Computergestütztes wissenschaftliches Rechnen 2

Hausarbeit Pendel Chaosübergang

Fabian Heimann [‡]

[‡] *Universität Göttingen, Fabian.Heimann@stud.uni-goettingen.de*

Sommersemester 2013

Zusammenfassung

In der folgenden Arbeit wurde das Verhalten eines periodisch angetriebenen, gedämpften Pendels bei verschieden starken Anregungen numerisch untersucht. Das zugehörige System von Differentialgleichungen wird dann zuerst numerisch mit dem klassischen Runge-Kutta-Verfahren gelöst. Tests zeigen jedoch, dass die Ergebnisse dieses Verfahrens bei chaotischem Pendelverhalten in absehbarer Rechenzeit keine verwertbaren Ergebnisse liefern. Daher wurde das Bulirsch-Stoer-Verfahren implementiert und erfolgreich getestet. Durch Auswertung der Trajektorien im Phasenraum wurde dann untersucht, ab welcher Stärke der Anregung das Verhalten des Pendels chaotisch wird. Abschließend wurde der Lyapunov-Exponent des Systems für verschieden starke Anregungen bestimmt.

Inhaltsverzeichnis

1 Einleitung

Im Folgenden betrachten wir ein einfaches Pendel mit Auslenkung θ und Winkelgeschwindigkeit $\omega = \dot{\theta}$. Es sei zum einen mit einem Parameter q gedämpft. Außerdem sei es periodisch angetriebenen mit einer Anregung der Frequenz Ω und Stärke f_c. Dann erhalten wir für die Bewegung des Pendels folgendes System von Differentialgleichungen:

$$\frac{\mathrm{d}\omega}{\mathrm{d}t} = -\sin\theta - q\omega + f_c \sin(\Omega t) \tag{1a}$$

$$\frac{\mathrm{d}\theta}{\mathrm{d}t} = \omega. \tag{1b}$$

Thema der folgenden Arbeit soll es zuerst sein, ein Programm in C++ zu implementieren, dass diese Bewegungsgleichungen numerisch löst. Das Programm soll als Ergebnis die Auslenkung für die Zeitpunkte

$$t_n = \frac{2\pi}{\Omega} \cdot n \tag{2}$$

ausgeben.

Danach soll anhand dieses Programmes der Wert für f_c bestimmt werden, bei dem die Bewegung im Phasenraum in eine chaotische Bewegung umschlägt. Die zu betrachtenden Rahmenbedingungen dafür sind

$$\Omega = \frac{2}{3}, \quad q = \frac{1}{2}. \tag{3}$$

In einem letzten Teil soll der Wert für den sog. Lyapunov-Exponenten λ für verschiedene f_c bestimmt werden. Dieser beschreibt die Divergenz der Bahn zweier Pendel, die bei leicht unterschiedlichen Anfangsauslenkungen und der Winkelgeschwindigkeit 0 gestartet werden. Sei $\Delta\theta_n = \theta^{(1)}(t_n) - \theta^{(2)}(t_n)$. Dann gilt folgendes:

$$|\Delta\theta_n| = |\Delta\theta_0| \cdot e^{\lambda n}. \tag{4}$$

2 Methodik

2.1 Numerische Integration

Wie oben motiviert sollen nun die Differentialgleichungen (1) numerisch gelöst werden. Dazu muss dieses System aus Differentialgleichungen erster Ordnung zuerst in eine einfache Differentialgleichung umgeschrieben werden, um die Lösung durch ein Standardverfahren zu ermöglichen. Dazu definieren wir uns einen Vektor, der den Zustand unseres Pendels zum Zeitpunkt t_n beschreibt: $y(t) = (\theta(t), \omega(t))^T$. Dann ist

$$\dot{y}(t, y) = \begin{pmatrix} \omega \\ -\sin\theta - q\omega + f_c \sin(\Omega t) \end{pmatrix} \tag{5}$$

Damit haben unsere einfache Differentialgleichung $\dot{y}(y)$.

2.1.1 Runge-Kutta-Verfahren

Diese soll nun zuerst mit dem klassischen Runge-Kutta-Verfahren gelöst werden. Dazu diskretisieren wir die gesuchte Lösungsfunktion $y(t)$ auf ein Zeitraster mit der Schrittweite h. Es ist dann: $t_i = t_0 + i \cdot h$ und $y_i = y(t_i)$.

Das Verfahren folgt folgender Idee [1, S. 120]: An vier Stützstellen innerhalb des Intervalles wird ein Näherungswert der Ableitung berechnet. Als erstes nehmen wir dazu den Punkt am Intervallanfang:

$$k_1 = f(t_i, y_i) \tag{6a}$$

Die zweite Stützstelle liegt in der Intervallmitte. Den Wert dort berechen wir wie durch Benutzung von k_1

$$k_2 = f(t_i + \frac{h}{2}, y_i + k_1 \cdot \frac{h}{2}) \tag{6b}$$

Als dritte Stützstelle berechnen wir den Wert für die Intervallmitte mithilfe von k_2:

$$k_3 = f(t_i + \frac{h}{2}, y_i + k_2 \cdot \frac{h}{2}) \tag{6c}$$

Die dritte Stützstelle liegt am Ende des Intervalles und wird mithilfe von k_3 berechnet:

$$k_4 = f(t_i + h, y_i + k_3 \cdot h) \tag{6d}$$

Nun bilden wir einen gewichteten Mittelwert und erhalten unseren neuen Funktionswert

$$y_{i+1} = y_i + \frac{h}{6}\left[k_1 + 2k_2 + 2k_3 + k_4\right]. \tag{6e}$$

Dieses klassiche Runge-Kutta-Verfahren ist von der Ordnung 4. Der Fehler fällt also mit h^4. [1, S. 120] Für eine detailliertere und anschaulichere Darstellung sei auf [2, S. 711] verwiesen.

2.1.2 Fehlerbetrachtung

Wie bei jedem numerischen Verfahren zur Integration gewöhnlicher Differentialgleichungen entsteht auch bei diesem Verfahren ein numerischer Fehler in der Lösung. Ziel ist es, diesen Fehler durch Anpassung der Schrittweite so klein werden zu lassen, dass er die Ergebnisse nicht beeinflusst.

Eine Fehlerabschätzung ist immer schwierig, wenn keine analytische Lösung oder Erhaltungsgrößen vorliegen. Beides ist hier nicht der Fall. Dann bieten sich vor allem zwei Möglichkeiten an: Rückwärtsrechnen nach dem eigentlichen Ende der Simulation oder paralleles Rechnen mit einer höheren Schrittweite. Hier soll letzteres durchgeführt werden.

Die Schrittweite h des Runge-Kutta-Verfahrens wird relativ zur Periodendauer der Anregung mit H wie folgt angegeben:

$$h = \frac{2\pi}{\Omega} \cdot \frac{1}{H}. \tag{7}$$

Nun wird in dem Programm eine Lösung für einen typischen Startfall durchgerechnet. Zusätzlich zu der Berechnung mit der aktuellen Schrittweite wird jeweils auch mit einer Schrittweite gerechnet, die um einen Faktor 100 kleiner ist. Dann wird die Differenz der beiden Lösungen als Betrag der Differenz der Zustandsvektoren y bestimmt. Dies erlaubt eine gute Abschätzung des numerischen Fehlers.

Zu unterscheiden ist dabei der geordnete und der chaotische Fall. Betrachten wir zuerst den nicht-chaotischen Fall. Dieser ist zwar nicht sehr aussagekräftig für die Genauigkeit in der späteren Simulation (denn auch wenn dieser Fall exakt berechnet wird, könnten Fehler im chaotischen Bereich auftreten), er erlaubt aber Aussagen über das Verhalten des

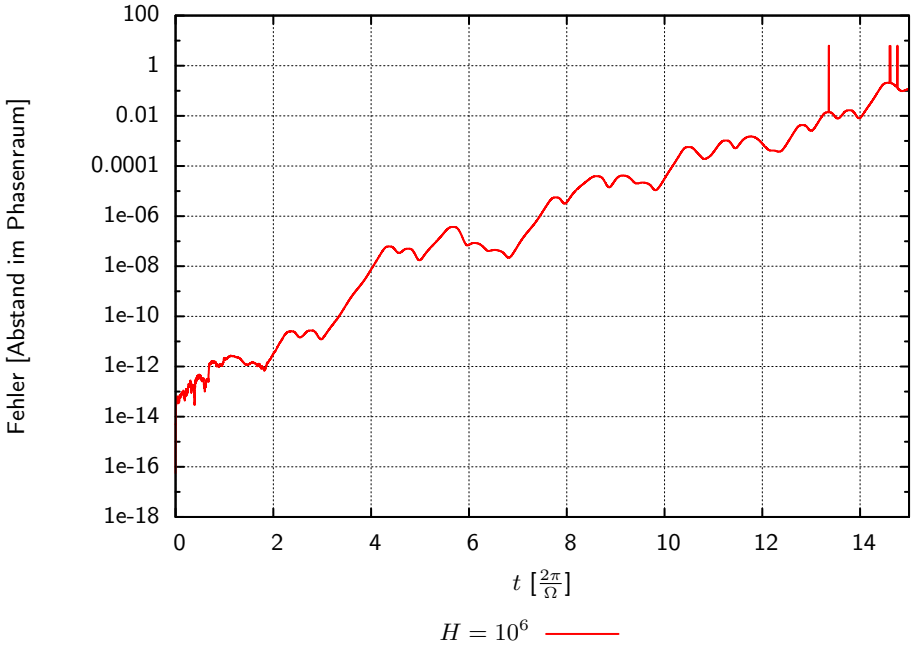

Abbildung 1: Fehlerabschätzung des Runge-Kutta-Verfahrens für verschiedene Werte für H bei $f_c = 1.2$

Fehlers, wenn h gegen 0 geht. In Abb. 7 sind die Fehler für verschiedene H bei $f_c = 0.5$ (nicht-chaotisch) dargestellt. Es zeigt sich, dass ab einem Wert von $H = 10^5$ die Fehler im Bereich der Rundungsfehler eines doubles liegen. Inklusive der obigen Kontrolle braucht eine solche Rechnung auf dem verwendeten 2-GHz-Notebook etwa 2 Minuten. Interessant ist am Graphen auch der Abfall des Fehlers, der der erwarteten Proportionalität zu h^4 entspricht. Dies ist ein starkes Indiz dafür, dass das Runge-Kutta-Verfahren korrekt implementiert wurde.

Betrachten wir nun den chaotischen Fall bei $f_c = 1.2$. Hier versagt das Runge-Kutta-Verfahren. Bei einem H-Wert von 10^6 erhalten wir über die simulierten 11 Perioden einen Fehler von 0.1 als Abstand im Phasenraum bei einer Rechenzeit von 90 Minuten. Der Verlauf ist in Abb. 1 dargestellt.

2.2 Bulirsch-Stoer-Verfahren

Um mit absehbarer Rechenzeit trotzdem verwertbare Ergebnisse zu erzielen, wird das Bulirsch-Stoer-Verfahren implementiert.[2, S. 724ff.] [3, S. 85ff.] Es ist ein fortgeschrittenes Verfahren, dass auf der modified midpoint method basiert. Aus Platzgründen wollen wir es hier nur kurz umreißen: Beim Verfahren wird ein Zeitschritt h in viele kleine Zeitschritte $h' = \frac{h}{N}$ aufgeteilt. Dadurch kann für eine beliebige Ordnung N eine Näherung für den

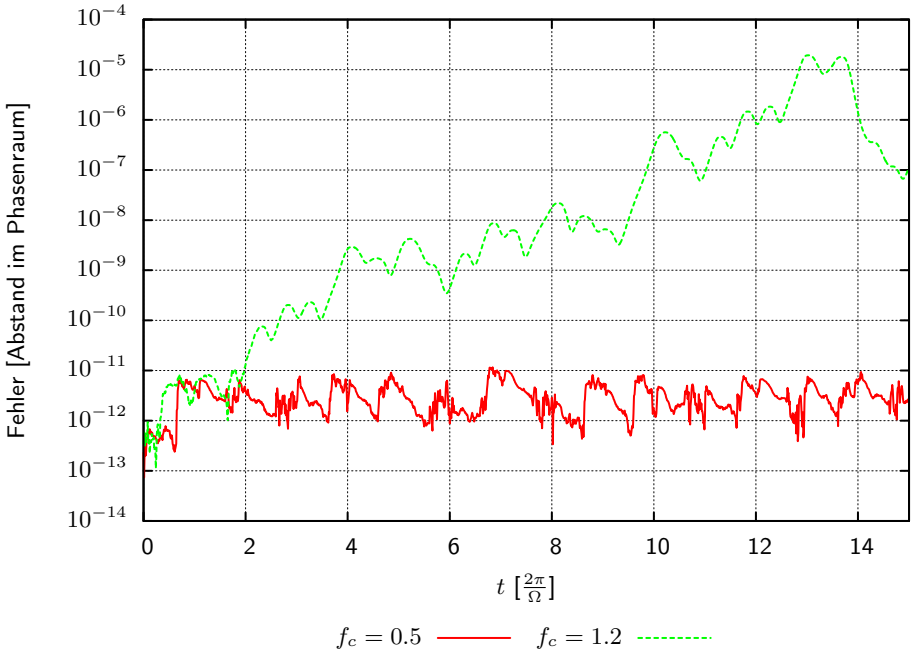

Abbildung 2: Fehlerabschätzung des Bulirsch-Stoer-Verfahrens für f_c =0.5 und 1.2

neuen Wert von y bestimmt werden:

$$y_0 = y(t), y_1 = y_0 + h' \cdot f(t, y_0) \tag{8a}$$

$$y_n = y_{n-2} + 2h' f(t + (n-1)h', y_{n-1}) \quad (n = 2, \ldots, N) \tag{8b}$$

$$y(t+h) = \frac{1}{2}\left[y_N + y_{N-1} + hf(t+h, y_N)\right] \tag{8c}$$

Die zentrale Idee des Bulirsch-Stoer-Verfahrens ist folgende: Wir berechnen für verschiedene Ordnungen N eine Näherung $y(t+h)$. Dann berechnen wir zu jeder dieser Ordnungen h' und extrapolieren gegen $h \to 0$. Dazu bietet sich polynomiale Interpolation nach Lagrange an, konkret z.B. nach Nevilles Methode (siehe dafür [4, S. 878f.]). Ein weiterer Trick des Verfahrens besteht darin, dann in der Interpolation einmal kleine Schrittweiten wegzulassen, um so zu ermitteln, wie schnell das Verfahren zu diesem Zeitschritt konvergiert. Ist der Unterschied zwischen den beiden Lösungen kleiner als ein Schwellenwert ϵ, wird die Lösung als konvergent betrachtet und zum neuen Schritt übergegangen. Ansonsten berechnet die Schleife höhere Ordnungen bis die Lösung konvergiert. Das sorgt dafür, dass das Verfahren automatisch dort genau rechnet, wo große Abweichungen auftauchen.

Die Genauigkeit des Bulirsch-Stoer-Verfahrens bestimmen wir nach dem selben Prinzip wie bei Runge-Kutta. In Abb. 2 sind die Graphen für eine Schrittweite $H = 100$ dargestellt. Eine solche Simulation benötigt inklusive Fehlerabschätzung etwa 5 s Rechenzeit.

2.3 Implementierung

Das Konzept des Status-Vektors, der sowohl Auslenkung als auch Winkelgeschwindigkeit enthält, wird in einer entsprechenden Vektor-Klasse implementiert. Um das spätere Pro-

grammieren zu erleichtern, werden Elementfunktion zum addieren, multiplizieren usw. definiert.

Dann wird die Differentialgleichung (5) als Funktion implementiert.

Außerdem wird zuerst eine Funktion definiert, die nach dem oben gezeigten Schema einen Runge-Kutta-Schritt durchführt. Dies macht es z.B. wesentlich einfacher Runge-Kutta-Schritte mit verschiedener Schrittweite durchzuführen, wie dies für die obigen Genauigkeitstest erforderlich ist.

Für das Bulirsch-Stoer-Verfahren werden diverse Unterfunktionen implementiert; eine für die Berechnung eines Schrittes mit der modified midpoint method, eine für die Lagrange-Extrapolation und eine für den tatsächlichen Bulirsch-Stoer-Schritt.

2.4 Trajektorien im Phasenraum

Zur Bestimmung der Trajektorien gibt das Programm in einer Version wie in der Aufgabenstellung gefordert die Auslenkungen zu den Zeiten t_n aus (2) aus. In einer anderen Version werden allerdings auch Zwischenergebnisse ausgegeben. Diese ermöglichen es, die gesamte Bewegung des Pendels im Phasenraum nachzuvollziehen, denn so ist eine geschlossene Bahn erkennbar.

2.5 Bestimmung des Lyapunov-Exponenten

Zuletzt soll noch für verschiedene Werte von f_c der Lyapunov-Exponent bestimmt werden. Vorgegeben ist dazu zuerst ein Tupel $(\theta_0^{(1)}, \Delta\theta_0)$. Dann verfolgt das Programm folgendes Prinzip: Als erstes wird durch eine Schleife ein Wert für f_c vorgegeben. Dann wird nach dem obigen Verfahren eine Bahn numerisch integriert.

Zu jedem Zeitpunkt t_n bestimmt das Programm dann den lokalen Wert für den Lyapunov-Exponenten als

$$\lambda_n = \frac{1}{n} \cdot \ln \frac{|\Delta\theta_n|}{|\Delta\theta_0|} \tag{9}$$

Zum Schluss wird dann über alle λ_n gemittelt, die empirische Standardabweichung bestimmt und das Ergebnis für das spezielle f_c abgespeichert.

3 Ergebnisse

3.1 Trajektorien im Phasenraum

Die Trajektorien im Phasenraum werden einmal für die vorgegeben t_n und einmal über ein dichteres Zeitgitter ausgegeben. Die Ergebnisse über ein dichteres Zeitgitter erscheinen informativer; daher werden sie hier in Abb. 3 und 4 gezeigt. Die anderen Plots sind im Anhang in Abb. 8 dargestellt. Das Ergebnis ist bei beiden Darstellungsformen das selbe: Eine chaotische Bewegung beginnt bei f_c =1.0 bis 1.1.

3.2 Bestimmung des Lyapunov-Exponenten

Für die Bestimmung des Lyapunov-Exponenten wurden zuerst verschiedene Parameter für $(\theta_0^{(1)}, \Delta\theta_0)$ getestet. Diese führen alle zu relativ ähnlichen Ergebnissen. Ähnliches gilt für Verwendung unterschiedlicher Simulationszeiträume. Durch die automatisierte Berechnung für verschiedene f_c und das extrem schnelle Bulirsch-Stoer-Verfahren ist es möglich,

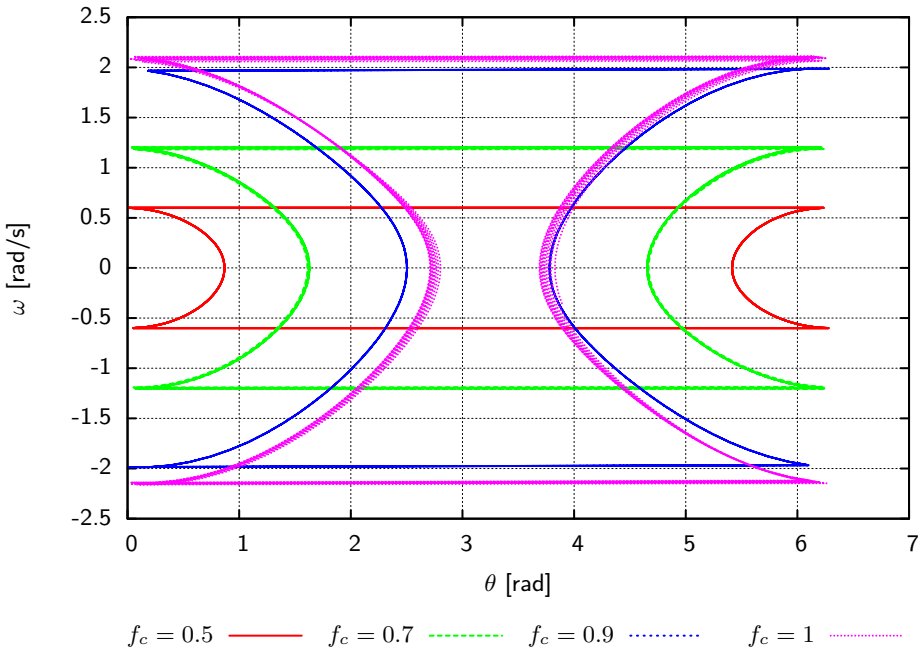

Abbildung 3: Stabile Trajektorien im Phasenraum. Bei f_c=1 ist bereits ein leicht chaotisches Verhalten zu erahnen. Die anderen Kurven sind geschlossen.

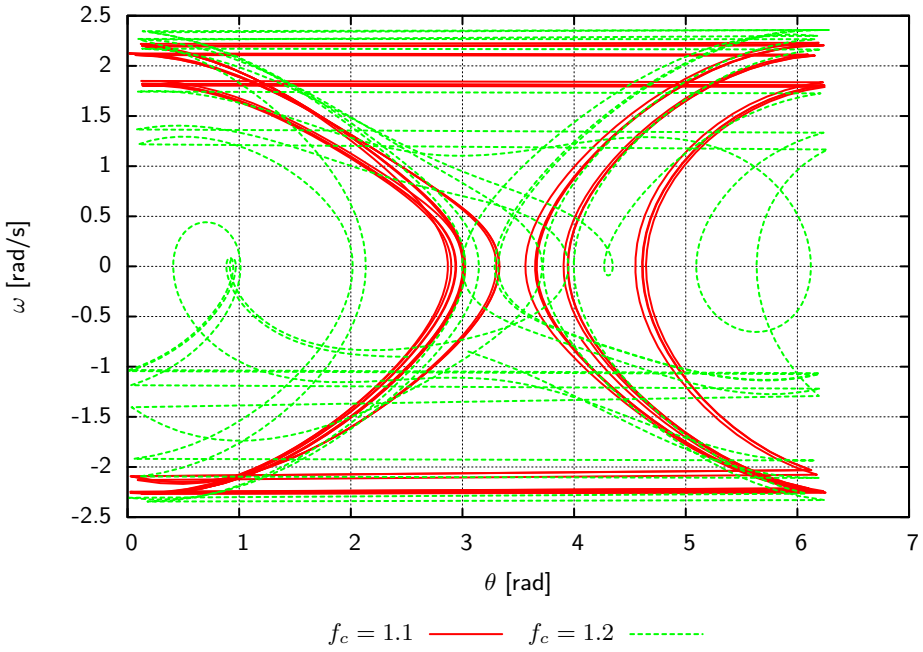

Abbildung 4: Chaotische Trajektorien im Phasenraum bei f_c=1.1.

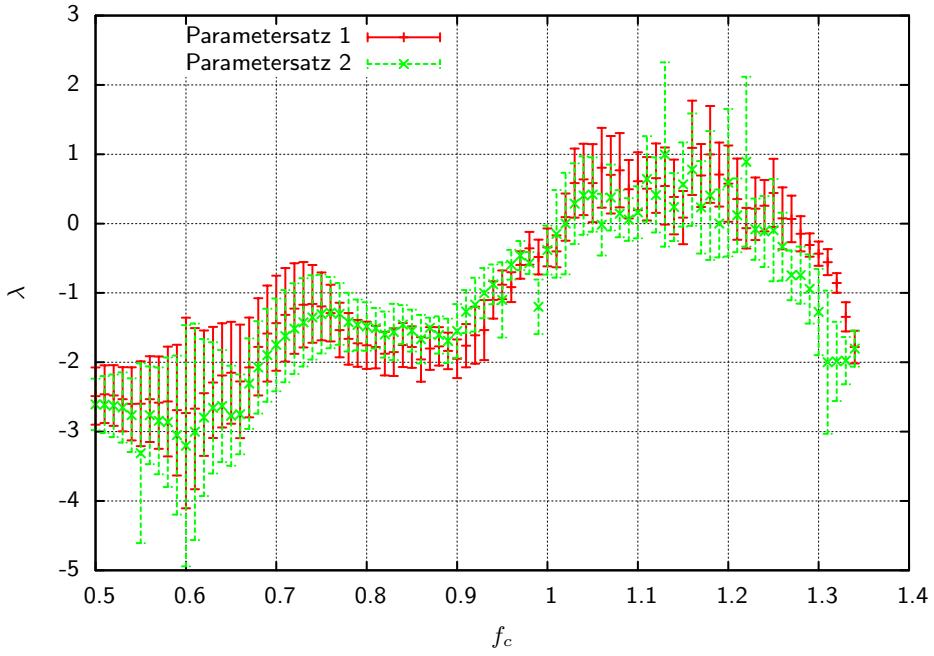

Abbildung 5: Plot für den Zusammenhang des Lyapunov-Exponenten von f_c. Parametersatz 1: $(\theta_0^{(1)}, \Delta\theta_0) = (\frac{\pi}{4}, 10^{-3})$, 10 Perioden, Parametersatz 2: $(\theta_0^{(1)}, \Delta\theta_0) = (\frac{\pi}{8}, 10^{-6})$, 5 Perioden

den Zusammenhang $\lambda(f_c)$ einfach und schnell für verschiedene Rahmenbedingungen berechnen zu lassen. Exemplarisch ist in Abb. 5 der Zusammenhang für 2 verschiedene Parametersätze dargestellt.

Um die Entstehung der verschiedenen Werte in dem Diagramm nachzuvollziehen, sind in Abb. 6 für verschiedene f_cs die Differenzen $\Delta\theta_n$ gegen die Zeit aufgetragen.

Für diese Werte für f_c wurden die Bestwerte für λ ebenfalls in gnuplot nach der χ^2-Methode berechnet. Die Ergebnisse dieser Auswertung sind in Tab. 1 dargestellt.

4 Diskussion

f_c	λ
0.5	-(3.24 ± 0.06)
0.7	-(2.8 ± 0.2)
0.9	-(1.18 ± 0.04)
1.0	-(0.3 ± 0.2)
1.1	0.68 ± 0.03
1.2	1.621 ± 0.002

Tabelle 1: Zusammenhang von λ und f_c mit χ^2-fit aus gnuplot.

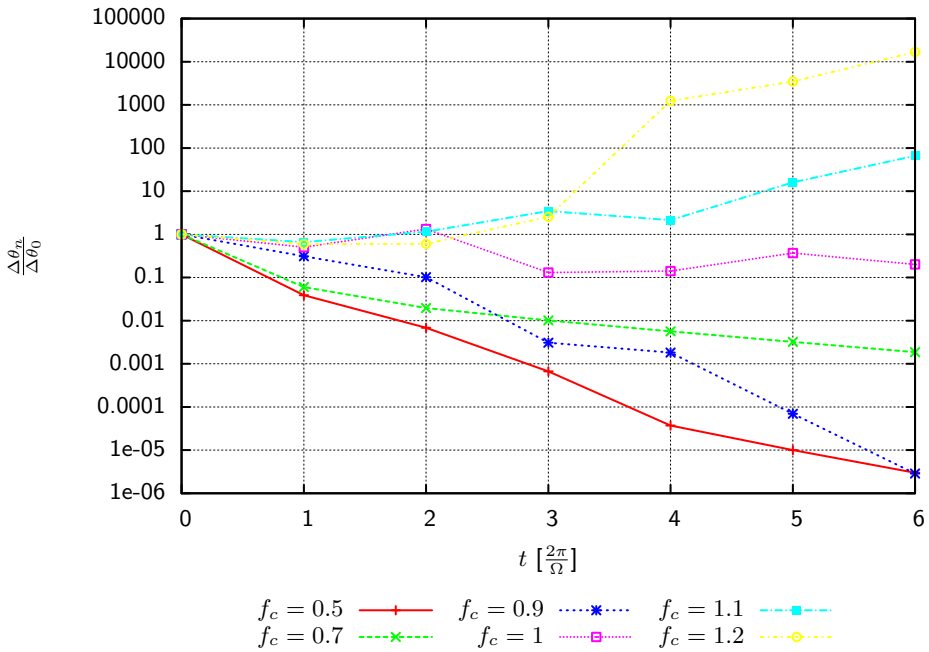

Abbildung 6: Übersicht über die zeitliche Entwicklung von $\Delta\theta_n$ bei $(\theta_0^{(1)}, \Delta\theta_0) = (\frac{\pi}{8}, 10^{-6})$

4.1 Numerisches Verfahren

Die Wahl des numerischen Verfahrens weicht von der Aufgabenstellung ab. Dies hat jedoch gute Gründe, da das verwendete Bulirsch-Stoer-Verfahren auf jeden Fall eine ausreichende Genauigkeit hat und in wesentlich kürzerer Zeit zu besseren Ergebnissen kommt.

4.1.1 Aussagekräftigkeit der Tests

Es mag in Frage gestellt werden, wie aussagekräftig die gemachten Tests sind. Dazu ist zu sagen, dass die generierten Abweichungen als Obergrenze für den Fehler gesehen werden sollten, denn als Vergleich lag keine exakte analytische Lösung, sondern ebenfalls eine numerische vor. Vor allem bei der langen Runge-Kutta-Simulation ist zu vermuten, dass in der als exakt angenommenen Lösung Rundungsfehler auftreten.

4.1.2 Fehlerquellen

Betrachtet man die Tiefe an Ordnungen, in die das Bulirsch-Stoer-Verfahren jeweils geht und vergleicht sie mit dem jeweils entstehenden Fehler, fällt auf, dass das Verfahren mit wenigen Ausnahmen bei konstanter Ordnung arbeitet. Das liegt daran, dass die Genauigkeit des doubles nicht für eine vernünftige Beurteilung der Konvergenz des Verfahrens liefert. Eine Möglichkeit, dies zu lösen, wäre z.B. die qd-Bibliothek [5], die jedoch aufgrund des Aufgabenrahmens nicht verwendet werden durfte.

4.2 Bestimmung des Lyapunov-Exponenten

Zur Bestimmung des Lyapunov-Exponenten ist zu sagen, dass die in Abb. 5 gezeigten Ergebnisse vermutlich wesentlich ungenauer sind als die in der Tab. 1 gezeigten. Denn hier wird die absolute Abweichung der Werte berücksichtigt. In der Mittelung der Lyapunov-Exponenten im C++-Programm werden durch den Logarithmus die Daten verzerrt. Die Mittelung entspricht in etwa dem Legen einer Ausgleichsgerade bei logarithmischer Skalierung, was natürlich zu ungenauen Ergebnissen führt. So ist der relativ unstetige Verlauf des Graphen eventl. nicht zentral auf die Messdaten zurück zu führen. Die Daten in der Tabelle sind also als wesentlich signifikanter zu bewerten.

Literatur

[1] Josef Stoer, Roland Bulirsch (2005): *Einführung in die Numerische Mathematik II*, 5. Auflage, Springer-Verlag Berlin, Heidelberg, New York

[2] W. Press, S. Teukolsky et al. (2002): *Numerical Recipes in C - The Art of Scientific Computing*, 2nd Edition, Camebridge University Press

[3] Peter Bodenheimer et al. (2007): *Numerical methods in astrophysics - an introduction* New York London: Taylor & Francis

[4] M. Abramowitz, I. Stegun (1972): *Handbook of mathematical functions*, 10. Auflage

[5] Yozo Hida, Xiaoye Li, David Bailey (2008): *Library for Double-Double and Quad-Double Arithmetic*, Dokumentation zum qd-Softwarepaket. Abrufbar unter `http://crd.lbl.gov/~dhbailey/mpdist/`

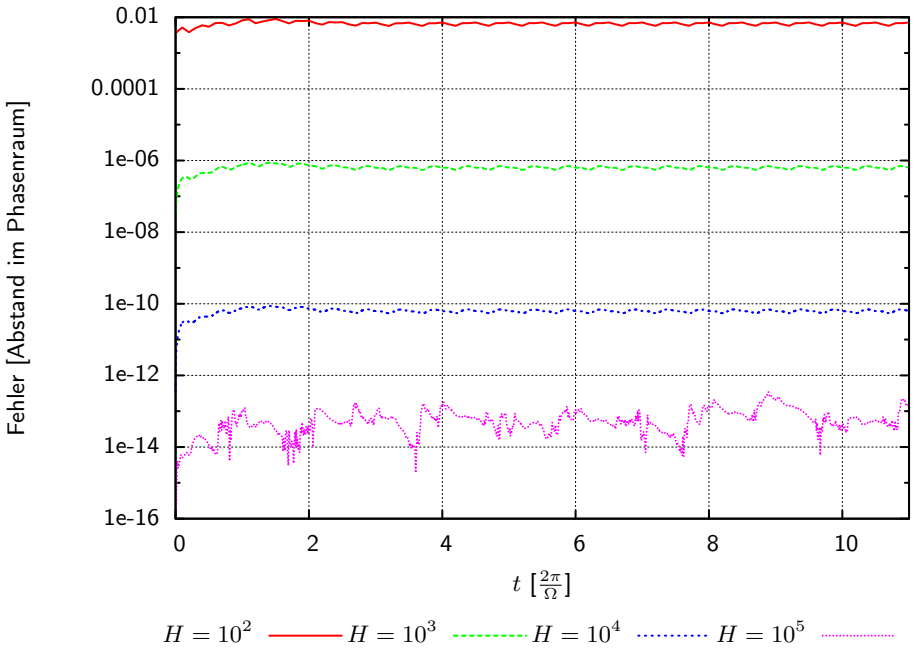

Abbildung 7: Fehlerabschätzung des Runge-Kutta-Verfahrens für verschiedene Werte für H bei $f_c = 0.5$

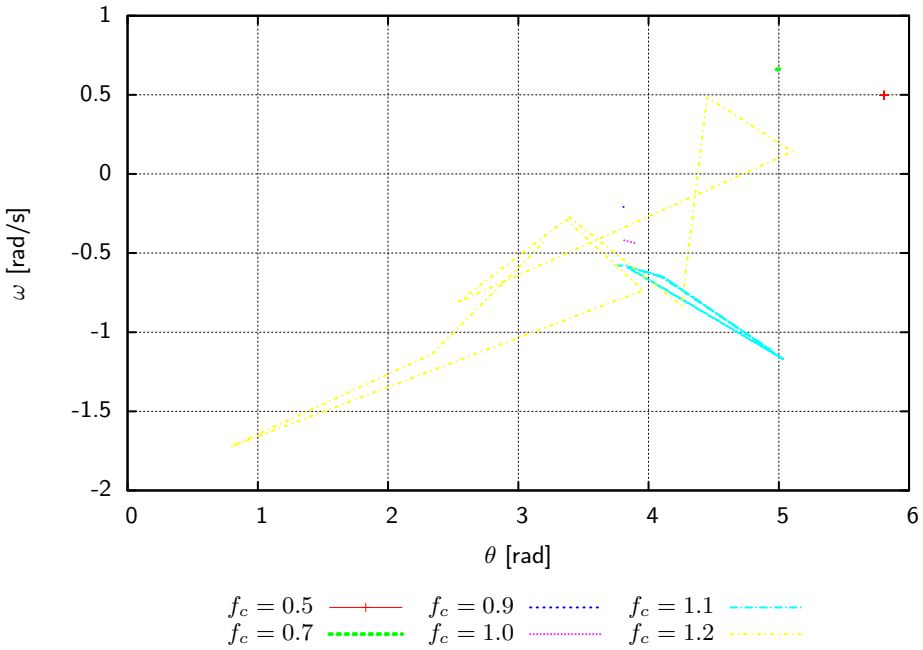

Abbildung 8: Trajektorien im Phasenraum