Interfejsy i Multimedia w Technice Gr.22

Grzegorz Kupczyk Adam Bernard Jarosław Bartkowski

Kwiecień 2021

1 Wprowadzenie

Celem projektu jest symulacja zbiornika z wodą zgodna z dynamiką obiektu oraz sporządzenie dokumentu w języku LaTeX zawierającego infografikę z wykorzystaniem paczki *tikz*. Zasady dotyczące dokumentu zawarto w tabeli 1.

2 Dynamika obiektu

Zbiornik z cieczą przedstawiony na rysunku 1. można opisać następującym równaniem różniczkowym:

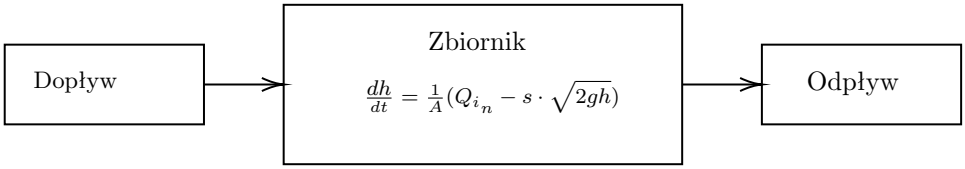
$$\frac{dh}{dt} = \frac{1}{A}(Q_{in} - s \cdot \sqrt{2gh}) \tag{1}$$

gdzie:

- A - powierzchnia lustra wody,
- g - wartość przyspieszenia ziemskiego,
- s - pole powierzchni przekroju wypływu,
- h - wysokość słupa cieczy,
- Q_{in} - dopływ wody

	Wymagania
1	Dokument zapisany w języku LaTeX
2	Schemat układu wykonany z użyciem biblioteki tikz
3	Symulacja obiektu w czasie rzeczywistym z wykorzystaniem Pythona oraz paczki numpy
4	Animacja oparta o symulacje, zapisana w Pythonie z wykorzystaniem biblioteki matplotlib

Tabela 1: Wymagania dotyczące projektu



Rysunek 1: Schemat obiektu

3 Symulacja

W celu wyznaczenia odpowiedniej wysokości słupa cieczy napisano algorytm wykorzystujący biblioteki *numpy* oraz *matplotlib*. Program zaprezentowano na listingu 1. Można go pobrać z repozytorium Github.

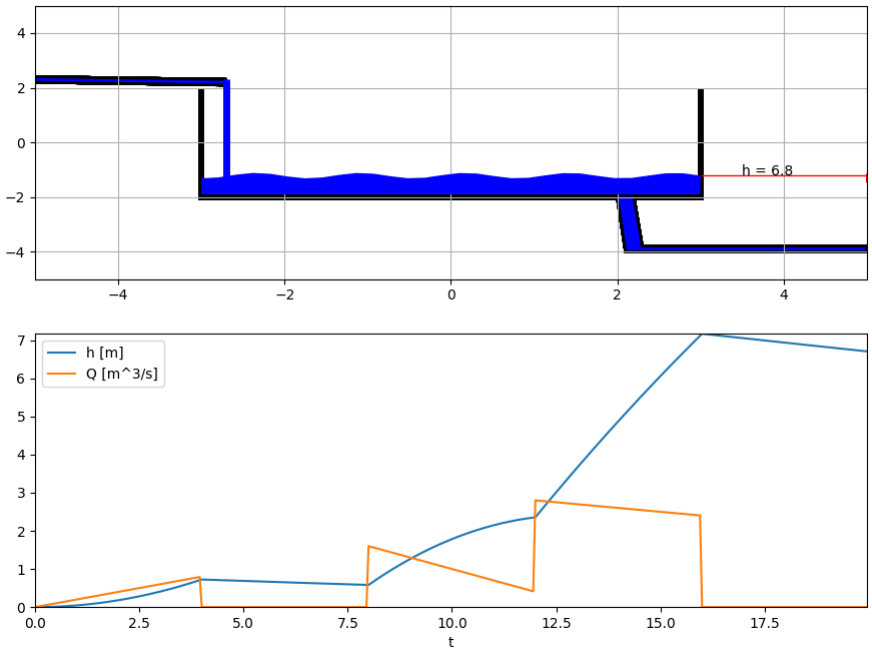
```
1  import numpy as np
2  import matplotlib.pyplot as plt
3  from scipy.integrate import odeint
4  from math import sqrt, sin
5  from matplotlib.animation import FuncAnimation
6
7
8  def Q(t):
9      if t < 4:
10         return 0.2 * t
11     elif t < 8:
12         return 0
13     elif t < 12:
14         return 4 - 0.3 * t
15     elif t < 16:
16         return 4 - 0.1 * t
17     return 0
18
19
20 def zbiornik_model(x, t):
21     A = 2  # pole powierzchni lustra wody
22     g = 9.81  # waro    przy pieszenia ziemskiego
23     s = 0.02  # pole powierzchni przekroju wyp ywu
24     Qin = Q(t)  # dop yw wody
25     h = x[0]  # wysoko    s upa wody
26
27     dhdt = 1 / A * (Qin - s * sqrt(2 * g * h))
28
29     return [dhdt]
30
31
32 class SuperContainer:
33     def __init__(self, h, q_in, ax):
34         container_points = [[-3, 2], [-3, -2], [3, -2], [3, 2]]
35         pipe_in_points = [[-2.7, 2.3], [-5, 2.4], [-5, 2.2], [-2.7,
       2.1]]
36         pipe_out_points = [
37             [2, -2],
38             [2.1, -4],
39             [5, -4],
```

```
            [5, -3.8],
            [2.3, -3.8],
            [2.2, -2],
        ]
        fill_points = self.__get_fill_points()
        pointer_points = [[3, 2], [5, 2]]
        stream_points = self.__stream(0)

        self.outline = plt.Polygon(
            container_points, closed=None, edgecolor="k", fill=
False, lw=4
        )
        self.pointer = plt.Polygon(
            pointer_points, closed=None, edgecolor="r", fill=False,
 lw=1
        )
        self.fill = plt.Polygon(fill_points, facecolor="b",
edgecolor=None, fill=True)
        self.pipe_in = plt.Polygon(
            pipe_in_points, closed=None, edgecolor="k", facecolor="
b", lw=2
        )
        self.pipe_out = plt.Polygon(
            pipe_out_points, closed=None, edgecolor="k", facecolor=
"b", lw=2
        )

        self.stream = plt.Polygon(
            stream_points, closed=None, edgecolor="b", fill=False,
lw=5
        )
        self.text = ax.text(4, 2, "h=0")
        self.__h = h
        self.__q = q_in

    def add_path(self, gca):
        gca.add_patch(self.fill)
        gca.add_patch(self.outline)
        gca.add_patch(self.pipe_in)
        gca.add_patch(self.pipe_out)
        gca.add_patch(self.pointer)
        gca.add_patch(self.stream)

    def __wavy(self, x1, x2, y0, points, amp=1, offset=0, reverse=
False):
        ax = np.linspace(x1, x2, points)
        ay = np.sin(np.linspace(0, (x2 - x1) * 5, points) + offset)
 * amp
        ay = ay + y0

        if reverse:
            ax = np.flip(ax)

        return np.vstack((ax, ay)).T

    def __stream(self, q_in):
        q_max = 5
```

```
        q = q_in / q_max
        w_max = 1
        points = 20
        x = np.linspace(-2.7, -2.7 + q * w_max, points)
        y = -4 * np.sin(np.linspace(0, np.pi / 2, points)) + 2.3

        return np.vstack((x, y)).T

    def __get_fill_points(self, offset=0, h=0):
        h_max = 10
        y_min = -1.9
        y = y_min + h / h_max
        return [
            [-3, y],
            [-3, y_min],
            [3, y_min],
            *self.__wavy(-3, 3, y, 30, 0.1, offset * 0.1, True),
        ]

    def animate(self, i):
        h = self.__h[i]
        q = self.__q[i]

        h_max = 10
        y_min = -1.9
        y = y_min + h / h_max

        self.fill.set_xy(self.__get_fill_points(i, h))
        self.pointer.set_xy([[3, y], [5, y]])
        self.stream.set_xy(self.__stream(q))
        self.text.set_position([3.5, y])
        self.text.set_text("h = %.1f" % h)

        return (self.fill, self.pointer, self.stream, self.text)


def init():
    global superContainer, h, Q, ax0
    superContainer = SuperContainer(h, Q, ax0)
    superContainer.add_path(plt.gca())
    return superContainer.animate(0)


def animate(i):
    return superContainer.animate(i)


fig = plt.figure()

dt = 0.05
t = np.arange(0.0, 20, dt)

x0 = [0]
x = odeint(zbiornik_model, x0, t)
superContainer = None

h = x[:, 0]
```

```
146 Q = [Q(t) for t in t]
147 ax1 = fig.add_subplot(212, autoscale_on=True)
148 ax1.plot(t, h, label="h [m]")
149 ax1.plot(t, Q, label="Q [m^3/s]")
150 ax1.set_xlim([t[0], t[-1]])
151 ax1.set_ylim((0, np.amax(h)))
152 ax1.set_xlabel("t")
153 ax1.legend(loc="upper left")
154
155 ax0 = fig.add_subplot(211, autoscale_on=False, xlim=(-5, 5), ylim
        =(-5, 5))
156 ax0.grid()
157
158 ani = FuncAnimation(
159     fig, animate, np.arange(1, len(t)), interval=25, blit=True,
        init_func=init
160 )
161
162 plt.show()
```

Listing 1: Kod programu symulacji i animacji



Rysunek 2: Pojedyncza klatka animacji

4 Podsumowanie

Kod spełnia swoje zadanie animując obiekt zbiornika z wodą, który jest uzupełniany w czasie rzeczywistym. Animacja odzwierciedla ten proces wykorzystując równanie różniczkowe podane w równaniu 1 . Cały program funkcjonuje zgodnie z przeznaczeniem, dlatego można wywnioskować, iż zadanie zostało wykonane w sposób ***poprawny***.

Literatura

[1] mgr inż. Karol Miądlicki, Instytut Technologii Mechanicznej
Ćwiczenie laboratoryjne nr 1: Symulacja zmian poziomu cieczy w zbiorniku oraz układzie zbiorników.
Zachodniopomorski Uniwersytet Technologiczny w Szczecinie, 2021.

[2] Parul Pandey: *Animations with Matplotlib*,
`https://towardsdatascience.com/animations-with-matplotlib-d96375c5442c`