04/13/18 02:08:25 /home/jorrit/git/StarandPlanetform/Orbit_calculations/Kriekscode/RungeKutta2.py

```
  1  from __future__ import division
  2  import initialOrbitals as ic
  3  import numpy as np
  4  import math
  5  import theforacc2 as forc
  6
  7
  8  # Perform the Runge-Kutta calculations for both the star and the planet
  9  # def calcK(xp,yp,vxp,vyp,axp,ayp,xs,ys,vxs,vys,axs,ays,dt):
 10
 11  def RKiter(dx,dy,x,y,dt,kuttax,kuttay,kuttadx,kuttady,j,i,instances):
 12      """
 13      Calculate the Runge-Kutta terms for the RK4-Method
 14      """
 15  # #  if at the first kutta term do this:
 16  #     if kuttax ==0 and kuttay ==0:
 17  #         kx = dx + kuttadx*dt
 18  #         ky = dy + kuttady*dt
 19  #         kdx,kdy = calcaccx(x + kuttax*dt,y+kuttay*dt,j,i,instances,dt)
 20  # # if at any other kutta term do this:
 21  #     else:
 22      kx = dx + kuttadx*dt
 23      ky = dy + kuttady*dt
 24      kdx,kdy = calcaccx(x + kuttax*dt,y+kuttay*dt,j,i,instances,dt,dx + kuttadx*dt,dy +
    kuttady*dt)
 25
 26      return kx,ky,kdx,kdy
 27
 28
 29  def calcK(instances,dt):
 30      """Here we call on all the RK functions"""
 31      # Reset the rungevalues atributes for each object
 32      for j in instances:
 33          j.rungevalues = []
 34          j.rungevalues.append([0,0,0,0])
 35
 36  # loop through the runge kutta terms
 37      for i in np.arange(1,5):
 38          # loop through the objects
 39          for j in instances:
 40              # as long as we haven't reached the final runge kutta term do this:
 41              if i != 4:
 42                  kuttax,kuttay,kuttadx,kuttady =
    RKiter(j.vx,j.vy,j.x,j.y,dt*0.5,j.rungevalues[i-1][0],j.rungevalues[i-1][1],j.rungevalues[i-1]
    [2],j.rungevalues[i-1][3],j,i,instances)
 43                  j.rungevalues.append([kuttax,kuttay,kuttadx,kuttady])
 44
 45              if i == 4:
 46                  kuttax,kuttay,kuttadx,kuttady = RKiter(j.vx,j.vy,j.x,j.y,dt,j.rungevalues[i-1]
    [0],j.rungevalues[i-1][1],j.rungevalues[i-1][2],j.rungevalues[i-1][3],j,i,instances)
 47                  j.rungevalues.append([kuttax,kuttay,kuttadx,kuttady])
 48
 49  # loop through all the instances and then change coordinates and list
 50      for j in instances:
 51
 52          # if 'HW' in j.name:
 53              # continue
 54
 55          j.x = j.x + ((1./6)*(j.rungevalues[1][0]+(2*j.rungevalues[2][0])+(2*j.rungevalues[3]
    [0])+j.rungevalues[4][0]))*dt
 56          j.y= j.y + ((1./6)*(j.rungevalues[1][1]+(2*j.rungevalues[2][1])+(2*j.rungevalues[3]
    [1])+j.rungevalues[4][1]))*dt
 57          j.vx = j.vx + ((1./6)*(j.rungevalues[1][2]+(2*j.rungevalues[2][2])+(2*j.rungevalues[3]
    [2])+j.rungevalues[4][2]))*dt
 58          j.vy = j.vy + ((1./6)*(j.rungevalues[1][3]+(2*j.rungevalues[2][3])+(2*j.rungevalues[3]
    [3])+j.rungevalues[4][3]))*dt
 59          j.xlist.append(j.x)
 60          j.ylist.append(j.y)
 61
 62
 63
 64  # Calculate R,theta,F and acceleration and return the acceleration
 65  def calcaccx(x, y, j, i, instances, dt,vx,vy):
 66  # """Function call for calculating acceleration"""
 67      axlist = []
```

```
68      aylist = []
69      # loop through the instances that are not the instance which we called this object
70      # and calculate the Force ,R theta and acceleration from having put the instance at the
    coordinate (0,0)
71
72      for g in instances:
73          if g != j:
74              # print j.name, g.name
75
76              if 'HW' in g.name:
77                  continue
78              else:
79
80                  if 'HW' in j.name and g.name == "Earth":
81                      continue
82
83                  else:
84                      # print j.expl_name, ':', g.expl_name
85                      x_diff = g.x + g.rungevalues[i-1][0]*dt - x
86                      y_diff = g.y + g.rungevalues[i-1][1]*dt - y
87
88                      R = forc.calcDist(x_diff, y_diff)
89                      theta = forc.calcTheta(x_diff, y_diff)
90                      F = forc.calcForce(R, j.mass, g.mass)
91
92                      ax, ay = forc.calcAcc(x_diff, y_diff, F, j.mass, theta)
93
94                      axlist.append(ax)
95                      aylist.append(ay)
96
97      if j.name == "Earth":
98          agasx,agasy = forc.calcDrag(x,y,j,vx,vy,dt)
99          axlist.append(agasx)
100         aylist.append(agasy)
101
102     if 'HW' in j.name:
103         agasx,agasy = forc.calcDragHW(x,y,j,vx,vy,dt, j.headwind)
104         axlist.append(agasx)
105         aylist.append(agasy)
106
107     axlist = np.array(axlist)
108     aylist = np.array(aylist)
109
110     return np.sum(axlist),np.sum(aylist)
111
112 # Runge-kutta function
113 def calcRK(dt):
114     """stupid function please ignore"""
115     calcK(ic.Orbitals.instances,dt)
```