Section 2.6: The Runge-Kutta Method (RK4)

Prior Knowledge

To approximate an ordinary differential equation of the form

$$\frac{dy}{dx} = f(x, y), \quad y(x_0) = y_0$$

we need to recall the limit definition of the derivative, which is

$$\frac{dy}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

On Friday's class, we use the following algorithm,

Algorithm: The Euler Method (Edwards, Penney, Calvis, 2023)

Given the initial value problem

$$\frac{dy}{dx} = f(x, y), \quad y(x_0) = y_0$$

Euler's method with step size h consists of applying the iterative formula

$$y_{n+1} = h \cdot f(x_n, y_n) + y_n \quad (n \geq 0)$$

to calculate successive approximations $y_1, y_2, y_3, \cdot$ to the [true] values $y(x_1), y(x_2), y(x_3), \cdot$ of the [exact] solution $y = y(x)$ at the points $x_1, x_2, x_3, \ldots$, respectively.

Then on Monday, we increased accuracy using the Improved Euler method (also known as Runge-Kutta Method 2nd order).

Recap: Watch the following video (From 5 mins to 9:30 mins) from Phanimations https://youtu.be/dShtlMl69kY?si=I53pfrTqZN3fbHXR&t=304

Which gave use the new algorithm:

Algorithm: The Improved Euler Method (Edwards, Penney, Calvis, 2023)

Given the initial value problem

$$\frac{dy}{dx} = f(x, y), \quad y(x_0) = y_0$$

the improved Euler method with step size h consists in applying the iterative formulas

$$\begin{aligned} k_1 &= f(x_n, y_n) \\ u_{n+1} &= y_n + h \cdot k_1 \\ k_2 &= f(x_{n+1}, u_{n+1}) \\ y_{n+1} &= y_n + h \cdot \frac{1}{2}(k_1 + k_2) \end{aligned}$$

to compute successive approximations $y_1, y_2, y_3, \ldots$ to the (true) values $y(x_1), y(x_2), y(x_3), \ldots$ of the (exact) solution $y = y(x)$ at the points $x_1, x_2, x_3, \ldots$, respectively.

Can we further improve our accuracy???

Let's watch the following video (From 10 mins to 12 mins): https://youtu.be/dShtIMl69kY?si=IQ0b8NjsjcYtXEb_&t=613

Algorithm: The Runge-Kutta Method 4th Order (RK4) (Edwards, Penney, and Calvis, 2023)

The Runge-Kutta methd involves a weighted average of values of $f(t, y)$ at different points in the interval $t_n \leq t \leq t_{n+1}$. It is given by

$$y_{n+1} = y_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

where

$$\begin{aligned} k_1 &= f(t_n, y_n) \\ k_2 &= f\left(t_n + \frac{1}{2}h, y_n + \frac{1}{2}h \cdot k1\right) \\ k_3 &= f\left(t_n + \frac{1}{2}h, y_n + \frac{1}{2}h \cdot k2\right) \\ k_3 &= f\left(t_n + h, y_n + h \cdot k3\right) \end{aligned}$$

Note: The sum $\dfrac{k_1 + 2k_2 + 2k_3 + k_4}{6}$ can be interpreted as an average slope where

- k_1 is the slope at the left end of the interval,
- k_2 is the slope at the midpoint using the Euler formula to go from t_n to $t_n + \dfrac{h}{2}$,
- k_3 is the second approximation to the slope at the midpoint, and
- k_4 is the slope at $t_n + h$ using the Euler formula and the slope k_3 to go from t_n to t_{n+h}

(The following code was developed and/or tweaked from Dr Jon Shiach's video https://www.youtube.com/watch?v=jrMzo5REU9g)

Problem 1:

Apply the RK4 method to approximate the solution to the initial value problem on the interval $\left[0, \dfrac{1}{2}\right]$, with step size $h = 0.1$. Compare the three-decimal-place values of the three approximations (One found with Euler, the other with Improved Euler, and one with RK4) at $x = \dfrac{1}{2}$ with the value of $y(1/2)$ of the actual solution.

$$y' = y, \quad y(0) = 3, \quad y(x) = 3e^x$$

In [19]:
```
import numpy as np
import math

# RK4 method
def rk4(f, tn, yn, h):
    k1 = f(tn, yn)
    k2 = f(tn + 0.5 * h, yn + 0.5 * h * k1)
```

```
    k3 = f(tn + 0.5 * h, yn + 0.5 * h * k2)
    k4 = f(tn + h, yn + h * k3)
    return yn + h/6 * (k1 + 2*k2 + 2*k3 + k4)

# Improved Euler method
def improved_euler(f, tn, yn, h):
    k1 = f(tn, yn)
    k2 = f(tn + h, yn + h * k1)
    return yn + h * 0.5 * (k1 + k2)

#Solver function
def solveIVP(f, tspan, y0, h, solver):
    t = np.arange(tspan[0], tspan[1] + h, h)
    y = np.zeros(len(t))
    y[0] = y0

    for n in range(len(t) - 1):
        y[n+1] = solver(f, t[n], y[n], h)

    return t, y

# Euler method
def euler(f, tn, yn, h):
    return yn + h * f(tn,yn)

# Define IVP
def f(t,y):
    return y # becuase y'=f(t,y)=y

# Problem Specific
tspan = [0, 0.5]
x0 = 0
y0 = 3
x = 0.5
h = 0.1

# Exact solution
def exact(x):
    return 3 * math.exp(x)

# Collect Euler results
t1, y1 = solveIVP(f, tspan, y0, h, euler)

# Collect Improved Euler results
t2, y2 = solveIVP(f, tspan, y0, h, improved_euler)

# Collect RK4 results
t3, y3 = solveIVP(f, tspan, y0, h, rk4)

# Print table
print(" Steps (h) | Euler y(0.5) | Improved Euler y(0.5) | RK4 y(0.5)    |   Exact y(0.5) ")
print("-------------------------------------------------------------------------------------")
for n in range(len(t1)):
    print(f" {t1[n]:6.2f}    |    {y1[n]:9.3f} |   {y2[n]:9.3f}           | {y3[n]:9.3f}     |    {ex
```

```
 Steps (h) | Euler y(0.5) | Improved Euler y(0.5) | RK4 y(0.5)    |   Exact y(0.5)
-------------------------------------------------------------------------------------
   0.00    |        3.000 |       3.000           |     3.000     |        4.946
   0.10    |        3.300 |       3.315           |     3.316     |        4.946
   0.20    |        3.630 |       3.663           |     3.664     |        4.946
   0.30    |        3.993 |       4.048           |     4.050     |        4.946
   0.40    |        4.392 |       4.473           |     4.475     |        4.946
   0.50    |        4.832 |       4.942           |     4.946     |        4.946
```

As you could see the RK4's Method is better at approximating our function!!!

Problem 2:

$y' = e^{-y}$, y(0)=0, $y = \ln(x+1)$ on interval [0,0.5] with h=0.25 and x=0.25

In [27]:
```
# Define IVP
def f(t,y):
    return math.exp(-1*y) # becuase y'=f(t,y)=e^{-y}

# Problem Specific
tspan = [0, 0.5]
x0 = 0
y0 = 0
h = 0.25

# Collect Euler results
t1, y1 = solveIVP(f, tspan, y0, h, euler)

# Collect Improved Euler results
t2, y2 = solveIVP(f, tspan, y0, h, improved_euler)

# Collect RK4 results
t3, y3 = solveIVP(f, tspan, y0, h, rk4)

# Exact solution
def exact(x):
   return math.log(x+1) # math.log = ln

# Print table
print(" Steps (h) | Euler y(0.25) | Improved Euler y(0.25) | RK4 y(0.25)    |   Exact y(0.25) ")
print("-------------------------------------------------------------------------------------------")
for n in range(len(t1)):
    print(f" {t1[n]:6.2f}    |     {y1[n]:9.3f} |   {y2[n]:9.3f}            | {y3[n]:9.3f}      |
```

```
 Steps (h) | Euler y(0.25) | Improved Euler y(0.25) | RK4 y(0.25)    |   Exact y(0.25)
-------------------------------------------------------------------------------------------
   0.00    |         0.000 |       0.000            |     0.000      |        0.405
   0.25    |         0.250 |       0.222            |     0.223      |        0.405
   0.50    |         0.445 |       0.404            |     0.405      |        0.405
```

Sources:

1. Differential Equations and Boundary Value Problems (2023) by Edwards, Penney, Calvis
2. Dr. Jon Shiach's Euler Method (Python) video found at https://www.youtube.com/watch?v=N7Oh0mk4YGc&t=378s
3. Phanimations video found at https://www.youtube.com/watch?v=N7Oh0mk4YGc&t=378s
4. Elementary Differential Equations (1997) by Boyce and DiPrima

In [ ]: