

Chapter 3

Comparison and Analysis of Multibody Dynamics Formalisms for Solving Optimal Control Problem



Quentin Docquier, Olivier Brùls, and Paul Fiset

Abstract Optimal Control methods are increasingly used for the control of multi-body systems (MBS). This work analyzes the different dynamic formulations and compare their performances in solving Optimal Control Problem. The focus is on minimal coordinates and the derivation of the dynamics via the recursive methods for tree-like MBS (i.e., the so-called Newton-Euler and Order-N recursive algorithms). The different formulations are introduced and their derivations are discussed. A benchmark case study (i.e., a 3D series manipulator balancing an inverted pendulum) is modeled and a series of manipulation tasks (movement of the end effector in the 3D space) are performed. The OCP is formulated and solved with the help of the CasADi software while the dynamic formulations are generated by the Robotran software. Results show that the implicit and semi-explicit formulations derived via the Newton-Euler recursive algorithm lead to faster computation of the OCP than the explicit formulations. This is explained by a more compact expression for the implicit dynamics. However, a lower number of high local minima is observed with the explicit formulations for the most extreme robot manipulations.

Keywords Multibody system dynamics · Direct optimal control · Newton-Euler recursive algorithm · Tree-like system · 3D serial robot · Inverted pendulum

3.1 Introduction

Nowadays, Optimal Control (OC) methods are increasingly used for the control of mechanical systems. Many industrial applications already make use of it, e.g., for the trajectory planning of industrial robots, while the scientific community continues

Q. Docquier (✉) · P. Fiset
Université Catholique de Louvain, Louvain-la-Neuve, Belgium
e-mail: quentin.docquier@uclouvain.be; paul.fisette@uclouvain.be

O. Brùls
Université de Liège, Liège, Belgium
e-mail: o.bruls@uliege.be

© Springer Nature Switzerland AG 2019
E. Zahariev, J. Cuadrado (eds.), *IUTAM Symposium on Intelligent Multibody Systems – Dynamics, Control, Simulation*, IUTAM Bookseries 33,
https://doi.org/10.1007/978-3-030-00527-6_3

to investigate new possibilities in the fields of active orthoses and autonomous driving. Optimal Control of multibody systems (MBS) is of particular interest for many mechanical systems such as vehicles, robots or the human body as it offers the possibility to compute beforehand a control sequence on a given time interval such that a certain optimality criterion is achieved. This makes OC methods suitable for multibody systems whose dynamics involves non-linearities and rotations in the tridimensional space. Moreover, the computation of control command inputs via optimization enables the control of underactuated MBS characterized by a lower number of inputs than degrees of freedom. However, the performances of the optimized control inputs strongly depend on the mathematical model used to represent the dynamics of the controlled system.

Optimal Control is also essential for Model Predictive Control (MPC) of non-linear systems. MPC consists in determining the current control inputs by solving an Optimal Control Problem (OPC) on a finite future time horizon. In case of mechanical systems with fast dynamics, fast updates of the control inputs are required, which means that the OCP online solving must be carried out in real-time. This leads to many investigations on the OCP solving efficiency including for multibody control purpose.

Optimal Control of multibody systems has already been approached on many different angles. There have been a multitude of works to develop general purpose algorithms for optimal trajectory planning of MBS. Among them, studies have focused on *constrained* MBS characterized by loops of bodies and algebraic equations [12]. For such MBS, different methods of index reduction for the differential algebraic equations (DAE) system have been investigated (e.g., coordinate partitioning [21] and Baumgarte stabilization [3]). Such methods are also of great importance in the case of non-minimal parameterization of the system motion. In [11] and [14], different parameterizations of the special orthogonal group $SO(3)$ have been investigated with special integrators adapted to the parametrization. These non-minimal parametrizations are of particular interest for many applications (e.g., flying vehicles such as drones) as they offer a singular-free representation for the rotations.

On the other hand, minimal coordinates have also drawn the attention of researchers as they are appropriate for the representation of many industrial applications (e.g., serial manipulators). During the two past decades, the optimal trajectory planning of industrial robots has been a very popular research subject [2, 6, 18, 19]. These robots generally possess a tree-like structure (i.e., without loop of bodies) which leads to *unconstrained* systems in the case of a joint coordinates approach.

In robotics, the trajectory planning is usually referred as the inverse dynamics problem for which the objective is to determine the joint torques for a prescribed motion of the controlled system [2]. The use of optimization methods is often necessary for the control of underactuated or overactuated systems for which there is still a latitude in the way to actuate them. Among the studied problems, the case of non-minimal phase systems represents an additional challenge as the inverse

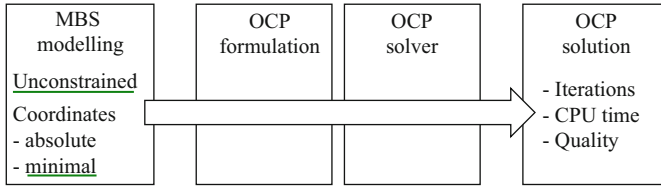


Fig. 3.1 In this work, the focus is on the influence of the MBS modeling on the solving OCP performances

dynamics is, in that case, non-causal which means that the control should start before the beginning of the trajectory [2].

Generally, the research on OC for MBS has mainly focused on the development of new methods for Numerical Optimal Control that are based on a given formulation of the multibody dynamics [2, 5, 19] and [17]. Nevertheless, in [6], Diehl gives a good overview on the different dynamic formulations and how they could influence the Optimal Control solver. He suggests that an implicit formulation of the dynamics through the Newton-Euler Recursive scheme should lead to a lower evaluation cost per iteration. However, to our knowledge, there has been no thorough investigation on the dynamic formulations and its influence on the solving of optimal control problem for *unconstrained* multibody systems. Regarding the numerical formulation of OCP, two methods exist: the multiple shooting and the direct collocation [4]. For the multiple shooting, the integration and the optimization are treated separately. The choice of the integrator is left free and can be adapted to the system dynamics. On the other hand, the direct collocation method consists in imposing the dynamics equations at intermediate collocation points. The integration and the optimization are treated at the same level. The advantages of collocation methods are that they lead to a very sparse NLP and that they show fast local convergence. In addition, they treat unstable system well and can easily cope with state and terminal constraints [6]. These reasons led us to consider the collocation method in the case study of this paper (see Sect. 3.4).

In this work, we analyze different multibody dynamic formulations and compare their performances in terms of convergence, accuracy and computation cost in the OCP framework. The aim is not to develop new numerical methods for the formulation of the OCP or to propose a new type of solver but to provide some insights on existing MBS formulations and their suitability in the formulation of Optimal Control Problems (see Fig. 3.1). Among the different MBS formalisms, we focus on minimal (i.e., relative) coordinates and the derivation of the dynamics via the recursive methods for tree-like MBS (i.e., the so-called Newton-Euler and Order-N recursive algorithms). The derivation is done symbolically through the use of the Robotran software [7]. The symbolic equation generator of Robotran has already been proved to be particularly efficient in the derivation of compact expressions for the dynamics of multibody system [15]. However, its performances for OCP solving have never been thoroughly analyzed.

In order to compare the formulations, a benchmark has been developed. It consists of a 3D serial robot arm with an inverted pendulum attached to the end effector. An important point is the underactuated nature of the system (due to the free rotation of the pendulum) which requires the use of optimization to determine the robot control inputs. In addition, the balancing of an inverted pendulum is a well-known task that has already been largely investigated. The system being tridimensional, its non-linear and complex dynamics should highlight the differences between the considered dynamic formulations.

This chapter is organized as follows: first the theory related to the formulation of the OCP and the multibody modeling are respectively presented in Sects. 3.2 and 3.3. The different formulations are introduced and their derivations are discussed. Then, Sect. 3.4 presents a benchmark case study for an *unconstrained* multibody system. First the MBS system (i.e., a 3D robot arm with an inverted pendulum) is described, then the optimal control task is defined. To this extent, the CasADi software [1] is used to formulate and solve the OCP while the dynamic formulations are provided by the symbolic MBS software Robotran [7]. Finally, results are presented for a series of tasks realized with the different representations of the dynamics. These are compared in terms of solver performances: number of iterations, CPU time to convergence and quality of the solutions. Finally, the differences observed for the different formulations are discussed in light of theoretical considerations.

3.2 Optimal Control of Multibody Systems

There are many ways to formulate and solve an Optimal Control Problem. In this Section, we introduce the main principle and define the trajectory planning problem to solve. In Sect. 3.2.1, the continuous problem is presented analytically then the discrete formulation is introduced in Sect. 3.2.2.

3.2.1 Optimal Control Formulation

In a multibody context, the optimal Control Problem (OCP) is formulated as an optimization problem whose solution provides the control trajectory $\mathbf{u}(t)$ that minimizes a given cost function (3.1) and steers the state $\mathbf{x}(t)$ of the multibody system and its algebraic variables $\mathbf{z}(t)$ from a given initial state (3.2) to a given terminal state (3.3), within a given time horizon T (see Fig. 3.2). The OCP then reads:

$$\underset{\mathbf{x}(t), \mathbf{u}(t), \mathbf{z}(t)}{\text{minimize}} \int_0^T L(\mathbf{x}(t), \mathbf{u}(t), \mathbf{z}(t)) dt \quad (3.1)$$

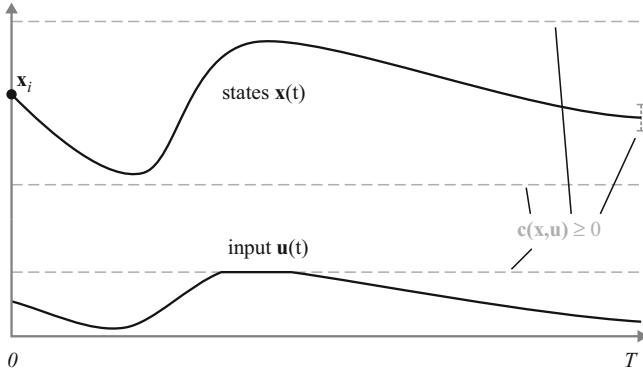


Fig. 3.2 The continuous optimal control problem consists in finding the optimal state and control trajectory on the time interval $[0, T]$

subject to possible path constraints:

$$\mathbf{x}(0) - \mathbf{x}_i = \mathbf{0} \quad (3.2)$$

$$\mathbf{x}(T) - \mathbf{x}_d = \mathbf{0} \quad (3.3)$$

$$\mathbf{c}(\mathbf{x}(t), \mathbf{u}(t), \mathbf{z}(t)) \geq \mathbf{0} \quad t \in [0, T] \quad (3.4)$$

The trajectory $\mathbf{x}(t), \mathbf{z}(t)$ and the control inputs $\mathbf{u}(t)$ are subjected to equality constraints denoted \mathbf{g} , which typically represent the system dynamics expressed as a set of Differential Algebraic Equations (DAE).

$$\mathbf{g}(\mathbf{x}(t), \mathbf{u}(t), \mathbf{z}(t)) : \begin{cases} \dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, \mathbf{z}) \\ \mathbf{a}(\mathbf{x}, \mathbf{z}) = \mathbf{0} \end{cases} \quad t \in [0, T] \quad (3.5)$$

In this equation, \mathbf{f} and \mathbf{a} respectively denote the differential and the algebraic equations that described the physics of the system in terms of the states \mathbf{x} , the algebraic variables \mathbf{z} and the control inputs \mathbf{u} . This general framework for the dynamics is at the core of the CasADi OC program and its permits the modeling of the dynamics of the multibody system as a system of DAE (see Sect. 3.3). The optimization problem defined by the set of Eqs. (3.1), (3.2), (3.3), (3.4) and (3.5) has in general no analytical solution. However it can be solved numerically through its discretization on the time interval $[0, T]$.

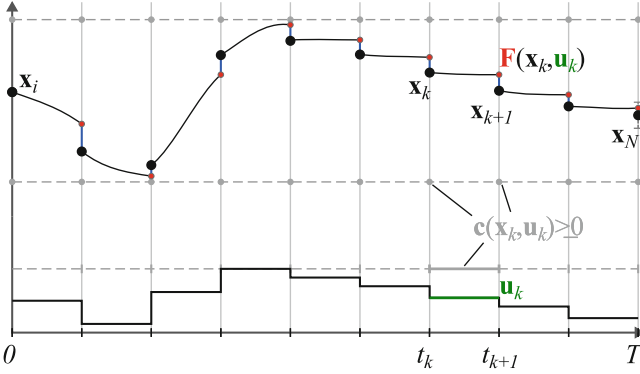


Fig. 3.3 Numerical OCP with the dynamic constraints (3.10) that are unsatisfied. The decision variables are the states at the discrete points and the piecewise constant control inputs

3.2.2 Numerical Optimal Control Methods

There are several approaches to obtain a numerical solution of an OCP. Among them, the *direct methods* consist in discretizing the time horizon $[0, T]$ into N time intervals $[t_k, t_{k+1}]$ such that $t_0 = 0$, $t_N = T$ and $k \in 0, 1, \dots, N - 1$. The time functions $\mathbf{x}(t)$, $\mathbf{u}(t)$ and $\mathbf{z}(t)$ that were the unknowns of the system (3.1), (3.2), (3.3), (3.4) and (3.5) are replaced by a sequence of $N + 1$ discrete states \mathbf{x}_k , \mathbf{z}_k and N piecewise constant controls \mathbf{u}_k which are now the decision variables of the optimization problem in its discrete form (see Fig. 3.3).

The cost function (3.1) is replaced by a numerical approximation p which is a function of the decision variables vector $\mathbf{w} = \{\mathbf{x}_0, \mathbf{u}_0, \mathbf{z}_0, \dots, \mathbf{x}_k, \mathbf{u}_k, \mathbf{z}_k \dots \mathbf{u}_{N-1}, \mathbf{z}_{N-1}, \mathbf{x}_N\}$. Finally, the path constraints (3.2) and (3.3) are re-expressed in terms of the decision variables vector \mathbf{w} . This parameterization of the original continuous OCP is referred as a nonlinear program (NLP).

$$\underset{\mathbf{w}}{\text{minimize}} p(\mathbf{w}) \quad (3.6)$$

subject to path constraints on the decision variables:

$$\mathbf{x}_0 - \mathbf{x}_i = \mathbf{0} \quad (3.7)$$

$$\mathbf{x}_N - \mathbf{x}_d = \mathbf{0} \quad (3.8)$$

$$\mathbf{c}(\mathbf{w}) \geq \mathbf{0} \quad (3.9)$$

In order to respect the system dynamics, the constant control input \mathbf{u}_k on the time interval $[t_k, t_{k+1}]$ is linked to the two adjacent differential states \mathbf{x}_k and \mathbf{x}_{k+1} through the explicit numerical integration \mathbf{F} of the continuous dynamics (3.5) on the same discrete time interval (see Fig. 3.3).

$$\mathbf{x}_{k+1} - \mathbf{F}(\mathbf{x}_k, \mathbf{u}_k, \mathbf{z}_k) = 0 \quad (3.10)$$

In addition, the differential and algebraic states are linked at each discrete time through the set of algebraic equations.

$$\mathbf{a}(\mathbf{x}_k, \mathbf{z}_k) = 0 \quad (3.11)$$

As mentioned in the introduction, a collocation method is used to deal with the constraints (3.10) and (3.11). This technique consists in defining collocation points between t_k and t_{k+1} at which the states, represented as polynomial functions, have to satisfy the continuous dynamics (3.5). The integration scheme implies the definition of extra variables (i.e., the polynomial coefficients) and additional constraint equations. This leads to a very large but sparse NLP.

The optimization problem defined by Eqs. (3.6), (3.7), (3.8), (3.9), (3.10) and (3.11) can be solved with different techniques (e.g., an Interior Point method) which generally require the derivatives of the cost function and constraint equations with respect to the decision variables. To this extent, the software CasADi [1] provides sensitivity analysis tools for ODE and DAE system that allows the automatic generation of the derivatives necessary to solve the NLP. To do so, CasADi uses state-of-the-art algorithmic differentiation [10].

3.3 Multibody Systems Modeling

It is well-established that several formalisms can be used to model multibody systems (MBS). Among them, both the absolute and relative coordinates can be considered for optimal control, resulting in different structures for the NLP to solve [2]. In this study, the focus is on minimal coordinates and the different formulations of the dynamics. First the relative (i.e., minimal) coordinates approach is briefly described. Then the different forms for the MBS dynamics that can be obtained through recursive algorithms are presented. Finally, the symbolic generation of these dynamics equations is discussed.

3.3.1 Relative Coordinates Approach

In the relative coordinates approach, the parametric configuration of a body is defined relatively to another one called its parent body. The motion degrees of freedom (dof) are defined via the joints that connect the different bodies of the system. If the MBS does not contain any loops of bodies, it inherits a tree-like structure as each body can only have one parent body (see Fig. 3.4). The number of coordinates is minimal and thus reduced compared to the absolute coordinates approach, as the joint constraints are implicitly taken into account by the relative formulation, leading to a pure set of ordinary differential equations.

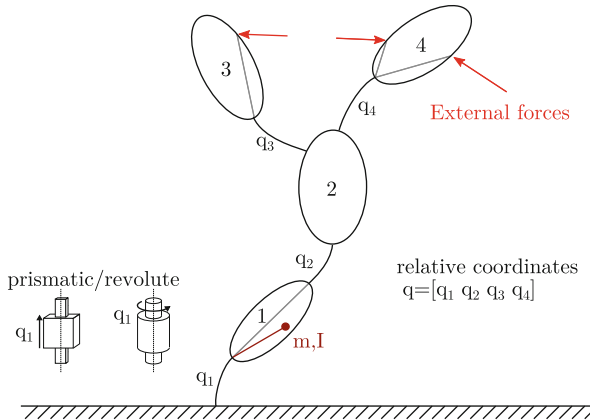


Fig. 3.4 The relative coordinates approach consists in defining the motion of each body relative to its parent body with the help of a joint coordinate q

In the relative coordinates approach, a MBS is fully characterized by two types of features:

- bodies, defined by their mass, center of mass and inertia tensor (10 parameters in total),
- joints, defined by their location on the body and their nature (i.e., prismatic or revolute, free or constrained).

A tree-like structure might be quite restrictive since many mechanical systems contain kinematic loops (e.g., a four-bar linkage). The relative coordinates approach deals with such closed-loop systems by introducing loop constraint equations. The tree-like structure is restored in two steps. First the loop is cut resulting into two independent branches of bodies. Afterwards the loop closure is ensured by applying algebraic constraints, for instance forcing two points of the created branches to coincide or imposing a given distance between two points. Hence, the derivation of the *differential* equations is based on the tree-like structure of the MBS whether or not the system is constrained. In this work, only unconstrained tree-like MBS are considered within the frame of OCP formulation. However, as the derivation of the differential equations is the same for constrained system, the study of the different formulations of the dynamics will be insightful for both constrained and unconstrained MBS.

3.3.2 Formulations of the Dynamic Equations

In this section, different formulations of the equations of motion are presented for the case of *unconstrained* MBS. Among those formulations, the so-called *Newton-Euler recursive* scheme denoted NER is particularly suitable for minimal coordinates [13]. This method implements two consecutive recursions on the MBS

(a forward kinematics followed by a backward dynamics). It is well-known for its facility of computer implementation and its reduced number of operations necessary to obtain the equations of motion. It expresses the vector of generalized forces \mathbf{Q} as a function of the system kinematics $(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})$.

$$\Phi(\ddot{\mathbf{q}}, \dot{\mathbf{q}}, \mathbf{q}) = \mathbf{Q} \quad (3.12)$$

This very efficient formulation, also called inverse dynamics has only a $\mathcal{O}(N)$ complexity thanks to its recursive formulation, N being the size of the MBS (i.e., the number of joints). It is extensively used by roboticists to determine the actuation torques \mathbf{Q} necessary to track a given trajectory $(\mathbf{q}(t), \dot{\mathbf{q}}(t), \ddot{\mathbf{q}}(t))$ for a fully actuated robot.

3.3.2.1 Implicit Formulation

More generally, Eq.(3.12) can be used to represent the dynamics of any MBS systems, in a residual form. In that case, the joint torques/forces \mathbf{Q} can be configuration-dependent (i.e., an elastic joint or a feedback control law) and are generally expressed as a function of the generalized coordinates \mathbf{q} and generalized velocities $\dot{\mathbf{q}}$. It results in an implicit formulation of the equations of motion in terms of the generalized acceleration $\ddot{\mathbf{q}}$:

$$\Phi(\ddot{\mathbf{q}}, \dot{\mathbf{q}}, \mathbf{q}) - \mathbf{Q}(\dot{\mathbf{q}}, \mathbf{q}) = \mathbf{0} \quad (3.13)$$

According to Sect. 3.2.1, some control inputs $\mathbf{u}(t)$ are considered for the OCP formulation. To this extent, the joint forces \mathbf{Q} can be partitioned into two subsets.

- \mathbf{Q}_a for the active or actuated joints (input of the system)
- \mathbf{Q}_p for the passive joints that can contain dry friction, damping force, etc.

$$\mathbf{Q}(\dot{\mathbf{q}}, \mathbf{q}) = \mathbf{Q}_a + \mathbf{Q}_p(\dot{\mathbf{q}}, \mathbf{q}) \quad (3.14)$$

The system (3.13) can be re-expressed as a DAE system of the form (3.5) with the generalized coordinates and velocities representing the differential states and the accelerations considered as the algebraic states.

$$\mathbf{x} = \begin{bmatrix} \dot{\mathbf{q}} \\ \mathbf{q} \end{bmatrix} \quad \mathbf{z} = \ddot{\mathbf{q}} \quad \mathbf{u} = \mathbf{Q}_a$$

$$\begin{cases} \mathbf{f}(\mathbf{x}, \mathbf{u}, \mathbf{z}) = \begin{bmatrix} \mathbf{z} \\ \dot{\mathbf{q}} \end{bmatrix} \\ \mathbf{a}(\mathbf{x}, \mathbf{z}) = \Phi(\mathbf{q}, \dot{\mathbf{q}}, \mathbf{z}) - \mathbf{Q}(\mathbf{q}, \dot{\mathbf{q}}, \mathbf{u}) = \mathbf{0} \end{cases} \quad (3.15)$$

3.3.2.2 Semi-explicit Formulation

The original Newton-Euler recursive scheme leading to the formulation (3.12) can be modified in order to obtain a semi-explicit form of the dynamics equations for tree-like multibody systems. This new form still preserves the full recursivity even for the computation of the mass matrix [9].

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{c}(\mathbf{q}, \dot{\mathbf{q}}) = \mathbf{Q}(\dot{\mathbf{q}}, \mathbf{q}) \quad (3.16)$$

\mathbf{M} is the symmetric generalized mass matrix of the system and \mathbf{c} is the non-linear dynamic vector which contains the gyroscopic, centripetal and gravity terms as well as the contribution of the resultant external forces. The difference compared to Eq. (3.12) is that the algorithm first computes \mathbf{M} and \mathbf{c} individually before the evaluation of the left-hand side.

This semi-explicit formulation is often preferred for time integration purposes. However, the factorization of the mass matrix in the recursive scheme requires additional computational efforts of $\mathcal{O}(N^2)$ complexity [9]. The semi-explicit system (3.16) can be re-expressed as a DAE system in a similar way as for the implicit formulation.

$$\begin{aligned} \mathbf{x} &= \begin{bmatrix} \dot{\mathbf{q}} \\ \mathbf{q} \end{bmatrix} & \mathbf{z} &= \ddot{\mathbf{q}} & \mathbf{u} &= \mathbf{Q}_a \\ \left\{ \begin{array}{l} \mathbf{f}(\mathbf{x}, \mathbf{u}, \mathbf{z}) &= \begin{bmatrix} \mathbf{z} \\ \dot{\mathbf{q}} \end{bmatrix} \\ \mathbf{a}(\mathbf{x}, \mathbf{z}) &= \mathbf{M}(\mathbf{q}) \mathbf{z} + \mathbf{c}(\mathbf{q}, \dot{\mathbf{q}}) - \mathbf{Q}(\dot{\mathbf{q}}, \mathbf{q}, \mathbf{u}) = \mathbf{0} \end{array} \right. \end{aligned} \quad (3.17)$$

3.3.2.3 Explicit Formulation

The system of Eq. (3.16) can be solved in terms of the accelerations $\ddot{\mathbf{q}}$ via a Cholesky decomposition of the mass matrix leading to the explicit expression $\boldsymbol{\gamma}$ for the acceleration $\ddot{\mathbf{q}}$ in terms of the position \mathbf{q} and velocity $\dot{\mathbf{q}}$. This resolution step yields an additional $\mathcal{O}(N^3)$ complexity compared to the semi-explicit form.

$$\ddot{\mathbf{q}} = \mathbf{M}^{-1} [\mathbf{Q} - \mathbf{c}] (\dot{\mathbf{q}}, \mathbf{q}) \triangleq \boldsymbol{\gamma}(\dot{\mathbf{q}}, \mathbf{q}, \mathbf{Q}) \quad (3.18)$$

Another way to derive the explicit form of the equations of motion in a fully recursive way can be obtained through the Order-N algorithm [16]. This method is based on three recursive steps (instead of two for the NER scheme) on the multibody system. In short, first a forward recursion covers the MBS from the inertial body to the leaves to compute the bodies *kinematics* (position, orientation, linear and angular velocities). A second backward recursion computes the bodies *dynamics* on

the basis of the classical Newton-Euler equations within which a subtle embedded block factorization of the mass matrix is performed. This process prepares the third final *kinetic* recursion that straightforwardly computes the generalized acceleration $\ddot{\mathbf{q}}$ as a function \mathbf{v} of the joint positions and velocities. This three-steps algorithm has a $\mathcal{O}(N)$ complexity. Several implementations of it have been proposed in the literature, the one used in this paper is the Schwertassek-Rulka method [16].

$$\ddot{\mathbf{q}} = \mathbf{v}(\dot{\mathbf{q}}, \mathbf{q}, \mathbf{Q}) \quad (3.19)$$

As the system is expressed in a purely explicit form, the accelerations $\ddot{\mathbf{q}}$ do not *necessarily* need to be considered as an algebraic state. As a result, the dynamics can be assimilated to either set of ODE or DAE. The explicit dynamics is generally denoted ξ and can either represent the NER explicit dynamics γ or the Order-N dynamics \mathbf{v} . The dynamics is represented by a set of differential algebraic equations (DAE) as

$$\begin{aligned} \mathbf{x} &= \begin{bmatrix} \dot{\mathbf{q}} \\ \mathbf{q} \end{bmatrix} & \mathbf{z} &= \ddot{\mathbf{q}} & \mathbf{u} &= \mathbf{Q}_a \\ \left\{ \begin{array}{l} \mathbf{f}(\mathbf{x}, \mathbf{u}, \mathbf{z}) &= \begin{bmatrix} \mathbf{z} \\ \dot{\mathbf{q}} \end{bmatrix} \\ \mathbf{a}(\mathbf{x}, \mathbf{z}) &= \mathbf{z} - \xi(\dot{\mathbf{q}}, \mathbf{q}, \mathbf{u}) = \mathbf{0} \end{array} \right. \end{aligned} \quad (3.20)$$

or by a set of ordinary differential equations (ODE) as

$$\begin{aligned} \mathbf{x} &= \begin{bmatrix} \dot{\mathbf{q}} \\ \mathbf{q} \end{bmatrix} & \mathbf{u} &= \mathbf{Q}_a \\ \mathbf{f}(\mathbf{x}, \mathbf{u}) &= \begin{bmatrix} \xi(\dot{\mathbf{q}}, \mathbf{q}, \mathbf{u}) \\ \dot{\mathbf{q}} \end{bmatrix} \end{aligned} \quad (3.21)$$

3.3.3 Symbolic Generation

In practice, the MBS equations of motion can be computed either numerically or symbolically. In specific cases, the recursive nature of the relative coordinates can be deeply exploited by a symbolic program dedicated to multibody systems. This is the purpose of the Robotran software [7] that offers symbolic tools for an extensive simplification of the equations of motion. The latter are written under the form of a function (C, Matlab, Python) with adequate input/output arguments. These functions, internally optimized thanks to the symbolic approach, can be used externally as a black-box by another program or computer environment.

Table 3.1 Formulations for the dynamics equations of *unconstrained* multibody systems

	Algorithm	Section	Dynamics	Complexity
IMPL	Original NER	3.3.2.1	$\Phi - \mathbf{Q} = \mathbf{0}$	N
SEMI	Modified NER	3.3.2.2	$\mathbf{M}\ddot{\mathbf{q}} + \mathbf{c} - \mathbf{Q} = \mathbf{0}$	N^2
EXP1	Modified NER + Cholesky	3.3.2.3	$\ddot{\mathbf{q}} - \boldsymbol{\gamma} = \mathbf{0}$	$N^2 + N^3$
EXP2	Order-N	3.3.2.3	$\ddot{\mathbf{q}} - \mathbf{v} = \mathbf{0}$	N

In this paper, the function generated by Robotran will be used by the CasADi software [1] to express the system dynamics for the OCP formulation.

The different formulations of the dynamics yield different levels of complexity as summarized in Table 3.1.

3.4 Case Study: Unconstrained MBS

Given their different complexities, the formulations of the dynamics should lead to distinct performances for the OCP solving. To this extent, the Optimal Control of a simple unconstrained multibody system has been extensively tested.

3.4.1 System Presentation

The system of interest is a 3D robot arm with an inverted pendulum attached to the end effector as represented in Fig. 3.5. The 3D arm has a morphology that resembles the human arm. It is composed of 3 bodies and 4 actuated rotational joints (with relative coordinates q_1, q_2, q_3 and q_4). The first element is connected to the base with two rotational joints enabling to point in any direction of space. Then the second and third elements are connected in series via a single rotational joint. The relative motion of the pendulum with respects to the end effector is characterized by two Cardan-type angles (q_5 and q_6) which are considered as free joints (i.e., frictionless and non-actuated). The MBS is thus composed of 4 bodies and 6 joints. The system is underactuated as only the 4 joints of the arm are actuated. The mass, inertia and length of the different bodies are given in Table 3.2.

3.4.2 OCP Formulation

There are many possible ways to formulate the Optimal Control Problem for this application. In this study, the focus is on the trajectory optimization of the 3D arm. The task is to move the system from an initial pose and position its end effector at a given location in the 3D space after an imposed time T . At this final time, the system must be at equilibrium with the pendulum in a perfect upright position.

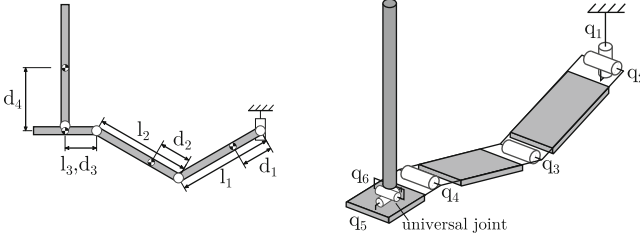


Fig. 3.5 Geometrical representation of the studied MBS system

Table 3.2 Geometric and inertia parameters for the 3D serial robot and the pendulum

	Length [m]	c.o.m. position [m]	Mass [kg]	Inertia [kg m ²]
Upperarm	$l_1 = 0.3$	$d_1 = 0.1$	2.5	[0.06,0.1,0.1]
Forearm	$l_2 = 0.3$	$d_2 = 0.1$	2	[0.03,0.05,0.05]
Hand	$l_3 = 0.1$	$d_3 = 0.1$	0.5	[0.004,0.006,0.006]
Pendulum		$d_4 = 0.2$	1	[0.1,0.1,0.05]

c.o.m. stands for the center of mass

The objective function is designed to minimize the actuation via the use of the square of the actuation torques and its time derivative:

$$L(\mathbf{x}, \mathbf{u}, \mathbf{z}) = k_1 \sum_{i=0}^4 Q_i^2 + k_2 \sum_{i=0}^4 \dot{Q}_i^2 \tag{3.22}$$

The first term aims at minimizing the torques in the four actuated joints while the second term minimizes the derivative of the controls which smooths the optimal trajectory and limits the mechanical power transmitted into the joints. In practice, the control torques are piece-wise constant on each discrete time interval, which implies the computation of the derivative via finite differences.

To achieve the equilibrium at the final time T , constraints are applied on the differential states $\mathbf{x} = [\dot{\mathbf{q}} \ \mathbf{q}]$, the inputs $\mathbf{u} = \mathbf{Q}_a$ and the algebraic states \mathbf{z} when appropriate.

- The absolute position of the end effector can be expressed as a function of the angular joint positions, $\mathbf{s}_{ee}(\mathbf{q})$. Thus, the end effector can be constrained to a desired absolute Cartesian position $\mathbf{p}_d = [x_d \ y_d \ z_d]$:

$$\mathbf{p}_d - \mathbf{s}_{ee}(\mathbf{q}(T)) = \mathbf{0} \tag{3.23}$$

- The absolute velocity of the end effector being a linear function of the joint angular velocities, equating the angular velocities at the final time T to zero should then be sufficient.

$$\dot{\mathbf{q}}(T) = \mathbf{0} \tag{3.24}$$

- The equilibrium of the whole system is achieved by canceling the joint accelerations at the final time, $\ddot{\mathbf{q}}(T) = \mathbf{0}$. To this extent, the joint accelerations are either available as algebraic variables in the case of a DAE system or expressed with the function ξ in the case of a pure ODE system (see Sect. 3.3.2.3 on explicit formulations).

$$\mathbf{z}(T) = \mathbf{0} \quad \text{or} \quad \xi(\dot{\mathbf{q}}, \mathbf{q}, \mathbf{Q}_a) = \mathbf{0} \quad (3.25)$$

At time $t = 0$, the robot is in a stable equilibrium. The initial condition of the system is given by the joint angles, \mathbf{q}_i .

$$\mathbf{q}(0) = \mathbf{q}_i = [0; \alpha; -2\alpha; \alpha; 0; 0] \quad \dot{\mathbf{q}}(0) = \mathbf{0} \quad \ddot{\mathbf{q}}(0) = \mathbf{0} \quad (3.26)$$

In addition, path inequality constraints (3.27) are applied to the joint angles to keep a physical sense for the arm and the pendulum motions.

$$\begin{aligned} -\infty &\leq q_1 \leq \infty \\ -\pi/2 &\leq q_2 \leq \pi/2 \\ -5\pi/12 &\leq q_3 \leq 0 \\ -\pi/2 &\leq q_4 \leq \pi/2 \\ -\pi/4 &\leq q_5 \leq \pi/4 \\ -\pi/4 &\leq q_6 \leq \pi/4 \end{aligned} \quad (3.27)$$

The equilibrium condition (3.25), the terminal constraint (3.23) and the path constraints ensure all together that the pendulum is in an upright position at the final time.

3.4.3 Numerical Results

In order to assess the performances of the different formulations, we have compared the solutions obtained for the above 3D arm benchmark. For the comparison, a total of six formulations for the system dynamics have been considered. From Sect. 3.3, four different formulations for the equations of motion have been derived:

- IMPL: Implicit formulation from the Newton-Euler Recursive scheme (NER),
- SEMI: Semi-explicit formulation from the NER scheme,
- EXP1: Explicit formulation from the NER scheme,
- EXP2: Explicit formulation from the Order-N formalism.

While the first two formulations imply a DAE system for its numerical representation, the two explicit formulations can either be numerically treated as a system

of DAE or as a system of pure ODE. The difference lies in the number of decision variables and constraint equations for the numerical NLP but also in the treatment of the terminal constraints.

- EXPi-O: system of ODE with the position and velocity in the differential states,
- EXPi-A: system of DAE with the joint accelerations as algebraic variables.

The OCP described in Sect. 3.4.2 is formulated with the help of the software environment CasADi [1]. Based on the functions generated by Robotran, symbolic expressions of the dynamics are written and symbolically derived with respect to the states and input variables. Then, the numerical OCP is formulated using a direct collocation scheme [4] allowing the treatment of either a DAE or a pure ODE representation of the dynamics. The resulting NLP is then solved using the Interior Point solver IPOPT [20].

3.4.3.1 Simulation Settings

The performances of each formulation are evaluated through the trajectory optimization of the 3D arm robot. The task is described in Sect. 3.4.2. The time interval length T is 2 s and is divided into 100 time intervals for the numerical solving via a direct collocation method.

To compare the formulations, the optimization was performed for a series of end-effector terminal locations. This set of points is defined in cylindrical coordinates (φ, r, z) around a vertical axis passing through the robot arm origin. The angular position φ varies in the range $[-\pi/2, \pi/2]$ rad. On the other hand, the two other coordinates have relatively small variation ranges because the end effector has to remain in the robot working space: the radius is fixed to 0.63 m and the height varies in the range $[-0.1, 0.1]$ m (see Fig. 3.6).

Among the defined lateral surface, many points can be tested (see Fig. 3.6). The number of trajectory optimizations was chosen on the basis of a convergence analysis. The optimizations have been performed on a grid of different sizes and it has been shown that the mean number of iterations does not vary significantly between two refinements after a certain size (see Fig. 3.7). The chosen grid is composed of 21 angular orientations and 11 vertical positions for a total of 231 points in the 3D space (see Fig. 3.6).

In addition, two different sets of results are generated by tuning the k_1 and k_2 coefficients. The first cost function is defined by $(k_1 = 1e0, k_2 = 0)$ and corresponds to a more aggressive actuation while the second cost function is defined by $(k_1 = 1e0, k_2 = 1e5)$ and should result in a smoother actuation. In both cases, the initial condition (i.e., the α angle) is such that the end effector is positioned at 0.65 m from the origin with zero lateral rotation of the arm (i.e., $\varphi = 0$ rad).

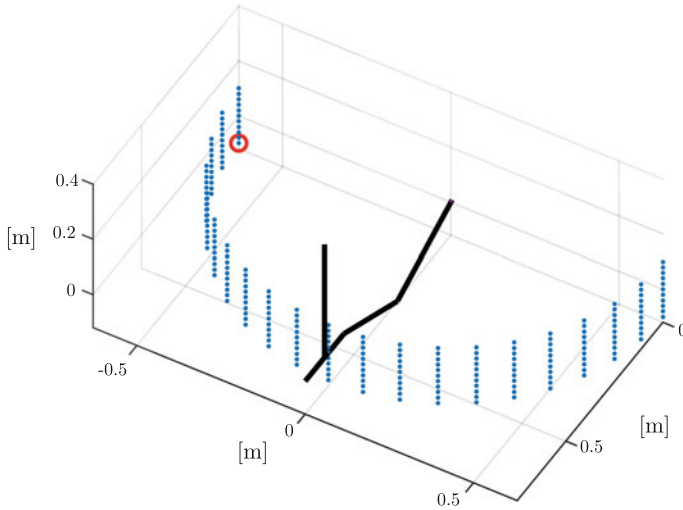


Fig. 3.6 Grid of the 3D point to reach. The multibody system is represented in its initial condition

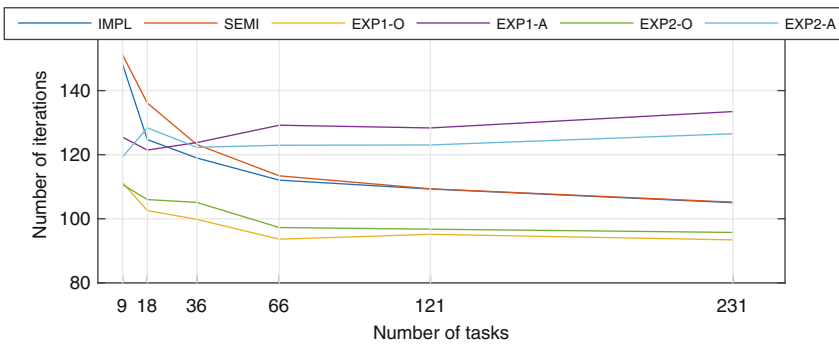


Fig. 3.7 Convergence analysis for the different formulations. After a given number of tasks, the number of iterations does not vary significantly

3.4.3.2 Simulation Results

Figure 3.8 shows the optimal trajectory obtained to reach the point $\mathbf{p}_d = \{0, 0.63, -0.1\}$ (see red point in Fig. 3.6) for the two different definitions of the cost function. The second definition leads to a smoother but more important actuation. The torques variations are reduced and there are less oscillations at the end of the trajectory. This smoother actuation is reflected in the joint velocities with lower values than for the non-smoothed actuation. Finally, for the two cost definitions, the same final angular positions and the same final actuation are obtained as the equilibrium to reach after 2 s is identical.

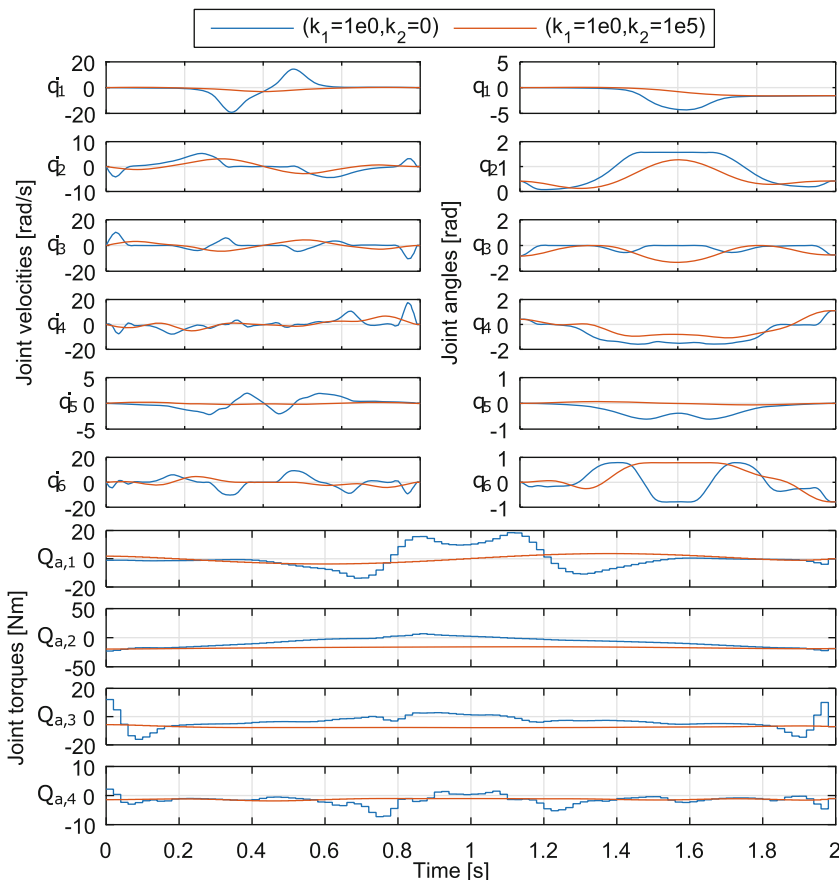


Fig. 3.8 Solution for the two different sets of cost function coefficients. A non-zero coefficient k_2 on the control derivative leads to a smoother but more important actuation

More than the solution itself, the interest of the study is to compare the different formulation performances to solve the OCP described in Sect. 3.4.2. To do so, the number of iterations before convergence, the cost function value at the optimum and the mean evaluation times of the dynamics are evaluated for the two sets of results.

First, the optimal values of the cost function are compared for the six considered representations of the dynamics. Figure 3.9 shows a box-plot¹ of the cost value for the two sets of results (i.e., different coefficients in the cost function). The OCP formulation with the smoothed actuation leads to more constant results with only

¹On the box plot, the central mark indicates the median, and the bottom and top edges of the box indicate the 25th and 75th percentiles, respectively. The whiskers extend to the most extreme data points not considered outliers, and the outliers are plotted individually using the ‘+’ symbol.

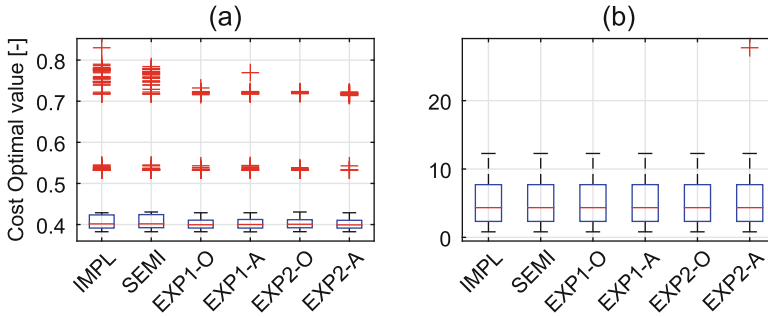


Fig. 3.9 Optimal values of the cost function represented in the form of a box plot for the two sets of solutions. (a) First cost function without smoothing: $k_1 = 1e0$, $k_2 = 0$. (b) Second cost function with smoothing: $k_1 = 1e0$, $k_2 = 1e5$

one formulation presenting an outlier which corresponds to a higher local minima (see Fig. 3.9b). On the other hand, in the case of the non-smoothed control, the solution leads to a higher number of outliers especially in the case of the implicit and the semi-explicit formulations (see Fig. 3.9a). Note that the higher group of outliers corresponds to the tasks where the point to reach is either at the far left or the far right and the lower group corresponds to the central point (i.e., no lateral movement needed for the arm). The higher outliers clearly correspond to higher local minima as lower minima are reached by the explicit formulations for the same tasks. Regarding the problem with no lateral movement, it is not clear whether it is a local minimum as all the formulations seem to struggle to solve this problem. In general, the actuation smoothing seems to reduce the problem of higher local minima.

Another important aspect of the performances is the computation efforts needed to reach the optimal solution. Figure 3.10 shows the number of iterations to convergence and the CPU time for the two sets of results. Figure 3.11 shows the CPU time per iteration. For the first set (see Fig. 3.10a), the number of iterations is characterized by a high number of outliers corresponding to the high local minima (see Fig. 3.9a). On this same set, the DAE explicit formulations result in a larger average number of iterations than the four other formulations. For the second set of tasks, the implicit and semi-explicit formulation generally perform better than the explicit ones (see Fig. 3.10b). This difference could be due to the high local minima that are encountered by the first two formulations while solving the first set of tasks. Regarding the CPU time, the implicit and the semi-explicit formulations lead to better results than the explicit ones for both sets of tasks. This tendency is also observed for the CPU time per iteration (see Fig. 3.11).

Regarding the numerical formulations, the ODE representation gives better results in terms of iterations for the first set of tasks (see Fig. 3.10a) but this difference is not so clear for the second set of tasks (see Fig. 3.10b). The two explicit

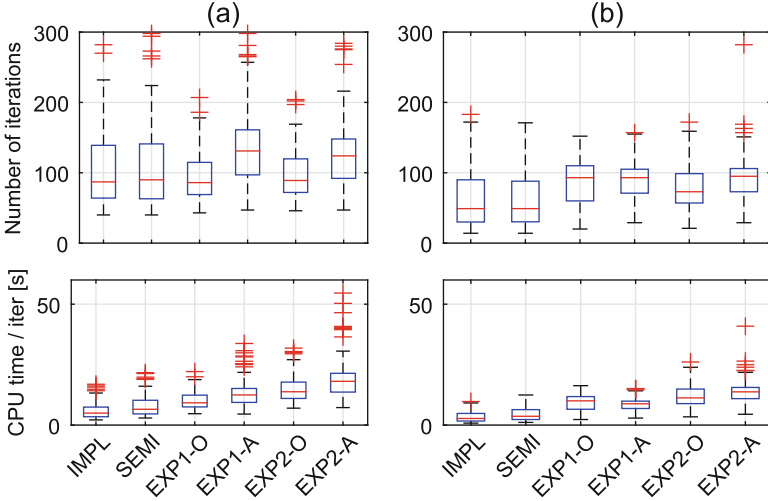


Fig. 3.10 Number of iterations and total CPU time to convergence represented in the form of a box plot for the two sets of solutions. (a) First cost function without smoothing: $k_1 = 1e0$, $k_2 = 0$. (b) Second cost function with smoothing: $k_1 = 1e0$, $k_2 = 1e5$

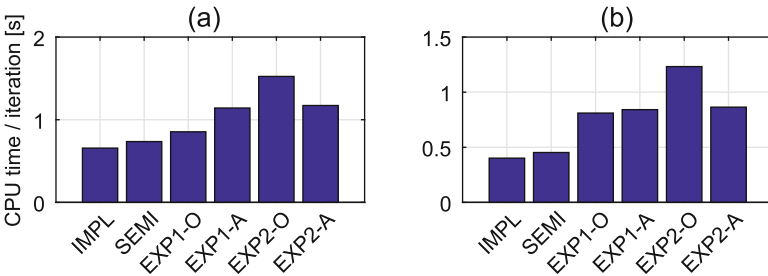


Fig. 3.11 Mean CPU time per iteration for the two sets of solutions. (a) First cost function without smoothing: $k_1 = 1e0$, $k_2 = 0$. (b) Second cost function with smoothing: $k_1 = 1e0$, $k_2 = 1e5$

forms exhibit similar performances in terms of CPU time when formulated as ODE and DAE systems (see Fig. 3.10). However, the Order-N implementation yields a lower CPU time per iteration when formulated as a DAE system (see Fig. 3.11). Its poor performances in terms of total CPU time are explained by its high number of iterations to reach convergence.

3.4.3.3 Discussion

In this section, we aim at giving some insights about the obtained results. From the simulations, it is clear that the implicit and the semi-explicit formulations perform better than the explicit ones. From a numerical point of view, the DAE form leads

Table 3.3 Size of the NLP associated to the OCP for the second cost definition (smoothed actuation)

Number of	IMP	SEMI	EXP1-O	EXP1-A	EXP2-O	EXP2-A
Variables	8812	8812	6412	8812	6412	8812
Constraint equations	8427	8427	6027	8427	6027	8427
nzt ^a in constraint Jacobian	72,435	72,435	62,519	69,635	62,519	69,635
nzt in Lagrangian Hessian	39,021	39,021	35,286	35,021	35,286	35,021

^anzt stands for non-zero terms

Table 3.4 Dynamic formulations and their generation for the studied MBS

	Algorithm	Dynamic terms	# Flops	# Eqs
IMPL	Original NER	$\Phi(\ddot{\mathbf{q}}, \dot{\mathbf{q}}, \mathbf{q})$	460	114
SEMI	Modified NER	$\mathbf{M}(\mathbf{q})$ and $\mathbf{c}(\dot{\mathbf{q}}, \mathbf{q})$	830	264
EXP1	Modified NER + Cholesky	$\gamma(\dot{\mathbf{q}}, \mathbf{q}, \mathbf{Q})$	991	328
EXP2	Order-N	$\nu(\dot{\mathbf{q}}, \mathbf{q}, \mathbf{Q})$	1784	395

to a larger problem as both the number of equations and variables is higher (see Table 3.3). This difference results from the treatment of joint accelerations as algebraic variables which is mandatory for the implicit and semi-explicit dynamics. Nevertheless, compared to a pure ODE representation, the differential equation $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, \mathbf{z})$ is trivial in the DAE case. In addition, the DAE formulation allows the definition of the terminal equilibrium constraint directly in terms of the algebraic variables, which can be advantageous as the expression of the equilibrium constraints is linear in that case.

The explanation for the performance differences lies in the derivation of the dynamics itself. The different formulations are all written in the form of a function generated symbolically with Robotran. Because they inherit different complexity, the number of equations and floating point operations associated with each formulation differ as shown in Table 3.4. The $\mathcal{O}(N)$ computation of the Φ term does in fact lead to a lower number of operations, about half of the \mathbf{M} and \mathbf{c} terms of the semi-explicit dynamics for the system at hand. Regarding the explicit formulations, the NER modified algorithm with Cholesky decomposition leads to a number of flops comparable with the semi-explicit form. This is because the considered MBS is not too large and the Cholesky decomposition is not critical for the number of joints (i.e., 6 for the benchmark). However, the Order-N method yields a high number of operations, that is more than 4 times the number of operations for the implicit form. It may seem strange as the number of operations for the Order-N method is supposed to increase linearly with the size of the system. The problem is that this linear behavior becomes only interesting for a greater number of joints in the MBS (i.e., around 15 according to [8]).

While the explicit expressions clearly differ as they are derived differently, the implicit and the semi-explicit equations are exactly the same but expressed in different forms (see Sects. 3.3.2.1 and 3.3.2.2). As these equations are identical, their derivative should also be the same. As a matter of fact, the implicit and semi-explicit

formulations lead to the exact same number of non-zero terms in the expression of the Jacobian and the Hessian for the NLP, which is already a good indicator (see Table 3.3). The fact that the equations are identical should lead to the same iteration path while solving the numerical OCP. This is observed in the case of the smoothed actuation (see Fig. 3.10b) with an almost exact same number of iterations in the 231 tasks. For the case of non-smoothed actuation (see Fig. 3.10a), a similar but non-identical number of iterations is observed for the two formulations. These differences are attributed to the high numerical sensitivity of this Optimal Control Problem. This could cause the NLP solver to use different strategies for the two dynamic formulations. Finally, the implicit and semi-explicit formulations are more frequently trapped in higher local minima than the explicit ones. Some additional investigations would be necessary to explain this observation.

We are aware that treating the explicit dynamics with a collocation scheme may not be the best choice and that a multiple shooting could be more appropriated [6]. As a matter of fact, multiple shooting for the explicit formulations have also been investigated in this work but was abandoned as it led to lower performances in terms of robustness (i.e., lower success rate for the extreme point and slower convergence). The reason could be the high dynamics of the system and the strong terminal equilibrium constraint. In order to verify this assumption, a simpler multibody system could be studied with a smoother formulation of the OCP (i.e., with no terminal constraint).

3.5 Conclusion

This paper analyzes different formulations of the multibody dynamics and compare their performances in terms of convergence, accuracy and computation cost in the context of Optimal Control. The aim is to provide some insights on existing MBS formulations and their relevance applicability in the formulation and solution of Optimal Control Problems. Among the different modeling techniques, we focus on the derivation of the equations, using a relative coordinates approach, with the Newton-Euler Recursive scheme and the Order-N algorithm.

First, a framework for the formulation of Optimal Control Problem has been introduced. It was based on a semi-explicit DAE representation of the dynamics well suited for the treatment of multibody systems. Other formulations could have been considered (e.g., full implicit) but would have required specific integrators (i.e., DAE integrators). Regarding the discretization of the OCP, the direct collocation scheme is selected here. This decision is taken following the considerations found in [6], but also because multiple shooting methods are less robust. Then, the different formulations for the multibody dynamics are introduced and their derivations are discussed. From the Newton-Euler recursive scheme, three formulations are derived: the implicit, the semi-explicit and an explicit one. Then, a second explicit formulation is derived from the Order-N algorithm.

Finally, a case study is presented in a benchmark approach for the unconstrained MBS Optimal Control. The system consists in a 3D robot arm with an inverted pendulum attached to the end effector. An OCP is formulated with the goal to reach a point in the 3D space for the end effector after a preset time of 2 s. At the end, the system has to be at equilibrium with zero acceleration and the actuation is minimized on the overall time interval. This task is realized for a grid of target points in the 3D space and for two different definitions of the cost function resulting in two sets of tasks.

Results showed that the implicit and semi-explicit formulations are generally performing better in terms of CPU time and number of iterations. This can be explained by a more compact formulation of the dynamics which leads to a lower CPU cost per iteration. However, these two formulations appear to be more sensitive to higher local minima when the dynamics is stronger (i.e., in the case of non-smoothed actuation). This could be a drawback in applications such as offline trajectory planning where the quality of the solution is more important than its computation cost. On the other hand, the explicit formulations show larger CPU time to convergence but ensure a higher quality of the solution, as a lesser number of high local minima are observed. It seems that the Order-N explicit form works better with a DAE representation, while a pure ODE seems more appropriate for the NER explicit form. For the first set of solutions, with non-smoothed actuation, the differences in the number of iterations for the implicit and semi-explicit formulations are attributed to the strong numerical sensitivity of the problem. More investigation should be carried out on that subject. Nevertheless, this study shows the importance of the dynamic formulations on the OCP solving performances for unconstrained MBS.

There are many perspectives to extend this work. First, it would be appropriate to carry out additional simulations with different OCP formulations and different multibody systems in order to verify the observed tendencies. It has been shown that the way the dynamics is derived has a huge impact on the solver for the OCP as it strongly influences the CPU time per iteration. A recent release of Robotran offers symbolic tools for obtaining a fully analytical sensitivity of MBS equations based on the recursive differentiation of the multibody formulation. It will be interesting to see whether this could lead to better performances than when using the algorithmic differentiation provided by an optimization software like CasADi.

Acknowledgements We would like to thank Dr. Joris Gillis from KUL for his help in the formulation and solving of the Optimal Control Problems with CasADi. Quentin Docquier is a FRIA Grant Holder of the Fonds de la Recherche Scientifique – FNRS, Belgium.

References

1. Andersson, J., Gillis, J., Horn, G., Rawings, J.B., Diehl, M.: CasADi – software framework for nonlinear optimization and optimal control. In: *Mathematical Programming Computation* (2018)
2. Bastos, G., Seifried, R., Bruls, O.: Inverse dynamics of serial and parallel underactuated multibody system using a DAE optimal control approach. *Multibody Sys. Dyn.* **30**, 359–376 (2013)

3. Baumgarte, J.: Stabilization of constraints and integrals of motion. *Comput. Methods Appl. Mech. Eng.* **1**, 1–16 (1972)
4. Betts, J.T.: *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming*, 2nd edn. SIAM, Philadelphia (2010)
5. Björkenstam, S., Carlson, J.S., Lennartson, B.: Exploiting sparsity in the discrete mechanics and optimal control method with application to human motion planning. In: *Proceedings of 11th IEEE International Conference on Automation Science and Engineering*, Gothenburg, pp. 769–774 (2015)
6. Diehl, M., Bock, H., Diedam, H., Wieber, P.B.: Fast direct multiple shooting algorithms for optimal robot control. In: Diehl M., Mombaur, K. (eds.) *Fast Motions in Biomechanics and Robotics*. Lecture Notes in Control and Information Sciences, vol. 340. Springer, Berlin/New York (2006)
7. Docquier, N., Poncelet, A., Fisettes, P.: Robotran: a powerful symbolic generator of multibody models. *Mech. Sci.* **4**, 199–219 (2013)
8. Fisettes, P.: *Génération Symbolique des Equations du Mouvement de Systèmes Multicorps et Application dans le Domaine Ferroviaire*. Ph.D. thesis, Université catholique de Louvain, Louvain-la-Neuve, Belgium (1994)
9. Fisettes, P., Samin, J.C.: Symbolic generation of large multibody system dynamic equations using a new semi-explicit Newton/Euler recursive scheme. *Arch. Appl. Mech.* **66**, 187–199 (1996)
10. Griewank, A., Walther, A.: *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. Other Titles in Applied Mathematics, vol. 105. SIAM, Philadelphia (2008)
11. Gros, S., Zanon, M., Diehl, M.: Nonlinear MPC and MHE for mechanical multi-body systems with application to fast tethered airplanes. In: *IFAC Conference on Nonlinear Model Predictive Control*, Noordwijkerhout, pp. 86–93 (2012)
12. Iwamura, M., Eberhard, P., Schiehlen W., Seifried R.: A general purpose algorithm for optimal trajectory planning of closed loop multibody systems. In: Arczewski, K. (ed.) *Multibody Dynamics: Computational Methods and Applications*. Computational Methods in Applied Sciences, vol. 23. Springer, Dordrecht (2011)
13. Luh, J.-Y.-S., Walker, N.-W., Paul, R.-P.-C.: On-line computational scheme for mechanical manipulators. *J. Dyn. Syst. Meas. Control* **102**, 69–76 (1980)
14. Manara, S., Gabiccini, M., Artoni, A., Diehl, M.: On the integration of singularity-free representations of SO(3) for direct optimal control. *Nonlinear Dyn.* **90**, 1223–1241 (2017)
15. Samin, J.C., Fisettes, P.: *Symbolic Modeling of Multibody Systems*. Kluwer, Dordrecht (2003)
16. Schwertassek, R., Rulka, W.: Aspects of efficient and reliable multibody systems simulation. In: Haug, E.J., Deyo, R.C. (eds.) *Real-Time Integration Methods for Mechanical System Simulation*. NATO ASI Series F, vol. 69, pp. 55–96. Springer, Berlin/Heidelberg (1989)
17. Siebert, R., Betsch, P.: A Hamiltonian conserving indirect optimal control method for multibody dynamics. In: *Proceedings in Applied Mathematics and Mechanics*, vol. 12. Wiley-VCH Verlag GmbH & Co. KGaA, Weinheim (2012)
18. Verscheure, D., Demeulenaere, B., Swevers, J., De Schutter, J., Diehl M.: Time-energy optimal path tracking for robots: a numerically efficient optimization approach. In: *10th IEEE International Workshop on Advanced Motion Control*, Trento, pp. 727–732 (2008)
19. von Stryk, O., Schlemmer, M.: Optimal control of the industrial robot manutec R3. In: Bulirsch, R. (ed.) *Computational Optimal Control*. ISNM International Series of Numerical Mathematics, vol. 115. Birkhäuser, Basel/Boston (1994)
20. Wächter, A.: *An interior point algorithm for large-scale nonlinear optimization with applications in process engineering*, Ph.D. thesis, Carnegie Mellon University (2002)
21. Wehage, R.-A., Haug, E.-J.: Generalized coordinate partitioning for dimension reduction in analysis of constrained dynamic systems. *J. Mech. Des.* **134**, 247–255 (1982)