Solving differential equations and introduction to SciPy

Python 4

General Plan

Assignments. Any issue with the last assignment? How long did it take you?

Plan.

0. Feedback on last week's exercises
1. First order Ordinary Differential Equations (ODEs):
   Euler Method and RK4 method
2. Systems of coupled first order ODEs
3. Second order ODEs
4. Solving ODEs with SciPy

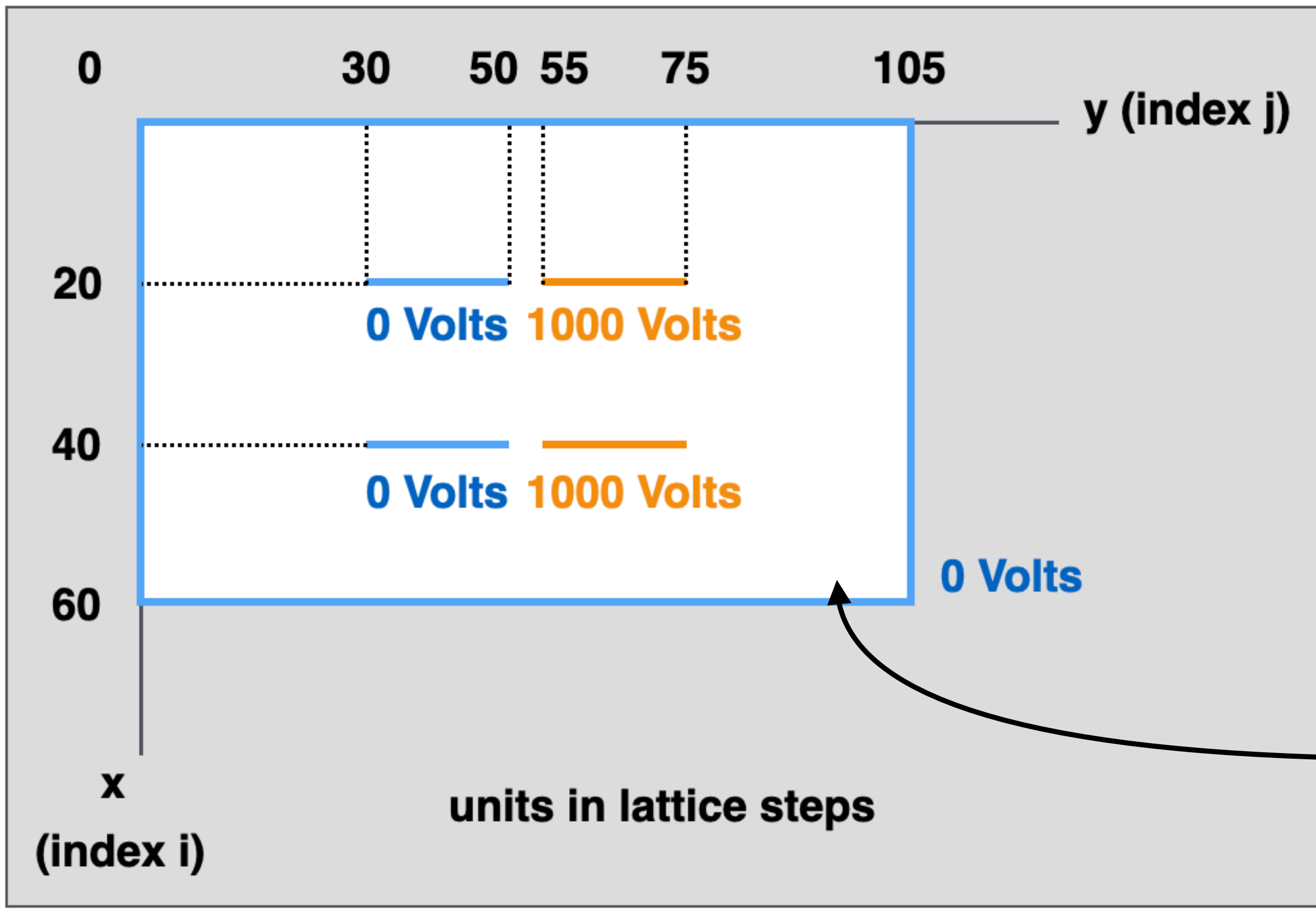
Part 0

Last week's exercises

Do you have any question?

Is there anything you are not sure you have understood?

Exercise 4



Start with:

$V_{i,j} = 0$ everywhere inside the cavity

but on the conductors

At each iteration:

$$V_{i,j} = \frac{V_{i-1,j} + V_{i,j-1} + V_{i+1,j} + V_{i,j+1}}{4}$$

Exo 4: Python Loops

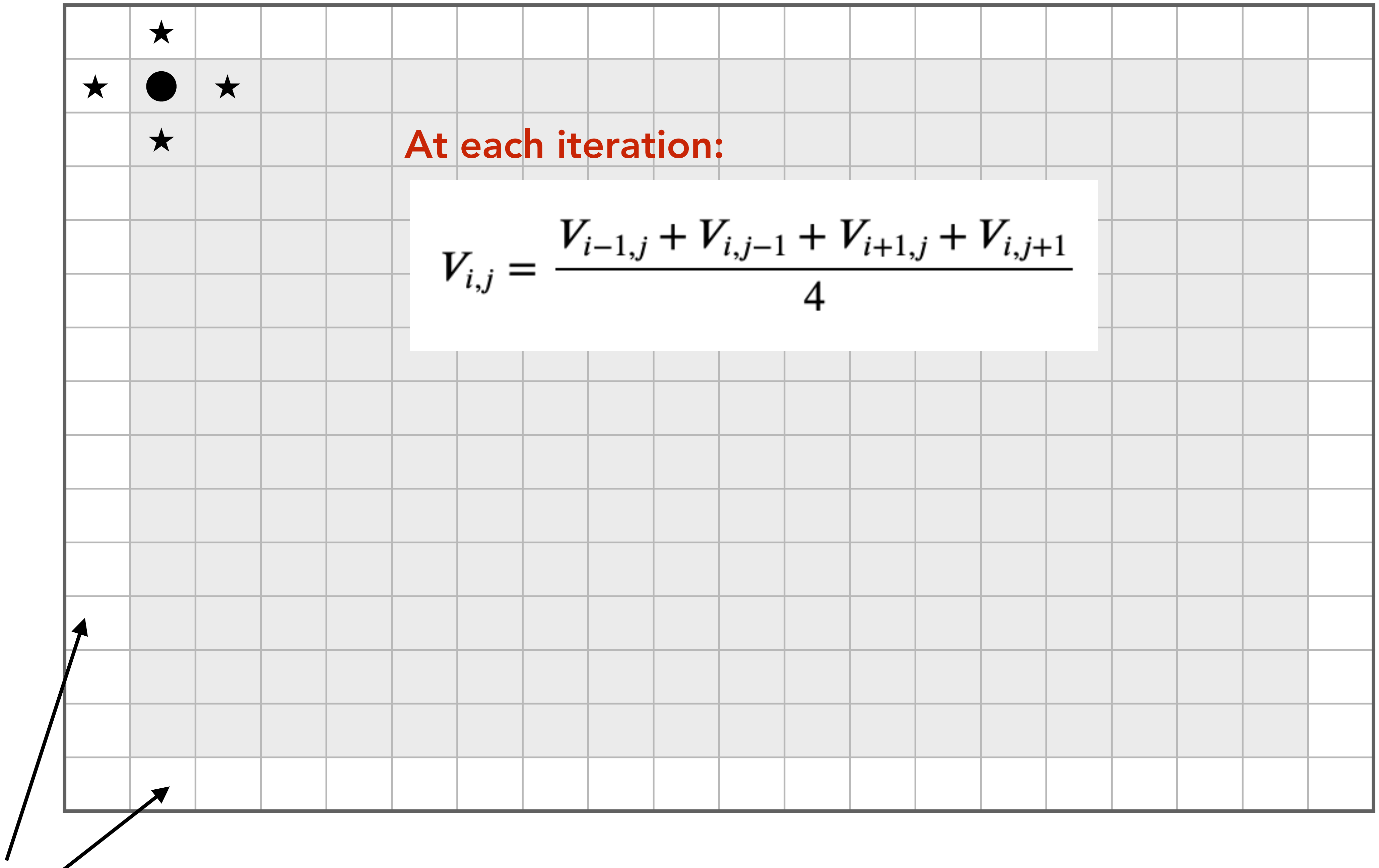
Exo 4: Python Loops

```python
# Update the potential inside the cavity:
def Update(V):
    V2=np.copy(V)
    for i in range(1,60):
        for j in range(1,105):
            if (i!=20 and i!=40) or (j<30) or (j>50 and j<55) or (j>75):
                V[i,j]=(V2[i,j-1]+V2[i,j+1]+V2[i-1,j]+V2[i+1,j])/4.    # Local update rule
```

```python
# Iterations:
def Solve_Laplace(V_init, Nit):
    V=np.copy(V_init)          # Make an initial copy
    for _ in range(Nit):       # Loop over the number of iterations
        Update(V)
    return V
```

Exo 4: NumPy operations

Exo 4: NumPy operations



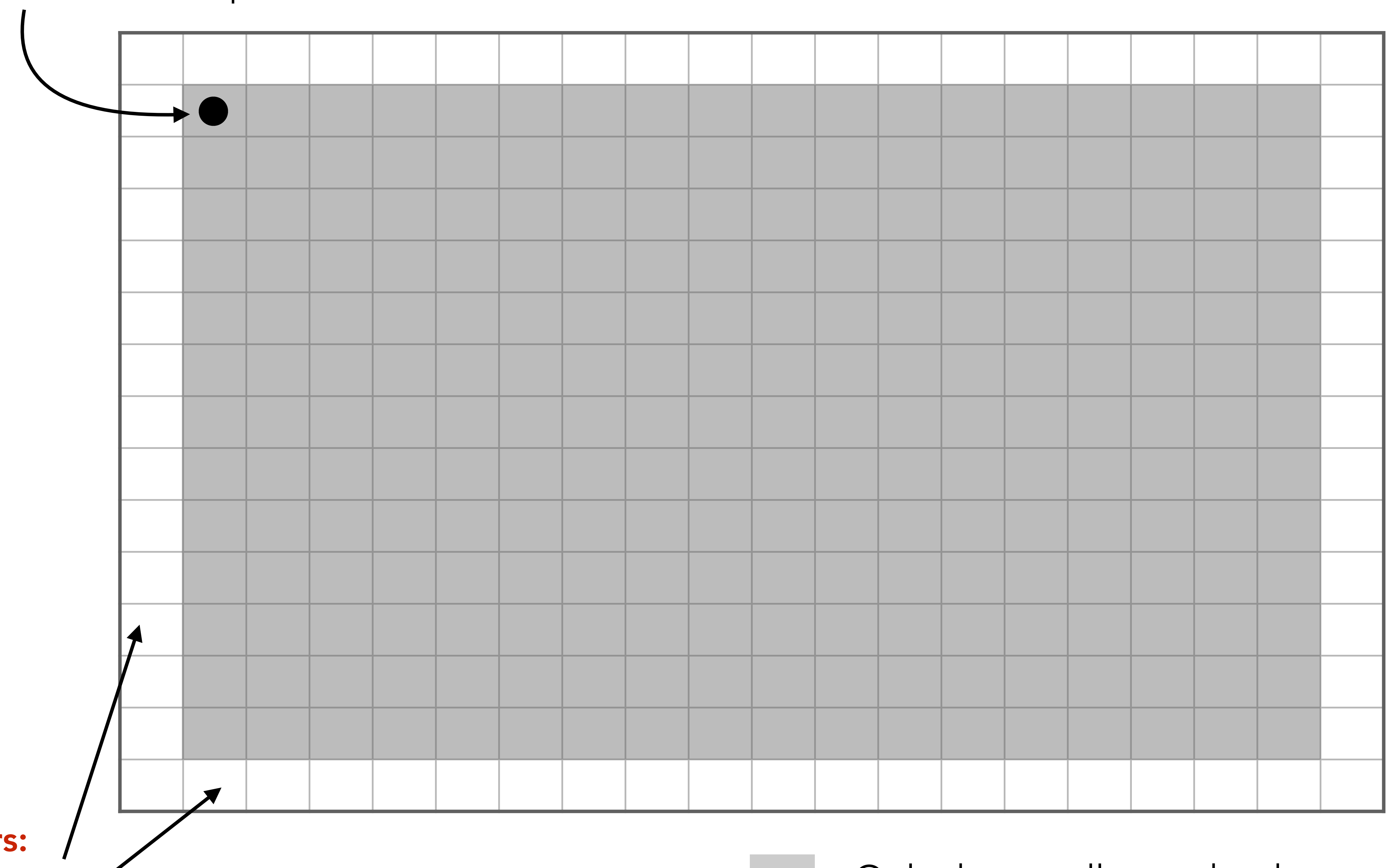
At each iteration:

$$V_{i,j} = \frac{V_{i-1,j} + V_{i,j-1} + V_{i+1,j} + V_{i,j+1}}{4}$$

Borders: stay at V=0 —> value not updated

Exo 4: NumPy operations

First elements: top left of this slice



Borders:

stay at V=0

—> value not updated

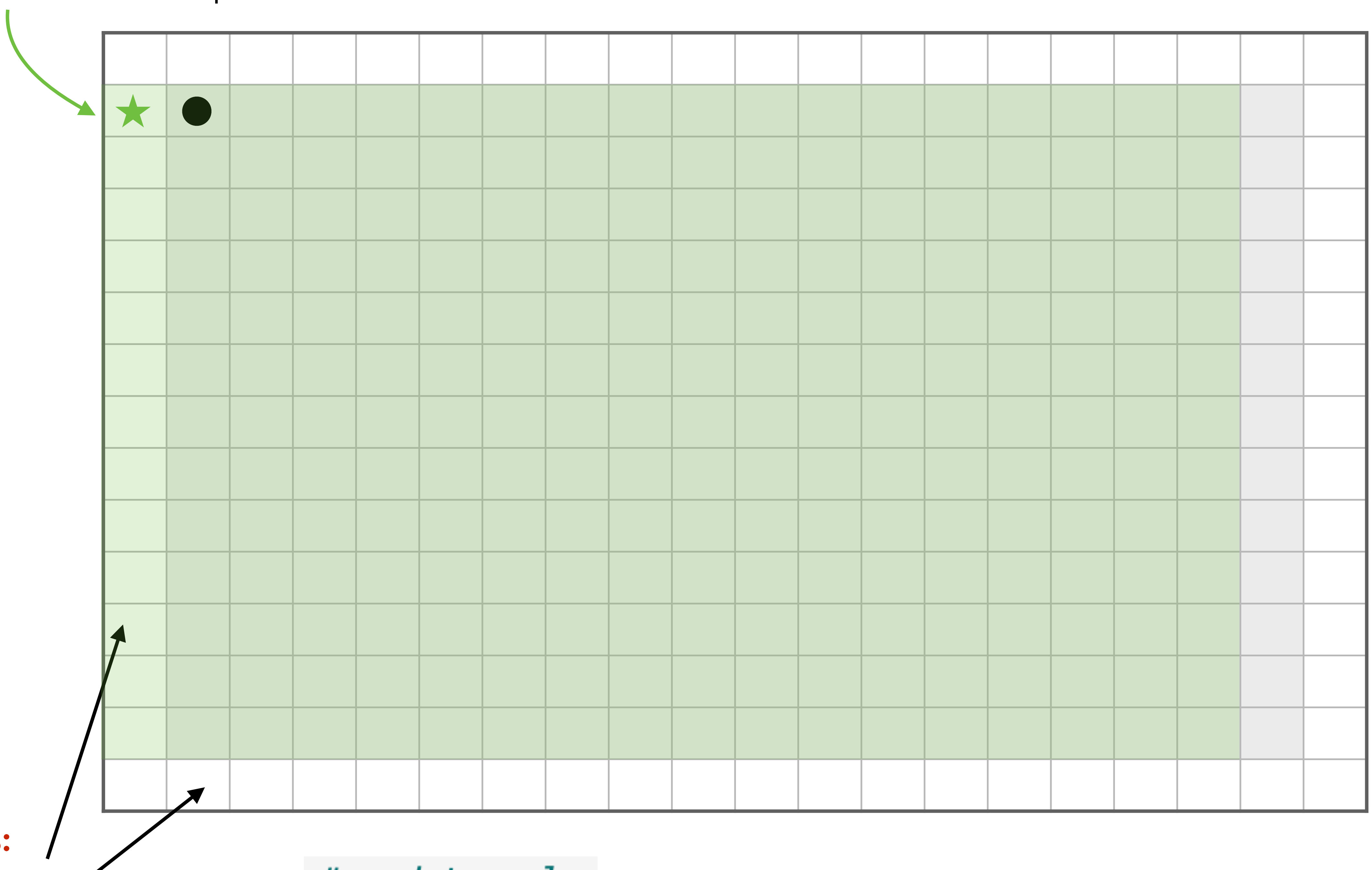
```
V[1:-1,1:-1]
```

Only these cells need to be updated

Exo 4: NumPy operations

First elements: top left of this slice

● = ★ +



Borders:

stay at V=0

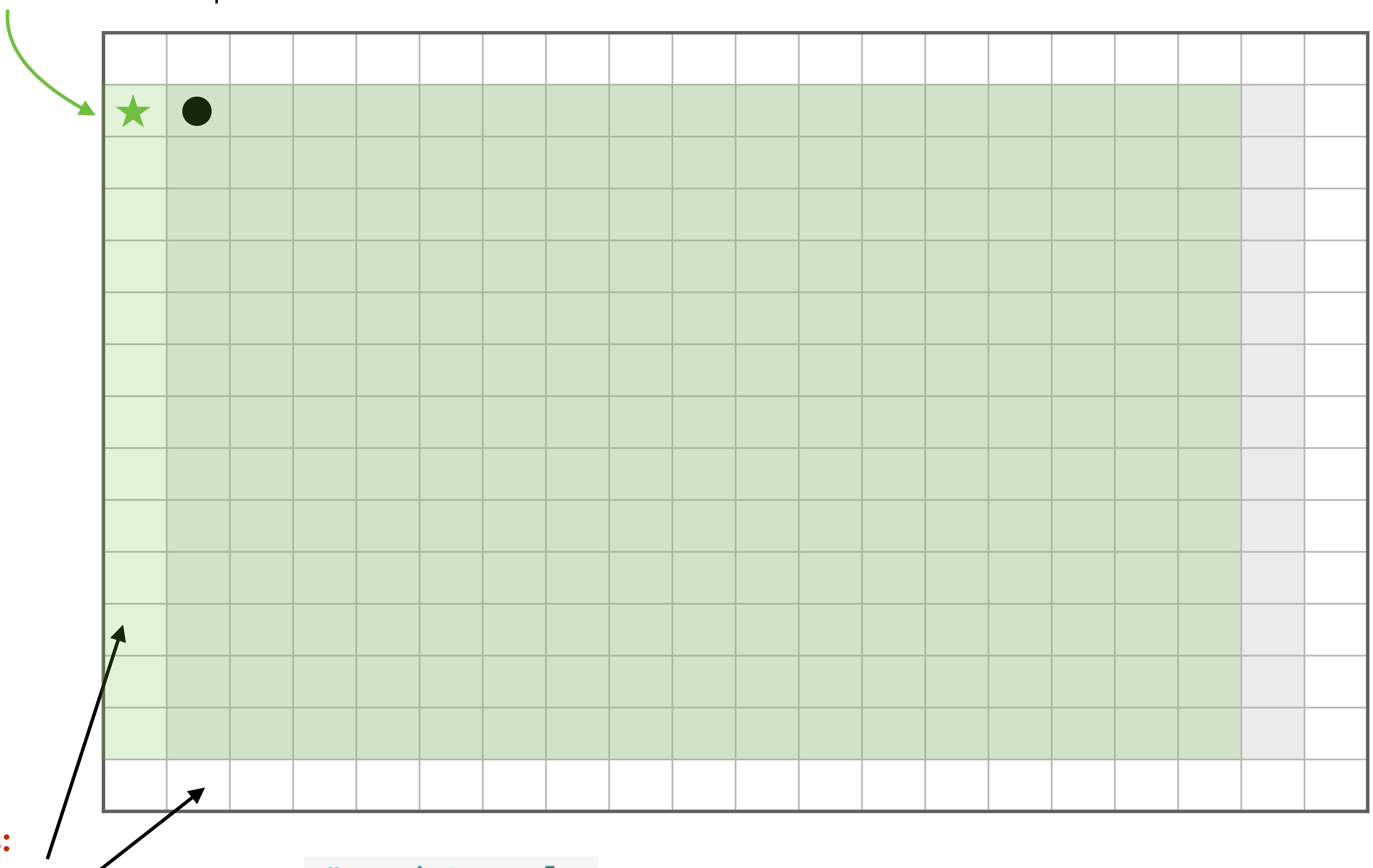
—> value not updated

```
# update rule
V[1:-1,1:-1] =
```

Exo 4: NumPy operations

First elements: top left of this slice

● = ★ +



Borders:

stay at V=0

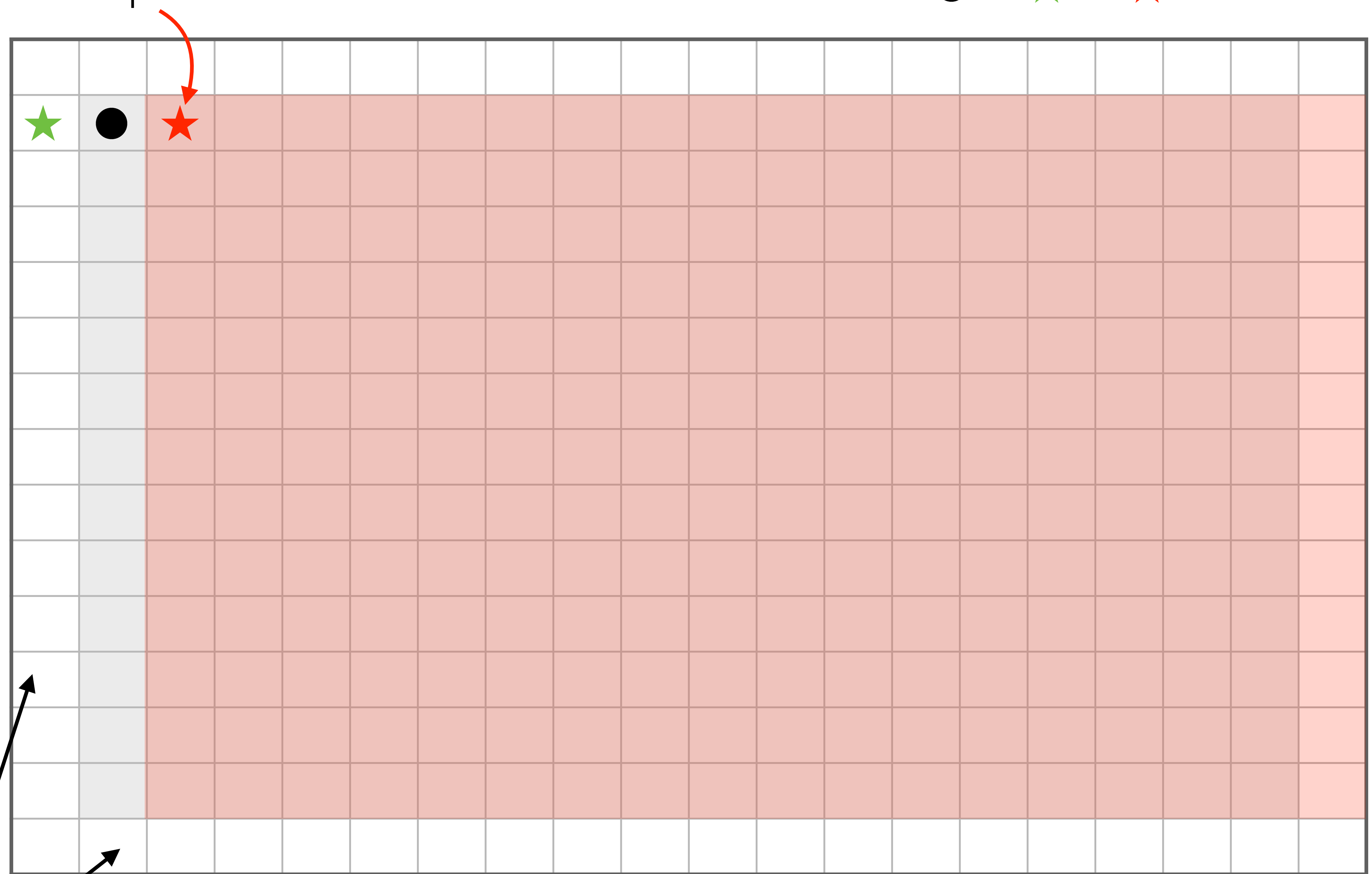
—> value not updated

```python
# update rule
V[1:-1,1:-1] = (V2[:-2,1:-1] +
```

Exo 4: NumPy operations

First elements: top left of this slice

● = ★ + ★ +



Borders:

stay at V=0

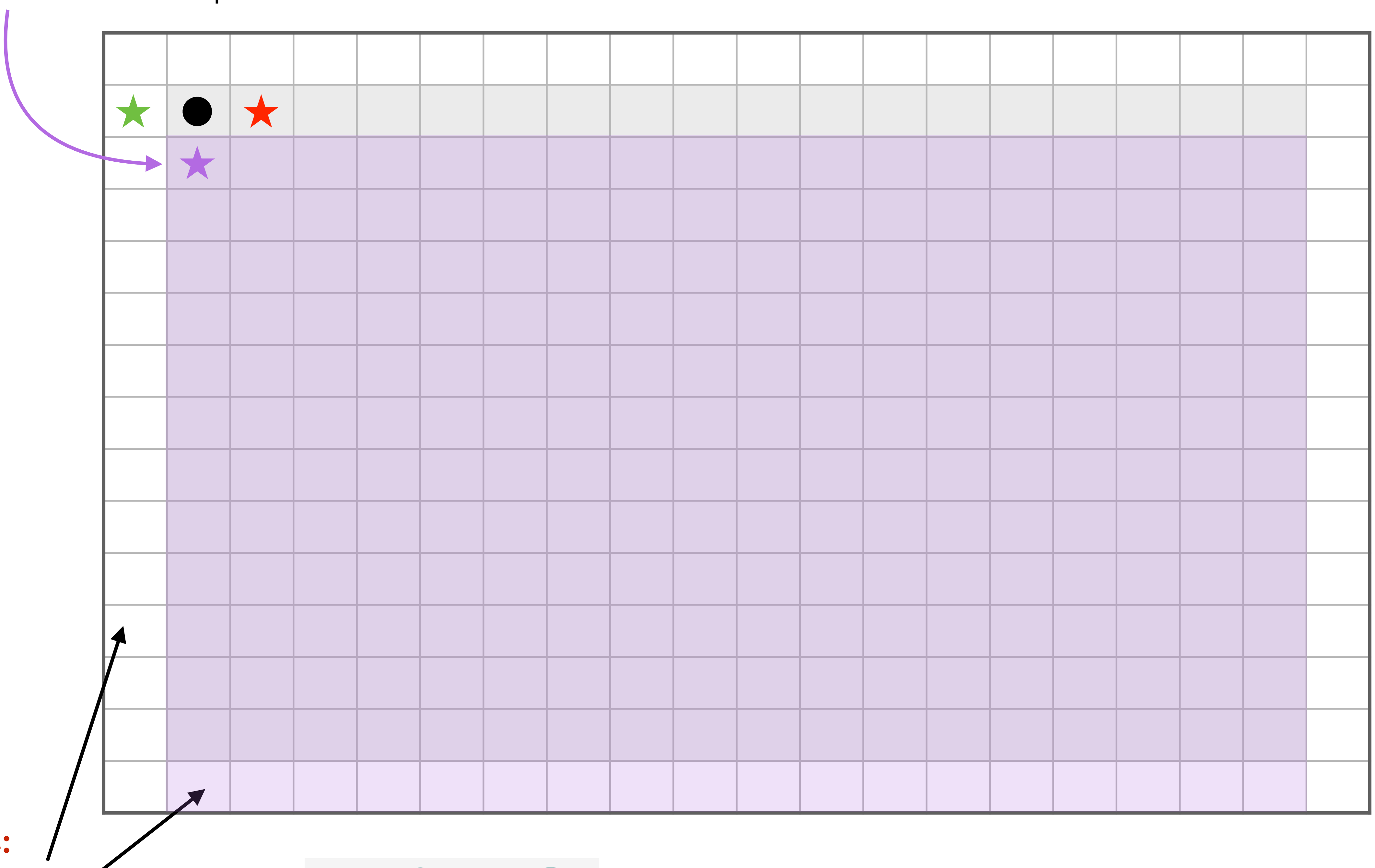
—> value not updated

```
# update rule
V[1:-1,1:-1] = (V2[:-2,1:-1] + V2[2:,1:-1] +
```

Exo 4: NumPy operations

First elements: top left of this slice

● = ★ + ★ + ★ +



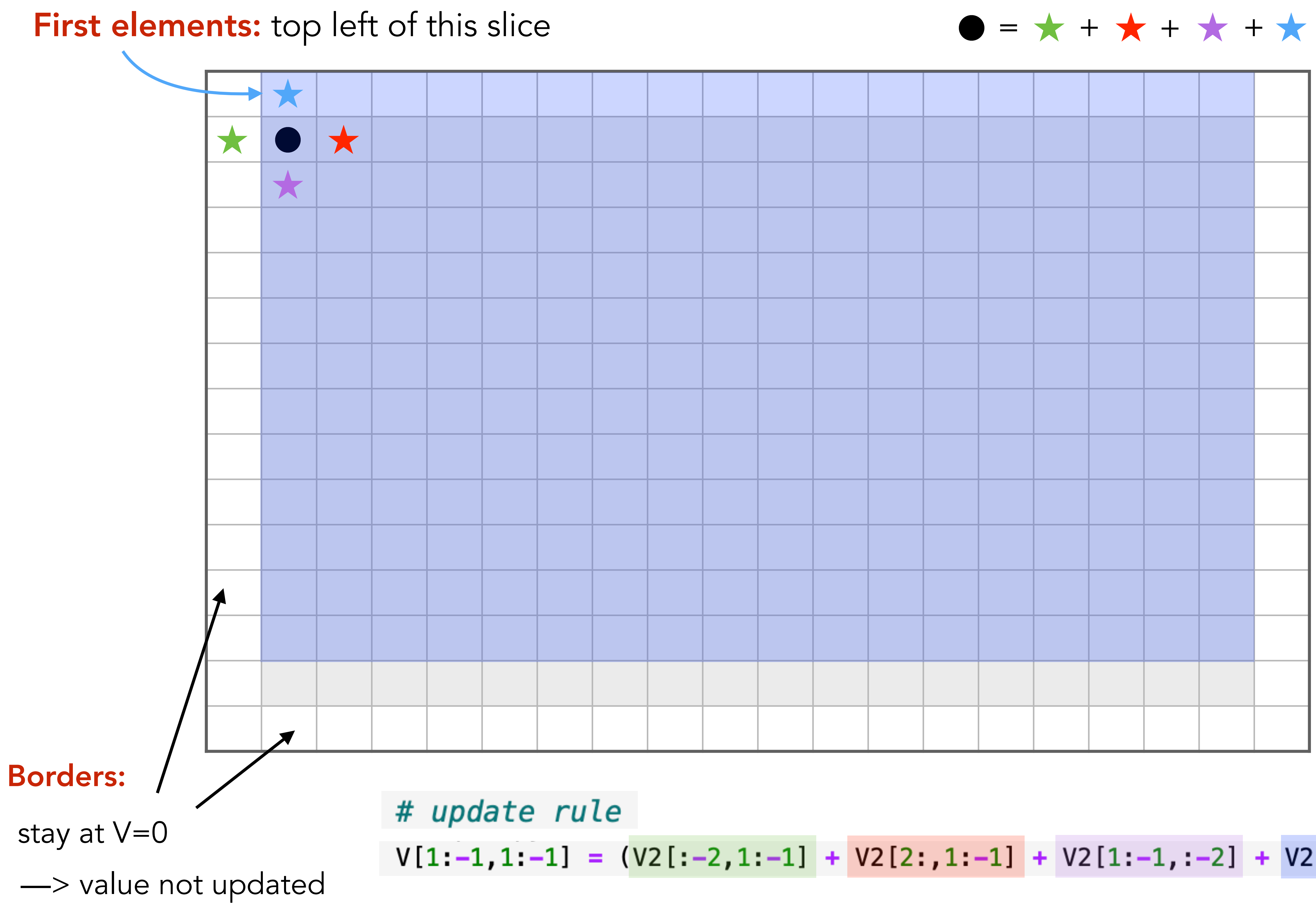
Borders:

stay at V=0

—> value not updated

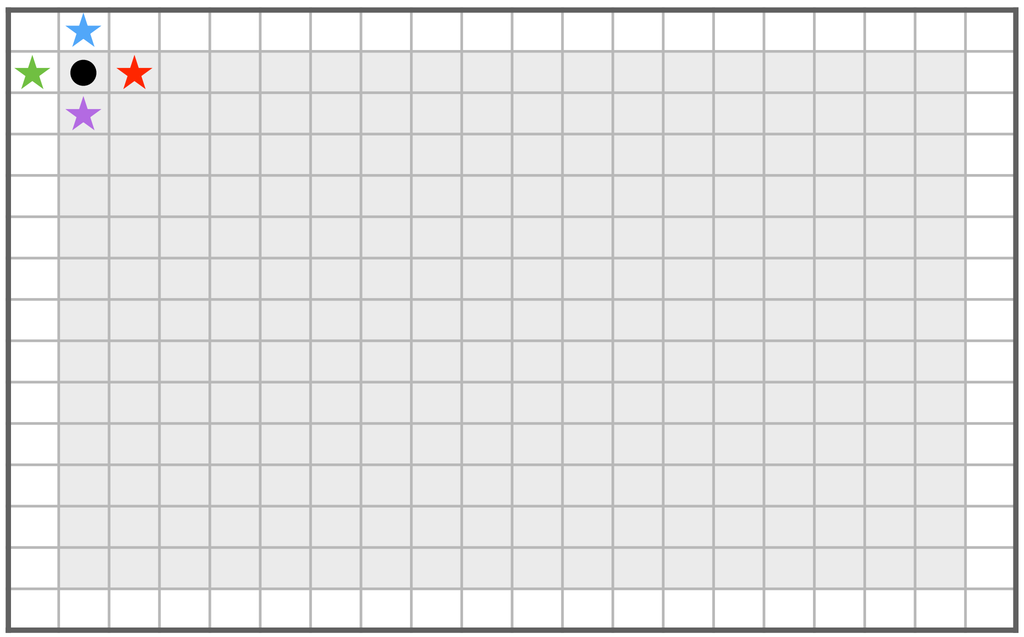
```
# update rule
V[1:-1,1:-1] = (V2[:-2,1:-1] + V2[2:,1:-1] + V2[1:-1,:-2] + V2[1:-1,2:])/4.
```

Exo 4: NumPy operations




```python
# update rule
V[1:-1,1:-1] = (V2[:-2,1:-1] + V2[2:,1:-1] + V2[1:-1,:-2] + V2[1:-1,2:])/4.
```

Exo 4: NumPy operations



At each iteration:

$$V_{i,j} = \frac{V_{i-1,j} + V_{i,j-1} + V_{i+1,j} + V_{i,j+1}}{4}$$

For each cell: ● = ★ + ★ + ★ + ★

For the whole matrix:

= + + +

```python
def Update2(V):
    V2=np.copy(V)
    V[1:-1,1:-1] = (V2[:-2,1:-1] + V2[2:,1:-1] + V2[1:-1,:-2] + V2[1:-1,2:])/4.

    # we re-fix the value of on the 4 conductors
    V[20, 55:76] = 1000
    V[40, 55:76] = 1000
    V[20, 30:51] = 0
    V[40, 30:51] = 0
```

>>> Q1

Matplotlib

pyplot:

```
import matplotlib.pyplot as plt
```

Option in Jupiter notebook:

```
%matplotlib inline
%matplotlib notebook
```

>>> Ex1

Matplotlib

pyplot:

```
import matplotlib.pyplot as plt
```

Option in Jupiter notebook:

```
%matplotlib inline
%matplotlib notebook
```

>>> Ex1

Simple plots:

```
x = np.linspace(0, 10)  # values linearly sampled from 0 to 10
y = np.sin(x)           # sine of these values
plt.plot(x, y);         # plot with line
```

2 arrays with same size

Matplotlib

pyplot:

```
import matplotlib.pyplot as plt
```

Option in Jupiter notebook:

```
%matplotlib inline
%matplotlib notebook
```

>>> Ex1

Simple plots:

```
x = np.linspace(0, 10)   # values linearly sampled from 0 to 10
y = np.sin(x)            # sine of these values
plt.plot(x, y);          # plot with line
```

2 arrays with same size

Examples:

```
x = np.linspace(0, 10, num=30)
plt.plot(x, np.sin(x), '-ok', label='sin(x)')

plt.xlabel("x")
plt.ylabel("sin(x)")
plt.legend(bbox_to_anchor=(1.3, 1.0))
```

>>> Q2

Explain

>>> Q3

Part 1

First Order
Ordinary Differential Equations (ODEs)

Euler method & Runge-Kunta 4 method (RK4)

1rst order ODEs

Equation: $y'(t) = F[y(t), t]$, with the initial condition (IC): $y(t_0) = y_0$.

1rst order ODEs

Equation: $y'(t) = F[y(t), t],$ with the initial condition (IC): $y(t_0) = y_0.$

Ex: $y'(t) = -t\, y(t),$ with the IC: $y(0) = 1$

Solution?

1rst order ODEs

Equation: $y'(t) = F[y(t), t]$, with the initial condition (IC): $y(t_0) = y_0$.

Ex: $y'(t) = -t\, y(t)$, with the IC: $y(0) = 1$

We already know the solution: $y(t) = \exp(-t^2/2)$

—>> We can use this to test our solvers!

Euler Method

Equation: $y'(t) = F[y(t), t]$, with the initial condition (IC): $y(t_0) = y_0$.

Euler method:

Euler Method

Equation: $y'(t) = F[y(t), t],$ with the initial condition (IC): $y(t_0) = y_0.$

Euler method:

$$\begin{aligned} y(t + dt) &= y(t) + y'(t)\,dt \\ &= \boxed{y(t)} + \boxed{F[y(t), t]}\,dt \end{aligned}$$

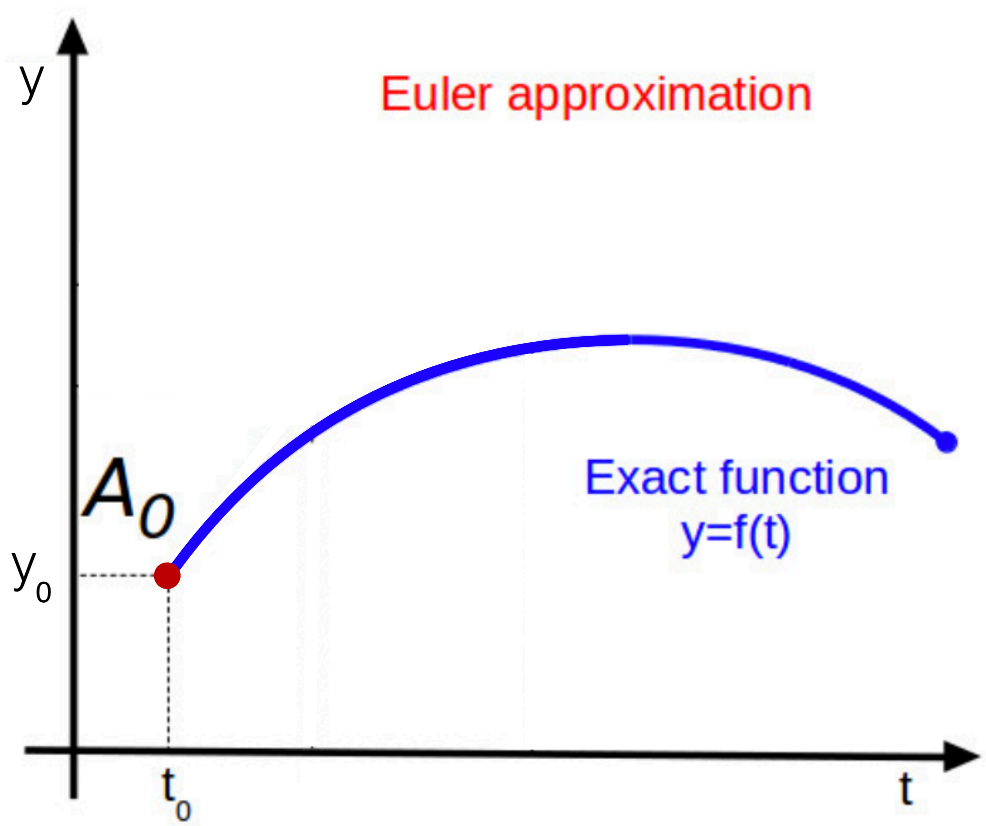
Euler Method

Equation: $y'(t) = F[y(t), t],$ with the initial condition (IC): $y(t_0) = y_0.$

Euler method:

$$\begin{aligned} y(t+dt) &= y(t) + y'(t)\,dt \\ &= \boxed{y(t)} + \boxed{F[y(t), t]}\,dt \end{aligned}$$

Start t_0: $y(t_0) = y_0.$



Euler Method

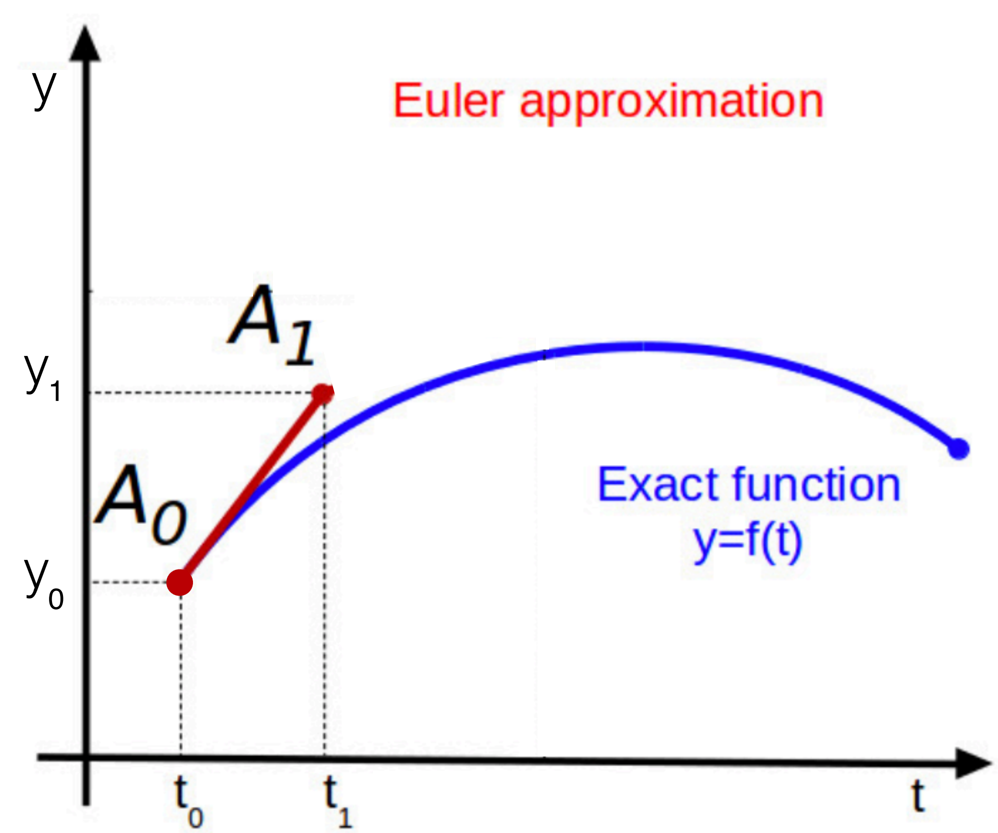
Equation: $y'(t) = F[y(t), t],$ with the initial condition (IC): $y(t_0) = y_0.$

Euler method:

$$
\begin{aligned}
y(t + dt) &= y(t) + y'(t)\, dt \\
&= \boxed{y(t)} + \boxed{F[y(t),\, t]}\, dt
\end{aligned}
$$

Start t_0: $y(t_0) = y_0.$

t_1: $y(t_1) = y(t_0 + dt) = y(t_0) + F[y(t_0), t_0]\, dt$



Euler Method

Equation: $y'(t) = F[y(t), t],$ with the initial condition (IC): $y(t_0) = y_0.$

Euler method:

$$\begin{aligned} y(t+dt) &= y(t) + y'(t)\,dt \\ &= \boxed{y(t)} + \boxed{F[y(t),\, t]}\,dt \end{aligned}$$

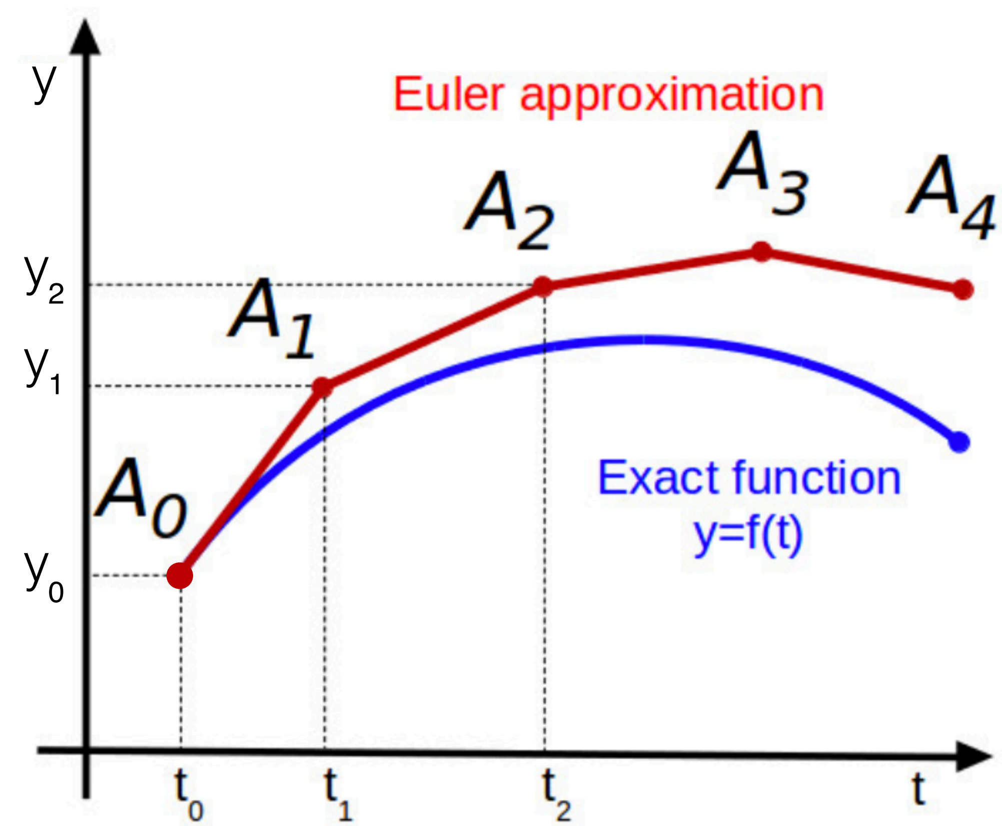
Start t_0: $y(t_0) = y_0.$

t_1: $y(t_1) = y(t_0 + dt) = y(t_0) + F[y(t_0), t_0]\,dt$

t_2: $y(t_2) = y(t_1 + dt) = y(t_1) + F[y(t_1), t_1]\,dt$

...

t_{n+1}: $y(t_{n+1}) = y(t_n + dt) = y(t_n) + F[y(t_n), t_n]\,dt$



Euler Method

>>> Q4-7

Equation: $y'(t) = F[y(t), t],$ with the initial condition (IC): $y(t_0) = y_0.$

Euler method:

$$y(t + dt) = y(t) + y'(t)\,dt$$
$$= \boxed{y(t)} + \boxed{F[y(t), t]}\,dt$$

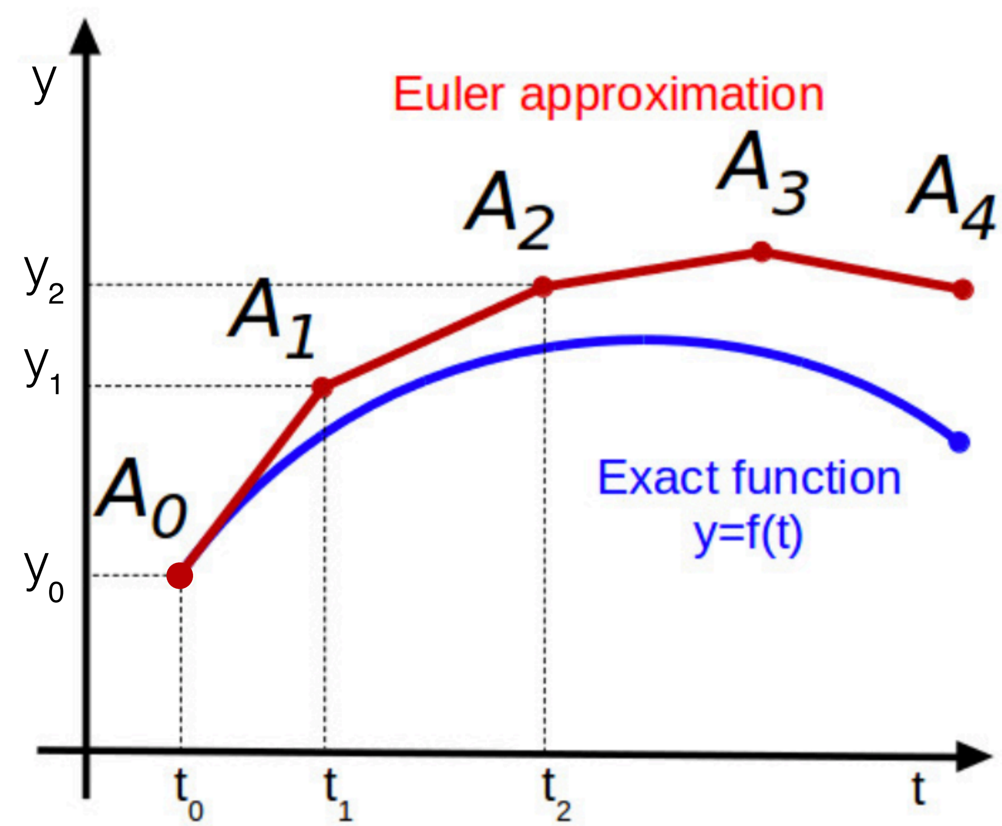
Start t_0: $y(t_0) = y_0.$

t_1: $y(t_1) = y(t_0 + dt) = y(t_0) + F[y(t_0), t_0]\,dt$

t_2: $y(t_2) = y(t_1 + dt) = y(t_1) + F[y(t_1), t_1]\,dt$

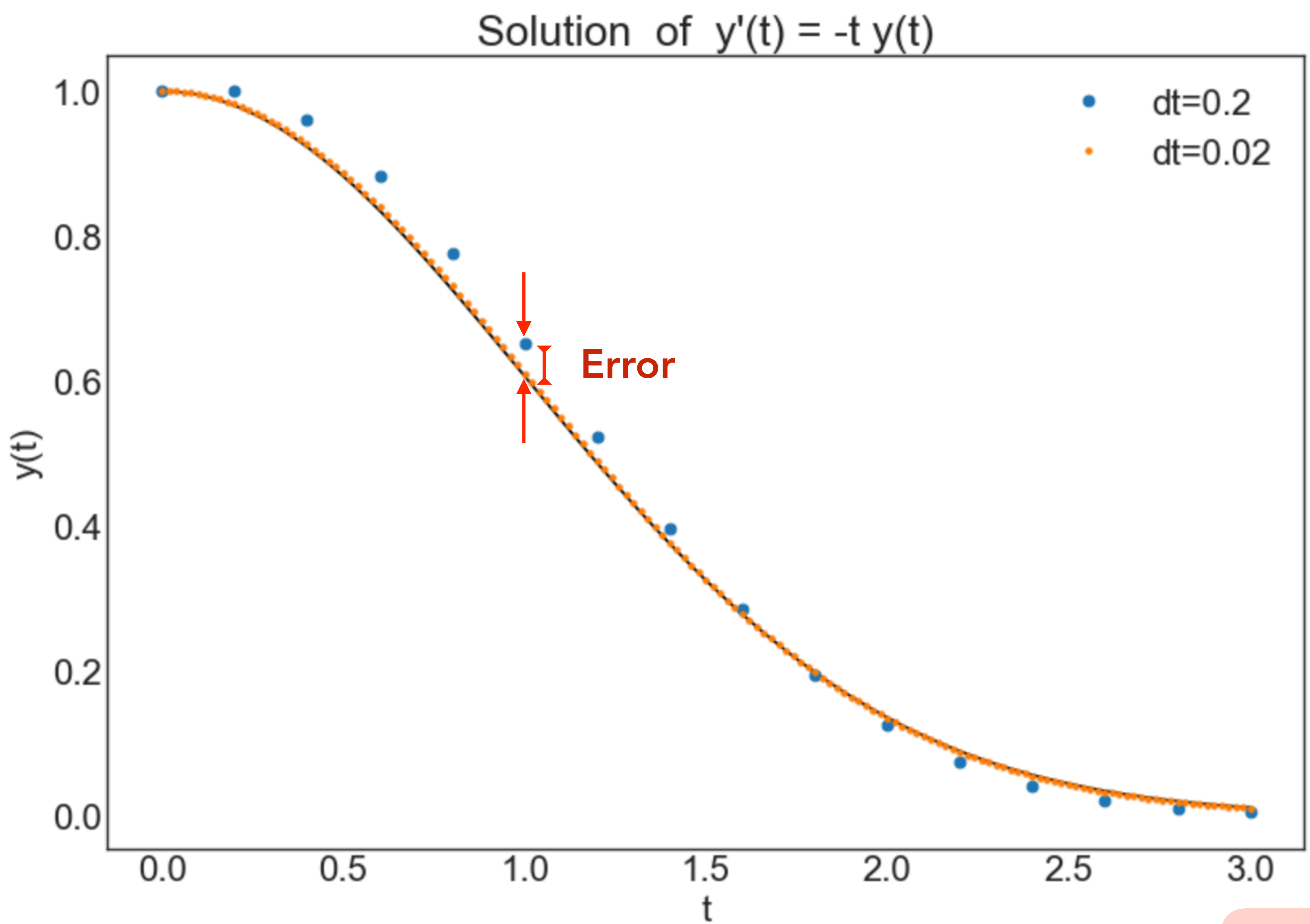
...

t_{n+1}: $y(t_{n+1}) = y(t_n + dt) = y(t_n) + F[y(t_n), t_n]\,dt$



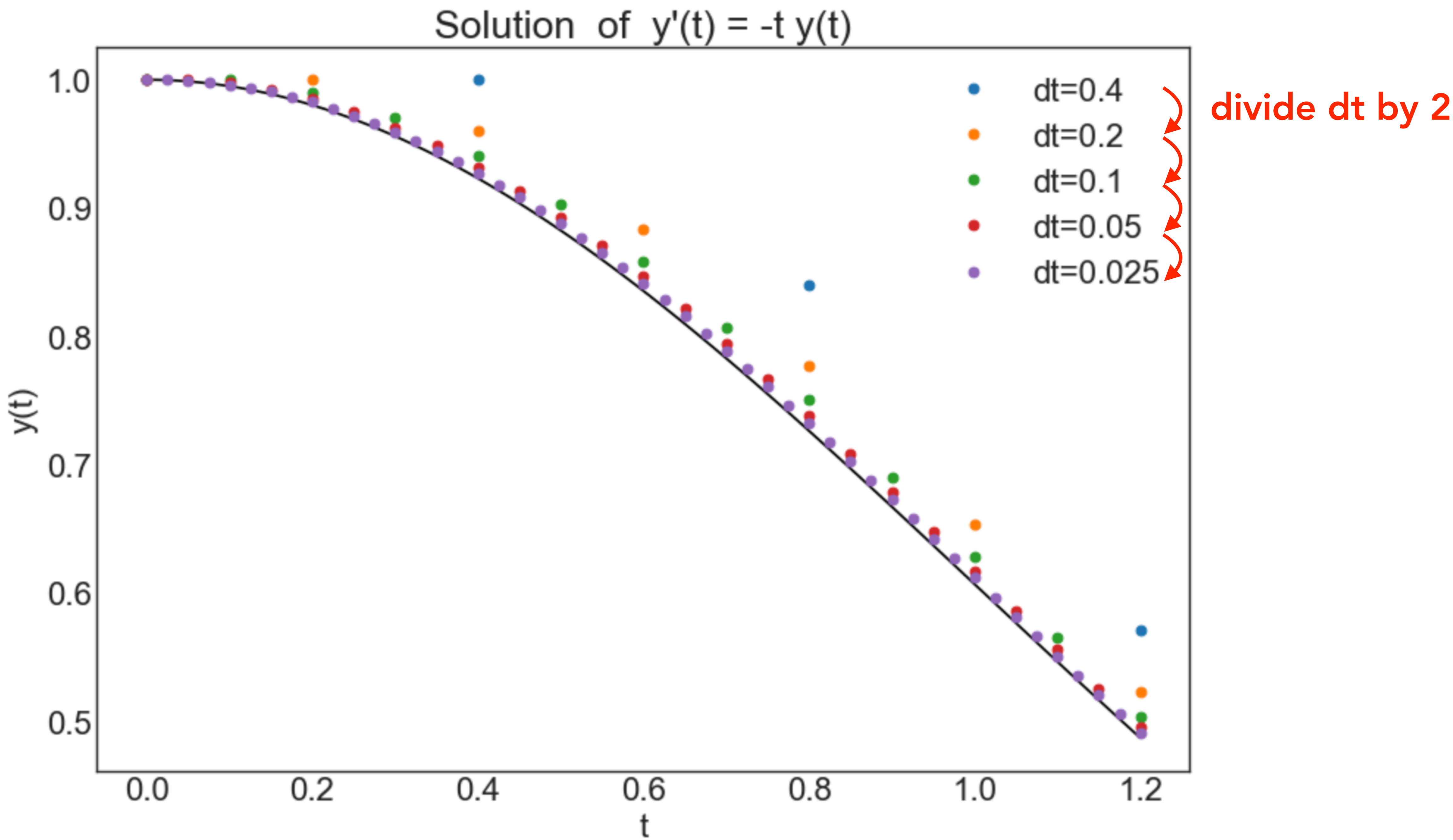
Euler method

Euler Method — Error



>>> Ex2

Euler Method — Error



"First order method": at fixed time, the error is proportional to dt.

Runge-Kutta 4 method

Equation: $y'(t) = F[y(t), t]$, with the initial condition (IC): $y(t_0) = y_0$.

Euler method:

$$\begin{aligned} y(t+dt) &= y(t) + y'(t)\,dt \\ &= \boxed{y(t)} + \boxed{F[y(t),\, t]}\,dt \end{aligned}$$

Runge-Kutta 4 method (RK4):

$$y(t+dt) = \boxed{y(t)} + \frac{dt}{6}\boxed{(k_1 + 2\,k_2 + 2\,k_3 + k_4)}$$

$$\begin{cases} k_1 = \boxed{F}[y(t)\,,\, t] \\ k_2 = \boxed{F}\left[y(t) + \dfrac{dt}{2}\,k_1\,,\, t + \dfrac{dt}{2}\right] \\ k_3 = \boxed{F}\left[y(t) + \dfrac{dt}{2}\,k_2\,,\, t + \dfrac{dt}{2}\right] \\ k_4 = \boxed{F}\left[y(t) + dt\,k_3\,,\, t + dt\right] \end{cases}$$

Runge-Kutta 4 method

Equation: $y'(t) = F[y(t), t],$ with the initial condition (IC): $y(t_0) = y_0.$

Euler method:

$$y(t+dt) = y(t) + y'(t)\,dt$$
$$= \boxed{y(t)} + \boxed{F[y(t), t]}\,dt$$

>>> Q8-10

Runge-Kutta 4 method (RK4):

$$y(t+dt) = \boxed{y(t)} + \frac{dt}{6}\boxed{(k_1 + 2\,k_2 + 2\,k_3 + k_4)}$$

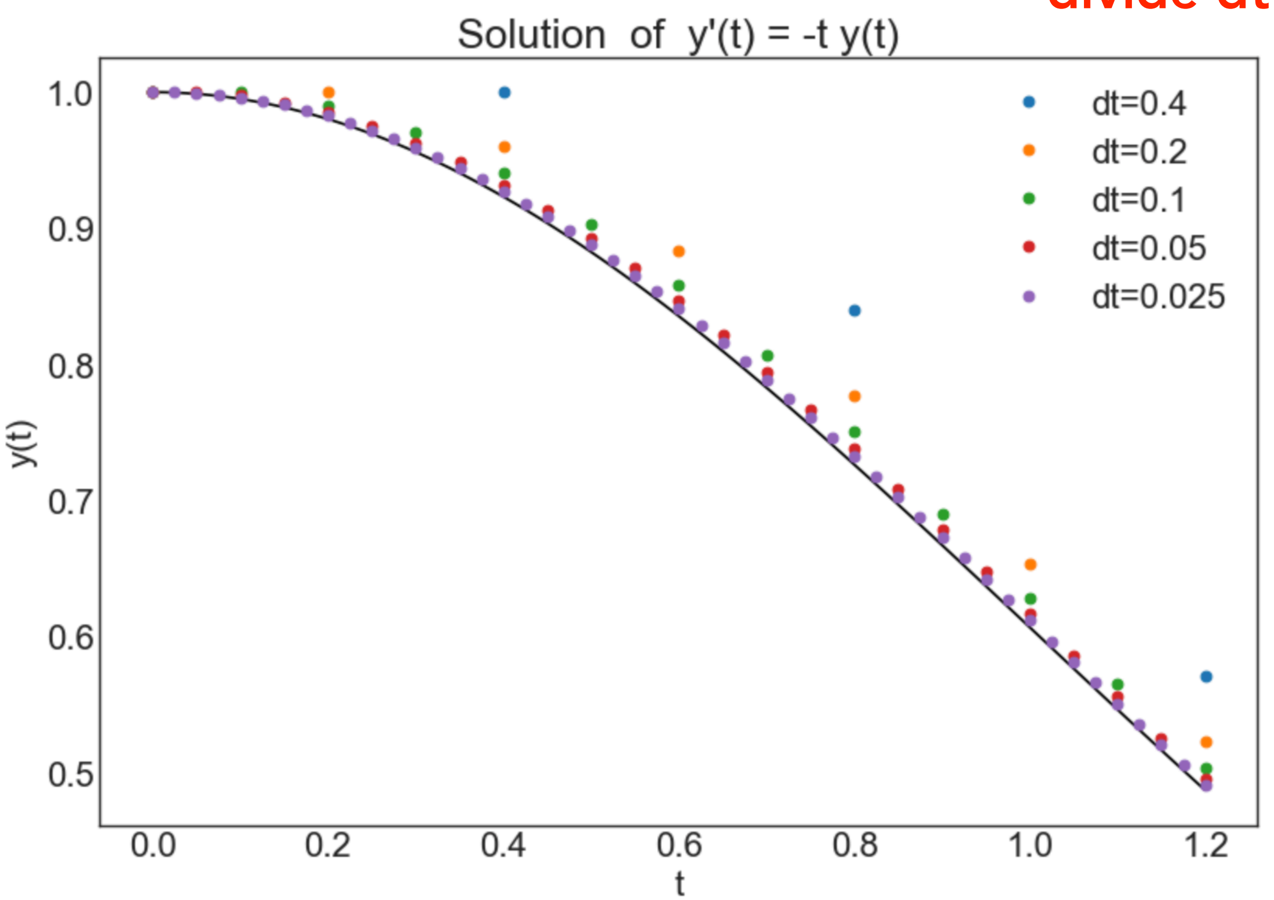
$$\begin{cases} k_1 = \boxed{F}[y(t)\,,\ t] \\ k_2 = \boxed{F}\left[y(t) + \dfrac{dt}{2}\,k_1\,,\ t + \dfrac{dt}{2}\right] \\ k_3 = \boxed{F}\left[y(t) + \dfrac{dt}{2}\,k_2\,,\ t + \dfrac{dt}{2}\right] \\ k_4 = \boxed{F}\left[y(t) + dt\,k_3\,,\ t + dt\right] \end{cases}$$

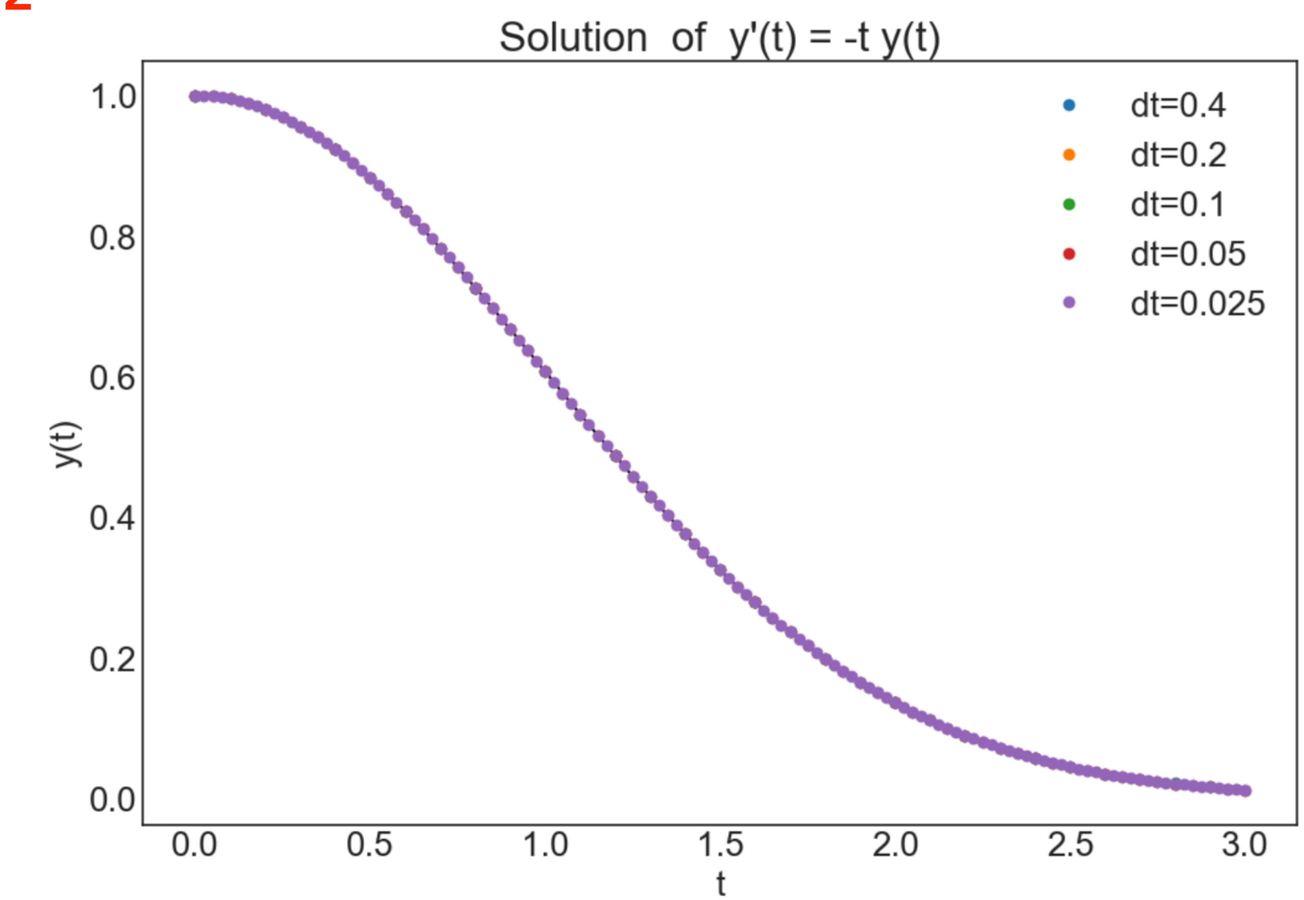
Euler method

RK4 method

Euler VS RK4 — Error

divide dt by 2





"First order method": at fixed time, the error is proportional to **dt**.

"Fourth order method": at fixed time, the total accumulated error grows as $\mathbf{dt^4}$.

In general, you shoould prefer RK4 method over Euler method

Summary

First order Ordinary Differential Equations (ODEs)

Equation: $y'(t) = F[y(t), t],$ with the initial condition (IC): $y(t_0) = y_0.$

Euler method:

$$\begin{aligned} y(t+dt) &= y(t) + y'(t)\,dt \\ &= \boxed{y(t)} + \boxed{F[y(t),\, t]}\,dt \end{aligned}$$

Runge-Kutta 4 method (RK4):

$$y(t+dt) = \boxed{y(t)} + \frac{dt}{6}\boxed{(k_1 + 2\,k_2 + 2\,k_3 + k_4)}$$

$$\begin{cases} k_1 = F\,[y(t)\,,\, t] \\ k_2 = F\left[y(t) + \dfrac{dt}{2}\,k_1\,,\, t + \dfrac{dt}{2}\right] \\ k_3 = F\left[y(t) + \dfrac{dt}{2}\,k_2\,,\, t + \dfrac{dt}{2}\right] \\ k_4 = F\Big[y(t) + dt\,k_3\,,\, t + dt\Big] \end{cases}$$

Part 2

Systems of Coupled First Order ODEs

Euler method & Runge-Kunta 4 method (RK4)

Coupled ODEs

Ex:

$$\begin{cases} y_0'(t) = y_1(t) \\ y_1'(t) = -y_0(t) \end{cases} \quad \text{with the initial condition:} \quad \begin{cases} y_0(0) = 1 \\ y_1(0) = 0 \end{cases}$$

Example

Coupled ODEs

Ex:

$$\begin{cases} y_0'(t) = y_1(t) \\ y_1'(t) = -y_0(t) \end{cases} \quad \text{with the initial condition:} \quad \begin{cases} y_0(0) = 1 \\ y_1(0) = 0 \end{cases}$$

Can be re-written as: $\boldsymbol{y}'(t) = \boldsymbol{F}[\boldsymbol{y}(t), t]$

where $\boldsymbol{y}(t) = \begin{pmatrix} y_0(t) \\ y_1(t) \end{pmatrix}$ and $\boldsymbol{F}[t, \boldsymbol{y}(t)] = \begin{pmatrix} y_1(t) \\ -y_0(t) \end{pmatrix}$

Example

Coupled ODEs

Ex:

$$\begin{cases} y_0'(t) = y_1(t) \\ y_1'(t) = -y_0(t) \end{cases} \quad \text{with the initial condition:} \quad \begin{cases} y_0(0) = 1 \\ y_1(0) = 0 \end{cases}$$

Can be re-written as: $\boldsymbol{y}'(t) = \boldsymbol{F}[\boldsymbol{y}(t), t]$

where

$$\boldsymbol{y}(t) = \begin{pmatrix} y_0(t) \\ y_1(t) \end{pmatrix} \quad \text{and} \quad \boldsymbol{F}[t, \boldsymbol{y}(t)] = \begin{pmatrix} y_1(t) \\ -y_0(t) \end{pmatrix}$$

Again, we already know the solution:

$$\begin{cases} y_0(t) = \cos(t) \\ y_1(t) = -\sin(t) \end{cases}$$

—>> We can use this to test our solvers!

Coupled ODEs

Ex:

$$\begin{cases} y_0'(t) = y_1(t) \\ y_1'(t) = -y_0(t) \end{cases} \quad \text{with the initial condition:} \quad \begin{cases} y_0(0) = 1 \\ y_1(0) = 0 \end{cases}$$

General form:

$$\begin{cases} y_0'(t) & = F_0[y_0, \cdots, y_{n-1}, t] \\ \cdots & \\ y_{n-1}'(t) & = F_{n-1}[y_0, \cdots, y_{n-1}, t] \end{cases}$$

$$\boldsymbol{y}'(t) = \boldsymbol{F}[\boldsymbol{y}(t), t]$$

"Vectorized version"

Summary

Coupled ODEs

Equation: $\boldsymbol{y}'(t) = \boldsymbol{F}[\boldsymbol{y}(t), t]$ with the initial condition (IC): $\boldsymbol{y}(t_0) = \boldsymbol{y}_0$,

Euler method: $\boldsymbol{y}(t+dt) = \boxed{\boldsymbol{y}(t)} + \boxed{\boldsymbol{F}[\boldsymbol{y}(t), t]}\, dt$

Runge-Kutta 4 method (RK4): $\boldsymbol{y}(t+dt) = \boxed{\boldsymbol{y}(t)} + \frac{dt}{6}\boxed{(\boldsymbol{k}_1 + 2\,\boldsymbol{k}_2 + 2\,\boldsymbol{k}_3 + \boldsymbol{k}_4)}$

$$
\begin{cases}
\boldsymbol{k}_1 = \boldsymbol{F}\,[\boldsymbol{y}(t)\,,\, t] \\
\boldsymbol{k}_2 = \boldsymbol{F}\left[\boldsymbol{y}(t) + \frac{dt}{2}\,\boldsymbol{k}_1\,,\, t + \frac{dt}{2}\right] \\
\boldsymbol{k}_3 = \boldsymbol{F}\left[\boldsymbol{y}(t) + \frac{dt}{2}\,\boldsymbol{k}_2\,,\, t + \frac{dt}{2}\right] \\
\boldsymbol{k}_4 = \boldsymbol{F}\left[\boldsymbol{y}(t) + dt\,\boldsymbol{k}_3\,,\, t + dt\right]
\end{cases}
$$

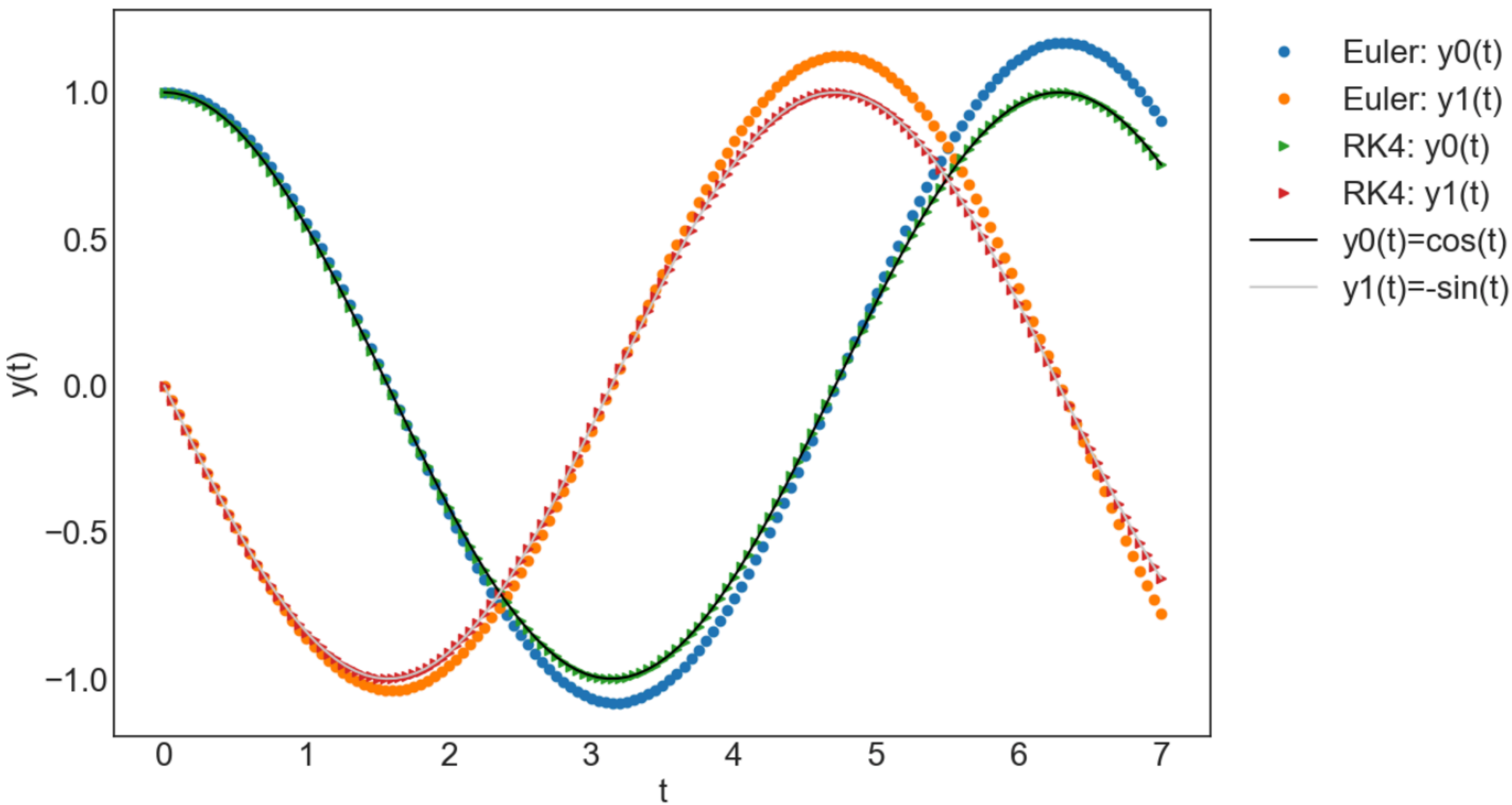
Same method,
but "Vectorized"!

>>> Q11-13

Coupled ODEs

Ex:

$$\begin{cases} y_0'(t) = y_1(t) \\ y_1'(t) = -y_0(t) \end{cases} \quad \text{with the initial condition:} \quad \begin{cases} y_0(0) = 1 \\ y_1(0) = 0 \end{cases}$$



Part 3

Second Order ODEs

Euler method & Runge-Kunta 4 method (RK4)

Example

Second order ODEs

Ex: $y''(t) = -y(t) + g(t)$ with the initial conditions: $\begin{cases} y(t=0) = 1 \\ y'(t=0) = 0 \end{cases}$

Driven harmonic oscillator.

Example

Second order ODEs

Ex: $y''(t) = -y(t) + g(t)$ with the initial conditions: $\begin{cases} y\,(t=0) = 1 \\ y'(t=0) = 0 \end{cases}$

Can be re-written as a system of two coupled ODEs:

$\begin{cases} y'(t) = z(t) \\ z'(t) = -y(t) + g(t) \end{cases}$ with the Initial Conditions: $\begin{cases} y(0) = 1 \\ z(0) = 0 \end{cases}$

Second order ODEs

Ex: $y''(t) = -y(t) + g(t)$ with the initial conditions: $\begin{cases} y\,(t=0) = 1 \\ y'(t=0) = 0 \end{cases}$

Can be re-written as a system of two coupled ODEs:

$\begin{cases} y'(t) = z(t) \\ z'(t) = -y(t) + g(t) \end{cases}$ with the Initial Conditions: $\begin{cases} y(0) = 1 \\ z(0) = 0 \end{cases}$

General form: $y''(t) = F[y(t), y'(t), t]$

with the IC: $y(t_0) = y_0$ and $y'(t_0) = y'_0$

Second order ODEs

Ex: $y''(t) = -y(t) + g(t)$ with the initial conditions: $\begin{cases} y(t=0) = 1 \\ y'(t=0) = 0 \end{cases}$

Can be re-written as a system of two coupled ODEs:

$$\begin{cases} y'(t) = z(t) \\ z'(t) = -y(t) + g(t) \end{cases} \quad \text{with the Initial Conditions:} \quad \begin{cases} y(0) = 1 \\ z(0) = 0 \end{cases}$$

General form: $y''(t) = F[y(t), y'(t), t]$

with the IC: $y(t_0) = y_0$ and $y'(t_0) = y'_0$

Can be re-written as a system of two coupled ODEs:

$$\begin{cases} y'(t) = z(t) \\ z'(t) = F[y(t), z(t), t] \end{cases} \quad \text{with the IC:} \quad \begin{cases} y(t_0) = y_0 \\ z(t_0) = y'_0 \end{cases}$$

>>> Q14

Part 4

Solving with SciPy

Using “odeint()”

Solving ODEs with SciPy

```
from scipy.integrate import odeint
```

General function call:

```
y=odeint(F, y0, t)
```

To compare, Euler and RK4 method:

```
t,y = Solve_ODE_Euler(F, y0, t0, tf, dt)
```

```
t,y = Solve_ODE_RK4(F, y0, t0, tf, dt)
```

Arguments of the function:

- `F` : a function $F(\boldsymbol{y}, t)$ that takes a 1d-array and a scalar and returns a 1d-array;
- `y0` : a 1d-array with the initial values of $\boldsymbol{y}$
- `t` : an array of the values of t at which $\boldsymbol{y}(t)$ will be computed. The first entry of this array must be the initial time at which the initial value `y0` applies;
- `y` : a 1d-array with the values of $\boldsymbol{y}(t)$ computed at the values of t specified in the `t` array.