Collocation Methods for Second Order Systems

Siro Moreno-Martín, Lluís Ros, and Enric Celaya
Institut de Robòtica i Informàtica Industrial (CSIC-UPC), 08028 Barcelona
Emails:{smorenom,ros,ecelaya}@iri.upc.edu

***Abstract*—Collocation methods for numerical optimal control commonly assume that the system dynamics is expressed as a first order ODE of the form $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t)$, where $\boldsymbol{x}$ is the state and $\boldsymbol{u}$ the control vector. However, in many systems in robotics, the dynamics adopts the second order form $\ddot{\boldsymbol{q}} = \boldsymbol{g}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{u}, t)$, where $\boldsymbol{q}$ is the configuration. To preserve the first order form, the usual procedure is to introduce the velocity variable $\boldsymbol{v} = \dot{\boldsymbol{q}}$ and define the state as $\boldsymbol{x} = (\boldsymbol{q}, \boldsymbol{v})$, where $\boldsymbol{q}$ and $\boldsymbol{v}$ are treated as independent in the collocation method. As a consequence, the resulting trajectories do not fulfill the mandatory relationship $\boldsymbol{v}(t) = \dot{\boldsymbol{q}}(t)$ for all times, and even violate $\ddot{\boldsymbol{q}} = \boldsymbol{g}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{u}, t)$ at the collocation points. This prevents the possibility of reaching a correct solution for the problem, and makes the trajectories less compliant with the system dynamics. In this paper we propose a formulation for the trapezoidal and Hermite-Simpson collocation methods that is able to deal with second order dynamics and grants the mutual consistency of the trajectories for $\boldsymbol{q}$ and $\boldsymbol{v}$ while ensuring $\ddot{\boldsymbol{q}} = \boldsymbol{g}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{u}, t)$ at the collocation points. As a result, we obtain trajectories with a much smaller dynamical error in similar computation times, so the robot will behave closer to what is predicted by the solution. We illustrate these points by way of examples, using well-established benchmark problems from the literature.**

I. Introduction

Direct collocation methods have proven to be powerful tools for solving optimal control problems in robotics [11, 22, 15]. Initially developed for aeronautics and astrodynamics applications [10, 8], 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, very often, are difficult to obtain accurately. Helpful tutorials and monographs like [11, 5], as well as open-source packages like [11, 2, 3, 24], are also contributing to the rapid dissemination of these methods.

Direct collocation methods involve the transcription of the continuous-time optimal control problem into a finite-dimensional nonlinear programming (NLP) problem [11]. 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 the NLP problem is solved for the discrete problem, a continuous solution is built with an interpolating polynomial that satisfies 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

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

where $\boldsymbol{x}$ and $\boldsymbol{u}$ are the state and control vectors [22]. However, in robotics, as in mechanics in general, the evolution of the system is often determined by a second order ODE of the form

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

where $\boldsymbol{q}$ is the configuration and $\dot{\boldsymbol{q}}$ is its time derivative. To apply a general collocation method, therefore, the usual procedure is to cast (2) into (1) by defining the state vector as $\boldsymbol{x} = (\boldsymbol{q}, \boldsymbol{v})$, where $\boldsymbol{v} = \dot{\boldsymbol{q}}$, so (2) can be written as

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

However, this raises a consistency issue. While the collocation method imposes (3) at the collocation points, it approximates $\boldsymbol{q}(t)$ and $\boldsymbol{v}(t)$ by polynomials of the same degree, so the functional relationship

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

is not granted over the continuous time horizon. 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) at these points, which contributes to increase the dynamic transcription error along the obtained trajectories [5]. This hinders the possibility to reach a correct solution since, even if the control function $\boldsymbol{u}(t)$ 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.

In this work, we present modified versions of the trapezoidal and Hermite-Simpson collocation methods specifically addressed to solve these issues for second order systems with the dynamics in (2). The new formulations grant that the collocation polynomials fulfill the condition in (4) while satisfying the dynamics in (2) at the collocation points. We call these methods second order, as opposed to classical ones that only guarantee (1) at the collocation points, which we call first order. We also use well-established benchmark problems to show that our second order methods reduce substantially the dynamic transcription 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 robot will follow the optimized trajectories more closely, and with less control effort, when they are tracked with the help of a feedback controller.

The rest of the paper is structured as follows. Section II formulates the optimal control to be solved and delimits the specific transcription problem that we face in this paper. To prepare the ground for later developments, Section III reviews the conventional trapezoidal and Hermite-Simpson collocation methods and pinpoints their limitations on transcribing second order differential equations. Improved versions of these methods are then developed in Section IV, and the performance of all methods is compared and discussed in detail in Section V with the help of examples. Finally, Section VI concludes the paper and enumerates a few points for future attention.

II. Problem formulation

Let $\boldsymbol{x} = (\boldsymbol{q}, \boldsymbol{v})$ be a tuple describing the robot state, where $\boldsymbol{q} \in \mathbb{R}^{n_q}$ gives the robot configuration and $\boldsymbol{v} = \dot{\boldsymbol{q}}$. We assume the robot dynamics is given by the ODE in (2), or equivalently by (1), where $\boldsymbol{u} \in \mathbb{R}^{n_u}$ is the control vector of motor forces and torques. Then, given an instantaneous cost function $L(\boldsymbol{x}(t), \boldsymbol{u}(t))$, a path constraint $\boldsymbol{h}(\boldsymbol{x}(t), \boldsymbol{u}(t)) \leq \boldsymbol{0}$, and a boundary constraint $\boldsymbol{b}(\boldsymbol{x}(0), \boldsymbol{x}(t_f), t_f) = \boldsymbol{0}$, our optimal control problem consists in finding state and action trajectories $\boldsymbol{x}(t)$ and $\boldsymbol{u}(t)$, and a final time t_f, that

$$\begin{aligned}
\underset{\boldsymbol{x}(\cdot), \boldsymbol{u}(\cdot), t_f}{\text{minimize}} \quad & J(\boldsymbol{x}(t), \boldsymbol{u}(t)) = \int_0^{t_f} L(\boldsymbol{x}(t), \boldsymbol{u}(t))\, dt && && \text{(5a)} \\
\text{subject to} \quad & \boldsymbol{h}(\boldsymbol{x}(t), \boldsymbol{u}(t)) \leq \boldsymbol{0}, && t \in [0, t_f], && \text{(5b)} \\
& \dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t), t), && t \in [0, t_f], && \text{(5c)} \\
& \boldsymbol{b}(\boldsymbol{x}(0), \boldsymbol{x}(t_f), t_f) = \boldsymbol{0}, && && \text{(5d)} \\
& t_f \geq 0, && && \text{(5e)}
\end{aligned}$$

Solving (5) via collocation involves partitioning the time history of the control and state variables into N intervals delimited by $N+1$ knot points $t_k, k = 0 \dots N$, then transcribing (5a)-(5c) 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 (5a) and (5b) are relatively straightforward and less relevant in the context of this paper. They can be done, for example, by approximating the integral in (5a) using some quadrature rule, and enforcing (5b) for all knot points t_k. The transcription of (5c), in contrast, is substantially more involved, and will be the main subject of the rest of the paper. In particular, we will seek to construct appropriate polynomial approximations of the solutions $\boldsymbol{x}(t)$ of the ODE in (5c) 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 (5c) over the whole time horizon $[0, t_f]$.

III. First order methods

Two of the most widely used transcriptions of (5c) are those of the trapezoidal and Hermite-Simpson methods. To see where these transcriptions incur in dynamical error, and ease the development of our second order methods, we briefly explain how they approximate the dynamics equations and obtain their approximation polynomials for the state. While the results match those by Betts [5] or Kelly [11], we follow a derivation process that is closer to the one used in numerical analysis [9] and in [10], which facilitates the transition to our new methods in Section IV.

A. Trapezoidal collocation

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

$$x(\tau) = a + b\tau + c\tau^2, \tag{6a}$$
$$\dot{x}(\tau) = b + 2c\tau, \tag{6b}$$

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

$$x_k = x(0), \tag{7}$$
$$\dot{x}_k = \dot{x}(0), \tag{8}$$
$$\dot{x}_{k+1} = \dot{x}(h), \tag{9}$$

where $h = t_{k+1} - t_k$. Evaluating the right-hand sides of (7)-(9) using (6) we obtain

$$\begin{bmatrix} x_k \\ \dot{x}_k \\ \dot{x}_{k+1} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 2h \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix}, \tag{10}$$

so solving for a, b, c and substituting the expressions in (6a), we have

$$x(\tau) = x_k + \dot{x}_k \tau + \frac{\tau^2}{2h}(\dot{x}_{k+1} - \dot{x}_k). \tag{11}$$

Equation (11) 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 [9, page 30], we determine the three parameters of (11) by enforcing the initial value constraint $x(0) = x_k$ and two collocation constraints of the form

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

From (11) we see that $x(0) = x_k$ 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

$$\dot{x}_k = f_k, \tag{12}$$
$$\dot{x}_{k+1} = f_{k+1}, \tag{13}$$

where f_k is a shorthand of $f(\boldsymbol{x}_k, \boldsymbol{u}_k, t_k)$. The value x_{k+1}, then, is obtained by evaluating (11) for $\tau = h$. This results in the constraint

$$x_{k+1} = x_k + \frac{h}{2}(\dot{x}_{k+1} + \dot{x}_k), \tag{14}$$

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

Note that equations (12)-(14) 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 (12) and (13) into (14) to obtain the more compact expression

$$x_{k+1} = x_k + \frac{h}{2}(f_{k+1} + f_k), \tag{15}$$

which we recognize as the common transcription rule in trapezoidal collocation [11, 5]. 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.

B. Hermite-Simpson collocation

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

$$x(\tau) = a + b\tau + c\tau^2 + d\tau^3, \tag{16a}$$
$$\dot{x}(\tau) = b + 2c\tau + 3d\tau^2. \tag{16b}$$

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

$$\begin{aligned} x_k &= x(0), \\ \dot{x}_k &= \dot{x}(0), \\ \dot{x}_c &= \dot{x}(h/2), \\ \dot{x}_{k+1} &= \dot{x}(h), \end{aligned}$$

where the extra parameter $\dot{x}_c$ is added since four parameters are needed to determine a third degree polynomial. Evaluating these identities using (16), solving for $a, \ldots, d$, and substituting the expressions in (16a), we obtain the interpolation polynomial

$$\begin{aligned} x(\tau) = x_k + \dot{x}_k\tau - \frac{\tau^2}{2h}(3\dot{x}_k - 4\dot{x}_c + \dot{x}_{k+1}) + \\ + \frac{\tau^3}{3h^2}(2\dot{x}_k - 4\dot{x}_c + 2\dot{x}_{k+1}). \end{aligned} \tag{17}$$

In order to determine the four parameters of (17), four conditions have to be imposed, and the Hermite-Simpson method makes this by fixing $x(0) = x_k$ (which holds by construction) and imposing the dynamics at the two bounding knot points and the midpoint between them:

$$\dot{x}_k = f_k, \tag{18}$$
$$\dot{x}_{k+1} = f_{k+1}, \tag{19}$$
$$\dot{x}_c = f_c. \tag{20}$$

In the latter equation, $f_c = f(\boldsymbol{x}_c, \boldsymbol{u}_c, t_c)$, where $\boldsymbol{x}_c = \boldsymbol{x}(h/2)$, $\boldsymbol{u}_c = \boldsymbol{u}(h/2)$, and $t_c = t_k + h/2$. Moreover, the values x_c that are needed in f_c can be expressed in terms of the above four parameters by evaluating (17) for $\tau = h/2$, which yields

$$x_c = x_k + \frac{h}{24}(5\dot{x}_k + 8\dot{x}_c - \dot{x}_{k+1}). \tag{21}$$

Finally, the continuity constraint between intervals k and $k+1$ is obtained by evaluating (17) for $\tau = h$:

$$x_{k+1} = x_k + \frac{h}{6}(\dot{x}_k + 4\dot{x}_c + \dot{x}_{k+1}). \tag{22}$$

Equations (18)-(22) already form a transcription of our ODE in $[t_k, t_{k+1}]$, but a transcription involving less variables can be obtained by substituting (18)-(20) in (22) and (21), which gives

$$x_{k+1} = x_k + \frac{h}{6}(f_k + 4f_c + f_{k+1}), \tag{23a}$$
$$x_c = x_k + \frac{h}{24}(5f_k + 8f_c - f_{k+1}). \tag{23b}$$

If preferred, we can also remove the dependence on f_c in (23b). This is achieved by isolating f_c from (23a) and substituting the result in (23b), which yields the alternative transcription

$$x_{k+1} = x_k + \frac{h}{6}(f_k + 4f_c + f_{k+1}), \tag{24a}$$
$$x_c = \frac{1}{2}(x_k + x_{k+1}) + \frac{h}{8}(f_k - f_{k+1}). \tag{24b}$$

Both transcriptions in (23) and (24) 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 (24) allows us to eliminate x_c by substituting (24b) in (24a), which results in a single equation that is known as the compressed form of Hermite-Simpson collocation [11, 5]. 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 [11].

Note that, despite the polynomial approximation into 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.

C. 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 (12)-(13), or (18)-(20), in the corresponding interpolating polynomials (11) and (17), 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 case of Hermite-Simpson, different options are possible. Some authors use a quadratic interpolation of the three control values available in each interval [11], while others prefer a linear interpolation and enforce the midpoint value to be the mean of the two bounding values [23].

D. Downsides of using first order methods

In a first order dynamic system, imposing (12)-(13) or (18)-(20) grants that the system dynamics is effectively satisfied at all 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 the collocation points, their derivatives may be different at these points. 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 $\dot{v}_k = g(\boldsymbol{q}_k, \boldsymbol{v}_k, \boldsymbol{u}_k, t_k)$ are satisfied, this does not imply that $\ddot{q}_k = g(\boldsymbol{q}_k, \boldsymbol{v}_k, \boldsymbol{u}_k, t_k)$, what means that, with a transcription based on (3), the system dynamics (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 $q(t)$ and $v(t)$ are approximated with their interpolation polynomials, 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 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 $\boldsymbol{u}_k$.

IV. Second order methods

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 and configuration variables are not treated as independent from each other, but explicitly defined as $v(t) = \dot{q}(t)$. 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.

A. The trapezoidal method for 2nd 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 will consist 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 $q(\tau)$ must be of degree three at least. So, we will have, for a given interval

$$q(\tau) = a + b\tau + c\tau^2 + d\tau^3, \tag{25a}$$
$$\dot{q}(\tau) = b + 2c\tau + 3d\tau^2, \tag{25b}$$
$$\ddot{q}(\tau) = 2c + 6d\tau. \tag{25c}$$

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 v_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:

$$\begin{aligned} q_k &= q(0) \\ v_k &= \dot{q}(0) \\ \ddot{q}_k &= \ddot{q}(0) \\ \ddot{q}_{k+1} &= \ddot{q}(h). \end{aligned}$$

Evaluating these identities using (25) and solving for a, b, c, d, we can write the interpolation polynomial $q(\tau)$ as:

$$q(\tau) = q_k + v_k\tau + \ddot{q}_k\frac{\tau^2}{2} + \frac{\tau^3}{6h}(\ddot{q}_{k+1} - \ddot{q}_k). \tag{26}$$

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

$$q_{k+1} = q_k + v_k h + \frac{h^2}{6}(\ddot{q}_{k+1} + 2\ddot{q}_k), \tag{27a}$$
$$v_{k+1} = v_k + \frac{h}{2}(\ddot{q}_{k+1} + \ddot{q}_k), \tag{27b}$$

and imposing the collocation constraints

$$\ddot{q}_k = g_k, \tag{28a}$$
$$\ddot{q}_{k+1} = g_{k+1}, \tag{28b}$$

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

$$q_{k+1} = q_k + v_k h + \frac{h^2}{6}(g_{k+1} + 2g_k), \tag{29a}$$
$$v_{k+1} = v_k + \frac{h}{2}(g_{k+1} + g_k). \tag{29b}$$

Note that, in this case, the trapezoidal rule only applies for the velocity, but not for the configuration itself, which is given by equation (29a).

It is worth observing that, 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 [7, 12, 4].

Method	Collocation equations	N. col. pts.	Num. vars.	Num. eqs.	Num. DOF
Trapezoidal (1st order)	$x_{k+1} = x_k + \frac{h}{2}(f_{k+1} + f_k)$	$N+1$	$(N+1)(2n_q + n_u)$	$2Nn_q$	$2n_q + (N+1)n_u$
Trapezoidal (2nd order)	$q_{k+1} = q_k + v_k h + \frac{h^2}{6}(g_{k+1} + 2g_k)$ $v_{k+1} = v_k + \frac{h}{2}(g_{k+1} + g_k)$				
Hermite-Simpson (1st order)	$x_{k+1} = x_k + \frac{h}{6}(f_k + 4f_c + f_{k+1})$ $x_c = \frac{1}{2}(x_k + x_{k+1}) + \frac{h}{8}(f_k - f_{k+1})$	$2N+1$	$(2N+1)(2n_q + n_u)$	$4Nn_q$	$2n_q + (2N+1)n_u$
Hermite-Simpson (2nd order)	$q_{k+1} = q_k + v_k h + \frac{h^2}{6}(g_k + 2g_c)$ $v_{k+1} = v_k + \frac{h}{6}(g_k + 4g_c + g_{k+1})$ $q_c = q_k + \frac{h}{32}(13v_k + 3v_{k+1}) + \frac{h^2}{192}(11g_k - 5g_{k+1})$ $v_c = \frac{1}{2}(v_k + v_{k+1}) + \frac{h}{8}(g_k - g_{k+1})$				

TABLE I
COLLOCATION EQUATIONS AND PROBLEM SIZE INDICATORS FOR ALL METHODS IN TERMS OF THE NUMBER N OF INTERVALS

B. The Hermite-Simpson method for 2nd 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 $q(\tau)$, 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

$$q(\tau) = a + b\tau + c\tau^2 + d\tau^3 + e\tau^4, \tag{30}$$
$$\dot{q}(\tau) = b + 2c\tau + 3d\tau^2 + 4e\tau^3, \tag{31}$$
$$\ddot{q}(\tau) = 2c + 6d\tau + 12e\tau^2. \tag{32}$$

Since five parameters are needed to determine the five coefficients of $q(\tau)$, 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 v_k at the initial point:

$$\begin{aligned} q_k &= q(0) \\ v_k &= \dot{q}(0) \\ \ddot{q}_k &= \ddot{q}(0) \\ \ddot{q}_c &= \ddot{q}(h/2) \\ \ddot{q}_{k+1} &= \ddot{q}(h). \end{aligned}$$

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

$$\begin{aligned} q(\tau) = q_k + v_k\tau + \frac{\tau^2}{2}\ddot{q}_k - \frac{\tau^3}{6h}(3\ddot{q}_k - 4\ddot{q}_c + \ddot{q}_{k+1}) + \\ + \frac{\tau^4}{6h^2}(\ddot{q}_k - 2\ddot{q}_c + \ddot{q}_{k+1}). \end{aligned} \tag{33}$$

Evaluating (33) and its derivative for $\tau = h$ we get

$$q_{k+1} = q_k + v_k h + \frac{h^2}{6}(\ddot{q}_k + 2\ddot{q}_c), \tag{34a}$$
$$v_{k+1} = v_k + \frac{h}{6}(\ddot{q}_k + 4\ddot{q}_c + \ddot{q}_{k+1}), \tag{34b}$$

and imposing the collocation constraints

$$\ddot{q}_k = g_k, \tag{35a}$$
$$\ddot{q}_c = g_c, \tag{35b}$$
$$\ddot{q}_{k+1} = g_{k+1}, \tag{35c}$$

yields

$$q_{k+1} = q_k + v_k h + \frac{h^2}{6}(g_k + 2g_c), \tag{36a}$$
$$v_{k+1} = v_k + \frac{h}{6}(g_k + 4g_c + g_{k+1}), \tag{36b}$$

where we recognize that (36b) is the Simpson quadrature for the velocity. The terms g_c in these equations involve the midpoint coordinate $q_c = q(h/2)$, and the velocity $v_c = \dot{q}(h/2)$, but these can be obtained by evaluating (33) and its derivative for $\tau = h/2$, and imposing (35), which yields

$$q_c = q_k + \frac{h}{2}v_k + \frac{h^2}{96}(7g_k + 6g_c - g_{k+1}), \tag{37a}$$
$$v_c = v_k + \frac{h}{24}(5g_k + 8g_c - g_{k+1}). \tag{37b}$$

The equations in (36) and (37) together constitute the separated form of the second order Hermite-Simpson method. Note however that, since q_c and v_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 (36b) and substitute the result in (37) to yield:

$$q_c = q_k + \frac{h}{32}(13v_k + 3v_{k+1}) + \frac{h^2}{192}(11g_k - 5g_{k+1}), \tag{38a}$$
$$v_c = \frac{1}{2}(v_k + v_{k+1}) + \frac{h}{8}(g_k - g_{k+1}). \tag{38b}$$

Written in this way, q_c and v_c can be replaced in the expression of g_c in (36) to transcribe the problem in compressed form, i.e., eliminating the need to treat q_c and v_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

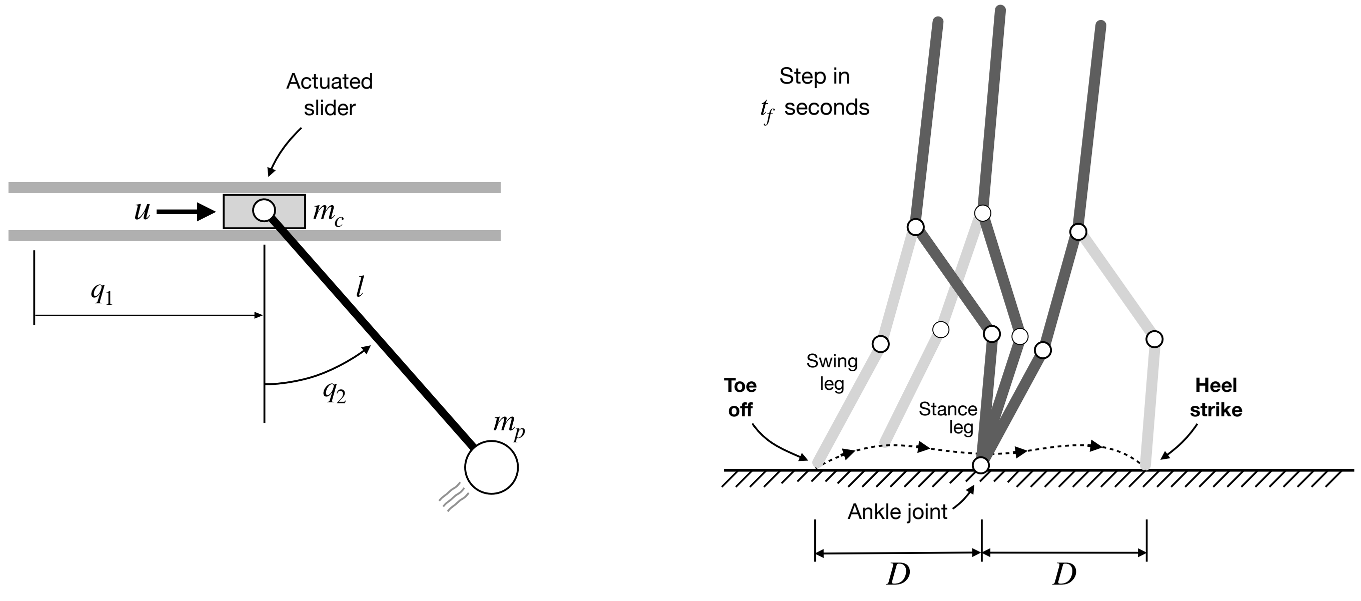

Fig. 1. Test cases. Left: A cart-pole system that has to perform a swing-up motion. Right: a five-link biped walking under a periodic gait. The three snapshots illustrate the motion that occurs between the *toe off* and *heel strike* events defining a period of the gait.

derivative imposed by the collocation constraints, what gives rise to smooth, continuous acceleration trajectories just like in the second order trapezoidal method.

Table I provides a summary of the equations for each collocation scheme, together with the number of collocation points, variables, equations and degrees of freedom they introduce in the NLP problem, in terms of N (the number of intervals of the discretization).

V. Test Cases

The performance of the first and second order methods is next evaluated and compared using two trajectory optimization problems documented in detail in [11], namely, the cart-pole swing-up and the five-link bipedal walking problems (Fig. 1). Analytical solutions for these problems are not available, so in order to compare the performance of the different collocation methods, we will compute the dynamic transcription error produced by each of them. To this end, we define the errors of the dynamics constraints as follows.

The first order dynamic error of the q_i coordinate is

$$\varepsilon^{[1]}_{q_i}(t) = \dot{q}_i(t) - v_i(t). \tag{39}$$

In general, this error is non-null in first order methods, as they do not enforce $v_i(t) = \dot{q}_i(t)$ for all t. For the same coordinate, the second order dynamic error is

$$\varepsilon^{[2]}_{q_i}(t) = \ddot{q}_i(t) - g_i(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{u}, t). \tag{40}$$

We found this error more meaningful than the $\varepsilon^{[1]}_{q_i}(t)$ error reported in [11], since it reflects the deviation from the actual system dynamics, which is expected to be minimized with the optimization process. When all coordinates in $\boldsymbol{q}$ have the same units, it also makes sense to define corresponding joint errors for all coordinates. A plausible definition for these errors is

$$\varepsilon^{[r]}(t) = |\varepsilon^{[r]}_{q_1}(t)| + \ldots + |\varepsilon^{[r]}_{q_{n_q}}(t)|, \quad r = 1, 2. \tag{41}$$

Finally, to summarize the error functions in just one number, we compute their integrals over $[0, t_f]$:

$$E^{[r]}_{q_i} = \int_0^{t_f} |\varepsilon^{[r]}_{q_i}(t)|\, dt, \qquad r = 1, 2, \tag{42}$$

$$E^{[r]} = \int_0^{t_f} \varepsilon^{[r]}(t)\, dt, \qquad r = 1, 2. \tag{43}$$

To perform the comparisons of the four collocation methods, we have implemented them in Python, using the symbolic package SymPy [13] to model the systems, and the toolbox CasADi [2] to solve the constrained optimization problems that result. 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 [24] in conjunction with the linear solver MUMPS [1]. The reader can interactively reproduce our results through the online Jupyter notebooks [19, 18]. The execution times we report in this paper have been obtained on a single-thread implementation running on an iMac computer with an Intel i7, 8-core 10th generation processor at 3.8 GHz.

In what follows, and in order to simplify the explanations, we use the shorthands *Tz1* and *HS1* for the first order trapezoidal and Hermite-Simpson methods, and *Tz2* and *HS2* for the corresponding second order ones.

	N	t_c	$E^{[1]}_{q1}$	$E^{[1]}_{q2}$	$E^{[2]}_{q1}$	$E^{[2]}_{q2}$
		(s)	(m)	(rad)	(m/s)	(rad/s)
Tz1	50	0.025	0.0066	0.0167	0.504	1.281
Tz2	50	0.025	0	0	0.052	0.170
HS1	25	0.020	0.0014	0.0043	0.113	0.338
HS2	25	0.023	0	0	0.016	0.052

TABLE II
PERFORMANCE VALUES FOR THE CART-POLE SWING-UP PROBLEM.

A. 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

$$J(u(t)) = \int_0^{t_f} u^2(t)dt, \tag{44}$$

where u is the force applied to the cart, and we adopt the same dynamic equations and problem parameters as in [11]. An animation of the solution obtained with *HS2* and $N = 25$ can be seen in [21].

Figure 2 compares the errors $\varepsilon^{[1]}_{q_i}(t)$ and $\varepsilon^{[2]}_{q_i}(t)$ obtained by all methods for the variables q_1 and q_2 shown in Fig. 1. The number of intervals used in the comparison is $N = 50$ for the trapezoidal scheme and $N = 25$ for Hermite-Simpson. This yields a fair comparison, as then the number of collocation points, variables, and degrees of freedom of the optimization are the same for the four problems (cf. Table I). The plots corresponding to the first order error $\varepsilon^{[1]}_{q_i}(t)$ of Fig. 2, in the first and third row respectively, confirm that *Tz1* and *HS1* present a non-negligible first order error, while in *Tz2* and *HS2* this error is exactly zero as expected.

The plots in the second and fourth rows clearly show a discontinuity at the knot points of the second order error $\varepsilon^{[2]}_{q_i}(t)$ for *Tz1* and *HS1*, reflecting the discontinuity of $\ddot{\boldsymbol{q}}(t)$ at these points. In contrast, for *Tz2* and *HS2*, the error functions are continuous and vanish at the collocation points, evidencing that, as anticipated in Section III-D, the system dynamics is exactly satisfied at all collocation points for the second order methods, but not for the first order ones.

The figure also shows the dramatic reductions of $\varepsilon^{[2]}_{q_i}(t)$ for the second order methods when compared with the corresponding first order ones. The numerical evaluation of the results appears in Table II, which provides the computation times t_c and the integral errors $E^{[2]}_{q_i}$ for this problem. It can be seen that the errors $E^{[2]}_{q_i}$ are almost one order of magnitude lower for the second order methods than for their first order counterparts, despite using a very similar computation time. It is interesting to see that the errors $E^{[2]}_{q_i}$ achieved by *Tz2* are about a half of those of *HS1* for the same number of collocation points. The comparison is relevant since both methods use polynomials of the same degree to approximate $q_i(t)$.

B. The 5-link bipedal walking problem

We next apply the four methods to optimize a periodic gait for the planar biped robot shown in Fig. 1. 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, therefore, this robot is underactuated, but it is substantially more complex. The system is commonly used as a testbed when studying bipedal walking [25, 26, 14, 17].

For this example we use the dynamic model given by Kelly [11], which matches the one in [25] with parameters corresponding to the RABBIT prototype [6]. 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 [11], 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 [11], 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

$$J(\boldsymbol{u}(t)) = \int_0^{t_f} \boldsymbol{u}(t)^\top \boldsymbol{u}(t)\, dt. \tag{45}$$

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

$$\boldsymbol{x}_0 = \boldsymbol{f}_H(\boldsymbol{x}_f), \tag{46}$$

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 (46), which is the particular form of the boundary constraint (5d) 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 [11]. 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. 1 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 [20].

Figure 3 shows the second order joint errors $\varepsilon^{[2]}$ for the different collocation methods. As before, the number of intervals used in the trapezoidal cases is twice those used in Hermite-Simpson so as to have identical number of collocation points and have balanced comparisons. The results are qualitatively similar to those of the cart-pole, though here the error diminution obtained by the second order methods is even

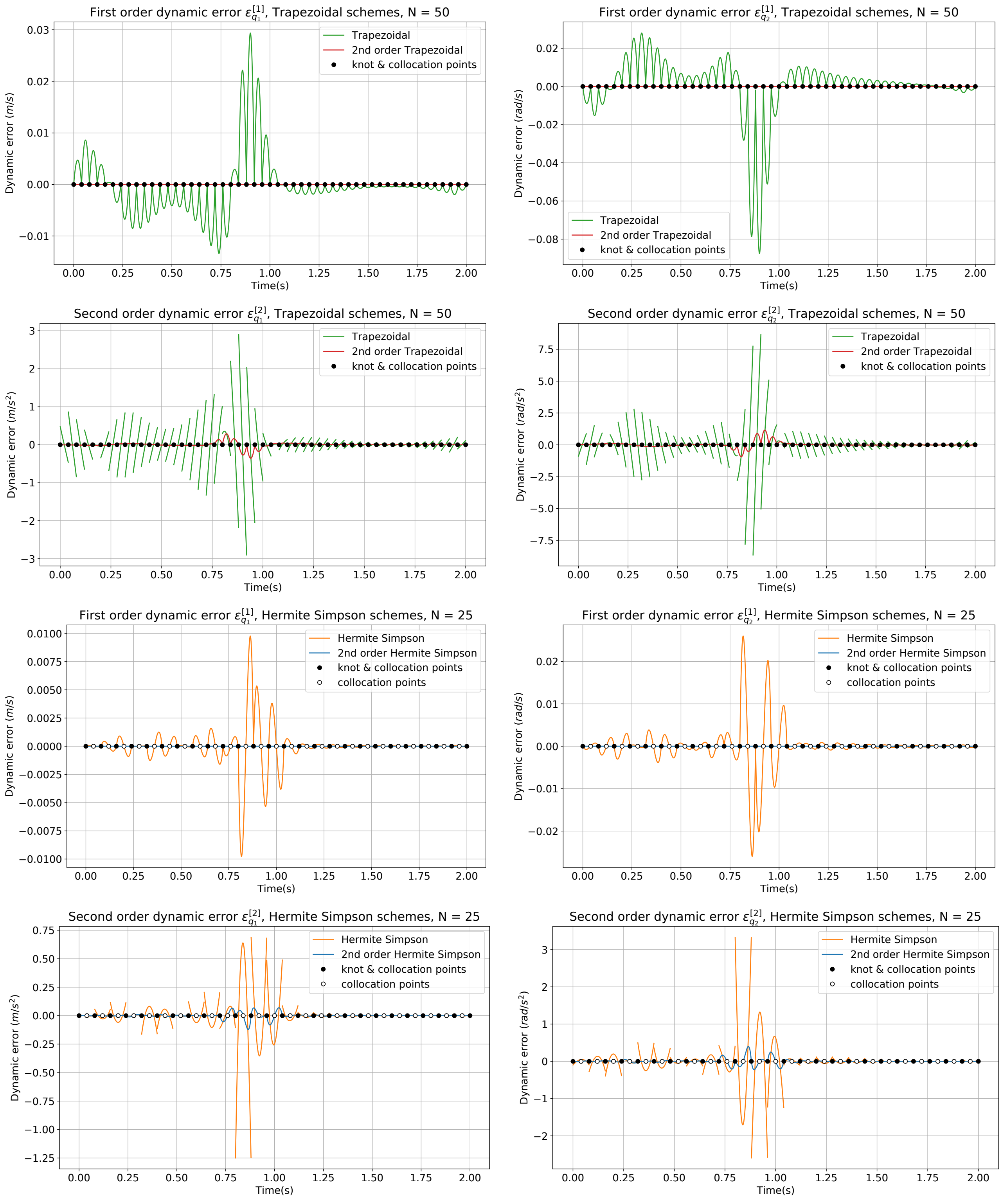

Fig. 2. Cart-pole problem: Plots of the first and second order dynamic errors $\varepsilon^{[1]}_{q_i}(t)$ and $\varepsilon^{[2]}_{q_i}(t)$ for q_1 and q_2 (left and right columns, respectively), using the trapezoidal and Hermite-Simpson methods. To compare the results with those in [11], note that [11] actually provides the plots of $-\varepsilon^{[1]}_{q_i}(t)$ for *HS1*.

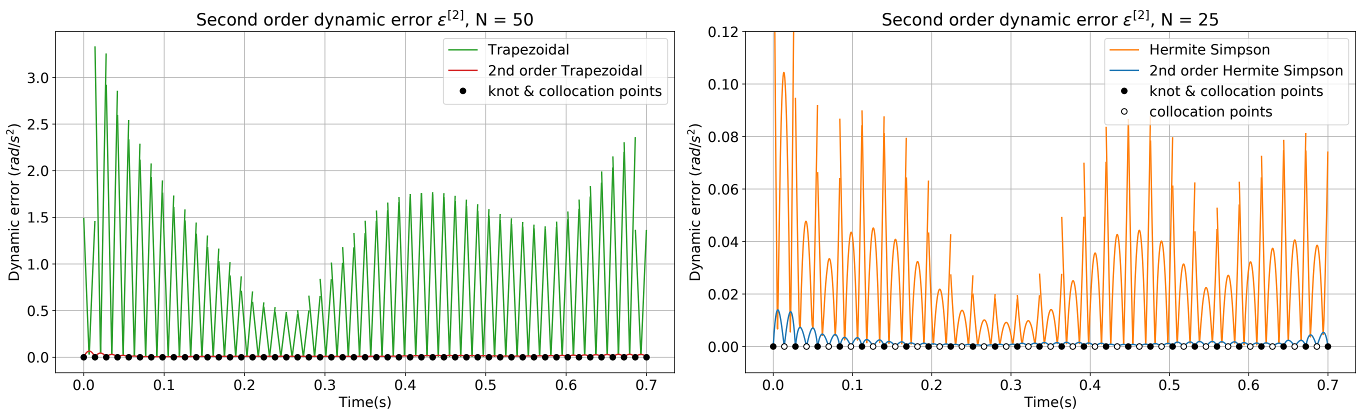

Fig. 3. Second order dynamic errors for the biped problem.

	N	t_c	$E^{[1]}$	$E^{[2]}$
		(s)	(rad)	(rad/s)
Tz1	50	0.122	0.0025	0.5328
Tz2	50	0.125	0	0.0081
HS1	25	0.131	8.2×10^{-5}	0.0182
HS2	25	0.127	0	0.0011

TABLE III
PERFORMANCE VALUES FOR THE BIPEDAL WALKING PROBLEM.

more accentuated. As can be seen in Table III, the integral joint second order error $E^{[2]}$ of *HS2* improves in more than one order of magnitude that of *HS1* even using a slightly lower computation time. In the case of *Tz2*, its improvement over *Tz1* is still higher, reaching a reduction factor near 66, and using a computation time only slightly longer.

C. Performance scaling with the number of intervals

To evaluate the evolution of the performance of the different methods when the number of intervals increases, a series of experiments have been conducted by progressively rising N from 20 to 200. Each experiment has been launched 100 times and the average of the computation times and integral second order errors $E^{[2]}$ obtained are represented in Figures 4 and 5 (note the logarithmic scale in the plots of $E^{[2]}$). In both test problems, the best results for $E^{[2]}$ are those of *HS2*, which improves the results of *HS1* in about one order of magnitude in all cases, and the improvement rate tends to increase with the number of intervals N. The same behavior is observed for *Tz2* with respect to *Tz1*. Interestingly, in all cases the performance of *Tz2* produces, for the same number of intervals N, only about twice the error of *HS1*, 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 *HS1* with the $2N$ value of *Tz2*. A close look at the plots will convince the reader that the result of this comparison is favorable for *Tz2* over *HS1* in all cases.

The bottom plots in Figs. 4 and 5 show the growth of the computation times with the number of intervals N. The higher complexity of the five link biped problem (the HS methods involve $14(2N+1)$ variables and $20N$ equations) is reflected into computation times which are about one order of magnitude longer that those of the cart-pole (involving $5(2N+1)$ variables and $8N$ equations). The comparison of the costs of the four methods shows that *HS2* is the most time-expensive, but comparable to *HS1*, while *Tz1* and *Tz2* are very close together and less expensive than the Hermite-Simpson methods for the same N.

In both test cases, the computation times when N is increased grow faster for the Hermite-Simpson than for the trapezoidal methods, but the increasing rate is not so different, and even gets inverted, when comparing the computation times for the same number of collocation points.

VI. CONCLUSIONS

Trapezoidal and Hermite-Simpson collocation methods are very popular in the robotics community. However, they are conceived for dynamic systems of first order, while the dynamics of the systems found in robotics are very often second order. The transcription of a second order system as a first order one has the unexpected effect that the second order dynamic equations are not actually imposed at the collocation points. Directly imposing the second order constraints at the same such points as the original algorithms requires increasing the degree of the polynomials approximating the configuration trajectory, while keeping the same degree for those of the velocity and control trajectories. This is achieved with the second order methods we propose, which grant the functional consistency between the configuration and velocity trajectories not only at the collocation points, but also along the computed trajectory. We have also shown, using benchmark problems from the literature, that the new methods provide trajectories with a much smaller dynamic error than traditional methods, despite they require a comparable amount of computation time. This implies that the obtained trajectories will be more

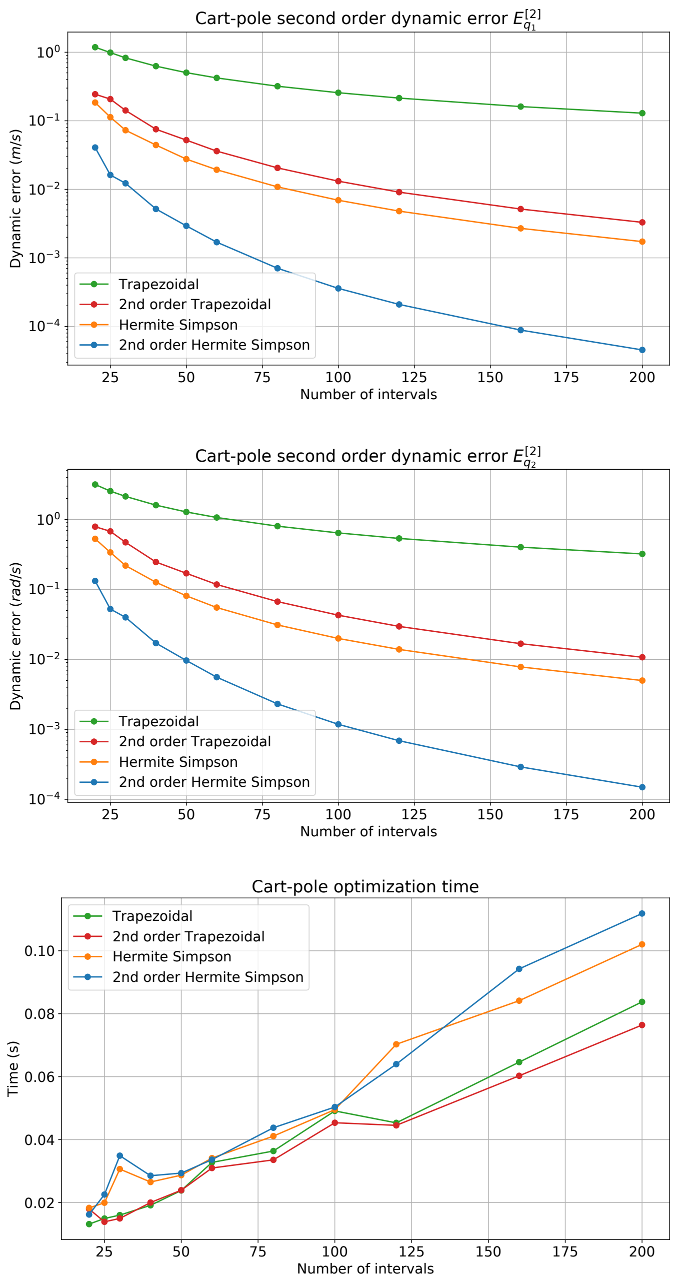

Fig. 4. Cart-pole problem: performance for increasing values of N.

compliant with the system dynamics, so they will use less control effort when they are tracked in practice with the help of a controller. Moreover, the trajectories of the new methods are twice differentiable, so in addition to enjoying smooth velocities, their accelerations will be continuous, which are very desirable properties from a control perspective.

Points that deserve further attention are the extension of these ideas to pseudospectral collocation methods [16], which allow approximations of arbitrary degree, the inclusion of constraints to limit the jerk and higher-order derivatives of the trajectories, and the theoretical analysis of the approximation error of the second order methods to see how they compare to first order ones.

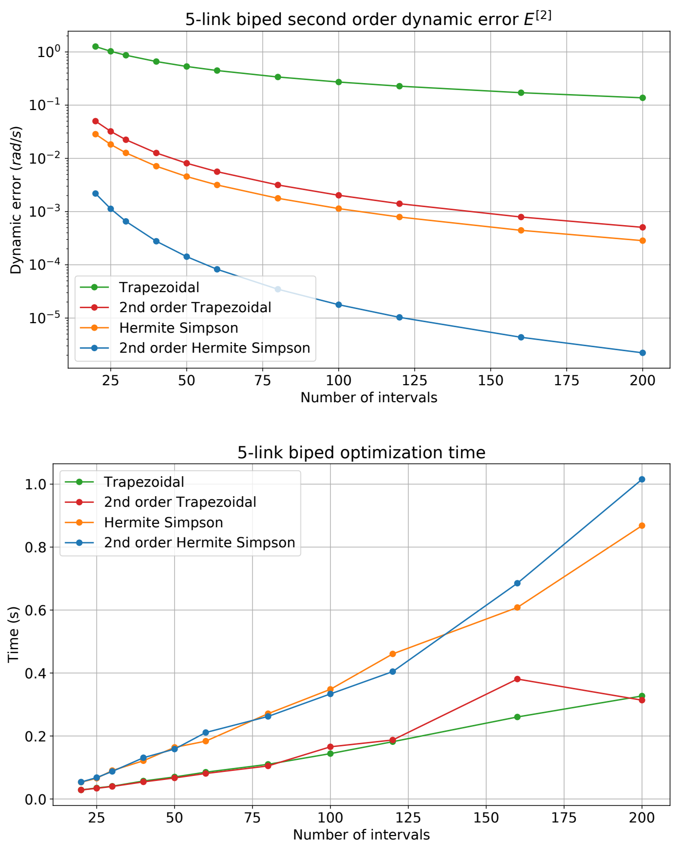

Fig. 5. Biped problem: performance for increasing values of N.

Acknowledgements

This work has been partially supported by Agencia Estatal de Investigación under project *Kinodyn+*, with reference PID2020-117509GB-I00 / AEI / 10.13039/50110001103.

References

[1] Patrick R. Amestoy, Iain S. Duff, Jean-Yves L'Excellent, and Jacko Koster. A Fully Asynchronous Multifrontal Solver Using Distributed Dynamic Scheduling. *SIAM Journal on Matrix Analysis and Applications*, 23(1):15–41, 2001.

[2] Joel A. E. Andersson, Joris Gillis, Greg Horn, James B. Rawlings, and Moritz Diehl. CasADi – A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 11(1):1–36, 2019.

[3] Victor M. Becerra. Solving complex optimal control problems at no cost with PSOPT. In *2010 IEEE International Symposium on Computer-Aided Control System Design*, pages 1391–1396, 2010.

[4] Lars Berscheid and Torsten Kröger. Jerk-limited Real-time Trajectory Generation with Arbitrary Target States. In *Robotics: Science and Systems*, 2021.

[5] John T. Betts. *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming*. SIAM, jan 2010.

[6] Christine Chevallereau, Gabriel Abba, Yannick Aoustin, Franck Plestan, Eric R. Westervelt, Carlos Canudas De Wit, and Jessy Grizzle. RABBIT: a testbed for advanced control theory. *IEEE Control Systems Magazine*, 23(5):57–79, 2003.

[7] Daniela Constantinescu and Elizabeth A. Croft. Smooth and time-optimal trajectory planning for industrial manipulators along specified paths. *Journal of Robotic Systems*, 17(5):233–249, 2000.

[8] Bruce A. Conway and Stephen W. Paris. Spacecraft Trajectory Optimization Using Direct Transcription and Nonlinear Programming. In Bruce Conway, editor, *Spacecraft Trajectory Optimization*, pages 37–78. Cambridge University Press, 2010.

[9] Ernst Hairer, Gerhard Wanner, and Christian Lubich. *Geometric Numerical Integration*. Springer Berlin Heidelberg, 2002.

[10] Charles R. Hargraves and Stephen W. Paris. Direct trajectory optimization using nonlinear programming and collocation. *Journal of guidance, control, and dynamics*, 10(4):338–342, 1987.

[11] Matthew Kelly. An Introduction to Trajectory Optimization: How to Do Your Own Direct Collocation. *SIAM Review*, 59(4): 849–904, 2017.

[12] Sonja Macfarlane and Elizabeth A. Croft. Jerk-bounded manipulator trajectory planning: design for real-time applications. *IEEE Transactions on Robotics and Automation*, 19(1):42–52, 2003.

[13] Aaron Meurer, Christopher P. Smith, Mateusz Paprocki, Ondřej Čertík, Sergey B. Kirpichev, Matthew Rocklin, Amit Kumar, Sergiu Ivanov, Jason K. Moore, Sartaj Singh, Thilina Rathnayake, Sean Vig, Brian E. Granger, Richard P. Muller, Francesco Bonazzi, Harsh Gupta, Shivam Vats, Fredrik Johansson, Fabian Pedregosa, Matthew J. Curry, Andy R. Terrel, Štěpán Roučka, Ashutosh Saboo, Isuru Fernando, Sumith Kulal, Robert Cimrman, and Anthony Scopatz. SymPy: symbolic computing in Python. *PeerJ Computer Science*, 3:e103, January 2017.

[14] Hae-Won Park, Koushil Sreenath, Alireza Ramezani, and Jessy W. Grizzle. Switching control design for accommodating large step-down disturbances in bipedal robot walking. In *2012 IEEE International Conference on Robotics and Automation*, pages 45–50. IEEE, 2012.

[15] Michael Posa, Scott Kuindersma, and Russ Tedrake. Optimization and stabilization of trajectories for constrained dynamical systems. In *IEEE International Conference on Robotics and Automation*, pages 1366–1373, 2016.

[16] Isaac M. Ross and Mark Karpenko. A review of pseudospectral optimal control: From theory to flight. *Annual Reviews in Control*, 36(2):182–197, 2012.

[17] Cenk Oguz Saglam and Katie Byl. Robust Policies via Meshing for Metastable Rough Terrain Walking. In *Robotics: Science and Systems*, 2014.

[18] Siro Moreno-Martín. Online Jupyter notebook for the bipedal walking problem, 2022. URL https://mybinder.org/v2/gh/AunSiro/Second-Order-Schemes/HEAD?labpath=Five-Link-Biped-demo.ipynb.

[19] Siro Moreno-Martín. Online Jupyter notebook to reproduce the results of the cart-pole swing-up problem, 2022. URL https://mybinder.org/v2/gh/AunSiro/Second-Order-Schemes/HEAD?labpath=Cartpole-demo.ipynb.

[20] Siro Moreno-Martín. Animation of the solution obtained for the biped walking problem, 2022. URL https://youtu.be/dtS-WbESiW0.

[21] Siro Moreno-Martín. Animation of the solution obtained for the cart-pole problem, 2022. URL https://youtu.be/M0ivg_8s-I8.

[22] R. Tedrake. *Underactuated Robotics: Algorithms for Walking, Running, Swimming, Flying, and Manipulation (Course Notes for MIT 6.832).* MIT, 2022. Downloaded on 13 January 2022 from http://underactuated.mit.edu/.

[23] Francesco Topputo and Chen Zhang. Survey of Direct Transcription for Low-Thrust Space Trajectory Optimization with Applications. *Abstract and Applied Analysis*, 2014:1–15, 2014.

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

[25] Eric R. Westervelt, Jessy W. Grizzle, and Daniel E. Koditschek. Hybrid zero dynamics of planar biped walkers. *IEEE Transactions on Automatic Control*, 48(1):42–56, 2003.

[26] Tao Yang, Eric. R. Westervelt, Andrea Serrani, and James P. Schmiedeler. A framework for the control of stable aperiodic walking in underactuated planar bipeds. *Autonomous Robots*, 27(3):277–290, 2009.