zu lösen. Wenn wir untersuchen wollen, ob die Gleichungen analytisch lösbar sind, müssen wir uns ansehen, was das genau bedeutet. Im Grunde bedeutet eine analytische Lösung einfach, dass die Lösung durch bestimmte, als elementar definierte Funktionen ausgedrückt werden kann. Die Liouville-Lösbarkeit beispielsweise lässt Komplexe Zahlen und die logarithmische Erweiterung zu[14] [15]. Die Lösbarkeit ist also durch die limitierte Anzahl an elementaren Funktionen eingeschränkt. Selbst wenn eine Lösung mit irgendwelchen konstruierten Funktionen möglich wäre, wären sie oft zu kompliziert, um charakteristische Eigenschaften des Systems zu erkennen[16]. Ausserdem wird das Problem nur darauf verschoben, diese neuen Funktionen numerisch zu ermitteln. Daher ist es am einfachsten, ein solches System direkt numerisch zu berechnen.

[11] siehe Appendix

[12] Lüdde, 2000, S.4.

[13] Lüdde, 2000, S. 18.

[14] Khovanskii, 2018, S. 6.

[15] Nach ihr wäre unser System nicht lösbar, was hier aber aufgrund des eingeschränkten Rahmens nicht hergeleitet wird.

[16] Jordan und Smith, 2007, S. 1.

Kapitel 2

Numerische Methoden

Das System soll also numerisch bestimmt werden. In diesem Kapitel sollen die grundlegenden Ideen:

1. Iteration
2. Runge-Kutta-Ansatz
3. Koeffizientenbestimmung durch Vergleich der Taylorentwicklung

eines numerischen Verfahrens und speziell eines Runge-Kutta-Verfahrens vorgestellt werden. Da die Herleitung des schlussendlich verwendeten Verfahrens sehr langwierig und nicht sonderlich fruchtbar ist, wird es für einen einfacheren Fall gemacht. Diese Herleitung kann aber analog auf die Herleitung anderer Runge-Kutta-Verfahren angewendet werden. Das letztlich verwendete Verfahren verwendet zudem eine automatische Schrittweitenkontrolle, die jedoch nicht erläutert wird. Ausserdem wird für eine bessere Intuition von numerischen Methoden das explizite Eulerverfahren erklärt. Die Herleitung wurde inspiriert durch diese Quelle[1].

2.1 Grundlegende Ideen der Runge-Kutta-Verfahren

2.1.1 Ausgangslage

Unsere Bewegungsgleichungen 1.6 und 1.7 können wir mit einem Trick als vier Differentialgleichungen erster Ordnung schreiben. Dafür setzen wir $\dot{\gamma} = z_1$

[1] Cryer, 2008.

und $\dot{\theta} = z_2$. Dann können wir unser System schreiben als:

$$\begin{aligned}\dot{\gamma} &= z_1 \\ \dot{z}_1 &= f(\gamma, \theta) \\ \dot{\theta} &= z_2 \\ \dot{z}_2 &= g(\gamma, \theta)\end{aligned}$$

Unser Ziel ist es $\gamma(t)$, $z_1(t)$, $\theta(t)$ und $z_2(t)$ für gewisse Anfangsbedingungen herauszufinden. Wir können das System kompakt darstellen als:

$$\frac{d}{dt}\vec{x}(t) = \vec{F}(\vec{x}(t), t) \tag{2.1}$$

mit

$$\vec{x}(t) = \begin{bmatrix}\gamma \\ z_1 \\ \theta \\ z_2\end{bmatrix}, \quad \vec{F}(\vec{x}(t), t) = \begin{bmatrix}z_1 \\ f(\gamma, \theta) \\ z_2 \\ g(\gamma, \theta)\end{bmatrix}.$$

mit gewissen Anfangsbedingungen $\vec{x}(t_0) = x_0$. Das ist ein Beispiel für ein System sogenannter Anfangswertprobleme[2]. Das kann verallgemeinert werden, wenn wir einen Zustandsvektor $q(t)$ und dessen Ableitung $f(q(t), t)$ definieren. Zur späteren Übersichtlichkeit werden Sie in dieser einfachen Form gezeigt, d.h. ohne Vektorpfeil. Dann handelt es sich allgemein um ein Anfangswertproblem, das in der folgenden Form geschrieben werden kann:

$$\begin{aligned}\frac{d}{dt}q(t) &= f(q(t), t), \\ q(t_0) &= q_0\end{aligned}$$

Mit dem Endpunkt t_f. Gesucht ist $q(t_f)$.
Unser Ziel ist es, das $q(t)$ zu finden, für das die Anfangsbedingungen gelten. Für verschiedene Anfangsbedingungen $q(t_0)$, die man wählt, gibt es verschiedene Funktionen, $q(t)$ die die Bewegung des Körpers beschreiben. Jedem Vektorpunkt $q(t)$ wird ein Vektor $f(q(t), t)$ zugeordnet. Die erste Gleichung beschreibt also ein Vektorfeld. $f(q(t), t)$ sagt uns die Änderungsrate und Änderungsrichtung des Vektors $q(t)$ aller Bahnpunkte aller Bahnen. Es beschreibt also die Bahn, entlang der sich der Körper mit bestimmten Startwerten bewegt. Bei einer möglichen numerischen Methode wäre also unser Ziel, die

[2]Mathepedia, o.J.

Bewegung eines Körpers im Vektorfeld mit Startpunkt $q(t_0)$ möglichst genau zu folgen. Doch das sieht man am besten an einem Beispiel.

Beispiel (Vektorfeld) Ein einfaches eindimensionales Beispiel für ein Anfangswertproblem wäre:

$$\dot{y}(y, x) = yx^2 - y$$

Mit der Anfangsbedingung:

$$y(1) = 1$$

y ist in diesem einfachen Fall nur 1-dimensional. Die erste Gleichung ordnet also jedem Punkt (x, y) einen Vektor $\dot{y}(x)$ zu. Wie Sie in Abbildung 2.1 sehen können, bestimmen diese Vektoren Bahnen. Die Anfangsbedingung bestimmt, welche der Bahnen genommen wird. Durch Separieren der Gleichung und Integrieren beider Seiten ergibt sich die Lösung:

$$y(x) = Ce^{\frac{1}{3}x(x^2-3)}$$

Die Integrationskonstante C kann man nun so modulieren, dass die Anfangsbedingung erfüllt wird. Mit $C = e^{\frac{2}{3}}$ ergibt sich als Lösung für unser Anfangswertproblem.

$$y(x) = e^{\frac{1}{3}(x^3-3x+2)}$$

Wie Sie in Abbildung 2.1 in Rot sehen können, folgt diese Funktion einer Bahn des Vektorfeldes, wobei sie die Anfangsbedingung erfüllt.

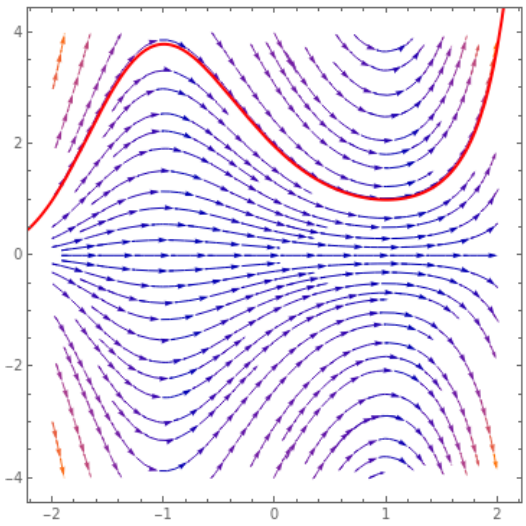


Abbildung 2.1: Vektorfeld für $\dot{y}(x) = yx^2 - y$ und in Rot die Funktion $y(x) = e^{\frac{1}{3}(x^3-3x+2)}$ welche das Anfangswertproblem erfüllt.

2.1.2 Iteration

Idee 1: Eine Idee besteht darin, das Intervall $[t_0, t_f]$ in Teilintervalle $t_0 < t_1 < t_2 < \ldots < t_n = t_f$ zu unterteilen und $q(t_0),\ q(t_1),\ q(t_2),\ \ldots\ ,\ q(t_f)$ (auf andere Art geschrieben: $q_0,\ q_1,\ q_2,\ \ldots\ ,\ q_f$) sukzessive zu bestimmen. Damit haben wir das Problem in viele kleine Teilprobleme unterteilt. Wir haben es sozusagen von einem globalen in ein lokales Problem verwandelt. Allgemein kann zur Berechnung eines dieser Teilprobleme der Ansatz:

$$q(t + \Delta t) = q(t) + \Delta t \Phi_m(q(t), t) \tag{2.2}$$

verwendet werden. Das allein hilft einem aber noch nichts, da man die Funktion $\Phi_m(q(t), t)$ nicht kennt. $\Phi_m(q(t), t)$ müsste die Änderungsrate zwischen den Punkten $q(t)$ und $q(t + \Delta t)$ sein. Eine perfekte Lösung, d.h. eine Lösung ohne Fehler für $\Phi_m(q(t), t)$ sähe also folgendermassen aus:

$$\Delta(q(t), t) := \begin{cases} \dfrac{q(t + \Delta t) - q(t)}{\Delta t}, & \text{falls } \Delta t \neq 0 \\ f(q(t), t), & \text{falls } \Delta t = 0 \end{cases} \tag{2.3}$$

Zu beachten ist, dass $\Phi_m = \Delta$ nichts bringt, da es nach einsetzen in Gleichung 2.2 per Konstruktion zu dem Ausdruck $q(t+\Delta t) = q(t+\Delta t)$ führt. Das zeigt auch, weshalb $\Delta(q(t), t)$ eine perfekte Lösung ist. Der Nutzen der Konstruktion ist, dass damit Fehler F_L eines Iterationsschrittes eines noch zu bestimmenden $\Phi_m(q(t), t)$ durch:

$$F_L = \Delta(q(t), t) - \Phi_m(q(t), t) \tag{2.4}$$

ermittelt werden kann. Diesen Fehler eines Iterationsschrittes nennt man auch den lokalen Fehler. Dem gegenüber steht der globale Fehler. Er ist der Unterschied zwischen dem numerisch berechneten und dem wahren Endewert. Damit können wir unsere Anforderungen an das numerische Verfahren etwas spezifizieren.

Ziel: Unser Ziel ist es, eine Konstruktion für $\Phi_m(q(t), t)$ zu finden. Dabei sollte sie effizient sein, d.h. einen möglichst kleinen Fehler bei möglichst kleinem Rechenaufwand besitzen.

Der kleine Rechenaufwand soll dadurch erreicht werden, dass das Verfahren keine Ableitungen von $f(q(t), t)$ enthalten soll. Später werden wir auch eine Idee für die Kontrolle des Fehlers bekommen.

2.1.3 Runge-Kutta-Verfahren

Idee 2: Die Runge-Kutta-Methoden verwenden einen bestimmten Ansatz für $\Phi_m(q(t), t)$ und damit auch für $q(t+\Delta t)$. Dieser Ansatz wird im Folgenden direkt und ohne Erläuterung dargestellt und erst danach wird erläutert, woher die einzelnen Teile stammen. Hier zeige ich eine leicht vereinfachte Version des Ansatzes:

$$\Phi_m(q(t), t) = \sum_{i=1}^{m} b_i k_i \tag{2.5}$$

und damit:

$$q(t + \Delta t) = q(t) + \Delta t \sum_{i=1}^{m} b_i k_i \tag{2.6}$$

Die k_i sind die sogenannten Stufen, die wie folgt definiert sind:

$$k_i = f(q(t) + a_i k_{i-1} \Delta t, t + a_i \Delta t) \tag{2.7}$$

Ausgeschrieben sehen die Stufen folgendermassen aus:

$$\begin{aligned} k_1 &= f(q(t), t), \\ k_2 &= f(q(t) + a_2 k_1 \Delta t, t + a_2 \Delta t), \\ k_3 &= f(q(t) + a_3 k_2 \Delta t, t + a_3 \Delta t), \\ &\vdots \\ k_m &= f(q(t) + a_m k_{m-1} \Delta t, t + a_m \Delta t) \end{aligned} \tag{2.8}$$

a_i, b_i sind noch zu ermittelnde Konstanten. k_i sind Vektoren. Das sieht komplizierter aus, als es ist. Die k_i sagen einem grundsätzlich einfach den Vektor $f(q(t), t)$ an verschiedenen Punkten. Diese Punkte werden durch das vorherige k, d.h. durch k_{i-1} ermittelt. Das Vorgehen wird in Abbildung 2.2 schematisch visualisiert. Als Erstes wird $k_1 = f(q(t), t)$ ermittelt. Das entspricht der Änderungsrate am Punkt $q(t)$. Dann nimmt man an die Kurve geht in einer geraden Linie weiter, also mit einer Steigung k_1 und ermittelt so den nächsten Punkt zur Zeit $t + a_2 \Delta t$ als $q(t) + a_2 k_1 \Delta t$. Falls die Kurve aber eine Krümmung hat, landet man nicht auf der gesuchten Kurve, sondern man befindet sich auf einer anderen Bahnkurve, die aus anderen Anfangsbedingungen resultiert. Die Hoffnung ist, dass bei genügend kleinem Δt die Abweichung nicht gross ist. An diesem neuen Punkt $(q(t) + a_2 k_1 \Delta t, t + a_2 \Delta t)$ ermittelt man $k_2 = f(q(t) + a_2 k_1 \Delta t, t + a_2 \Delta t)$. Mit k_2 findet man mit der gleichen iterativen Methode k_3, damit k_4 usw. bis man k_m gefunden hat.

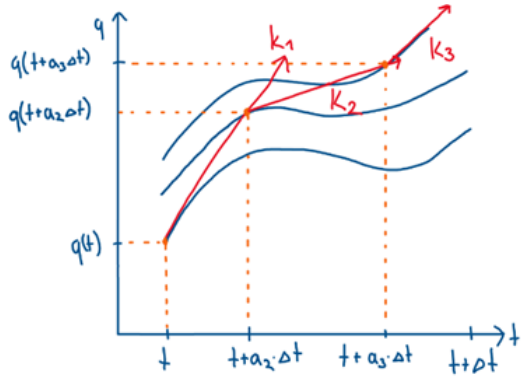


Abbildung 2.2: Vereinfachte Darstellung der ersten drei Schritte eines allgemeinen Runge-Kutta-Verfahrens. q ist im Allgemeinen ein Vektor, hier wird er 1-dimensional dargestellt.

Die Gleichung 2.5 soll die verschiedenen k_i so kombinieren und mit den b_i gewichten, dass ihre Summe möglichst genau der Steigung der Verbindung zwischen $q(t)$ und $q(t + \Delta t)$ entspricht. Es ist ein gewichtetes Mittel. Es stellt sich also die Frage nach der Wahl der Koeffizienten a_i, b_i. Sie sollen, wie in unserem Ziel gesagt, so gewählt werden, dass der lokale Fehler $F_L = \Delta(q(t), t) - \Phi_m(q(t), t)$ möglichst klein bleibt. Um den Fehler zu berechnen, muss Δ und Φ vergleichbar gemacht werden.

2.1.4 Taylorreihen von $\Phi_m(q(t), t)$ und $\Delta(q(t), t)$

Idee 3: Die dritte Idee besteht darin, $\Phi_m(q(t), t)$ und $\Delta(q(t), t)$ durch eine Taylorreihe anzunähern und dann zu vergleichen, um die Koeffizienten a_i, b_i so zu bestimmen, dass der lokale Fehler minimiert oder die Ordnung des Fehlers maximiert wird (das ist identisch, weil $\Delta t < 1$ ist). Um ein numerisches Verfahren mit lokaler Genauigkeit der Ordnung $O(\Delta t^{p+1})$ zu haben, müssen die ersten p Terme die Taylorentwicklungen von $\Phi_m(q(t), t)$ und $\Delta(q(t), t)$ übereinstimmen. Das nennt man ein Runge-Kutta-Verfahren der Ordnung p oder kurz RKp.

Zuerst soll die Taylorreihe von $\Delta(q(t), t)$ (Gleichung 2.3) bestimmt werden. Hierfür wird die Taylorreihe von $q(t + \Delta t)$ bestimmt. Zur Erinnerung: die Taylorreihe von $q(t_v)$ einem Punkt t_0 ist[3]:

$$Tq(t_v) = \sum_{n=0}^{p} \frac{q^{(n)}(t_0)}{n!}(t_v - t_0)^n + O(\Delta t^{p+1})$$

Obwohl $q(t)$ ein Vektor ist, kann die klassische Taylorentwicklung verwendet werden, da jede Operation komponentenweise durchgeführt wird. Im Prinzip wird also jede Komponente einzeln entwickelt. Da man eine unendliche Summe mit $p \to \infty$ nicht immer berechnen kann, nimmt man oft nur die ersten p Terme und vernachlässigt alle Terme höherer Ordnung. Dadurch ergibt sich ein Fehler. Das sogenannte Restglied $O(\Delta t^{p+1})$ repräsentiert diese Terme höherer Ordnung und gibt den Fehler an, der dabei in Kauf genommen wird. Wie genau dieses Restglied angenähert werden kann, ist eine andere Frage. Intuitiv kann aber gesagt werden, dass je grösser die Ordnung des Restglieds, desto kleiner der Fehler. Wenn man nun das Taylorpolynom von

[3]Deiser, 2024.

$q(t_v)$ um den Punkt $t_0 = t$ bis zur dritten Ordnung entwickelt und für die Variabel $t_v = t + \Delta t$ setzt, ergibt das:

$$q(t + \Delta t) = q(t) + \dot{q}(t)\Delta t + \ddot{q}(t)\frac{\Delta t^2}{2} + \dddot{q}(t)\frac{\Delta t^3}{6} + O(\Delta t^4)$$

Damit ist:

$$\Delta(q(t), t) = \frac{q(t + \Delta t) - q(t)}{\Delta t} = \dot{q}(t) + \ddot{q}(t)\frac{\Delta t}{2} + \dddot{q}(t)\frac{\Delta t^2}{6} + O(\Delta t^3)$$

Da wir die Funktion $q(t)$ nicht kennen, wollen wir dafür $f(q(t), t)$ einsetzen:

$$\Delta(q(t), t) = f(q(t), t) + \dot{f}(q(t), t)\frac{\Delta t}{2} + \ddot{f}(q(t), t)\frac{\Delta t^2}{6} + O(\Delta t^3)$$

Beachten Sie hier, dass f eine Funktion von $q(t)$ und t ist. Unter Verwendung der Produkt- und Kettenregel ergibt es für die zeitlichen Ableitungen von f:

$$\dot{f} = f_q f + f_t$$
$$\ddot{f} = f_{qq} f^2 + {f_q}^2 f + f_q f_t + 2 f_{qt} + f_{tt}$$

Beachten Sie die kurze Schreibweise für partielle Ableitung: Beispielsweise ist $f_t = \frac{\partial f}{\partial t}$ die partielle Ableitung jeder Komponente von f nach der Zeit. f_q wäre $\frac{\partial f}{\partial q}$ die Jacobi-Matrix von f nach q[4].
Damit wird $\Delta(q(t), t)$ zu:

$$\Delta(q(t), t) = f + (f_q f + f_t)\frac{\Delta t}{2} + (f_{qq} f^2 + {f_q}^2 f + f_q f_t + 2 f_{qt} + f_{tt})\frac{\Delta t^2}{6} + O(\Delta t^3) \tag{2.9}$$

Nun wollen wir das Taylorpolynom von $\Phi_m(q(t), t)$ aus Gleichung 2.5 entwickeln. Dafür entwickeln wir Gleichung 2.7, indem wir $f(x, y)$ um den Punkt (x_0, y_0) entwickeln und dann $(q(t)+a_i k_{i-1}\Delta t, t+a_i\Delta t)$ einsetzen. Erinnern Sie sich hierfür an die Taylorentwicklung für Gleichungen mit zwei Variablen[5]:

$$\begin{aligned} f(x, y) = {} & f(x_0, y_0) + \Big(f_x(x_0, y_0)(x - x_0) + f_y(x_0, y_0)(y - y_0)\Big) \\ & + \frac{1}{2}\Big(f_{xx}(x_0, y_0)(x - x_0)^2 + 2 f_{xy}(x_0, y_0)(x - x_0)(y - y_0) + f_{yy}(x_0, y_0)(y - y_0)^2\Big) \\ & + O(\Delta t^3) \end{aligned}$$

[4]Studyflix, o.J.
[5]Wagner, o.J.

Auch sie gilt auch für vektorielle Funktionen, da jede Operation komponentenweise durchgeführt wird. Mit $x = q(t) + a_i k_{i-1} \Delta t$, $y = t + a_i \Delta t$ und $x_0 = q(t)$, $y_0 = t$ ergibt es:

$$\begin{aligned} f(q(t) + a_i k_{i-1} \Delta t, t + a_i \Delta t) = f &+ f_q a_i k_{i-1} \Delta t + f_t a_i \Delta t \\ &+ \frac{1}{2}(f_{qq}(a_i k_{i-1} \Delta t)^2 + 2 f_{qt} k_{i-1} {a_i}^2 \Delta t^2 + f_{tt}(a_i \Delta t)^2) \\ &+ O(\Delta t^3) \end{aligned} \tag{2.10}$$

Wir haben $\Phi_m(q(t), t)$ und $\Delta(q(t), t)$ mit Taylorreihen angenähert. Nun kann die Differenz der beiden Gleichungen gebildet werden, um somit den lokalen Fehler zu berechnen (Gleichung 2.4) und dann zu minimieren. Das hängt aber noch davon ab, wie viele Stufen man in Gleichung 2.5 verwenden möchte. Denken Sie hier noch einmal an die Forderung nach Effizienz, die wir in unserem Ziel beschrieben haben. Wir wollen möglichst wenig Rechenaufwand für einen möglichst kleinen Fehler. Die genaue Abschätzung davon ist problemabghängig, doch ich möchte jetzt eine Intuition dafür geben. Je mehr Stufen verwendet werden, desto höher der Aufwand. Wenn ich ein 1-stufiges Verfahren gegenüber einem 4-stufigen Verfahren verwende, ist der Rechenaufwand des 4-stufigen Verfahrens etwa 4-mal höher. Gleichzeitig kann das 4-Stufige Verfahren aber eine deutlich grössere Schrittweite verwenden, da die Ordnung des lokalen Fehlers viel kleiner ist. Wenn das 1-stufige Verfahren eine Schrittweite von 0.0001 verwendet, ergibt das einen lokalen Fehler $O(\Delta t) = O(0.0001)$. Das 4-stufige Verfahren erreicht die gleiche Genauigkeit mit einer Schrittweite von 0.1: $O(\Delta t^4) = O(0.0001)$. Obwohl das 4-stufige Verfahren also pro Schritt 4-mal so viel Rechenaufwand braucht, braucht es 100-mal weniger Schritte um auf die gleiche Genauigkeit zu kommen. Die Potenz "besiegt"das Lineare. Diese Logik funktioniert aber nur bis zu einer bestimmten Grenze, da die Stufenzahl schneller wächst als die Ordnungszahl[6].

2.2 Beispiele Runge-Kutta-Verfahren

Im Folgenden werden die im letzten Abschnitt entwickelten Methoden angewendet, um zwei der einfachsten Runge-Kutta-Verfahren herzuleiten und zu

[6]Eggert, 2007, S. 56.

erklären. Zunächst wird das sogenannte explizite Euler-Verfahren (ein Runge-Kutta-Verfahren 1. Ordnung) vorgestellt, um ein Gefühl für die numerischen Verfahren zu bekommen. Dann wird das Runge-Kutta-Verfahren 2. Ordnung hergeleitet, um zu verstehen, wie die Herleitung funktioniert.

2.2.1 Explizites Eulerverfahren

Um ein Gefühl für numerische Methoden zu bekommen, soll jetzt der einfachste Fall der Runge-Kutta-Verfahren betrachtet werden, nämlich wenn man nur eine Stufe verwendet, d.h. bei Gleichung 2.6 $m = 1$ setzt. Historisch wird die Methode nicht als Runge-Kutta-Verfahren bezeichnet. Man nennt es das explizite Eulerverfahren.

Wenn wir bei Gleichung 2.6 $m = 1$ setzen, ergibt es für Φ:

$$\Phi_1(q(t), t) = b_1 k_1 = b_1 f(q(t), t)$$

Jetzt wollen wir die Koeffizienten so wählen, dass es möglichst effizient ist. Da die Anzahl Stufen und damit der Rechenaufwand für ein gegebenes Δt unveränderlich ist, müssen wir die Koeffizienten so bestimmen, dass der Fehler minimiert oder anders gesagt die Ordnung des Fehlers maximiert wird. Nach Idee 3 verwenden wir also die Taylorentwicklung von Δ und $\Phi_1(q(t), t)$. Da bei k_1 der Faktor $a_1 = 0$ ist (siehe Gleichung 2.8), ist die Taylorentwicklung von k_1 bzw. von $\Phi_1(q(t), t)$ gleichwertig mit der obigen Gleichung (siehe Gleichung 2.10). Ausserdem wird es reichen, die Taylorentwicklung von Δ aus Gleichung 2.9 nur bis zur zweiten Ordnung aufzuschreiben. Der lokale Fehler (siehe Gleichung 2.4) ist also:

$$\begin{aligned} F_L = \Delta - \Phi_1 &= (f + (f_q f + f_t)\frac{\Delta t}{2}) - b_1 f + O(\Delta t^3) \\ &= (1 - b_1) f + (f_q f + f_t)\frac{\Delta t}{2} + O(\Delta t^3) \end{aligned}$$

Dieser Ausdruck wird minimiert, wenn

$$b_1 = 1$$

da mit dieser Koeffizientenwahl der erste Term vollständig wegfällt. Daraus ergibt sich der lokale Fehler:

$$F_L = (f_q f + f_t)\frac{\Delta t}{2} + O(\Delta t^3) = O(h^2)$$

und:

$$\Phi_1 = f(q(t), t)$$

Damit haben wir ein erstes numerisches Verfahren zur Lösung eines Anfangswertproblems.

Jetzt werde ich noch einmal Schritt für Schritt erklären, wie das Verfahren genau funktioniert, bzw. wie das Verfahren für ein Anfangswertproblem mit einem Anfangswert $q(t_0) = q_0$ auf $q(t_f)$ kommt.

Zuerst wird eine Schrittweite Δt gewählt und das Intervall $[q(t_0), q(t_f)]$ in $\frac{t_f - t_0}{\Delta t}$ gleich grosse Teilintervalle unterteilt. Nun werden sukzessiv die Funktionswerte von $q(t + \Delta t))$, angefangen mit dem Anfangswert $q(t_0) = q_0$ und der oben entwickelten Iterationsformel, bestimmt:

$$q(t + \Delta t) = q(t) + \Delta t f(q(t), t)$$

Und anschaulicher geschrieben:

$$t_k = t_0 + k\Delta t$$

$$q_{k+1} = q_k + \Delta t f(q_k, t_k)$$

mit $k \in \mathbb{N}$. Wie in Abbildung 2.3 schematisch dargestellt wird bei dieser Methode davon ausgegangen, dass dir nächste Iterationspunkt der Funktion für ein kleines Δt durch eine Tangente an dem vorangegangenen Iterationspunkt angenähert werden kann. Ein Iterationsschritt verursacht einen Fehler $O(\Delta t^2)$. Das ist im Vergleich zu anderen numerischen Verfahren allerdings nicht besonders gut.

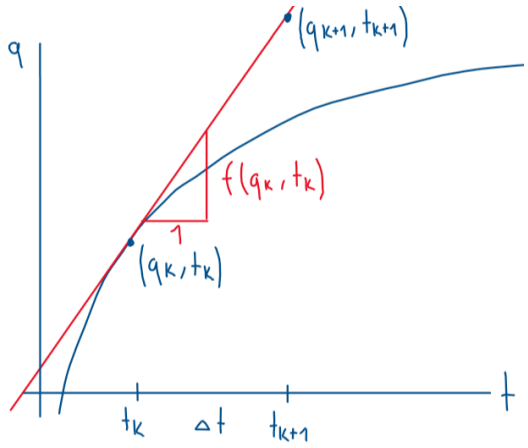


Abbildung 2.3: Schematische Darstellung eines Iterationsschrittes eines expliziten Eulerverfahrens. q kann allgemein auch mehr als nur eine Dimension haben.

2.2.2 Runge-Kutta-Verfahren 2. Ordnung

Das Runge-Kutta-Verfahren 2. Ordnung verwendet zwei Stufen. Wir setzen in Gleichung 2.5 also $m = 2$:

$$\begin{aligned}\Phi_2(q(t), t) &= \sum_{i=1}^{2} b_i k_i = b_1 k_1 + b_2 k_2 \\ &= b_1 f(q(t), t) + b_2 f(q(t) + a_2 k_1 \Delta t, t + a_2 \Delta t)\end{aligned}$$

mit den zwei Stufen:

$$\begin{aligned}k_1 &= f(q(t), t), \\ k_2 &= f(q(t) + a_2 k_1 \Delta t, t + a_2 \Delta t)\end{aligned}$$

Unser Ziel ist es nun, die Koeffizienten a_2, b_1 und b_2 ($a_1 = 0$ gilt nach wie vor) so zu bestimmen, dass der lokale Fehler möglichst klein, bzw. die Fehlerordnung möglichst hoch ist. Hierfür verwenden wir Idee 3 und bestimmen den lokalen Fehler F_L mit der Taylorreihe von Δ und Φ_2. Die Taylorreihe von Δ haben wir bereits als Gleichung 2.9 bestimmt. Die Taylorreihe von Φ_2 ist mit Gleichung 2.10 und der obigen zweistufigen Form des Runge-Kutta-

Verfahrens:

$$\begin{aligned}\Phi_2 = b_1 f + b_2 (f &+ f_q\, a_2 k_1 \Delta t + f_t\, a_2 \Delta t \\ &+ \frac{1}{2}\left(f_{qq}(a_2 k_1 \Delta t)^2 + 2 f_{qt} k_1 {a_2}^2 \Delta t^2 + f_{tt}(a_2 \Delta t)^2\right) + O(\Delta t^3)\end{aligned}$$

Wenn wir bedenken, dass $k_1 = f$ kann man damit den lokalen Fehler F_L aus Gleichung 2.4 folgendermassen angeben:

$$\begin{aligned}F_L = \Delta - \Phi_2 = &\left(f + (f_q f + f_t)\frac{\Delta t}{2} + (f_{qq} f^2 + {f_q}^2 f + f_q f_t + 2 f_{qt} + f_{tt})\frac{\Delta t^2}{6} + O(\Delta t^4)\right) \\ &- b_1 f - b_2 \Bigg(f + f_q f a_2 \Delta t + f_t a_2 \Delta t \\ &+ \frac{1}{2}\Big(f_{qq} f^2 (a_2 \Delta t)^2 + 2 f_{qt} f (a_2 \Delta t)^2 + f_{tt}(a_2 \Delta t)^2\Big) + O(\Delta t^3)\Bigg)\end{aligned}$$

Wenn wir jeweils die Terme der gleichen Ordnung zusammennehmen, ergibt das nach einigem Umformen:

$$\begin{aligned}F_L = \Delta - \Phi_2 = &\Big(1 - (b_1 + b_2)\Big) f \\ &+ \Big(\frac{1}{2} - b_2 a_2\Big)(f_q f + f_t)\Delta t \\ &+ (f_{qq} f^2 (1 - 3{a_2}^2 b_2) + f_{tt}(1 - 3{a_2}^2 b_2) + {f_q}^2 f + f_q f_t + 2 f_{qt} - 6 f_{qt} f {a_2}^2 b_2)\frac{\Delta t^2}{6} \\ &+ O(\Delta t^4)\end{aligned}$$

Hier sieht man, dass die ersten beiden Summanden verschwinden, wenn die folgenden beiden Koeffizientenbedingungen eingehalten werden:

$$b_1 + b_2 = 1$$

$$b_2 a_2 = \frac{1}{2}$$

Der dritte Summand kann mit keiner Änderung der Koeffizienten auf 0 gesetzt werden. Der lokale Fehler wird unter Einhaltung der Koeffizientenbedingungen zu:

$$\begin{aligned} F_L =& (f_{qq}f^2(1-3{a_2}^2b_2) + f_{tt}(1-3{a_2}^2b_2) + {f_q}^2 f + f_q f_t + 2f_{qt} - 6f_{qt}f{a_2}^2b_2)\frac{\Delta t^2}{6} + O(\Delta t^4) \\ =& O(\Delta t^3) \end{aligned}$$

Ein Beispiel für eine Koeffizientenwahl wäre:

$$b_1 = b_2 = \frac{1}{2}$$

$$a_2 = 1$$

Damit würde das Verfahren folgendermassen aussehen:

$$q(t+\Delta t) = q(t) + \frac{\Delta t}{2}(f(q(t),t) + f(q(t) + f(q(t),t)\Delta t, t+\Delta t))$$

Das ist das sogenannte Heun-Verfahren[7].

2.2.3 Runge-Kutta-Verfahren höherer Ordnung und automatische Schrittweitenkontrolle

Für komplexere Systeme benötigt man noch genauere Verfahren. Eine Möglichkeit, dies zu erreichen, ist die Verwendung von Verfahren höherer Ordnung. Die Schritte

1. Wahl der Anzahl Stufen
2. Taylorentwicklung
3. Koeffizientenvergleich

lassen sich analog zu den hier durchgeführten Herleitungen auch auf die Herleitung von Runge-Kutta-Verfahren höherer Ordnung anwenden. Wie sie aber sicher bemerkt haben, war es für die 2. Ordnung schon einigermassen mühsam, weshalb ich es hier nicht tun werde. Wir werden jedoch das DOP853

[7] Nipp, 2006, S. 82.

verwenden. Es ist auch ein Runge-Kutta-Verfahren. Spezifisch hat es die Verfahren 8., 5. und 3. Ordnung eingebettet. Es verwendet dann aber eine automatische Schrittweitenkontrolle. Das Verfahren schätzt also den lokalen Fehler bei jedem Iterationsschritt und basierend auf dieser Fehlerabschätzung wird die Schrittweite automatisch angepasst, um die gewünschte Genauigkeit zu gewährleisten.

2.3 Geeignetes Verfahren für das planare, gekoppelte Federpendel

Nun soll ermittelt werden, welches Verfahren für unser System geeignet ist. Der entscheidende Faktor ist die Effizienz. Es soll also möglichst genau, mit möglichst wenig Rechenzeit sein. Die Rechenzeit lässt sich in Python leicht messen (siehe Anhang). Ein Eindruck der Genauigkeit kann erlangt werden, indem die totale Energie gegen die Zeit geplottet wird. Je genauer das System ist, desto konstanter ist die totale Energie. Wir können also die relative Änderung der totalen Energie, d.h.:

$$\text{Relative Änderung von } E_{tot}(t) = \frac{E_{tot}(t) - E_{tot}(t=0)}{E_{tot}(t=0)}$$

Gegen die Zeit plotten, um einen Eindruck des globalen Fehlers zu bekommen.

2.3.1 Eignung des Runge-Kutta-Verfahrens 2. Ordnung

Als Erstes soll untersucht werden, ob das Runge-Kutta-Verfahren 2. Ordnung, welches wir hergeleitet haben, wirklich nicht geeignet ist. Wie sie in Abbildung 2.4 sehen können, hat es nach 10 Sekunden nur einen relativen Fehler von 10^{-4} Prozent. Als Rechenzeit wurde **2.41 Sekunden** berechnet (aus dem gleichen Code, der zur Berechnung von Abbildung 2.4 gebraucht wurde). Das ist schon relativ gut. Doch nun will ich schauen, ob es auch für Langzeitanalysen geeignet wäre. Wie sie in Abbildung 2.5 sehen können, hat die Simulation nach 1000 Sekunden einen Fehler von 6 Prozent. Das ist zu viel. Deshalb werden jetzt die Integrationsmethoden von `scipy.integrate.solve_ivp` untersucht.

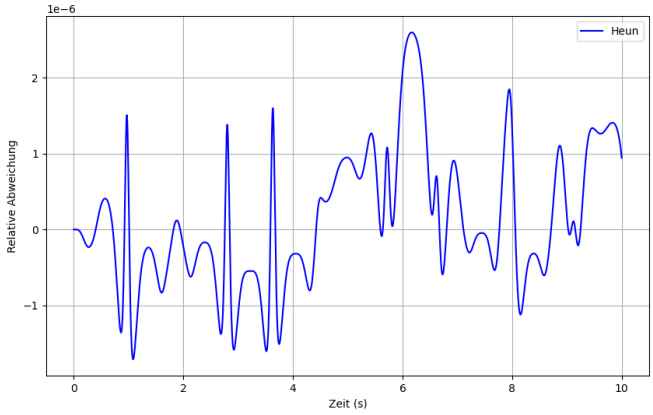


Abbildung 2.4: Relative Änderung der totalen Energie von dem Runge-Kutta-Verfahren 2. Ordnung über 10 Sekunden. Beachten Sie die Achsenskalierung um 10^{-6}

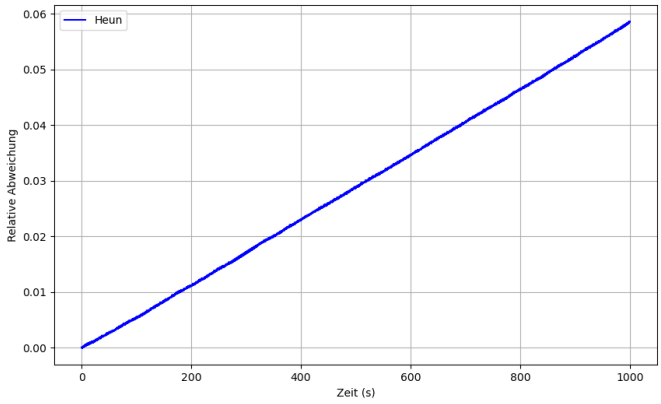


Abbildung 2.5: Relative Änderung der totalen Energie von dem Runge-Kutta-Verfahren 2. Ordnung über 1000 Sekunden.

2.3.2 Eignung der Verfahren von `scipy.integrate.solve_ivp`

In diesem Abschnitt sollen zwei der Verfahren untersucht werden, die `scipy.integrate.solve_ivp` zur Verfügung stellt[8]. Es wurden RK5(4) mit dem DOP853 verglichen. Beides sind Runge-Kutta-Verfahren mit automatischer Schrittweitenkontrolle. Sie sind also Methoden, die den Fehler bei jedem Iterationsschritt abschätzen und je nachdem wie gross er ist, passen sie die Schrittweite des Verfahrens an. Nun wurden noch zwei Varianten, nämlich RK5(4) mit hoher Präzision und DOP853 mit hoher Präzision, angeschaut. Bei diesen wurde die relative Fehlertoleranz (`rtol`) und die absolute Fehlertoleranz (`atol`) verkleinert. In der Quelle[9] kann das genauer nachgelesen werden. Je kleiner man `rtol` und `atol`, desto genauer wird die Lösung. Gleichzeitig erhöht sich aber dadurch auch der Rechenaufwand. In meinem Fall habe ich: `rtol` $= 1e-10$, `atol` $= 1e-10$ gesetzt, da ich eine hohe Genauigkeit möchte. Hier die Ergebnisse:

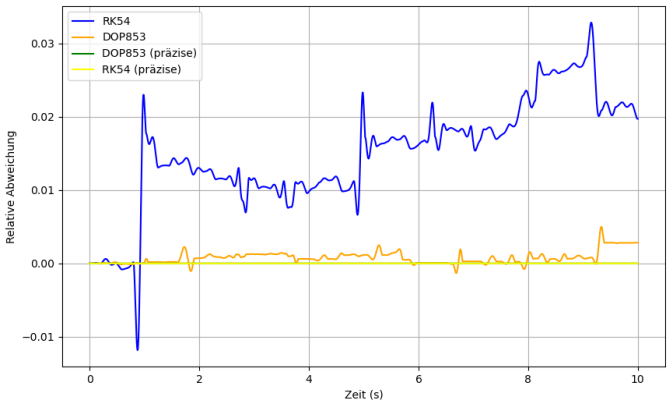


Abbildung 2.6: Relative Änderung der totalen Energie für die Verfahren RK5(4) und DOP853 und ihre "präzisen"Varianten mit `rtol` $= 1e-10$, `atol` $= 1e-10$

Da in Abbildung 2.6 die präzisen Varianten nicht erkennbar sind, wurde im Folgenden nur die präzisen Varianten geplottet. Es ist aber der gleiche Graph wie aus Abbildung 2.6:

[8]SciPy, o.J.
[9]SciPy, o.J.

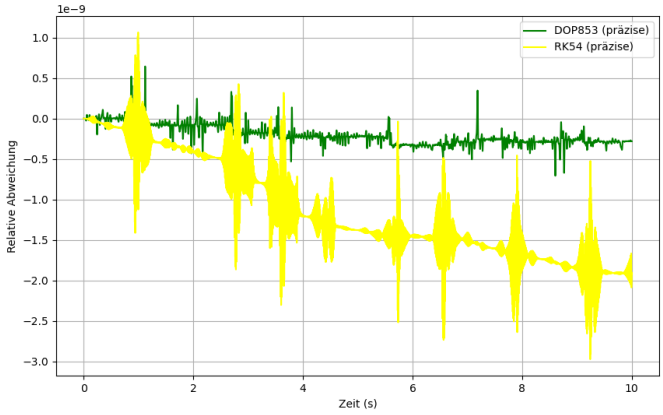


Abbildung 2.7: Die präzisen Varianten aus Abbildung 2.6, beachten sie die Skalierung der y-Achse um 10^{-9}

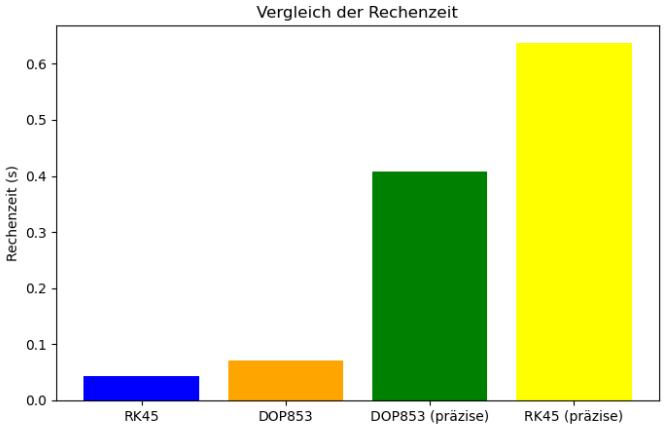


Abbildung 2.8: Rechenzeit für die Berechnung mit den verschiedenen Methoden aus Abbildung 2.6

Wie sie in Abbildung 2.6 sehen können, weicht RK5(4) nach nur 10 Sekunden um mehr als 2 Prozent ab. DOP853 ist da schon deutlich besser. Es weicht um etwa $0.2 - 0.3$ Prozent ab. Beide sind aber erstaunlicherweise deutlich ungenauer als das Runge-Kutta-Verfahren 2. Ordnung. Die präzisen Varianten sind noch einmal deutlich genauer (Abbildung 2.7). Sie bewegen

sich in der Grössenordnung von 10^{-7} Prozent. RK5(4) (präzise) weicht nach 10 Sekunden um etwa $2 \cdot 10^{-7}$ Prozent ab. Aber am besten ist die präzise Variante von DOP853. Es hat nach 10 Sekunden einen relativen Fehler in der Grössenordnung von $2 \cdot 10^{-8}$ bis $3 \cdot 10^{-8}$ Prozent, ist also um eine Grössenordnung 10^4 genauer als das Runge-Kutta-Verfahren 2. Ordnung und das, obwohl die Rechenzeit so viel kleiner ist (Abbildung 2.8). Die höhere Präzision der präzisen Varianten gegenüber ihrer nicht präzisen Varianten spiegelt sich auch in der Rechenzeit wider. Durch das Hochschrauben der Fehlertoleranzen dauern die Verfahren deutlich länger. Ausserdem sind alle vier Varianten deutlich weniger rechenintensiv als Runge-Kutta-Verfahren 2. Ordnung. Überraschenderweise dauert die RK5(4) (präzise) aber noch einmal deutlich länger als DOP853 (präzise). Das liegt daran, dass DOP853 eine höhere Ordnung besitzt (8 gegenüber 5) und dadurch weniger Schritte für die gleiche Genauigkeit benötigt. Ausserdem nutzt DOP853 eine effizientere Fehlerkontrolle und kann die Schrittweite besser anpassen. Insbesondere bei hohen Präzisionsanforderungen führt dies dazu, dass RK45 deutlich mehr Schritte ausführen muss, was die Rechenzeit erhöht. Ausserdem können wir die präzisen Methoden über eine längere Zeit anschauen:

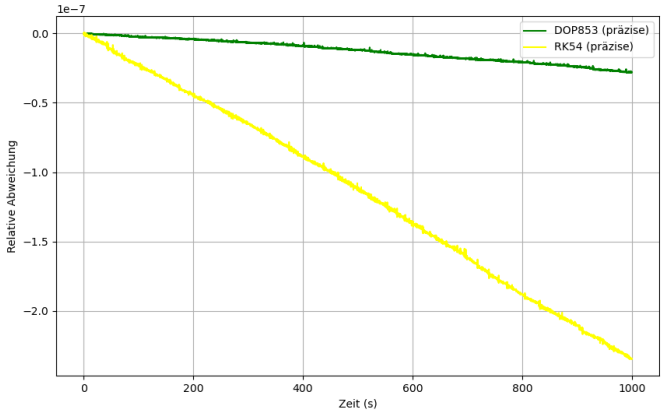


Abbildung 2.9: Die präzisen Varianten aus Abbildung 2.6 über 1000 Sekunden beachten sie die Skalierung der y-Achse um 10^{-7}

Auch für Langzeitanalysen ist also DOP853 (präzise) deutlich genauer als RK5(4) (präzise) und das Runge-Kutta-Verfahren 2. Ordnung. DOP853 (präzise) vereint also die hohe Genauigkeit mit relativ kurzer Rechenzeit. Es ist

also am effizientesten, weshalb im Folgenden dieses Runge-Kutta-Verfahren verwendet wird.

Kapitel 3

Visualisierung und Analyse dynamischer Systeme

Wir können jetzt also die Bewegungsgleichungen aufstellen und diese mit den numerischen Methoden lösen. Trotzdem gibt es immer noch zwei Probleme. Einerseits die unvermeidbare Ungenauigkeit der Simulation durch Messfehler und Fehler der numerischen Methoden. Wann das ein Problem ist, wird im Kapitel 'Deterministisches Chaos und Ljapunow-Exponenten' genauer angeschaut. Das andere Problem liegt in der vollständigen Darstellung komplexer Systeme. Im Allgemeinen lassen sich die Bewegungsgleichungen eines Systems mit f Freiheitsgraden, wie wir es mit unserem System auch schon gemacht haben (Gleichung 2.1), folgendermassen kompakt schreiben:

$$\frac{d}{dt}\vec{x}(t) = \vec{F}(\vec{x}(t), t, \lambda_k)$$

Wobei $\vec{x} = (x_1, ..., x_{2f})^T$ der Zustandsvektor ist. Sein Verlauf über die Zeit, die sogenannte Trajektorie, ist die Lösung der obigen Gleichung. Der Zustand eines Systems ist zu jedem Zeitpunkt eindeutig durch einen solchen Punkt $\vec{x}$ im sogenannten Phasenraum beschrieben. Man braucht f generalisierte Koordinaten für die Position und f weitere Koordinaten, um zu beschreiben, wie sich die Koordinaten verändern, was dem Zustandsvektor und seinen Komponenten entspricht. Hinter x_i können sich verschiedene Grössen verbergen, z.B. in dem Fall des planaren Federpendels Winkel und ihre Ableitungen. λ_k steht für die k verschiedenen Parameter. Stellen wir uns nun vor, wir wollen alle Informationen über das Verhalten in einem Diagramm darstellen. Wenn wir bestimmte Parameter auswählen, dann braucht man $2f$ Achsen für den

Zustandsvektor und eine Zeitachse. Es wäre ein Raum mit $2f + 1$ Dimensionen. Zusätzlich verhält sich ein System für verschiedene Kombinationen von Parametern teilweise sehr unterschiedlich. Um die vollständige Vielfalt eines Systems darzustellen, bräuchte man noch einmal k Koordinatenachsen für die Parameter. Offensichtlich ist das bei den meisten Systemen für den Menschen eine Überforderung, da dieser Raum mehr als drei Dimensionen hätte. Dazu kommt noch, dass das Verhalten von komplexeren Systemen oftmals sehr unstetig (man würde vielleicht chaotisch sagen) und nur schwer fassbar ist. Die Methoden zur Darstellung und Analyse von dynamischen Systemen drehen sich alle darum, die Informationen dieses $(2f + 1 + k)$-dimensionalen Raums anschaulich zu machen und gewisse Informationen zu isolieren, also sichtbarer zu machen. Im Folgenden werden wir einige dieser Methoden kennenlernen.

3.1 Methoden zur Beschreibung dynamischer Systeme

Eine offensichtliche Methode zur Darstellung des Systems wird dadurch erreicht, dass man den Phasenraum auf zwei oder drei Dimensionen projiziert oder Komponenten davon gegen die Zeit plottet. Oft werden sogenannte Phasenraumdiagramme gemacht, bei welchen eine Koordinate gegen ihre zeitliche Ableitung geplottet wird[1]. Das wird später auch gebraucht um das planare, gekoppelte Federpendel zu beschreiben. Nun sollen einige alternative Methoden zur Beschreibung dynamischer Systeme erläutert werden.

3.1.1 Poincaré-Schnitte

Der Poincaré-Schnitt ist eine Methode zur Darstellung dynamischer Systeme. Die Idee besteht darin, die Zeitabhängigkeit zu diskreditieren. Hierfür wird eine $N - 1$ dimensionale Hyperfläche in den Phasenraum gelegt. Dann wird jedes Mal ein Punkt auf der Hyperfläche markiert, an dem die Trajektorie von $\vec{x}(t)$ die Hyperfläche schneidet[2]. Wie Sie in Abbildung 3.1 sehen können, entsteht dadurch eine Reihe an Punkten $\vec{x}_1$, $\vec{x}_2$, $\vec{x}_3$. Man kann versuchen einen Zusammenhang in der Abfolge der einzelnen Schnitte zu erkennen oder

[1] Russell, o.J.
[2] Greiner, 2008, S. 447.

man zeichnet einfach die Hyperfläche mit allen Schnittpunkten und versucht global Muster zu erkennen. Später wird zweiteres gemacht. Prinzipiell kann der Schnitt auch weniger als $N-1$ Dimensionen haben, nur wird es dann immer unwahrscheinlicher, dass die Trajektorie den Schnitt auch schneidet.

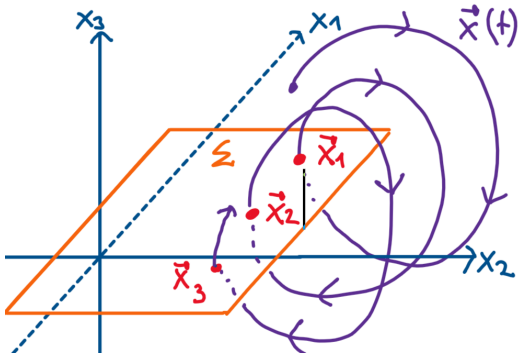


Abbildung 3.1: Visualisierung eines Poincaré-Schnitts in 3 Dimensionen (x_1, x_2, x_3) mit einer Trajektorie $\vec{x}(t)$, einer Schnittfläche $\sum$ welche mit der $\vec{x}_1$-$\vec{x}_2$-Ebene identisch ist und drei Durchstosspunkten $\vec{x}_1$, $\vec{x}_2$, $\vec{x}_3$

3.1.2 Deterministisches Chaos und Ljapunow-Exponenten

Deterministisches Chaos scheint erst ein Widerspruch zu sein. Gemeint ist, dass obwohl das System eindeutig durch die Bewegungsgleichung beschrieben ist, scheinbar zufälliges Verhalten auftritt. Charakteristisch für solche Systeme ist, dass kleine Änderungen der Anfangsbedingungen grosse Auswirkungen auf den Verhalten des Systems haben. Wenn sich aber ein System aufgrund kleiner Änderungen der Anfangsbedingen längerfristig stark unterscheidet, dann führt das sowohl auf experimenteller als auch auf numerischer Ebene zu Problemen. Auf experimenteller Ebene ist das Problem, dass man die Anfangsbedingungen für ein System nicht mit unbegrenzter Genauigkeit bestimmen kann, da es immer einen Messfehler geben wird. Man kann nur sagen, dass die Anfangsbedingungen in einem bestimmten Bereich liegen. Also z.B. der Ball liegt in einer Höhe von 10(1) Meter. Bei chaotischen Systemen kommt es aber sehr darauf an, was genau die Anfangsbedingungen sind. Wenn man das System mit den unklaren Anfangsbedingungen versucht zu simulieren, ergibt sich ein zweites Problem. Bei numerischen Verfahren gibt es zwangsläufig bei jedem Iterationsschritt einen gewissen lokalen Fehler. Nun addieren sich diese Fehler aber nicht mehr linear zu einem globalen Fehler,

sondern jeder lokale Fehler wächst, da zwei unterschiedliche Trajektorien im Phasenraum sich in chaotischen Systemen auseinander bewegen. Dadurch werden die numerischen Vorhersagen für chaotische Systeme schneller unvertrauenswürdig. Das Problem chaotischer Systeme besteht also darin, dass Messfehler und numerische Verfahrensfehler nicht vermieden werden können und diese Fehler a priori schnell zu falschen Ergebnissen führen.

Als Beispiel kann man unser System verwenden. Wie Sie in Abbildung 3.2 sehen können, verursacht ein Unterschied in den Anfangsbedingungen um 0.0001 bereits nach spätestens 14 Sekunden vollkommen unterschiedliches Verhalten von γ. Es ist eindeutig chaotisch. Nun kann man sich vorstellen, dass verschiedene Systeme unterschiedlich schnell, unterschiedlich stark auf Änderungen der Anfangsbedingungen reagieren. Das kann mit dem sogenannten Ljapunow-Exponenten bestimmt werden.

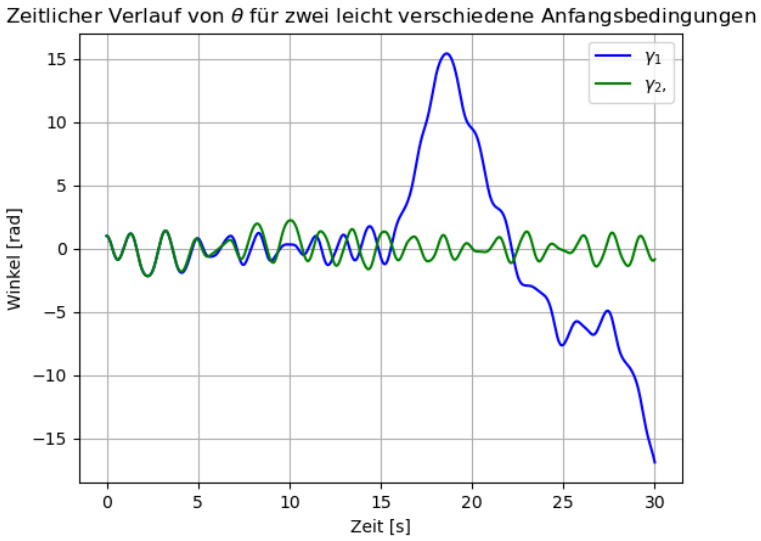


Abbildung 3.2: Verläufe für γ. γ_1 und γ_2 unterscheiden sich nur in der Anfangsbedingung für $\gamma(t = 0)$, nähmlich $\gamma_1(t = 0) = 1$ und $\gamma_2(t = 0) = 1.0001$.

Ljapunow-Exponent

Wir wollen untersuchen, wie empfindlich ein System auf eine Änderung der Anfangsbedingungen reagiert. Wir können dafür zwei Bahnen $\vec{x}_1$ und $\vec{x}_2$ berechnen. Diese zwei Bahnen unterscheiden sich in ihren Anfangsbedingungen

um eine kleine Verschiebung $\vec{\epsilon}(0) = \vec{x}_2(0) - \vec{x}_1(0)$. Wir wollen untersuchen, wie sich die Verschiebung $\vec{\epsilon}(t)$ zwischen den beiden zwei Bahnen mit dem Zusammenhang:

$$\vec{x}_2(t) - \vec{x}_1(t) = \vec{\epsilon}(t)$$

entwickelt. Wird die Norm $\|\vec{\epsilon}(t)\|$ mit der Zeit grösser, so ist das System chaotisch. Wenn wir $\vec{\epsilon}(t)$ linear approximieren, dann gilt im Bereich der linearen Approximation der folgende Zusammenhang zwischen der Ausgangsverschiebung $\vec{\epsilon}(0)$ und $\vec{\epsilon}(t)$:

$$\|\vec{\epsilon}(t)\| \approx \|\vec{\epsilon}(0)\| e^{\lambda t},$$

Wobei λ der sogenannte Ljapunow-Exponent ist[3]. Je nach Richtung der Anfangsverschiebung gibt es verschiedene Ljapunow-Exponenten, weshalb man von einem Spektrum an Ljapunow-Exponenten spricht. Den grössten nennt man maximalen Ljapunow-Exponent (MLE), und er ist folgendermassen gegeben[4]:

$$\lambda_{\max} = \lim_{t \to \infty} \lim_{\|\vec{\epsilon}(0)\| \to 0} \frac{1}{t} \ln \frac{\|\vec{\epsilon}(t)\|}{\|\vec{\epsilon}(0)\|}.$$

Es ist einfach die obere Gleichung nach λ aufgelöst. Dann lässt man t gegen unendlich gehen, da dann der MLE dominiert. Dafür muss aber $\|\vec{\epsilon}(0)\|$ gegen null gehen, da man sich nur dann über den gesamten Zeitraum im Bereich der linearen Approximation befindet.

Numerische Berechnung des MLE

Es gibt mehrere Gründe, weshalb die obige Definition für numerische Kalkulationen nicht geeignet ist. Es ist nicht sinnvoll entsprechend $\lim_{t\to\infty}$, $\lim_{\|\vec{\epsilon}(0)\|\to 0}$ sehr grosse t und kleine $\|\vec{\epsilon}(0)\|$ auszuwählen. Bei zu kleinem $\|\vec{\epsilon}(0)\|$ und zu grossem t ist der Fehler schnell grösser als die tatsächliche Separation. Ausserdem weiss man nicht genau, wie lang der Bereich der linearen Approximation ist, da $\|\vec{\epsilon}(0)\|$ nicht tatsächlich der Limes gegen null ist. Zudem passiert es bei chaotischen Systemen oft, dass die beiden Trajektorien sehr schnell voneinander entfernen[5]. Um also stabile und exakte Ergebnisse zu erhalten, muss der Ansatz leicht angepasst werden.

Man bestimmt wieder eine Startverschiebung $\|\vec{\epsilon}(0)\|$. Nun berechnet man eine Iteration der beiden Trajektorien $\vec{x}_1$ und $\vec{x}_2$ und berechnet damit $\|\vec{\epsilon}(t_1)\|$.

[3] Guan, 2014, S. 1.
[4] Guan, 2014, S. 2.
[5] Greiner, 2008, S. 464.

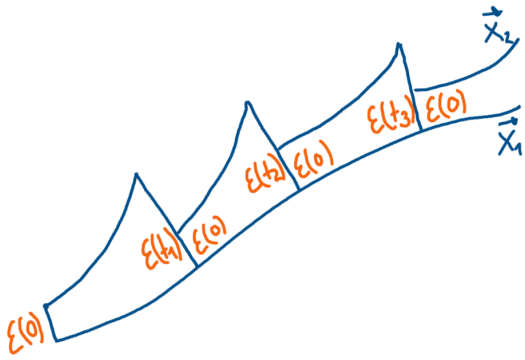


Abbildung 3.3: Bei der numerischen Berechnung des MLE nimmt man bei jedem Iterationsschritt eine Renormierung vor.

Damit kann ein erster MLE berechnet werden. Nun wird die Verschiebung normalisiert, d.h. es wird wieder von Startverschiebung ausgegangen (man kann es auch mit einer Konstante $c \ll 0$ multiplizieren, um die Richtung der Verschiebung zu erhalten). Dann wird mit dem gleichen Verfahren ein MLE für die alle Iterationsschritte berechnet. In Abbildung 3.3 sieht man sehr anschaulich, wie das Verfahren funktioniert. Am Ende berechnet man das Mittel aus den verschiedenen Ljapunow-Exponenten der Iterationsschritte, um eine gute und stabile Schätzung des MLE's zu erhalten.[6]

3.2 Analyse des planaren, gekoppelten Federpendels für unterschiedliche Anfangsbedingungen

In diesem Abschnitt werden die entwickelten Methoden dafür verwendet das planare, gekoppelte Federpendel zu untersuchen. Es wird untersucht, wie verschiedene Anfangsbedingungen das System beeinflussen. Hierfür wurden bestimmte Parameter ausgewählt ($M = 3, l_0 = 1, k = 20, m_1 = 2, m_2 = 2, r_1 = 1, r_2 = 1$) und auch alle Anfangsbedingungen bis auf $\gamma(0)$ bestimmt ($\theta(0) = \pi, \dot{\gamma}(0) = 0, \dot{\theta}(0) = 0$) . Spezifisch wurde untersucht, wie sich das System für $0 < \gamma(0) < \pi$ verhält. Nun stellt sich die Frage nach einer übersichtlichen Darstellung. Wegen der Symmetrie des Systems wird es reichen,

[6] Sprott, 2015.

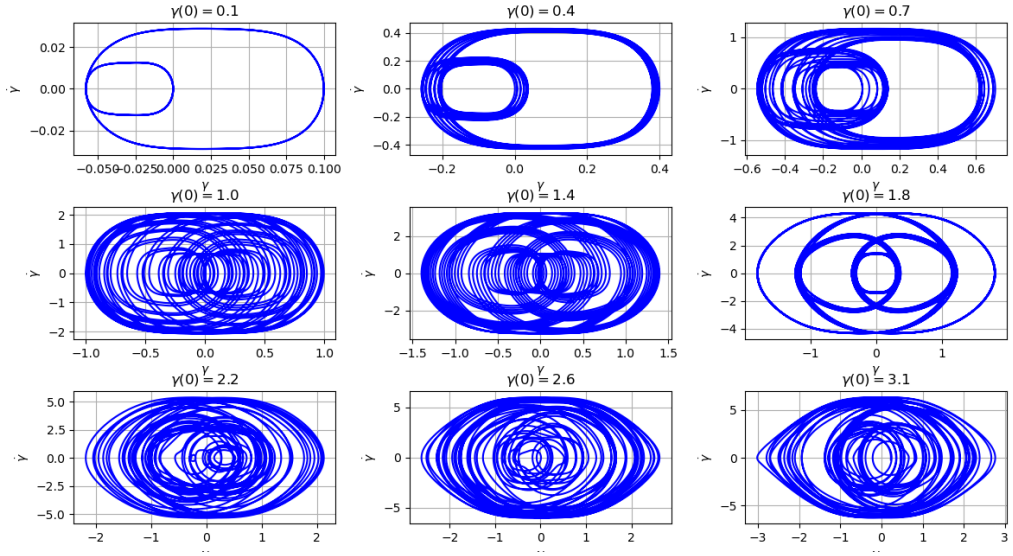


Abbildung 3.4: Phasendiagramme für verschiedene Anfangsbedingungen $\gamma(0)$. Auf der horizontalen Achse ist γ und auf der vertikalen $\dot{\gamma}$ aufgetragen

einen der Winkel anzuschauen, um einige Charakteristika des Systems herauszufinden. Man könnte die Zeitabhängigkeit von $\gamma(t)$ oder die Trajektorie im dreidimensionalen Phasenraum von γ, $\dot{\gamma}$, t darstellen. Eine übersichtliche Darstellung der Trajektorie kann in diesem Fall durch ein Phasenraumdiagramm von γ-$\dot{\gamma}$ erreicht werden, da dort auch periodisches Langzeitverhalten kompakt dargestellt werden kann. In Abbildung 3.4 sehen Sie, wie dieses Phasenraumdiagramm für verschiedene Anfangsbedingungen $\gamma(0)$ aussieht.

Das System wurde dafür über 100 Sekunden berechnet. Die ersten 10 Sekunden wurden aber weggelassen, da uns nur das Langzeitverhalten und nicht der Einschwingvorgang interessiert. Die Phasendiagramme sind jeweils auf einen gewissen Bereich eingeschränkt. Das liegt primär an der Konstanz der totalen Energie. Wegen der speziellen Anfangsbedingungen kann sich das System auch nicht überschlagen, da die dafür gebrauchte potentielle Energie zu gross wäre. Das System ist also in Richtung beider Achsen eingeschränkt. Man sieht, dass das System bei $\gamma(0) = 0.1$ periodisch ist, da die Trajektorie geschlossen ist. Es hat einen grossen und einen kleinen Zyklus, auf welchem die maximale Auslenkung und Winkelgeschwindigkeit kleiner sind. Während das eine Pendel auf dem grossen Zyklus ist, ist das andere Pendel auf dem

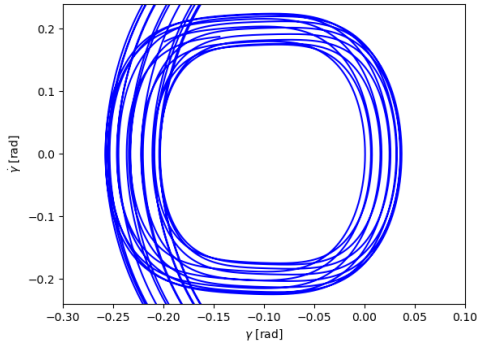


Abbildung 3.5: Ausschnitt aus dem Phasendiagramm für $\gamma(0) = 0.4$.

kleinen Zyklus. Auch das kann mit Energieerhaltung begründet werden. Mit grösserem $\gamma(0) = 0.4$ tritt eine Periodenvervielfachung auf, d.h. das System hat jetzt mehre Schwingungszüge, auf denen sich das System bewegt und wechselt. Das Wechseln sieht man in dem Ausschnitt (Abbildung 3.5) aus dem Phasendiagramm sehr schön. Für $\gamma(0) = 0.7$ gab es noch mehr Periodenvervielfachung. Ausserdem scheint es einen zweiten markanten grossen Zyklus zu geben. Für $\gamma(0) = 1$ ist durch die vielen Periodenvervielfachungen fast der ganze Raum aufgefüllt. Nun passiert etwas sehr Interessantes. Für $\gamma(0) = 1.4$ sind wieder gewisse geschlossene Bahnen erkennbar, denen das System aber nicht exakt folgt. Man nennt ein solches Verhalten quasiperiodisch[7]. Und für $\gamma(0) = 1.8$ ist das System wieder periodisch, da man geschlossene Bahnen hat, denen das System relativ genau folgt. Für grössere $\gamma(0)$ ist keine klare Struktur mehr erkennbar. Das System schwankt flüssig zwischen verschieden grossen Zyklen und füllt so den Phasenraum aus. Sie sind chaotisch. Wie chaotisch, kann mit dem Ljapunow-Exponenten untersucht werden. Mit dem maximalen Ljapunow-Exponenten berechnet für die verschiedenen Anfangsbedingungen, lässt sich dieser Übergang ins Chaos zeigen (Abbildung 3.6).

[7]Spektrum, 2000a.

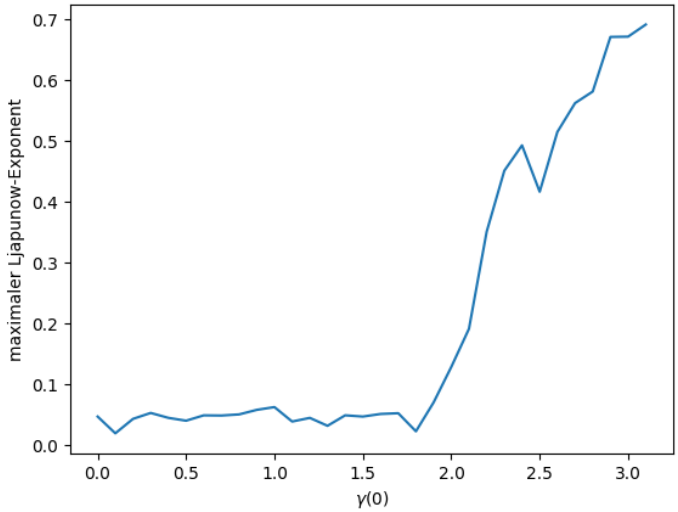


Abbildung 3.6: Maximaler Lyapunov Exponent für verschiedene Anfangsbedingungen $\gamma(0)$.

Für $0 < \gamma(0) < 1.8$ ist der maximale Ljapunow-Exponent relativ konstant niedrig. Nur bei den periodischen Lösungen $\gamma(0) = 0.1, 1.8$ ist er noch etwas kleiner. Doch für $1.8 < \gamma(0)$ schnellt der maximale Ljapunow-Exponent nach oben. Dort wird das System richtig chaotisch.

Das Gleiche lässt sich auch mit Poincaré-Schnitten visualisieren. Als Schnittbedingung wurde $\gamma = 0$ gewählt. Als Schnittebene wurde die θ-$\dot{\theta}$-Ebende gewählt. Es sollten nur die Schnittpunkte aufgetragen werden, bei denen die Trajektorie von derselben Seite der Ebene kommen, weshalb eine zweite Forderung $\dot{\gamma} > 0$ hinzugefügt wurde[8]. Für periodisches Verhalten wären einzelne Punkte erwartet, für quasiperiodisches Verhalten eine Kurve die kein Fläche füllt und für chaotisches Verhalten sollte es ganze Flächen auffüllen[9]. In der Abbildung 3.7 sieht man wie erwartet (fast) periodisches Verhalten. In Abbildung 3.8 sieht man auch wie erwartet eine geschlossene Kurve die dem quasiperiodischen Verhalten entspricht. In beiden Abbildungen sind es aber nicht exakt Punkte oder Linien. Das deutet darauf hin, dass nicht die exakten periodischen/quasiperiodischen Bereiche getroffen wurden.

[8]Richter und Scholz, o.J. S. 11.
[9]Stein, o.J.

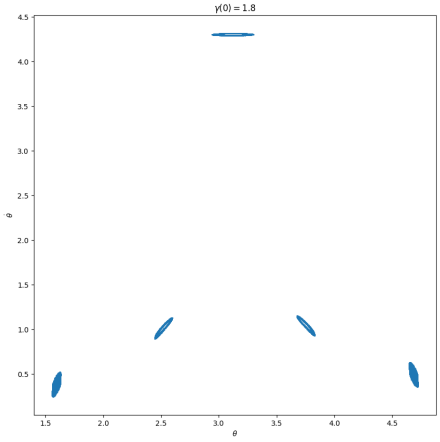


Abbildung 3.7: Poincare-Schnitt für $\gamma(0) = 1.8$, x-Achse: θ, y-Achse: $\dot{\theta}$

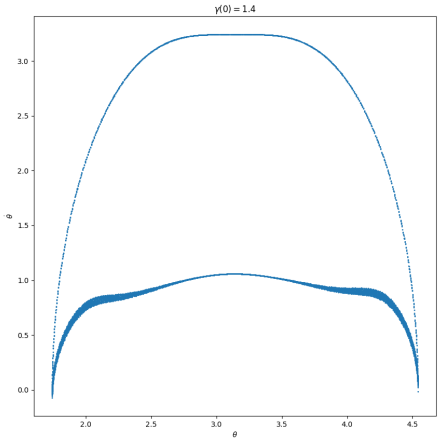


Abbildung 3.8: Poincaré-Schnitt für $\gamma(0) = 1.4$, x-Achse: θ, y-Achse: $\dot{\theta}$

In Abbildung 3.9 sieht man das chaotische Regime. Die Schnittpunkte füllen den homogen fast den ganzen Raum auf.

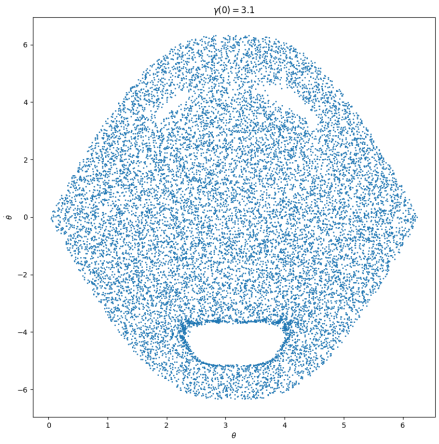


Abbildung 3.9: Poincare-Schnitt für $\gamma(0) = 3.1$, x-Achse: θ, y-Achse: $\dot{\theta}$

Noch besser sieht man den Übergang in einem sogenannten Bifurkationsdiagramm. Hierfür wurde das System auch wieder durchgerechnet für verschiedene $\gamma(0)$. Da sich $\dot{\gamma}$ und $\dot{\theta}$ immer um null bewegen, ist zu erwarten, dass sie regelmässig den Wert Null annehmen. In dem Bifurkationsdiagramm wurde angeschaut, welchen Wert $\dot{\gamma}$ annimmt, wenn $\dot{\theta} = 0$ ist. Es ist also eine Art Poincaré-Schnitt. Wenn das System periodisch ist, dann wäre erwartet, dass $\dot{\gamma}$ immer die gleichen bestimmte Werte annimmt, wenn $\dot{\theta} = 0$. In Abbildung 3.10 sieht man ein solches Diagramm. Hier sieht man sehr schön, wie die Trajektorien durch die verschiedenen Anfangsbedingungen zustande kommen. Für kleine Werte von $\gamma(0)$ ist das System periodisch. Dann kommt es zu den ersten Periodenvervielfachungen, was den Verästelungen im Diagramm entspricht, da dann $\dot{\gamma}$ verschiedene Werte hat, wenn $\dot{\theta}(0) = 0$. Für $\gamma(0) \approx 0.7$ konzentriert es sich wieder etwas auf zwei Regionen, weshalb man in den Phasendiagrammen die beiden öfter befahrenen Zyklen hat. Es konzentriert sich aber nicht auf zwei Punkte, weshalb kein periodisches Verhalten auftritt. Dann kommen sehr viele Verästelungen, die sich dann überlappen, was zu den vielen Periodenvervielfachungen führt. Bei $\gamma(0) \approx 1.4$ ist es teilweise wieder konzentrierter, weshalb dort das quasiperiodische Verhalten auftritt. Dann verästelt es sich aber wieder und somit hängt es nicht mit dem periodischen Verhalten bei $\gamma(0) \approx 1.8$ zusammen. Dort konzentriert es sich wie bei dem Poincaré-Schnitt noch einmal auf fünf Punkte. Für $1.8 < \gamma(0)$

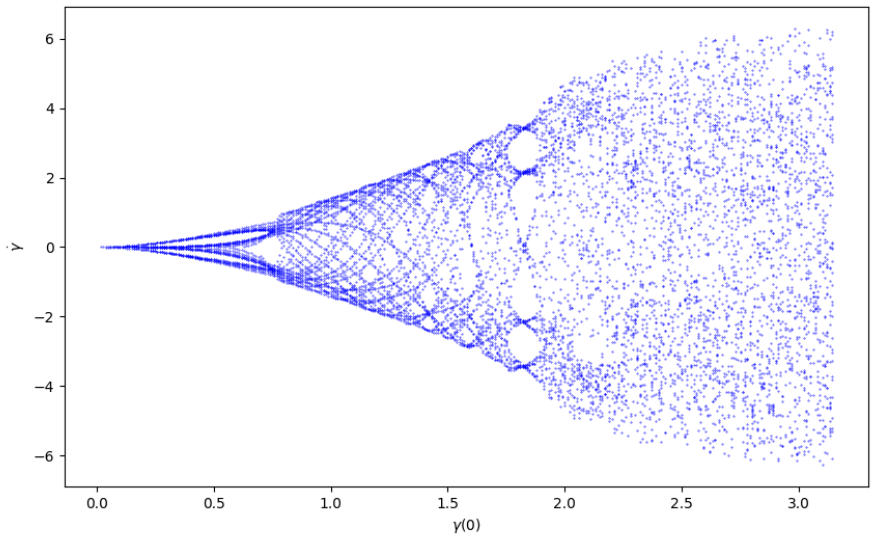


Abbildung 3.10: Bifurkationsdiagramm für verschiedene $\gamma(0)$

verteilen sich die Punkte fast homogen, was dem chaotischen Bereich entspricht. Man erkennt also klar, dass das Beispiel ein Fall von Chaos durch Periodenverdopplung ist[10].

Eine Analyse, wie sie hier gemacht wurde, könnte jetzt auch für die anderen Parameter gemacht werden, um ein ganzheitlicheres Bild des Systems zu bekommen.

[10]Schuster, 2000.

Schlusswort

Um zu verstehen, wie ein System wie das planare, gekoppelte Federpendel beschrieben werden kann, mussten einige zentrale Konzepte verstanden werden. Sie können unter der Frage:

Wie können komplexe, dynamische Systeme beschrieben werden?

zusammengefasst werden. Die Frage wird in der Arbeit zwar minimalistisch, aber in den wichtigsten Punkten trotzdem vollständig beantwortet.

Dazu wurde im ersten Kapitel ein praktischer Weg zur Bestimmung der Bewegungsgleichung von holonomen Systemen entwickelt. Unter Zuhilfenahme der holonomen Zwangsbedingungen können die generalisierten Koordinaten bestimmt werden. Mit diesen muss die Lagrangefunktion:

$$L = T - V$$

ausgedrückt werden. Für diese Funktion wurde über das d'Alembertsche Prinzip die Lagrange-Gleichung hergeleitet:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_\alpha}\right) - \frac{\partial L}{\partial q_\alpha} = 0 \quad \text{für jedes } \alpha$$

Diese wurde am Ende von Kapitel 1 auf das planare, gekoppelte Federpendel angewendet.

Von den Lagrange-Gleichungen erhält man einen Satz an Bewegungsgleichungen. Wenn sie mit einem Anfangswert spezifiziert wird, dann erhält man ein Anfangswertproblem:

$$\begin{aligned} \frac{d}{dt}q(t) &= f(q(t), t), \\ q(t_0) &= q_0 \end{aligned}$$

Dieses ist für viele Systeme aber analytisch nicht lösbar. Deshalb wurden numerische Verfahren eingeführt. Es wurden die grundlegenden Ideen eines Runge-Kutta-Verfahrens vorgestellt. Diese sind die Iteration, der Runge-Kutta-Ansatz und die Koeffizientenbestimmung durch Vergleich der Taylorentwicklung. Man unterteilt in einem solchen Verfahren das System in Teilschritte. Dann wird versucht, diesen Teilschritt durch ein gewichtetes Mittel verschiedener $f(q(t), t)$ zu approximieren. Das ist der Runge-Kutta-Ansatz. Um die Koeffizienten, die Komponenten und die Gewichtung der Komponenten bestimmen, optimal zu wählen, wird die Differenz der Taylorentwicklung einer perfekten Lösung mit der des Runge-Kutta-Ansatzes verglichen und dann minimiert. Nun stellte sich die Frage nach der Darstellung dieser numerisch erlangten Daten. Wegen der vielen Dimensionen, die im Spiel sind, und der Komplexität (Chaos) des Verhaltens ist das ein Problem. Mit Phasenraumdiagrammen, Poincaré-Schnitten und dem maximalen Ljapunow-Exponenten wurden drei Wege vorgestellt, damit umzugehen.

Die theoretischen Teile wurden auf das planare, gekoppelte Federpendel angewendet. Es wurden die Bewegungsgleichungen und ein passendes numerisches Verfahren ermittelt. Im dritten Kapitel wurde der Übergang ins Chaos für verschiedene Anfangsbedingungen gezeigt und erklärt.

Es wurde also die zentralsten Punkte vom d'Alembertschen Prinzip bis zur Beschreibung von Chaos theoretisch erläutert und am gekoppelten planaren Federpendel praktisch angewendet.

Literatur

Cryer, P. D. C. W. (2008). *Herleitung des Runge-Kutta-Verfahrens*. Verfügbar 15. Dezember 2024 unter https://www.uni-muenster.de/AMM/num/Vorlesungen/cryer/Numerik2_SS06/RungeKutta.pdf

Deiser, O. (2024). *Einführung in die Mathematik*. Verfügbar 15. Dezember 2024 unter https://www.aleph1.info/?call=Puc&permalink=ema11_2_3_Z3

Eggert, R. (2007). *Implizite Runge-Kutta-Verfahren und ihre Anwendung auf Steuerungsprobleme* [Magisterarb., Universität Bayreuth].

Greiner, W. (2008). *Klassische Mechanik II*. Harri Deutsch.

Guan, K. (2014). Important Notes on Lyapunov Exponents. https://arxiv.org/abs/1401.3315

Jordan, D., & Smith, P. (2007). *Nonlinear Ordinary Differential Equations*. Oxford.

Khovanskii, A. (2018). Integrability in Finite Terms And Actions of Lie Groups. https://arxiv.org/abs/1808.06326

Lüdde, H. J. (2000). *Mathematische Ergänzung zur Theoretischen Physik II*. Verfügbar 15. Dezember 2024 unter https://itp.uni-frankfurt.de/~luedde/E-Skript/Dgl/Dgl.pdf

Mathepedia. (o.J.). *Anfangswertprobleme*. Verfügbar 14. Dezember 2024 unter https://mathepedia.de/Anfangswertprobleme.html

Nipp, K. (2006). *Gewöhnliche Differentialgleichungen*. Verfügbar 15. Dezember 2024 unter https://people.math.ethz.ch/~grsam/Numerik_MAVT_SS06/skript/kn-num-skript-kap8.pdf

Richter, P. H., & Scholz, H. J. (o.J.). *Das ebene Doppelpendel*. Verfügbar 12. Dezember 2024 unter https://www.itp.uni-bremen.de/prichter/download/DoppelpendelIWF.pdf

Russell, D. A. (o.J.). *Phase Space Diagrams for an Oscillator (undamped and damped)*. Verfügbar 15. Dezember 2024 unter https://www.acs.psu.edu/drussell/Demos/phase-diagram/phase-diagram.html

Schuster, H. G. (2000). *Chaos)*. Verfügbar 15. Dezember 2024 unter https://www.spektrum.de/lexikon/physik/chaos/2292

SciPy. (o.J.). *solve_ivp*. Verfügbar 15. Dezember 2024 unter https://docs.scipy.org/doc/scipy/reference/generated/scipy.integrate.solve_ivp.html

Serlo. (o.J.). *Zwangsbedingungen und ihre Folgen* [Lizenz: CC BY-SA 4.0]. Verfügbar 11. Dezember 2024 unter https://de.serlo.org/physik/104026/zwangsbedingungen-und-ihre-folgen

Spektrum. (2000a). *Quasiperiodizität*. Verfügbar 15. Dezember 2024 unter https://www.spektrum.de/lexikon/physik/quasiperiodizitaet/11932

Spektrum. (2000b). *Virtuelle Verschiebung*. Verfügbar 10. Dezember 2024 unter https://www.spektrum.de/lexikon/physik/virtuelle-verschiebung/15276

Sprott, J. C. (2015). *Numerical Calculation of Largest Lyapunov Exponent*. Verfügbar 15. Dezember 2024 unter https://sprott.physics.wisc.edu/chaos/lyapexp.htm

Stein, L. C. (o.J.). *Poincaré section clicker for the double pendulum*. Verfügbar 12. Dezember 2024 unter https://duetosymmetry.com/tool/poincare-section-clicker-toy/

Studyflix. (o.J.). *Jacobi-Matrix*. Verfügbar 15. Dezember 2024 unter https://studyflix.de/mathematik/jacobi-matrix-1349

Udwadia, F. E., & Kalaba, R. E. (2001). Explicit Equations of Motion for Mechanical Systems With Nonideal Constraints. *Journal of Applied Mechanics*, *68(3)*, 462–467. https://doi.org/10.1115/1.1364492

Universität Innsbruck. (o.J.). *Zwangsbedingungen und ihre Folgen*. Verfügbar 14. Dezember 2024 unter http://physik.uibk.ac.at/01-02/physik1/pdfs/kap2_14.pdf

Wagner, S. (o.J.). *Taylorpolynome in mehreren Variablen*. Verfügbar 15. Dezember 2024 unter https://www.math.tugraz.at/~wagner/Taylor.pdf

Abbildungsverzeichnis

Hinweis zu den Abbildungen

Alle in dieser Arbeit verwendeten Abbildungen wurden vom Autor selbst erstellt.

Anhang A

Code

Hier wird der Code für die Erstellung der Abbildungen aufgeführt. Über den jeweiligen Boxen wird angegeben, für welche Abbildung(en) der Code gebraucht wurde.

A.1 Mathematica Code

Abbildung 2.1

```
sol = DSolveValue[{y'[x] == y[x]*x^2 - y[x], y[1] ==
   1}, y[x], x]
Show[StreamPlot[{1, y*x^2 - y}, {x, -2, 2}, {y, -4,
   4}],
    Plot[sol, {x, -4, 4}, PlotStyle -> Red]]
```

A.2 Python Code

Das ist der Basiscode, der für alle anderen Berechnungen benutzt wurde. Er wird hier einmal angeführt, um unnötige Repetition zu vermeiden.

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import solve_ivp
import sympy as smp

```

```
t, r1, r2, m1, m2, k, M0, l0 = smp.symbols('t r_1 r_2
    m_1 m_2 k M_0 l_0')

the1, the2 = smp.symbols(r'\theta_1 \theta_2', cls=
   smp.Function)

the1 = the1(t)
the2 = the2(t)
the1_d = smp.diff(the1, t)
the2_d = smp.diff(the2, t)
the1_dd = smp.diff(the1_d, t)
the2_dd = smp.diff(the2_d, t)
x1 = -smp.sin(the1)*r1
y1 = smp.cos(the1)*r1
x2 = -smp.sin(the2)*r2
y2 = smp.cos(the2)*r2 + M0
T=smp.Rational(1,2) * m1 * (smp.diff(x1,t)**2 + smp.
   diff(y1,t)**2) \
    +smp.Rational(1,2) * m2 * (smp.diff(x2,t)**2 +
       smp.diff(y2,t)**2)
V = smp.Rational(1,2)*k*(smp.sqrt((smp.sin(the1)*r1-
   smp.sin(the2)*r2)**2 + (smp.cos(the2)*r2+M0-smp.
   cos(the1)*r1)**2)-l0)**2
L=T-V
LE1 = smp.diff(L, the1) - smp.diff(smp.diff(L, the1_d
   ), t).simplify()
LE2 = smp.diff(L, the2) - smp.diff(smp.diff(L, the2_d
   ), t).simplify()
sols = smp.solve([LE1, LE2], (the1_dd, the2_dd),
                 simplify=False, rational=False)

#zur Berechnung der Bewegungsgleichung
#print(sols)

dz1dt_f = smp.lambdify((t,k,l0,m1,m2,r1,r2, M0, the1,
   the2,the1_d,the2_d), sols[smp.Derivative(the1, (t,
    2))])
dz2dt_f = smp.lambdify((t,k,l0,m1,m2,r1,r2, M0, the1,
   the2,the1_d,the2_d), sols[smp.Derivative(the2, (t,
    2))])
```

```
dthe1dt_f = smp.lambdify(the1_d, the1_d)
dthe2dt_f = smp.lambdify(the2_d, the2_d)

def dSdt(t, S, k, l0, m1, m2, r1, r2):
    the1, z1, the2, z2 = S
    dthe1_dt = dthe1dt_f(z1)
    dz1_dt = dz1dt_f(t,k,l0,m1,m2,r1,r2,M0,the1,the2,
       z1,z2)
    dthe2_dt = dthe2dt_f(z2)
    dz2_dt = dz2dt_f(t,k,l0,m1,m2,r1,r2,M0,the1,the2,
       z1,z2)
    return [dthe1_dt, dz1_dt, dthe2_dt, dz2_dt]

t = np.linspace(0, 40, 1001)
k = 20
m1=2
m2=2
r1 = 1
r2 = 1
l0 = 1
M0 = 3
the10 = 1.82
the20 = np.pi
the1_d0 = 0
the2_d0 = 0
initial_conditions = [the10, the1_d0, the20, the2_d0]
ans = solve_ivp(dSdt, (0, t.max()), t_eval=t, args=(k
   , l0, m1, m2, r1, r2), y0=initial_conditions)
the1 = ans.y[0]
the1_d = ans.y[1]
the2 = ans.y[2]
the2_d = ans.y[3]
```

Abbildungen 2.4, 2.5

```
def heun_schritte(t, S, dt, func, k, l0, m1, m2, r1,
   r2):
    S_pred = S + dt * np.array(func(t, S, k, l0, m1,
       m2, r1, r2))
    S_corr = S + (dt / 2) * (
        np.array(func(t, S, k, l0, m1, m2, r1, r2)) +
```

```
        np.array(func(t + dt, S_pred, k, l0, m1, m2,
            r1, r2))
    )
    return S_corr

dt = t[1] - t[0]
schritte = len(t)
S = np.zeros((schritte, 4))
S[0] = initial_conditions

time_start = time.time()
for i in range(schritte - 1):
    S[i + 1] = heun_schritte(t[i], S[i], dt, dSdt, k,
        l0, m1, m2, r1, r2)
time_end = time.time()
#Berechnung der Rechenzeit
print("Rechenzeit: ", time_end - time_start)

def Toatale_Energie(S, r1, r2, m1, m2, k, l0, M0):
    the1_werte, the1_d_werte, the2_werte,
        the2_d_werte = S.T
    the1 = the1_werte
    the2 = the2_werte
    the1_d = the1_d_werte
    the2_d = the2_d_werte

    x1_d = -np.cos(the1) * the1_d * r1
    y1_d = -np.sin(the1) * the1_d * r1
    x2_d = -np.cos(the2) * the2_d * r2
    y2_d = -np.sin(the2) * the2_d * r2
    T_werte = 0.5 * m1 * (x1_d**2 + y1_d**2) + 0.5 *
        m2 * (x2_d**2 + y2_d**2)

    x1 = -np.sin(the1) * r1
    y1 = np.cos(the1) * r1
    x2 = -np.sin(the2) * r2
    y2 = np.cos(the2) * r2 + M0
    V_werte = 0.5 * k * (np.sqrt((np.sin(the1)*r1-np.
        sin(the2)*r2)**2 +
                (np.cos(the2)*r2+M0-np.cos(the1)*r1)
```

```
                    **2)-l0)**2

    E_tot = T_werte + V_werte
    E_diff = (E_tot - E_tot[0]) / E_tot[0]
    return E_diff

E_diff = Toatale_Energie(S, r1, r2, m1, m2, k, l0, M0
    )
plt.figure(figsize=(10, 6))
plt.plot(t, E_diff, label= "Heun", color="blue")
plt.legend()
plt.xlabel("Zeit (s)")
plt.ylabel("Relative Abweichung")
plt.grid(True)
plt.show()
```

Abbildungen 2.6, 2.7, 2.8, 2.9

```
def Toatale_Energie(ans, r1, r2, m1, m2, k, l0, M0):
    the1 = ans.y[0]
    the2 = ans.y[2]
    the1_d = ans.y[1]
    the2_d = ans.y[3]

    x1_d = -np.cos(the1) * the1_d * r1
    y1_d = -np.sin(the1) * the1_d * r1
    x2_d = -np.cos(the2) * the2_d * r2
    y2_d = -np.sin(the2) * the2_d * r2
    T_werte = 0.5 * m1 * (x1_d**2 + y1_d**2) + 0.5 *
       m2 * (x2_d**2 + y2_d**2)

    x1 = -np.sin(the1) * r1
    y1 = np.cos(the1) * r1
    x2 = -np.sin(the2) * r2
    y2 = np.cos(the2) * r2 + M0
    V_werte = 0.5 * k * (np.sqrt((np.sin(the1)*r1-np.
       sin(the2)*r2)**2 +
                (np.cos(the2)*r2+M0-np.cos(the1)*r1)
                   **2)-l0)**2

    E_tot = T_werte + V_werte
```

```
    E_diff = (E_tot - E_tot[0]) / E_tot[0]
    return E_diff

timing_results = {}

start_time = time.time()
ans = solve_ivp(dSdt, (0, t.max()), y0=
   initial_conditions, t_eval=t, method='RK45', args
   =(k, l0, m1, m2, r1, r2))
timing_results["RK45"] = time.time() - start_time

start_time = time.time()
ans1 = solve_ivp(dSdt, (0, t.max()), y0=
   initial_conditions, t_eval=t, method='DOP853',
   args=(k, l0, m1, m2, r1, r2))
timing_results["DOP853"] = time.time() - start_time

start_time = time.time()
ans2 = solve_ivp(dSdt, (0, t.max()), y0=
   initial_conditions, t_eval=t, method='DOP853',
                args=(k, l0, m1, m2, r1, r2), rtol=1
                   e-10, atol=1e-10)
timing_results["DOP853 (präzise)"] = time.time() -
   start_time

start_time = time.time()
ans3 = solve_ivp(dSdt, (0, t.max()), y0=
   initial_conditions, t_eval=t, method='RK45', args
   =(k, l0, m1, m2, r1, r2),  rtol=1e-10, atol=1e-10)
timing_results["RK45 (präzise)"] = time.time() -
   start_time

E_diff = Toatale_Energie(ans, r1, r2, m1, m2, k, l0,
   M0)
E_diff_1 = Toatale_Energie(ans1, r1, r2, m1, m2, k,
   l0, M0)
E_diff_2 = Toatale_Energie(ans2, r1, r2, m1, m2, k,
   l0, M0)
E_diff_3 = Toatale_Energie(ans3, r1, r2, m1, m2, k,
   l0, M0)
```

```
47
48 plt.figure(figsize=(8, 5))
49 plt.bar(timing_results.keys(), timing_results.values
      (), color=["blue", "orange", "green", "yellow"])
50 plt.ylabel("Rechenzeit (s)")
51 plt.title("Vergleich der Rechenzeit")
52 plt.show()
53
54
55 plt.figure(figsize=(10, 6))
56 plt.plot(t, E_diff, label= "RK54", color="blue")
57 plt.plot(t, E_diff_1, label="DOP853", color="orange")
58 plt.plot(t, E_diff_2, label="DOP853 (präzise)", color
      ="green")
59 plt.plot(t, E_diff_3, label="RK54 (präzise)", color="
      yellow")
60 plt.xlabel("Zeit (s)")
61 plt.ylabel("Relative Abweichung")
62 plt.grid(True)
63 plt.legend()
64 plt.show()
```

Abbildung 2.5, 2.9

```
1     t = np.linspace(0, 1000, 100001, dtype=np.float64
         )
```

Abbildung 3.2

```
1 t = np.linspace(0, 30, 1001, dtype=np.float64)
2 the10 = 1
3 the20 = 0
4 the1_d0 = 0
5 the2_d0 = 0
6 initial_conditions = [the10, the1_d0, the20, the2_d0]
7 ans = solve_ivp(dSdt, (0, t.max()), y0=
      initial_conditions, t_eval=t, method='DOP853',
      args=(k, l0, m1, m2, r1, r2), rtol=1e-10, atol=1e
      -10)
8 the1 = ans.y[0]
9 the2 = ans.y[2]
10 the1_d = ans.y[1]
```

```
the2_d = ans.y[3]

#leichte änderung der anfangsbedingungen
the100 = the10+0.0001
the200 = 0
the1_d00 = 0
the2_d00 = 0
initial_conditions_pert = [the100, the1_d00, the200,
   the2_d00]
ans0 = solve_ivp(dSdt, (0, t.max()), y0=
   initial_conditions_pert, t_eval=t, method='DOP853'
   , args=(k, l0, m1, m2, r1, r2), rtol=1e-10, atol=1
   e-10)
the10 = ans0.y[0]
the20 = ans0.y[2]
the1_d0 = ans0.y[1]
the2_d0 = ans0.y[3]
plt.figure()
plt.plot(t, the1, label=r'$\gamma_1$', color='b')
plt.plot(t, the10, label=r'$\gamma_2,$', color='g')
plt.xlabel('Zeit [s]')
plt.ylabel(r'Winkel [rad]')
plt.title(r'Zeitlicher Verlauf von $\theta$ für zwei
   leicht verschiedene Anfangsbedingungen')
plt.legend()
plt.grid(True)
plt.show()
```

Abbildung 3.4

```
t = np.linspace(0, 100, 20000, dtype=np.float64)
the20 = np.pi
the1_d0 = 0
the2_d0 = 0

angles = (0.1, 0.4, 0.7, 1.0, 1.4, 1.8, 2.2, 2.6,
   3.1)
num_plots = len(angles)
fig, axs = plt.subplots((num_plots + 2) // 3, 3,
   figsize=(12, 12))

```

```
for idx, the10 in enumerate(angles):
    initial_conditions = [the10, the1_d0, the20,
       the2_d0]
    ans = solve_ivp(dSdt, (0, t.max()), y0=
       initial_conditions, t_eval=t, method='DOP853',
        args=(k, l0, m1, m2, r1, r2, M0), rtol=1e-10,
        atol=1e-10)
    the1 = ans.y[0]
    the1_d = ans.y[1]

    cutoff = int(0.1 * len(the1_d))
    the1_d = the1_d[cutoff:]
    the1 = the1[cutoff:]

    ax = axs[idx // 3, idx % 3]
    ax.plot(the1, the1_d, color='b')
    ax.set_title(f"$\gamma (0)={the10}$")
    ax.set_xlabel(r"$\gamma$")
    ax.set_ylabel(r"$\dot{\gamma}$")
    ax.grid(True)

plt.subplots_adjust(hspace=1)
plt.tight_layout(rect=[0, 0, 1, 0.95])
plt.show()
```

Abbildung 3.6

```
def lyapunov_exponent(anfangsbedingung, t_span, dt, *
   args):
    delta0 = 1e-8
    S0 = np.array(anfangsbedingung)
    perturbation = delta0 * np.random.randn(len(S0))
    total_lyapunov = 0
    steps = int((t_span[1] - t_span[0]) / dt)

    for i in range(steps):
        t0 = t_span[0] + i * dt
        t1 = t0 + dt
        sol1 = solve_ivp(dSdt, [t0, t1], S0, args=
           args, t_eval=[t1], rtol=1e-10, atol=1e-10)
        sol2 = solve_ivp(dSdt, [t0, t1], S0 +
```

```
            perturbation, args=args, t_eval=[t1], rtol
            =1e-10, atol=1e-10)

        S0 = sol1.y[:, -1]
        perturbation = sol2.y[:, -1] - S0
        diff = np.linalg.norm(perturbation) + 1e-12
        total_lyapunov += np.log(diff / delta0)
        perturbation *= delta0 / diff

    return total_lyapunov / (steps * dt)

t_span = (0, 100)
dt = 0.1
winkel = np.arange(0, np.pi, 0.1)
lyapunov_array = []
for i in tqdm(range(len(winkel))):
    anfangsbedingung = [winkel[i], 0.0, np.pi, 0.0]
    lyapunov = lyapunov_exponent(anfangsbedingung,
       t_span, dt, k, l0, m1, m2, r1, r2, M0, dz1dt_f
       , dz2dt_f, dthe1dt_f, dthe2dt_f)
    lyapunov_array.append(lyapunov)

plt.plot(winkel, lyapunov_array)
plt.xlabel(r'$\gamma (0)$')
plt.ylabel('maximaler Ljapunow-Exponent')
plt.show()
```

Abbildungen 3.8, 3.7, 3.9

```
t_stop = 10000
t = np.linspace(0, t_stop, 1000000)
#für the10 = 3.1 wurde länger simuliert
#the10 = 3.1
#t_stop = 30000
#t = np.linspace(0, t_stop, 3000000)
the10 = 1.4
#the10 = 1.8
the20 = np.pi
the1_d0 = 0
the2_d0 = 0
initial_conditions = [the10, the1_d0, the20, the2_d0]
```

```
sol = solve_ivp(dSdt, (0, t_stop), y0=
    initial_conditions, t_eval=t, method='DOP853',
    args=(k, l0, m1, m2, r1, r2), rtol=1e-13, atol=1e
    -13)

the1 = sol.y[0]
z1 = sol.y[1]
the2 = sol.y[2]
z2 = sol.y[3]


poincare_points = np.array([[the2[i], z2[i]] for i in
     range(len(t) - 1) if the1[i]*the1[i+1] < 0 and z1
    [i] > 0])

plt.figure(figsize=(10, 10))
plt.scatter(poincare_points[:,0], poincare_points
    [:,1], s=0.1)
plt.xlabel(r'$\theta$')
plt.ylabel(r'$\dot{\theta}$')
plt.title('$\gamma (0) = 3.1$')
plt.show()
```

Abbildung 3.10

```
t = np.linspace(0, 40, 11000, dtype=np.float64)

#Bifurkations diagramm
winkel = np.arange(0, np.pi, 0.01)
bifurkation = []

for the10 in winkel:
    initial_conditions = [the10, 0, np.pi, 0]
    ans = solve_ivp(dSdt, (0, t.max()), y0=
        initial_conditions, t_eval=t, method='DOP853',
         args=(k, l0, m1, m2, r1, r2, M0), rtol=1e-10,
         atol=1e-10)

    the1 = ans.y[0]
    the1_d = ans.y[1]
    the2 = ans.y[2]
```

```
    the2_d = ans.y[3]

    cutoff = int(0.2 * len(the2_d))
    the1 = the1[cutoff:]
    the1_d = the1_d[cutoff:]
    the2 = the2[cutoff:]
    the2_d = the2_d[cutoff:]

    for i in range(1, len(the2_d)):
        if the2_d[i-1] * the2_d[i] < 0:
            bifurkation.append((the10, the1_d[i]))

bifurcation_data = np.array(bifurkation)
x_vals, y_vals = bifurcation_data[:, 0],
   bifurcation_data[:, 1]

plt.figure(figsize=(10, 6))
plt.scatter(x_vals, y_vals, s=0.1, color='b')
plt.xlabel(r'$\gamma (0)$')
plt.ylabel(r'$\dot{\gamma}$')
plt.show()
```