Warm-starting multi-start procedure using penalties instead of constraints to find more optimal trajectories

Eve Charbonneau[1], **Francisco Pascoa**[2], **Mickaël Begon**[1]

[1] Faculty of Medicine
Université de Montréal
2900 Bd. Édouard-Montpetit,
H3T 1J4 Montréal, Canada
eve.charbonneau.1@umontreal.ca
mickael.begon@umontreal.ca

[2] Faculty of arts and sciences
Université de Montréal
2900 Bd. Édouard-Montpetit,
H3T 1J4 Montréal, Canada
francisco.pascoa@umontreal.ca

ABSTRACT

For non-convex trajectory optimization, it is rarely possible to know if the optimal solution found using gradient descent corresponds to the global optimum. While searching for the global minimum, it is possible to use a multi-start approach by randomly initializing the problem. The initial guesses generated with this method might be far from any optimum. Since gradient descent methods are affected by the initial guess provided, this random initialization might impair the convergence. This study proposed a two-step optimization procedure aiming to find more optimal solutions to optimal control problems. First, the problem is modified to relax certain constraints by replacing them with a penalty term; this problem is solved with random initialization. This first step enables the generation of an initial guess closer to an optimum which is then used to initialize and solve the original fully constrained problem. This method was applied to a multiple shooting optimal control problem aiming to invert a pendulum without hitting several obstacles. The obstacle avoidance or continuity conditions were expressed as penalties when applying the two-step optimization procedure. The proposed method outperformed a randomly initialized multi-start regarding convergence rate, optimal cost, and computational time. Expressing the obstacle avoidance condition as a penalty reduced the computational time by 67%. Expressing the continuity or obstacle avoidance conditions as a penalty allowed discovering up to 3 better solutions (lower cost). It also increased the rate of efficient solutions found by 24% and 70%, respectively. Researchers interested in solving constrained optimal control problems to find innovative trajectories could benefit from using the two-step optimization procedure proposed in this study.

Keywords: Optimal control, Initial guess, Interior point method, Underconstrained optimization, Multi-start.

1 INTRODUCTION

Optimal control theory has been used by researchers from various fields, from aeronautics to biomechanics [1, 2, 3, 4]. This numerical method minimizes optimality criteria (*i.e.*, cost function) while complying with context-specific constraints. It is usually performed with a gradient descent method to find a local minimum [5]. However, when applying this method to non-convex problems, it is unknown if the local optimum found corresponds to the global one. Hence, a multi-start approach that solves the problem multiple times with different initial guesses is generally recommended as it increases the chance of finding a more global optimum [6, 7]. This multi-start approach is usually performed through a random initialization of the variables. However, the low quality of the random initial guess might affect the optimal control problem (OCP) convergence.

Finding fast optimization algorithms with a high convergence rate for non-convex problems is an open research topic. While the perfect optimization algorithm does not exist, some are better suited

for certain types of OCP problems. When a good initial guess cannot be acquired experimentally or predicted by the researchers, the interior point method (IPM) is usually preferred due to its high convergence rate and its robustness to the initialization [8]. However, IPM might be unable to find minima hidden in highly non-convex constrained domains. Indeed, if the non-convex set of constraints creates cavities, IPM cannot jump over the constraints to reach another portion of the feasible domain (Fig. 6, in Appendix 6.1). It would be advantageous to benefit from the IPM's high convergence rate while getting around its drawbacks. Starting the optimization at points beneath the constraint barriers by relaxing the constraints might achieve the desired goal.

In the last decades, several numerical transcriptions have arisen like parametric optimization [9], temporal finite elements [10], direct single shooting [11], direct multiple shooting (DMS) [12], direct collocation [13], differential dynamic programming [14], and discrete mechanics and optimal control for constrained systems [15]. Since implementing these transcriptions is not trivial, researchers usually choose one implementation that fits their needs better. The focus of this article is not to compare or improve these implementations, but rather to present an innovative problem initialization method. Indeed, it has been shown for direct single shooting that generating initial guesses closer to an optimum can enlarge the chances of finding an appropriate optimal solution [16]. In the present paper, the DMS implementation was chosen, but the improvements brought by the suggested initialization method are not specific to the chosen transcription method.

Implementing a DMS problem involves discretizing the continuous state ($\mathbf{x}$) and control ($\mathbf{u}$) variables into vectors of discrete variables $[\mathbf{x}_0, \mathbf{x}_1, ..., \mathbf{x}_{N_i+1}]$ and $[\mathbf{u}_0, \mathbf{u}_1, ..., \mathbf{u}_{N_i}]$. These variable vectors enable the discretization of the problem into sub-problems which are then solved independently [17]. To ensure that the state trajectories are continuous between the sub-problems, a continuity condition constrains the states at the end of the i^{th} shooting node to be equal to the states at the beginning of the $(i+1)^{th}$ sub-problem (Fig. 1). Note that, in the DMS transcription, the states at the end of the sub-intervals $\mathbf{x}_{i,end}$ are estimated through numerical integration. Consequently, states at the end of the sub-intervals depend on the states and controls at the beginning of the sub-interval.

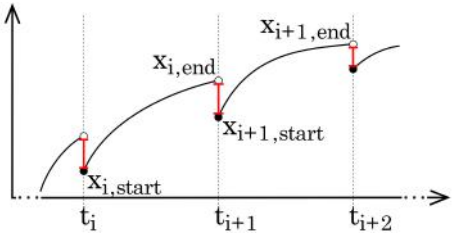

Figure 1: DMS implementation of the continuity constraint (red), where i is the number of the sub-problem ranging from 0 to N_i and $\mathbf{x}$ the state vector.

$$\mathbf{x}_{i+1,start} - \mathbf{x}_{i,end}(\mathbf{x}_{i,start},\ \mathbf{u}_i,\ T) = 0 \qquad (1)$$

Subdividing the problem improves numerical stability and computational efficiency [17]. Where direct single shooting transcription integrates the trajectory over its whole duration, DMS transcription only integrates over small intervals of the trajectory at a time. Given that the interval discretization is fine enough, this strategy prevents the integration from diverging largely when the controls are far from their optimal values.

As suggested in [18], the inter-shooting interval continuity condition could be softened by transforming the constraints into a penalty of the following form:

$$\sum_{i=1}^{N_i-1} \sum_{k=1}^{N_k} (x^k_{i+1,start} - x^k_{i,end}(\mathbf{x}_{i,start},\ \mathbf{u}_i,\ T))^2, \qquad (2)$$

with k the number of the state ranging from 0 to N_k. Formulating the continuity condition as

a penalty term in the cost function has the advantage of simplifying the resolution by removing constraints. However, for non-convex problems, it does not guarantee the continuity of the optimal solution, possibly resulting in dynamically inconsistent trajectories. This idea of simplifying the OCP by transforming a constraint into a penalty objective can also be applied to other types of constraints. It would be interesting to take advantage of the computational benefits from this underconstrained formulation while ensuring that the dynamics of the system and the task requirements are respected.

The objective of this study was to present a method generating initial guesses which are spread out over the variable domain but positioned close to a local minimum. Such a method would enlarge the chance of finding the global optimum while being independent of the researcher's insights on the problem. The method proposed is composed, first of an underconstrained optimization randomly initialized, followed by a fully constrained optimization initialized with the result from the first optimization. We hypothesized that this method would allow finding a variety of relevant local minima. The proposed method was compared with a usual multi-start approach regarding convergence rate, optimal cost, and computational time. A secondary objective was to assess the effect of the penalty weight and the maximum number of iterations during the first optimization on the optimal solutions.

2 METHODS

2.1 Workflow

The workflow is composed of the following stages:

Random initialization – A random initial guess is generated for the states and controls within their bounded domains. This variety of initial guesses independent of the researcher's knowledge of the problem increases the chance of finding various local minima.

First optimization – Some constraints of the original OCP are softened by replacing them with weighted penalties included in the cost function. This underconstrained OCP is easier to solve and less restrained by the IPM constraint barriers. The solution found might not be feasible. Still, due to the gradient descent, this solution is likely closer to an optimum and to respecting the constraints than the random initialization.

Second optimization – The solution from the **first optimization** step is provided as the initial guess for the fully constrained OCP.

This workflow is performed in a multi-start approach. As many random initial guesses as judged necessary are generated to create different optimal solutions.

2.2 Application to an OCP

The two-step optimization workflow was applied to a planar DMS trajectory optimization implemented in Bioptim [19]. The OCP consisted of inverting a rigid pendulum mounted on a cart without hitting obstacles along the trajectory (Fig. 2). The cart could translate on a rail, and the rigid pendulum could passively rotate without friction. The pendulum started hanging still below the cart and should end with a rotation of 180° above the cart. Horizontal forces were applied to the cart to invert the pendulum. The objective of the OCP was to minimize the horizontal forces and the duration of the movement. One point of the pendulum was constrained to avoid collision with four obstacles, namely spheres.

The four states ($\mathbf{x}$) of the problem were the cart translation and the pendulum rotation (positions and velocities) (Tab. 1). The control ($\mathbf{u}$) was the horizontal force applied to the cart. The problem was discretized into 500 shooting intervals ($N_i = 500$). Each shooting interval was integrated using five steps of a fourth-order Runge-Kuta method.

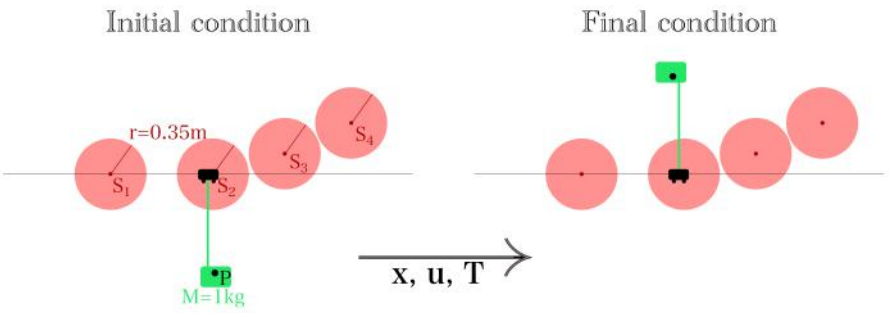

Figure 2: Illustration of the OCP. The pendulum (green) starts hanging below the cart (black). The cart moves on the rail (black line) until the pendulum is upside down. The point on the pendulum labeled P (black dot) cannot get through the spherical obstacles labeled S_i (red). The pendulum has a mass of 1 kg, and the radius of the spheres is 0.35 m.

Table 1: Bounds on the variables of the OCP

		First node	Middle nodes	Last node
x^0	Translation of the cart [m]	0	$[-2, 2]$	0
x^1	Rotation of the pendulum [rad]	0	$[-\pi, \pi]$	π
x^2	Velocity of the cart [m/s]	0	$[-10\pi, 10\pi]$	$[-10\pi, 10\pi]$
x^3	Velocity of the pendulum [rad/s]	0	$[-10\pi, 10\pi]$	$[-10\pi, 10\pi]$
u^0	Force on the cart [N]	$[-300, 300]$	$[-300, 300]$	*NA*
T	Final time of the simulation [s]	*NA*	*NA*	$[0, 10]$

NA: not applicable since the time parameter refers to the total duration of the movement, and there is no control applied to the last node of the simulation.

2.3 Implementations comparison

To assess the benefits of the two-step optimization workflow, the OCP was initialized 100 times and solved with three different implementations:

i DMS continuity and sphere avoidance conditions included in the problem as constraints (***Constrained***).

ii DMS continuity condition (***penalized continuity***) included in the problem as a penalty term during the **first optimization** step.

iii Sphere avoidance condition (***penalized obstacles***) included in the problem as a penalty term during the **first optimization** step.

For *ii* and *iii*, the weight of the penalty term and the maximum number of iterations in the **first optimization** step were varied in line with our secondary objective (Tab. 2).

Table 2: Weight of penalty terms (Eq. 3) in the cost function and maximum iterations for the **first optimization** step. Each combination of parameters was tested

	Maximum iterations	Continuity penalty weight (ω_3)	Obstacles penalty weight (ω_4)
Constrained	NA	constrained	constrained
Penalized continuity	$100, 1000, 10000$	$10000, 100000, 1000000$	constrained
Penalized obstacles	$100, 1000, 10000$	constrained	$10000, 100000, 1000000$

2.4 OCP transcription

The mathematical transcription of the OCP was:

$$\min_{\mathbf{x},\mathbf{u},T} \quad \omega_1 T + \omega_2 \sum_{i=0}^{N_i} \mathbf{u}_i^2 + \overbrace{\omega_3 \sum_{i=1}^{N_i-1} \sum_{k=1}^{N_k} (x_{i+1,start}^k - x_{i,end}^k(\mathbf{x}_{i,start},\ \mathbf{u}_i,\ T))^2}^{Continuity\ penalty} + \underbrace{\omega_4 \sum_{i=0}^{N_i+1} \sum_{j=1}^{4} \left(\begin{cases} \|\mathbf{P}_i(\mathbf{x}_{i,start}) - \mathbf{S}_j\| - r, & \text{if negative,} \\ 0, & \text{otherwise} \end{cases} \right)^2}_{Obstacles\ penalty} \tag{3a}$$

$$s.t. \quad x_{i+1,start}^k - x_{i,end}^k(\mathbf{x}_{i,start},\ \mathbf{u}_i,\ T) = 0 \tag{3b}$$

$$\mathbf{P}_i(\mathbf{x}_{i,start}) - \mathbf{S}_j \geq r \tag{3c}$$

$$\mathbf{x} \in \mathscr{X}, \mathbf{u} \in \mathscr{U}, T \in \mathscr{T} \tag{3d}$$

where r was the spheres' radius, $\mathbf{P}_i$ was the position of the point on the pendulum at the i^{th} node, $\mathbf{S}_j$ was the j^{th} sphere's center position ω_i were the weightings ($\omega_1 = 100$, $\omega_2 = 1$, $\omega_3 \in \{10000, 100000, 1000000\}$ and $\omega_4 \in \{1000, 100000, 10000000\}$), and $\mathscr{X}$, $\mathscr{U}$ $\mathscr{T}$ were the decision variable domains (see Tab. 1 and Tab. 2 for numerical values). The first two objective terms (i.e., minimizing the movement duration and the horizontal forces applied on the cart) were included in all implementations of the OCP. Their sum was used to compare the cost between implementations. The continuity term was included only to the ***penalized continuity*** OCP, while, for the other two OCPs, it was enforced by the continuity condition constraints (Eq. 3b). The sphere avoidance term was included only for the ***penalized obstacles*** OCP, while, for the other two OCPs, it was enforced by obstacle avoidance constraints (Eq. 3c). The tolerance on all constraints was set to 10^{-6} when solving the nonlinear program with IPOPT [20].

2.5 Optimal solutions analysis

Each optimized trajectory was reintegrated to measure the position of the point on the pendulum at 5000 points (500 shooting nodes x 10 sub-discretization) equally distributed in time. The distance between each point position and the spheres' surfaces was measured to obtain the spheres' penetration using the following equation:

$$\sum_{i=0}^{N_i \times 10+1} \sum_{j=1}^{4} \left| \begin{cases} \|\mathbf{P}_i(\mathbf{x}_{i,start}) - \mathbf{S}_j\| - r, & \text{if negative,} \\ 0, & \text{otherwise} \end{cases} \right|, \tag{4}$$

Solutions without DMS continuity condition violation and no penetration of the spheres were qualified as *admissible* solutions.

The convergence rate, optimal cost, computational time, and maximum sphere penetration for all solutions of the three implementations were reported. The most optimal solutions, referred to as *recommended* solutions, were further analyzed.

3 RESULTS

All implementations found optimal solutions (Fig. 3). The convergence rate of the ***constrained*** implementation was 93.0%. The convergence rate was 95.1% and 98.9% after the **second optimization** step of the ***penalized continuity*** and ***penalized obstacles*** implementations, respectively (Fig. 4 a.). The mean convergence time was 93.65 ± 81.39 *s* for the ***constrained*** implementation. The mean combined convergence time for both optimization steps were 148.94 ± 342.55 *s* and 30.79 ± 72.95 *s* for the ***penalized continuity*** and ***penalized obstacles*** implementations, respectively. Solutions with the same kinematics and cost were grouped into clusters to facilitate their

analysis. The optimal solutions belonging to the four most optimal clusters of solutions found were considered as *recommended* solutions (Fig. 5 and animation videos in supplementary material). These *recommended* solutions were at least as optimal as the best solution provided by the ***constrained*** implementation.

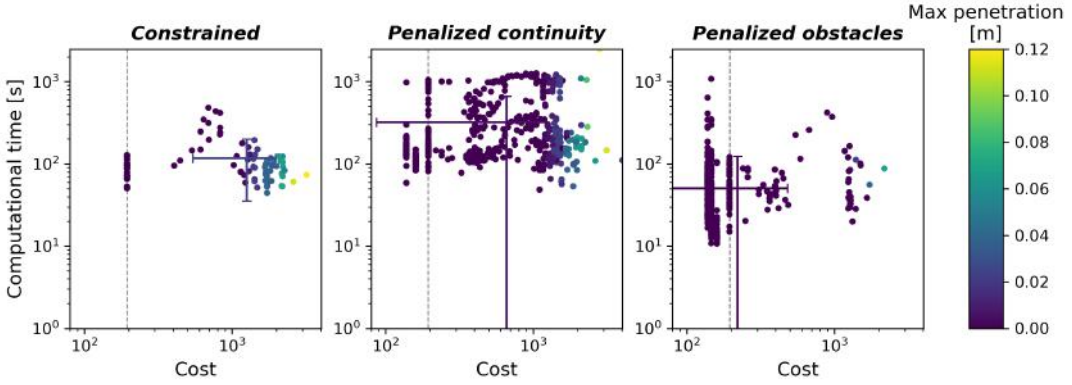

Figure 3: Cost (x-axis), computational time (y-axis) of the two-step optimization workflow, and maximum spheres' penetration (color gradient) for the ***constrained*** (left), ***penalized continuity*** (middle) and ***penalized obstacles*** (right) implementations. Means and standard deviations are presented with error bars. The dashed vertical line indicates the threshold for solutions considered as *recommended*. Only solutions that reached convergence are presented.

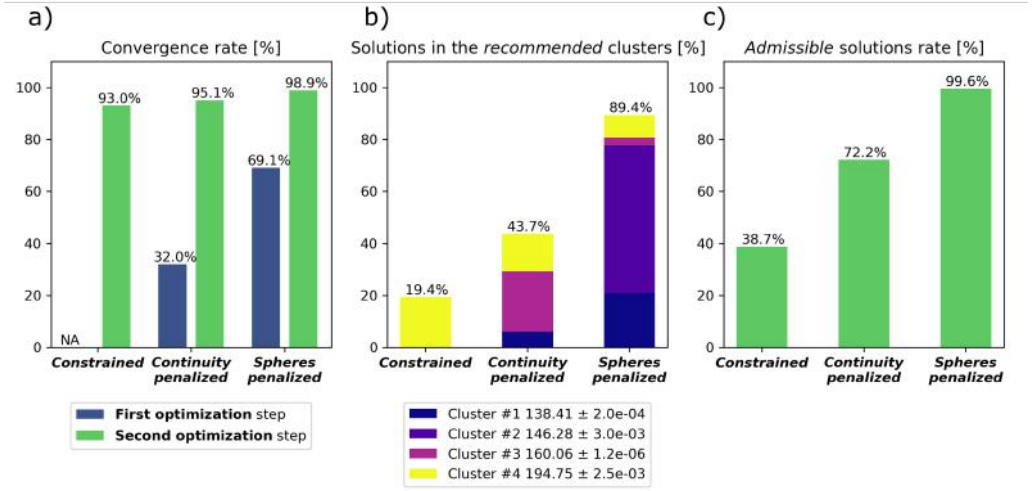

Figure 4: Proportion of a) OCPs that converged, b) solutions in the *recommended* clusters, and c) *admissible* solutions for the three OCP implementations.

The ***constrained*** implementation only found solutions in cluster #4, i.e., the worst cluster. The ***penalized continuity*** implementation found solutions in the clusters #1, #3, and #4 . The ***penalized obstacles*** implementation found solutions in all four clusters. Out of the solutions that converged, *recommended* solutions were found 19.4%, 43.7% and 89.4% of the time, for ***constrained***, ***penalized continuity*** and ***penalized obstacles*** implementations, respectively (Fig. 4 b.). Out of the solutions that converged, 38.7%, 72.2% and 99.6% were *admissible* solutions for ***constrained***, ***penalized continuity*** and ***penalized obstacles*** implementations, respectively (Fig. 4 c.).

The most optimal solutions were found with the shortest computational time with weight=1000, maximum iteration number=[1000, 10000] for ***penalized continuity*** and weight=100000, maximum iteration number=[1000, 10000] for ***penalized obstacles*** implementations (Fig. 8 and 9 in Appendix 6.3).

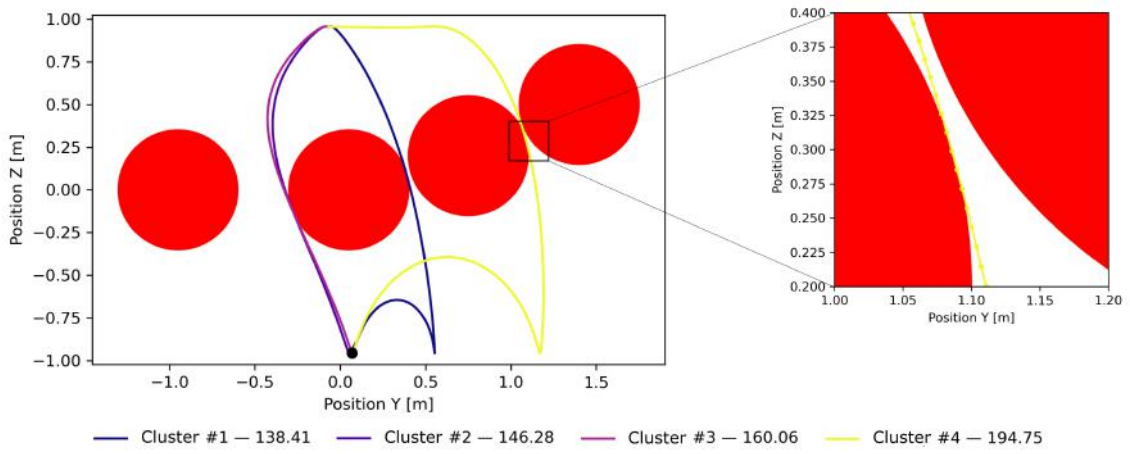

Figure 5: Trajectories of the pendulum during its inversion for the four clusters of *recommended* solutions. The cost associated with each cluster is presented in the legend. The red circles represent the obstacles, and the black dot is the initial position of the pendulum. A section of the trajectory is enlarged on the right-hand side to show how the pendulum got around the spherical obstacles. The shooting nodes where the constraints are applied are presented as dots along the trajectories.

4 DISCUSSION

A two-step optimization algorithm was presented and assessed by optimizing the trajectory of a rigid pendulum avoiding obstacles. The algorithm consists in solving an underconstrained version of the OCP, then warm-starting the fully constrained OCP with this solution as the initial guess. Our two-step optimization method led to more optimal solutions and a larger proportion of *recommended* solutions. Despite the two optimizations needed, our method did not require longer computational time; it was even faster for the ***penalized obstacles*** implementation than the usual random initialization.

4.1 Exploring relevant local minima

Trajectory optimization is often used to generate trajectories without *a priori* knowledge of which strategy should be used. In this context, the choice of the initial guess is important because it may impact the optimal strategies found. Indeed, when using gradient descent methods, if the initial guess is chosen too close to one feasible solution, the optimization will always converge to this solution ignoring more optimal strategies. On the other hand, if the initial guess is chosen too far from any feasible solution, it might impair convergence. The workflow presented here has shown to be an excellent compromise to generate a variety of initial guesses close enough to different local optima. The proposed workflow slightly increased the convergence rate, but more importantly, it has resulted in four times more *recommended* solutions than the usual multi-start method.

4.2 Simplifying constrained OCP

In most OCPs, continuity and path constraints are necessary to generate relevant trajectories that respect the dynamics and satisfy the task requirements. The optimization problem resolution is complexified when constraints are included. Computational tricks were suggested to work around this limitation. Some implementations involve expressing the constraints as penalty terms in the cost function allowing to solve an unconstrained OCP [21]. The penalty weightings can be modified iteratively to pressure the solutions to approach a feasible optimal solution [22, 23]. However, even with these improvements, implementations replacing constraints with penalties still do not guarantee feasibility for non-convex problems. Others suggested modifying the descent direction to exclude the constrained regions from the search path [24]. Modifying the descent method re-

moves the need for constraints while ensuring the optimal solution is in the constrained domain. Still, like IPM, it might disadvantage local minima which are hidden in non-linear constraint cavities. Recently, [25] modified the descent direction by considering the normalized gradient of the constraints; unfortunately, it does not apply to equality constraints. The warm-starting strategy proposed in this study was a good compromise as it simplifies the resolutions of the OCP, enables the exploration of hidden local minima, and provides solutions respecting the constraints after the **second optimization** step.

4.3 Non-penetration constraints

Constraints commonly used in robotics OCP aim to avoid collisions by refraining sections of the robot to access specified regions of the environment [26]. It is a usual practice to slightly overestimate the forbidden regions as a security precaution. It ensures that objects in the environment are entirely covered by the constraints. This overestimation is necessary in the case of DMS OCP since the constraints are usually applied at the shooting nodes only, meaning that the mathematical constraints of the problem might be transgressed between the nodes. With a fine temporal discretization and a small overestimation of the penetration constraints, it is possible to assume that the constrained regions are not accessible, thus preventing collisions. To measure the penetration allowed by our implementation of the problem, the maximum penetration distance of the pendulum was measured at 10 sub-interval for each DMS interval. We found that most trajectories (including all *recommended* solutions) did not penetrate the forbidden spherical obstacles. The solutions penetrating the spheres generally had a higher cost. Their strategy consisted in extending the duration to spread apart the shooting nodes and to use large forces (large cost) to numerically jump *over* the obstacles (Fig. 11 in Appendix 6.5). Due to the fine time discretization (500 nodes) and the bounds on the controls and velocities, this unadmissible strategy was impossible with a shorter duration. Since the ***penalized obstacles*** implementation had the greatest rate of *recommended* solutions found, it was not surprising that it also had the greatest rate of *admissible* solutions.

4.4 Effect of the parameters on the optimal solutions

Finding appropriate weightings for each objective term in an OCP's cost function is challenging [27]. In a concern for objectivity in the choice of the weightings of the penalty terms, we chose to vary them (Tab. 2) and analyzed their effect on the optimal solutions (Appendix 6.3). The solutions resulting from a larger penalty weight tended to have a higher cost. This could be explained by the other objective terms becoming too small compared to the penalty term. Consequently, the solutions of the **first optimization** step would be drawn farther from an optimum by the largely weighted penalty term. At the same time, the maximum iteration number was also varied. We expected that a few iterations in the right direction during the **first optimization** step would be sufficient to generate a good initial guess since the purpose of this step was only to improve the starting point without any feasibility guarantee. However, this hypothesis was shown wrong; it seems preferable to choose a higher maximal iteration number (> 1000) as it gave a better starting point. Thus, supplementary iterations during the **first optimization** step were beneficial to find a better starting point, which reduced the computational time of the **second optimization** step and led to solutions that are more optimal. Consequently, we recommend using a maximal iteration number greater than 1000 and a penalty weight between 1000 and 100000 depending on the order of magnitude of the penalty.

4.5 Limits and perspectives

Two main limitations are worth noticing. *i)* The workflow was applied to a simple OCP regarding the multibody system dynamics. While the generalization to other OCPs is not possible. Yet, there is no reason to think that this problem is a particular case. Further studies with more complex OCPs are needed to demonstrate the advantages of the proposed workflow over random initializa-

tion. *ii)* The process being time-consuming, a limited number of parameter combinations (penalty weight and maximum iteration number) were tested. Hence, it is not possible to provide the best parameters to optimize this OCP.

5 CONCLUSION

We developed a two-step optimization workflow in which the first optimization is solved by expressing constraints as penalties included in the cost function. The fully constrained OCP is then solved using the solution from the underconstrained OCP as the initial guess. Applying this two-step optimization algorithm to a trajectory optimization problem solved with an interior point method, we found more optimal solutions, more often and more rapidly. Consequently, researchers using constrained optimal control to search for innovative trajectories could reduce manual fine-tuning, spear computational time, and get more global solutions using or adapting the workflow proposed in this study.

ACKNOWLEDGMENTS

This work was supported by Mitacs and Own the Podium under Grant FR73046 and NSERC through the CREATE OPSIDIAN program.

REFERENCES

[1] Porsa, S., Lin, Y.C., Pandy, M.G.: Direct methods for predicting movement biomechanics based upon optimal control theory with implementation in opensim. Annals of biomedical engineering **44**(8) (2016) 2542–2557

[2] Ashby, B.M., Delp, S.L.: Optimal control simulations reveal mechanisms by which arm movement improves standing long jump performance. Journal of biomechanics **39**(9) (2006) 1726–1734

[3] Stryk, O.v., Schlemmer, M.: Optimal control of the industrial robot manutec r3. In: Computational optimal control. Springer (1994) 367–382

[4] Wang, R., Liu, C., Shi, Y.: Optimal control of aero-engine systems based on a switched lpv model. Asian Journal of Control **23**(5) (2021) 2239–2250

[5] Betts, J.T.: Practical methods for optimal control and estimation using nonlinear programming. SIAM (2010)

[6] Martí, R.: Multi-start methods. In: Handbook of metaheuristics. Springer (2003) 355–368

[7] Huchez, A., Haering, D., Holvoët, P., Barbier, F., Begon, M.: Local versus global optimal sports techniques in a group of athletes. Computer methods in biomechanics and biomedical engineering **18**(8) (2015) 829–838

[8] Forsgren, A., Gill, P.E., Wright, M.H.: Interior methods for nonlinear optimization. SIAM review **44**(4) (2002) 525–597

[9] Jackson, R.: An approach to the numerical solution of time-dependant optimisation problems in two-phase contacting devices. Chemical Engineering Research and Design **45a** (1967) 160–168

[10] Hodges, D.H., Bless, R.R.: Weak hamiltonian finite element method for optimal control problems. Journal of Guidance, Control, and Dynamics **14**(1) (1991) 148–156

[11] Bard, Y.: Nonlinear parameter estimation. Technical report (1974)

[12] Bock, H.G., Plitt, K.J.: A multiple shooting algorithm for direct solution of optimal control problems. IFAC Proceedings Volumes **17**(2) (1984) 1603–1608

[13] Biegler, L.T.: Solution of dynamic optimization problems by successive quadratic programming and orthogonal collocation. Computers & chemical engineering **8**(3-4) (1984) 243–247

[14] Bellman, R.: Dynamic programming, princeton, 1957. BellmanDynamic Programming1957 (1960)

[15] Leyendecker, S., Ober-Blöbaum, S., Marsden, J.E., Ortiz, M.: Discrete mechanics and optimal control for constrained systems. Optimal Control Applications and Methods **31**(6) (2010) 505–528

[16] Pan, B., Wang, Y., Tian, S.: A high-precision single shooting method for solving hypersensitive optimal control problems. Mathematical Problems in Engineering **2018** (2018)

[17] Liberzon, D.: Calculus of variations and optimal control theory: a concise introduction. Princeton university press (2011)

[18] Kelley, M.T., Baldick, R., Baldea, M.: A direct transcription-based multiple shooting formulation for dynamic optimization. Computers & Chemical Engineering **140** (2020) 106846

[19] Michaud, B., Bailly, F., Charbonneau, E., Ceglia, A., Sanchez, L., Begon, M.: Bioptim, a python framework for musculoskeletal optimal control in biomechanics. IEEE Transactions on Systems, Man, and Cybernetics: Systems **53**(1) (2022) 321–332

[20] Wächter, A., Biegler, L.T.: On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. Mathematical programming **106**(1) (2006) 25–57

[21] Gen, M., Cheng, R.: A survey of penalty techniques in genetic algorithms. In: Proceedings of IEEE international conference on evolutionary computation, IEEE (1996) 804–809

[22] Joines, J.A., Houck, C.R.: On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with ga's. In: Proceedings of the first IEEE conference on evolutionary computation. IEEE world congress on computational intelligence, IEEE (1994) 579–584

[23] Fabien, B.C.: Indirect solution of inequality constrained and singular optimal control problems via a simple continuation method. Journal of Dynamic Systems, Measurement, and Control **136**(2) (2014) 021003

[24] Dragan, A.D., Ratliff, N.D., Srinivasa, S.S.: Manipulation planning with goal sets using constrained trajectory optimization. In: 2011 IEEE International Conference on Robotics and Automation, IEEE (2011) 4582–4588

[25] Chen, L., Bletzinger, K.U., Gauger, N.R., Ye, Y.: A gradient descent akin method for constrained optimization: algorithms and applications. arXiv preprint arXiv:2302.11898 (2023)

[26] Raibail, M., Rahman, A.H.A., AL-Anizy, G.J., Nasrudin, M.F., Nadzir, M.S.M., Noraini, N.M.R., Yee, T.S.: Decentralized multi-robot collision avoidance: A systematic review from 2015 to 2021. Symmetry **14**(3) (2022) 610

[27] Mamdouh, M., Abido, M., Hamouz, Z.: Weighting factor selection techniques for predictive torque control of induction motor drives: A comparison study. Arabian Journal for Science and Engineering **43** (2018) 433–445

6 Appendix

6.1 Interior point constraints

The problem illustrated in Fig. 6 was solved with an IPM. It exemplifies its failure to find the global minimum because of the constraints' configuration. At the beginning (x_{init}), the constraints are not active since the yellow point is not neighboring a constraint. Then, the point would slide linearly in the gradient's direction (white line) until a bound is hit. The point would then slide on the constraint until the cost function is minimized along the constraint frontier. At this point, the IPM exits (x_{found}, magenta point). The solution found is better than the initial guess; however, if it had been possible to jump over the constraints, a better solution would have been found (x_{opt}, purple points).

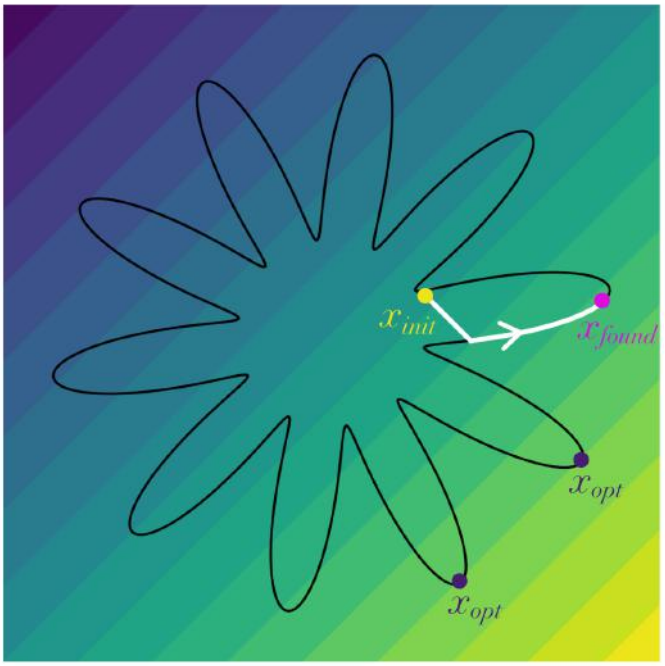

Figure 6: Illustration of the path followed by the IPM to find a locally optimal solution. The feasible variable domain is constrained by the black line, and the colored surface lines illustrate the gradient of the cost function. The optimization is initialized at the yellow point (x_{init}), then follows the white line to reach a locally optimal solution on the magenta point (x_{found}). More optimal solutions existed at purple points (x_{opt}).

6.2 Kinematics of the *recommended* solutions

Fig. 7 presents the kinematics used in the four clusters of *recommended* solutions.

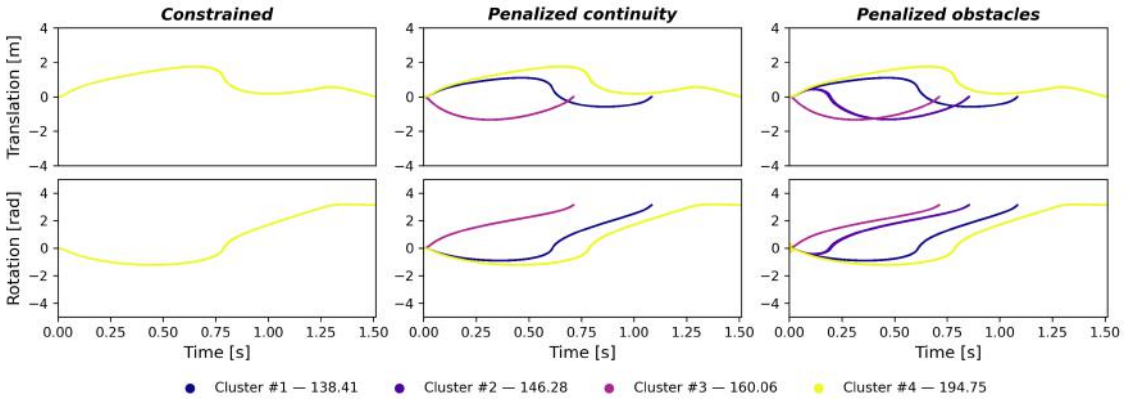

Figure 7: The pendulum's kinematics for the four clusters of *recommended* solutions. The cost associated with each cluster is presented in the legend.

6.3 Impact of the penalty weight and maximal number of iterations on the optimal solutions

During the **first optimization** step, the weight of the penalty term and the maximal number of iterations were varied. Fig. 8 and Fig. 9 present the effect of these parameters on the solutions that converged for the ***penalized continuity*** and ***penalized obstacles*** implementations, respectively.

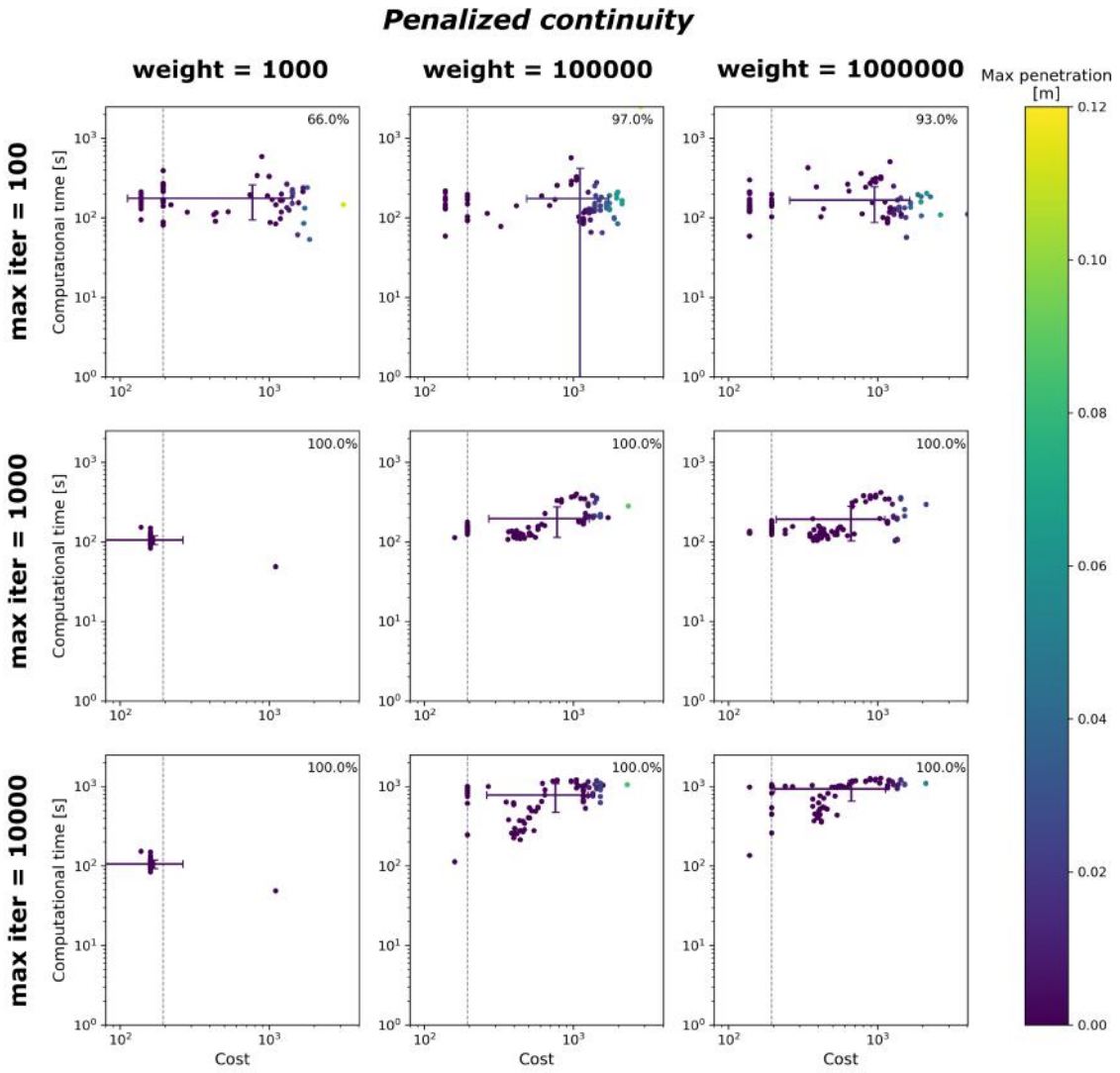

Figure 8: Cost (x-axis), computational time (y-axis) of the two-step optimization workflow, and maximum spheres' penetration (color gradient) for the ***penalized continuity*** implementation. Optimal solutions are separated according to the maximum number of iterations and the weight associated with the continuity penalty during the **first optimization** step. Mean and standard deviations are presented with error bars. The convergence rate of each case is presented in the top right corner of the figures. The dashed vertical lines indicate the threshold for solutions considered as *recommended*. Only solutions that reached convergence are presented.

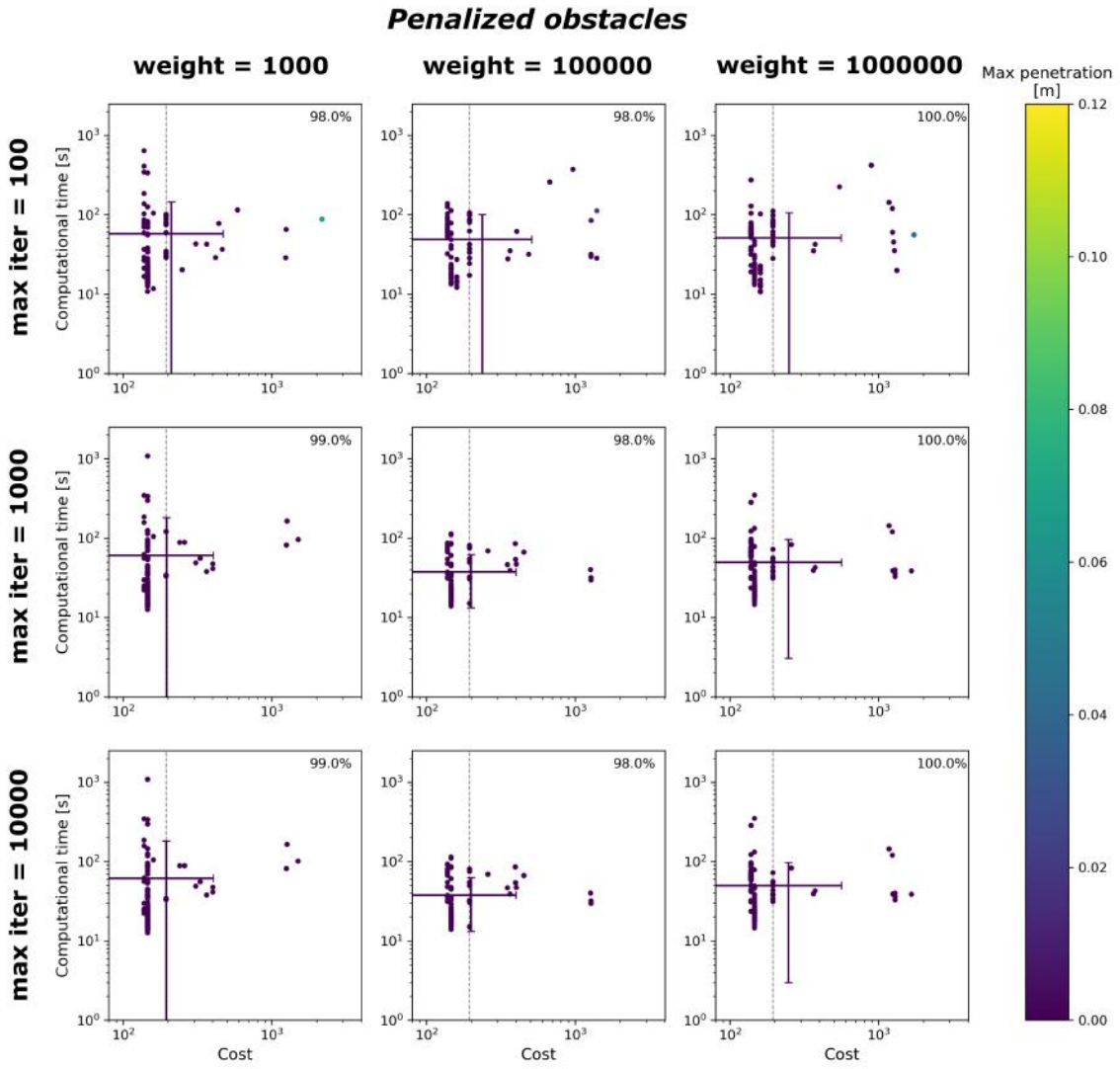

Figure 9: Cost (x-axis), computational time (y-axis) of the two-step optimization workflow, and maximum spheres' penetration (color gradient) for the ***penalized obstacles*** implementation. Optimal solutions are separated according to the maximum number of iterations and the weight associated with the continuity penalty during the **first optimization** step. Mean and standard deviations are presented with error bars. The convergence rate of each case is presented in the top right corner of the figures. Only solutions that reached convergence are presented.

6.4 Computational time per iteration

The median computational time was 93.65 s, 148.94 s, and 30.79 s for the ***constrained***, ***penalized continuity*** and ***penalized obstacles*** implementations, respectively . The median computational time per iteration is similar for each implementation (***constrained***=0.0949 s, ***penalized continuity***=0.0973 s, and ***penalized obstacles***=0.0961 s). However, the number of iterations to convergence varied across implementations (***constrained***=1216.9, **second optimization** step of ***penalized continuity***=1100.7 and **second optimization** step of ***penalized obstacles***=236.5). It can be concluded that the shorter computational time for the ***penalized obstacles*** implementation comes from the reduced number of iterations needed to reach convergence (Fig. 10 b.). Thus, on average, the **first optimization** step of the ***penalized obstacles*** implementation found initial guesses that were closer to an optimal solution than the ***penalized continuity***.

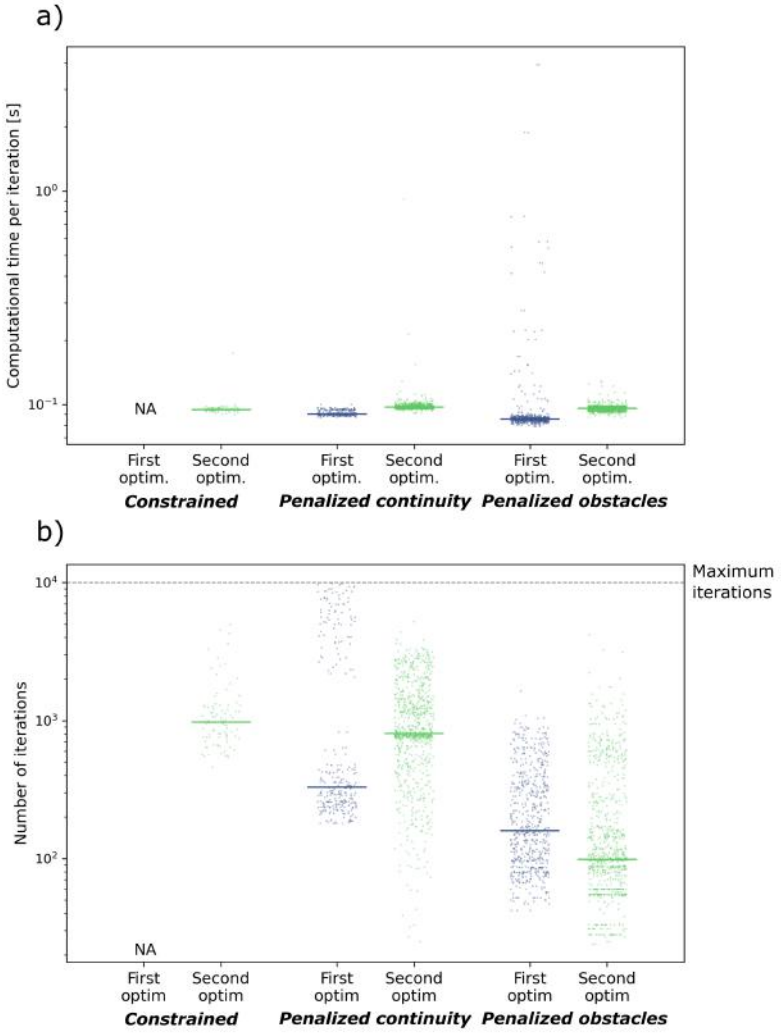

Figure 10: a. Computational time per iteration b. Number of iterations. Median values are presented with horizontal lines. Only the solutions that reached convergence after the **second optimization** step are presented.

6.5 Penetrating: a poorly efficient strategy

Most of the solutions showing sphere penetration had a high cost. Fig. 11 presents an example of a penetrating solution (animation video in supplementary material). This strategy had a high cost because it required a long movement duration and large forces applied to the cart. The time interval between two shooting nodes was 0.019 s, and the integral of the horizontal force applied on the cart was 17.39 Ns compared to 0.002 s and 4.04 Ns for the best *recommended* solution. The point on the pendulum penetrated the sphere S_3 and S_4 between shooting nodes 495, 496, and 497.

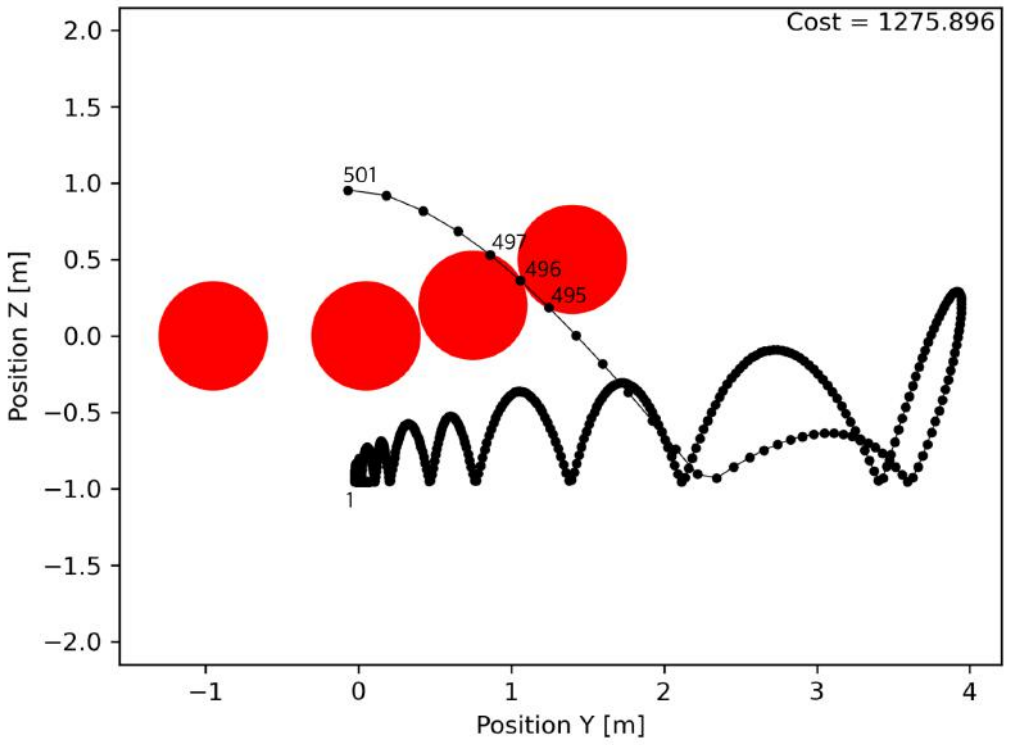

Figure 11: A less efficient solution where the pendulum penetrates through the spherical obstacles between the shooting nodes. Relevant shooting nodes are numbered for reference.