Beispiele

Beispiel 1: Phasenebene des ungedämpften harmonischen Oszillators

Der ungedämpfte harmonische Oszillator ist das einfachste Beispiel eines zweidimensionalen dynamischen Systems. Seine Phasenebene zeigt charakteristische geschlossene Trajektorien, die die Energieerhaltung des Systems widerspiegeln.

Problemstellung

Wir betrachten einen ungedämpften harmonischen Oszillator mit Masse $m = 1\,\mathrm{kg}$ und Federkonstante $k = 1\,\mathrm{N/m}$. Das System wird aus verschiedenen Anfangszuständen gestartet. Wir möchten die Trajektorien in der Phasenebene visualisieren und verstehen, wie sie mit der Energie des Systems zusammenhängen.

Mathematisches Modell

Das System lautet:

$$\begin{aligned} \dot{y}_1 &= y_2 \\ \dot{y}_2 &= -\omega_0^2 y_1 \end{aligned}$$

mit $\omega_0 = \sqrt{k/m} = 1\,\mathrm{rad/s}$. Die analytische Lösung ist bekannt:

$$\begin{aligned} y_1(t) &= y_{1,0}\cos(\omega_0 t) - \frac{y_{2,0}}{\omega_0}\sin(\omega_0 t) \\ y_2(t) &= y_{1,0}\omega_0 \sin(\omega_0 t) + y_{2,0}\cos(\omega_0 t) \end{aligned}$$

Dabei sind $y_{1,0}$ und $y_{2,0}$ die Anfangswerte von Auslenkung und Geschwindigkeit.

Interpretation

Die Trajektorien in der Phasenebene sind Ellipsen, deren Form durch die Energie des Systems bestimmt wird. Jede Ellipse entspricht einem festen Energieniveau. Das System durchläuft die Ellipse periodisch mit der Periode $T = 2\pi/\omega_0$.

Die Form der Ellipsen ergibt sich aus der Energieerhaltung. Die Gesamtenergie ist:

$$E = \frac{1}{2}my_2^2 + \frac{1}{2}ky_1^2 = \text{const}$$

In der Phasenebene sind dies Ellipsen mit Halbachsen, die von der Anfangsenergie abhängen. Alle Trajektorien sind geschlossen, was die periodische Natur des ungedämpften Oszillators widerspiegelt.

```
import numpy as np
import matplotlib.pyplot as plt

# Parameter
k = 1.0  # Federkonstante
m = 1.0  # Masse
omega_0 = np.sqrt(k/m)

# Verschiedene Anfangsbedingungen
initial_conditions = [
    (1.0, 0.0),    # Maximale Auslenkung
    (0.0, 1.0),    # Maximale Geschwindigkeit
    (0.7, 0.7),    # Diagonal
    (0.5, -0.5),   # Andere Richtung
]

fig, ax = plt.subplots(figsize=(7, 4))

# Zeit für Integration
t = np.linspace(0, 10, 1000)

# Plot Trajektorien für verschiedene Anfangsbedingungen
colors = ['red', 'blue', 'green', 'purple']
for (y1_0, y2_0), color in zip(initial_conditions, colors):
    y1 = y1_0 * np.cos(omega_0 * t) - (y2_0 / omega_0) * np.sin(omega_0 * t)
    y2 = y1_0 * omega_0 * np.sin(omega_0 * t) + y2_0 * np.cos(omega_0 * t)

    # Elliptische Trajektorie
    ax.plot(y1, y2, color=color, linewidth=2, label=f'$(y_1(0), y_2(0)) = ({y1_0}, {y2_0})$'
```

```
    # Startpunkt markieren
    ax.plot(y1_0, y2_0, 'o', color=color, markersize=8)

    # Pfeilrichtung zeigen (Vektor an einigen Positionen)
    n_arrows = 5
    indices = np.linspace(0, len(t)-1, n_arrows, dtype=int)
    for idx in indices[:-1]:
        dy1 = (y1[idx+10] - y1[idx]) / 10 if idx+10 < len(t) else (y1[idx+1] - y1[idx])
        dy2 = (y2[idx+10] - y2[idx]) / 10 if idx+10 < len(t) else (y2[idx+1] - y2[idx])
        ax.arrow(y1[idx], y2[idx], dy1*0.08, dy2*0.08,
                 head_width=0.05, head_length=0.05, fc=color, ec=color, alpha=0.6)

ax.set_xlabel('Auslenkung $y_1 = x$', fontsize=13)
ax.set_ylabel(r'Geschwindigkeit $y_2 = \dot{x}$', fontsize=13)
ax.set_title('Phasenebene des ungedämpften harmonischen Oszillators', fontsize=14)
ax.grid(True, alpha=0.3)
ax.axhline(y=0, color='k', linewidth=0.5)
ax.axvline(x=0, color='k', linewidth=0.5)
ax.legend(fontsize=11, loc='upper right')
ax.set_aspect('equal')

plt.tight_layout()
plt.show()
```

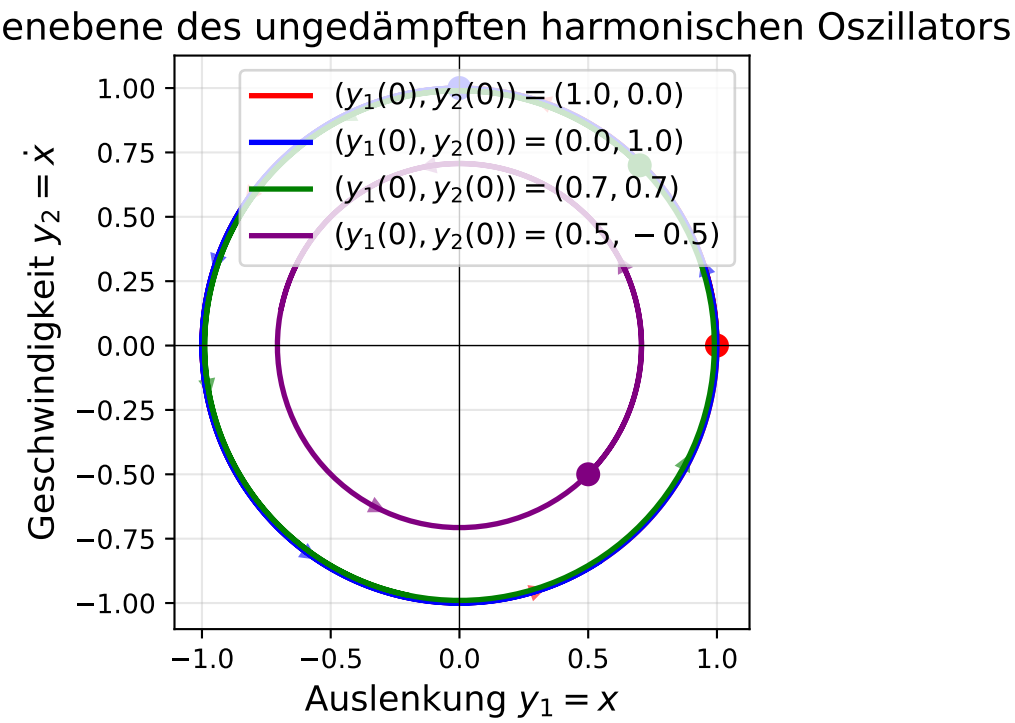

Abbildung 1: Phasenebene des ungedämpften harmonischen Oszillators

Beispiel 2: Gedämpfter Oszillator in der Phasenebene

Die Dämpfung verändert das Verhalten in der Phasenebene fundamental. Anstelle geschlossener Ellipsen entstehen Spiralen, die zum Ursprung konvergieren. Die Form dieser Spiralen hängt stark von der Stärke der Dämpfung ab.

Problemstellung

Wir untersuchen denselben Oszillator wie zuvor, aber mit drei verschiedenen Dämpfungsstärken. Diese entsprechen den drei fundamentalen Dämpfungsregimen: Unterdämpfung, kritische Dämpfung und Überdämpfung.

Mathematisches Modell

Das System lautet:

$$\dot{y}_1 = y_2$$
$$\dot{y}_2 = -\omega_0^2 y_1 - \gamma y_2$$

Für die drei Fälle wählen wir: - Unterdämpfung: $\gamma = 0.2 < \omega_0 = 1$ - Kritische Dämpfung: $\gamma = 1.0 = \omega_0$ - Überdämpfung: $\gamma = 2.0 > \omega_0$

Interpretation

Bei Unterdämpfung spiralt das System zum Ursprung, wobei es mehrfach um diesen kreist. Die Anzahl der Umläufe hängt von der Dämpfungsstärke ab. Je schwächer die Dämpfung, desto mehr Umläufe werden ausgeführt.

Bei kritischer Dämpfung erreicht das System den Ursprung am schnellsten, ohne zu überschwingen. Dies ist der optimale Fall für viele technische Anwendungen wie Stoßdämpfer oder Messgeräte.

Bei Überdämpfung nähert sich das System dem Ursprung monoton, ohne Oszillationen. Die Annäherung ist jedoch langsamer als im kritisch gedämpften Fall. Das System kriecht gewissermaßen zur Ruhelage.

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

# Parameter
k = 1.0
m = 1.0
omega_0 = np.sqrt(k/m)

# Drei Dämpfungsfälle
gamma_values = [0.2, 1.0, 2.0]  # c/m
labels = ['Unterdämpfung ( =0.2)', 'Kritische Dämpfung ( =1.0)', 'Überdämpfung ( =2.0)']

fig, axes = plt.subplots(1, 3, figsize=(7, 4))

for ax, gamma, label in zip(axes, gamma_values, labels):
    # Differentialgleichung für das System
    def oscillator(t, y):
        y1, y2 = y
        dy1 = y2
```

```
        dy2 = -omega_0**2 * y1 - gamma * y2
        return [dy1, dy2]

    # Verschiedene Anfangsbedingungen
    initial_conditions = [
        (1.0, 0.0),
        (0.0, 1.5),
        (0.8, 0.8),
        (-0.8, 0.8),
        (0.5, -1.0),
    ]

    colors = plt.cm.rainbow(np.linspace(0, 1, len(initial_conditions)))

    for (y1_0, y2_0), color in zip(initial_conditions, colors):
        # Numerisch integrieren
        sol = solve_ivp(oscillator, [0, 10], [y1_0, y2_0], dense_output=True, max_step=0.01)

        # Phasenebene plotten
        ax.plot(sol.y[0], sol.y[1], color=color, linewidth=2, alpha=0.8)
        ax.plot(y1_0, y2_0, 'o', color=color, markersize=8)

    # Ursprung markieren
    ax.plot(0, 0, 'k*', markersize=15, label='Gleichgewicht')

    ax.set_xlabel('Auslenkung $y_1$', fontsize=12)
    ax.set_ylabel('Geschwindigkeit $y_2$', fontsize=12)
    ax.set_title(label, fontsize=12)
    ax.grid(True, alpha=0.3)
    ax.axhline(y=0, color='k', linewidth=0.5)
    ax.axvline(x=0, color='k', linewidth=0.5)
    ax.set_xlim([-1.2, 1.2])
    ax.set_ylim([-1.5, 1.5])

plt.tight_layout()
plt.show()
```

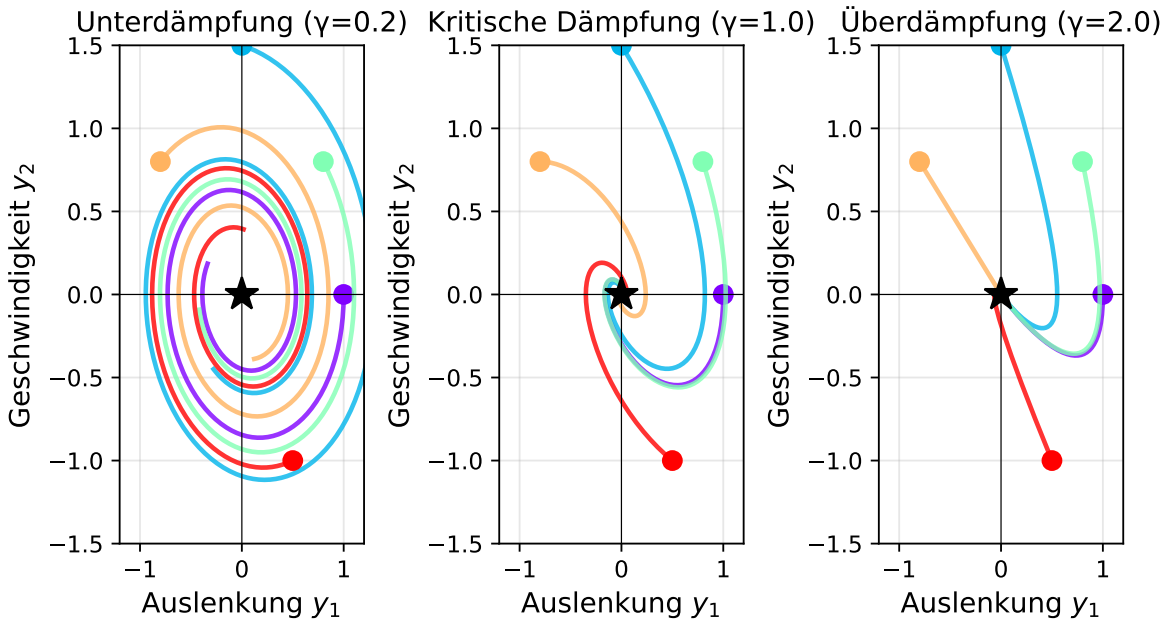

Abbildung 2: Phasenebene des gedämpften harmonischen Oszillators

Beispiel 3: Der RLC-Schaltkreis

Elektrische Schaltkreise mit Widerstand, Spule und Kondensator bilden ein wichtiges Anwendungsgebiet für Systeme von Differentialgleichungen. Die mathematische Beschreibung ist strukturell identisch mit dem mechanischen Oszillator, was eine tiefe Analogie zwischen beiden Systemen offenbart.

Problemstellung

Ein RLC-Schaltkreis besteht aus einem Widerstand R, einer Induktivität L und einer Kapazität C in Reihenschaltung. Wir möchten das zeitliche Verhalten von Ladung und Strom als System von Differentialgleichungen erster Ordnung beschreiben.

Mathematische Modellierung

Das Kirchhoffsche Spannungsgesetz besagt, dass die Summe aller Spannungen im Kreis gleich der angelegten Spannung $U(t)$ ist:

$$U(t) = U_R + U_L + U_C$$

Die einzelnen Spannungen sind: - Widerstand: $U_R = RI$ nach dem Ohmschen Gesetz - Induktivität: $U_L = L\frac{\mathrm{d}I}{\mathrm{d}t}$ für die Spule - Kapazität: $U_C = Q/C$ für den Kondensator

Da der Strom die Ableitung der Ladung ist, $I = \mathrm{d}Q/\mathrm{d}t$, erhalten wir eine Differentialgleichung zweiter Ordnung:

$$L\frac{\mathrm{d}^2Q}{\mathrm{d}t^2} + R\frac{\mathrm{d}Q}{\mathrm{d}t} + \frac{Q}{C} = U(t)$$

Um dies als System erster Ordnung zu schreiben, definieren wir:

$$y_1 = Q \quad \text{(Ladung)}, \quad y_2 = I = \dot{Q} \quad \text{(Strom)}$$

Damit ergibt sich:

$$\begin{aligned} \dot{y}_1 &= y_2 \\ \dot{y}_2 &= -\frac{1}{LC}y_1 - \frac{R}{L}y_2 + \frac{1}{L}U(t) \end{aligned}$$

Analogie zum mechanischen Oszillator

Der Vergleich mit dem gedämpften Oszillator zeigt eine bemerkenswerte Entsprechung:

Mechanisches System	Elektrisches System
Masse m	Induktivität L
Dämpfung c	Widerstand R
Federkonstante k	reziproke Kapazität $1/C$
Auslenkung x	Ladung Q
Geschwindigkeit $\dot{x}$	Strom I
Kraft F	Spannung U

Die mathematische Struktur beider Systeme ist identisch. Erkenntnisse über mechanische Oszillatoren lassen sich direkt auf elektrische Schaltkreise übertragen und umgekehrt.

Der LC-Schaltkreis

Ein besonders einfacher Fall ist der LC-Schaltkreis ohne Widerstand ($R = 0$). Dieser entspricht dem ungedämpften Oszillator. Das System lautet:

$$\begin{aligned} \dot{y}_1 &= y_2 \\ \dot{y}_2 &= -\frac{1}{LC}y_1 \end{aligned}$$

Die Eigenfrequenz ist $\omega_0 = 1/\sqrt{LC}$. Ladung und Strom oszillieren harmonisch. Die Energie pendelt zwischen dem elektrischen Feld im Kondensator ($\frac{1}{2}Q^2/C$) und dem magnetischen Feld in der Spule ($\frac{1}{2}LI^2$).

Die Trajektorien in der Phasenebene (Q, I) sind Ellipsen, analog zum mechanischen Oszillator. Jede Ellipse entspricht einem festen Energieniveau.

Der RLC-Schaltkreis mit Dämpfung

Mit Widerstand $R > 0$ tritt Dämpfung auf. Die Energie wird in Wärme dissipiert. Je nach Verhältnis von R, L und C ergeben sich drei Regime:

- Unterdämpfung ($R < 2\sqrt{L/C}$): Gedämpfte Oszillationen
- Kritische Dämpfung ($R = 2\sqrt{L/C}$): Schnellste Rückkehr ohne Überschwingen
- Überdämpfung ($R > 2\sqrt{L/C}$): Exponentieller Abfall ohne Oszillation

Diese Klassifikation ist identisch mit den Dämpfungsregimen des mechanischen Oszillators. Die Phasenraumtrajektorien zeigen dasselbe Verhalten: Spiralen bei Unterdämpfung, direkte Annäherung zum Ursprung bei Überdämpfung.

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

fig, axes = plt.subplots(1, 2, figsize=(7, 4))

# === Linke Seite: LC-Schaltkreis (ungedämpft) ===
ax = axes[0]
L, C = 1.0, 1.0
omega_0 = 1.0 / np.sqrt(L * C)

# Verschiedene Anfangsbedingungen
initial_conditions = [
    (1.0, 0.0),    # Maximale Ladung
    (0.0, 1.0),    # Maximaler Strom
    (0.7, 0.7),    # Diagonal
    (0.5, -0.5),   # Andere Richtung
]

t = np.linspace(0, 10, 1000)
colors = ['red', 'blue', 'green', 'purple']

for (Q0, I0), color in zip(initial_conditions, colors):
```

```
    Q = Q0 * np.cos(omega_0 * t) - (I0 / omega_0) * np.sin(omega_0 * t)
    I = Q0 * omega_0 * np.sin(omega_0 * t) + I0 * np.cos(omega_0 * t)

    ax.plot(Q, I, color=color, linewidth=2, label=f'$(Q_0, I_0) = ({Q0}, {I0})$')
    ax.plot(Q0, I0, 'o', color=color, markersize=8)

ax.set_xlabel('Ladung $Q(t)$ [C]', fontsize=12)
ax.set_ylabel('Strom $I(t)$ [A]', fontsize=12)
ax.set_title('LC-Schaltkreis (konservativ)', fontsize=12)
ax.grid(True, alpha=0.3)
ax.axhline(y=0, color='k', linewidth=0.5)
ax.axvline(x=0, color='k', linewidth=0.5)
ax.legend(fontsize=10, loc='upper right')
ax.set_aspect('equal')

# === Rechte Seite: RLC-Schaltkreis mit verschiedenen Dämpfungen ===
ax = axes[1]
L, C = 1.0, 1.0
omega_0 = 1.0 / np.sqrt(L * C)

# Für RLC-Schaltkreis (nur eine Subgrafik gezeigt)
R = 0.5  # Leichte Dämpfung für Demonstration
def rlc_circuit(t, y):
    Q, I = y
    dQ = I
    dI = -(1 / (L * C)) * Q - (R / L) * I
    return [dQ, dI]

initial_conditions = [
    (1.0, 0.0),
    (0.0, 1.5),
    (0.7, 0.7),
    (0.5, -1.0),
]

colors_rlc = plt.cm.viridis(np.linspace(0, 1, len(initial_conditions)))

for (Q0, I0), color in zip(initial_conditions, colors_rlc):
    sol = solve_ivp(rlc_circuit, [0, 15], [Q0, I0], dense_output=True, max_step=0.01)
    ax.plot(sol.y[0], sol.y[1], color=color, linewidth=2, alpha=0.8)
    ax.plot(Q0, I0, 'o', color=color, markersize=8, label=f'$(Q_0, I_0) = ({Q0}, {I0})$')
```

```
ax.plot(0, 0, 'k*', markersize=15, label='Gleichgewicht')

ax.set_xlabel('Ladung $Q(t)$ [C]', fontsize=12)
ax.set_ylabel('Strom $I(t)$ [A]', fontsize=12)
ax.set_title(f'RLC-Schaltkreis mit Dämpfung (R={R})', fontsize=12)
ax.grid(True, alpha=0.3)
ax.axhline(y=0, color='k', linewidth=0.5)
ax.axvline(x=0, color='k', linewidth=0.5)
ax.legend(fontsize=9, loc='upper right')
ax.set_xlim([-1.2, 1.2])
ax.set_ylim([-1.5, 1.5])

plt.tight_layout()
plt.show()
```

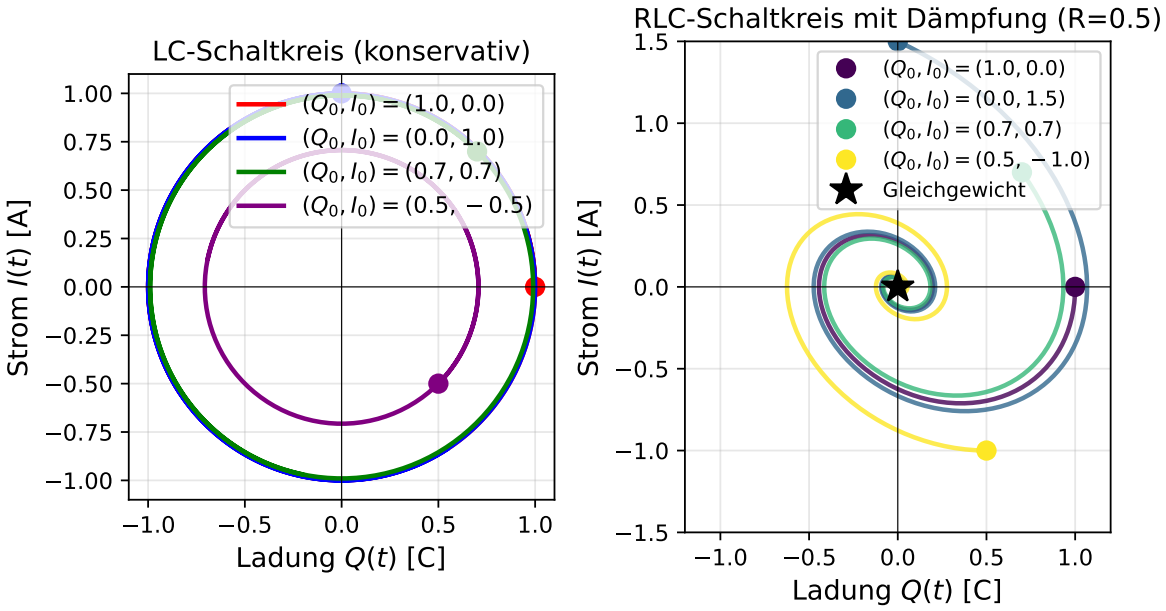

Abbildung 3: Phasenebene eines LC-Schaltkreises (links) und eines RLC-Schaltkreises mit verschiedenen Dämpfungen (rechts)

Der RC-Schaltkreis

Ein einfacherer Fall ist der RC-Schaltkreis, bei dem die Induktivität fehlt ($L = 0$). Die Differentialgleichung reduziert sich auf:

$$R\frac{\mathrm{d}Q}{\mathrm{d}t} + \frac{Q}{C} = U(t)$$

Dies ist bereits eine Gleichung erster Ordnung und benötigt keine Reduktion. Für konstante Spannung U_0 lautet die Lösung:

$$Q(t) = CU_0(1 - e^{-t/(RC)})$$

Die Zeitkonstante $\tau = RC$ charakterisiert die Geschwindigkeit des Ladevorgangs. Nach einer Zeitkonstante hat der Kondensator etwa 63 Prozent seiner Endladung erreicht. Der RC-Schaltkreis hat nur eine Zustandsvariable und benötigt daher keine Phasenebene.