Experiment report about a parabolic model in python

Clearly define the objectives of the project in your own words

Implement a parabolic model with air resistance in python

Outline the minimum expectations of the system in the program description

1.Able to read various system parameters

2.Physical parameters at each moment of the ball can be calculated by using physical formulas

3.Able to print the calculation results in the window

4.Able to save the calculation results into a file

5.It can display the trajectory of ball statically

6.Dynamic display of ball movement

You can also include a list of requirements required to extend the system

1.Read system parameters through files

2.Support multiple balls to simulate together

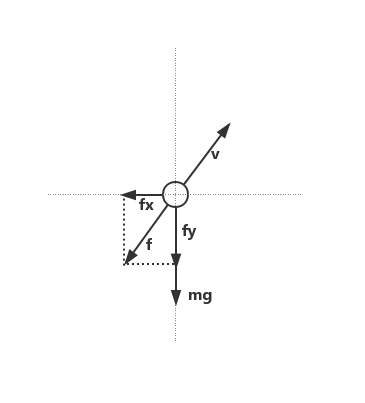
3.The process of bouncing back after hitting the ground

4.Design more user-friendly interfaces

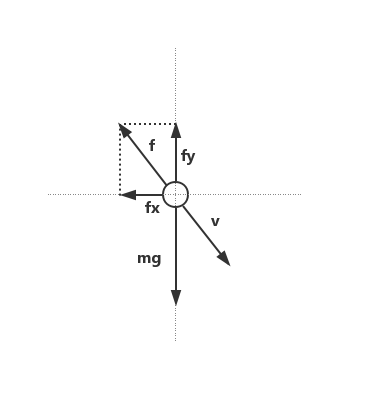
This should make full use of academic resources

To includes an explanation of the forces acting on the particle (you can include an annotated diagram)

1. When the ball is going up, the vertical component of resistance has the same direction as gravity.



1. When the ball is going down, the vertical component of resistance and gravity are in the opposite directions.



Design

An overview of all parameters and variables of the system, which should include units of measure per unit

The input:

mass(kg)

gravity\_acc(m/s^2)

velocity(m/s)

angle(degrees)

drag\_cff

time\_step(s)

total\_time(s)

The output：

Time array t[]

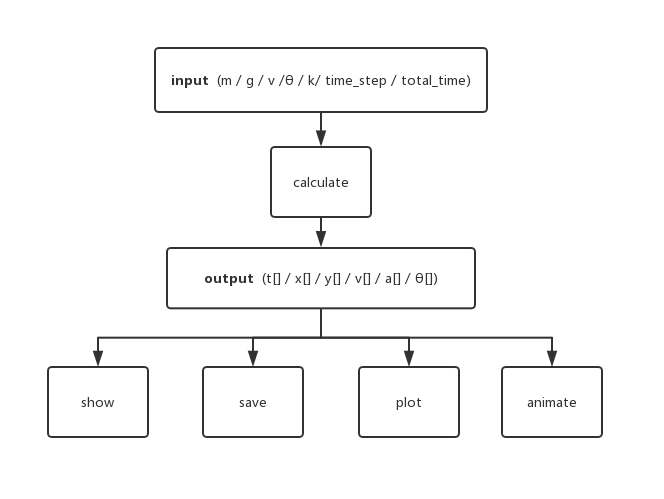
Horizontal coordinate array x[]

Array y[]

Array of velocity v[]

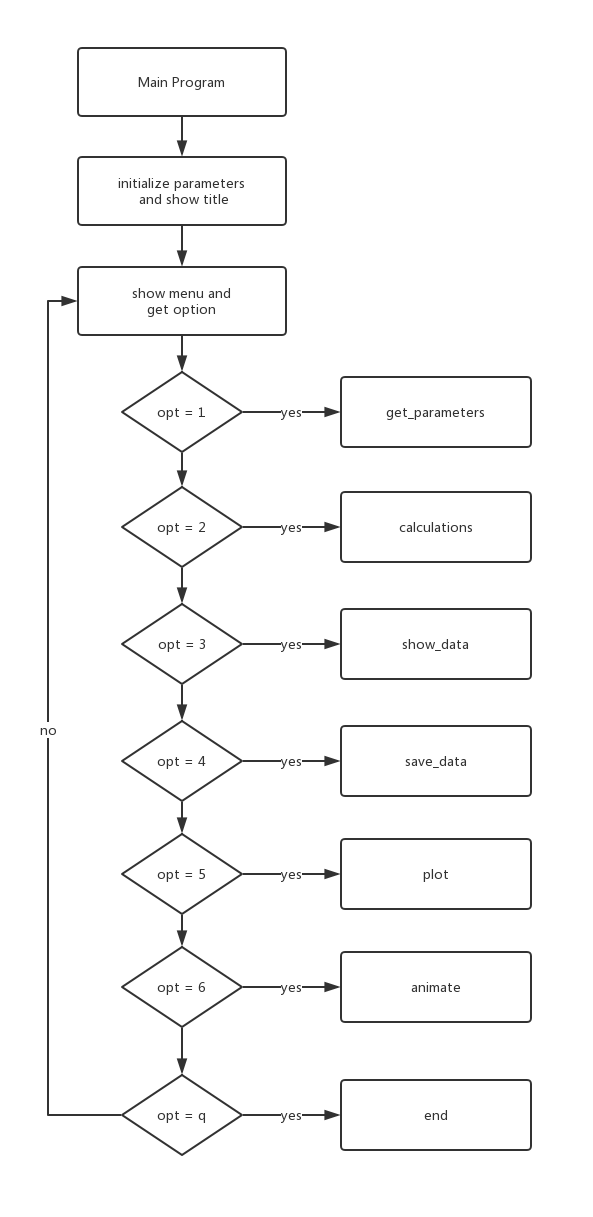
Acceleration array a[]

Theta array θ[]



Create a structure diagram that shows an outline of the component

Your system



Equations outline all related equations and explain their use (step by step)

At any given moment, the ball satisfies the following equations

Stress：The force of gravity(mg ) and air resistance f = k \* v^2

Break down:

In the horizontal direction:m \* ax = f \* cos(θ)

In the vertical direction:m \* ay = f \* sin(θ) + mg

Get the acceleration:

ax = f \* cos(θ) / m

ay = f \* sin(θ) / m + g

The total acceleration is:

a[i] =

Take a small ball in an interval of time as a straight line of uniform acceleration:

Get the current displacement:

x[i] = x[i-1] + 0.5 \* ax[i-1] \* dt^2

y[i] = y[i-1] + 0.5 \* ay[i-1] \* dt^2

And current speed:

vx[i] = vx[i-1] + ax[i-1] \* dt

vy[i] = vy[i-1] + ay[i-1] \* dt

And the total velocity:

v[i] =

And update the theta in the next intervalθ

θ = atan(vy[i-1] / vx[i-1])

Test plan appropriate test plan that shows whether the test results are normal, incorrect, and correct

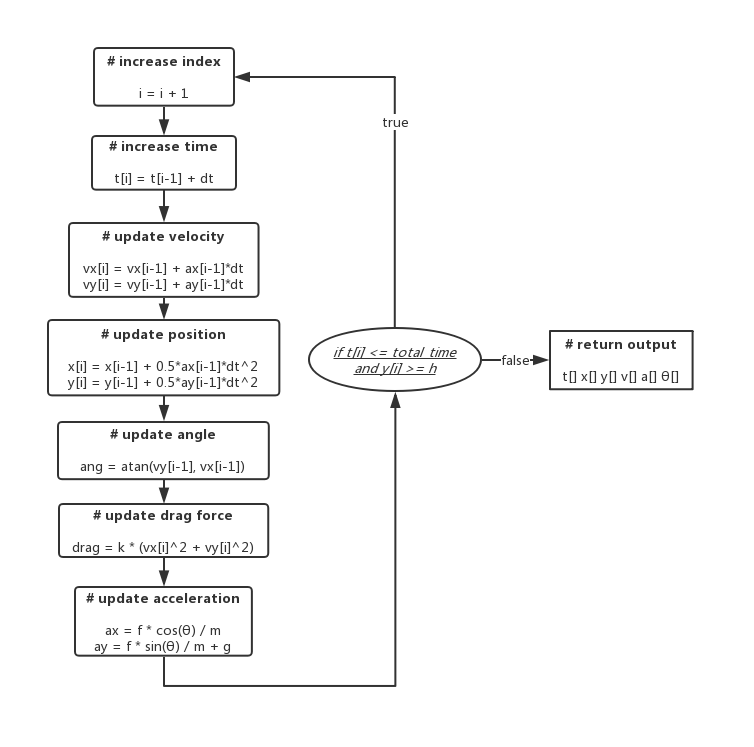
Boundary/extreme data

The following boundary conditions are all tested normal

|  |  |  |
| --- | --- | --- |
| θ | -30 | Throw down |
| θ | 0 | Horizontal cast |
| θ | 90 | The vertical up |
| θ | 120 | Flip it up |

The implementation section should include the following:

elements including flowchart fully explain and demonstrate the flowchart shown in your code by the main algorithm



The default value for parameters:

* m = 1.00 (kg)
* g = 9.8 (m/s^2)
* h = 200.00 (m)
* v = 100.00 (m/s)
* angle = 60.00 (degrees)
* k = 0.005

The results:

i = 0 time = 0.00 x = 0.00 y = 0.00 velocity = 100.00 acceleration = 59.68 angle = 1.40

i = 1 time = 0.20 x = 3.30 y = 18.52 velocity = 88.07 acceleration = 48.46 angle = 1.40

i = 2 time = 0.40 x = 6.29 y = 34.89 velocity = 78.39 acceleration = 40.40 angle = 1.39

i = 3 time = 0.60 x = 9.04 y = 49.50 velocity = 70.32 acceleration = 34.40 angle = 1.39

i = 4 time = 0.80 x = 11.59 y = 62.64 velocity = 63.45 acceleration = 29.81 angle = 1.38

i = 5 time = 1.00 x = 13.97 y = 74.49 velocity = 57.50 acceleration = 26.21 angle = 1.38

i = 6 time = 1.20 x = 16.21 y = 85.24 velocity = 52.28 acceleration = 23.35 angle = 1.37

i = 7 time = 1.40 x = 18.34 y = 95.00 velocity = 47.63 acceleration = 21.03 angle = 1.36

i = 8 time = 1.60 x = 20.36 y = 103.88 velocity = 43.45 acceleration = 19.12 angle = 1.35

i = 9 time = 1.80 x = 22.29 y = 111.96 velocity = 39.66 acceleration = 17.55 angle = 1.34

i = 10 time = 2.00 x = 24.15 y = 119.32 velocity = 36.18 acceleration = 16.23 angle = 1.33

i = 11 time = 2.20 x = 25.94 y = 126.00 velocity = 32.98 acceleration = 15.13 angle = 1.32

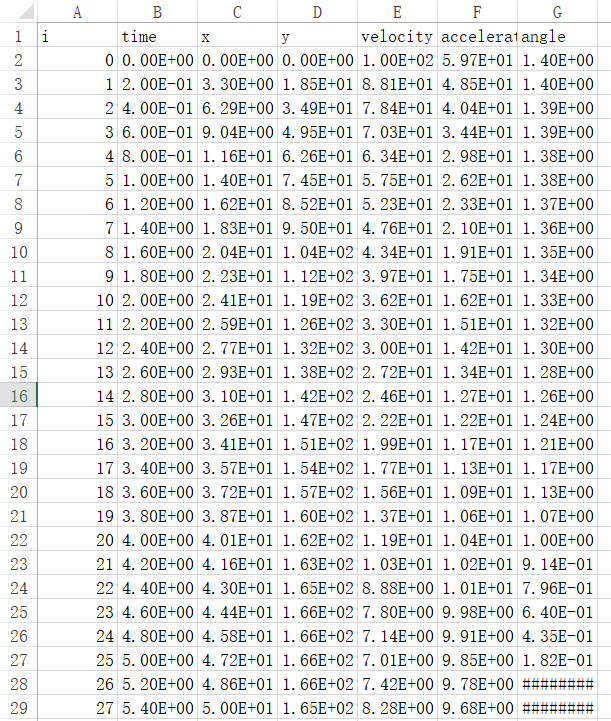
………………..

i = 75 time = 15.00 x = 87.10 y = -144.88 velocity = 43.38 acceleration = 0.49 angle = -1.54

i = 76 time = 15.20 x = 87.35 y = -153.56 velocity = 43.45 acceleration = 0.46 angle = -1.54

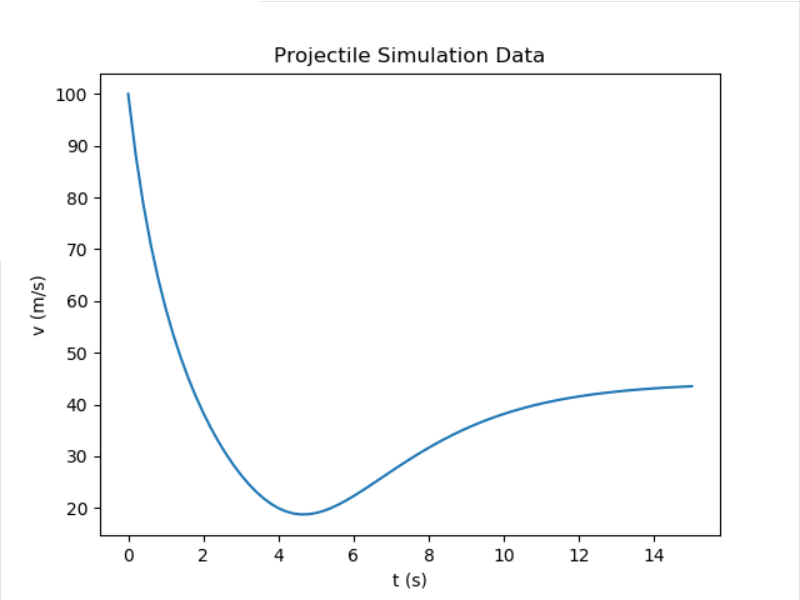
And the results can be save into files:



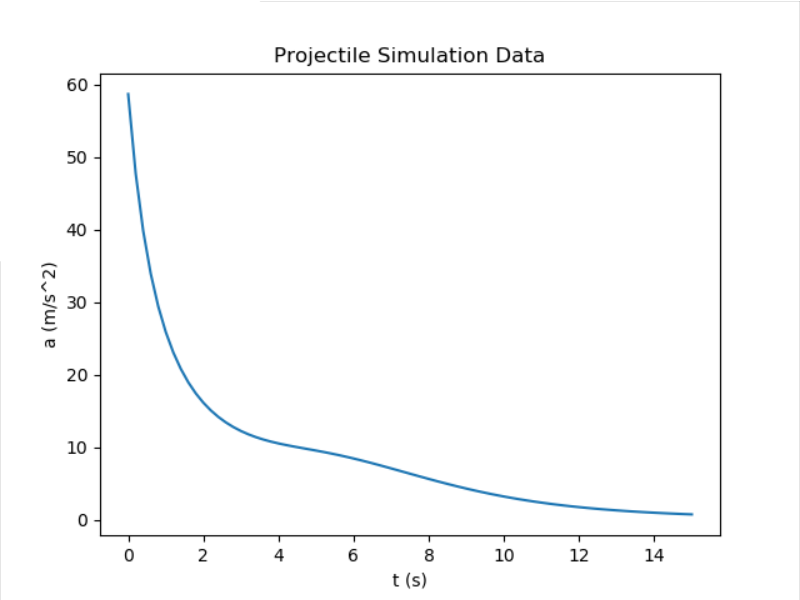


The conclusion：

1. If the time is long enough, the parabolic motion considering air resistance will eventually become a uniform and straight motion falling vertically.



v - t



a - t

1. Under the condition that the air resistance coefficient remains unchanged, the greater the initial velocity, the greater the Angle included in the up-throw, and the greater the maximum height the object can reach.

If the variables value as follows

* m = 1.00 (kg)
* h = 200.00 (m)
* v = 100.00 (m/s)
* angle = 60.00 (degrees)

The result is:

* max height = 132.87948437148643
* max length = 223.0482409083623
* total time = 14.999999999999979
* final velocity = 43.51576218308847

If v goes up to 200.00 (m/s), the result changes to:

* max height = 220.52371638199998
* max length = 287.36900714317454
* total time = 15.199999999999978
* final velocity = 43.32899716352561

If angle goes up to 80.00 (degrees), the result changes to:

* max height = 166.10185627215444
* max length = 87.34751451580493
* total time = 15.199999999999978
* final velocity = 43.453989618710864

Future improvement:

1.Read system parameters through files

2.Support multiple balls to simulate together

3.The process of bouncing back after hitting the ground

4. Design more user-friendly interfaces

Reference:

[1] https://www.fisicalab.com/en/section/uarm-equations#contents

[2] https://courses.lumenlearning.com/physics/chapter/3-4-projectile-motion/

[3] https://matplotlib.org/tutorials/introductory/pyplot.html

[4] https://www.tutorialspoint.com/python/python\_graphs.htm

The appendix