(A)

import numpy as np

import math

def getInitalVals(a, h):

    return [f(a + 3\*h), f(a + 2\*h), f(a + h), f(a)]

def f(t):

    return -1 + math.log(2 - math.exp(1 - math.exp(t)))

def f\_p(t, y):

    return math.exp(t)\*(2\*math.exp(-1\*(y+1)) - 1)

def AdamsB(h, temp\_arr, t):

    w3, w2, w1, w0 = temp\_arr

    return w3 + (h/24)\*(55\*f\_p(t + 3\*h, w3) - 59\*f\_p(t + 2\*h, w2) +

                     37\*f\_p(t + h, w1) - 9\*f\_p(t, w0))

def AdamsM(h, temp\_arr, t):

    wp, w3, w2, w1 = temp\_arr

    return w3 + (h/24)\*(9\*f\_p(t + 4\*h, wp) + 19\*f\_p(t + 3\*h, w3)

                     - 5\*f\_p(t + 2\*h, w2) + f\_p(t + h, w1))

def TrueV(a):

    return f(a)

def Error(true, pred):

    return abs(pred - true)

def AdamsPredCorr(a, b, h, p\_arr):

    n = int((b - h\*3)/h)

    temp\_arr = np.zeros((len(p\_arr)))

    temp\_arr = p\_arr

    for i in range(0, n + 1):

        t = h\*i

        wp = AdamsB(h, temp\_arr, t)

        temp\_arr = np.roll(temp\_arr, 1)

        temp\_arr[0] = wp

        w = AdamsM(h, temp\_arr, t)

    return w

def get\_h\_arr():

    h\_arr = np.zeros((7))

    for i in range(3, 10):

        h\_arr[i - 3] = 1/2\*\*i

    return h\_arr

def main():

    a = 0

    b = 1

    h\_arr = get\_h\_arr()

    true = TrueV(b)

    for i in range(0, len(h\_arr)):

        h = h\_arr[i]

        InitVals = getInitalVals(a, h)

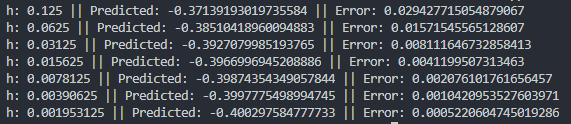
        w = AdamsPredCorr(a, b, h, InitVals)

        error = Error(true, w)

        print('h:', h, '|| Predicted:', w, '|| Error:', error)

if \_\_name\_\_ == "\_\_main\_\_":

    main()

Output:

# Notice that Error is directly proportionally to h (1/2 h -> 1/2 Error)

(B) Below in Math work

#4(a)(b)(c)(d)

Below in Math work

import numpy as np

import matplotlib.pyplot as plt

import math

def Method(a, b, alp, h):

*# Finding total number of iterations in [a,b]*

    N = int((b - a) / h)

*# Initial Condidtions*

    w = alp

    w1 = 1 - math.exp(-h)

*# Using difference iteration method*

    w\_arr = np.zeros((N + 1))

    t\_arr = np.zeros((N + 1))

    w\_arr[0] = w

    w\_arr[1] = w1

    t\_arr[0] = a

    t\_arr[1] = a + h

    for i in range(0, N - 1):

        w2 = 4\*w1 -3\*w - 2\*h\*(1 - w)

        w\_arr[i + 2] = w2

        t\_arr[i + 2] = t\_arr[i + 1] + h

        w = w1

        w1 = w2

    return w2, w\_arr, t\_arr

def true\_func(t):

    x = np.zeros((len(t)))

    for i in range(0, len(x)):

        x[i] = -math.exp(-t[i]) + 1

    return x

def main():

    a = 0

    b = 1

    alpha = 0

    h = 0.1

    approx, w\_arr, t\_arr = Method(a, b, alpha, h)

    h = 0.01

    approx2, w\_arr2, t\_arr2 = Method(a, b, alpha, h)

    true\_arr = true\_func(t\_arr)

    plt.plot(t\_arr, w\_arr, label='h=0.1')

    plt.plot(t\_arr, true\_arr, label='true')

    plt.plot(t\_arr2, w\_arr2, label='h=0.01')

    plt.legend()

    plt.show()

if \_\_name\_\_ == "\_\_main\_\_":

    main()

A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generated

#2c

import numpy as np

import math

def f1(t, u1, u2, u3):

    return u1 + 2\*u2 - 2\*u3 + math.exp(-t)

def f2(t, u1, u2, u3):

    return u2 + u3 - 2\*math.exp(-t)

def f3(t, u1, u2, u3):

    return u1 + 2\*u2 + math.exp(-t)

def k11 (h, t, u1, u2, u3):

    return h\*f1(t, u1, u2, u3)

def k12 (h, t, u1, u2, u3):

    return h\*f2(t, u1, u2, u3)

def k13 (h, t, u1, u2, u3):

    return h\*f3(t, u1, u2, u3)

def k21 (t, h, u1, k11x, u2, k12x, u3, k13x):

    return h\*f1(t + h/2, u1 + k11x/2, u2 + k12x/2, u3 + k13x/2)

def k22 (t, h, u1, k11x, u2, k12x, u3, k13x):

    return h\*f2(t + h/2, u1 + k11x/2, u2 + k12x/2, u3 + k13x/2)

def k23 (t, h, u1, k11x, u2, k12x, u3, k13x):

    return h\*f3(t + h/2, u1 + k11x/2, u2 + k12x/2, u3 + k13x/2)

def k31 (t, h, u1, k21x, u2, k22x, u3, k23x):

    return h\*f1(t + h/2, u1 + k21x/2, u2 + k22x/2, u3 + k23x/2)

def k32 (t, h, u1, k21x, u2, k22x, u3, k23x):

    return h\*f2(t + h/2, u1 + k21x/2, u2 + k22x/2, u3 + k23x/2)

def k33 (t, h, u1, k21x, u2, k22x, u3, k23x):

    return h\*f3(t + h/2, u1 + k21x/2, u2 + k22x/2, u3 + k23x/2)

def k41 (t, h, u1, k31x, u2, k32x, u3, k33x):

    return h\*f1(t + h, u1 + k31x, u2 + k32x, u3 + k33x)

def k42 (t, h, u1, k31x, u2, k32x, u3, k33x):

    return h\*f2(t + h, u1 + k31x, u2 + k32x, u3 + k33x)

def k43 (t, h, u1, k31x, u2, k32x, u3, k33x):

    return h\*f3(t + h, u1 + k31x, u2 + k32x, u3 + k33x)

def w1 (u1, k11x, k21x, k31x, k41x):

    return u1 + (1/6)\*(k11x + 2\*k21x + 2\*k31x + k41x)

def w2 (u2, k12x, k22x, k32x, k42x):

    return u2 + (1/6)\*(k12x + 2\*k22x + 2\*k32x + k42x)

def w3 (u3, k13x, k23x, k33x, k43x):

    return u3 + (1/6)\*(k13x + 2\*k23x + 2\*k33x + k43x)

def u1\_true(t):

    return -3\*math.exp(-t) - 3\*math.sin(t) + 6\*math.cos(t)

def u2\_true(t):

    return 3\*math.exp(-t)/2 + 0.3\*math.sin(t) - 2.1\*math.cos(t) - 0.4\*math.exp(2\*t)

def u3\_true(t):

    return -math.exp(-t) +12\*math.cos(t)/5 + 9\*math.sin(t)/5 - 2\*math.exp(2\*t)/5

def Error(u1, u2, u3):

    return abs(u1 - u1\_true(1)), abs(u2 - u2\_true(1)), abs(u3 - u3\_true(1))

def RK4\_Systems3(u1, u2, u3, a, b, h):

    N = int((b - a) / h)

    for i in range(0, N):

        t = a + i\*h

        k11x = k11(h, t, u1, u2, u3)

        k12x = k12(h, t, u1, u2, u3)

        k13x = k13(h, t, u1, u2, u3)

        k21x = k21(t, h, u1, k11x, u2, k12x, u3, k13x)

        k22x = k22(t, h, u1, k11x, u2, k12x, u3, k13x)

        k23x = k23(t, h, u1, k11x, u2, k12x, u3, k13x)

        k31x = k31(t, h, u1, k21x, u2, k22x, u3, k23x)

        k32x = k32(t, h, u1, k21x, u2, k22x, u3, k23x)

        k33x = k33(t, h, u1, k21x, u2, k22x, u3, k23x)

        k41x = k41(t, h, u1, k31x, u2, k32x, u3, k33x)

        k42x = k42(t, h, u1, k31x, u2, k32x, u3, k33x)

        k43x = k43(t, h, u1, k31x, u2, k32x, u3, k33x)

        u1 = w1(u1, k11x, k21x, k31x, k41x)

        u2 = w2(u2, k12x, k22x, k32x, k42x)

        u3 = w3(u3, k13x, k23x, k33x, k43x)

    return u1, u2, u3

def main():

    a = 0

    b = 1

    u1 = 3

    u2 = -1

    u3 = 1

    h = 0.1

    w1, w2, w3 = RK4\_Systems3(u1, u2, u3, a, b, h)

    e1, e2, e3 = Error(w1, w2, w3)

    print('Pred u1:', w1, '|| Error:', e1)

    print('Pred u2:', w2, '|| Error:', e2)

    print('Pred u3:', w3, '|| Error:', e3)

if \_\_name\_\_ == "\_\_main\_\_":

    main()

Output:



(C) Below in math work

(D) Below in math work