Direct Collocation Methods for Trajectory Optimization in Constrained Robotic Systems

Ricard Bordalba, Tobias Schoels, Lluís Ros, Josep M. Porta, and Moritz Diehl

***Abstract*—Direct collocation methods are powerful tools to solve trajectory optimization problems in robotics. While their resulting trajectories tend to be dynamically accurate, they may also present large kinematic errors in the case of constrained mechanical systems, i.e., those whose state coordinates are subject to holonomic or nonholonomic constraints, like loop-closure or rolling-contact constraints. These constraints confine the robot trajectories to an implicitly-defined manifold, which complicates the computation of accurate solutions. Discretization errors inherent to the transcription of the problem easily make the trajectories drift away from this manifold, which results in physically inconsistent motions that are difficult to track with a controller. This paper reviews existing methods to deal with this problem and proposes new ones to overcome their limitations. Current approaches either disregard the kinematic constraints (which leads to drift accumulation) or modify the system dynamics to keep the trajectory close to the manifold (which adds artificial forces or energy dissipation to the system). The methods we propose, in contrast, achieve full drift elimination on the discrete trajectory, or even along the continuous one, without artificial modifications of the system dynamics. We illustrate and compare the methods using various examples of different complexity.**

***Index terms*— Trajectory optimization, motion planning, constrained system, holonomic, nonholonomic, direct collocation, manifold, drift, basic, Baumgarte, projection, PKT, local coordinates.**

I. Introduction

TRAJECTORY synthesis is one of the most fundamental problems in robotics. The goal is to find a sequence of control actions able to move a robot from a start to a goal state, while avoiding collisions with obstacles and incurring in minimum costs during the way. The problem is difficult in general. The control actions should be carefully selected to ensure the kinodynamic feasibility of the trajectory, which requires to take many aspects into account. A full robot model should be accounted for—including inertial and friction effects, motor saturations, joint or velocity limits, and other kinematic or dynamic constraints of relevance—as well as a rich-enough model of the environment. Traditionally, the robotics community has approached this problem with motion planners of various sorts, like those using potential fields, probabilistic roadmaps, or randomized tree techniques, among others [1]. Recently, however, the advances in computing power and mathematical programming are also giving rise to a new family of planners based on trajectory optimization methods [2]–[4]. A strong point of these methods is versatility. Constraints and cost functions of various forms can all be managed under a same paradigm, allowing to design fast, smooth, or control-efficient motions in broad classes of systems.

In essence, all trajectory optimization methods solve an instance of the variational problem of optimal control. Two strategies are mainly followed [5]. Indirect approaches initially derive the Pontryagin conditions of optimality and then solve the resulting boundary-value problem numerically. Direct approaches, in contrast, discretize the optimal control problem at the outset, and then tackle the discrete problem with nonlinear programming (NLP) methods. While indirect approaches tend to be more accurate on optimizing the cost function, they also require good guesses of the solution, which are difficult to provide in general. Direct approaches, in contrast, may yield slightly suboptimal trajectories, but show larger regions of convergence, which makes them preferable to solve problems in robotics. In these approaches, the dynamic constraints can be discretized using shooting methods, which often use explicit integrators to estimate the evolution of the system, or collocation methods, which avoid costly integration rules via spline approximations. Collocation methods are relatively fast and effectively solve a wide variety of problems, which justifies the growing interest they arouse [6]–[10], and why they constitute the main focus of this paper.

When computing a solution trajectory, a main concern is the consistency of the motor actions with the trajectory states, so the actions can closely reproduce such states when executed in the real robot. This calls for the use of realistic robot models, but also for an accurate satisfaction of the kinematic and dynamic constraints of such models along the entire trajectory. While direct collocation methods are good at achieving dynamic accuracy, little work has been devoted to also ensure their kinematic accuracy on constrained mechanical systems; i.e., systems with dependent state coordinates, like those involving closed kinematic chains, rolling contacts, or non-minimal representations of spatial orientations. In such systems, the state vector must satisfy a set of holonomic and nonholonomic constraints, which in general cannot be

This work has been partially funded by the Spanish Ministry of Science, Innovation, and Universities under projects DPI2017-88282-P, and PID2020-117509GB-I00/AEI/10.13039/50110001103, and by the German Federal Ministry for Economic Affairs and Energy (BMWi) via DyConPV (0324166B).

Ricard Bordalba, Lluís Ros, and Josep M. Porta are with the Institut de Robòtica i Informàtica Industrial (CSIC-UPC). C. Llorens Artigas 4-6, 08028 Barcelona, Catalonia (e-mails: rbordalba@iri.upc.edu, ros@iri.upc.edu, and porta@iri.upc.edu).

Tobias Schoels is with the Systems Control and Optimization Laboratory, Department of Microsystems Engineering (IMTEK), University of Freiburg, Georges-Koehler-Allee 102, 79110 Freiburg, Germany (e-mail: tobias.schoels@imtek.de).

Prof. Dr. Moritz Diehl is with the Systems Control and Optimization Laboratory, Department of Microsystems Engineering (IMTEK) and Department of Mathematics, University of Freiburg, Georges-Koehler-Allee 102, 79110 Freiburg, Germany (e-mail: moritz.diehl@imtek.uni-freiburg.de).

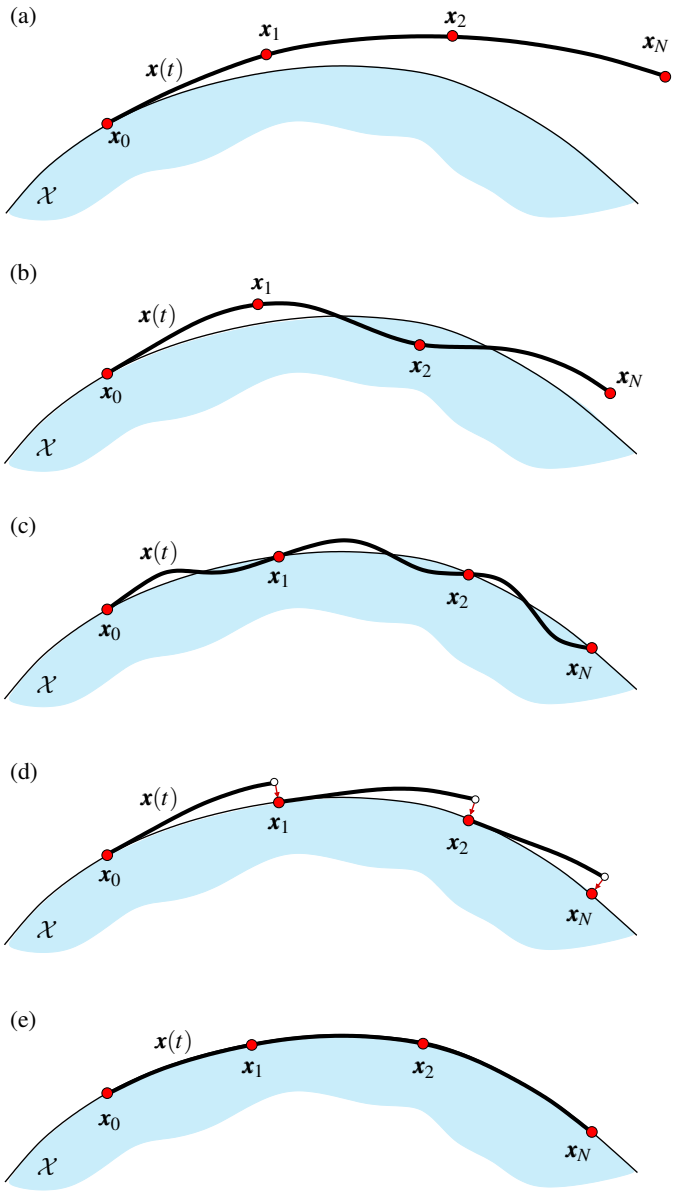


Fig. 1. Qualitative form of the trajectories $\boldsymbol{x}(t)$ obtained by existing methods (a, b, and c) and those proposed in this paper (d and e). From top to bottom: output of the "basic", "Baumgarte", "PKT", "projection", and "local coordinates" methods. The blue surface $\mathcal{X}$ is the state space manifold on which $\boldsymbol{x}(t)$ should evolve. The red dots indicate the knot states $\boldsymbol{x}_0,\dots,\boldsymbol{x}_N$ used to discretize $\boldsymbol{x}(t)$. In the basic method, the knot states, and so $\boldsymbol{x}(t)$, may increasingly drift away from $\mathcal{X}$. The knot states are kept near $\mathcal{X}$ in the Baumgarte method (b), and exactly on $\mathcal{X}$ in the PKT method (c), but at the cost of modifying the system dynamics. The methods we propose also eliminate drift at the knot states (d), or even along the entire trajectory (e), without modifying the system dynamics.

solved in closed form. This confines the robot trajectories to a nonlinear manifold implicitly defined in a larger ambient space, so any drift from such manifold caused by discretization errors will lead to unrealistic trajectories that are difficult to stabilise with a controller [7, 9].

In the literature, three main strategies have been given to mitigate the problem of drift. In a first strategy, which we call the basic method, the kinematic constraints are simply not enforced, but fine discretizations or high-order integrators can be used to keep trajectory drift to a minimum [8, 10]–[14]. This approach is easy to implement, but it increases the computational cost of solving the problem, and much drift can accumulate despite the precautions [15, 16]. In a second strategy, Baumgarte stabilization is used to modify the dynamic vector field of the system and make it convergent to the manifold [17, 18]. The method is also easy to implement, but it has three shortcomings: it adds artificial compliance and energy dissipation to the system, its parameters may be difficult to tune, and trajectory instabilities may arise as a result [18, 19]. A third strategy, which we call the PKT method, adapts the classical Hermite-Simpson collocation scheme to keep the trajectory knot points on the manifold [7]. This approach is a notable improvement over the Baumgarte method, but it also applies artificial modifications of the system dynamics in half of the collocation points, so the obtained trajectories may suffer from dynamic inconsistencies. Another drawback is that it can only use cubic polynomials, so it cannot be applied in hp-adaptive meshing schemes that improve the trajectory by increasing the polynomial degree, in addition to the mesh resolution [20, 21].

The goal of this paper is to review the previous methods and to propose new ones to overcome their limitations. By using projections and local charts of the manifold, we present two methods that keep the trajectory knot points exactly on the manifold without modifying the system dynamics (Fig. 1). The two methods, which we refer to as the "projection" and "local coordinates" methods, can compute approximation polynomials of arbitrary degree. While the former is easier to implement and usually faster, the latter achieves full drift elimination even for the continuous-time trajectory, which is beneficial when high-quality solutions are needed.

Both methods leverage techniques to solve differential algebraic equations (DAEs) that can be expressed as ODEs on manifolds [15, 22]. The projection method is based on the work by Hairer [23], who used orthogonal projections to cancel the drift from the algebraic manifold of a DAE. The local coordinates method, in turn, has its roots in Potra and Rheinboldt [24], and Yen et al. [25], who introduced the tangent space parameterization as a means to obtain trajectories lying continuously on such manifold. While the work in [15, 22]–[25] was mainly developed for dynamic system simulation, our goal is to show how it can be adapted to trajectory optimization in robotics.

The rest of the paper is structured as follows. Section II formally describes the class of systems we consider and formulates the continuous optimal control problem to be solved. Section III recalls background techniques to transcribe this problem into a nonlinear programming problem. Using these techniques, Section IV reviews the basic, Baumgarte, and PKT methods and discussses their strengths and weaknesses. Section V explains the methods we propose, which overcome such weaknesses. Important points to be considered in their implementation are then given in Section VI, and the performance of all methods is compared in Section VII by means of examples. Section VIII finally provides the conclusions of the paper and highlights points deserving further attention.

II. Problem formulation and assumptions

Let us describe the robot state by means of a tuple $\boldsymbol{x} = (\boldsymbol{q}, \dot{\boldsymbol{q}})$, where $\boldsymbol{q}$ is a vector of n_q generalized coordinates encoding the positions and orientations of all links at a given instant of time, and $\dot{\boldsymbol{q}} = \frac{d}{dt}\boldsymbol{q}$. We restrict our attention to constrained robotic systems, i.e., those in which $\boldsymbol{q}$ and $\dot{\boldsymbol{q}}$ are subject to equations of the form

$$\begin{cases} \boldsymbol{\Phi}(\boldsymbol{q}) = \boldsymbol{0}, & \text{(1a)} \\ \boldsymbol{B}(\boldsymbol{q})\,\dot{\boldsymbol{q}} = \boldsymbol{0}, & \text{(1b)} \end{cases}$$

where $\boldsymbol{\Phi}(\boldsymbol{q}) : \mathbb{R}^{n_q} \to \mathbb{R}^{n_p}$ and $\boldsymbol{B}(\boldsymbol{q})\,\dot{\boldsymbol{q}} : \mathbb{R}^{2n_q} \to \mathbb{R}^{n_v}$ are differentiable maps defining configuration and velocity constraints. Here, Eq. (1a) encompasses the holonomic constraints of the system (like joint-assembly or loop-closure constraints), and Eq. (1b) includes the time derivative of Eq. (1a) and all nonholonomic constraints intervening (like those arising from rolling contacts). Thus, $n_v = n_p + n_r$, where n_r is the number of non-holonomic constraints of the robot.

For ease of explanation, let us write the system in (1) as

$$\boldsymbol{F}(\boldsymbol{x}) = \boldsymbol{0}, \tag{2}$$

so $\boldsymbol{F} : \mathbb{R}^{n_x} \to \mathbb{R}^{n_e}$, where $n_x = 2\,n_q$, and $n_e = n_p + n_v$. The state space of the robot is then the set

$$\mathcal{X} = \{\boldsymbol{x} : \boldsymbol{F}(\boldsymbol{x}) = \boldsymbol{0}\}. \tag{3}$$

In the rest of the paper, we assume that $\boldsymbol{F}_{\boldsymbol{x}} = \partial \boldsymbol{F}/\partial \boldsymbol{x}$ is full rank for all $\boldsymbol{x} \in \mathcal{X}$, so $\mathcal{X}$ is a smooth manifold of dimension $d_{\mathcal{X}} = n_x - n_e$. This assumption is not too restrictive, as geometric singularities can often be removed by judicious mechanical design [26], or using singularity-avoidance constraints [27]–[29]. The assumption is needed to ensure that the tangent space of $\mathcal{X}$ at $\boldsymbol{x}$,

$$\mathcal{T}_{\boldsymbol{x}}\mathcal{X} = \{\dot{\boldsymbol{x}} \in \mathbb{R}^{n_x} : \boldsymbol{F}_{\boldsymbol{x}}\,\dot{\boldsymbol{x}} = \boldsymbol{0}\}, \tag{4}$$

is well-defined and $d_{\mathcal{X}}$-dimensional for any $\boldsymbol{x} \in \mathcal{X}$, a property we shall exploit in the sequel.

We also model the dynamics of the robot using Lagrange's equation with multipliers

$$\boldsymbol{M}(\boldsymbol{q})\,\ddot{\boldsymbol{q}} + \boldsymbol{B}(\boldsymbol{q})^\top \boldsymbol{\lambda} = \boldsymbol{\tau}(\boldsymbol{u}, \boldsymbol{q}, \dot{\boldsymbol{q}}), \tag{5}$$

where $\boldsymbol{M}(\boldsymbol{q})$ is the positive-definite mass matrix of the system, $\boldsymbol{\lambda} \in \mathbb{R}^{n_v}$ is a vector of Lagrange multipliers, $\boldsymbol{u} \in \mathbb{R}^{n_u}$ collects the motor forces and torques, and $\boldsymbol{\tau}(\boldsymbol{u}, \boldsymbol{q}, \dot{\boldsymbol{q}}) \in \mathbb{R}^{n_q}$ accounts for the generalized Coriolis, gravity, friction, and actuation forces.

Since Eq. (5) is a system of n_q equations in $n_q + n_v$ unknowns ($\ddot{\boldsymbol{q}}$ and $\boldsymbol{\lambda}$), we need n_v additional constraints in order to solve it. These can be obtained by taking the time derivative of Eq. (1b), which yields the acceleration constraint

$$\boldsymbol{B}(\boldsymbol{q})\,\ddot{\boldsymbol{q}} = \boldsymbol{\xi}(\boldsymbol{q}, \dot{\boldsymbol{q}}), \tag{6}$$

where $\boldsymbol{\xi}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = -\dot{\boldsymbol{B}}(\boldsymbol{q})\,\dot{\boldsymbol{q}}$. Eqs. (5) and (6) can then be combined into

$$\begin{bmatrix} \boldsymbol{M}(\boldsymbol{q}) & \boldsymbol{B}(\boldsymbol{q})^\top \\ \boldsymbol{B}(\boldsymbol{q}) & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \ddot{\boldsymbol{q}} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\tau}(\boldsymbol{u}, \boldsymbol{q}, \dot{\boldsymbol{q}}) \\ \boldsymbol{\xi}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \end{bmatrix}, \tag{7}$$

and, since $\boldsymbol{F}_{\boldsymbol{x}}$ is full rank, so is $\boldsymbol{B}(\boldsymbol{q})$, and we can write

$$\ddot{\boldsymbol{q}} = \begin{bmatrix} \boldsymbol{I}_{n_q} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{M}(\boldsymbol{q}) & \boldsymbol{B}(\boldsymbol{q})^\top \\ \boldsymbol{B}(\boldsymbol{q}) & \boldsymbol{0} \end{bmatrix}^{-1} \begin{bmatrix} \boldsymbol{\tau}(\boldsymbol{u}, \boldsymbol{q}, \dot{\boldsymbol{q}}) \\ \boldsymbol{\xi}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \end{bmatrix}. \tag{8}$$

By writing the state as $\boldsymbol{x} = (\boldsymbol{q}, \boldsymbol{v})$ and adding the constraint $\dot{\boldsymbol{q}} = \boldsymbol{v}$, Eq. (8) can be expressed in the common form

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t)), \tag{9}$$

where $\boldsymbol{x}(t)$ and $\boldsymbol{u}(t)$ are the state and action trajectories. These trajectories may be further constrained by a path constraint

$$\boldsymbol{h}(\boldsymbol{x}(t), \boldsymbol{u}(t)) \geq \boldsymbol{0} \tag{10}$$

that models, for example, joint or force limits, or collision constraints, and by a boundary constraint

$$\boldsymbol{b}(\boldsymbol{x}(0), \boldsymbol{x}(t_f)) = \boldsymbol{0} \tag{11}$$

that restricts the values taken by $\boldsymbol{x}(t)$ at the initial and final times $t = 0$ and $t = t_f$.

With these definitions, the problem we confront can be posed as follows. Given a running cost function $L(\boldsymbol{x}(t), \boldsymbol{u}(t))$, find trajectories $\boldsymbol{x}(t)$ and $\boldsymbol{u}(t)$, and a final time t_f, that

$$\begin{aligned} \underset{\boldsymbol{x}(\cdot), \boldsymbol{u}(\cdot), t_f}{\text{minimize}} \quad & \int_0^{t_f} L(\boldsymbol{x}(t), \boldsymbol{u}(t))\,\mathrm{d}t && && \text{(12a)} \\ \text{subject to} \quad & \boldsymbol{h}(\boldsymbol{x}(t), \boldsymbol{u}(t)) \geq \boldsymbol{0}, && t \in [0, t_f], && \text{(12b)} \\ & \dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t)), && t \in [0, t_f]. && \text{(12c)} \\ & \boldsymbol{b}(\boldsymbol{x}(0), \boldsymbol{x}(t_f)) = \boldsymbol{0}, && && \text{(12d)} \\ & t_f \geq 0. && && \text{(12e)} \end{aligned}$$

Note that Eq. (2) is not added to this formulation because it is already accounted for implicitly by Eq. (12c). However, our goal will be to obtain solutions of this problem that satisfy both (2) and (12c) as accurately as possible.

III. Transcription techniques

A. Problem discretization

In this paper we solve Problem (12) by transcribing it into an NLP problem, which entails approximating all functionals in (12a)-(12c) by functions of discrete states and actions. To this end, we discretize the time horizon $[0, t_f]$ into N intervals defined by the time instants

$$t_0, \ldots, t_k, \ldots, t_N,$$

where $t_0 = 0$ and $t_N = t_f$, and represent $\boldsymbol{x}(t)$ and $\boldsymbol{u}(t)$ by the $N+1$ states

$$\boldsymbol{x}_0, \ldots, \boldsymbol{x}_k, \ldots, \boldsymbol{x}_N$$

and actions

$$\boldsymbol{u}_0, \ldots, \boldsymbol{u}_k, \ldots, \boldsymbol{u}_N$$

at those instants. The values $t_0, \ldots, t_N$ are known as the knot points, and for simplicity we assume them to be uniformly spaced, so $\Delta t = t_{k+1} - t_k$ takes the same value for $k = 0, \ldots, N-1$. If Δt is constant, the time horizon $[0, t_f]$ is fixed, but a variable time horizon can also be allowed by treating Δt as a decision variable of the problem [21].

The transcriptions of Eqs. (12a) and (12b) are relatively straightforward and less relevant for our purposes. They can be done, for example, by approximating the integral in Eq. (12a) using some quadrature rule, and by setting $\boldsymbol{h}(\boldsymbol{x}_k, \boldsymbol{u}_k) \geq \boldsymbol{0}$ for $k = 0, \ldots, N$. The transcription of (12c), in contrast, is substantially more involved, and will be the main subject of the rest of this paper. We next recall background techniques to carry it out, which will be needed hereafter.

B. Transcription of differential constraints

To approximate Eq. (12c), the first step is to define $\boldsymbol{u}(t)$ in terms of $\boldsymbol{u}_0,\dots,\boldsymbol{u}_N$. While many choices are possible here, we use a first-order hold filter due to its good balance between simplicity and accuracy of representation. For all $t\in[t_k,t_{k+1}]$ it will thus be

$$\boldsymbol{u}(t)=\boldsymbol{u}_k+\frac{t-t_k}{\Delta t}\cdot(\boldsymbol{u}_{k+1}-\boldsymbol{u}_k). \tag{13}$$

The second step is to determine the state $\boldsymbol{x}_{k+1}$ that would be reached from $\boldsymbol{x}_k$ under the actions in Eq. (13). This can be done in many ways, but in this paper we opt for the Gauss-Legendre collocation scheme because it has the lowest integration error for a fixed number of dynamics function evaluations [22]. Our methods, however, should be easy to adapt to other collocation schemes [6, 21].

Transcription via Gauss-Legendre collocation works as follows. The form of $\boldsymbol{x}(t)$ in the interval $[t_k,t_{k+1}]$ is not known a priori, but we assume it to be approximated by a polynomial of degree d taking the value $\boldsymbol{x}_k$ for $t=t_k$. This polynomial is defined as the one that interpolates $d+1$ states

$$\boldsymbol{x}_{k,0},\dots,\boldsymbol{x}_{k,d},$$

corresponding to $d+1$ time instants

$$t_{k,0},\dots,t_{k,d}$$

in the interval $[t_k,t_{k+1}]$ (Fig. 2), where $t_{k,0}=t_k$, and

$$t_{k,0}<t_{k,1}<\cdots<t_{k,d}. \tag{14}$$

Using Lagrange's interpolation formula [30], we thus can write this polynomial as

$$\boldsymbol{p}_k(t,\boldsymbol{c}_k)=\boldsymbol{x}_{k,0}\cdot\ell_0(t-t_k)+\cdots+\boldsymbol{x}_{k,d}\cdot\ell_d(t-t_k), \tag{15}$$

where

$$\boldsymbol{c}_k=(\boldsymbol{x}_{k,0},\dots,\boldsymbol{x}_{k,d}) \tag{16}$$

is the vector of polynomial coefficients, and

$$\ell_0(t-t_k),\dots,\ell_d(t-t_k) \tag{17}$$

is the basis of Lagrange polynomials of degree d defined for $t\in[t_k,t_{k+1}]$. Recall that these polynomials only depend on $t_{k,0},\dots,t_{k,d}$, and that

$$\ell_j(t_{k,i}-t_k)=\begin{cases}1, & \text{if } j=i\\ 0, & \text{otherwise}\end{cases} \tag{18}$$

so $\boldsymbol{p}_k(t_{k,i},\boldsymbol{c}_k)=\boldsymbol{x}_{k,i}$ [30]. Since we wish $\boldsymbol{p}_k(t_k,\boldsymbol{c}_k)=\boldsymbol{x}_k$, it must be $\boldsymbol{x}_{k,0}=\boldsymbol{x}_k$ in Eq. (15), and we shall assume so hereafter. The remaining coefficients $\boldsymbol{x}_{k,1},\dots,\boldsymbol{x}_{k,d}$ are unknown, but they can be determined by forcing $\boldsymbol{p}_k(t,\boldsymbol{c}_k)$ to satisfy $\dot{\boldsymbol{x}}(t)=\boldsymbol{f}(\boldsymbol{x}(t),\boldsymbol{u}(t))$ for all $t=t_{k,1},\dots,t_{k,d}$. This yields the d collocation constraints

$$\dot{\boldsymbol{p}}_k(t_{k,i},\boldsymbol{c}_k)=\boldsymbol{f}(\boldsymbol{x}_{k,i},\boldsymbol{u}_{k,i}),\qquad i=1,\dots,d, \tag{19}$$

where $\boldsymbol{u}_{k,i}=\boldsymbol{u}(t_{k,i})$. The values $t_{k,1},\dots,t_{k,d}$ are called the collocation points, and correspond to the roots of the Legendre polynomial of degree d [6]. The left-hand side of Eq. (19), in turn, is easy to formulate, as $\dot{\boldsymbol{p}}_k(t_{k,i},\boldsymbol{c}_k)$ can be written as

$$\dot{\boldsymbol{p}}_k(t_{k,i},\boldsymbol{c}_k)=\boldsymbol{J}(t_{k,0},\dots,t_{k,d})\cdot\boldsymbol{c}_k, \tag{20}$$

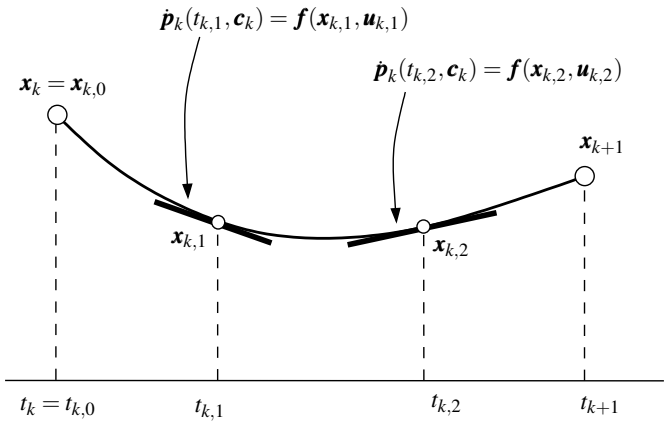


Fig. 2. In the interval $[t_k,t_{k+1}]$, the state trajectory $\boldsymbol{x}(t)$ is approximated by a polynomial of degree d that interpolates $d+1$ states $\boldsymbol{x}_{k,0},\dots,\boldsymbol{x}_{k,d}$, where $\boldsymbol{x}_{k,0}=\boldsymbol{x}_k$ (the figure depicts the case $d=2$). The derivative of this polynomial must match the system dynamics at the collocation points $t_{k,1},\dots,t_{k,d}$.

where $\boldsymbol{J}$ is a constant differentiation matrix that solely depends on $t_{k,0},\dots,t_{k,d}$ [6, 30]. The state $\boldsymbol{x}_{k+1}$ is finally given by

$$\boldsymbol{x}_{k+1}=\boldsymbol{p}_k(t_{k+1},\boldsymbol{c}_k). \tag{21}$$

In each interval $[t_k,t_{k+1}]$, therefore, Eq. (12c) is transcribed into Eqs. (19) and (21), where $\boldsymbol{p}_k(t,\boldsymbol{c}_k)$ is defined by Eq. (15). Such a transcription is very precise, as the local integration error of the Gauss-Legendre scheme is $O(\Delta t^{2d+1})$ [22]. In the end, once the transcribed problem is solved, the coefficients $\boldsymbol{c}_k$ will be known, and the trajectory $\boldsymbol{x}(t)$ will be the spline defined by $\boldsymbol{p}_0(t,\boldsymbol{c}_0),\dots,\boldsymbol{p}_{N-1}(t,\boldsymbol{c}_{N-1})$.

IV. Previous Transcription Methods

We next review the three principal methods for transcribing Problem (12) given so far in the literature, for the case of constrained mechanical systems (Sections IV-A to IV-C).

A. Basic collocation

A direct way of transcribing Problem (12) consists in applying the methods in Section III to each one of its equations. This results in the optimization problem

$$\begin{aligned}
\underset{\boldsymbol{w}}{\text{minimize}}\quad & C(\boldsymbol{w}) && && \text{(22a)}\\
\text{subject to}\quad & \boldsymbol{h}(\boldsymbol{x}_k,\boldsymbol{u}_k)\geq\boldsymbol{0}, && k=0,\dots,N, && \text{(22b)}\\
& \dot{\boldsymbol{p}}_k(t_{k,i},\boldsymbol{c}_k)=\boldsymbol{f}(\boldsymbol{x}_{k,i},\boldsymbol{u}_{k,i}), && k=0,\dots,N-1, && \text{(22c)}\\
& && i=1,\dots,d, &&\\
& \boldsymbol{p}_k(t_{k+1},\boldsymbol{c}_k)=\boldsymbol{x}_{k+1}, && k=0,\dots,N-1, && \text{(22d)}\\
& \boldsymbol{b}(\boldsymbol{x}_0,\boldsymbol{x}_N)=\boldsymbol{0}, && && \text{(22e)}
\end{aligned}$$

where $C(\boldsymbol{w})$ is the discrete version of the integral cost in Eq. (12a), and $\boldsymbol{w}$ encompasses all the decision variables intervening: the actions $\boldsymbol{u}_0,\dots,\boldsymbol{u}_N$, the states $\boldsymbol{x}_0,\dots,\boldsymbol{x}_N$, and the collocation states $\boldsymbol{x}_{k,i}$ for $k=0,\dots,N-1$, $i=1,\dots,d$. This transcription will be called the basic method hereafter.

The basic method is simple, but it presents an important limitation. As shown in Fig. 1(a), the knot states $\boldsymbol{x}_0,\dots,\boldsymbol{x}_N$ tend to drift off from $\mathcal{X}$ due to the discretization errors inherent

to Eqs. (22c) and (22d), which leads to unrealistic trajectories that can be difficult to control. Unfortunately, this problem cannot be solved by just adding

$$\boldsymbol{F}(\boldsymbol{x}_k) = \boldsymbol{0}, \quad k = 0, \ldots, N, \tag{23}$$

to the transcribed formulation, as Eqs. (22c) and (22d) already determine $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$ when $\boldsymbol{x}_0$ is known and a control sequence is fixed. Therefore, the addition of (23) would introduce redundant constraints, thus violating the linear independence constraint qualification (LICQ) required by the Karush-Kuhn-Tucker conditions of optimality [31]. Despite the problem of drift, however, the simplicity of the basic method makes it a good tool to provide initial guesses for more elaborate methods.

B. Collocation with Baumgarte stabilization

An alternative to the basic method is to resort to Baumgarte stabilization. This technique consists in altering the system dynamics by adding artificial forces to damp the trajectory drift from $\mathcal{X}$. To achieve so, Eq. (6) is modified by adding stabilizing terms for the residuals of Eqs. (1a) and (1b). The modified equation takes the form

$$\boldsymbol{B}(\boldsymbol{q})\,\ddot{\boldsymbol{q}} + \alpha\,\boldsymbol{B}(\boldsymbol{q})\dot{\boldsymbol{q}} + \beta \begin{bmatrix} \boldsymbol{\Phi}(\boldsymbol{q}) \\ \boldsymbol{0}_{n_r} \end{bmatrix} - \boldsymbol{\xi}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \boldsymbol{0}, \tag{24}$$

where $\boldsymbol{0}_{n_r}$ is a column vector of n_r zeros, and α and β are constant paramaters that have to be tuned to ensure a stable trajectory near $\mathcal{X}$. Following steps analogous to those in Eqs. (7)-(9) we then obtain a stabilized dynamics equation

$$\dot{\boldsymbol{x}} = \boldsymbol{f}_{\text{stab}}(\boldsymbol{x}, \boldsymbol{u}). \tag{25}$$

Using Baumgarte stabilization, thus, the transcription in Problem (22) is improved by replacing Eq. (22c) by

$$\dot{\boldsymbol{p}}_k(t_{k,i}, \boldsymbol{c}_k) = \boldsymbol{f}_{\text{stab}}(\boldsymbol{x}_{k,i}, \boldsymbol{u}_{k,i}) \tag{26}$$

for $k = 0, \ldots, N-1$ and $i = 1, \ldots, d$.

The simplicity of the technique has made it a common approach to mitigate the problem of drift, but the method presents three main drawbacks. First, the tuning of α and β is not trivial, as the dynamics from the stabilizing terms should be faster than the drift dynamics produced by the transcribed equations, otherwise the trajectory might depart from $\mathcal{X}$ in an unstable manner. Despite the importance of this point, only heuristic rules have been given to choose α and β so far [18, 19]. Second, Eq. (25) has a more complex structure than the one in Eq. (9), which complicates the computation of gradients and Hessians needed by the optimizer. Third, and most notably, the approach alters the dynamics of the system artificially, as the use of Eq. (26), instead of Eq. (22c), implies that the actual dynamics will not be fulfilled at the collocation points. This point is illustrated in Section VII with the help of examples.

C. The PKT method

In [7], Posa et al. modify the classical Hermite-Simpson collocation method [32] to deal with constrained systems. In their case, the system can only have holonomic constraints (so $n_v = n_p$ and $d_{\mathcal{X}} = n_x - 2n_p$) and the trajectories are approximated by polynomials of third degree. We next review this approach, which we refer to as the PKT method.

As in [32], the PKT method approximates the trajectory for $t \in [t_k, t_{k+1}]$ by a cubic polynomial $\boldsymbol{p}_k(t, \boldsymbol{c}_k)$. This polynomial interpolates $\boldsymbol{x}_k$ and $\boldsymbol{x}_{k+1}$ while satisfying the slopes $\dot{\boldsymbol{x}}_k$ and $\dot{\boldsymbol{x}}_{k+1}$ determined by the collocation constraints

$$\dot{\boldsymbol{x}}_k = \boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{u}_k), \tag{27a}$$
$$\dot{\boldsymbol{x}}_{k+1} = \boldsymbol{f}(\boldsymbol{x}_{k+1}, \boldsymbol{u}_{k+1}). \tag{27b}$$

In terms of $\boldsymbol{c}_k = (\boldsymbol{x}_k, \boldsymbol{x}_{k+1})$, the polynomial $\boldsymbol{p}_k(t, \boldsymbol{c}_k)$ can be written as follows

$$\begin{aligned} \boldsymbol{p}_k(t, \boldsymbol{c}_k) = \boldsymbol{x}_k + \boldsymbol{f}_k(t - t_k) - \\ - \tfrac{(t-t_k)^2}{\Delta t^2}(3\boldsymbol{x}_k - 3\boldsymbol{x}_{k+1} + 2\Delta t\, \boldsymbol{f}_k + \Delta t\, \boldsymbol{f}_{k+1}) + \\ + \tfrac{(t-t_k)^3}{\Delta t^3}(2\boldsymbol{x}_k - 2\boldsymbol{x}_{k+1} + \Delta t\, \boldsymbol{f}_k + \Delta t\, \boldsymbol{f}_{k+1}), \end{aligned} \tag{28}$$

where $\boldsymbol{f}_k = \boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{u}_k)$. To determine the value of $\boldsymbol{x}_{k+1}$ in $\boldsymbol{p}_k(t, \boldsymbol{c}_k)$, a third collocation constraint is imposed at the midpoint $t_{k,c} = \frac{1}{2}(t_k + t_{k+1})$, which can be expressed as

$$\dot{\boldsymbol{p}}_k(t_{k,c}, \boldsymbol{c}_k) = \boldsymbol{f}(\boldsymbol{x}_{k,c}, \boldsymbol{u}_{k,c}), \tag{29}$$

where $\boldsymbol{x}_{k,c} = \boldsymbol{p}(t_{k,c}, \boldsymbol{c}_k)$ and $\boldsymbol{u}_{k,c} = \frac{1}{2}(\boldsymbol{u}_k + \boldsymbol{u}_{k+1})$. Since Eqs. (27) have already been already been taken into account in Eq. (28), Eq. (29) is the sole constraint needed to transcribe the dynamics in the usual Hermite-Simpson method [32].

Posa et al. note in [7] that, while Eq. (29) poses no challenge in unconstrained systems, it is problematic when holonomic constraints are present. In the former case the n_x components of $\boldsymbol{x}_{k+1}$ are independent, so the n_x conditions in (29) properly determine $\boldsymbol{x}_{k+1}$. In the latter case, however, one must impose

$$\boldsymbol{F}(\boldsymbol{x}_k) = \boldsymbol{0}, \qquad k = 0, \ldots, N, \tag{30}$$

to ensure all knot states lie on $\mathcal{X}$, so $\boldsymbol{x}_{k+1}$ will only have $d_{\mathcal{X}} = n_x - 2n_p$ freedoms and Eq. (29) will overconstrain its value. To circumvent this problem, the PKT method relaxes Eq. (29) to include $2n_p$ new variables for each time interval, $\boldsymbol{\gamma}_{k,c} \in \mathbb{R}^{n_p}$ and $\boldsymbol{\lambda}_{k,c} \in \mathbb{R}^{n_p}$, which provide the required freedoms to eliminate the overconstraint. Specifically, [7] reformulates (29) as

$$\dot{\boldsymbol{q}}_k(t_{k,c}, \boldsymbol{c}_k) = \boldsymbol{v}_k(t_{k,c}, \boldsymbol{c}_k) + \boldsymbol{B}(\boldsymbol{q}_{k,c})^\top \cdot \boldsymbol{\gamma}_{k,c}, \tag{31a}$$
$$\dot{\boldsymbol{v}}_k(t_{k,c}, \boldsymbol{c}_k) = \boldsymbol{g}(\boldsymbol{x}_{k,c}, \boldsymbol{u}_{k,c}, \boldsymbol{\lambda}_{k,c}), \tag{31b}$$

where $\boldsymbol{q}_k(t, \boldsymbol{c}_k)$ and $\boldsymbol{v}_k(t, \boldsymbol{c}_k)$ refer to the configuration and velocity components of $\boldsymbol{p}_k(t, \boldsymbol{c}_k)$, and

$$\boldsymbol{g}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\lambda}) = \boldsymbol{M}(\boldsymbol{q})^{-1} \cdot \left[\boldsymbol{\tau}(\boldsymbol{u}, \boldsymbol{q}, \dot{\boldsymbol{q}}) - \boldsymbol{B}(\boldsymbol{q})^\top \boldsymbol{\lambda} \right] \tag{32}$$

is the acceleration function determined by Eq. (5). Under the PKT method, therefore, the transcription in (22) is improved by adding the constraints in (30) and replacing (22c) by (31) for $k = 0, \ldots, N-1$.

The strong point of the PKT method is that it keeps the knot states $\boldsymbol{x}_0, \ldots, \boldsymbol{x}_N$ on $\mathcal{X}$, while ensuring, through (27), that

such states satisfy the actual dynamics of the system. This is a clear improvement over the Baumgarte method, which only keeps $\boldsymbol{x}_0,\ldots,\boldsymbol{x}_N$ near $\mathcal{X}$ approximately, and violates the actual dynamics at the collocation points. The main weakness of the PKT method, however, is that it modifies the system dynamics at the midpoint of each interval. In particular, Eq. (31b), relaxes the second order dynamics constraint, as it only accounts for Lagrange's equation in (5) but not for the acceleration constraint in (6). This implies that, at each midpoint $t_{k,c}$, the robot is treated as an unconstrained system subject to the forces defined by the $\boldsymbol{\lambda}_{k,c}$ multipliers (see [33] which interprets these multipliers as external forces applied at the breakpoints of kinematic loops). Thus, while the collocation constraints at $t_0,\ldots,t_N$ are exactly enforced, those at the midpoints $t_{0,c},\ldots,t_{N-1,c}$ are not. Additional limitations of the PKT method are that, as defined in [7], it can only handle holonomic systems, and it uses approximation polynomials of a fixed degree. The latter, in particular, impedes its application in hp-adaptive meshing schemes.

V. New transcription methods

We next propose two new methods to solve the problem of drift. The two methods, which are referred to as the projection and local coordinates methods, satisfy the actual system dynamics at the collocation points, and they can use approximation polynomials of arbitrary degree. While the projection method guarantees that the knot states $\boldsymbol{x}_0,\ldots,\boldsymbol{x}_N$ lie on $\mathcal{X}$ exactly (Section V-A), the local coordinates method achieves full drift elimination over the entire trajectory $\boldsymbol{x}(t)$ (Section V-B).

A. The projection method

The projection method cancels the drift at each knot point by projecting the end state of each Lagrange interpolation orthogonally to $\mathcal{X}$. The method only requires modifying Problem (22) as follows. First, we add the constraints

$$\boldsymbol{F}(\boldsymbol{x}_k)=\boldsymbol{0}, \qquad k=0,\ldots,N, \tag{33}$$

to ensure all knot states $\boldsymbol{x}_0,\ldots,\boldsymbol{x}_N$ lie on $\mathcal{X}$. Second, we add N auxiliary states

$$\boldsymbol{x}'_1,\ldots,\boldsymbol{x}'_N,$$

and n_e-dimensional vectors

$$\boldsymbol{\mu}_1,\ldots,\boldsymbol{\mu}_N,$$

to the decision variables $\boldsymbol{w}$. Finally, we replace Eq. (22d) by

$$\boldsymbol{x}'_{k+1}=\boldsymbol{p}_k(t_{k+1},\boldsymbol{c}_k) \qquad k=0,\ldots,N-1, \tag{34}$$

so $\boldsymbol{x}'_{k+1}$ is the end state of the Lagrange interpolation, and add the constraint

$$\boldsymbol{x}_{k+1}=\boldsymbol{x}'_{k+1}+\boldsymbol{F}_{\boldsymbol{x}}(\boldsymbol{x}_{k+1})^\top\boldsymbol{\mu}_k \qquad k=0,\ldots,N-1 \tag{35}$$

to ensure that $\boldsymbol{x}_{k+1}$ is the orthogonal projection of $\boldsymbol{x}'_{k+1}$ on $\mathcal{X}$ (Fig. 3).

Note that the rows of $\boldsymbol{F}_{\boldsymbol{x}}$ provide a basis of the normal space of $\mathcal{X}$ at $\boldsymbol{x}_{k+1}$, so $\boldsymbol{F}_{\boldsymbol{x}}(\boldsymbol{x}_{k+1})^\top\boldsymbol{\mu}_k$ will be a normal vector of $\mathcal{X}$ at $\boldsymbol{x}_{k+1}$. Thus, although $\boldsymbol{x}'_1,\ldots,\boldsymbol{x}'_N$ may deviate from $\mathcal{X}$, the drift will not accumulate because the joint effect of Eqs. (33) and (35) will fully remove it after every time step. Actually, the local integration error before the projection is $O(\Delta t^{2d+1})$, and Hairer shows that the projection step does not affect this bound [22, 23].

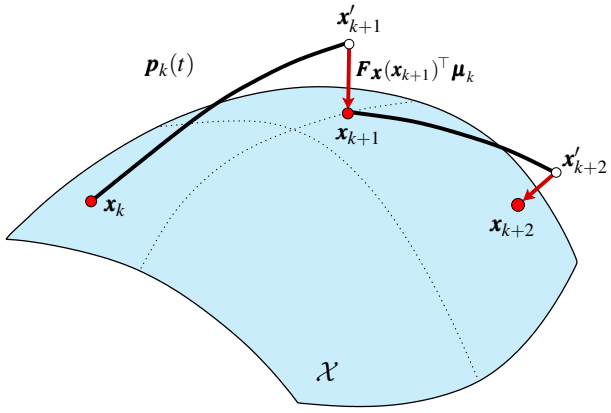


Fig. 3. The projection method. The end state of each polynomial is projected orthogonally to $\mathcal{X}$ to eliminate the accumulated drift.

Regarding the problem of drift, both the projection and the PKT methods achive a similar result, since both ensure $\boldsymbol{x}_1,\ldots,\boldsymbol{x}_N$ lie on $\mathcal{X}$ exactly. However, the projection method is advantageous in that it fulfills the actual dynamics at all collocation points, and its approximation polynomial, $\boldsymbol{p}_k(t,\boldsymbol{c}_k)$, can be of arbitrary degree. The weak point of the projection method is that, as in the PKT method, some drift from $\mathcal{X}$ will persist despite the projections. If a driftless continuous trajectory is required, however, we can always resort to the following method.

B. The local coordinates method

The idea of this method is to transcribe Eq. (12c) by using integration in local coordinates [22, 24, 25]. In trajectory optimization, this concept was applied to $SO(3)$ and multiple shooting in [34], and we here extend it to general manifolds using collocation methods. We first explain the idea using generic maps (Section V-B.1) and then give the collocation equations for particular choices of such maps (Section V-B.2).

B.1. Collocation in local coordinates

Let $\boldsymbol{y}=\boldsymbol{\varphi}_k(\boldsymbol{x})$ be a local chart of $\mathcal{X}$ defined at some point $\boldsymbol{x}_k\in\mathcal{X}$. This implies that $\boldsymbol{\varphi}_k$ is a local diffeomorphism

$$\boldsymbol{\varphi}_k : V_k \to P_k,$$

where V_k and P_k are open neighbourhoods in $\mathcal{X}$ and $\mathbb{R}^{d_{\mathcal{X}}}$, respectively, including $\boldsymbol{x}_k$ and $\boldsymbol{\varphi}_k(\boldsymbol{x}_k)$. Without loss of generality, we will assume that $\boldsymbol{\varphi}_k(\boldsymbol{x}_k)=\boldsymbol{0}$, so $\boldsymbol{\varphi}_k$ maps V_k to a neighborhood of the origin of $\mathbb{R}^{d_{\mathcal{X}}}$ (Fig. 4). Since $\boldsymbol{\varphi}_k$ is a diffeomorphism, it has an inverse map

$$\boldsymbol{\psi}_k : P_k \to V_k,$$

which provides a local parametrization of $V_k\subset\mathcal{X}$. The fact that $\mathcal{X}$ is a smooth manifold guarantees that both $\boldsymbol{\varphi}_k$ and $\boldsymbol{\psi}_k$ exist for any point $\boldsymbol{x}_k\in\mathcal{X}$ [35].

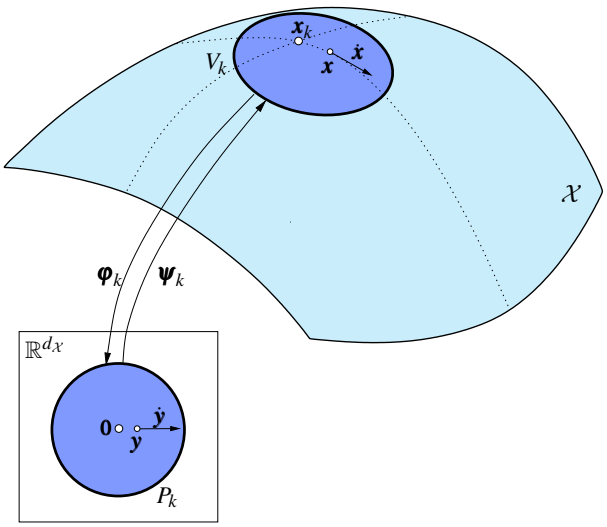


Fig. 4. A chart of $\mathcal{X}$ is a map φ_k from an open neighbourhood V_k of $\mathcal{X}$ to an open neighbourhood P_k of $\mathbb{R}^{d_\mathcal{X}}$. The inverse map ψ_k gives a local parametrization of $V_k \subset \mathcal{X}$.

We next see that, using φ_k and ψ_k, we can compute the state x_{k+1} that would be reached from x_k while ensuring that $F(x_{k+1}) = 0$ precisely. To this end, we first take the time derivative of

$$y = \varphi_k(x) \tag{36}$$

and obtain

$$\dot{y} = S(x) \cdot \dot{x}, \tag{37}$$

where $S(x) = \partial \varphi_k(x)/\partial x$. If we substitute $\dot{x} = f(x,u)$ in Eq. (37), we get

$$\dot{y} = S(x) \cdot f(x,u), \tag{38}$$

and using $x = \psi_k(y)$ we arrive at

$$\dot{y} = S(\psi_k(y)) \cdot f(\psi_k(y),u), \tag{39}$$

which we compactly write as

$$\dot{y} = g_k(y,u). \tag{40}$$

Eq. (40) provides the dynamic equation in y coordinates, and we can use it as follows to compute x_{k+1} [Fig. 5(a)]: using φ_k, we first map x_k to the origin of $\mathbb{R}^{d_\mathcal{X}}$; we then integrate Eq. (40) to find the state $y(t_{k+1}) = y_{k+1}$ that would be reached from $y(t_k) = 0$ under the actions $u(t)$; and, finally, we project y_{k+1} back to $\mathcal{X}$ using

$$x_{k+1} = \psi_k(y_{k+1}). \tag{41}$$

Using this process, it is clear that $x_{k+1} \in \mathcal{X}$ by construction.

To integrate Eq. (40) using collocation, we assume that the trajectory $y(t)$ is well approximated by a polynomial that starts at the origin of $\mathbb{R}^{d_\mathcal{X}}$ and interpolates d unknown states

$$y_{k,1}, \ldots, y_{k,d}$$

corresponding to the Gauss-Legendre collocation times

$$t_{k,1} < \cdots < t_{k,d} \tag{42}$$

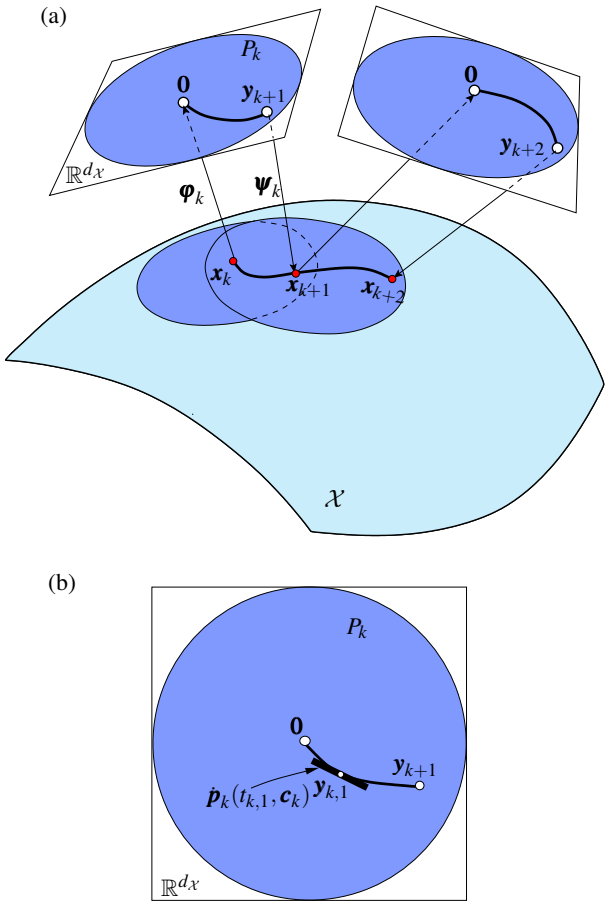


Fig. 5. Transcription using local coordinates. (a) To obtain x_{k+1}, we map x_k to $\varphi_k(x_k) = 0 \in \mathbb{R}^{d_\mathcal{X}}$, then integrate Eq. (40) to find y_{k+1}, and finally project y_{k+1} to $\mathcal{X}$ using ψ_k. (b) The collocation constraint is now enforced in $\mathbb{R}^{d_\mathcal{X}}$. In the figure, we assume $d = 1$, so $\dot{p}_k(t_{k,1})$ must match $g_k(y_{k,1}, u(t_{k,1}))$.

from $[t_k, t_{k+1}]$ [Fig. 5(b)]. This polynomial can be written as

$$p_k(t, c_k) = y_{k,1} \cdot \ell_1(t - t_k) + \cdots + y_{k,d} \cdot \ell_d(t - t_k), \tag{43}$$

where now

$$c_k = (y_{k,1}, \ldots, y_{k,d}). \tag{44}$$

Note that the polynomial in Eq. (43) satisfies $p_k(t_k, c_k) = 0$ as required since, due to Eq. (18), $\ell_1(t - t_k), \ldots, \ell_d(t - t_k)$ are all zero for $t = t_k$. To determine $y_{k,1}, \ldots, y_{k,d}$, we only have to impose the d collocation constraints

$$\dot{p}_k(t_{k,i}, c_k) = g_k(y_{k,i}, u_{k,i}), \quad \text{for} \quad i = 1, \ldots, d. \tag{45}$$

The value of y_{k+1} will then be given by

$$y_{k+1} = p_k(t_{k+1}, c_k), \tag{46}$$

so Eq. (41) can be written as

$$x_{k+1} = \psi_k(p_k(t_{k+1}, c_k)). \tag{47}$$

With the earlier procedure, thus, we can achieve a drift-free transcription of Problem (12) by replacing Eqs. (22c) and (22d) by

$$\dot{\boldsymbol{p}}_k(t_{k,i},\boldsymbol{c}_k) = \boldsymbol{g}_k(\boldsymbol{y}_{k,i},\boldsymbol{u}_{k,i}), \quad k=0,\dots,N-1, \quad i=1,\dots,d, \tag{48a}$$

$$\boldsymbol{x}_{k+1} = \boldsymbol{\psi}_k(\boldsymbol{p}_k(t_{k+1},\boldsymbol{c}_k)), \quad k=0,\dots,N-1. \tag{48b}$$

Once the transcribed problem is solved we can use $\boldsymbol{p}_0(t,\boldsymbol{c}_0),\dots,\boldsymbol{p}_{N-1}(t,\boldsymbol{c}_{N-1})$ to construct a continuous spline for $\boldsymbol{x}(t)$ lying in $\mathcal{X}$ for all t, something that the earlier methods were unable to obtain. This spline will be given by

$$\boldsymbol{\psi}_0(\boldsymbol{p}_0(t,\boldsymbol{c}_0)),\dots,\boldsymbol{\psi}_{N-1}(\boldsymbol{p}_{N-1}(t,\boldsymbol{c}_{N-1})). \tag{49}$$

B.2. Collocation in tangent space coordinates

For some manifolds, the maps $\boldsymbol{\psi}_k$ and $\boldsymbol{\varphi}_k$ can be defined in closed form (for example, by expressing some variables of Eq. (2) as a function of the others). For the sake of generality, however, we here define them using tangent space coordinates, which work for any manifold [22, 25]. Using these coordinates, the map $\boldsymbol{y}=\boldsymbol{\varphi}_k(\boldsymbol{x})$ is obtained by projecting $\boldsymbol{x}$ orthogonally onto $\mathcal{T}_{\boldsymbol{x}_k}\mathcal{X}=\mathbb{R}^{d_\mathcal{X}}$, as shown in Fig 6, and takes the form

$$\boldsymbol{y} = \boldsymbol{U}_k(\boldsymbol{x}_k)^\top\,(\boldsymbol{x}-\boldsymbol{x}_k), \tag{50}$$

where $\boldsymbol{U}_k(\boldsymbol{x}_k)$ is an $n_x\times d_\mathcal{X}$ matrix whose columns provide an orthonormal basis of $\mathcal{T}_{\boldsymbol{x}_k}\mathcal{X}$ (see Appendix A). The inverse map $\boldsymbol{x}=\boldsymbol{\psi}_k(\boldsymbol{y})$ is implicitly determined by the system of nonlinear equations

$$\begin{cases} \boldsymbol{y}-\boldsymbol{U}_k(\boldsymbol{x}_k)^\top(\boldsymbol{x}-\boldsymbol{x}_k)=\boldsymbol{0}, & \text{(51a)}\\ \boldsymbol{F}(\boldsymbol{x})=\boldsymbol{0}, & \text{(51b)}\end{cases}$$

which will be written as

$$\boldsymbol{G}_k(\boldsymbol{x},\boldsymbol{y}) = \boldsymbol{0} \tag{52}$$

for short. The time derivative of Eq. (50) provides the particular form of Eq. (37),

$$\dot{\boldsymbol{y}} = \boldsymbol{U}_k(\boldsymbol{x}_k)^\top\,\dot{\boldsymbol{x}}, \tag{53}$$

where $\boldsymbol{U}_k(\boldsymbol{x}_k)^\top$ corresponds to $\boldsymbol{S}(\boldsymbol{x})$. Eq. (38) is then given by

$$\dot{\boldsymbol{y}} = \boldsymbol{U}_k(\boldsymbol{x}_k)^\top\,\boldsymbol{f}(\boldsymbol{x},\boldsymbol{u}). \tag{54}$$

Since the $\boldsymbol{\psi}_k$ map is only defined implicitly by Eq. (52), we cannot obtain Eq. (39) explicitly using these coordinates. However, using Eq. (52), we can still impose Eq. (48a) via

$$\boldsymbol{G}_k(\boldsymbol{x}_{k,i},\boldsymbol{y}_{k,i}) = \boldsymbol{0}, \quad k=0,\dots,N-1, \quad i=1,\dots,d, \tag{55a}$$

$$\dot{\boldsymbol{p}}_k(t_{k,i},\boldsymbol{c}_k) = \boldsymbol{U}_k(\boldsymbol{x}_k)^\top\boldsymbol{f}(\boldsymbol{x}_{k,i},\boldsymbol{u}_{k,i}), \quad k=0,\dots,N-1, \quad i=1,\dots,d, \tag{55b}$$

and Eq. (48b) via

$$\boldsymbol{G}_k(\boldsymbol{x}_{k+1},\boldsymbol{p}_k(t_{k+1},\boldsymbol{c}_k)) = \boldsymbol{0}, \quad k=0,\dots,N-1, \tag{56}$$

where $\boldsymbol{y}_{k,1},\dots,\boldsymbol{y}_{k,d}$ must be added to the decision variables $\boldsymbol{w}$ of the problem.

In [36], constraints are considered that help to delimit the domains in which the local parameterizations are valid. These constraints could be added to the transcription if necessary, but we omit them to simplify the presentation, and because they are seldom useful when the mesh of knot points is dense enough.

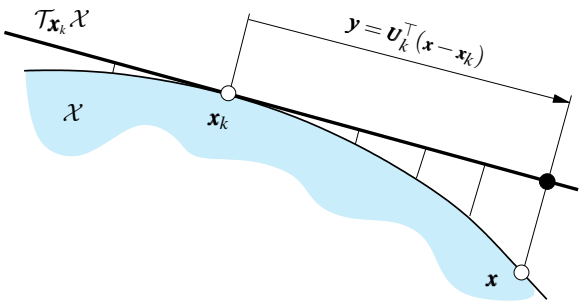


Fig. 6. The tangent space parametrization. The map $\boldsymbol{\varphi}_k$ is defined by the projection of $\boldsymbol{x}$ onto the tangent space $\mathcal{T}_{\boldsymbol{x}_k}\mathcal{X}$.

The local error of this method is the one of Gauss-Legendre collocation, i.e. $O(\Delta t^{2d+1})$ (Section III-B), but in tangent space. For multi-step methods it has been proved that, under mild conditions, the mapping back to the manifold does not increase the order of this error [37]. This result probably also holds in general, and in fact is taken for granted in [18], but providing a formal proof of this point is left out of the scope of this paper.

VI. Implementation

To compare the techniques in Section V with those in Section IV, we have implemented them using MATLAB and the symbolic toolbox for nonlinear optimization and algorithmic differentiation CasADi [38]. CasADi provides the necessary means to formulate the problems and to compute the gradients and Hessians of the transcribed equations using automatic differentiation. These derivatives are necessary to solve the NLP problems that result, a task for which we rely on the interior-point solver IPOPT [39] in conjunction with the linear solver MA-27 [40]. Our implementation can be accessed through [41].

We next discuss important aspects that must be considered when implementing and solving the transcribed problems, regardless of the software platform employed.

A. Explicit versus implicit dynamics

In all transcriptions so far, the collocation constraints have been formulated using the explicit form of the system dynamics in Eq. (9). The derivation of Eq. (9), however, requires finding the matrix inverse in Eq. (8), which often complicates the expressions of the gradients and Hessians needed by the optimizer. Unless such an inverse is simple enough, it may be preferable to write the collocation constraints using the implicit form of the dynamics given by Eq. (7). This is easy to do, as it only requires the substitution of each term $\boldsymbol{f}(\boldsymbol{x}_{k,i},\boldsymbol{u}_{k,i})$ appearing in a collocation constraint by a new variable $\dot{\boldsymbol{x}}_{k,i}$ subject to

$$\boldsymbol{D}(\boldsymbol{x}_{k,i},\dot{\boldsymbol{x}}_{k,i},\boldsymbol{u}_{k,i},\boldsymbol{\lambda}_{k,i}) = \boldsymbol{0}, \tag{57}$$

TABLE I
SUMMARY OF THE VARIABLES AND EQUATIONS INTRODUCED BY THE FIVE METHODS USING THE IMPLICIT FORMULATION

Method	Variables / Equations	Range	Size
Basic and Baumgarte	$\boldsymbol{u}_k,\ \boldsymbol{x}_k$	$k=0,\dots,N$	$(n_u+2n_q)\,(N+1)$
	$\boldsymbol{x}_{k,i},\ \ddot{\boldsymbol{q}}_{k,i},\ \boldsymbol{\lambda}_{k,i}$	$k=0,\dots,N-1;\ i=1,\dots,d$	$(3n_q+n_v)\,Nd$
	$\boldsymbol{F}(\boldsymbol{x}_0)=\boldsymbol{0}$		n_e
	$\boldsymbol{p}_k(t_{k+1},\boldsymbol{c}_k)=\boldsymbol{x}_{k+1}$	$k=0,\dots,N-1$	$2n_qN$
	$\dot{\boldsymbol{p}}_k(t_{k,i},\boldsymbol{c}_k)=\dot{\boldsymbol{x}}_{k,i}$	$k=0,\dots,N-1;\ i=1,\dots,d$	$2n_qNd$
	$\boldsymbol{D}(\boldsymbol{x}_{k,i},\dot{\boldsymbol{x}}_{k,i},\boldsymbol{u}_{k,i},\boldsymbol{\lambda}_{k,i})=\boldsymbol{0}$	$k=0,\dots,N-1;\ i=1,\dots,d$	$(n_q+n_v)\,Nd$
PKT	$\boldsymbol{u}_k,\ \boldsymbol{x}_k,\ \boldsymbol{\lambda}_k$	$k=0,\dots,N$	$(n_u+2n_q+n_p)\,(N+1)$
	$\boldsymbol{\lambda}_{k,c},\ \boldsymbol{\gamma}_{k,c}$	$k=0,\dots,N-1$	$2n_pN$
	$\boldsymbol{F}(\boldsymbol{x}_k)=\boldsymbol{0}$	$k=0,\dots,N$	$2n_p\,(N+1)$
	$\boldsymbol{B}(\boldsymbol{q}_k)\,\boldsymbol{g}(\boldsymbol{x}_k,\boldsymbol{u}_k,\boldsymbol{\lambda}_k)=\boldsymbol{\xi}(\boldsymbol{q}_k,\dot{\boldsymbol{q}}_k)$	$k=0,\dots,N$	$n_p\,(N+1)$
	$\dot{\boldsymbol{q}}_k(t_{k,c},\boldsymbol{c}_k)=\boldsymbol{v}_k(t_{k,c},\boldsymbol{c}_k)+\boldsymbol{B}(\boldsymbol{q}_{k,c})^\top\,\boldsymbol{\gamma}_{k,c}$	$k=0,\dots,N-1$	n_qN
	$\dot{\boldsymbol{v}}_k(t_{k,c},\boldsymbol{c}_k)=\boldsymbol{g}(\boldsymbol{x}_{k,c},\boldsymbol{u}_{k,c},\boldsymbol{\lambda}_{k,c})$	$k=0,\dots,N-1$	n_qN
Projection	$\boldsymbol{u}_k,\ \boldsymbol{x}_k$	$k=0,\dots,N$	$(n_u+2n_q)\,(N+1)$
	$\boldsymbol{x}'_k,\ \boldsymbol{\mu}_k$	$k=1,\dots,N$	$(2n_q+n_e)\,N$
	$\boldsymbol{x}_{k,i},\ \ddot{\boldsymbol{q}}_{k,i},\boldsymbol{\lambda}_{k,i}$	$k=0,\dots,N-1;\ i=1,\dots,d$	$(3n_q+n_v)\,Nd$
	$\boldsymbol{F}(\boldsymbol{x}_k)=\boldsymbol{0}$	$k=0,\dots,N$	$n_e\,(N+1)$
	$\boldsymbol{x}'_{k+1}=\boldsymbol{p}_k(t_{k+1},\boldsymbol{c}_k)$	$k=0,\dots,N-1$	$2n_qN$
	$\boldsymbol{x}_{k+1}=\boldsymbol{x}'_{k+1}+\boldsymbol{F}_{\boldsymbol{x}}(\boldsymbol{x}_{k+1})^\top\,\boldsymbol{\mu}_k$	$k=0,\dots,N-1$	$2n_qN$
	$\dot{\boldsymbol{p}}_k(t_{k,i},\boldsymbol{c}_k)=\dot{\boldsymbol{x}}_{k,i}$	$k=0,\dots,N-1;\ i=1,\dots,d$	$2n_qNd$
	$\boldsymbol{D}(\boldsymbol{x}_{k,i},\dot{\boldsymbol{x}}_{k,i},\boldsymbol{u}_{k,i},\boldsymbol{\lambda}_{k,i})=\boldsymbol{0}$	$k=0,\dots,N-1;\ i=1,\dots,d$	$(n_q+n_v)\,Nd$
Local coordinates	$\boldsymbol{u}_k,\ \boldsymbol{x}_k$	$k=0,\dots,N$	$(n_u+2n_q)\,(N+1)$
	$\boldsymbol{x}_{k,i},\ \boldsymbol{y}_{k,i},\ \ddot{\boldsymbol{q}}_{k,i},\ \boldsymbol{\lambda}_{k,i}$	$k=0,\dots,N-1;\ i=1,\dots,d$	$(3n_q+d_{\mathcal{X}}+n_v)\ \ Nd$
	$\boldsymbol{F}(\boldsymbol{x}_0)=\boldsymbol{0}$		n_e
	$\boldsymbol{G}_k(\boldsymbol{x}_{k+1},\boldsymbol{p}_k(t_{k+1},\boldsymbol{c}_k))=\boldsymbol{0}$	$k=0,\dots,N-1$	$2n_qN$
	$\boldsymbol{G}_k(\boldsymbol{x}_{k,i},\boldsymbol{y}_{k,i})=\boldsymbol{0}$	$k=0,\dots,N-1;\ i=1,\dots,d$	$2n_qNd$
	$\dot{\boldsymbol{p}}_k(t_{k,i},\boldsymbol{c}_k)=\boldsymbol{U}_k(\boldsymbol{x}_k)^\top\,\dot{\boldsymbol{x}}_{k,i}$	$k=0,\dots,N-1;\ i=1,\dots,d$	$d_{\mathcal{X}}Nd$
	$\boldsymbol{D}(\boldsymbol{x}_{k,i},\dot{\boldsymbol{x}}_{k,i},\boldsymbol{u}_{k,i},\boldsymbol{\lambda}_{k,i})=\boldsymbol{0}$	$k=0,\dots,N-1;\ i=1,\dots,d$	$(n_q+n_v)\,Nd$

where $\boldsymbol{D}(\boldsymbol{x},\dot{\boldsymbol{x}},\boldsymbol{u},\boldsymbol{\lambda})=\boldsymbol{0}$ denotes Eq. (7), or its stabilized version in the Baumgarte method, and $\boldsymbol{\lambda}_{k,i}$ is an auxiliary multiplier. Clearly, this adds more variables and equations to the transcribed problems, but the resulting system is much sparser, which often improves the convergence of the optimizer. Moreover, bounds on the constraint forces are easier to set in this form, as they directly relate to the $\boldsymbol{\lambda}_{k,i}$ values. To set such bounds, however, the $\ddot{\boldsymbol{q}}_{k,i}$ components of $\dot{\boldsymbol{x}}_{k,i}$ are not necessary, so some approaches, and in particular the PKT method, use a semi-implicit approach to eliminate them. This can be achieved by substituting each variable $\ddot{\boldsymbol{q}}_{k,i}$ of the previous formulation by $\boldsymbol{g}(\boldsymbol{x}_{k,i},\boldsymbol{u}_{k,i},\boldsymbol{\lambda}_{k,i})$, where $\boldsymbol{g}(\boldsymbol{x},\boldsymbol{u},\boldsymbol{\lambda})$ is the acceleration function we obtained in Eq. (32), and adding Eq. (6) evaluated for $t=t_{k,i}$, which yields

$$\boldsymbol{B}(\boldsymbol{q}_{k,i})\,\boldsymbol{g}(\boldsymbol{x}_{k,i},\boldsymbol{u}_{k,i},\boldsymbol{\lambda}_{k,i})=\boldsymbol{\xi}(\boldsymbol{q}_{k,i},\dot{\boldsymbol{q}}_{k,i}). \tag{58}$$

This formulation reduces the number of added variables, but also the sparsity of the system, so it may be less effective than the fully implicit formulation in our experience.

Table I summarizes the variables and equations introduced by each method, assuming the implicit form of the dynamics is used. The only exception is the PKT method, which, in accordance to [7], is written using the semi-implicit form. In all cases, we only give the variables and equations introduced by the dynamics, so boundary or path constraints do not appear in the table. By adding the number of variables minus the number of equations in each method, it is easy to see that the solution space of the NLP problem is of dimension $(N+1)\,n_u+d_{\mathcal{X}}$ in all transcriptions. Thus, for a fixed N, all methods allow the same freedom to compute the optimal solution.

B. Ensuring proper projections

For both the projection and local coordinates methods, we may find situations in which a projection ray intercepts the $\mathcal{X}$ manifold in multiple points (Fig. 7). In these situations, we need to ensure that the position of $\boldsymbol{x}_{k+1}$ chosen by the optimizer (red point) is the one that is closest to the projection point (white point), otherwise the trajectory would suddenly jump to farther regions in $\mathcal{X}$ (green point). To this end, we define the distance d_k from the projection point to $\boldsymbol{x}_{k+1}$ as

$$d_k=\|\boldsymbol{x}'_{k+1}-\boldsymbol{x}_{k+1}\| \tag{59}$$

in the projection method, and as

$$d_k=\|(\boldsymbol{x}_k+\boldsymbol{U}_k(\boldsymbol{x}_k)\,\boldsymbol{y}_{k+1})-\boldsymbol{x}_{k+1}\| \tag{60}$$

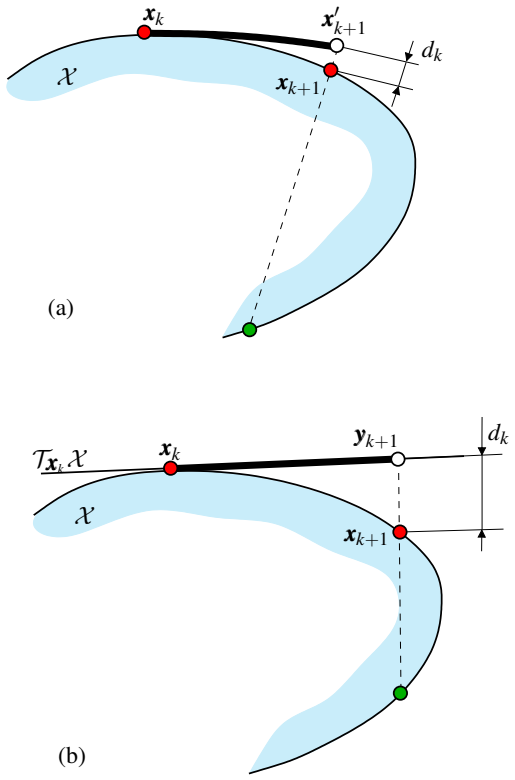


Fig. 7. Projection rays in the projection and local coordinates methods [(a) and (b), respectively]. Whereas in (a) the ray is orthogonal to $\mathcal{X}$, in (b) it is orthogonal to $\mathcal{T}_{\boldsymbol{x}_k}\mathcal{X}$. In both cases there may be more than one solution to the projection step to $\mathcal{X}$ (red and green points). The transcriptions should ensure the optimizer selects the red point for $\boldsymbol{x}_{k+1}$, i.e., the one that is closest to the projection point (in white). See the text for details.

in the local coordinates method, and we add a small penalty term proportional to

$$\sum_{k=1}^{N} d_k^2 \tag{61}$$

in the cost function $C(\boldsymbol{w})$ in Eq. (22a). In this way, the optimizer selects the closest point to the projection point in the ray.

C. Setting the boundary conditions

Many trajectory optimization problems require fixing the end point of the trajectory $\boldsymbol{x}_N$ to a particular state $\boldsymbol{x}_g$. However, the direct transcription of this constraint through imposing $\boldsymbol{x}_N = \boldsymbol{x}_g$ is not suitable in constrained systems. In such systems, all transcription methods will implicitly or explicitly constrain $\boldsymbol{x}_N$ to be a point of $\mathcal{X}$. For instance, in the basic method, only the drift prevents $\boldsymbol{x}_N$ to be on $\mathcal{X}$ once a control sequence is fixed, and other transcription methods explicitly include the constraint $\boldsymbol{F}(\boldsymbol{x}_N) = \boldsymbol{0}$. In all cases, directly using $\boldsymbol{x}_N = \boldsymbol{x}_g$ would result on an overconstrained system, thus violating the required LICQ conditions [31]. A way around this problem consists in replacing $\boldsymbol{x}_N = \boldsymbol{x}_g$ by

$$\boldsymbol{U}_g^\top(\boldsymbol{x}_N - \boldsymbol{x}_g) = \boldsymbol{0}, \tag{62}$$

where $\boldsymbol{U}_g$ is an $n_x \times d_{\mathcal{X}}$ matrix whose columns provide an orthogonal basis of $\mathcal{T}_{\boldsymbol{x}_g}\mathcal{X}$. Equation (62) constrains $\boldsymbol{x}_N$ to lie in the normal space of $\mathcal{X}$ at $\boldsymbol{x}_g$, so it removes the $d_{\mathcal{X}}$ degrees of freedom of $\boldsymbol{x}_N$ on $\mathcal{X}$. In this case, however, a projection ambiguity may appear which can be avoided including a small penalty in the cost function proportional to $\|\boldsymbol{x}_N - \boldsymbol{x}_g\|^2$. For the PKT, projection, and local coordinates methods, which include the constraint $\boldsymbol{F}(\boldsymbol{x}_N) = \boldsymbol{0}$, an alternative solution consists in replacing this constraint by $\boldsymbol{x}_N = \boldsymbol{x}_g$. Similiar considerations apply when fixing $\boldsymbol{x}_0$ to a particular value.

D. Finding an initial guess

Like most trajectory optimization methods, those we propose require reasonable initial guesses of $\boldsymbol{u}(t)$ and $\boldsymbol{x}(t)$ to converge to a locally-optimal solution. A typical way to find them consists in constructing an approximate trajectory $\boldsymbol{x}(t)$ satisfying the boundary conditions (for example by interpolating the start and goal states or using some educated guess), then evaluating $\boldsymbol{x}(t)$ for $t = t_0, \ldots, t_N$ to obtain values for $\boldsymbol{x}_0, \ldots, \boldsymbol{x}_N$, and finally deriving consistent actions $\boldsymbol{u}_0, \ldots, \boldsymbol{u}_N$ by solving the inverse dynamics using Eq. (7). This also provides initial guesses of $\boldsymbol{\lambda}_0, \ldots, \boldsymbol{\lambda}_N$, which are needed if we resort to implicit forms of the dynamic equations (see Section VI-A).

The previous initializations work well in simple situations, but in constrained robots the joint angles tend to be highly coupled, so guessing their trajectories is difficult, and they may be underactuated, or have singularities, which prevents the solution of the inverse dynamics in general [42]. One way to circumvent these difficulties is to resort to a randomized kinodynamic planner like the one in [36, 42], which finds a dynamically-feasible trajectory respecting the force limits of the actuators even across forward singularities. This trajectory will often be jerky, and far from optimal, but in many cases it is good enough to allow the convergence of the optimizer. This has been our method of choice in the examples below (Section VII). A good account of other initialization techniques is given in [6].

E. Accuracy metrics

To evaluate the quality of a trajectory it is essential to define proper accuracy metrics. These metrics allow us to compare the five transcription methods described in this paper. We here use two common error functions to quantify how well $\boldsymbol{x}(t)$ and $\boldsymbol{u}(t)$ satisfy Eqs. (2) and (9). The logic is that if these two equations are accurately fulfilled (both at the knot points and in between them) then the spline for $\boldsymbol{x}(t)$ will provide an accurate representation of the motion induced by $\boldsymbol{u}(t)$. Therefore, the lower the errors, the lower the control effort needed to stabilize the trajectories a posteriori.

Specifically, we define the kinematic error as

$$e_K(t) = \|\boldsymbol{F}(\boldsymbol{x}(t))\|, \tag{63}$$

and the dynamic error as

$$e_D(t) = \|\dot{\boldsymbol{x}}(t) - \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t))\|. \tag{64}$$

The averages of these two errors over $[0, t_f]$,

$$E_K = \frac{1}{t_f}\int_0^{t_f} e_K(t)\,\mathrm{d}t, \tag{65}$$

$$E_D = \frac{1}{t_f}\int_0^{t_f} e_D(t)\,\mathrm{d}t, \tag{66}$$

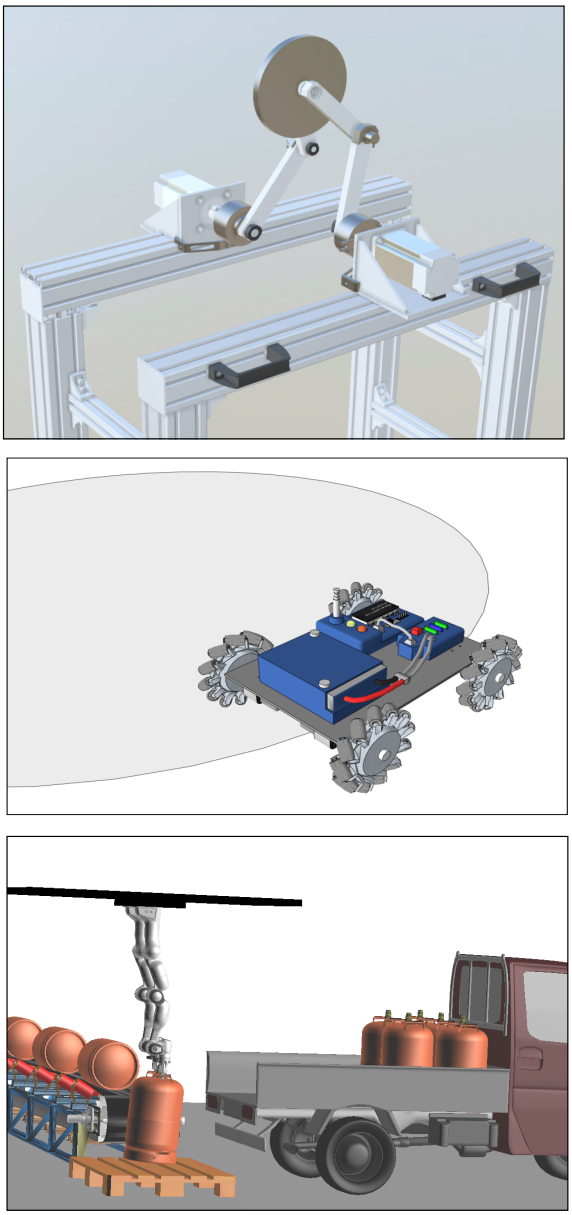
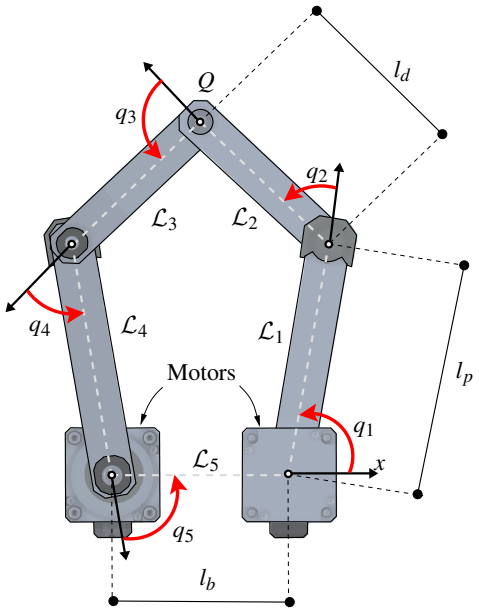


Fig. 8. Examples. Top: a parallel robot used to lift a heavy load. Middle: a robot with omni-directional wheels constrained to follow a circular path. Bottom: two Franka Emika manipulators used to lift gas bottles onto a truck.



Parameter	Symbol	Value	Units
Base distance	l_b	0.12	[m]
Proximal link length	l_p	0.20	[m]
Distal link length	l_d	0.15	[m]
Mass of proximal link	m_p	1.20	[kg]
Mass of distal link	m_d	0.90	[kg]
Moment of inertia of proximal link	I_p	0.007	[kg·m^2]
Moment of inertia of distal link	I_d	0.002	[kg·m^2]
Mass of the circular weight at Q	m_w	0.5	[kg]
Viscous friction coefficient	b	0.07	[N·m·s/rad]
Maximum torque of each motor	τ_{max}	1.4	[N·m]

Fig. 9. Geometry and dynamic parameters of the parallel robot. All joints are revolute. Joints q_1 and q_5 are actuated, and the remaining joints are passive. The motors are fixed to the ground, which acts as a fifth bar. All moments of inertia are given about the center of mass of the corresponding link.

provide global quantities summarizing the two types of errors over the whole trajectories $\boldsymbol{x}(t)$ and $\boldsymbol{u}(t)$. In our implementation, these integrals are computed using Shampine's adaptive quadrature method included in Matlab [44], under an absolute error tolerance of 10^{-10}.

VII. Examples

We next compare the performance of all methods in solving trajectory optimization problems for the three systems shown in Fig. 8: a parallel robot used to lift a heavy weight, a Mecanum-wheeled vehicle following a prescribed path, and a dual-arm system that loads gas bottles onto a truck. Whereas the first system illustrates the methods' handling of holonomic constraints (in this case a loop-closure constraint), the second shows how they cope with nonholonomic constraints (the rolling contacts), and the third provides a case with many degrees of freedom. All optimization problems have been formulated as in Table I, and they have been solved using a MacBook Pro with an Intel i9, 6-core processor running at 2.9 GHz. Except where noted, the final time t_f is fixed beforehand.

A. A system with holonomic constraints

The parallel robot we consider has the geometry shown in Fig 9. It consists of a closed loop of five links $\mathcal{L}_1,\dots,\mathcal{L}_5$ pairwise connected by revolute joints. In the figure, q_i denotes the relative angle at the i-th joint, which connects $\mathcal{L}_i$ with $\mathcal{L}_{i-1}$. Joints 1 and 5 are actuated, allowing the control of the (x,y) position of point Q, which is regarded as the end effector. The rest of joints are passive. A heavy disk is mounted on the axis at Q, as shown in Fig. 8, top. The goal is to lift this disk from a bottom position in which the robot is at rest, to an upright position. To complicate the task, the robot is set to move on a vertical plane, so it must overcome gravity, and we limit the motor torques to a small range $[-\tau_{\max},\tau_{\max}]$ that impedes direct trajectories to the goal. In all computations we assume the parameters given in Fig 9.

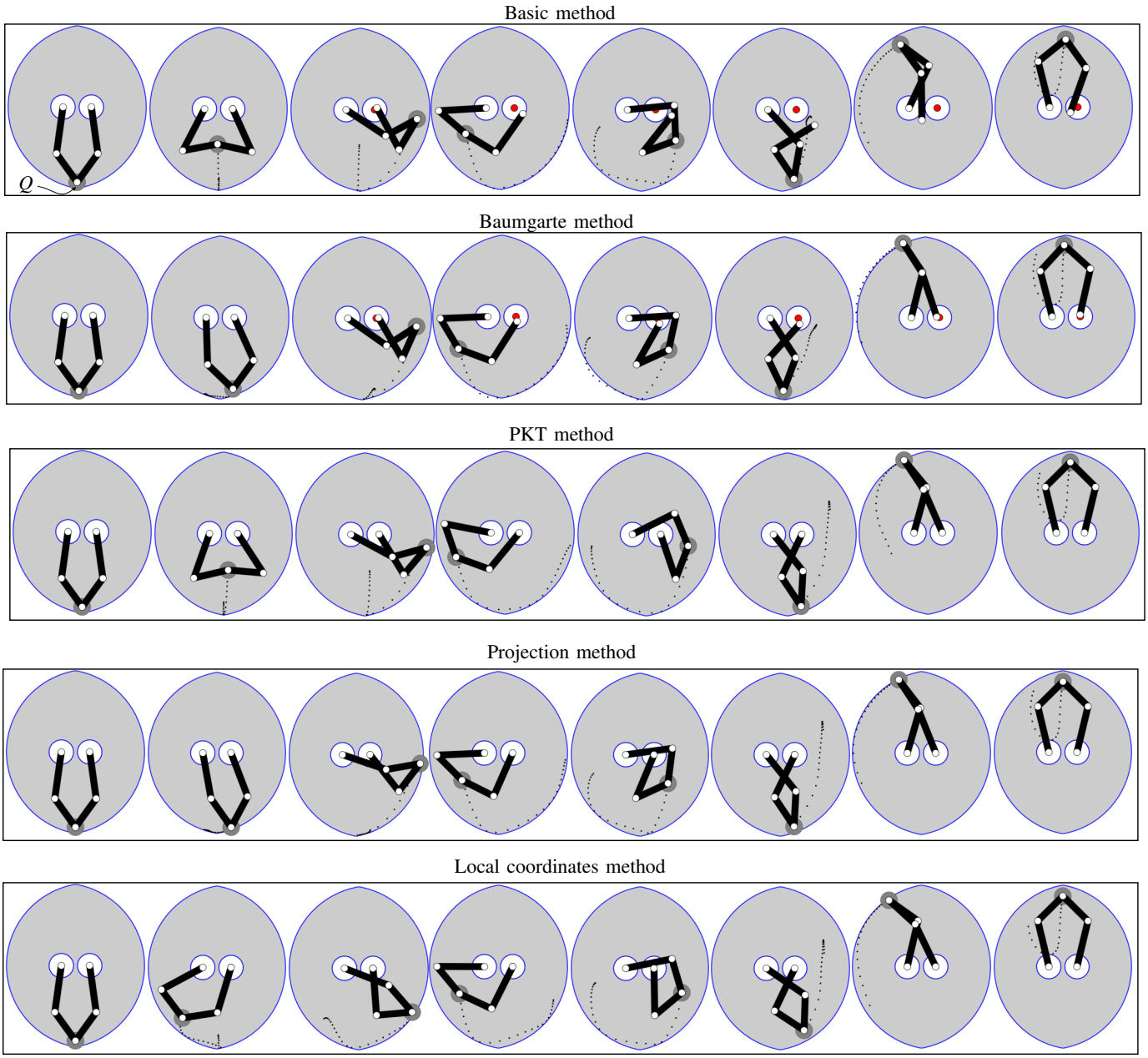


Fig. 10. Trajectories obtained by the five methods in the weight lifting task. See [43] for a video of this figure. The grey area in each snapshot is the workspace of the robot, i.e., the set of positions that point Q can attain. The two base joints are actuated. The trajectory from the basic method shows that, due to the accumulation of drift, the mechanism disassembles at the right motor joint (see the mismatch between the expected and obtained positions of this joint in several snapshots, in red and white respectively). In the trajectory from the Baumgarte method we see that, despite the stabilization of drift, the mechanism slightly disassembles and Q leaves its workspace at times. In the PKT and projection methods, in contrast, the mechanism is kept almost assembled (only tiny disassemblies arise in between knot points in the video). The local coordinates method finally removes all disassemblies.

In this robot, Eq. (1a) consists in the closure condition imposed by the kinematic loop formed by the five links. This condition can be expressed as

$$\begin{bmatrix} l_p\,\mathrm{c}(\bar{q}_1)+l_d\,\mathrm{c}(\bar{q}_2)+l_d\,\mathrm{c}(\bar{q}_3)+l_p\,\mathrm{c}(\bar{q}_4)+l_b \\ l_p\,\mathrm{s}(\bar{q}_1)+l_d\,\mathrm{s}(\bar{q}_2)+l_d\,\mathrm{s}(\bar{q}_3)+l_p\,\mathrm{s}(\bar{q}_4) \\ s(\bar{q}_5) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \quad (67)$$

where $\mathrm{s}(\cdot)$ and $\mathrm{c}(\cdot)$ denote the sine and cosine of their argument and $\bar{q}_i = \sum_{j=1}^{i} q_j$ is the angle of i-th link relative to ground. In this case, Eq. (1b) contains the time derivative of Eq. (67) only, as all robot constraints are holonomic. The generalized coordinates of this robot are $\boldsymbol{q} = (q_1, \ldots, q_5)$, so $n_x = 10$, $n_p = n_v = 3$, and $\mathcal{X}$ is 4-dimensional in this problem. To set up Eq. (8), moreover, we have used the methods in [45], which efficiently calculate all terms intervening.

Fig. 10 shows weight-lifting trajectories obtained by all methods using the running cost $L(\boldsymbol{x}(t), \boldsymbol{u}(t)) = \boldsymbol{u}(t)^\top \boldsymbol{u}(t)$ for Eq. (12a). For a fair comparison, we have set $d = 3$ in all methods, as the PKT method can only work with cubic polynomials. As we see in the figure, in all trajectories the torque limits force the generation of swinging motions to reach

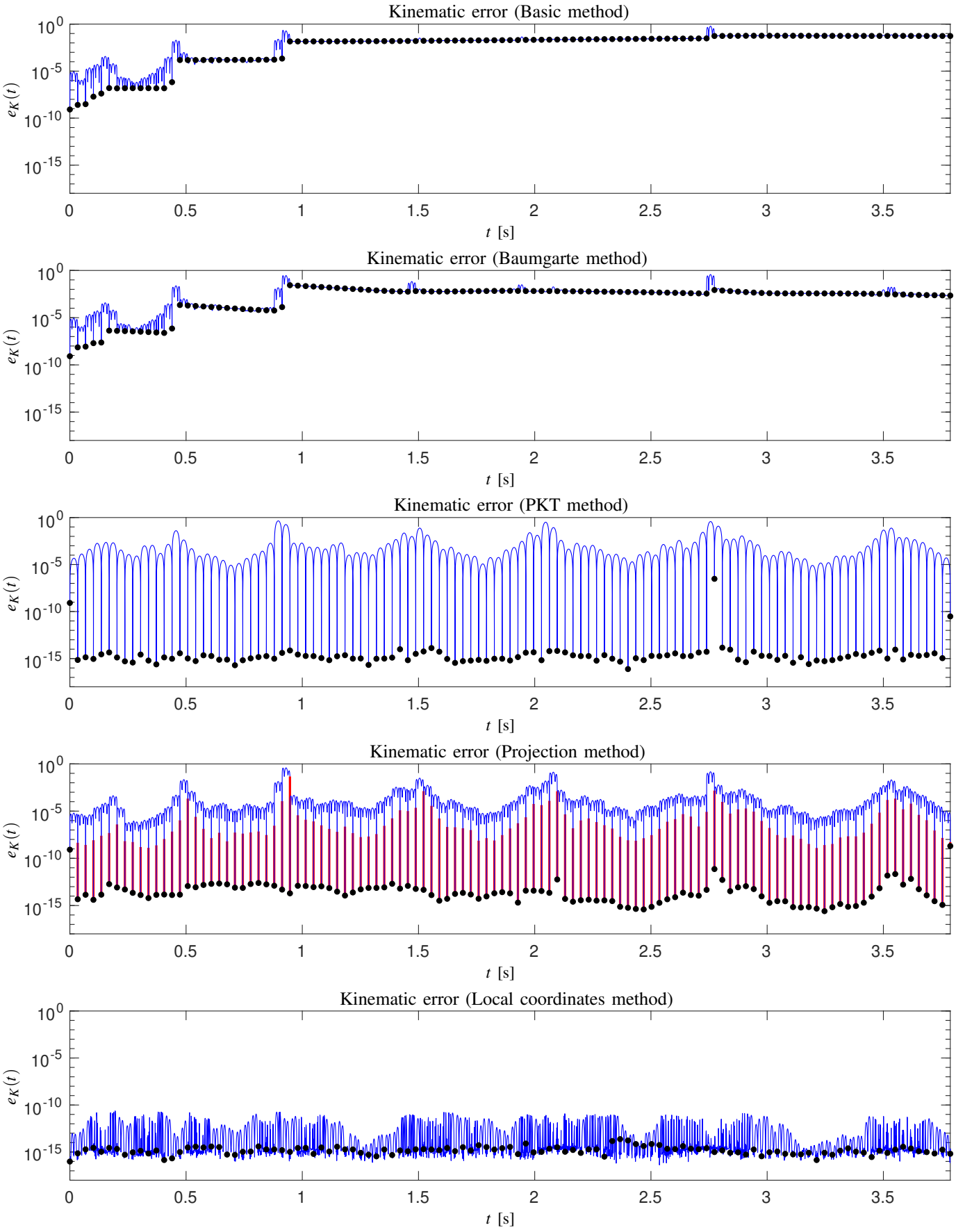


Fig. 11. Logarithmic plot of the kinematic error $e_K(t)$ for the weight-lifting trajectories obtained by all methods (shown in blue and using $d = 3$). The black dots indicate the values of $e_K(t)$ at the knot points. In the fourth plot, the red segments correspond to the projections from $\boldsymbol{x}'_k$ to $\boldsymbol{x}_k$ shown in Fig. 3.

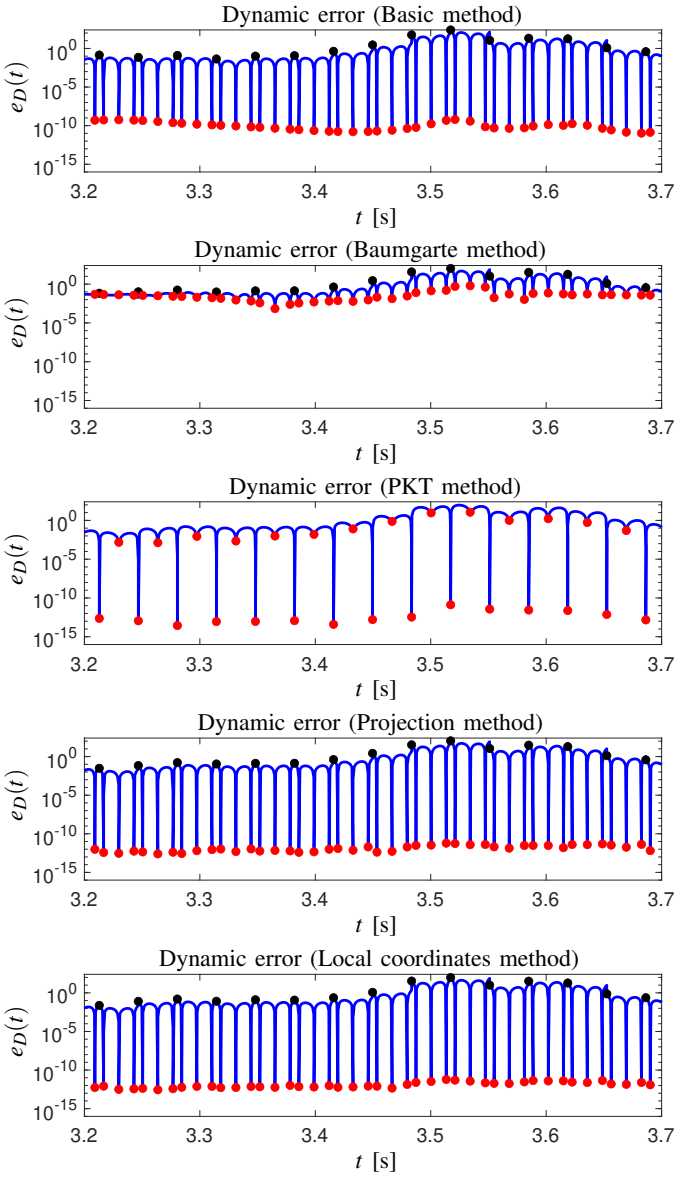


Fig. 12. Logarithmic plot of the dynamic error $e_D(t)$ for the weight-lifting task in all methods, using polynomial splines of degree $d = 3$. We only depict $e_D(t)$ for the time span $[3.2, 3.7]$ to better appreciate this error at the collocation points (in red), but the trends are similar in the whole trajectory. The black dots indicate the value of $e_D(t)$ for the knot points $t_0, \ldots, t_N$. In the PKT method the knot points coincide with collocation points (those where $e_D(t)$ is negligible), so their black dots are occluded.

the goal. The trajectory from the basic method shows that the drift inevitably accumulates, so the mechanism disassembles and the goal state cannot be reached (see the mismatch between the expected and obtained positions of the right motor joint in several snapshots, in red and white respectively). The trajectory from the Baumgarte method reveals that, although the drift is mitigated, the mechanism still disassembles slightly. The PKT and projection methods behave similarly, as both methods eliminate the drift at the knot points, so only tiny disassemblies are visible in between such points (see the the animation of this figure in [43]). The local coordinates method finally eliminates any drift along the continuous time trajectory.

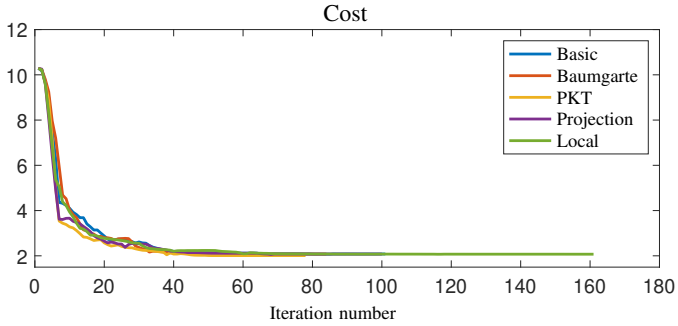


Fig. 13. Evolution of the cost as a function of the iteration number.

The kinematic error corresponding to the previous trajectories is shown in Fig. 11. Note that a large value of $e_K(t)$ implies that Eq. (67) is violated, which results in unrealistic trajectories that are difficult to track with a controller. Thus, a method resulting in low values of $e_K(t)$ is preferable. From the figure we see that each method performs as expected. In the basic method, $e_K(t)$ tends to accumulate over time, which in Fig. 10 corresponds to the progressive disassembly of the right motor joint. In the Baumgarte method, the stabilizing terms reduce the accumulation of drift from $t = 1$ onwards, but many knot points still deviate substantially from the manifold (black dots). In the PKT and projection methods, instead, $e_K(t) \approx 0$ at the knot points, but not in between them. Finally, $e_K(t)$ is negligible for all $t \in [0, t_f]$ in the local coordinates method, so this is the most accurate of all methods.

To see how the different methods compare in terms of satisfying the collocation constraints, Fig. 12 provides the dynamic error $e_D(t)$ for all methods in the time span $[3.2, 3.7]$, which is representative of the rest of the trajectory. As expected, in the basic method the value of $e_D(t)$ is negligible at the collocation points (red dots in the figure) because it imposes the actual dynamics at such points. In the Baumgarte method, in contrast, $e_D(t)$ is much larger at the collocation points, because this method enforces a modified version of the dynamics to mitigate the problem of drift. The PKT method is an improvement over the Baumgarte method because $e_D(t)$ is negligible in one of every two collocation points. In the projection and local coordinates methods, instead, $e_D(t)$ is negligible in all collocation points.

Table II provides global performance measures for the five methods when using polynomials of increasing degree d (except for the PKT method, which requires $d = 3$) using $N = 112$ and $\Delta t = 0.034$ [s] in all cases. For each value of d we provide the number of variables n_w and constraints n_c in the transcriptions, the average integral kinematic and dynamic errors, E_K and E_D, given by Eqs. (65) and (66) respectively, the cost C^* of the optimized trajectory (as a reference, the initial guess has $C = 10.54$), the CPU time t_{opt} used to solve the NLP problem, the number of optimization iterations n_i, and the average time per iteration t_i. As expected, both E_K and E_D decrease when increasing d in each method. For a same value of d, the projection method (and also the PKT method when $d = 3$) has values of E_K that are smaller than those of the basic and Baumgarte methods. In contrast, E_K is negligible in the local coordinates method, and only depends

TABLE II
PERFORMANCE STATISTICS FOR THE WEIGHT LIFTING TASK IN THE PARALLEL ROBOT AS d INCREASES

Method	d	n_w	n_c	E_K	E_D	C^*	t_{opt} [s]	n_i	t_i [s]
Basic	2	5388	5166	$31.44 \cdot 10^{-3}$	58.57	0.73	3.02	55	0.055
	3	7404	7182	$29.06 \cdot 10^{-3}$	43.70	1.07	5.65	54	0.105
	4	9420	9198	$8.40 \cdot 10^{-3}$	26.46	1.74	15.83	76	0.208
Baumgarte	2	5388	5166	$19.09 \cdot 10^{-3}$	61.21	0.64	4.14	56	0.074
	3	7404	7182	$15.47 \cdot 10^{-3}$	31.59	1.68	9.75	71	0.137
	4	9420	9198	$2.38 \cdot 10^{-3}$	14.12	2.06	18.53	98	0.189
PKT	3	2367	2145	$10.90 \cdot 10^{-3}$	33.19	0.61	6.69	46	0.145
Projection	2	7180	6958	$11.47 \cdot 10^{-3}$	63.90	0.61	4.02	54	0.074
	3	9196	8974	$4.10 \cdot 10^{-3}$	30.91	1.77	4.89	47	0.104
	4	11212	10990	$1.37 \cdot 10^{-3}$	14.30	2.05	13.54	72	0.188
Local coordinates	2	6284	6062	$4.13 \cdot 10^{-13}$	39.36	2.05	20.51	80	0.256
	3	10698	10426	$5.04 \cdot 10^{-13}$	18.36	2.07	16.14	65	0.248
	4	11212	10990	$3.82 \cdot 10^{-13}$	10.03	2.09	21.80	78	0.279

on the tolerance used by the optimizer when solving Eq. (52). Also for a same d, the local coordinates method has the smallest value of E_D, which is in agreement with its higher accuracy. This higher accuracy comes at the cost of a larger optimization time due to computing the tangent space bases for all knot points in each iteration (see Appendix A). As a result, the cost of this operation dominates the execution time for this method. This is why its time per iteration is almost constant in the tests, irrespective of d. This is not the case for the rest of methods. Note that n_e and n_w indicate the size of the corresponding NLP problem. The projection and the local coordinates methods define larger NLP problems, but this does not necessarily translate into larger execution times. This effect is remarkable for the PKT method, which is the one with smallest programs, but not the one with the lowest execution times. This can be attributed to the fact that the PKT method uses the semi-implicit form of the dynamics, instead of the fully-implicit one (Section VI-A)

As shown in the companion video of this paper (see [43]), the different methods obtain solution trajectories in a same homotopy class. The slight differences between the trajectories, and in the values of C^*, can be attributed to the violation of the kinematic constraints, as only the local coordinates method can find a solution that is fully compliant with such constraints. As the precision is increased, either by increasing d or by decreasing Δt, the kinematic error decreases and all methods converge to a same value C^*. This is illustrated in Fig. 13 which shows the evolution of the cost for $d = 3$, but using a smaller time step $\Delta t = 0.01$ [s]. Clearly, all methods converge to a solution with the same cost.

Fig. 14 shows that the methods can cope with different cost functions, including those yielding bang bang controls. The figure shows the initial guess of the action trajectory (top plot) and the optimized action trajectories $\boldsymbol{u}^*(t)$ that we obtain when using the following running costs in Eq. (12a):

$$L_1(\boldsymbol{x}(t), \boldsymbol{u}(t)) = \boldsymbol{u}(t)^\top \boldsymbol{u}(t), \tag{68}$$

$$L_2(\boldsymbol{x}(t), \boldsymbol{u}(t)) = \dot{\boldsymbol{u}}(t)^\top \dot{\boldsymbol{u}}(t), \tag{69}$$

$$L_3(\boldsymbol{x}(t), \boldsymbol{u}(t)) = 1. \tag{70}$$

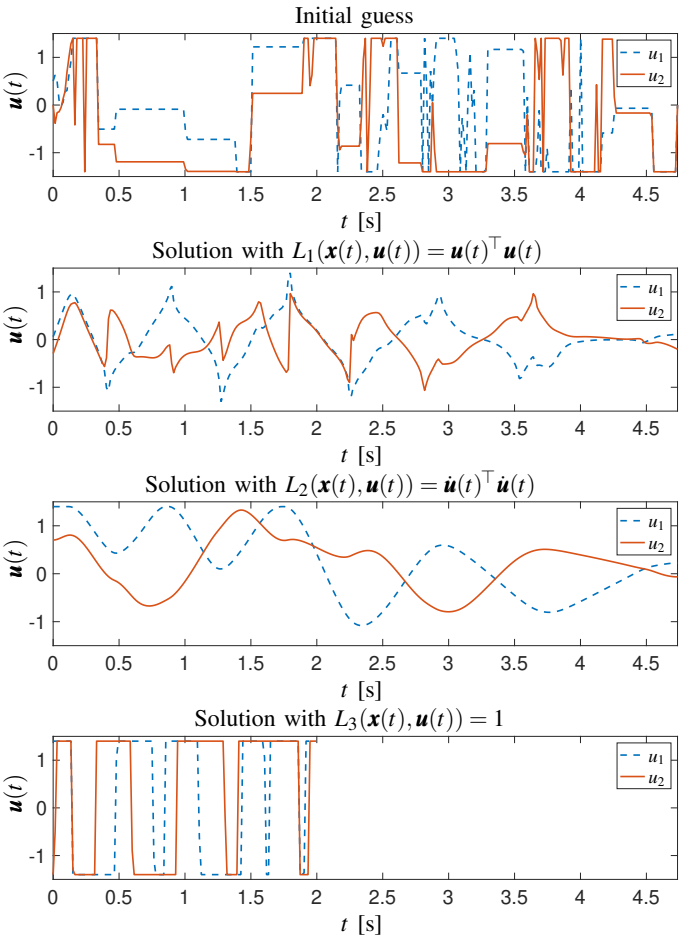
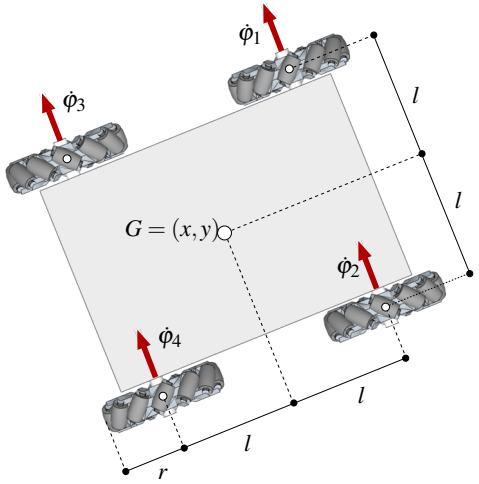


Fig. 14. Top plot: the initial guess of the action trajectory $\boldsymbol{u}(t)$. Bottom plots: the optimized function $\boldsymbol{u}(t)$ obtained by the projection method with $d = 4$ when using different running costs. See [43] for the animated trajectories corresponding to each cost.

As we see, the use of L_1 results in an action trajectory $\boldsymbol{u}^*(t)$ that is smoother in comparison to the initial guess, so the control signal will be easier to follow. If further smoothing is needed, we can use L_2, which will minimize the derivative of $\boldsymbol{u}(t)$. The curves in the third plot of Fig. 14 are clearly

TABLE III
PERFORMANCE STATISTICS FOR THE CIRCLE FOLLOWING TASK IN THE WHEELED ROBOT WHEN d INCREASES

Method	d	n_w	n_c	E_K	E_D	C^*	t_{opt} [s]	n_i	t_i [ms]
Basic	2	630	600	$1384.6\cdot10^{-3}$	97.4	2188.3	0.1	30	4.1
	3	855	825	$358.3\cdot10^{-3}$	18.9	2191.6	0.2	33	4.8
	4	1080	1050	$62.5\cdot10^{-3}$	3.5	2191.7	0.1	27	5.4
Baumgarte	2	630	600	$1366.4\cdot10^{-3}$	98.0	2188.3	0.1	27	2.8
	3	855	825	$344.8\cdot10^{-3}$	18.7	2191.6	0.2	28	6.6
	4	1080	1050	$61.1\cdot10^{-3}$	3.4	2191.7	0.2	29	5.9
Projection	2	792	762	$1094.1\cdot10^{-3}$	83.1	2189.0	0.2	39	5.1
	3	1017	987	$335.7\cdot10^{-3}$	20.0	2191.0	0.4	54	6.6
	4	1242	1212	$55.8\cdot10^{-3}$	3.4	2191.7	0.4	48	8.9
Local coordinates	2	810	780	$1.7\cdot10^{-13}$	46.4	2193.4	1.2	23	50.4
	3	1125	1095	$1.7\cdot10^{-13}$	3.9	2195.1	3.9	64	61.3
	4	1440	1410	$4.0\cdot10^{-13}$	0.2	2195.1	2.7	29	92.2



Parameter	Symbol	Value	Units
Side length of the chassis	l	0.21	[m]
Wheel radius	r	0.066	[m]
Vehicle mass	m	15.75	[kg]
Moment of inertia of the chassis	I_z	0.461	[kg·m^2]
Moment of inertia of the wheels	I_w	0.0026	[kg·m^2]
Viscous friction at the wheel axes	b	0.1	[N·m·s/rad]

Fig. 15. Geometric and dynamic parameters of the Mecanum-wheeled robot in the second example. They correspond to a Mecanum Wheel Vectoring Robot IG42 from SuperDroid Robots Inc. See [46] for details. The moments of inertia I_z and I_w are relative to an axis perpendicular to the chassis and wheel planes, meeting G and the wheel centers respectively.

smoother than those in the second plot. Finally, if we need to minimize the total trajectory time, we can free t_f and use L_3 to obtain the optimized trajectory $\boldsymbol{u}^*(t)$ of the fourth plot. A bang-bang control arises in which at least one motor works either at its upper or lower torque limit, and the robot achieves the goal state in only two seconds. The plots were obtained with the projection method, but similar results are achieved with the local coordinates method.

B. A system with nonholonomic constraints

To illustrate the proposed methods on nonholonomic systems, we next apply them to a Mecanum-wheeled robot with the parameters shown in Fig 15. The vehicle pose is given by the (x,y) coordinates of the centre of mass G of the vehicle, and by the angle ψ of the chassis, all relative to an absolute frame. We denote the wheel angles as $\varphi_1,\dots,\varphi_4$.

In this example, the robot's position is constrained to a circle of radius R centered at the origin, so the trajectory must fulfil

$$x(t)^2+y(t)^2=R^2. \tag{71}$$

The chassis orientation $\psi(t)$ can freely be chosen instead, as the vehicle has omni-directional wheels. In all tests we use $R=2.5$ [m].

Note that Eq. (71) cannot be considered as part of Eq. (1a) in this case because it is not intrinsic to the robot structure. Instead, it must be viewed as a path constraint modelled by Eq. (22b). In the transcribed problem, thus, Eq. (71) will intervene in the form

$$x_k^2+y_k^2=R^2, \qquad k=1,\dots,N. \tag{72}$$

To formulate the kinematic constraints, we denote the chassis configuration by $\boldsymbol{q}_r=(x,y,\psi)$ and the wheels configuration by $\boldsymbol{q}_w=(\varphi_1,\dots,\varphi_4)$. The robot configuration is then given by $\boldsymbol{q}=(\boldsymbol{q}_r,\boldsymbol{q}_w)$. As is common in mobile robots, we wish the wheels do not slip, so according to [46] $\boldsymbol{q}_r$ and $\boldsymbol{q}_w$ must be coupled by the nonholonomic constraint

$$\boldsymbol{K}\,\boldsymbol{R}_Z^\top(\psi)\,\dot{\boldsymbol{q}}_r-\dot{\boldsymbol{q}}_w=\boldsymbol{0}, \tag{73}$$

where

$$\boldsymbol{K}=\frac{1}{r}\begin{bmatrix}1&-1&-2l\\1&1&2l\\1&1&-2l\\1&-1&2l\end{bmatrix}$$

and $\boldsymbol{R}_Z(\psi)$ is the 3×3 rotation of angle ψ about the Z axis.

The optimizer must guarantee that Eq. (73) is fulfilled as closely as possible in the final trajectory, otherwise slippage of the robot relative to the ground occurs, which generates

TABLE IV
PERFORMANCE STATISTICS FOR THE TRUCK LOADING TASK IN THE DUAL-ARM SYSTEM WHEN d INCREASES

Method	d	n_w	n_c	E_K	E_D	C^*	t_{opt} [s]	n_i	t_i [s]
Basic	2	5838	5252	$1.17 \cdot 10^{-3}$	0.63	11699.49	392.70	83	4.731
	3	7854	7268	$0.17 \cdot 10^{-3}$	0.13	11731.70	1221.90	100	12.219
	4	9870	9284	$0.12 \cdot 10^{-3}$	0.11	12418.45	1815.59	96	18.912
Baumgarte	2	5838	5252	$0.86 \cdot 10^{-3}$	0.63	11700.70	833.62	126	6.616
	3	7854	7268	$0.15 \cdot 10^{-3}$	0.13	11731.65	1264.76	181	6.988
	4	9870	9284	$0.09 \cdot 10^{-3}$	0.11	12418.51	1090.45	81	13.462
PKT	3	2568	1982	$3.24 \cdot 10^{-3}$	1.89	13058.05	792.99	611	1.298
Projection	2	7518	6932	$1.16 \cdot 10^{-3}$	1.61	12425.24	476.90	700	0.681
	3	9534	8948	$0.14 \cdot 10^{-3}$	0.13	11731.23	492.29	453	1.087
	4	11550	10964	$0.03 \cdot 10^{-3}$	0.04	11729.36	1546.32	457	3.384
Local coordinates	2	7182	6596	$0.10 \cdot 10^{-13}$	0.69	12039.85	28.84	63	0.458
	3	9870	9284	$0.12 \cdot 10^{-13}$	0.15	12979.68	52.56	81	0.649
	4	12558	11972	$0.13 \cdot 10^{-13}$	0.04	12603.92	56.18	64	0.878

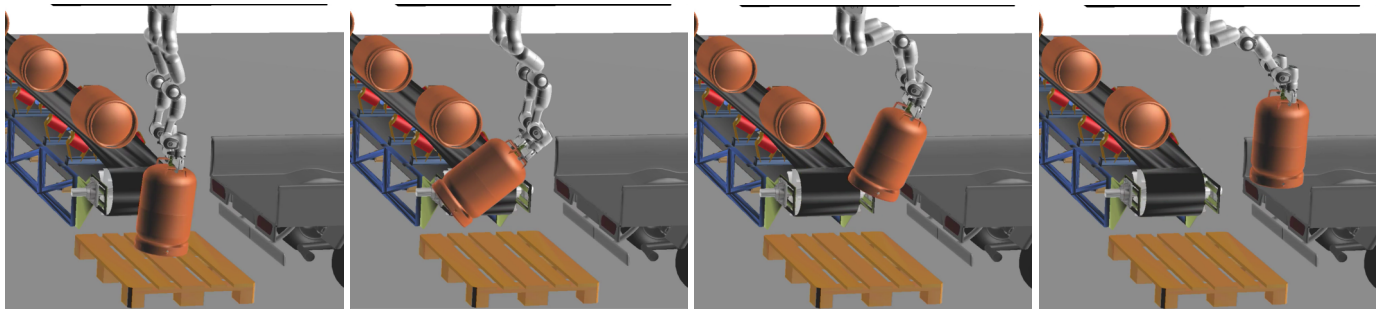


Fig. 16. A trajectory obtained for the truck loading task in the dual arm system using the local coordinates method and $d = 3$. To gain momentum, the robot first moves the bottle backwards, to then lift it forward onto the truck. The companion video of this paper provides an animated version [43].

odometry problems during navigation. This equation can be brought into the form of Eq. (1b) by defining

$$\boldsymbol{B}(\boldsymbol{q}) = [\boldsymbol{K}\boldsymbol{R}_Z^\top(\psi) \quad -\boldsymbol{I}_4], \tag{74}$$

where $\boldsymbol{I}_4$ is the 4×4 identity matrix. By taking the time derivative of Eq. (73) we obtain Eq. (6) in the form

$$\boldsymbol{B}(\boldsymbol{q})\,\ddot{\boldsymbol{q}} = -\boldsymbol{K}\dot{\boldsymbol{R}}_Z^\top(\psi)\,\dot{\boldsymbol{q}}_r. \tag{75}$$

Using the derivations in [46], Lagrange's equation for this robot is then given by

$$\boldsymbol{M}\ddot{\boldsymbol{q}} + \boldsymbol{B}(\boldsymbol{q})^\top \boldsymbol{\lambda} = \boldsymbol{\tau}(\boldsymbol{u}, \dot{\boldsymbol{q}}), \tag{76}$$

where

$$\boldsymbol{M} = \begin{bmatrix} \boldsymbol{M}_r & \\ & \boldsymbol{M}_w \end{bmatrix}, \quad \boldsymbol{M}_r = \begin{bmatrix} m & & \\ & m & \\ & & I_z \end{bmatrix}, \quad \boldsymbol{M}_w = I_w \boldsymbol{I}_4,$$

and

$$\boldsymbol{\tau}(\boldsymbol{u}, \dot{\boldsymbol{q}}) = \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{u} - b\dot{\boldsymbol{q}}_w \end{bmatrix}. \tag{77}$$

In these equations, $\boldsymbol{u} = (u_1, u_2, u_3, u_4)$ is the vector of wheel torques applied by the motors, and $b\dot{\boldsymbol{q}}_w$ models the viscous friction at the wheels. Thus, in this example $n_u = 4$, $n_x = 14$, $n_p = 0$, $n_v = 4$, and $d_{\mathcal{X}} = 10$. As explained in Section II, Eqs. (75) and (76) can be combined to find the explicit form of the dynamics equation $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})$.

Table III provides the performance statistics for this test case using $N = 9$, $\Delta t = 0.2$ [s], and the running cost

$$L(\boldsymbol{x}(t), \boldsymbol{u}(t)) = \boldsymbol{u}(t)^\top \boldsymbol{u}(t), \tag{78}$$

under increasing values of d. Results from the PKT method do not appear in this table because, as we said before, it cannot be applied to nonholonomic systems. All methods converge to trajectories of a similar cost in this example, and the results confirm that the projection and local coordinates methods reduce the kinematic error, or make it negligible, respectively. While the latter method is again more costly, it also achieves a remarkable decrease in the dynamic error. Compare the values of E_D for $d = 3$ or 4 for example.

C. A system with many degrees of freedom

To compare the methods in a system with many degrees of freedom, we now address a task in which two Franka Emika 7-DOF arms have to move a gas bottle cooperatively. Initially, the two arms are holding the bottle in a bottom position at rest, and the goal is to load it onto a truck, arriving with zero velocity at the goal configuration. The grippers in each arm rigidly grasp the bottle, so only the robot joints can move.

For this example, the $\boldsymbol{q}$ vector is formed by the 14 internal joint angles, and Eq. (1a) contains 6 loop-closure constraints. Thus, $n_x = 28$, $n_p = n_v = 6$, and $d_{\mathcal{X}} = 16$. The mass of the

bottle is twice the added payload of the two arms, so by designing this trajectory, we allow the system to move much beyond its static capabilities.

Table IV provides performance statistics for the solutions we obtain using $N = 42$, $\Delta t = 0.04$ [s], and the running cost $L(\boldsymbol{x}(t), \boldsymbol{u}(t)) = \dot{\boldsymbol{u}}^\top \dot{\boldsymbol{u}}$ under increasing values of d. We use this running cost to promote trajectories with smooth accelerations, which are useful to reduce stress, wear and tear of the robot arms. With the aim of obtaining highly accurate results without incurring in a too high computational cost, in this example we use the output trajectory from the projection method as the input for the local coordinates method, obtaining the result in Fig. 16. As in previous examples, all methods improve E_D and E_K as d increases. For $d > 2$, the projection method reduces the average kinematic error with respect to the basic, Baumgarte, and PKT methods, and the subsequent application of the local coordinates method eliminates such error up to numerical accuracy. Moreover, the total execution time of first applying the projection method and then the local coordinates methods is similar (or even shorter in some cases) than the execution time of the alternative approaches. Thus, the proposed methods provide an efficient and accurate pipeline for trajectory optimization in constrained robotic systems.

VIII. Conclusions

Existing trajectory optimization methods have difficulties when dealing with the holonomic and nonholonomic constraints that appear in many problems of robotics. As we have shown in this paper, these constraints confine the robot states to a nonlinear manifold, but transcription errors easily produce trajectories that drift away from this manifold. Such a drift typically translates into unrealistic behaviour of the robot, and complicates, or even prevents, the control of the trajectory a posteriori. To address these problems, this paper has introduced two methods that cancel the drift without affecting, or even reducing, the dynamic error of the trajectories. The two methods leverage techniques from geometric integration on manifolds [15, 22]–[25]. The projection method is simpler and typically faster, but it only cancels the drift at the knot points of the trajectory. In contrast, the local coordinates method cancels the drift all along the continuous-time trajectory. Both methods can employ polynomials of arbitrary degree, which allows their application to hp-adaptive meshing schemes [20, 21].

A key point in any trajectory optimizer is the quality of the initial guess of the solution. To provide such a guess we have used the trajectories obtained with the planner in [36], which are not optimal in any specific sense but satisfy all kinematic and dynamic constraints of the problem. According to our experience, this guess is helpful to converge to the optimal trajectories in reasonable times. For complex systems with many degrees of freedom, however, a better strategy is to improve this guess with the projection method (using a coarse mesh resolution or low-degree polynomials) and then use it as a warm start for the local coordinates method. Thus, the two approaches are complementary in this respect.

As usual in numerical optimal control, our final trajectories are local optima in the set of trajectories homotopic to the initial guess. To obtain a globally-optimal trajectory one should resort to asymptotically optimal planners like those in [47]–[49]. In this context, the methods we propose could be used as a steering method for a global planner, so the local connections could comply with both the kinematic and dynamic constraints of the problem. This is a research direction that, we believe, is worth exploring in the future.

Appendix

Computing a basis of the tangent space

To compute a basis of $\mathcal{T}_{\boldsymbol{x}_k}\mathcal{X}$, i.e., of the null space of $\boldsymbol{F}_{\boldsymbol{x}}(\boldsymbol{x}_k)$, we can use the QR decomposition. Using this decomposition, the $n_e \times n_x$ matrix $\boldsymbol{F}_{\boldsymbol{x}}(\boldsymbol{x}_k)$ can be expressed as

$$\boldsymbol{F}_{\boldsymbol{x}}(\boldsymbol{x}_k)^\top = \boldsymbol{Q}_k \boldsymbol{R}_k, \tag{79}$$

where $\boldsymbol{Q}_k$ is an $n_x \times n_x$ orthonormal matrix, so $\boldsymbol{Q}_k^\top \boldsymbol{Q}_k = \boldsymbol{I}_{n_x}$, and $\boldsymbol{R}_k$ is an $n_x \times n_e$ upper triangular matrix. From Eq. (79) we have that

$$\boldsymbol{F}_{\boldsymbol{x}}(\boldsymbol{x}_k)\, \boldsymbol{Q}_k = \boldsymbol{R}_k^\top, \tag{80}$$

which can be written in block form as

$$\boldsymbol{F}_{\boldsymbol{x}}(\boldsymbol{x}_k)\, [\boldsymbol{V}_k\, \boldsymbol{U}_k] = [\boldsymbol{L}_k\, \boldsymbol{0}], \tag{81}$$

where $\boldsymbol{V}_k$ includes the first n_e columns of $\boldsymbol{Q}_k$, $\boldsymbol{U}_k$ includes the remaining $d_{\mathcal{X}}$ columns, and $\boldsymbol{L}_k$ is an $n_e \times n_e$ lower triangular matrix. Since $\boldsymbol{F}_{\boldsymbol{x}}(\boldsymbol{x}_k)\, \boldsymbol{U}_k = \boldsymbol{0}$ and $\boldsymbol{U}_k^\top \boldsymbol{U}_k = \boldsymbol{I}_{d_{\mathcal{X}}}$, $\boldsymbol{U}_k$ provides the desired orthonormal basis of $\mathcal{T}_{\boldsymbol{x}_k}\mathcal{X}$.

Note that the basis $\boldsymbol{U}_k$ is not unique and typical implementations of the QR decomposition apply column reordering with the aim of improving the numerical stability of the procedure. Therefore, applying the process just described on nearby points may produce significantly different bases, so the procedure does not guarantee the continuity and smoothness of the outputs. This is inconvenient for the optimization process, which requires the derivatives of $\boldsymbol{U}_k$ with respect to $\boldsymbol{x}_k$. Therefore, once $\boldsymbol{U}_k$ is computed for a given point $\boldsymbol{x}_k$, i.e., after the initialization of the optimization process, it is more convenient to use Gram-Schmidt orthonormalization to update the basis $\boldsymbol{U}_l$ for any point $\boldsymbol{x}_l$ close enough to the previous estimation of $\boldsymbol{x}_k$. In this process, the columns $\boldsymbol{u}_l^j$ of $\boldsymbol{U}_l$ for $j = \{1, \ldots, d_{\mathcal{X}}\}$ are computed in sequence from the columns $\boldsymbol{u}_k^j$ of $\boldsymbol{U}_k$ in two steps. We first compute

$$\boldsymbol{w}_l^j = \boldsymbol{u}_k^j - \boldsymbol{A}_j \boldsymbol{A}_j^\top \boldsymbol{u}_k^j,$$

with $\boldsymbol{A}_j = [\boldsymbol{F}_{\boldsymbol{x}}(\boldsymbol{x}_l)^\top\, \boldsymbol{u}_l^1 \ldots \boldsymbol{u}_l^{j-1}]$, and then obtain

$$\boldsymbol{u}_l^j = \frac{\boldsymbol{w}_l^j}{\|\boldsymbol{w}_l^j\|}.$$

The first step projects $\boldsymbol{u}_k^j$ to the null space of $\boldsymbol{A}_j$, i.e., of $\boldsymbol{F}_{\boldsymbol{x}}(\boldsymbol{x}_l)$ and the vectors of $\boldsymbol{U}_l$ already computed. The second step just normalizes the resulting vector. This process is well-defined as long as none of the $\boldsymbol{w}_l^j$ vectors is zero, i.e., provided $\boldsymbol{U}_k$ and $\boldsymbol{U}_l$ are relatively similar.

REFERENCES

[1] S. M. LaValle, *Planning algorithms*. New York: Cambridge University Press, 2006.

[2] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "STOMP: Stochastic trajectory optimization for motion planning," in *IEEE International Conference on Robotics and Automation*, 2011, pp. 4569–4574.

[3] M. Zucker, N. Ratliff, A. D. Dragan, M. Pivtoraiko, M. Klingensmith, C. M. Dellin, J. A. Bagnell, and S. S. Srinivasa, "CHOMP: Covariant Hamiltonian optimization for motion planning," *The International Journal of Robotics Research*, vol. 32, no. 9-10, pp. 1164–1193, 2013.

[4] J. Schulman, D. Y, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel, "Motion planning with sequential convex optimization and convex collision checking," *The International Journal of Robotics Research*, vol. 33, no. 9, pp. 1251–1270, 2014.

[5] D. A. Benson, G. T. Huntington, T. P. Thorvaldsen, and A. V. Rao, "Direct trajectory optimization and costate estimation via an orthogonal collocation method," *Journal of Guidance, Control, and Dynamics*, vol. 29, no. 6, pp. 1435–1440, 2006.

[6] M. Kelly, "An introduction to trajectory optimization: How to do your own direct collocation," *SIAM Review*, vol. 59, no. 4, pp. 849–904, 2017.

[7] M. Posa, S. Kuindersma, and R. Tedrake, "Optimization and stabilization of trajectories for constrained dynamical systems," in *IEEE International Conference on Robotics and Automation*, 2016, pp. 1366–1373.

[8] D. Pardo, M. Neunert, A. Winkler, R. Grandia, and J. Buchli, "Hybrid direct collocation and control in the constraint-consistent subspace for dynamic legged robot locomotion," in *Robotics: Science and Systems*, vol. 13. MIT Press Journals, 2017.

[9] M. Posa, C. Cantu, and R. Tedrake, "A direct method for trajectory optimization of rigid bodies through contact," *The International Journal of Robotics Research*, vol. 33, no. 1, pp. 69–81, 2014.

[10] A. Patel, S. L. Shield, S. Kazi, A. M. Johnson, and L. T. Biegler, "Contact-implicit trajectory optimization using orthogonal collocation," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 2242–2249, apr 2019.

[11] M. L. Felis, K. Mombaur, and A. Berthoz, "An optimal control approach to reconstruct human gait dynamics from kinematic data," in *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*. IEEE, 2015, pp. 1044–1051.

[12] W. Xi, Y. Yesilevskiy, and C. D. Remy, "Selecting gaits for economical locomotion of legged robots," *The International Journal of Robotics Research*, vol. 35, no. 9, pp. 1140–1154, 2016.

[13] W. Xi and C. D. Remy, "Optimal gaits and motions for legged robots," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014, pp. 3259–3265.

[14] R. Bonalli, A. Bylard, A. Cauligi, T. Lew, and M. Pavone, "Trajectory optimization on manifolds: A theoretically-guaranteed embedded sequential convex programming approach," in *Robotics: Science and systems*, 2019.

[15] E. Hairer, "Geometric integration of ordinary differential equations on manifolds," *BIT Numerical Mathematics*, vol. 41, no. 5, pp. 996–1007, 2001.

[16] S. Gros, M. Zanon, and M. Diehl, "Baumgarte stabilisation over the SO(3) rotation group for control," in *2015 54th IEEE Conference on Decision and Control (CDC)*. IEEE, 2015, pp. 620–625.

[17] J. Baumgarte, "Stabilization of constraints and integrals of motion in dynamical systems," *Computer Methods in Applied Mechanics and Engineering*, vol. 1, no. 1, pp. 1–16, 1972.

[18] P. Rabier and W. Rheinboldt, "Theoretical and numerical analysis of differential-algebraic equations," *Handbook of Numerical Analysis*, vol. 8, pp. 183–540, 2002.

[19] W. Blajer, "Methods for constraint violation suppression in the numerical simulation of constrained multibody systems - A comparative study," *Computer Methods in Applied Mechanics and Engineering*, vol. 200, no. 13-16, pp. 1568–1576, 2011.

[20] C. L. Darby, W. W. Hager, and A. V. Rao, "An hp-adaptive pseudospectral method for solving optimal control problems," *Optimal Control Applications and Methods*, vol. 32, no. 4, pp. 476–502, 2011.

[21] J. T. Betts, *Practical methods for optimal control and estimation using nonlinear programming*. SIAM Publications, 2010.

[22] E. Hairer, C. Lubich, and G. Wanner, *Geometric numerical integration: structure-preserving algorithms for ordinary differential equations*. Springer Science, 2006.

[23] E. Hairer and G. Wanner, *Solving ordinary differential Equations II: stiff and differential-algebraic equations*. Springer-Verlag, 1996.

[24] F. A. Potra and W. C. Rheinbold, "On the numerical solution of Euler-Lagrange equations," *Journal of Structural Mechanics*, vol. 19, no. 1, pp. 1–18, 1991.

[25] F. A. Potra and J. Yen, "Implicit numerical integration for Euler-Lagrange equations via tangent space parametrization," *Journal of Structural Mechanics*, vol. 19, no. 1, pp. 77–98, 1991.

[26] O. Bohigas, M. Manubens, and L. Ros, *Singularities of robot mechanisms: numerical computation and avoidance path planning*, ser. Mechanisms and Machine Science. Springer, 2017, vol. 41.

[27] R. Bordalba, J. M. Porta, and L. Ros, "Randomized planning of dynamic motions avoiding forward singularities," in *Advances in Robot Kinematics*, 2018, pp. 170–178.

[28] O. Bohigas, M. E. Henderson, L. Ros, M. Manubens, and J. M. Porta, "Planning singularity-free paths on closed-chain manipulators," *IEEE Transactions on Robotics*, vol. 29, no. 4, pp. 888–898, 2013.

[29] O. Bohigas, M. Manubens, and L. Ros, "Planning wrench-feasible motions for cable-driven hexapods," *IEEE Transactions on Robotics*, vol. 32, no. 2, pp. 442–451, 2016.

[30] J.-P. Berrut and L. N. Trefethen, "Barycentric Lagrange interpolation," *SIAM review*, vol. 46, no. 3, pp. 501–517, 2004.

[31] J. Nocedal and S. Wright, *Numerical optimization*. Springer Science & Business Media, 2006.

[32] C. R. Hargraves and S. W. Paris, "Direct trajectory optimization using nonlinear programming and collocation," *Journal of guidance, control, and dynamics*, vol. 10, no. 4, pp. 338–342, 1987.

[33] R. Featherstone, *Robot dynamics algorithms*. Kluwer, Norwell, MA, 1987.

[34] S. Manara, M. Gabiccini, A. Artoni, and M. Diehl, "On the integration of singularity-free representations of SO(3) for direct optimal control," *Nonlinear Dynamics*, vol. 90, pp. 1–19, 08 2017.

[35] J. M. Lee, *Introduction to smooth manifolds*. Springer, 2001.

[36] R. Bordalba, L. Ros, and J. M. Porta, "A randomized kinodynamic planner for closed-chain robotic systems," *IEEE Transactions on Robotics*, pp. 1–17, 2020.

[37] J. Yen, "Constrained equations of motion in multibody dynamics as ODEs on manifolds," *SIAM Journal on Numerical Analysis*, vol. 30, no. 2, pp. 553–568, 1993.

[38] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi – A software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, 2019.

[39] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical programming*, vol. 106, no. 1, pp. 25–57, 2006.

[40] HSL, "A collection of Fortran codes for large-scale scientific computation," 2007. [Online]. Available: http://www.hsl.rl.ac.uk

[41] R. Bordalba. Git repository of the software in this paper. [Online]. Available: https://github.com/rbordalba/collocation_constrained_systems

[42] ——, "Kinodynamic planning and control of closed-chain robotic systems," Ph.D. thesis. [Online]. Available: https://www.tdx.cat/handle/10803/674372

[43] ——. Companion video of this paper. [Online]. Available: https://drive.google.com/file/d/1J47pTHuqaan69s3b1M_lNbBvYAMhHZuI

[44] L. F. Shampine, "Vectorized adaptive quadrature in MATLAB," *Journal of Computational and Applied Mathematics*, vol. 211, no. 2, pp. 131–140, 2008.

[45] R. Featherstone and D. E. Orin, "Dynamics," in *Springer Handbook of Robotics*. Springer, 2016, pp. 37–66.

[46] I. Moreno-Caireta, E. Celaya, and L. Ros, "Model predictive control for a Mecanum-wheeled robot navigating among obstacles," *IFAC-PapersOnLine*, vol. 54, no. 6, pp. 119–125, 2021.

[47] Y. Li, Z. Littlefield, and K. E. Bekris, "Asymptotically optimal sampling-based kinodynamic planning," *The International Journal of Robotics Research*, vol. 35, no. 5, pp. 528–564, 2016.

[48] K. Hauser and Y. Zhou, "Asymptotically optimal planning by feasible kinodynamic planning in a state-cost space," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1431–1443, 2016.

[49] S. Karaman, M. R. Walter, A. Perez, E. Frazzoli, and S. Teller, "Anytime motion planning using the RRT*," in *IEEE International Conference on Robotics and Automation*, 2011, pp. 1478–1483.

Ricard Bordalba received the B.Sc. degree in Automation and Industrial Electronic Engineering in 2014 from the University of Lleida, Spain, the M.Sc. degree in Control and Automation in 2016 from Aalborg University, Denmark, and the Ph.D. degree in Robotics and Automation from the Universitat Politècnica de Catalunya (UPC) in 2021. He is currently working at Beta Robots. His research interests include kinematics, dynamics and control, with applications to robotics.

Tobias Schoels studied electrical engineering at RWTH Aachen from 2011-2014. After completing his B.Sc. degree, he joined the autonomous driving team at Mercedes-Benz Research & Development North America for nine months. Returning to Germany in 2015 he enrolled into a M.Sc. programme in Embedded Systems Engineering at Albert-Ludwigs-Universität Freiburg. Since fall 2018 he is a Ph.D. student at SYSCOP continuing his research on model predictive control based robot motion planning and control in dynamic environments.

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 and control of general multibody systems.

Josep M. Porta received the Engineering degree in Computer Science in 1994, and the Ph.D. degree (with honors) in Artificial Intelligence in 2001, both from the Universitat Politècnica de Catalunya, Spain. From 2001 to 2003 he held a postdoctoral position at the University of Amsterdam, pursuing research in autonomous robot localization using vision. Currently, he is an Associate Researcher of the Spanish National Research Council at the Institut de Robòtica i Informàtica Industrial (IRI, CSIC-UPC), Barcelona, Spain. His current research interests include planning under uncertainty and computational kinematics.

Moritz Diehl studied physics and mathematics at Heidelberg and Cambridge University from 1993-1999, and received his Ph.D. degree from Heidelberg University in 2001, at the Interdisciplinary Center for Scientific Computing. From 2006 to 2013, he was a professor with the Department of Electrical Engineering, KU Leuven University Belgium, and served as the Principal Investigator of KU Leuven's Optimization in Engineering Center OPTEC. In 2013 he moved to the University of Freiburg, Germany, where he heads the Systems Control and Optimization Laboratory, in the Department of Microsystems Engineering (IMTEK), and is also affiliated to the Department of Mathematics. His research interests are in optimization and control, spanning from numerical method development to applications in different branches of engineering, with a focus on embedded and on renewable energy systems.