BPhO Computational Challenge

2024 Projectiles

Welcome to the **British Physics Olympiad Computational Challenge 2024.** The goal is to build *computer models* based upon the instructions in the **Challenge Presentation document**. These can mostly be achieved using a *spreadsheet* such as Microsoft Excel, although you are very much encouraged to use a *programming language* of your choice*.

The challenge runs from **Easter 2024 till August 2024.** To submit an entry you will need to fill in a web form at **https://www.bpho.org.uk/.**

Additional resources can be found at: **http://www.eclecticon.info/physics_bpho_compphys.htm**

The deliverable of the challenge is to produce a **screencast** of ***maximum length two minutes*** which describes your response to the challenge, i.e. the graphs and the code & spreadsheets and your explanation of these. Your video should make it really clear *how you* have arrived at your solutions to the tasks set. This is what we need evidence for in your video. All credit is for 'show your working' !

The videos must be uploaded to **YouTube**, and we recommend you set these as *Unlisted* with *Comments disabled*. **Your entry will comprise a YouTube link.** To produce the screencast, we recommend the Google Chrome add-on **Screencastify.**

You can enter the challenge **individually** or in **pairs**. If you opt for the latter, *both* of you must make equal contributions to the screencast.

Gold, **Silver** or **Bronze** e-certificates will be emailed to each complete entry, and the **top five** Golds will be invited to present their work at a special ceremony. You should receive a result by December 2024. Note no additional feedback will be provided, and the decision of the judges is final.

Bronze: Initial spreadsheet-based challenge elements attempted, some basic coding.

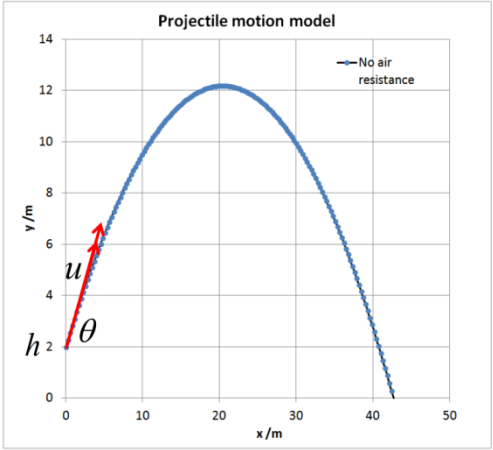
Silver: All the spreadsheet-based elements completed, and a commendable attempt at the programming-based elements. Most tasks completed to a reasonable standard.

Gold: All tasks completed to a high standard, with possible extension work such as the construction of apps (i.e. programs with graphical user interfaces), significant development of the models, attempt at extension work, short research papers etc.

*MATLAB or Python is recommended, although any system that can easily execute code in loops and plot graphs will do. e.g. Octave, Java, Javascript, C#, C++, Mathematica... Use what you can access and feel comfortable with. These Programming resources might be a helpful start.

INSTRUCTIONS *** First download the Challenge Presentation from the BPhO website ***

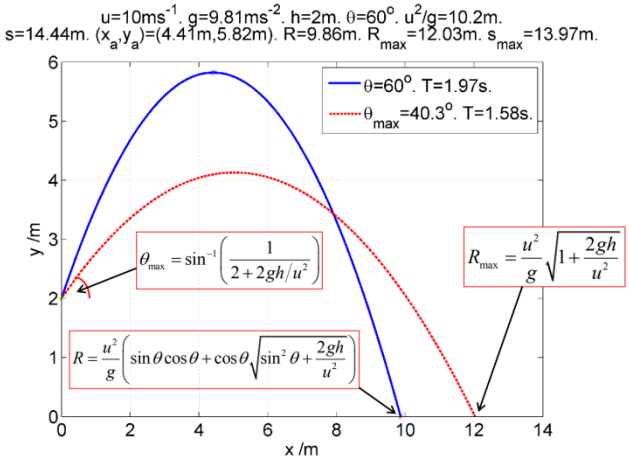
Summary of tasks (each will have Bronze, Silver and Gold aspects - although each task is more involved than the previous). All mathematical details for tasks are provided in the Challenge Presentation pack. All the tasks can be completed by coding up the formulae for the y vs x trajectory, maximum range, apogee etc that are mentioned in the pack. **Although we would certainly encourage all students to work through the derivations (which are provided), and indeed a short paper incorporating all the projectile motion derivations is a recommended extension activity.**



TASK 1: Create a simple model of drag-free projectile motion in a spreadsheet or via a programming language. Inputs are: launch angle from horizontal θ, strength of gravity g, launch speed u and launch height h. Use a fixed increment of time Δt. The graph must automatically update when inputs are changed.

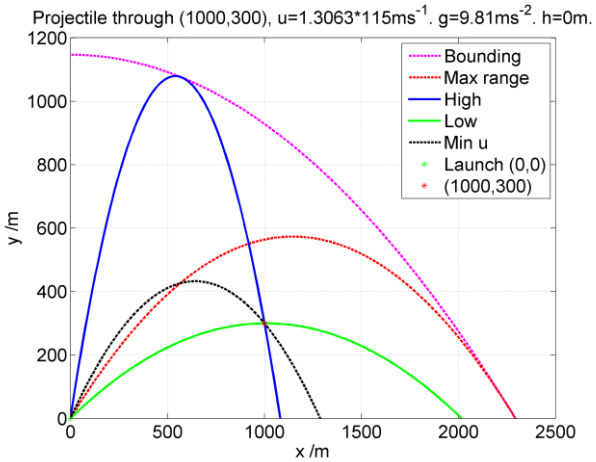
TASK 2: Create a more sophisticated exact ('analytic') model using $y(x)$ equations for the projectile trajectory. In this case define a equally spaced array of x coordinate values between 0 and the maximum horizontal range R. Plot the trajectory and the apogee.

TASK 3: Create a new projectile model which is based upon calculating trajectories that are launched from $(0,0)$ and pass through a fixed position (X,Y). Calculate the *minimum launch speed* to achieve this, and hence determine 'low ball' and 'high ball' trajectories.

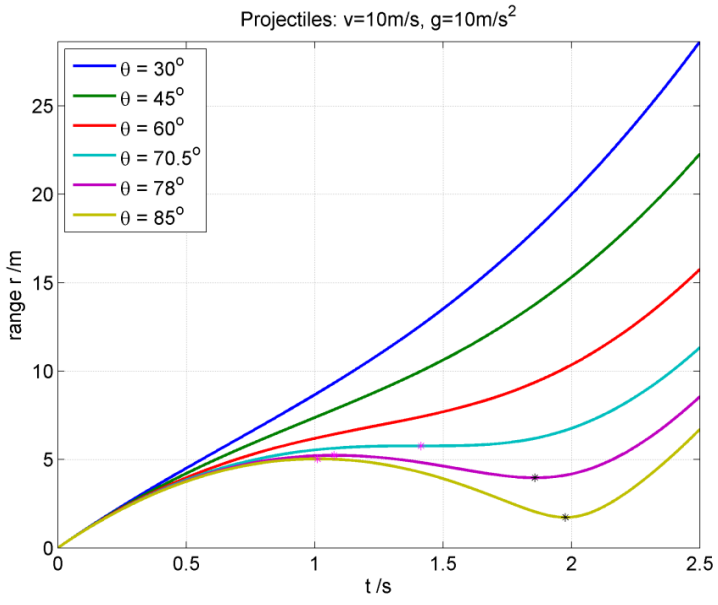


TASK 4: Create a new projectile model which compares a trajectory to the trajectory which *maximizes horizontal range* given the same launch height and launch speed. Inputs are u,h,g and θ. For the *maximum range trajectory* you need to calculate the optimum angle. For $h > 0$ note this is *not* 45° ...

TASK 5: Update your projectile model of a trajectory which passes through (X,Y) with the *bounding parabola*, in addition to minimum speed, max range and high and low ball curves. The bounding parabola marks the region where possible (X,Y) coordinates could be reached given u,h,g inputs.



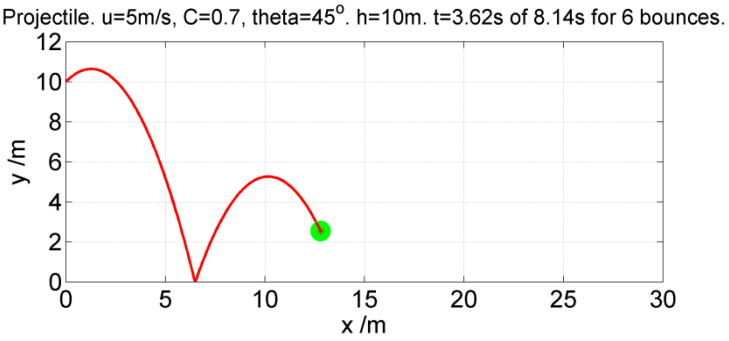
TASK 6: Now update your projectile model with a calculation of the *distance travelled* by the projectile i.e. the length of the inverted parabolic arc. This can be computed exactly!



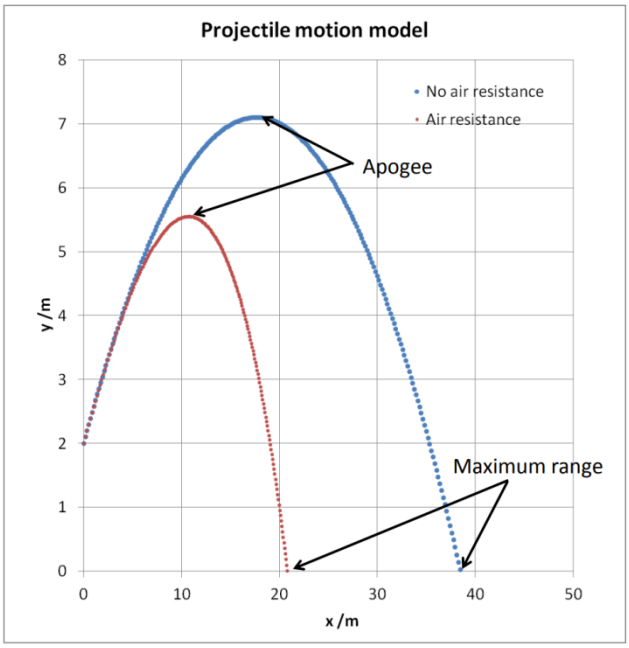
TASK 7:

A curious fact is that the range r of a projectile from the launch point, plotted against time t can, for launch angles greater than about 70.5°, actually pass through a *local maximum and then a minimum,* before increasing with increasing gradient. Use the derivations to recreate the graphs of r vs t. Work out the times, x, y, and r values for these maxima and minima and plot these via a suitable marker.

TASK 8: Use a numerical method assuming constant acceleration motion between small, discrete timesteps (e.g. the 'Verlet' method) to compute a projectile trajectory which includes the possibility of a *bounce*. Define the *coefficient of restitution* C to be the vertical speed of separation divided by the vertical speed of approach. Assume a constant horizontal velocity, and stop the simulation after N bounces. *Extension:* Modify your code to *animate* the trajectory, and ideally, create a video file for efficient future playback.

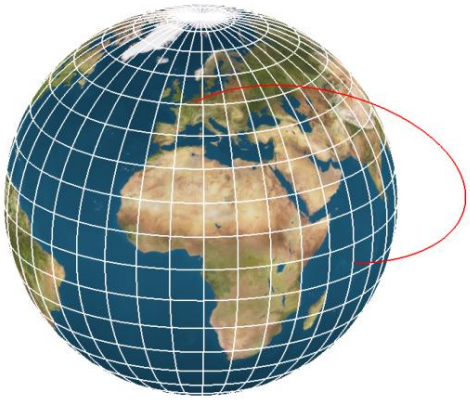


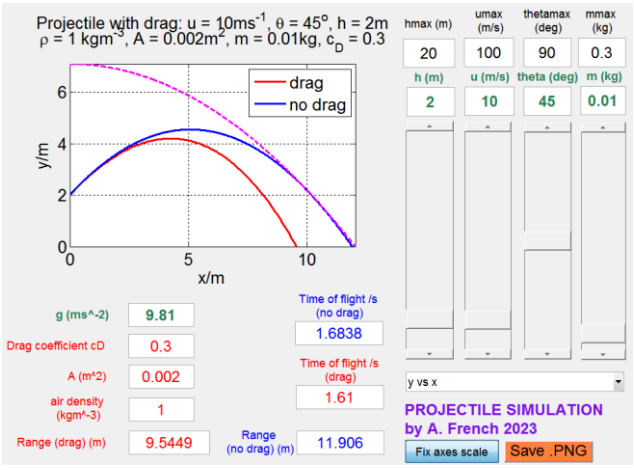
TASK 9: Write a new projectile model which compares a drag-free model (use what you have already done in previous challenges) with a model incorporating the effect of air resistance. Use a *Verlet* method to solve the air-resistance case with a v^2 drag dependence. It is possible to solve motion under drag which varies with the square of velocity analytically in 1D (see here) but in 2D projectile motion drag always opposes the velocity vector, which makes the maths much harder. So write a numerical recipe instead.



POSSIBLE EXTENSION OPPORTUNITIES:

- Consider projectile motion in an atmosphere with a model of air density that diminishes with altitude. See the **2022 BPhO Computational Challenge ('A Standard Atmosphere')** for details.
- Consider projectile motion launched from a spherical planet, which rotates about a fixed axis. Work out the latitude and longitude where the projectile lands, and animate the motion. Texture-map a planet surface e.g. Earth, Mars, the Moon....
- Write a graphical user interface (GUI) for the projectile model and encode this as an 'app'. Coding up an iOS/Android smartphone app will particularly impress the judges.





- Write up your model as a short paper. (Aim for about 10 sides of A4, two columns).

AF 14/7/2023