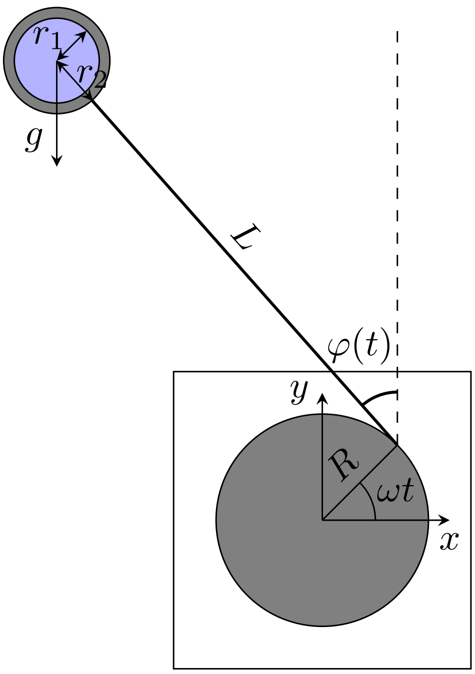
Mechanics 2 Lab

Group 2
Andreas Åberg

August 16, 2023

1 Part A: Theory



$$L_{tot} = L + r_2 \tag{1}$$
$$\hat{x} = R\cos(\omega t) - L_{tot}\sin(\varphi) \tag{2}$$
$$\hat{y} = R\sin(\omega t) + L_{tot}\cos(\varphi) \tag{3}$$

$$\dot{x} = -R\omega\sin(\omega t) - L_{tot}\dot{\varphi}\cos(\varphi) \tag{4}$$
$$\dot{y} = R\omega\cos(\omega t) - L_{tot}\dot{\varphi}\sin(\varphi) \tag{5}$$

$$v^2 = \dot{x}^2 + \dot{y}^2 \tag{6}$$

Merging equation (4), (5) and (6):

$$v^2 = L_{tot}^2 \cdot \dot{\varphi}^2 - 2L_{tot}R\omega \cdot \dot{\varphi}\sin(\varphi - \omega t) + R^2\omega^2 \tag{7}$$

Kinetic energy where T_1 is the rotational energy of the ball inside the shell and T_2 the kinetic energy of the translational speed:

$$T = T_1 + T_2 \tag{8}$$
$$T_1 = \frac{1}{2}I_2\dot{\varphi}^2 \tag{9}$$
$$T_2 = \frac{1}{2}m \cdot v^2 \tag{10}$$
$$T = \frac{1}{2}\left(I_2 \cdot \dot{\varphi}^2 + m \cdot v^2\right) \tag{11}$$
$$T = \frac{1}{2}\left(I_2\dot{\varphi}^2 + m(\overbrace{L_{tot}^2 \cdot \dot{\varphi}^2 - 2L_{tot}R\omega \cdot \dot{\varphi}\sin(\varphi - \omega t) + R^2\omega^2}^{v^2})\right) \tag{12}$$

Since the potential energy of the system only depends on the ball and shell (since it has all the mass), the potential energy can be written as:

$$V = mg \cdot \hat{y}. \tag{13}$$

Inserting equation (3):

$$V = mg\left(R\sin(\omega t) + L_{tot}\cos(\varphi)\right) \tag{14}$$

With the Lagrangian L defined as L = T-V, with eq. (12) and (14):

$$\begin{aligned} L =& \frac{1}{2}\bigg(I_2\dot{\varphi}^2 + m(L_{tot}^2 \cdot \dot{\varphi}^2 - 2L_{tot}R\omega \cdot \dot{\varphi}\sin(\varphi - \omega t) + R^2\omega^2)\bigg) \\ &- mg(R\sin(\omega t) + L_{tot}\cos(\varphi)) \end{aligned} \tag{15}$$

The Euler-Lagrange defined as:

$$\frac{\partial L}{\partial \varphi} - \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\varphi}}\right) = 0 \Leftrightarrow \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\varphi}}\right) - \frac{\partial L}{\partial \varphi} = 0 \tag{16}$$

The parts of Euler-Lagrange:

$$\frac{\partial L}{\partial \varphi} = \frac{\partial}{\partial \varphi}\left(\frac{m}{2}\Big(-2L_{tot}R\omega \cdot \dot{\varphi}\sin(\varphi - \omega t)\Big) - mgL_{tot}\cos(\varphi)\right) \tag{17}$$

$$= mL_{tot}\big(g\sin(\varphi) - R\omega \cdot \dot{\varphi}\cos(\omega t - \varphi)\big) \tag{18}$$

$$\frac{\partial L}{\partial \dot{\varphi}} = \frac{\partial}{\partial \dot{\varphi}}\left(\frac{1}{2}\Big(I_2 \cdot \dot{\varphi}^2 + mL_{tot}^2\dot{\varphi}^2 - 2mL_{tot}R\omega \cdot \dot{\varphi}\sin(\varphi - \omega t)\Big)\right) \tag{19}$$

$$= I_2\dot{\varphi} + mL_{tot}^2\dot{\varphi} - mL_{tot}R\omega\sin(\varphi - \omega t) \tag{20}$$

Deriving eq. 20 with respect to time:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\varphi}}\right) = I_2\ddot{\varphi} + mL_{tot}^2\ddot{\varphi} - mL_{tot}R\omega\cos(\omega t - \varphi)(\dot{\varphi} - \omega) \tag{21}$$

Inserting the parts eq. (18) and (21) into Euler-Lagrange eq. (16)

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\varphi}}\right) - \frac{\partial L}{\partial \varphi} = 0 \Rightarrow \tag{22}$$

$$\begin{aligned} &I_2\ddot{\varphi} + mL_{tot}^2\ddot{\varphi} - mL_{tot}R\omega\cos(\omega t - \varphi)(\dot{\varphi} - \omega) - \\ &mL_{tot}\big(g\sin(\varphi) - R\omega \cdot \dot{\varphi}\cos(\omega t - \varphi)\big) = 0 \end{aligned} \tag{23}$$

After simplifying:

$$I_2\ddot{\varphi} + mL_{tot}^2\ddot{\varphi} + mL_{tot}\left(R\omega^2\cos(\omega t - \varphi) - g\sin(\varphi)\right) = 0 \tag{24}$$

$$\ddot{\varphi} = -\frac{mL_{tot}\left(R\omega^2\cos(\omega t - \varphi) - g\sin(\varphi)\right)}{I_2 + mL_{tot}^2}. \tag{25}$$

Introducing ℓ as variable:

$$\ell := \frac{I_2 + mL_{tot}^2}{mL_{tot}} \tag{26}$$

$$\ddot{\varphi} = -\frac{R\omega^2\cos(\omega t - \varphi) - g\sin(\varphi)}{\ell} \tag{27}$$

$$\cos(\omega t - \varphi) = \cos(\omega t)\cos(\varphi) + \sin(\omega t)\sin(\varphi) \tag{28}$$

$$\ddot{\varphi} = -\frac{(R\omega^2\sin(\omega t) - g)\sin(\varphi) + R\omega^2\cos(\omega t)\cos(\varphi)}{\ell} \tag{29}$$

Assume $|\varphi(t)| << 1$:

$$\ddot{\varphi} = -\frac{(R\omega^2\sin(\omega t) - g)\varphi + R\omega^2\cos(\omega t)}{\ell} \tag{30}$$

2 Part B: Numerical solution in Python

Calculations were done with given parameters:

$$L = 1m$$
$$R = 0.05m$$
$$r_1 = 0.04m$$
$$r_2 = 0.05m$$
$$\rho = 7800kg/m^3,$$

and the initial conditions (at $t = 0$) as:

$$\varphi(0) = 1^\circ, \quad \dot{\varphi}(0) = 0$$

2.1 Calculation 1

The maximum deviation of the pendulum from vertical position with the angular velocity $\omega = 100$ rad/s is:

```
Y max: 0.11619935627748722, for omega: 100.0
```

Figure 1: Maximum deviation for $\omega = 100.0$

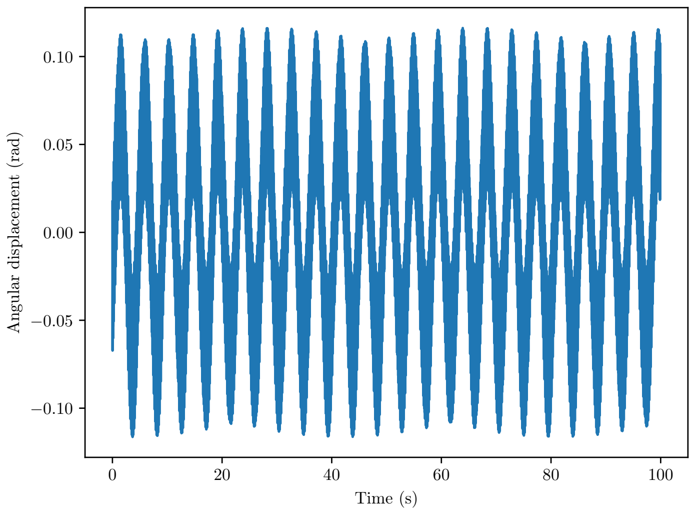


Figure 2: Stable result for $\omega = 100$

Where **Y max** is the maximum deviation in radians.

2.2 Finding minimum angular velocity

Finding the minimum angular velocity depends highly on the tolerance of angular deviation. The model requires $|\varphi(t)| << 1$, and if the step size is precise (eg. checking $\Delta\omega = 0.01$) it will require a larger ω to find a solution. Checking for values $|\varphi(t)| < 1$ from $\omega = 0$ to 100 with $\Delta\omega = 5$:

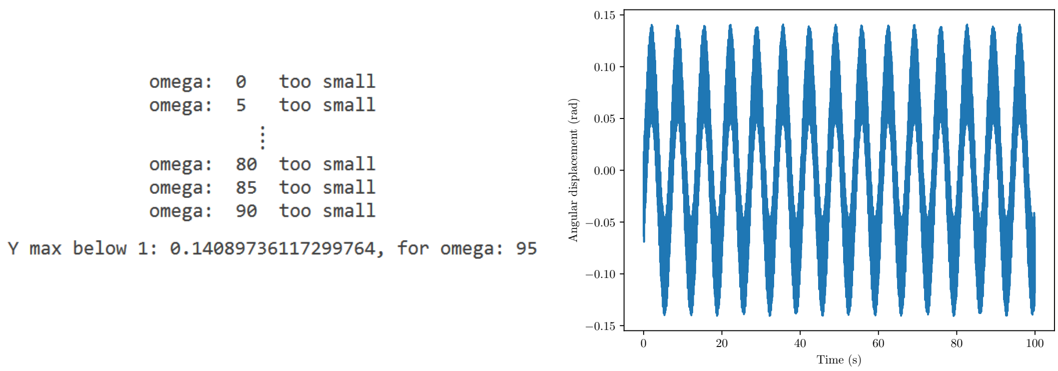


Figure 3: Results from first iteration with $\Delta\omega = 5$

This concludes that a stable point must be within $90 \leq \omega \leq 95$, since 95 is stable.

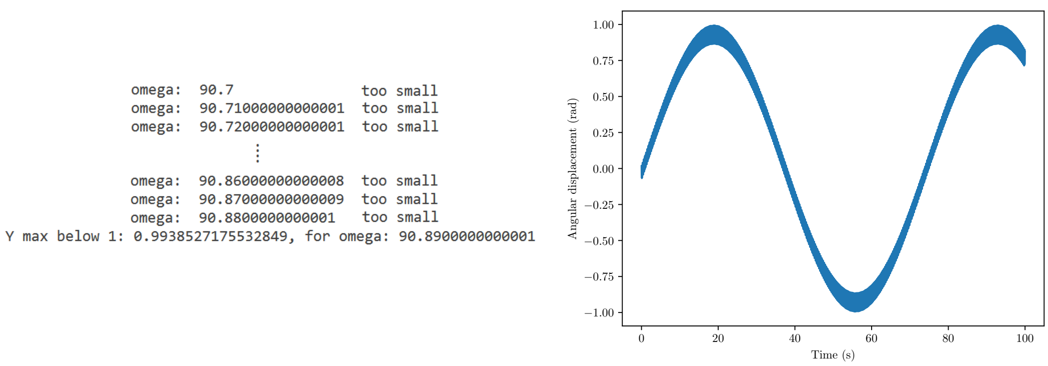


Figure 4: Last iteration with $\Delta\omega = 0.01$

There is a stable point for $\omega \approx 90.9$, however the value of the angular maximum deviation is very close to 1 when the model specifies $|\varphi(t)| << 1$. Depending on what value we say $\varphi(t)$ is allowed to have we will have a different result. In the case when $\omega = 100$, the maximum angle deviation was calculated to be $\varphi \approx 0.1162 \approx 0.12$. Setting $|\varphi(t)| < 0.12$ the lowest possible value of ω will be close to $\omega = 100$. This is proven in following figure 5.

```
omega:  91  too small
omega:  92  too small
            ⋮
omega:  98  too small
Y max below 0.12: 0.11923833883078272, for omega: 99
```

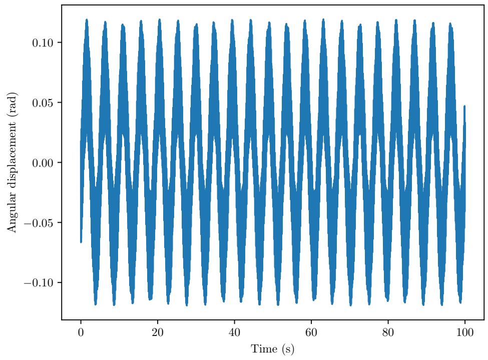


Figure 5: Stable system for $\varphi(t) < 0.12$

3 Discussion

Asking the question to find the minimum angular velocity of ω, in which the inverted pendulum stays stable is somewhat vague, since the answer is not unique. Allowing values of $|\varphi(t)| < 1$ compared to $|\varphi(t)| < 0.12$ generates different results. The requirement $|\varphi(t)| << 1$ is ambiguous, since a much smaller value than something else is not exactly defined. When describing a mathematical system in a programming language such as Python, this is a problem because it requires a specified limit for how small or big $\varphi(t)$ is allowed to be.

Something a bit unrelated but noteworthy: by some unofficial testing and the result in figure 5, is that the angular deviation decrease when increasing ω. So if you want to minimize the deviation of φ one could argue that you should use as high value as possible for ω. This seems to be similar to Kapitza's pendulum where the pendulum arm is oscillating strictly in up and down motion (compared to our case when the arm is fixed to a rotation).

4 Python Code

```
import numpy as np
from numpy import sin, cos, pi
from scipy.integrate import odeint
from matplotlib import pyplot as plt
#import matplotlib  Remove comment to export to pgf
#matplotlib.use("pgf")
#matplotlib.rcParams.update({
#     "pgf.texsystem": "pdflatex",
#     'font.family': 'serif',
#     'text.usetex': True,
#     'pgf.rcfonts': False,
#})

def find_omega(omegas, y_max):  #find omega for y_max
    for omega in omegas:
        phiOpt = odeint(equations,[phi0, x0], time, args=(omega,))
                                                  #find Optimised phi
        max_value = np.max(phiOpt[:,0])     #Positive max of phiOpt
        min_value = -np.min(phiOpt[:,0])    #Max for negative
        if abs(max_value) < y_max and min_value < y_max:
            return omega #Check if max is below y_max
        else:
            print("omega: " , omega, " too small")
            #Prints too low omegas for debugging input parameters

#define equations
def equations(y0, t, omega):
#boundary y0 = [intial angle, angle velocity]
    phi, x = y0
    f = [x, -((R*omega**2 * sin(omega*t)-g)*phi + R*omega**2 * cos
                                     (omega*t))/l] #[initial
                                     velocity, ode]
    return f

def plot_results(time, phi1):   #Plotting radian displacement/time
    plt.plot(time, phi1[:,0])
    plt.xlabel("Time (s)")
    plt.ylabel("Angular displacement (rad)")
    #plt.savefig('LaTeXPlot.pgf')    #remove comment to export to
                                     pgf
    plt.show()
```

```
#parameters
g = 9.81          #m/s^2
rho = 7800        #kg/m^3
L = 1             #Length of pendulum (m)
r1 = 0.04         #radius of sphere (m)
r2 = 0.05         #radius of spherical shell (m)
R = 0.05          #Engine radius (m)
m1 = (4/3)*pi*rho*r1**3                  #mass off sphere (kg)
m2 = (4/3)*pi*rho*r2**3 - m1             #mass of spherical shell (kg)
m = m1+m2                                #total mass of the system
I2 = (2/3)*m2*r2**2      #Mass moment of inertia of spherical shell
l = (I2 + m*(L+r2)**2)/(m*(L+r2))   #Constant for differential eq.

#inital conditions
initial_angle = 1.0
#Initial angle for pendulum arm (degrees)

phi0 = np.radians(initial_angle) #Degree to radian convertion
initial_angle_speed = 0.0   #Initial angle speed for pendulum arm
                                 (degrees/s)
x0 = np.radians(initial_angle_speed)  #Degree to radian convertion
t_max = 100  #Max time for plotting on x-axis (s)
time = np.arange(0,t_max, 0.025)  #Step function of time (s)

#find the sol.
omega = 0   #Intialize omega to something
omegas = np.arange(90,91,0.1)  #Array for different omegas
y_max = 1  #Maximum allowed displacement for pendulum arm
omega = find_omega(omegas, y_max)   #Calling find_omega to find
                                 required omega for y_max
phi1 = odeint(equations, [phi0, x0], time, args=(omega,))  #Solve
max_value = np.max(phi1[:,0])

print(f"Y max below {y_max}: {max_value}, for omega: {omega}")
#Print solution parameters

plot_results(time, phi1) #Function to plot the results
```