Nguyễn Minh Đức - ITITIU21045 _ Lab9 Ngày .... Tháng ..... Năm........

Q1.

- First-order central difference approx. of ~~$O(h^4)$~~ $O(h^4)$:

$$f'(x_i) = \frac{-f(x_{i+2}) + 8f(x_{i+1}) - 8f(x_{i-1}) + f(x_{i-2})}{12h} \quad (*)$$

a) $y = x^3 + 3x - 15$ at $x = 0$, $h = 0.25$

- $x_{i+2} = 0.5$; $f(x_{i+2}) = -13.375$
- $x_{i+1} = 0.25$; $f(x_{i+1}) = -14.234375$
- $x_{i-1} = -0.25$; $f(x_{i-1}) = -15.765625$
- $x_{i-2} = -0.5$; $f(x_{i-2}) = -16.625$

Using equation $(*)$, we have:

$f'(0) = 3 = f'(0)_{analytic}$

b) $y = x^2 \cos x$ at $x = 0.5$, $h = 0.1$

- $x_{i+2} = 0.7$; $f(x_{i+2}) = 0.49$
- $x_{i+1} = 0.6$; $f(x_{i+1}) = 0.36$
- $x_{i-1} = 0.4$; $f(x_{i-1}) = 0.16$
- $x_{i-2} = 0.3$; $f(x_{i-2}) = 0.09$

Using equation $(*)$, we have:

$f'(0.5) = 1 > f'(0.5)_{analytic}$

c) $y = \tan\left(\frac{x}{3}\right)$ at $x = 2$, $h = 0.5$

- $x_{i+2} = 3$; $f(x_{i+2}) = 0.0175$
- $x_{i+1} = 2.5$; $f(x_{i+1}) = 0.0145$
- $x_{i-1} = 1.5$; $f(x_{i-1}) = 0.0087$
- $x_{i-2} = 1$; $f(x_{i-2}) = 0.0058$

Using equation $(*)$, we have:

$f'(2) = 0.005783 < f'(2)_{analytic}$

d) $y = \frac{\sin(0.5\sqrt{x})}{x}$ at $x = 1$, $h = 0.2$

- $x_{i+2} = 1.4$; $f(x_{i+2}) = 0.0074$
- $x_{i+1} = 1.2$; $f(x_{i+1}) = 0.008$
- $x_{i-1} = 0.8$; $f(x_{i-1}) = 0.0098$
- $x_{i-2} = 0.6$; $f(x_{i-2}) = 0.0113$

$\rightarrow$ ~~$f'(0.5$~~ $f'(1) = -4.375 \times 10^{-3} < \boxed{f'(1)_{analytic}}$

e) $y = e^x + x$ at $x = 3$, $h = 0.2$

- $x_{i+2} = 3.4$; $f(x_{i+2}) = 33.3641$
- $x_{i+1} = 3.2$; $f(x_{i+1}) = 27.7325$
- $x_{i-1} = 2.8$; $f(x_{i-1}) = 19.2446$
- $x_{i-2} = 2.6$; $f(x_{i-2}) = 16.0637$

$\rightarrow f'(3) = 21.0845 < f'(3)_{analytic}$

Q2.:

Q3.:

Q2:

t, s	0	25	50	75	100	125
y, km	0	32	58	78	92	100

~~• At t = 0~~

~~$v(0) = y'(0) = \frac{y(25) - y(0)}{25-0} = \frac{32-0}{25-0}$~~

Velocity is differential of distance of time

Acceleration is differential of velocity of time.

Using forward, center and backward difference to calculate v, a

forward for the first point, backward for the last point and center for the rest, we ~~have~~ calculate table of velocity and acceleration.

• Forward: $f'(x_i) = \frac{f(x_{i+1}) - f(x_i)}{h}$

• Backward: $f'(x_i) = \frac{f(x_i) - f(x_{i-1})}{h}$

• Center: $f'(x_i) = \frac{f(x_{i+1}) - f(x_{i-1})}{2h}$

v, km/s	1.28	1.16	0.92	0.68	0.44	0.32
a, km/s²	-4.8×10^{-3}	-7.2×10^{-3}	-9.6×10^{-3}	-9.6×10^{-3}	-7.2×10^{-3}	-4.8×10^{-3}

$< \boxed{f'(0)_{analytic}}$

Q3: (a) $\frac{dy}{dx} = (1+2t)\sqrt{x}$

$\Rightarrow dy = (1+2t)\sqrt{x}\, dx$

$\Rightarrow \int dy = \int (1+2t)\sqrt{x}\, dx$

$\Rightarrow y = (1+2t)\frac{2}{3}x^{\frac{3}{2}} + C$

Substitute $y(0) = 1$ to the equation, we calculate C:

$\Rightarrow C = 1$

$\Rightarrow y = (1+2t)\frac{2}{3}x^{\frac{3}{2}} + 1$

x	y
0	1
0.25	$t/6 + 13/12$
0.5	$0.47t + 1.24$
0.75	$0.87t + 1.43$
1	$4/3\,t + 5/3$

Ngày .... Tháng ..... Năm........

b) Euler's method:

- $y(0.25) = y(0) + f(0,1)h$
- $f(0,1) = (1+2t)\sqrt{0} = 0$

$\Rightarrow y(0.25) = 1 + 0 \times 0.25 = 1$

- $y(0.5) = y(0.25) + f(0.25, 1 \cancel{.25}) \times 0.25$
- $f(0.25, 1) = (1+2t)\sqrt{0.25} = 0.5 + t$

$\Rightarrow y(0.5) = 1 + (t + 0.5) \times 0.25 = 0.25t + 1.125$

x	y	dy/dx
0	1	0
0.25	1	$t + 0.5$
0.5	$0.25t + 1.125$	$\cancel{\sqrt{2}t + 1}$ $1.41t + 0.7$
0.75	$0.6t + 1.3$	$1.73t + 0.87$
1	$1.03t + 1.5$	$2t + 1$

c) $x_{i+1} = x_i + h$ | $y^*_{i+1} = y_i + h[f(x_i, y_i)]$ (Predictor) | $y_{i+1} = y_i + \frac{h}{2}[f(x_i, y_i) + f(x_{i+1}, y^*_{i+1})]$ (Corrector)

$x_{i+1} = x_i + h$	y^*_{i+1} (Predictor)	y_{i+1} (Corrector)
0	$1 + 0.25 \times 0 = 1$	$1 + \frac{0.25}{2}(0 + t + 0.5) = 0.125t + 1.0625$
0.25	~~0.125t + 1~~ 0.375t + 1.1875	0.43t + 1.21
0.5	0.78t + 1.39	0.82t + 1.41
0.75	0.36t + 1.63	1.29t + 1.64
1	~~1.79t + 1.89 1.79t + 1.89~~	

d) Ralston's method:

$$y_{i+1} = y_i + (a_1 k_1 + a_2 k_2)h \qquad a_1 = \frac{1}{3};\ a_2 = \frac{2}{3}$$

$$\Rightarrow y_{i+1} = y_i + \left(\frac{1}{3}k_1 + \frac{2}{3}k_2\right)h \qquad k_1 = f(x_i, y_i)$$

$$k_2 = f\left(x_i + \frac{3}{4}h,\ y_i + \frac{3}{4}hk_1\right)$$

x_i	k_1	k_2	y_{i+1}
0	0	0.87t + 0.43	0.145t + 1.072
0.25	t + 0.5	1.32t + 0.66	0.45t + 1.22
0.5	1.4t + 0.7	1.66t + 0.83	0.84t + 1.42
0.75	1.73t + 0.87	1.94t + 0.97	1.3t + 1.65
1	1 + 2t	2.18t + 1.09	1.83t + 1.92

e) $f(x, y) = (1+2t)\sqrt{x}$, $y(0) = 1$, $h = 0.25$

$f(x_0, y_0) = f(0, 1) = (1 + 2t)\sqrt{0} = 0$

$k_1 = h \cdot f(x_0, y_0) = 0.25 \times 0 = 0$

$$k_2 = h\,f\left[x_0 + \frac{h}{2}, y_0 + \frac{k_1}{2}\right] = 0.25 \times f\left[0 + \frac{0.25}{2}, 1 + \frac{0}{2}\right] = 0.18t + 0.09$$

$k_3 = hf\left[x_0 + \frac{h}{2}, y_0 + \frac{k_2}{2}\right] = 0.25 \times f\left[0 + \frac{0.25}{2}, y_0 + \frac{k_2}{2}\right] = 0.18t + 0.09$

$k_4 = hf\left[x_0 + h, y_0 + k_3\right] = 0.25 \times f\left[0 + 0.25, y_0 + k_3\right] = 0.25t + 0.125$

Finally,

$K = \frac{1}{6}\left[K_1 + 2K_2 + 2K_3 + K_4\right]$

$= \frac{1}{6}\left[0 + 2\times(0.18t + 0.09) + 2\times(0.18t + 0.09) + 0.25t + 0.125\right]$

$= 0.16t + 0.08$

Therefore $y = y_0 + k = 1 + 0.16t + 0.08 = 0.16t + 1.08$

$y(0.25) = 0.16t + 1.08$

Calculate other points we have table:

x_i	$f(x_i, y_i)$	k_1	k_2	k_3	k_4	k	y
0	0	0	0.18t + 0.09	0.18t + 0.09	0.25t + 0.125	0.16t + 0.08	0.16t + 1.08
0.25	t + 0.25	0.25t + 0.0625	0.3t + 0.15	0.3t + 0.15	0.35t + 0.18	0.3t + 0.14	0.46t + 1.22
0.5	1.41t + 0.7	0.35t + 0.18	0.4t + 0.2	0.4t + 0.2	0.43t + 0.22	0.4t + 0.2	0.86t + 1.42
0.75	1.73t + 0.87	0.4t + 0.2	0.47t + 0.23	0.47t + 0.23	0.5t + 0.25	0.46t + 0.23	1.32t + 1.65
1	1 + 2t	0.5t + 0.25	0.53t + 0.27	0.53t + 0.27	0.56t + 0.28	0.53t + 0.27	1.85t + 1.92

```
#Codes created by Nguyen Minh Duc - ITITIU21045 - TMCLab9
#(a)

def function_y(x):
    return x**3 + 3*x - 15

def derivative_y(x):
    return 3*x**2 + 3

def central_difference_approximation(f, x, h):
    numerator = -f(x + 2*h) + 8*f(x + h) - 8*f(x - h) + f(x - 2*h)
    return numerator / (12 * h)

# Given values
x_0 = 0
h = 0.25

# Calculate the central difference approximation
approximation = central_difference_approximation(function_y, x_0, h)

# Calculate the exact derivative at x = 0
exact_derivative = derivative_y(x_0)

# Output the results
print("Central Difference Approximation:", approximation)
print("Exact Derivative:", exact_derivative)
```

```
Central Difference Approximation: 3.0
Exact Derivative: 3
```

```
# (b)

import math

def function_y(x):
    return x**2 * math.cos(x)

def derivative_y(x):
    return 2*x*math.cos(x) - x**2*math.sin(x)

def central_difference_approximation(f, x, h):
    numerator = -f(x + 2*h) + 8*f(x + h) - 8*f(x - h) + f(x - 2*h)
    return numerator / (12 * h)

# Given values
x_0 = 0.5
h = 0.1

# Calculate the central difference approximation
approximation = central_difference_approximation(function_y, x_0, h)

# Calculate the exact derivative at x = 0.5
exact_derivative = derivative_y(x_0)

# Output the results
print("Central Difference Approximation:", approximation)
print("Exact Derivative:", exact_derivative)
```

```
Central Difference Approximation: 0.7576800923900712
Exact Derivative: 0.757726177239322
```

```
# (c)

import math

def function_y(x):
    return math.tan(x/3)

def derivative_y(x):
    return (1/3) * (1 / math.cos(x/3)**2)

def central_difference_approximation(f, x, h):
    numerator = -f(x + 2*h) + 8*f(x + h) - 8*f(x - h) + f(x - 2*h)
    return numerator / (12 * h)

# Given values
x_0 = 2
h = 0.5

# Calculate the central difference approximation
approximation = central_difference_approximation(function_y, x_0, h)

# Calculate the exact derivative at x = 2
exact_derivative = derivative_y(x_0)

# Output the results
print("Central Difference Approximation:", approximation)
print("Exact Derivative:", exact_derivative)
```

```
Central Difference Approximation: 0.5374421427372936
Exact Derivative: 0.5397072442380613
```

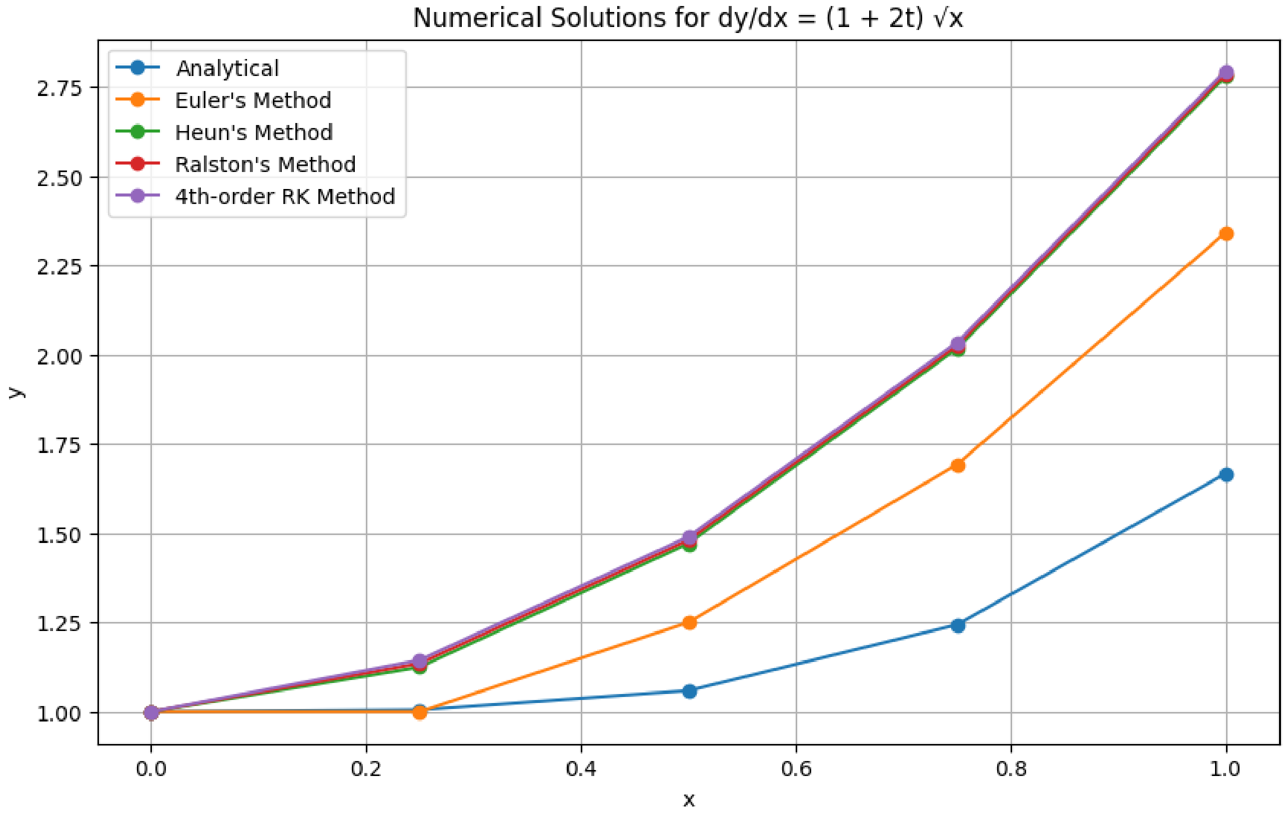
```
# (d)

import math

def function_y(x):
    return math.sin(0.5 * math.sqrt(x)) / x

def derivative_y(x):
    return (math.cos(0.5 * math.sqrt(x)) / (2 * math.sqrt(x))) - (math.sin(0.5 * math.sqr

# (e)

import math

def function_y(x):
    return math.exp(x) + x

def derivative_y(x):
    return math.exp(x) + 1
```

```
# Codes created by Nguyen Minh Duc - ITITIU21045 - TMCLab9
import numpy as np

# Given data
t = np.array([0, 25, 50, 75, 100, 125])
y = np.array([0, 32, 58, 78, 92, 100])

# Function to calculate velocity using central difference
def calculate_velocity(t, y):
    h = t[1] - t[0]
    velocity = (y[2:] - y[:-2]) / (2 * h)
    return np.concatenate(([0], velocity, [0]))

# Function to calculate acceleration using central difference
def calculate_acceleration(t, y):
    h = t[1] - t[0]
    acceleration = (y[2:] - 2 * y[1:-1] + y[:-2]) / (h**2)
    return np.concatenate(([0], acceleration, [0]))

# Calculate velocity and acceleration
velocity = calculate_velocity(t, y)
acceleration = calculate_acceleration(t, y)

# Output the results
for i in range(len(t)):
    print(f"Time: {t[i]}s, Velocity: {velocity[i]} km/s, Acceleration: {accelerat
```

```
Time: 0s, Velocity: 0.0 km/s, Acceleration: 0.0 km/s²
Time: 25s, Velocity: 1.16 km/s, Acceleration: -0.0096 km/s²
Time: 50s, Velocity: 0.92 km/s, Acceleration: -0.0096 km/s²
Time: 75s, Velocity: 0.68 km/s, Acceleration: -0.0096 km/s²
Time: 100s, Velocity: 0.44 km/s, Acceleration: -0.0096 km/s²
Time: 125s, Velocity: 0.0 km/s, Acceleration: 0.0 km/s²
```

```
# Codes created by Nguyen Minh Duc - ITITIU21045 - TMCLab9
import numpy as np
import matplotlib.pyplot as plt

# Given ODE
def f(t, y, x):
    return (1 + 2*t) * np.sqrt(x)

# (a) Analytical solution (if possible)
def analytical_solution(t, x):
    return (2/3) * (x**(3/2)) * (t**2) + 1

# (b) Euler's method
def euler_method(t_values, h):
    y_values = [1]  # Initial condition
    for i in range(1, len(t_values)):
        x_i = t_values[i-1]
        t_i = t_values[i]
        y_i = y_values[-1]
        y_next = y_i + h * f(t_i, y_i, x_i)
        y_values.append(y_next)
    return y_values

# (c) Heun's method without iteration
def heun_method(t_values, h):
    y_values = [1]  # Initial condition
    for i in range(1, len(t_values)):
        x_i = t_values[i-1]
        t_i = t_values[i]
        y_i = y_values[-1]
        k1 = f(t_i, y_i, x_i)
        k2 = f(t_i + h, y_i + h * k1, x_i + h)
        y_next = y_i + 0.5 * h * (k1 + k2)
        y_values.append(y_next)
    return y_values

# (d) Ralston's method
def ralston_method(t_values, h):
    y_values = [1]  # Initial condition
    for i in range(1, len(t_values)):
        x_i = t_values[i-1]
        t_i = t_values[i]
        y_i = y_values[-1]
        k1 = f(t_i, y_i, x_i)
        k2 = f(t_i + 0.75 * h, y_i + 0.75 * h * k1, x_i + 0.75 * h)
        y_next = y_i + (1/3) * h * (k1 + 2*k2)
        y_values.append(y_next)
    return y_values

# (e) Fourth-order Runge-Kutta (RK) method
def runge_kutta_method(t_values, h):
    y_values = [1]  # Initial condition
    for i in range(1, len(t_values)):
        x_i = t_values[i-1]
```

```
        t_i = t_values[i]
        y_i = y_values[-1]
        k1 = f(t_i, y_i, x_i)
        k2 = f(t_i + 0.5 * h, y_i + 0.5 * h * k1, x_i + 0.5 * h)
        k3 = f(t_i + 0.5 * h, y_i + 0.5 * h * k2, x_i + 0.5 * h)
        k4 = f(t_i + h, y_i + h * k3, x_i + h)
        y_next = y_i + (h/6) * (k1 + 2*k2 + 2*k3 + k4)
        y_values.append(y_next)
    return y_values

# Interval and step size
t_values = np.arange(0, 1.25, 0.25)

# Analytical solution
analytical_values = analytical_solution(t_values, t_values)

# Numerical solutions
euler_values = euler_method(t_values, 0.25)
heun_values = heun_method(t_values, 0.25)
ralston_values = ralston_method(t_values, 0.25)
rk_values = runge_kutta_method(t_values, 0.25)

# Plotting
plt.figure(figsize=(10, 6))
plt.plot(t_values, analytical_values, label='Analytical', marker='o')
plt.plot(t_values, euler_values, label="Euler's Method", marker='o')
plt.plot(t_values, heun_values, label="Heun's Method", marker='o')
plt.plot(t_values, ralston_values, label="Ralston's Method", marker='o')
plt.plot(t_values, rk_values, label='4th-order RK Method', marker='o')

plt.title('Numerical Solutions for dy/dx = (1 + 2t) √x')
plt.xlabel('x')
plt.ylabel('y')
plt.legend()
plt.grid(True)
plt.show()
```


Numerical Solutions for dy/dx = (1 + 2t) √x
Analytical
Euler's Method
Heun's Method
Ralston's Method
4th-order RK Method
2.75
2.50
2.25
2.00
1.75
1.50
1.25
1.00
y
0.0
0.2
0.4
0.6
0.8
1.0
x