Applied Computational Methods in Mechanical Sciences

(ME466)

Assignment 6

Himanshu Kumar

16ME234

October 8, 2019

**Problem Statement:**

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

with initial condition as: . The temperature after from analytical solution is:

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)

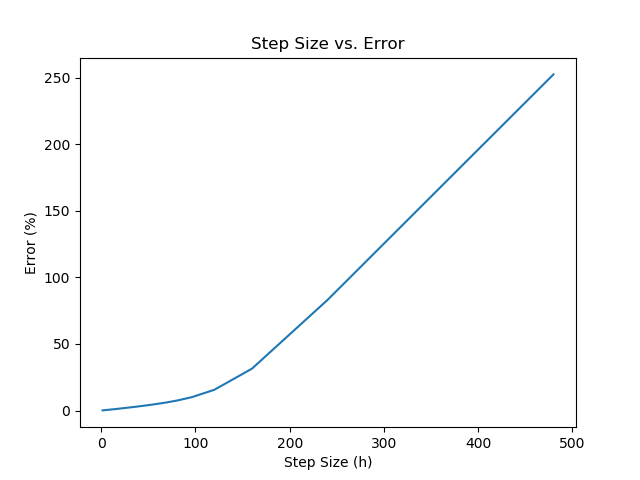
#rk2(f\_dashx,ivc,120,4,const\_args)

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. CPU time: 0.78125 s



1. **With Modified Euler’s method:**
2. RESULTS FOR OPTIMALITY: step\_size = 30.0 no. of iterations = 16
3. CPU time: 0.609375 s

