Applied Computational Methods in Mechanical Sciences

(ME466)

Assignment 6

Himanshu Kumar

16ME234

October 8, 2019

Problem Statement:

A ball at $1200K$ is allowed to cool down in air at ambient temperature of $300K$. Assuming heat is lost only due to radiation, the governing differential equation for heat loss is given as:

$$\frac{d\theta}{dt} = -2.2067 \times 10^{-12} \ (\theta^4 - 81 \times 10^8)$$

with initial condition as: $\theta \ 0 \ = 1200K$. The temperature after $480s$ from analytical solution is:

$$\theta(480s) = 647.57K$$

Using Euler's method and Modified Euler's method, find the optimal value of step size and number of iteration taken such that the result well approximates the true value.

Error limit is taken as 0.1%.

Python Code:

```
import math
import time
import matplotlib.pyplot as plt

# function derivative as a method of the function "f" itself
def f_dashx(a,b,f,x):
    f_dash = a*((f*f*f*f) -b)
    return(f_dash)

#calculation of fucntion value using euler method/ RK-1
# ivc stands for initial value condition
```

```
def rk1(f_dash,ivc,step,iters,args_const):
    fun_val = ivc[0]
    x_val = ivc[1]
    for i in range(iters):
        args_fdash = args_const + (fun_val,x_val,)
        fun_val = fun_val + f_dashx(*args_fdash)*step
        #moving to new point.
        x_val = x_val + step
        #print("\t f=",fun_val,"\t x =",x_val);
    #loop end

    #print("\n Final Val = ",fun_val,"\t at x=",x_val,"\t iterations = ",iters)
    return(fun_val)

def rk2(f_dash,ivc,step,iters,args_const):
    fun_val = ivc[0]
    x_val = ivc[1]

    for i in range(iters):
        sudo_args_fdash1 = args_const + (fun_val,x_val,)
        slope1 = f_dashx(*sudo_args_fdash1)
        sudo_fun_val1 = fun_val + slope1*step

        #moving to new sudo point1.
        sudo_x_val1 = x_val + step

        sudo_args_fdash2 = args_const + (sudo_fun_val1,sudo_x_val1,)
        slope2 = f_dashx(*sudo_args_fdash2)
        sudo_fun_val2 = sudo_fun_val1 + slope2*step

        #moving to new sudo point2.
        sudo_x_val2 = sudo_x_val1 + step

        #calculation of mean slope
        slope_mean = (slope1 + slope2)/2

        #moving to new point.
        fun_val = fun_val + slope_mean*step
        x_val = x_val + step
```

```
        #print("\t f=",fun_val,"\t x =",x_val);
    #loop end

    #print("\n Final Val = ",fun_val,"\t at x=",x_val,"\t iterations = ",iters)
    return(fun_val)


#optimal step size finding
def optimum_rk(rk,f_dashx,ivc,x_range,const_args):
    i=0
    true_val = 647.57
    step_size=0
    px=[]
    py=[]
    while (1):
        i=i+1
        step_size = (x_range[1] - x_range[0])/i
        numeric_val = rk(f_dashx,ivc,step_size,i,const_args)

        #calculation of error
        err = abs((numeric_val - true_val)/true_val)
        #print("error = ",err*100,"%")

        px.append( step_size )
        py.append(  err*100 )
        if(err<0.001):
            #print("\n ERROR LESS THAN 0.1%, OVER")
            break

    plt.plot(px, py)
    plt.xlabel('Step Size (h)')
    plt.ylabel('Error (%)')
    plt.title('Step Size vs. Error')
    plt.show()

    print("\n RESULTS FOR OPTIMALITY: step_size = ",step_size,"\t no. of iterations = ",i)
    print ("\n CPU time: ", time.process_time(),'s')
```

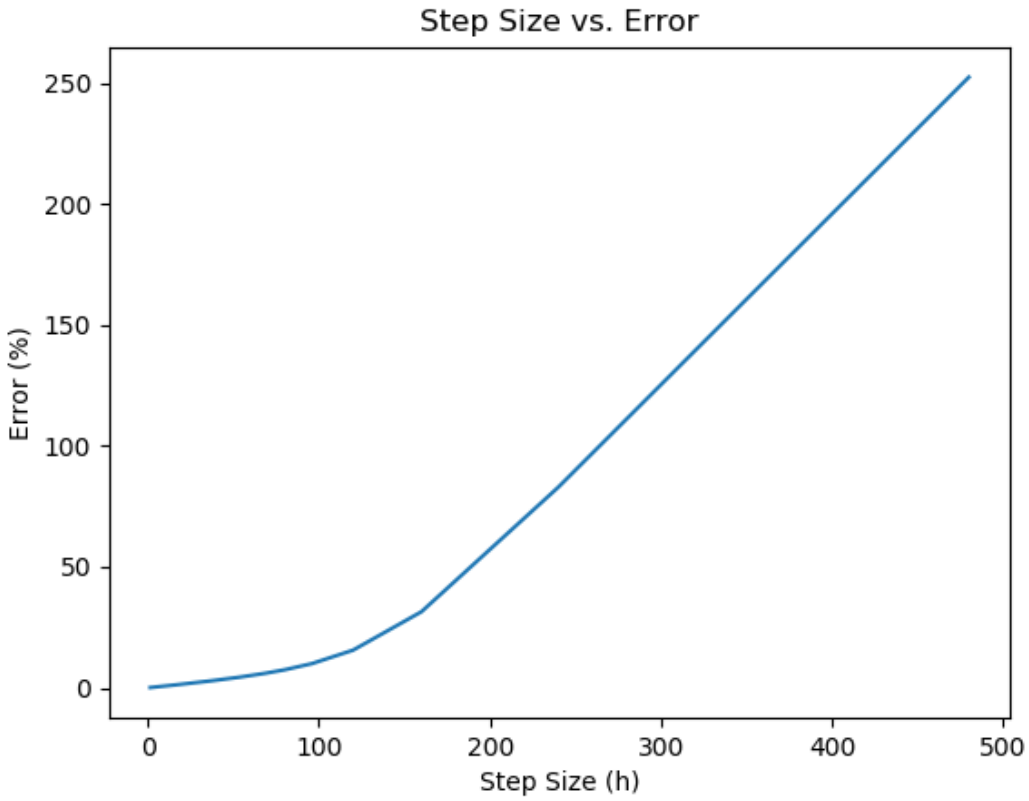
```
#calling in main program
a = -2.2067*pow(10,-12)
b = 81*pow(10,8)
ivc = (1200,0)
x_range = (0,480)
const_args = (a,b)

#rk2(f_dashx,ivc,120,4,const_args)
optimum_rk(rk1,f_dashx,ivc,x_range,const_args)
#optimum_rk(rk2,f_dashx,ivc,x_range,const_args)
```

Results:

1. With Euler's method:

```
2.  RESULTS FOR OPTIMALITY: step_size =  1.3953488372093024       no. of iterations =  344
3.
4.   CPU time:  0.78125 s
```



2. With Modified Euler's method:

```
3.  RESULTS FOR OPTIMALITY: step_size =  30.0          no. of iterations =  16
4.
5.   CPU time:  0.609375 s
```