Falling particle

We use a simple numerical scheme to compute the velocity and position of the particle. It is not the best numerical scheme, but if suffices for our purpose. We try to find a new value for the velocity and for the position by letting the time advance by a small value of dt.
The basic idea is that we solve a first order differential equation by reworking its mathematical definition to ‘taking the limit of dt almost to zero’.
Suppose we have a differential equation of the form

$$\frac{d\mathrm{x}}{dt} = f \Rightarrow \frac{x(\mathrm{t}+\mathrm{dt}) - \mathrm{x}(\mathrm{t})}{\mathrm{dt}} \approx f(t) \Rightarrow x(t+dt) \approx x(t) + f(t) * dt$$

In computer coding it is easier to number the elements of x and use as notation x[i]. This means the value of x corresponding to time t[i]. The latter is t[i] = t[i-1]+dt = i*dt

We apply the above to N2 by writing it as:

$$\frac{dv}{dt} = F/m \;\Rightarrow v[i] = v[i-1] + \frac{F[i\text{-}1]}{m}dt$$

The smaller the value of dt, the more accurate our calculation.
From the definition of the velocity, we can find the position:

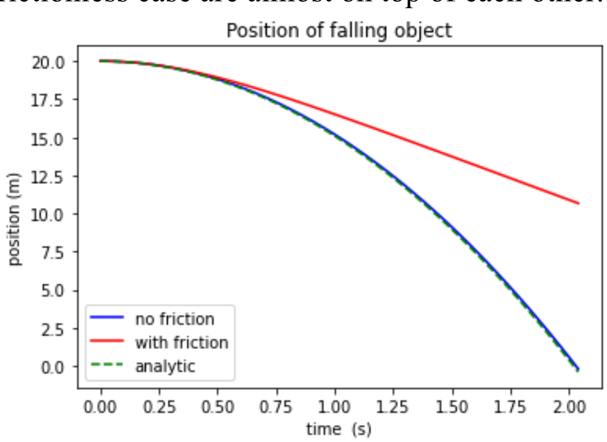
$$\mathrm{v} = \frac{dx}{dt} \;\Rightarrow \mathrm{x}[i] = \mathrm{x}[i-1] + \mathrm{v}[\mathrm{i\text{-}1}]dt$$

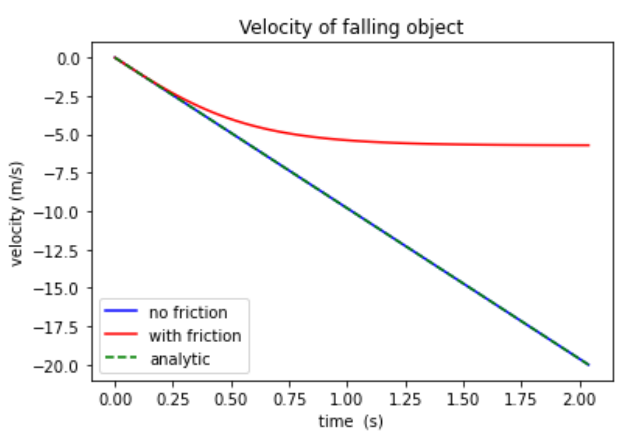
If we do this in an iterative scheme, we get values for position and velocity on discrete times, (0,dt,2dt, 3dt, …)

We will write a program in which a particle of density 1.0 10^3 kg/m^3 and diameter of 3mm will fall from a height H =20 straight down. The particle has zero initial velocity.
N2 for this case reads as

$$m\frac{dv}{dt} = -mg - A_{\perp}C_D\frac{1}{2}\rho_{air}v^2$$

We compute velocity and position with drag on the particle and without. Besides, we use the analytical solution to see how good our numerical one is in case of no-friction.
The results are given below. Note that the analytic solution and the computed one for the frictionless case are almost on top of each other.





A Python code is given on the next page.

```python
"""
dropping a stone to find gravity's acceleration, taking air friction into account
"""


import numpy as np
import matplotlib.pyplot as plt


pi=np.pi                            #set pi=3.1415...                   [-]
rho_p = 1e3                         #density of particle                [kg/m3]
rho_air=1.2                         #density of air                     [kg/m3]
D=3.0e-3                            #diameter of (spherical) particle   [m]
Aperp = pi*D*D/4.0                  #frontal area of a sphere           [m2]
CD = 1                              #drag coefficient                   [-]
m=pi/6.0*rho_p*D*D*D                #mass of particle                   [kg]
grav=9.81                           #gravity's acceleration             [m/s2]
H=20.0                              #initial height at t=0              [0]

dt=0.02                             #time step                          [s]
N=100                               #number of iterations
v=[]                                #particle velocity
x=[]                                #particle position
t=[]                                #time
y=[]                                #frictionless particle position
w=[]                                #frictionless particle velocity
z=[]                                #analytic solution (frictionless)
u=[]                                #analytic solution (frictionless)
t.append(0.0)
x.append(0.0)
v.append(0.0)
y.append(0.0)
w.append(0.0)



#compute drag force per unit mass and add gravity per unit mass
def f(v):
   f=-grav - CD*Aperp*0.5*rho_air*v*v*np.sign(v)/m
   return f
```

```
#start iterations
for i in range (1,N):
        x.append(x[i-1] + dt*v[i-1])
        v.append(v[i-1] + dt*f(v[i-1]))
        t.append(t[i-1] + dt)
        y.append(y[i-1] + dt*w[i-1])
        w.append(-grav*t[i])
        if y[i]<0:
                break           #stop iterating if free falling particle has reached the ground
        print("i= ",i,"  x=", x[i],"  y=",y[i],"   v= ",v[i],"   w= ",w[i])

#plot velocity
plt.plot(t,w,'b-',t,v,'r-',t,u,'g—')
# Add title and axis names
plt.title('Velocity of falling object')
plt.xlabel('time  (s)')
plt.ylabel('velocity (m/s)')
plt.legend(["no friction", "with friction","analytic"], loc ="lower left")
plt.show()

#plot position
plt.plot(t,y,'b-',t,x,'r-')
# Add title and axis names
plt.title('Position of falling object')
plt.xlabel('time  (s)')
plt.ylabel('position (m)')
plt.legend(["no friction", "with friction","analytic"], loc ="lower left")
plt.show()

v_inf=v[N-1]
print("v_inf = ",v_inf)
```