Question 1

Given

Analytical solution:

Given Initial condition

Then,

**Euler's Modified Method (EMM)**

Algorithm

Predictor

Corrector 1

Corrector 2

Where

Algorithm Steps

1. Compute initial slope at beginning of interval

Next value

1. First prediction

1. Correction 1

Slope @ predicted point

Average slope for next value

1. Correction 2

1. Final value

Parameters

* Step sizes: and
* Function:

import numpy as np

import pandas as pd

import argparse

def get\_equations(key: str):

    if key.lower() == "assignment1":

        def f(x, y):

            return -3 \* x\*\*2 \* y\*\*2

        def y\_exact(x):

            return 2 / (2 \* x\*\*3 + 1)

        return f, y\_exact

    else:

        raise ValueError(f"Unknown equation keyword: {key}")

def run\_modified\_euler(f, y\_exact, h: float, y0: float = 2.0, x\_end: float = 1.0) -> pd.DataFrame:

    x\_vals = np.arange(0, x\_end + h, h)

    y\_vals = [y0]

    errors = [0]

    f\_xy\_vals = [f(0, y0)]

    y\_p\_vals = [np.nan]

    f\_new\_vals = [np.nan]

    y\_new\_vals = [y0]

    for i in range(1, len(x\_vals)):

        x\_i = x\_vals[i - 1]

        y\_i = y\_vals[-1]

        f\_xy = f(x\_i, y\_i)

        y\_p = y\_i + h \* f\_xy

        y\_c1 = y\_i + (h / 2) \* (f\_xy + f(x\_i + h, y\_p))

        y\_next = y\_i + (h / 2) \* (f\_xy + f(x\_i + h, y\_c1))

        f\_new = f(x\_i + h, y\_c1)

        true\_val = y\_exact(x\_vals[i])

        error = abs(true\_val - y\_next)

        y\_vals.append(y\_next)

        errors.append(error)

        f\_xy\_vals.append(f\_xy)

        y\_p\_vals.append(y\_p)

        f\_new\_vals.append(f\_new)

        y\_new\_vals.append(y\_next)

    return pd.DataFrame({

        'x': x\_vals,

        'y': y\_vals,

        'Exact y': y\_exact(x\_vals),

        'Error': errors,

        'f(x,y)': f\_xy\_vals,

        'yp': y\_p\_vals,

        'fnew': f\_new\_vals,

        'ynew': y\_new\_vals

    })

if \_\_name\_\_ == "\_\_main\_\_":

    parser = argparse.ArgumentParser(description="Modified Euler Method with 2 corrections")

    parser.add\_argument('--equation', type=str, required=True,

                        help="Choose a predefined equation, e.g., 'assignment1'")

    parser.add\_argument('--step-size', type=float, required=True,

                        help="Step size (e.g. 0.1)")

    parser.add\_argument('--y0', type=float, default=2.0, help="Initial y value (default: 2.0)")

    parser.add\_argument('--x-end', type=float, default=1.0, help="Final x value (default: 1.0)")

    args = parser.parse\_args()

    f, y\_exact = get\_equations(args.equation)

    df = run\_modified\_euler(f, y\_exact, args.step\_size, args.y0, args.x\_end)

    print(df.to\_string(index=False))

python EMM.py --equation assignment1 --step-size 0.1

x y Exact y Error f(x,y) yp fnew ynew

0.0 2.000000 2.000000 0.000000 0.000000 NaN NaN 2.000000

0.1 1.994036 1.996008 0.001972 -0.000000 2.000000 -0.119281 1.994036

0.2 1.964916 1.968504 0.003588 -0.119285 1.982107 -0.463111 1.964916

0.3 1.893422 1.897533 0.004111 -0.463307 1.918585 -0.966568 1.893422

0.4 1.770042 1.773050 0.003007 -0.967963 1.796626 -1.499641 1.770042

0.5 1.599271 1.600000 0.000729 -1.503864 1.619656 -1.911551 1.599271

0.6 1.398064 1.396648 0.001416 -1.918252 1.407446 -2.105897 1.398064

0.7 1.188615 1.186240 0.002375 -2.110949 1.186969 -2.078039 1.188615

0.8 0.990227 0.988142 0.002085 -2.076823 0.980932 -1.890925 0.990227

0.9 0.814805 0.813670 0.001136 -1.882656 0.801962 -1.625783 0.814805

1.0 0.666788 0.666667 0.000122 -1.613295 0.653476 -1.347045 0.666788

x y Exact y Error f(x,y) yp fnew ynew

0.0 2.000000 2.000000 0.000000 0.000000 NaN NaN 2.000000

0.2 1.954276 1.968504 0.014228 -0.000000 2.000000 -0.457236 1.954276

0.4 1.762801 1.773050 0.010249 -0.458304 1.862616 -1.456453 1.762801

0.6 1.407367 1.396648 0.010718 -1.491584 1.464484 -2.062758 1.407367

0.8 0.997893 0.988142 0.009750 -2.139135 0.979540 -1.955605 0.997893

1.0 0.662607 0.666667 0.004059 -1.911916 0.615509 -1.440936 0.662607

python EMM.py --equation assignment1 --step-size 0.1

Question 2

Given

Derivatives at

Compute at

Question 3

Given

**Convert to First-Order System**

Then,

And,

**2nd Order Runge-Kutta Method (General Form)**

Given

Then,

**Applying Algorithm**

Initial values

**Compute k1 values**

**Intermediate step values for k2**

**Compute k2 values**

**Final values using weighted combination, at t=0.2**