OCULUS - A unified visual solver for Optimal Control Problems via indirect methods

76th International Astronautical Congress (IAC), Sydney, Australia, 29 Sep-3 Oct 2025.

IAC–25–E1,4,4,x103041

OCULUS - A unified visual solver for Optimal Control Problems via indirect methods

Luigi Mascolo[a*]

[a] *Dipartimento di Ingegneria Meccanica e Aerospaziale (DIMEAS), Politecnico di Torino, Corso Duca degli Abruzzi 24, 10129, Torino, IT, luigi.mascolo@polito.it*

[*] *Corresponding author*

Abstract

Indirect Methods (IMs) for solving Optimal Control Problems (OCPs) are computationally efficient but often require human intervention to initiate the optimization process. One of the primary challenges in using IMs, regardless of the problem being solved, is selecting appropriate initial costates. Indeed, the advantage of a highly efficient computational tool comes at the cost of these methods being extremely sensitive to initial guesses. OCULUS (Optimal Control with User-friendly Layout Unified Solver) addresses this issue by quasi-automating the initialization process and providing an interactive interface that streamlines the solution of single- and multiple-point boundary value problems. OCULUS is a versatile visualizer and solver capable of handling any indirect optimal control problem where the dynamical model features either autonomous bang-bang control (where control switches based on the sign of the switching function) or predefined switching decision arcs. A major drawback of traditional IM implementations is the trade-off between computational efficiency and user-friendliness. Codes relying on compiled languages like Fortran or C++ offer high performance but lack intuitive visualization, while Python and MATLAB provide better visualization but suffer from slow computational speed. OCULUS bridges this gap by allowing users to integrate compiled dynamic files while maintaining real-time visualization capabilities within a dynamically updated GUI that adapts to user-defined setups. Users can adjust costates between integration steps, apply differential corrections incrementally, and refine parameters manually before resuming optimization.

OCULUS has already demonstrated its effectiveness in academia. Students using the solver for their thesis research have reduced problem setup time from several months to just a few weeks, eliminating the need for repeated manual iterations in the early stages of learning the subject. Moreover, OCULUS features an automatically populated database that stores previously converged solutions, enabling interpolation-based or neighboring initial guesses for new boundary conditions once at least one previous solution has been found. Users can select different integration methods based on precision, speed, and robustness requirements and fine-tune the finite-difference method with user-defined sensitivity thresholds.

By automating critical aspects of IMs and providing an interactive, adaptable environment, OCULUS makes solving complex optimal control problems more accessible. It allows researchers and engineers to focus on problem formulation and analysis rather than struggling with initialization and convergence challenges (however instructive those challenges may be). The presented work includes an Earth-Moon n-body transfer case study, illustrating the solver's capabilities.

Nomenclature

$\mathcal{H}$	Hamiltonian
J	Jacobian
$\mathcal{J}$	functional, merit index
$\boldsymbol{p}$	guess vector
S_F	Switching Function
u	control
$\boldsymbol{x}$	state vector
ζ	Levenberg-Marquardt damping
$\boldsymbol{\lambda}$	costate vector
$\boldsymbol{\chi}$	boundary conditions vector

Acronyms/Abbreviations

AW	Armijo-Wolfe
BC	Boundary Condition
FD	Finite Difference
IM	Indirect Method
LM	Levenberg-Marquardt
OCP	Optimal Control Problem
OCT	Optimal Control Theory
OCULUS	Optimal Control with User-friendly Layout Unified Solver
ODE	Ordinary Differential Equation
RF	Reference Frame
SC	Spacecraft
STM	State Transition Matrix

1. Introduction and Motivation

Students pursuing advanced research in space flight mechanics and astrodynamics often face a steep learning curve when transitioning from theory to practical implementation. In particular, IMs for trajectory optimization are powerful, fascinating for sure and mathematically elegant, but also highly sensitive to initialization and tightly linked to problem-specific formulations. Consequently, students and researchers rarely develop implementations from scratch; instead, they rely on legacy codes within research groups. Unfortunately, such codes are often poorly documented or based on a non-efficient version control, making it difficult for newcomers to grasp underlying methods quickly and to link all the necessary information. This creates a dual barrier: mastering the theoretical foundations of OCPs while simultaneously navigating non-intuitive software environments.

Most undergraduate and graduate education courses typically emphasize the use of accessible tools such as Python for industry applications or MATLAB in academic settings, with increasing adoption over the last decade. Exposure to compiled languages persists but appears to be declining in some curricula [1]. While compiled codes offer the computational efficiency required for challenging problems, they typically lack the visualization aids that support learning, especially in the early stages. At the opposite end of the spectrum, fully predefined optimization suites can deliver quick and reliable solutions, but they remove students from the formative process of deriving, initializing, and refining an OCP. Such tools risk turning learning into mere application, depriving students of the opportunity to grapple with the structure of the problems they are facing and the experience-based art of trajectory optimization. This challenge is particularly acute in academic settings where the theoretical backbone is predominant [2], and where the transition from university to research or industry requires students to acquire complementary skills not emphasized during their studies.

From a pedagogical perspective, this bridging difficulty can be interpreted through the framework of biologically primary and secondary knowledge [3, 4]. Primary knowledge encompasses innate or easily acquired skills, such as language and basic motor actions, whereas secondary knowledge, such as programming and scientific reasoning, requires sustained effort and intentional practice to master. Deriving, implementing, and solving an OCP clearly falls into this latter category. When students are introduced to such material, the combined cognitive load of theory and technical implementation can exceed working memory capacity. Engaging with both dimensions nonetheless provides a valuable learning opportunity: while this process embodies the principle of "learning by doing", research in educational psychology highlights a critical caveat, namely that activity-based learning fosters deep understanding only when adequately scaffolded; if poorly supported, it can overwhelm learners with extraneous demands. Indeed, several studies have shown that generative activities may even yield worse outcomes than studying pre-packaged examples when task demands are too high [5, 6]. This risk has been described as the danger of "doing without learning" [7]. In the present context, and following the Cognitive Load Theory [8, 9], extraneous tasks such as debugging code in unfamiliar languages, navigating through poorly maintaned and non version-controlled codes, or compensating for missing visualization, consume cognitive resources that should instead be devoted to understanding the structure of the project at hand, in the present analysis IMs, especially in the early stages. The challenge is further amplified by the advent of LLMs and AI-assisted tools: although they can accelerate routine tasks, they often create a false sense of progress by providing rapid solutions at the cost of risking superficial engagement and undermining the development of essential problem-solving skills.

To address this challenge, in 2021, within MSc thesis projects conducted at the end of the degree program, a project-based initiative tasked candidates with progressively building simplified Python solvers with the explicit aim of solving OCPs via IM. The activity provided authentic practice in algorithm design and debugging and aligned with generative-learning principles [10]. In practice, however, progress was constrained by an undestandable limited programming fluency, the sensitivity of shooting-based IM to initial costates and boundary conditions in the absence of robust visualization and warm-starts, and the lack of an integrated environment to store and reuse nominal trajectories. Under thesis deadlines, candidates ultimately pivoted to pre-packaged direct-method toolboxes to complete their studies, and the intended IM implementations could not be brought to convergence within the available time and tooling.

These observations motivated, in late 2022, the development of the first prototype (`v0.0.1`) of Optimal Control with User-friendly Layout Unified Solver (OCULUS), designed to address the pain points identified in earlier MSc thesis projects. The design emphasized clarity of the core building blocks over automation: templates for defining generic problems (ODEs, boundary conditions, parameters, and cost functionals), example problems and unit-test stubs to encourage best practices, and strong visualization tools to diagnose costate guesses and boundary mismatches during shooting iterations. This first release enabled faster prototyping and made the structure of IMs more transparent to learners. Nonetheless, `v0.0.1`

remained performance-limited: its predominantly Python implementation incurred substantial interpreter overhead, restricting the size and fidelity of case studies feasible within thesis timelines. These limitations led to a sequence of private releases up to `v0.3.6` (from the end of 2023 up to early 2025), culminating recently in `v1.0.0`, which integrates compiled dynamics with a Python-based GUI.

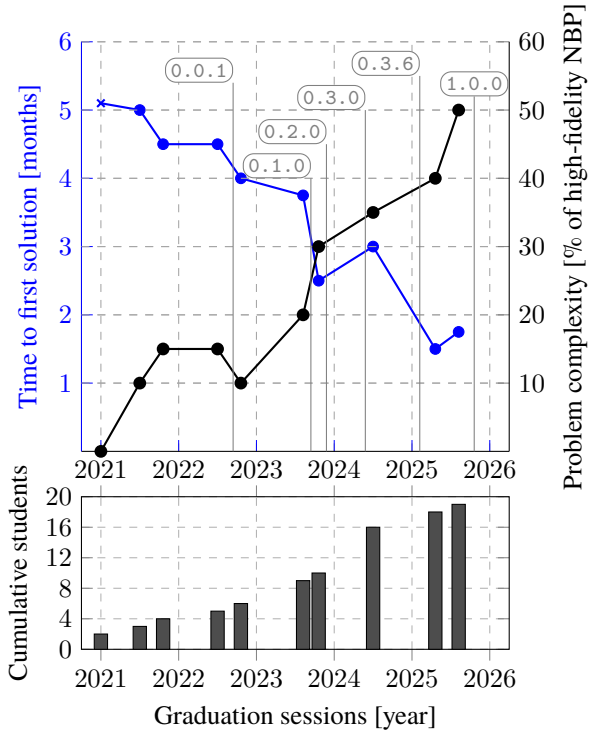

Fig. 1. Reduction in time from problem implementation to first solution (blue, left axis) and increase in problem complexity (black, right axis) achieved by MSc students using OCULUS.

The educational impact of OCULUS has already been observed in academic settings, as illustrated in Figure 1. Here, the x-axis data points represent successive MSc thesis sessions (from mid-2021 to September 2025), each corresponding to a specific graduation cohort. Students using the solver for their thesis research reduced problem-setup times from several months to a few weeks and engaged more deeply with the theoretical aspects of IMs rather than with the mechanics of software operation. Over time, they were able to progress from a simple unperturbed two-body problem (2BP) in mid-2021 to a double-perturbed medium-fidelity formulation, all coded independently starting from the Ordinary Differential Equations (ODEs), Boundary Conditions (BCs), and the OCP itself.

The notion of problem complexity in Figure 1 reflects this progression. Here, 100% corresponds to a high-fidelity n-body problem, i.e., a trajectory optimization problem including atmospheric drag, Earth's spherical harmonics, solar radiation pressure (SRP), and at least two perturbing bodies such as the Sun and the Moon. A 50% fidelity case corresponds, for example, to a problem including drag and SRP only, while intermediate percentages represent partial combinations of these perturbations. Importantly, even with OCULUS, the coding component is preserved: students must still define their own problem formulation, including ODEs, BCs, and cost functional, rather than relying on a "black-box" solver.

2. Methodology

The following case study, adapted from Refs. [11–13], is presented as a sufficiently complex example to demonstrate the usefulness and the the benefits of OCULUS. It shows how, despite a high dimensionality of a problem, the overall structure can be immediately visualized and studied. At the same time, the overview below outlines the key steps required for a student to actively implement an OCP using IMs.

Consider a generic OCP for space trajectory optimization, defined by a system of first-order ODEs describing the evolution of n state variables under m control inputs $\boldsymbol{u}$, with time t as the independent variable:

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t), t) \tag{1}$$

An optimal solution is a trajectory $\boldsymbol{x}^\star(t)$ under control $\boldsymbol{u}^\star(t)$ satisfying all BCs at the initial and final states,

$$\boldsymbol{\chi}(\boldsymbol{x}_0, \boldsymbol{x}_f, t_0, t_f) = \boldsymbol{0} \tag{2}$$

where $\boldsymbol{\chi} : [\mathbb{R}^n, \mathbb{R}^n, \mathbb{R}, \mathbb{R}] \to \mathbb{R}^q$, and possibly with bounds on $\boldsymbol{u}$, while optimizing a merit index

$$\mathcal{J} = \varphi(\boldsymbol{x}_0, \boldsymbol{x}_f, t_0, t_f) + \int_{t_0}^{t_f} \Phi(\boldsymbol{x}(t), \boldsymbol{u}(t), t)\, \mathrm{d}t \tag{3}$$

Problems of this form belong to Bolza's class, and Eq. (3) reduces to a Mayer problem with $\Phi = 0$ when the cost depends only on boundary values. Let $\varphi \triangleq \varphi(\boldsymbol{x}_0, \boldsymbol{x}_f, t_0, t_f)$ and $\boldsymbol{\chi} \triangleq \boldsymbol{\chi}(\boldsymbol{x}_0, \boldsymbol{x}_f, t_0, t_f)$ for brevity; IMs introduce adjoint variables $\boldsymbol{\lambda}(t) \in \mathbb{R}^n$ and multipliers $\boldsymbol{\mu} \in \mathbb{R}^q$ for the boundary conditions, yielding the augmented functional

$$\mathcal{J}^\star = \varphi + \boldsymbol{\mu}^\top \boldsymbol{\chi} + \int_{t_0}^{t_f} \left[\Phi + \boldsymbol{\lambda}^\top (\boldsymbol{f} - \dot{\boldsymbol{x}})\right] \mathrm{d}t \tag{4}$$

Integrating by parts the $-\boldsymbol{\lambda}^\top \dot{\boldsymbol{x}}$ term gives

$$\begin{aligned} \mathcal{J}^\star = \varphi + \boldsymbol{\mu}^\top \boldsymbol{\chi} + (\boldsymbol{\lambda}_0^T \boldsymbol{x}_0 - \boldsymbol{\lambda}_f^\top \boldsymbol{x}_f) + \\ + \int_{t_0}^{t_f} \left(\Phi + \boldsymbol{\lambda}^\top \boldsymbol{f} - \dot{\boldsymbol{\lambda}}^\top \boldsymbol{x}\right) \mathrm{d}t \end{aligned} \tag{5}$$

with the Hamiltonian

$$\mathcal{H} \triangleq \Phi + \boldsymbol{\lambda}^\top \boldsymbol{f} \tag{6}$$

Stationarity of $\mathcal{J}^\star$ leads to the first-order variation

$$\begin{aligned}\delta\mathcal{J}^\star = &\left(\frac{\partial\varphi}{\partial t_0} + \boldsymbol{\mu}^\top\frac{\partial\boldsymbol{\chi}}{\partial t_0} - \mathcal{H}_0\right)\delta t_0 + \\ &+ \left(\frac{\partial\varphi}{\partial t_f} + \boldsymbol{\mu}^\top\frac{\partial\boldsymbol{\chi}}{\partial t_f} + \mathcal{H}_f\right)\delta t_f + \\ &+ \left(\frac{\partial\varphi}{\partial \boldsymbol{x}_0} + \boldsymbol{\mu}^\top\frac{\partial\boldsymbol{\chi}}{\partial \boldsymbol{x}_0} + \boldsymbol{\lambda}_0^\top\right)\delta\boldsymbol{x}_0 + \\ &+ \left(\frac{\partial\varphi}{\partial \boldsymbol{x}_f} + \boldsymbol{\mu}^\top\frac{\partial\boldsymbol{\chi}}{\partial \boldsymbol{x}_f} - \boldsymbol{\lambda}_f^\top\right)\delta\boldsymbol{x}_f + \\ &+ \int_{t_0}^{t_f}\left[\left(\frac{\partial\mathcal{H}}{\partial\boldsymbol{x}} + \dot{\boldsymbol{\lambda}}^\top\right)\delta\boldsymbol{x} + \frac{\partial\mathcal{H}}{\partial\boldsymbol{u}}\delta\boldsymbol{u}\right]\,\mathrm{d}t\end{aligned} \tag{7}$$

which vanishes when the multiplying coefficients are zero. This produces:

- 2 transversality conditions at t_0 and t_f,
- $2n$ optimality conditions on $\boldsymbol{x}_0$ and $\boldsymbol{x}_f$,
- n adjoint ODEs for $\dot{\boldsymbol{\lambda}}$, the Euler-Lagrange equations,
- m algebraic equations for the optimal control $\boldsymbol{u}(t)$.

A user wishing to solve this problem has to deal with all the points above, one by one. For example, nullifying the coefficients of $\delta t_0, \delta t_f, \delta\boldsymbol{x}_0, \delta\boldsymbol{x}_f$ in Eq. (7) yields the $2 + 2n$ conditions

$$\frac{\partial\varphi}{\partial t_0} + \boldsymbol{\mu}^\top\frac{\partial\boldsymbol{\chi}}{\partial t_0} - \mathcal{H}_0 = 0 \tag{8a}$$

$$\frac{\partial\varphi}{\partial t_f} + \boldsymbol{\mu}^\top\frac{\partial\boldsymbol{\chi}}{\partial t_f} + \mathcal{H}_f = 0 \tag{8b}$$

$$\frac{\partial\varphi}{\partial \boldsymbol{x}_0} + \boldsymbol{\mu}^\top\frac{\partial\boldsymbol{\chi}}{\partial \boldsymbol{x}_0} - \boldsymbol{\lambda}_0^\top = \boldsymbol{0} \tag{8c}$$

$$\frac{\partial\varphi}{\partial \boldsymbol{x}_f} + \boldsymbol{\mu}^\top\frac{\partial\boldsymbol{\chi}}{\partial \boldsymbol{x}_f} + \boldsymbol{\lambda}_f^\top = \boldsymbol{0} \tag{8d}$$

which control how times and states, respectively with 2 transversality and $2n$ optimality conditions, should behave at the trajectory extremal points. Some easy rules can be found in Figure 3.

By nullifying the coefficient of $\delta\boldsymbol{x}$ one retrieves the Euler-Lagrange equations for the adjoint variables

$$\frac{\mathrm{d}\boldsymbol{\lambda}}{\mathrm{d}t} = -\left(\frac{\partial\mathcal{H}}{\partial\boldsymbol{x}}\right)^\top, \tag{9}$$

which have to be manually derived per each new problem and set of ODEs, except for simpler cases where symbolic or automatic differentiation can be performed. Lastly, by nullifying the coefficient of $\delta\boldsymbol{u}$, a total of m algebraic equations for the controls are found, namely

$$\left(\frac{\partial\mathcal{H}}{\partial\boldsymbol{u}}\right)^\top = 0 \tag{10}$$

In order to give an example for the solution of Equation (10), a generic ODE dynamical system for Spacecraft (SC) trajectory optimization can be presented in the form

$$\frac{\mathrm{d}r}{\mathrm{d}t} = u \tag{11a}$$

$$\frac{\mathrm{d}\vartheta}{\mathrm{d}t} = \frac{v}{r\cos\varphi} \tag{11b}$$

$$\frac{\mathrm{d}\varphi}{\mathrm{d}t} = \frac{w}{r} \tag{11c}$$

$$\frac{\mathrm{d}u}{\mathrm{d}t} = -\frac{\mu}{r^2} + \frac{v^2}{r} + \frac{w^2}{r} + \frac{T_u}{m} + (\mathrm{a}_p)_u \tag{11d}$$

$$\frac{\mathrm{d}v}{\mathrm{d}t} = -\frac{uv}{r} + \frac{vw}{r}\tan\varphi + \frac{T_v}{m} + (\mathrm{a}_p)_v \tag{11e}$$

$$\frac{\mathrm{d}w}{\mathrm{d}t} = -\frac{uw}{r} - \frac{v^2}{r}\tan\varphi + \frac{T_w}{m} + (\mathrm{a}_p)_w \tag{11f}$$

$$\frac{\mathrm{d}m}{\mathrm{d}t} = -\frac{T}{c} \tag{11g}$$

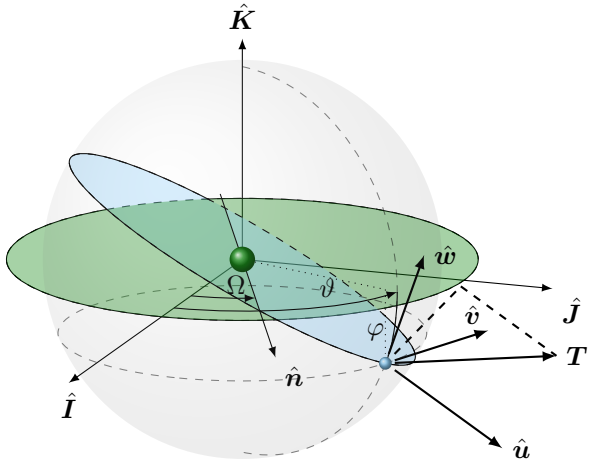

Fig. 2. SC position, velocity, and thrust components [13]

where the SC position is defined in a spherical Reference Frame (RF) and the velocity vector in a Zenith-East-North RF, as shown in Figure 2. Applying Optimal Control Theory (OCT) to Eq. (11), the optimal control $\boldsymbol{u}^\star(t)$ maximizes the spacecraft final mass. The state vector is

$$\boldsymbol{x} = \begin{Bmatrix} r & \vartheta & \varphi & u & v & w & m \end{Bmatrix}^\top, \tag{12}$$

the augmented state vector is

$$\boldsymbol{y} = \begin{bmatrix}\boldsymbol{x} \\ \boldsymbol{\lambda}\end{bmatrix}, \quad \boldsymbol{\lambda} = \begin{Bmatrix}\lambda_r & \lambda_\vartheta & \lambda_\varphi & \lambda_u & \lambda_v & \lambda_w & \lambda_m\end{Bmatrix}^\top \tag{13}$$

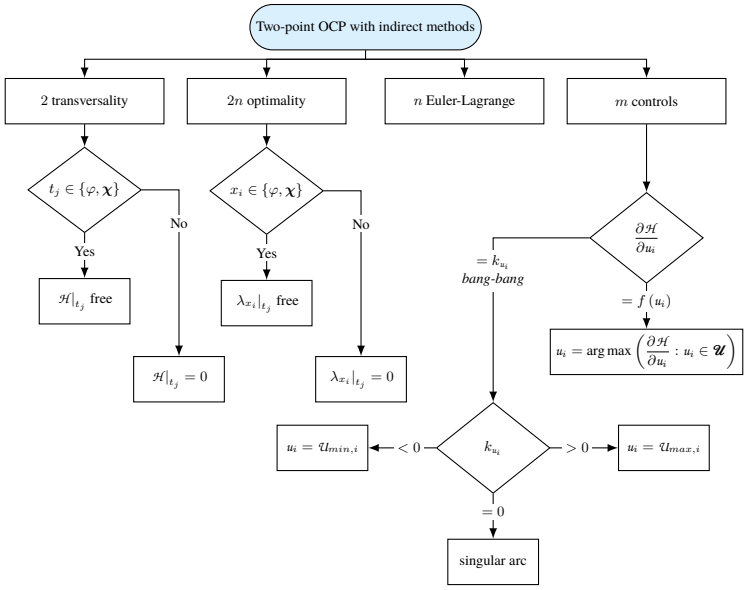

Fig. 3. Schematic flowchart representation of a two-point OCP [13]

and, in the Mayer form $\Phi = 0$, the merit index reduces to the final mass:

$$\mathcal{J} = \varphi = m_f \tag{14}$$

The Hamiltonian is, from Equations (6) and (11),

$$\mathcal{H} = \boldsymbol{\lambda}_r^\top \boldsymbol{V} + \boldsymbol{\lambda}_V^\top \left(\frac{\boldsymbol{T}}{m} - \mu \frac{\boldsymbol{r}}{r^3} + \mathbf{a}_p \right) - \lambda_m \frac{T}{c}. \tag{15}$$

Grouping the thrust terms yields

$$\mathcal{H} = \boldsymbol{\lambda}_r^\top \boldsymbol{V} + \boldsymbol{\lambda}_V^\top \left(-\mu \frac{\boldsymbol{r}}{r^3} + \mathbf{a}_p \right) + \frac{T}{m} S_F, \tag{16}$$

with

$$S_F = \boldsymbol{\lambda}_V^\top \frac{\boldsymbol{T}}{T} - \lambda_m \frac{m}{c}. \tag{17}$$

The control $\boldsymbol{u}(t)$ consists of thrust magnitude and direction. By the Pontryagin's Maximum Principle, the optimal control maximizes Eq. (16). Since $\mathcal{H}$ is linear in T, a bang-bang law arises:

$$T = T_{\max} \quad \text{if } S_F > 0, \qquad T = 0 \quad \text{if } S_F < 0. \tag{18}$$

The optimal thrust direction is parallel to $\boldsymbol{\lambda}_V$ (primer vector), so Eq. (17) reduces to

$$S_F = \lambda_V - \lambda_m \frac{m}{c}. \tag{19}$$

The optimal values for thrust angles are obtained by deriving the Hamiltonian in Eq. (16) with respect to the thrust angles in Figure 4, obtaining:

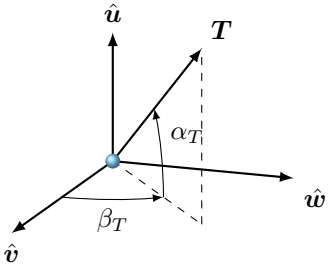

Fig. 4. Thrust vector in the SC ZEN RF [13]

$$\sin \alpha_T = \frac{\lambda_u}{\lambda_V} \tag{20a}$$

$$\cos \alpha_T \cos \beta_T = \frac{\lambda_v}{\lambda_V} \tag{20b}$$

$$\cos \alpha_T \sin \beta_T = \frac{\lambda_w}{\lambda_V}. \tag{20c}$$

3. User options for the correction process

The previous section has presented all the required elements to fully define a complex OCP, which requires the implementation and manual derivation of multiple elements. As a last but very important step, to solve the generic problem, one must determine the optimal initial values of the design vector that yield the desired final conditions while satisfying all constraints. Let this design vector $\boldsymbol{p}_0 \in \mathbb{R}^n$ be the initial guess. A single-shooting method is employed to find the optimal design vector $\boldsymbol{p}^\star$

that satisfies the BCs, $\boldsymbol{\chi}(\boldsymbol{p}^\star) = \mathbf{0}$. During the iterative correction process, two ingredients are required at each r-th iteration: the Jacobian $\boldsymbol{J}(\boldsymbol{p}) = \dfrac{\partial \boldsymbol{\chi}}{\partial \boldsymbol{p}} \in \mathbb{R}^{q\times n}$ and a rule to compute the correction $\Delta\boldsymbol{p}$. The next sections deal with all the modes that are available and exposed to the users to choose how the step direction and length, as well as the Jacobian, could be computed. Throughout the text, the first occurrence of ⊙ denotes an available choice in OCULUS, whereas ⊙ identifies a user-modifiable coefficient.

3.1 Step direction

The step direction is obtained from the damped normal equations of Levenberg-Marquardt (LM). Such an algorithm is chosen by default due to its robustness and reliability, and also because it is able to find a solution with coarser guesses compared to a standard Netwon. Let $\boldsymbol{\chi}_r = \boldsymbol{\chi}(\boldsymbol{p}_r)$ and $\boldsymbol{J}_r = \boldsymbol{J}(\boldsymbol{p}_r)$ for brevity. The direction $\Delta\boldsymbol{p}_r$ at step r and the update to the next iterate $\boldsymbol{p}_{r+1}$ are

$$\begin{aligned} \Delta\boldsymbol{p}_r &= -\boldsymbol{J}_r^\top \left(\boldsymbol{J}_r\boldsymbol{J}_r^\top + \zeta \boldsymbol{I}\right)^{-1} \boldsymbol{\chi}_r, \\ \boldsymbol{p}_{r+1} &= \boldsymbol{p}_r + \Delta\boldsymbol{p}_r\,. \end{aligned} \tag{21}$$

Here the damping (ζ) ≥ 0 controls the conditioning and the orientation of the step. Setting $\zeta = 0$ recovers the Newton/Gauss-Newton direction (respectively for $q = n$ and $q \neq n$); in this case, the local curvature can produce larger (and, thus, faster) steps when the local quadratic model is reliable, but it may overshoot or become unstable if $\boldsymbol{J}_r^\top \boldsymbol{J}_r$ is ill-conditioned. A positive ζ, on the other hand, stabilizes the linear system and biases the direction toward steepest descent, although it might slow down the convergence.

Three modes are available via ζ, as illustrated in Figure 5: Newton for $\zeta = 0$, LM with fixed $\zeta > 0$, and LM adaptive, in which ζ is increased when the achieved reduction is insufficient and decreased otherwise.

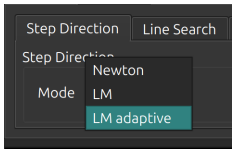

Fig. 5. Step-direction modes available to the user.

As a quick note, a distinctive feature of OCULUS is its integrated image-handling capability. Through shortcuts, users can capture screenshots either by selecting a portion of the screen or by saving in high quality directly from the GUI widgets. In addition, single-click export is available for automatically-recognized sections such as tables, tab selectors, plots, or the log window. All figures of OCULUS presented in this paper have been captured using these features.

3.1.1 Adaptive damping ζ

Regarding the last choice of the adaptive damping parameter, define at each r-th step the directional slope g_r and a gain ratio ρ_r that compares the actual decrease in $\|\boldsymbol{\chi}\|_2^2$ with its first-order prediction along $\Delta\boldsymbol{p}_r$:

$$\begin{aligned} g_r &= \boldsymbol{\chi}_r^\top \boldsymbol{J}_r \Delta\boldsymbol{p}_r < 0, \\ \rho_r &= \frac{\|\boldsymbol{\chi}_r\|_2^2 - \|\boldsymbol{\chi}_{r+1}\|_2^2}{-2g_r}\,. \end{aligned} \tag{22}$$

Here, $\boldsymbol{\chi}_{r+1}$ is the BC vector evaluated at the accepted update $\boldsymbol{p}_{r+1}$. Using $0 <$ (ηL) $<$ (ηH) < 1 and a multiplier (γ) > 1, OCULUS updates ζ after accepting $\boldsymbol{p}_{r+1}$ as

$$\begin{cases} \rho_r < \eta_L \implies \zeta_{r+1} = \min\left[\zeta_{\max}, \max\left(\zeta_0, \gamma\,\zeta_r\right)\right], \\ \rho_r > \eta_H \implies \zeta_{r+1} = \max\left(0, \dfrac{\zeta_r}{\gamma}\right), \end{cases} \tag{23}$$

and leaves ζ unchanged otherwise. Small values of ρ_r indicate that the local model overpredicts the decrease and call for stronger damping; $\rho_r < 0$ indicates an increase in the boundary-condition residual. Conversely, consistently large ρ_r justify relaxing ζ.

When different modes are selected, the interface exposes only the relevant controls; this automatic presentation highlights which coefficients govern each mode, as shown in Figure 6, helping students grasp the interdependencies and the effects of parameter changes.

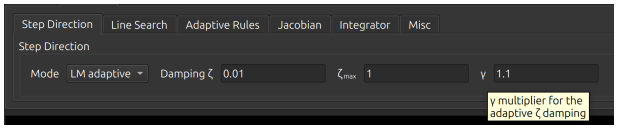

Fig. 6. Contextual options shown for the selected step-direction mode.

Tooltips are available throughout the interface and appear on hover, providing brief reminders for each control or rich hyperlinks to references for further reading.

	Newton	LM	LM adaptive
ζ		✓	✓
$\zeta_{\max}$			✓
γ			✓
η_L			✓
η_H			✓

Table 1. Parameters available for each step direction mode.

3.2 Line search for the step length

The correction $\Delta \boldsymbol{p}_r$ from Equation (21) can be scaled by a relaxation factor $k_{1,r} \in (0, 1]$, either fixed (k_1) or chosen by a backtracking line search, so that:

$$\boldsymbol{p}_{r+1} = \boldsymbol{p}_r + k_{1,r}\, \Delta \boldsymbol{p}_r. \tag{24}$$

The available modes control whether $k_{1,r}$ is Fixed or adaptively selected under four variants of the Armijo-Wolfe (AW) conditions, namely Armijo, AW, qAW, and Auto AW. To clarify the use of AW, the Armijo condition can be enforced alone or together with the Wolfe curvature condition, which can be enabled manually or automatically.

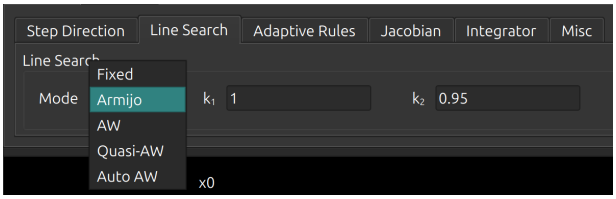

Fig. 7. Line-search modes available to the user.

Armijo's sufficient decrease selects at iteration r the largest $k_{1,r} \in \{1, \beta, \beta^2, \ldots\}$, $0 < \beta < 1$, satisfying

$$\|\boldsymbol{\chi}_{r+1}\|_2^2 \le \|\boldsymbol{\chi}_r\|_2^2 + 2\,c_1\, k_{1,r}\, g_r, \quad c_1 \in (0, 1), \tag{25}$$

and the user can specify the initial β. In the Armijo mode, only Equation (25) is enforced. However, close to a solution it is often advantageous to add the Wolfe curvature condition because Armijo alone may accept steps that are too short when $|g_r|$ is small; enforcing curvature ensures that the directional derivative is sufficiently reduced at $\boldsymbol{p}_{r+1}$, preventing unnecessarily small steps and promoting Newton-like speed. The AW mode augments Equation (25) with the Wolfe curvature condition

$$|g_{r+1}| \le c_2\, |g_r|, \qquad c_1 < c_2 < 1, \tag{26}$$

which avoids excessively short or poorly oriented steps. Since evaluating g_{r+1} requires both $\boldsymbol{\chi}_{r+1}$ and $\boldsymbol{J}_{r+1}$ (while $\boldsymbol{\chi}_{r+1}$ is already available from Armijo), the qAW option uses an approximate curvature

$$\tilde{g}_{r+1} = \boldsymbol{\chi}_{r+1}^{\top} \boldsymbol{J}_r \Delta \boldsymbol{p}_r, \tag{27}$$

reducing cost. When a fixed relaxation is preferred, a constant k_2 can be used and safeguarded by accepting the update only if the max-norm decreases, e.g., $E_{\max,r+1} < k_2\, E_{\max,r}$ with $E_{\max,r} = \max_i |\chi_i(\boldsymbol{p}_r)|$ and $k_2 \lesssim 1$.

3.2.1 Auto Wolfe

Lastly, the Auto AW option automatically toggles the Wolfe curvature condition to preserve robustness without exposing advanced choices to the user. If Wolfe is active (either by user selection, AW, or automatically, Auto AW), the line search at step r must enforce Equation (26) in addition to Armijo, in Equation (25). The decision to enable or disable Wolfe at the next step (or to keep it on or off) is based on simple rules tied to easily observable quantities. The Wolfe condition is automatically activated if one of the following holds

$$\begin{cases} \|\boldsymbol{\chi}_r\|_2 \le \tau_{\text{abs}} \\ \dfrac{\|\boldsymbol{\chi}_r\|_2}{\|\boldsymbol{\chi}_0\|_2} \le \tau_{\text{rel}} \\ k_{1,r} \le k_{\text{trig}} \\ \rho_r < \eta_L \end{cases} \tag{28}$$

and deactivated if

$$\rho_r > \eta_H \;\text{ and }\; k_{1,r} = 1. \tag{29}$$

The thresholds τ_{abs} and τ_{rel} detect proximity to the solution in absolute and relative terms; $k_{\text{trig}} \in (0, 1]$ is a direct threshold on the accepted step, namely if Armijo-only backtracking repeatedly shrinks the step, $k_{1,r}$ becomes small and Wolfe is forced; the pair $\eta_L < \eta_H$ introduces hysteresis based on the predictive quality of the local model.

	Fixed	Armijo	AW	qAW	Auto AW
k_1	✓				
k_2	✓				
β		✓	✓	✓	✓
c_1		✓	✓	✓	✓
c_2			✓	✓	✓
η_L					✓
η_H					✓
τ_{abs}					✓
τ_{rel}					✓
k_{trig}					✓

Table 2. Availability of parameters across modes.

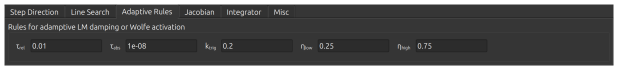

Fig. 8. Shared adaptive rules for the step direction (LM adaptive) and line search (auto AW) modes.

3.3 Jacobian computation

Because OCULUS must be problem-independent, computing the Jacobian via a state-transition matrix, while accurate, is not advisable, as it depends directly on the problem's ODEs. The Jacobian is therefore computed numerically. Each entry $J_{ij} = \partial \chi_i / \partial p_j$ is approximated by

Method	Level ℓ	Order	# FD calls of $\boldsymbol{\chi}$ per column j	Total calls per J	Total cost per J
F/B	0	1	$1^{\dagger}$	$n+1^{\ddagger}$	$\mathcal{O}(qn)$
C	0	2	2	$2n$	$\mathcal{O}(2qn)$
F/B + Richardson	ℓ	$1+\ell$	$\ell{+}1^{\dagger}$	$n(\ell{+}1)+1^{\ddagger}$	$\mathcal{O}((\ell+1)\,qn)$
C + Richardson	ℓ	$2(1+\ell)$	$2(\ell{+}1)$	$2n(\ell{+}1)$	$\mathcal{O}(2\,(\ell+1)\,qn)$

Table 3. Exact counts of $\boldsymbol{\chi}$ evaluations. The cost of one call to $\boldsymbol{\chi}$ includes integrating n ODEs and forming q residuals; wall time scales with the number of calls. †One-sided formulas reuse $\boldsymbol{\chi}(\boldsymbol{p})$, so each column needs only the perturbed call at each step size; ‡the single unperturbed call $\boldsymbol{\chi}(\boldsymbol{p})$ is shared across all columns.

a unified Finite Difference (FD) interface parameterized by $\theta \in \{0, \frac{1}{2}, 1\}$, respectively for Backward B, Central C, and Forward F:

$$J_{ij}^{[0]}(h;\theta) = \frac{\chi_i\big(\boldsymbol{p}+\theta h\,\boldsymbol{e}_j\big) - \chi_i\big(\boldsymbol{p}-(1-\theta)h\,\boldsymbol{e}_j\big)}{h}. \tag{30}$$

Here, $[0]$ denotes the base FD level. Equation (30) is the standard FD formulation and has truncation error $\mathcal{O}(h^{\xi})$, with $\xi = 1$ for one-sided B/F and $\xi = 2$ for C. Many implementations use only $[0]$, which is often sufficient but can limit accuracy and robustness with respect to h. The available choices to the user are shown in Figure 10.

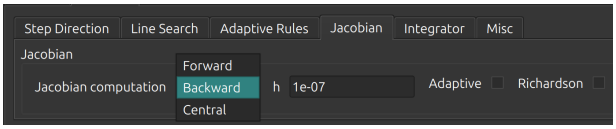

Fig. 9. Jacobian modes and options available to the user.

3.3.1 Richardson extrapolation of level ℓ

To reduce truncation error and sensitivity to h, Richardson extrapolation [14] is applied elementwise up to level $\ell = 1, 2, \ldots$ by combining evaluations at h and $h/2$. The base FD approximation admits a Taylor expansion of the form

$$J_{ij}^{[0]}(h;\theta) = J_{ij} + w_1\,h^{\xi} + w_2\,h^{2\xi} + w_3\,h^{3\xi} + \ldots. \tag{31}$$

The leading neglected term $w_1 h^{\xi}$ dominates the truncation error. Richardson extrapolation cancels this term by combining evaluations at h and $h/2$; after level ℓ the diagonal estimate satisfies

$$J_{ij}^{[\ell]}(h;\theta) = J_{ij} + w_\ell\,h^{\xi(1+\ell)} + \mathcal{O}\big(h^{\xi(2+\ell)}\big), \tag{32}$$

so the accuracy order increases from ξ at $[0]$ to $\xi(1+\ell)$ at level ℓ. The recursion used in OCULUS replaces the base estimate with

$$J_{ij}^{[\ell]}(h;\theta) = \frac{2^{\xi\ell}\,J_{ij}^{[\ell-1]}(h/2;\theta) - J_{ij}^{[\ell-1]}(h;\theta)}{2^{\xi\ell}-1}, \quad \ell \geq 1, \tag{33}$$

with $J_{ij}^{[0]}(h;\theta)$ as above and ξ dependent on the stencil (again, 1 for F/B and 2 for C). Here, ℓ counts how many extrapolations are applied (recursion depth). The resulting accuracy order after level ℓ is

$$\xi\,(1+\ell), \tag{34}$$

so each level increases the order by ξ. Only the diagonal value of the Richardson table for the given (i,j), namely $J_{ij}^{[\ell]}(h;\theta)$, is retained as the final approximation; off-diagonal entries are intermediate quantities useful for error estimation and early stopping, and are not needed to assemble the Jacobian.

3.3.2 Columnwise step size

A columnwise adaptive step can improve robustness across scales:

$$h_j = \varepsilon^{\,1/[\xi(1+\ell)+1]}\,\max\big(|p_j|, 1\big), \tag{35}$$

with machine epsilon ε (e.g., $\varepsilon \approx 10^{-16}$ in double precision). These options help students in different ways: Richardson cancels dominant truncation terms, while the adaptive h_j balances truncation and round-off errors.

3.4 Integrators

OCULUS implements both the Backward Differentiation Formula (BDF) and Adams-Moulton integrators, suitable respectively for stiff and non-stiff problems, through the CVODE library [15].

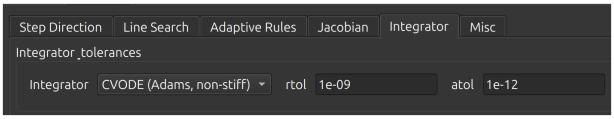

Fig. 10. Integrator options available to the user.

4. Evaluation and Results

4.1 Rationale

A question could, and probably should, arise at this point: why provide a series of different options to the user when, ideally, the most robust combination would employ

- adaptive LM for the step direction,
- automatic AW for the step size,
- central-FD with Richardson extrapolation and adaptive step size for the Jacobian.

The key point is both pedagogical and practical: the newer the problem, or the less experience a student has (especially with initializing an IM), the less likely it is that suitable initial guesses will be identified quickly. Providing access to more robust solver options may increase computational cost, but it tolerates imperfect guesses and dramatically reduces the human cost discussed above, which is the true optimization sought after with OCULUS. As students gain experience with their specific problem and learn the nuances of its dynamics and control, they can progressively tune solver options more aggressively, trading robustness for speed and achieving faster convergence on the foundation of stronger problem-specific knowledge.

To understand this human cost more thoroughly, it is useful to recap the structure described so far. Under the first-order ODEs in Equation (11), augmented to a $2n$ ODEs system via the Euler-Lagrange equations in Equation (9), trajectory optimization with user-defined and problem-specific BCs, both explicit and implicit from Equation (8), is performed by controlling thrust activation, potentially in bang-bang fashion via Equation (19), and enforcing optimal directions from Equation (20). If a more robust setup is desired, a predefined number N of thrust and coasting arcs can be imposed, which introduces $N-1$ internal switching boundaries where S_F must vanish.

With N arcs, the internal variables comprise the state and costate values at each shooting node. Counting all nodes (initial, internal, and final) yields $2nN$ unknowns; since the initial state $\boldsymbol{x}_0$ is known, this reduces to $2nN-n$ unknowns. In addition, there are $N-1$ switching times to locate the internal bang-bang transitions, and, if the final time is free, one more unknown for t_f. Let $\delta_t \in \{0,1\}$ indicate whether the final time is free ($\delta_t = 1$) or fixed ($\delta_t = 0$). Hence, the total number of unknowns is

$$U = (2nN - n) + (N - 1) + \delta_t. \tag{36}$$

Constraints $\boldsymbol{\chi}$ consist of:

- continuity of state and costate across the $N-1$ internal nodes, contributing $2n(N-1)$ conditions;
- q terminal boundary conditions, where q includes both the explicit terminal goals and the implicit optimality/transversality conditions;
- the $N-1$ zero crossings of the switching function at the internal boundaries;
- and the transversality condition on t_f when the final time is free.

Therefore,

$$C = 2n(N-1) + \underbrace{q + (N-1) + \delta_t}_{\text{human cost}}. \tag{37}$$

Subtracting gives the identity

$$U - C = n - q. \tag{38}$$

which, again, shows as per the damped LM in Equation (21), that, if $q < n$, the system is underdetermined and admits infinitely many solutions; if $q = n$, the system is square, which is the most common scenario. If $q > n$, the system is overdetermined and must be solved in the least-squares sense by minimizing terminal residuals.

From a student's perspective, supposing $q = n$, this balance translates into the need to supply $n+(N-1)+\delta_t$ initial guesses: the n initial costates, the $N-1$ switching times, and the final time if free. At the same time, students have to define also explicitly the soft, implicit BCs deriving from the optimality and transversality conditions; some quick rules can be followed, as per Figure 3, but more complex conditions require manual intervention. All of these are the true "human cost" of indirect methods. OCULUS directly alleviates this burden: it eliminates the need to hand-code the switching-arc conditions, if a pre-defined switching structure is provided, and it provides immediate visual feedback on states, costates, and the switching function, making guess exploration faster and more targeted. Consequently, coarser or imperfect guesses might still lead to convergence, reducing trial-and-error time and letting students focus on problem structure rather than manual initialization.

The remaining question is how tolerant the solver remains to suboptimal guesses while still converging to an optimal solution.

4.2 Example: high-fidelity Sun-Earth-Moon escape

For this last issue, from Ref. [13], the first optimal solution for a high-fidelity Earth-Moon escape problem was obtained as

$$\begin{Bmatrix} \lambda_{r0} \\ \lambda_{\vartheta 0} \\ \lambda_{\varphi 0} \\ \lambda_{u0} \\ \lambda_{v0} \\ \lambda_{w0} \\ \lambda_{m0} \\ t_1 \\ t_f \end{Bmatrix} = \begin{Bmatrix} 5.051\,495 \times 10^{-3} \\ -6.518\,380 \times 10^{-2} \\ 2.619\,603 \times 10^{-2} \\ 0.806\,805 \\ 0.476\,741 \\ -0.179488 \\ 0.999998 \\ 182.002777 \\ 8416.126742 \end{Bmatrix} \tag{39}$$

with a consumed propellant mass of $m_p = 3.004$ kg. The legacy solver, implemented in Fortran, featured:

- Adams-Bashforth-Moulton adaptive degree and order integrator,
- Fixed k_1,
- Fixed k_2,
- B FD Jacobian with step size $h = 1 \times 10^{-7}$, tuned to $h_t = 1 \times 10^{-8}$ for all time variables.

The convergence radius $\mathcal{C}_R$ of this solver for each variable is reported below, where the $\pm$ value denotes, on average, how far a variable can deviate from the optimal solution while still reconverging to the same trajectory without divergence:

$$\begin{cases} \mathcal{C}_R = \pm 2.4 \times 10^{-2}, & \text{for } \lambda_r, \lambda_\vartheta, \lambda_\varphi \\ \mathcal{C}_R = \pm 4.1 \times 10^{-1}, & \text{for } \lambda_u, \lambda_v, \lambda_w \\ \mathcal{C}_R = \pm 1 \times 10^{-2}, & \text{for } \lambda_m \\ \mathcal{C}_R = \pm 7.7 \times 10^{1}, & \text{for } t_1, t_f \end{cases} \tag{40}$$

When solving the same problem in OCULUS, the coarse search space can be extended to the following convergence radii, using $\zeta = 0.5$, AW, and adaptive Richardson extrapolation:

$$\begin{cases} \mathcal{C}_R = \pm 1.0 \times 10^{-1}, & \text{for } \lambda_r, \lambda_\vartheta, \lambda_\varphi \\ \mathcal{C}_R = \pm 7.5 \times 10^{-1}, & \text{for } \lambda_u, \lambda_v, \lambda_w \\ \mathcal{C}_R = \pm 3 \times 10^{-1}, & \text{for } \lambda_m \\ \mathcal{C}_R = \pm 1.2 \times 10^{2}, & \text{for } t_1, t_f \end{cases} \tag{41}$$

This improvement comes at the cost of only a $+11\%$ increase in the number of convergence steps, with negligible additional computational time (approximately one extra second). The enhancement in search space is evident: initial guesses can be on average up to five times broader than with the legacy solver, while benefiting from interactive plots that update in real time during the convergence process. A radar graphical representation of these results is shown in Figure 11.

4.3 Example: minimum-effort car

A simplified 2D longitudinal model of an electric vehicle is considered to showcase OCULUS' flexibility beyond the astrodynamics problem just presented. A vehicle must choose its acceleration profile along a road with piecewise-constant slope by minimizing the control effort. The dynamics with gravity and quadratic drag are

$$\dot{x} = v, \tag{42}$$

$$\dot{v} = u - g \sin\theta(x) - k_{\text{drag}}\, v\, |v|, \tag{43}$$

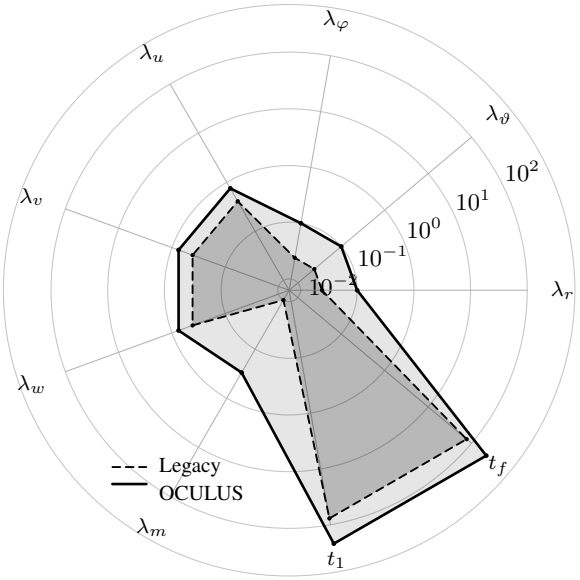

Fig. 11. Convergence-radius radar per variable. Radial scale is $\log_{10}(\mathcal{C}_R)$.

with g the gravitational constant and $k_{\text{drag}} = 0.02\,\text{m}^{-1}$. The inclined road is modeled by

$$\sin\theta(x) = \begin{cases} \sin(0°), & x < 20 \text{ and } x \geq 60, \\ \sin(10°), & 20 \leq x < 30, \\ \sin(-1°), & 30 \leq x < 60, \end{cases} \tag{44}$$

as shown in Figure 12.

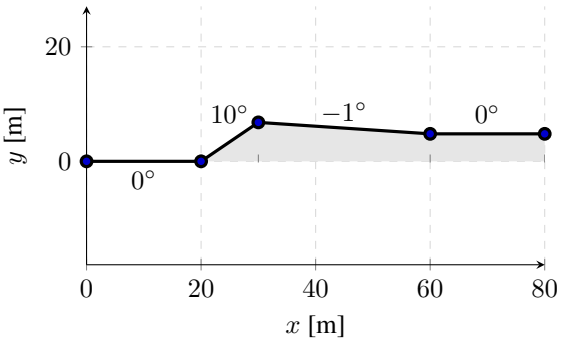

Fig. 12. Street profile

The cost functional is chosen as minimum effort,

$$J = \frac{1}{2} \int_{t_0}^{t_f} u(t)^2 \, \text{d}t, \tag{45}$$

subject to control bounds $u_{\min} \leq u \leq u_{\max}$ and the bound-

ary conditions

$$x_0 = 0\,\mathrm{m}, \quad v_0 = 0\,\mathrm{m/s}, \quad x_f = 80\,\mathrm{m}, \quad v_f = 0\,\mathrm{m/s}. \tag{46}$$

The car has a maximum acceleration and deceleration of $-u_{\min} = u_{\max} = 3\,\mathrm{m/s^2}$.

The Hamiltonian is

$$H = \frac{1}{2}u^2 + \lambda_x\, v + \lambda_v\, [u - g\, \sin\theta(x) - k_{\text{drag}}\, v\, |v|]\,. \tag{47}$$

The adjoint equations are

$$\dot{\lambda}_x = -\frac{\partial H}{\partial x} = 0, \tag{48}$$

$$\dot{\lambda}_v = -\frac{\partial H}{\partial v} = -\lambda_x + 2\, k_{\text{drag}}\, v\, \lambda_v, \tag{49}$$

while the stationarity condition is

$$\frac{\partial H}{\partial u} = u + \lambda_v = 0 \quad \Longrightarrow \quad u^\star_{\text{uncon}} = -\lambda_v. \tag{50}$$

With bounds $u_{\min} \le u \le u_{\max}$, the unconstrained $u^\star_{\text{uncon}} = -\lambda_v$ is simply clipped to the admissible interval:

$$u^\star(t) = \begin{cases} u_{\min}, & -\lambda_v(t) < u_{\min}, \\ -\lambda_v(t), & u_{\min} \le -\lambda_v(t) \le u_{\max}, \\ u_{\max}, & -\lambda_v(t) > u_{\max}. \end{cases}$$

After implementing the model and inspecting a few exploratory plots, a coarse initial guess with two decimal digits is

$$\lambda_{x0} = -0.22, \qquad \lambda_{v0} = -1.52, \qquad t_f = 21. \tag{51}$$

This produces the state timeseries in Figure 13. Even with a coarse guess, $\dot{x} > 0$ due to $v > 0$ from the dynamics, while $\lambda_v < 0$ tends to push the unconstrained control $u^\star_{\text{uncon}} = -\lambda_v > 0$, accelerating. The transient reduction of v for $t \approx 6 \div 9\,\mathrm{s}$ is caused by the steep uphill segment.

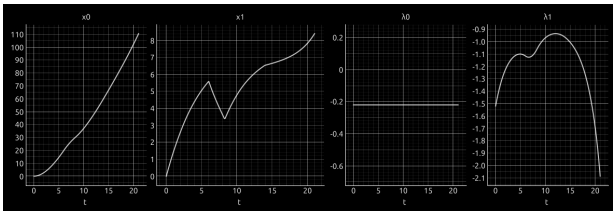

Fig. 13. Example trajectories from a coarse guess.

Using a simple Gauss-Newton step with fixed k_1 and a central-difference Jacobian, the optimal solution shown in Figure 14 is obtained, with solver logs as follows:

```
242 p=[-0.22599251 -1.52639157 21.00025861] E=4.0e-06
243 p=[-0.22599251 -1.52639157 21.00025861] E=3.3e-06
244 p=[-0.22599249 -1.52639149 21.00025861] E=1.6e-06
245 p=[-0.22599249 -1.52639149 21.00025861] E=9.8e-07
Solver finished in 0.76s (avg 3.1 ms/iter)
```

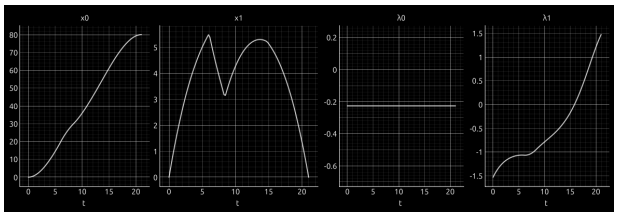

Fig. 14. Converged trajectories using Gauss-Newton with fixed k_1 and a central-difference Jacobian.

Since the problem is not non-dimensionalized, the reported residual of $\mathcal{O}(10^{-7})$ corresponds to errors of order 1×10^{-7} m and 1×10^{-7} m/s in the respective state variables.

An admissible initial guess was found after only a few minutes of trial and error, and the solver converged in less than one second, with an average computational cost of about 3 ms per iteration, including graphical updates.

5. Conclusions

This paper has presented the use of OCULUS as a continuously evolving software environment designed to support students and researchers in the formulation and solution of indirect methods for trajectory optimization. Since its earliest alpha releases, OCULUS has been well received by students. While the first versions required non-negligible user effort to set up, today its use has become nearly immediate, and the overall "human cost" has been significantly reduced. Beyond providing visualization and robust solver options, OCULUS also acts as a didactic scaffold, lowering the barrier to entry while still exposing the full structure of indirect methods, thereby encouraging a deeper engagement with the underlying theory.

The software has already demonstrated its educational value by reducing the time to first solution and enabling students to tackle increasingly complex problems within academic timelines. At the same time, it preserves manual control and transparency: the user must still define their problem formulation and can fine-tune the correction process, ensuring that the educational and formative aspects are not lost in a black-box approach.

A distinctive strength of OCULUS lies in its generality: any system of ODEs expressible in Bolza form can be embedded in the framework and solved through indirect methods. This opens applications far beyond the examples shown here. Topics already explored include solar array orientation for maximizing harvested energy, four-dimensional aircraft trajectory optimization under variable winds and weather conditions, energy management

in hybrid-electric propulsion systems, spacecraft attitude control, robotics, and advanced mission concepts such as nuclear propulsion with variable specific impulse for interstellar transfers. Such versatility allows the same core infrastructure to serve both as a research testbed and as a prototyping tool for industrial contexts, wherever dynamical systems and constrained optimization intersect.

Development of OCULUS proceeds iteratively to address emerging user needs and improve robustness. A public release under license, including a dedicated system for community feedback and bug reporting, is planned in the near future. Looking ahead, OCULUS has the potential to extend well beyond academic settings.

The author hopes that the adoption of this tool will allow a wider community to engage with the fascinating field of optimal control and trajectory optimization, combining the accessibility of modern software with the rigor and educational richness of indirect methods.

List of references

[1] R. Mason, Simon, B. A. Becker, T. Crick, and J. H. Davenport, "A Global Survey of Introductory Programming Courses," 2024. DOI: 10.1145/3626252.3630761.

[2] S. Fantinelli, M. Cortini, T. Di Fiore, S. Iervese, and T. Galanti, "Bridging the gap between theoretical learning and practical application: A qualitative study in the italian educational context," *Education Sciences*, vol. 14, no. 2, 2024. DOI: 10.3390/educsci14020198.

[3] D. C. GEARY, "An evolutionarily informed education science," *Educational Psychologist*, vol. 43, no. 4, pp. 179–195, 2008. DOI: 10.1080/00461520802392133.

[4] F. Lespiau and A. Tricot, "Primary vs. secondary knowledge contents in reasoning: Motivated and efficient vs. overburdened," *Acta Psychologica*, vol. 227, p. 103 610, 2022, ISSN: 0001-6918. DOI: 10.1016/j.actpsy.2022.103610.

[5] A. T. Stull and R. E. Mayer, "Learning by doing versus learning by viewing: Three experimental comparisons of learner-generated versus author-provided graphic organizers," *Journal of Educational Psychology*, vol. 99, no. 4, pp. 808–820, 2007. DOI: 10.1037/0022-0663.99.4.808.

[6] R. Ploetzner and B. Fillisch, "Not the silver bullet: Learner-generated drawings make it difficult to understand broader spatiotemporal structures in complex animations," *Learning and Instruction*, vol. 47, pp. 13–24, 2017. DOI: 10.1016/j.learninstruc.2016.10.002.

[7] A. Skulmowski, "Learning by doing or doing without learning? the potentials and challenges of activity-based learning," *Educational Psychology Review*, vol. 36, p. 28, 2024. DOI: 10.1007/s10648-024-09869-y.

[8] J. Sweller, J. J. G. van Merrienboer, and F. G. W. C. Paas, "Cognitive architecture and instructional design," *Educational Psychology Review*, vol. 10, no. 3, pp. 251–296, 1998. DOI: 10.1023/A:1022193728205.

[9] J. Sweller, J. J. G. van Merrienboer, and F. G. W. C. Paas, "Cognitive architecture and instructional design: 20 years later," *Educational Psychology Review*, vol. 31, no. 2, pp. 261–292, 2019. DOI: 10.1007/s10648-019-09465-5.

[10] L. Fiorella and R. E. Mayer, "Eight ways to promote generative learning," *Educational Psychology Review*, vol. 28, no. 4, pp. 717–741, 2016. DOI: 10.1007/s10648-015-9348-9.

[11] L. Casalino and L. Mascolo, "Pontryagin's Principle for the Optimization of Escape Trajectories from Earth-Moon L2," in *New Trends and Challenges in Optimization Theory Applied to Space Engineering*, P. Cannarsa, A. Celletti, G. Fasano, L. Mazzini, M. Pontani, and E. Trélat, Eds., Cham: Springer Nature Switzerland, 2025, pp. 9–22, ISBN: 978-3-031-81253-8. DOI: 10.1007/978-3-031-81253-8_2.

[12] L. Mascolo and L. Casalino, "Optimal Escape from Sun-Earth and Earth-Moon L2 with Electric Propulsion," *Aerospace*, vol. 9, no. 4, 2022, ISSN: 2226-4310. DOI: 10.3390/aerospace9040186.

[13] L. Mascolo, "Low-Thrust Optimal Escape Trajectories from Lagrangian Points and Quasi-Periodic Orbits in a High-Fidelity Model," IRIS institutional repository, Ph.D. dissertation, Politecnico di Torino, 2023. [Online]. Available: https://iris.polito.it/handle/11583/2976595.

[14] R. Fößmeier, "On Richardson Extrapolation for Finite Difference Methods on Regular Grids," *Numerische Mathematik*, vol. 55, no. 4, pp. 451–462, 1987. DOI: 10.1007/BF01396048.

[15] S. D. Cohen and A. C. Hindmarsh, "CVODE, A Stiff/Nonstiff ODE Solver in C," *Computers in Physics*, vol. 10, no. 2, pp. 138–143, 1996. DOI: 10.1063/1.4822377.