

The deadline for this exercise is **Monday 28th November 2022** at 12:30 p.m. A short report and all program (*.py) files should be uploaded into Blackboard at the appropriate point in the *PHYS_20032_30009_2022_TB-4: Introduction to Computational Physics (PHYS20032 + PHYS30009) 2022 Course*.

S. Hanna

Objectives of the exercise

- To become familiar with some basic tools for solving ordinary differential equations;
- To apply the Euler method to a 1D free fall problem with varying air resistance;
- To explore the accuracy of the finite difference method applied to solving the wave equation;
- To gain experience in the writing of technical reports; N.B. a report will be required for this exercise.

Problem 1 (60% of marks): Free-fall with fixed or varying drag

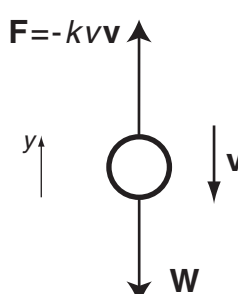
On 14th October 2012, Felix Baumgartner set the world record for falling from a great height. He jumped from a helium balloon at a height of 39045 m, fell for 4 minutes and 19 seconds and reached a maximum speed of 373 m.s^{-1} . In this problem, you will solve the equations of motion for a free-falling object, and use your program to confirm, or otherwise, Felix Baumgartner's statistics.¹

For a projectile travelling at speed through the air, the air resistance is proportional to the square of the velocity and acts in the opposite direction. i.e. $\mathbf{F} = -k\mathbf{v}^2\hat{\mathbf{v}} = -k|\mathbf{v}|\mathbf{v} = -k\mathbf{v}\mathbf{v}$. The constant, k , is given by:

$$k = \frac{C_d \rho_0 A}{2} \quad (1)$$

in which C_d is the drag coefficient (~ 0.47 for a sphere; $\sim 1.0 - 1.3$ for a sky diver or ski jumper), A is the cross sectional area of the projectile and ρ_0 is the air density ($\sim 1.2 \text{ kg m}^{-3}$ at ambient temperature and pressure).

In this problem, the acceleration is varying, so Newton's equations of motion produce a second order ODE to solve. As illustrated in lectures, we can do this if we separate it into two first order equations, one for the derivative of the velocity, the other for the position:



$$m\mathbf{a} = \mathbf{W} + \mathbf{F}$$

$$\text{i.e.} \quad m \frac{dv_y}{dt} = -mg - k|v_y|v_y \quad (2)$$

$$\text{and} \quad \frac{dy}{dt} = v_y \quad (3)$$

The y coordinate is taken vertically upwards. Euler's method for solving:

$$\frac{dy}{dt} = f(y, t)$$

is summarised by:

$$y_{n+1} = y_n + \Delta t \cdot f(y_n, t_n) \quad ; \quad t_{n+1} = t_n + \Delta t \quad (4)$$

in which we are determining y and t at the $(n+1)$ th step from their values at the n th step. Applying this scheme to Eqs. (2) and (3), we obtain:

$$v_{y,n+1} = v_{y,n} - \Delta t \left(g + \frac{k}{m} |v_{y,n}| v_{y,n} \right) \quad (5)$$

$$y_{n+1} = y_n + \Delta t \cdot v_{y,n} \quad (6)$$

$$t_{n+1} = t_n + \Delta t \quad (7)$$

If we provide the initial conditions i.e. y_0 and $v_{y,0}$, we can use the above scheme repeatedly to find y and v_y for all t .

¹In fact Felix Baumgartner's record stood until 24th October 2014, when Alan Eustace (a senior Google vice president) jumped from 41419 m, reaching a maximum speed of 367 m.s^{-1} . A sonic boom was heard by observers on the ground. However, Eustace's attempt used a drogue parachute, whereas Baumgartner's did not, so both records still stand.

Attempt the following programming tasks and address the points raised in the script in your report. Incorporate your code in a menu system similar to that used in the Exercise 1. Each of the sections (a) to (d) should correspond to a separate menu item:

- a) Write a Python program to plot the following analytical predictions for height y and vertical speed v_y as a function of time, for a free-falling object under constant gravity and constant drag factor, k :

$$y = y_0 - \frac{m}{k} \log_e \left[\cosh \left(\sqrt{\frac{kg}{m}} \cdot t \right) \right] \quad (8)$$

$$v_y = -\sqrt{\frac{mg}{k}} \tanh \left(\sqrt{\frac{kg}{m}} \cdot t \right) \quad (9)$$

N.B. These equations follow directly from solving Eqs. (2) and (3). You can set $y_0 = 1$ km, say, and calculate y and v_y for *any* t .

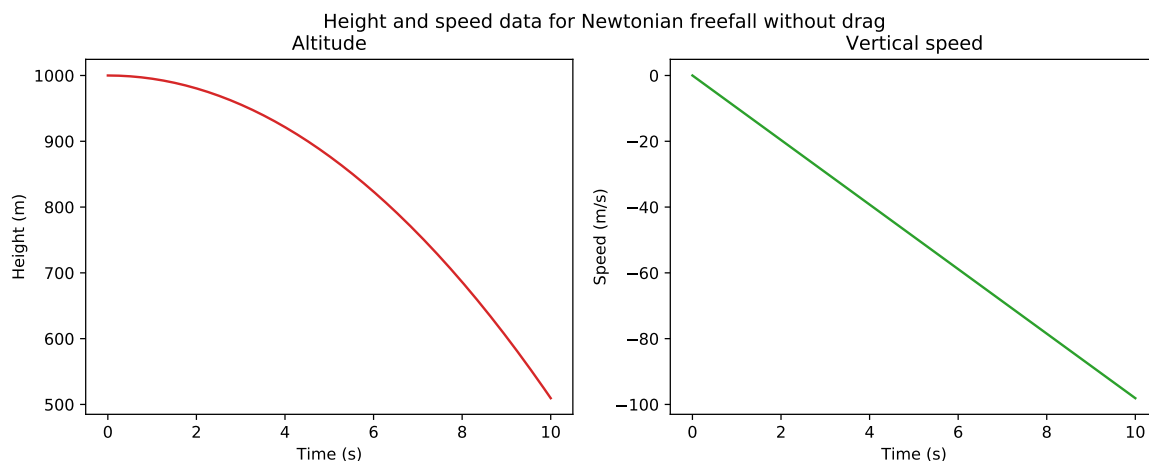
HINT: To generate the plot, you will need to create three arrays, two for the y and v_y values and the other for the time values. These can either be Python lists or Numpy arrays. This example demonstrates plotting of Numpy arrays (with sample output below):

```
import numpy as np
import matplotlib.pyplot as plt

g = 9.81 # m/s^2, acceleration
y0 = 1000 # m, initial height
NumPoints = 200
tmin = 0.0 # seconds
tmax = 10.0 # seconds
tvals = np.linspace(tmin, tmax, NumPoints)
yvals = np.zeros(NumPoints)
vyvals = np.zeros(NumPoints)

# equations of motion without drag
for i in range(NumPoints):
    vyvals[i] = -g*tvals[i] # you should define functions for Eqs (8) and (9)
    yvals[i] = y0 - 0.5*g*tvals[i]**2 # this will make comparison plots easier later on

# this is just one approach to the plotting
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12,4))
fig.suptitle('Height and speed data for Newtonian freefall without drag')
ax1.set(xlabel='Time (s)', ylabel='Height (m)', title='Altitude')
ax2.set(xlabel='Time (s)', ylabel='Speed (m/s)', title='Vertical speed')
ax1.plot(tvals, yvals, 'tab:red')
ax2.plot(tvals, vyvals, 'tab:green')
plt.show() # May not be needed
```



b) Now solve the free-fall problem using the Euler method, as outlined in Eqs. (5) to (7). Using a starting height of 1 km and zero initial velocity, calculate $y(t)$ and $v_y(t)$ for the falling body:

- You will need to create a loop and iterate repeatedly through Eqs. (5), (6) and (7), until you have filled your arrays of y , v_y and t values.
- You will need to provide sensible values for C_d , A and m .
- You will also need to specify a condition for ending the simulation i.e. when the body reaches the ground.
- Make sure you plot your results.

In your report you should consider the following:

- Verify the correct functioning of your program by comparing your results with the analytical prediction.
 - Examine the effect on your solution of varying the step size Δt . Investigate how closely your solution tracks the real trajectory for a range of step sizes. What happens to the simulation when Δt is very large?
 - Explore the effect on the motion of varying the ratio k/m .
- c) Now you are going to make the problem more realistic. Baumgartner jumped from very high altitude, where the air density is very low, and so the drag factor, k should be replaced with a function $k(y)$. The simplest way to approach this is to make use of the scale height for the atmosphere, h , and model the variation of density as an exponential decay:

$$\rho(y) = \rho_0 \exp(-y/h) \quad (10)$$

from which $k(y)$ follows using Eq. (1). An appropriate value of h appears to be 7.64 km.²

Adapt your program to use $\rho(y)$ instead of ρ_0 . Test your program using the parameters for Baumgartner's jump. Plot $y(t)$ and $v_y(t)$ against time. The most interesting feature of the motion is that the downwards speed goes through a maximum that is much greater than the terminal velocities predicted for constant ρ . Do you observe this?

d) There was great interest in whether Baumgartner would break the sound barrier during his jump. The speed of sound in a gas varies with the temperature:

$$v_s = \sqrt{\frac{\gamma RT}{M}}$$

where γ is 1.4 for air, R is the molar gas constant and T is the absolute temperature in Kelvin. M is the molar mass of the gas, which for dry air is about 0.0289645 kg/mol. The atmospheric temperature varies with altitude, H , as follows:

Troposphere:	$H \leq 11000 \text{ m}$	$T(\text{K}) = 288.0 - 0.0065H$
Lower Stratosphere:	$11000 < H \leq 25100 \text{ m}$	$T(\text{K}) = 216.5$
Upper Stratosphere:	$H > 25100 \text{ m}$	$T(\text{K}) = 141.3 + 0.0030H$

Using the information above, adapt your program to determine Baumgartner's maximum Mach number (fraction of the speed of sound) as he falls. Investigate the effect of varying the jump parameters (the jump height and the quantity $C_d A/m$) on the maximum speed achieved, and the total duration of the jump. According to your simulation, does Baumgartner manage to break the sound barrier?

Problem 2 (40% of marks): Waves

In this problem you will use a finite difference method to solve the wave equation, following the method discussed in lectures, for waves propagating along a 1-dimensional string. The equation to be solved is:

$$\frac{\partial^2 u}{\partial x^2} = \frac{1}{v^2} \frac{\partial^2 u}{\partial t^2}$$

Expressing the partial derivatives as finite differences and re-arranging leads to:

$$u(x, t + \tau) = \gamma [u(x + h, t) + u(x - h, t)] + 2u(x, t)(1 - \gamma) - u(x, t - \tau) \quad (11)$$

²see http://en.wikipedia.org/wiki/Scale_height

where:

$$\gamma = \frac{\tau^2 v^2}{h^2}$$

with τ being the time step, Δt , and h the lattice increment, Δx . v is the phase velocity, which for a string will be given by:

$$v = \sqrt{\frac{T}{\mu}}$$

where T is the tension in the string and μ is its mass per unit length. As noted in lectures, setting $\gamma = 1.0$ leads to an exact solution of the wave equation, but this puts some limits on the choice of v , τ and h .

e) Write a program to solve Eqn. (11) for a general value of γ , noting the following points:

- Choose sensible values for μ , T and h , and keep them fixed so that $\gamma \approx 1$.
- Use hard boundaries at the ends of your string i.e. $u(0, t) = u(L, t) = 0$.
- Initialise your simulation with a Gaussian wave function in the centre of your string, as shown in the example in lectures, but set the initial derivatives $u'(x, 0)$ so that the wave will propagate to the right.
- Plot graphs showing the wave propagation for $\gamma = 1.0$ after different time intervals.

Make sure you integrate your code with the menu from the first problem.

f) Repeat your simulations with values of $\gamma \neq 1$ and examine the accuracy of your solution compared with that for $\gamma = 1$. Do you observe changes in either the phase velocity, as measured in your graphs, or the shape (e.g. width and height) of the wave function? Is your simulation stable for all values of γ ?

Report

Your report should include, but not be restricted to, the following:

- A *brief* statement of the problem as you understand it;
- A *brief* description of the method used to solve the problem;
- Presentation of your results, in graphical format where applicable;
- A discussion of the results, which should include a critique of the method and ideas for improvement;
- Answers to any of the questions in the script, including appropriate discussion.

Generally, you should aim for conciseness, with most emphasis being placed on the presentation of your results and the discussion.

Submitting your work

You should submit the following to Blackboard:

1. A concise report, in MS Word or pdf format;
2. Final version of your program which should address both the free-fall problem and the wave equation.

As before, please note:

- Please upload your "prog.py" file, but first please rename your file with a ".txt" extension, for the benefit of the Blackboard plagiarism checker.
- **Please note, there is only a single upload point for your programs, so please ensure both problems are addressed in a single program and upload a single *.txt file to Turnitin.**
- Please also give your programs sensible distinguishing names, including your name or userid e.g. "my_userid_ex2_prob_1_and_2.txt".

If you have any problems submitting your work, please contact Dr. Hanna (s.hanna@bristol.ac.uk) or ask a demonstrator.