Autonomous Robots (2024) 48:2
https://doi.org/10.1007/s10514-023-10155-z

Collocation methods for second and higher order systems

Siro Moreno-Martín[1] · Lluís Ros[1] · Enric Celaya[1]

Received: 15 February 2023 / Accepted: 17 December 2023 / Published online: 28 January 2024

Abstract
It is often unnoticed that the predominant way to use collocation methods is fundamentally flawed when applied to optimal control in robotics. Such methods assume that the system dynamics is given by a first order ODE, whereas robots are often governed by a second or higher order ODE involving configuration variables and their time derivatives. To apply a collocation method, therefore, the usual practice is to resort to the well known procedure of casting an Mth order ODE into M first order ones. This manipulation, which in the continuous domain is perfectly valid, leads to inconsistencies when the problem is discretized. Since the configuration variables and their time derivatives are approximated with polynomials of the same degree, their differential dependencies cannot be fulfilled, and the actual dynamics is not satisfied, not even at the collocation points. This paper draws attention to this problem, and develops improved versions of the trapezoidal and Hermite–Simpson collocation methods that do not present these inconsistencies. In many cases, the new methods reduce the dynamics transcription error in one order of magnitude, or even more, without noticeably increasing the cost of computing the solutions.

Keywords Collocation methods · Trajectory optimization · Optimal control · Second and higher order systems

1 Introduction

Direct collocation methods have proven to be powerful tools for solving optimal control problems in robotics (Posa et al., 2016; Pardo et al., 2016; Kelly, 2017; Hereid et al., 2018; Tedrake, 2023). Initially developed for aeronautics and astrodynamics applications (Hargraves and Paris, 1987; Conway and Paris, 2010), these methods have become very popular and of widespread use in the context of trajectory optimization and model predictive control, thanks to a few key advantages over indirect approaches based on the Pontryagin conditions of optimality: in general, they are easier to implement and show larger regions of convergence, and do not require estimations of the costate variables, which may be difficult to obtain accurately. Helpful tutorials and monographs like Kelly (2017) or Betts (2010), as well as open-source packages for nonlinear optimization (Wächter and Biegler, 2006), numerical optimal control (Kelly, 2017; Becerra, 2010; Andersson et al., 2019), or model-based design and verification (The Drake Team, 2023), are also contributing to their rapid dissemination among the community.

Direct collocation methods involve the transcription of the continuous-time optimal control problem into a finite-dimensional nonlinear programming (NLP) problem (Kelly, 2017). The transcription is based on partitioning the time history of the control and state variables into a number of intervals delimited by knot points. The system dynamics is then discretized in each interval by imposing the differential constraints at a set of collocation points, which may coincide, or not, with the chosen knot points. The cost function is also approximated using the values taken by the variables at such points, and the NLP problem is formulated using them. Once this problem is solved, a continuous solution is built using interpolating polynomials that satisfy the dynamics equations at the collocation points.

The general formulation of most collocation methods assumes that the system dynamics is governed by a first order ODE of the form

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

✉ Siro Moreno-Martín
smorenom@iri.upc.edu

Lluís Ros
ros@iri.upc.edu

Enric Celaya
enric.celaya@gmail.com

[1] Institut de Robòtica i Informàtica Industrial (CSIC-UPC), Llorens Artigas 4-6, 08028 Barcelona, Catalonia, Spain

where $\boldsymbol{x}(t)$ and $\boldsymbol{u}(t)$ are the state trajectory and the control function, respectively (Tedrake, 2023). In robotics, however, as in mechanics in general, the evolution of the system is often determined by a second order ODE of the form

$$\ddot{\boldsymbol{q}}(t) = \boldsymbol{g}\left(\boldsymbol{q}(t), \dot{\boldsymbol{q}}(t), \boldsymbol{u}(t), t\right), \tag{2}$$

where $\boldsymbol{q}(t)$ is the configuration trajectory and $\dot{\boldsymbol{q}}(t)$ is its time derivative. To apply a general collocation method, therefore, the usual procedure is to define $\boldsymbol{v}(t) = \dot{\boldsymbol{q}}(t)$ and write (2) as

$$\begin{cases} \dot{\boldsymbol{q}}(t) = \boldsymbol{v}(t), & \text{(3a)} \\ \dot{\boldsymbol{v}}(t) = \boldsymbol{g}(\boldsymbol{q}(t), \boldsymbol{v}(t), \boldsymbol{u}(t), t). & \text{(3b)} \end{cases}$$

which, if we define $\boldsymbol{x}(t) = (\boldsymbol{q}(t), \boldsymbol{v}(t))$, corresponds formally to (1). Yet, this raises a consistency issue. Since the collocation method locally approximates $\boldsymbol{q}(t)$ and $\boldsymbol{v}(t)$ by polynomials of the same degree, imposing

$$\boldsymbol{v}(t) = \dot{\boldsymbol{q}}(t) \tag{4}$$

only at the collocation points does not grant the satisfaction of (4) over the continuous time domain. Even more striking, perhaps, is the fact that, as we demonstrate in this paper, imposing (3) at the collocation points does not imply the satisfaction of (2), not even at these points, which contributes to increase the dynamic transcription error along the obtained trajectories. This hinders the possibility to reach a correct solution since, even if $\boldsymbol{u}(t)$ produces the expected trajectory for $\boldsymbol{v}(t)$, its integration will rarely coincide with the function obtained for $\boldsymbol{q}(t)$. In other words, the state trajectory $\boldsymbol{x}(t)$ will be inconsistent in general. Note also that, while second order ODEs could be discretized using Nyström methods (Hairer et al., 1993), the main advantage of these methods arises when the right-hand side of (2) does not depend on $\dot{\boldsymbol{q}}$, which seldom occurs in robotics.

In this paper we present modified versions of the trapezoidal and Hermite–Simpson collocation methods specifically addressed to guarantee that the collocation polynomials fulfill (4), while satisfying (2) at the collocation points, thereby increasing the accuracy of the obtained solutions. The paper is an extended version of an earlier work we presented in RSS'2022 (Moreno-Martín et al., 2022). In this new version, the original methods for second order ODEs are further generalized to deal with ODEs of arbitrary order M

$$\boldsymbol{q}^{(M)}(t) = \boldsymbol{g}(\boldsymbol{q}(t), \dot{\boldsymbol{q}}(t), \ldots, \boldsymbol{q}^{(M-1)}(t), \boldsymbol{u}(t), t), \tag{5}$$

which are less common but may arise in flexible, elastic, or soft robots for example (De Luca and Book, 2016; Della Santina, 2020), or when increased smoothness is sought in the computed solutions (Sect. 7.4). In addition, we also study the theoretical accuracy of all methods and offer a more thorough comparison between them.

By means of illustrative benchmark problems, the paper demonstrates that the new methods reduce substantially the dynamics error (in one order of magnitude or even more depending on the number of knot points) without noticeably increasing the computational time needed to solve the transcribed NLP problems. As a result, the state and control trajectories $\boldsymbol{x}(t)$ and $\boldsymbol{u}(t)$ will be mutually more consistent, which facilitates their tracking with a feedback controller.

The rest of the paper is structured as follows. Section 2 formulates the optimal control problem to be solved and delimits the specific transcription problem that we face in this paper. To prepare the ground for later developments, Sect. 3 reviews the conventional trapezoidal and Hermite–Simpson methods and pinpoints their limitations on transcribing 2nd order ODEs. Improved versions of these methods are then developed in Sect. 4 for 2nd order systems, and for Mth order ones in Sect. 5. The methods are summarized and compared in Sect. 6, where tools to assess their accuracy are also provided. The performance of all methods is analyzed in Sect. 7 with the help of examples, and the paper conclusions are finally given in Sect. 8.

2 Problem formulation

The optimal control problem that concerns us in this paper consists of finding state and action trajectories $\boldsymbol{x}(t)$ and $\boldsymbol{u}(t)$, and a final time t_f, that

minimize

$$K(\boldsymbol{x}_f, t_f) + \int_0^{t_f} L(\boldsymbol{x}(t), \boldsymbol{u}(t))\, dt \tag{6a}$$

subject to

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t), t), \qquad t \in [0, t_f] \tag{6b}$$

$$\boldsymbol{p}(\boldsymbol{x}(t), \boldsymbol{u}(t)) \leq \boldsymbol{0}, \qquad t \in [0, t_f] \tag{6c}$$

$$\boldsymbol{b}(\boldsymbol{x}_0, \boldsymbol{x}_f, t_f) = \boldsymbol{0}, \tag{6d}$$

$$t_f \geq 0, \tag{6e}$$

where $\boldsymbol{x}_0 = \boldsymbol{x}(0)$, and $\boldsymbol{x}_f = \boldsymbol{x}(t_f)$, the terms $K(\boldsymbol{x}_f, t_f)$ and $L(\boldsymbol{x}(t), \boldsymbol{u}(t))$ are terminal and running cost functions, respectively, (6b) is an ODE modeling the system dynamics, and (6c) and (6d) encompass the path and boundary constraints.

We note that, while Eq. (6b) has the appearance of a first order ODE, in robotics it often takes the form

$$\left.\begin{aligned} \dot{\boldsymbol{x}}_1 &= \boldsymbol{x}_2 \\ \dot{\boldsymbol{x}}_2 &= \boldsymbol{x}_3 \\ &\vdots \\ \dot{\boldsymbol{x}}_{M-1} &= \boldsymbol{x}_M \\ \dot{\boldsymbol{x}}_M &= \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{u}, t) \end{aligned}\right\} \tag{7}$$

where

$$\boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_M) = (\boldsymbol{q}, \dot{\boldsymbol{q}}, \ldots, \boldsymbol{q}^{(M-1)}), \tag{8}$$

so in such cases it actually encodes an Mth order ODE like (5), or (2) if $M = 2$.

Solving Problem (6) via collocation involves partitioning the time history of the control and state variables into N intervals delimited by $N + 1$ knot points t_k, $k = 0, \ldots, N$, then transcribing Eqs. (6a)–(6c) into appropriate discretizations expressed in terms of the values $\boldsymbol{x}_k = \boldsymbol{x}(t_k)$ and $\boldsymbol{u}_k = \boldsymbol{u}(t_k)$, and finally solving the constrained optimization problem that results.

The transcriptions of (6a) and (6c) are relatively straightforward and less relevant in the context of this paper. They can be done, for example, by approximating the integral in (6a) using some quadrature rule, and enforcing (6c) for all knot points t_k. The transcription of (6b), in contrast, is substantially more involved, and will be the main subject of the rest of this paper. In particular, we seek to construct appropriate polynomial approximations of the solutions $\boldsymbol{x}(t)$ of (6b) for each interval $[t_k, t_{k+1}]$. These approximations will be defined as solutions of systems of equations which, when considered together for all intervals, will form a proper transcription of (6b) over the whole time domain $[0, t_f]$.

In what follows, for each interval $[t_k, t_{k+1}]$ we shall use the shifted time variable $\tau = t - t_k$, and the interval width $h = t_{k+1} - t_k$.

3 Methods for first order systems

Two of the most widely used transcriptions of (6b) are those of the trapezoidal and Hermite–Simpson methods, which assume no particular form for (6b). To see where these transcriptions incur in dynamical error, and ease the development of the new methods, we briefly explain how they approximate (6b) and obtain their approximation polynomials for the state. Our results match those by Betts (2010) and Kelly (2017), but we follow a derivation process that is closer to Hargraves and Paris (1987), which facilitates the transition to our new methods in Sects. 4 and 5.

3.1 Trapezoidal collocation

In trapezoidal collocation, the state trajectories are approximated by quadratic polynomials. For $t \in [t_k, t_{k+1}]$, we can write the polynomial approximation for a component x of the state, and its temporal derivative, as

$$x(t) = a + b\tau + c\tau^2, \tag{9a}$$
$$\dot{x}(t) = b + 2c\tau, \tag{9b}$$

where a, b, and c are real coefficients. To facilitate the application of collocation constraints, however, we will rewrite $x(t)$ using the three parameters

$$x_k = x(t_k), \tag{10}$$
$$\dot{x}_k = \dot{x}(t_k), \tag{11}$$
$$\dot{x}_{k+1} = \dot{x}(t_{k+1}). \tag{12}$$

Evaluating the right-hand sides of (10)–(12) using (9) we obtain

$$\begin{bmatrix} x_k \\ \dot{x}_k \\ \dot{x}_{k+1} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 2h \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix}, \tag{13}$$

so solving for a, b, c and substituting the resulting expressions in (9a), we have

$$x(t) = x_k + \dot{x}_k\tau + \frac{\tau^2}{2h}(\dot{x}_{k+1} - \dot{x}_k). \tag{14}$$

Equation (14) is known as the interpolation polynomial, as it allows us to estimate the intermediate states for $t \in [t_k, t_{k+1}]$, once the NLP problem has been solved.

Now, following Hairer et al. (2002, p. 30), we determine the three parameters of (14) by enforcing the initial value constraint $x(t_k) = x_k$ and two collocation constraints of the form

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

for two different time instants $t \in [t_k, t_{k+1}]$. From (14) we see that $x(t_k) = x_k$ by construction. As for the collocation constraints, the trapezoidal method imposes them at the knot points t_k and t_{k+1}, so it must be

$$\dot{x}_k = f_k, \tag{15}$$
$$\dot{x}_{k+1} = f_{k+1}, \tag{16}$$

where f_k is a shorthand for $f(\boldsymbol{x}_k, \boldsymbol{u}_k, t_k)$. The value x_{k+1}, then, is obtained by evaluating (14) for $\tau = h$. This results

in the constraint

$$x_{k+1} = x_k + \frac{h}{2}(\dot{x}_{k+1} + \dot{x}_k), \tag{17}$$

which ensures the continuity of the trajectory across intervals k and $k+1$.

Note that Eqs. (15)–(17) already form a transcription of our ODE in the interval $[t_k, t_{k+1}]$ since, if $\boldsymbol{x}_k$, $\boldsymbol{u}_k$, and $\boldsymbol{u}_{k+1}$ were known, these equations would suffice to determine the three unknowns $\dot{x}_k$, $\dot{x}_{k+1}$, and x_{k+1}. However, we can also substitute (15) and (16) into (17) to obtain the more compact expression

$$x_{k+1} = x_k + \frac{h}{2}(f_{k+1} + f_k), \tag{18}$$

which we recognize as the common transcription rule in trapezoidal collocation (Kelly, 2017; Betts, 2010). Observe that the continuity between the polynomials of intervals k and $k+1$ is granted for the first derivative as, by construction, they both satisfy $\dot{x}_{k+1} = f_{k+1}$. However, second and higher order continuity is not preserved in general.

3.2 Hermite–Simpson collocation

In Hermite–Simpson collocation, the state trajectories in each interval are approximated by cubic polynomials:

$$x(t) = a + b\tau + c\tau^2 + d\tau^3, \tag{19a}$$
$$\dot{x}(t) = b + 2c\tau + 3d\tau^2. \tag{19b}$$

By analogy with the trapezoidal method, we first express the polynomial coefficients in terms of the parameters

$$\begin{aligned} x_k &= x(t_k), \\ \dot{x}_k &= \dot{x}(t_k), \\ \dot{x}_c &= \dot{x}(t_c), \\ \dot{x}_{k+1} &= \dot{x}(t_{k+1}), \end{aligned}$$

where $t_c = t_k + h/2$, and the extra parameter $\dot{x}_c$ is added because four parameters are needed to determine a third degree polynomial. Evaluating these identities using (19), solving for $a, \ldots, d$, and substituting the expressions in (19a), we obtain the interpolation polynomial

$$\begin{aligned} x(t) = x_k + \dot{x}_k\tau - \frac{\tau^2}{2h}(3\dot{x}_k - 4\dot{x}_c + \dot{x}_{k+1}) \\ + \frac{\tau^3}{3h^2}(2\dot{x}_k - 4\dot{x}_c + 2\dot{x}_{k+1}). \end{aligned} \tag{20}$$

In order to determine the four parameters of (20), four conditions have to be imposed, and the Hermite–Simpson method makes this by fixing $x(t_k) = x_k$ (which holds by construction) and imposing the dynamics at the two bounding knot points and the midpoint between them:

$$\dot{x}_k = f_k, \tag{21}$$
$$\dot{x}_{k+1} = f_{k+1}, \tag{22}$$
$$\dot{x}_c = f_c. \tag{23}$$

In the latter equation, $f_c = f(\boldsymbol{x}_c, \boldsymbol{u}_c, t_c)$, where $\boldsymbol{x}_c = \boldsymbol{x}(t_c)$, and $\boldsymbol{u}_c = \boldsymbol{u}(t_c)$. Moreover, the values x_c that are needed in f_c can be expressed in terms of the above four parameters by evaluating (20) for $\tau = h/2$, which yields

$$x_c = x_k + \frac{h}{24}(5\dot{x}_k + 8\dot{x}_c - \dot{x}_{k+1}). \tag{24}$$

Finally, the continuity constraint between intervals k and $k+1$ is obtained by evaluating (20) for $\tau = h$:

$$x_{k+1} = x_k + \frac{h}{6}(\dot{x}_k + 4\dot{x}_c + \dot{x}_{k+1}). \tag{25}$$

Equations (21)–(25) already form a transcription of our ODE in $[t_k, t_{k+1}]$, but a transcription involving less variables can be obtained by substituting (21)–(23) in (25) and (24), which gives

$$x_{k+1} = x_k + \frac{h}{6}(f_k + 4f_c + f_{k+1}), \tag{26a}$$
$$x_c = x_k + \frac{h}{24}(5f_k + 8f_c - f_{k+1}). \tag{26b}$$

If preferred, we can also remove the dependence on f_c in (26b). This is achieved by isolating f_c from (26a) and substituting the result in (26b), which yields the alternative transcription

$$x_{k+1} = x_k + \frac{h}{6}(f_k + 4f_c + f_{k+1}), \tag{27a}$$
$$x_c = \frac{1}{2}(x_k + x_{k+1}) + \frac{h}{8}(f_k - f_{k+1}). \tag{27b}$$

Both transcriptions in (26) and (27) are called separated forms of Hermite–Simpson collocation, in the sense they both keep x_c as a decision variable of the problem. They are equivalent, but the one in (27) allows us to eliminate x_c by substituting (27b) in (27a), which results in a single equation that is known as the compressed form of Hermite–Simpson collocation (Kelly, 2017; Betts, 2010). While the use of a separated form tends to be better when working with a small number of intervals, the compressed form is preferable when such a number is large (Kelly, 2017).

Note that, despite the polynomial approximation for each interval between consecutive knot points is of third degree,

continuity through knot points is only granted for the state trajectory and its first derivative.

3.3 Trajectory interpolation

After solving the NLP problem, values of the state and control variables at all collocation points are available. A continuous approximation to the optimal trajectory for the state is then obtained by substituting (15)–(16), or (21)–(23), in the corresponding interpolating polynomials (14) and (20), for the trapezoidal and Hermite–Simpson methods, respectively. The approximation of the control trajectory within each interval is obtained, in the trapezoidal case, by linear interpolation of the control values. In the Hermite–Simpson case, different options are possible. Some authors handle the midpoint control as an independent variable and use a quadratic interpolation of the three control values available in each interval (Kelly, 2017), while others prefer a linear interpolation and enforce the midpoint value to be the mean of the two bounding values (Topputo and Zhang, 2014). In this paper we follow the former option.

3.4 Downsides of the methods

In a first order dynamical system, imposing (15)–(16) or (21)–(23) grants that the system dynamics is effectively satisfied at the collocation points. The same is not true when a second order system is cast into a first order one via (3). To see why, note that the constraint $\dot{q}(t) = v(t)$ is only imposed at the collocation points, but not in between them, so that, even if the curves $\dot{q}(t)$ and $v(t)$ coincide at such points, their derivatives may be different in them (Fig. 1). Therefore, $\ddot{q}(t) \neq \dot{v}(t)$ in general and, in particular, also at the collocation points. As a consequence, even if $\dot{q}_k = v_k$ and $\dot{v}_k = g(\boldsymbol{q}_k, \boldsymbol{v}_k, \boldsymbol{u}_k, t_k)$, this does not imply that the expected relation $\ddot{q}_k = g(\boldsymbol{q}_k, \dot{\boldsymbol{q}}_k, \boldsymbol{u}_k, t_k)$ is satisfied, what means that, with a transcription based on (3), the system dynamics in (2) is not granted, not even at the collocation points. This problem is solved in the second order collocation methods introduced in the next section.

A related problem of first order methods is that, when the trajectories are approximated with their interpolation polynomials $q(t)$ and $v(t)$, the difference $v(t) - \dot{q}(t) \neq 0$ makes the state trajectory inconsistent, so that, if we try to follow it with a controller, since the configuration and velocity trajectories are incompatible, both cannot be followed at the same time. An attempt to solve this may consist in ignoring the configuration trajectory and replacing it by the integral of the velocity, but the resulting configuration trajectory may violate the problem constraints, e.g., the final configuration may be different from the expected one. Alternatively, one can try to replace the velocity trajectory by the derivative of $q(t)$, but in this case, since the dynamic constraint satisfied

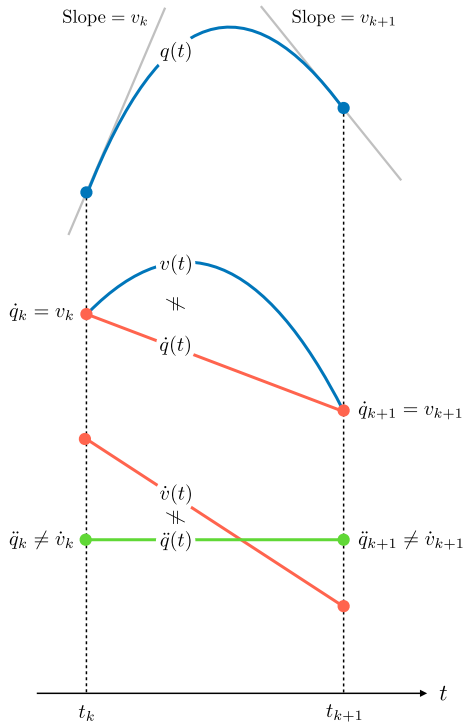

Fig. 1 Inconsistencies that arise when a collocation method for first order systems is applied to a second order ODE $\ddot{\boldsymbol{q}} = \boldsymbol{g}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{u}, t)$. The figure illustrates the case of the trapezoidal method, whose quadratic approximations $q(t)$ and $v(t)$ are depicted in blue. The red and green curves correspond to first and second derivatives of these trajectories, respectively

at collocation point k is $\dot{v}_k = g(\boldsymbol{q}_k, \boldsymbol{v}_k, \boldsymbol{u}_k, t_k)$, and $v_k = \dot{q}_k$ but $\dot{v}_k \neq \ddot{q}_k$, the dynamic constraint $\ddot{q}_k = g(\boldsymbol{q}_k, \dot{\boldsymbol{q}}_k, \boldsymbol{u}_k, t_k)$ will not be satisfied with the computed $\boldsymbol{u}_k$.

4 Methods for second order systems

To solve the inconsistency problems just explained, we propose alternative formulations for the trapezoidal and Hermite–Simpson collocation methods in which the dynamic constraints are directly imposed on the second derivative of the configuration variables, instead of on the first derivative of the state variables. By doing so, the velocity variables are not treated as independent from the configuration ones, but explicitly defined as $v(t) \equiv \dot{q}(t)$. In this way, the discrepancy between $q(t)$ and $v(t)$ is fully removed, and the second order dynamics is satisfied at each collocation point.

4.1 Trapezoidal method for second order systems

The essential feature characterizing trapezoidal collocation is that the dynamics is imposed just at the knot points or, otherwise said, that each interval bound is a collocation point. When the dynamics is governed by the second order ODE

in (2), using the same strategy as the trapezoidal method consists in imposing (2) at each interval bound. This means that, for each interval, two constraints have to be imposed on the second derivative of the polynomial approximating each component q of the configuration. But, since the second derivative of a quadratic polynomial is constant, only one constraint could be imposed on it. This implies that the interpolating polynomial $q(t)$ must be of degree three at least. So, we will have, for a given interval $[t_k, t_{k+1}]$,

$$q(t) = a + b\tau + c\tau^2 + d\tau^3, \tag{28a}$$
$$\dot{q}(t) = b + 2c\tau + 3d\tau^2, \tag{28b}$$
$$\ddot{q}(t) = 2c + 6d\tau. \tag{28c}$$

To determine the coefficients a, b, c, d, we need to impose four conditions. While in the trapezoidal method three conditions were used (the value x_k at the initial bound and the derivatives $\dot{x}_k$ and $\dot{x}_{k+1}$ at the two bounds), here we will impose, in addition to the initial value q_k and the second derivative at the interval bounds $\ddot{q}_k$ and $\ddot{q}_{k+1}$, the value $\dot{q}_k$ of the first derivative at the initial bound. Note that, for a cubic polynomial, no more than two independent conditions can be fulfilled by its second derivative, so imposing the dynamics at the midpoint of the interval as in the Hermite–Simpson method is not possible here. Thus we will use as parameters:

$$\begin{aligned} q_k &= q(t_k) \\ \dot{q}_k &= \dot{q}(t_k) \\ \ddot{q}_k &= \ddot{q}(t_k) \\ \ddot{q}_{k+1} &= \ddot{q}(t_{k+1}). \end{aligned}$$

Evaluating these identities using (28) and solving for a, b, c, d, we can write the interpolation polynomial $q(t)$ as:

$$q(t) = q_k + \dot{q}_k\tau + \ddot{q}_k\frac{\tau^2}{2} + \frac{\tau^3}{6h}(\ddot{q}_{k+1} - \ddot{q}_k). \tag{29}$$

The evaluation of this polynomial and its derivative $\dot{q}(t)$ for $\tau = h$ yields

$$q_{k+1} = q_k + \dot{q}_k h + \frac{h^2}{6}(\ddot{q}_{k+1} + 2\ddot{q}_k), \tag{30a}$$
$$\dot{q}_{k+1} = \dot{q}_k + \frac{h}{2}(\ddot{q}_{k+1} + \ddot{q}_k), \tag{30b}$$

and imposing the collocation constraints

$$\ddot{q}_k = g_k, \tag{31a}$$
$$\ddot{q}_{k+1} = g_{k+1}, \tag{31b}$$

where $g_k = g(\boldsymbol{q}_k, \dot{\boldsymbol{q}}_k, \boldsymbol{u}_k, t_k)$, we finally obtain the trapezoidal method for second order systems:

$$q_{k+1} = q_k + \dot{q}_k h + \frac{h^2}{6}(g_{k+1} + 2g_k), \tag{32a}$$
$$\dot{q}_{k+1} = \dot{q}_k + \frac{h}{2}(g_{k+1} + g_k). \tag{32b}$$

Note that, in this case, the trapezoidal rule only applies for the velocity, but not for the configuration itself, which is given by Eq. (32a).

As opposed to the trapezoidal method for first order systems, the continuity between neighboring polynomials at the knot points is of second order in this case, since the collocation constraints impose the coincidence of the second derivative of $q(t)$. Second order continuity for the configuration trajectory implies smooth velocity profiles and continuous accelerations, which are desirable properties in many robotics applications (Constantinescu and Croft, 2000; Macfarlane and Croft, 2003; Berscheid and Kröger, 2021).

4.2 Hermite–Simpson method for second order systems

Our purpose now is to impose the second order dynamics on the two bounds and the midpoint of each interval, in similarity with the conventional Hermite–Simpson method. Clearly, if we want to impose three conditions to the second derivative of a polynomial $q(t)$, such a derivative must be quadratic at least, what implies that the polynomial must have degree four at least. Thus, we propose to approximate the configuration trajectory, and its derivatives, by

$$q(t) = a + b\tau + c\tau^2 + d\tau^3 + e\tau^4, \tag{33}$$
$$\dot{q}(t) = b + 2c\tau + 3d\tau^2 + 4e\tau^3, \tag{34}$$
$$\ddot{q}(t) = 2c + 6d\tau + 12e\tau^2. \tag{35}$$

Since five parameters are needed to determine the five coefficients of $q(t)$, we will use, in addition to the three accelerations $\ddot{q}_k, \ddot{q}_c, \ddot{q}_{k+1}$, the values of the configuration coordinate q_k and its derivative $\dot{q}_k$ at the initial point:

$$\begin{aligned} q_k &= q(t_k) \\ \dot{q}_k &= \dot{q}(t_k) \\ \ddot{q}_k &= \ddot{q}(t_k) \\ \ddot{q}_c &= \ddot{q}(t_c) \\ \ddot{q}_{k+1} &= \ddot{q}(t_{k+1}). \end{aligned}$$

Solving for the coefficients $a, \ldots, e$, we obtain the following expression for the interpolating polynomial:

$$q(t) = q_k + \dot{q}_k \tau + \frac{\tau^2}{2}\ddot{q}_k - \frac{\tau^3}{6h}(3\ddot{q}_k - 4\ddot{q}_c + \ddot{q}_{k+1}) + \frac{\tau^4}{6h^2}(\ddot{q}_k - 2\ddot{q}_c + \ddot{q}_{k+1}). \tag{36}$$

Evaluating (36) and its derivative for the value $\tau = h$ results in

$$q_{k+1} = q_k + \dot{q}_k h + \frac{h^2}{6}(\ddot{q}_k + 2\ddot{q}_c), \tag{37a}$$

$$\dot{q}_{k+1} = \dot{q}_k + \frac{h}{6}(\ddot{q}_k + 4\ddot{q}_c + \ddot{q}_{k+1}), \tag{37b}$$

and imposing the collocation constraints

$$\ddot{q}_k = g_k, \tag{38a}$$

$$\ddot{q}_c = g_c, \tag{38b}$$

$$\ddot{q}_{k+1} = g_{k+1}, \tag{38c}$$

yields

$$q_{k+1} = q_k + \dot{q}_k h + \frac{h^2}{6}(g_k + 2g_c), \tag{39a}$$

$$\dot{q}_{k+1} = \dot{q}_k + \frac{h}{6}(g_k + 4g_c + g_{k+1}), \tag{39b}$$

where we recognize that (39b) is the Simpson quadrature for the velocity. The terms g_c in these equations involve the midpoint coordinate $q_c = q(t_c)$, and the velocity $\dot{q}_c = \dot{q}(t_c)$, but these can be obtained by evaluating (36) and its derivative for $\tau = h/2$, and imposing (38), which yields

$$q_c = q_k + \frac{h}{2}\dot{q}_k + \frac{h^2}{96}(7g_k + 6g_c - g_{k+1}), \tag{40a}$$

$$\dot{q}_c = \dot{q}_k + \frac{h}{24}(5g_k + 8g_c - g_{k+1}). \tag{40b}$$

Note however that, since q_c and $\dot{q}_c$ are to be used in the evaluation of g_c, we may prefer not to express them in terms of g_c itself. For this we simply isolate g_c from (39b) and substitute the result in (40) to obtain:

$$q_c = q_k + \frac{h}{32}(13\dot{q}_k + 3\dot{q}_{k+1}) + \frac{h^2}{192}(11g_k - 5g_{k+1}), \tag{41a}$$

$$\dot{q}_c = \frac{1}{2}(\dot{q}_k + \dot{q}_{k+1}) + \frac{h}{8}(g_k - g_{k+1}). \tag{41b}$$

Equations (39) and (41) together constitute a separated form of the Hermite–Simpson method for 2nd order systems. Written in this way, (41) can be replaced in the expression of g_c in (39) to transcribe the problem in compressed form, which eliminates the need to treat q_c and $\dot{q}_c$ as decision variables.

In this collocation scheme, the continuity across knot points is also of second order due to the coincidence of the second derivative imposed by the collocation constraints, what gives rise to smooth, continuous acceleration trajectories just like in the second order trapezoidal method.

5 Extensions for higher order systems

Second order systems are, by far, the most common in robotics, but sometimes it may be necessary to deal with dynamical systems of a higher order M, whose dynamics is described by an ODE like (5), which we recall for convenience:

$$\boldsymbol{q}^{(M)}(t) = \boldsymbol{g}\left(\boldsymbol{q}(t), \dot{\boldsymbol{q}}(t), \ldots, \boldsymbol{q}^{(M-1)}(t), \boldsymbol{u}(t), t\right). \tag{42}$$

We next see how the new methods can be extended to transcribe (42).

5.1 The generalized trapezoidal method

To derive the trapezoidal method for Mth order systems we proceed as in Sect. 4.1. For each time interval $[t_k, t_{k+1}]$ we approximate each component q of the solution of (42) by a polynomial $q(t)$ whose Mth time derivative $q^{(M)}(t)$ is linear, so its two coefficients may be determined by imposing (42) at each interval bound. This implies that $q(t)$ must be of order $M+1$. Using $a_0, \ldots, a_{M+1}$ as coefficients, this polynomial, and its derivatives, may be written as

$$q(t) = \frac{a_0}{0!} + \frac{a_1}{1!}\tau + \ldots + \frac{a_{M+1}}{(M+1)!}\tau^{M+1} \tag{43a}$$

$$\dot{q}(t) = \frac{a_1}{0!} + \frac{a_2}{1!}\tau + \ldots + \frac{a_{M+1}}{M!}\tau^{M} \tag{43b}$$

$$\vdots$$

$$q^{(M)}(t) = a_M + a_{M+1}\tau, \tag{43c}$$

or, more compactly as

$$q^{(j)}(t) = \sum_{i=j}^{M+1} \frac{a_i}{(i-j)!}\tau^{i-j}, \tag{44}$$

for $j = 0, \ldots, M$. We then can determine a_M and a_{M+1} by imposing the two collocation constraints

$$q^{(M)}(t_k) = g_k, \tag{45}$$

$$q^{(M)}(t_{k+1}) = g_{k+1}, \tag{46}$$

where $g_k = g(\boldsymbol{q}_k, \dot{\boldsymbol{q}}_k, \ldots, \boldsymbol{q}_k^{(M-1)}, \boldsymbol{u}_k, t_k)$. With simple calculations we find that

$$a_M = g_k, \tag{47a}$$
$$a_{M+1} = \tfrac{1}{h}(g_{k+1} - g_k). \tag{47b}$$

The remaining coefficients $a_0, \ldots, a_{M-1}$ are determined by imposing the initial value constraints

$$q^{(j)}(t_k) = q_k^{(j)} \tag{48}$$

for $j = 0, \ldots, M-1$. Using (43) we see that the left hand side of (48) is a_j, so we readily obtain

$$a_j = q_k^{(j)} \tag{49}$$

for $j = 0, \ldots, M-1$. Finally, by evaluating (44) for $\tau = h$ we find that the generalized versions of the Tz-2 formulas in (32) are given by

$$q_{k+1}^{(j)} = \sum_{i=j}^{M+1} \frac{a_i}{(i-j)!} h^{i-j}, \tag{50}$$

for $j = 0, \ldots, M-1$.

One can check that, by particularizing (50) for $M = 1$ and $M = 2$, we obtain the equations of the trapezoidal method for first and second order systems given in (17) and (32), respectively.

5.2 The generalized Hermite–Simpson method

An analogous route can be followed to obtain a Hermite–Simpson method for Mth order systems. In this case, $q^{(M)}(t)$ must be quadratic in order to determine its coefficients by imposing the collocation constraints at t_k, t_{k+1}, and $t_c = t_k + h/2$. This means that $q(t)$ must be of degree $M+2$ now, so $q(t)$, and its derivatives, will take the form

$$q^{(j)}(t) = \sum_{i=j}^{M+2} \frac{a_i}{(i-j)!} \tau^{i-j} \tag{51}$$

for $j = 0, \ldots, M$. The last equation in (51) is

$$q^{(M)}(t) = a_M + a_{M+1}\tau + \frac{a_{M+2}}{2}\tau^2, \tag{52}$$

and its coefficients a_M, a_{M+1}, and a_{M+2} can be determined by imposing

$$q^{(M)}(t_k) = g_k, \tag{53}$$
$$q^{(M)}(t_c) = g_c, \tag{54}$$
$$q^{(M)}(t_{k+1}) = g_{k+1}, \tag{55}$$

where

$$g_c = g(\boldsymbol{q}_c, \dot{\boldsymbol{q}}_c, \ldots, \boldsymbol{q}_c^{(M-1)}, \boldsymbol{u}_c, t_c), \tag{56a}$$
$$\boldsymbol{q}_c^{(j)} = \boldsymbol{q}^{(j)}(t_c), \quad j = 0, \ldots, M-1. \tag{56b}$$

After simple calculations we find that

$$a_M = g_k, \tag{57a}$$
$$a_{M+1} = -\tfrac{1}{h}\left(3g_k - 4g_c + g_{k+1}\right), \tag{57b}$$
$$a_{M+2} = \tfrac{4}{h^2}(g_k - 2g_c + g_{k+1}). \tag{57c}$$

As in the trapezoidal method, the remaining coefficients are determined by the initial value constraints, and we have

$$a_j = q_k^{(j)}, \tag{58}$$

for $j = 0, \ldots, M-1$. The generalized versions of Eqs. (39) can then be obtained by evaluating the expressions up to order $M-1$ in (51) for $\tau = h$, and using $q^{(j)}(t_{k+1}) = q_{k+1}^{(j)}$. This yields

$$q_{k+1}^{(j)} = \sum_{i=j}^{M+2} \frac{a_i}{(i-j)!} h^{i-j} \tag{59}$$

for $j = 0, \ldots, M-1$.

As it happens in the Hermite–Simpson method for 2nd order systems, g_c in (57) requires the midpoint values $\boldsymbol{q}_c, \dot{\boldsymbol{q}}_c, \ldots, \boldsymbol{q}_c^{(M-1)}$, but these are easily obtained by evaluating (51) for $\tau = h/2$, which results in

$$q_c^{(j)} = \sum_{i=j}^{M+2} \frac{a_i}{(i-j)!} \left(\frac{h}{2}\right)^{i-j} \tag{60}$$

for $j = 0, \ldots, M-1$.

The terms a_{M+1} and a_{M+2} in (60) involve g_c and thus the midpoint coordinate q_c and its derivatives. However, we can remove the dependence of $q_c^{(j)}$ on g_c by using the last equation in (59), which is

$$q_{k+1}^{(M-1)} = q_k^{(M-1)} + \frac{h}{6}(g_k + 4g_c + g_{k+1}). \tag{61}$$

By isolating g_c from this equation we have

$$g_c = \frac{g_{k+1} - g_k}{4} + \frac{3q_{k+1}^{(M-1)} - 3q_k^{(M-1)}}{2h}, \tag{62}$$

which we can substitute in the expressions of a_{M+1} and a_{M+2} involved in (60). With these substitutions applied, (59)

and (60) form a separated form of the Hermite–Simpson method for Mth order systems. The condensed form is finally achieved by substituting the new version of (60) in the expressions of a_{M+1} and a_{M+2} in (59).

Again, one can verify that, for $M = 1$ and $M = 2$, Eqs. (59) and (60) yield the Hermite–Simpson formulas for first and second order systems given in (26), and in (39) and (41), respectively.

6 Comparison of the methods

Table 1 summarizes the equations for all methods of the trapezoidal and Hermite–Simpson families. For short, we refer to the methods in each family by *TZ-* and *HS-*, followed by a number that indicates the order assumed for the system dynamics. For the general *TZ-M* and *HS-M* methods, the table provides the equations for $q_{k+1}^{(M-l)}$, as well as $q_{k+1}^{(M-l)}$ and $q_c^{(M-l)}$, respectively, where l runs from 1 to M in all cases. We also specialize these equations for $l = 1, 2$, so the reader can realize that, within each family, the equations for a same value of l coincide for all orders.

In the table, the equations for the Hermite–Simpson methods are given in their separated form, and in the *HS-M* method we show those that result from applying the manipulations described in Sect. 5.2.

6.1 Problem size

It is not difficult to see that, for all methods in a same family, the number of variables (n_v), equations (n_e), and degrees of freedom (n_{DOF}) is the same in the resulting transcriptions of

Table 1 Collocation equations for all methods of the trapezoidal and Hermite–Simpson families

Method	Collocation equations
Tz-1	$\left\{ x_{k+1} = x_k + \frac{h}{2}(f_{k+1} + f_k) \right.$
Tz-2	$\begin{cases} \dot{q}_{k+1} = \dot{q}_k + \frac{h}{2}(g_{k+1} + g_k) \\ q_{k+1} = q_k + \dot{q}_k h + \frac{h^2}{6}(g_{k+1} + 2g_k) \end{cases}$
Tz-M	$\begin{cases} q_{k+1}^{(M-1)} = q_k^{(M-1)} + \frac{h}{2}(g_{k+1} + g_k) \\ q_{k+1}^{(M-2)} = q_k^{(M-2)} + q_k^{(M-1)} h + \frac{h^2}{6}(g_{k+1} + 2g_k) \\ \quad \vdots \\ q_{k+1}^{(M-l)} = \left(\sum_{i=0}^{l-1} \frac{h^i}{i!} q_k^{(i+M-l)} \right) + \frac{h^l}{(l+1)!}(l\, g_k + g_{k+1}) \end{cases}$
HS-1	$\begin{cases} x_{k+1} = x_k + \frac{h}{6}(f_k + 4f_c + f_{k+1}) \\ x_c = \frac{1}{2}(x_k + x_{k+1}) + \frac{h}{8}(f_k - f_{k+1}) \end{cases}$
HS-2	$\begin{cases} \dot{q}_{k+1} = \dot{q}_k + \frac{h}{6}(g_k + 4g_c + g_{k+1}) \\ \dot{q}_c = \frac{1}{2}(\dot{q}_k + \dot{q}_{k+1}) + \frac{h}{8}(g_k - g_{k+1}) \\ q_{k+1} = q_k + \dot{q}_k h + \frac{h^2}{6}(g_k + 2g_c) \\ q_c = q_k + \frac{h}{32}(13\dot{q}_k + 3\dot{q}_{k+1}) + \frac{h^2}{192}(11g_k - 5g_{k+1}) \end{cases}$
HS-M	$\begin{cases} q_{k+1}^{(M-1)} = q_k^{(M-1)} + \frac{h}{6}(g_k + 4g_c + g_{k+1}) \\ q_c^{(M-1)} = \frac{1}{2}\left(q_k^{(M-1)} + q_{k+1}^{(M-1)}\right) + \frac{h}{8}(g_k - g_{k+1}) \\ q_{k+1}^{(M-2)} = q_k^{(M-2)} + q_k^{(M-1)} h + \frac{h^2}{6}(g_k + 2g_c) \\ q_c^{(M-2)} = q_k^{(M-2)} + \frac{h}{32}\left(13q_k^{(M-1)} + 3q_{k+1}^{(M-1)}\right) + \frac{h^2}{192}(11g_k - 5g_{k+1}) \\ \quad \vdots \\ q_{k+1}^{(M-l)} = \left(\sum_{i=0}^{l-1} \frac{h^i}{i!} q_k^{(i+M-l)} \right) + \frac{h^l\left(l^2 g_k + 4l g_c + (2-l) g_{k+1}\right)}{(l+2)!} \\ q_c^{(M-l)} = \left(\sum_{i=0}^{l-2} \frac{h^i}{2^i\, i!} q_k^{(i+M-l)} \right) + \frac{h^{l-1}\left(3q_{k+1}^{(M-1)} + (2l^2+4l-3) q_k^{(M-1)}\right)}{2^l\, l!\, (l+2)} + \frac{h^l\left((2l^2+2l-1) g_k - (2l+1) g_{k+1}\right)}{2^{l+1}\, (l+2)!} \end{cases}$

For the *Tz-M* and *HS-M* methods, we provide the general equation of $q_{k+1}^{(M-l)}$ (where l is meant to run up to M) but also the particular instances of this equation for $l = 1, 2$. The equation of $q_c^{(M-l)}$, and its instances for $l = 1, 2$, are also provided in the *HS-M* method (where, again, $l = 1, \ldots, M$). This arrangement allows us to realize that, within each family, the equations for the same l coincide for all orders

Table 2 Number of variables (n_v), equations (n_e), and degrees of freedom (n_{DOF}) in the two families of methods

Family	n_v	n_e	n_{DOF}
Trapezoidal	$(N+1)\,(n_x+n_u)$	$n_x\,N-n_b$	$n_x+(N+1)\,n_u-n_b$
Hermite–Simpson	$(2N+1)\,(n_x+n_u)$	$2n_x\,N-n_b$	$n_x+(2N+1)\,n_u-n_b$

Problem (6). If n_x and n_u are the dimensions of $\boldsymbol{x}$ and $\boldsymbol{u}$, and n_b is the number of boundary constraints in Eq. (6d), we obtain the values in Table 2. Note that for an M-th order ODE, the state $\boldsymbol{x}$ includes the configuration vector $\boldsymbol{q}$ and its derivatives, so that $n_x = Mn_q$, where n_q is the dimension of $\boldsymbol{q}$. The improved formulas, as compared to those of the *Tz*-1 and *HS*-1 methods, neither increase the problem size, nor reduce the freedom to find the optimal solution. Moreover, since the dynamic function must be evaluated at each collocation point, the number of evaluations is the same in all methods of a same family, so the new methods should not increase the cost of each iteration when solving the transcribed NLP problem. This point is also supported by the computational experiments that we present in Sect. 7.

6.2 Accuracy of the approximations

While the new methods introduced in this paper are explicitly designed to preserve the consistency between the configuration trajectory and its derivatives, a further question is how the application of these methods may affect the accuracy of the solution approximations and its rate of convergence as $h \to 0$. To answer this question, we draw upon the concept of order of accuracy (Betts, 2010), or simply order (Hairer et al., 2002) of a collocation method, which, in turn, relies on the definition of local error of an approximation.

The local error ϵ_k of a collocation method at interval k is defined as the difference between the computed value q_{k+1} and the value for $t = t_{k+1}$ of the exact solution of the ODE, $\hat{q}(t)$, that passes through the computed point q_k. If a collocation method approximates the solution with polynomials of degree d, we have for interval k:

$$q(t) = q_k + a_1\tau + \frac{a_2}{2}\tau^2 + \cdots + \frac{a_d}{d!}\tau^d,$$

and the computed value q_{k+1} is obtained by setting $t = t_k + h$, which means setting $\tau = h$:

$$q_{k+1} = q_k + a_1 h + \frac{a_2}{2}h^2 + \cdots + \frac{a_d}{d!}h^d. \tag{63}$$

On the other hand, the Taylor expansion of the exact solution $\hat{q}(t)$ that passes through the computed point q_k is

$$\hat{q}(t_k+t) = q_k + \dot{\hat{q}}(t_k)\,\tau + \frac{\ddot{\hat{q}}(t_k)}{2}\tau^2 + \cdots + \frac{\hat{q}^{(d)}(t_k)}{d!}\tau^d + \mathcal{O}(\tau^{d+1}),$$

and evaluating for $\tau = h$ we have:

$$\hat{q}(t_k+h) = q_k + \dot{\hat{q}}(t_k)\,h + \frac{\ddot{\hat{q}}(t_k)}{2}h^2 + \cdots + \frac{\hat{q}^{(d)}(t_k)}{d!}h^d + \mathcal{O}(h^{d+1}), \tag{64}$$

thus, the local error ϵ_k is given by the difference of the two Taylor expansions (63) and (64):

$$\epsilon_k = \left(a_1 - \dot{\hat{q}}(t_k)\right)h + \frac{a_2 - \ddot{\hat{q}}(t_k)}{2}h^2 + \ldots + \frac{a_d - \hat{q}^{(d)}(t_k)}{d!}h^d + \mathcal{O}(h^{d+1}). \tag{65}$$

A collocation method is said to have order of accuracy p if the sum of the first p terms of (65) is zero. Note that this does not imply that each term vanishes by itself: when h takes a specific numerical value, different non-null terms of the sum may add to zero. In the hypothetical case that the exact solution $\hat{q}(t_k+t)$ was a polynomial of degree p, a method of order p would have no local error. For this reason, the order of accuracy is also called the degree of exactness (Dahlquist and Björck, 2008), and an equivalent definition for it is that a collocation method has order of accuracy p if it is exact for all polynomials of degree $\leq p$.

In the limit, when $h \to 0$, the error of the approximation will converge to zero. The rate of this convergence is an important property of a method, and is directly given by its order. If a method has order p, the lower power of h appearing in ϵ_k is $p+1$, so that, when $h \to 0$, the local error decreases as h^{p+1}:

$$\epsilon_k = \mathcal{O}(h^{p+1}).$$

In all collocation methods discussed here, the interpolating polynomial used in each interval has degree $d = M + s - 1$, where M is the order of the ODE and s is the number of collocation points of each interval. This value d

ensures the unique determination of the $d+1$ polynomial coefficients given the s collocation constrains and the M initial conditions. In the event that the exact solution happens to be a polynomial $\hat{q}(t)$ of degree d, it must necessarily coincide with the interpolating polynomial, and q_{k+1} will coincide with the exact value $\hat{q}(t_k+h)$. This shows that the order of accuracy of any method is at least $p = d = M+s-1$, so the orders of Tz-1 and HS-1 are at least 2 and 3, respectively, while the orders of Tz-2 and HS-2 are at least 3 and 4. However, these lower bounds can be surpassed in some cases. For example, the HS-1 method is known to have order 4 (Hairer et al., 2002), while its corresponding lower bound is 3. This is because it takes advantage of a special property of a family of polynomials of fourth degree. It can be proved that any fourth degree polynomial satisfying

$$\hat{q}(t_k) = q_k, \tag{66a}$$
$$\dot{\hat{q}}(t_k) = \dot{q}_k, \tag{66b}$$
$$\dot{\hat{q}}(t_k + h/2) = \dot{q}_c, \tag{66c}$$
$$\dot{\hat{q}}(t_k + h) = \dot{q}_{k+1} \tag{66d}$$

takes always the same value $\hat{q}(t_k+h) = q_{k+1}$. Since the only third degree polynomial satisfying these same conditions is a particular case of this family, it satisfies $q(t_k+h) = q_{k+1}$, so its order of accuracy is 4.

Even if the HS-2 method does not benefit from a similar property, it is granted that its order is at least as large as that of HS-1, i.e., 4.

So, we can say that the order of accuracy of the presented methods for second order systems is equal or higher than that of the corresponding methods for first order systems. In general, for the same number of collocation points, a method for Mth order systems has this lower bound $M-1$ units higher than the corresponding method for first order systems.

6.3 Consistency errors

The order of accuracy of a method is useful, but it only provides hints on how the local errors ϵ_k converge to zero in terms of h. For a particular problem, obtaining the local errors of the computed splines $\boldsymbol{q}(t)$ and $\boldsymbol{u}(t)$ is seldom possible, as this requires knowing the actual solutions $\hat{\boldsymbol{q}}(t)$ and $\hat{\boldsymbol{u}}(t)$, which are rarely available. For this reason, to see the extent to which $\boldsymbol{q}(t)$ and $\boldsymbol{u}(t)$ are consistent with the system dynamics, we will compute the residual of Eq. (5),

$$\boldsymbol{\varepsilon}(t) = \boldsymbol{q}^{(M)}(t) - \boldsymbol{g}(t), \tag{67}$$

where $\boldsymbol{g}(t) = \boldsymbol{g}\left(\boldsymbol{q}(t), \dot{\boldsymbol{q}}(t), \ldots, \boldsymbol{q}^{(M-1)}(t), \boldsymbol{u}(t), t\right)$. Some authors, like Kelly (2017) or Betts (2010), refer to $\boldsymbol{\varepsilon}(t)$ as the "error in the differential equations", though they restrict their attention to the case $M=1$.

In those situations in which Eq. (5) has been discretized via Tz-1 and HS-1, we can define additional residuals to assess whether the obtained trajectories for the velocity, acceleration, and remaining components of the state, match those of the corresponding derivatives of the configuration. To define these residuals, recall from (7) that when a 1st order ODE encodes an Mth order one, the computed trajectory for the state takes the form

$$\boldsymbol{x}(t) = (\boldsymbol{x}_1(t), \boldsymbol{x}_2(t), \ldots, \boldsymbol{x}_M(t)), \tag{68}$$

where $\boldsymbol{x}_1(t), \boldsymbol{x}_2(t), \ldots, \boldsymbol{x}_M(t)$ approximate the configuration trajectory and its first and higher order derivatives, respectively. For the trajectories $\boldsymbol{x}_i(t)$ to be compatible among themselves, therefore, they should verify

$$\begin{aligned} \dot{\boldsymbol{x}}_1(t) &= \boldsymbol{x}_2(t), \\ \ddot{\boldsymbol{x}}_1(t) &= \boldsymbol{x}_3(t), \\ &\vdots \\ \boldsymbol{x}_1^{(M-1)}(t) &= \boldsymbol{x}_M(t), \end{aligned} \tag{69}$$

which leads us to computing

$$\boldsymbol{\varepsilon}^{[r]}(t) = \boldsymbol{x}_1^{(r)}(t) - \boldsymbol{x}_{r+1}(t), \tag{70}$$

for $r = 1, \ldots, M-1$ to verify their consistency.

In what follows, we will refer to $\boldsymbol{\varepsilon}(t)$ and $\boldsymbol{\varepsilon}^{[r]}(t)$ as the dynamics error and the rth order compatibility error, respectively, and we will use them to compare the presented methods in illustrative situations.

When reporting our results, we will sometimes use $\varepsilon_{q_i}(t)$ and $\varepsilon_{q_i}^{[r]}(t)$ to refer to the dynamics error and rth order compatibility error for the q_i component of $\boldsymbol{q}$. However, when all components of $\boldsymbol{q}$ have the same units, we will provide the values of the joint errors

$$\varepsilon(t) = |\varepsilon_{q_1}(t)| + \cdots + |\varepsilon_{q_{n_q}}(t)|, \tag{71}$$
$$\varepsilon^{[r]}(t) = |\varepsilon_{q_1}^{[r]}(t)| + \cdots + |\varepsilon_{q_{n_q}}^{[r]}(t)|. \tag{72}$$

Finally, when an error function needs to be summarized in just one number, we will compute the integral of its absolute value over $[0, t_f]$. Such a quantity will be denoted by prepending a small integral symbol to the error in consideration. Thus, for example, "$\int \varepsilon_{q_i}$" will be a shorthand for

$$\int_0^{t_f} |\varepsilon_{q_i}(t)|\, dt. \tag{73}$$

7 Test cases

The performance of all methods is next evaluated and compared using three trajectory optimization problems shown in Fig. 2. We refer to them as the cart-pole, bipedal walking, and ball throwing problems, respectively. The first two problems are solved and documented in detail by Kelly (2017), and thus serve to compare our results with those in the literature. The third problem is proposed by the authors to illustrate the methods on a widely-used robot with a complex dynamics. The cart-pole problem is also used to exemplify a situation in which a third-order ODE arises, which calls for the application of *Tz*-3 or *HS*-3.

To compare the methods, we have implemented them in Python, using CasADi to solve the constrained optimization problems that result (Andersson et al., 2019). CasADi provides the necessary means to formulate such problems and to compute the gradients and Hessians of the transcribed equations using automatic differentiation. These are necessary to solve the optimization problems, a task for which we rely on the interior-point solver IPOPT (Wächter and Biegler, 2006) in conjunction with the linear solver MUMPS (Amestoy et al., 2001). The whole implementation can be downloaded from https://github.com/AunSiro/optibot, but the reader can also reproduce the results for the cart-pole and bipedal walking problems through interactive Jupyter notebooks online (Moreno-Martín, 2023a, b, c). The execution times we report have been obtained with a single-thread implementation running on an iMac computer with an Intel i7, 8-core 10th generation processor at 3.8 GHz. In all cases we have set the "desired" and "acceptable" tolerances of IPOPT to 10^{-16} and 10^{-6}, respectively (The IPOPT Team, 2023).

7.1 The cart-pole swing-up problem

The cart-pole system comprises a cart that travels along a horizontal track and a pendulum that hangs freely from the cart. A motor drives the cart forward and backward along the track. Starting with the pendulum hanging below the cart at rest at a given position, the goal is to reach a final configuration in a given time t_f, with the pendulum stabilized at a point of inverted balance and the cart staying at rest at a distance d from the initial position. The cost to be minimized is

$$\int_0^{t_f} u^2(t)dt, \tag{74}$$

where u is the force applied to the cart, and we adopt the same dynamics equations and problem parameters as in Kelly (2017). An animation of the solution obtained with *HS*-2 and $N = 25$ can be seen in https://youtu.be/M0ivg_8s-I8.

Table 3 Performance data for the cart-pole problem

	N	T_c (s)	$\int \varepsilon_{q_1}^{[1]}$ (m)	$\int \varepsilon_{q_2}^{[1]}$ (rad)	$\int \varepsilon_{q_1}$ (m/s)	$\int \varepsilon_{q_2}$ (rad/s)
Tz-1	50	0.038	0.0066	0.0167	0.504	1.281
Tz-2	50	0.038	0	0	0.052	0.170
HS-1	25	0.045	0.0014	0.0043	0.113	0.338
HS-2	25	0.044	0	0	0.016	0.052

Figure 3 compares the compatibility and dynamics errors obtained by the methods for the variables q_1 and q_2 shown in Fig. 2. The number N of intervals used in the comparison is 50 for the trapezoidal scheme, and 25 for the Hermite–Simpson one. This yields a fair comparison, as then the number of collocation points, variables, and degrees of freedom are the same in all NLP problems (cf. Table 2). The plots of the compatibility errors $\varepsilon_{q_i}^{[1]}(t)$, in the first and third rows of Fig. 3, confirm that *Tz*-1 and *HS*-1 present a non-negligible value for these errors, while in *Tz*-2 and *HS*-2 these errors are exactly zero as expected.

The plots in the second and fourth rows of Fig. 3 clearly show a discontinuity at the knot points of the dynamics error $\varepsilon_{q_i}(t)$ for *Tz*-1 and *HS*-1, reflecting the discontinuity of $\ddot{\boldsymbol{q}}(t)$ at these points. In contrast, for *Tz*-2 and *HS*-2, the error functions are continuous and vanish at the collocation points, evidencing that, as anticipated in Sect. 3.4, the system dynamics is exactly satisfied at all collocation points for the new methods, but not for the conventional ones.

The figure also shows the dramatic reductions of $\varepsilon_{q_i}(t)$ for the new methods when compared with the corresponding *Tz*-1 and *HS*-1 ones. The numerical evaluation of the results appears in Table 3, which provides the computation times T_c and the integral errors $\int \varepsilon_{q_i}^{[1]}$ and $\int \varepsilon_{q_i}$ for this problem. It can be seen that the values of $\int \varepsilon_{q_i}$ are about one order of magnitude lower for *Tz*-2 and *HS*-2 than for their counterparts *Tz*-1 and *HS*-1, despite using a very similar computation time. It is interesting to see that the errors $\int \varepsilon_{q_i}$ achieved by *Tz*-2 are about a half of those of *HS*-1 for the same number of collocation points. The comparison is relevant since both methods use polynomials of the same degree to approximate $q_i(t)$.

7.2 The bipedal walking problem

We next apply the methods to optimize a periodic gait for the planar biped robot shown in Fig. 2. The robot involves five links pairwise connected with revolute joints, forming two legs and a torso. All joints are powered by torque motors, with the exception of the ankle joint, which is passive. Like the cart pole system, this robot is underactuated, but it is substantially more complex. The system is commonly used as a testbed when studying bipedal walking (Westervelt et

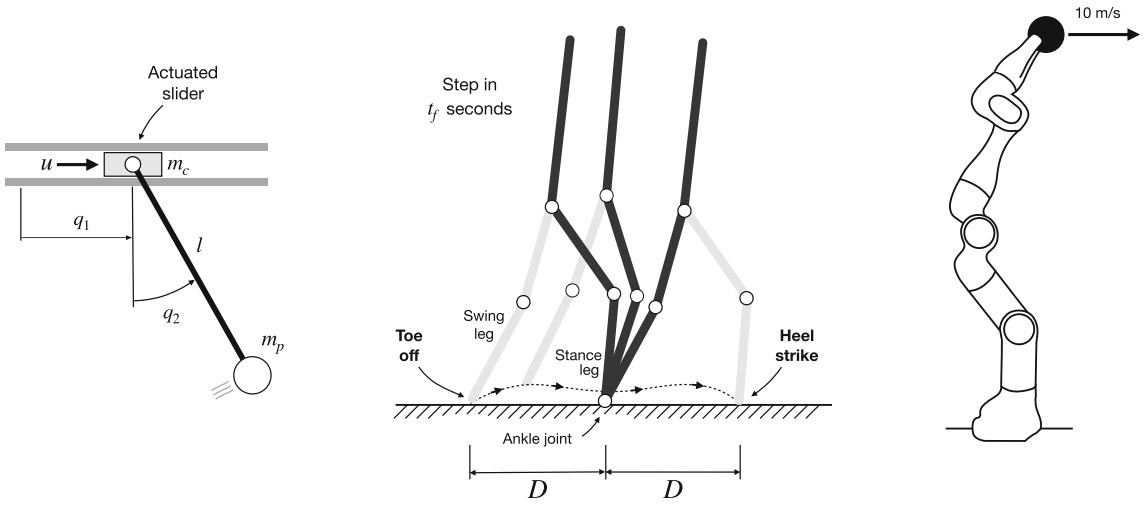

Fig. 2 Benchmark problems. Left: A cart-pole system that has to perform a swing-up motion. Center: a walking biped whose periodic gait must be optimized (the three snapshots illustrate the motion that occurs between the *toe off* and *heel strike* events defining a period of the gait). Right: A 7R Panda robot that has to pick a ball at the shown configuration, and throw it from the same configuration at 10m/s horizontally

al., 2003; Yang et al., 2009; Park et al., 2012; Saglam and Byl, 2014).

For this example we use the dynamic model given by Kelly (2017), which matches the one in Westervelt et al. (2003) with parameters corresponding to the RABBIT prototype (Chevallereau et al., 2003). We assume the robot is left-right symmetric, so we can search for a periodic gait using a single step, as opposed to a stride, which involves two steps. This means that the state and torque trajectories will be the same on each successive step.

As in Kelly (2017), we define $\boldsymbol{q}$ as the vector that contains the absolute angles of all links relative to ground, while $\boldsymbol{u}$ encompasses all motor torques. Also as in Kelly (2017), and similarly to the cart-pole problem, our goal is to find state and action trajectories $\boldsymbol{x}(t)$ and $\boldsymbol{u}(t)$ that define an optimal gait under the cost

$$\int_0^{t_f} \boldsymbol{u}(t)^\mathsf{T} \boldsymbol{u}(t)\, dt. \tag{75}$$

Several constraints are added to ensure a feasible gait. First of all, we require the gait to be periodic, so

$$\boldsymbol{x}_0 = \boldsymbol{f}_H(\boldsymbol{x}_f), \tag{76}$$

where $\boldsymbol{x}_0$ and $\boldsymbol{x}_f$ are the initial and final states of the robot, and $\boldsymbol{f}_H$ is the heel-strike map. The states $\boldsymbol{x}_0$ and $\boldsymbol{x}_f$ are unknown a priori, but constrained by (76), which is the particular form of the boundary constraint (6d) in this case. To construct $\boldsymbol{f}_H$ it is assumed that, at heel strike, an impulsive collision occurs that changes the joint velocities but not their angles, and that, as soon as the leading foot impacts the ground, the trailing foot loses contact with it. The collision conserves angular momentum but introduces an instantaneous drop of kinetic energy in the system (Kelly, 2017). Next, we require the robot to march at a certain speed, which is achieved by setting the final time of the period to $t_f = 0.7$ s, and the length D in Fig. 2 to 0.5m. We also constrain the vertical velocity component of the trailing foot to be positive at $t = 0$, and negative when it touches the ground for $t = t_f$. Finally, we require the swing foot to be above the ground at all times. An animation of the solution we obtain can be seen in https://youtu.be/dtS-WbESiW0.

Figure 4 shows the dynamics errors $\varepsilon(t)$ for the different collocation methods. As before, the number of intervals used in the trapezoidal cases is twice that used in the Hermite–Simpson ones so as to have an identical number of collocation points and achieve balanced comparisons. The results are qualitatively similar to those of the cart-pole, though here the error diminution obtained by the new methods is even more accentuated. As we can see in Table 4, the integral dynamics error $\int \varepsilon$ of *HS*-2 improves in more than one order of magnitude that of *HS*-1, and still using a similar computation time. In the case of *Tz*-2, its improvement over *Tz*-1 is still higher, reaching a reduction factor near 66, and using a slightly lower computation time.

7.3 A ball throwing problem

As a third example, we apply the methods to compute an object-throwing trajectory for a 7R Panda manipulator. The robot is initially at rest, grasping a ball with its gripper, and its task is to throw the ball from the same configuration after 1 second, with an horizontal velocity of 10 m/s. Since the dynamic model is complex in this case, we rely on the advanced dynamics engine Pinocchio (Carpentier et

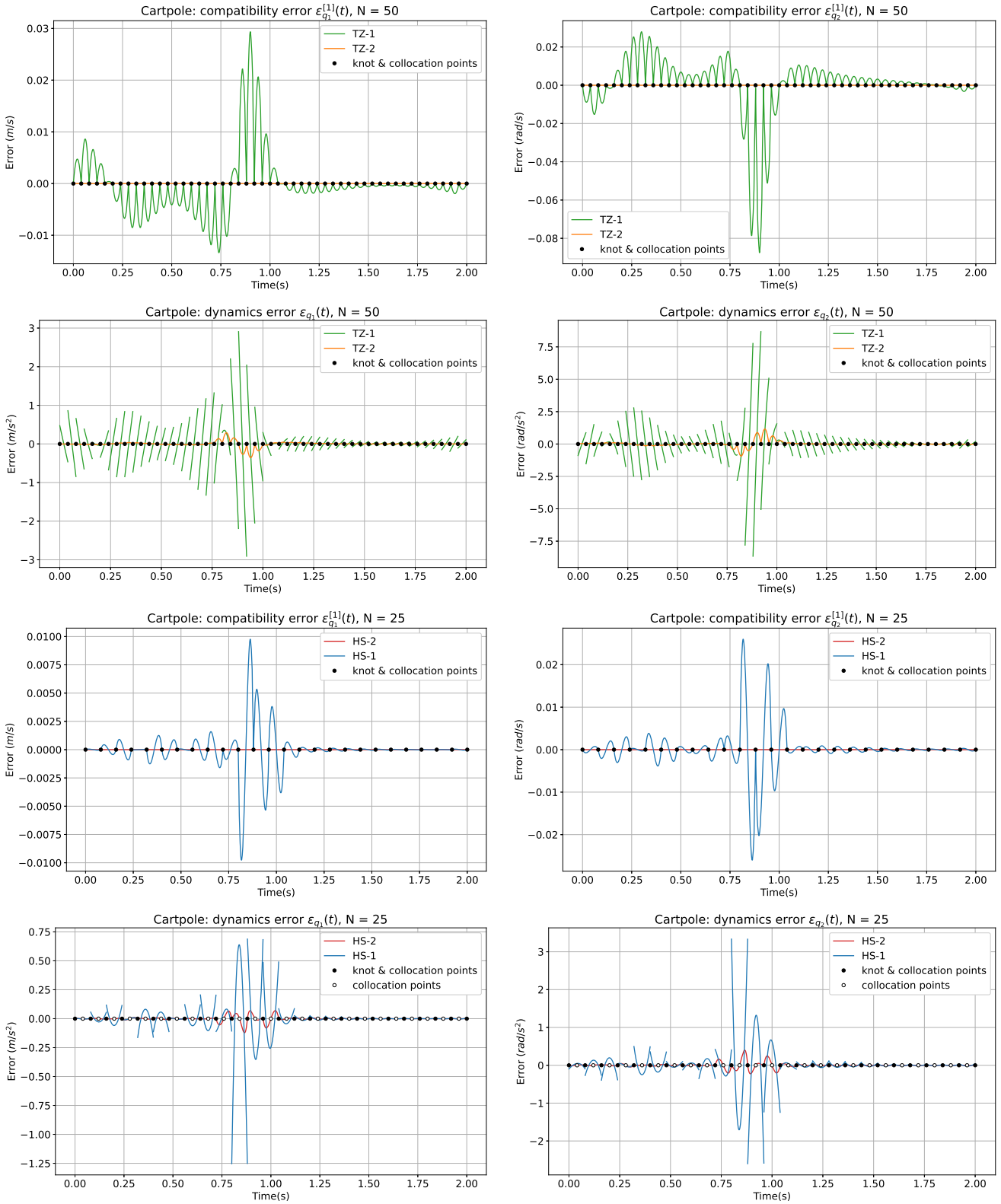

Fig. 3 Cart-pole problem: Plots of the compatibility errors $\varepsilon_{q_i}^{[1]}(t)$ and the dynamics errors $\varepsilon_{q_i}(t)$ for the coordinates q_1 (left column) and q_2 (right column), using the trapezoidal and Hermite–Simpson methods.

To compare the results with those of Kelly (2017), note that this author actually provides the plots of $-\varepsilon_{q_i}^{[1]}(t)$ for *HS*-1

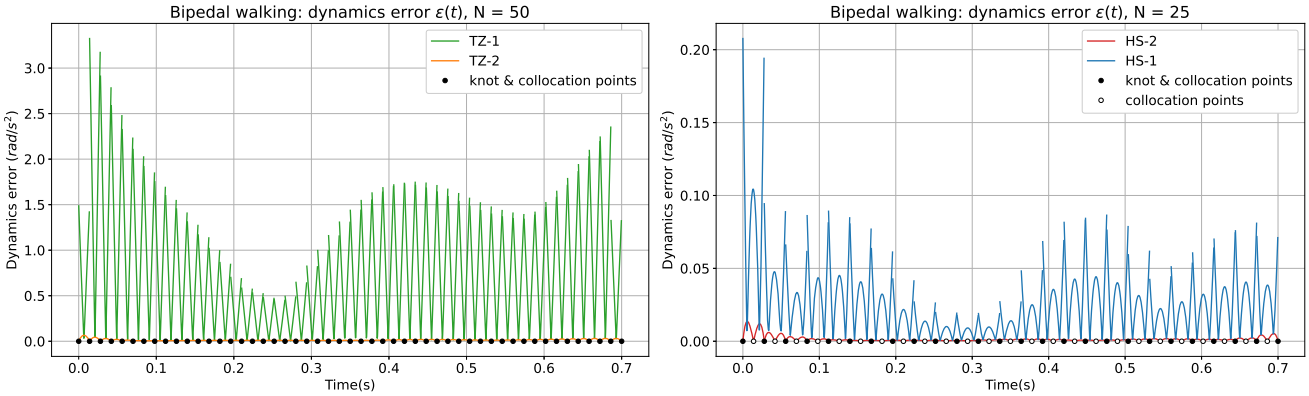

Fig. 4 Dynamics errors $\varepsilon(t)$ for the bipedal walking problem

Table 4 Performance data for the bipedal walking problem ($\int \varepsilon$ is the integral of $\varepsilon(t)$ in Fig. 4)

Method	N	T_c (s)	$\int \varepsilon^{[1]}$ (rad)	$\int \varepsilon$ (rad/s)
Tz-1	50	0.117	0.0025	0.5328
Tz-2	50	0.111	0	0.0081
HS-1	25	0.110	8.2×10^{-5}	0.0182
HS-2	25	0.113	0	0.0011

Table 5 Performance data for the ball throwing problem ($\int \varepsilon$ is the integral of $\varepsilon(t)$ in Fig. 6)

	N	T_c (s)	$\int \varepsilon^{[1]}$ (rad)	$\int \varepsilon$ (rad/s)
Tz-1	100	6.362	0.0189	6.1229
Tz-2	100	6.064	0	1.0644
HS-1	50	6.269	0.0034	1.2388
HS-2	50	6.268	0	0.4275

Table 6 Performance data for the cart-pole problem with 3rd order dynamics

	N	T_c (s)	$\int \varepsilon_{q_1}^{[1]}$ (m)	$\int \varepsilon_{q_2}^{[1]}$ (rad)	$\int \varepsilon_{q_1}$ (m/s^2)	$\int \varepsilon_{q_1}$ (rad/s^2)
Tz-1	50	0.07	0.0060	0.0169	45.926	127.614
Tz-3	50	0.06	0	0	1.075	3.174
HS-1	25	0.06	0.0011	0.0033	8.614	26.451
HS-3	25	0.06	0	0	0.497	1.616

al., 2019) to compute the f_k and g_k values in the collocation formulas. This engine implements the forward dynamics algorithms by Featherstone (2008) in C++, which speeds up the computations considerably. As for the cost function, we use

$$\int_0^{t_f} \left[\boldsymbol{u}(t)^\mathsf{T}\boldsymbol{u}(t) + K_a \ddot{\boldsymbol{q}}(t)^\mathsf{T}\ddot{\boldsymbol{q}}(t) \right] dt, \tag{77}$$

where K_a is a small value that we fixed to 0.1. While the first term in the integrand penalizes large control torques, the second helps to achieve smoother trajectories for the state.

To compare the methods on an equal footing, in all runs we feed the NLP solver with an initial guess that allows the convergence to a similar solution. This guess is obtained using the *Tz*-1 method with $N = 25$, initialized with $u_k = 0$ for all k, and using x_k values that interpolate the initial state, a guessed state for $t = 0.5s$, and the final state. The trajectory obtained via *Tz*-1 is then used to warm start all methods in the comparisons. As a reference, Fig. 5 shows the trajectory obtained using *HS*-2 and $N = 100$. Note how the robot performs a circular motion, exploiting gravity to gain momentum so as to get back to the launch point with the required speed.

Figure 6 compares the dynamics error $\varepsilon(t)$ for the trapezoidal and Hermite–Simpson methods (left and right plots, respectively). As in the previous problems, the new methods notably outperform the conventional ones in terms of this error. The integral errors $\int \varepsilon$ corresponding to these figures can be seen in Table 5, together with those of $\int \varepsilon^{[1]}$ and T_c, confirming similar trends as in the earlier problems.

7.4 A third order example

As an example of a higher order system, we take again the problem of the cart-pole of Sect. 7.1 and impose the additional requirement of the control function $u(t)$ to be smooth, i.e., not only continuous as in the standard approach, but with continuous derivative. A smooth trajectory $u(t)$ will give rise to smooth accelerations and continuous jerks for the configuration variables, all of which are desirable properties in many robotics applications. To achieve these properties, we include the applied force u as a state variable and define a new control variable w as the temporal derivative of u by imposing $w(t) = \dot{u}(t)$. Thus, the continuity of $w(t)$ will

Fig. 5 Trajectory obtained for the ball throwing task. The robot, initially at rest, progressively gains momentum assisted by gravity so as to get back to the initial configuration to throw the ball at the required speed. An animation of the trajectory can be seen in https://youtu.be/NsEv6JrSN8c

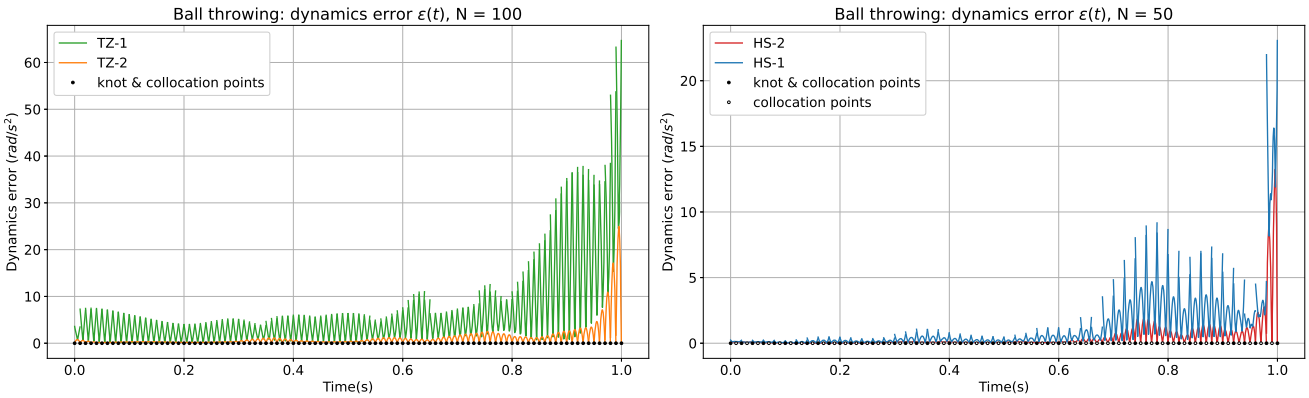

Fig. 6 Dynamics errors $\varepsilon(t)$ for the ball throwing problem

grant the smoothness of $u(t)$. By differentiating the dynamics equations with respect to time, we obtain the differential equations involving the new control variable w in the form:

$$\dddot{\boldsymbol{q}}(t) = \boldsymbol{g}(\boldsymbol{q}(t), \dot{\boldsymbol{q}}(t), \ddot{\boldsymbol{q}}(t), u(t), w(t), t). \tag{78}$$

The resulting system is a 3rd order ODE which will provide the same solution as the original 2nd order one except for an arbitrary choice of the initial state. By imposing the condition that the initial state satisfies the original 2nd order ODE, both systems become completely equivalent.

Note that the use of (78) now allows us to apply Tz-3 and HS-3, both of which ensure the continuity of the obtained trajectory for $\dddot{\boldsymbol{q}}(t)$, and thus the smoothness of $\ddot{\boldsymbol{q}}(t)$ as desired. In other words, whereas the splines $\boldsymbol{q}(t)$ computed with Tz-1 and HS-1 will be just once differentiable, those obtained with Tz-3 and HS-3 will be three times differentiable. To avoid high rates of change in the applied force, moreover, in this example we minimize

$$\int_0^{t_f} w(t)^2 \, dt.$$

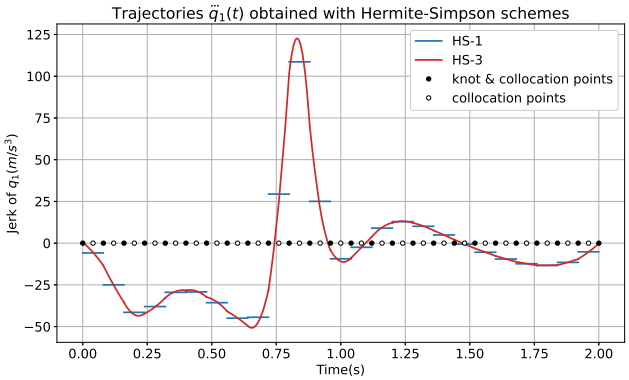

Fig. 7 Comparison of the jerk trajectories $\dddot{q}_1(t)$ obtained by HS-1 and HS-3 in the cart-pole problem with 3rd order dynamics

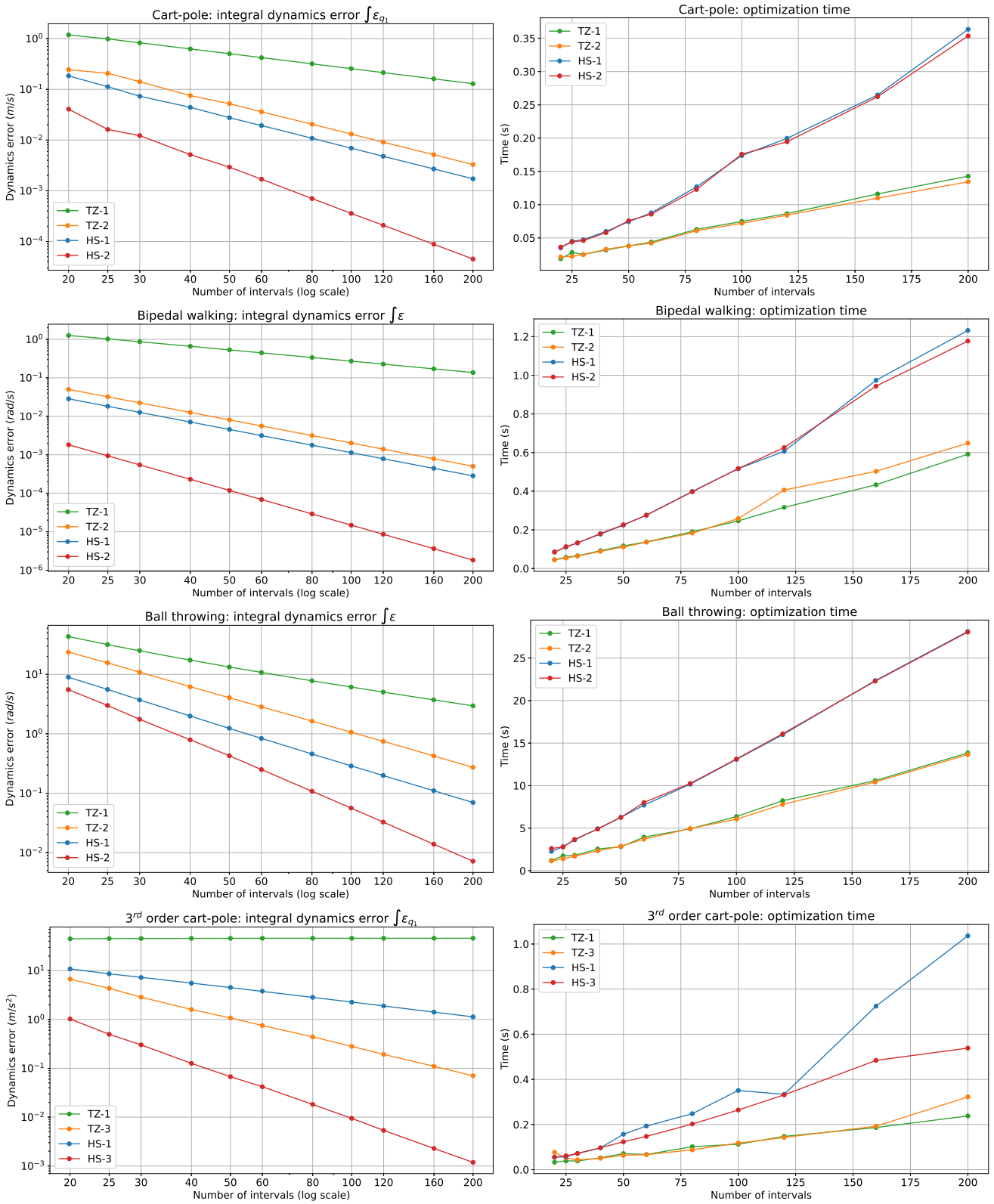

Fig. 8 Dynamics error (left) and optimization time (right) for the four test problems, as N is increased

Table 6 shows the computation times T_c and the errors we obtain for Tz-1 and HS-1, compared with those for Tz-3 and HS-3. As we see, when Tz-3 and HS-3 are used, the dynamics errors are reduced in more than one order of magnitude with respect to those of Tz-1 and HS-1, respectively, while the values of T_c remain similar.

To assess the difference in the continuity of the trajectories, Fig. 7 shows the curves obtained for the third derivative of the position, $\dddot{q}_1(t)$, both for HS-1 and HS-3. As expected, while HS-1 gives piece-wise constant and discontinuous values for $\dddot{q}_1(t)$, HS-3 provides a high-quality trajectory with continuous jerk.

7.5 Performance scaling with *N*

To evaluate the performance of the methods when the number N of intervals increases, a series of experiments have been conducted by progressively rising N from 20 to 200. Each experiment has been launched several times and the average of the integral dynamics errors and computation times are shown in Fig. 8 as a function of N. For the bipedal walking and ball throwing problems we plot $\int \varepsilon$. For the cart-pole problem we do not use $\int \varepsilon$ as the coordinates q_1 and q_2 have different units. Instead we provide only the plot of $\int \varepsilon_{q_1}$, as the one of $\int \varepsilon_{q_2}$ is very similar.

In all test problems, the best results for the dynamics error (shown on the left column of Fig. 8 using logarithmic scale on both axes) are those of HS-2 and HS-3, which, in many cases, improve the results of HS-1 in about one order of magnitude, or even more, and the improvement rate tends to increase with the number N of intervals. The same behavior is observed for Tz-2 and Tz-3 with respect to Tz-1. Interestingly, in all cases the performance of Tz-2 produces, for the same number of intervals N, only about twice the error of HS-1, and this rate is kept rather constant with N. However, a more balanced comparison would be to look at experiments with equal number of collocation points, what means to compare each N value of HS-1 with the $2N$ value of Tz-2. A close look at the plots will convince the reader that this comparison gives equal or better results for Tz-2 in all cases. Noticeably, as evidenced in the last row of Fig. 8, the results for Tz-3 outperform those for HS-1 even for the same number of intervals.

The plots on the right hand side of Fig. 8 show the growth of the computation times with the number N of intervals. The plots consistently show that the difference in computation time between a method for first order systems and the corresponding method for second or third order systems is not relevant. In all cases, the growth is nearly linear in N, but the increase rate is higher for the HS methods than for the Tz ones. Despite the different complexity of the four problems analyzed, reflected in the different time scales involved, in all cases the increase rate of the HS methods is nearly twice that of the Tz methods. In other words, the increasing rate is very similar for all methods when comparing the computation times for the same number of collocation points.

8 Conclusions

Trapezoidal and Hermite–Simpson collocation methods are very popular in the robotics community. However, they are conceived for dynamical systems of first order, while the dynamics of the systems found in robotics are often Mth order, with $M > 1$. The transcription of an Mth order ODE as a first order one has the unexpected effect that the dynamic equations are not actually imposed at the collocation points. Properly imposing the Mth order constraints at the same such points as in the original algorithms requires increasing the degree of the polynomials approximating the configuration trajectory, while keeping the implied degrees for its time derivatives. This is achieved with the methods we propose, which grant the functional consistency between the trajectories of all the state coordinates, not only at the collocation points, but also along the whole time horizon. Using benchmark problems of increasing complexity, we have also shown that the new methods provide trajectories with a much smaller dynamic error than those of conventional methods, despite they require a comparable amount of computation time. This implies that the obtained trajectories will be more compliant with the system dynamics, so they should be easier to track with a feedback controller. Moreover, the trajectories of the new methods are M times differentiable, so in addition to enjoying smooth velocities, their accelerations will be continuous, or even smooth if $M \geq 3$, which are very desirable properties from a control perspective.

Points that deserve further attention are the extension of these ideas to pseudospectral collocation methods, which we initially explored for $M = 2$ in Moreno-Martín et al. (2022), or generalizations to deal with constrained multibody systems (Posa et al., 2016; Bordalba et al., 2023), or systems involving SO(3) (Manara et al., 2017).

Author Contributions Mr. Moreno-Martín carried out the implementations and the computational experiments, and all authors developed the theory and prepared the paper.

Funding Open Access funding provided thanks to the CRUE-CSIC agreement with Springer Nature. This work has been partially funded by Agencia Estatal de Investigación under project *Kinodyn+*, with reference PID2020-117509GB-I00/AEI /10.13039/50110001103, and by a Ph.D. contract supporting the first author, with reference PRE2018-085582.

Declarations

Conflict of interest The authors declare no competing interests.

References

Amestoy, P. R., Duff, I. S., L'Excellent, J.-Y., & Koster, J. (2001). A fully asynchronous multifrontal solver using distributed dynamic scheduling. *SIAM Journal on Matrix Analysis and Applications, 23*(1), 15–41. https://doi.org/10.1137/S0895479899358194

Andersson, J. A. E., Gillis, J., Horn, G., Rawlings, J. B., & Diehl, M. (2019). CasADi: A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation, 11*(1), 1–36. https://doi.org/10.1007/s12532-018-0139-4

Becerra, V. M. (2010). Solving complex optimal control problems at no cost with PSOPT. In *2010 IEEE International Symposium on Computer-aided Control System Design* (pp. 1391–1396). https://doi.org/10.1109/CACSD.2010.5612676.

Berscheid, L. & Kröger, T. (2021). Jerk-limited real-time trajectory generation with arbitrary target states. In *Robotics: Science and Systems*. https://doi.org/10.15607/RSS.2021.XVII.015

Betts, J. T. (2010). *Practical methods for optimal control and estimation using nonlinear programming*. Philadelphia: SIAM. https://doi.org/10.1137/1.9780898718577

Bordalba, R., Schoels, T., Ros, L., Porta, J. M., & Diehl, M. (2023). Direct collocation methods for trajectory optimization in constrained robotic systems. *IEEE Transactions on Robotics, 39*(1), 183–202. https://doi.org/10.1109/TRO.2022.3193776

Carpentier, J., Saurel, G., Buondonno, G., Mirabel, J., Lamiraux, F., Stasse, O., & Mansard, N. (2019). The Pinocchio C++ library: A fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives. In *IEEE International Symposium on System Integrations (SII)*. https://doi.org/10.1109/SII.2019.8700380.

Chevallereau, C., Abba, G., Aoustin, Y., Plestan, F., Westervelt, E. R., De Wit, C. C., & Grizzle, J. (2003). RABBIT: A testbed for advanced control theory. *IEEE Control Systems Magazine, 23*(5), 57–79. https://doi.org/10.1109/MCS.2003.1234651

Constantinescu, D., & Croft, E. A. (2000). Smooth and time-optimal trajectory planning for industrial manipulators along specified paths. *Journal of Robotic Systems, 17*(5), 233–249. https://doi.org/10.1115/IMECE1999-0065

Conway, B. A., & Paris, S. W. (2010). Spacecraft trajectory optimization using direct transcription and nonlinear programming. In B. Conway (Ed.), *Spacecraft trajectory optimization* (pp. 37–78). Cambridge: Cambridge University Press. https://doi.org/10.1017/cbo9780511778025.004

Dahlquist, G., & Björck, A. (2008). *Numerical Methods in Scientific Computing, Volume I*. Society for Industrial and Applied Mathematics.

De Luca, A., & Book, W. J. (2016). Robots with flexible elements. In *Springer handbook of robotics* (pp. 243–282. Springer. https://doi.org/10.1007/978-3-319-32552-1_11.

Della Santina, C. (2020). Flexible Manipulators. In *Encyclopedia of robotics*. Springer. https://doi.org/10.1007/978-3-642-41610-1_182-1.

Featherstone, R. (2008). *Rigid body dynamics algorithms*. Berlin: Springer. https://doi.org/10.1007/978-1-4899-7560-7

Hairer, E., Wanner, G., & Lubich, C. (2002). *Geometric numerical integration*. Berlin: Springer. https://doi.org/10.1007/978-3-662-05018-7

Hairer, E., Wanner, G., & Nørsett, S. P. (1993). *Solving ordinary differential equations. I*. Berlin: Springer. https://doi.org/10.1007/978-3-540-78862-1

Hargraves, C. R., & Paris, S. W. (1987). Direct trajectory optimization using nonlinear programming and collocation. *Journal of Guidance, Control, and Dynamics, 10*(4), 338–342. https://doi.org/10.2514/3.20223

Hereid, A., Hubicki, C. M., Cousineau, E. A., & Ames, A. D. (2018). Dynamic humanoid locomotion: A scalable formulation for HZD gait optimization. *IEEE Transactions on Robotics, 34*(2), 370–387. https://doi.org/10.1109/TRO.2017.2783371

Kelly, M. (2017). An introduction to trajectory optimization: How to do your own direct collocation. *SIAM Review, 59*(4), 849–904. https://doi.org/10.1137/16M1062569

Macfarlane, S., & Croft, E. A. (2003). Jerk-bounded manipulator trajectory planning: Design for real-time applications. *IEEE Transactions on Robotics Automation, 19*(1), 42–52. https://doi.org/10.1109/TRA.2002.807548

Manara, S., Gabiccini, M., Artoni, A., & Diehl, M. (2017). On the integration of singularity-free representations of SO(3) for direct optimal control. *Nonlinear Dynamics, 90*, 1223–1241.

Moreno-Martín, S. (2023a). Online Jupyter notebook for cart-pole problem (with 3rd order ODE). https://mybinder.org/v2/gh/AunSiro/Second-Order-Schemes/HEAD?labpath=Cartpole-3rd-order-demo.ipynb.

Moreno-Martín, S. (2023b). Online Jupyter notebook for the cart-pole problem (standard version). https://mybinder.org/v2/gh/AunSiro/Second-Order-Schemes/HEAD?labpath=Cartpole-demo.ipynb.

Moreno-Martín, S. (2023c). Online Jupyter notebook for the bipedal walking problem. https://mybinder.org/v2/gh/AunSiro/Second-Order-Schemes/HEAD?labpath=Five-Link-Biped-demo.ipynb.

Moreno-Martín, S., Ros, L., & Celaya, E. (2022). A Legendre-Gauss pseudospectral collocation method for trajectory optimization in second order systems. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 13335–13340). https://doi.org/10.1109/IROS47612.2022.9981255.

Moreno-Martín, S., Ros, L., & Celaya, E. (2022). Collocation methods for second order systems. In *Robotics: Science and Systems*, New York. http://www.roboticsproceedings.org/rss18/p038.html.

Pardo, D., Möller, L., Neunert, M., Winkler, A. W., & Buchli, J. (2016). Evaluating direct transcription and nonlinear optimization methods for robot motion planning. *IEEE Robotics and Automa-*

tion Letters, 1(2), 946–953. https://doi.org/10.1109/LRA.2016.2527062

Park, H.-W., Sreenath, K., Ramezani, A., & Grizzle, J. W. (2012). Switching control design for accommodating large step-down disturbances in bipedal robot walking. In *2012 IEEE International Conference on Robotics and Automation* (pp. 45–50). IEEE. https://doi.org/10.1109/ICRA.2012.6225056.

Posa, M., Kuindersma, S., & Tedrake, R. (2016). Optimization and stabilization of trajectories for constrained dynamical systems. In *IEEE International Conference on Robotics and Automation* (pp. 1366–1373). https://doi.org/10.1109/ICRA.2016.7487270.

Saglam, C. O. & Byl, K. (2014). Robust policies via meshing for metastable rough terrain walking. In *Robotics: Science and Systems*. https://doi.org/10.15607/RSS.2014.X.049.

Tedrake, R. (2023). *Underactuated robotics: Algorithms for walking, running, swimming, flying, and manipulation (Course Notes for MIT 6.832).* MIT. Accessed 16 June 2023 from http://underactuated.mit.edu/.

The Drake Team (2023). Drake: Model-based design and verification for robotics. https://drake.mit.edu/.

The IPOPT Team (2023). IPOPT Documentation. Accessed 16 June 2023 from https://coin-or.github.io/Ipopt/OPTIONS.html.

Topputo, F., & Zhang, C. (2014). Survey of direct transcription for low-thrust space trajectory optimization with applications. *Abstract and Applied Analysis, 2014*, 1–15. https://doi.org/10.1155/2014/851720

Wächter, A., & Biegler, L. T. (2006). On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming, 106*(1), 25–57. https://doi.org/10.1007/s10107-004-0559-y

Westervelt, E. R., Grizzle, J. W., & Koditschek, D. E. (2003). Hybrid zero dynamics of planar biped walkers. *IEEE Transactions on Automatic Control, 48*(1), 42–56. https://doi.org/10.1109/TAC.2002.806653

Yang, T., Westervelt, E. R., Serrani, A., & Schmiedeler, J. P. (2009). A framework for the control of stable aperiodic walking in underactuated planar bipeds. *Autonomous Robots, 27*(3), 277–290. https://doi.org/10.1007/s10514-009-9126-y

Siro Moreno-Martín received the Aeronautics Engineering degree in 2015 from the Universidad Politécnica de Madrid (UPM), Spain. From 2016 to 2018 he worked as R&D Engineer at the Instituto Nacional de Técnica Aeroespacial (INTA). He is currently a PhD candidate at the Computational Robotics Group of the Institut de Robòtica i Informàtica Industrial (IRI, CSIC-UPC), Barcelona. His research interests include kinematics, dynamics, and control, with applications to robotics.

Lluís Ros received the Mechanical Engineering degree in 1992, and the Ph.D. degree (with honors) in Industrial Engineering in 2000, both from Universitat Politècnica de Catalunya (UPC). From 1993 to 1996 he worked with the Control of Resources Group of Institut de Cibernètica (Barcelona). He joined the Institut de Robòtica i Informàtica Industrial (Barcelona) in 1997, where he is an Associate Researcher of the Spanish National Research Council (CSIC) since 2004. He has been a visiting scholar at York University (Toronto), University of Tokyo (Tokyo), and the Laboratoire d'Analyse et Architecture des Systèmes (Toulouse). His current research interests include kinodynamic motion planning, trajectory optimization, and control, of general multibody systems.

Enric Celaya received the B.S. degree in Theoretical Physics from the Universitat de Barcelona in 1979 and the Ph.D. in Artificial Intelligence from de Universitat Politécnica de Catalunya (UPC) in 1992. Currently retired, he has hold a position of Scientific Researcher of the CSIC at the Institut de Robòtica i Informàtica Industrial (CSIC-UPC) in Barcelona. Along its professional career he has been working in the fields of Kinematics, legged robots, computer vision, reinforcement learning, and trajectory optimization.