Physikalisches Pendel_2

December 8, 2022

Notebook erstellt von A. Naber am 25.11.2022, modifiziert von Y. Augenstein

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

1.2.1 Setup

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

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

1.2.2 Funktionen

[2]:
```
# Definition der Differentialgleichungen für den gedämpften Oszillator mit␣
↪harmonischer Anregung
def pendulum(alpha, t, b, omega, f, omega0=1.0):
    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


def run_pendulum(t, omega, A, b, f):
    # Anfangsbedingungen [Winkelauslenkung, Winkelgeschwindigkeit]
    alpha_0 = [A, 0]
```

```
    # Lösen der Differentialgleichung
    alpha = odeint(pendulum, alpha_0, t, args=(b, omega, f)).T
    return alpha


def spectrum(x, tmax):
    xf = fftfreq(len(x), tmax / len(x))
    xf = 2 * np.pi * np.fft.fftshift(xf)
    yf = np.abs(np.fft.fftshift(fft(x)))
    return xf, yf


def wrap2pi(x):
    return x - 2 * np.pi * np.rint(x / (2 * np.pi))


def make_broken_axis(ax1, ax2):
    ax1.spines["right"].set_visible(False)
    ax1.grid(False)
    ax2.spines["left"].set_visible(False)
    ax2.set_yticks([])
    ax2.set_yticklabels([])
    ax2.grid(False)

    d = 0.02
    kwargs = dict(transform=ax1.transAxes, color="k", clip_on=False)
    ax1.plot((1 - d, 1 + d), (1 - d, 1 + d), **kwargs)
    ax1.plot((1 - d, 1 + d), (-d, +d), **kwargs)
    kwargs.update(transform=ax2.transAxes)
    ax2.plot((-d, +d), (1 - d, 1 + d), **kwargs)
    ax2.plot((-d, +d), (-d, +d), **kwargs)
```

1.2.3 Simulationsparameter

[3]:
```
omega = 1.0      # Anregungsfrequenz
n_int = 100      # Anzahl der Zeitwerte für eine Periode; > 1000 für hohe␣
 ↪Qualität (begrenzt durch Speicherplatz)
n_cycles = 500  # Anzahl der Perioden; 50000 für hohe Qualität der␣
 ↪PoincarÃ¯-Map
N = n_int * n_cycles
tmax = (2 * np.pi / omega) * n_cycles
t = np.linspace(0, tmax, N)  # Das entsprechende Zeitinervall als array
```

1.3 Darstellung der Ergebnisse

1.3.1 Obere Reihe: Winkelamplitude und Winkelgeschwindigkeit als Funktion der Zeit

Hier werden zunächst der Winkel α und die Winkelgeschwindigkeit $\dot{\alpha}$ als Funktion der Zeit geplottet. Meist genügt es, 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. Daher zeigen wir hier den Anfang sowie das Ende der Schwingung, wobei die Anzahl der angezeigten Zeitpunkte hier mit dem Parameter `n` frei gewählt werden kann.

1.3.2 Unten links: 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 nichtlineare 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.

1.3.3 Unten rechts: PoincarÃľ-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 PoincarÃľ-Map wird nach dem Einschwingvorgang immer nur der gleiche Punkt erzeugt. Eine komplexe Verteilung von Punkten in der PoincarÃľ-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 PoincarÃľ-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.

[8]:
```
A = [0, 1e-3, -1e-3]  # muss eine Liste sein, da wir mehrere Werte angeben␣
↪können
b = 0.22
f = 2.68
```

```
alphas = [run_pendulum(t, omega, Ai, b, f) for Ai in A]

spectra = []
poincare_maps = []
for alpha in alphas:
    spectra.append(spectrum(alpha[0], tmax))

    # Für die PoincarÃľ-Map
    cycles = n_int * np.arange(n_cycles)
    al0_cyc = wrap2pi(alpha[0][cycles])
    al1_cyc = alpha[1][cycles]
    poincare_maps.append([al0_cyc, al1_cyc])

n = 600  # Anzahl der Zeitpunkte, die angezeigt werden sollen

# Wir bereiten alle Achsen zum plotten vor
plt.close("all")
fig = plt.figure(figsize=(10, 8), dpi=100)
gs = fig.add_gridspec(2, 2, wspace=0.3, hspace=0.3)
gs1 = gs[0, 0].subgridspec(1, 2, wspace=0.05)
ax11 = fig.add_subplot(gs1[0])
ax11.set_xlabel(r"t (s)", x=1.05)
ax11.set_ylabel(r"Winkel $\alpha$")
ax12 = fig.add_subplot(gs1[1])
make_broken_axis(ax11, ax12)

gs2 = gs[0, 1].subgridspec(1, 2, wspace=0.05)
ax21 = fig.add_subplot(gs2[0])
ax21.set_xlabel(r"t (s)", x=1.05)
ax21.set_ylabel(r"Winkelgeschw. $\dot{\alpha}$")
ax22 = fig.add_subplot(gs2[1])
make_broken_axis(ax21, ax22)

ax3 = fig.add_subplot(gs[1, 0])
ax3.set_xlabel("Winkelfrequenz (Hz)")
ax3.set_ylabel("Amplitude")
ax3.set_yscale("log")
ax3.set_xlim(0, 4)

ax4 = fig.add_subplot(gs[1, 1])
ax4.set_xlabel(r"Winkel $\alpha$")
ax4.set_ylabel(r"Winkelgeschw. $\dot{\alpha}$")
ax4.axis("equal")

# Wir plotten die Daten in die entsprechenden Achsen
```

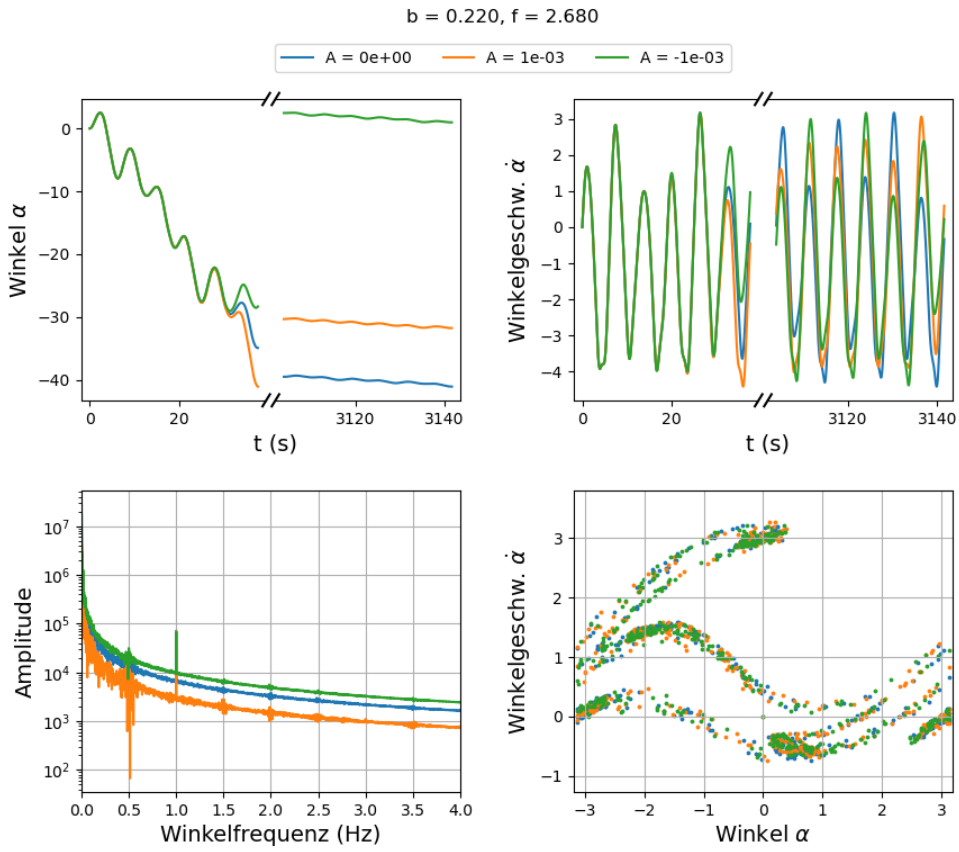
```
for Ai, alpha, (xf, mag), (al0_cyc, al1_cyc) in zip(A, alphas, spectra,␣
 ↪poincare_maps):
    ax11.plot(t[:n], alpha[0][:n])
    ax12.plot(t[-n:], alpha[0][-n:], label=f"A = {Ai:.0e}")

    ax21.plot(t[:n], alpha[1][:n])
    ax22.plot(t[-n:], alpha[1][-n:])

    ax3.plot(xf, mag)

    ax4.scatter(al0_cyc, al1_cyc, marker=".", lw=0, s=30)
    ax4.set_xlim(min(al0_cyc) - 0.05, max(al0_cyc) + 0.05)
    ax4.set_ylim(min(al1_cyc) - 0.05, max(al1_cyc) + 0.05)

fig.suptitle(f"b = {b:.3f}, f = {f:.3f}")
fig.legend(
    ncol=len(A),
    loc="upper center",
    bbox_to_anchor=(0.5, 0.95),
    bbox_transform=fig.transFigure,
)
plt.show()
```



1.3.4 Speichern der Daten

Falls erwünscht, können wir die hier berechneten Ergebnisse in einer Datei abspeichern.

[5]:
```
def file_writer(filename, data, labels):
    with open(filename, "w", newline="") as f:
        writer = csv.writer(f)
        writer.writerow(labels)
        for row in zip(*data):
            writer.writerow(row)


# for Ai, alpha, (xf, mag) in zip(A, alphas, spectra):
#     file_writer(
#         f"alpha_A={Ai:.0e}.csv",
#         [t, *alpha],
#         ["time", "amplitude", "angular velocity"],
```

```
#     )
#     file_writer(
#         f"spectrum_A={Ai:.0e}.csv",
#         [xf, mag],
#         ["frequency", "amplitude"],
#     )
```

[6]:
```
print(plt.__version__)
```

[ ]:
```

```