$f(x, y) = e^{-x}$

interval: $[0, 10]$

101 steps

step size: 0.1

init condition $y(0) = -1$

$y(0) = -1$

$y(1) = -1 + 0.1\left(e^{-(-1)}\right) \approx -0.728$

$y(2) = -0.728 + 0.1\left(e^{-(-0.728)}\right) \approx -0.521$

Analytical Solution

$\frac{dy}{dx} = e^{-x}$

$\int \frac{dy}{dx} = \int e^{-x}$

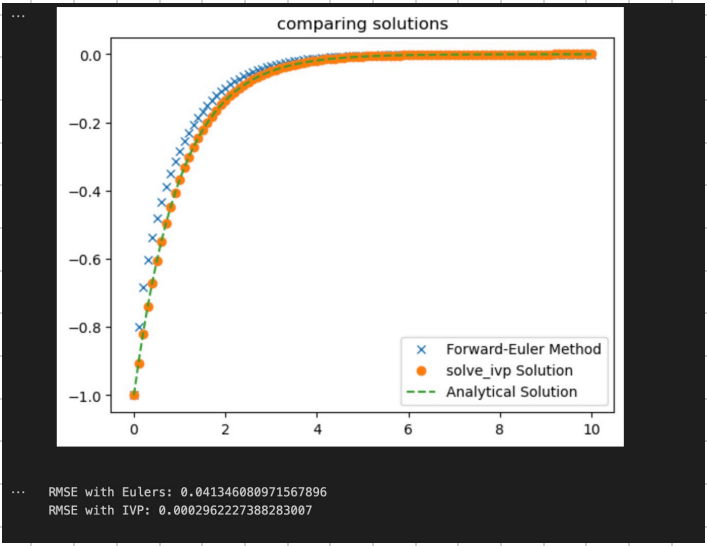
$y = -e^{-x} + c$

$y(0) = -1, \quad -1 = -e^{-0} + c$

$-1 = -1 + c$

$0 = c$

$y = -e^{-x}$



You can see visually, what the RMSE Confirms, that the Forward Eulers method is much less accurate than scipy's solve_ivp

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import solve_ivp
```

[1] ✓ 2.2s Python

```
x0, xf = 0, 10

x_step = 101
step_size = (xf-x0)/(x_step-1)

x = np.linspace(x0, xf, x_step)
y_eul = np.zeros(x_step)

y_eul[0] = -1

for i in range(x_step-1):
    y_eul[i+1] = y_eul[i] + step_size*(-y_eul[i]+np.exp(-i))

plt.plot(x, y_eul, "x")


y0 = [-1]
def fun(x,y):
    return np.exp(-x)

sol = solve_ivp(fun, [x0, xf], y0, t_eval=x)

plt.plot(x, sol.y[0], "o")

y_analytic = -np.exp(-x)
plt.plot(x, y_analytic, "--")

plt.legend(["Forward-Euler Method", "solve_ivp Solution", "Analytical Solution"])
plt.title("comparing solutions")
plt.show()

rmse_eul = np.sqrt(np.mean((y_analytic-y_eul) ** 2))
rmse_ivp = np.sqrt(np.mean((y_analytic-sol.y[0]) ** 2))

print(f"RMSE with Eulers: {rmse_eul}")
print(f"RMSE with IVP: {rmse_ivp}")
```

[6] ✓ 0.0s Python