VETOR

Multi-Objective Optimization of an Adaptive Numerical Integrator Using Optuna☆

Otimização Multiobjetiva de um Integrador Numérico Adaptativo Utilizando o Optuna

Daniel Augusto de Sousa Mendes[1†], Rafael Pereira Bianchin[1], Ângelo de Carvalho Paulino[1], Angelo Passaro[1]

[1]*Instituto de Estudos Avançados, São José dos Campos, Brasil*
[†]**Corresponding author:** adsmendesdaniel@gmail.com

Abstract

The numerical integration of ordinary differential equations is one of the most common problems in numerical analysis. Numerous methods and techniques are available in the literature, as are different combinations of heuristics and parameters. Therefore, appropriate choices must be made to minimize both numerical error and computational time. In this work, Optuna was employed as a multi-objective optimization framework to search for parameters of an adaptive fourth- and fifth-order Runge–Kutta time integrator (RK45) in three selected test cases. Subsequently, a Pareto front analysis methodology based on simple geometric metrics was developed using an expert-system-like approach. This methodology uses the Pareto-optimal set obtained with Optuna to select parameter configurations that effectively balance numerical error and simulation execution time. It was also tested with four benchmark objective functions to assess its capabilities, providing a straightforward way to select parameters that balance error and execution time. Optuna proved to be an effective tool for prototyping optimization-based methodologies, demonstrating its underexplored potential in tasks beyond machine learning.

Keywords

Multi-Objective Optimization • Adaptive Time Integration • Runge–Kutta Methods • Optuna • Numerical Simulation

Resumo

A integração numérica de equações diferenciais ordinárias é um dos problemas mais comuns em análise numérica. Diversos métodos e técnicas estão disponíveis na literatura, assim como diferentes combinações de heurísticas e parâmetros. Portanto, escolhas adequadas devem ser feitas para minimizar tanto o erro numérico quanto o tempo computacional. Neste trabalho, o Optuna foi empregado como um *framework* de otimização multiobjetivo para buscar os parâmetros de um integrador temporal adaptativo Runge–Kutta de quarta e quinta ordem (RK45) em três casos de teste selecionados. Em seguida, uma metodologia de análise da fronteira de Pareto baseada em métricas geométricas simples foi desenvolvida usando uma abordagem semelhante a um sistema especialista. Essa metodologia usa o conjunto Pareto-ótimo obtido com o Optuna para selecionar configurações de parâmetros que equilibram efetivamente o erro numérico e o tempo de execução da simulação. Ela também foi testada com quatro funções objetivo de referência para avaliar suas capacidades, fornecendo uma forma direta de selecionar parâmetros que equilibram erro e tempo de execução. O Optuna mostrou-se uma ferramenta efetiva para prototipar metodologias baseadas em otimização, demonstrando seu potencial ainda pouco explorado em tarefas além do aprendizado de máquina.

☆This article is an extended version of the work presented at the XXVIII National Meeting on Computational Modeling (ENMC) & XVI Meeting on Science and Technology of Materials (ECTM), held in Montes Claros, Brazil, October 21–24, 2025.

Palavras-chave

Otimização Multiobjetiva • Integração Temporal Adaptativa • Métodos de Runge–Kutta • Optuna • Simulação Numérica

1 Introduction

In numerical simulations of physical phenomena, numerical integration is often of utmost importance regarding accuracy and performance. The computation of the derivatives is usually expensive, and one desires to use large step sizes to avoid computing them too often. To ensure accuracy, adaptive integrators are used, which allow for the use of the largest step size that will keep the truncation error within tolerance [1, 2, 3].

To fine-tune the behavior of such adaptive integrators, especially in terms of error tolerance and step size control, optimization frameworks can be used. Optuna is an open-source hyperparameter optimization framework designed to be flexible and efficient for automated machine learning (ML) workflows [4, 5], especially neural networks (tuning number of neurons, number of layers, optimizer, architecture, etc). It supports single and multi-objective optimization tasks, and it is known for its ease of use and integration with scientific tools in Python [6, 7]. Despite its popularity in ML, its use in a broader scientific computing context remains unexplored.

The optimization of hyperparameters to obtain reliable results in a reduced time is essential for the application of computational optimization methods in solving engineering problems, as illustrated by the multi-objective design and performance optimization of hypersonic airbreathing vehicles and scramjet inlets [8, 9]. Many other engineering models, such as the stabilization of flexible hypersonic vehicle dynamics, are governed by systems of nonlinear ordinary differential equations, whose accurate numerical solution is essential for control design and simulation [10]. This is particularly relevant in inverse problems, for instance, when an ordinary equation must be solved several times in order to find a solution [11, 12].

In this context, we present an implementation of the Runge–Kutta method of fourth and fifth order (RK45) [1, 13, 14] featuring seven adjustable parameters that govern step size adaptation and truncation error control. To evaluate the influence of these parameters on performance and accuracy, a set of ODE test problems with distinct numerical characteristics is considered. In addition, standard multi-objective optimization benchmark problems are employed to analyze the behavior of the Pareto front and to assess the proposed selection methodology independently of the numerical integration context [15, 16]. A multi-objective optimization strategy is then applied to explore trade-offs between execution time and numerical error (measured against analytical solutions) [17, 18]. A Pareto front analysis methodology is developed following a specialist-system-like approach. This method takes a Pareto optimal set generated by Optuna and applies simple metrics of a geometric nature to select an optimal point that balances error (RMSE) and execution time [16, 19]. Optuna proved to be an effective optimization tool, particularly suitable for prototyping routines that involve multi-objective optimization. For each selected ODE test case, a distinct configuration of RK45 parameters is identified, leading to improvements in both numerical error and execution time, effectively leveraging trade-offs between these two objectives.

2 Theory

The focus of this work is the analysis of the RK45 integrator applied to ODE-based benchmark problems, described in Section 2.1 and Section 2.2. In order to assess the behavior of the proposed Pareto front analysis and selection methodology in a controlled setting, a set of standard multi-objective optimization benchmark problems is considered, presented in Section 2.5.

2.1 Runge–Kutta Integrator

The Runge–Kutta class of methods is very vast, but the most used are certainly adaptive methods of fourth and fifth order [1]. Being an adaptive method, it is well suited for most classes of problems with the exception, possibly, of symplectic problems (such as those that involve conserved variables in mechanics) or stiff problems (that often arise in chemical kinetics, for instance). In this report, a particular implementation of the Runge–Kutta methods will be discussed, the Cash–Karp method, which is of fifth order, using a fourth order embedded method to compute the truncation error [1]. The coefficients of the Cash–Karp RK45 scheme, represented by its Butcher tableau, follow the formulation presented in Section 16.2 of *Numerical Recipes in C* [1] and were not modified in this work. To achieve a good balance between accuracy and computational cost, the optimized parameters are:

a) relative and absolute tolerances (*rtol* and *atol*);
b) maximum and minimum step sizes (*max_step* and *minstep*);
c) a multiplier for the calculated new step size (*multiplier*);
d) the maximum ratio between consecutive steps (*maxratio*); and

e) the maximum number of iterations within a step (*maxit*).

2.2 Selected ODE Test Cases

The RK45 integrator was used to solve three distinct ordinary differential equation problems. The first problem, called 'Spring-mass', is a model of two masses connected by a spring [20]. The problem is described by the following set of equations and initial conditions:

$$\begin{bmatrix}2\\4\end{bmatrix}\ddot{\mathbf{X}}+\begin{bmatrix}12&-4\\-4&4\end{bmatrix}\mathbf{X}=0,\quad \mathbf{X}(0)=\begin{bmatrix}0\\1\end{bmatrix},\quad \dot{\mathbf{X}}(0)=\begin{bmatrix}0\\0\end{bmatrix} \tag{1}$$

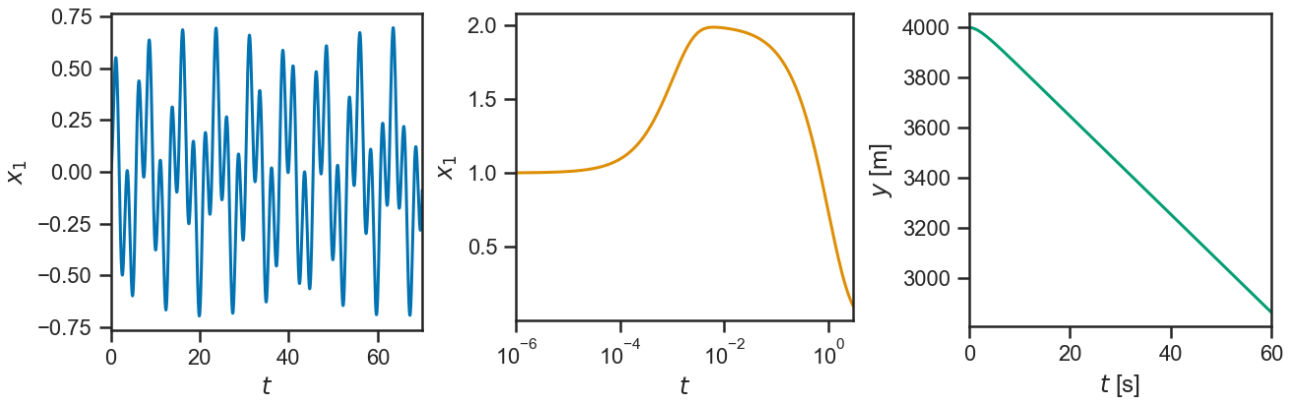
This case represents an oscillatory problem which is 'well-behaved', but cannot have arbitrarily large step sizes. The exact solution for x_1 is $x_1 = 0.3481\,(\cos(0.7923\,t)-\cos(2.5243\,t))$, which is shown in Fig. 1a. The second problem, called 'Stiff', is a simple stiff set of differential equations proposed by [21]:

$$\dot{\mathbf{X}}+\begin{bmatrix}998&1998\\-999&-1999\end{bmatrix}\mathbf{X}=0,\quad \mathbf{X}(0)=\begin{bmatrix}1\\0\end{bmatrix} \tag{2}$$

The exact solution for x_1 is $x_1 = 2\exp(-t)-\exp(-1000\,t)$. As Fig. 1b shows, there are solution components with very different time scales, typical of stiff problems. This problem was included because explicit solvers such as RK45 usually have poor performance with stiff problems. The third problem, called 'Parachute', is a simple model of a falling body experiencing drag [22]:

$$\ddot{y}=-g-\kappa\dot{y},\quad y(0)=4000\text{ m},\quad \dot{y}(0)=0 \tag{3}$$

where $g = 9.81\text{ m/s}^2$ and $\kappa = 0.5\text{ s}^{-1}$. As the body reaches terminal velocity, the numerical solution becomes nearly exact, due to the order of the integrator. The exact solution, $y = y(0)+g\kappa^{-2}(1-\kappa t-\exp(-\kappa t))$, shown in Fig. 1c, is mostly linear except near $t = 0$.



(a) Spring-mass (b) Stiff (c) Parachute

Figure 1: Exact solutions for the test cases.

2.3 Multi-Objective Optimization Problem (MOP)

A multi-objective optimization problem (MOP) involves minimizing or maximizing several conflicting objective functions simultaneously, subject to a set of inequality and equality constraints, as well as bounds on the decision variables [23, 24]. The solution is not a single optimum but a Pareto optimal set—solutions that are not dominated by any others, meaning improving one objective would degrade at least one other [18, 25]. The Pareto front represents these solutions in objective space and can take various forms, including convex, concave, or discontinuous shapes [23]. To solve MOPs, optimization techniques are broadly categorized into deterministic and stochastic methods. Deterministic methods, often gradient-based, can become trapped in local optima, especially in multimodal landscapes [26]. Stochastic methods, such as metaheuristic algorithms, aim to efficiently explore the search space to find near-optimal solutions and avoid premature convergence [24]. These methods are particularly useful for approximating well-distributed Pareto fronts, and frameworks like Optuna are commonly used to implement them effectively.

2.4 Optuna

Optuna is structured around a few core concepts: A trial represents a single evaluation of the objective function with a specific set of parameters values. Each trial is recorded with its inputs and results. A study is the overall optimization tasks, managing the collection of trials, tracking progress, storing the best results and providing a basic analysis of it. To decide which parameters to test next, Optuna employs samplers [7], state-of-art optimization algorithms to guide the search for optimal parameter configurations, like the 'TPESampler' [27]. Also, Optuna offers native support for parallel execution, enabling easy distribution of trials across multiple CPUs or computing nodes.

2.5 Multi-Objective Benchmark Problems

Classical multi-objective optimization benchmark problems are widely used to evaluate the performance of optimization algorithms and to analyze the structure of Pareto fronts under different constraint conditions. In this work, four standard benchmark problems are employed to assess the capability of Optuna in solving classical multi-objective optimization problems: ConstrEx, Osyczka–Kundu, CTP1, and Binh–Korn.

The **ConstrEx problem** is a constrained test problem with unbalanced ranges of the objective functions [28]. Its Pareto front is convex and exhibits a single knee point, which is similar to the Pareto fronts obtained from the temporal integration problem addressed in this work. Its two objective functions $f_1(x_i)$ and $f_2(x_i)$ are given by:

$$\begin{aligned} f_1(x_i) &= x_1, \\ f_2(x_i) &= \frac{1+x_2}{x_1}, \end{aligned} \tag{4}$$

where x_i denotes the components of the decision vector $\mathbf{x} = (x_1, x_2)$. The search space is defined as $x_1 \in [0.1, 1]$ and $x_2 \in [0, 5]$, and the problem is subject to the following constraints:

$$\begin{aligned} 9x_1 - x_2 &\geq 1, \\ 9x_1 + x_2 &\geq 6. \end{aligned} \tag{5}$$

The **Osyczka–Kundu problem**, also referred to as OSY, is a multi-constrained optimization problem proposed by Osyczka and Kundu [29]. It exhibits a non-convex and continuous Pareto front with multiple knee points [30]. In this context, multiple knee points refer to distinct regions of the Pareto front where the local curvature is more pronounced or the trade-off between objectives changes markedly, indicating successive trade-off transitions rather than a single dominant compromise point. Its objective functions are given by [29]:

$$\begin{aligned} f_1(x_i) &= -25(x_1-2)^2 - (x_2-2)^2 - (x_3-1)^2 - (x_4-4)^2 - (x_5-1)^2, \\ f_2(x_i) &= \sum_{i=1}^{6} x_i^2, \end{aligned} \tag{6}$$

in the space domain given by $x_1, x_2, x_6 \in [0, 10]$, $x_3, x_5 \in [1, 5]$, and $x_4 \in [0, 6]$ and subject to the following six constraints:

$$\begin{aligned} x_1 + x_2 - 2 &\geq 0, \\ 6 - x_1 - x_2 &\geq 0, \\ 2 - x_2 + x_1 &\geq 0, \\ 2 - x_1 + 3x_2 &\geq 0, \\ 4 - (x_3 - 3)^2 - x_4 &\geq 0, \\ (x_5 - 3)^2 + x_6 - 4 &\geq 0. \end{aligned} \tag{7}$$

The **CTP1 problem** was designed to introduce difficulties for optimization algorithms near the Pareto-optimal front due to its constraint formulation [31]. Its objective functions are given by:

$$\begin{aligned} f_1(x_i) &= x_1, \\ f_2(x_i) &= (1+x_2)\exp\left(-\frac{x_1}{1+x_2}\right), \end{aligned} \tag{8}$$

with the search space given by $x_1, x_2 \in [0, 1]$, subject to the following constraints:

$$\begin{aligned} \frac{f_2(x_i)}{0.858\exp\left(-0.541\, f_1(x_i)\right)} &\geq 1, \\ \frac{f_2(x_i)}{0.728\exp\left(-0.295\, f_1(x_i)\right)} &\geq 1. \end{aligned} \tag{9}$$

Proposed by Binh and Korn [32], the **Binh–Korn problem** has its objective functions defined as follows:

$$\begin{aligned} f_1(x_i) &= 4x_1^2 + 4x_2^2, \\ f_2(x_i) &= (x_1 - 5)^2 + (x_2 - 5)^2 \end{aligned} \tag{10}$$

in the search domain $x_1 \in [0, 5]$ and $x_2 \in [0, 3]$. In this problem, some solutions lie on the boundaries of the feasible region, and the objectives are strongly conflicting, such that a small reduction in $f_2(x_i)$ leads to a significant increase in $f_1(x_i)$. This problem also has multiple knee points, but its Pareto front is convex. It is subject to the following constraints:

$$\begin{aligned} (x_1 - 5)^2 + x_2^2 &< 25, \\ (x_1 - 8)^2 + (x_2 + 3)^2 &\geq 7.7. \end{aligned} \tag{11}$$

Each of the selected benchmark cases presents its own challenges for optimization algorithms. They were chosen to enable a comparison between the results obtained with Optuna and those reported in the literature, allowing an assessment of the reliability of the proposed framework and highlighting its potential limitations when compared to other optimization methods. In particular, the ConstrEx problem is of special interest, as it exhibits a single knee point, resembling the behavior observed in the test cases associated with the temporal integration problem.

3 Methods

The Runge–Kutta method with Carsh-Karp parameters was implemented as a Python function. This method uses truncation error estimation to adjust the step size and verify if a step error is within tolerance. The number of attempts to correct the same step is limited, as well as the growth rate of the step size. After the solution is obtained, the root mean square error (RMSE) of a trial is calculated based on the corresponding analytical solution.

3.1 Optimization Process

For the ODE-based benchmark problems, the formulated optimization problem to be solved with Optuna can be characterized as a multi-objective problem corresponding to the minimization of the RMSE — between the predicted solution and its corresponding exact solution — and the simulation execution time (in seconds), as shown in Eq. 12. Table 1 shows the values used in RK45, obtained through empirical testing, and a sampling range defined around the standard value, either on logarithmic or linear scale. The values defined empirically are named standard values from now on. Trials were also bounded by a 30 s timeout.

$$\min_x \left(f_1(x), f_2(x)\right) = \min_x \left(\text{RMSE}, \text{Time}(x)\right) \tag{12}$$

Table 1: Summarization of the tunable hyperparameters used in the optimization routine.

Parameters	Standard Value	Search Domain	Sampling
rtol, atol	1×10^{-6}	$\left[1 \times 10^{-8}, 1 \times 10^{-2}\right]$	Logarithmic
max_step	∞	$\left[1 \times 10^{-6}, 1\right]$	Logarithmic
minstep	5×10^{-1}	$\left[1 \times 10^{-10}, 1 \times 10^{-2}\right]$	Logarithmic
multiplier	0.8	$[0.1, 1.2]$	Linear
maxratio	10	$[1, 50]$	Linear
maxit	10	$[1, 50]$	Linear

The chosen sampler was the metaheuristic '*TPESampler*' [27] as it supports float and integer parameters as inputs, multi-objective and multivariate optimization and constrained optimization. For each test case a study with 10,000 trials was planned. All the 30,000 simulations (10,000 per ODE test case) were run on an Intel Core i7-8700 CPU personal computer, parallelizing the optimization pipeline with 12 processes. The total time of computing was 4h58 for the Spring-mass case, 4h19 for the Stiff and 4h07 for the Parachute case.

For the multi-objective benchmark problems described in Section 2.5, a similar optimization procedure was adopted, with 10,000 trials performed for each case. However, the search domain was defined individually for each problem, as specified in the corresponding section. The feasibility of each trial was determined solely by the constraints associated with each benchmark problem. The same metaheuristic, *TPESampler*, was employed, as in the ODE-based tests, with all variables sampled using a linear scale. These experiments were carried out on an Intel

Core i5-10210U CPU, with the optimization process parallelized across 12 processes, resulting in a total execution time of approximately 1 hour per benchmark case.

3.2 Choosing an Ideal Value on the Pareto Front

With all the Optuna studies complete, a post processing of the pareto front was initiated with three simple metrics, named here as Knee-point Detection [16], Distance-to-ideal Selection — similar to the 'utopia' point method proposed by [18] — and ϵ-Constraint Filtering [24].

In the Knee-point Detection, a Pareto optimal set $\{(e_i, t_i)\}$ (RMSE vs. execution time) is approximated as a 2D curve. A chord is drawn between its two anchor points and each intermediate point's perpendicular distance to that chord is computed, and the point of maximum distance is returned as the 'knee'. This point marks the threshold where any further reduction in time incurs a steep increase in error, and vice versa.

The Distance-to-ideal Selection corresponds to taking an ideal reference point (e^*, t^*) (in this work, this ideal point was set to $(0, 0)$) and calculate every Pareto distance to this point. The last implemented method, the ϵ-Constraint Filtering, is applied when a threshold $e_{\max}$ or $t_{\max}$ is provided so all points on the Pareto front violating this bounds are discarded (leaving us with all points with $e_i \leq e_{\max}$ and/or $t_i \leq t_{\max}$). In the current implementation, the single best value index, k, of the other objective is chosen:

$$k = \begin{cases} \text{argmin}_{t_i \leq t_{\max}} e_i & \text{if } t_{\max} \text{ is provided} \\ \text{argmin}_{e_i \leq e_{\max}} t_i & \text{if } e_{\max} \text{ is provided.} \end{cases} \tag{13}$$

To extract a single best solution from the Optuna-derived Pareto front, a rule-based expert-system pipeline was developed: If a maximum execution-time threshold $t_{\max}$ is specified, 'ϵ-Constraint Filtering' excludes all points with $t_i \geq t_{\max}$ and selects the one with minimum error; if only an error threshold $e_{\max}$ is provided, points with $e_i \geq e_{\max}$ are removed and the fastest set of parameters is chosen. When both thresholds are given, dual 'ϵ-Constraint Filtering' filtering is applied before further selection. In the absence of any threshold (or once filtering is complete), 'Knee-point Detection' (maximizing the perpendicular distance to the chord between the extreme front endpoints) and 'Distance-to-ideal Selection' (minimizing Euclidean distance to a reference utopia) are executed to yield the final best compromise solution(s). This process is shown in Fig. 2.

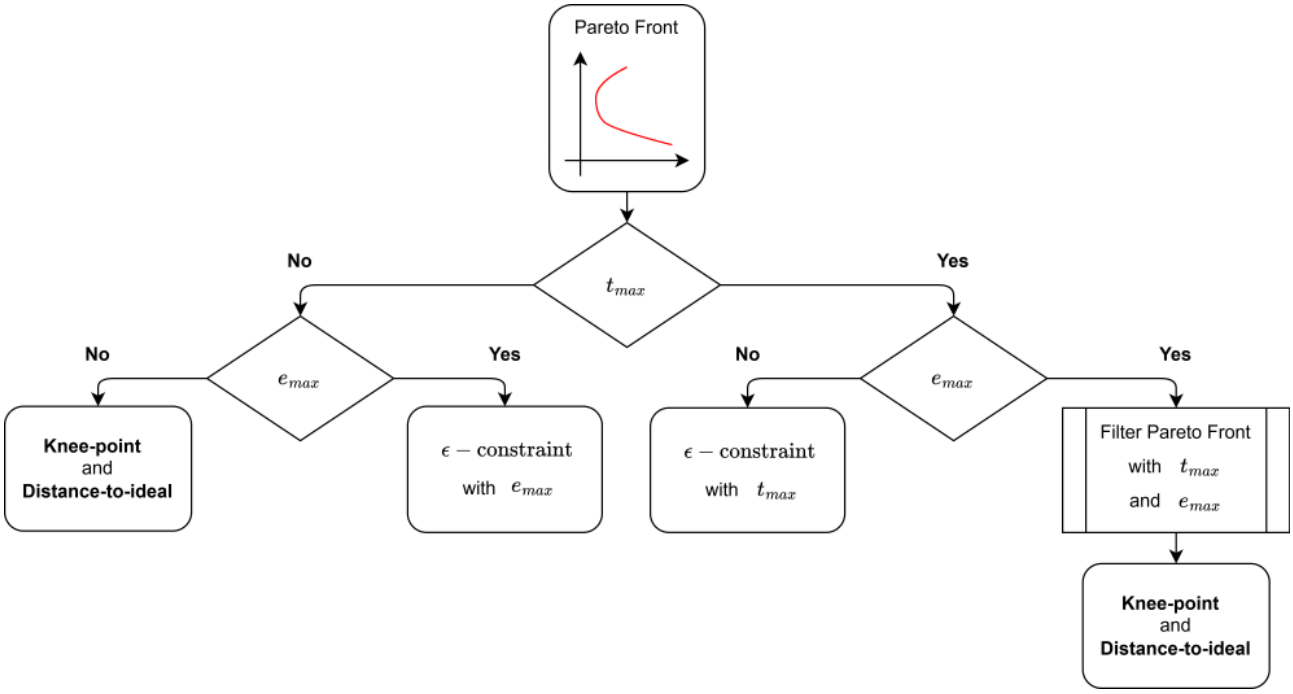


Figure 2: Flowchart of the expert-system-like selection pipeline applied to the Optuna-derived Pareto optimal set $\{(e_i, t_i)\}$.

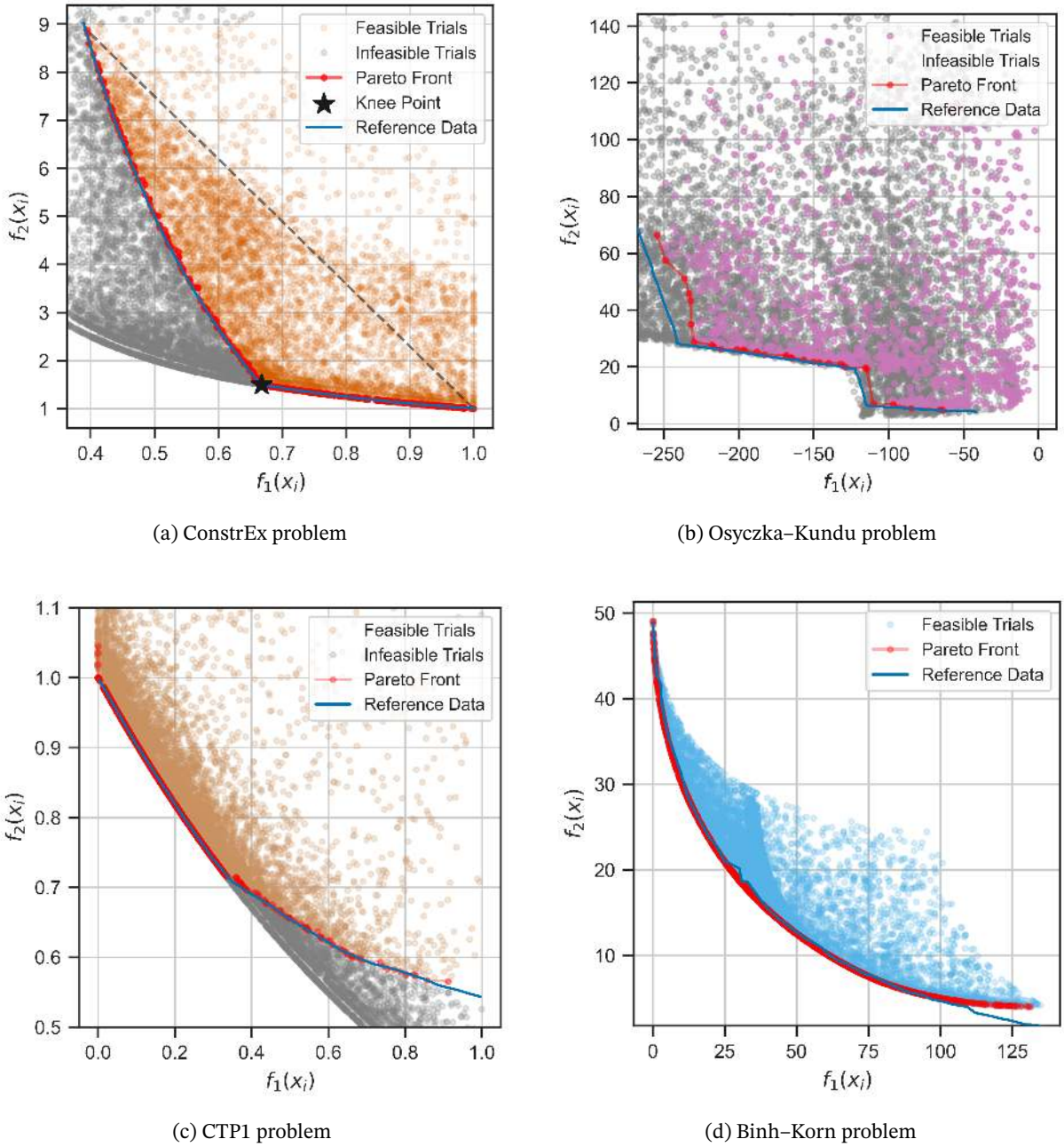
For every study done in this work, some points of the Pareto front were dropped out after the sorting, to ensure its convexity, which is necessary for the application of some metrics. In the Spring-mass case, 9 points were dropped out, 0 for the Stiff case and 2 for the Parachute.

4 Results and Discussion

In this section, the methodology described in Section 3 will be applied first to the benchmark problems as a test to assess the behavior of the proposed analysis. Afterwards, the methodology will be applied to the optimization of the Runge–Kutta integrator for each of the ODE test cases.

4.1 Multi-Objective Benchmark Results

This section discusses the results obtained for the multi-objective benchmark problems described in Section 2.5, presented in Fig. 3. In each plot in Fig. 3, the red line and markers represent the Pareto front obtained with Optuna, while the blue line corresponds to the reference data for each benchmark case. Colored markers indicate feasible trials, whereas gray markers denote trials that violate the constraint conditions.



(a) ConstrEx problem

(b) Osyczka–Kundu problem

(c) CTP1 problem

(d) Binh–Korn problem

Figure 3: Multi-objective benchmark results.

Figure 3a shows the results obtained for the ConstrEx problem, along with the knee point identified using the chord method. This problem is convex and exhibits a single knee point, making it well suited for the selection method described in Section 3.2, and similar to the behavior observed in the ODE test cases. The results indicate that the knee point is correctly identified, and the obtained Pareto front correctly matches the reference data extracted from [28].

The results obtained for the Osyczka–Kundu problem, shown in Fig. 3b, do not exhibit the same level of agreement with the reference data [30] as observed for the ConstrEx problem. Nevertheless, the obtained Pareto front presents the expected shape, with three knee regions. This problem is known to be challenging for optimization algorithms, both in locating solutions near the Pareto front and in exploring the entire search space [31]. This difficulty is reflected in the large number of infeasible solutions observed across the domain in the plot. Despite these challenges, a comparison with the original study [29] shows that 33 points were obtained on the Pareto front, compared to 28 reported in the original work, while using a smaller number of trials (10,000 in this work versus 20,000 in the original study).

Figure 3c represents the results for the CTP1 problem. This problem has two subtle knees [31]. In the results obtained by Optuna, a third knee was identified around $(f_1, f_2) = (0, 1)$, and the two original knees are less noticeable. This is possibly due to an insufficient number of trials. Neverthless, the results demonstrate excellent agreement between the obtained Pareto front and the reference data.

Figure 3d represents the results for the Binh–Korn problem. For this problem, no infeasible trials were found by Optuna, possibly due to the objective functions and constraints being more 'well-behaved'. The results agree very well with the reference data from [32], with a Pareto front that is smoother than that presented in the original study for this problem. Moreover, the seemingly spurious behavior found in the Pareto front presented by [32] for $f_1 \gtrsim 100$ is not reproduced by Optuna. This spurious behavior can be attributed to inaccuracies introduced during the extraction of the reference data from the original plot.

Among the benchmark problems considered, the ConstrEx case is the one that most closely resembles the characteristics of the ODE-based optimization problems addressed in this work, as will be shown in the next section, particularly due to being convex and the presence of a single, well-defined knee point. For this case, the proposed methodology demonstrated excellent agreement with the reference data and accurately identified the representative solution on the Pareto front. Although some discrepancies were observed in the other benchmark problems (especially the Osyczka-Kundu and CTP1 problems), these are associated with known difficulties of such cases. Overall, the results indicate that the proposed framework is sufficiently robust to capture the main features of the Pareto front and to support the selection of representative solutions, thus providing confidence in its application to the ODE-based problems considered in this study.

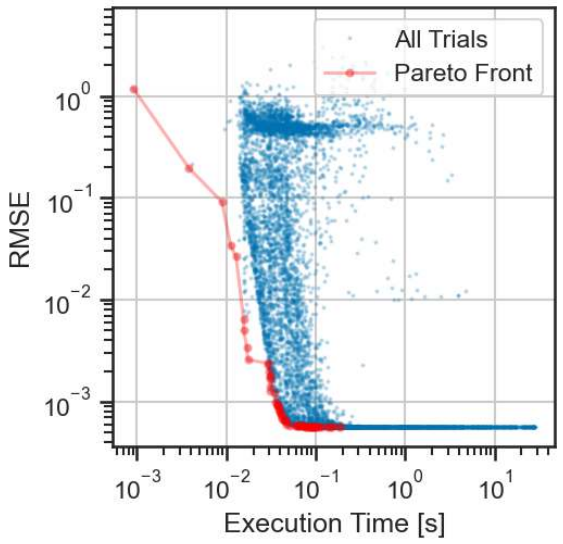
4.2 ODE Test Cases Results

The Optuna-driven parameter search resulted in a Pareto front of RMSE vs. execution time for each test case. Figure 4b illustrates the distribution of all trials and the non-dominated Pareto set for the spring-mass case. In this case, the front revealed a sharp 'knee', at the approximate execution time of 5×10^{-2} s. The 'knee' point can be automatically obtained with the methodology described in Section 3.2, without the specification of a $e_{\max}$ or $t_{\max}$, as can be seen in Fig. 4a and Fig. 4b, for both linear and log scale. Figure 4a shows all trials and the full Pareto optimal set. Figure 4b shows the region near the closest-to-ideal point (without the outliers), which is defined as the point that is closest to the ideal point at origin (i.e., where both error and execution time are zero).

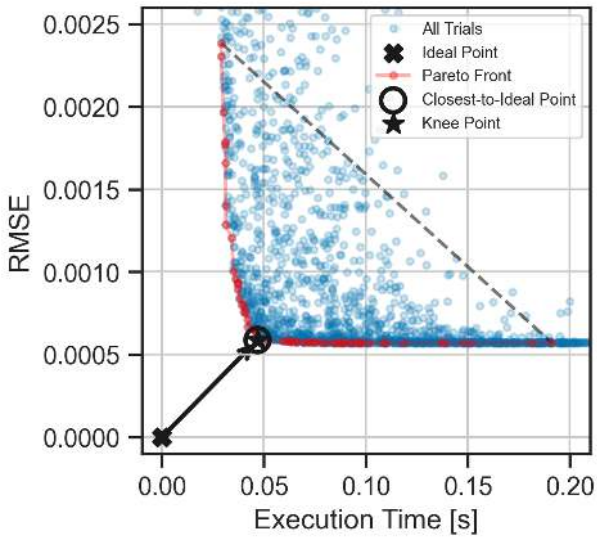
Figure 5a presents the results of the Optuna optimization for the stiff test case. The region preceding this knee exhibits a dense cluster of configurations, indicating that variations in the time integrator's parameters produce relatively short execution times but with significant fluctuations in error. Additionally, a second region, toward the bottom right, can be observed, representing a secondary front composed entirely of dominated solutions. The presence of these two regions highlights the necessity of sampling a large number of configurations in order to recover the true Pareto-optimal front. Moreover, the discontinuity in the distribution of these points indicates that achieving low error generally requires a minimum execution time (on the order of 10^1 seconds). However, Optuna was able to identify configurations with low RMSE and execution times on the order of 10^{-1} second. Additionally, the results for the Parachute case exhibit a well-defined curvature, indicating a smooth trade-off between execution time and accuracy, as can be seen in Fig. 5b, with all the evaluated configurations shown in green and the resulting Pareto front in red.

4.3 Influence of Truncation Error

The performance of the integrator, which is of fifth order method for RK45, is limited by the truncation error. Assuming, for simplicity, a constant step size, one has RMSE $\propto h^{1/5}$. Since the execution time is proportional to h^{-1}, one can expect the RMSE to be bounded by RMSE $\propto t_{\text{execution}}^{-1/5}$. This relation can be visualized in Fig. 6a for the spring-mass case. In the parachute case, however, the error is found to increase with execution time for times larger than 0.1 s. Since the right-hand side of Eq. 3 becomes constant as velocity reaches its terminal value, the integrator

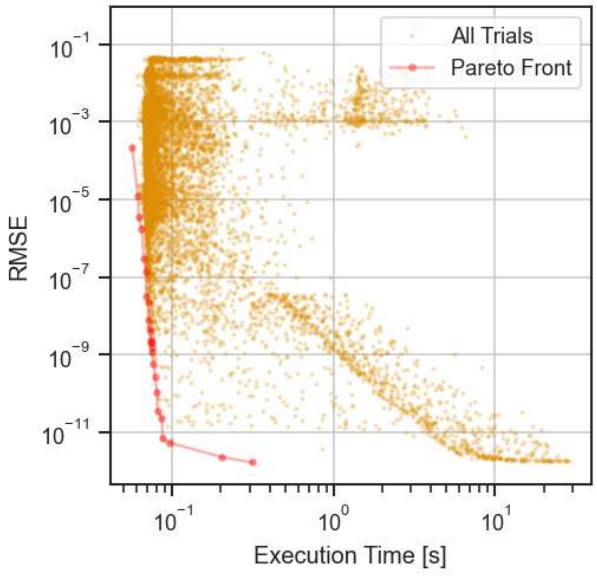


(a) Plot of all trials, along with the corresponding non-dominated solutions for the Spring-mass test case.

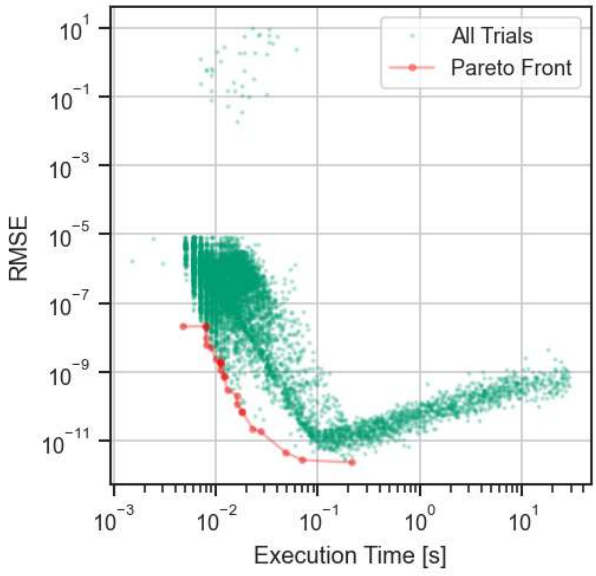


(b) Post-optimization results. The Pareto optimal set is shown along with the 'Knee Point' and the 'Closest-to-Ideal Point'.

Figure 4: Plot of all trials, spring-mass test case.



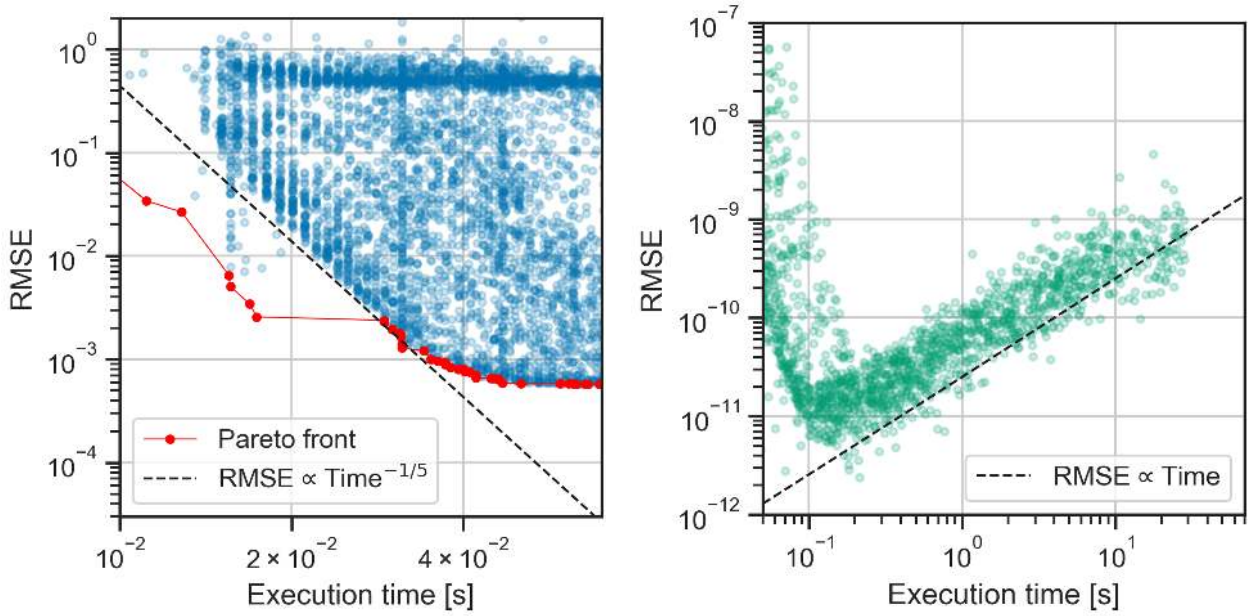
(a) Stiff case.



(b) Parachute case.

Figure 5: Plot of all trials along with the Pareto optimal set.

is able to obtain an exact solution (except for the initial transient). Therefore, decreasing the step size will not decrease truncation error or RMSE. However, a decrease of the step size leads to an increase of total steps, which in turn accumulates floating point errors. In that case, one can assume that the error added at each integration step is constant and, therefore, the overall error is proportional to the number of integration steps, which in turn is inversely proportional to execution time, giving RMSE $\propto t_{\text{execution}}$ (as shown in Fig. 6b).



(a) Mass-spring case.

(b) Parachute test case.

Figure 6: Detail of the objective space showing the relation of RMSE with execution time.

4.4 Best Parameters for Each Test Case

The best configuration of RK45 parameters for each test case can be seen in Table 2. The methodology described in Section 3.2 was applied without specifying $e_{\max}$ and $t_{\max}$ constraints. Values for the errors (RMSE), execution time and number of steps obtained with the standard values (Table 1) and with the best values can be seen in Table 3. For the spring-mass case, there is virtually no difference in the absolute error after optimization. There is, however, a significant improvement in execution time, as seen in Table 3.

Table 2: Best hyperparameters obtained by Pareto front analysis.

Parameters	Spring-Mass	Stiff	Parachute
rtol	1.80×10^{-6}	8.86×10^{-5}	1.39×10^{-8}
atol	4.12×10^{-6}	1.20×10^{-3}	1.03×10^{-6}
max_step	9.90×10^{-1}	9.11×10^{-2}	9.94×10^{-1}
minstep	1.46×10^{-5}	1.01×10^{-10}	8.22×10^{-8}
multiplier	0.711	0.202	0.201
maxratio	5.98	17.06	41.24
maxit	22	44	14

Table 3: Execution times, RMSEs and number of steps for each case using the standard and the optimized parameters. Values below 1×10^{-8} are considered 0.

	Execution Time [s]		RMSE		Number of Steps	
	Standard	Optimized	Standard	Optimized	Standard	Optimized
Spring-Mass	1.15×10^{-1}	6.32×10^{-2}	5.70×10^{-4}	5.88×10^{-4}	784	432
Stiff	7.69×10^{-2}	6.66×10^{-2}	3.92×10^{-7}	3.60×10^{-6}	827	819
Parachute	4.24×10^{-3}	1.41×10^{-2}	1.71×10^{-6}	0	41	153

For the stiff problem, optimization led to an increase in absolute error, despite a very small decrease in the number

of steps (Table 3). This is due to the stiffness of the problem, which prevents efficient integration. The optimization drastically decreased the absolute error in the parachute problem, while the number of steps increased.

4.5 Parameter Importance

Figure 7 represents the hyperparameter importance analysis for the spring-mass test case, with based on the 'PED-ANOVA', one of the methods to verify the parameter importance implemented in Optuna [7]. The results reveal that the *max_step* is the most influential parameter for reducing the RMSE, likely due to its role in controlling the maximum allowed step size of the integrator, where smaller steps yield more accurate solutions in stiff or sensitive dynamics. Decreasing *max_step* helps the integrator to control the truncation error, which can make the step more accurate than specified by *atol* and *rtol*, while increasing the execution time due to the use of smaller step sizes. The mean values obtained with PED-ANOVA for each test parameter can be seen on the bar chart of Fig. 7.

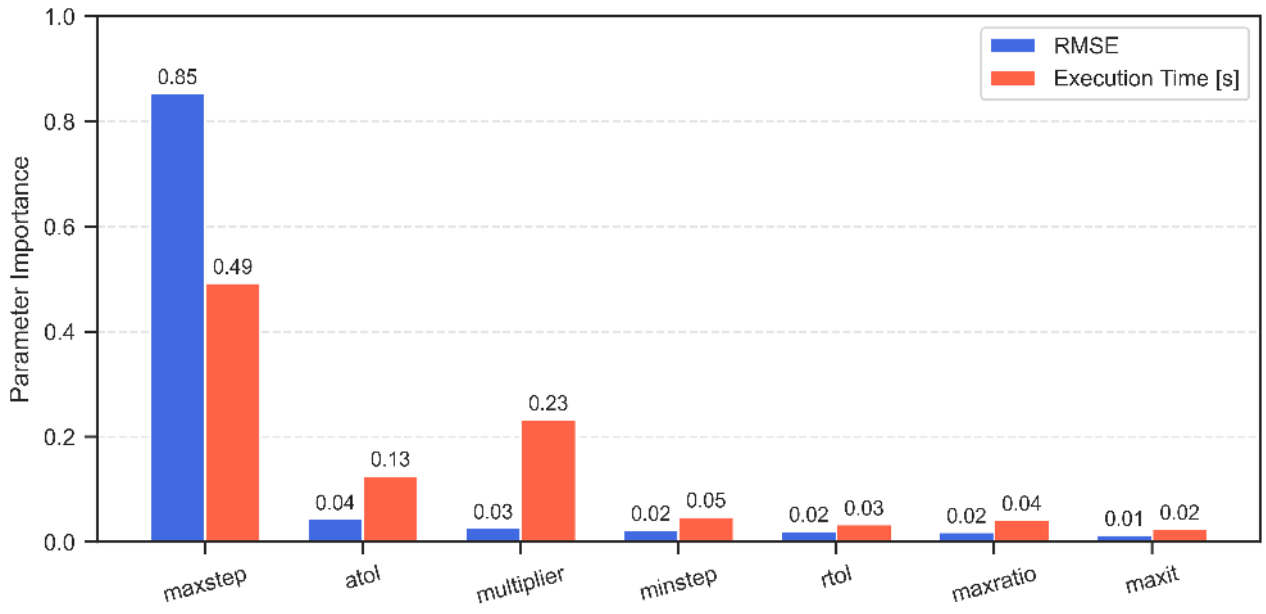


Figure 7: Bar chart illustrating the mean parameter importance scores for all the test cases as computed with PED-ANOVA method in Optuna, presented in decreasing order of importance.

Regarding execution time, both *max_step* and *multiplier* were the most influential parameters, in terms of execution time. This is expected, as *max_step* directly limits the number of integration steps, while *multiplier* adjusts the scaling factor in the adaptive time-stepping mechanism, affecting how aggressively the step size is increased or decreased. The value of *multiplier* is chosen in general to ensure the stability of the method and to keep the truncation error below tolerance. However, its small effect on the RMSE suggests that in the studied cases multiplier is not very important for stability.

A value of *max_step* in RK45 that yielded good results for one problem did not necessarily produce similar performance when applied to other problems. Although the integrator is adaptive by design, the results suggest that it may not be sufficiently adaptive for cross-problem generalization. This highlights the need to fine-tune the hyperparameters for each specific application. The complexity of the patterns observed in the objective space, along with the possible evidence of multimodality in the optimization landscape, indicates that manually selecting suitable parameter values may be impractical. Consequently, the use of an optimization algorithm becomes essential for achieving efficient configurations.

5 Conclusion

In this work, the Optuna hyperparameter optimization framework, commonly used for tuning machine learning models, was employed to optimize the parameters of the adaptive RK45 time integrator for selected test cases, after demonstrating its capabilities through four benchmark problems. The application presented here extended beyond typical machine learning tasks, demonstrating high utility in prototyping methods involving optimization algorithms.

Regarding the developed methodology, it was possible to automate the selection of an 'ideal' point for the adaptive RK45 time integrator, balancing both RMSE and simulation execution time, using a rule-based expert-system-like scheme built on top of simple geometric metrics. Across the test cases, some configurations led to reductions in execution time, while others led to reductions in error. Compared with the standard parameter configuration, the selected optimized configurations reduced execution time by approximately 45% for the Spring-Mass case and 13% for the Stiff case, whereas the Parachute case favored accuracy instead of speed, yielding negligible RMSE at the cost of a higher execution time. A *max_step* value in RK45 that worked well for one problem could fail to deliver similar performance on others, indicating limited cross-problem adaptability. Given the complexity and potential multi-modality of the optimization landscape, manual tuning is impractical, making automated optimization essential for efficient parameter selection.

Although the results obtained for the ODE-based problems were considered satisfactory, Optuna did not exhibit equally strong performance across all standard multi-objective benchmark cases. For the problems characterized by a single knee point or no pronounced knee region (ConstrEx and Binh–Korn, respectively), the results showed excellent agreement with the reference data. In contrast, for problems exhibiting multiple knee regions (Osyczka–Kundu and CTP1), the agreement with the reference data was less accurate. This behavior can be attributed to the inherent difficulty of these problems, particularly the Osyczka–Kundu case, which is specifically designed to challenge optimization algorithms across the entire search space. It is worth noting that the results may be further improved through a more refined configuration of the optimization process, such as exploring alternative samplers or different sampling strategies (such as logarithmic sampling). Nevertheless, the presence of complex constraint structures in these benchmark problems highlights limitations that are not evident in the ODE-based cases, which do not exhibit the same level of constraint complexity.

Finally, all the results presented in this work demonstrate that Optuna has the ability to provide effective optimization methods and holds significant untapped potential for problems beyond ML applications, particularly in prototyping environments or contexts that demand rapid development. The conclusions regarding the selected configurations are tied to the use of RMSE as the numerical-error objective. Evaluating the effect of alternative error metrics was outside the scope of this work and may be considered in future studies. In future studies, Optuna could be replaced by more customizable optimization tools that offer greater control over their internal mechanisms and source code. The expert-system-like decision pipeline can also be expanded with additional strategies for selecting points on Pareto front, as well as variations in input processing, such as applying a logarithmic transformation to RMSE while leaving execution time in its original scale.

Acknowledgements

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001 and Academic Cooperation Program in National Defense (PROCAD-DEFENSE), grant 88881.387753/2019-01.

References

[1] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C*, 2nd ed. Cambridge, UK: Cambridge University Press, 2002. Available at: https://s3.amazonaws.com/nrbook.com/book_C210.html

[2] E. Hairer, S. P. Nørsett, and G. Wanner, *Solving Ordinary Differential Equations I: Nonstiff Problems*, 2nd ed. Springer Berlin, Heidelberg, 1993. Available at: https://doi.org/10.1007/978-3-540-78862-1

[3] L. F. Shampine and M. W. Reichelt, "The MATLAB ODE suite," *SIAM Journal on Scientific Computing*, vol. 18, no. 1, pp. 1–22, 1997. Available at: https://doi.org/10.1137/S1064827594276424

[4] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," *arXiv, Computer Science, Machine Learning*, 2019. Available at: https://doi.org/10.48550/arXiv.1907.10902

[5] Optuna, "Optuna: A hyperparameter optimization framework," GitHub, 2025. Available at: https://github.com/optuna/optuna

[6] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. Anchorage, USA: Association for Computing Machinery, 2019, pp. 2623–2631. Available at: https://doi.org/10.1145/3292500.3330701

[7] Optuna Contributors, "Optuna: A hyperparameter optimization framework," Optuna Documentation, 2018. Available at: https://optuna.readthedocs.io/en/stable/index.html

[8] P. P. B. Araújo, R. Y. Tanaka, C. A. Silva, A. Passaro, and P. G. P. Toro, "Multi-objective optimization of a hypersonic airbreathing vehicle," *Physics of Fluids*, vol. 36, no. 2, p. 026120, 2024. Available at: https://doi.org/10.1063/5.0181366

[9] P. P. B. Araújo, M. V. S. Pereira, G. S. Marinho, J. F. A. Martos, and P. G. P. Toro, "Optimization of scramjet inlet based on temperature and mach number of supersonic combustion," *Aerospace Science and Technology*, vol. 116, p. 106864. Available at: https://doi.org/10.1016/j.ast.2021.106864

[10] J.-W. Wang and H.-N. Wu, "Exponentially stabilizing fuzzy controller design for a nonlinear ODE-beam cascaded system and its application to flexible air-breathing hypersonic vehicle," *Fuzzy Sets and Systems*, vol. 385, pp. 127–147, 2020. Available at: https://doi.org/10.1016/j.fss.2019.02.023

[11] R. C. Aster, B. Borchers, and C. H. Thurber, *Parameter Estimation and Inverse Problems*, 3rd ed. Elsevier, 2018. Available at: https://doi.org/10.1016/C2015-0-02458-3

[12] A. Raue, C. Kreutz, T. Maiwald, J. Bachmann, M. Schilling, U. Klingmüller, and J. Timmer, "Lessons learned from quantitative dynamical modeling in systems biology," *PLoS ONE*, vol. 8, no. 9, p. e74335, 2013. Available at: https://doi.org/10.1371/journal.pone.0074335

[13] J. R. Cash and A. H. Karp, "A variable order Runge-Kutta method for initial value problems with rapidly varying right-hand sides," *ACM Transactions on Mathematical Software*, vol. 16, no. 3, pp. 201–222, 1990. Available at: https://doi.org/10.1145/79505.79507

[14] J. R. Dormand and P. J. Prince, "A family of embedded Runge-Kutta formulae," *Journal of Computational and Applied Mathematics*, vol. 6, no. 1, pp. 19–26, 1980. Available at: https://doi.org/10.1016/0771-050X(80)90013-3

[15] E. Zitzler, K. Deb, and L. Thiele, "Comparison of multiobjective evolutionary algorithms: Empirical results," *Evolutionary Computation*, vol. 8, no. 2, pp. 173–195, 2000. Available at: https://doi.org/10.1162/106365600568202

[16] G. Yu, Y. Jin, and M. Olhofer, "Benchmark problems and performance indicators for search of knee points in multiobjective optimization," *IEEE Transactions on Cybernetics*, vol. 50, no. 8, pp. 3531–3544, 2020. Available at: https://doi.org/10.1109/TCYB.2019.2894664

[17] K. Deb, *Multi-Objective Optimization Using Evolutionary Algorithms*. John Wiley & Sons, 2001.

[18] N. Gunantara, "A review of multi-objective optimization: Methods and its applications," *Cogent Engineering*, vol. 5, no. 1, p. 1502242, 2018. Available at: https://doi.org/10.1080/23311916.2018.1502242

[19] I. Das and J. E. Dennis, "Normal-boundary intersection: A new method for generating the Pareto surface in nonlinear multicriteria optimization problems," *SIAM Journal on Optimization*, vol. 8, no. 3, pp. 631–657, 1998. Available at: https://doi.org/10.1137/S1052623496307510

[20] S. S. Rao, *Mechanical Vibrations*, 5th ed. Upper Saddle River, USA: Prentice-Hall, 2011.

[21] C. W. Gear, *Numerical Initial Value Problems in Ordinary Differential Equations*. Hoboken, USA: Prentice Hall, 1971.

[22] D. B. Meade, "ODE models for the parachute problem," *SIAM Review*, vol. 40, no. 2, pp. 327–332, 1998. Available at: https://doi.org/10.1137/S0036144596316248

[23] K.-L. Du and M. N. S. Swamy, *Search and Optimization by Metaheuristics: Techniques and Algorithms Inspired by Nature*. Birkhäuser Cham, 2016. Available at: https://doi.org/10.1007/978-3-319-41192-7

[24] E.-G. Talbi, *Metaheuristics: From Design to Implementation*. John Wiley & Sons, 2009. Available at: https://doi.org/10.1002/9780470496916

[25] Q. Liu, X. Li, H. Liu, and Z. Guo, "Multi-objective metaheuristics for discrete optimization problems: A review of the state-of-the-art," *Applied Soft Computing*, vol. 93, p. 106382, 2020. Available at: https://doi.org/10.1016/j.asoc.2020.106382

[26] X.-S. Yang, *Artificial Intelligence, Evolutionary Computing and Metaheuristics: In the Footsteps of Alan Turing*, 1st ed. Springer Berlin, Heidelberg, 2013. Available at: https://doi.org/10.1007/978-3-642-29694-9

[27] S. Watanabe, "Tree-structured parzen estimator: Understanding its algorithm components and their roles for better empirical performance," *arXiv, Computer Science, Machine Learning*, 2023. Available at: https://doi.org/10.48550/arXiv.2304.11127

[28] F. Rudzinski, "Finding sets of non-dominated solutions with high spread and well-balanced distribution using generalized strength pareto evolutionary algorithm," in *Proceedings of the 2015 Conference of the International Fuzzy Systems Association and the European Society for Fuzzy Logic and Technology*, Gijón, Spain, 2015, pp. 178–185. Available at: https://doi.org/10.2991/ifsa-eusflat-15.2015.28

[29] A. Osyczka and S. Kundu, "A new method to solve generalized multicriteria optimization problems using the simple genetic algorithm," *Structural optimization*, vol. 10, no. 2, pp. 94–99, 1995. Available at: https://doi.org/10.1007/BF01743536

[30] Z. Guo, Z. Zhou, L. Song, J. Li, and Z. Feng, "Aerodynamic analysis and multi-objective optimization design of a high pressure ratio centrifugal impeller," in *ASME Turbo Expo 2014: Turbine Technical Conference and Exposition*, Düsseldorf, Germany, 2014, p. V02DT42A013. Available at: https://doi.org/10.1115/GT2014-25496

[31] F. Jimenez, A. Gomez-Skarmeta, G. Sanchez, and K. Deb, "An evolutionary algorithm for constrained multi-objective optimization," in *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No.02TH8600)*, vol. 2, Honolulu, USA, 2002, pp. 1133–1138. Available at: https://doi.org/10.1109/CEC.2002.1004402

[32] T. T. Binh and U. Korn, "MOBES: A multiobjective evolution strategy for constrained optimization problems," in *The third international conference on genetic algorithms (Mendel 97)*, vol. 25, Brno, Czech Republic, 1997.