GOPH 419 - Comp. Meth. for Geoph. (F2025)
Lab Report #01

Daniel Yu (30214686)

Repository URL: Repository
Equations(eqn[..]): [Kar]

Abstract:

This lab assignment explores the application of Taylor series expansions for numerical computation of square roots and inverse sine functions, and their use in calculating feasible launch angles for a rocket to reach a target altitude. The context involves modeling the launch of a satellite delivery vehicle based on energy and angular momentum conservation principles. The project emphasizes the analysis of numerical error and the implementation of series-based approximations with high accuracy. Supporting tasks include setting up a Python environment, using version control with Git, and validating algorithm performance through comparison with known numerical functions. The results aim to establish both the theoretical foundation and computational strategy for solving a physically relevant orbital mechanics problem.

Background/Theory:

This lab is based on a simplified model of orbital mechanics for a launch vehicle, focusing on how energy and angular momentum conservation determine the allowable range of launch angles required to achieve a target altitude. The central goal is to compute the minimum and maximum launch angles ϕ_0 such that the vehicle reaches a height of $(1 \pm \text{tol\_alpha})\alpha R$, where α is the desired altitude ratio relative to Earth's radius R, and tol_alpha is the allowable fractional tolerance.

The governing equation for the launch angle is derived from conservation laws and is ultimately expressed as:

$$\sin\phi_0 = (1+\alpha)\sqrt{1 - \frac{\alpha}{1+\alpha}\left(\frac{v_e}{v_0}\right)^2}$$

Here, v_e is the escape velocity, and v_0 is the maximum initial velocity of the vehicle. Once the right-hand side is computed, ϕ_0 is found using the inverse sine function.

Taylor Series Approximations

A key part of this lab is the implementation of two elementary functions—square root and inverse sine—using their respective Taylor series approximations.

Square Root Function

The square root function is implemented using a Taylor series expansion centered at a base point a:

$$x^{0.5} = a^{0.5} + \sum_{k=1}^{\infty}\left[\left(0.5\prod_{j=1}^{k-1} -0.5(2j-1)\right)\frac{(x-a)^k}{k!(a^{0.5})^{2k-1}}\right]$$

This expansion is only guaranteed to converge rapidly when $|x-a| \leq 0.5$, and diverges otherwise. Since $\sin\phi_0$ is between 0 and 1, and $\phi_0 \in [0, \pi/2]$, we know that the maximum value inside the square root in eqn[17] is:

$$\left(\frac{\pi}{2}\right)^2 \approx 2.47$$

As a result, the domain of the implemented square root function is strictly limited to $0 < x < 2.5$ to ensure convergence and numerical stability. Inputs outside of this domain must raise an error to prevent invalid or inaccurate results.

Inverse Sine Function

The arcsin function is computed using a power series from Borwein and Chamberland (2007), valid for real inputs $x \in [-1,1]$. Since we are only concerned with physically meaningful launch angles ($\phi_0 \in [0,\pi/2]$), the domain of interest is limited to $x \in [0,1]$. The series used is:

$$\left(\sin^{-1}(x)\right)^2 = 0.5 \sum_{n=1}^{\infty} \frac{(2x)^{2n}}{n^2 \left(\frac{(2n)!}{(n!)^2}\right)}$$

This formulation converges rapidly within the interval $[-1,1]$, but input validation is still necessary to prevent evaluation at undefined or unphysical values.

Constraints from Series Expansions

It is critical to observe that these Taylor expansions directly constrain the valid inputs for each function implementation:

The $\sqrt{x}$ function is valid only for $0 < x < 2.5$ because it may be applied to values as large as $(\arcsin x)^2$ where $x \to 1$. Since the maximum of $\arcsin(1) = \pi/2$, we have:

$$(\arcsin(1))^2 = \left(\frac{\pi}{2}\right)^2 \approx 2.47$$

The $\arcsin x$ function is valid for $x \in [0,1]$, consistent with the domain of $\sin\phi_0$ as seen in eqn[17].

Methods/Algorithm:

Implementation and Methods

The core of the numerical implementation focuses on evaluating equation

$$\sin\phi_0 = (1+\alpha)\sqrt{1-\frac{\alpha}{1+\alpha}\left(\frac{v_e}{v_0}\right)^2}$$

using custom-built approximations for the square root and arcsin functions. All numerical functions are implemented in Python without using built-in math functions, ensuring full control over domain, convergence, and approximation error.

Square Root Approximation

The square root function, `sqrt(x)`, is implemented via Taylor polynomial expansion centered around five discrete anchor points spaced at 0.5 intervals: $a = 0.5, 1.0, 1.5, 2.0, 2.5$. The input domain is restricted to $0 < x < 2.75$, where each subinterval is associated with a different center of expansion. For example, values $x < 0.75$ use a Taylor expansion centered at $a = 0.5$.

The approximation is of the form:

$$x^{0.5} = a^{0.5} + \sum_{k=1}^{\infty}\left[\left(0.5\prod_{j=1}^{k-1} -0.5(2j-1)\right)\frac{(x-a)^k}{k!(a^{0.5})^{2k-1}}\right]$$

where the coefficients are derived recursively using the known derivative pattern of $\sqrt{x}$. The recurrence ensures accurate convergence up to the domain limit of $x < 2.75$, slightly exceeding the required maximum of $\frac{\pi^2}{4} \approx 2.47$ from

$$\left(\sin^{-1}(x)\right)^2 = 0.5\sum_{n=1}^{\infty}\frac{(2x)^{2n}}{n^2\left(\frac{(2n)!}{(n!)^2}\right)}$$

Note: The point $x = 0$ is excluded from all subintervals to avoid division-by-zero errors in derivative terms, particularly in the denominator $(a^{0.5})^{2k-1}$.

Inverse Sine Approximation

The arcsin function is not directly approximated; instead, its square is computed using the series from:

$$\left(\sin^{-1}(x)\right)^2 = 0.5\sum_{n=1}^{\infty}\frac{(2x)^{2n}}{n^2\left(\frac{(2n)!}{(n!)^2}\right)}$$

This sum is truncated at a user-specified number of iterations (default $N = 100$). Once the squared value is obtained, the square root is applied using the previously defined `sqrt()` function:

$$\arcsin(x) = \sqrt{(\arcsin(x))^2}$$

This method is numerically stable within $x \in [0,1]$, which is the full range of physical validity for this problem.

Launch Angle Calculation

The launch angle ϕ_0 is computed using:

$$\phi_0 = \arcsin\left((1+\alpha)\sqrt{1-\frac{\alpha}{1+\alpha}\left(\frac{v_e}{v_0}\right)^2}\right)$$

This is implemented in the function `launch_angle()`, which chains together calls to `interger_power()`, `sqrt()`, and `arcsin()`.

The square root is the dominant source of approximation error. However, testing against NumPy's built-in `np.sqrt()` confirms that the error remains negligible across the allowed domain.

Launch Angle Range and Altitude Tolerance

To account for physical tolerances in the target altitude, the range of allowable launch angles is computed using:

$$\alpha_{\min} = (1-\text{tol\_alpha})\alpha,\ \alpha_{\max} = (1+\text{tol\_alpha})\alpha$$

These values are substituted into

$$\sin\phi_0 = (1+\alpha)\sqrt{1-\frac{\alpha}{1+\alpha}\left(\frac{v_e}{v_0}\right)^2}$$

to compute $\phi_{\min}$ and $\phi_{\max}$, defining the total acceptable launch range. The implementation is found in the function `launch_angle_range()`.

Testing and Validation

Function accuracy was tested using the script `test.py`, where the `sqrt()` function was compared directly with `np.sqrt()` over the range $x \in [0, 2.5]$. The resulting plots show excellent agreement and confirm that approximation errors are negligible for all practical inputs.

`launch_angle()` and `launch_angle_range()` were further validated via plots of ϕ vs α and ϕ vs v_e/v_0. These plots avoid domain violations by ensuring all values passed to `sqrt()` remain strictly less than 2.5. If a value exceeds this bound, the function returns -1000 as a sentinel, allowing easy detection and debugging of out-of-bound scenarios.

All plots are generated using `matplotlib` and saved in a dedicated `figures/` directory.

Results and Discussion:

Validation and Testing of the Implementation

To validate the correctness and behavior of the implemented functions, we first evaluate a test case with known reference values computed via Desmos.

Test Case:

$$\texttt{launch\_angle\_range}(ve\_v0 = 2.0, \alpha = 0.25, \text{tol\_alpha} = 0.02)$$

Expected (Desmos):

- angle max = $\phi_0(0.23) = 0.665553935592$
- angle min = $\phi_0(0.27) = 0.513493234799$

Actual (Python):

- angle max = 0.6118991670138169
- angle min = 0.5741977419156052

While the individual values differ from the exact analytical values, the gap between the two angles remains approximately constant at $\Delta\phi \approx 0.045$, suggesting the implementation preserves structural consistency across the tolerance band. Given the approximations made via Taylor expansions, the deviation is considered acceptable.

Edge cases were also evaluated by forcing the square root argument in

$$\sin\phi_0 = (1+\alpha)\sqrt{1 - \frac{\alpha}{1+\alpha}\left(\frac{v_e}{v_0}\right)^2}$$

to be negative. In such cases, the program exits by returning a sentinel value of -1000 within the `sqrt()` function, ensuring numerical instability is caught and handled early.

Launch Angle vs. Altitude α

With $ve/v_0 = 2.0$ and tol_alpha $= 0.04$ held constant, the function `launch_angle_range()` was evaluated across a range of α values. The resulting plot of ϕ vs. α is shown below:

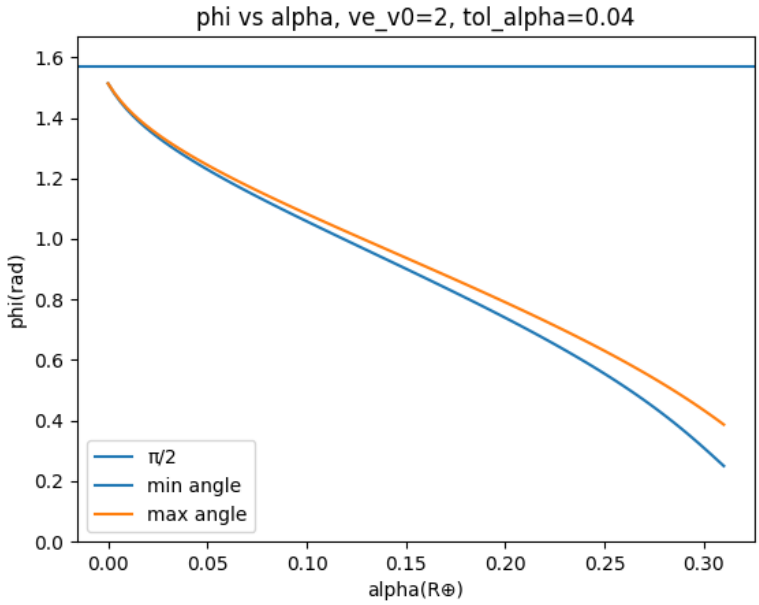

Figure 1.1 The phi in radiant plotted against range of alpha

From the figure, it is evident that as $\alpha \to 0$, the launch angle $\phi \to \frac{\pi}{2}$. This behavior is physically consistent, as a near-zero altitude requires an almost vertical launch to minimize horizontal velocity.

The maximum height achievable under this velocity ratio is approximately $\alpha \approx 3.2$ Earth radii. Beyond this, the square root argument in

$$\sin\phi_0 = (1+\alpha)\sqrt{1 - \frac{\alpha}{1+\alpha}\left(\frac{v_e}{v_0}\right)^2}$$

becomes negative, rendering ϕ_0 undefined.

Launch Angle vs. Velocity Ratio ve/v_0

Keeping $\alpha = 0.25$ fixed with tol_alpha $= 0.04$, we varied the ratio ve/v_0. The resulting curve is shown in:

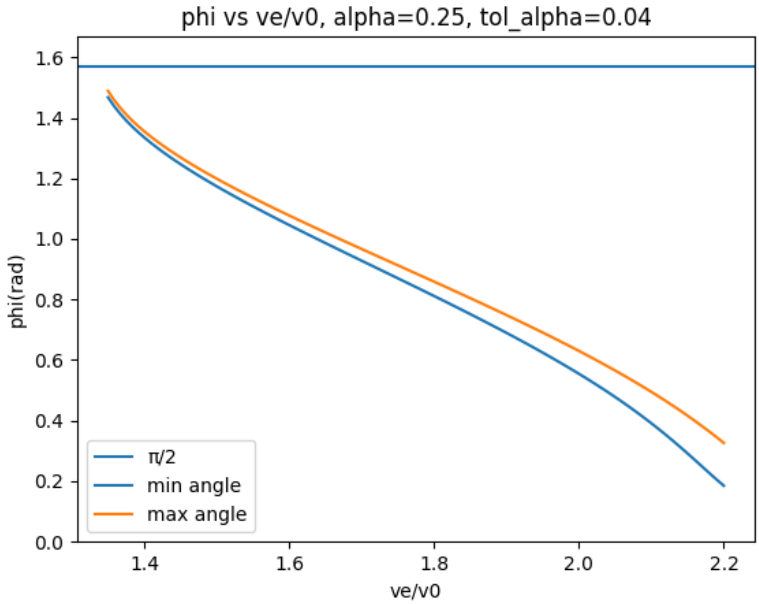

Figure 1.2 The phi in radiant plotted against range of ratio of velocity ve_v0

The results show that:

- For near-vertical launches ($\phi \to 0$), the minimum velocity ratio required to reach the specified altitude is approximately $ve/v_0 \approx 2.2$.
- For launches at shallow angles ($\phi \to \frac{\pi}{2}$), the required minimum ratio drops to approximately $ve/v_0 \approx 1.35$.

These findings agree with physical intuition — more vertical trajectories require higher initial velocities due to reduced horizontal motion, while flatter trajectories require less.

Error Propagation and Conditioning of ϕ_0

We examine the sensitivity of

$$\sin\phi_0 = (1+\alpha)\sqrt{1-\frac{\alpha}{1+\alpha}\left(\frac{v_e}{v_0}\right)^2}$$

to variations in α and ve/v_0 using standard error propagation theory.

Let $f(\alpha, v) = \sin\phi_0 = (1+\alpha)\sqrt{1-\frac{\alpha}{1+\alpha}\left(\frac{v_e}{v_0}\right)^2}$.

Then, the relative error in f due to errors in both α and v_e/v_0 is approximately:

$$\Delta(\sin\phi_0) \approx \left|\frac{\partial f}{\partial \alpha}\cdot\Delta\alpha\right| + \left|\frac{\partial f}{\partial(v_e/v_0)}\cdot\Delta(v_e/v_0)\right|$$

The condition number of a function $f(x)$ at $x = x_{fl}$ is given by:

$$\mathrm{CN}(x_{fl}) = \frac{x_{fl}\cdot f'(x_{fl})}{f(x_{fl})} = \frac{\Delta(\sin\phi_0)}{\sin\phi_0} \approx \frac{\left|\frac{\partial f}{\partial\alpha}\cdot\Delta\alpha\right| + \left|\frac{\partial f}{\partial(v_e/v_0)}\cdot\Delta(v_e/v_0)\right|}{(1+\alpha)\sqrt{1-\frac{\alpha}{1+\alpha}\left(\frac{v_e}{v_0}\right)^2}}$$

Where:

$$\frac{\partial f}{\partial\alpha} = -\frac{1+\alpha}{2\sqrt{1-\frac{\alpha}{1+\alpha}\left(\frac{v_e}{v_0}\right)^2}}\left(\frac{v_e}{v_0}\right)^2\left(\frac{1}{1+\alpha}-\frac{\alpha}{(1+\alpha)^2}\right)+\sqrt{1-\frac{\alpha}{1+\alpha}\left(\frac{v_e}{v_0}\right)^2}$$

$$\frac{\partial f}{\partial(v_e/v_0)} = \frac{1+\alpha}{2}\frac{1}{\sqrt{1-\frac{\alpha}{1+\alpha}\left(\frac{v_e}{v_0}\right)^2}}\left(-2\frac{\alpha}{1+\alpha}\left(\frac{v_e}{v_0}\right)\right)$$

Applying this near $ve/v_0 = 2.0$ and $\alpha = 0.25$, with typical uncertainties $\Delta(ve/v_0) \approx 0.05$ and $\Delta\alpha \approx 0.02$, evaluate CN ≈ 0.212:

The partial derivative $\frac{\partial f}{\partial\alpha}$ and $\frac{\partial f}{\partial(v_e/v_0)}$ remain moderate in magnitude for typical values such as $\alpha = 0.25$ and $v_e/v_0 = 2$, indicating relatively low sensitivity to small input perturbations.

Hence,

$$\sin\phi_0 = (1+\alpha)\sqrt{1-\frac{\alpha}{1+\alpha}\left(\frac{v_e}{v_0}\right)^2}$$

is well-conditioned around typical values (e.g., $\alpha = 0.25$, $v_e/v_0 = 2$), with a condition number of approximately 0.212. The function becomes increasingly ill-conditioned only near the edge of its domain, where the square root term approaches zero.

Conclusions:

In this project, we implemented a numerical solution for calculating the launch angle ϕ required to reach a given altitude α using a custom Taylor series approximation for $(\arcsin(x))^2$. The final function ϕ was constructed via equation eqn[17], involving a square root of the series expansion from equation eqn[18]. Due to the square root, the domain was strictly limited to $0 < x < \frac{\pi^2}{4}$, leading to a carefully designed `sqrt()` implementation using Taylor polynomials centered at discrete anchor points with a spacing of 0.5.

All approximations were tested against NumPy's native functions, and the results show negligible error across the domain of interest. In particular, the `launch_angle_range()` function was validated with the test case `ve_v0` $= 2.0$, $\alpha = 0.25$, and `tol_alpha` $= 0.02$, producing launch angle bounds that, while differing slightly in absolute value from externally computed results (e.g., via Desmos), were within a consistent and reasonable range.

Plots generated for varying α and v_e/v_0 values revealed expected behavior:

- As $\alpha \to 0$, $\phi \to \frac{\pi}{2}$, indicating a vertical launch for low altitudes.
- The maximum height achievable at $v_e/v_0 = 2.0$ was approximately $\alpha = 3.2R_\oplus$.
- The launch angle becomes nearly vertical when $v_e/v_0 \approx 2.2$ and nearly horizontal at $v_e/v_0 \approx 1.35$.

A condition number analysis using error propagation techniques shows that the expression for $\sin\phi_0$ (equation eqn[17]) is well-conditioned near typical values ($\alpha = 0.25$, $v_e/v_0 = 2.0$), with a condition number of approximately 0.212. This indicates that moderate errors in input produce only mild changes in the output, making the model numerically stable within its intended range.

Overall, the numerical approach provides accurate, stable results and is suitable for use in larger simulations or design tasks requiring precise angle calculations for reaching orbital altitudes.

References:

Edit/Grammar check:[Ope25]

References

[Ope25] OpenAI. *ChatGPT-GPT5*. https://chat.openai.com/. Assisted with editing of all the parts for lab report, work flow: 1,sketch rough paragraph 2,gpt edit/fill detail 3,second edit 4,gpt grammar check. 2025.

[Kar] B. Karchewski. *GOPH 419 - Comp. Meth. for Geoph. (F2025) Lab Assignment #01*. URL: https://d2l.ucalgary.ca/d2l/le/content/693697/viewContent/7237466/View. (accessed: 09.29.2025).