

PB1010 - Fysikk 1

Oblig 1

Sølve Kjelseth

February 16, 2025

0. Introduction

This is for Oblig 1 in Physics where the mechanics of a free falling ball is dissected.

Note: The \LaTeX and python code is open source, [github link](#).

0.0. Table of Contents

1. Task 1	3
1.0.Part a	3
1.1.Part b	7
1.2.Part c	10
1.3.Part d	13
2. Task 2	15
2.0.Part a	16
2.1.Part b	17
2.2.Part c	17
2.3.Ekstra	18
2.4.Source code	19

0.1. List of Figures

1.0	Sketch of sphere falling from building	4
1.1	Sketch of sphere falling with initial horizontal velocity	8
1.2	Free body diagram of sphere with air resistance	10
2.0	Simulations of height both with and without air resistance	16
2.1	Simulations of velocity both with and without air resistance	17
2.2	Simulations of acceleration both with and without air resistance	18
2.3	Simulation of falling spheres, over engineered to a high degree	24

1. Task 1

I do not understand why the assignment wants to have ground level at $h = 0m$ and positive direction downwards at the same time(as stated in part a), that means the sphere has a starting position at $h = -50m$. The math works either way and even if it seems counterintuitive I choose to adhere to the requirements of the task and assumes this direction going forward unless otherwise stated.

1.0. Part a

To solve part a I choose to use the ISEE method.

1.0.0. Identify

I find some important variables, such as ground level, $h_{ground} = 0m$ and distance between the ground and the sphere $x_{sphere-ground} = 50m$. I also see that positive direction is downwards, that means $h > 0$ will be meters under ground. That makes starting position $h_0 = -50m$. Air resistance is omitted, this way only gravity affects acceleration. This is constant acceleration, downwards, thereby positive $a = g = 9.825ms^{-2}$ (This is the official number for g in Oslo). Therefore the equations for motion in constant acceleration can be used. I also note that the start time of the free fall is at $t_0 = 0s$ and it has no initial velocity $v_0 = 0ms^{-1}$. Lastly I see that everything in this part is single dimension motion.

1.0.1. Set up

For the first question, find $h(t)$, we know the goal variable is position, we "know" the time variable as all possible values will be put in, and finally the last known is acceleration. this means the velocity less equation can be used.

$$x = x_0 + v_0t + \frac{1}{2}at^2 \quad (1)$$

For the next question, find $v(t)$, we know the goal variable is velocity, and here time is also "known" as it is the function input and since we know acceleration as well the position less equation can be used.

$$v = v_0 + at \quad (2)$$

For the final question, find t_{ground} , we can rearrange Equation 1 to find the time when we calculate further. A sketch of the situation is provided on the next page.

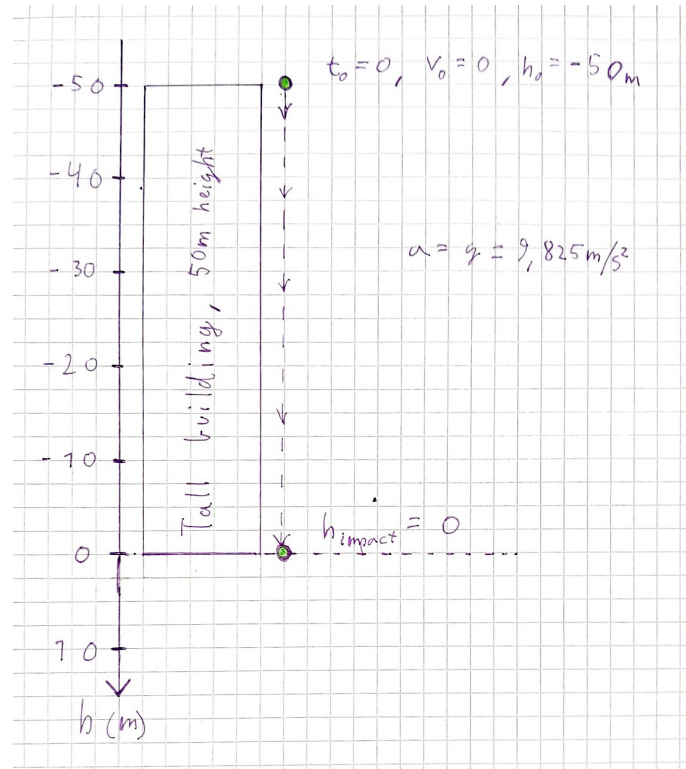


Figure 1.0: Sketch of sphere falling from building

1.0.2. Execute

Filling the used variables in Equation 1, and making it into a function:

$$h(t) = h_0 + v_0 t + \frac{1}{2} g t^2$$

Filling in values, here $v_0 = 0 \text{ m s}^{-1}$ simplifies the equation by removing a term.

$$h(t) = \frac{1}{2} 9.825 \text{ m s}^{-2} t^2 - 50 \text{ m} \quad (3)$$

Moving on to the next, filling in the used variables in Equation 2 and making it a function.

$$v(t) = v_0 + g t$$

Filling in values, $v_0 = 0 \text{ m s}^{-1}$ simplifies this equation too.

$$v(t) = 9.825 \text{ m s}^{-2} t \quad (4)$$

The time of impact will be at $h(t) = 0m$, filling this in Equation 3 and solving for t .

$$0m = \frac{1}{2}9.825ms^{-2}t^2 - 50m$$

$$50m = \frac{1}{2}9.825ms^{-2}t^2$$

$$100m = 9.825ms^{-2}t^2$$

$$\frac{100m}{9.825ms^{-2}} = t^2$$

$$t = \pm \sqrt{\frac{100m}{9.825ms^{-2}}} \approx 3.19s \quad (5)$$

Now as time starts at 0, the negative solution for t does not make sense in this task and is therefore omitted from the results.

1.0.3. Evaluate

I feel like the magnitude of my answers is in the correct order and $3.19s$ seems quite reasonable time for an object to fall 50 meters, it is maybe a bit fast, but this is also with no air resistance. I will also want to check units, starting with Equation 3.

$$m = m \cdot s^{-2} \cdot s^2 - m$$

$$m = m - m$$

$$m = m$$

This checks out correct, onto Equation 4.

$$m \cdot s^{-1} = m \cdot s^{-2} \cdot s$$

$$m \cdot s^{-1} = m \cdot s^{-1}$$

And the final one, Equation 5.

$$s = \sqrt{\frac{m}{m \cdot s^{-2}}}$$

$$s = \sqrt{\frac{1}{s^{-2}}}$$

$$s = \sqrt{s^2}$$

$$s = s$$

This also ended with correct units, This makes me conclude that the answers are correct with a very high probability.

1.0.4. Solution

This is the function for $h(t)$:

$$h(t) = \frac{1}{2}9.825ms^{-2}t^2 - 50m \quad (3)$$

This is the function for $v(t)$:

$$v(t) = 9.825ms^{-2}t \quad (4)$$

The sphere hits the ground approximately $3.19s$ after letting it go.

$$t = \pm\sqrt{\frac{100m}{9.825ms^{-2}}} \approx 3.19s \quad (5)$$

1.1. Part b

To solve part b I choose to use the ISEE method and to build upon the finds in part a.

1.1.0. Identify

I find the same initial variables as in part a, $t_0 = 0s$, $h_0 = -50m$, $v_{0h} = 0ms^{-1}$, this time I have specified the direction of v and a as we are now dealing with two dimensional motion. I choose to keep the vectors decomposed and calculate separately for each axis. The new information I have is that $v_{0x} = 7.2ms^{-1}$ and I choose to set the initial position for the sphere at $x_0 = 0m$ and positive direction along the horizontal movement. As there is no air resistance we can see that only gravity affects the acceleration, $a_h = g = 9.825ms^{-2}$ and $a_x = 0ms^{-2}$. This is constant and therefore I can still use equations for motion with constant acceleration.

1.1.1. Set up

The first question, find the time until impact, is more of a trick question, as the horizontal movement have no impact on the vertical movement, this means the time will be the same as calculated in part a. The second question, find the horizontal movement at time of impact, the goal variable is known, x and we know time of impact from part a, we also know that since there is no acceleration the speed is always the same, I can therefore use any of the equations and I decided to use the velocity less as it involves initial velocity as it is known and it simplifies nicely when acceleration is zero.

$$x = x_0 + v_0t + \frac{1}{2}at^2 \quad (1)$$

A sketch of the situation is provided on the next page.

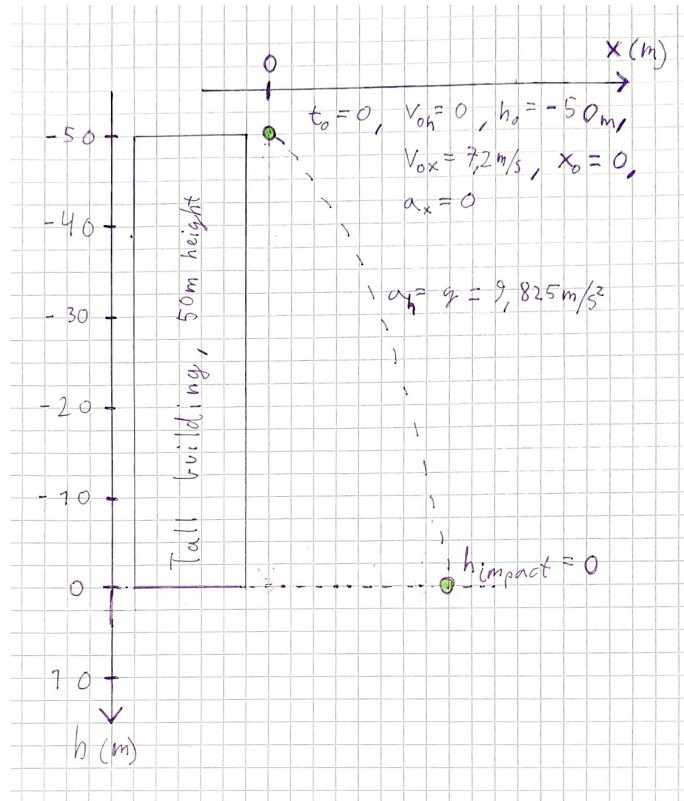


Figure 1.1: Sketch of sphere falling with initial horizontal velocity

1.1.2. Execute

I start by simplifying this equation as both $x_0 = 0$ and $a_x = 0$ this removes two terms. Also filling in the used variables.

$$x_{\text{impact}} = v_0 t_{\text{impact}}$$

Now with values is filled in.

$$x_{\text{impact}} \approx 7.2 \text{ms}^{-1} \cdot 3.19 \text{s} \approx 22.97 \text{m} \quad (6)$$

1.1.3. Evaluate

I feel like the magnitude of my answer is in the correct order and $22.97m$ seems quite reasonable for an object to move that distance when the velocity is $v_0 = 7.2ms^{-1}$, and travels for $t_{impact} = 3.19s$. I will also want to check units to Equation 6.

$$m = m \cdot s^{-1} \cdot s$$

$$m = m$$

Here the units check out. This makes me conclude that the answer is correct with a very high probability.

1.1.4. Solution

The sphere hits the ground after approximately $3.19s$, the same as in part a.

$$t = \pm \sqrt{\frac{100m}{9.825ms^{-2}}} \approx 3.19s \quad (5)$$

The sphere moved approximately $22.97m$ horizontally.

$$x_{impact} \approx 7.2ms^{-1} \cdot 3.19s \approx 22.97m \quad (6)$$

1.2. Part c

To solve part c I choose to use the ISEE method.

1.2.0. Identify

This part does not need any numerical values as the task is to make a drawing, and set up an equation, therefore no important known values are needed. It is specified that correct sign should be used such that positive direction is upwards.

1.2.1. Set up

I start with the free body diagram, as it provides some context.

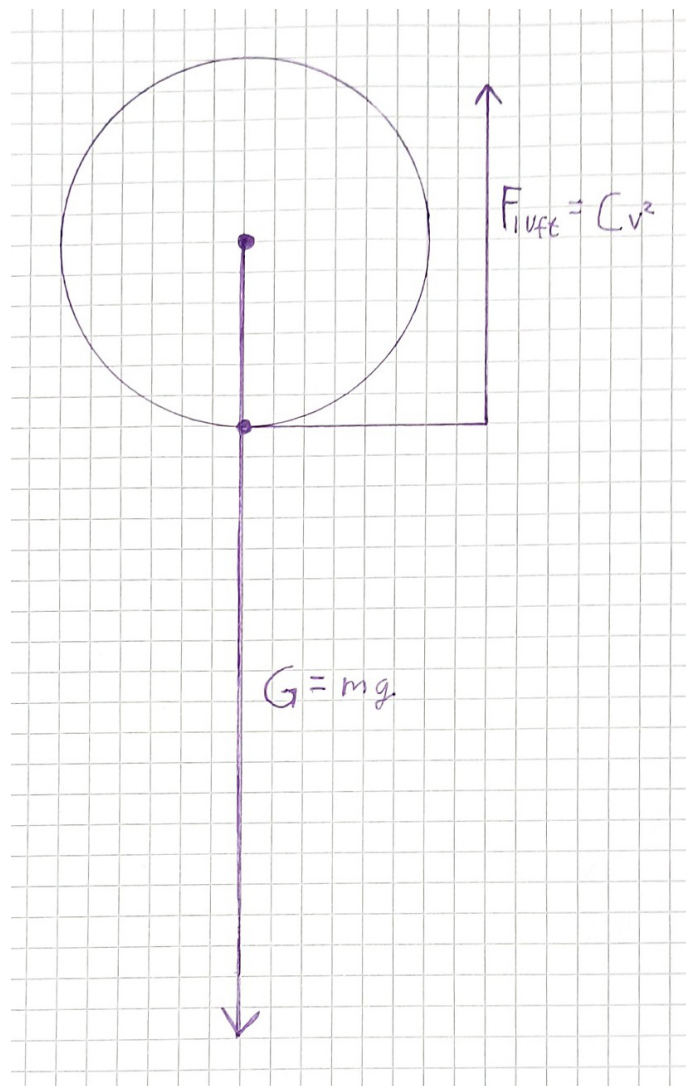


Figure 1.2: Free body diagram of sphere with air resistance

We only need to use one equation, Newtons 2nd law.

$$\sum F = ma \quad (7)$$

1.2.2. Execute

I start with modifying Equation 7, replacing the $\sum F$ with the forces drawn in Figure 1.2.

$$F_{lift} - G = ma \quad (8)$$

Now filling in for the forces and doing general arithmetic.

$$Cv^2 - mg = ma$$

$$\frac{Cv^2}{m} - g = a$$

$$a = \frac{dv}{dt} = -g + \frac{Cv^2}{m} \quad (9)$$

1.2.3. Evaluate

In Figure 1.2 I made G larger than F_{luft} as without high starting speed or other forces, F_{luft} can never be larger than G . Checking units for Equation 9.

$$m \cdot s^{-2} = m \cdot s^{-2} + \frac{kg \cdot m^{-1} \cdot (m \cdot s^{-1})^2}{kg}$$

$$m \cdot s^{-2} = m \cdot s^{-2} + m^{-1} \cdot m^2 \cdot s^{-2}$$

$$m \cdot s^{-2} = m \cdot s^{-2} + m \cdot s^{-2}$$

$$m \cdot s^{-2} = m \cdot s^{-2}$$

Here the units check out. This makes me conclude that the answer is correct with a very high probability.

1.2.4. Solution

Free body diagram is drawn in Figure 1.2.

Newtons 2nd law for this situation:

$$F_{luft} - G = ma \quad (8)$$

Can be rewritten to this:

$$a = \frac{dv}{dt} = -g + \frac{Cv^2}{m} \quad (9)$$

1.3. Part d

To solve part d I choose to use the ISEE method and use results from part c.

1.3.0. Identify

I find v_T to be the terminal velocity. I also find a value for $C = 5.00 \cdot 10^{-2} \text{kgm}^{-1}$ and $m = 100\text{g}$. I need to use Equation 9 from part c.

1.3.1. Set up

The first thing to do is to correct $m = 100\text{g}$ into $m = 0.1\text{kg}$ such as to keep all units as SI-units. Terminal velocity will cause no acceleration as there is no change in speed, this means that $\frac{dv}{dt} = 0$ at terminal velocity. It can be set up like this.

$$0 = -g + \frac{Cv^2}{m}$$

1.3.2. Execute

solving this for v using arithmetic.

$$g = \frac{Cv^2}{m}$$

$$mg = Cv^2$$

$$\frac{mg}{C} = v^2$$

$$v = \pm \sqrt{\frac{mg}{C}} \quad (10)$$

Only the negative answer makes sense with positive direction upwards. Filling in values for m , g and C .

$$v = -\sqrt{\frac{0.1\text{kg} \cdot 9.825\text{ms}^{-2}}{5.00 \cdot 10^{-2}\text{kgm}^{-1}}} \approx -4.43\text{ms}^{-1} \quad (11)$$

1.3.3. Evaluate

I feel like the magnitude of my answer is in the correct order and $-4.43ms^{-1}$ seems quite reasonable terminal speed for a object with mass $m = 100g$, it is maybe a bit slower than intuitivly expected for me. I will also want to check units for Equation ??.

$$m \cdot s^{-1} = \sqrt{\frac{kg \cdot m \cdot s^{-2}}{kg \cdot m^{-1}}}$$

$$m \cdot s^{-1} = \sqrt{\frac{m \cdot s^{-2}}{m^{-1}}}$$

$$m \cdot s^{-1} = \sqrt{m^2 \cdot s^{-2}}$$

$$m \cdot s^{-1} = \sqrt{\frac{m^2}{s^2}}$$

$$m \cdot s^{-1} = \frac{m}{s}$$

$$m \cdot s^{-1} = m \cdot s^{-1}$$

1.3.4. Solution

Using Equation 9 with $\frac{dv}{dt} = 0$ you can get equation for terminal velocity.

$$v = \pm \sqrt{\frac{mg}{C}} \quad (10)$$

Filling in values for c we find that the teminal velocity is approximately $-4.43ms^{-1}$.

$$v = -\sqrt{\frac{0.1kg \cdot 9.825ms^{-2}}{5.00 \cdot 10^{-2}kgm^{-1}}} \approx -4.43ms^{-1} \quad (11)$$

2. Task 2

Now I may have overdone this section as I taught this was a good exercise to extend my python skills, in other words, the code does exactly as described, and more. But it does not reassemble the example code a whole lot. If clarifications on how the code works is needed, feel free to reach out. In general the code as shown in Subsection 2.4. runs two simulations one with positive direction upwards named Sphere 1 and one with positive direction downwards, named Sphere 2 as the program handles both.

2.0. Part a

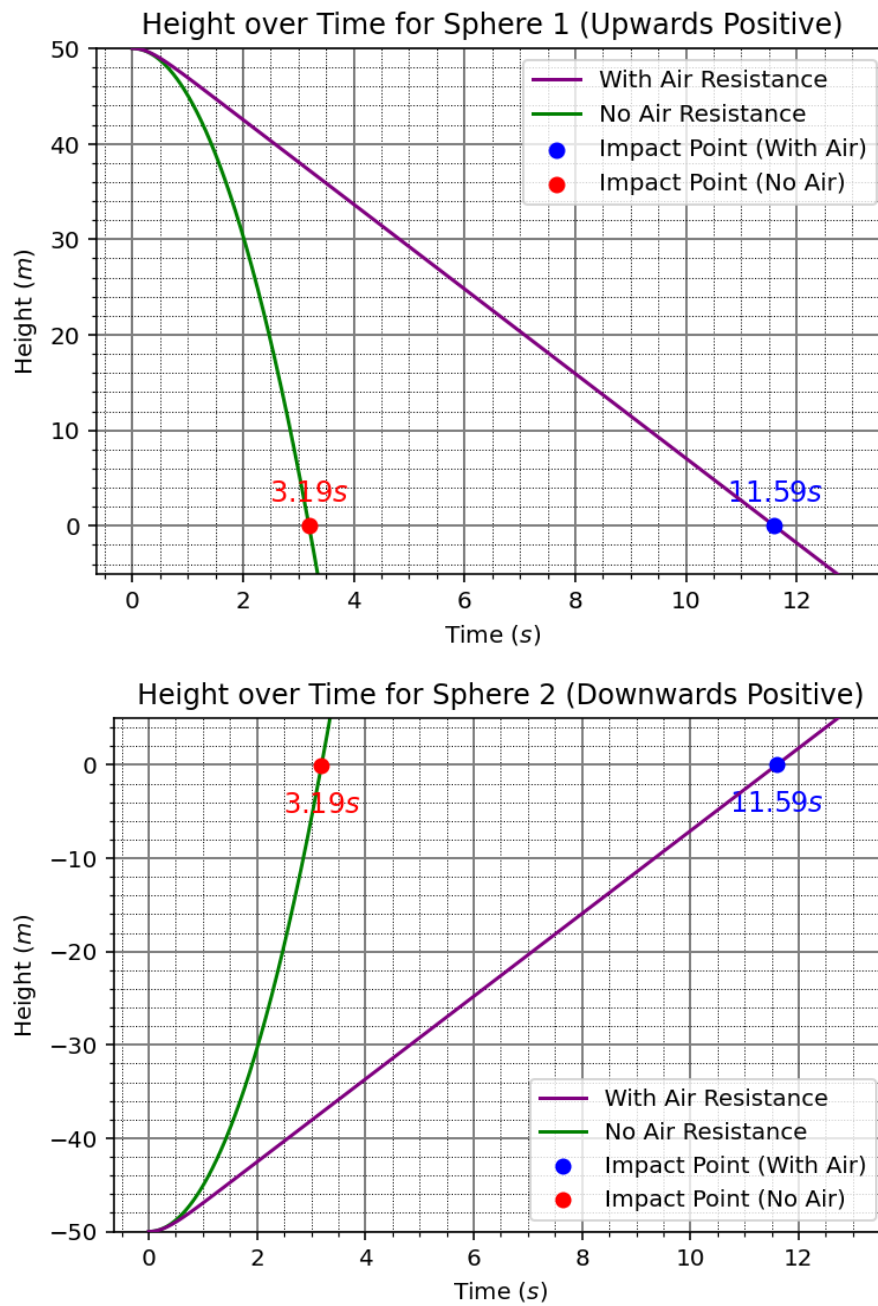


Figure 2.0: Simulations of height both with and without air resistance

2.1. Part b

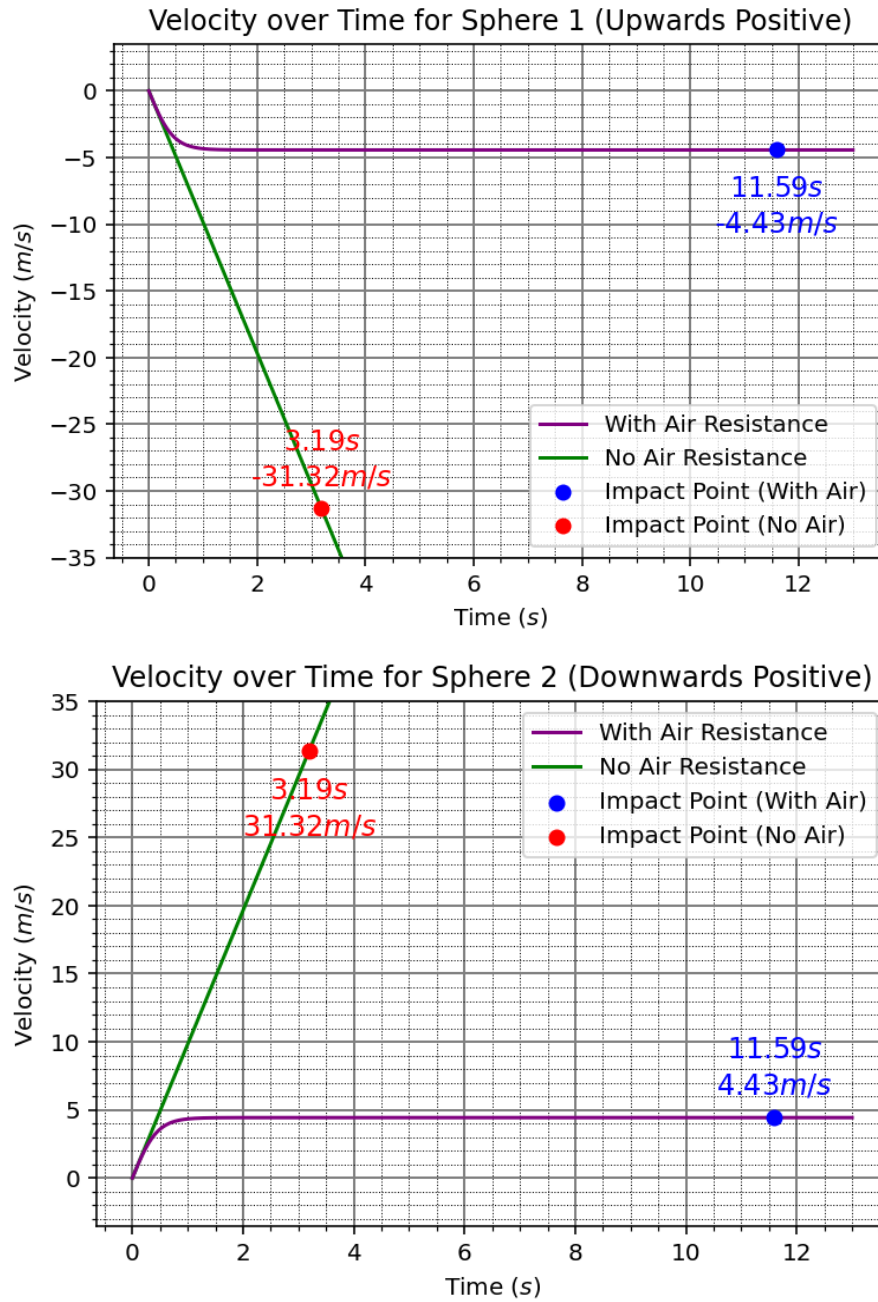


Figure 2.1: Simulations of velocity both with and without air resistance

2.2. Part c

We can see from the graph that terminal velocity v_T is achieved and the size is 4.43ms^{-1} . This corresponds well with results of Equation 11

$$v = -\sqrt{\frac{0.1\text{kg} \cdot 9.825\text{ms}^{-2}}{5.00 \cdot 10^{-2}\text{kgm}^{-1}}} \approx -4.43\text{ms}^{-1} \quad (11)$$

2.3. Ekstra

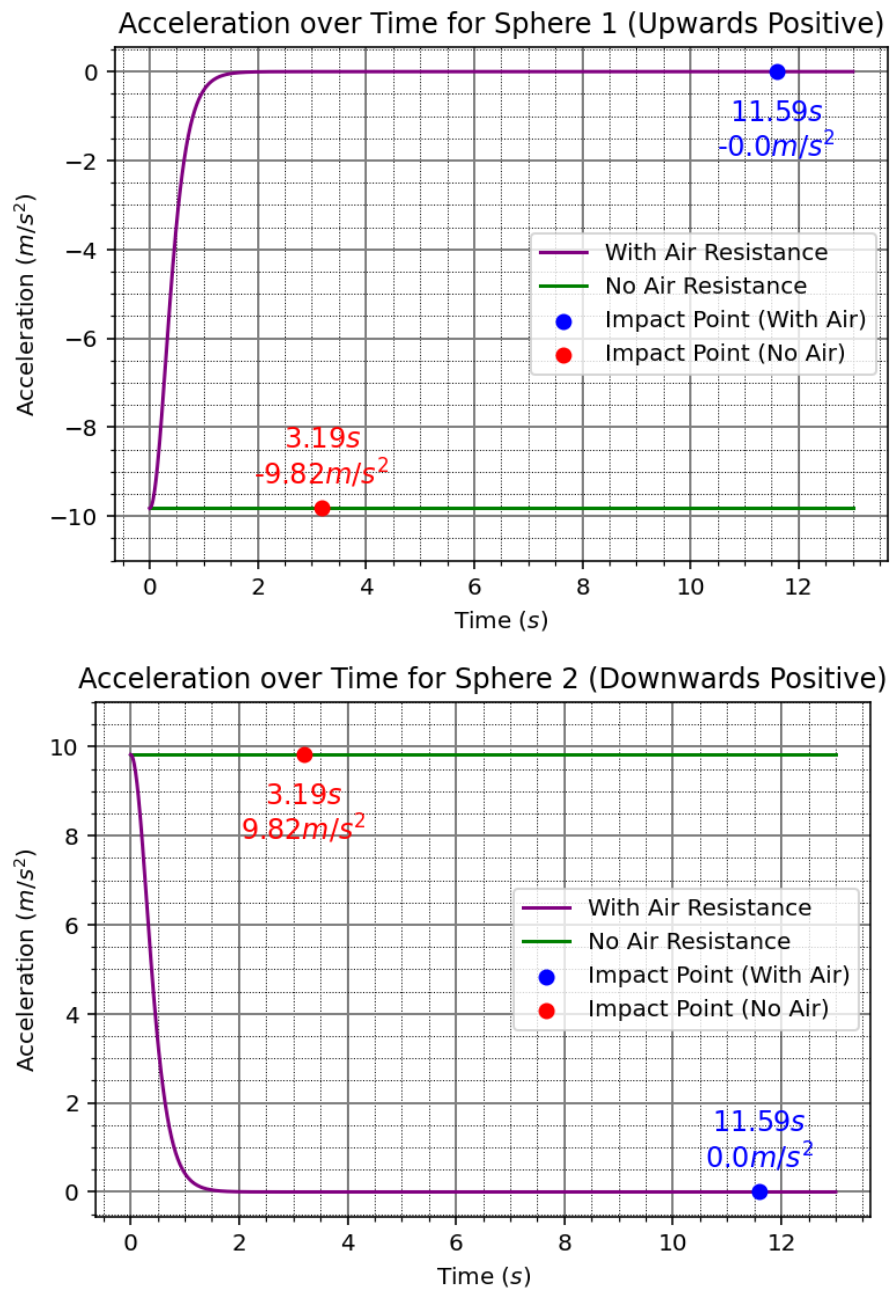


Figure 2.2: Simulations of acceleration both with and without air resistance

2.4. Source code

Python

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 class FallingSphere:
5     def __init__(self, name="Sphere", resolution=1000,
6                 start_time=0.0, end_time=10.0, distance_from_ground=100,
7                 speed_towards_ground=0.0, g=9.825, mass=1.0, C=0.05,
8                 positive_direction="upwards"):
9
10         """
11         Initialize the FallingSphere simulation.
12         g = 9.825 is used as this is the actual value for Oslo
13         """
14         if positive_direction not in ["upwards", "downwards"]:
15             raise ValueError(
16                 "direction must be either 'upwards' or 'downwards'")
17
18         if distance_from_ground <= 0:
19             raise ValueError(
20                 "Height above ground must be greater than zero.")
21
22         if g < 0 or (g == 0 and speed_towards_ground <= 0):
23             raise ValueError(
24                 "Gravity must be positive"
25                 "or zero with initial speed towards ground")
26
27         if mass <= 0:
28             raise ValueError(
29                 "Mass needs to be positive")
30
31         if C < 0:
32             raise ValueError(
33                 "C needs to be zero (for no resistance) or positive")
34
35         if start_time >= end_time:
36             raise ValueError(
37                 "End time needs to be larger than start time")
38
39         # Controls positive direction and how calculations change
40         self.positive_direction = positive_direction
41         self.sign = 1 if positive_direction == "upwards" else -1
42
43         # General properties of object
44         self.name = name
45         self.resolution = resolution
46
47         # Time based parameters
48         self.t = np.linspace(start_time, end_time, resolution) # All time values
49         self.dt = self.t[1] - self.t[0] # Time step size
50
51         # Constants
52         self.g = g
53         self.mass = mass
54         self.C = C
55
56         # Initial conditions with positive direction in mind
57         self.h0 = self.sign * distance_from_ground
58         self.v0 = self.sign * speed_towards_ground
59         self.a0 = self.sign * -g
60
61         # Arrays for simulation (with and without air resistance)
62         self.h = np.zeros(resolution)
63         self.v = np.zeros(resolution)
64         self.a = np.zeros(resolution)
65         self.h_noair = np.zeros(resolution)
66         self.v_noair = np.zeros(resolution)
```

```

66     self.a_noair = np.zeros(resolution)
67
68     # Initial conditions stored in array
69     self.h[0] = self.h0
70     self.v[0] = self.v0
71     self.a[0] = self.a0
72
73     # Impact times
74     self.t_impact = None
75     self.t_noair_impact = None
76
77     def calculate(self):
78         """
79         Compute the motion of the sphere over time.
80         """
81         # For loop starts at index 1 as initial value at index 0 is set
82         for i in range(1, self.resolution):
83             self.h[i] = self.h[i - 1] + self.v[i - 1] * self.dt
84             self.v[i] = self.v[i - 1] + self.a[i - 1] * self.dt
85             self.a[i] = self.sign * (
86                 -self.g + (self.C * (self.v[i] ** 2)) / self.mass)
87
88         # Without air resistance (vectorized calculations, does not need looping)
89         self.h_noair = self.h[0] + (
90             self.v0 * self.t + 0.5 * self.sign * -self.g * (self.t ** 2))
91         self.v_noair = self.sign * -self.g * self.t
92         self.a_noair = self.sign * -self.g * np.ones_like(self.t)
93
94     def find_impact_times(self):
95         """
96         Find the time indices when the sphere impacts the ground.
97         """
98         self.t_impact = np.argmin(np.abs(self.h))
99         self.t_noair_impact = np.argmin(np.abs(self.h_noair))
100
101         # Checks if simulation is long enough for impact
102         if abs(self.h_noair[self.t_noair_impact]) > 0.5:
103             raise ValueError(
104                 "Object does not hit the ground, within current timeframe")
105         elif abs(self.h[self.t_impact]) > 0.5:
106             raise ValueError(
107                 "Object does not hit the ground with air resistance, "
108                 "within current timeframe")
109
110
111     def display_results(self):
112         """
113         Print the time and velocity at impact for both cases.
114         """
115         print(
116             f'The sphere hits the ground at '
117             f'{round(self.t[self.t_impact], 3)}s, '
118             f'reaching a speed of {round(self.v[self.t_impact], 3)}m/s'
119             f'with {self.positive_direction} positive direction')
120         print(
121             f'and without air resistance at '
122             f'{round(self.t[self.t_noair_impact], 3)}s, '
123             f'reaching a speed of {round(self.v_noair[self.t_noair_impact], 3)}m/s '
124             f'with {self.positive_direction} positive direction')
125
126     def plot_height(self):
127         """
128         Plot height vs. time for both cases.
129         """
130         plt.figure(0)
131
132         max_val = np.ceil(abs(self.h0) / 5.0) * 5

```

```

133     y_step = max_val * 0.05
134     min_val = 2 * y_step
135
136     if self.sign == 1:
137         plt.ylim(-min_val, max_val)
138         plt.text(
139             self.t[self.t_impact],
140             self.h[self.t_impact] + 1 * y_step,
141             f'{round(self.t[self.t_impact], 2)}s$',
142             color="blue", fontsize=12, ha="center")
143         plt.text(
144             self.t[self.t_noair_impact],
145             self.h_noair[self.t_noair_impact] + 1 * y_step,
146             f'{round(self.t[self.t_noair_impact], 2)}s$',
147             color="red", fontsize=12, ha="center")
148     else:
149         plt.ylim(-max_val, min_val)
150         plt.text(
151             self.t[self.t_impact],
152             self.h[self.t_impact] - 2 * y_step,
153             f'{round(self.t[self.t_impact], 2)}s$',
154             color="blue", fontsize=12, ha="center")
155         plt.text(
156             self.t[self.t_noair_impact],
157             self.h_noair[self.t_noair_impact] - 2 * y_step,
158             f'{round(self.t[self.t_noair_impact], 2)}s$',
159             color="red", fontsize=12, ha="center")
160
161     plt.plot(
162         self.t, self.h,
163         color="purple", label="With Air Resistance", zorder=3)
164     plt.plot(
165         self.t, self.h_noair,
166         color="green", label="No Air Resistance", zorder=2)
167
168     plt.scatter(
169         self.t[self.t_impact], self.h[self.t_impact],
170         color="blue", zorder=4, label="Impact Point (With Air)")
171     plt.scatter(
172         self.t[self.t_noair_impact], self.h_noair[self.t_noair_impact],
173         color="red", zorder=3, label="Impact Point (No Air)")
174
175     plt.grid(which="both", linestyle="-", color="gray", linewidth=1)
176     plt.minorticks_on()
177     plt.grid(which="minor", linestyle=":", color="black", linewidth=0.5)
178
179     plt.title(
180         f'Height over Time for {self.name} '
181         f'({self.positive_direction.capitalize()} Positive)')
182     plt.legend()
183     plt.xlabel(r'Time ($s$)')
184     plt.ylabel(r'Height ($m$)')
185     plt.show()
186
187     def plot_velocity(self):
188         """
189         Plot velocity vs. time for both cases.
190         """
191         plt.figure(1)
192         max_val = np.ceil(abs(self.v_noair[self.t_noair_impact]) / 5.0) * 5
193         y_step = max_val * 0.05
194         min_val = 2 * y_step
195         if self.sign == 1:
196             plt.ylim(-max_val, min_val)
197             plt.text(
198                 self.t[self.t_impact],

```

```

199         self.v[self.t_impact] - 2 * y_step,
200         f'{round(self.t[self.t_impact], 2)}s$',
201         color="blue", fontsize=12, ha="center")
202     plt.text(
203         self.t[self.t_impact],
204         self.v[self.t_impact] - 3.5 * y_step,
205         f'{round(self.v[self.t_impact], 2)}m/s$',
206         color="blue", fontsize=12, ha="center")
207     plt.text(
208         self.t[self.t_noair_impact],
209         self.v_noair[self.t_noair_impact] + 2.5 * y_step,
210         f'{round(self.t[self.t_noair_impact], 2)}s$',
211         color="red", fontsize=12, ha="center")
212     plt.text(
213         self.t[self.t_noair_impact],
214         self.v_noair[self.t_noair_impact] + 1 * y_step,
215         f'{round(self.v_noair[self.t_noair_impact], 2)}m/s$',
216         color="red", fontsize=12, ha="center")
217 else:
218     plt.ylim(-min_val, max_val)
219     plt.text(
220         self.t[self.t_impact],
221         self.v[self.t_impact] + 2.5 * y_step,
222         f'{round(self.t[self.t_impact], 2)}s$',
223         color="blue", fontsize=12, ha="center")
224     plt.text(
225         self.t[self.t_impact],
226         self.v[self.t_impact] + 1 * y_step,
227         f'{round(self.v[self.t_impact], 2)}m/s$',
228         color="blue", fontsize=12, ha="center")
229     plt.text(
230         self.t[self.t_noair_impact],
231         self.v_noair[self.t_noair_impact] - 2 * y_step,
232         f'{round(self.t[self.t_noair_impact], 2)}s$',
233         color="red", fontsize=12, ha="center")
234     plt.text(
235         self.t[self.t_noair_impact],
236         self.v_noair[self.t_noair_impact] - 3.5 * y_step,
237         f'{round(self.v_noair[self.t_noair_impact], 2)}m/s$',
238         color="red", fontsize=12, ha="center")
239
240
241 plt.plot(
242     self.t, self.v,
243     color="purple", label="With Air Resistance", zorder=3)
244 plt.plot(
245     self.t, self.v_noair,
246     color="green", label="No Air Resistance", zorder=2)
247
248 plt.scatter(
249     self.t[self.t_impact], self.v[self.t_impact],
250     color="blue", zorder=4, label="Impact Point (With Air)")
251 plt.scatter(
252     self.t[self.t_noair_impact], self.v_noair[self.t_noair_impact],
253     color="red", zorder=3, label="Impact Point (No Air)")
254
255 plt.grid(which="both", linestyle="-", color="gray", linewidth=1)
256 plt.minorticks_on()
257 plt.grid(which="minor", linestyle=":", color="black", linewidth=0.5)
258
259 plt.title(
260     f'VeLOCITY over Time for {self.name} '
261     f'({self.positive_direction.capitalize()} Positive)')
262 plt.legend()
263 plt.xlabel(r'Time (s)')
264 plt.ylabel(r'VeLOCITY (m/s)')

```

```

265 plt.show()
266
267 def plot_acceleration(self):
268     """
269     Plot acceleration vs. time for both cases.
270     """
271     plt.figure(2)
272     max_val = np.ceil(self.g / 2.0) * 2 + 1
273     y_step = max_val * 0.05
274     min_val = y_step
275     if self.sign == 1:
276         plt.ylim(-max_val, min_val)
277         plt.text(
278             self.t[self.t_impact],
279             self.a[self.t_impact] - 2 * y_step,
280             f'{round(self.t[self.t_impact], 2)}s$',
281             color="blue", fontsize=12, ha="center")
282         plt.text(
283             self.t[self.t_impact],
284             self.a[self.t_impact] - 3.5 * y_step,
285             f'{round(self.a[self.t_impact], 2)}m/s^2$',
286             color="blue", fontsize=12, ha="center")
287         plt.text(
288             self.t[self.t_noair_impact],
289             self.a_noair[self.t_noair_impact] + 2.5 * y_step,
290             f'{round(self.t[self.t_noair_impact], 2)}s$',
291             color="red", fontsize=12, ha="center")
292         plt.text(
293             self.t[self.t_noair_impact],
294             self.a_noair[self.t_noair_impact] + 1 * y_step,
295             f'{round(self.a_noair[self.t_noair_impact], 2)}m/s^2$',
296             color="red", fontsize=12, ha="center")
297     else:
298         plt.ylim(-min_val, max_val)
299         plt.text(
300             self.t[self.t_impact],
301             self.a[self.t_impact] + 2.5 * y_step,
302             f'{round(self.t[self.t_impact], 2)}s$',
303             color="blue", fontsize=12, ha="center")
304         plt.text(
305             self.t[self.t_impact],
306             self.a[self.t_impact] + 1 * y_step,
307             f'{round(self.a[self.t_impact], 2)}m/s^2$',
308             color="blue", fontsize=12, ha="center")
309         plt.text(
310             self.t[self.t_noair_impact],
311             self.a_noair[self.t_noair_impact] - 2 * y_step,
312             f'{round(self.t[self.t_noair_impact], 2)}s$',
313             color="red", fontsize=12, ha="center")
314         plt.text(
315             self.t[self.t_noair_impact],
316             self.a_noair[self.t_noair_impact] - 3.5 * y_step,
317             f'{round(self.a_noair[self.t_noair_impact], 2)}m/s^2$',
318             color="red", fontsize=12, ha="center")
319
320     plt.plot(
321         self.t, self.a,
322         color="purple", label="With Air Resistance", zorder=3)
323     plt.plot(
324         self.t, self.a_noair,
325         color="green", label="No Air Resistance", zorder=2)
326
327     plt.scatter(
328         self.t[self.t_impact], self.a[self.t_impact],
329         color="blue", zorder=4, label="Impact Point (With Air)")
330

```

```

331     plt.scatter(
332         self.t[self.t_noair_impact], self.a_noair[self.t_noair_impact],
333         color="red", zorder=3, label="Impact Point (No Air)")
334
335     plt.grid(which="both", linestyle="-", color="gray", linewidth=1)
336     plt.minorticks_on()
337     plt.grid(which="minor", linestyle=":", color="black", linewidth=0.5)
338
339     plt.title(
340         f'Acceleration over Time for {self.name} '
341         f'({self.positive_direction.capitalize()} Positive)')
342     plt.legend()
343     plt.xlabel(r'Time ($s$)')
344     plt.ylabel(r'Acceleration $(m/s^2)$')
345     plt.show()
346
347     def run_simulation(self):
348         """
349         Run the full simulation.
350         """
351         self.calculate()
352         self.find_impact_times()
353         self.display_results()
354         self.plot_height()
355         self.plot_velocity()
356         self.plot_acceleration()
357
358     # Run the simulation
359     if __name__ == "__main__":
360         sim_1 = FallingSphere(
361             "Sphere 1",
362             distance_from_ground = 50, end_time = 13.0, mass = 0.1, C = 0.05)
363         sim_1.run_simulation()
364
365         sim_2 = FallingSphere(
366             "Sphere 2",
367             distance_from_ground = 50, end_time = 13.0, mass = 0.1, C = 0.05,
368             positive_direction = "downwards")
369         sim_2.run_simulation()

```

Code 2.3: Simulation of falling spheres, over engineered to a high degree