# AE 370 Project 1 (Group Component)

John Klis, Andrew Myers, Luke Brown, Matthew Weippert, Jack Stevenson*
*University of Illinois Urbana-Champaign, Champaign, Illinois, 61820*

**To model non-linear rocket dynamics, our project aims to use an explicit numerical method to first model a rocket under ideal rocket conditions. After a high degree of accuracy is achieved using this numerical method, we will evaluate the contribution of the drag assumption and initial condition to the overall error of the rocket trajectory.**

## Contents



*Department of Aerospace Engineering, University of Illinois at Urbana-Champaign

# I. Dynamical System

## A. Dynamical System Importance and Relevance

The process of understanding any complex system starts similarly; assumptions and simplifications are made to get a grasp on the core physical mechanisms driving the behavior of that system. As the understanding of that system solidifies, many of these assumptions are removed to reveal a more complex system beneath, a system which will take longer to fully understand the mechanics of. In the field of aerospace engineering, the true nature of these systems is nearly always nonlinear, which poses an enormous barrier to a full physical understanding of the system in question.

The topic of this paper, as well as an interesting dynamical system that has this process of understanding attached, is modeling rocket dynamics. Upon first being introduced to rocket dynamics, the ideal rocket equation, or the Tsiolkovsky rocket equation is usually the first equation learned due to the simplifications it makes. In this equation, there are only five variables, allowing for a straightforward calculation of the velocity. From velocity information and a set of initial conditions, the trajectory of a rocket can be easily calculated. Even if you were to convert this equation into a vector-valued equation, this type of analysis seems so simple it is almost boring. The reason this is the case is the sheer number of assumptions required to make such a model possible in the first place. These assumptions may seem to obscure the true answer to a significant degree; however without such a solution increasingly complicated mathematical computation the degree of impact of these assumptions is unknown. For this reason, our project aims to investigate the impact various assumptions of the ideal rocket equation have on the true solution, which we will approximate numerically. The chief assumption this paper will examine is the drag assumption, and its implications on various rocket conditions.

## B. Dynamical System Questions

In implementing a numerical method to approximate this system, we seek to demonstrate how the presence of drag affects the trajectory of an ideal single-stage rocket. We will explore different air densities, different drag coefficients, and the percentage error of the ideal solution and one estimated with drag. Put another way, our questions are:

- What is the relation between drag factors and the error between these models?
- Does the ideal model estimate true behavior better for rockets with certain initial conditions (chiefly fuel-mass ratio and exhaust velocity)?

To answer these questions, we must first establish the dynamical system in both the ideal model and in the model with drag, along with their complete derivation to ensure accuracy.

## C. Dynamical System Equations

In a realistic setting, a rocket is subject to thrust, drag, and gravity. The equation of motion is [1]:

$$\frac{d}{dt}(mv) = F_{\text{thrust}} - F_{\text{drag}} - F_{\text{gravity}}, \tag{1}$$

where:

$$\begin{aligned} F_{\text{thrust}} &= \dot{m} v_e, \\ F_{\text{drag}} &= \frac{1}{2} C_D \rho(h) A v^2, \\ F_{\text{gravity}} &= mg. \end{aligned}$$

Expanding the derivative on the left-hand side:

$$m\frac{dv}{dt} + v\frac{dm}{dt} = \dot{m} v_e - \frac{1}{2} C_D \rho(h) A v^2 - mg. \tag{2}$$

Dividing through by $m$ and noting that $\dot{m} = \frac{dm}{dt} < 0$, we obtain:

$$\frac{dv}{dt} = (v_e - v)\frac{\dot{m}}{m} - \frac{1}{2m} C_D A \rho(h) v^2 - g. \tag{3}$$

**Atmospheric Density Model**

To account for the decrease in air density with altitude, we use the exponential model:

$$\rho(h) = \rho_0 e^{-h/H}, \tag{4}$$

where:

- $\rho_0$ is the sea-level air density (typically $1.225\,\text{kg/m}^3$),
- $h$ is the altitude,
- $H$ is the scale height of the atmosphere (typically 8500 m).

The altitude $h$ evolves with velocity:

$$\frac{dh}{dt} = v. \tag{5}$$

The full system of coupled differential equations becomes:

$$\frac{dv}{dt} = (v_e - v)\frac{\dot{m}}{m} - \frac{1}{2m} C_D A \rho_0 e^{-h/H} v^2 - g \tag{6}$$

$$\frac{dh}{dt} = v \tag{7}$$

$$\tag{8}$$

This nonlinear system captures the rocket's dynamics with realistic drag variation due to altitude. Analytical solutions are generally not possible, and numerical methods are typically employed to solve the system.

If we neglect the drag term, we use the following derivation to create an analytical solution [2] [3]:

We define the useful quantity:

$$\xi = \frac{m_p}{m_0} = \frac{m_0 - m_f}{m_0} = 1 - \frac{m_f}{m_0}, \tag{9}$$

where:

- $m_0$ is the initial mass of the rocket,
- $m_f$ is the final mass of the rocket after burnout,
- $m_p$ is the propellant mass of the rocket,
- $\xi$ represents the fraction of initial mass lost as fuel.

Assuming that the launch angle $\theta$ is fixed during the burn period, the forces acting on the rocket are:

- Thrust: $T = -\dot{m} v_e$
- Gravity: $mg$, acting downward.

Applying Newton's Second Law along the rocket's direction of motion:

$$m\frac{dv}{dt} = -\dot{m} v_e - mg. \tag{10}$$

Rewriting using $\dot{m} = \frac{dm}{dt}$:

$$dv = -v_e \frac{dm}{m} - g\,dt. \tag{11}$$

*1. Integrating for Velocity During Burn Period*

Integrating both sides from $m_0$ to $m_f$ and from 0 to $t_b$:

$$\int_0^{v_b} dv = -v_e \int_{m_0}^{m_f} \frac{dm}{m} - g \int_0^{t_b} dt. \tag{12}$$

Solving:

$$v_b - v_0 = -v_e \ln \frac{m_f}{m_0} - g t_b. \tag{13}$$

Using $\xi = 1 - \frac{m_f}{m_0}$, we rewrite:

$$v_b = -v_e \ln(1 - \xi) - g t_b. \tag{14}$$

*2. Integrating for Position During Burn Period*

We integrate:

$$\frac{dy}{dt} = v. \tag{15}$$

Substituting $v$:

$$\frac{dy}{dt} = \left[ -v_e \ln \frac{m}{m_0} - gt \right]. \tag{16}$$

Integrating both sides:

$$y_b - y_0 = \int_0^{t_b} \left[ -v_e \ln \frac{m}{m_0} - gt \right] dt. \tag{17}$$

Using the given integral:

$$\int_0^a \ln(1 - qt), dt = -a + \left( a - \frac{1}{q} \right) \ln(1 - aq), \tag{18}$$

we obtain:

$$y_b = y_0 + v_e \left[ t_b - \left( t_b - \frac{t_b}{\xi} \right) \ln(1 - \xi) \right] - \frac{1}{2} g t_b^2. \tag{19}$$

*3. Motion After Burnout (Projectile Motion)*

After fuel exhaustion, the rocket follows projectile motion under gravity.

- Acceleration:

$$a = -g. \tag{20}$$

- Velocity:

$$v(t) = v_b - gt. \tag{21}$$

- Position:

$$y(t) = y_b + v_b t - gt^2. \tag{22}$$

*4. Final Summary of Equations*

**During Burn Period** ($0 \leq t \leq t_b$)**:**

$$v(t) = -v_e \ln(1 - \xi(t/t_b)) - gt, \tag{23}$$

$$y(t) = y_0 + v_e \left[ t - \left( t - \frac{t}{\xi} \right) \ln(1 - \xi) \right] - \frac{1}{2} g t^2. \tag{24}$$

**After Burnout** ($t > t_b$)**:**

$$v(t) = v_b - g(t - t_b), \tag{25}$$

$$y(t) = y_b + v_b(t - t_b) - \frac{1}{2} g (t - t_b)^2. \tag{26}$$

## II. Appropriate Numerical Method

To arrive at a true-to-life estimate of the extent of error caused by ideal rocket assumptions, we are choosing to use the fourth-order Runge-Kutta method (RK4). Modeling a rocket trajectory does not qualify as a stiff problem, as the trajectory will be free from oscillations and sudden changes. Thus, an explicit method, like RK4, will likely remain stable without requiring an extremely small step size. Additionally, being a fourth-order method, the error will be sufficiently small with little computational cost. The choices for this method will be further explained in this section.

## A. Justification for Runge-Kutta 4

Runge-Kutta 4 is a single-step, explicit method that achieves fourth-order accuracy, making it well-suited for problems involving smooth but nonlinear dynamics like rocket motion, where mass and velocity change continuously. Unlike multistep methods, RK4 does not require data from previous time steps, which simplifies implementation and is ideal for problems with evolving dynamics or changing control parameters. While implicit methods are generally more stable for stiff equations, the rocket dynamics in this scenario are not stiff, so the extra computational cost of solving nonlinear systems at each step is unnecessary. RK4 offers a good trade-off between stability and computational cost, with stable performance for moderately sized time steps, which is critical for long-duration simulations where small errors can accumulate. Its explicit nature also avoids matrix inversions or iterative solvers, reducing complexity and execution time. Overall, RK4 provides reliable, high-accuracy solutions for the non-stiff, nonlinear ODEs describing rocket motion, making it a practical and efficient method for this problem [1].

## B. Derivation of the Method

*1. Deriving the Fourth-Order Method*

To derive the Runge-Kutta 4 (RK4) method from first principles, we start with the ordinary differential equation [4]:

$$\dot{u}(t) = f(u(t), t), \tag{27}$$

with an initial condition $u(t_0) = u_0$. We integrate both sides over a small time step from $t_k$ to $t_{k+1} = t_k + h$:

$$\int_{t_k}^{t_{k+1}} \dot{u}\, dt = \int_{t_k}^{t_{k+1}} f(u(t), t)\, dt. \tag{28}$$

The left-hand side (LHS) is straightforward:

$$u(t_{k+1}) - u(t_k) = \int_{t_k}^{t_{k+1}} f(u(t), t)\, dt. \tag{29}$$

Thus, the solution at the next step is:

$$u_{k+1} = u_k + \int_{t_k}^{t_{k+1}} f(u(t), t)\, dt. \tag{30}$$

**Approximating the Integral Using Local Polynomial Interpolation**

The key idea in RK4 is to approximate the integral $\int_{t_k}^{t_{k+1}} f(u(t), t)\, dt$ using a weighted average of $f$ evaluated at carefully chosen points in the interval $[t_k, t_{k+1}]$. Specifically, RK4 uses a **fourth-order** approximation, which requires evaluating $f$ at four intermediate points.

**Step 1: Approximate $f$ Using a Weighted Average**

We approximate the integral as:

$$\int_{t_k}^{t_{k+1}} f(u(t), t)\, dt \approx \Delta t \left(w_1 f_1 + w_2 f_2 + w_3 f_3 + w_4 f_4\right), \tag{31}$$

where:

- $\Delta t = t_{k+1} - t_k$ is the step size,
- $w_1, w_2, w_3, w_4$ are weights,
- $f_1, f_2, f_3, f_4$ are evaluations of $f$ at intermediate points.

**Step 2: Choose Intermediate Points (Stages)**

RK4 uses the following four evaluations:

1) **First evaluation (Euler step):**

$$f_1 = f(u_k, t_k). \tag{32}$$

2) **Second evaluation (midpoint estimate):**

$$f_2 = f\left(u_k + \frac{\Delta t}{2} f_1, t_k + \frac{\Delta t}{2}\right). \tag{33}$$

3) **Third evaluation (improved midpoint estimate):**

$$f_3 = f\left(u_k + \frac{\Delta t}{2} f_2, t_k + \frac{\Delta t}{2}\right). \tag{34}$$

4) **Fourth evaluation (full-step estimate):**

$$f_4 = f\left(u_k + \Delta t f_3, t_k + \Delta t\right). \tag{35}$$

**Step 3: Determine the Weights**

To achieve fourth-order accuracy, the weights must satisfy certain conditions derived from Taylor expansions. The standard RK4 weights are:

$$w_1 = \frac{1}{6}, \quad w_2 = \frac{1}{3}, \quad w_3 = \frac{1}{3}, \quad w_4 = \frac{1}{6}. \tag{36}$$

These weights ensure that the approximation matches the Taylor expansion of the true solution up to $O(h^5)$.

**Step 4: Combine into the RK4 Formula**

Substituting the weights and evaluations, the RK4 update formula is:

$$u_{k+1} = u_k + \frac{\Delta t}{6}\left(f_1 + 2f_2 + 2f_3 + f_4\right), \tag{37}$$

where:

- $f_1 = f(u_k, t_k)$,
- $f_2 = f\left(u_k + \frac{\Delta t}{2} f_1, t_k + \frac{\Delta t}{2}\right)$,
- $f_3 = f\left(u_k + \frac{\Delta t}{2} f_2, t_k + \frac{\Delta t}{2}\right)$,
- $f_4 = f\left(u_k + \Delta t f_3, t_k + \Delta t\right)$.

**Final Explicit RK4 Update Formula**

The complete RK4 method for computing $u_{k+1}$ is:

$$k_1 = \Delta t f(u_k, t_k), \tag{38}$$

$$k_2 = \Delta t f\left(u_k + \frac{k_1}{2}, t_k + \frac{\Delta t}{2}\right), \tag{39}$$

$$k_3 = \Delta t f\left(u_k + \frac{k_2}{2}, t_k + \frac{\Delta t}{2}\right), \tag{40}$$

$$k_4 = \Delta t f\left(u_k + k_3, t_k + \Delta t\right), \tag{41}$$

$$u_{k+1} = u_k + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4). \tag{42}$$

*2. Error Analysis*

To analyze the error accumulation in the Runge-Kutta 4 (RK4) method, we derive both the local truncation error ($\tau_k$) and global error using a Taylor series approach with step size $\Delta t$ [4].

**Local Truncation Error (LTE) Derivation**

The local truncation error $\tau_k$ is the error introduced in a single step. We compare the exact solution $u(t_{k+1})$ to the RK4 approximation $u_{k+1}$ to derive the error for this method.

**Exact Solution (Taylor Expansion)**

The true solution at $t_{k+1} = t_k + \Delta t$ is:

$$u(t_{k+1}) = u(t_k) + \Delta t\, f + \frac{\Delta t^2}{2}\frac{df}{dt} + \frac{\Delta t^3}{6}\frac{d^2 f}{dt^2} + \frac{\Delta t^4}{24}\frac{d^3 f}{dt^3} + O(\Delta t^5), \tag{43}$$

where $f = f(u(t_k), t_k)$ and higher derivatives are evaluated at $t_k$.

**RK4 Approximation (Taylor Expansion)**

The RK4 method computes:

$$u_{k+1} = u_k + \frac{\Delta t}{6}(k_1 + 2k_2 + 2k_3 + k_4), \tag{44}$$

Expanding $k_1, k_2, k_3, k_4$ via Taylor series and substituting, we find:

$$u_{k+1} = u_k + \Delta t\, f + \frac{\Delta t^2}{2}\frac{df}{dt} + \frac{\Delta t^3}{6}\frac{d^2 f}{dt^2} + \frac{\Delta t^4}{24}\frac{d^3 f}{dt^3} + O(\Delta t^5). \tag{45}$$

**LTE Calculation**

Subtracting the exact solution from the RK4 approximation:

$$\tau_k = u(t_{k+1}) - u_{k+1} = O(\Delta t^5). \tag{46}$$

Thus, the LTE is $\tau_k = O(\Delta t^5)$ per step

**Global Error Derivation**

The global error $E$ is the cumulative error after $N$ steps (total time $T = N\Delta t$).

- LTE per step: $O(\Delta t^5)$.
- Number of steps: $N = \frac{T}{\Delta t}$.

Assuming errors add linearly:

$$E \approx N \cdot \tau_k = \frac{T}{\Delta t} \cdot O(\Delta t^5) = O(\Delta t^4). \tag{47}$$

Thus, the global error is $E = O(\Delta t^4)$.

**Key Results:**

| Error Type | Order | Explanation |
|---|---|---|
| Local ($\tau_k$) | $O(\Delta t^5)$ | Truncation error per single time step. |
| Global ($E$) | $O(\Delta t^4)$ | Cumulative error over the total time, $T$. |

The RK4 method is fourth-order accurate globally because the LTE accumulates as $1/\Delta t$ steps.

*3. Stability Derivation*

**Detailed Derivation of RK4's Absolute Stability Region**

The absolute stability region for an initial value problem numerical method consists of all complex values $z = \Delta t\lambda$ (where $\lambda$ is an eigenvalue of the system) for which the numerical solution remains bounded as $n \to \infty$. For RK4, we derive this region by analyzing how the method behaves when applied to the scalar test equation [4].

**Linear Test Equation**

We start with the model problem:

$$\frac{du}{dt} = \lambda u, \quad u(0) = u_0, \tag{48}$$

where $\lambda \in \mathbb{C}$. The exact solution is:

$$u(t) = u_0 e^{\lambda t}. \tag{49}$$

For stability, we require $\mathrm{Re}(\lambda) \leq 0$.

**RK4 Applied to the Test Equation**

The RK4 method computes:

$$u_{k+1} = R(z)u_k, \tag{50}$$

where $z = \Delta t\lambda$, and $R(z)$ is the stability function. To derive $R(z)$, we apply RK4 to the test equation:

**Stage 1** (Euler step):

$$k_1 = \Delta t\lambda u_k = z u_k. \tag{51}$$

**Stage 2** (Midpoint estimate):

$$k_2 = \Delta t\lambda \left(u_k + \frac{k_1}{2}\right) = z\left(1 + \frac{z}{2}\right) u_k. \tag{52}$$

**Stage 3** (Improved midpoint):

$$k_3 = \Delta t\lambda \left(u_k + \frac{k_2}{2}\right) = z\left(1 + \frac{z}{2} + \frac{z^2}{4}\right) u_k. \tag{53}$$

**Stage 4** (Full-step estimate):

$$k_4 = \Delta t\lambda \left(u_k + k_3\right) = z\left(1 + z + \frac{z^2}{2} + \frac{z^3}{4}\right) u_k. \tag{54}$$

**RK4 Update**:

$$u_{k+1} = u_k + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4). \tag{55}$$

Substituting $k_1, k_2, k_3, k_4$, we get:

$$R(z) = 1 + z + \frac{z^2}{2} + \frac{z^3}{6} + \frac{z^4}{24}. \tag{56}$$

**Stability Condition**

The method is absolutely stable if the numerical solution remains bounded, i.e., $|R(z)| \leq 1$. Thus, we solve:

$$\left|1 + z + \frac{z^2}{2} + \frac{z^3}{6} + \frac{z^4}{24}\right| \leq 1. \tag{57}$$

**Analyzing** $R(z)$

1. **For real** $z$: The method is stable when $R(z) \in [-1, 1]$. Numerically, this occurs for $z \in [-2.785, 0]$.
2. **For complex** $z$: We evaluate $|R(z)|$ on a grid in $\mathbb{C}$. The boundary of the stability region is where $|R(z)| = 1$.

**Key Observations in Fig. 1**

1. **Physical Stability**: The red region shows where RK4 is stable **and** $\mathrm{Re}(z) \leq 0$. The method is mathematically stable for $\mathrm{Re}(z) > 0$, but those systems are unphysical and can be ignored.

2. **Comparison with Exact Solution**: For $\mathrm{Re}(\lambda) \leq 0$, the exact solution decays, and RK4 matches this behavior within the red region. For $\mathrm{Re}(\lambda) > 0$, the exact solution grows unbounded and will not be stable.

3. **Imaginary Axis Stability**: RK4 remains stable for purely imaginary $z$ up to $|z| \approx 2.828$, making it suitable for oscillatory problems. While useful in general, this will not aid us in modeling the rocket's trajectory.

**C. Time Stepping Analysis**

The fourth-order Runge-Kutta (RK4) method advances the solution from $u_k$ to $u_{k+1}$ through four carefully designed slope calculations. Each step improves the approximation by capturing different aspects of the solution's behavior [4]:

1) **Initial Slope Estimation** ($k_1$ - Euler step):

$$k_1 = \Delta t\, f(u_k, t_k) \tag{38}$$

Computes the slope at the current point using Euler's method. This provides a first-order approximation that serves as the baseline for subsequent refinements.

2) **Midpoint Slope Prediction** ($k_2$ - First correction):

$$k_2 = \Delta t\, f\left(u_k + \tfrac{k_1}{2}, t_k + \tfrac{\Delta t}{2}\right) \tag{39}$$

Estimates the slope at the midpoint using the initial Euler step. This captures the solution's curvature by evaluating at an intermediate time level.

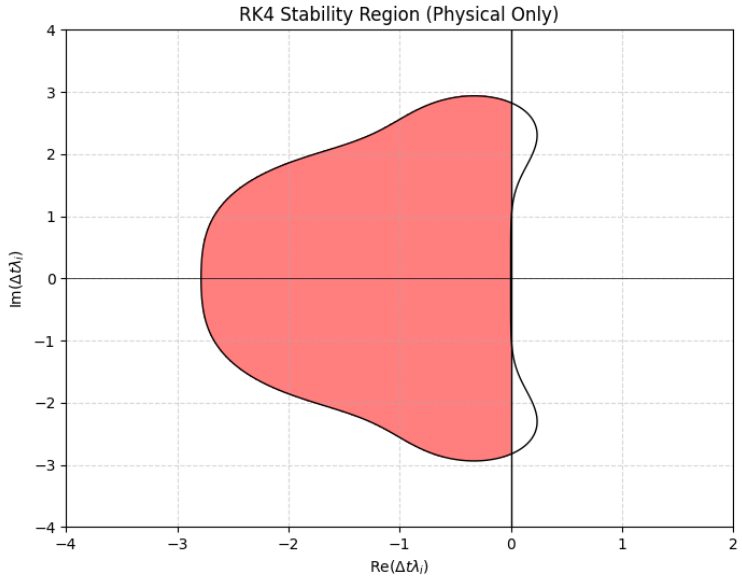


**Fig. 1 This graph showcases the physical region of absolute stability for the RK4 method.**

3) **Improved Midpoint Slope** ($k_3$ - Second correction):

$$k_3 = \Delta t\, f\left(u_k + \tfrac{k_2}{2}, t_k + \tfrac{\Delta t}{2}\right) \tag{40}$$

Recomputes the midpoint slope using the improved $k_2$ estimate. This refinement reduces error by incorporating more accurate intermediate information.

4) **Endpoint Slope Estimation** ($k_4$ - Final projection):

$$k_4 = \Delta t\, f(u_k + k_3, t_k + \Delta t) \tag{41}$$

Calculates the slope at the next time point using the best available approximation. This ensures the method accounts for the solution's behavior across the entire interval.

5) **Weighted Combination** (Fourth-order update):

$$u_{k+1} = u_k + \tfrac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{42}$$

Combines all four slopes with optimal weights (1:2:2:1 ratio) to achieve fourth-order accuracy. This Simpson's rule-like averaging cancels lower-order error terms through careful coefficient selection.

## III. Implementation

### A. Error Convergence Study

A convergence study was conducted to verify the correctness of the fourth-order Runge-Kutta (RK4) method implementation. A convergence study examines how the global error changes as the time step size is reduced. For a numerical method of known theoretical order, the error should decrease predictably with step size, even when the exact solution is unknown. This provides strong evidence that the method has been correctly implemented.

In this study, the rocket trajectory problem, including thrust, gravity, atmospheric drag, and varying mass, was solved over the time interval $t \in [0, 30]$ seconds. The global error at the final time was measured using the infinity norm, comparing each solution to a highly resolved reference solution computed with 12,800 steps. The problem was solved for 30 different step sizes, logarithmically spaced between 100 and 3200 total steps, to produce a smooth convergence curve.

The results were plotted on a log-log scale, showing the global error versus the step size. A straight-line fit to the data yielded an observed slope of approximately 3.88, which closely matches the theoretical expectation of fourth-order

accuracy for RK4. Minor deviations from an exact slope of 4.0 are expected due to nonlinear effects in the rocket model and finite step size artifacts at coarser resolutions. The convergence study confirms that the RK4 method is correctly implemented and accurately captures the rocket dynamics.

A summary of selected step sizes and their corresponding global errors is provided in Section III.B, where these results are used to inform step size selection for final simulations.

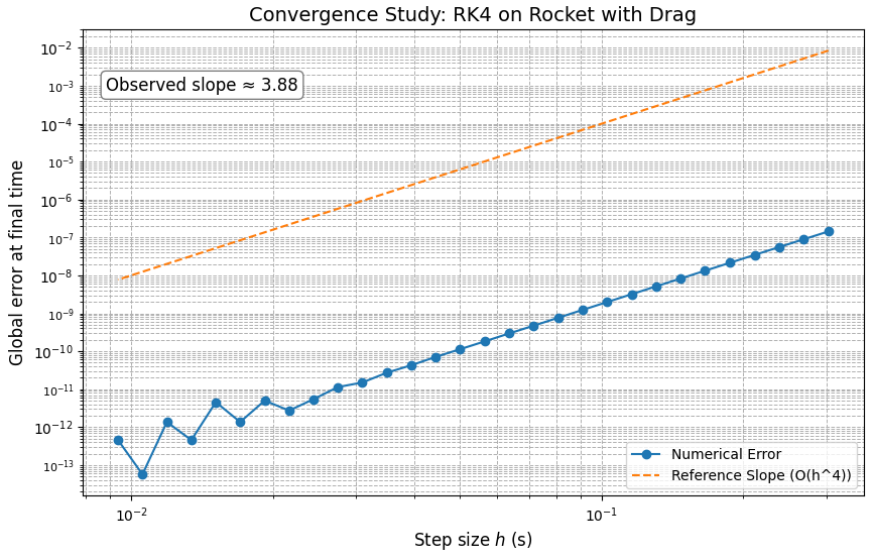


**Fig. 2 Log-log plot of global error vs. step size for RK4. The observed convergence rate is 3.88 (approximately 4.0), consistent with the method's theoretical fourth-order accuracy.**

When including burnout logic in our rocket equation, our RK4 method doesn't converge quite as nicely, but as you can see in Fig. 3, the convergence does trend towards 4.0 as the step size increases.

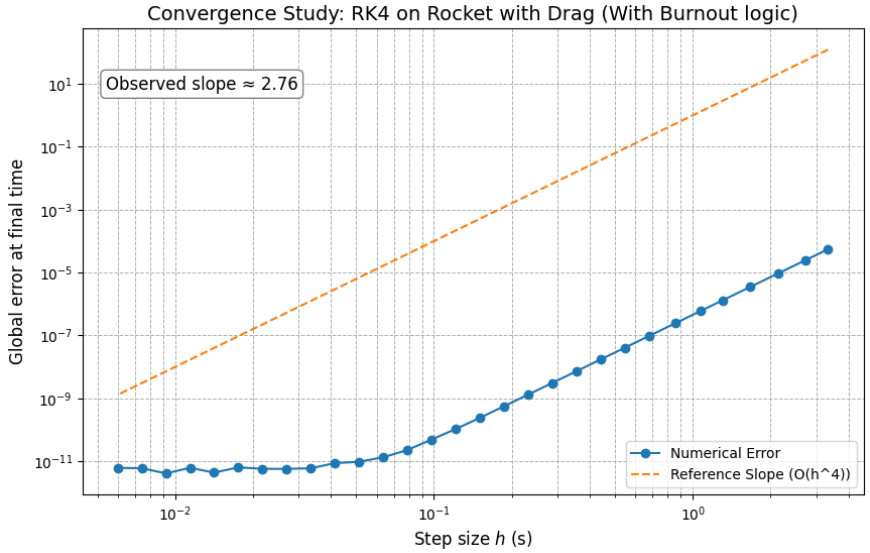


**Fig. 3 Log-log plot of global error vs. step size for RK4 with bunrout logic included. The observed convergence rate is 2.76, but trends towards 4 as the step size increases.**

This study provides strong evidence that our RK4 solver is implemented correctly and can reliably simulate the rocket's dynamics for various time step resolutions.

**B. Step Size Evaluation**

To select an appropriate time step for our final rocket simulations, we used the results of our convergence study to balance accuracy with computational efficiency. As discussed in Section III.A, the convergence plot confirmed that our RK4 implementation achieves fourth-order accuracy. However, choosing a time step also involves evaluating the computational cost associated with each level of accuracy.

Although smaller step sizes yield more accurate results, they also significantly increase the number of steps and the runtime required to complete the simulation. To better understand this trade-off, we evaluated a subset of step sizes and measured the global error at the final time against a highly resolved reference solution using 12,800 steps. We also recorded the number of RK4 steps used in each case. These results are summarized in Table 1.

| Step Size $h$ | Global Error (Infinity Norm) | Number of Steps |
|---|---|---|
| 0.3030 | 1.44e-07 | 100 |
| 0.1508 | 9.11e-09 | 200 |
| 0.0752 | 5.73e-10 | 400 |
| 0.0375 | 3.77e-11 | 801 |
| 0.0188 | 3.18e-12 | 1597 |

**Table 1 Global error at final time and number of RK4 steps for selected time step sizes.**

To further assess the cost of increasing accuracy, we measured the runtime required for each step size and plotted runtime versus global error on a log-log scale. As shown in Figure 4, the observed slope of approximately -0.22 indicates that runtime increases only modestly as accuracy improves. This confirms that the RK4 method remains computationally efficient, even for relatively small step sizes.

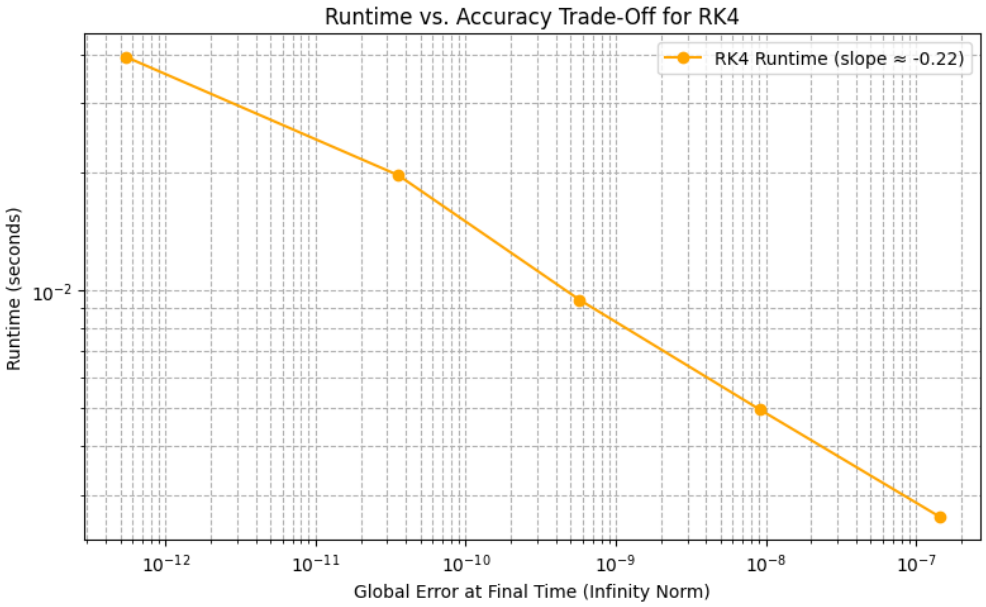


**Fig. 4 Runtime vs. global error for various RK4 step sizes. The observed slope of -0.22 shows that runtime increases slowly as error decreases.**

Based on these results, we selected a step size of $h = 0.0752$ for our final simulations. This value provides a global error on the order of $10^{-10}$, which is negligible relative to the physical effects being modeled, while also maintaining a reasonable computational cost.

Although the inclusion of burnout logic introduces a discontinuity in the thrust profile and causes the observed convergence rate to fall slightly below the ideal fourth-order behavior, the overall trend remains consistent with a

convergent method. As shown in Figure 3, the error continues to decrease as the step size is refined, and at our selected step size of $h = 0.0752$, the global error remains on the order of $10^{-9}$. This level of accuracy is sufficient for capturing the key physical dynamics of the system, and the time step remains appropriate even under more realistic simulation conditions.

# IV. Results

## A. Drag Factor impact on Percentage Error

Without loss of generality, the cross-sectional area and drag coefficient are constant multipliers of the square of velocity in the drag force. This means for error analysis, the following relation can be made to simplify the graphical analysis [5]:

$$F_{drag} = \frac{1}{2} C_D A \rho_0 e^{-h/H} v^2 \Rightarrow F_{drag} = k \rho_0 e^{-h/H} v^2. \tag{58}$$

Since the density of the air changes with altitude, this factor must be accounted for separately from the drag coefficient and cross-sectional area. The impact of modifying $k$ and $\rho_0$ can be seen visually in Figures 5 and 6.

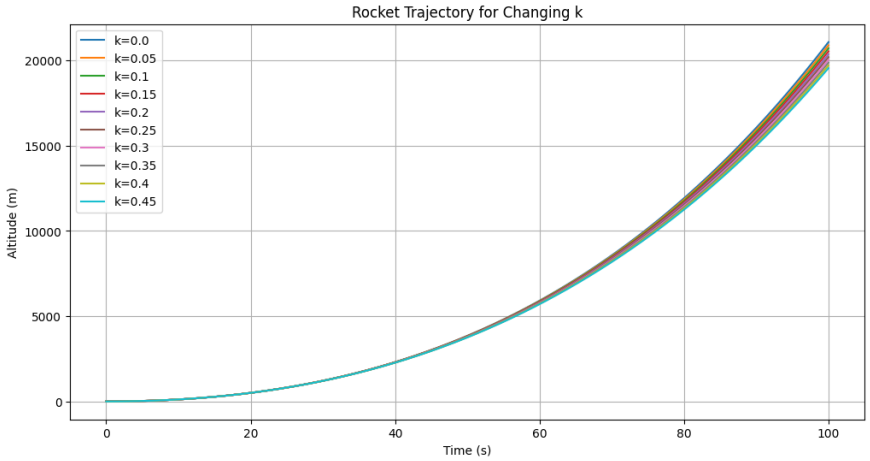


**Fig. 5 Rocket Trajectory with Changing Constant K**

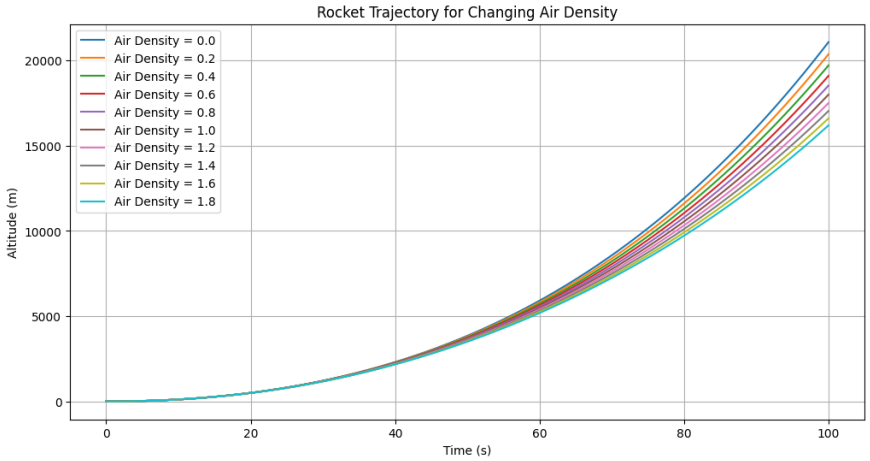


**Fig. 6 Rocket Trajectory with Changing Density $\rho_0$**

From the above figures, we can see the constants' effect on trajectory. It is obvious that a higher $k$ and $\rho_0$ both result in a higher drag throughout the entire phase of flight. This higher drag is seen in the above graphs as the rocket trajectory decreases with a higher drag constant, reaching lesser altitudes over the same time.

To answer our questions posed in Section I.B, we need to find the relation between drag factors and the error between these models [6].

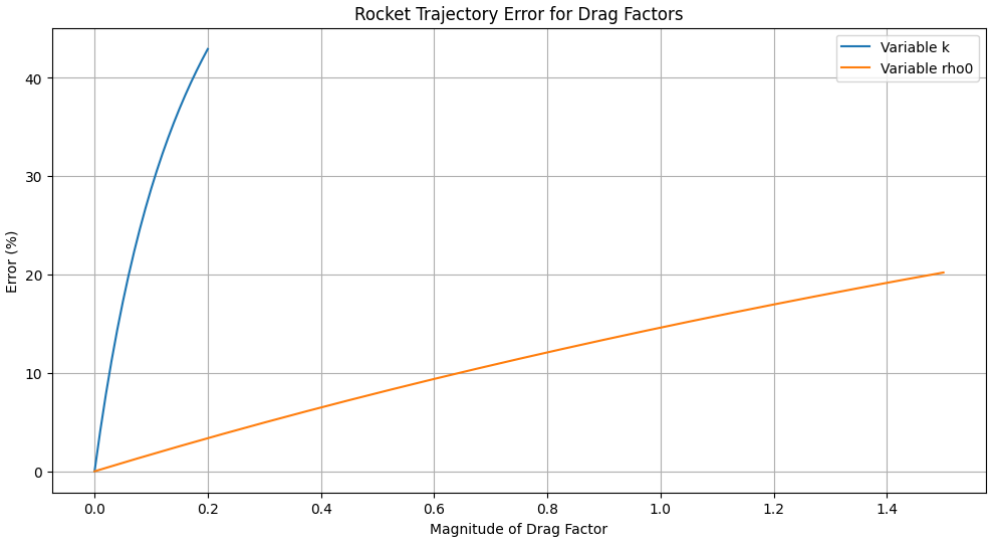


**Fig. 7 Error in Ideal Rocket Equation with Changing Drag Constants**

This plot begins with both constants, $k$, and $\rho_0$ set to zero, resulting in the convergence to 0% error seen in the bottom left corner. Then, as both $k$ and $\rho$ grow to and beyond the assumed values in our code, we can see the change. It is clear that a change in our $k$ will result in a much higher error than a change of the same magnitude to $\rho_0$. We can see that our max $k$ of 0.2 results in an error higher than 40%. This means that a change to reference area $A$ or to the drag constant $C_D$ will result in a far more ideal launch as compared to a change in $\rho_0$.

### B. Initial Condition Impact on Error

In this section, the extent to which the initial conditions impact the error between the drag model and the ideal no-drag model are examined.

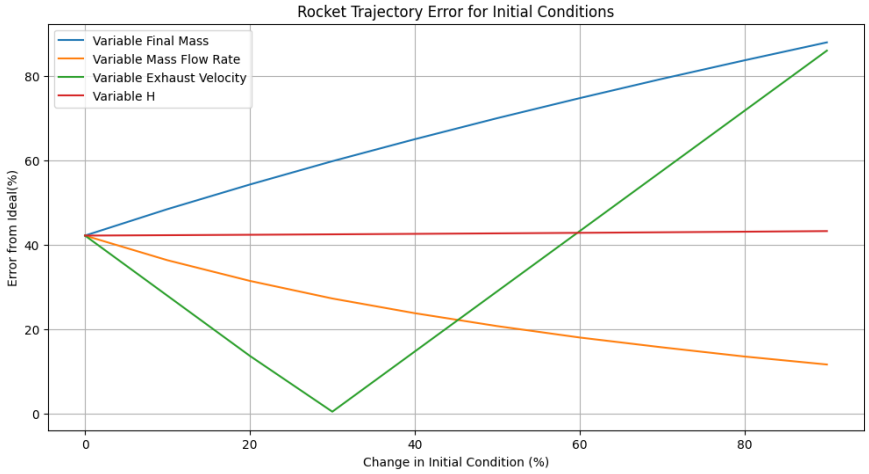


**Fig. 8 Error in Ideal Rocket Equation with Changing Initial Values**

The graph above was created by independently increasing each variable at a ten-percent increment of its so-called base condition until the variable is at 100% of its base-condition value. At each increment, the difference between the ideal solution with no drag and the numerical solution with drag, divided by the value for the ideal solution with drag was plotted. This way, each variable could have its impact on the variance from the ideal solution measured. As can be seen in the graph, a doubling in the final mass roughly doubles the change in variance from the ideal solution, showing a linear impact. Changing the exhaust velocity only appears to account for errors between the ideal and numerical models, but only to a point. It appears that a 30% increase in exhaust velocity perfectly accounts for this error. The slope of this line reveals the extent of its impact, as it has a degree of impact of roughly 1.125. The increase of mass flow rate appears to force the relationship to converge to its ideal value, which makes sense in theory as the full expulsion of energy at once would limit the time over which drag could take effect. In complete contrast to this, changing the value of $H$ has nearly no effect on the system. While these changes seem drastic, examining an initial variable's impact on the on the trajectories for which the error has been plotted in Figure 8 would be helpful.

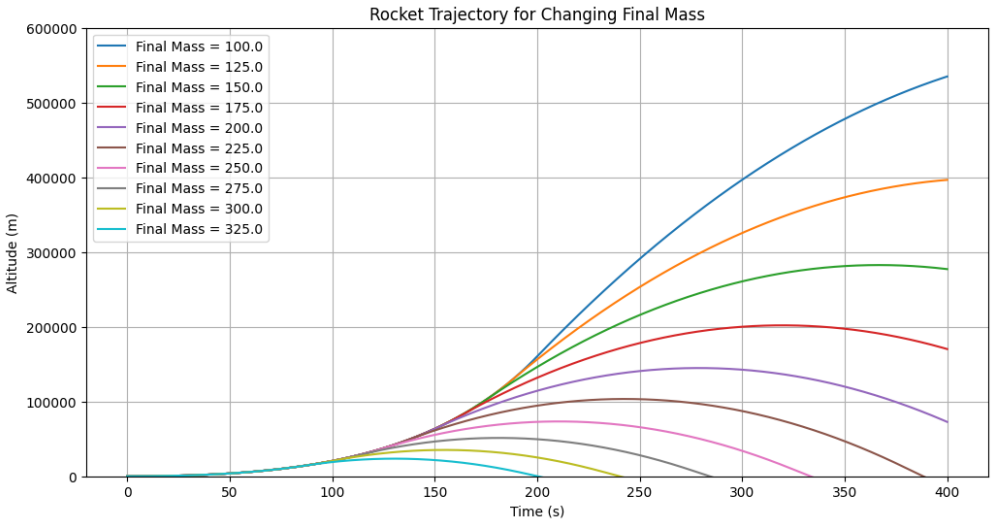


**Fig. 9 Trajectories of Rocket with Varying Final Mass**

With these trajectories plotted in 9, the significance in variance can be understood to be a natural consequence of the increase in mass ratio of the rocket. In keeping the initial mass constant, increasing the final mass of the rocket decreases the available fuel mass. Thus, less fuel can be burned, less thrust can be generated, and more mass is left after burnout, weighting the rocket down. The final mass of the rocket has a large impact on the trajectory compared to other variables.

## V. Group Member Roles

As a team, all five members of the group met together in the Aerolab to complete Section I: Dynamical System. Each member meaningfully contributed to the discussion and choice of our system as well as the numerical method we would use for the project. Thus, this section will be split evenly between all five members, giving each member roughly 5% completion. Distinctions and roles were defined for the remainder of the project, and the contributions are listed below.

**John Klis** completed the step size evaluation in Section III: Implementation. Combining this with his work on Section I, John completed roughly 18% of the project.

**Andrew Myers** completed all of Section II: Appropriate Numerical Method. After our group decided on what method to use, it was his role to gather all of the information about the method and its derivation. While this portion represents a quarter of the project length, its implementation is simpler when compared to Sections III and IV, thus decreasing its relative value. Andrew additionally worked on formatting the Acknowledgments and References sections. Combining this work with his work on Section I, Andrew completed roughly 26% of the project.

**Luke Brown** completed the convergence study in Section III. He also contributed to the Python implementation of RK4 and set up the GitHub repository. Combining this with his work on Section I, Luke completed roughly 19% of the project.

**Matthew Weippert** completed the code and report expository relating to the analysis of the results of our model, including Figures 5-8. The dynamical system importance and questions asked were also contributed by him. 19% of the total report was completed by him.

**Jack Stevenson** completed the answering of Question 1 in Section IV: Results. He also contributed to the Python implementation of RK4 and the subsequent plots. Combining this with his work on Section I, Jack completed roughly 18% of the project.

## VI. Reproducibility

A link to a GitHub repository containing our project code can be found here: https://github.com/lukecb13/ae370-rk4-rocket-simulation.

## Acknowledgments

We would like to acknowledge that artificial intelligence, AI, was used in the completion of this project, mainly through ChatGPT and DeepSeek. Thus, the references list may not be comprehensive of all of the sources that the AI drew from. These tools were used mainly in the derivation and the implementation of the numerical method. Throughout the entire process, AI helped to format our ideas in LaTeX and aided in plotting the results that answered the questions that we posed. The outputs from these models were verified for validity.

## References


[1] ChatGPT., Available: https://chatgpt.com/, [Accessed: March 31, 2025]

[2] Ideal Rocket Equation, Available: https://www1.grc.nasa.gov/beginners-guide-to-aeronautics/ideal-rocket-equation/, [Accessed: March 31, 2025]

[3] Sutton, G. P. and Biblarz, O., "Flight Performance," *Rocket Propulsion Elements 7th Ed.*, John Wiley & Sons, 2001, pp. 102-159

[4] DeepSeek., Available: https://chat.deepseek.com/, [Accessed: April 10, 2025]

[5] Han, S., Hwang, M., Lee, B., Ahn, J., and Tahk, M., "Analytic Solution of Projectile Motion with Quadratic Drag and Unity Thrust," *IFAC-PapersOnLine*, Vol. 49, No. 17, 2016, pp. 40-45, https://www.sciencedirect.com/science/article/pii/S240589631631480X

[6] Shape Effects on Drag, Available: https://www1.grc.nasa.gov/beginners-guide-to-aeronautics/shape-effects-on-drag/, [Accessed: April 10, 2025]