Physikalisches Pendel

November 26, 2022

Notebook erstellt von A. Naber am 25.11.2022

1 Physikalisches Pendel und deterministisches Chaos

1.1 Einleitung

In diesem Notebook wird die Bewegung des gedämpften physikalischen Pendels mit äußerer periodischer Antriebskraft numerisch gelöst. Interessant sind dabei vor allem die Verhaltensweisen für große Amplituden und insbesondere auch für Überschläge. Diese reichen von einfachen zu komplexeren periodischen Bewegungen bis zu chaotischem Verhalten ohne erkennbare Muster.

Für das verwendete numerische Verfahren bringen wir die Differentialgleichung (DGL) aus der Vorlesung in eine andere Form. Zunächst werden für die Vorfaktoren einfachere Parameter eingeführt:

$$f = \frac{F_0}{ml} \; ; \quad b = \frac{\gamma}{ml} \; ; \quad \omega_0^2 = \frac{g}{l} \quad .$$

Damit wird

$$\frac{\mathrm{d}^2\alpha}{\mathrm{d}t^2} + b\,\frac{\mathrm{d}\alpha}{\mathrm{d}t} + \omega_0^2\,\sin(\alpha) = f\,\sin(\omega t) \quad .$$

Diese DGL zweiter Ordnung wird zur numerischen Berechnung in zwei lineare DGL erster Ordnung zerlegt. Mit

$$\alpha_1 = \alpha \quad \text{und} \quad \alpha_2 = \frac{\mathrm{d}\alpha_1}{\mathrm{d}t}$$

erhalten wir dann

$$\alpha_2 = \frac{\mathrm{d}\alpha_1}{\mathrm{d}t}$$

$$\frac{\mathrm{d}\alpha_2}{\mathrm{d}t} = f\,\sin(\omega t) - b\,\alpha_2 - \omega_0^2\,\sin(\alpha_1) \quad .$$

Diese Gleichungen werden im Skript mit dem Modul `odeint` aus *SciPy* integriert. Die Schrittweite wird darin bestimmt durch ein Array t, welches die diskreten Zeitwerte für die zu berechnenden

Winkel und Winkelgeschwindigkeiten enthält. Je kleiner die Schrittweite zwischen benachbarten Zeiten ist, umso präziser wird die numerische Berechnung, aber umso länger dauert sie dann natürlich auch.

Die erste Python-Zelle des Skripts ("Lösung der DGL") berechnet die Lösungen als Funktion der gegebenen Parameter. Weisen Sie daher vor jeder neuen Berechnung (mittels Shift-Return) den Parametern zunächst die gewünschten Werte zu.

Die darauf folgenden Zellen zur Darstellung der Ergebnisse sind voneinander unabhängig. Sie können sich also darauf beschränken, nur die auszuführen, die für Sie interessant sind, und die anderen überspringen. Wenn Sie die Darstellung z.B. eines Plots verändern wollen, dann können Sie das tun ohne Neuberechnung der Lösung. Das kann sehr viel Zeit sparen, wenn Sie eine kleine Schrittweite für die Zeit und damit eine hohe Genauigkeit verwenden.

1.2 Lösung der Differentialgleichung

[1]:
```
%matplotlib widget
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import odeint
from scipy.fft import fft, fftfreq
from time import perf_counter
import csv
```

[2]:
```
# Definition der Differentialgleichungen für den
# gedämpften Oszillator mit harmonischer Anregung
def pendulum(alpha, t, b, omega0, omega, f):
    dalpha1_dt = alpha[1]
    dalpha2_dt = f * np.cos(omega * t) - b * alpha[1] - (omega0**2) * np.
 ↪sin(alpha[0])
    return dalpha1_dt, dalpha2_dt


# Simulationsparameter
f = 2.7       # Anregungsamplitude
b = 0.22      # Reibungskoeffizient
omega0 = 1.0  # Winkelfrequenz der Resonanz für ungedämpften Oszillator;
              # Nicht ändern: bewirkt nur andere Zeitskala der Ergebnisse
omega = 1.0   # Winkelfrequenz der Anregung;
              # wähle omega=omega0 wenn chaotisches Verhalten gewünscht ist

# Anfangsbedingungen [Winkelauslenkung, Winkelgeschwindigkeit]
alpha_0 = [0, 0]

# Zeitintervall der Integration
```

```
n_int = 100      # Anzahl der Zeitwerte für eine Periode; > 1000 für hohe␣
 ↪Qualität (begrenzt durch Speicherplatz)
n_cycles = 5000  # Anzahl der Perioden; 50000 für hohe Qualität der␣
 ↪Poincaré-Map
N = n_int * n_cycles
tmax = (2 * np.pi / omega) * n_cycles
t = np.linspace(0, tmax, N)  # Das entsprechende Zeitinervall als array

# Lösen der Differentialgleichung
t0 = perf_counter()  # timer
alpha = odeint(pendulum, alpha_0, t, args=(b, omega0, omega, f)).T
print(f"Ausführungszeit: {perf_counter() - t0}")
```

```
Ausführungszeit: 4.765115239
```

1.3 Darstellung der Ergebnisse

1.3.1 Winkelamplitude und Winkelgeschwindigkeit als Funktion der Zeit

Im Folgenden werden zunächst der Winkel α und die Winkelgeschwindigkeit $\dot{\alpha}$ als Funktion der Zeit geplottet. Meist genügt es hier, einen vergleichsweise kurzen Zeitraum darzustellen, um z.B. ein periodisches Muster zu erkennen. Chaotisches Verhalten kann dagegen eventuell erst für vergleichsweise große Zeiträume identifiziert werden.

Öndern Sie in der folgenden Zelle die Limits der Zeitachse für die Plots, um diese Ihren Vorstellungen anzupassen. Wenn Sie die Werte in einem anderen Programm verwenden möchten, geben Sie für `filename` einen Namen ein, dann werden die Daten als csv-Datei gespeichert.

Hinweis: Die Darstellung der Winkelamplitude kann mit dem Parameter `reduce_to_2pi = True` so geändert werden, dass alle Winkel auf den Bereich $\pm\pi$ reduziert werden. So können die Schwingungen besser visualisiert werden, aber die Folge von Überschlägen in eine Richtung ist nicht mehr erkennbar. Für die Fouriertransformation sollte immer `reduce_to_2pi = False` gewählt werden.

[3]:
```
xlim = (0, tmax / 10)  # Hier können Sie das zu plottende Zeitinterval ändern

reduce_to_2pi = False
filename = ""  # angeben, um Daten als .csv zu speichern

# Die Winkel alpha1 werden auf den Bereich -pi bis +pi reduziert
# ACHTUNG: verändert Ergebnisse von FFT
if reduce_to_2pi:
    alpha[0, :] = alpha[0] - 2 * np.pi * np.rint(alpha[0] / (2 * np.pi))

if filename:
```

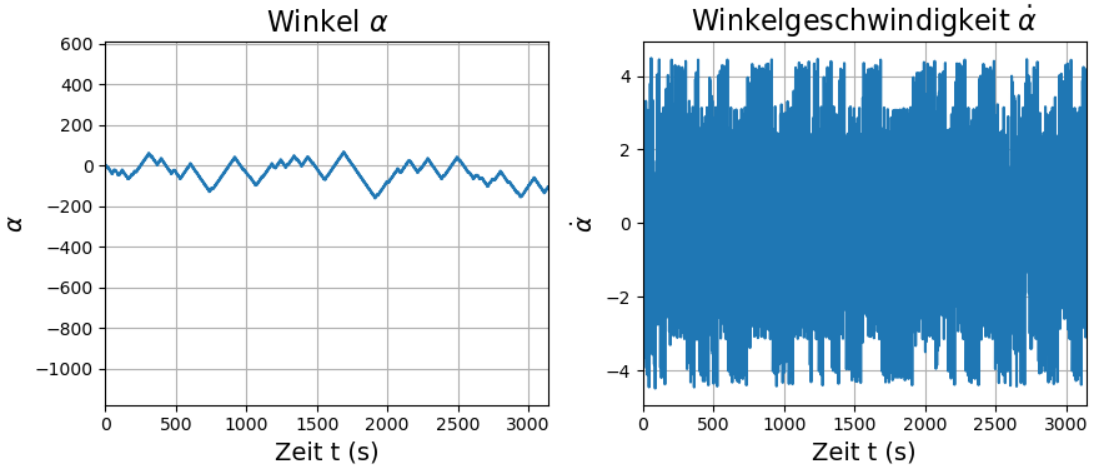
```
    with open(filename, "w", newline="") as f:
        writer = csv.writer(f)
        header = ["time", "amplitude", "angular velocity"]
        writer.writerow(header)
        for i in range(N):
            writer.writerow([t[i], alpha[0, i], alpha[1, i]])

# Plotting
plt.rcParams["axes.grid"] = True
plt.rcParams["axes.labelsize"] = 14
plt.rcParams["axes.titlesize"] = 16

fig, ax = plt.subplots(1, 2, figsize=(9, 4))

ax[0].plot(t, alpha[0])
ax[0].set_title(r"Winkel $\alpha$")
ax[0].set_ylabel(r"$\alpha$")
ax[1].plot(t, alpha[1])
ax[1].set_title(r"Winkelgeschwindigkeit $\dot{\alpha}$")
ax[1].set_ylabel(r"$\dot{\alpha}$")

plt.setp(ax, xlim=xlim, xlabel="Zeit t (s)")
fig.tight_layout()
plt.show()
```



1.3.2 Fouriertransformation der Winkelamplitude

Die Fouriertransformierte der Winkelamplitude bestimmt deren Frequenzkomponenten. Im Fall schwach gedämpfter Eigenschwingungen des Pendels für kleine Amplituden und ohne externe Anregung reduziert sich das Frequenzspektrum auf die Eigenfrequenz f_0 des harmonischen Oszillators. Für größere Amplituden und nichtlinearer Rückstellkraft wird die zunächst reine Sinusfunktion zunehmend "verzerrt" - die Eigenfrequenz f_0 nimmt ab und es gibt zusätzliche Frequenzkomponenten (ungerade Harmonische, also $3f_0$, $5f_0$, etc.). Im Falle externer Anregung und Überschlägen des Pendels wird das Frequenzspektrum nochmal komplexer, insbesondere wenn die Amplitude als Funktion der Zeit für spezielle Parameter aperiodisch und "chaotisch" wird.

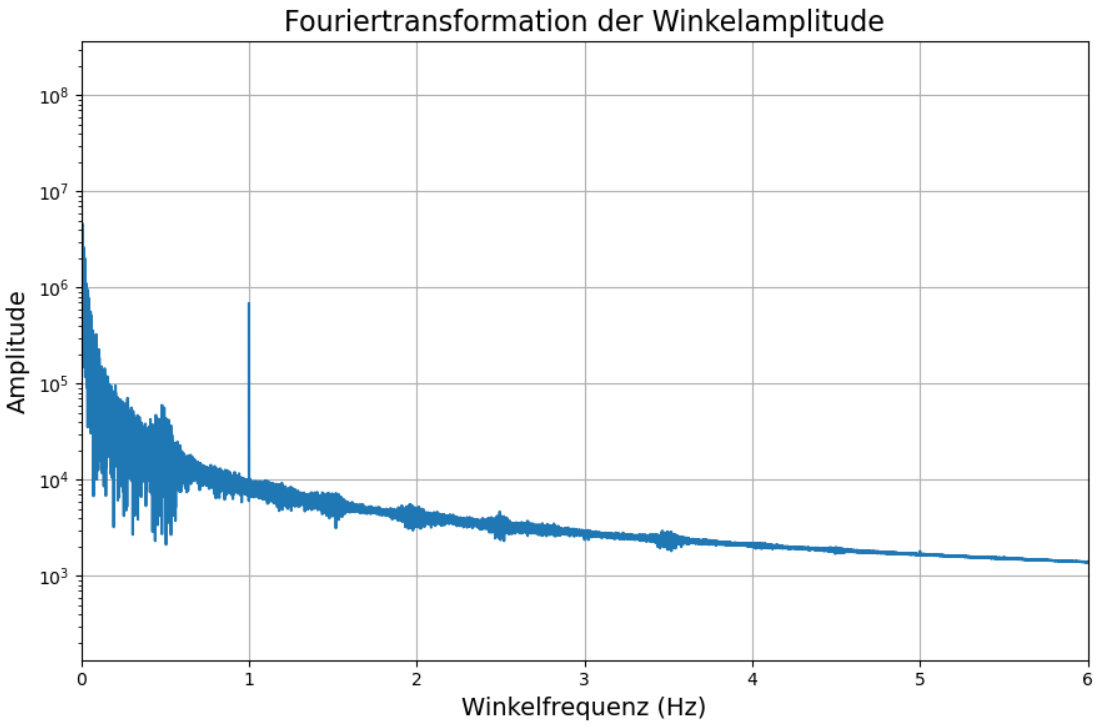
[4]:
```
filename = ""

# FFT
yf = fft(alpha[0])
xf = fftfreq(N, tmax / N)
# Sortiere Freq
yf = np.fft.fftshift(yf)
xf = 2 * np.pi * np.fft.fftshift(xf)

mag = np.abs(yf)

if filename != "":
    with open(filename, "w", newline="") as f:
        writer = csv.writer(f)
        header = ["frequency", "amplitude"]
        writer.writerow(header)
        for i in range(N):
            writer.writerow([xf[i], mag[i]])

fig, ax = plt.subplots(1, 1, figsize=(9, 6))
ax.plot(xf, mag)
ax.set_xlabel("Winkelfrequenz (Hz)")
ax.set_ylabel("Amplitude")
ax.set_title("Fouriertransformation der Winkelamplitude")
ax.set_yscale("log")
ax.set_xlim(0, 6)
fig.tight_layout()
plt.show()
```



1.3.3 Poincaré-Map

Hier wird die Winkelgeschwindigkeit $\dot{\alpha}$ aufgetragen gegen die Winkelamplitude α für Zeiten $t_n = n \cdot T$, worin T die Periodendauer der externen Anregung ist. Falls das Pendel periodisch mit der Anregungsfrequenz schwingt, was bei moderater Anregung ohne Überschläge meist der Fall ist, befindet sich das Pendel zu den Zeiten t_n im selben Zustand, d.h. in der Poincaré-Map wird nach dem Einschwingvorgang immer nur der gleiche Punkt erzeugt. Eine komplexe Verteilung von Punkten in der Poincaré-Map ist daher ein Indiz für chaotisches Verhalten. Berechnet man die Map mit hoher zeitlicher Auflösung (kleine zeitliche Iterationsschritte, viele Anregungszyklen), dann wird eine Feinstruktur in den Punkten erkennbar - Linien, die sich zu immer feineren Linien aufspalten. In dem scheinbar chaotischen Verhalten werden erstaunliche Regelmäßigkeiten erkennbar. Durch den limitierten Speicherplatz sind den dafür nötigen Berechnungen leider enge Grenzen gesetzt.

Experimentieren Sie mit den Parametern! Als Startwerte sind z.B. $b = 0.22$ und $f = 2.7$ geeignet. Zoomen Sie in "turbulente" Abschnitte der Poincaré-Map hinein.

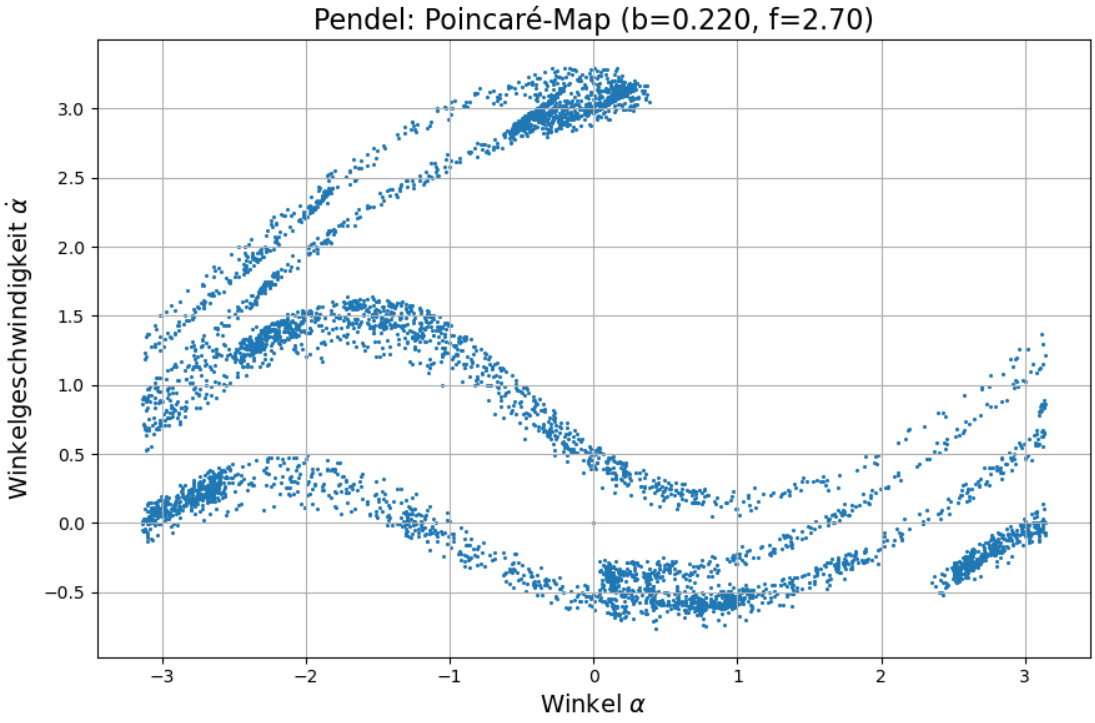
Hinweis: In der Voreinstellung werden die Punkte großdargestellt, um sie gut erkennbar zu machen. Wenn Sie sehr viele Zyklen berechnen und damit sehr viele Punkte in der Map erzeugen, dann sollten Sie die Punktgröße mittels `s = 1` auf einen einzigen Pixel verringern.

[5]:
```
# Für Poincaré-Map werden Werte mit Zeitabstand von genau einer␣
↪Periodendauer T gewählt
```

```
cycles = n_int * np.arange(n_cycles)
al0_cyc = alpha[0][cycles]
al1_cyc = alpha[1][cycles]

# Falls noch nicht geschehen, werden Winkel auf den Bereich -pi bis +pi␣
 ↪reduziert
if not reduce_to_2pi:
    al0_cyc[:] = al0_cyc - 2 * np.pi * np.rint(al0_cyc / (2 * np.pi))

fig, ax = plt.subplots(1, 1, figsize=(9, 6))
ax.scatter(al0_cyc, al1_cyc, marker=".", lw=0, s=20)
ax.set_xlabel(r"Winkel $\alpha$")
ax.set_ylabel(r"Winkelgeschwindigkeit $\dot{\alpha}$")
ax.set_title(f"Pendel: Poincaré-Map (b={b:.3f}, f={f:.2f})")
fig.tight_layout()
plt.show()
```




```
[ ]:
```