Comparison of OpenMDAO and AeroSandbox for Trajectory Optimization

Devin Mroz and Kabir Khwaja

December 16, 2025

---

1 Abstract

This study compares Dymos and AeroSandbox for trajectory optimization using a two-dimensional rocket ascent problem with identical dynamics, constraints, and optimizer settings. Both frameworks solve a minimum-time ascent to orbit via direct collocation and IPOPT, differing in transcription schemes and derivative computation. Results across multiple mesh sizes show that Dymos provides higher accuracy and stricter constraint enforcement, while AeroSandbox achieves significantly lower computational cost and faster convergence. Trajectory and convergence comparisons highlight trade-offs between solution fidelity and efficiency, indicating that framework selection should depend on whether accuracy or speed is the primary priority.

2 Background and motivation

Trajectory optimization is central to aerospace engineering, particularly for launch vehicle ascent, where thrust direction, mass depletion, and vehicle motion must be coordinated to meet mission objectives such as reaching a target altitude and velocity or achieving orbit while minimizing time or propellant usage. Because propulsion resources and performance margins are limited, even modest improvements in ascent trajectories can lead to significant gains in payload capability and overall mission efficiency.

Ascent trajectory optimization is inherently multidisciplinary, coupling translational dynamics, propulsion, and control. Thrust magnitude and direction determine vehicle acceleration in both the vertical and horizontal directions, while propellant consumption reduces vehicle mass and feeds back into the equations of motion. These interactions form a coupled, nonlinear system that must be solved simultaneously rather than sequentially. Even in simplified two-dimensional settings, this structure makes ascent optimization a representative test case for multidisciplinary design and optimization (MDO) methods.

In this project, we explore the applicability of MDO methods to a two-dimensional rocket ascent model. While this is simplified relative to full six-degree-of-freedom launch vehicle simulations, the 2D model captures essential trajectory optimization features, including pitch control, nonlinear coupled dynamics, mass depletion, and boundary constraints associated with orbital insertion. By excluding three-dimensional effects, detailed aerodynamics, and

guidance complexity, the problem remains computationally efficient and interpretable, enabling focused study of solver behavior, derivative computation, and numerical robustness.

This project compares two modern trajectory optimization frameworks: Dymos [1] and AeroSandbox [2]. Dymos integrates direct collocation within a general-purpose MDO architecture from OpenMDAO, offering strong support for analytic derivatives, sparsity exploitation, and scalable multidisciplinary coupling. AeroSandbox emphasizes rapid development and flexibility through CasADi-based automatic differentiation and concise problem formulations. By holding the physical model constant, the comparison isolates differences in derivative calculation and solver performance, providing insight into how framework architecture influences optimization behavior and informing tool selection for future trajectory optimization applications.

3 Problem formulation

3.1 Optimization problem statement

This project considers a two-dimensional rocket ascent into orbit, where the goal is to determine an optimal pitch angle history that enables the vehicle to reach a specified altitude while minimizing time in ascent. A diagram of an example trajectory is shown in Fig. (1).

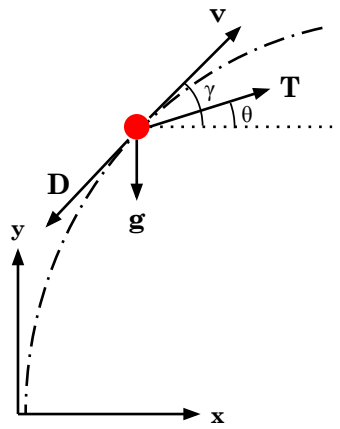

Figure 1: Example 2D rocket ascent trajectory.

The system state vector is defined as

$$x(t) = \begin{bmatrix} x(t) \\ y(t) \\ v_x(t) \\ v_y(t) \\ m(t) \end{bmatrix}$$

where $x(t)$ and $y(t)$ denote horizontal and vertical position, $v_x(t)$ and $v_y(t)$ denote horizontal and vertical velocity, and $m(t)$ is the vehicle mass. The control variable is the pitch angle $\theta(t)$, defined as the angle between the thrust vector and the local horizontal direction. Thrust magnitude T is assumed constant throughout the ascent.

The ascent dynamics are governed by the following ordinary differential equations:

$$\dot{x}(t) = v_x(t)$$

$$\dot{y}(t) = v_y(t)$$

$$\dot{v_x}(t) = \frac{T}{m(t)}\cos(\theta(t)) - \frac{\rho C_D A}{2m(t)} v(t)^2 \cos(\gamma)$$

$$\dot{v_y}(t) = \frac{T}{m(t)}\sin(\theta(t)) - g - \frac{\rho C_D A}{2m(t)} v(t)^2 \sin(\gamma)$$

$$\dot{m}(t) = -\frac{T}{I_{sp} g_0}$$

where

$$v(t) = \sqrt{v_x(t)^2 + v_y(t)^2}$$

$$C_D = 0.5,\ A = 7.069\ \text{m}^2$$

The constant $g = 9.80665$ m/s is the gravitational acceleration and $I_{sp} = 265.2$ s denotes the specific impulse. The objective is to minimize the total ascent time required to reach orbit:

$$\min[t_f]$$

where t_f is the final time of the ascent phase. The optimization problem is subject to the following constraints:

- Initial conditions:

$$x(0) = 0,\ y(0) = 0,\ v_x(0) = 0,\ v_y(0) = 0,\ m(0) = 117000\ \text{kg}$$

- Terminal altitude constraint:

$$y(t_f) \geq h_{orbit} = 185\ \text{km}$$

- Orbital velocity constraint:

$$v_x(t_f) \geq v_{orbit} = 7796.6961\ \text{m/s}$$

$$v_y(t_f) = 0$$

- Dynamic constraints - state trajectories must satisfy the equations of motion for all $t \in [0, t_f]$.

Since the ascent is modeled using multiple phases, continuity constraints enforce consistency of all state variables across phase boundaries. Minimum mass bounds are not enforced because the constant thrust ensures fuel is not depleted, and since we are minimizing time, an additional mass bound would be an unnecessary constraint.

3.2 Problem classification

The 2-dimensional rocket ascent problem considered in this study is formulated as a continuous-time optimal control problem with nonlinear dynamics and constraints. From an MDO perspective, the problem is classified as:

- A nonlinear programming (NLP) problem after transcription.
- Nonconvex due to nonlinear dynamics and trigonometric control terms.
- Constrained optimization with both equality and inequality constraints.
- Gradient-based optimization enabled by analytic or automatic derivatives.

3.3 Description of models and coupling

The 2D rocket ascent problem is modeled as a tightly coupled multidisciplinary system consisting of flight dynamics, propulsion, and control. These disciplines are not independent as they interact continuously through shared state variables and governing equations, requiring a simultaneous solution within the trajectory optimization framework.

The flight dynamics model describes the planar translational motion of the vehicle under the influence of thrust, gravity, and aerodynamic drag. The vehicle's accelerations are determined by the net force acting on the vehicle, with the aerodynamic drag modeled as a quadratic function of the vehicle's velocity and acts opposite the direction of motion, introducing nonlinear coupling between the horizontal and vertical velocity components.

The propulsion model provides constant thrust magnitude and governs mass depletion through a fixed specific impulse. The thrust enters the dynamics through both magnitude and direction determined by the pitch angle. The reduction of mass over time directly influences acceleration, amplifying the coupling between propulsion and dynamics.

The control model specifies the time history of the pitch angle, which serves as the primary design variable, and direction determines the direction of the thrust vector. These three models are strongly and bidirectionally coupled. Vehicle mass affects acceleration, acceleration determines velocity, velocity determines drag, and drag feeds back into acceleration. The pitch angle control influences all force components simultaneously, making the

system highly sensitive to control history. As a result, the coupled system cannot be solved sequentially and must be solved simultaneously.

Within both Dymos and AeroSandbox, this coupling is handled by enforcing the full set of nonlinear equations of motion as constraints at collocation points. All states and controls are solved simultaneously, allowing the optimizer to account for the interactions between dynamics, propulsion, and control throughout the ascent trajectory.

3.4 Proposed optimization algorithm

In this study, the nonlinear programming problems resulting from trajectory transcription are solved using IPOPT (Interior Point OPTimizer) in both Dymos and AeroSandbox. IPOPT is a widely used, open-source, gradient-based optimizer designed for large-scale constrained nonlinear optimization, making it well suited for direct collocation formulations of optimal control problems. The use of a common optimizer ensures that both frameworks solve the exact same mathematical model, enabling a fair and controlled comparison. Differences in optimization behavior therefore arise from how derivatives are computed (analytic derivatives in Dymos versus automatic differentiation in AeroSandbox) and from the choice of collocation schemes, with Dymos employing Legendre–Gauss–Lobatto collocation and AeroSandbox using a uniform linear time discretization. This approach isolates the impact of transcription and derivative handling on solver performance and numerical robustness rather than differences in the underlying physics or optimization algorithm.

4 Investigation of the Coupled Models

Prior to analyzing optimization results, it is important to examine the numerical and structural properties of the coupled trajectory optimization problem. The behavior of gradient-based solvers is strongly influenced by factors such as variable bounds, problem modality, derivative quality, and numerical conditioning. This section outlines the investigations performed to characterize these properties and to ensure that differences observed between frameworks can be attributed to transcription and derivative handling rather than unintended modeling artifacts.

4.1 Variable Bounds and Scaling

Bounds are applied to state and control variables to ensure physical realism and improve numerical conditioning. The pitch angle is limited to avoid unrealistic thrust directions, while altitude, velocity, and mass are constrained through initial and terminal conditions. Careful attention is given to scaling differences among states, as poor scaling can produce

ill-conditioned Jacobians and slow convergence. Variable magnitudes and nondimensionalization are therefore chosen to maintain consistent conditioning in both frameworks.

4.2 Modality

The 2D rocket ascent problem is inherently nonconvex due to nonlinear dynamics, trigonometric control terms, aerodynamic drag, and free final time. Multiple locally optimal trajectories may exist depending on the pitch angle history and timing of the gravity turn. This motivates the use of consistent initial guesses and identical problem formulations when comparing frameworks, ensuring that differences arise from numerical implementation rather than convergence to different local minima.

4.3 Choice of Coupled Solver Formulation

Because the flight dynamics, propulsion, and control models are strongly coupled, all state variables and controls are solved simultaneously within the nonlinear programming problem, with the governing equations enforced as equality constraints at collocation points. This formulation avoids the convergence difficulties associated with sequential approaches and is well suited to gradient-based optimization with nearly exact derivative information.

4.4 Numerical Noise and Discretization Effects

Although the governing equations are deterministic and smooth, numerical noise can arise from time discretization, constraint enforcement, and finite resolution of the control history. The choice of collocation scheme influences how accurately the dynamics are represented and how sensitive the NLP is to perturbations in the design variables.

4.5 Solver Performance Metrics

To enable consistent comparison, solver performance is evaluated using metrics such as convergence reliability, iteration count, and sensitivity to initial guesses. These metrics provide insight into how effectively each framework handles the coupled dynamics and constraints, independent of the final optimal solution. Performance is assessed under identical problem definitions and solver settings to isolate the effects of transcription and derivative computation.

4.6 Derivative Computation

Accurate first-order derivatives are essential for the performance of interior-point methods such as IPOPT. In this study, Dymos computes derivatives using a complex-step approach

within the OpenMDAO framework and explicitly exploits Jacobian sparsity, while AeroSandbox relies on CasADi's automatic differentiation to generate exact derivatives symbolically. Differences in derivative computation and Jacobian structure directly affect convergence behavior, robustness, and scalability. This investigation examines how each framework's derivative strategy interacts with the tightly coupled ascent dynamics.

5 Optimization of Simplified Problem

To enable a controlled investigation of solver behavior and transcription effects, a simplified version of the 2D rocket ascent problem is considered prior to analyzing the full problem. The simplified formulation retains the same governing equations, objective, and boundary conditions, but reduces problem complexity by limiting the temporal resolution of the control and state discretization. This reduction allows numerical behavior to be examined without the effects of large problem size or excessive degrees of freedom. The simplified problem is intended to expose sensitivity to discretization choices, derivative computation, and constraint enforcement, rather than to produce a high-fidelity ascent trajectory.

All problem parameters, bounds, and solver settings are held consistent between Dymos and AeroSandbox to ensure a fair comparison. The same mathematical model and objective function are used in both frameworks, with differences arising only from the collocation schemes and derivative computation methods. This simplified optimization therefore serves as an initial benchmark for assessing convergence behavior and numerical robustness before proceeding to higher-resolution formulations.

5.1 Simplified Problem Results

In order to get a fair comparison between AeroSandbox and Dymos for the simplified problem, we selected the smallest number of design variables that allowed both frameworks to converge to a feasible optimum, resulting in 61 design variables. For AeroSandbox, this corresponded to 10 collocation points, or discrete time locations at which the equations of motion are enforced, while for Dymos this required 11 collocation points to achieve the same number of design variables. Because this discretization leads to a high-dimensional design space, traditional contour plots are not informative and are therefore omitted; instead, results are evaluated using trajectory histories and convergence metrics. The results of this comparison are presented below.

The simplified problem results highlight a clear tradeoff between computational efficiency and solution accuracy when comparing AeroSandbox and Dymos. AeroSandbox demonstrated significantly faster performance, converging in 20 IPOPT iterations compared to 30 iterations for Dymos, with a wall-clock time of about 0.49 seconds versus about 3.60 seconds. Time spent inside IPOPT followed a similar trend, with AeroSandbox requiring about 0.24

Table 1: Comparison of Optimization Metrics for the Simplified 2D Rocket Ascent Problem

Metric	**AeroSandbox**	**Dymos**
Final time t_f (s)	142.777130	142.3127295
Final altitude $y(t_f)$ (m)	185000.0485	185000.0000
Final speed $v_x(t_f)$ (m/s)	8214.0632	7796.6961
Final mass $m(t_f)$ (kg)	1712.09516	2087.08336
Wall-clock time (s)	0.49352	3.60465
IPOPT iterations	20	30
Objective evaluations	21	31
Gradient evaluations	21	31
Time spent in IPOPT (s)	0.238	0.685

seconds compared to roughly 0.69 seconds for Dymos. Despite this speed advantage, the optimal ascent time obtained by AeroSandbox was slightly slower—by less than half a second—than that obtained using Dymos. The number of objective and gradient evaluations scaled consistently with iteration count in both frameworks, indicating comparable optimizer usage patterns.

While AeroSandbox converged more quickly, Dymos produced a notably more accurate and efficient solution. Dymos satisfied the terminal altitude constraint exactly, achieving the prescribed orbital altitude of 185,000 m, whereas AeroSandbox slightly overshot this value. More importantly, Dymos met the orbital velocity requirement precisely, converging to the specified value of 7,796.6961 m/s, while AeroSandbox exceeded this target with a final speed of 8,214.0632 m/s. This overshoot translated directly into increased propellant consumption: the final mass achieved by Dymos was roughly 2,087.1 kg, compared to only about 1,712.1 kg for AeroSandbox, representing a fuel savings of over 300 kg. These differences indicate that Dymos more effectively enforced boundary constraints and identified a more efficient ascent profile within the same mathematical model.

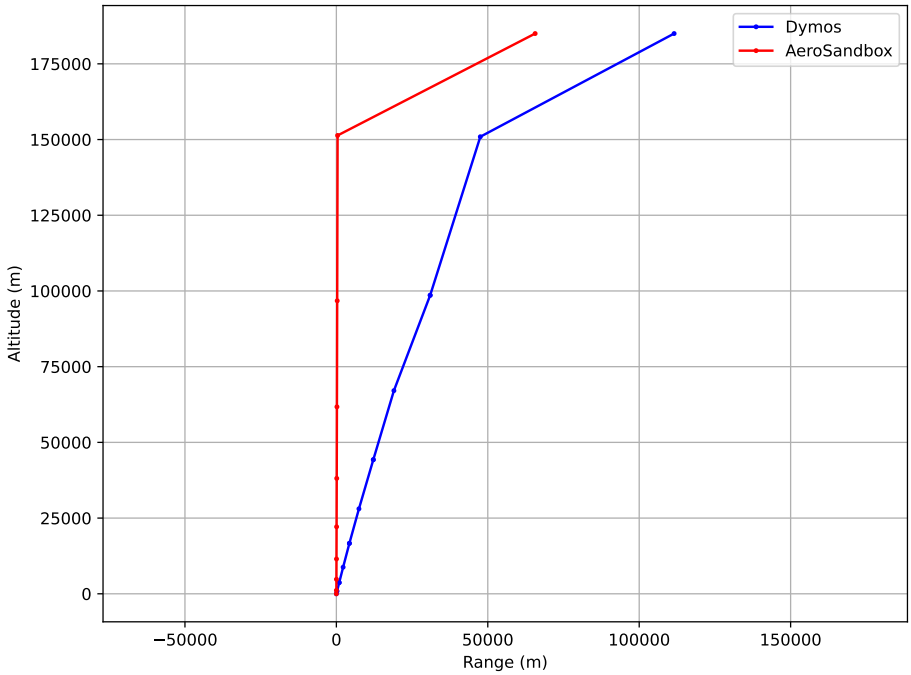

Figure 2: Comparison of AeroSandbox and Dymos optimized 2D trajectories for the simplified problem.

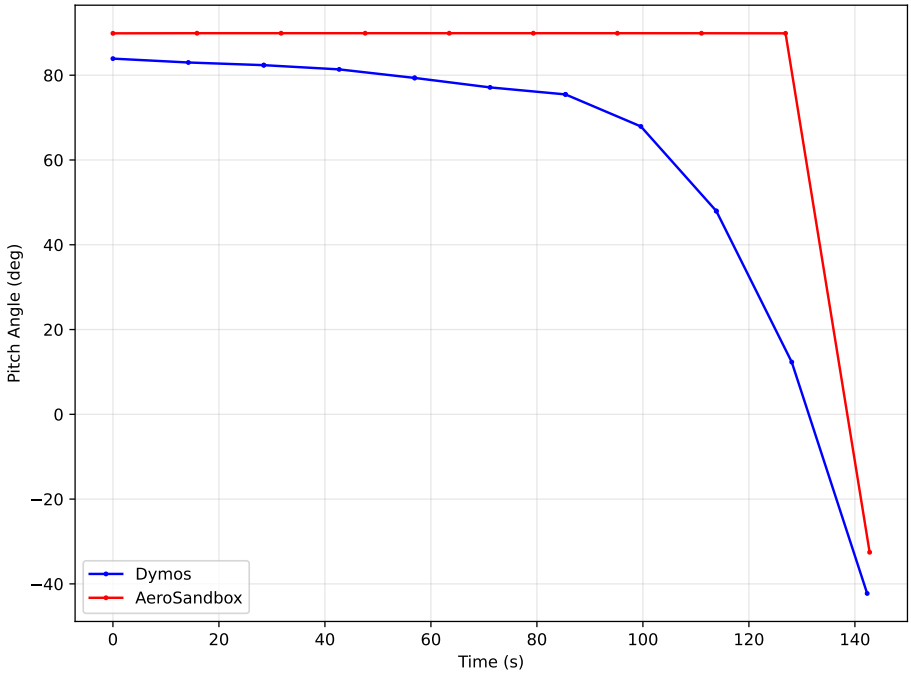

Figure 3: Comparison of AeroSandbox and Dymos optimized thrust angle histories for the simplified problem.

The trajectories produced are shown in Fig. (2), and the pitch controls over time are shown in Fig. (3). The differences in trajectory and control histories further illustrate the contrasting numerical behavior of the two frameworks. AeroSandbox's solution exhibits an aggressive strategy in which the vehicle ascends nearly vertically for most of the trajectory, followed by a sharp pitch-over maneuver near the final segment. In contrast, the Dymos solution shows a slightly smoother and more gradual gravity-turn-like behavior, initiating a slight pitch earlier in the ascent and transitioning progressively toward horizontal flight. This smoother control profile enables Dymos to satisfy terminal constraints more precisely while avoiding excess acceleration late in the trajectory.

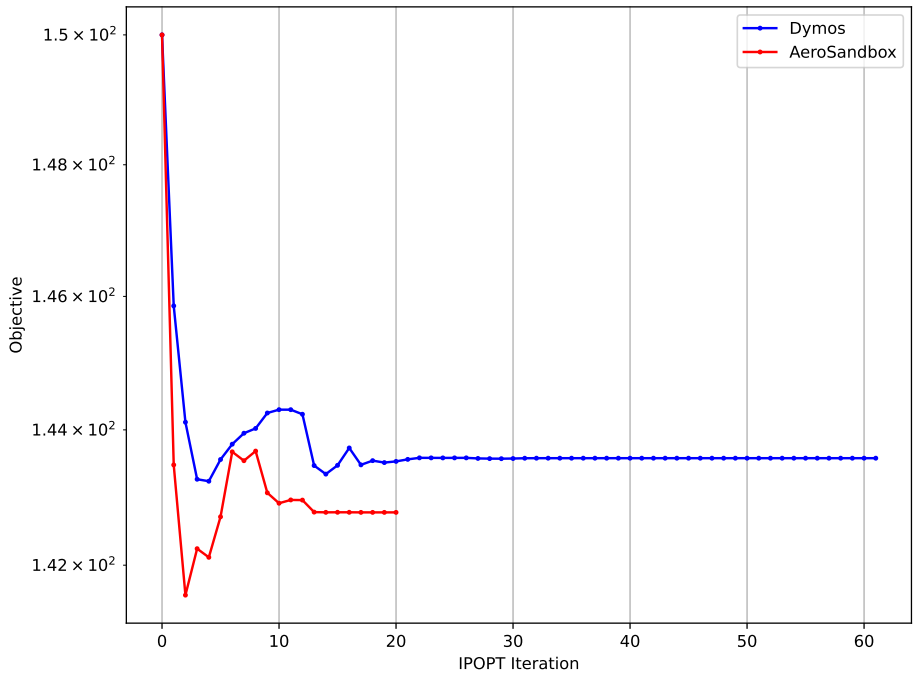

Figure 4: Convergence plot comparison of AeroSandbox and Dymos for the simplified problem, showing objective value (final ascent time) versus iterations.

Convergence histories shown in Fig. (4) provide additional insight into solver behavior. Both frameworks exhibited transient violations of constraints during the optimization process, as evidenced by objective values temporarily dropping below the physically feasible optimum before rising to the final solution. However, AeroSandbox displayed larger overshoots during these phases, ultimately converging to a less optimal solution. Dymos, by comparison, exhibited smaller deviations and more controlled corrections, allowing it to settle at a superior optimum. Notably, AeroSandbox reached the constraint-violating regime earlier in the iteration process, reflecting faster early convergence but reduced robustness near feasibility boundaries.

Overall, the simplified problem demonstrates that AeroSandbox excels in rapid conver-

gence and low computational cost, making it attractive for exploratory studies and rapid prototyping. Dymos, while computationally more expensive, exhibits greater robustness and accuracy in constraint enforcement, leading to more efficient and physically consistent solutions. These results suggest that the choice between frameworks depends strongly on whether speed or solution fidelity is the primary priority, a distinction that becomes increasingly important for higher-fidelity trajectory optimization problems. More robust testing is explored in the next section for more complex problems to verify these initial conclusions.

6 Optimization results

This section presents optimization results for the 2D rocket ascent problem using AeroSandbox and Dymos across multiple mesh sizes (mesh size meaning the number of design variables). With identical problem formulations and a common optimizer, the comparison focuses on how solution accuracy, convergence behavior, and computational cost scale with the number of design variables. Results include error, iteration count, wall time, final time, and final mass trends, as well as trajectory, thrust angle, and convergence plots. Together, these results highlight the impact of discretization and transcription choices on optimization performance in each framework.

6.1 Optimizer Performance

The L2 error convergence is shown in Fig. (5), calculated using Eq. (1)

$$.e_{L2} = \sqrt{(x_{truth} - x_i)^2 + (y_{truth} - y_i)^2} \tag{1}$$

The error convergence graph shows that both AeroSandbox and Dymos exhibit decreasing error as the number of design variables increases, consistent with the expectation that finer meshes better approximate the "true" trajectory defined by the finest examined mesh. Across all mesh sizes, Dymos achieves lower error than AeroSandbox, indicating higher accuracy per design variable. However, Dymos data is limited to meshes with fewer than 10^3 design variables, as larger meshes led to prohibitively long run times exceeding ten minutes. In contrast, AeroSandbox maintained feasible runtimes at higher mesh resolutions, allowing error trends to be observed across a wider range of design variables.

The iterations versus number of design variables plot in Fig. (6) shows that, in general, both AeroSandbox and Dymos require more iterations as the mesh is refined and the number of design variables increases. Except for the coarsest mesh, Dymos consistently exhibits a higher iteration count than AeroSandbox, reflecting its stricter enforcement of collocation and more accurate derivative computations. AeroSandbox displays an interesting behavior where the iteration count temporarily decreases for one or two intermediate mesh sizes before

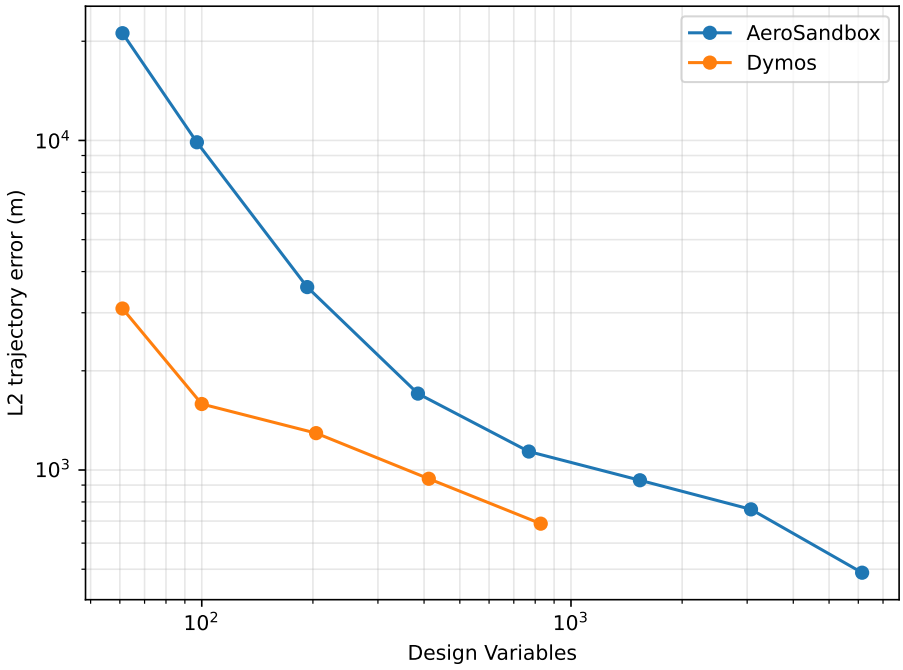

Figure 5: Error convergence of both frameworks versus number of design variables, corresponding with increasing mesh size. Error was measured as the combined difference from the trajectory position values to the "true" optimal trajectory, which was from the finest mesh size we examined.

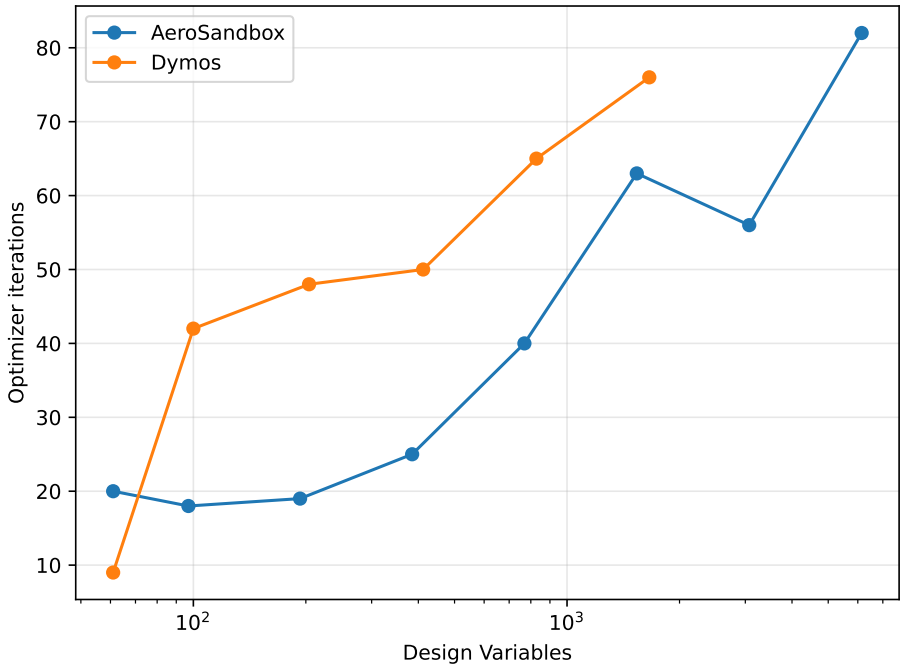

Figure 6: Iterations versus number of design variables for both frameworks, captured across all mesh sizes.

resuming the overall increasing trend. This non-monotonic behavior likely reflects interactions between mesh resolution, transcription, and the optimizer's step size adjustments.

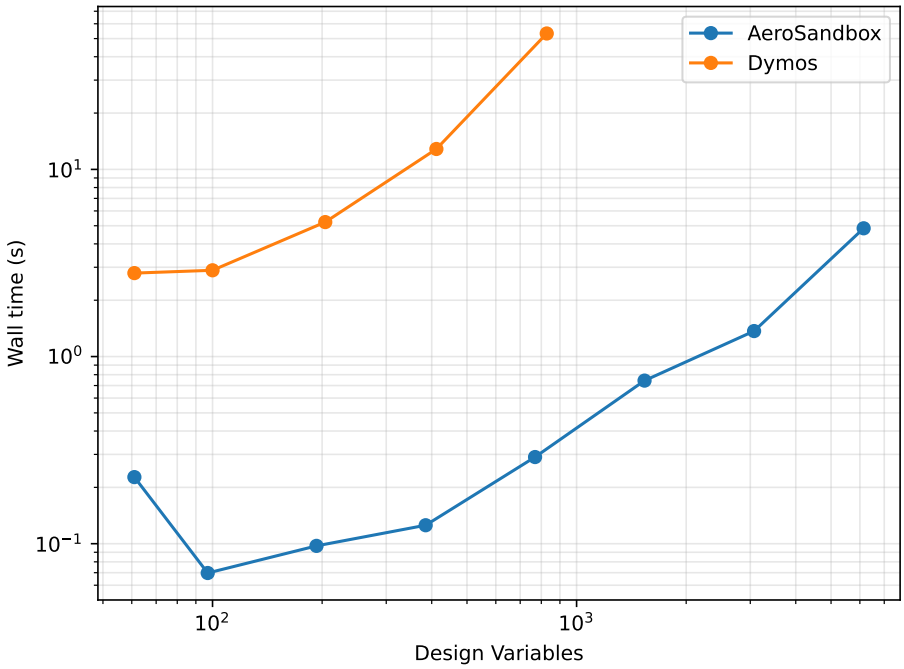

Figure 7: Wall time versus number of design variables for both frameworks.

The wall time versus number of design variables plot in Fig. (7) shows an overall increase in computation time for both frameworks as the mesh is refined. Dymos remains significantly slower than AeroSandbox across all mesh sizes, reflecting its more computationally intensive collocation scheme and derivative computations. AeroSandbox exhibits an unusual behavior in which the wall time for the first data point is higher than for the next three mesh sizes before increasing steadily beyond that, likely due to initial setup overhead that is amortized over slightly larger but still small meshes. Overall, the trend highlights the trade-off between accuracy and computational cost for the two frameworks.

The convergence plot of the objective value for the finest mesh size in Fig. (8) shows that both frameworks reach rapid convergence, reflecting efficient optimization at high fidelity. Dymos exhibits two brief dips below the true optimum, indicating temporary constraint violations, before settling at its final optimal value. AeroSandbox displays an interesting plateau slightly above the optimum for over 30 iterations before eventually decreasing to match Dymos' final value, suggesting the presence of a shallow local minimum or a region of slow convergence. These behaviors highlight subtle differences in how each framework navigates the high-dimensional optimization landscape at fine mesh resolution.

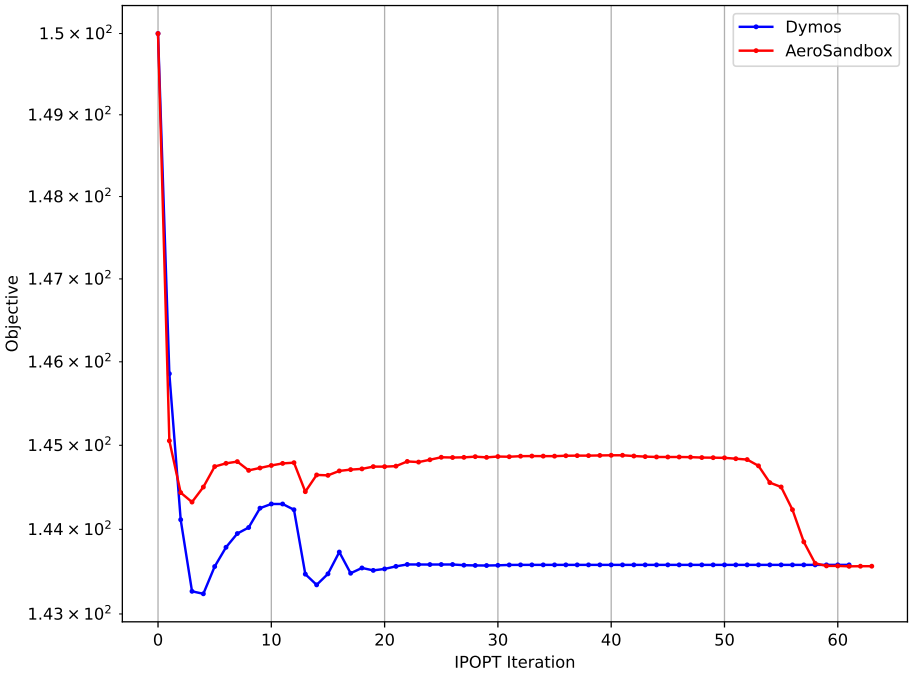

Figure 8: Convergence of the objective value for the finest mesh size for both frameworks.

6.2 Investigation of the Optimum

As shown in Fig. (9), the plot of AeroSandbox optimal trajectories across all mesh sizes (N = 10 to 2048) illustrates how mesh refinement affects trajectory shape. The coarsest trajectory follows a nearly vertical ascent before sharply veering toward horizontal, reflecting the limited resolution of the pitch control. As N increases, the trajectories become progressively smoother, showing a more gradual reduction in pitch angle over the ascent. The finest mesh trajectory lies below all others, reflecting this gradual maneuvering and more efficient use of thrust to reach the target orbit. This trend demonstrates how increased mesh resolution allows the optimizer to capture subtler control variations and produce more physically realistic trajectories.

As shown in Fig (10), the plot of Dymos optimal trajectories across all mesh sizes (5 to 128 segments) highlights the effect of mesh refinement within the framework's higher-order collocation scheme. Due to the increased computational cost of Dymos' third-order approximation and final simulation, fewer segments are used compared to AeroSandbox. Interestingly, unlike AeroSandbox, Dymos trajectories become steeper as the mesh is refined: the coarsest trajectory lies below the others, while the finest mesh flies above all, reflecting a more aggressive ascent profile. Despite this trend, the trajectories gradually become smoother with increasing segment count, indicating that higher-resolution collocation allows the optimizer to resolve subtler variations in pitch control and produce more refined trajectories.

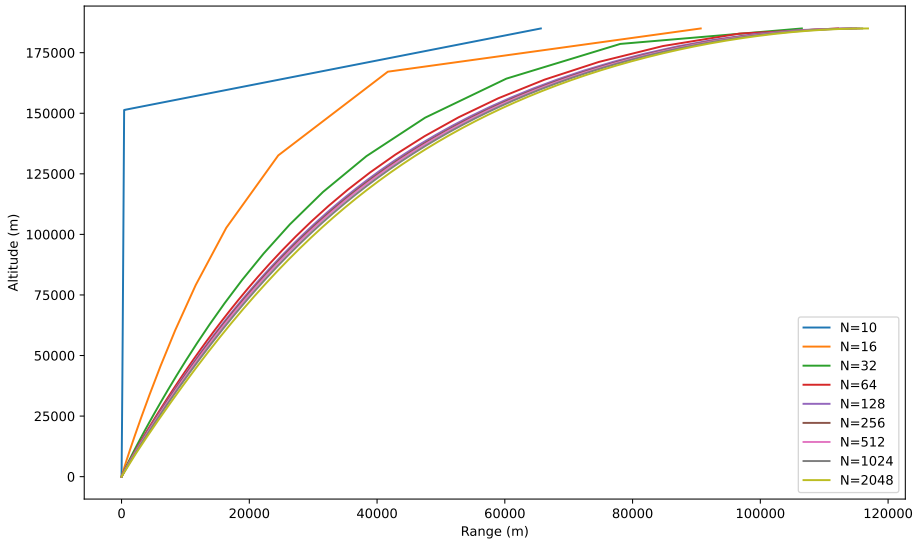

Figure 9: Optimal trajectories of AeroSandbox across all mesh sizes.

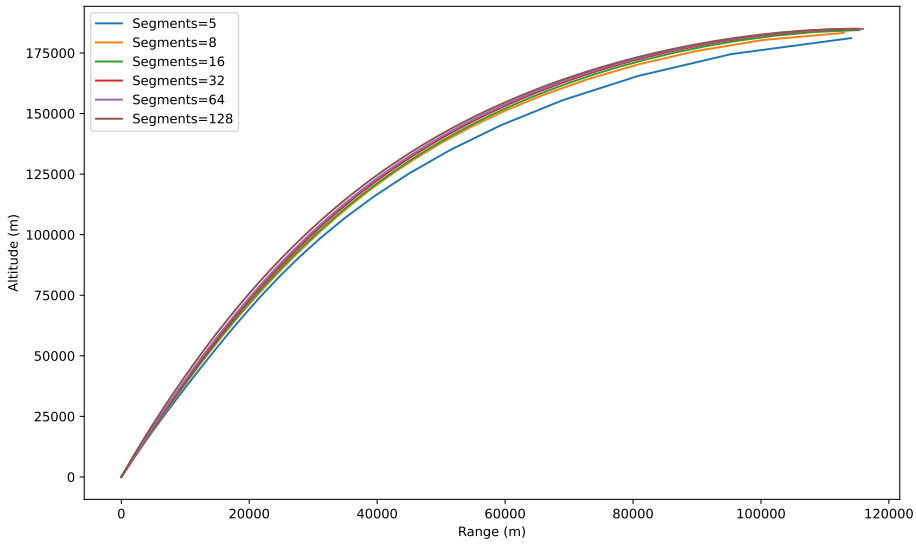

Figure 10: Optimal trajectories of Dymos across all mesh sizes.

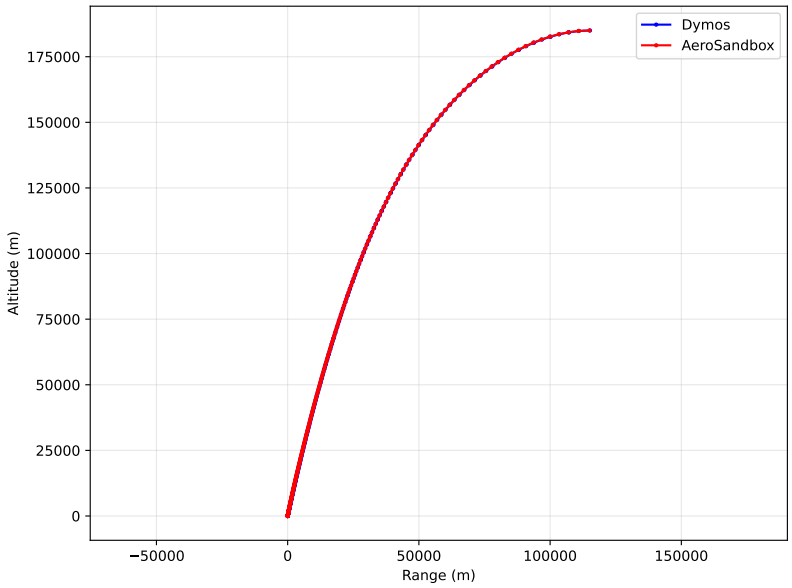

Figure 11: Comparison of the optimal trajectory of the finest mesh size for both frameworks.

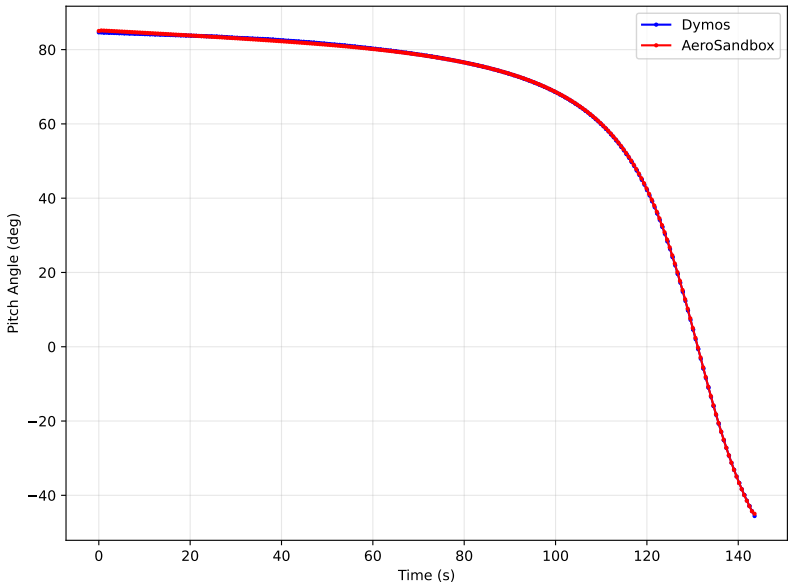

Figure 12: Comparison of the optimal pitch angle histories of the finest mesh size for both frameworks.

As seen in Fig. (11) and Fig. (12), the optimal trajectory and pitch angle history of the finest mesh for each framework is essentially identical. Despite the difference in mesh sizes, the number of design variables for each is similar, and at this level of fidelity we see that both frameworks offer the same robust solution.

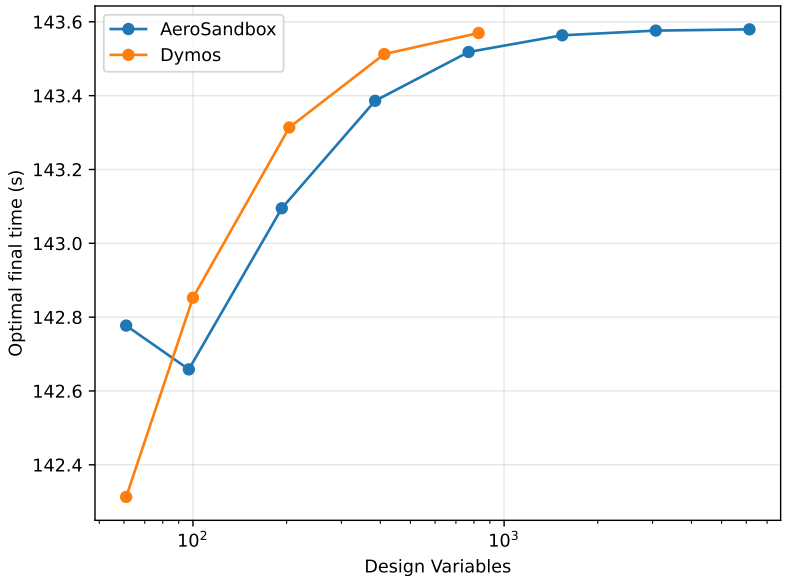

Figure 13: Optimum value of ascent time for both frameworks across all mesh sizes.

The plot in Fig. (13) of optimal ascent time versus mesh size illustrates how each framework approaches the true minimum as the number of design variables increases. Dymos starts with the lowest final time for the coarsest mesh, then gradually increases above AeroSandbox before flattening out, reflecting its stricter adherence to constraints and consistent with the steeper trajectories observed for finer meshes. AeroSandbox begins with a higher ascent time than Dymos, drops below Dymos for intermediate mesh sizes, and then increases, eventually converging to the same final value as Dymos but at a higher number of design variables. This behavior highlights the differences in how the two frameworks balance trajectory feasibility, constraint satisfaction, and optimization accuracy across mesh resolutions.

The plot in Fig. (14) of final mass versus number of design variables closely mirrors the trends observed in the optimal ascent time plot. Dymos begins with the highest final mass, exceeding 2000 kg, and then decreases toward a more realistic value below 1200 kg as the mesh is refined, eventually flattening out. AeroSandbox starts with a lower initial mass around 1800 kg, briefly exceeds Dymos' mass for intermediate mesh sizes, and then decreases and stabilizes below 1200 kg for the finest meshes. These trends reflect how both frameworks progressively converge to physically realistic solutions as the number of design variables increases, while also highlighting the differences in transient behavior due to their distinct transcription and derivative methods.

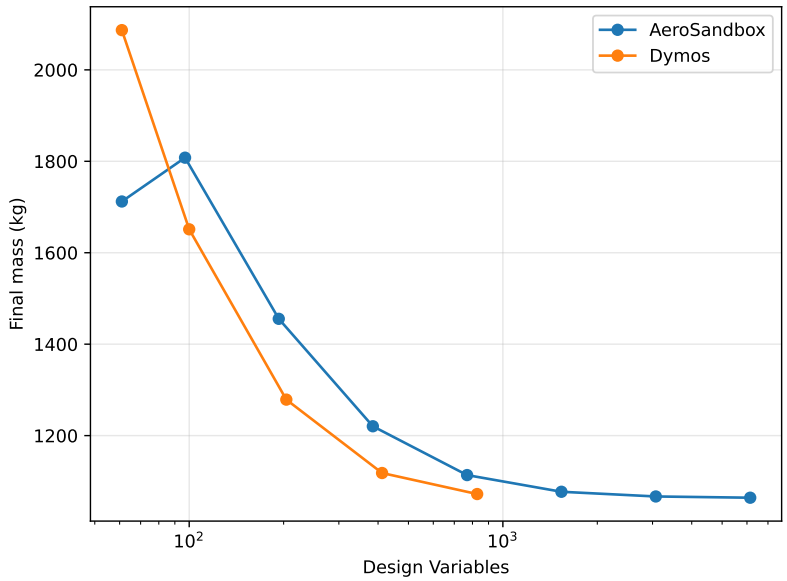

Figure 14: Resultant final rocket mass for both frameworks across all mesh sizes.

Guess	Initial θ Range (radians)	Initial x Range (m)	Initial y Range (m)
0 (original)	[1.5, -0.76]	[0, 115000]	[0, 185000]
1 (shallow trajectory)	[0.3, -0.3]	[0, 140000]	[0, 120000]
2 (steep trajectory)	[1.4, 0.2]	[0, 80000]	[0, 225000]
3 (low energy)	[0, 0]	[0, 70000]	[0, 80000]
4 (high energy)	[1.2, -1]	[0, 20000]	[0, 250000]

Table 2: Summary of initial guesses and variable ranges for testing solver robustness.

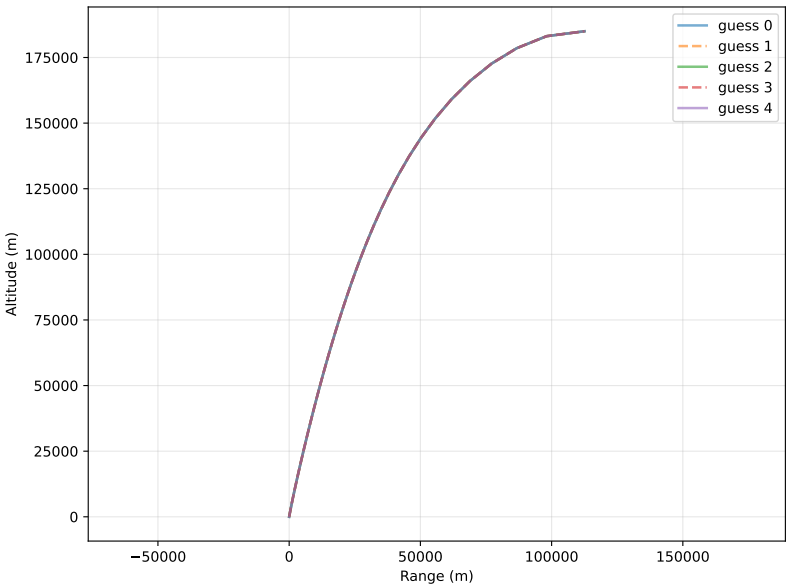

Figure 15: Optimized trajectories from AeroSandbox using various initial conditions.

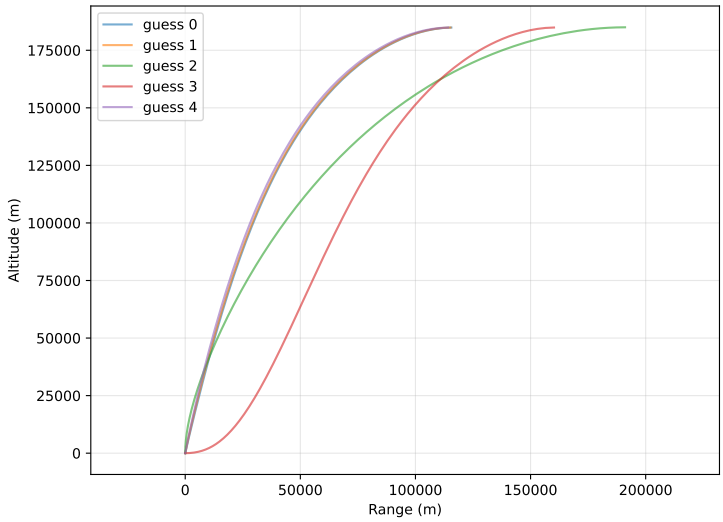

Figure 16: Optimized trajectories from Dymos using various initial conditions.

Guess	Final Time ASB	Final Time Dymos
0	143.4068711831387	143.51290283
1	143.40687118307926	143.51272548
2	143.4068711830763	141.68143616
3	143.40687118301457	143.40019827
4	143.4068711830794	143.51274207

Table 3: Optimal values of final ascent time for each framework and initial guess.

The plots in Fig. (15) and Fig. (16) comparing trajectories obtained from multiple initial guesses highlights a clear difference in robustness between AeroSandbox and Dymos. For AeroSandbox, all initial guesses converge to the same optimal trajectory, indicating a stable solution and strong insensitivity to initialization. In contrast, Dymos exhibits sensitivity to the initial pitch angle profile: steep and shallow initial guesses converge to noticeably different optimal trajectories. These alternative Dymos solutions travel farther in the horizontal direction and achieve slightly lower final ascent times than the nominal optimum, with one steep initial guess producing a significant outlier with a final time below 142 seconds compared to all other cases above 143 seconds. This behavior suggests the presence of multiple local minima in the high-dimensional optimization landscape, which Dymos' higher-order collocation and stricter constraint enforcement may resolve differently depending on initialization. It also indicates that Dymos' increased fidelity can expose alternative feasible trajectories that are not captured by AeroSandbox's lower-order transcription, highlighting a trade-off between robustness to initial guesses and sensitivity to solution structure.

7 Overall Conclusions and Recommendations

In conclusion, this study compared Dymos and AeroSandbox for a 2D rocket ascent problem, with emphasis on how collocation schemes, derivative computation, solver behavior, and sensitivity to initial guesses affect optimization performance. Overall, Dymos produced more accurate and physically realistic solutions, with tighter adherence to terminal constraints, smoother trajectories, and higher-fidelity final mass estimates, even at moderate mesh resolutions. Its use of higher-order collocation and accurate derivatives enables robust handling of the coupled nonlinear dynamics. However, Dymos is significantly more computationally expensive, exhibiting longer wall times, limited scalability to very fine meshes, and increased sensitivity to initial guesses. In particular, different initial pitch-angle guesses led to distinct local optima, including outlier trajectories with lower final times, indicating that Dymos may converge to different feasible solutions depending on initialization.

AeroSandbox, by contrast, demonstrated substantially faster runtimes and greater scalability, allowing convergence at much higher mesh resolutions with manageable computational cost. While its solutions were slightly less accurate—showing small overshoots in terminal altitude, velocity, and mass—they were consistently physically reasonable. Notably, AeroSandbox exhibited strong robustness to initial guesses, with all tested initial conditions converging to essentially the same optimal trajectory and final time. Its primary strengths are speed, flexibility, and robustness, while its main limitations include lower solution fidelity and occasional non-monotonic convergence behavior.

Based on these results, Dymos is best suited for high-fidelity trajectory optimization problems where accuracy, strict constraint satisfaction, and detailed trajectory structure are paramount, such as mission-critical launch vehicle design or high-accuracy multidisciplinary optimization. AeroSandbox is better suited for early-stage design, rapid prototyping, parametric studies, and sensitivity analyses, where computational efficiency and robustness to initialization are more important than marginal gains in optimality.

Future work could investigate hybrid approaches that combine AeroSandbox's efficiency with Dymos's accuracy, explore adaptive or hp-mesh refinement to improve Dymos scalability, and extend the comparison to three-dimensional trajectories, variable-thrust models, or stochastic perturbations. A more systematic study of sensitivity to initial guesses, collocation strategies, and solver settings would also provide deeper insight into the trade space between robustness, accuracy, and computational cost in trajectory optimization frameworks.

References

[1] OpenMDAO Dymos. GitHub repository, https://github.com/OpenMDAO/dymos. Accessed 16 Dec. 2025.

[2] Sharpe, P. D. AeroSandbox. GitHub repository, https://github.com/peterdsharpe/AeroSandbox. Accessed 16 Dec. 2025.

[3] Kelly, M. "A Collection of Tutorials on Trajectory Optimization." https://www.matthewpeterkelly.com/tutorials/trajectoryOptimization/index.html. Accessed 16 Dec. 2025.

[4] Gray, J. S., Hwang, J. T., Martins, J. R. R. A., Moore, K. T., and Naylor, B. A. "OpenMDAO: An Open-Source Framework for Multidisciplinary Design, Analysis, and Optimization." *Structural and Multidisciplinary Optimization*, Vol. 59, No. 4, 2019, pp. 1075–1104.

[5] Falck, R., Gray, J. S., Ponnapalli, K., and Wright, T. "Dymos: A Python Package for Optimal Control of Multidisciplinary Systems." *Journal of Open Source Software*, Vol. 6, No. 59, 2021, Article 2809.

[6] Sharpe, P. D. *AeroSandbox: A Differentiable Framework for Aircraft Design Optimization*. Master's Thesis, Massachusetts Institute of Technology, Cambridge, MA, 2021. Available at https://dspace.mit.edu/handle/1721.1/140023.

[7] Kelly, M. "An Introduction to Trajectory Optimization: How to Do Your Own Direct Collocation." *SIAM Review*, Vol. 59, No. 4, 2017, pp. 849–904.

8 Appendix

8.1 AI Use Disclaimer

Generative AI was utilized on this assignment to improve writing style and ensure a professional tone, as well as for its expertise with Python in debugging assistance and plotting syntax. AI was not used for generating any original ideas, problem formulation, modeling decisions, or analysis of results.