Kinematic equations and projectile motion

Software 2 – Mathematics Python Labs

Sakari Lukkarinen

Metropolia University of Applied Sciences

Contents

- Constant velocity and acceleration
- Example – a motorist and a police car
- Projectile motion
- Example – tossing a ball
- Creating animations (bonus)
- Next steps

Constant velocity and acceleration

Kinematic equation (1D)

$$x(t) = x_0 + v_0 t + \frac{1}{2} a t^2$$

- x_0 = initial position (m)
- v_0 = constant velocity (m/s)
- a = constant acceleration (m/s^2)

Travelled distance

How to calculate, when

(a) velocity is constant?

(b) acceleration is constant?

(c) in general (i.e. $v(t)$ continuous)?

(a) If v constant, then $\boxed{x = v \cdot t}$

(b) If acceleration is constant, then

$$\boxed{v(t) = v_0 + at}$$

$$\boxed{x = v_{ave} \cdot t = v_0 \cdot t + \frac{1}{2} a t^2}$$

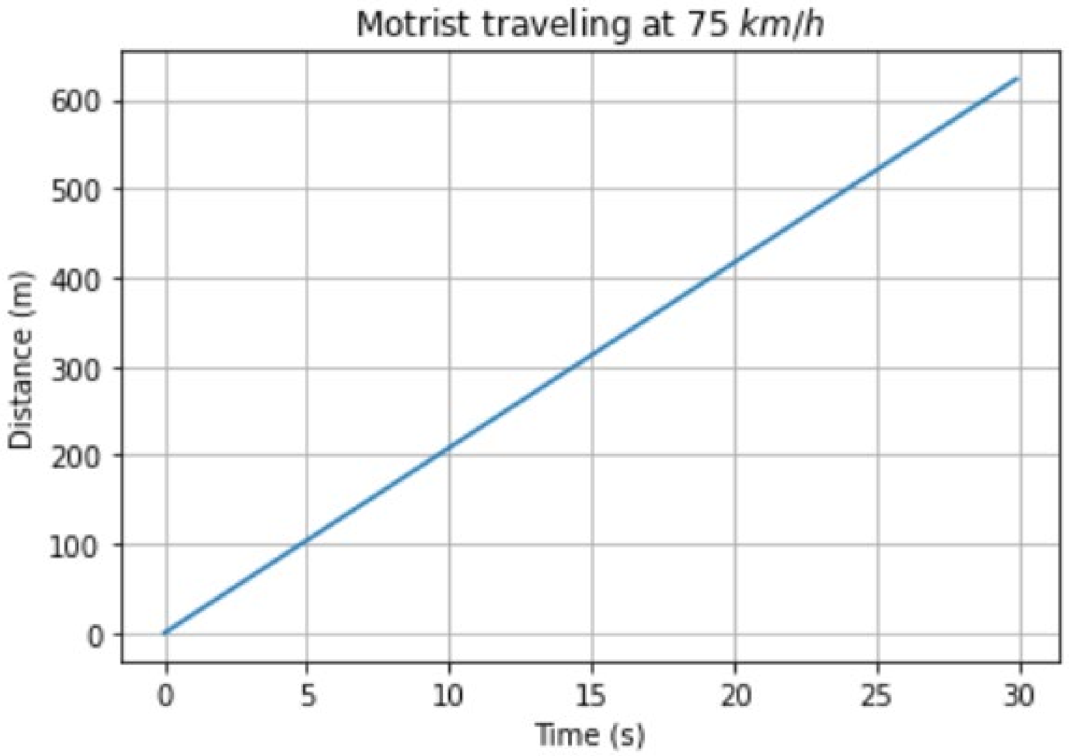
Source: Salin, Timo. Physics lecture notes

Example – a motorist and a police car

EXAMPLE A speeding motorist zooms through a $50\frac{km}{h}$ zone at $75\frac{km}{h}$ without noticing a stationary police car. The police officer immediately heads after the speeder at $a = 2{,}5\frac{m}{s^2}$. When the officer catches up to the speeder, how far down the road are they, and how fast is the police car going?

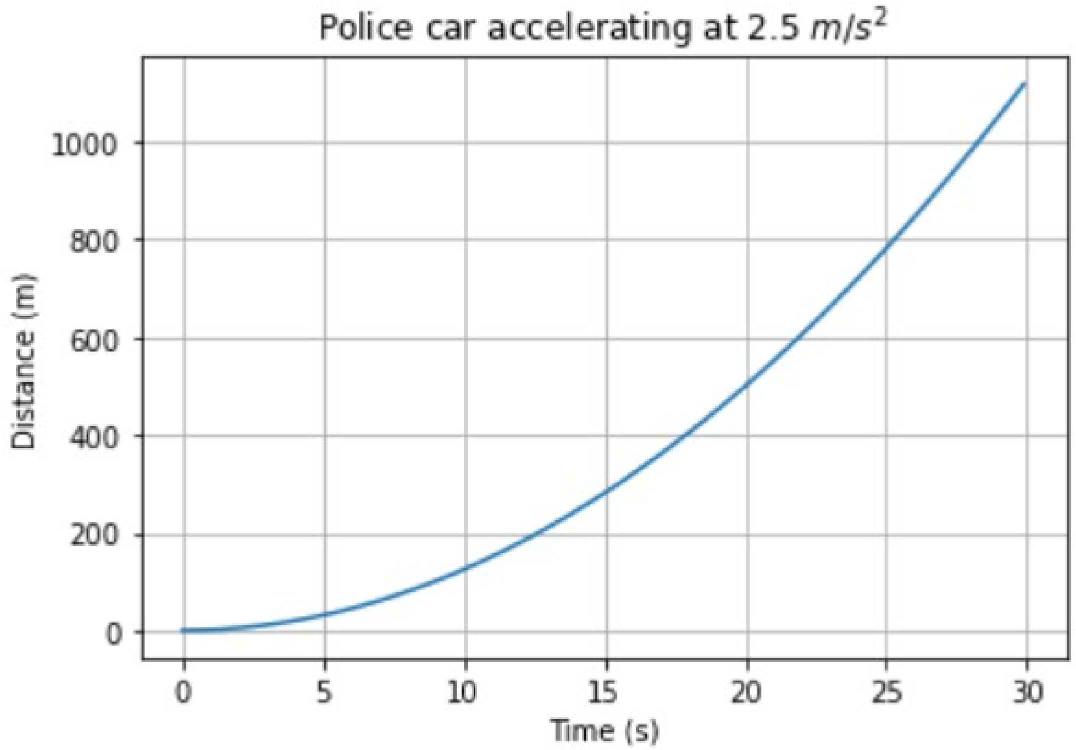
Constant velocity – the motorist

```python
v0 = 75 * 1000/3600 # km/h => m/s
t = np.arange(0, 30, 0.1)
x = v0*t
plt.plot(t, x)
plt.xlabel('Time (s)')
plt.ylabel('Distance (m)')
plt.title('Motrist traveling at 75
plt.grid()
plt.show()
```



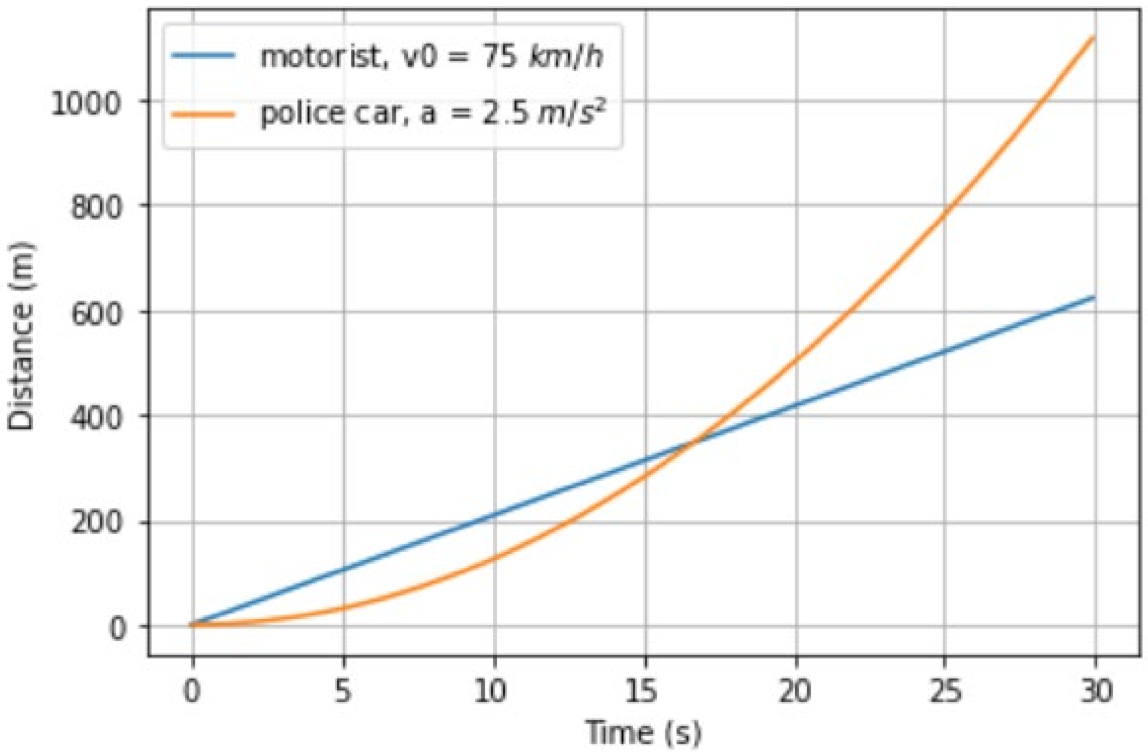
Constant acceleration – the police car

```
a = 2.5 # m/s^2
x2 = 1/2*a*t**2
plt.plot(t, x2)
plt.xlabel('Time (s)')
plt.ylabel('Distance (m)')
plt.title('Police car accelerating
plt.grid()
plt.show()
```



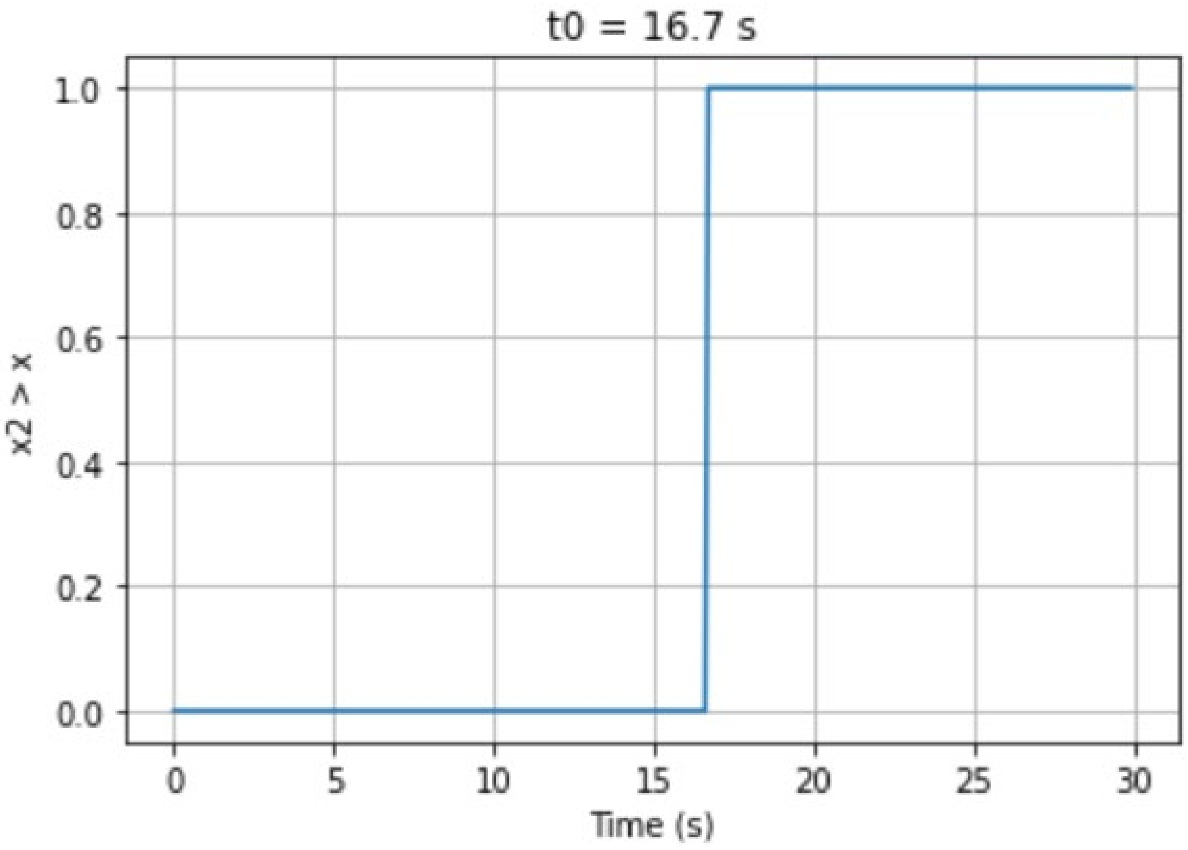
Comparison

```
plt.plot(t, x, label = 'motorist,
plt.plot(t, x2, label = 'police c
plt.xlabel('Time (s)')
plt.ylabel('Distance (m)')
plt.legend()
plt.grid()
plt.show()
```



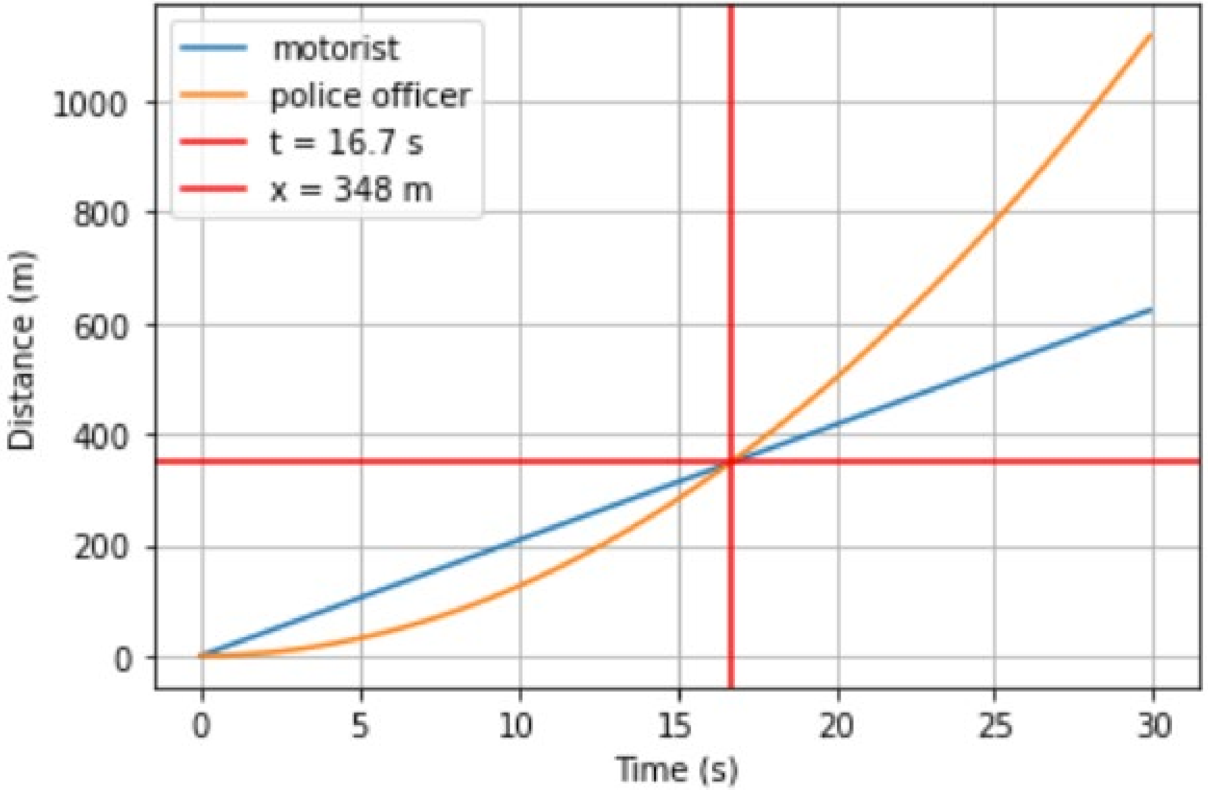
When does the officer catch the speeder?

```
# When is x2 greater than x?
i = (x2 > x)
plt.plot(t, i)
plt.xlabel('Time (s)')
plt.ylabel('x2 > x')
plt.grid()

# When does that happen?
t0 = np.min(t[i])
plt.title(f't0 = {t0} s')
plt.show()
```



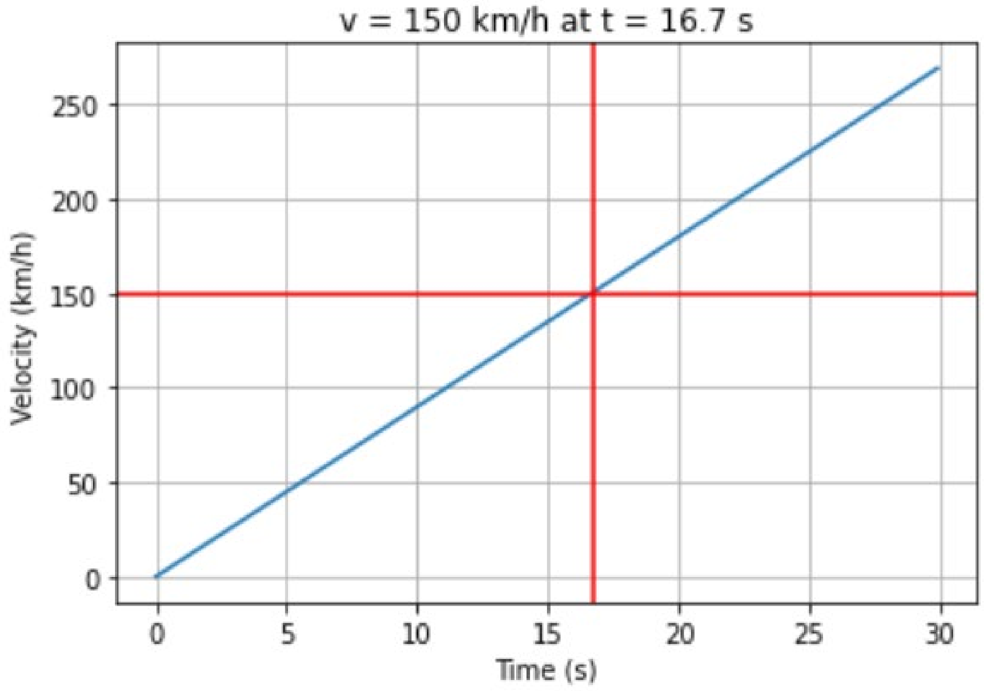
Catching time

```
i2 = (t == t0)
x0 = x[i2]
plt.plot(t, x, label = 'motorist')
plt.plot(t, x2, label = 'police officer'
plt.axvline(t0, color = 'red', label = f
plt.axhline(x0, color = 'red', label = f
plt.xlabel('Time (s)')
plt.ylabel('Distance (m)')
plt.legend()
plt.grid()
plt.show()
```



Police car's velocity at the end

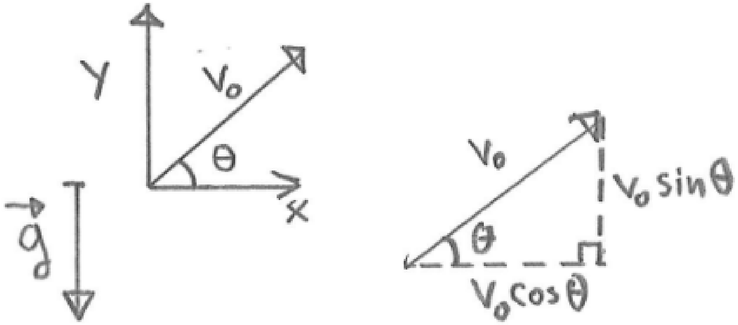
```
v = a*t *3600/1000 # m/s ==> km/h
i2 = (t == t0)
v2 = v[i2]

plt.plot(t, v)
plt.axvline(t0, color = 'red')
plt.axhline(v2, color = 'red')
plt.xlabel('Time (s)')
plt.ylabel('Velocity (km/h)')
plt.title(f'v = {v2[0]:.0f} km/h at t = {t0} s')
plt.grid()
plt.show()
```



Projectile (2D) motion

Projectile (2D) motion

A projectile is an object that is launched into the air and then moves predominantly under the influence of gravity.



$$
\begin{aligned}
v_x &= v_0 \cos(\theta) \\
v_y &= v_0 \sin(\theta) - gt \\
x &= x_0 + v_0 \cos(\theta) t \\
y &= y_0 + v_0 \sin(\theta) t - \frac{1}{2} g t^2
\end{aligned}
$$

Salin, T. (2022) . Physics, Period 1.

Trigonometric functions and angles

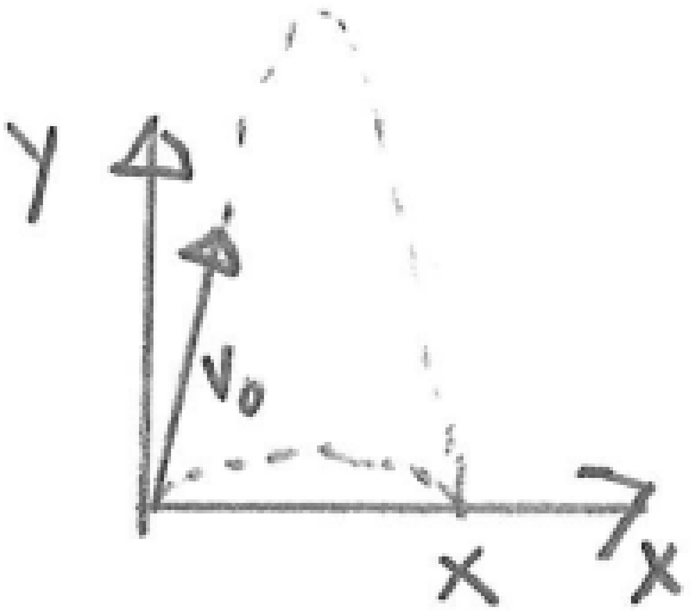
NOTE: The angle for trigonometric functions (sin, cos, tan) should be given in radians.

For that reason we need np.deg2rad() function to convert the degrees to radians. See also: Conversions of angles.

Turns	Radians	Degrees
0 turn	0 rad	0°
1/12 turn	π/6 rad	30°
1/8 turn	π/4 rad	45°
1/6 turn	π/3 rad	60°
1/4 turn	π/2 rad	90°
1/3 turn	2π/3 rad	120°
1/2 turn	π rad	180°
3/4 turn	3π/2 rad	270°
1 turn	2π rad	360°

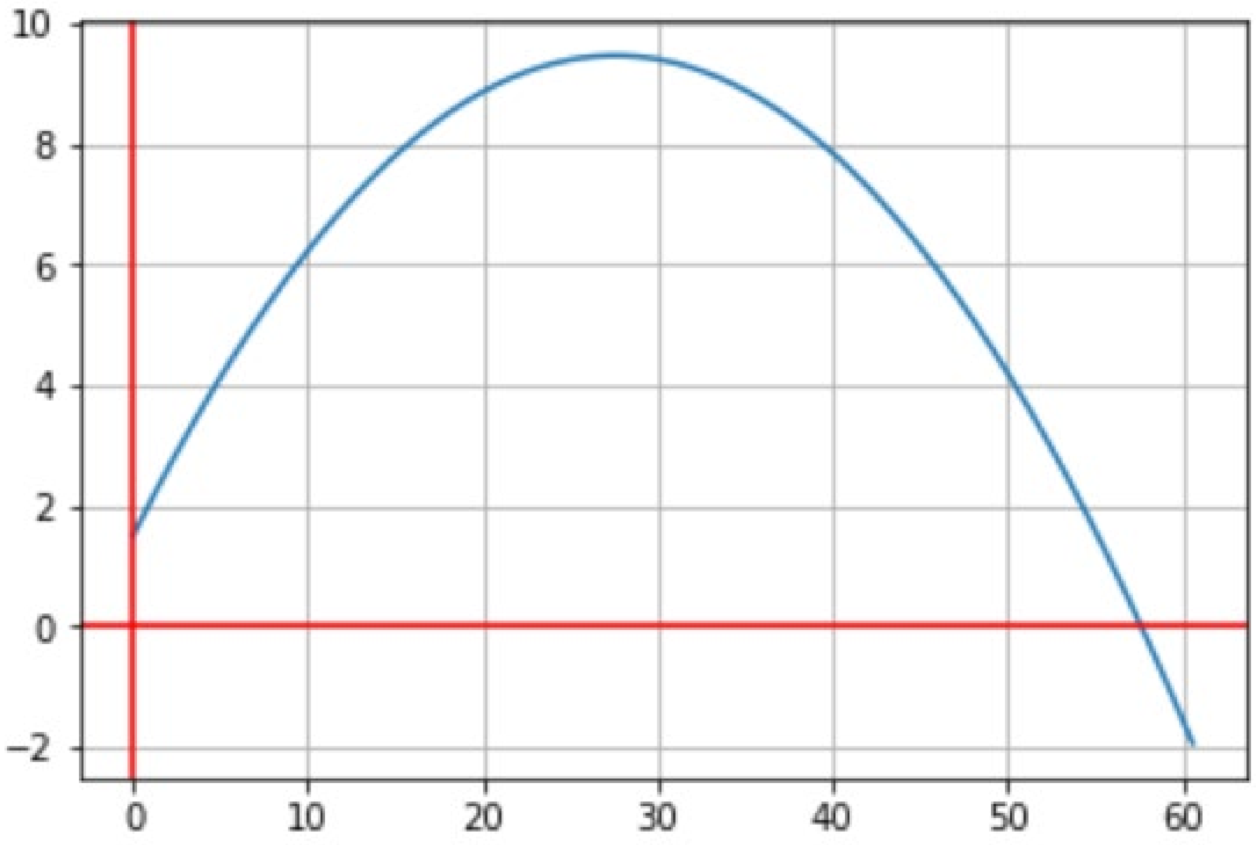
Example

You toss a ball at speed of 25.0 m/s and It leaves your hand at 1.5 m above a floor in angle of 30 degrees. How far does the ball flight? Draw the trajectory of the ball.



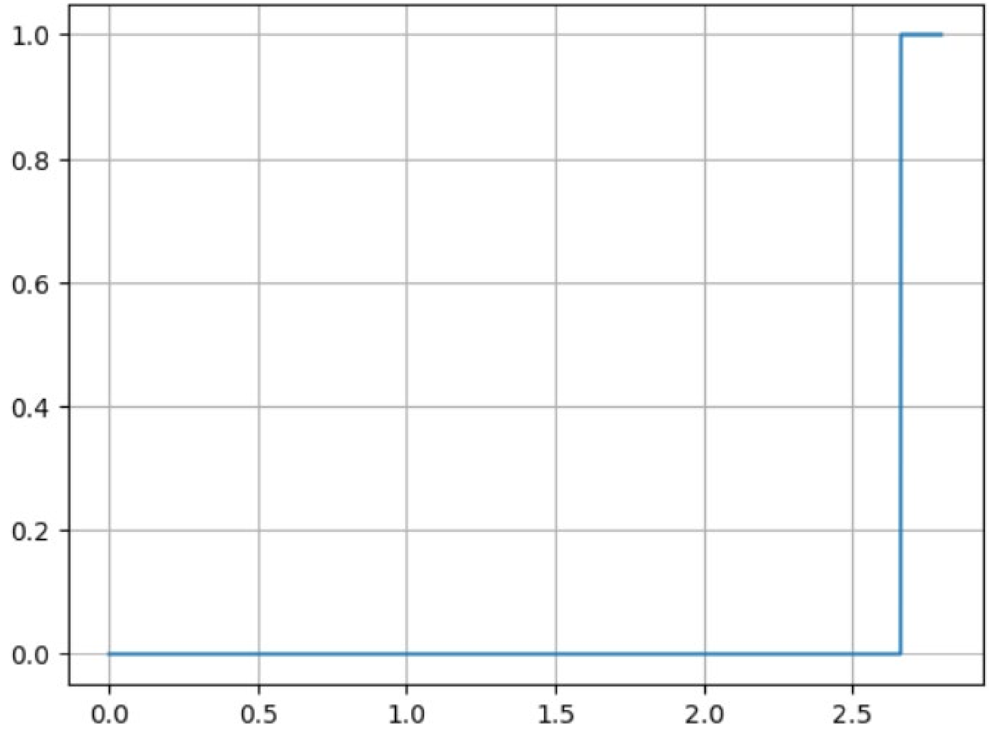
Example – tossing a ball (2D)

```
x0, y0 = 0, 1.5
v0 = 25.0
theta = np.deg2rad(30)
g = 9.81

t = np.arange(0, 2.8, 0.001)

vx = v0*np.cos(theta)
vy = v0*np.sin(theta) - g*t
x = x0 + v0*np.cos(theta)*t
y = y0 + v0*np.sin(theta)*t - 1/2*g*t**2

plt.plot(x, y)
plt.axhline(0, color = 'red')
plt.axvline(0, color = 'red')
plt.grid()
```



When does the ball land to the floor?

```python
i = (y <= 0)
plt.plot(t, i)
plt.grid()
```




```python
t_end = np.min(t[i])
print(f't_end = {t_end} s')
```

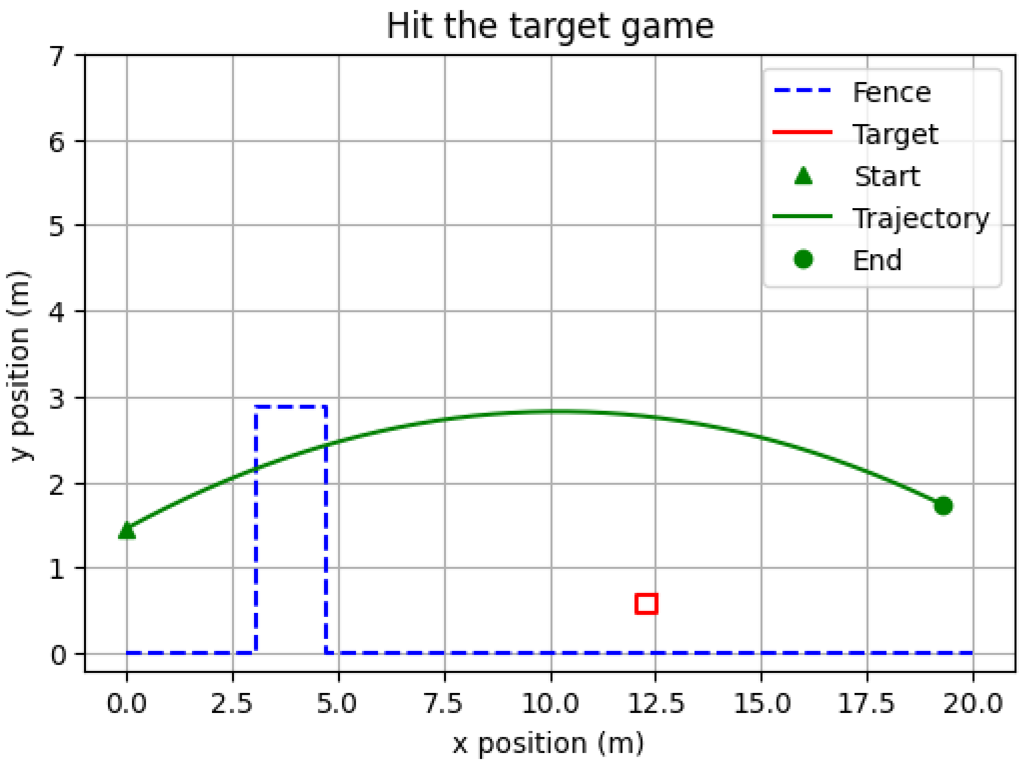
```
t_end = 2.664 s
```

What is the landing location?

```python
i_end = (t == t_end)
x_end = x[i_end][0]
print(f'The ball lands at x = {x_end:.1f} m')
```

```
The ball lands at x = 57.7 m
```

Hit the target game (Problem 2)



In the second problem, your task is to hit the target by throwing the ball and try to avoid to hit the fence.

The code for calculating and displaying the trajectory is given, but you should find the right initial and end values.

Creating animations (BONUS)

FuncAnimation makes an animation by repeatedly calling a given graphics function. The animation is then converted to HTML presentation by using IPython.display module's HTML class.

We use the same data as in previous example, but now we reduce the time step in order to make the simulation run smoother.

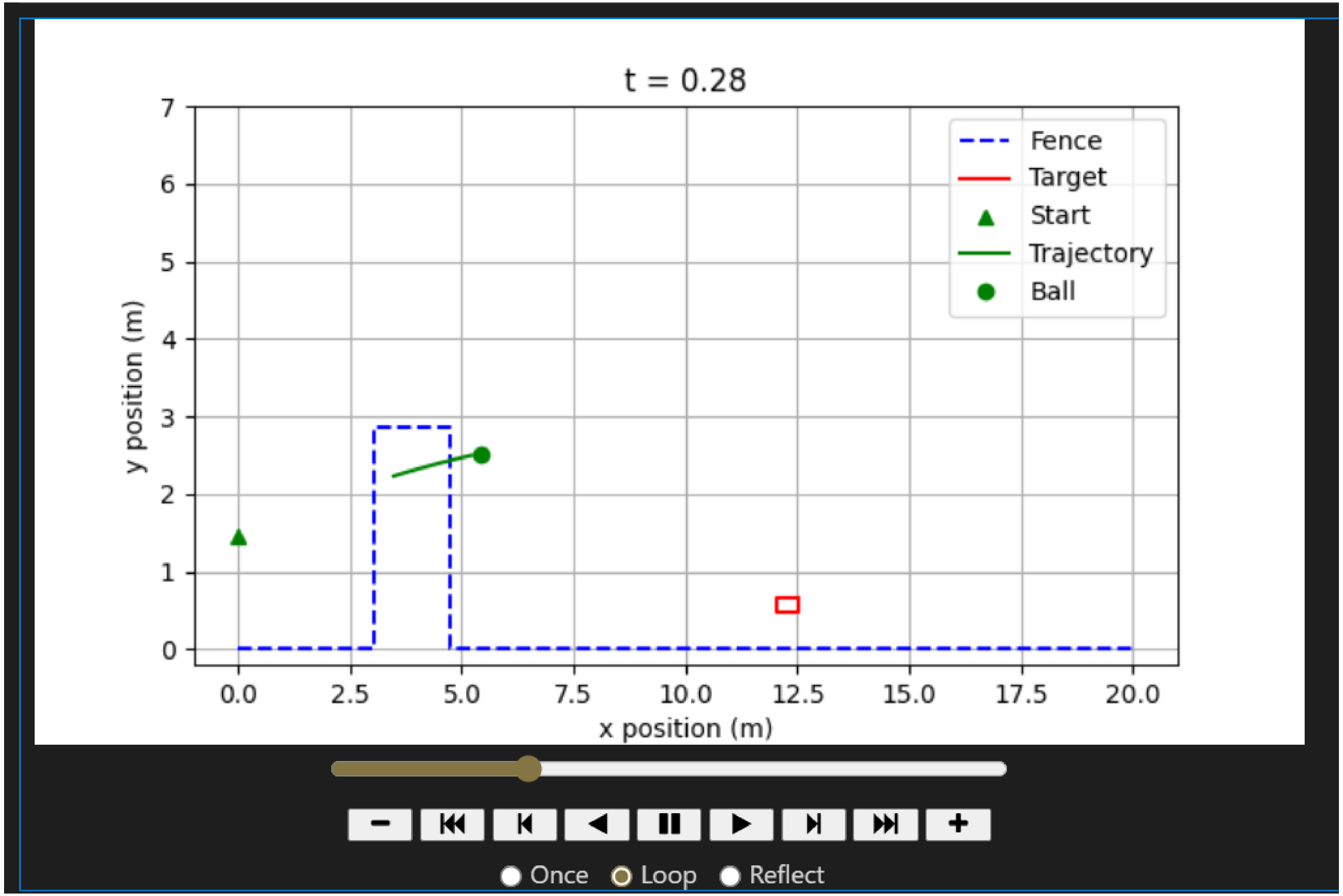
```
from IPython.display import HTML
from matplotlib.animation import FuncAnimation
```

Animation example

```
t = np.arange(0, 2.67, 0.02)

vx = v0*np.cos(theta)
vy = v0*np.sin(theta) - g*t
x = x0 + v0*np.cos(theta)*t
y = y0 + v0*np.sin(theta)*t - 1/2*g*t**2

# Initialize the graph
fig, ax = plt.subplots()
l, = ax.plot(x, y)
l2, = ax.plot(x[-1], y[-1], 'bo')

plt.axhline(0, color = 'red')
plt.axvline(0, color = 'red')
plt.grid(True)
```

```
# Animation function
def animate(i):
    l.set_data(x[:i], y[:i])
    l2.set_data(x[i], y[i])
    ax.set_title(f't = {t[i]:.2f} s')

# Create animation
ani = FuncAnimation(fig, animate, frames=len(x))

# Show the animation
HTML(ani.to_jshtml())
```

Animation controls



Next steps

- Practice – Lab 5
  - Moodle Quiz for practise
  - Notebook can be found from OMA assignments
- Read more
  - Salin, T. Physics lecture notes.
  - What are the kinematic formulas? (article) | Khan Academy
  - Projectile motion - Wikipedia
- Extra
  - matplotlib.animation — Matplotlib documentation