sciendo

Int. J. Appl. Math. Comput. Sci., 2025, Vol. 35, No. 1, 83–96
DOI: 10.61822/amcs-2025-0007

amcs

DMOC–BASED ROBOT TRAJECTORY OPTIMIZATION WITH ANALYTICAL FIRST–ORDER INFORMATION

CARLA VILLANUEVA-PIÑON[a,*], GUSTAVO ARECHAVALETA[a]

[a] Robotics and Advanced Manufacturing Group
Center for Research and Advanced Studies of the National Polytechnic Institute
1062 Avenida Industrial Metalúrgica, 25900 Saltillo, Coahuila, Mexico
e-mail: carla.villanueva@cinvestav.edu.mx,
garechav@cinvestav.mx

Discrete mechanics and optimal control (DMOC) is a numerical optimal control framework capable of solving robot trajectory optimization problems. This framework has advantages over other direct collocation and multiple-shooting schemes. In particular, it works with a reduced number of decision variables due to the use of the forced discrete Euler–Lagrange (DEL) equation. Also, the transcription mechanism inherits the numerical benefits of variational integrators (i.e., momentum and energy conservation over a long time horizon with large time steps). We extend the benefits of DMOC to solve trajectory optimization problems for highly articulated robotic systems. We provide analytical evaluations of the forced DEL equation and its partial differentiation with respect to decision variables. The Lie group formulation of rigid-body motion and the use of multilinear algebra allow us to efficiently handle sparse tensor computations. The arithmetic complexity of the proposed strategy is analyzed, and it is validated by solving humanoid motion problems.

Keywords: discrete mechanics and optimal control, variational integrators, humanoid robot motions.

1. Introduction

Robot trajectory optimization represents an active research area. The problem to be solved consists of generating dynamically feasible robot motions under a wide variety of constraints such as torque and joint limits, obstacle avoidance, contacts, dynamic balance, among others. The aim of this work is to take advantage of variational integrators (Marsden and West, 2001) to formulate the underlying trajectory optimization problem in terms of discrete mechanics and optimal control (DMOC) (Ober-Blöbaum *et al*., 2011). Unlike previous works (Johnson and Murphey, 2009; Johnson *et al*., 2015), which are computationally expensive due to nested loops and multiple conditionals that must be handled with scalar operations, we propose a multivariable formulation that avoids the inherent sparsity of DMOC. The impact of the proposed sparsity-free DMOC is more notorious for tree-like kinematic structures (e.g., humanoid robots) in terms of arithmetic complexity and computational performance.

*Corresponding author

1.1. Problem formulation.

Let us formulate an optimal control problem (OCP) as follows: Given a highly articulated robotic system with n degrees of freedom (DoF) and a state vector

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{q} \\ \dot{\boldsymbol{q}} \end{bmatrix} \in \mathbb{R}^{2n}, \tag{1}$$

where $\{\boldsymbol{q}, \dot{\boldsymbol{q}}\} \in \mathbb{R}^n$ are the robot configuration, and joint velocities, respectively, find a control law (if it exists) that leads the system from its initial state to a final one by solving the following OCP:

$$\underset{\boldsymbol{u}(t)\in\boldsymbol{U}}{\text{minimize}} \quad \int_{t_0}^{t_f} C(\boldsymbol{x}(t), \boldsymbol{u}(t), t)\,\mathrm{d}t, \tag{2}$$

$$\begin{aligned} \text{subject to} \quad & \dot{\boldsymbol{x}}(t) = \boldsymbol{f}_x(\boldsymbol{x}(t), \boldsymbol{u}(t), t), \\ & \boldsymbol{g}(\boldsymbol{x}(t), \boldsymbol{u}(t), t) \leq 0, \\ & \boldsymbol{x}_{lb} \leq \boldsymbol{x}(t) \leq \boldsymbol{x}_{ub}, \\ & \boldsymbol{x}(t_0) = \boldsymbol{x}_0, \;\; \boldsymbol{x}(t_f) = \boldsymbol{x}_f, \end{aligned} \tag{3}$$

where $\boldsymbol{U} \subset \mathbb{R}^n$ defines the admissible control domain, $C(\cdot)$ stands for the integrand of the cost functional to be minimized, $\boldsymbol{g}(\cdot) \in \mathbb{R}^p$ contains the set of path constraints,

and $\dot{\boldsymbol{x}}(t) = \boldsymbol{f}_x(\boldsymbol{x}(t), \boldsymbol{u}(t), t)$ stands for the robot's dynamics expressed in its state equation form. The initial and final time are expressed as t_0 and t_f, respectively. The constraint vectors $\boldsymbol{x}_{lb} \in \mathbb{R}^{2n}$ and $\boldsymbol{x}_{ub} \in \mathbb{R}^{2n}$ are the lower and upper bounds of the system's state, and $\{\boldsymbol{x}_0, \boldsymbol{x}_f\} \in \mathbb{R}^{2n}$ are the initial and final states, respectively.

The continuous OCP stated in (2) and (3) could be transformed into a nonlinear program (NLP), such that the robot's equations of motion become defect constraints to be satisfied as follows:

$$\begin{aligned} \underset{\boldsymbol{z}}{\text{minimize}} \quad & J(\boldsymbol{z}), \\ \text{subject to} \quad & \boldsymbol{c}_l \leq \boldsymbol{c}(\boldsymbol{z}) \leq \boldsymbol{c}_u, \\ & \boldsymbol{z}_l \leq \boldsymbol{z} \leq \boldsymbol{z}_u, \end{aligned} \tag{4}$$

where J is the cost function, $\boldsymbol{c}(z) \in \mathbb{R}^{n_{cns}}$ is a stack of defect, path, and boundary constraints as $\boldsymbol{\Phi}$, $\boldsymbol{\Upsilon}$, and $\boldsymbol{e}$, respectively. Lower and upper bounds over constraints are $\boldsymbol{c}_U$ and $\boldsymbol{c}_L$, respectively. The vector of decision variables is $\boldsymbol{z} = [\boldsymbol{x}_0^\top \dots \boldsymbol{x}_{N-1}^\top \boldsymbol{u}_0^\top \dots \boldsymbol{u}_{N-1}^\top]^\top \in \mathbb{R}^{3nN}$ where $\boldsymbol{x}_k$ and $\boldsymbol{u}_k$ represent the state and control input evaluated at time step k, respectively, and k iterates from 0 to $N-1$. The vectors $\boldsymbol{z}_l$ and $\boldsymbol{z}_u$ denote the lower and upper bounds of $\boldsymbol{z}$, respectively. The number of discrete points is N.

Since the NLP stated in (4) turns out to be a large-scale optimization problem that demands an important amount of computation, it is desirable to analytically evaluate the robot's equations of motion together with their partial derivatives for decreasing the computation time without affecting the numerical sensitivity of the solution.

1.2. Related work. The maturity of direct collocation and shooting methods has been translated into out-of-the-box numerical optimal control (NOC) solvers (Houska *et al*., 2011; Andersson *et al*., 2019; Agamawi and Rao, 2020). In general, they transcribe the original continuous optimal control problem into a nonlinear program (NLP), such that the system dynamics become defect constraints to be satisfied (Kelly, 2017; Betts, 2010). It has been well established that direct methods offer superior convergence properties compared to indirect transcription mechanisms, that are sensitive to initial conditions (Rao, 2009). Regardless of the preferred transcription method, there exist two main strategies known as multiple shooting and collocation. Thus, significant work has been done to develop sophisticated NOC frameworks for solving small to medium size optimal control problems based on direct methods. As a result, out-of-the-box optimal control solvers are now available such as ACADO toolkit (Houska *et al*., 2011), DirCol5i (Kelly, 2019), MUSCOD-II (Leineweber *et al*., 2003), PSOPT (Becerra, 2010), CasADi (Andersson *et al*., 2019) and CGPOPS (Agamawi and Rao, 2020). Some of them are open-source projects such as PSOPT and CasADi. ACADO toolkit is no longer supported while MUSCOD-II and CGPOPS are commercial options. All of them have been conceived to solve general purpose optimal control problems with special attention to chemical (Leineweber *et al*., 2003) and aerospace (Betts and Erb, 2003) engineering. Other trajectory optimization strategies have been suggested for linear systems for Liu *et al*. (2022).

Recent efforts have been dedicated to develop robot trajectory optimization strategies based on direct shooting and collocation (Howell *et al*., 2019; Mastalli *et al*., 2020; Hereid and Ames, 2017; Cardona-Ortiz *et al*., 2020). In particular, the direct shooting strategy introduced by Mayne (1966), known as differential dynamic programming (DDP), has been widely applied for robot motion generation (Budhiraja *et al*., 2018; Howell *et al*., 2019; Mastalli *et al*., 2020), and it is sparsity-free by design. However, path constraints together with simple state and control bounds cannot be added to the problem in a straightforward manner.

On the other hand, variational integrators have been suggested for computing the robot's equations of motion in generalized coordinates (Johnson and Murphey, 2009), and their linearization has been studied for control design purposes, e.g., the LQR for moving an articulated marionette robot (Johnson *et al*., 2015). Special attention has been given to improve the computational demands to calculate the discrete Euler–Lagrange (DEL) equation and its linearization for generating dynamically feasible robot motions (Lee *et al*., 2020; Fan *et al*., 2018). It is important to mention that recursive algorithms are based on the geometric and spatial algebra formulations (Park *et al*., 2018; Featherstone, 2014). In particular, the geometric and variational integrators approaches could work together to cope with geometric discrete optimal control problems over Lie groups (Kobilarov and Marsden, 2011).

DMOC, which is based on variational integrators, has been successfully applied for robot motion generation under different conditions (Sun *et al*., 2016; Zhang *et al*., 2018; Manns and Mombaur, 2017; Manchester *et al*., 2019). The methods proposed by Sun *et al*. (2016) treated the walking gait of five degrees of freedom planar biped robots (i.e., support and swing legs). In DMOC, holonomic constraints are studied by Leyendecker *et al*. (2010), and non-holonomic constraints have been also considered to solve trajectory optimization problems for car-like vehicles as described by Kobilarov and Sukhatme (2007). The multi-phase DMOC method proposed by Zhang *et al*. (2018) demonstrates how to optimize the trajectories of quadrotor aerial vehicles. In the context of contact dynamics, the method proposed by Manchester *et al*. (2019) deals with linear complementarity conditions for robot motion generation under contact constraints. Following the same vein, the main contributions of

our work are twofold. First, the inherent sparsity of DMOC, when highly articulated robots come to play, is carefully removed. Second, efficient recursive algorithms are proposed to analytically evaluate the robot's DEL equation and its linearization.

The paper is organized as follows. First, DMOC is briefly described in Section 2. In Section 3, we apply Lie groups and algebras for robot motion to analytical evaluate the partial derivatives of robot's DEL equation with respect to decision variables. In Section 4 we use multilinear algebra to avoid naive tensor calculations, i.e., sparsity. Numerical results are provided in Section 5. Finally, some concluding remarks are given in Section 6.

2. Variational integrators and DMOC

Variational integrators can be used to obtain the robot's equations of motion. Since the robot is considered as a non-conservative mechanical system subject to external forces, we apply the d'Alembert–Lagrange principle as follows:

$$\delta \int_{t_0}^{t_f} L\left(\boldsymbol{q}(t), \dot{\boldsymbol{q}}(t)\right)\, \mathrm{d}t + \int_{t_0}^{t_f} \boldsymbol{F}(\boldsymbol{q}(t), \dot{\boldsymbol{q}}(t), \boldsymbol{u}(t)) \cdot \delta \boldsymbol{q}(t)\, \mathrm{d}t = 0, \quad (5)$$

where L is the Lagrangian function of a mechanical system defined as $L(\boldsymbol{q}(t), \dot{\boldsymbol{q}}(t)) = K(\boldsymbol{q}(t), \dot{\boldsymbol{q}})(t) - U(\boldsymbol{q}(t))$, such that $K(\boldsymbol{q}(t), \dot{\boldsymbol{q}}(t))$ and $U(\boldsymbol{q}(t))$ are the kinetic and potential energies (Marsden and West, 2001). The second term in (5) represents the virtual work exerted on the mechanical system, where $\boldsymbol{F}(\boldsymbol{q}(t), \dot{\boldsymbol{q}}(t), \boldsymbol{u}(t)) \in \mathbb{R}^n$ stands for the Lagrangian control force and the variation of $\boldsymbol{q}(t)$ is denoted by $\delta\boldsymbol{q}(t)$ (Ober-Blöbaum *et al.*, 2011). Thus, variational integrators are derived from the discrete version of equation (5) such that

$$L_d(\boldsymbol{q}_k, \boldsymbol{q}_{k+1}) \approx \int_{k\Delta t}^{(k+1)\Delta t} L(\boldsymbol{q}(t), \dot{\boldsymbol{q}}(t))\, \mathrm{d}t, \quad (6)$$

where $\boldsymbol{q}(t)$ is discretized into N segments $\boldsymbol{q}_k|_{k=0}^{N}$, and L_d is known as the discrete Lagrangian. This function can be defined by any quadrature rule. In this work we apply the midpoint approximation

$$L_d(\boldsymbol{q}_k, \boldsymbol{q}_{k+1}) = L\left(\boldsymbol{q}_d, \dot{\boldsymbol{q}}_d\right) \Delta t, \quad (7)$$

such that

$$\boldsymbol{q}_d = \frac{q_k + q_{k+1}}{2}, \quad \dot{\boldsymbol{q}}_d = \frac{q_{k+1} - q_k}{\Delta t}$$

with Δt as the time increment. Similarly, the virtual work is approximated with a combination of a midpoint and forward rectangle rules as follows:

$$\boldsymbol{F}_d(\boldsymbol{q}_k, \boldsymbol{q}_{k+1}, \boldsymbol{u}_k)(\delta\boldsymbol{q}_k + \delta\boldsymbol{q}_{k+1}) \approx \int_{k\Delta t}^{(k+1)\Delta t} \boldsymbol{F}(\boldsymbol{q}(t), \dot{\boldsymbol{q}}(t), \boldsymbol{u}(t)) \cdot \delta\boldsymbol{q}(t)\mathrm{d}t, \quad (8)$$

where $\boldsymbol{F}_d$ is known as the discrete Lagrangian control force,

$$\boldsymbol{F}_d(\boldsymbol{q}_k, \boldsymbol{q}_{k+1}, \boldsymbol{u}_k) = \boldsymbol{F}\left(\boldsymbol{q}_d, \dot{\boldsymbol{q}}_{\boldsymbol{d}}, \boldsymbol{u}_d\right) \Delta t. \quad (9)$$

such that $\boldsymbol{u}_d$ is the control input approximation given by a numerical integration rule.

The discrete d'Alembert–Lagrange principle stands for

$$\sum_{k=0}^{N-1} \left(D_1 L_d(\boldsymbol{q}_k, \boldsymbol{q}_{k+1}) \cdot \delta\boldsymbol{q}_k + D_2 L_d(\boldsymbol{q}_k, \boldsymbol{q}_{k+1}) \cdot \delta\boldsymbol{q}_{k+1}\right) + \sum_{k=0}^{N-1} \boldsymbol{F}_d(\boldsymbol{q}_k, \boldsymbol{q}_{k+1}, \boldsymbol{u}_k) \cdot (\delta\boldsymbol{q}_k + \delta\boldsymbol{q}_{k+1}) = 0, \quad (10)$$

for all variations $\delta\boldsymbol{q}(t)$ of $\boldsymbol{q}(t)$ that vanish at endpoints $\boldsymbol{q}_0$ and $\boldsymbol{q}_N$. The slot derivative is denoted as D_i, which represents the partial differentiation of L_d with respect to its i-th argument. By applying a discrete integration by parts, we obtain

$$\sum_{k=1}^{N-1} \left(D_2 L_d(\boldsymbol{q}_{k-1}, \boldsymbol{q}_k) + D_1 L_d(\boldsymbol{q}_k, \boldsymbol{q}_{k+1}) + \boldsymbol{F}_d(\boldsymbol{q}_{k-1}, \boldsymbol{q}_k, \boldsymbol{u}_k) + \boldsymbol{F}_d(\boldsymbol{q}_k, \boldsymbol{q}_{k+1}, \boldsymbol{u}_k)\right) \cdot \delta\boldsymbol{q}_k = 0. \quad (11)$$

From the fact that (11) must hold for all variations $\delta\boldsymbol{q}_k$, the forced discrete Euler–Lagrange (DEL) equation is

$$D_2 L_d(\boldsymbol{q}_{k-1}, \boldsymbol{q}_k) + D_1 L_d(\boldsymbol{q}_k, \boldsymbol{q}_{k+1}) + \boldsymbol{F}_d(\boldsymbol{q}_{k-1}, \boldsymbol{q}_k, \boldsymbol{u}_k) + \boldsymbol{F}_d(\boldsymbol{q}_k, \boldsymbol{q}_{k+1}, \boldsymbol{u}_k) = \boldsymbol{0}. \quad (12)$$

The computation of slot derivatives in (12) uses the chain rule to differentiate (7). Therefore,

$$D_1 L_d(\boldsymbol{q}_k, \boldsymbol{q}_{k+1}) = \left(\frac{\partial L}{\partial \boldsymbol{q}_d}\frac{\partial \boldsymbol{q}_d}{\partial \boldsymbol{q}_k} + \frac{\partial L}{\partial \dot{\boldsymbol{q}}_d}\frac{\partial \dot{\boldsymbol{q}}_d}{\partial \boldsymbol{q}_k}\right)\Delta t, = \frac{1}{2}\frac{\partial L}{\partial \boldsymbol{q}_d}\Delta t - \frac{\partial L}{\partial \dot{\boldsymbol{q}}_d}, \quad (13)$$

$$D_2 L_d(\boldsymbol{q}_k, \boldsymbol{q}_{k+1}) = \left(\frac{\partial L}{\partial \boldsymbol{q}_d}\frac{\partial \boldsymbol{q}_d}{\partial \boldsymbol{q}_{k+1}} + \frac{\partial L}{\partial \dot{\boldsymbol{q}}_d}\frac{\partial \dot{\boldsymbol{q}}_d}{\partial \boldsymbol{q}_{k+1}}\right)\Delta t, = \frac{1}{2}\frac{\partial L}{\partial \boldsymbol{q}_d}\Delta t + \frac{\partial L}{\partial \dot{\boldsymbol{q}}_d}. \quad (14)$$

Notice that the form of (13) and (14) depends on the quadrature rule, in this case the midpoint rule.

2.1. Direct transcription. The trajectory optimization problem stated in (4) implies the fulfillment of the robot's equations of motion along the optimized trajectory. For DMOC this means that the forced DEL equation given in (12) must be satisfied. Thus, a vector of stacked defect constraints is defined as

$$\boldsymbol{\Phi} = \begin{bmatrix} \boldsymbol{f}_1(\boldsymbol{q}_0, \boldsymbol{q}_1, \boldsymbol{q}_2, \boldsymbol{u}_1) \\ \boldsymbol{f}_2(\boldsymbol{q}_1, \boldsymbol{q}_2, \boldsymbol{q}_3, \boldsymbol{u}_2) \\ \vdots \\ \boldsymbol{f}_{N-1}(\boldsymbol{q}_{N-2}, \boldsymbol{q}_{N-1}, \boldsymbol{q}_N, \boldsymbol{u}_{N-1}) \end{bmatrix}, \tag{15}$$

where $\boldsymbol{f}_k$ corresponds to (12), such that k iterates from 1 to $N-1$ for guaranteeing the feasibility of system dynamics along the optimized trajectory.

Similarly to the discrete Lagrangian defined in (6) and (7), discrete variational principles can be applied to evaluate the functional (2) at each time step as

$$C_d(\boldsymbol{q}_k, \boldsymbol{q}_{k+1}, \boldsymbol{u}_k) \approx \int_{k\Delta t}^{(k+1)\Delta t} C(\boldsymbol{q}(t), \dot{\boldsymbol{q}}(t), \boldsymbol{u}(t))\, \mathrm{d}t. \tag{16}$$

Then, the expression (16) is used to obtain the discrete version of (2), i.e., the cost function in (4), as

$$J(\boldsymbol{q}_k, \boldsymbol{u}_k) = \sum_{k=0}^{N-1} C_d(\boldsymbol{q}_k, \boldsymbol{q}_{k+1}, \boldsymbol{u}_k),$$
$$C_d(\boldsymbol{q}_k, \boldsymbol{q}_{k+1}, \boldsymbol{u}_k) = C\left(\boldsymbol{q}_d, \dot{\boldsymbol{q}}_d, \boldsymbol{u}_d\right) \Delta t.$$

Variational integrators are also used to obtain discrete path constraints as a set of algebraic equations acting on the discrete curve. The vector of stacked constraints is

$$\boldsymbol{\Upsilon} = \begin{bmatrix} \boldsymbol{g}_{d_0} \\ \boldsymbol{g}_{d_1} \\ \vdots \\ \boldsymbol{g}_{d_{N-1}} \end{bmatrix} \in \mathbb{R}^{pN} \tag{17}$$

where

$$\boldsymbol{g}_{d_k}(\boldsymbol{q}_k, \boldsymbol{q}_{k+1}, \boldsymbol{u}_k) \approx \int_{k\Delta t}^{(k+1)\Delta t} \boldsymbol{g}(\boldsymbol{x}(t), \boldsymbol{u}(t))\, \mathrm{d}t \in \mathbb{R}^p, \tag{18}$$

such that p is the number of constraints at each node of the discrete curve given by $\{\boldsymbol{q}_k\}_{k=0}^N$. Now, the boundary conditions appearing in (3), i.e., $\{\boldsymbol{q}(t_0), \dot{\boldsymbol{q}}(t_0)\}$ and $\{\boldsymbol{q}(t_f), \dot{\boldsymbol{q}}(t_f)\}$, should be consistent with the discrete representation. Notice that joint velocities are not directly available in (12). However, joint velocities can be transformed into momentum by applying the standard Legendre transform

$$(\boldsymbol{q}, \dot{\boldsymbol{q}}) \to (\boldsymbol{q}, \boldsymbol{p}) = (\boldsymbol{q}, D_2 L(\boldsymbol{q}, \dot{\boldsymbol{q}})). \tag{19}$$

From (12), the discrete Legendre transform implies (Ober-Blöbaum *et al.*, 2011):

$$\begin{aligned}(\boldsymbol{q}_{k-1}, \boldsymbol{q}_k) \to (\boldsymbol{q}_k, \boldsymbol{p}_k) =& (\boldsymbol{q}_k, D_2 L_d(\boldsymbol{q}_{k-1}, \boldsymbol{q}_k) \\ &+ \boldsymbol{F}_d(\boldsymbol{q}_{k-1}, \boldsymbol{q}_k, \boldsymbol{u}_{k-1}), \\ (\boldsymbol{q}_{k-1}, \boldsymbol{q}_k) \to (\boldsymbol{q}_{k-1}, \boldsymbol{p}_{k-1}) =& (\boldsymbol{q}_{k-1}, -D_1 L_d(\boldsymbol{q}_{k-1}, \boldsymbol{q}_k) \\ &- \boldsymbol{F}_d(\boldsymbol{q}_{k-1}, \boldsymbol{q}_k, \boldsymbol{u}_{k-1}).\end{aligned}$$

By equating the continuous and discrete momenta, the following momentum conditions are obtained:

$$\begin{aligned} D_2 L(\boldsymbol{q}(t_0), \dot{\boldsymbol{q}}(t_0)) + D_1 L_d(\boldsymbol{q}_0, \boldsymbol{q}_1) \\ + \boldsymbol{F}_d(\boldsymbol{q}_0, \boldsymbol{q}_1, \boldsymbol{u}_0) = \boldsymbol{0}, \end{aligned} \tag{20}$$

$$\begin{aligned} -D_2 L(\boldsymbol{q}(t_f), \dot{\boldsymbol{q}}(t_f)) + D_2 L_d(\boldsymbol{q}_{N-1}, \boldsymbol{q}_N) \\ + \boldsymbol{F}_d(\boldsymbol{q}_{N-1}, \boldsymbol{q}_N, \boldsymbol{u}_{N-1}) = \boldsymbol{0}, \end{aligned} \tag{21}$$

where the expression (20) could be rewritten as $\boldsymbol{p}(t_0) - \boldsymbol{p}_0 = 0$ and (21) as $\boldsymbol{p}(t_f) - \boldsymbol{p}_N = 0$. In addition, boundary conditions related to the initial and final configurations can be directly expressed as

$$\boldsymbol{q}(t_0) - \boldsymbol{q}_0 = \boldsymbol{0}, \tag{22}$$

$$\boldsymbol{q}(t_f) - \boldsymbol{q}_N = \boldsymbol{0}. \tag{23}$$

The stack of boundary conditions is represented as $\boldsymbol{e} = [\boldsymbol{e}_0^\top,\ \boldsymbol{e}_N^\top]^\top \in \mathbb{R}^{4n}$ where $\boldsymbol{e}_0$ contains the expressions (20) and (22), while $\boldsymbol{e}_N$ considers (21) and (23).

2.2. Sparsity patterns of first-order terms. Since the NLP solver asks for first-order information of the problem, the computation of the constraint Jacobian, denoted as $\frac{\partial \boldsymbol{c}(\boldsymbol{z})}{\partial \boldsymbol{z}}$, is of great importance. Its sparse structure follows the patterns

$$\begin{cases} \dfrac{\partial \boldsymbol{f}_i}{\partial \boldsymbol{z}_k} \in \mathbb{R}^{n \times 2n} & \text{if } i = k, \\ \boldsymbol{0} & \text{otherwise,} \end{cases}$$

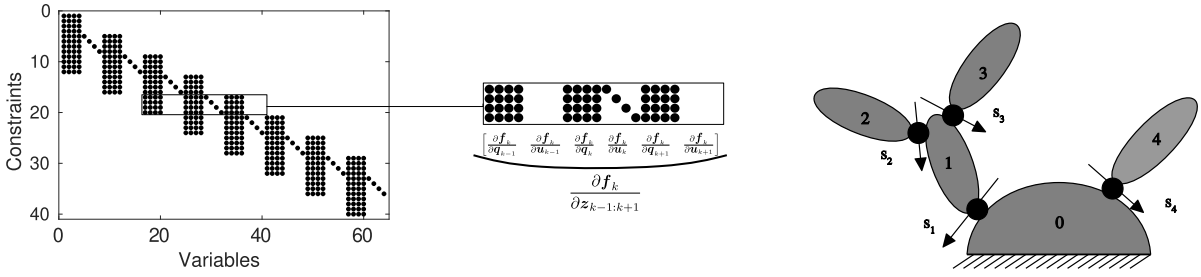
$$\begin{cases} \dfrac{\partial \boldsymbol{g}_{d_i}}{\partial \boldsymbol{z}_k} \in \mathbb{R}^{p \times 2n} & \text{if } i = k, \\ \boldsymbol{0} & \text{otherwise,} \end{cases}$$

$$\begin{cases} \dfrac{\partial \boldsymbol{e}}{\partial \boldsymbol{q}_k} \in \mathbb{R}^{2n \times n} & \text{if } k = 0 \text{ or } k = N, \\ \boldsymbol{0} & \text{otherwise,} \end{cases}$$

$$\frac{\partial \boldsymbol{e}}{\partial \boldsymbol{u}_k} \in \mathbb{R}^{2n \times n} = \boldsymbol{0},$$

where $\boldsymbol{z}_k = [\boldsymbol{q}_k^\top, \boldsymbol{u}_k^\top]^\top \in \mathbb{R}^{2n}$. Figure 1(a) illustrates the structure of $\partial \boldsymbol{c}(\boldsymbol{z})/\partial \boldsymbol{z}$ with $N = 8$ for a tree-like mechanism depicted in Fig. 1(b). Note that the patterns form dense blocks that correspond to the partial derivatives of the system dynamics with respect to the discrete robot configuration and control at $k-1$, k and $k+1$ as

$$\frac{\partial \boldsymbol{f}_k}{\partial \boldsymbol{z}_{k-1:k+1}} = \begin{bmatrix} \frac{\partial \boldsymbol{f}_k}{\partial \boldsymbol{q}_{k-1}} & \frac{\partial \boldsymbol{f}_k}{\partial \boldsymbol{u}_{k-1}} & \frac{\partial \boldsymbol{f}_k}{\partial \boldsymbol{q}_k} & \frac{\partial \boldsymbol{f}_k}{\partial \boldsymbol{u}_k} & \frac{\partial \boldsymbol{f}_k}{\partial \boldsymbol{q}_{k+1}} & \frac{\partial \boldsymbol{f}_k}{\partial \boldsymbol{u}_{k+1}} \end{bmatrix}, \tag{24}$$



(a) (b)

Fig. 1. Sparsity of the constraint Jacobian depends on the number of constraints, discretization points, and degrees of freedom of the articulated rigid-body system. The sparsity of the constraint Jacobian due to the DMOC transcription: it is observed that each block along Jacobian's diagonal also contains the induced sparsity of the articulated rigid-body system (a). A 4 DoF open-chain rigid-body system composed by 5 bodies and 4 axes of admissible motion denoted by $\boldsymbol{s}_i$: the body labeled with 0 is fixed, and it corresponds to the root of the tree-like kinematic mechanism (b).

where the partial derivatives are calculated as follows:

$$\frac{\partial \boldsymbol{f}_k}{\partial \boldsymbol{q}_{k-1}} = D_1D_2L_d(\boldsymbol{q}_{k-1},\boldsymbol{q}_k) + D_1\boldsymbol{F}_d(\boldsymbol{q}_{k-1},\boldsymbol{q}_k,\boldsymbol{u}_{k-1}), \tag{25}$$

$$\frac{\partial \boldsymbol{f}_k}{\partial \boldsymbol{q}_k} = D_2D_2L_d(\boldsymbol{q}_{k-1},\boldsymbol{q}_k) + D_1D_1L_d(\boldsymbol{q}_k,\boldsymbol{q}_{k+1}) + D_1\boldsymbol{F}_d(\boldsymbol{q}_k,\boldsymbol{q}_{k+1},\boldsymbol{u}_k), \tag{26}$$

$$\frac{\partial \boldsymbol{f}_k}{\partial \boldsymbol{u}_k} = D_3\boldsymbol{F}_d(\boldsymbol{q}_k,\boldsymbol{q}_{k+1},\boldsymbol{u}_k), \tag{27}$$

$$\frac{\partial \boldsymbol{f}_k}{\partial \boldsymbol{q}_{k+1}} = D_2D_1L_d(\boldsymbol{q}_k,\boldsymbol{q}_{k+1}) + D_2\boldsymbol{F}_d(\boldsymbol{q}_k,\boldsymbol{q}_{k+1},\boldsymbol{u}_k). \tag{28}$$

Note that second slot derivatives appear in (25), (26) and (28). Thus, the chain rule of differentiation is employed to obtain the following expressions:

$$D_1D_1L_d(\boldsymbol{q}_k,\boldsymbol{q}_{k+1}) = \frac{1}{4}\frac{\partial^2 L(\cdot)}{\partial \boldsymbol{q}_d\partial \boldsymbol{q}_d}\Delta t - \frac{1}{2}\frac{\partial^2 L(\cdot)}{\partial \dot{\boldsymbol{q}}_d\partial \boldsymbol{q}_d} - \frac{1}{2}\frac{\partial^2 L(\cdot)}{\partial \boldsymbol{q}_d\partial \dot{\boldsymbol{q}}_d} + \frac{\partial^2 L(\cdot)}{\partial \dot{\boldsymbol{q}}_d\partial \dot{\boldsymbol{q}}_d}\frac{1}{\Delta t}, \tag{29}$$

$$D_2D_1L_d(\boldsymbol{q}_k,\boldsymbol{q}_{k+1}) = \frac{1}{4}\frac{\partial^2 L(\cdot)}{\partial \boldsymbol{q}_d\partial \boldsymbol{q}_d}\Delta t + \frac{1}{2}\frac{\partial^2 L(\cdot)}{\partial \dot{\boldsymbol{q}}_d\partial \boldsymbol{q}_d} - \frac{1}{2}\frac{\partial^2 L(\cdot)}{\partial \boldsymbol{q}_d\partial \dot{\boldsymbol{q}}_d} - \frac{\partial^2 L(\cdot)}{\partial \dot{\boldsymbol{q}}_d\partial \dot{\boldsymbol{q}}_d}\frac{1}{\Delta t}, \tag{30}$$

$$D_1D_2L_d(\boldsymbol{q}_k,\boldsymbol{q}_{k+1}) = \frac{1}{4}\frac{\partial^2 L(\cdot)}{\partial \boldsymbol{q}_d\partial \boldsymbol{q}_d}\Delta t - \frac{1}{2}\frac{\partial^2 L(\cdot)}{\partial \dot{\boldsymbol{q}}_d\partial \boldsymbol{q}_d} + \frac{1}{2}\frac{\partial^2 L(\cdot)}{\partial \boldsymbol{q}_d\partial \dot{\boldsymbol{q}}_d} - \frac{\partial^2 L(\cdot)}{\partial \dot{\boldsymbol{q}}_d\partial \dot{\boldsymbol{q}}_d}\frac{1}{\Delta t}, \tag{31}$$

$$D_2D_2L_d(\boldsymbol{q}_k,\boldsymbol{q}_{k+1}) = \frac{1}{4}\frac{\partial^2 L(\cdot)}{\partial \boldsymbol{q}_d\partial \boldsymbol{q}_d}\Delta t + \frac{1}{2}\frac{\partial^2 L(\cdot)}{\partial \dot{\boldsymbol{q}}_d\partial \boldsymbol{q}_d} + \frac{1}{2}\frac{\partial^2 L(\cdot)}{\partial \boldsymbol{q}_d\partial \dot{\boldsymbol{q}}_d} + \frac{\partial^2 L(\cdot)}{\partial \dot{\boldsymbol{q}}_d\partial \dot{\boldsymbol{q}}_d}\frac{1}{\Delta t}. \tag{32}$$

3. Computation of robot dynamics

The aim here is to describe a geometric procedure to analytically evaluate the robot equations of motion based on variational integrators. The geometric procedure involves the use of matrix Lie groups $SO(3)$ and $SE(3)$ and their associated algebra that are employed in robotics to represent articulated rigid bodies (Johnson and Murphey, 2009; Park *et al.*, 2018). From the geometric formulation, a recursive and sparsity-free computation of the robot dynamics can be performed.

3.1. Discrete Lagrangian computation. The discrete Lagrangian $L(\boldsymbol{q}_d,\dot{\boldsymbol{q}}_d)$ is defined as

$$L(\boldsymbol{q}_d,\dot{\boldsymbol{q}}_d) = K(\boldsymbol{q}_d,\dot{\boldsymbol{q}}_d) - U(\boldsymbol{q}_d), \tag{33}$$

where $K(\boldsymbol{q}_d,\dot{\boldsymbol{q}}_d)$ and $U(\boldsymbol{q}_d)$ represent respectively the kinetic and potential energies of the robotic system composed by articulated rigid bodies. Thus, the kinetic energy can be calculated through the chain of rigid bodies as

$$K(\boldsymbol{q}_d,\dot{\boldsymbol{q}}_d) = \frac{1}{2}\sum_i \boldsymbol{\nu}_i(\boldsymbol{q}_d,\dot{\boldsymbol{q}}_d)^\top \boldsymbol{M}_i\boldsymbol{\nu}_i(\boldsymbol{q}_d,\dot{\boldsymbol{q}}_d), \tag{34}$$

where $\boldsymbol{M}_i \in \mathbb{R}^{6\times 6}$ and $\boldsymbol{\nu}_i \in \mathbb{R}^6$ are the inertia matrix and twist of the i-th body, respectively. The twist $\boldsymbol{\nu}_i = [\boldsymbol{v}_i^T\ \boldsymbol{\omega}_i^T]^T \in \mathbb{R}^6$ contains the linear and angular velocities of i-th body, $\boldsymbol{v}_i \in \mathbb{R}^3$ and $\boldsymbol{\omega}_i \in \mathbb{R}^3$, respectively. From the following recursion, the twist can be computed as

$$\boldsymbol{\nu}_i(\boldsymbol{q}_d,\dot{\boldsymbol{q}}_d) = \mathrm{Ad}_{G_i^{\lambda(i)}}\boldsymbol{\nu}_{\lambda(i)}(\boldsymbol{q}_d,\dot{\boldsymbol{q}}_d) + \boldsymbol{s}_i\dot{q}_{d_i} \in \mathbb{R}^6, \tag{35}$$

where $\boldsymbol{s}_i \in \mathbb{R}^6$ stands for the axis of motion expressed in local coordinates (i.e., the reference frame attached to the i-th body), $\dot{q}_{d_i}$ is the i-th discrete joint velocity, and $\mathrm{Ad}_{G_i^{\lambda(i)}}$ is known as the Adjoint operator, which expresses the twist $\boldsymbol{\nu}_{\lambda(i)}$ in the i-th reference frame of the corresponding rigid body. The Adjoint operator is constructed by means of $\boldsymbol{G} \in SE(3)$ that encodes the position and orientation of the rigid body (Selig, 2004; Park *et al.*, 2018). Notice that the body parent of the i-th body towards the root is denoted by $\lambda(i)$. For example, according to Fig. 1(b), $\lambda(2) = \lambda(3) = 1$, which means that bodies labeled as 2 and 3 have the same body parent, i.e., their predecessor.

The evaluation of the potential energy in (33) can be obtained from the following expression:

$$U(\boldsymbol{q}_d) = \sum_i m_i \, \boldsymbol{g}^\top \boldsymbol{r}_i(\boldsymbol{q}_d), \tag{36}$$

where $\boldsymbol{g} \in \mathbb{R}^3$ is the gravity vector, m_i is the mass of the i-th body along the kinematic chain, and $\boldsymbol{r}_i(\boldsymbol{q}_d) \in \mathbb{R}^3$ represents the center of mass (CoM) in terms of the discrete configuration. Since the CoM in (36) is expressed in the inertial frame, its computation implies

$$\begin{bmatrix} \boldsymbol{r}_i(\boldsymbol{q}_d) \\ 1 \end{bmatrix} = \boldsymbol{G}_0^i(\boldsymbol{q}_d) \begin{bmatrix} \boldsymbol{r}_{b_i} \\ 1 \end{bmatrix}, \tag{37}$$

where $\boldsymbol{r}_{b_i} \in \mathbb{R}^3$ is the CoM of the i-th body expressed in local coordinates, and $\boldsymbol{G}_0^i(\boldsymbol{q}_d)$ represents the product of exponentials formula (Brockett, 2005), which encodes the local transformations along the robot kinematics from the inertial to the i-th reference frames attached to the robot as

$$\boldsymbol{G}_0^i(\boldsymbol{q}_d) = \boldsymbol{G}_0^1(0)\, e^{[s_1]q_1} \boldsymbol{G}_1^2(0)\, e^{[s_2]q_2} \cdots \boldsymbol{G}_{i-1}^i(0)\, e^{[s_i]q_i}, \tag{38}$$

where $\boldsymbol{e}^{[s_i]q_i}$ is the matrix exponential map, $[\boldsymbol{s}_i]$ is the matrix form of the screw axis of motion expressed in local coordinates, q_i is the i-th element of the robot configuration, and $\boldsymbol{G}_{i-1}^i(0)$ is the i-th reference frame home configuration (for details, see the work of Park *et al.*, (2018)).

3.2. Computation of the DEL equation. The evaluation of (12), which is the DEL equation, implies the partial differentiation of the discrete Lagrangian with respect to $\boldsymbol{q}_d$ and $\dot{\boldsymbol{q}}_d$ as it is observed in (13) and (14). As a remark, hereafter we omit the explicit dependence on $\boldsymbol{q}_d$ and $\dot{\boldsymbol{q}}_d$ in terms like $\boldsymbol{r}_i$ and $\boldsymbol{\nu}_i$ for readability purposes. By plugging (34), (35) and (36) in (33), the expressions of the resulting partial differentiation are

$$\frac{\partial L}{\partial \boldsymbol{q}_d} = \sum_i \left(\boldsymbol{\nu}_i^\top \boldsymbol{M}_i \, \frac{\partial \boldsymbol{\nu}_i}{\partial \boldsymbol{q}_d} - m_i \, \boldsymbol{g}^\top \, \frac{\partial \boldsymbol{r}_i}{\partial \boldsymbol{q}_d} \right) \in \mathbb{R}^n, \tag{39}$$

$$\frac{\partial L}{\partial \dot{\boldsymbol{q}}_d} = \sum_i \left(\boldsymbol{\nu}_i^T \boldsymbol{M}_i \, \frac{\partial \boldsymbol{\nu}_i}{\partial \dot{\boldsymbol{q}}_d} \right) \in \mathbb{R}^n. \tag{40}$$

The partial derivatives of (35) with respect to $\boldsymbol{q}_d$ and $\dot{\boldsymbol{q}}_d$ in (39) and (40), respectively, can be analytically obtained

$$\frac{\partial \boldsymbol{\nu}_i}{\partial \boldsymbol{q}_d} = \mathrm{Ad}_{G_i^{\lambda(i)}} \frac{\partial \boldsymbol{\nu}_{\lambda(i)}}{\partial \boldsymbol{q}_d} - \mathrm{ad}_{s_i} \boldsymbol{\nu}_i \, \frac{\partial q_{d_i}}{\partial \boldsymbol{q}_d} \in \mathbb{R}^{6\times n}, \tag{41}$$

$$\frac{\partial \boldsymbol{\nu}_i}{\partial \dot{\boldsymbol{q}}_d} = \mathrm{Ad}_{G_i^{\lambda(i)}} \frac{\partial \boldsymbol{\nu}_{\lambda(i)}}{\partial \dot{\boldsymbol{q}}_d} + \boldsymbol{s}_i \, \frac{\partial \dot{q}_{d_i}}{\partial \dot{\boldsymbol{q}}_d} \in \mathbb{R}^{6\times n}, \tag{42}$$

where

$$\frac{\partial q_{d_i}}{\partial \boldsymbol{q}_d} \in \mathbb{R}^{1\times n}, \quad \frac{\partial \dot{q}_{d_i}}{\partial \dot{\boldsymbol{q}}_d} \in \mathbb{R}^{1\times n}$$

are the *i*-th rows of

$$\frac{\partial \boldsymbol{q}_d}{\partial \boldsymbol{q}_d} \in \mathbb{R}^{n\times n}, \quad \frac{\partial \dot{\boldsymbol{q}}_d}{\partial \dot{\boldsymbol{q}}_d} \in \mathbb{R}^{n\times n},$$

respectively. The operator ad_{s_i} is known as the Lie bracket (Selig, 2004; Park *et al.*, 2018). Also note that the screw $\boldsymbol{s}_i$ is constant, since it is expressed in its local reference frame. In addition, the computation of (39) requires the partial differentiation of (37) with respect to $\boldsymbol{q}_d$.

Write the time derivative of (37) as

$$\begin{bmatrix} \dot{\boldsymbol{r}}_i \\ 0 \end{bmatrix} = \boldsymbol{G}_0^i[\boldsymbol{\nu}_i] \begin{bmatrix} \boldsymbol{r}_{b_i} \\ 1 \end{bmatrix}. \tag{43}$$

Therefore, the time derivative of the CoM stands for

$$\dot{\boldsymbol{r}}_i = \boldsymbol{R}_0^i \left(\boldsymbol{v}_i + [\boldsymbol{\omega}_i]\boldsymbol{r}_{b_i} \right), \tag{44}$$

where $\boldsymbol{R}_0^i \in SO(3)$. Observe that the linear and angular velocities, $\boldsymbol{v}_i$ and $\boldsymbol{\omega}_i$, respectively, are elements of the twist $\boldsymbol{\nu}_i$ defined in (35). Consequently, the partial differentiation of (37) with respect to $\boldsymbol{q}_d$ can be performed by applying (44) and (35) such that

$$\frac{\partial \boldsymbol{r}_i}{\partial \boldsymbol{q}_d} = \boldsymbol{R}_0^i \left(\bar{\boldsymbol{v}}_i - [\boldsymbol{r}_{b_i}]\bar{\boldsymbol{\omega}}_i \right), \tag{45}$$

where $\{\bar{\boldsymbol{v}}_i, \bar{\boldsymbol{\omega}}_i\} \in \mathbb{R}^{3\times n}$ are the linear and angular elements of the following twist-like expression

$$\bar{\boldsymbol{\nu}}_i = \begin{bmatrix} \bar{\boldsymbol{v}}_i \\ \bar{\boldsymbol{\omega}}_i \end{bmatrix} = \mathrm{Ad}_{G_i^{\lambda(i)}} \bar{\boldsymbol{\nu}}_{\lambda(i)} + \boldsymbol{s}_i \frac{\partial q_{d_i}}{\partial \boldsymbol{q}_d} \in \mathbb{R}^{6\times n}. \tag{46}$$

3.3. Sparse computation of the DEL equation. First, let us introduce the use of the colon operator; that refers to indexing and accessing the data of matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ similar to MATLAB language style. Thus, $\boldsymbol{A}_{:,n}$ refers to the *n*-th column of $\boldsymbol{A}$. Also, let us define the partitioned matrix as follows. Given two matrices $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$ of adequate dimensions, they are horizontally or vertically stacked with the partitioned matrix notation as

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{A}_1 & | & \boldsymbol{A}_2 \end{bmatrix} \quad \text{or} \quad \boldsymbol{A} = \begin{bmatrix} \boldsymbol{A}_1 \\ \boldsymbol{A}_2 \end{bmatrix}. \tag{47}$$

Algorithm 1. DEL equation computation.

1: */* Initialization*/*
2: $\frac{\partial L}{\partial q_d} \leftarrow \mathbf{0}^{1\times n} \frac{\partial L}{\partial \dot{q}_d} \leftarrow \mathbf{0}^{1\times n}$
3: **for** $i \leftarrow 1$ to n **do**
4: */* Compute Eqn. (51)*/*
5: $\frac{\partial L}{\partial q_d}[:, 1{:}i] \leftarrow \frac{\partial L}{\partial q_d}[:, 1{:}i]$ $+\nu_i^\top M_i \frac{\partial \nu_i}{\partial q_d}[:, 1{:}i] \; - \; m_i g^\top \frac{\partial r_i}{\partial q_d}[:, 1{:}i]$
6: */* Compute Eqn. (52)*/*
7: $\frac{\partial L}{\partial \dot{q}_d}[:, 1{:}i] \leftarrow \frac{\partial L}{\partial \dot{q}_d}[:, 1{:}i] + \nu_i^\top M_i \frac{\partial \nu_i}{\partial \dot{q}_d}[:, 1{:}i]$
8: **end for**
9: **return** $\frac{\partial L}{\partial q_d}, \frac{\partial L}{\partial \dot{q}_d}$

Now, consider an example to observe the avoidable operations with zero elements when $i = 1$ in (41):

$$\frac{\partial \boldsymbol{\nu}_1}{\partial \boldsymbol{q}_d} = \begin{bmatrix} \frac{\partial \boldsymbol{\nu}_1}{\partial q_{d_1}} & \cancelto{0}{\frac{\partial \boldsymbol{\nu}_1}{\partial q_{d_2}}} & \cancelto{0}{\frac{\partial \boldsymbol{\nu}_1}{\partial q_{d_3}}} & \cancelto{0}{\frac{\partial \boldsymbol{\nu}_1}{\partial q_{d_4}}} \end{bmatrix}. \tag{48}$$

This happens due to the fact that $\boldsymbol{\nu}_i$ only depends on the predecessor joints along the kinematic chain of the robot up to the current i-th body as deduced from (35). To overcome the unnecessary calculations, the chain rule for partial differentiation can be applied as follows:

$$\frac{\partial \boldsymbol{\nu}_i}{\partial \boldsymbol{q}_d} = \frac{\partial \boldsymbol{\nu}_i}{\partial \boldsymbol{q}_{d_{1:i}}} \frac{\partial \boldsymbol{q}_{d_{1:i}}}{\partial \boldsymbol{q}_d}, \tag{49}$$

where $\frac{\partial \boldsymbol{\nu}_i}{\partial \boldsymbol{q}_{d_{1:i}}} \in \mathbb{R}^{6\times i}$ gathers dense terms (i.e. non-zero) while $\frac{\partial \boldsymbol{q}_{d_{1:i}}}{\partial \boldsymbol{q}_d} \in \mathbb{R}^{i\times n}$ contains sparse terms. Then, we recursively collect dense terms as

$$\begin{aligned} \frac{\partial \boldsymbol{\nu}_i}{\partial \boldsymbol{q}_{d_{1:i}}} &= \mathrm{Ad}_{G_i^{\lambda(i)}} \frac{\partial \boldsymbol{\nu}_{\lambda(i)}}{\partial \boldsymbol{q}_{d_{1:i}}} - \mathrm{ad}_{s_i} \boldsymbol{\nu}_i \frac{\partial q_{d_i}}{\partial \boldsymbol{q}_{d_{1:i}}}, \\ &= \begin{bmatrix} \mathrm{Ad}_{G_i^{\lambda(i)}} \frac{\partial \boldsymbol{\nu}_{\lambda(i)}}{\partial \boldsymbol{q}_{d_{1:i-1}}} & \Big| & -\mathrm{ad}_{s_i} \boldsymbol{\nu}_i \end{bmatrix}. \end{aligned} \tag{50}$$

The same procedure applies for (42) and (45) to collect dense terms. As a result, (39) and (40) can be rewritten by means of the partitioned notation where dense and sparse terms are clearly identified:

$$\frac{\partial L}{\partial \boldsymbol{q}_d} = \sum_i \begin{bmatrix} \boldsymbol{\nu}_i^\top \boldsymbol{M}_i \frac{\partial \boldsymbol{\nu}_i}{\partial \boldsymbol{q}_{d_{1:i}}} - m_i \, \boldsymbol{g}^\top \frac{\partial \boldsymbol{r}_i}{\partial \boldsymbol{q}_{d_{1:i}}} & \Big| & \mathbf{0}^{1\times\kappa} \end{bmatrix}, \tag{51}$$

$$\frac{\partial L}{\partial \dot{\boldsymbol{q}}_{\boldsymbol{d}}} = \sum_i \begin{bmatrix} \boldsymbol{\nu}_i^\top \boldsymbol{M}_i \frac{\partial \boldsymbol{\nu}_i}{\partial \dot{\boldsymbol{q}}_{d_{1:i}}} & \Big| & \mathbf{0}^{1\times\kappa} \end{bmatrix}, \tag{52}$$

where $\kappa = n - i$. Algorithm 1 illustrates the procedure for avoiding unnecessary arithmetic operations to compute (51) and (52). In particular, the vector addition at each iteration only operates with dense terms since zero elements are directly assigned in the initialization step.

4. Linearization of the DEL equation

The aim is to analytically proceed with the linearization of the DEL equation. The algorithms suggested by Johnson and Murphey (2009) or Johnson *et al*. (2015) perform scalar operations. By contrast, our multivariable formulation of the discrete Euler–Lagrange equation and its linearization allows us to directly differentiate with respect to the vectors of generalized position and velocity, respectively. In particular, the expressions (39) and (40) should be differentiated with respect to $\boldsymbol{q}_d$ and $\dot{\boldsymbol{q}}_d \in \mathbb{R}^n$. However, their computation implies second partial derivatives where third-order tensors appear. Since tensor operations are computationally demanding, the proposed strategy applies the tensor operations defined in Section 4.1 to isolate dense from sparse elements. Once the sparsity patterns are identified, only dense operations are computed in a recursive manner, as described in Section 4.2. As a result, the arithmetic complexity is reduced. The impact of the proposed factorization is more notorious for tree-like kinematic structures (e.g., humanoid robots) due to the fact that $\boldsymbol{\nu}_i$ and $\boldsymbol{r}_i$ of the i-th body only depend on the predecessor joints along the kinematic chain of the robot.

4.1. Tensor notation. Let $\mathcal{B} \in \mathbb{R}^{6\times m\times p}$ be a third-order tensor and $\boldsymbol{a}$ a given vector belonging to $\mathbb{R}^6$. Then

$$\boldsymbol{C} = \boldsymbol{a} \boxplus \mathcal{B} \in \mathbb{R}^{m\times p}, \tag{53}$$

is the tensor contraction where each element of the resulting matrix is computed as

$$\boldsymbol{C}_{k,l} = \sum_\alpha \boldsymbol{a}_\alpha \, \mathcal{B}_{\alpha,k,l}, \tag{54}$$

where (53) involves p matrix-vector products between 6-by-m slide of $\mathcal{B}$ and the vector $\boldsymbol{a}$. Notice that capitalized calligraphic letters refer to tensor objects. Now, let us denote by $\circ$ the product of a vector $\boldsymbol{b} \in \mathbb{R}^p$ and a matrix $\boldsymbol{A} \in \mathbb{R}^{6\times m}$ such that

$$\mathcal{C} = \boldsymbol{b} \circ \boldsymbol{A} \in \mathbb{R}^{6\times m\times p}, \tag{55}$$

where the scalar-matrix product of each element of $\boldsymbol{b}$ and $\boldsymbol{A}$ is stored as a slice of the resulting tensor $\mathcal{C}$. Therefore, the elements of $\mathcal{C}$ are expressed as

$$\mathcal{C}_{j,k,l} = \boldsymbol{b}_l \, \boldsymbol{A}_{j,k} \, . \tag{56}$$

Similarly, the tensor operation denoted by $\bar{\circ}$ stands for

$$\mathcal{C} = \boldsymbol{A} \, \bar{\circ} \, \boldsymbol{b} \in \mathbb{R}^{6\times p\times m}, \tag{57}$$

where the matrix-scalar product is stored in the resulting tensor $\mathcal{C}$ as

$$\mathcal{C}_{j,k,l} = \boldsymbol{A}_{j,l} \, \boldsymbol{b}_k. \tag{58}$$

The next tensor operation is denoted $\circledast$, which stands for

$$\mathcal{C} = \boldsymbol{A} \circledast \mathcal{B} \in \mathbb{R}^{6\times m\times q}, \tag{59}$$

where $\boldsymbol{A} \in \mathbb{R}^{m\times p}$ and $\mathcal{B} \in \mathbb{R}^{6\times p\times q}$. This tensor operation implies 6 matrix-matrix products between $\boldsymbol{A} \in \mathbb{R}^{m\times p}$ and a slice of $\mathcal{B}$ given by $\boldsymbol{B} \in \mathbb{R}^{p\times q}$. The resulting matrices are stored as slices of tensor $\mathcal{C}$. The scalar form of (59) is

$$\mathcal{C}_{j,k,l} = \sum_{\alpha} \boldsymbol{A}_{k,\alpha}\, \mathcal{B}_{j,\alpha,l}\,. \tag{60}$$

The following tensor operation denoted by $\circledast$ is

$$\mathcal{C} = \mathcal{A} \circledast \boldsymbol{B} \in \mathbb{R}^{6\times p\times m}\,, \tag{61}$$

where $\mathcal{A} \in \mathbb{R}^{6\times p\times q}$ and $\boldsymbol{B} \in \mathbb{R}^{q\times m}$. This tensor operation implies 6 matrix-matrix products between a slice of $\mathcal{A}$ given by $\boldsymbol{A} \in \mathbb{R}^{p\times q}$ and $\boldsymbol{B}$. The resulting matrices are stored as slices of tensor $\mathcal{C}$. The scalar form of (61) is

$$\mathcal{C}_{j,k,l} = \sum_{\alpha} \mathcal{A}_{j,k,\alpha}\, \boldsymbol{B}_{\alpha,l}\,. \tag{62}$$

Let us define a cross-product-like operation as follows:

$$\mathcal{C} = \boldsymbol{A}\widetilde{\otimes}\boldsymbol{B} \in \mathbb{R}^{3\times n\times n}, \tag{63}$$

where $\boldsymbol{A} \in \mathbb{R}^{3\times n}$ and $\boldsymbol{B} \in \mathbb{R}^{3\times n}$. The operator $\widetilde{\otimes}$ transforms the k-th column of matrix $\boldsymbol{A}$ into its skew-symmetric form $[\boldsymbol{A}_{:,k}] \in SO(3)$ to be multiplied by $\boldsymbol{B}$, and the resulting matrix is stored as a slice of tensor $\mathcal{C}$ as

$$\mathcal{C}_{:,:,l} \leftarrow [\boldsymbol{A}_{:,k}]\boldsymbol{B}. \tag{64}$$

4.2. Second order partial derivatives of DEL. The partial derivatives of the system dynamics with respect to $\boldsymbol{q}_d$ and $\dot{\boldsymbol{q}}_d$ imply the evaluation of (29)–(32). Thus, second order partial derivatives arise by differentiating (39) and (40) with respect to $\boldsymbol{q}_d$ and $\dot{\boldsymbol{q}}_d$ as follows:

$$\frac{\partial^2 L}{\partial \boldsymbol{q}_d \partial \boldsymbol{q}_d} = \sum_i \left[\frac{\partial \boldsymbol{\nu}_i^\top}{\partial \boldsymbol{q}_d}\, \boldsymbol{M}_i \frac{\partial \boldsymbol{\nu}_i}{\partial \boldsymbol{q}_d} + \boldsymbol{\nu}_i^\top \boldsymbol{M}_i \boxplus \frac{\partial^2 \boldsymbol{\nu}_i}{\partial \boldsymbol{q}_d \partial \boldsymbol{q}_d} - m_i\, \boldsymbol{g}^\top \boxplus \frac{\partial^2 \boldsymbol{r}_i}{\partial \boldsymbol{q}_d \partial \boldsymbol{q}_d} \right], \tag{65}$$

$$\frac{\partial^2 L}{\partial \boldsymbol{q}_d \partial \dot{\boldsymbol{q}}_d} = \sum_i \left[\frac{\partial \boldsymbol{\nu}_i^\top}{\partial \boldsymbol{q}_d}\, \boldsymbol{M}_i \frac{\partial \boldsymbol{\nu}_i}{\partial \dot{\boldsymbol{q}}_d} + \boldsymbol{\nu}_i^\top \boldsymbol{M}_i \boxplus \frac{\partial^2 \boldsymbol{\nu}_i}{\partial \boldsymbol{q}_d \partial \dot{\boldsymbol{q}}_d} \right], \tag{66}$$

$$\frac{\partial^2 L}{\partial \dot{\boldsymbol{q}}_d \partial \dot{\boldsymbol{q}}_d} = \sum_i \left[\frac{\partial \boldsymbol{\nu}_i^\top}{\partial \dot{\boldsymbol{q}}_d}\, \boldsymbol{M}_i \frac{\partial \boldsymbol{\nu}_i}{\partial \dot{\boldsymbol{q}}_d} \right], \tag{67}$$

$$\frac{\partial^2 L}{\partial \dot{\boldsymbol{q}}_d \partial \boldsymbol{q}_d} = \frac{\partial^2 L}{\partial \boldsymbol{q}_d \partial \dot{\boldsymbol{q}}_d}^\top, \tag{68}$$

where $\frac{\partial^2 \boldsymbol{\nu}_i}{\partial \boldsymbol{q}_d \partial \boldsymbol{q}_d}$ and $\frac{\partial^2 \boldsymbol{\nu}_i}{\partial \boldsymbol{q}_d \partial \dot{\boldsymbol{q}}_d}$ are third-order tensors belonging to $\mathbb{R}^{6\times n\times n}$ while $\frac{\partial^2 \boldsymbol{r}_i}{\partial \boldsymbol{q}_d \partial \boldsymbol{q}_d}$ belongs to $\mathbb{R}^{3\times n\times n}$. The operator $\boxplus$ refers to tensor contraction (53).

According to the Schwarz symmetry theorem for matrices composed by second order partial derivatives (see the work of Carlier (2022, pp. 65–69) for more details), expression (68) can be easily evaluated by transposing (66). Note that expressions (41) and (42) can be used to recursively compute the partial derivatives as

$$\begin{aligned}\frac{\partial^2 \boldsymbol{\nu}_i}{\partial \boldsymbol{q}_d \partial \boldsymbol{q}_d} &= \mathrm{Ad}_{G_i^{\lambda(i)}} \circledast \frac{\partial^2 \boldsymbol{\nu}_{\lambda(i)}}{\partial \boldsymbol{q}_d \partial \boldsymbol{q}_d} \\ &\quad - \mathrm{ad}_{s_i} \circledast \left(\frac{\partial q_{d_i}}{\partial \boldsymbol{q}_d} \circ \mathrm{Ad}_{G_i^{\lambda(i)}} \frac{\partial \boldsymbol{\nu}_{\lambda(i)}}{\partial \boldsymbol{q}_d} + \frac{\partial \boldsymbol{\nu}_i}{\partial \boldsymbol{q}_d} \bar{\circ} \frac{\partial q_{d_i}}{\partial \boldsymbol{q}_d} \right),\end{aligned} \tag{69}$$

$$\begin{aligned}\frac{\partial^2 \boldsymbol{\nu}_i}{\partial \boldsymbol{q}_d \partial \dot{\boldsymbol{q}}_d} &= \mathrm{Ad}_{G_i^{\lambda(i)}} \circledast \frac{\partial^2 \boldsymbol{\nu}_{\lambda(i)}}{\partial \boldsymbol{q}_d \partial \dot{\boldsymbol{q}}_d} \\ &\quad - \mathrm{ad}_{s_i} \circledast \left(\frac{\partial q_{d_i}}{\partial \boldsymbol{q}_d} \circ \mathrm{Ad}_{G_i^{\lambda(i)}} \frac{\partial \boldsymbol{\nu}_{\lambda(i)}}{\partial \dot{\boldsymbol{q}}_d} \right),\end{aligned} \tag{70}$$

where the operators $\circ$ and $\bar{\circ}$ in (69)–(70) apply (55) and (57), respectively. Note that the second partial derivative of the CoM with respect to $\boldsymbol{q}_d$ can be obtained by partially differentiating (45) with respect to $\boldsymbol{q}_d$ as

$$\begin{aligned}\frac{\partial^2 \boldsymbol{r}_i}{\partial \boldsymbol{q}_d \partial \boldsymbol{q}_d} &= \boldsymbol{R}_0^i \circledast \Big([\bar{\boldsymbol{\omega}}_i] \widetilde{\otimes} (\bar{\boldsymbol{v}}_i - [\boldsymbol{r}_{b_i}]\bar{\boldsymbol{\omega}}_i) \\ &\quad + \left(\frac{\partial \bar{\boldsymbol{v}}_i}{\partial \boldsymbol{q}_d} - [\boldsymbol{r}_{b_i}] \circledast \frac{\partial \bar{\boldsymbol{\omega}}_i}{\partial \boldsymbol{q}_d} \right) \Big),\end{aligned} \tag{71}$$

where $\partial \bar{\boldsymbol{v}}_i / \partial \boldsymbol{q}_d \in \mathbb{R}^{3\times n\times n}$ and $\partial \bar{\boldsymbol{\omega}}_i / \partial \boldsymbol{q}_d \in \mathbb{R}^{3\times n\times n}$ are the linear and angular elements of a tensor obtained by partially differentiating the recursive twist-like expression given in (46) with respect to $\boldsymbol{q}_d$ as

$$\begin{aligned}\frac{\partial \bar{\boldsymbol{\nu}}_i}{\partial \boldsymbol{q}_d} &= \begin{bmatrix} \frac{\partial \bar{\boldsymbol{v}}_i}{\partial \boldsymbol{q}_d} \\ \frac{\partial \bar{\boldsymbol{\omega}}_i}{\partial \boldsymbol{q}_d} \end{bmatrix} \\ &= \mathrm{Ad}_{G_i^{\lambda(i)}} \circledast \frac{\partial \bar{\boldsymbol{\nu}}_{\lambda(i)}}{\partial \boldsymbol{q}_d} - \frac{\partial q_{d_i}}{\partial \boldsymbol{q}_d} \circ \mathrm{ad}_{s_i}\, \bar{\boldsymbol{\nu}}_i.\end{aligned} \tag{72}$$

It is important to point out that the direct computation of (65)–(67) does not consider the induced sparsity of the mechanical system. Consequently, it does not avoid unnecessary arithmetic operations. We cope with this problem by factorizing dense and sparse terms. The factorization makes use of the tensor operations described in Section 4.1. Let us sketch the proposed factorization to evaluate (69). From the chain rule, it is possible to rewrite (69) as

$$\frac{\partial^2 \boldsymbol{\nu}_i}{\partial \boldsymbol{q}_d \partial \boldsymbol{q}_d} = \frac{\partial \boldsymbol{q}_{d_{1:i}}}{\partial \boldsymbol{q}_d}^\top \circledast \frac{\partial^2 \boldsymbol{\nu}_i}{\partial \boldsymbol{q}_{d_{1:i}} \partial \boldsymbol{q}_{d_{1:i}}} \circledast \frac{\partial \boldsymbol{q}_{d_{1:i}}}{\partial \boldsymbol{q}_d}, \tag{73}$$

where

$$\frac{\partial^2 \boldsymbol{\nu}_i}{\partial \boldsymbol{q}_{d_{1:i}} \partial \boldsymbol{q}_{d_{1:i}}} \in \mathbb{R}^{6\times i\times i}$$

is the dense part of the tensor while

$$\frac{\partial \boldsymbol{q}_{d_{1:i}}}{\partial \boldsymbol{q}_d} \in \mathbb{R}^{i\times n}$$

contains sparse terms. The same procedure can be applied to (70)-(72) for grouping dense terms. Then, simple algebraic manipulation is performed to factorize expression (65) for obtaining the following sparsity-free computation:

$$\begin{aligned}\frac{\partial^2 L}{\partial \boldsymbol{q}_d \partial \boldsymbol{q}_d} = \sum_i \frac{\partial \boldsymbol{q}_{d_{1:i}}}{\partial \boldsymbol{q}_d}^\top \Bigg(& \frac{\partial \boldsymbol{\nu}_i^\top}{\partial \boldsymbol{q}_{d_{1:i}}} \boldsymbol{M}_i \frac{\partial \boldsymbol{\nu}_i}{\partial \boldsymbol{q}_{d_{1:i}}} \\ & + \boldsymbol{\nu}_i^\top \boldsymbol{M}_i \boxplus \frac{\partial^2 \boldsymbol{\nu}_i}{\partial \boldsymbol{q}_{d_{1:i}} \partial \boldsymbol{q}_{d_{1:i}}} \\ & - m_i \, \boldsymbol{g}^\top \boxplus \frac{\partial^2 \boldsymbol{r}_i}{\partial \boldsymbol{q}_{d_{1:i}} \partial \boldsymbol{q}_{d_{1:i}}} \Bigg) \frac{\partial \boldsymbol{q}_{d_{1:i}}}{\partial \boldsymbol{q}_d}.\end{aligned} \tag{74}$$

The Hessian matrix (74) encodes the sparsity-free computation of two tensors within an iterative process. Consequently, the outer sparse term $\frac{\partial \boldsymbol{q}_{d_{1:i}}}{\partial \boldsymbol{q}_d}$ in (74) does not have to be computed. The same factorization strategy is applied to rewrite expressions (66) and (67). Algorithm 2 illustrates the computation of the linearization of DEL equations. The factorization used in (73) isolates dense and sparse components of a third-order tensor that appears in (65). Thus, the sparsity due to successors of the i-th body along the kinematic chain is eliminated. From lines 6 to 8 of Algorithm 2 the dense term $\frac{\partial^2 \nu_i}{\partial q_{d_{1:i}} \partial q_{d_{1:i}}}$ is computed in a recursive manner. It can be observed that these lines share the structure of expression (69). The only difference is on the elimination of the inner sparsity observed in (69) due to vector $\frac{\partial q_{d_i}}{\partial q_d}$ that contains one and zeros. Similarly, from lines 18 to 22 of Algorithm 2, the same strategy for removing sparse terms has been applied to compute (65)–(68).

5. Results

This section focuses on validating the performance and numerical accuracy of the proposed sparsity-free DMOC strategy with analytical evaluation of the forced DEL equation and its linearization. First, we quantify the arithmetic complexity to compute the linearization of the DEL equation. Then, we evaluate the accuracy of optimized trajectories in terms of the discretization error. In particular, we compare a classic direct collocation method provided in CasADi (Andersson *et al.*, 2019) against the proposed sparsity-free DMOC strategy. Also, we illustrate the use of DMOC to generate dynamically feasible humanoid motions. Finally, we show numerical comparisons to evaluate the forced DEL equation and its linearization between the methods introduced by Johnson and Murphey (2009) as well as Johnson *et al.* (2015) versus the proposed strategy.

5.1. Arithmetic complexity. The arithmetic complexity has been considered to obtain a quantitative comparison of evaluating the forced DEL and its linearization with and without sparsity exploitation. Table 5.1 shows the number of arithmetic operations in both cases together with the corresponding average reduction. The quantities in Table 5.1 have been obtained by considering 50 serial robots with $n = \{1, 2, \ldots, 50\}$ degrees of freedom (DoF). Figure 2 presents the number of multiplications and additions as a function of the number of DoF for both cases. Table 5.1 provides the number of arithmetic operations to evaluate individual second-order terms as a function of the robot's DoF.

Table 1. Arithmetic complexity: total number of cumulative scalar multiplications and additions to evaluate the forced DEL and its linearization versus the proposed sparsity-free strategy for a set of 50 serial robots.

	Multiplications	Additions
Forced DEL	345917700	305599650
Exploiting sparsity	121569975	107366475
Average reduction	**64.8**[%]	**64.8**[%]

Table 2. Arithmetic operations to compute sparsity-free second-order terms for $1 \leq i \leq n$, where n is the number of DoF. The reduction is 64%.

	Exploiting sparsity	
Second-order terms	Multiplications	Additions
$\frac{\partial^2 \nu_i}{\partial q_{d_{1:i}} \partial q_{d_{1:i}}}$	$84i^2 + 36i$	$72i^2 + 30i$
$\frac{\partial^2 \nu_i}{\partial q_{d_{1:i}} \partial \dot{q}_{d_{1:i}}}$	$78i^2 + 36i$	$66i^2 + 30i$
$\frac{\partial^2 L_i}{\partial q_d \partial q_d}$	$15i^2 + 36i$	$8i^2 + 37i + 30$
$\frac{\partial^2 L_i}{\partial q_d \partial \dot{q}_d}$ & $\frac{\partial^2 L_i}{\partial \dot{q}_d \partial q_d}$	$12i^2 + 36i + 36$	$7i^2 + 35i + 30$
$\frac{\partial^2 L_i}{\partial \dot{q}_d \partial \dot{q}_d}$	$6i^2 + 36i$	$5i^2 + 30i$

5.2. DMOC accuracy against trapezoidal collocation. The accuracy of optimized robot trajectories depends on several factors such as the number of discrete points N, the size of time increment Δt, and the number of decision variables $\boldsymbol{z}$, among others. Here, the accuracy is calculated by measuring the violation of system dynamics along optimized robot trajectories. In other words, the error is estimated by assessing how well the optimized trajectories satisfy the system dynamics between discrete points.

5.2.1. Evaluating DMOC accuracy. Transcription methods commonly approximate the state $\boldsymbol{x}(t)$ with cubic

Algorithm 2. Differentiation of DEL equations.

/* *Initialization*/*
$\frac{\partial^2 L}{\partial q_d \partial q_d} \leftarrow \mathbf{0}^{n\times n}, \quad \frac{\partial^2 L}{\partial \dot{q}_d \partial q_d} \leftarrow \mathbf{0}^{n\times n}, \quad \frac{\partial^2 L}{\partial q_d \partial \dot{q}_d} \leftarrow \mathbf{0}^{n\times n}, \quad \frac{\partial^2 L}{\partial \dot{q}_d \partial \dot{q}_d} \leftarrow \mathbf{0}^{n\times n}$
for $i \leftarrow 1$ to n **do**
  $\frac{\partial^2 \nu_i}{\partial q_d \partial q_d} \leftarrow \mathbf{0}^{6\times n\times n}, \quad \frac{\partial^2 \nu_i}{\partial q_d \partial \dot{q}_d} \leftarrow \mathbf{0}^{6\times n\times n}, \quad \frac{\partial \bar{\nu}_i}{\partial q_d} \leftarrow \mathbf{0}^{6\times n\times n}, \quad \frac{\partial^2 r_i}{\partial q_d \partial q_d} \leftarrow \mathbf{0}^{3\times n\times n}$
  /* *Compute Eqn. (69)* */
  $\frac{\partial^2 \nu_i}{\partial q_d \partial q_d}[:, 1{:}i{-}1, 1{:}i{-}1] \leftarrow \mathrm{Ad}_{G_i^{\lambda(i)}} \circledast \frac{\partial^2 \nu_{\lambda(i)}}{\partial q_d \partial q_d}[:, 1{:}i{-}1, 1{:}i{-}1]$
  $\frac{\partial^2 \nu_i}{\partial q_d \partial q_d}[:, 1{:}i{-}1, i] \leftarrow -\mathrm{ad}_{s_i} \circledast \mathrm{Ad}_{G_i^{\lambda(i)}} \frac{\partial \nu_{\lambda(i)}}{\partial q_d}[:, 1{:}i{-}1]$
  $\frac{\partial^2 \nu_i}{\partial q_d \partial q_d}[:, i, :] \leftarrow -\mathrm{ad}_{s_i} \frac{\partial \nu_i}{\partial q_d}[:, 1{:}i]$
  /* *Compute Eqn. (70)* */
  $\frac{\partial^2 \nu_i}{\partial q_d \partial \dot{q}_d}[:, 1{:}i{-}1, 1{:}i{-}1] \leftarrow \mathrm{Ad}_{G_i^{\lambda(i)}} \circledast \frac{\partial^2 \nu_{\lambda(i)}}{\partial q_d \partial \dot{q}_d}[:, 1{:}i{-}1, 1{:}i{-}1]$
  $\frac{\partial^2 \nu_i}{\partial q_d \partial \dot{q}_d}[:, 1{:}i{-}1, i] \leftarrow -\mathrm{ad}_{s_i} \circledast \mathrm{Ad}_{G_i^{\lambda(i)}} \frac{\partial \nu_{\lambda(i)}}{\partial \dot{q}_d}[:, 1{:}i{-}1]$
  /* *Compute Eqn. (72)* */
  $\frac{\partial \bar{\nu}_i}{\partial q_d}[:, 1{:}i{-}1, 1{:}i{-}1] \leftarrow \mathrm{Ad}_{G_i^{\lambda(i)}} \circledast \frac{\partial \bar{\nu}_{\lambda(i)}}{\partial q_d}[:, 1{:}i{-}1, 1{:}i{-}1]$
  $\frac{\partial \bar{\nu}_i}{\partial q_d}[:, :, i] \leftarrow \mathrm{ad}_{s_i} \bar{\nu}_i[:, 1{:}i]$
  /* *Compute Eqn. (71)* */
  $\frac{\partial^2 r_i}{\partial q_d \partial q_d}[:, 1{:}i, 1{:}i] \leftarrow R_0^i \circledast \Big([\bar{\omega}_i[:, 1{:}i]] \widetilde{\otimes} (\bar{v}_i[:, 1{:}i] - [r_{b_i}] \bar{\omega}_i[:, 1{:}i]) + (\frac{\partial \bar{\nu}_i}{\partial q_d}[1{:}3, 1{:}i, 1{:}i] - [r_{b_i}] \circledast \frac{\partial \bar{\nu}_i}{\partial q_d}[4{:}6, 1{:}i, 1{:}i]) \Big)$
  /* *Compute Eqn. (65), (66) and (67)* */
  $\frac{\partial^2 L}{\partial q_d \partial q_d}[1{:}i, 1{:}i] \leftarrow \frac{\partial^2 L}{\partial q_d \partial q_d}[1{:}i, 1{:}i] + \frac{\partial \nu_i^\top}{\partial q_d}[:, 1{:}i]\, M_i\, \frac{\partial \nu_i}{\partial q_d}[:, 1{:}i] + \nu_i^\top M_i \boxplus \frac{\partial^2 \nu_i}{\partial q_d \partial q_d}[:, 1{:}i, 1{:}i] - m_i\, g^\top \boxplus \frac{\partial^2 r_i}{\partial q_d \partial q_d}[:, 1{:}i, 1{:}i]$
  $\frac{\partial^2 L}{\partial q_d \partial \dot{q}_d}[1{:}i, 1{:}i] \leftarrow \frac{\partial^2 L}{\partial q_d \partial \dot{q}_d}[1{:}i, 1{:}i] + \frac{\partial \nu_i^\top}{\partial q_d}[:, 1{:}i]\, M_i\, \frac{\partial \nu_i}{\partial \dot{q}_d}[:, 1{:}i] + \nu_i^\top M_i \boxplus \frac{\partial^2 \nu_i}{\partial q_d \partial \dot{q}_d}[:, 1{:}i, 1{:}i]$
  $\frac{\partial^2 L}{\partial \dot{q}_d \partial \dot{q}_d}[1{:}i, 1{:}i] \leftarrow \frac{\partial^2 L}{\partial \dot{q}_d \partial \dot{q}_d}[1{:}i, 1{:}i] + \frac{\partial \nu_i^\top}{\partial \dot{q}_d}[:, 1{:}i]\, M_i\, \frac{\partial \nu_i}{\partial \dot{q}_d}[:, 1{:}i]$
end for
/* *Compute Eqn. (68)* */
$\frac{\partial^2 L}{\partial \dot{q}_d \partial q_d} \leftarrow \frac{\partial^2 L}{\partial q_d \partial \dot{q}_d}^\top$
return $\frac{\partial^2 L}{\partial q_d \partial q_d}, \frac{\partial^2 L}{\partial \dot{q}_d \partial q_d}, \frac{\partial^2 L}{\partial q_d \partial \dot{q}_d}, \frac{\partial^2 L}{\partial \dot{q}_d \partial \dot{q}_d}$

B-splines while the control $\boldsymbol{u}(t)$ is approximated with linear splines (Betts, 2010). The error can be defined as

$$\eta_k = \int_{t_k}^{t_{k+1}} |\boldsymbol{\varepsilon}(s)| \mathrm{d}s, \tag{75}$$

with

$$\boldsymbol{\varepsilon}(t) = \bar{\dot{\boldsymbol{x}}}(t) - \boldsymbol{f}_x(\bar{\boldsymbol{x}}(t), \bar{\boldsymbol{u}}(t)), \tag{76}$$

as the local error at step k, and the bar refers to time-varying profiles evaluated with the corresponding interpolation. In addition, $\bar{\dot{\boldsymbol{x}}}(t)$ is the time-derivative of the interpolated state $\bar{\boldsymbol{x}}(t)$. Since DMOC solves for robot configuration $\boldsymbol{q}(t)$, momentum $\boldsymbol{p}(t)$ and control input $\boldsymbol{u}(t)$, it is not possible to directly utilize (76). Thus, the standard Legendre transform (19) is used to get $\dot{\boldsymbol{q}}(t)$ from $\boldsymbol{p}(t)$ as

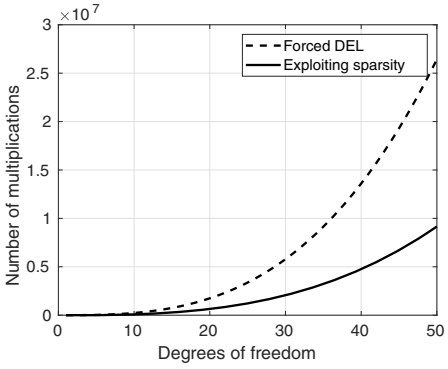
$$\dot{\boldsymbol{q}}(t) = \boldsymbol{H}(\boldsymbol{q}(t))^{-1} \boldsymbol{p}(t), \tag{77}$$

where $\boldsymbol{H}(\boldsymbol{q}(t)) \in \mathbb{R}^{n\times n}$ is the generalized inertia matrix.

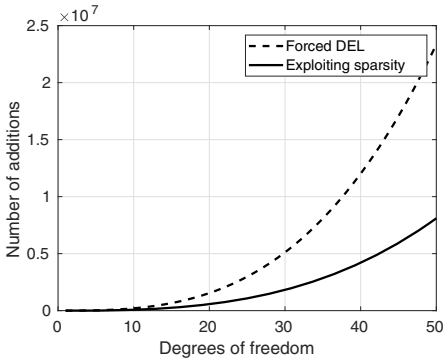
5.2.2. DMOC versus trapezoidal collocation. A relatively simple optimal control problem has been considered to compare the convergence efficiency of DMOC (with the simplest variational integrator known as the midpoint rule) versus trapezoidal collocation, which is a classic direct collocation method available in CasADi. It is also important to mention that CasADi computes analytical derivatives by means of automatic differentiation techniques, while the proposed DMOC strategy computes analytical derivatives with sparsity-free recursive algorithms. In both cases, the optimization solver IPOPT is used (Biegler and Zavala, 2009), which is the one that requests the evaluation of the system dynamics and its linearization at any time. The system dynamics corresponds to the equations of motion of a serial robot with 3 DoF. The final time was 5 seconds, and the cost function stands for the L_2-norm of control inputs as

$$J_d(\boldsymbol{u}_k) = \sum_{k=0}^{N-1} ||\boldsymbol{u}_k||^2. \tag{78}$$

Table 5.2.2 reports the number of iterations, the optimal cost function, and the error accuracy while increasing the number of discrete points. It is clear that the proposed DMOC strategy arrives at a better local



(a)



(b)

Fig. 2. Scalar operations between the forced DEL and its linearization versus the proposed sparsity-free strategy.

minimum in fewer iterations and with higher accuracy than the trapezoidal collocation.

5.3. DMOC-based humanoid motion generation. The proposed DMOC solver has been executed on a laptop with standard computation capabilities, Intel i7 at 2.60 GHz. The optimized trajectories are directly sent to the NAO robot (a tree-like kinematic structure with 24 DoF) in the simulation environment CoppeliaSim for validating that the robot is balanced (Rohmer *et al*., 2013).

Since NAO is a biped robot, its equilibrium is handled by means of its centroidal momentum, which consists of summing the momenta of all robot bodies, and then the resulting momentum is projected onto a reference frame attached to the robot's center of mass (Orin *et al*., 2013). According to the midpoint approximation, the centroidal momentum is evaluated as

$$\boldsymbol{\mu}_k(\boldsymbol{q}_k, \boldsymbol{q}_{k+1}) = \boldsymbol{\mu}\left(\frac{\boldsymbol{q}_k + \boldsymbol{q}_{k+1}}{2}, \frac{\boldsymbol{q}_{k+1} - \boldsymbol{q}_k}{\Delta t}\right)\Delta t. \tag{79}$$

Thus, the cost function (4) has been chosen as

$$J_d(\boldsymbol{q}_k, \boldsymbol{q}_{k+1}, u_k) = \sum_{k=0}^{N-1} ||\boldsymbol{\mu}_k||^2 + ||\boldsymbol{q}_{k+1} - \boldsymbol{q}_k||^2. \tag{80}$$

Figure 3 shows a motion sequence for the humanoid robot NAO reaching a desired posture while maintaining its balance through one foot contact.

Table 4 shows the computational performance of DMOC to get optimized trajectories for the NAO robot with different resolutions (i.e., $N = \{20, 30, 40, 50\}$). For comparison purposes, the same optimization problems have been solved by computing the forced DEL equation and its linearization with the methods introduced by Johnson and Murphey (2009) as well as Johnson *et al*. (2015). Moreover, Table 5 shows the computational time to evaluate the constraint Jacobian in DMOC with finite differences, the methods introduced by Johnson and Murphey (2009) as well as Johnson *et al*. (2015), and the proposed sparsity-free strategy. It is observed how much time is saved with our methods for different sizes of the problem.

To evaluate the accuracy of the proposed DMOC strategy, three different trajectory optimization problems were solved according to three resolutions $N = \{30, 60, 100\}$. For each resulting trajectory, the sum of errors between discrete points (75) was computed. The average error is shown in Table 6. As is expected, it decreases when the number of discrete points increases due to the mesh refinement. Finally, it is important to mention that, for the specific case of NAO robot, the reduction of arithmetic operation was 65%.

6. Conclusions

We have provided efficient algorithms for analytically evaluating the forced DEL equation and its linearization, which are requested to solve DMOC for robot trajectory optimization problems. The proposed derivation enables the possibility to exploit the sparsity of the constraint Jacobian associated with DMOC, which turns to be important when highly articulated rigid-body systems are considered. In particular, geometric and multilinear operators allowed us to factorize dense and sparse terms in a recursive manner. The numerical validation involved the arithmetic complexity and the accuracy of the optimized trajectories in terms of mesh refinement and the computational performance.

Future research is targeted at analytically computing higher-order terms of the DEL equation in order to evaluate sufficient conditions for optimality. Also, it is desirable to implement the proposed strategy on real robotic platforms. Thus, model uncertainties and disturbances should be considered as suggested by Aguilar-Ibanez *et al*. (2024).

Acknowledgment

Carla Villanueva-Piñon acknowledges the financial support of the National Council of Humanities, Science and Technology (CONACHyT) under the scholarship no. 779478.

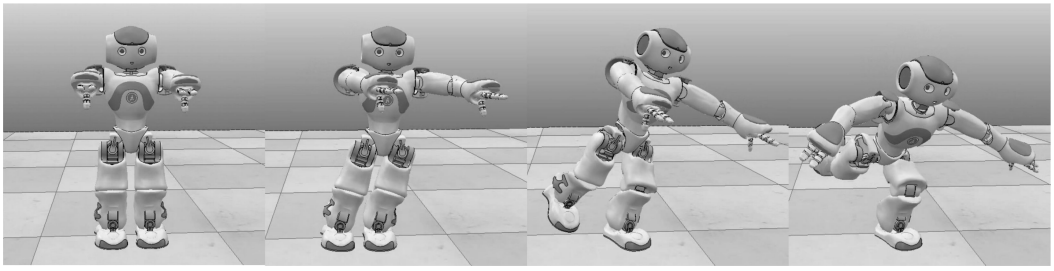


Fig. 3. DMOC-based optimized trajectory for the humanoid robot NAO reaching a desired posture while maintaining its balance through one-foot contact.

Table 3. Optimization report for a 3 DoF serial robot.

	Trapezoidal / CasADi			DMOC / Exploiting sparsity		
N	Iters	Cost	Error	Iters	Cost	Error
20	18	1.20956	5.3091e-3	10	1.16425	3.8177e-4
50	18	1.17615	2.0953e-3	10	1.16425	2.3348e-5
100	17	1.17160	1.0375e-3	12	1.16663	2.8470e-6
150	17	1.17077	6.8972e-4	13	1.16767	8.3619e-7

Table 4. Computational time of DMOC to get optimized trajectories for a NAO robot with different resolutions.

Computational time in seconds				
Method	$N = 20$	$N = 30$	$N = 40$	$N = 50$
Johnson *et al.*, 2015	290.21	655.44	907.19	1370.32
Exploiting sparsity	39.42	77.98	122.77	188.88

Table 5. Computational time to evaluate the constraint Jacobian in DMOC.

Jacobian evaluation in seconds					
			Methods		Exploiting
N	n_z	n_{cns}	Numerical	Johnson *et al.*, 2009; 2015	sparsity
30	1440	768	17.92	1.13	0.10
60	2880	1488	70.46	2.27	0.24
100	4800	2448	191.42	3.92	0.46

Table 6. Accuracy evaluation of DMOC.

	Number of discrete points		
Trajectory	$N = 30$	$N = 60$	$N = 100$
1	1.136	1.814e-1	6.15e-2
2	1.479	1.94e-1	4.62e-2
3	1.547	2.186e-1	5.77e-2

References

Agamawi, Y.M. and Rao, A.V. (2020). CGPOPS: A C++ software for solving multiple-phase optimal control problems using adaptive Gaussian quadrature collocation and sparse nonlinear programming, *ACM Transactions on Mathematical Software* **46**(3): 1–38.

Aguilar-Ibanez, C., Suarez-Castanon, M.S., Saldivar, B., Jimenez-Lizarraga, M.A., de Jesus Rubio, J. and Mendoza-Mendoza, J. (2024). Algebraic active disturbance rejection to control a generalized uncertain second-order flat system, *International Journal of Applied Mathematics and Computer Science* **34**(2): 185–198, DOI: 10.61822/amcs-2024-0013.

Andersson, J.A., Gillis, J., Horn, G., Rawlings, J.B. and Diehl, M. (2019). CasADi: A software framework for nonlinear optimization and optimal control, *Mathematical Programming Computation* **11**(1): 1–36.

Becerra, V.M. (2010). Solving complex optimal control problems at no cost with PSOPT, *2010 IEEE International Symposium on Computer-Aided Control System Design, Yokohama, Japan*, pp. 1391–1396.

Betts, J.T. (2010). *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming*, SIAM, Philadelphia.

Betts, J.T. and Erb, S.O. (2003). Optimal low thrust trajectories to the moon, *SIAM Journal on Applied Dynamical Systems* **2**(2): 144–170.

Biegler, L.T. and Zavala, V.M. (2009). Large-scale nonlinear programming using IPOPT: An integrating framework

for enterprise-wide dynamic optimization, *Computers & Chemical Engineering* **33**(3): 575–582.

Brockett, R.W. (2005). Robotic manipulators and the product of exponentials formula, *in* P.A. Fuhrmanni (Ed.), *Mathematical Theory of Networks and Systems*, Springer, Berlin, pp. 120–129.

Budhiraja, R., Carpentier, J., Mastalli, C. and Mansard, N. (2018). Differential dynamic programming for multi-phase rigid contact dynamics, *2018 IEEE–RAS 18th International Conference on Humanoid Robots (Humanoids), Beijing, China*, pp. 1–9.

Cardona-Ortiz, D., Paz, A. and Arechavaleta, G. (2020). Exploiting sparsity in robot trajectory optimization with direct collocation and geometric algorithms, *2020 IEEE International Conference on Robotics and Automation (ICRA), Paris, France*, pp. 469–475.

Carlier, G. (2022). *Classical and Modern Optimization*, World Scientific, London.

Fan, T., Schultz, J. and Murphey, T. (2018). Efficient computation of higher-order variational integrators in robotic simulation and trajectory optimization, *in* M. Morales *et al.* (Eds), *Algorithmic Foundations of Robotics*, Springer, Cham, pp. 689–706.

Featherstone, R. (2014). *Rigid Body Dynamics Algorithms*, Springer, New York.

Hereid, A. and Ames, A.D. (2017). FROST*: Fast robot optimization and simulation toolkit, *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, Canada*, pp. 719–726.

Houska, B., Ferreau, H.J. and Diehl, M. (2011). ACADO toolkit—An open-source framework for automatic control and dynamic optimization, *Optimal Control Applications and Methods* **32**(3): 298–312.

Howell, T.A., Jackson, B.E. and Manchester, Z. (2019). ALTRO: A fast solver for constrained trajectory optimization, *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Macau, China*, pp. 7674–7679.

Johnson, E.R. and Murphey, T.D. (2009). Scalable variational integrators for constrained mechanical systems in generalized coordinates, *IEEE Transactions on Robotics* **25**(6): 1249–1261.

Johnson, E., Schultz, J. and Murphey, T. (2015). Structured linearization of discrete mechanical systems for analysis and optimal control, *IEEE Transactions on Automation Science and Engineering* **12**(1): 140–152.

Kelly, M. (2017). An introduction to trajectory optimization: How to do your own direct collocation, *SIAM Review* **59**(4): 849–904.

Kelly, M.P. (2019). DirCol5i: Trajectory optimization for problems with high-order derivatives, *Journal of Dynamic Systems, Measurement, and Control* **141**(3).

Kobilarov, M.B. and Marsden, J.E. (2011). Discrete geometric optimal control on Lie groups, *IEEE Transactions on Robotics* **27**(4): 641–655.

Kobilarov, M. and Sukhatme, G. (2007). Optimal control using nonholonomic integrators, *Proceedings 2007 IEEE International Conference on Robotics and Automation, Rome, Italy*, pp. 1832–1837.

Lee, J., Liu, C.K., Park, F.C. and Srinivasa, S.S. (2020). A linear-time variational integrator for multibody systems, *in* K. Goldberg *et al.* (Eds), *Algorithmic Foundations of Robotics XII*, Springer, Cham, pp. 352–367.

Leineweber, D. B., Schäfer, A., Bock, H.G. and Schlöder, J. P. (2003). An efficient multiple shooting based reduced SQP strategy for large-scale dynamic process optimization. Part II: Software aspects and applications, *Computers & Chemical Engineering* **27**(2): 167–174.

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

Liu, G., Wu, S., Zhu, L., Wang, J. and Lv, Q. (2022). Fast and smooth trajectory planning for a class of linear systems based on parameter and constraint reduction, *International Journal of Applied Mathematics and Computer Science* **32**(1): 11–21, DOI: 10.34768/amcs-2022-0002.

Manchester, Z., Doshi, N., Wood, R.J. and Kuindersma, S. (2019). Contact-implicit trajectory optimization using variational integrators, *International Journal of Robotics Research* **38**(12-13): 1463–1476.

Manns, P. and Mombaur, K. (2017). Towards discrete mechanics and optimal control for complex models, *IFAC-PapersOnLine* **50**(1): 4812–4818.

Marsden, J.E. and West, M. (2001). Discrete mechanics and variational integrators, *Acta Numerica* **10**(10): 357–514.

Mastalli, C., Budhiraja, R., Merkt, W., Saurel, G., Hammoud, B., Naveau, M., Carpentier, J., Righetti, L., Vijayakumar, S. and Mansard, N. (2020). Crocoddyl: An efficient and versatile framework for multi-contact optimal control, *2020 IEEE International Conference on Robotics and Automation (ICRA), Paris, France*, pp. 2536–2542.

Mayne, D. (1966). A second-order gradient method for determining optimal trajectories of non-linear discrete-time systems, *International Journal of Control* **3**(1): 85–95.

Ober-Blöbaum, S., Junge, O. and Marsden, J.E. (2011). Discrete mechanics and optimal control: An analysis, *ESAIM: Control, Optimisation and Calculus of Variations* **17**(2): 322–352.

Orin, D.E., Goswami, A. and Lee, S.-H. (2013). Centroidal dynamics of a humanoid robot, *Autonomous Robots* **35**(2-3): 161–176.

Park, F.C., Kim, B., Jang, C. and Hong, J. (2018). Geometric algorithms for robot dynamics: A tutorial review, *Applied Mechanics Reviews* **70**(1): 010803.

Rao, A.V. (2009). A survey of numerical methods for optimal control, *Advances in the Astronautical Sciences* **135**(1): 497–528.

Rohmer, E., Singh, S.P.N. and Freese, M. (2013). V-REP: A versatile and scalable robot simulation framework, *International Conference on Intelligent Robots and Systems (IROS), Tokyo, Japan*, pp. 1321–1326.

Selig, J.M. (2004). *Geometric Fundamentals of Robotics*, Springer, New York.

Sun, Z., Tian, Y., Li, H. and Wang, J. (2016). A superlinear convergence feasible sequential quadratic programming algorithm for bipedal dynamic walking robot via discrete mechanics and optimal control, *Optimal Control Applications and Methods* **37**(6): 1139–1161.

Zhang, W., Wang, D. and Inanc, T. (2018). A multiphase DMOC-based trajectory optimization method, *Optimal Control Applications and Methods* **39**(1): 114–129.

Carla Villanueva-Piñon obtained her bachelor's degree in mechatronics engineering in 2015 from Universidad Politécnica de Victoria. She received her MSc degree in robotics and advanced manufacturing from Centro de Investigación y de Estudios Avanzados del IPN (CINVESTAV), Saltillo, Mexico, in 2019. She is currently a PhD student in robotics and advanced manufacturing at CINVESTAV. Her research interests include robot dynamics, robot trajectory optimization, humanoid whole-body motion generation.

Gustavo Arechavaleta received his MS degree from Tecnológico de Monterrey (ITESM), Campus Estado de México, in 2003, and his PhD degree from Institut National des Sciences Appliquées de Toulouse, France, for his work on the computational principles of human walking via optimal control within the GEPETTO Group, Laboratoire d'Analyse et d'Architecture des Systèmes, CNRS, Toulouse, France, in 2007. In 2008 he joined the Robotics and Advanced Manufacturing Group at CINVESTAV, Saltillo, Mexico, where he is a researcher. His scientific interests are mainly focused on robot trajectory optimization and vision-based robot navigation.

Received: 21 February 2024
Revised: 7 June 2024
Accepted: 8 July 2024