A $p_n h$-Adaptive Refinement Procedure for Numerical Optimal Control Problems

Lorenzo Bartali
Dipartimento di Ingegneria,
Civile e Industriale
Università di Pisa,
Pisa PI 56122, Italy
e-mail: lorenzo.bartali@phd.unipi.it

Marco Gabiccini[1]
Dipartimento di Ingegneria
Civile e Industriale
Università di Pisa,
Pisa PI 56122, Italy
e-mail: marco.gabiccini@unipi.it

Massimo Guiggiani
Dipartimento di Ingegneria,
Civile e Industriale,
Università di Pisa,
Pisa PI 56122, Italy
e-mail: massimo.guiggiani@unipi.it

This paper presents an automatic procedure to enhance the accuracy of the numerical solution of an optimal control problem (OCP) discretized via direct collocation at Gauss–Legendre points. First, a numerical solution is obtained by solving a nonlinear program (NLP). Then, the method evaluates its accuracy and adaptively changes both the degree of the approximating polynomial within each mesh interval and the number of mesh intervals until a prescribed accuracy is met. The number of mesh intervals is increased for all state vector components alike, in a classical fashion. Instead, improving on state-of-the-art procedures, the degrees of the polynomials approximating the different components of the state vector are allowed to assume, in each finite element, distinct values. This explains the $p_n h$ definition, where n *is the state dimension. With respect to the approaches found in the literature, where the degree is always raised to the highest order for all the state components, our methods allow a sensible reduction of the overall number of variables of the resulting NLP, with a corresponding reduction of the computational burden. Numerical tests on three OCP problems highlight that, under the same maximum allowable error, by independently selecting the degree of the polynomial for each state, our method effectively picks lower degrees for some of the states, thus reducing the overall number of variables in the NLP. Accordingly, various advantages are brought about, the most remarkable being: (i) an increased computational efficiency for the final enhanced mesh with solution accuracy still within the prescribed tolerance, (ii) a reduced risk of being trapped by local minima due to the reduced NLP size, and (iii) a gain of the robustness of the convergence process due to the better-behaved solution landscapes.* [DOI: 10.1115/1.4062227]

1 Introduction

Over the last three decades, direct collocation methods have become popular for the numerical solution of nonlinear optimal control problems, see, e.g., Refs. [1–4], to mention but a few. In direct collocation methods, state and control functions are discretized at a set of appropriately chosen points in the time horizon considered. Then, the continuous-time optimal control problem (OCP) is *transcribed* to a finite dimensional nonlinear program (NLP), and the NLP is coded and solved using different available software packages, such as Refs. [5–7], typically interfaced with interior-point [8], or SQP [9] back-end solvers.

A rich variety of approaches has already been proposed in the literature to obtain highly accurate solutions. A naïve approach would be to resort to the use of high-resolution (dense) discretization grids, with a requirement of a large amount of CPU and memory resources, especially if the resulting NLP is not sparse. Solutions to reduce the computational burden associated with uniform grid discretizations have been proposed in Ref. [10], where new grid points are introduced by solving an integer programming problem that minimizes the maximum discretization error by subdividing the current grid. In Ref. [11], the pseudospectral knotting method, initially proposed in Ref. [12], has been generalized by exchanging information across the patches in the form of event conditions associated with the optimal control problem. In Ref. [4], the user specifies the number of nodes to be increased within a particular phase, in case the error of the computed optimal control between two successive iterations is greater than a prescribed threshold. Here, the gradient of the control is employed to determine the location of the knots. In Refs. [13] and [14], a wavelet-Galerkin approach is used to discretize the OCP into an NLP, and local error analysis of the states and a wavelet analysis of the control profile decides whether to add or remove wavelet basis functions. When state and control path constraints are present, Ref. [15] uses wavelet analysis of the control profile to determine the regions that require refinement. With the specific objective to trade numerical accuracy for robustness and complexity for execution speed, multiresolution techniques have been proposed in Refs. [16] and [17] for the application that is time-critical, such as onboard real-time guidance and during emergency maneuvers.

Recently, a great deal of research effort has been devoted to the class of Gauss quadrature orthogonal collocation methods [1,2,18–20]. In these approaches, the state is typically approximated using a Lagrange polynomial where the support points are chosen as Gauss quadrature points. The most advanced employ either Gauss–Legendre, Gauss–Radau, or Gauss–Lobatto points [21]. Gauss quadrature collocation methods have been originally implemented as p methods with a single interval for the whole time horizon. Convergence of these methods was achieved by increasing the degree of the approximating polynomial. Their converge rate is exponential for problems with smooth and well-behaved solutions [22,23]. On the contrary, h methods, where convergence is sought by increasing the number of mesh intervals, work well for problems whose solutions are nonsmooth, since even a low-order polynomial approximation can capture *wild* solution behaviors on an extremely fine mesh. However, their converge rate is slow for problems with smooth solutions as compared to p methods.

With the goal of getting the best of both worlds, hp methods have been developed. Originally they were introduced for solving PDEs [24], while more recently they have been applied to OCPs as well [19,20]. However, these approaches have been demonstrated to create a great deal of noise in the error estimate, thereby making them computationally intractable when high-accuracy solutions are desired. Furthermore, the error estimate defined for these methods does not take advantage of the exponential convergence rate of a Gaussian quadrature collocation method.

Monitoring the error only in the state vector, Patterson et al. [25] proposed a new ph-mesh refinement method that is very

[1]Corresponding author.

Manuscript received August 4, 2022; final manuscript received March 20, 2023; published online May 4, 2023. Assoc. Editor: Radu Serban.

computationally efficient, does not suffer from noisy error estimates in Ref. [19], and produces larger mesh intervals for a given accuracy tolerance when compared with traditional fixed-order h methods.

Motivated by the relative simple ideas behind [25], in this paper we present an automatic procedure to enhance the accuracy of the numerical solution of an OCP discretized via direct collocation at Gauss-Legendre points. First, a numerical solution is obtained by solving the associated NLP with an initial uniform discretization. Then, the method evaluates its accuracy and adaptively changes both the degree of the approximating polynomial within each mesh interval and the number of mesh intervals until a prescribed accuracy is met, thus following the *ph* strategy proposed in Ref. [25]. The number of mesh intervals is increased concurrently for all state vector components, in a classical fashion.

In this paper, taking inspiration from the methodology developed in Ref. [25], we propose a *generalization* of the procedure described therein. The main contribution of our work consists in a new algorithm that allows the degree of the approximating polynomial to assume *distinct values for each state vector component and for each finite element*. To the best of the authors' knowledge, in the research works found in the literature, the degree is always raised to the highest order for all the state components, with an associated waste of computational resources and no practical accuracy benefit. Correspondingly, our procedure, despite being a bit more involved in program, allows for a sensible reduction of the computational burden in the to-be-solved NLP while ensuring the same solution accuracy.

The procedure presented unfolds in two main steps: (i) componentwise state error estimates are computed based on the difference between the Lagrange polynomial approximation and a higher-order estimate obtained by quadrature of the dynamics at Gauss–Legendre points within each finite element; then, (ii) the derived error estimates are employed to establish whether (a) the degree of the approximating polynomial should be increased or (b) the mesh interval should be split into subintervals. Option (a) is selected if the degree suggested by the method remains below a given threshold for all state components. Otherwise, according to option (b), a finer mesh is created in those intervals where the excessive error is registered. The NLP problem is solved again with the enhanced mesh and the evaluation procedure described is repeated. The process stops when, with the refined mesh and/or updated degrees, the NLP solution meets the prescribed error tolerance from the outset. More details are provided in Algorithms 1 and 2.

To validate the proposed procedure, we analyze its numerical results on three OCP problems. The first one is the classical stabilization of the origin of the Van der Pol oscillator in a prescribed time horizon. The second is a classical nonholonomic trajectory planning problem which consists in driving, in a prescribed time horizon, a rolling upright penny on a horizontal plane from an initial to a final position and orientation by applying to steer and roll torques. The last one represents the trajectory planning in 2D for a simplified vertical takeoff and landing (VTOL) aircraft endowed with two thrust engines.

The mesh refinement outcomes highlight that, given a maximum error threshold, by allowing to increase in the degree of the approximating polynomial independently for each state, our method effectively chooses to raise them frugally, thus reducing the overall number of NLP variables needed with clear benefits in terms of reduced CPU time, memory usage and reduced risk of being trapped in local minima.

2 Optimal Control Problem

In this work, the $p_n h$ adaptive refinement algorithm is applied to a general OCP. Therefore, with the aim to introduce the notation used in the rest of the paper, in this section the overall structure of an OCP and its main components are briefly recalled.

The general form of an OCP can be written as

$$\underset{X,U}{\text{minimize}} \quad J = \int_0^T g(X(t), U(t))\, dt + E(X(T)) \tag{1a}$$

$$\text{subject to} \quad \dot{X}(t) - f(X(t), U(t)) = 0, t \in [0, T] \tag{1b}$$

$$h(X(t), U(t)) \leq 0, \quad t \in [0, T] \tag{1c}$$

$$r(X(T), U(T)) \leq 0, \tag{1d}$$

where t is the independent variable (time), T is the terminal time, $X \in \mathbb{R}^{n_x}$ and $U \in \mathbb{R}^{n_u}$ are the states and the controls, respectively, and $f : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}$ is the dynamic vector field.

The OCP is characterized by an integral cost function, with a running cost $g : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}$, a terminal cost $E : \mathbb{R}^{n_x} \to \mathbb{R}$, and three kinds of constraints: dynamics $\dot{X} - f(X, U)$, inequality $h : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_h}$, and terminal constraints $r : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \in \mathbb{R}^{n_r}$. The solution of the OCP consists of the controls $U^*(t)$ and the states $X^*(t)$ minimizing the integral cost function while satisfying the above constraints.

3 Gauss–Legendre Collocation Method With Tailored State Degrees

The continuous optimal control problem introduced in Sec. 2 is discretized using the *direct collocation* method at Gauss–Legendre collocation points. As a consequence, the original OCP becomes a NLP. For discussions about existence and uniqueness of the NLP solution, we refer the interested reader to Refs. [26] and [21].

In *direct collocation* method, the time horizon is discretized on a fixed mesh, where the generic interval is denoted as $S_k = [T_{k-1}, T_k]$, with mesh step $h_k = T_{k-1} - T_k$ $(k = 1, \ldots, N)$. The values at mesh points (nodal values) are indicated by $X_k \in \mathbb{R}^{n_x}$, $U_k \in \mathbb{R}^{n_u}$, and $T_k \in \mathbb{R}$, for states, controls, and time, respectively. The parameterization of the controls on the same mesh is chosen as piecewise constant, which yields a constant control $U(t) = U_k$ on each interval S_k.

As far as states in S_k are concerned, classical approaches appearing in the literature approximate all of them by polynomials of the same degree d. Instead, in this paper, we can pick different degrees for each state and for each mesh interval. Hence, the ith state component on the S_k subinterval, is approximated by a polynomial of degree $d_{k,i}$ defined as follows:

$$X_{k,i}^{d_{k,i}}(\tau) = \sum_{m=1}^{d_{k,i}^+} L_m^{d_{k,i}}(\tau) x_{k,i,m}, \quad \text{with} \tag{2a}$$

$$L_m^{d_{k,i}}(\tau) = \prod_{r=1, r\neq m}^{d_{k,i}^+} \frac{\tau - \bar{\tau}_r^{d_{k,i}}}{\bar{\tau}_m^{d_{k,i}} - \bar{\tau}_r^{d_{k,i}}}, \tag{2b}$$

where $d_{k,i}^+ = d_{k,i} + 1$, $\tau \in [0, 1]$ is the normalized time on S_k such that $T_k(\tau) = T_{k-1} + h_k \tau$. Basis functions $L_m^{d_{k,i}}(\tau) : \mathbb{R} \to \mathbb{R}$ are the Lagrange polynomials of degree $d_{k,i}$, calculated considering the points $\bar{\tau}^{d_{k,i}} = [\tau_0, \tau^{d_{k,i}}]$, where $\tau_0 = 0$ and $\tau^{d_{k,i}}$ are the $d_{k,i}$ Gauss–Legendre collocation points. In our notation, $\tau^{d_{k,i}}$ represents the vector of collocation points and $\bar{\tau}^{d_{k,i}}$ represents its *augmented* version obtained by appending τ_0 in the first position. Then, $\bar{\tau}_j^{d_{k,i}}$ indicates the jth component of $\bar{\tau}^{d_{k,i}}$ vector. Finally, $x_{k,i,m} \in \mathbb{R}$ is the values of the ith state polynomial in interval S_k at $\bar{\tau}_m^{d_{k,i}}$.

In order to apply collocation equations, ensuring that the system dynamics are satisfied at discrete points, the derivative of the generic polynomial $X_{k,i}^{d_{k,i}}(\tau) \in \mathbb{R}$ comes into play. This is defined as

$$V_{k,i}(\tau) = \frac{d}{d\tau}\left(X_{k,i}^{d_{k,i}}(\tau)\right) = \sum_{m=1}^{d_{k,i}^+} \left(\frac{d}{d\tau} L_m^{d_{k,i}}(\tau)\right) x_{k,i,m} \tag{3}$$

Moreover, it is useful to define quantity $X_k^d(\tau) \in \mathbb{R}^{n_x}$ as

$$X_k^d(\tau) = [X_{k,1}^{d_{k,1}}(\tau), \cdots X_{k,i}^{d_{k,i}}(\tau), \cdots X_{k,n_x}^{d_{k,n_x}}(\tau)]^{\mathrm{T}} \tag{4}$$

which represents, in the generic interval S_k and for a given τ, the evaluation of all state polynomials each of its own degree $(d_{k,1}, \ldots, d_{k,i}, \ldots, d_{k,n_x})$.

In *direct collocation* Eqn. (1b), for the ith state on the kth interval, reads as

$$V_{k,i}(\tau_j^{d_{k,i}}) = f_i(X_k^d(\tau_j^{d_{k,i}}), U_k)\, h_k, \quad j = 1, \ldots, d_{k,i} \tag{5}$$

where $\tau_j^{d_{k,i}}$ is the jth collocation point for the ith state and $f_i : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}$ is the ith component of the vector field representing system dynamics. Equation (5) has to be imposed for all state components $(i = 1, \ldots, n_x)$ and for all mesh interval $(k = 1, \ldots, N)$.

To completely define the state polynomial, it is necessary to add the continuity equations as equality constraint of the resulting NLP as follows:

$$X_k^d(1) - X_k = 0, \quad k = 1, \ldots, N-1 \tag{6}$$

where the nodal values of the state are linked to the $x_{k,i,m}$ by the following equation:

$$X_k = [x_{k+1,1,1}, \ldots, x_{k+1,i,1}, \ldots, x_{k+1,n_x,1}], \quad k = 1, \ldots, N-1 \tag{7}$$

For $k=1$ and $k=N$, the following boundaries conditions are imposed:

$$X_0 = \bar{X}_0 \tag{8}$$

$$X_N^d(1) = X_N \tag{9}$$

where $X_0 = X(0) = [x_{1,1,1}, \ldots, x_{1,i,1}, \ldots, x_{1,n_x,1}]$, $\bar{X}_0$ is an assigned initial value and $X_N = X(T)$ is the final nodal value.

The generic path constraint of Eq. (1*c*), for each S_k, becomes

$$h(X_k, U_k) \le 0. \quad k = 1, \ldots, N \tag{10}$$

Then, the terminal constraints of Eq. (1*d*) are

$$r(X_N, U_N) \le 0 \tag{11}$$

As regards the cost function, the integral of Eq. (1*a*) is numerically approximated with a quadrature scheme that employs, for each interval, the collocation points associated with the state components with the highest degree. In such manner, the best integral approximation is guaranteed. Hence, the running cost $g(t)$ in the S_k interval can be rewritten as

$$g_k(\tau) = \sum_{m=1}^{\bar{d}_k} \ell_m^{\bar{d}_k^-}(\tau) g_{k,m}, \quad \text{with} \tag{12a}$$

$$\ell_m^{\bar{d}_k^-}(\tau) = \prod_{r=1, r\neq m}^{\bar{d}_k} \frac{\tau - \tau_r^{\bar{d}_k}}{\tau_m^{\bar{d}_k} - \tau_r^{\bar{d}_k}}, \tag{12b}$$

where $\bar{d}_k = \max_{i=1,\ldots,n_x}(d_{k,i})$, $\bar{d}_k^- = \bar{d}_k - 1$, $\ell_m^{\bar{d}_k^-}$ are the Lagrange polynomials of degree $\bar{d}_k^-$, $\tau^{\bar{d}_k}$ are the $\bar{d}_k$ Gauss–Legendre collocation points, and $g_{k,m}$ is the value of $g_k(\tau)$ at $\tau_m^{\bar{d}_k}$.

From this polynomial approximation, the numerical integral J_k of the running cost function $g(t)$ in the kth element can be obtained from the following equation:

$$J_k = \sum_{m=1}^{\bar{d}_k} \left(\int_0^1 \ell_m^{\bar{d}_k^-}(\tau) d\tau \right) g_{k,m} h_k \tag{13}$$

With the equations shown in this section, it is possible to transform the original OCP into an NLP by writing them for $k = 1, \ldots, N$, and considering that the total cost function J, which has to be minimized, is given by $J = \sum_{k=1}^{N} J_k$.

It is worth noting that, unlike [25] where the dynamics are collocated using the *implicit integral form*, here the *differential form* is used. Furthermore, even if the collocation method was generalized by using a different degree for the polynomial approximation of ith state on the kth mesh interval, by choosing a degree d in a way that $d_{k,i} = d$, for $i = 1, \ldots, n_x$ and $k = 1, \ldots, N$, the classical collocation method can be implemented.

4 $p_n h$-Adaptive Mesh Refinement Method

In this section, a $p_n h$-adaptive mesh refinement method is developed. In particular, the generic OCP problem is first transcribed into an NLP using the *direct collocation* method, explained above, then the mesh refinement method is applied to the obtained optimal solution. The $p_n h$ method explained here is closely related to Ref. [25]. In particular, the method for estimating the error in the current solution is adapted from the aforementioned reference. The key novelty here is the refinement strategy. In Ref. [25], the adjustment of the polynomial degree is performed such that all states are represented by the same degree. Instead, in this work, the p-refinement is different for each state. Then, similarly to aforementioned work, the mesh spacing is adjusted only if the approach to reach convergence by increasing the polynomial degree fails.

4.1 Error Estimate. In this section, an estimate of the relative error in the solution within a mesh interval is derived. The error is calculated in each mesh interval and for each component of the state. The key idea is to compare the optimal state obtained from the NLP solution, with a more accurate approximation of the state.

To explain in detail this aspect, let us consider the ith state on the generic S_k mesh interval. The solution of NLP gives as an output a polynomial approximation of the state, namely, $X_{k,i}^{d_{k,i}}(\tau)$. Under the assumption of a smooth solution, a polynomial approximation obtained with an increased number of Gauss-Legendre points should yield a state that more accurately complies with the dynamics. Suppose that the error has to be estimated at a set of $d_{k,i}^+ = d_{k,i} + 1$ collocation points denoted as $\tau^{d_{k,i}^+}$. Then, on the one hand, the values of the NLP solution at these points are computed as $X_{k,i}^{d_{k,i}}(\tau_j^{d_{k,i}^+})$, $(j = 1, \ldots, d_{k,i}^+)$.

On the other hand, a higher-order approximation of the state can be constructed by sampling the dynamics at these new Gauss–Legendre points $\tau^{d_{k,i}^+}$ and employing a Gauss quadrature formula. Let $\bar{X}_{k,i}^{d_{k,i}^+}(\tau)$ be a polynomial of degree $d_{k,i}^+$ that is defined for the ith state on the S_k interval. If the derivative of this polynomial matches the dynamics at each $\tau^{d_{k,i}^+}$ collocation points, $\bar{X}_{k,i}^{d_{k,i}^+}(\tau)$ can be obtained by a numerical integration using the resulting quadrature scheme

$$\bar{X}_{k,i}^{d_{k,i}^+}(\tau) = \sum_{m=1}^{d_{k,i}^+} \left(\int_0^\tau \ell_m^{d_{k,i}}(\tau) d\tau \right) f_i\left(X_k^d\left(\tau_m^{d_{k,i}^+}\right), U_k \right), \quad \text{with} \tag{14a}$$

$$\ell_m^{d_{k,i}}(\tau) = \prod_{r=1, r\neq m}^{d_{k,i}^+} \frac{\tau - \tau_r^{d_{k,i}^+}}{\tau_m^{d_{k,i}^+} - \tau_r^{d_{k,i}^+}} \tag{14b}$$

where $\int_0^\tau \ell_m^{d_{k,i}}(\tau) d\tau$ are the integrals of the Lagrange polynomials and represent the weights of a classical quadrature formula.

Hence $\bar{X}_{k,i}^{d_{k,i}^+}(\tau)$ represents the improved approximation and its values in $\tau^{d_{k,i}^+}$ will be denoted by $\bar{X}_{k,i}^{d_{k,i}^+}(\tau_j^{d_{k,i}^+})$, $(j = 1, \ldots, d_{k,i}^+)$.

With the above quantities, the *absolute* error $E_{k,i,j}$ and the *relative* error $e_{k,i,j}$ computed at each $\tau_j^{d_{k,i}^+}$ for the ith state in the kth interval are defined as

$$E_{k,i,j} = \left| \bar{X}_{k,i}^{d_{k,i}^+}(\tau_j^{d_{k,i}^+}) - X_{k,i}^{d_{k,i}}(\tau_j^{d_{k,i}^+}) \right| \tag{15a}$$

$$e_{k,i,j} = \frac{E_{k,i,j}}{1 + \max_{j=1,\ldots,d_{k,i}^+} |X_{k,i}^{d_{k,i}}\left(\tau_j^{d_{k,i}^+}\right)|} \tag{15b}$$

The relative error in particular is the key ingredient used in the next Sec. 4.2 to implement the p_nh-refinement strategy.

4.2 Algorithm. In this section, the algorithm implemented for mesh refining is illustrated. Let us suppose that our goal is to keep the relative error $e_{k,i,j}$ below a given tolerance ϵ in each mesh interval S_k, $(k = 1, \ldots, N)$. If the tolerance ϵ is not met at least in one mesh interval, then the next step is to refine the current mesh, either by dividing the mesh interval or increasing the degree of the approximating polynomial within the mesh interval.

For the sake of clarity, the method is shown as a pseudocode for the generic S_k mesh interval. The Algorithm unfolds in two main passes: (i) the *Exploration* pass and (ii) the *Execution* pass. To better understand the pseudocode, some quantities appearing in it are worth explaining from the outset.

With reference to Algorithm 1, $d_{k,i}^*$ is the *new* polynomial degree that is suggested by the algorithm, for the ith state in the S_k interval, to reach the prescribed tolerance ϵ. The quantity $B_{k,i}$ represents the number of subinterval into which S_k should be split in order to meet the prescribed ϵ tolerance for the error of the ith state component. If $B_{k,i} = 1$, no subdivisions in S_k are necessary. Correspondingly, if $d_{k,i}^* = d_{k,i}$, no increment in the polynomial degree is required. Instead, $d_{\min}$ and $d_{\max}$ are the maximum and the minimum allowable polynomial degrees provided by the user.

Algorithm 1 *Exploration* pass of the p_nh mesh refinement

1: **for** $i = 1$ to n_x **do**
2: **if** $\max_{j=1,\ldots,d_{k,i}^+}(e_{k,i,j}) \leq \epsilon$ **then**
3: $d_{k,i}^* \leftarrow d_{k,i}$ ▷ degree ok
4: $B_{k,i} \leftarrow 1$ ▷ mesh ok
5: **else**
6: $P_{k,i} \leftarrow \log_{d_{k,i}}, \left[\max_{j=1,\ldots,d_{k,i}^+} \left(\frac{e_{k,i,j}}{\epsilon} \right) \right]$
7: $d_{k,i}^* \leftarrow d_{k,i} + P_{k,i}$ ▷ increase degree
8: **if** $d_{k,i}^* \leq d_{\max}$ **then**
9: $B_{k,i} \leftarrow 1$
10: **else**
11: $B_{k,i} \leftarrow \max\left(\left[\frac{d_{k,i}^*}{d_{\min}} \right], 2 \right)$ ▷ split mesh
12: **end if**
13: **end if**
14: **end for**

According to Algorithm 1, the outcome of the exploration pass is to collect three vectorial quantities: $D_k^* = [d_{k,1}^*, \ldots, d_{k,i}^*, \ldots, d_{k,n_x}^*]$, $D_k = [d_{k,1}, \ldots, d_{k,i}, \ldots, d_{k,n_x}]$, and $[B_{k,1}, \ldots, B_{k,i}, \ldots, B_{k,n_x}]$. The quantity $P_{k,i}$, playing a key role to compute $d_{k,i}^*$, is calculated as suggested by Ref. [25], in accordance with the convergence theory summarized in Refs. [27] and [28].

Roughly speaking, the outcome of the *exploration* pass does not provide a final answer *per se* but is only instrumental to the process of deciding whether the degree of the polynomial for each state component should be raised and/or the interval S_k should be split. The actual action to take is performed in the *execution* pass and benefits from the comparison of the collected vectorial quantities.

The execution pass, presented in the form pseudocode in Algorithm 2 is now illustrated.

Algorithm 2 Execution pass of the p_nh mesh refinement

1: **if** $\max_{i=1,\ldots,n_x}(B_{k,i}) = 1$ **then**
2: **if** $D_k^* = D_k$ **then**
3: algorithm.stop() ▷ convergence reached
4: **else**
5: **for** $i = 1$ to n_x **do**
6: $d_{k,i} \leftarrow d_{k,i}^*$ ▷ p_n-refinement
7: **end for**
8: **end if**
9: **else**
10: **for** $i = 1$ to n_x **do**
11: $d_{k,i} \leftarrow d_{k,i}$
12: $B_{k,i} \leftarrow \max_{i=1,\ldots,n_x}(B_{k,i})$ ▷ h-refinement
13: **end for**
14: **end if**

The main idea is that, if all $B_{k,i}$'s are equal to 1 and $D_k^* = D_k$, then the current mesh and polynomial degrees are sufficient to meet the prescribed tolerance. Therefore, no refinement is needed (convergence reached).

Instead, if only all $B_{k,i}$'s are equal to 1, then the collected $d_{k,i}^*$ should replace the degrees of the polynomials, according to a p_n-refinement strategy (p_n-refinement).

Alternatively, if at least one of the $B_{k,i}$'s is greater than 1—for at least one state component a degree greater than $d_{\max}$ was required (cf. line 11 in Algorithm 1)—a subdivision of the "responsible" interval/intervals takes place while the degrees are kept to the starting values $d_{k,i}$, thus following a h-refinement strategy (h-refinement).

In particular, both in the ph and the p_nh method used in this paper, we adopt a slightly different strategy with respect to Ref. [25]. In fact, in Ref. [25] the degrees of each state are *restarted* to $d_{\min}$, if at least one of the $B_{k,i}$'s is greater than 1. In our work, instead, the degrees are kept to the previous values $d_{k,i}$ for the sake of reducing the number of algorithmic iterations. This logic, despite being in general more eager in terms of variables with respect to the restarting process adopted in Ref. [25], still indicates a clear advantage of the p_nh method over to the ph one.

In any event, at the end of the execution pass, the new mesh (to be employed in the discretization of the next NLP to be solved), is encoded in the quantity $d_{k,i}$ and $B_{k,i}$ for $i = 1, \ldots, n_x$ and for $k = 1, \ldots, N$.

After the solution of the new NLP with the refined mesh, the same strategy can be applied until convergence. Converge is obtained when hitting line 3 of Algorithm 2.

It is worth observing that the proposed algorithm can match the original mesh refinement method in Ref. [25] if the following assumptions are made: (i) the initial NLP is setup such that all the state components have the same degree d_k in S_k, therefore, $d_{k,i} = d_k$ $(i = 1, \ldots, n_x)$; (ii) line 6 of Algorithm 2 is modified according to $d_{k,i} = \max_{i=1,\ldots,n_x}(d_{k,i}^*)$. In this manner, the polynomial approximation maintains always the same degree for all the states.

5 Examples

In this section, three different examples are discussed. Their purpose is to numerically showcase that our method allows us to adaptively refine the mesh to accurately solve an OCP, similarly to Ref. [25], with the remarkable advantage of an increased computational efficiency for the final enhanced mesh without hindering the accuracy of the final solution. In fact, by allowing to increase in the degrees of the approximating polynomials independently for each state, our method effectively adopts to raise them carefully and in a tailored manner. This allows us to reduce the overall number of

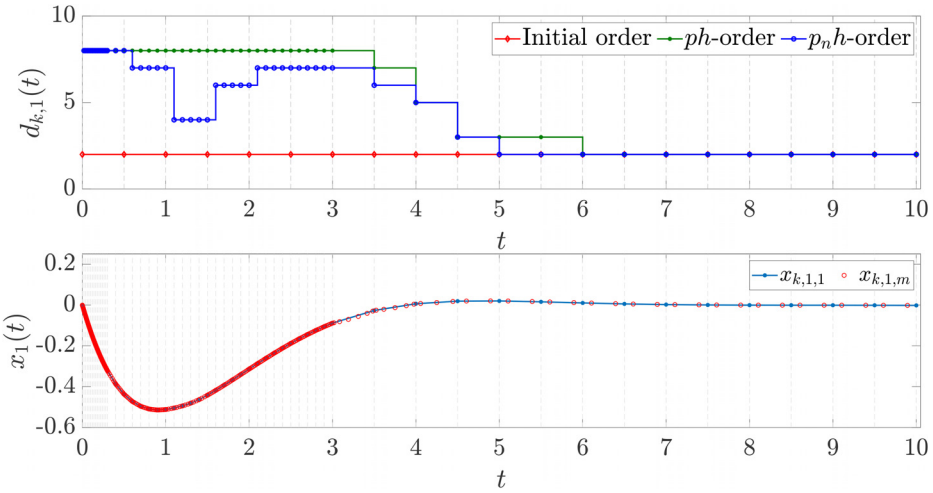

Fig. 1 Example 1: (top panel) polynomial degree for state $x_1(t)$ for the initial mesh (red diamonds markers), outcome degrees of the p_nh algorithm (blue empty circles markers), outcome degrees of the *ph* algorithm (green filled circles markers). Both algorithms prescribe to split the mesh intervals in an equal manner, i.e., for $0 \le t \le 3$ s. The *ph* algorithm (green filled circles markers) always raises the degree for all the states components if, in the generic interval, at least one states requires it. Consistently, the green curves in top panels of Figs. 1 and 2 are equal. Instead, the p_nh algorithm (blue empty circles markers) in the generic interval raises the degree only for that particular state that requires it. Comparing the blue curves in bottom panels of Figs. 1 and 2 it is evident, in each time interval, which is the state that requires the increase in the degree; (bottom panel) optimal $x_1(t)$ profile obtained with p_nh algorithm. Being equal to the mesh density (same dimension of the time intervals) more collocation points $x_{k,1,m}$ are added where the polynomial order required is higher.

variables of the resulting NLP, with clear benefits in terms of reduced CPU time, memory usage, and reduced risk of being trapped by local minima.

Let us now direct our attention to the details. Each problem is transcribed in its initial NLP version by choosing $N = 20$ mesh intervals and by assigning a degree $d_{k,i} = 2$ everywhere, i.e., for all the state components $i = 1, \ldots, n_x$ and in all mesh intervals $k = 1, \ldots, N$. The mesh refinement algorithm is then applied with $d_{\min} = 2$, $d_{\max} = 8$ with a prescribed maximum allowable relative error $\epsilon = 1 \times 10^{-3}$.

It is worth observing that the p_nh and *ph* tags indicate our mesh refinement method and the one that would be suggested by applying the algorithm proposed by Patterson et al. [25], respectively. It is worth noting that the original Patterson's algorithms are obtained here as a particular case of ours (see the explanation in Sec. 4).

It is also worth remarking that the final refined mesh for all three presented examples (both in terms of number and size of S_k intervals), turns out to be the same by rolling out both the p_nh and the *ph* approaches. This arises partly as a result that both algorithms (ours and the original proposed in Ref. [25]) share the last section (lines 9–14) of the execution pass in Algorithm 2. This is in accordance with the rule that for the generic S_k interval to be split it is sufficient that just one state component in the generic S_k tries to exceed the maximum degree $d_{\max}$.

Each transcribed optimal control problem tested was coded in a scripting environment using the MATLAB interface to the open-source CASADI framework [5]. The CASADI suite can perform automatic differentiation on the code to compute gradients and Jacobians and provides building blocks to efficiently formulate and solve large-scale optimization problems. The back-end solver adopted is IPOPT [8] which implements an interior-point algorithm. All the associated NLPs are solved on a laptop with a 2.30 GHz Intel Core i7-10875H CPU and 32 GB of RAM.

5.1 Example 1: Van Der Pol. The first optimal control problem considered is taken from Ref. [5] and considers driving a *Van der Pol* oscillator to the origin. It can be expressed as follows:

$$\underset{X \in \mathbb{R}^2, U \in \mathbb{R}}{\text{minimize}} \quad J = \int_0^T \left(x_1^2 + x_2^2 + u^2\right) dt \tag{16a}$$

$$\text{subject to} \quad \dot{x}_1 = \left(1 - x_2^2\right)x_1 - x_2 + u, t \in [0, T] \tag{16b}$$

$$\dot{x}_2 = x_1, \quad t \in [0, T] \tag{16c}$$

$$-1 \le u \le 1, \quad t \in [0, T] \tag{16d}$$

$$x_2 \ge -0.25, \quad t \in [0, T] \tag{16e}$$

$$x_1(0) = 0, \quad x_2(0) = 1 \tag{16f}$$

where $T = 10$ s, x_1 and x_2 are the two-state components (hence $X(t) = [x_1(t), x_2(t)] \in \mathbb{R}^2$), $u \in \mathbb{R}$ indicates the control, and the dynamics constraints and the initial conditions are shown explicitly for each state component. Clearly, $\dot{X}(t) = [\dot{x}_1(t), \dot{x}_2(t)]$.

In Figs. 1 and 2 (top panels) the final degrees of the first (x_1) and the second (x_2) state components, respectively, are shown. Here, the blue line is the outcome of the p_nh-mesh refinement algorithm whereas the green line is the result of the *ph*-one. With respect to the red line, which represents the initial degree, each method increases the degrees of the polynomials approximating the states. It is worth noting that the green line is the same for both figures, meaning that, according to the *ph* logic, all the states share the same (maximum) polynomial order in the same interval S_k. Instead, the blue lines are different, meaning that each state has reached an independent degree for its polynomial approximation. Furthermore, with respect to the initial S_k steps delimited by the dashed gray lines, the more packed green and blue markers of the final solutions reveal where the *h*-strategy was adopted, thus producing a finer mesh.

In Figs. 1 and 2 (bottom panels) the state profiles obtained with the p_nh final mesh are shown. Here, the dashed gray lines delimit the *final* mesh intervals. In particular, the nodal values are shown in blue, whereas the collocation points are plotted in red (empty circles markers). These figures clearly show that the p_nh-adaptive mesh refinement method increases the polynomial order (more collocation points) and increases the mesh density (more mesh points)

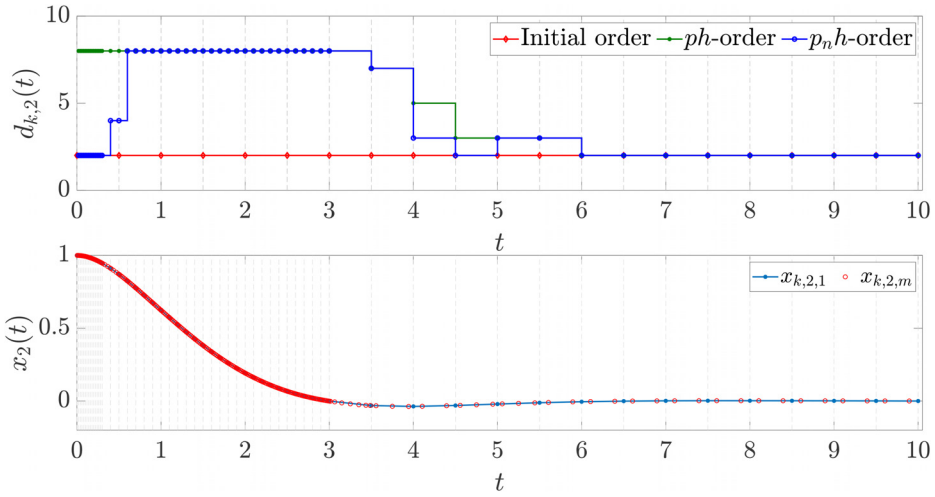

Fig. 2 Example 1: (top panel) polynomial degree for state $x_2(t)$ for the initial mesh (red diamonds markers), outcome degrees of the p_nh algorithm (blue empty circles markers), outcome degrees of the ph algorithm (green filled circles markers). Both algorithms prescribe to split the mesh intervals in an equal manner, i.e., for $0 \le t \le 3$ s. The ph algorithm (green filled circles markers) always raises the degree for all the states components if, in the generic interval, at least one states requires it. Consistently, the green curves in top panels of Figs. 1 and 2 are equal. Instead, the p_nh algorithm (blue empty circles markers) in the generic interval raises the degree only for that particular state that requires it. Comparing the blue curves in bottom panels of Figs. 1 and 2 it is evident, in each time interval, which is the state that requires the increase in the degree; (Bottom Panel) optimal $x_2(t)$ profile obtained with p_nh algorithm. Being equal to the mesh density (same dimension of the time intervals) more collocation points $x_{k,2,m}$ are added where the polynomial order required is higher.

selectively for each state only where they are needed (higher gradient). The advantages of the p_nh method with respect to the ph one are shown in Table 1. Here, the mesh column indicates the type of discretization mesh. In particular, *initial* refers to the mesh of the initial problem, whereas ph and p_nh refer to the final meshes produced by applying the methods in Ref. [25] and ours, respectively. It is worth remarking that both of them comply with the prescribed tolerance $\epsilon = 10^{-3}$. Furthermore, *vars* indicate the number of variables of the NLP associated with the final transcribed OCP. Then, *soln. time* indicates the time to solution of the optimal control problem with the refined mesh. Hence, *iter* refers to the number of algorithm iterations needed to reach the prescribed accuracy, whereas *tot.time* is the total CPU time elapsed to reach the imposed convergence criterion. It is worth remarking that it includes the necessary time to build and solve all the iter + 1 number of successive NLP problems and the time spent by the application of a number of iter mesh refinement algorithms. Finally, $e_{\max}$ is the maximum of the local error $e_{k,i,j}$, hence

$$e_{\max} = \max_{\substack{j=1,\dots,d_{k,i}^{+} \\ i=1,\dots,n_x \\ k=1,\dots,N}} (e_{k,i,j}) \tag{17}$$

It is worth stressing that the solution associated with a *coarser* discretization, produced by the p_nh method, still complies with the given tolerance ϵ while leading to an NLP with sensibly fewer variables than the ph algorithm. Hence, as a general benefit, the final NLP obtained from our method requires a reduced computational cost and a shorter CPU time. Furthermore, considering that ph and p_nh have the same final mesh in terms of S_k number and size, these two solutions can be compared at the nodal values. In Fig. 3, the deviations between the p_nh and ph final optimal solutions, are shown. Given that both the states x_1 and x_2 and the control u are in the order of 10^0, the deviations $\Delta(\cdot)$ are absolutely negligible and the two solutions are practically the same.

Table 1 Example 1: mesh refinement results for the three different types of mesh (initial mesh, ph final mesh, and p_nh final mesh)

Mesh	Vars	Soln. time	Iter	Tot. time	$e_{\max}$
Initial	140	138 ms	/	/	3.7×10^{-2}
ph	918	500 ms	4	23.1 s	9.1×10^{-4}
p_nh	769	400 ms	4	19.0 s	9.5×10^{-4}

5.2 Example 2: Rolling (Not Falling) Disk. The second optimal control problem, see Fig. 4, consists in driving a rolling (not falling) disk on a horizontal xy-plane from an initial position to a final one, while minimizing the control action. This can be expressed as follows:

$$\underset{X\in\mathbb{R}^6, U\in\mathbb{R}^2}{\text{minimize}} J = \int_0^T \left(u_1^2 + u_2^2\right) dt \tag{18a}$$

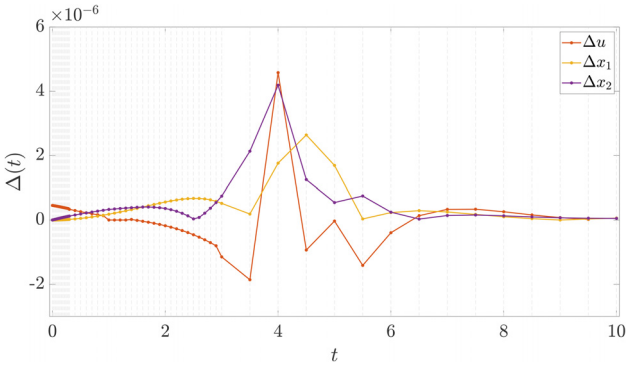

Fig. 3 Example 1: nodal differences between p_nh and ph final solutions. In this figure, we consider the solution obtained with the ph method and the one obtained with our approach (p_nh). The lines shown here represent the differences between these two solution at the nodal value (X_k, U_k), for the control input u, and the two states x_1 and x_2. It is evident that the two algorithm leads to the same solutions as the differences are negligible.

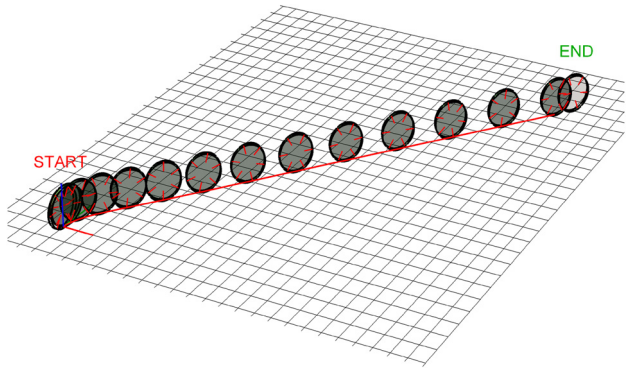

Fig. 4 Example 2: A rolling (not falling) disk—a typical nonholonomic benchmark system—controlled by a steering torque u_1 and a rolling torque u_2, is required to move from the stationary initial pose (START) to the stationary final pose (END). Snapshots of the optimal trajectory obtained by solving the associated NLP are depicted. The optimal trajectory obtained from the resulting NLP is shown in the accompanying video.[2]

$$\text{subject to} \quad \dot{x}_1 = \rho x_6 \cos(x_3) t \in [0, T] \tag{18b}$$

$$\dot{x}_2 = \rho x_6 \sin(x_3) \quad t \in [0, T] \tag{18c}$$

$$\dot{x}_3 = x_5 \quad t \in [0, T] \tag{18d}$$

$$\dot{x}_4 = x_6 \quad t \in [0, T] \tag{18e}$$

$$\dot{x}_5 = u_1 / I_\theta \quad t \in [0, T] \tag{18f}$$

$$\dot{x}_6 = u_2 / (I_\phi + m\rho^2) \quad t \in [0, T] \tag{18g}$$

$$x_6 \geq 0 \quad t \in [0, T] \tag{18h}$$

$$X(0) = [0, 0, \pi/2, 0, 0, 0] \tag{18i}$$

$$x_1(T) = 10, \quad x_2(T) = 10, \quad x_3(T) = \pi/4 \tag{18j}$$

where $T = 10$ s, m is the mass of the disk, ρ is the disk radius, I_θ is the diametral moment of inertia and I_ϕ is the axial disk moment of inertia. States x_1 and x_2 represent the trajectory of the contact point of the disk on the plane, x_3 is the heading angle measured with respect to the x-axis and x_4 is the rolling angle. Then, x_5, x_6 are the steering and rolling speed, respectively. The disk is controlled with two torques, u_1 for the steering action and u_2 for the rolling action. Starting at the origin ($x_1(0) = 0$, $x_2(0) = 0$) and heading along the y-axis ($x_3(0) = \pi/2$), the disk has to reach the final position described by Eq. (18*j*). The optimal trajectory obtained from the resulting NLP is shown in the accompanying video.[2]

It is worth recalling that, for a more compact notation, $X = [x_1, x_2, x_3, x_4, x_5, x_6]$ and $U = [u_1, u_2]$. In Figs. 5 and 6 (top panels) shown, for the p_nh algorithm and for the ph one, the final degree of state x_1 (horizontal coordinate) and of state x_3 (heading angle), respectively. With reference to Fig. 5 (top panel), both p_nh and ph return, for state x_1, almost the same polynomial degree for the whole time horizon. However, as evident from Fig. 6 (top panel), for state x_3 the two algorithms provide very different polynomial degrees for most of the time horizon. In particular, the green line is always at $d_{\max} = 8$. This implies that, in each mesh interval, there is at least one state (different from x_3, surely x_1) that requires that particular polynomial order. Instead, by applying the p_nh method, each state is approximated with a polynomial degree that is *just right*. More in detail, considering the solution in the range 6–9 s, the state x_1 requires the maximum achievable order $d_{\max} = 8$. This leads to an overkill in the approximation of the state x_3 with the same degree for the ph method, while a lower degree, as suggested by the p_nh method, is sufficient.

In Figs. 5 and 6 (bottom panels) the optimal profile of x_1 and x_3 are shown. In particular, considering the gray line and the collocation points in red, it is evident where the subdivision of the mesh intervals kicks in. The advantages of the p_nh method with respect to the ph one are summarized in Table 2. It is interesting to note that, even if p_nh method needs an extra iteration, it maintains a total time equal to the ph method. Furthermore, under the same tolerance $e_{\max}$, the number of variables involved in the last NLP (leading to the most accurate solution) of p_nh are 25% less than the original method. This represents a noteworthy reduction.

For what concerns the nodal deviations of the p_nh and ph solutions at nodal values, the maximum value is registered for u_2. Again, this difference is in the order of 10^{-6}, which is absolutely negligible given the control signal u_2 in the order of 10^0. Therefore, the two methods lead to equally accurate solutions.

5.3 Example 3: Free-Flying Robot. As the last example, we consider optimally controlling the pose of a 2D free-flying robot endowed with a propulsion system, see Fig. 7. This problem is taken from Ref. [26], with a modified cost function. Analytically, it can be framed as follows:

$$\underset{X \in \mathbb{R}^6, U \in \mathbb{R}^2}{\text{minimize}} \quad J = \int_0^T \left(u_1^2 + u_2^2\right) dt \tag{19a}$$

$$\text{subject to} \quad \dot{x}_1 = x_4 t \in [0, T] \tag{19b}$$

$$\dot{x}_2 = x_5 \quad t \in [0, T] \tag{19c}$$

$$\dot{x}_3 = x_6 \quad t \in [0, T] \tag{19d}$$

$$\dot{x}_4 = (u_1 + u_2)\cos(x_3) \quad t \in [0, T] \tag{19e}$$

$$\dot{x}_5 = (u_1 + u_2)\sin(x_3) \quad t \in [0, T] \tag{19f}$$

$$\dot{x}_6 = \alpha u_1 - \beta u_2 \quad t \in [0, T] \tag{19g}$$

$$X(0) = [-10, -10, \pi/2, 0, 0, 0] \tag{19h}$$

$$X(T) = [0, 0, 0, 0, 0, 0] \tag{19i}$$

where x_1 and x_2 are the inertial coordinates of the center of gravity, x_4 and x_5 are the corresponding velocities, and x_3 and x_6 are the thrust direction and the angular velocity, respectively. The thrust from the two engines, denoted by u_1 and u_2, serves as the control variables. Finally, $T = 12$ s and $\alpha = \beta = 1$. The optimal trajectory obtained from the resulting NLP is shown in the accompanying video.[3]

Due to space limitations, in Fig. 8, only the results for the thrust direction x_3 are depicted. Clearly, the final order required by the p_nh method is very different from the ph counterpart. Furthermore, considering the state profile, it is worth observing that the h-refinement strategy is favored at the beginning and at the end of the trajectory.

The advantages of the p_nh method with respect to the ph one, are shown in Table 3. Under the same practical accuracy, p_nh method guarantees about 20% less NLP variables involved, a 20% reduction in the final solution time, and about 27% reduction in total time with respect to the ph analog.

As for nodal differences between p_nh and ph solutions, the maximum value is reached for state x_1. A maximum difference in the order of 10^{-7} reveals that it is practically negligible given a state x_1 in the order of 10^0. Hence, the two methods lead to the same solution.

[2] https://youtu.be/K2kI8s5FbYY

[3] https://youtu.be/B8QKOD6EUo4

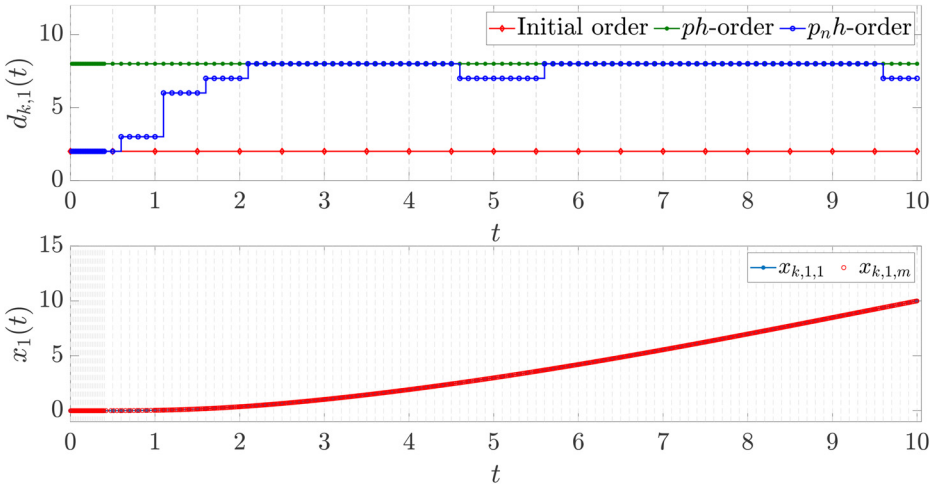

Fig. 5 Example 2: (top panel) polynomial degree for state $x_1(t)$ for the initial mesh (red diamonds markers), outcome degrees of the p_nh algorithm (blue empty circles markers), outcome degrees of the *ph* algorithm (green filled circles markers). Both algorithms prescribe to split the mesh intervals in an equal manner, i.e., for $0 \le t \le 10$ s and more densely for $0 \le t \le 0.5$. The *ph* algorithm (green filled circles markers) always raises the degree for all the states components if, in the generic interval, at least one states requires it. Consistently, the green curves in top panels of Figs. 5 and 6 are equal. Instead, the p_nh algorithm (blue empty circles markers) in the generic interval raises the degree only for that particular state that requires it. Comparing the blue curves in bottom panels of Figs. 5 and 6 it is evident, in each time interval, which is the state that requires the increase in the degree. As an example, with reference to the solution in the range of 6–9 s, while $x_1(t)$ requires the maximum achievable order $d_{max} = 8$, for the state $x_3(t)$ (see Fig. 6) order $d = 2$ is sufficient. This brings about an important saving of computational resources; (bottom panel) optimal $x_1(t)$ profile obtained with p_nh algorithm. Being equal to the mesh density (same dimension of the time intervals) more collocation points $x_{k,1,m}$ are added where the polynomial order required is higher.

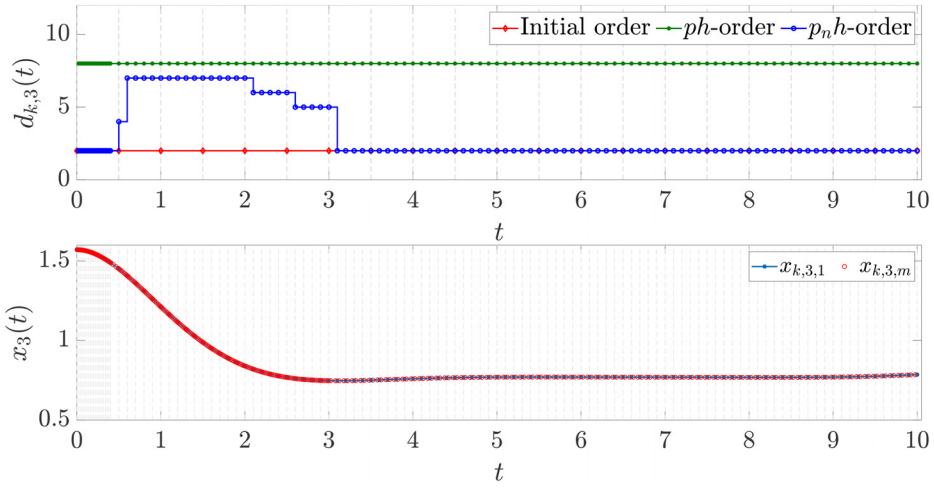

Fig. 6 Example 2: (top panel) polynomial degree for state $x_3(t)$ for the initial mesh (red diamonds markers), outcome degrees of the p_nh algorithm (blue empty circles markers), outcome degrees of the *ph* algorithm (green filled circles markers). Both algorithms prescribe to split the mesh intervals in an equal manner, i.e., for $0 \le t \le 10$ s and more densely for $0 \le t \le 0.5$. The *ph* algorithm (green filled circles markers) always raises the degree for all the states components if, in the generic interval, at least one states requires it. Consistently, the green curves in top panels of Figs. 5 and 6 are equal. Instead, the p_nh algorithm (blue empty circles markers) in the generic interval raises the degree only for that particular state that requires it. Comparing the blue curves in bottom panels of Figs. 5 and 6 it is evident that state $x_3(t)$ needs an increased polynomial degree only for high gradients, i.e., in the range 0–3 s. With reference to the solution in the range 3–9 s, while $x_1(t)$ (see Fig. 5) requires almost everywhere the maximum achievable order $d_{max} = 8$, for the state $x_3(t)$ (see Fig. 6) order $d = 2$ is sufficient. This brings about an important saving of computational resources; (bottom panel) optimal $x_3(t)$ profile obtained with p_nh algorithm. Being equal to the mesh density (same dimension of the time intervals) collocation points $x_{k,3,m}$ gather where the polynomial order required is higher.

Table 2 Example 2: mesh refinement results for the three different types of mesh (initial mesh, *ph* final mesh and p_nh final mesh)

Mesh	Vars	Soln. time	Iter	Tot. time	$e_{\max}$
Initial	400	0.643 s	/	/	3.6×10^{-2}
ph	6496	61.6 s	4	287.7 s	9.8×10^{-4}
p_nh	4810	32.4 s	5	287.1 s	9.8×10^{-4}

Table 3 Example 3: mesh refinement results for the three different types of mesh (initial mesh, *ph* final mesh and p_nh final mesh)

Mesh	Vars	Soln. time	Iter	Tot. time	$e_{\max}$
Initial	400	0.643 s	/	/	3.7×10^{-2}
ph	6944	79.8 s	4	367.1 s	8.9×10^{-4}
p_nh	5550	63.6 s	4	271.3 s	9.2×10^{-4}

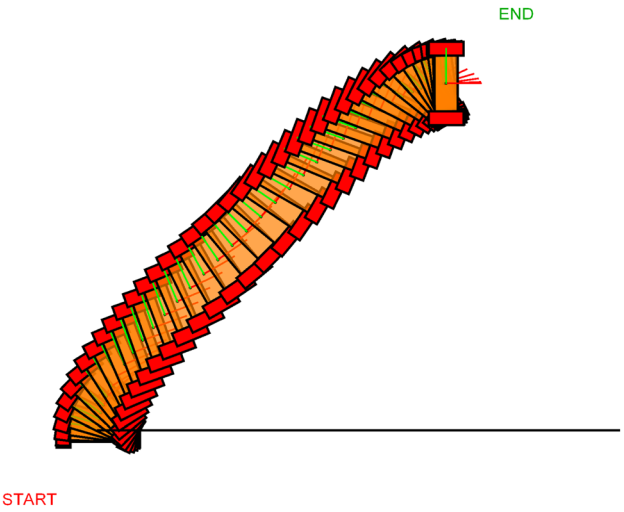

Fig. 7 Example 3: a 2D free-flying robot, representative of a VTOL aircraft endowed with a propulsion system, is required to move from the stationary initial pose (START) to the stationary final pose (END). Snapshots of the optimal trajectory obtained by solving the associated NLP are depicted. The optimal trajectory obtained from the resulting NLP is shown in the accompanying video.[3]

6 Conclusions

A p_nh mesh refinement method for numerical optimal control has been developed. The associated Algorithms 1 and 2 detail an innovative approach where the degrees of the polynomials approximating the solution of an OCP can assume distinct values for each state vector component and for each finite element.

The examples discussed to highlight the advantages of this approach in producing a final refined mesh which, while coping with the prescribed accuracy ϵ, allows a sensible reduction of the overall number of variables in the NLP, thus reducing the computational burden. More in detail, the most remarkable advantages are (i) an increased computational efficiency for the final enhanced mesh with solution accuracy still within the prescribed tolerance, (ii) a reduced risk of being trapped in local minima due to the reduced NLP size, and (iii) a gain of robustness of the convergence process due to the better-behaved solution landscapes.

These advantages are more evident for systems with a larger number of states, as shown in Examples 2 and 3. In Example 2, with respect to the classical *ph*-method, our p_nh-method registered a 26% reduction in the number of variables (from 6496 to 4810) and a 49% reduction of the solution time (from 61.6 s to 32.4 s). In Example 3, with respect to the classical *ph*-method, our p_nh-method registered a 20% reduction in the number of variables (from 6944 to 5550) and 20% reduction of the final solution time (from 79.8 s to 63.6 s). Therefore, when compared to the results obtained following the approach in Ref. [25], which is the closest contribution in the literature and inspired our improved method, at least in our

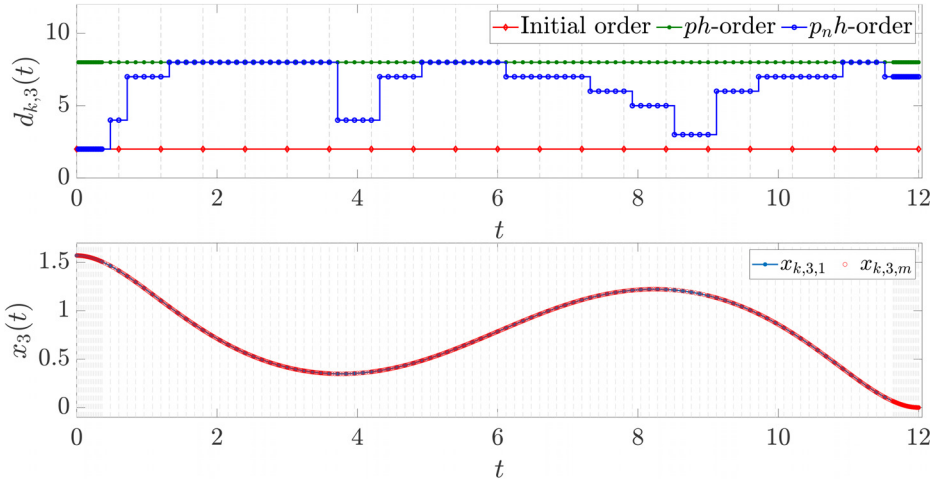

Fig. 8 Example 3: (top panel) polynomial degree for state $x_3(t)$ for the initial mesh (red diamonds markers), outcome degrees of the p_nh algorithm (blue empty circles markers), outcome degrees of the *ph* algorithm (green filled circles markers). Both algorithms prescribe to split the mesh intervals in an equal manner, i.e., for $0 \le t \le 12$ s and more densely for $0 \le t \le 0.2$ and for $11.8 \le t \le 12$. The *ph* algorithm (green filled circles markers) always raises the degree for all the states components if, in the generic interval, at least one state requires it. Instead, the p_nh algorithm (blue empty circles markers) in the generic interval raises the degree only for that particular state that requires it. With reference to the solution in the range 0–0.2 s, while other states require almost the maximum achievable order $d_{\max} = 8$, for the state $x_3(t)$ order $d = 2$ is sufficient. This brings about an important saving of computational resources; (bottom panel) optimal $x_3(t)$ profile obtained with p_nh algorithm. Being equal to the mesh density (same dimension of the time intervals) collocation points $x_{k,3,m}$ gather where the polynomial order required is higher.

numerical experiments, the solution are practically the same (differences are six orders of magnitude smaller than any state or control values). With the only drawback of being programmatically a little bit more involved, our method provides refined solutions we a remarkable gain in efficiency and robustness.

Data Availability Statement

The datasets generated and supporting the findings of this article are obtainable from the corresponding author upon reasonable request.

References

[1] Fahroo, F., and Ross, I. M., 2002, "Direct Trajectory Optimization by a Chebyshev Pseudospectral Method," J. Guid., Control, Dyn., **25**(1), pp. 160–166.
[2] Elnagar, G., Kazemi, M. A., and Razzaghi, M., 1995, "The Pseudospectral Legendre Method for Discretizing Optimal Control Problems," IEEE Trans. Autom. Control, **40**(10), pp. 1793–1796.
[3] Fahroo, F., and Ross, I. M., 2000, "A Spectral Patching Method for Direct Trajectory Optimization," J. Astronaut. Sci., **48**(2–3), pp. 269–286.
[4] Gong, Q., and Ross, I. M., 2006, "Autonomous Pseudospectral Knotting Methods for Space Mission Optimization," Adv. Astronaut. Sci., **124I**, pp. 779–794.
[5] Andersson, J. A. E., Gillis, J., Horn, G., Rawlings, J. B., and Diehl, M., 2019, "CasADi: A Software Framework for Nonlinear Optimization and Optimal Control," Math. Prog. Comput., **11**(1), pp. 1–36.
[6] Rao, A. V., Benson, D. A., Darby, C., Patterson, M. A., Francolin, C., Sanders, I., and Huntington, G. T., 2010, "Algorithm 902: GPOPS, a Matlab Software for Solving Multiple-Phase Optimal Control Problems Using the Gauss Pseudospectral Method," ACM Trans. Math. Software, **37**(2), pp. 1–39.
[7] Patterson, M. A., Weinstein, M., and Rao, A. V., 2013, "GPOPS-II: A Matlab Software for Solving Multiple-Phase Optimal Control Problems Using hp-Adaptive Gaussian Quadrature Collocation Methods and Sparse Nonlinear Programming," ACM Trans. Math. Software, **39**(3), pp. 1–36.
[8] Biegler, L. T., and Zavala, V. M., 2009, "Large-Scale Nonlinear Programming Using IPOPT: An Integrating Framework for Enterprise-Wide Dynamic Optimization," Comput. Chem. Eng., **33**(3), pp. 575–582.
[9] Gill, P. E., Murray, W., and Saunders, M. A., 2005, "SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization," SIAM Rev., **47**(1), pp. 99–131.
[10] Betts, J. T., Biehn, N., Campbell, S. L., and Huffman, W. P., 2000, "Compensating for Order Variation in Mesh Refinement for Direct Transcription Methods," J. Comput. Appl. Math., **125**(1–2), pp. 147–158.
[11] Ross, I. M., Fahroo, F., and Strizzi, J., 2003, "Adaptive Grids for Trajectory Optimization by Pseudospectral Methods," Adv. Astronaut. Sci., **25**(1), pp. 160–166.
[12] Fahroo, F., and Ross, I. M., 2000, "Trajectory Optimization by Indirect Spectral Collocation Methods," AIAA Paper No. 2000-4028.
[13] Binder, T., Cruse, A., Villar, C. A. C., and Marquardt, W., 2000, "Dynamic Optimization Using a Wavelet Based Adaptive Control Vector Parameterization Strategy," Comput. Chem. Eng., **24**(2–7), pp. 1201–1207.
[14] Binder, T., Blank, L., Dahmen, W., and Marquardt, W., 2000, "Grid Refinement in Multiscale Dynamic Optimization," Elsevier, Amsterdam, The Netherlands, Report No. LPT-2000-11.
[15] Schlegel, M., Stockmann, K., Binder, T., and Marquardt, W., 2005, "Dynamic Optimization Using Adaptive Control Vector Parameterization," Comput. Chem. Eng., **29**(8), pp. 1731–1751.
[16] Jain, S., Tsiotras, P., and Zhou, H. M., 2007, "Adaptive Multiresolution Mesh Refinement for the Solution of Evolution PDEs," Proceedings of the IEEE Conference on Decision and Control, New Orleans, LA, Dec. 12–14, pp. 3525–3530.
[17] Jain, S., and Tsiotras, P., 2008, "Trajectory Optimization Using Multiresolution Techniques," J. Guid. Control Dyn., **31**(5), pp. 1424–1436.
[18] Garg, D., Patterson, M., Hager, W. W., Rao, A. V., Benson, D. A., and Huntington, G. T., 2010, "A Unified Framework for the Numerical Solution of Optimal Control Problems Using Pseudospectral Methods," Automatica, **46**(11), pp. 1843–1851.
[19] Darby, C. L., Hager, W. W., and Rao, A. V., 2011, "Direct Trajectory Optimization Using a Variable Low-Order Adaptive Pseudospectral Method," J. Spacecr. Rockets, **48**(3), pp. 433–445.
[20] Darby, C. L., Hager, W. W., and Rao, A. V., 2011, "An *hp*-Adaptive Pseudospectral Method for Solving Optimal Control Problems," Opt. Control Appl. Methods, **32**(4), pp. 476–502.
[21] Biegler, L. T., 2010, "*Nonlinear Programming: Concepts, Algorithms, and Applications to Chemical Processes*," Society for Industrial and Applied Mathematics, Philadelphia, PA.
[22] Canuto, C., Hussaini, M. Y., Quarteroni, A., and Zang, T. A., 1991, *Spectral Methods in Fluid Dynamics*, Springer, Heidelberg, Germany.
[23] Fornberg, B., 1996, *A Practical Guide to Pseudospectral Methods*, Cambridge University Press, New York.
[24] Babuška, I., and Suri, M., 1990, "The p- and h-p Versions of the Finite Element Method, an Overview," Comput. Methods Appl. Mech. Eng., **80**(1–3), pp. 5–26.
[25] Patterson, M. A., Hager, W. W., and Rao, A. V., 2015, "A ph Mesh Refinement Method for Optimal Control," Optimal Control Appl. Methods, **36**(4), pp. 398–421.
[26] Betts, J. T., 2010, *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming* (Advances in Design and Control), Siam, Philadelphia, PA.
[27] Hou, H., Hager, W. W., and Rao, A. V., 2012, "Convergence of a Gauss Pseudospectral Method for Optimal Control," AIAA Paper No. 2012–4452.
[28] Hou, H., 2013, "Convergence Analysis of Orthogonal Collocation Methods for Unconstrained Optimal Control," Ph.D. thesis, University of Florida, Gainesville, FL.