Numerical Modelling of a Stratospheric Balloon

Alex Howells and Callum Geissmar and Hariram Gnanachandran

ABSTRACT

This report presents a comprehensive analysis of the trajectory of a high-altitude stratospheric weather balloon, completing both 2D and 3D analysis using numerical modelling techniques. The trajectory modelling is based on the solution of a fully derived coupled Ordinary Differential Equation (ODE) system in conjunction with a wind forecast in 2D and 3D. The numerical solutions involved addressing initial value problems through the application of the Runge-Kutta 4 method and boundary value problems using the secant method, with both methods implemented and modelled using MATLAB. Furthermore, the development of a Graphical User Interface as well as a Balloon popping function was developed to create a more realistic model. Despite the accurate modelling of the balloon, the model relies on major fundamental assumptions which reduce the realistic application of the model.

Keywords: Numerical Modelling, Ordinary Differential Equations, RK4 Method, Secant Method

1. INTRODUCTION

Stratospheric balloons are tools for scientific exploration and experimentation in the Earth's upper atmosphere. These balloons reach altitudes far beyond conventional aircraft and serve as highly economical vessels for a wide range of applications. They contribute to the observation of the earth as well as facilitate communication networks. To control their trajectory, their vertical and horizontal movement must be described. The balloon's vertical movement can be effectively managed by manipulating buoyancy and weight forces, achieved by the venting of helium or dropping of sand, yet the control of their horizontal motion relies on the movement of the wind. The initial conditions of the balloon include a constant mass of 15kg, 10kg of sand and 1000 moles of helium. In this report, the numerical modelling and control of a stratospheric balloon's trajectory is achieved using these key assumptions:

- Masses were considered as a point mass.
- The balloon was assumed to be in equilibrium in the horizontal plane as the balloon moves within the same frame of reference as the wind and so horizontal drag forces were also taken to be negligible.
- Throughout one time step the horizontal velocity of the balloon does not change.
- The balloon is considered able to vent helium or drop sand instantly, reducing its weight.
- The balloon was assumed to be spherical.
- Assume helium in the balloon is an ideal gas.

2. DERIVATION OF ODE'S

To model the trajectory of the balloon, the forces acting on the balloon were initially determined. This was achieved by drawing a free-body diagram of the balloon at its initial position, shown in Figure 1:

By equating the forces acting in the y plane and using Newton's Second Law, $F = m * a$ [1], the resultant force equation was derived to be:

$$F_t - Mg - F_yD = Ma \tag{1}$$

F_t = Thrust (N), M =Total Mass (kg), g = Gravitational Constant [9.81], F_yD= Drag Force (N)

The Drag Force was defined as: [2]

$$F_yD = \frac{1}{2}C_d V_y^2 A \rho_{atm} \tag{2}$$

C_d = Coefficient of Drag, V_y = Velocity (ms^{-1}), A = Cross Sectional Area (m^2), ρ_{atm} = Density of Atmosphere (kg/m^3)

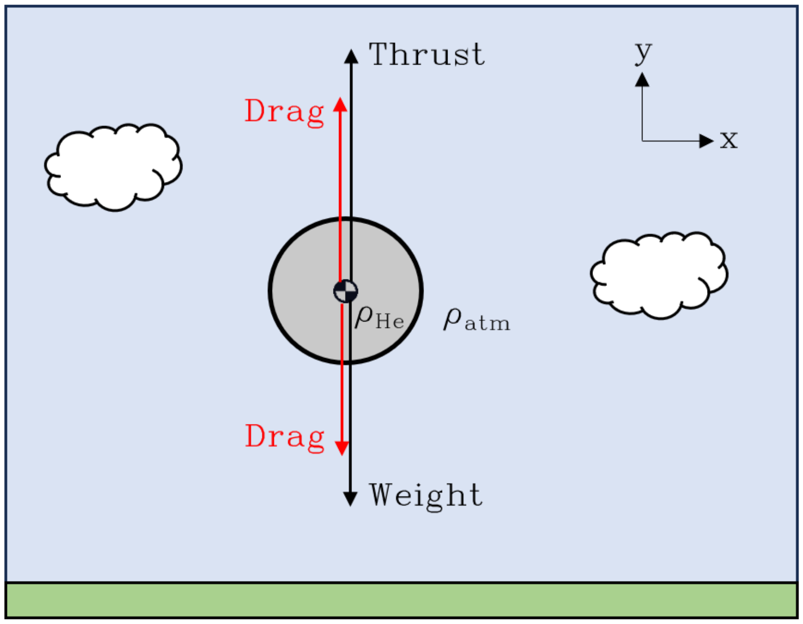


Figure 1. Free Body Diagram of the Forces Acting on the Balloon

To allow for changes in the drag force, due to the change in atmospheric conditions affecting ρ_{atm} and A of the balloon, using the ideal gas law $pV = nRT$ [3], the drag force was rearranged to be:

$$F_yD = \frac{\pi\rho_{atm}V_y^2}{8}(\frac{3nRT}{4\pi p})^{\frac{2}{3}} \tag{3}$$

n = Number of Moles, R = Gas Constant, T = Temperature (K), p_{atm}= Atmospheric Pressure (Pa)

Thrust was defined as: [4]

$$F_B = \rho_{atm}gV \tag{4}$$

V = Volume of Balloon Similarly, the equation was transformed to account for changing atmospheric conditions which would affect ρ_{atm} and V, and therefore thrust becomes:

$$F_B = \frac{\rho_{atm}gnRT}{p} \tag{5}$$

The resulting equation which models the forces acting on the balloon in the y plane is:

$$\frac{\rho_{atm}gnRT}{p} - (15 + M_{He} + M_S)g - (\frac{dx}{dt})^2 \frac{\pi\rho_{atm}V_y^2}{8}(\frac{3nRT}{4\pi p})^{\frac{2}{3}} = (15 + M_{He} + M_S)(\frac{d^2x}{dt^2}) \tag{6}$$

To model this second-order differential equation (ODE), it was deconstructed into two coupled first-order ODE equations. To decompose the second-order ODE to first order required a state space representation of the system. The state variables for this system are given as:

$$z_1 = x$$

$$z_2 = \dot{x}$$

The derivatives are:

$$z_1^{'} = \dot{x} = z_2$$

$$z_2^{'} = \ddot{x}$$

It follows that:

$$dz_1 = z_2 = \dot{x} \tag{7}$$

$$dz_2 = z_2' = \frac{\rho_{atm} gnRT}{p} - (15 + M_{He} + M_S)g - (\frac{dx}{dt})^2 \frac{\pi \rho_{atm} V_y^2}{8} (\frac{3nRT}{4\pi p})^{\frac{2}{3}} = (15 + M_{He} + M_S)(\frac{d^2x}{dt^2}) \tag{8}$$

These coupled first-order ODEs were implemented into MATLAB where they served as the foundation equation for the modelling of the balloon's trajectory.

3. METHOD - BASIC TASK

Given the coupled first-order ODE equations (7) and (8), the Runge-Kutta 4 (RK4) method aims to find a numerical approximate solution to x(t) (altitude of the balloon) at different points in time. In order to fully describe dz_1 and dz_2, the variables within the ODE must be assigned values. Temperature, pressure and density variables were dependent on the altitude of the balloon and so were updated after each step. This was achieved using the *standardAtmosphere* function, providing the values of temperature pressure and density for the current altitude. Similarly, helium mass was dependent on the number of moles, n, currently in the balloon, which changes with the amount of helium vented as well as the mass of sand which changes with the amount of sand dropped. Defining dz_1 and dz_2 into the *stateDeriv* function as well as the numerical values of each variable within the equation (8) allowed for the current state vector to be fully described. This current state vector is passed into the *stepRungeKutta* function.

The RK4 method involves updating the dependent variable x at each time step based on four weighted estimates of the state derivative, dz.(dz = [dz_1; dz_2]) [5]. These four estimate derivatives within the time step dt, are given as:

$$A = dt * f(t_n, X_n)$$

$$B = dt * f(t_n + \frac{dt}{2}, X_n + \frac{A}{2})$$

$$C = dt * f(t_n + \frac{dt}{2}, X_n + \frac{B}{2})$$

$$D = dt * f(t_n + dt, X_n + C)$$

Then combine these derivative approximations using a weighted average to calculate the new state derivative Z_{n+1} at the next time step using the RK4 update rule [5]:

$$X_{n+1} = X_n + \frac{1}{6}(A + 2B + 2C + D) \tag{9}$$

This is completed inside the *stepRungeKutta* function producing the next state derivative (where the new altitude and velocity is contained within that vector). To complete a full solution to the ODE, this process was iterated at each time step. This was achieved using the *ivpsolver* function.

As the horizontal displacement of the balloon is not modelled by an ODE; due to its dependence on the wind speed, it was modelled by updating the horizontal displacement after each time step. The horizontal speed of the balloon is constant relative to the wind as no resultant forces are acting and therefore by using the horizontal velocity during each time step, the horizontal distance travelled can be calculated for each step. This was achieved within the *finalHoriDist* function using the equation $distance = speed * time$ [6]. The speed at each altitude is extracted from the *exampleforecast.csv* file using the *evaluatewindforecast1D* function.

To model the balloon to a specific position the trajectory was altered by venting helium and dropping sand. Calculating the specific amount to unload became a boundary value problem (BVP), as the initial and final position of the balloon was known, however, the initial and final masses (number of moles and mass of sand) were unknown. To solve the boundary value problem, the secant method was used, which is an iterative approach that aims to minimize the errors generated by computed values at the boundaries of the system. [7], and is given by the formula: [7]

$$Z_{n+1} = Z_n - \varepsilon_n (\frac{Z_n - Z_{n-1}}{\varepsilon_n - \varepsilon_{n-1}}) \tag{10}$$

$Z_n + 1$ = New Mole Estimate, Z_n, Z_{n-1} = Initial Mole Estimates, $\varepsilon_n, \varepsilon_{n-1}$ = Errors of Corresponding Mole Estimate

Firstly, the number of moles required to vent was computed as the basic task required for the balloon to descend to a set altitude and horizontal displacement. To calculate the new required number of moles a relationship between the number of moles and horizontal distance reached with that number of moles was established. To generate this relationship for the desired balloon trajectory, the function *CreateMolesVsHeight* was used to create an array of the number of moles in steps of 20 from 0 to 1000, against the final height reached as well as the time at which it reaches. Then using *findHorDistances* function, these altitudes were given corresponding horizontal distance values for each by evaluating the horizontal distance after each time step and the velocity of the wind at each altitude.

To determine the actual number of moles needed to drop to reach the required altitude the horizontal and vertical distances reached at specific moles and at what time they are reached arrays were passed into the function *detMolesNeeded* to begin the secant root finding method.

To compute the iterative method, given in equation (10), initial estimates for the number of moles needed to complete the desired manoeuvre were created. These initial estimates (Z_n, Z_{n-1}) were generated by finding the corresponding number of moles which relates to the distances one step greater and lower than the required horizontal distance of the manoeuvre. Taking these initial mole estimates, the initial errors in these estimates, E0 and E1 were determined as the difference between the required horizontal distance, and the horizontal distances the initial estimate's amount of moles would reach.

The initial mole estimates and their corresponding errors were then used in equation (10) to calculate the next number of moles Z_{n+1} that minimizes the new error, assuming a linear correlation between the estimates and the errors[7]. The new mole estimate Z_{n+1} was then passed into the *ivpsolver* function to calculate what altitude and therefore what distance the new estimate would reach which was used to measure the new error between the required horizontal distance and the horizontal distance the new estimates amount of moles will reach.

This process was iterated until the magnitude of the error was below 0.5. The corresponding number of moles to meet this criterion became the approximate solution to the BVP and therefore the required number of moles to complete the given trajectory, a variable called *secantMoles*.

This method was repeated to calculate the mass of sand required for an upward movement of the balloon, where at each step the estimates and calculated number of moles were substituted for a variable of sand mass.

The final calculated number of moles and sand required to complete the given manoeuvre was then inputted into the *ivpsolver* function to find the final vertical altitudes through the RK4 method described above. The final vertical altitude was then passed into *finalhordistances* function giving the final horizontal distance and the final array containing the horizontal and vertical positions for the trajectory was created within the *createFinArr* function. The final trajectory of the balloon was then produced given the required manoeuvre parameters and shown in Figure 2.

4. METHOD - EXTENSION TASKS

3D Model

Modelling the balloon in three dimensions, added an additional direction in which the wind influences the balloon, and therefore the original ODE remained unchanged as the wind has no resultant force on the balloon. This new wind forecast was given by a new data set, *exampleForecast3D*. Likewise, the RK4 method derived in the basic task as well as its included components such as *stateDeriv* were unchanged. To model the balloon with a new wind dimension the horizontal distance from the initial position after each time step differed from the basic task as now the velocity at each axial location is a 2D vector.

The same array containing the number of moles vs the altitude reached by that number of moles from *createMolesVSHeight* function was therefore passed into the function *findHorDistancesM* to convert these altitudes into north and east horizontal directions using the corresponding directional velocity of the balloon at each time step. These velocity vectors are generated inside the *createvelocityvector* function. This function retrieves the velocities in all directions for each altitude from the 3D wind forecast and generates the corresponding direction velocity vector using *CreateVector* function and its axial position.

To complete the BVP in a 3D field, the relationship between the number of moles required and both the northing and easting positions were considered. This was completed within the *detMolesNeeded* function and the same secant method as outlined in the basic task was completed. However, the estimates of moles required were analyzed in separate northing and easting directions. In this way two sets of errors were iterated, one set for errors in the north direction and one set for errors in the east direction. The resultant value of *secantMoles* is determined when the balloon reaches the required northing or easting position while venting the least amount of helium as possible.

In addition to the modelling of helium dropped, the mass of sand underwent secant iteration to allow for upward trajectories of the balloon. This was achieved using the same method as the helium mole analysis.

By processing the new number of moles and mass of sand through *ivpsolver* and *findHoriPos3D* to create the horizontal and vertical position arrays, the optimal trajectory of the balloon was established. To ensure the balloon was not modelled outside the 3D forecasted field, the final 3D position arrays were passed through *createFinArrM* for moles and createFinArrS for sand, where the vertical altitude limits were imposed (top and bottom of forecast matrix). The final 3D trajectory of the balloon was then produced given the required manoeuvre parameters using the *plot3* function and is shown in Figure 4.

Graphical User Interface for 3D Model

To take advantage of the modularity of the designed code and allow for easy changes in the input parameters of the 3D balloon model, a Graphical User Interface (GUI) was developed within MATLAB app designer. Firstly, using the app layout tool, input boxes as well as the format of the app on the base layer were generated. This allowed for the input of parameters of initial and final locations in all axes, as well as initial values of helium moles and mass of sand, which can be seen in Figures (4) and (5).

These input values are assigned to their corresponding variable and are passed into the main script to give the script the values required to process the above 3D modelling of the balloon's trajectory. This allows the main 3D modelling script to calculate all the values required to plot the trajectory graph.

The arrays generated by the main script containing final positions, moles of helium required, mass of sand and time taken are saved and loaded into the app. After the data has been calculated, and the plot process has been activated, using MATLAB *plot3* function, the 3D trajectory of the balloon is plotted and displayed within the GUI, likewise, time taken for procedure, final helium moles and mass of sand required is also displayed.

Balloon Popping

Although the balloon model is limited by the maximum altitude of the 3D forecast, a function to evaluate whether the balloon would pop, *checkpop*, was implemented. The function evaluates the new radius of the balloon, R_1, at each altitude using:

$$R_1 = \sqrt[3]{\frac{3nRT}{4\pi p}} \tag{11}$$

This equation is derived from equating the volume of a sphere: $\frac{4}{3}\pi r^3$ [8], as well as the ideal gas law equation to calculate the radius of the sphere. The pressure and temperature variables within the equation are updated at each altitude using values given from the *standardAtmosphere* function. The calculated value R_1 as well as an input of the initial radius of the balloon R0 into the GUI, was then used to calculate the strain of the balloon using the formula $\frac{R_1 - R_0}{R_0} * 100$ [9] within the graphical user interface app. If the calculated strain at a given altitude exceeds the maximum percentage strain, determined by a user input, the balloon will pop and is indicated on the GUI as well as the altitude at which the balloon pops.

5. RESULTS

The results for the basic task, in a 2D wind forecast are shown in Figures (2) and (3) below:

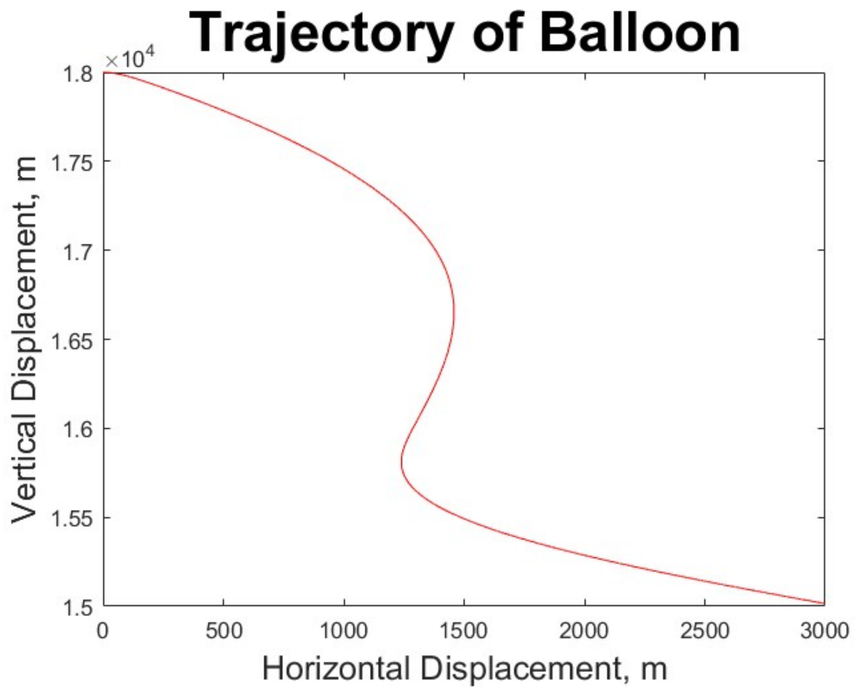


Figure 2. Basic Task - Trajectory of Balloon from initial position (0,18km) to (3km,15km)

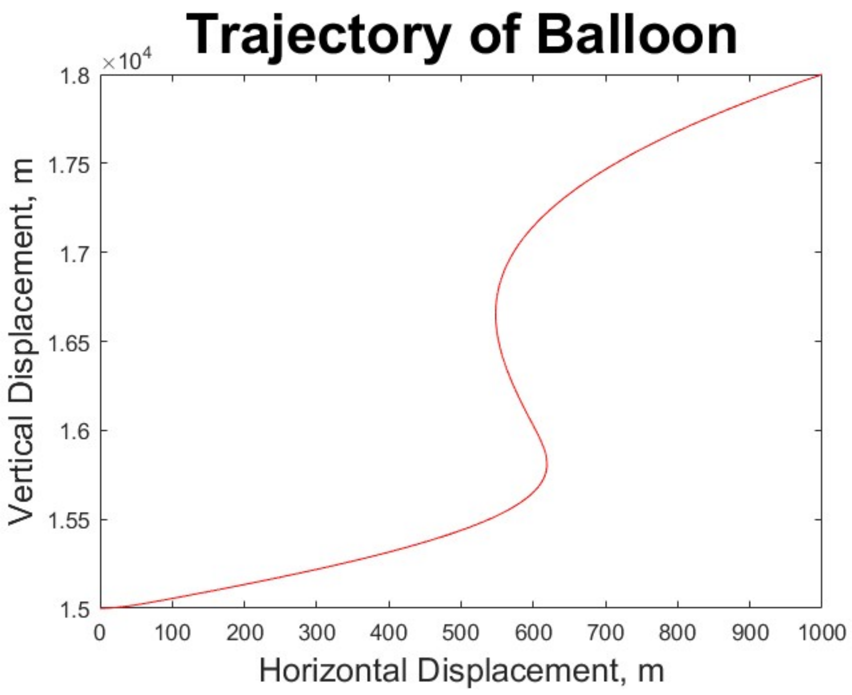


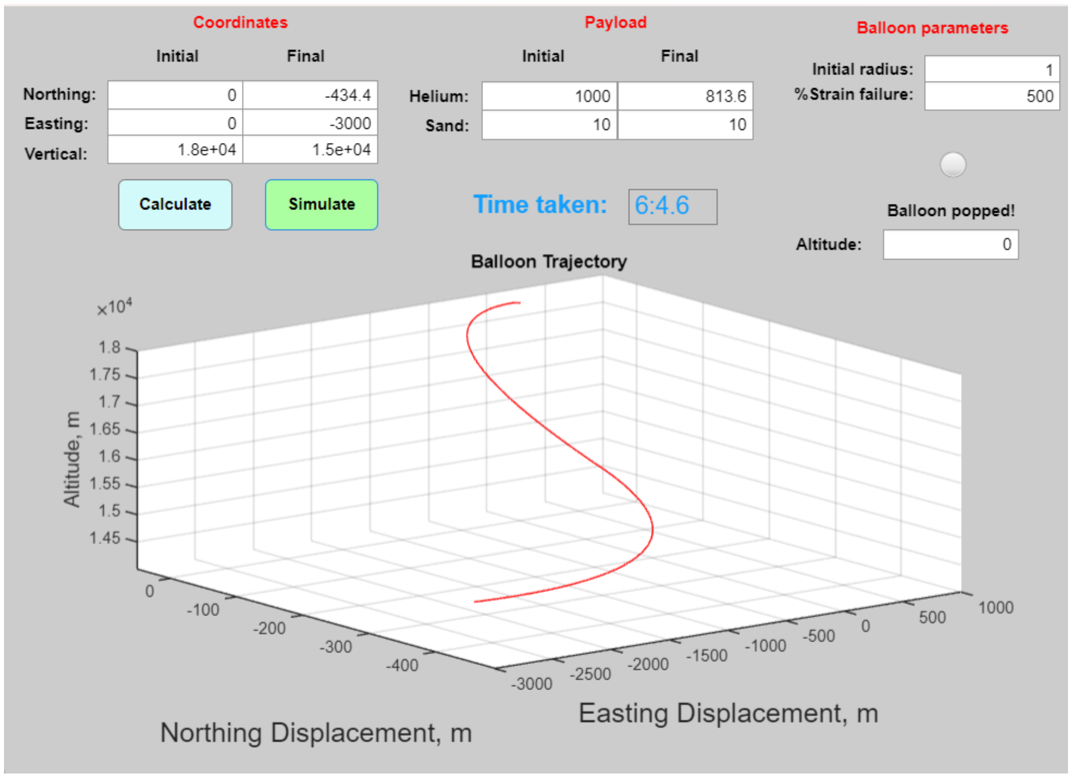
Figure 3. Trajectory of Balloon from initial position (0,15km) to (1km,18km)

The detailed results of the two 2D models are given in Table (1) below:

	Figure 2	Figure 3
Desired Input Positions (km)	(0.000, 18.000), (3.000, 15.000)	(0.000, 15.000), (1.000, 18.000)
Actual Final Position (km)	(3.000, 15.015)	(0.999, 15.000)
Duration of Maneuver		
Moles Vented	17.9226	0
Sand Dropped (kg)	0	4.7104

Table 1. Detailed Results of Trajectories displayed in Figures (2) and (3)

The results for the extension tasks, including 3D wind forecast, graphical user interface, and balloon popping analysis shown in Figures (5) and (6) below:



	Initial	Final
Northing:	0	-434.4
Easting:	0	-3000
Vertical:	1.8e+04	1.5e+04

	Initial	Final
Helium:	1000	813.6
Sand:	10	10

Initial radius: 1

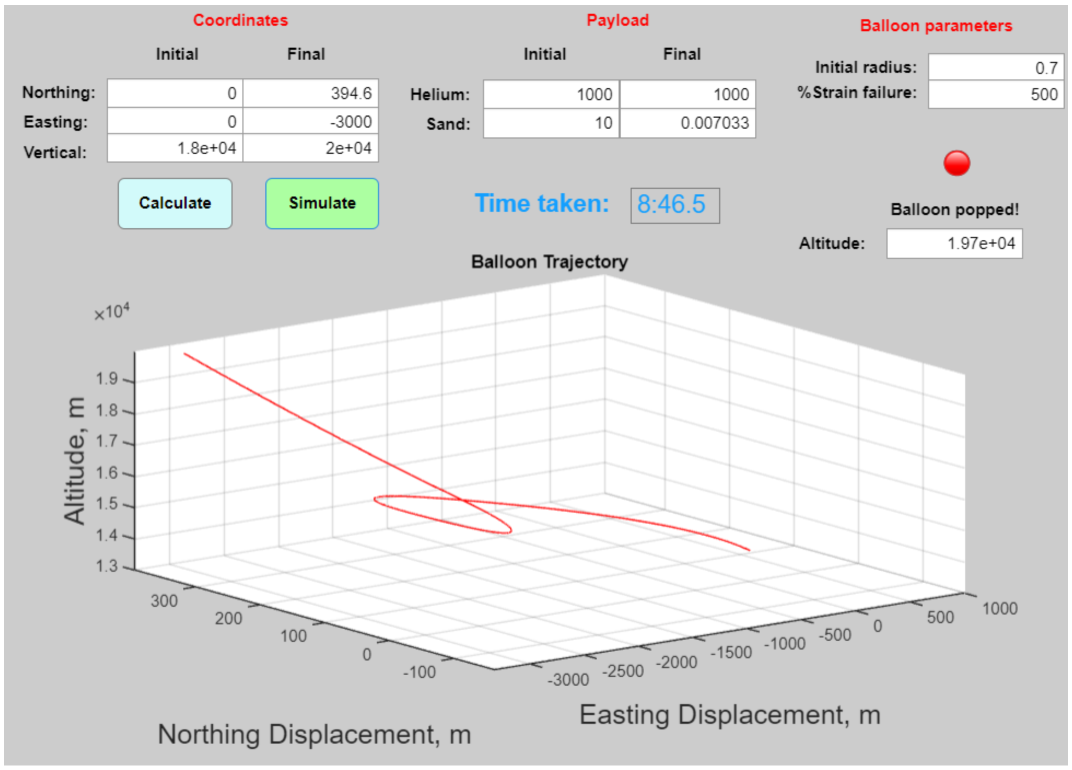
%Strain failure: 500

Time taken: 6:4.6

Balloon popped!

Altitude: 0

Figure 4. 3D Trajectory of Balloon



	Initial	Final
Northing:	0	394.6
Easting:	0	-3000
Vertical:	1.8e+04	2e+04

	Initial	Final
Helium:	1000	1000
Sand:	10	0.007033

Initial radius: 0.7

%Strain failure: 500

Time taken: 8:46.5

Balloon popped!

Altitude: 1.97e+04

Figure 5. 3D Trajectory of Balloon with Pop Criterion Met

6. DISCUSSION

Firstly, the trajectory of the basic task, as shown in Figure 2, accurately completes the given manoeuvre to within 15m of the final desired position, an error of 0.1%. This small discrepancy could be attributed to the accuracy of the RK4 method and secant methods used to derive the trajectory. The RK4 method provides a balance of accuracy and computational power required, however, to process a more accurate solution, a higher-order method such as Runge-Kutta 5 may be considered. In addition, an adaptive method such as the Runge-Kutta-Fehlberg Method (RKF) may also be considered as an alternative to RK4. The RKF method 'adjusts the step size using an estimate of the error at each step and a user-defined limit for the tolerable error.'[10]. In this way, not only would the step size adjust to produce a more accurate solution but would contribute to reducing the run time of the code making it more efficient. The secant method is an accurate root-finding method, however, to further improve the accuracy of its use, reducing the acceptable error below 0.5 between an estimate and its accepted value would increase the accuracy of the model.

The trajectory and implementation of the extension tasks suffer the same limitations as the basic task considering the accuracy of modelling techniques used. The 3D model was limited by the size of the wind forecast used, shown in Appendix A, which limited the possible flight trajectories which could be modelled. To make a more comprehensive model, a larger forecast for a larger forecast area would be used.

In addition to the limitations of the modelling techniques used, the initial assumptions made affect the accuracy of the model. For example, the imperfect assumption of the spherical balloon will affect the drag force as well as the radius of the balloon. Furthermore, simplifying assumptions such as not considering the effects such as the diurnal cycle, limit the real-world accuracy of the model.

7. CONCLUSION

In conclusion, the trajectory of a high-altitude stratospheric weather balloon was modelled in 2D and 3D using numerical modelling solutions to a fully derived coupled ODE given in Equations (7) and (8). These techniques included the solving of initial value problems using the Runge-Kutta 4 method, and the solving of boundary value problems using the secant method, both of which were implemented in MATLAB. These methods resulted in accurate and precise approximations to the ODE solution and the computed results are presented in Figures 2,3,4,5. In addition to 2D and 3D models, a graphical user interface was designed to allow for variable input parameters to allow for a range of balloon trajectories in 3D, as well as a balloon popping model which returned at what altitude the modelled balloon would pop with the input balloon radius. Although the modelled balloon is accurate, the modelling assumptions and techniques used could be improved by using more accurate ODE-solving methods such as the RKF method. Furthermore, the major assumptions made to model the balloon could have been improved to create a more realistic model. Overall the model is successful and accurate with the assumptions made.

8. REFERENCES

[1] Newton's laws of Motion [Internet]. NASA; 2023 [cited 2023 Dec 7]. Available from: https://www1.grc.nasa.gov/beginners-guide-to-aeronautics/newtons-laws-of-motion/
[2] Drag equation [Internet]. NASA; 2023 [cited 2023 Dec 7]. Available from: https://www1.grc.nasa.gov/beginners-guide-to-aeronautics/drag-equation-2/
[3] Ideal Gas Behaviour [Internet]. Kevin M. Tenny; Jeffrey S. Cooper. [cited 2023 Dec 7]. Available from: https://www.ncbi.nlm.nih.gov/books/NBK441936/
[4] Archimedes principle - explanation, Archimedes Law examples [Internet]. BYJU'S; 2023 [cited 2023 Dec 7]. Available from: https://byjus.com/physics/archimedes-principle/
[5] Dr Alan Hunter. Lecture 2 – Runge-Kutta Method. [Internet] Moodle 2023.
[6] Dr Eleanor Lingham [Internet]. [cited 2023 Dec 7]. Available from: https://www.mathcentre.ac.uk/resources/uploaded/22-speed-and-distance.pdf
[7] Dr Alan Hunter. Lecture 5 – The Shooting Method. [Internet] Moodle 2023.
[8] Admin. Volume of sphere - definition, formula, derivation and examples [Internet]. BYJU'S; 2022 [cited 2023 Dec 7]. Available from: https://byjus.com/maths/volume-of-sphere/
[9] Mechanics of materials: Strain " mechanics of Slender Structures: Boston University [Internet]. [cited 2023 Dec 7]. Available from: https://www.bu.edu/moss/mechanics-of-materials-strain/
[10] Dr Alan Hunter. Lecture 11 – Improving Accuracy and Efficiency. [Internet] Moodle 2023.
[11] Dr Alan Hunter. Image of example 3-D wind forecast [Internet] Moodle 2023.

APPENDIX A

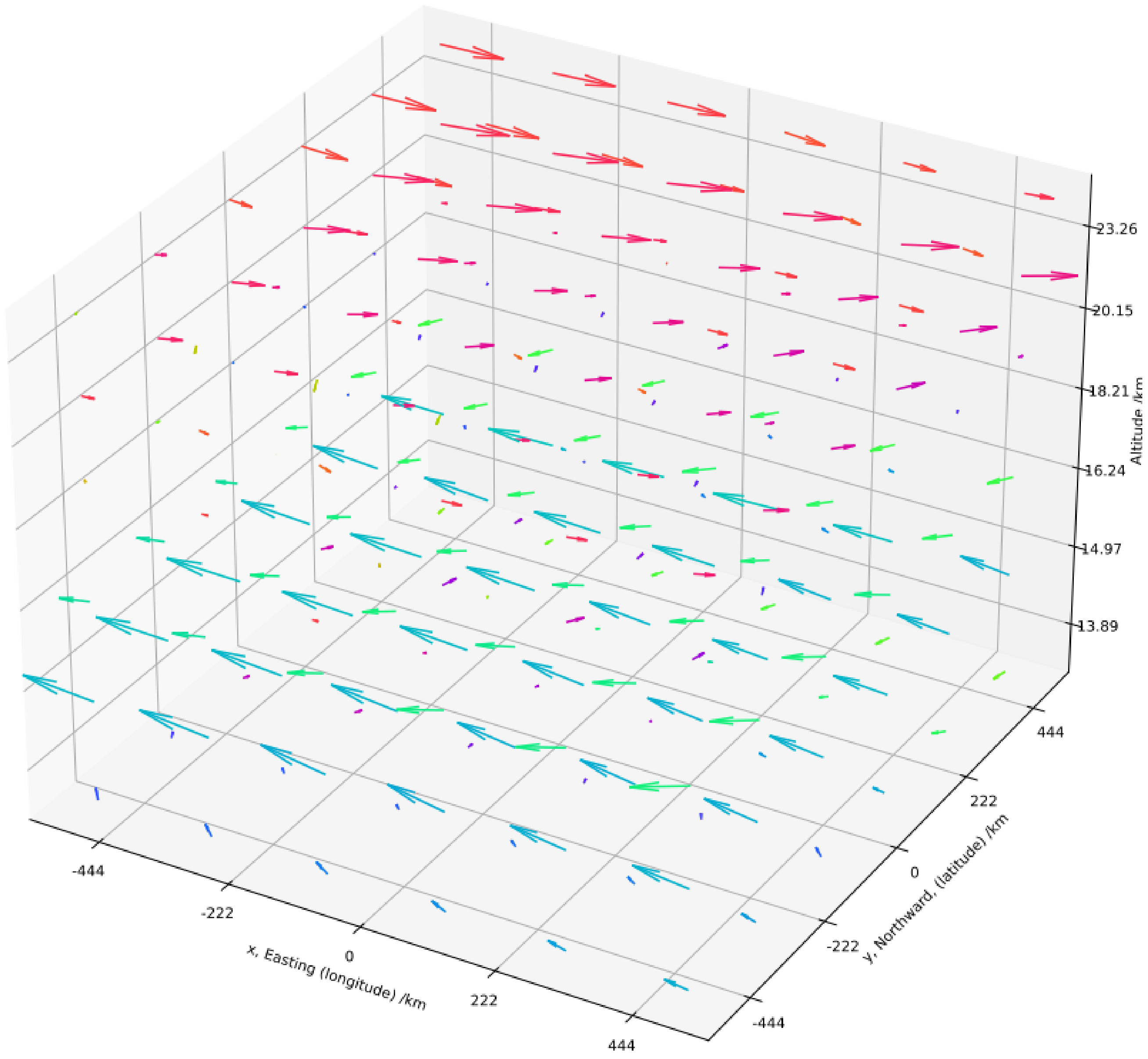


Figure 6. 3D Wind Forecast [11]

Figure 7 shows the full 3D wind forecast data used. It is clear that there are maximum and minimum limitations on where the balloon can move both vertically and horizontally. In this way, the model is limited by the forecast data used.

APPENDIX B

Basic Task Code:

APPENDIX C

Extension Task Code: