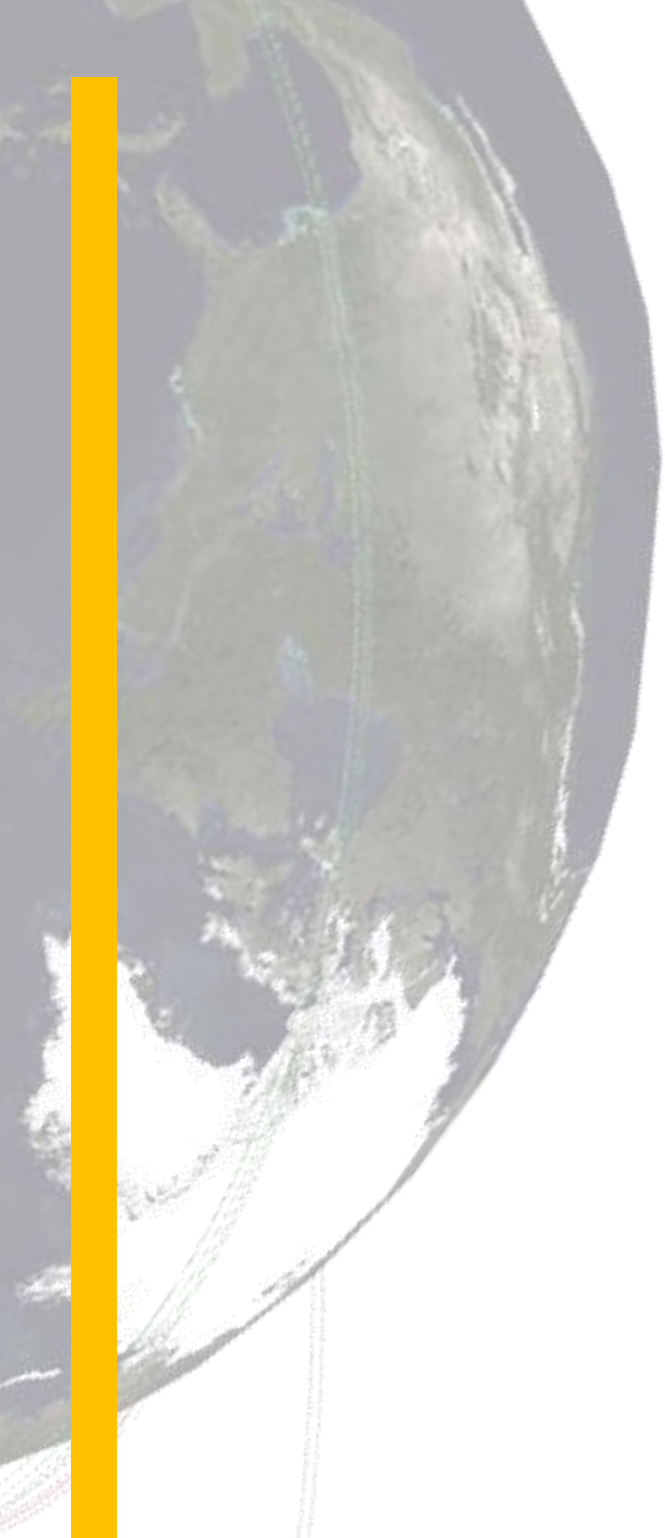


DECEMBER 7, 2022

NUMERICAL MODELLING OF A KINETIC SPACE LAUNCH

ASSIGNMENT 2

JOSHUA KEENAN, CRIS DIGNADICE, JORGE PALOP SUAREZ
UNIVERSITY OF BATH

SUMMARY

A model for the launch of a ballistic projectile was made. This model aimed to determine the angle required for the projectile to reach a max altitude of $190\ km$ and then return to the earth, deploying a parachute at an altitude of $10\ km$. Launching at a velocity of 2500 m/s the angle required to reach the required altitude was calculated to be of $54.18°$. At this angle and conditions, the distance travelled over the ground, time of flight and impact velocity were calculated to be of $547.26\ km, 579.8\ s$ and $41.57\ ms^{-1}$ respectively. In addition to the basic task, several additional components were added to the code to increase the accuracy of the model. This included introducing a third dimension and modelling the launch from a real location on earth, considering the effect of the Earth's rotation at different locations and directions and wind velocity. With this, the location for impact could be found and pinpointed on a map.

A second stage was introduced and modelled reaching orbital velocity and performing a Hohmann transfer manoeuvre. Entering a circular orbit directly from apogee required a velocity of $7790ms^{-1}$ resulting in a 'Delta V' of $6475.6ms^{-1}$. This was achieved through a constant burn time of 98.4135 seconds. During the acceleration process, $19682.7kg$ of fuel was used reducing the projectile mass to $5317.3kg$. At the end of the procedure, a slight overshoot of velocity was detected. As a result, the thruster was activated in the opposite direction to correct the trajectory to conclude the manoeuvre. An error of 0.0304% was present when comparing the theoretical orbital velocity to the one achieved by the code. In the scenario a manoeuvre isn't possible with the remaining resources, an error message will be displayed.

A realistic representation of the trajectory was then plotted on top of the 3D earth model along with the ISS trajectory for reference. A line originating from the origin is connected to the instantaneous position of the projectile and ISS allowing for easy tracking over the surface of the ellipsoid. Push buttons were also provided to allow the user control over the rotational speed of the earth.

List of symbols

$\rho = Density$

$C_d = Coefficent\ of\ Drag$

$A = Projected\ Area\ of\ projectile$

$\theta_x = Angle\ of\ Launch$

$m = Mass\ of\ Projectile$

$M = Mass\ of\ earth$

$\ddot{x} = Net\ Acceleration\ of\ Projectile$

$\dot{x} = Net\ Velocity\ of\ projectile$

$G = Gravitational\ Constant\ (6.67 \times 10^{-11} m^3 kg^{-1} s^{-2})$

$R_e = Radius\ of\ Earth$

$A_{perpendicular} = Area\ perpendicular\ to\ direction\ of\ motion$

1. Introduction

Launching of typical rockets requires the burning of a large amount of fuel to reach orbit, to the extent that over 90% of the weight of a rocket is fuel **(Holt & Monk, 2009)**. By imparting a large majority of the energy onto the rocket before launch instead of carrying fuel to generate more energy, the energy required to accelerate the increased mass is avoided. This was initially experimented with in the early 1960s under project HARP (High Altitude Research Project) run by the US and Canadian departments of defence. (Petrescu et al., 2017) Project HARP used a large gun to accelerate the object to high velocity to reach space. Following similar principles, SpinLaunch is developing a launch system that spins the projectile to high velocities and precisely releases it to get the projectile into space. Once in space, the second stage boosts into orbit. The decrease in the amount of fuel required for launch and increased reusability results in a cheap, fuel-efficient, and sustainable launch system that could be used in the future to launch satellites into space or resupply space stations, however, due to the high g-forces, it is not currently suitable for human flight.

The modelling of this kind of launch system is valuable as it can be used to ensure the viability of the design, help with calculating fuel required to reach orbits and find out where the first stage is going to land to ensure a safe landing spot and easy to access recovery for reuse. This report aimed to develop an approximated model capable of calculating the projectile trajectory during various initial conditions. The initial objectives were to use the model to calculate the angle of launch required to reach an apogee altitude of $190\,km$ and plot the profile of the trajectory, showing the point of parachute release and impact. The model was then further developed to refine the model and add extra functionality, aiming to increase the model's accuracy.

2. Basic model & Method

2.1. Discussion of physical problem and assumptions

The initial aim of the model is to calculate the angle that allows the projectile to reach precisely $190\,km$ from the earth surface. This was done with the initial conditions of velocity of $2500\,ms^{-1}$ and rocket mass of $2500\,kg$. It was assumed the launch was happening with no wind.

2.2. ODE Derivation

A free body diagram (FBD) for the moving projectile was developed, as shown by Figure 1.

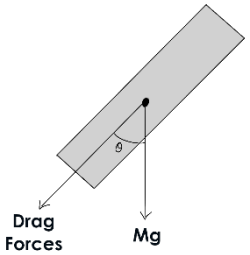


Figure 1: Free Body Diagram of Rocket

It was determined that the projectile only experienced a drag force acting parallel to the projectile's velocity ($Equation\ 1$) and a gravitational force acting vertically down.

$$F = \frac{1}{2}\rho \dot{x}^2 C_D A \tag{1}$$

Using this equation, and the FBD previously developed, two second-order ordinary differential equations IVPs were developed for the projectiles motion in the X and Y axis, as shown by Equation 2 and 3 respectively.

$$\ddot{x}_x = -\frac{\rho C_d A \cdot \cos(\theta)}{2m}\dot{x}^2 \tag{2}$$

$$\ddot{x}_y = -\frac{\rho C_d A \cdot \sin(\theta)}{2m}\dot{x}^2 - \frac{GM}{R^2} \tag{3}$$

The generated second-order ODEs were reduced to two coupled 1st-order ODEs from which the state variables for the projectiles motion in the X and Y direction were developed. Using initial conditions of $\dot{x} = 2500\, ms^{-1}$, $t = 0s$ and a step size of $\Delta t = 0.025s$, the Rung-Kutter 4th order method (Liu et al., 2011) was then implemented to calculate an approximation for the 1st-order state derivatives at a launch angle θ_x. The velocity and displacement of the projectile in the X and Y directions at $t = 0.025s$ were then determined and used to calculate the nominal velocity and angle of the projectile at $t = 0.025s$. These values were then used on the next iteration to calculate the projectile state values at $t_2 = t_1 + \Delta t$. This process was repeated until the projectile vertical displacement decreased to zero, thus calculating the projectile behaviour and trajectory through its flight.

The projectile behaviour and trajectory throughout its flight was therefore calculated, and the value for the maximum altitude was determined. With this value, a shooting method was then used to find the angle to reach precisely $190\, km$. This works by initially shooting the projectile at an angle of 75° then at an angle of 25° and calculating the error in max altitude. These values were then used to calculate an informed guess for the angle using Equation 4 (Edun & Akinlabi, 2021).

$$Guessed\ Angle\ = \theta_{x1} - \left(Error2 \cdot \left(\frac{\theta_{x2} - \theta_{x1}}{Error2 - Error1}\right)\right) \tag{4}$$

From this, the calculations to find max altitude were run again and the values for error and angle were substituted back into Equation 4 and repeated until the error in altitude was less than $2\, m$ as this was deemed to be sufficient to refer to this altitude as precisely $190km$ with an error of 0.0003% $(0.6219\, m)$.

The mass and values for drag changed during the launch, this was done by creating a condition of when the second stage detaches at the apogee the mass is reduced by $20500\, kg$. This drastically increased the effect of drag and wind on the object. Furthermore, once the parachute opened at $10\, km$ the coefficient of drag increased to 1.75 and the projected area increased to $24\, m$. Reducing the velocity of impact down to $41.57\, ms^{-1}$.

2.3. Accounting for curvature of the earth

To account for the curvature of the earth the first step was to calculate the distance from the centre of the earth to the calculated position at each time step. This was done using Pythagoras formula. With the altitude at every point the maximum altitude could be easily found and the loop that crated the path was called to stop at altitude = 0. In addition, the increase in surface displacement due to the curvature of the earth was accounted for by using the cosine rule.

Due to the small distances and angles present some simplifications were made in respect to the curvature of the earth. Gravity was assumed to only act within the Z axis and not always perpendicular to the surface of the earth. To improve this, we could find the angle of the surface of the earth at any given time step and apply trigonometry to find the components of gravity in X, Y and Z. We also assumed that wind only acted in the X and Y plane, wind should act parallel to the surface of the earth and therefore trigonometry should be used again to account for the change in angle of the surface of the earth relative to the launch location.

3. Results

With the conditions of launching at $2500\, ms^{-1}, and\ \Delta t = 0.025s$ the angle of launch θ_x required to reach an apogee of $190\, km$ was found to be $54.18°$ with an error of $62\, cm$ from the intended maximum altitude. This caused a surface displacement of $547.26\, km$, a velocity of impact of $41.57\, ms^{-1}$ and a time of flight of $579.8\, s$. A time step of $\Delta t = 0.025s$ was chosen as it provided an accuracy of $0.02°$ while maintaining a reasonable computational time. The profile of the launch has been displayed in figure 2. This shows the trajectory of the projectile, annotated is the point of maximum altitude with a line perpendicular to the surface of the earth. The parachute deployment at $10\, km$ has also been annotated along with the point of impact. The red line shows the

path the second stage would take. We decided to initially have the second stage reach orbital velocity to orbit at the altitude of apogee.

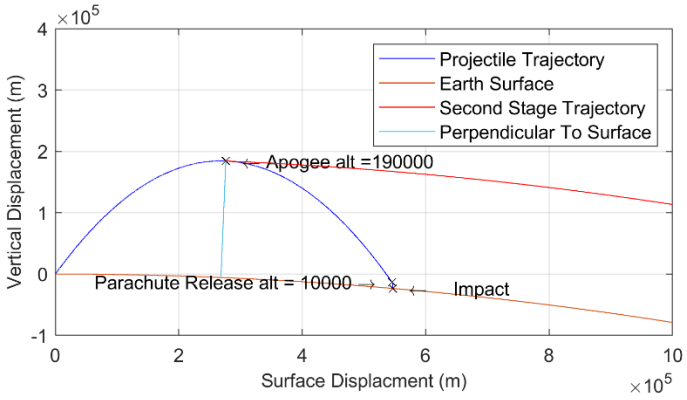


Figure 2: 2D plot of launch

4. Extensions & Discussion

4.1. Third Dimension

To make the model more general purpose the initial first step was to introduce a third dimension the launch. This was done by introducing an additional term to the z vector allowing the velocity and acceleration. With this third dimension, we were able to introduce the ability to change the direction of the launch by putting in a value for the heading between $180° \; and \; -180°$. We decided that the simplest way of displaying the 3D plot was to have the initial launch location as the origin. From this point, the Z-axis extends down towards the centre of the globe, the positive X-axis points east and the positive Y-axis points north.

4.2. Latitude and Longitude

The ability to calculate the landing location of a ballistic object is imperative to reduce the chance of the object harming people or damaging property. To calculate the impact location the launch location was first set as coordinates of latitude (Lat_0) and longitude $(Long_0)$.

As the globe plot initially had the launch location as the north pole it had to be rotate to match the selected launch position. The earth plot was first rotated by $Long_0$ in the Z-axis. It was then rotated $90°$ in the X-axis to get to the equator and then further rotated by Lat_0 in the same axis. This was simplified down to a single rotation of $Lat_0 - 90°$. This approach allowed the launch location to be changed to any point on the Earth's surface.

The projectile trajectory then was computed, and the $Long_0$ value was used to calculate the perpendicular distance from the launch position to the rotational axis of the Earth (R_x). The displacement in the X, Y and Z axis at impact were then used to determine the latitude and longitude at landing through trigonometry, as shown in equations 5 and 6.

$$Impact\ Latitude \; = \; Lat_0 \pm \cos^{-1}\left(1 - \frac{y^2 + z^2}{2 \cdot (R_e)^2}\right) \tag{5}$$

$$Impact\ Longitude \; = \; Long_0 \pm \cos^{-1}\left(1 - \frac{x^2 + z^2}{2 \cdot (R_x)^2}\right) \tag{6}$$

Equation 5 was derived by taking a cross section of Earth with radius R_e through the launch position and using the cosine formula to calculate the angle subtended by the launch position and the projected landing position. A positive or negative value was assigned to this angle depending on the direction of the net displacement in Y.

Equation 6 was derived using the same approach but taking a horizontal cross section of Earth through the launch position. A positive or negative value was assigned to this angle depending on the direction of the net

displacement in X. This approach assumes a small distance travelled and thus assumes that the difference between the value of R_x at the launch point and the impact point to be negligible.

Adjustments had to be made for when the launch went over the longitude line of $180°$ by removing $360°$ from the absolute value and multiplying by the signum function of the longitude. This enabled the calculations to be used for any launch scenario.

With the output of the longitude and latitude, the Mapbox static map API (Static Maps | Mapbox, n.d.) was used to plot the point of impact on a high-resolution satellite map with an overlay of roads and population centres. This allows the launch angles to be planned for the object to land in rural areas to reduce the risk of harm to humans as shown in figure 3.

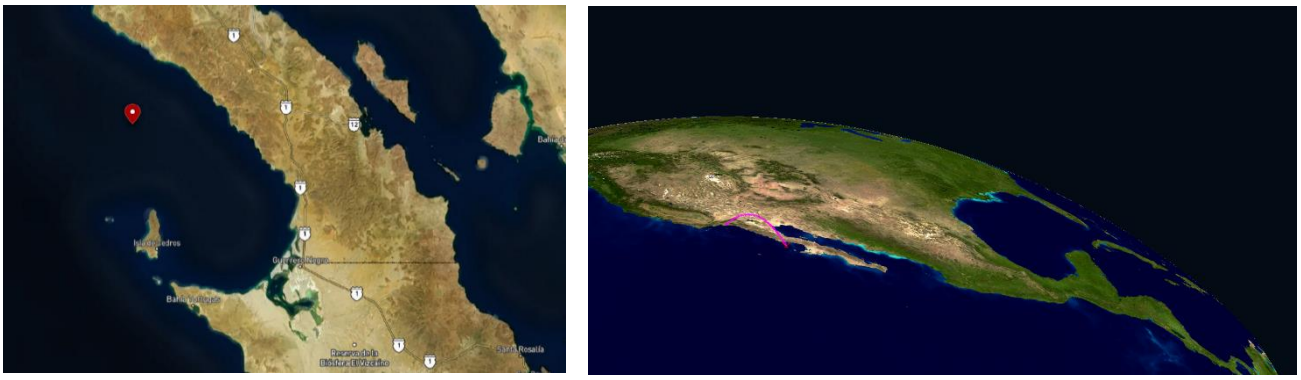


Figure 3: Satellite Image of Impact Location When Lunching from Long Beach at an angle of –78 Degrees from East.

There were some shortcomings in the method used to calculate and display the latitude and longitude. The main issue was the plot for the trace of the launch produces a line which doesn't follow lines of latitude. This is most apparent when the object launches directly east or west and therefore causes the Y-displacement to be zero. Due to the placement of the axis, the curvature of the earth causes the lines of latitude to curve away from the X-axis in relation to the latitude value. This would result in there being a Y-displacement that is unaccounted for within the trajectory model. This however is not an issue with the landing location satellite map due to the way the coordinates were calculated.

4.3. Rotation of Earth

Different points on the surface of a rotating sphere will have varying nominal velocities as the distance between them and the rotating axis changes. When launching projectiles from earth, the contribution of the rotation of the earth to the projectiles net velocity must be considered to decrease the required thrust needed to overcome Earth's gravitational attraction.

To calculate the nominal velocity (V_E) at any point on Earth's surface the Longitude value at the launch position was used to calculate the perpendicular distance from the launch position to the rotational axis. This approach assumed that Earth is a perfect sphere. The calculated value was then multiplied by the Earth's angular velocity to obtain the V_E.

As Earth rotates from W to E, it was determined that the velocity V_E acted along the positive X axis direction. The projectile velocity at launch $(2500\ ms^{-1})$ was decomposed into its X, Y and Z components using the projectile angle from the positive X and Z axis at launch. Depending on the projectile direction (E or W), the value of V_E was added to or subtracted from the projectile velocity in the X direction. The projectile net velocity at launch was then recalculated and used in the first ODE calculation. The effect of Earth's rotation, and no wind, at $\theta_x = 0°\ (W\ to\ E)\ and\ \theta_x = 180°\ (E\ to\ W)$ are shown by table 1.

Table 1. Effect of Earth's rotation on projectile trajectory

Launch direction	Net X displacement $(\times 10^5 m)$	Net Y displacement (m)	θ_z at apogee $(degrees)$

$W\ to\ E$	6.9258	0	54.1187
$E\ to\ W$	-4.0191	0	54.1872

The calculated results show in table 1 agree with those expected by theory. A more accurate answer could be achieved when calculating V_E by taking Earth to be an ellipsoid instead of a sphere. Further development could also use a geoid model of earth, so that the distance from the launch position to the rotational axis of Earth is dependent in both longitude and latitude as the Earth's surface varies in elevation.

4.4. Effects of wind on the projectile motion

To further increase the accuracy of this model the force exerted by wind on a projectile and its effect on the projectile trajectory were calculated. Due to small projectile displacement (relative to the earth's radius), and to decrease the complexity of the required calculations, wind was taken to be acting on the two-dimensional plane X-Y tangential to launch point. The force exerted by wind on the projectile was calculated using equation (x).

Unlike the previously completed calculations for air resistance, it was determined that both C_D and A varied depending on the surface of the projectile, with A also changing depending on the projectile position relative to the wind direction. To facilitate the calculations, the projectile geometry was a square prism with a square front face area of $A_1 = \pi\ m^2$ and a rectangular side face area of $A_2 = 20\ m^2$. The coefficient of drag (C_D) for the square faces and the rectangular faces were equated to those of the cylindrical projectile, $0.1\ and\ 1$ respectively.

At any given time, wind can only exert a force on a maximum of three surfaces of the prism. Assuming that the nominal velocity of the prism is normal to the square face, only one square surface and two rectangular faces will experience a force due to wind. The projected area (A) of both the rectangular surfaces and the square face varied from 0 to A_2 and A_1 respectively as the three-dimensional rotation of the projectile relative to the wind changed during its trajectory. The net projected area for the rectangular surfaces (A_R) and the square surface (A_S) in the X and Y directions at each time step were calculated using equations 7 and 8 respectively.

$$A_{Rx} = A_2[\sin(\theta_x) + \cos(\theta_x)\sin(\theta_z)]\ ,\ A_{Sx} = A_1[cos(\theta_x)cos(\theta_z)] \quad (7)$$

$$A_{Ry} = A_2[\cos(\theta_x) + \sin(\theta_x)\sin(\theta_z)]\ ,\ A_{Sy} = A_1[sin(\theta_x)cos(\theta_z)] \quad (8)$$

Where θ_x and θ_z are the projectile angle from the positive X and Z axis at that time step respectively.

To enable any wind direction on the X-Y plane to be used within these calculations all wind angles from the positive X axis (θ_w) with an absolute value greater than $90°$ (Wind travelling West) were automatically overwritten to the difference of their magnitude and $180°$.

Equations 7 and 8 were then used to calculate the net force on the X and Y direction and multiplied by a trigonometric function of θ_w to account for changes in wind direction. These forces were then divided by the projectile mass (M) and assigned a sign depending on θ_w and the sign of the velocity components in the X and Y direction. These values were then included into the derivative calculation.

At an altitude of $10{,}000\ km$ as the projectile returned to the surface both $A_1 and\ A_2$, and their respective drag coefficients, where increased to include the forces exerted by the wind on a parachute with a rectangular cross-section of $4m \times 6m$.

The final code enabled the effect of wind on the projectile trajectory to be calculated at any angle and combination of $\theta_w, \theta_x\ and\ \theta_z$. The effect of wind at $\theta_w = -45°\ (NW\ to\ SE)\ when\ \theta_x = 0°\ (W\ to\ E)\ and\ \theta_x = 180°\ (E\ to\ W)$ on the projectile trajectory is shown by table 2.

Table 2. Effect of wind on projectile trajectory

	Net X displacement $(\times \mathbf{10^5 m})$	Net Y displacement $(\boldsymbol{m})$	$\boldsymbol{\theta_z}$ at apogee $(\boldsymbol{degrees})$
W to E	6.9300	-438.2340	54.1184

E to W	-4.0151	-423.5282	54.1873

The calculated results shown in table 2 agree with those expected by theory when compared to the values in table 1. A more precise calculation would require the velocity of wind (v) to be maintained tangential to the earth surface, expanding the calculations to a third dimension, and thus calculating the Z component of the exerted force. The effects of weather cocking were also taken into consideration, however, due to the lack of stabilising wings on the projectile, its effects were neglected.

4.5. Orbital velocity and Circular orbit

To get the projectile from apogee and into Low Earth Orbit (LEO), the orbital velocity must be calculated. This was done by creating a function with inputs orbital height (oH) and initial mass (mI) and outputs final velocity (vFinal), final mass (mFinal), total burn time (burnTime) and velocity error (errorV). This function provided the opportunity to calculate whether a certain orbit was possible with the remaining resources that the projectile possesses.

Due to the projectile continuously expelling fuel, the calculation for acceleration was not as straightforward as dividing the force by the mass. An array was created ranging from the initial mass, mI, to the final mass, 3500, with equal increments of the exit mass rate, mE. The thrust force produced by the nozzle was constant and was calculated with the following:

$$F_{Nozzle} = \dot{m}V_e + (p_e - p_0)A_e \tag{9}$$

An array of the nozzle thrust was created with a constant thrust value in each column. Its length was specified to match that of the length of the massChange array, allowing for array division despite a change in the mass input. To get the acceleration at each mass reduction iteration, the force array was divided by the mass array. This method resulted in increasing acceleration values over time, allowing for the calculations to account for the change in mass when calculating the new velocity. The orbital velocity for an orbit height of 'r' has the following equation:

Similarly, to thrust, an array was also made for a timestep of one second. This idea became plausible when realising that one iteration could be read as one second, equating second to one run-through of the loop. It also provided a predictable and linear relationship between exit mass rate and time.

Multiple assumptions were made through these calculations. Assumption one was to treat the projectile as a point mass, ignoring the effects of propellant sloshing and externally created moments. Another assumption was to begin the second stage directly when the projectile arrived at its apogee, meaning the only velocity it possessed at that instant was horizontal velocity. The difference between this initial horizontal velocity and the orbital velocity calculated using the above formula would subsequently be named 'deltaV'.

A while loop was nested into the code to iteratively calculate the current velocity due to the instantaneous acceleration provided by the decreasing mass of the projectile. *'While Vc > Vf'* ensured that the projectile velocity would continuously increase until it was either greater than or equal to the orbital velocity. Two extra variables were set before the while loop: *'iterCount =0'* and *'n=1'*. With each loop completion, both variables would increase. Once the statement was met, the value of *iterCount* was taken to be the burn time. The velocity was updated once through each loop, increasing with the *n+1* value of the acceleration array.

Working in integers for both seconds and mass reduction always resulted in a slight overshoot of the final projectile velocity once the loop had finished. This difference was amended by taking the difference between final velocity and orbital velocity. This difference was then divided by the acceleration at *n-1,* which is the acceleration of the projectile just before ECO. Dividing the velocity difference by the acceleration gave the extra time for which the thrusters were on. This extra time would then be the same duration the thrusters would need to be applied in the opposite direction to motion. The iteration count and final mass were updated by accounting for this stabilising action. A velocity error was also calculated between the final velocity and orbital velocity. The error in the calculations resulted in the final velocity being ±0.03% off the theoretical orbital velocity. This translated to

approximately an orbit height deviation of ±0.4km. In the scenario that the resources remaining in the projectile aren't sufficient to produce the change in velocity required, an error message will be displayed asking for the user to recalculate and reconsider their desired orbit choice.

4.6. Orbital Decay

Creating a realistic orbit requires the orbit to decay over time. This was modelled by calculating the reduction in orbital velocity due to variables such as drag, geomagnetic index, solar flux and effect molar mass variable scale height and local density. To ensure mathematical accuracy, the governing equations on which the period reduction was dependent are derived for thermosphere use only and can be found in the appendix. The reduction in the orbital period due to these variables is given by:

$$dP = 3\pi\left(\frac{A_{perpendicular}}{mass}\right) \times orbit\ height \times \rho_{orbit\ height} \times Seconds\ in\ a\ day \qquad (10)$$

Multiple assumptions were made here. It was assumed that an orbit would 'fail' once its orbital period sent its height below 180km. This is due to the knowledge that orbits within 180km of the earth's surface fail within a day without interference from onboard thrusters. It was also assumed that the input variables, geomagnetic index, and solar flux are constant throughout the entire orbit.

4.7. Hohmann Transfer Manoeuvre (Relevant script and Example script)

The Hohmann Transfer is used to transfer orbits using the least amount of energy possible. Following the initial orbital insertion into 190km, this code can be used to transfer the projectile into higher orbits on the condition that the deltaV and timeToDeltaV are known. With initial inputs of deltaV = 19 and timeToDeltaV = 20, the total transfer time from 190km (our initial orbit) to 400km (ISS orbit), takes 1348 seconds. Another similar script is provided named 'hohmannTransferExample', where the manoeuvre can be seen in full effect traversing much larger distances during its transfer. The values of deltaV and timeToDeltaV provided in the example are calculated from the larger orbit.

Numerous things were calculated beforehand to ensure the elliptical orbit transfer was mathematically correct:

$$V\ at\ transfer\ apoapsis,\ V_{ta} = \sqrt{GM_e\left(\frac{2}{R_a} - \frac{1}{a}\right)} \quad (11), \qquad V\ at\ transfer\ periapsis,\ V_{tp} = \sqrt{GM_e\left(\frac{2}{R_p} - \frac{1}{a}\right)} \qquad (12)$$

$$Rocket\ equation\ in\ space,\ v = v_0 + u\ln\left(\frac{m_0}{m_0 - \frac{\Delta m}{\Delta t}t}\right) \quad (13), \qquad Burn\ time,\ \Delta T = \frac{m_0 \cdot v_e}{F} \cdot \left(1 - e^{\left(-\frac{\Delta v}{v_e}\right)}\right) \qquad (14)$$

Due to the nature of the math behind a transfer orbit, the visualisation of the plot was kept in 2D to keep the code simple. The projectile's X and Y coordinate continuously change as a result of the initial velocity applied. As such, the norm function was applied to the radius, calculating the Euclidean distance. The norm function returns the square root of the sum of the squares of the elements meaning it can be used in both 2D and 3D scenarios to calculate the magnitude of the radius when the coordinate values change. A dynamic array is created where a list of values is calculated using the initial parameters.

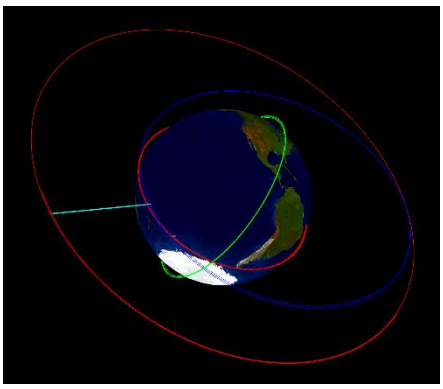
The script is split into three sections, one for each orbit. The code for the first and third orbits is nearly identical due to only plotting circles. The second orbit is plotted as an elliptical orbit which transitions from orbit one to orbit three. The loop in the second orbit section is stopped when the angle of the orbit reaches pi. This is due to only half of the transfer orbit being required. The angle of the trajectory is calculated using the new r and the previous r using the vector dot product. The built-in Matlab function, comet, was also modified by adding pauses in the for loops to be able to appreciate the animation.

4.8. 3D Circular Orbit

A 3D visualisation of the code above is compiled into '*3D Circular Orbit*'. Calculating the position of the orbiters was done by using functions where the state derivatives were calculated based on initial parameters. Coordinate

values for X, Y and Z were continuously changing, hence, the norm was used again. The functions named '*proj3DOrbit'* and '*ISSOrbit'* , received predefined inputs of the satellite's initial state, state, formatted [X, Y, Z, Xdot, Ydot, Zdot] and orbit timespan, t. Using the initial parameters provided, it would then output the projectile's next state formatted [Xdot, Ydot, Zdot, Xaccel, Yaccel, Zaccel]. ODE15s was chosen to be the solver due to the stiffness within the ODE. To ensure the accuracy of results, two subplots were plotted, one subplot with the cartesian displacements, and the other with their respective velocities. The Hohmann transfer from the initial orbit (Inner red line) to the final orbit (Outer red line) has been included.

Dotted reference lines for the X, Y and Z axes and equator were also added. Earth's tilt and rotation were also accounted for, and hence the reference lines also had rotation matrices applied in order to stay stationary during the entirety of the infinite while loop. The background of the plotting area and general space were turned black to simulate the environment of space.



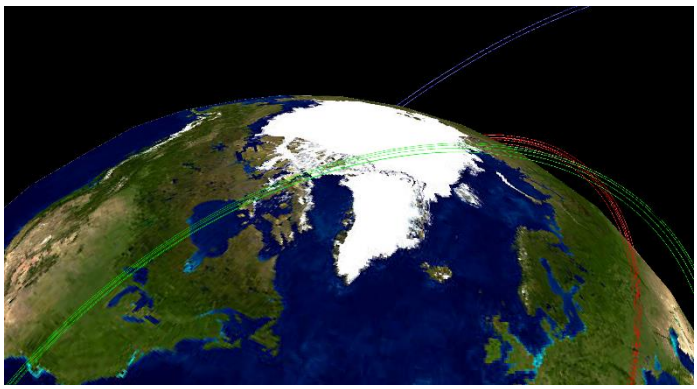


Figure 4: 3D representation of second stage orbit, ISS orbit and transfere orbit shown in red green and blue respecivly

To give the user some control over the simulation, three UI controls were added, two of which are interactive. The central UI displays the current rotation speed, which at default is set to 10000. The push buttons on either side increase or decrease the rotational speed by 500. The callback used for the buttons was '*Callback,@(~,~)'* suppressing both the input and output of the function and moving directly to the string-to-number then number-to-string conversion of the text box value.

5. Conclusion

An initial two-dimensional model for the projectile trajectory with initial conditions of $U = 2500\, ms^{-1}\, and\, \Delta t = 0.025s$ was developed using the Rung-Kutter 4th order method at various launch angles θ_x. The launch angle required to reach an apogee of $190\, km$ was calculated to be $54.18°$ with an error of $0.62\, m$ using the shooting method. At this angle and with a parachute deployment at $10\, km$ from the surface, the surface displacement, projectile velocity at impact and time of flight were calculated to be of $547.26\, km, 41.57\, ms^{-1}\, and\, 579.8\, s$ respectively. The accuracy of the model was then increased by calculating the trajectory in three dimensions, considering the effect of wind and the rotation of the Earth at launch position on the projectile trajectory. The final first stage model was able to compute the trajectory of a projectile launched at any direction, in any location on earth experiencing wind acting at any direction on the X-Y plane. The coordinates of the landing position were then calculated and used to display it on a satellite hybrid map.

The projectile was placed into a circular orbit by applying thrust for a total burn time of 98.4135 seconds, resulting in a total mass decrease of $19682.7kg$. Slight discrepancies were found between orbital velocity and final projectile velocity and were amended. Concluding the manoeuvre, a velocity error of 0.0304% was present.

A 3D visual representation of the extension tasks was then plotted around a dimensionally accurate, rotating, ellipsoidal earth. A line connecting the satellite position to the centre of the earth was also plotted, mimicking ground tracking.

References

Edun, I. F., & Akinlabi, G. O. (2021). Application of the shooting method for the solution of second order boundary value problems. *J. Phys*, 12020. https://doi.org/10.1088/1742-6596/1734/1/012020

Holt, J. B., & Monk, T. S. (2009). *Propellant Mass Fraction Calculation Methodology for Launch Vehicles and Application to Ares Vehicles*.

Liu, C., Wu, H., Feng, L., & Yang, A. (2011). LNCS 7030 - Parallel Fourth-Order Runge-Kutta Method to Solve Differential Equations. *LNCS*, *7030*, 192–199.

Petrescu, R. V., Aversa, R., Akash, B., Berto, F., Apicella, A., & Petrescu, F. I. T. (2017). Project HARP. *Journal of Aircraft and Spacecraft Technology*, *1*(4), 249–257. https://doi.org/10.3844/JASTSP.2017.249.257

Static Maps | Mapbox. (n.d.). Retrieved December 6, 2022, from https://www.mapbox.com/static-maps

Appendices

Appendix A – Running the code

1st stage – Open $BallisticModel.m$. At the top of the script all variable condition values can be changed. Run $BallisticModel.m$ to calculate the projectile trajectory.

2nd stage – Code must be run in order shown below.

Call function $orbitalVelocity.m$ and format like the following:

[vFinal, mFinal, burnTime, errorV] = orbitalVelocity(190000,25000)

Where: oH = height above surface in metres, mI = initial mass

Note: mI can be updated by user following first manoeuvre. mFinal = new mI.

Call function $orbitDecay$ and format like the following:

[yDays,yYears] = orbitDecay(aboveSurface,F10.7,Ap)

Where: aboveSurface = height above surface in metres, F10.7 = Solar flux (0 to 300),

Ap = Geomagnetic index (1 to 9)

Open $hohmannTransfer.m$ and press run.

This script shows the transfer manoeuvre from initial orbit to ISS orbit. For exaggerated view of transfer:

Open $hohmannTransferExample.m$ and press run. This script takes 15-20 seconds to load.

Open $circularOrbit3D.m$ and format like the following to achieve graph in 3D section above:

[orbPlot] = circularOrbit3D(190000,12)

Note: Values above achieve the same graph as shown in figure 4. Values can be changed to vary orbits. Red line = projectile orbit, Green line = ISS, Blue line = Hohmann transfer from initial orbit to final orbit (Works only for specified inputs).

Appendix B – Code equations

Circular Orbit

$$V_c = \sqrt{\frac{GM_e}{r}}$$

Orbital decay equations

$$T_{Thermosphere} = 900 + 2.5(F10.7 - 70) + 1.5A_p$$

$$Effective\ molar\ mass = 27 - 0.012\left(\frac{Height\ above\ surface}{1000} - 200\right)$$

$$Variable\ scale\ height,\ H = \frac{T_{Thermosphere}}{Effective\ molar\ mass}$$

$$\rho_{orbit\ height} = (6 \times 10^{-10})e^{-\frac{\frac{orbit\ height}{1000} - 175}{H}}$$

Hohmann transfer equations

$$Semi - major\ axis,\ a = \frac{R_a + R_p}{2}$$

$$Velocity\ at\ apoapsis,\ V_a = \sqrt{\frac{GM_e}{R_a}}$$

$$Velocity\ at\ periapsis,\ V_p = \sqrt{\frac{GM_e}{R_p}}$$

$$Earth's\ sphere\ of\ \text{influence},\ \mu = GM_e$$

$$General\ rocket\ equation,\ \Delta v = v_e \ln\left(\frac{m_0}{m_f}\right)$$