TPIb_Uebung04_coding_SS25

May 5, 2025

1 Theoretische Physik Ib - Analytische Mechanik

Marc Thome, Philipp Hövel, Saarland University
Date: **May 05, 2025**

1.0.1 Tasks

- Getting started with Python
- Pendulum equation

1.0.2 Importing helpful Python modules

```
In [1]: import numpy as np
        import matplotlib.pyplot as plt

        from scipy.integrate import odeint
```

1.0.3 Defining the right-hand side of the differential equation

Equations of motion:

$$\ddot{\varphi} = -\omega^2 \sin\varphi$$

```
In [2]: def derivatives_pendulum(x, t, omega):
            # x contains the values [phi, d/dt phi]
            phi, phi_dot = x

            # returns [d/dt phi, d^2/dt^2 phi]
            return [phi_dot, -omega**2 * np.sin(phi)]
```

1.0.4 Defining the solution for the small-angle approximation

```
In [3]: def solution_small_angle_approx(t, omega, amp):
            return amp * np.sin(omega * t), amp * omega * np.cos(omega * t)
```

1.0.5 Here comes the main part

1.0.6 Define parameters

In [4]:
```
# define Parameters
g = 9.81
# pendulum length
L = 5.0
# pendulum frequency
omega = np.sqrt(g / L)
# period length
T = 2.0 * np.pi / omega

# amplitude of the oscillation
init_angle = 15.0   # degrees
# init_angle = 80.0  # degrees (this angle will violate the small angle app
A = np.pi / 180.0 * init_angle # radians
# initial values [phi, d/dt phi]
x_init = [0.0, omega * A]
```

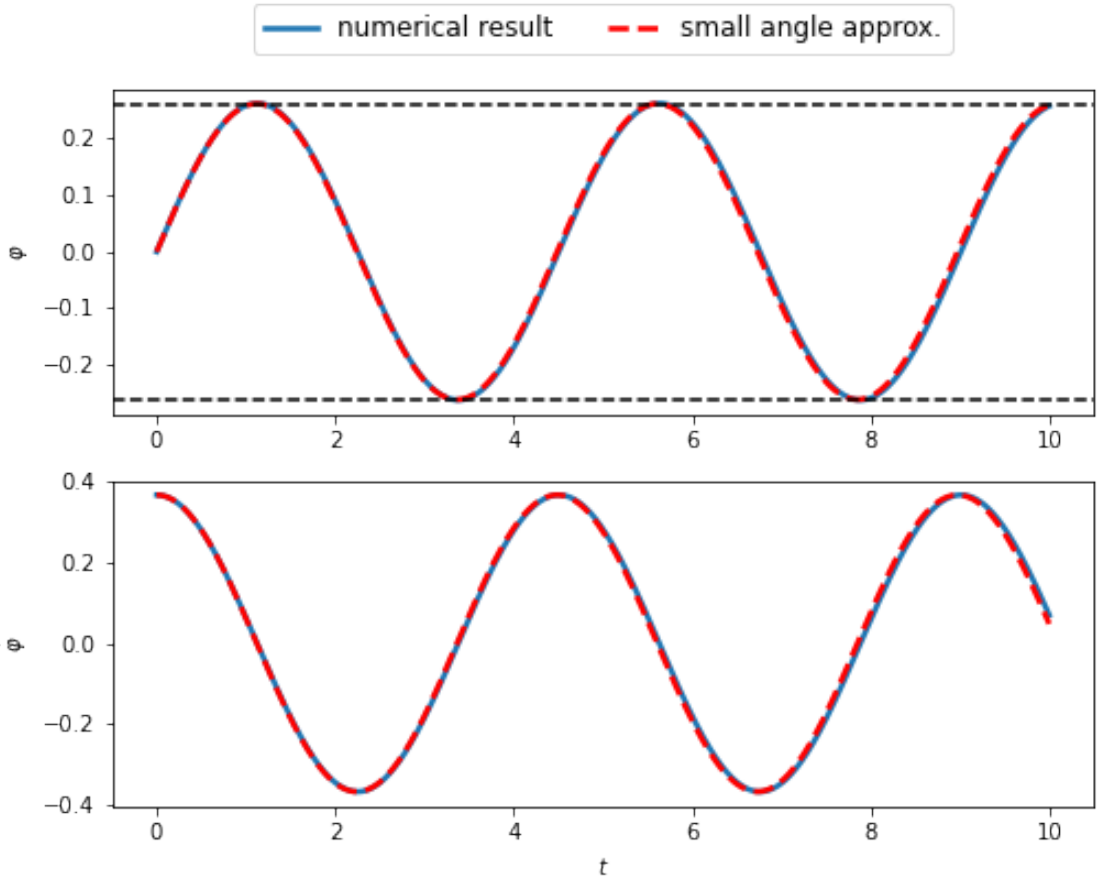
1.0.7 Let the magic begin

In [5]:
```
# this cell does the integration of out equations of motion by calling odei

# time axis
time = np.arange(0.0, 10.0, 1e-3)
# phi and d/dt phi for the small angle approximation
sas, sas_dot = solution_small_angle_approx(time, omega, A)

# integrate the exact equations of motion numerically
result = odeint(derivatives_pendulum, x_init, time, args=(omega,))
# split solution array into two arrays
phi, phi_dot = result.transpose()
```

1.0.8 Plotting the time series

In [6]:
```
# plot the results in two panels underneath each other
fig, axs = plt.subplots(nrows=2, ncols=1, figsize=(8,6))
# plot the numerical result for phi
axs[0].set_ylabel(r'$\varphi$')
axs[0].plot(time, phi, lw=2.5)
# plot the small angle approximation as red dashes line
axs[0].plot(time, sas, '--r', lw=2.5)

axs[0].legend(['numerical result', 'small angle approx.'],
              fontsize=12, ncol=2, loc='upper center', bbox_to_anchor=(0.5,

x_min, x_max = axs[0].get_xlim()
# plot  +/- maximum amplitude of the small angle solution as black dashed l
```

```
axs[0].plot([x_min, x_max], [A, A], '--k')
axs[0].plot([x_min, x_max], [-A, -A], '--k')
axs[0].set_xlim(x_min, x_max)

# plot the numerical and small angle solution for d/dt phi
axs[1].set_ylabel(r'$\dot\varphi$')
axs[1].set_xlabel(r'$t$')
axs[1].plot(time, phi_dot, lw=2.5)
axs[1].plot(time, sas_dot, '--r', lw=2.5)
plt.savefig('pendulum.png', dpi=300)
```



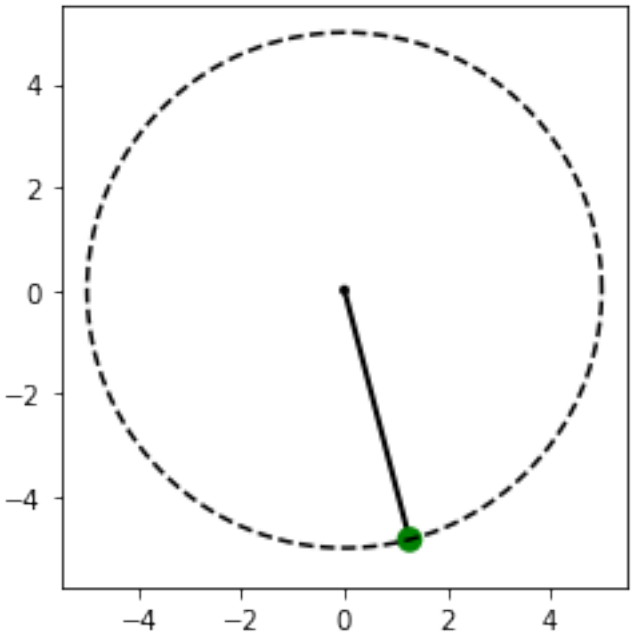
1.0.9 Let's do an animation (and save it as gif)

```
In [7]: # helpful modules
        from matplotlib.patches import Circle
        from matplotlib.animation import FuncAnimation
```

Never mind the details of the code. Just enjoy the output file.

In [8]:
```
# animation
figa, ax = plt.subplots()

# plot a black dashed circle around the origin (anker of the pendulum)
a = np.linspace(0.0, 2.0 * np.pi, 5000)
cx = L * np.cos(a)
cy = L * np.sin(a)

ax.set_aspect('equal')
ax.plot(cx, cy, '--k')

# calculate x- and y-coordinates for all values phi(t)
xp = [ L * np.sin(p) for p in phi]
yp = [-L * np.cos(p) for p in phi]

# plot the thread of the pendulum
thread, = ax.plot([0.0, xp[0]], [0.0, yp[0]], 'k', lw=2.0, zorder=-1000)

# plot a little black circle as anker for the pendulum
p_fix = Circle(xy=(0.0, 0.0),
               radius=0.1,
               facecolor='k')
ax.add_patch(p_fix)

# plot the pendulum mass aka the bob
p_bob = Circle(xy=(xp[0], yp[0]),
               radius=0.25,
               facecolor='g')
ax.add_patch(p_bob)

# since we have a lot of data points we only want to animate every skip_fra
# therefore we cut the data here
skip_frames = 20
xp = xp[::skip_frames]
yp = yp[::skip_frames]
num_frames = len(xp)

# function to update our plot objects (the pendulum bob and the thread)
def update(frame: int):
    p_bob.center = xp[frame], yp[frame]
    thread.set_xdata([0.0, xp[frame]])
    thread.set_ydata([0.0, yp[frame]])
    return (p_bob, thread)

ani = FuncAnimation(fig=figa, func=update, frames=num_frames, interval=1)

# save the animation as a gif, you can find it in the same folder as this n
ani.save('pendulum_animation.gif', fps=30)
```

```
MovieWriter ffmpeg unavailable; using Pillow instead.
```



In [ ]: