Computers and Structures 308 (2025) 107642

Contents lists available at ScienceDirect

Computers and Structures

journal homepage: www.elsevier.com/locate/compstruc

An analytical approach to the sensitivity analysis of semi-recursive ODE formulations for multibody dynamics

Álvaro López Varela [a,b,*], Daniel Dopico Dopico [a], Alberto Luaces Fernández [a]

[a] Laboratorio de Ingeniería Mecánica, Campus Industrial de Ferrol, CITENI, Universidade da Coruña, Mendizábal s/n, 15403, Ferrol, Spain
[b] Centro Mixto de Investigación Navantia-UDC, Campus Industrial de Ferrol, Universidade da Coruña, Batallones s/n, 15403, Ferrol, Spain

ARTICLE INFO

Keywords:
Multibody dynamics
Semi-recursive
Sensitivity analysis
Direct differentiation method
Adjoint variable method
Matrix R

ABSTRACT

Sensitivity analysis is an extremely powerful tool in many applications such as in the optimization of the dynamics of multibody systems with gradient-based methods. Sensitivity calculations are computationally burdensome and, depending on the method chosen for differentiation and the set of dynamic equations, they could result highly inefficient. Semi-recursive dynamic methods are seldom studied analytically in terms of sensitivity analysis due to their complexity, even though their dynamic performance is usually among the most efficient.

This work explores the sensitivity analysis of a particular multibody-dynamics formulation, the semi-recursive Matrix R formulation, which is based on the nullspace of constraint equations and leads to a system of ordinary differential equations. As a result, two sets of sensitivity equations are proposed, one based on the direct differentiation method (DDM) and other on the Adjoint Variable Method (AVM), being these sensitivity formulations the main novelty of this work. The main derivatives required in the sensitivity equations are listed in this document, paying special attention to conciseness, correctness and completeness. The methods proposed have been implemented in the general purpose multibody library MBSLIM (*Multibody Systems in Laboratorio de Ingeniería Mecánica*), and their performance has been tested in two numerical experiments, a five-bar benchmark problem and a four-wheeled buggy vehicle.

A review and generalization of constrained and unconstrained kinematic problems in relative coordinates is provided as an introduction to the generation of the semi-recursive Matrix R equations of motion. Due to the importance of the selection of the set of independent coordinates, a more general description of the Matrix R method is presented as a novel contribution as well.

1. Introduction

During the last few decades, the knowledge area which studies the dynamics of systems composed of multiple bodies, commonly known as multibody dynamics (MBD), has expanded its boundaries from pure direct and inverse dynamic problems to new topics related to design, optimization or control. In all these problems, it is essential to have a measure of the variation of the behavior of a model with respect to a set of parameters. The sensitivity analysis is the tool that permits to calculate these variations, but it involves a higher level of complexity than pure dynamic problems.

The quest for the most efficient method for the evaluation of the sensitivity analysis of a multibody system dynamics is still open nowadays since its performance is related to: the differentiation method, the sensitivity equations, the multibody model and, specially, the dynamic formulation considered. Dynamic recursive methods which exploit the topology of the multibody system usually display the best performance in terms of efficiency [1], but they are rarely considered for a sensitivity analysis since they imply much higher complexity than global methods. Moreover, constraint enforcement schemes like Matrix R [2], based on a projection from dependent to independent coordinates, are rarely addressed with analytical differentiation methods due to the complexity of the resulting derivatives and to the assumption that analytical methods are “time-consuming and error-prone” [3]. This work proves that a purely analytical sensitivity analysis of a semi-recursive matrix R formulation is viable, general for any multibody system and can be highly efficient.

* Corresponding author.
E-mail addresses: alvaro.lopez1@udc.es (Á. López Varela), ddopico@udc.es (D. Dopico Dopico), alberto.luaces@udc.es (A. Luaces Fernández).

https://doi.org/10.1016/j.compstruc.2024.107642
Received 19 June 2024; Accepted 30 December 2024

Available online 10 January 2025
0045-7949/© 2025 The Author(s). Published by Elsevier Ltd. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

Classical Matrix R formulations involve a selection of independent coordinates from the dependent coordinates vector. In some closed-chain multibody models, coordinates of points might represent better the degrees of freedom of a multibody system than relative coordinates. For that reason, another contribution of this work is the extension of the dynamic semi-recursive Matrix R formulation to support any relative or natural coordinates (Cartesian coordinates of points or vectors) as independent coordinates, and the incorporation of this feature in the sensitivity formulations developed.

The sensitivity analysis of the semi-recursive Matrix R formulation has been analytically studied before by Gutiérrez et al. in [4]. This seminal work presents a direct sensitivity method and lists the set of derivatives needed in the direct differentiation of the semi-recursive Matrix R formulation. Additionally, it provides a comparison between direct (analytical), automatic and numerical differentiation methods, but the comparison at efficiency level is hindered by the use of different languages (C and MATLAB). This work represents the first effort to compute analytical semi-recursive Matrix R sensitivities in a general form, but it has two main limitations: first, it only allows relative coordinates as independent coordinates (here, coordinates of points and vectors are also allowed); and second, it assumes that the Jacobian of the constraints is invertible (the generalization conveys significant modifications). The present work addresses these two limitations, leading to more general sensitivity formulations, and completes the study with the analytical semi-recursive adjoint method. Moreover, in [4] the dynamic and sensitivity equations are derived considering one particular reference point for the recursive evaluation of kinematic, dynamic and sensitivity terms (it uses the point that "instantaneously coincides with the origin of the inertial reference frame" [4]). In the current developments, an arbitrary reference point is considered, leading to a more general formulation of the equations of motion and their sensitivities.

There are several differentiation techniques which can be used to obtain the sensitivity analysis of a system. The simplest differentiation method consists in numerical differentiation (ND), which is based on the numerical perturbation of the parameters in order to approximate the derivative of an objective function by means of its variation using a finite difference scheme or similar. This technique suffers from two issues: first, it is extremely dependent on the magnitude of the perturbations used which yields the so called "step-size dilemma" [5]; and second, the computational effort is directly proportional to the number of parameters. Despite its problems, this method has been used combined with semi-analytical differentiation in works such as [6] or [7], and it is commonly used as test method for analytical implementations.

A second option is the automatic differentiation method (AD) (also called algorithmic differentiation method). Automatic differentiation is based on the decomposition of complex computations into elemental operations with known direct analytical expressions for their derivatives, and it offers high accuracy with low computational expense. Even though its implementation is substantially more complex than numerical methods, it is a suitable method for obtaining accurate derivatives with a low programming effort [8]. There are multiple examples of works in which AD libraries have been successfully used in the sensitivity analysis of the dynamics of multibody systems. Dürrbaum et al. compared the performance of ADOL-C (*Automatic Differentiation by OverLoading in C++*) against a symbolic differentiation software in [9], giving as result a better performance of the symbolic program. Ambrosio et al. applied ADIFOR (*Automatic DIfferentiation of FORtran*) for the optimization of flexible MBS in [10]. Callejo and collaborators explored the automatic differentiation for the dynamics of MBS in different works [8,3,11], one of which ([3]) represents the most recent attempt to achieve the sensitivity analysis of a semi-recursive Matrix R formulation. The list of works comprising automatic differentiation is steadily growing, specially in those problems where analytical differentiation is unmanageable.

The third possibility consists in the analytical differentiation of the dynamic equations of motion. In general, analytical approaches are usually the fastest and most accurate, but they involve an important theoretical and implementation effort. Despite their complexity, new studies devoted to the analytical sensitivity analysis of multibody system dynamics are published every year since the appearance of the first works in the field [12–16].

The sensitivity analysis of a set of equations can be accomplished following two different approaches, which are the direct differentiation method (DDM) and the adjoint variable method (AVM). The application of the DDM to the dynamic equations of motion of a multibody system delivers a set of systems of equations in which the unknowns are the sensitivities of the variables of the original dynamic problem. The simplicity of the DDM has made many authors to resort to this method, which converts it in the most spread option in the multibody community. The AVM, on the contrary, reformulates the sensitivity analysis adding a new set of variables, namely the adjoint variables. Thanks to this transformation, the number of systems of equations is independent of the number of parameters, which makes it the ideal method for highly parameterized sensitivity problems.

Sensitivity methods can be classified as well according to the order between differentiation and discretization [17]. In *differentiate-then-discretize* approaches, the continuous equations of motion are firstly differentiated and, secondly, they are discretized in time steps to be solved. This option delivers a numerical approximation of the sensitivities of the continuous problem, and leads to a set of differential equations independent of the numerical integrator chosen in the dynamics. The *discretize-then-differentiate* approach studies the sensitivity analysis of the discretized dynamics, thus yielding the exact derivatives of the numerical approximation of the dynamics. The second approach is expressed in terms of algebraic equations that are specific for the numerical integrator used to solve the dynamics. Despite their conceptual differences, both approaches converge to the same results as the time step is decreased. In this work, the *differentiate-then-discretize* approach is considered in both direct and adjoint methods due to its conciseness, generality and to the fact that, in practical applications, the differences between both approaches are almost negligible.

The computational effort dedicated to sensitivity analysis is related to the multibody model that describe the mechanism in study, as well as to the formulation used to obtain the dynamic response. Models based on relative coordinates usually lead to the fastest dynamic simulations, although they involve more complex computations than other models [18]. In this work, we study if this increase of efficiency is also present in sensitivity problems.

Among the most efficient methods to handle MBS are the semi-recursive methods, in which any multibody system can be described as an open-loop system subjected (or not) to a set of kinematic constraints [19,20]. In this method, dynamic terms are composed recursively, but the equations of motion are solved globally. Thanks to these recursive and global procedures, it is possible to combine relative coordinate modeling with multiple constraint enforcement schemes typical of global systems. Semi-recursive methods have been combined with the Matrix R formulation in [21–23] and more recently in [24], with the penalty approach in [25] or with the Augmented Lagrangian index-3 formulation with velocity and acceleration projections (ALI3-P) in [26,27,18,28].

In this study, the semi-recursive Matrix R formulation in relative coordinates is revisited and extended to support natural coordinates [2] as degrees of freedom as well. However, the main novelty of this paper is the development of direct and adjoint semi-recursive Matrix R sensitivity analysis formulations using an analytical differentiation method. Both dynamic and sensitivity formulations have been implemented in the general purpose multibody library MBSLIM [29], which can solve kinematic, dynamic, sensitivity analysis and optimization problems of rigid multibody systems, and which is being currently extended to support flexible bodies.

This work is structured as follows: section 2 presents the kinematics of relative coordinates and introduces the forward dynamic semi-recursive Matrix R formulation; section 3 covers the sensitivity analysis of the semi-recursive Matrix R formulation by means of both the direct differentiation and the adjoint variable methods; the dynamic and sensitivity formulations presented are tested in different numerical experiments in section 4; finally, section 5 gathers the main conclusions of this work.

2. Dynamic formulation

2.1. Kinematics of open-loop systems

Relative coordinate modeling usually constitutes the most natural and efficient method to describe the kinematics of a multibody system. However, despite the reduced number of coordinates, the generation of the equations of motion is not direct and other set of intermediate coordinates (as Cartesian coordinates) is needed.

The use of these intermediate coordinates encompasses a double kinematic problem that has to be addressed both in kinematic and dynamic analyses: first, it is necessary to calculate positions, velocities and accelerations of the intermediate Cartesian coordinates from the relative coordinates in order to generate the EoM; second, a coordinate transformation takes the equations from Cartesian to relative coordinates, which is carried out by means of recursive kinematic relations based on the topology of the system.

According to [18], the recursive kinematic relations can be summarized in the following set of equations, valid for any type of joint[1]:

$$\mathbf{V}_i = \mathbf{B}_i^v \mathbf{V}_{i-1} + \mathbf{b}_i^v \dot{\mathbf{z}}_i \tag{1a}$$

$$\dot{\mathbf{V}}_i = \mathbf{B}_i^v \dot{\mathbf{V}}_{i-1} + \mathbf{b}_i^v \ddot{\mathbf{z}}_i + \mathbf{d}_i^v \tag{1b}$$

$$\mathbf{B}_i^v = \begin{bmatrix} \mathbf{I} & \tilde{\mathbf{r}}_{i-1} - \tilde{\mathbf{r}}_i \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \tag{1c}$$

$$\dot{\mathbf{B}}_i^v = \begin{bmatrix} \mathbf{0} & \dot{\tilde{\mathbf{r}}}_{i-1} - \dot{\tilde{\mathbf{r}}}_i \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \tag{1d}$$

$$\mathbf{d}_i^v = \dot{\mathbf{B}}_i^v \mathbf{V}_{i-1} + \dot{\mathbf{b}}_i^v \dot{\mathbf{z}}_i \tag{1e}$$

with $\mathbf{V}_i = \begin{bmatrix} \dot{\mathbf{r}}_i^{\mathrm{T}} & \boldsymbol{\omega}_i^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}} \in \mathbb{R}^6$ and $\mathbf{z}_i$ identifying the set of relative coordinates of joint i. Remaining terms $\mathbf{b}_i^v$, $\dot{\mathbf{b}}_i^v$ and $\mathbf{d}_i^v$ are kinematic expressions related to each joint and body.

Observe that (1) allows a general and simple implementation, with a common structure for every joint type and with only one particular term required for each type of joint ($\mathbf{b}_i^v$) and its time derivative ($\dot{\mathbf{b}}_i^v$), whose expressions involve a reduced set of arithmetic operations with the entities that define the joint [18].

It is possible to reformulate (1) relating a vector $\mathbf{V} = \begin{bmatrix} \mathbf{V}_1^{\mathrm{T}} & \mathbf{V}_2^{\mathrm{T}} & \ldots \mathbf{V}_{n_b}^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}} \in \mathbb{R}^{6n_b}$ containing the linear and angular velocities of all the bodies of the multibody system with the vector of relative coordinates $\mathbf{z} = \begin{bmatrix} \mathbf{z}_1{}^{\mathrm{T}} & \mathbf{z}_2{}^{\mathrm{T}} & \ldots & \mathbf{z}_{n_b}{}^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}} \in \mathbb{R}^n$, with n_b the number of bodies and n the number of relative coordinates, as:

$$\mathbf{V} = \mathbf{R}^v \dot{\mathbf{z}} \tag{2}$$

$$\mathbf{V}^* = \mathbf{R}^v \dot{\mathbf{z}}^* \tag{3}$$

$$\dot{\mathbf{V}} = \mathbf{R}^v \ddot{\mathbf{z}} + \dot{\mathbf{R}}^v \dot{\mathbf{z}} \tag{4}$$

with the superscript $()^*$ denoting virtual velocities.[2] The expressions (2)-(4) allow a more compact notation that can be exploited in the dynamics for the generation of concise mass matrix and generalized force vector expressions. However, at implementation level, it is desirable not to build the matrices $\mathbf{R}^v$ and $\dot{\mathbf{R}}^v$ but to compute the kinematics recursively by means of (1).

It should be remarked that every closed-loop system can be expressed as an open-loop system subjected to a set of kinematic constraints, thus equations (1), (2), (3) and (4) are general to any MBS.

2.2. Kinematics for non-minimal relative coordinates

Almost any practical multibody system modeled in natural (or reference point) coordinates requires a set of constraint equations to be fully defined.[3] On the contrary, open-loop systems modeled with minimal relative coordinates do not need constraints. However, the presence of kinematic constraints is common in relative coordinate modeling for the imposition of loop-closure constraints or for complementing the definition of the multibody model.

In the following lines, the three main kinematic problems are briefly described (the finite displacements problem is assimilated to the position problem). The three constrained kinematic problems have been implemented in the MBSLIM multibody library as general kinematic formulations.

It is worth mentioning that only two out of these three kinematic problems are needed for the dynamics: initial position (subsection 2.2.1) and velocity (subsection 2.2.4) analyses are needed for obtaining the initial conditions of the dynamics DAE system.

2.2.1. Position kinematic analysis

The objective of the position kinematic analysis is to nullify the vector of m constraint equations $\boldsymbol{\Phi} \in \mathbb{R}^m$ for given values of the d degrees of freedom, $\mathbf{z}^i \in \mathbb{R}^d$, chosen as independent coordinates:

[1] These generic expressions represent the generalization of classical approaches like [14,25] to an arbitrary reference point.
[2] $\mathbf{V}_i^*$ and $\dot{\mathbf{z}}_i^*$ are also related by equation (1a).
[3] One single rigid body in natural coordinates or one single joint in reference point coordinates need constraints.

$$\mathbf{\Phi}\left(\mathbf{q},\mathbf{z},\boldsymbol{\rho},t\right)=\mathbf{0} \tag{5}$$

where t is the time variable, $\boldsymbol{\rho}$ a set of constant parameters and $\mathbf{q}$, $\mathbf{z}$ have been defined before. Expression (5) represents a system of nonlinear equations, which in general cannot be solved analytically due to the complexity of the equations, and has to be computed numerically. One of the most resorted approaches to solve these type of equations is the Newton-Raphson method, which exploits the expansion of (5) in a Taylor series around a given initial approximated solution $\mathbf{z}^0$:

$$\mathbf{\Phi}\left(\mathbf{q}\left(\mathbf{z}\right),\mathbf{z},\boldsymbol{\rho},t\right)\cong\mathbf{\Phi}\left(\mathbf{q}^0,\mathbf{z}^0,\boldsymbol{\rho},t\right)+\mathbf{\Phi}_{\hat{\mathbf{z}}}\left(\mathbf{q}^0,\mathbf{z}^0\boldsymbol{\rho},t\right)\left(\mathbf{z}-\mathbf{z}^0\right)=\mathbf{0} \tag{6}$$

where $\mathbf{\Phi}_{\hat{\mathbf{z}}}\in\mathbb{R}^{m\times n}$ is the Jacobian matrix of the constraint equations vector:

$$\mathbf{\Phi}_{\hat{\mathbf{z}}}=\frac{\partial\mathbf{\Phi}}{\partial\mathbf{q}}\frac{\partial\mathbf{q}}{\partial\mathbf{z}}+\frac{\partial\mathbf{\Phi}}{\partial\mathbf{z}}=\mathbf{\Phi}_{\mathbf{q}}\mathbf{q}_{\mathbf{z}}+\mathbf{\Phi}_{\mathbf{z}} \tag{7}$$

From (7) it can be observed that $\mathbf{\Phi}_{\hat{\mathbf{z}}}$ is not exactly a partial derivative, and for this reason the subscript $\hat{\mathbf{z}}$ is used, according to the notation (A.1).

Equation (6) can be reformulated as:

$$\mathbf{\Phi}_{\hat{\mathbf{z}}}\left(\mathbf{q}^{\{j\}},\mathbf{z}^{\{j\}},t\right)\Delta\mathbf{z}^{\{j+1\}}=-\mathbf{\Phi}\left(\mathbf{q}^{\{j\}},\mathbf{z}^{\{j\}},t\right) \tag{8}$$

being j and $j+1$ the iteration numbers and $\Delta\mathbf{z}^{\{j+1\}}=\mathbf{z}^{\{j+1\}}-\mathbf{z}^{\{j\}}$ the increment in the joint coordinates for the present iteration.

Equation (8) cannot be uniquely solved because it is rank deficient in the number d of degrees of freedom of the system, since no reference has been made to the imposition of the values of the degrees of freedom yet. The classical technique and a novel technique for imposing degrees of freedom are described in subsections 2.2.2 and 2.2.3.

2.2.2. Imposition of degrees of freedom: relative coordinates as DoF

The classical technique for imposing degrees of freedom is described in [2] and assumes that the degrees of freedom are a subset of the full set of dependent coordinates, $\mathbf{z}^i\in\mathbb{R}^d\subseteq\mathbf{z}$. Thus, a constant matrix $\mathbf{B}\in\mathbb{R}^{d\times n}$ composed of "1"s and "0"s can be defined.[4] This matrix is constant in time, straightforward to calculate with a given set of independent coordinates, and satisfies the following relations:

$$\mathbf{z}^i=\mathbf{B}\mathbf{z} \tag{9a}$$

$$\dot{\mathbf{z}}^i=\mathbf{B}\dot{\mathbf{z}} \tag{9b}$$

$$\ddot{\mathbf{z}}^i=\mathbf{B}\ddot{\mathbf{z}} \tag{9c}$$

Equations (5) can be completed with equations (9a) and thus (8) becomes a full-column-rank system of equations:

$$\mathbf{\Theta}\left(\mathbf{q}\left(\mathbf{z}\right),\mathbf{z},\mathbf{z}^i,\boldsymbol{\rho},t\right)=\begin{bmatrix}\mathbf{\Phi}\\ \mathbf{B}\mathbf{z}-\mathbf{z}^i\end{bmatrix}=\mathbf{0}\Rightarrow\mathbf{\Theta}_{\hat{\mathbf{z}}}^{\{j\}}\Delta\mathbf{z}^{\{j+1\}}=-\mathbf{\Theta}\left(\mathbf{z}^{\{j\}},\mathbf{z}^i,\boldsymbol{\rho},t\right)\Rightarrow\begin{bmatrix}\mathbf{\Phi}_{\hat{\mathbf{z}}}^{\{j\}}\\ \mathbf{B}\end{bmatrix}\Delta\mathbf{z}^{\{j+1\}}=\begin{bmatrix}-\mathbf{\Phi}^{\{j\}}\\ \mathbf{z}^i-\mathbf{B}\mathbf{z}^{\{j\}}\end{bmatrix} \tag{10}$$

Equations (10) can be further simplified by starting the iterations with an initial guess matching the desired degrees of freedom, $\mathbf{B}\mathbf{z}^{\{0\}}=\mathbf{z}^i$ and it can be easily proved, by induction, that $\mathbf{B}\mathbf{z}^{\{j\}}=\mathbf{z}^i\,\forall j$ resulting in the simplified system:

$$\begin{bmatrix}\mathbf{\Phi}_{\hat{\mathbf{z}}}^{\{j\}}\\ \mathbf{B}\end{bmatrix}\Delta\mathbf{z}^{\{j+1\}}=\begin{bmatrix}-\mathbf{\Phi}^{\{j\}}\\ \mathbf{0}\end{bmatrix} \tag{11}$$

2.2.3. Imposition of degrees of freedom: natural coordinates as DoF

In closed-loop systems, the degrees of freedom of a mechanism can sometimes be identified more directly with a set of Cartesian coordinates rather than joint coordinates. In addition, supporting natural coordinates as DoF is a requirement in the definition of MBSLIM models. Since natural coordinates are not in the set of relative coordinates, the degrees of freedom are not a subset of the full set of dependent coordinates, $\mathbf{z}^i\subseteq\mathbf{q},\mathbf{q}\cap\mathbf{z}=\emptyset\Rightarrow\mathbf{z}^i\nsubseteq\mathbf{z}$, and therefore equations (9) do not hold since $\mathbf{B}$ matrix cannot be defined like in section 2.2.2.

Taking into account that coordinates $\mathbf{z}$ define completely the kinematics of the system in positions, any other kinematic magnitude can be expressed as a function of them [5]:

$$\mathbf{z}^i=\mathbf{h}(\mathbf{z},\boldsymbol{\rho}) \tag{12}$$

Equations (5) can be completed with equations (12) forming a nonlinear system of equations. The application of the Newton-Raphson iteration to the system, converts (8) into a full-column-rank system of equations:

$$\mathbf{\Theta}\left(\mathbf{q}\left(\mathbf{z}\right),\mathbf{z},\mathbf{z}^i,\boldsymbol{\rho},t\right)=\begin{bmatrix}\mathbf{\Phi}\\ \mathbf{h}(\mathbf{z},\boldsymbol{\rho})-\mathbf{z}^i\end{bmatrix}=\mathbf{0}\Rightarrow\mathbf{\Theta}_{\hat{\mathbf{z}}}^{\{j\}}\Delta\mathbf{z}^{\{j+1\}}=-\mathbf{\Theta}\left(\mathbf{z}^{\{j\}},\mathbf{z}^i,\boldsymbol{\rho},t\right)\Rightarrow\begin{bmatrix}\mathbf{\Phi}_{\hat{\mathbf{z}}}^{\{j\}}\\ \mathbf{B}^{\{j\}}\end{bmatrix}\Delta\mathbf{z}^{\{j+1\}}=\begin{bmatrix}-\mathbf{\Phi}^{\{j\}}\\ \mathbf{z}^i-\mathbf{h}^{\{j\}}\end{bmatrix} \tag{13}$$

with $\mathbf{\Theta}_{\hat{\mathbf{z}}}=\begin{bmatrix}\mathbf{\Phi}_{\hat{\mathbf{z}}}\\ \mathbf{B}\end{bmatrix}$ the Jacobian matrix of $\mathbf{\Theta}$ and $\mathbf{B}\equiv\mathbf{h}_{\mathbf{z}}$ the Jacobian matrix of $\mathbf{h}$.

Observe that equations (13) are equivalent to (10) but they make it possible to use degrees of freedom which are not in the relative coordinates vector, $\mathbf{z}$, but at the prize of a non-constant and more complex matrix $\mathbf{B}$. If, otherwise, the degrees of freedom are selected from the relative coordinates vector, equations (10) and the simplified $\mathbf{B}$ from section 2.2.2, arise again.

[4] Please do not confuse with $\mathbf{B}_i^v$, an elemental term of the recursive accumulation.

[5] Even in the unusual case of an explicit function difficult to obtain, the implicit function theorem can be used, arriving at the same conclusion.

Concerning velocities and accelerations, equations (9b) are still valid with the new definition of $\mathbf{B}$, but (9c) needs to be modified:

$$\dot{\mathbf{z}}^i = \mathbf{h}_z \dot{\mathbf{z}} = \mathbf{B}\dot{\mathbf{z}} \tag{14}$$

$$\ddot{\mathbf{z}}^i = \mathbf{B}\ddot{\mathbf{z}} + \dot{\mathbf{B}}\dot{\mathbf{z}} \tag{15}$$

With this approach, any type of coordinate can be used as degree of freedom, as long as the matrix $\boldsymbol{\Theta}_{\hat{\mathbf{z}}}$ has a left inverse, $\boldsymbol{\Theta}_{\hat{\mathbf{z}}}^{+}$, which is guaranteed to exist if a proper selection of degrees of freedom is made:

$$\boldsymbol{\Theta}_{\hat{\mathbf{z}}}^{+}\boldsymbol{\Theta}_{\hat{\mathbf{z}}} = \begin{bmatrix} \mathbf{S}^{\boldsymbol{\Phi}} & \mathbf{R}^{\boldsymbol{\Phi}} \end{bmatrix} \begin{bmatrix} \boldsymbol{\Phi}_{\hat{\mathbf{z}}} \\ \mathbf{B} \end{bmatrix} = \mathbf{I}_n \tag{16}$$

with matrices $\mathbf{R}^{\boldsymbol{\Phi}} \in \mathbb{R}^{n\times d}$ and $\mathbf{S}^{\boldsymbol{\Phi}} \in \mathbb{R}^{n\times m}$ the last d and the first m columns of the inverse, respectively.

For the general case of redundant constraints, matrices $\mathbf{R}^{\boldsymbol{\Phi}}$ and $\mathbf{S}^{\boldsymbol{\Phi}}$ can be calculated by means of a least-squares problem:

$$\left[\boldsymbol{\Phi}_{\hat{\mathbf{z}}}^{\mathrm{T}}\boldsymbol{\Phi}_{\hat{\mathbf{z}}} + \mathbf{B}^{\mathrm{T}}\mathbf{B}\right] \begin{bmatrix} \mathbf{S}^{\boldsymbol{\Phi}} & \mathbf{R}^{\boldsymbol{\Phi}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\Phi}_{\hat{\mathbf{z}}}^{\mathrm{T}} & \mathbf{B}^{\mathrm{T}} \end{bmatrix} \tag{17}$$

The explicit calculation of matrix $\mathbf{S}^{\boldsymbol{\Phi}}$ is usually avoided for kinematic and dynamic analyses but it has an important role in sensitivity analyses.

2.2.4. Velocity kinematic analysis

The velocity problem aims to obtain the set of velocities $\dot{\mathbf{z}}$ such that:

$$\dot{\boldsymbol{\Phi}}(\dot{\mathbf{q}}(\mathbf{z},\dot{\mathbf{z}}),\mathbf{q}(\mathbf{z}),\dot{\mathbf{z}},\mathbf{z},\rho,t) = \boldsymbol{\Phi}_{\hat{\mathbf{z}}}\dot{\mathbf{z}} + \boldsymbol{\Phi}_t = \mathbf{0} \tag{18}$$

Differentiating (5) with respect to time and completing with equations (9b) or (14), velocity equations can be rewritten as:

$$\dot{\boldsymbol{\Theta}}\left(\dot{\mathbf{q}}(\mathbf{z},\dot{\mathbf{z}}),\mathbf{q}(\mathbf{z}),\dot{\mathbf{z}},\dot{\mathbf{z}}^i,\rho,t\right) = \begin{bmatrix} \dot{\boldsymbol{\Phi}} \\ \mathbf{B}\dot{\mathbf{z}} - \dot{\mathbf{z}}^i \end{bmatrix} = \mathbf{0} \Rightarrow \begin{bmatrix} \boldsymbol{\Phi}_{\hat{\mathbf{z}}} \\ \mathbf{B} \end{bmatrix} \dot{\mathbf{z}} = \begin{bmatrix} \mathbf{b} \\ \dot{\mathbf{z}}^i \end{bmatrix} \tag{19}$$

with $\dot{\mathbf{z}}^i$ representing the desired values of the DoF at velocity level, $\dot{\boldsymbol{\Theta}}_{\hat{\mathbf{z}}} = \boldsymbol{\Theta}_{\hat{\mathbf{z}}} = \begin{bmatrix} \boldsymbol{\Phi}_{\hat{\mathbf{z}}} \\ \mathbf{B} \end{bmatrix}$ the leading matrix for velocities and $\mathbf{b} \equiv -\boldsymbol{\Phi}_t$ is introduced in accordance with the classical compact notation of this problem[6], as presented in [2]. The velocity problem is solved by means of a linear system of equations and, unlike the position problem, does not need to be iterated.

2.2.5. Acceleration kinematic analysis

The set of dependent accelerations, $\ddot{\mathbf{z}}$, resulting from the kinematic acceleration problem, have to satisfy the acceleration constraints, which can be found differentiating (5) twice (or (18) once) with respect to time:

$$\ddot{\boldsymbol{\Phi}}(\ddot{\mathbf{q}}(\mathbf{z},\dot{\mathbf{z}},\ddot{\mathbf{z}}),\dot{\mathbf{q}}(\mathbf{z},\dot{\mathbf{z}}),\mathbf{q}(\mathbf{z}),\ddot{\mathbf{z}},\dot{\mathbf{z}},\mathbf{z},\rho,t) = \mathbf{0} \tag{20}$$

Completing with equation (15), a linear system of equations arises,

$$\ddot{\boldsymbol{\Theta}}\left(\ddot{\mathbf{q}}(\mathbf{z},\dot{\mathbf{z}},\ddot{\mathbf{z}}),\dot{\mathbf{q}}(\mathbf{z},\dot{\mathbf{z}}),\mathbf{q}(\mathbf{z}),\ddot{\mathbf{z}},\ddot{\mathbf{z}}^i,\dot{\mathbf{z}},\mathbf{z},\rho,t\right) = \begin{bmatrix} \ddot{\boldsymbol{\Phi}} \\ \mathbf{B}\ddot{\mathbf{z}} + \dot{\mathbf{B}}\dot{\mathbf{z}} - \ddot{\mathbf{z}}^i \end{bmatrix} = \mathbf{0} \Rightarrow \begin{bmatrix} \boldsymbol{\Phi}_{\hat{\mathbf{z}}} \\ \mathbf{B} \end{bmatrix} \ddot{\mathbf{z}} = \begin{bmatrix} \mathbf{c} \\ \ddot{\mathbf{z}}^i - \dot{\mathbf{B}}\dot{\mathbf{z}} \end{bmatrix} \tag{21}$$

with $\ddot{\mathbf{z}}^i$ the desired values of the DoF accelerations, $\ddot{\boldsymbol{\Theta}}_{\hat{\mathbf{z}}} = \boldsymbol{\Theta}_{\hat{\mathbf{z}}} = \begin{bmatrix} \boldsymbol{\Phi}_{\hat{\mathbf{z}}} \\ \mathbf{B} \end{bmatrix}$ the leading matrix for accelerations and $\mathbf{c} \equiv -\dot{\boldsymbol{\Phi}}_t - \dot{\boldsymbol{\Phi}}_{\hat{\mathbf{z}}}\dot{\mathbf{z}}$. In the acceleration problem, the new term $\dot{\boldsymbol{\Phi}}_{\hat{\mathbf{z}}}$ appears:

$$\dot{\boldsymbol{\Phi}}_{\hat{\mathbf{z}}} = \frac{\mathrm{d}\boldsymbol{\Phi}_{\hat{\mathbf{z}}}}{\mathrm{d}t} = \dot{\boldsymbol{\Phi}}_{\mathbf{q}}\mathbf{q}_{\mathbf{z}} + \boldsymbol{\Phi}_{\mathbf{q}}\dot{\mathbf{q}}_{\mathbf{z}} + \dot{\boldsymbol{\Phi}}_{\mathbf{z}} \tag{22}$$

Thus, $\mathbf{c}$ can be rewritten as:

$$\mathbf{c} = -\left(\dot{\boldsymbol{\Phi}}_{\mathbf{q}}\mathbf{q}_{\mathbf{z}} + \boldsymbol{\Phi}_{\mathbf{q}}\dot{\mathbf{q}}_{\mathbf{z}} + \dot{\boldsymbol{\Phi}}_{\mathbf{z}}\right)\dot{\mathbf{z}} - \dot{\boldsymbol{\Phi}}_t \tag{23}$$

In general, $\dot{\boldsymbol{\Phi}}$ is expressed in terms of positions and velocities of points, vectors, angles, distances and/or the time, and consequently, the assessment of the derivatives $\dot{\boldsymbol{\Phi}}_{\mathbf{q}}$, $\dot{\boldsymbol{\Phi}}_{\mathbf{z}}$ and $\dot{\boldsymbol{\Phi}}_t$ (all with respect to the explicit dependencies of each constraint) are straightforward to obtain. On the contrary, the derivatives of the natural coordinates $\dot{\mathbf{q}}_{\mathbf{z}}$ and $\mathbf{q}_{\mathbf{z}}$ depend on the topology of the mechanism and their evaluation is more challenging.

2.3. Semi-recursive Matrix R formulation

The formulation presented in this section was originally introduced by García de Jalón and Bayo in [30], based on previous results derived in [31]. A modern description with the notation used in this work can be found in [2] and later, in [21], [22] and more recently in [32] the formulation was applied to semi-recursive methods. As long as the method is profusely described in those works, only the main structure of the formulation is outlined. A new notation is employed here so as to avoid possible misunderstandings between the $\mathbf{R}$ matrix of the semi-recursive method (see [18]) and the projection matrix $\mathbf{R}^{\boldsymbol{\Phi}}$ of this formulation.

In this approach, a second velocity transformation (Matrix R transformation) is carried out in order to remove some dependent coordinates and all the constraints from the equations. Let us consider a multibody system modeled with n relative coordinates subjected to m constraints and with

[6] Do not confuse with $\mathbf{b}_i^v$, an elemental term of the recursive accumulation.

d DoF. A vector of degrees of freedom $\mathbf{z}^i \in \mathbb{R}^d$ can be selected such that the dependent velocities, $\dot{\mathbf{z}}$, accelerations, $\ddot{\mathbf{z}}$, and virtual velocities, $\dot{\mathbf{z}}^*$, can be expressed in terms of the independent ones by using the left inverse matrices $\mathbf{R}^{\mathbf{\Phi}}$ and $\mathbf{S}^{\mathbf{\Phi}}$ from (16) in equations (19) and (21):

$$\dot{\mathbf{z}} = \mathbf{R}^{\mathbf{\Phi}} \dot{\mathbf{z}}^i + \mathbf{S}^{\mathbf{\Phi}} \mathbf{b} \tag{24}$$

$$\ddot{\mathbf{z}} = \mathbf{R}^{\mathbf{\Phi}} \left(\ddot{\mathbf{z}}^i - \dot{\mathbf{B}} \dot{\mathbf{z}} \right) + \mathbf{S}^{\mathbf{\Phi}} \mathbf{c} \tag{25}$$

$$\dot{\mathbf{z}}^* = \mathbf{R}^{\mathbf{\Phi}} \dot{\mathbf{z}}^{*i} \tag{26}$$

being $\mathbf{b}$ and $\mathbf{c}$ the terms related to the temporal constraints derivatives described in sections 2.2.4 and 2.2.5 respectively.[7]

The expressions of the semi-recursive Matrix R formulation can be derived from the virtual power principle applied to a multibody system. For a general multibody system modeled with joint coordinates, the virtual power principle delivers the following system of equations:

$$\dot{\mathbf{z}}^{*\mathrm{T}} \left[\left(\mathbf{R}^{v\mathrm{T}} \mathbf{M}^v \mathbf{R}^v \right) \ddot{\mathbf{z}} - \mathbf{R}^{v\mathrm{T}} \left(\mathbf{Q}^v - \mathbf{M}^v \dot{\mathbf{R}}^v \dot{\mathbf{z}} \right) \right] = 0 \tag{27}$$

with $\mathbf{M}^v$ and $\mathbf{Q}^v$ denoting the mass matrix and generalized forces vector referred to the reference coordinates of each body [18]. Equation (27) can be rewritten using $\mathbf{M}^d = \mathbf{R}^{v\mathrm{T}} \mathbf{M}^v \mathbf{R}^v$, $\mathbf{Q}^d = \mathbf{R}^{v\mathrm{T}} \left(\mathbf{Q}^v - \mathbf{M}^v \dot{\mathbf{R}}^v \dot{\mathbf{z}} \right)$ and the relation (26) between virtual velocities, as:

$$\dot{\mathbf{z}}^{i*\mathrm{T}} \mathbf{R}^{\mathbf{\Phi}\mathrm{T}} \left[\mathbf{M}^d \ddot{\mathbf{z}} - \mathbf{Q}^d \right] = 0 \Rightarrow \mathbf{R}^{\mathbf{\Phi}\mathrm{T}} \left(\mathbf{M}^d \ddot{\mathbf{z}} - \mathbf{Q}^d \right) = \mathbf{0} \tag{28}$$

The last identity in (28) holds because the virtual velocities $\dot{\mathbf{z}}^{i*}$ are independent. Finally, replacing (25), one obtains

$$\left(\mathbf{R}^{\mathbf{\Phi}\mathrm{T}} \mathbf{M}^d \mathbf{R}^{\mathbf{\Phi}} \right) \ddot{\mathbf{z}}^i = \mathbf{R}^{\mathbf{\Phi}\mathrm{T}} \left(\mathbf{Q}^d - \mathbf{M}^d \left(\mathbf{S}^{\mathbf{\Phi}} \mathbf{c} - \mathbf{R}^{\mathbf{\Phi}} \dot{\mathbf{B}} \dot{\mathbf{z}} \right) \right) \tag{29}$$

which constitute the general Matrix R equations for closed loops or non-minimal joint coordinates. Observe that (29) supports the definition of any type of coordinates as independent variables (relative coordinates, natural coordinates, etc.) enhancing the generality of the classical approach.

In the dynamic formulation proposed, the kinematic position and velocity problems as well as the evaluation of the matrix $\mathbf{R}^{\mathbf{\Phi}}$ have to be computed at each time step with (17). Taking into account that the three problems have identical leading matrices, they can be factorized once (for the position problem) and then reused, minimizing the computational cost. The evaluation of the matrix $\mathbf{S}^{\mathbf{\Phi}}$ is not necessary in the dynamics, because the term $\mathbf{S}^{\mathbf{\Phi}}\mathbf{c}$ can be calculated directly by means of the kinematic acceleration analysis (21) with $\ddot{\mathbf{z}}^i - \dot{\mathbf{B}}\dot{\mathbf{z}} = \mathbf{0}$.

3. Sensitivity analysis

The semi-recursive Matrix R formulation for a non-constant $\mathbf{B}$ matrix is analytically differentiated in this section with respect to a set of parameters $\boldsymbol{\rho} \in \mathbb{R}^p$ applying a direct differentiation scheme and the adjoint variable method.

Let us consider an integral objective function dependent on a set of natural coordinates $\mathbf{q}$, $\dot{\mathbf{q}}$ and $\ddot{\mathbf{q}}$, a set of joint coordinates $\mathbf{z}$, $\dot{\mathbf{z}}$ and $\ddot{\mathbf{z}}$, a set of degrees of freedom $\mathbf{z}^i$, $\dot{\mathbf{z}}^i$ and $\ddot{\mathbf{z}}^i$ and a set of parameters $\boldsymbol{\rho}$:

$$\boldsymbol{\psi} = \int_{t_0}^{t_F} \mathbf{g} \left(\mathbf{q}(\mathbf{z}), \dot{\mathbf{q}}(\mathbf{z}, \dot{\mathbf{z}}), \ddot{\mathbf{q}}(\mathbf{z}, \dot{\mathbf{z}}, \ddot{\mathbf{z}}), \mathbf{z}, \dot{\mathbf{z}}, \ddot{\mathbf{z}}, \mathbf{z}^i, \dot{\mathbf{z}}^i, \ddot{\mathbf{z}}^i, \boldsymbol{\rho} \right) \mathrm{d}t \tag{30}$$

Note that each degree of freedom is usually part of the natural coordinates vector $\mathbf{q}$ or the relative coordinates vector, $\mathbf{z}$. In this regard, explicit dependencies of the objective function $\mathbf{g}$ on the degrees of freedom might not be necessary since they can be included in the natural and joint coordinate dependencies, but they are considered separately because they allow particular simplifications in the sensitivity equations. On the other hand, we avoid explicit dependencies with the natural coordinates, $\mathbf{q}$, considered as implicit dependencies on relative dependent coordinates, $\mathbf{z}$, in order to avoid longer expressions.

3.1. Forward sensitivity

Taking derivatives on (30) with respect to a set of parameters $\boldsymbol{\rho}$ and considering the implicit dependencies of $\mathbf{g}$, the sensitivity of the objective function (30) can be represented by the following gradient:

$$\boldsymbol{\psi}' = \int_{t_0}^{t_F} \left(\mathbf{g}_{\mathbf{z}^i} \mathbf{z}^{i\prime} + \mathbf{g}_{\dot{\mathbf{z}}^i} \dot{\mathbf{z}}^{i\prime} + \mathbf{g}_{\ddot{\mathbf{z}}^i} \ddot{\mathbf{z}}^{i\prime} + \mathbf{g}_{\mathbf{z}} \mathbf{z}' + \mathbf{g}_{\dot{\mathbf{z}}} \dot{\mathbf{z}}' + \mathbf{g}_{\ddot{\mathbf{z}}} \ddot{\mathbf{z}}' + \mathbf{g}_{\hat{\rho}} \right) \mathrm{d}t \tag{31}$$

where $(\cdot)'$ denotes state derivatives with respect to the design parameters and subscripts indicate partial derivatives.

The Matrix R formulation imposes to the dependent states, $\mathbf{z}$, $\dot{\mathbf{z}}$ and $\ddot{\mathbf{z}}$ the fulfillment of the constraints vectors at position, velocity and acceleration levels, in terms of a proper selection of degrees of freedom, $\mathbf{z}^i$, $\dot{\mathbf{z}}^i$ and $\ddot{\mathbf{z}}^i$, by means of equations (13), (19) and (21). The sensitivities, $\mathbf{z}'$, $\dot{\mathbf{z}}'$ and $\ddot{\mathbf{z}}'$, of these dependent states, in terms of the degrees of freedom sensitivities, $\mathbf{z}^{i\prime}$, $\dot{\mathbf{z}}^{i\prime}$ and $\ddot{\mathbf{z}}^{i\prime}$, read:

$$\mathbf{\Theta}' = \begin{bmatrix} \mathbf{\Phi}' \\ \mathbf{B}\mathbf{z}' + \mathbf{h}_{\hat{\rho}} - \mathbf{z}^{i\prime} \end{bmatrix} = \mathbf{0} \Rightarrow \begin{bmatrix} \mathbf{\Phi}_{\mathbf{z}} \\ \mathbf{B} \end{bmatrix} \mathbf{z}' = \begin{bmatrix} -\mathbf{\Phi}_{\hat{\rho}} \\ \mathbf{z}^{i\prime} - \mathbf{h}_{\hat{\rho}} \end{bmatrix} \Rightarrow \mathbf{z}' = \mathbf{R}^{\mathbf{\Phi}} \left(\mathbf{z}^{i\prime} - \mathbf{h}_{\hat{\rho}} \right) - \mathbf{S}^{\mathbf{\Phi}} \mathbf{\Phi}_{\hat{\rho}} \tag{32a}$$

$$\dot{\mathbf{\Theta}}' = \begin{bmatrix} \dot{\mathbf{\Phi}}' \\ \mathbf{B}\dot{\mathbf{z}}' + \mathbf{B}'\dot{\mathbf{z}} - \mathbf{z}^{i\prime} \end{bmatrix} = \mathbf{0} \Rightarrow \begin{bmatrix} \mathbf{\Phi}_{\mathbf{z}} \\ \mathbf{B} \end{bmatrix} \dot{\mathbf{z}}' = \begin{bmatrix} -\mathbf{b}^{\rho} \\ \dot{\mathbf{z}}^{i\prime} - \bar{\mathbf{b}}^{\rho} \end{bmatrix} \Rightarrow \dot{\mathbf{z}}' = \mathbf{R}^{\mathbf{\Phi}} \left(\dot{\mathbf{z}}^{i\prime} - \bar{\mathbf{b}}^{\rho} \right) - \mathbf{S}^{\mathbf{\Phi}} \mathbf{b}^{\rho} \tag{32b}$$

$$\ddot{\mathbf{\Theta}}' = \begin{bmatrix} \ddot{\mathbf{\Phi}}' \\ \mathbf{B}\ddot{\mathbf{z}}' + \mathbf{B}'\ddot{\mathbf{z}} + \dot{\mathbf{B}}'\dot{\mathbf{z}} + \dot{\mathbf{B}}\dot{\mathbf{z}}' - \ddot{\mathbf{z}}^{i\prime} \end{bmatrix} = \mathbf{0} \Rightarrow \begin{bmatrix} \mathbf{\Phi}_{\dot{\mathbf{z}}} \\ \mathbf{B} \end{bmatrix} \ddot{\mathbf{z}}' = \begin{bmatrix} -\mathbf{c}^{\rho} \\ \ddot{\mathbf{z}}^{i\prime} - \bar{\mathbf{c}}^{\rho} \end{bmatrix} \Rightarrow \ddot{\mathbf{z}}' = \mathbf{R}^{\mathbf{\Phi}} \left(\ddot{\mathbf{z}}^{i\prime} - \bar{\mathbf{c}}^{\rho} \right) - \mathbf{S}^{\mathbf{\Phi}} \mathbf{c}^{\rho} \tag{32c}$$

[7] Note that term $\mathbf{b}$ does not show up in the virtual velocity equations because they are instantaneous variations by definition.

where equation (16) has been considered and $\mathbf{b}^{\rho}$, $\mathbf{c}^{\rho}$, $\bar{\mathbf{b}}^{\rho}$ and $\bar{\mathbf{c}}^{\rho}$ defined as:

$$\mathbf{b}^{\rho} = \dot{\mathbf{\Phi}}_{\hat{\mathbf{z}}}\mathbf{z}' + \dot{\mathbf{\Phi}}_{\hat{\rho}} \tag{33}$$

$$\mathbf{c}^{\rho} = 2\dot{\mathbf{\Phi}}_{\hat{\mathbf{z}}}\dot{\mathbf{z}}' + \ddot{\mathbf{\Phi}}_{\hat{\mathbf{z}}}\mathbf{z}' + \ddot{\mathbf{\Phi}}_{\hat{\rho}} \tag{34}$$

$$\bar{\mathbf{b}}^{\rho} = \dot{\mathbf{B}}\mathbf{z}' + \mathbf{B}_{\hat{\rho}}\dot{\mathbf{z}} \tag{35}$$

$$\bar{\mathbf{c}}^{\rho} = 2\dot{\mathbf{B}}\dot{\mathbf{z}}' + \ddot{\mathbf{B}}\mathbf{z}' + \mathbf{B}_{\hat{\rho}}\ddot{\mathbf{z}} + \dot{\mathbf{B}}_{\hat{\rho}}\dot{\mathbf{z}} \tag{36}$$

with $\dot{\mathbf{B}} = \mathbf{B}_{\hat{\mathbf{z}}}\dot{\mathbf{z}}$ and $\ddot{\mathbf{B}} = \mathbf{B}_{\hat{\mathbf{z}}}\ddot{\mathbf{z}} + \dot{\mathbf{B}}_{\hat{\mathbf{z}}}\dot{\mathbf{z}}$.

Expanding terms to make explicit the dependencies on the sensitivities of the independent coordinates in positions, velocities and accelerations, equations (32) are transformed:

$$\mathbf{z}' = \mathbf{R}^{\mathbf{\Phi}}\left(\mathbf{z}^{i\prime} - \mathbf{h}_{\hat{\rho}}\right) - \mathbf{S}^{\mathbf{\Phi}}\mathbf{\Phi}_{\hat{\rho}} \tag{37a}$$

$$\dot{\mathbf{z}}' = \mathbf{R}^{\mathbf{\Phi}}\left(\dot{\mathbf{z}}^{i\prime} - \left(\dot{\mathbf{B}}\mathbf{z}' + \mathbf{B}_{\hat{\rho}}\dot{\mathbf{z}}\right)\right) - \mathbf{S}^{\mathbf{\Phi}}\left(\dot{\mathbf{\Phi}}_{\hat{\mathbf{z}}}\mathbf{z}' + \dot{\mathbf{\Phi}}_{\hat{\rho}}\right) \tag{37b}$$

$$\ddot{\mathbf{z}}' = \mathbf{R}^{\mathbf{\Phi}}\left(\ddot{\mathbf{z}}^{i\prime} - \left(2\dot{\mathbf{B}}\dot{\mathbf{z}}' + \ddot{\mathbf{B}}\mathbf{z}' + \mathbf{B}_{\hat{\rho}}\ddot{\mathbf{z}} + \dot{\mathbf{B}}_{\hat{\rho}}\dot{\mathbf{z}}\right)\right) - \mathbf{S}^{\mathbf{\Phi}}\left(2\dot{\mathbf{\Phi}}_{\hat{\mathbf{z}}}\dot{\mathbf{z}}' + \ddot{\mathbf{\Phi}}_{\hat{\mathbf{z}}}\mathbf{z}' + \ddot{\mathbf{\Phi}}_{\hat{\rho}}\right) \tag{37c}$$

Identifying terms in equations (37), the following relations can be inferred:

$$\mathbf{z}_{\mathbf{z}^i} = \dot{\mathbf{z}}_{\dot{\mathbf{z}}^i} = \ddot{\mathbf{z}}_{\ddot{\mathbf{z}}^i} = \mathbf{R}^{\mathbf{\Phi}} \tag{38a}$$

$$\mathbf{z}_{\rho} = -\mathbf{S}^{\mathbf{\Phi}}\mathbf{\Phi}_{\hat{\rho}} - \mathbf{R}^{\mathbf{\Phi}}\mathbf{h}_{\hat{\rho}} \tag{38b}$$

$$\dot{\mathbf{z}}_{\mathbf{z}} = -\mathbf{S}^{\mathbf{\Phi}}\dot{\mathbf{\Phi}}_{\hat{\mathbf{z}}} - \mathbf{R}^{\mathbf{\Phi}}\dot{\mathbf{B}} \tag{38c}$$

$$\dot{\mathbf{z}}_{\rho} = -\mathbf{S}^{\mathbf{\Phi}}\dot{\mathbf{\Phi}}_{\hat{\rho}} - \mathbf{R}^{\mathbf{\Phi}}\mathbf{B}_{\hat{\rho}}\dot{\mathbf{z}} \tag{38d}$$

$$\ddot{\mathbf{z}}_{\dot{\mathbf{z}}} = -2\mathbf{S}^{\mathbf{\Phi}}\dot{\mathbf{\Phi}}_{\hat{\mathbf{z}}} - 2\mathbf{R}^{\mathbf{\Phi}}\dot{\mathbf{B}} \tag{38e}$$

$$\ddot{\mathbf{z}}_{\mathbf{z}} = -\mathbf{S}^{\mathbf{\Phi}}\ddot{\mathbf{\Phi}}_{\hat{\mathbf{z}}} - \mathbf{R}^{\mathbf{\Phi}}\ddot{\mathbf{B}} \tag{38f}$$

$$\ddot{\mathbf{z}}_{\rho} = -\mathbf{S}^{\mathbf{\Phi}}\ddot{\mathbf{\Phi}}_{\hat{\rho}} - \mathbf{R}^{\mathbf{\Phi}}\left(\mathbf{B}_{\hat{\rho}}\ddot{\mathbf{z}} + \dot{\mathbf{B}}_{\hat{\rho}}\dot{\mathbf{z}}\right) \tag{38g}$$

The expressions whose sensitivity analysis is being addressed include the definition of a non-constant $\mathbf{B}$ matrix, hence the expressions developed are generic for any selection of degrees of freedom, including the particular case of a constant matrix $\mathbf{B}$, in which case the derivatives of $\mathbf{B}$ are null.

Expressions (37) and (38) allow removing $\mathbf{z}'$, $\dot{\mathbf{z}}'$, $\ddot{\mathbf{z}}'$ from the gradient of the objective function (31). Then, all the contributions involving $\mathbf{z}^{i\prime}$, $\dot{\mathbf{z}}^{i\prime}$ and $\ddot{\mathbf{z}}^{i\prime}$ can be gathered together:

$$\mathbf{g}_{\check{\mathbf{z}}^i} = \mathbf{g}_{\mathbf{z}^i} + \left[\mathbf{g}_{\hat{\mathbf{z}}} + \mathbf{g}_{\hat{\dot{\mathbf{z}}}}\dot{\mathbf{z}}_{\mathbf{z}} + \mathbf{g}_{\hat{\ddot{\mathbf{z}}}}\left(\ddot{\mathbf{z}}_{\dot{\mathbf{z}}}\dot{\mathbf{z}}_{\mathbf{z}} + \ddot{\mathbf{z}}_{\mathbf{z}}\right)\right]\mathbf{z}_{\mathbf{z}^i} \tag{39a}$$

$$\mathbf{g}_{\check{\dot{\mathbf{z}}}^i} = \mathbf{g}_{\dot{\mathbf{z}}^i} + \left[\mathbf{g}_{\hat{\dot{\mathbf{z}}}} + \mathbf{g}_{\hat{\ddot{\mathbf{z}}}}\ddot{\mathbf{z}}_{\dot{\mathbf{z}}}\right]\dot{\mathbf{z}}_{\dot{\mathbf{z}}^i} \tag{39b}$$

$$\mathbf{g}_{\check{\ddot{\mathbf{z}}}^i} = \mathbf{g}_{\ddot{\mathbf{z}}^i} + \mathbf{g}_{\hat{\ddot{\mathbf{z}}}}\ddot{\mathbf{z}}_{\ddot{\mathbf{z}}^i} \tag{39c}$$

$$\mathbf{g}_{\check{\rho}} = \mathbf{g}_{\hat{\rho}} + \mathbf{g}_{\hat{\mathbf{z}}}\mathbf{z}_{\rho} + \mathbf{g}_{\hat{\dot{\mathbf{z}}}}\left(\dot{\mathbf{z}}_{\mathbf{z}}\mathbf{z}_{\rho} + \dot{\mathbf{z}}_{\rho}\right) + \mathbf{g}_{\hat{\ddot{\mathbf{z}}}}\left(\ddot{\mathbf{z}}_{\dot{\mathbf{z}}}\left(\dot{\mathbf{z}}_{\mathbf{z}}\mathbf{z}_{\rho} + \dot{\mathbf{z}}_{\rho}\right) + \ddot{\mathbf{z}}_{\mathbf{z}}\mathbf{z}_{\rho} + \ddot{\mathbf{z}}_{\rho}\right) \tag{39d}$$

where $(\cdot)_{\check{\mathbf{x}}}$ identifies a partial derivative including implicit dependencies with respect to relative coordinates:

$$(\cdot)_{\check{\mathbf{x}}} = (\cdot)_{\hat{\mathbf{x}}} + (\cdot)_{\hat{\mathbf{z}}}\mathbf{z}_{\mathbf{x}} + (\cdot)_{\hat{\dot{\mathbf{z}}}}\left(\dot{\mathbf{z}}_{\mathbf{x}} + \dot{\mathbf{z}}_{\mathbf{z}}\mathbf{z}_{\mathbf{x}}\right) + (\cdot)_{\hat{\ddot{\mathbf{z}}}}\left(\ddot{\mathbf{z}}_{\mathbf{x}} + \ddot{\mathbf{z}}_{\dot{\mathbf{z}}}\dot{\mathbf{z}}_{\mathbf{x}} + \left(\ddot{\mathbf{z}}_{\mathbf{z}} + \ddot{\mathbf{z}}_{\dot{\mathbf{z}}}\dot{\mathbf{z}}_{\mathbf{z}}\right)\mathbf{z}_{\mathbf{x}}\right) \tag{40}$$

Finally, gathering the terms with respect to $\mathbf{z}^{i\prime}$, $\dot{\mathbf{z}}^{i\prime}$ and $\ddot{\mathbf{z}}^{i\prime}$, the expression of the gradient results:

$$\boldsymbol{\psi}' = \int_{t_0}^{t_F}\left(\mathbf{g}_{\check{\mathbf{z}}^i}\mathbf{z}^{i\prime} + \mathbf{g}_{\check{\dot{\mathbf{z}}}^i}\dot{\mathbf{z}}^{i\prime} + \mathbf{g}_{\check{\ddot{\mathbf{z}}}^i}\ddot{\mathbf{z}}^{i\prime} + \mathbf{g}_{\check{\rho}}\right)\mathrm{d}t \tag{41}$$

The sensitivities $\mathbf{z}^{i\prime}$, $\dot{\mathbf{z}}^{i\prime}$ and $\ddot{\mathbf{z}}^{i\prime}$ can be obtained differentiating (29) with respect to the vector of parameters. First of all, let us transform the system (29) to allow a more compact notation:

$$\bar{\mathbf{M}}\ddot{\mathbf{z}}^i = \bar{\mathbf{Q}} \tag{42}$$

with

$$\bar{\mathbf{M}} = \left(\mathbf{R}^{\mathbf{\Phi}\mathrm{T}}\mathbf{M}^d\mathbf{R}^{\mathbf{\Phi}}\right) \tag{43}$$

$$\bar{\mathbf{Q}} = \mathbf{R}^{\mathbf{\Phi}\mathrm{T}}\left(\mathbf{Q}^d - \mathbf{M}^d\left(\mathbf{S}^{\mathbf{\Phi}}\mathbf{c} - \mathbf{R}^{\mathbf{\Phi}}\dot{\mathbf{B}}\dot{\mathbf{z}}\right)\right) \tag{44}$$

Then, taking derivatives in (42) with respect to the set of parameters $\boldsymbol{\rho}$:

$$\bar{\mathbf{M}}'\ddot{\mathbf{z}}^i + \bar{\mathbf{M}}\ddot{\mathbf{z}}^{i\prime} = \bar{\mathbf{Q}}' \tag{45}$$

where:

$$\bar{\mathbf{Q}}' = \frac{\mathrm{d}\bar{\mathbf{Q}}}{\mathrm{d}\boldsymbol{\rho}} = \bar{\mathbf{Q}}_{\check{\mathbf{z}}^i}\mathbf{z}^{i\prime} + \bar{\mathbf{Q}}_{\check{\dot{\mathbf{z}}}^i}\dot{\mathbf{z}}^{i\prime} + \bar{\mathbf{Q}}_{\check{\rho}} = -\bar{\mathbf{K}}\mathbf{z}^{i\prime} - \bar{\mathbf{C}}\dot{\mathbf{z}}^{i\prime} + \bar{\mathbf{Q}}_{\check{\rho}} \tag{46}$$

$$\bar{\mathbf{M}}'\ddot{\mathbf{z}}^i = \frac{\mathrm{d}\bar{\mathbf{M}}}{\mathrm{d}\boldsymbol{\rho}}\ddot{\mathbf{z}}^i = \left(\bar{\mathbf{M}}_{\dot{\mathbf{z}}^i}\ddot{\mathbf{z}}^i\right)\mathbf{z}^{i\prime} + \bar{\mathbf{M}}_{\hat{\boldsymbol{\rho}}}\ddot{\mathbf{z}}^i \tag{47}$$

The resulting Tangent Linear Model (hereinafter TLM) takes the form:

$$\bar{\mathbf{M}}\ddot{\mathbf{z}}^{i\prime} + \bar{\mathbf{C}}\dot{\mathbf{z}}^{i\prime} + \left(\bar{\mathbf{K}} + \bar{\mathbf{M}}_{\dot{\mathbf{z}}^i}\ddot{\mathbf{z}}^i\right)\mathbf{z}^{i\prime} = \bar{\mathbf{Q}}_{\hat{\boldsymbol{\rho}}} - \bar{\mathbf{M}}_{\hat{\boldsymbol{\rho}}}\ddot{\mathbf{z}}^i \tag{48a}$$

$$\mathbf{z}^{i\prime}\left(t_0\right) = \mathbf{z}_0^{i\prime} \tag{48b}$$

$$\dot{\mathbf{z}}^{i\prime}\left(t_0\right) = \dot{\mathbf{z}}_0^{i\prime} \tag{48c}$$

wherein:

$$\bar{\mathbf{Q}}_{\hat{\boldsymbol{\rho}}} - \bar{\mathbf{M}}_{\hat{\boldsymbol{\rho}}}\ddot{\mathbf{z}}^i = \bar{\mathbf{Q}}_{\hat{\boldsymbol{\rho}}} - \bar{\mathbf{M}}_{\hat{\boldsymbol{\rho}}}\ddot{\mathbf{z}}^i + \left(\bar{\mathbf{Q}}_{\hat{\mathbf{z}}} + \bar{\mathbf{Q}}_{\hat{\dot{\mathbf{z}}}}\dot{\mathbf{z}}_{\mathbf{z}} - \bar{\mathbf{M}}_{\hat{\mathbf{z}}}\ddot{\mathbf{z}}^i\right)\mathbf{z}_{\boldsymbol{\rho}} + \bar{\mathbf{Q}}_{\hat{\dot{\mathbf{z}}}}\dot{\mathbf{z}}_{\boldsymbol{\rho}} \tag{49}$$

$$\bar{\mathbf{K}} = -\bar{\mathbf{Q}}_{\dot{\mathbf{z}}^i} = -\left(\bar{\mathbf{Q}}_{\hat{\mathbf{z}}} + \bar{\mathbf{Q}}_{\hat{\dot{\mathbf{z}}}}\dot{\mathbf{z}}_{\mathbf{z}}\right)\mathbf{z}_{\mathbf{z}^i} \tag{50}$$

$$\bar{\mathbf{C}} = -\bar{\mathbf{Q}}_{\dot{\mathbf{z}}^i} = -\bar{\mathbf{Q}}_{\hat{\dot{\mathbf{z}}}}\dot{\mathbf{z}}_{\dot{\mathbf{z}}^i} \tag{51}$$

$$\bar{\mathbf{M}}_{\dot{\mathbf{z}}^i}\ddot{\mathbf{z}}^i = \left(\bar{\mathbf{M}}_{\hat{\mathbf{z}}}\ddot{\mathbf{z}}^i\right)\mathbf{z}_{\mathbf{z}^i} \tag{52}$$

Back into expressions (49) to (52), the following derivatives are needed as well:

$$\bar{\mathbf{Q}}_{\hat{\mathbf{z}}} = \mathbf{R}_{\mathbf{z}}^{\boldsymbol{\Phi}\mathrm{T}}\left[\mathbf{Q}^d - \mathbf{M}^d\left(\mathbf{S}^{\boldsymbol{\Phi}}\mathbf{c} - \mathbf{R}^{\boldsymbol{\Phi}}\dot{\mathbf{B}}\dot{\mathbf{z}}\right)\right] - \mathbf{R}^{\boldsymbol{\Phi}\mathrm{T}}\left[\mathbf{K} + \mathbf{M}_{\hat{\mathbf{z}}}^d\left(\mathbf{S}^{\boldsymbol{\Phi}}\mathbf{c} - \mathbf{R}^{\boldsymbol{\Phi}}\dot{\mathbf{B}}\dot{\mathbf{z}}\right) + \mathbf{M}^d\left(\mathbf{S}^{\boldsymbol{\Phi}}\mathbf{c} - \mathbf{R}^{\boldsymbol{\Phi}}\dot{\mathbf{B}}\dot{\mathbf{z}}\right)_{\hat{\mathbf{z}}}\right], \tag{53}$$

$$\bar{\mathbf{Q}}_{\hat{\dot{\mathbf{z}}}} = -\mathbf{R}^{\boldsymbol{\Phi}\mathrm{T}}\left(\mathbf{C} + \mathbf{M}^d\left(\mathbf{S}^{\boldsymbol{\Phi}}\mathbf{c}_{\hat{\dot{\mathbf{z}}}} - 2\mathbf{R}^{\boldsymbol{\Phi}}\dot{\mathbf{B}}\right)\right), \tag{54}$$

$$\bar{\mathbf{Q}}_{\hat{\boldsymbol{\rho}}} = \mathbf{R}_{\boldsymbol{\rho}}^{\boldsymbol{\Phi}\mathrm{T}}\left[\mathbf{Q}^d - \mathbf{M}^d\left(\mathbf{S}^{\boldsymbol{\Phi}}\mathbf{c} - \mathbf{R}^{\boldsymbol{\Phi}}\dot{\mathbf{B}}\dot{\mathbf{z}}\right)\right] + \mathbf{R}^{\boldsymbol{\Phi}\mathrm{T}}\left[\mathbf{Q}_{\hat{\boldsymbol{\rho}}}^d - \mathbf{M}_{\hat{\boldsymbol{\rho}}}^d\left(\mathbf{S}^{\boldsymbol{\Phi}}\mathbf{c} - \mathbf{R}^{\boldsymbol{\Phi}}\dot{\mathbf{B}}\dot{\mathbf{z}}\right) - \mathbf{M}^d\left(\mathbf{S}^{\boldsymbol{\Phi}}\mathbf{c} - \mathbf{R}^{\boldsymbol{\Phi}}\dot{\mathbf{B}}\dot{\mathbf{z}}\right)_{\hat{\boldsymbol{\rho}}}\right] \tag{55}$$

The derivatives of matrix $\mathbf{R}^{\boldsymbol{\Phi}}$ can be attained using its definition as a basis of the nullspace of $\boldsymbol{\Phi}_{\hat{\mathbf{z}}}$ for the degrees of freedom selected:

$$\boldsymbol{\Theta}_{\hat{\mathbf{z}}}\mathbf{R}^{\boldsymbol{\Phi}} = \begin{bmatrix}\mathbf{0}_{m\times d}\\ \mathbf{I}_d\end{bmatrix} \Rightarrow \mathbf{R}_{\hat{\mathbf{z}}}^{\boldsymbol{\Phi}} = -\boldsymbol{\Theta}_{\hat{\mathbf{z}}}^{+}\boldsymbol{\Theta}_{\hat{\mathbf{z}}\hat{\mathbf{z}}}\mathbf{R}^{\boldsymbol{\Phi}} = -\left(\mathbf{S}^{\boldsymbol{\Phi}}\boldsymbol{\Phi}_{\hat{\mathbf{z}}\hat{\mathbf{z}}} + \mathbf{R}^{\boldsymbol{\Phi}}\mathbf{B}_{\hat{\mathbf{z}}}\right)\mathbf{R}^{\boldsymbol{\Phi}} \tag{56}$$

$$\mathbf{R}_{\hat{\boldsymbol{\rho}}}^{\boldsymbol{\Phi}} = -\boldsymbol{\Theta}_{\hat{\mathbf{z}}}^{+}\boldsymbol{\Theta}_{\hat{\mathbf{z}}\hat{\boldsymbol{\rho}}}\mathbf{R}^{\boldsymbol{\Phi}} = -\left(\mathbf{S}^{\boldsymbol{\Phi}}\boldsymbol{\Phi}_{\hat{\mathbf{z}}\hat{\boldsymbol{\rho}}} + \mathbf{R}^{\boldsymbol{\Phi}}\mathbf{B}_{\hat{\boldsymbol{\rho}}}\right)\mathbf{R}^{\boldsymbol{\Phi}} \tag{57}$$

Concerning the term $\left(\mathbf{S}^{\boldsymbol{\Phi}}\mathbf{c} - \mathbf{R}^{\boldsymbol{\Phi}}\dot{\mathbf{B}}\dot{\mathbf{z}}\right)_{\hat{\mathbf{z}}}$, it can be obtained from the definition of $\mathbf{S}^{\boldsymbol{\Phi}}\mathbf{c} - \mathbf{R}^{\boldsymbol{\Phi}}\dot{\mathbf{B}}\dot{\mathbf{z}}$ as a particular solution for the acceleration problem (21), with $\ddot{\mathbf{z}}^i = \mathbf{0}$:

$$\boldsymbol{\Theta}_{\hat{\mathbf{z}}}\left(\mathbf{S}^{\boldsymbol{\Phi}}\mathbf{c} - \mathbf{R}^{\boldsymbol{\Phi}}\dot{\mathbf{B}}\dot{\mathbf{z}}\right) = \begin{bmatrix}\mathbf{c}\\ -\dot{\mathbf{B}}\dot{\mathbf{z}}\end{bmatrix} \Rightarrow \left(\mathbf{S}^{\boldsymbol{\Phi}}\mathbf{c} - \mathbf{R}^{\boldsymbol{\Phi}}\dot{\mathbf{B}}\dot{\mathbf{z}}\right)_{\hat{\mathbf{z}}} = \mathbf{S}^{\boldsymbol{\Phi}}\mathbf{c}_{\hat{\mathbf{z}}} - \mathbf{R}^{\boldsymbol{\Phi}}\dot{\mathbf{B}}_{\hat{\mathbf{z}}}\dot{\mathbf{z}} - \left(\mathbf{S}^{\boldsymbol{\Phi}}\boldsymbol{\Phi}_{\hat{\mathbf{z}}\hat{\mathbf{z}}} + \mathbf{R}^{\boldsymbol{\Phi}}\mathbf{B}_{\hat{\mathbf{z}}}\right)\left(\mathbf{S}^{\boldsymbol{\Phi}}\mathbf{c} - \mathbf{R}^{\boldsymbol{\Phi}}\dot{\mathbf{B}}\dot{\mathbf{z}}\right) \tag{58}$$

$$\left(\mathbf{S}^{\boldsymbol{\Phi}}\mathbf{c} - \mathbf{R}^{\boldsymbol{\Phi}}\dot{\mathbf{B}}\dot{\mathbf{z}}\right)_{\hat{\boldsymbol{\rho}}} = \mathbf{S}^{\boldsymbol{\Phi}}\mathbf{c}_{\hat{\boldsymbol{\rho}}} - \mathbf{R}^{\boldsymbol{\Phi}}\dot{\mathbf{B}}_{\hat{\boldsymbol{\rho}}}\dot{\mathbf{z}} - \left(\mathbf{S}^{\boldsymbol{\Phi}}\boldsymbol{\Phi}_{\hat{\mathbf{z}}\hat{\boldsymbol{\rho}}} + \mathbf{R}^{\boldsymbol{\Phi}}\mathbf{B}_{\hat{\boldsymbol{\rho}}}\right)\left(\mathbf{S}^{\boldsymbol{\Phi}}\mathbf{c} - \mathbf{R}^{\boldsymbol{\Phi}}\dot{\mathbf{B}}\dot{\mathbf{z}}\right) \tag{59}$$

Finally, the derivatives of the mass matrix times acceleration $\bar{\mathbf{M}}_{\hat{\mathbf{z}}}\ddot{\mathbf{z}}^i$ and $\bar{\mathbf{M}}_{\hat{\boldsymbol{\rho}}}\ddot{\mathbf{z}}^i$ make use of some of the previously calculated terms as well.

$$\bar{\mathbf{M}}_{\hat{\mathbf{z}}}\ddot{\mathbf{z}} = \mathbf{R}_{\hat{\mathbf{z}}}^{\boldsymbol{\Phi}\mathrm{T}}\mathbf{M}^d\mathbf{R}^{\boldsymbol{\Phi}}\ddot{\mathbf{z}}^i + \mathbf{R}^{\boldsymbol{\Phi}\mathrm{T}}\mathbf{M}_{\hat{\mathbf{z}}}^d\mathbf{R}^{\boldsymbol{\Phi}}\ddot{\mathbf{z}}^i + \mathbf{R}^{\boldsymbol{\Phi}\mathrm{T}}\mathbf{M}^d\mathbf{R}_{\hat{\mathbf{z}}}^{\boldsymbol{\Phi}}\ddot{\mathbf{z}}^i \tag{60}$$

$$\bar{\mathbf{M}}_{\hat{\boldsymbol{\rho}}}\ddot{\mathbf{z}} = \mathbf{R}_{\hat{\boldsymbol{\rho}}}^{\boldsymbol{\Phi}\mathrm{T}}\mathbf{M}^d\mathbf{R}^{\boldsymbol{\Phi}}\ddot{\mathbf{z}}^i + \mathbf{R}^{\boldsymbol{\Phi}\mathrm{T}}\mathbf{M}_{\hat{\boldsymbol{\rho}}}^d\mathbf{R}^{\boldsymbol{\Phi}}\ddot{\mathbf{z}}^i + \mathbf{R}^{\boldsymbol{\Phi}\mathrm{T}}\mathbf{M}^d\mathbf{R}_{\hat{\boldsymbol{\rho}}}^{\boldsymbol{\Phi}}\ddot{\mathbf{z}}^i \tag{61}$$

Observe that if a constant $\mathbf{B}$ matrix is selected, many terms in the expressions will be canceled. Detailed expressions of the derivatives of masses ($\mathbf{M}^d$), forces ($\mathbf{Q}^d$) and constraints ($\boldsymbol{\Phi}$, $\dot{\boldsymbol{\Phi}}$, $\ddot{\boldsymbol{\Phi}}$) with respect to relative coordinates and parameters can be found in [33].

The direct sensitivity analysis of the semi-recursive Matrix R formulation has been implemented in the MBSLIM multibody library as a general sensitivity formulation considering a non-constant $\mathbf{B}$ matrix and the RTdyn0 (center of mass of each body as reference point) and RTdyn1 (global origin of coordinates as reference point) approaches.

Algorithm 1 Direct sensitivity.

procedure SOLVEDIRECTSENSITIVITIES
  $t \leftarrow t_0$
  $\mathbf{z}_0, \dot{\mathbf{z}}_0 \leftarrow iniPosVel(\mathbf{z}_0^i, \dot{\mathbf{z}}_0^i)$
  $\mathbf{z}_0', \dot{\mathbf{z}}_0' \leftarrow iniSensitPosVel(\mathbf{z}_0^{i\prime}, \dot{\mathbf{z}}_0^{i\prime})$
  $\ddot{\mathbf{z}}_0 \leftarrow iniAcel(\mathbf{z}_0, \dot{\mathbf{z}}_0)$
  $\ddot{\mathbf{z}}_0' \leftarrow \left(\bar{\mathbf{M}}\ddot{\mathbf{z}}^{i\prime} + \bar{\mathbf{C}}\dot{\mathbf{z}}^{i\prime} + \left(\bar{\mathbf{K}} + \bar{\mathbf{M}}_{\dot{\mathbf{z}}^i}\ddot{\mathbf{z}}^i\right)\mathbf{z}^{i\prime} = \bar{\mathbf{Q}}_{\hat{\boldsymbol{\rho}}} - \bar{\mathbf{M}}_{\hat{\boldsymbol{\rho}}}\ddot{\mathbf{z}}^i\right)$
  while $t <= t_{end}$ **do**
    $t = t + \Delta t$
    $\mathbf{z}, \dot{\mathbf{z}}, \ddot{\mathbf{z}} \leftarrow solveDynamics()$
    $\mathbf{z}_i^{i\prime}, \dot{\mathbf{z}}_i^{i\prime}, \ddot{\mathbf{z}}_i^{i\prime} \leftarrow predictor\left(\mathbf{z}_{i-1}^{i\prime}, \dot{\mathbf{z}}_{i-1}^{i\prime}, \ddot{\mathbf{z}}_{i-1}^{i\prime}\right)$
    $\bar{\mathbf{M}}_{\dot{\mathbf{z}}^i}\ddot{\mathbf{z}}^i, \bar{\mathbf{K}}, \bar{\mathbf{C}}, \bar{\mathbf{Q}}_{\hat{\boldsymbol{\rho}}} \leftarrow evalDerivatives()$
    $\mathbf{z}_i^{i\prime} \leftarrow \left(\bar{\mathbf{M}}\ddot{\mathbf{z}}^{i\prime} + \bar{\mathbf{C}}\dot{\mathbf{z}}^{i\prime} + \left(\bar{\mathbf{K}} + \bar{\mathbf{M}}_{\dot{\mathbf{z}}^i}\ddot{\mathbf{z}}^i\right)\mathbf{z}^{i\prime} = \bar{\mathbf{Q}}_{\hat{\boldsymbol{\rho}}} - \bar{\mathbf{M}}_{\hat{\boldsymbol{\rho}}}\ddot{\mathbf{z}}^i\right)$
    $\dot{\mathbf{z}}_i^{i\prime}, \ddot{\mathbf{z}}_i^{i\prime} \leftarrow corrector\left(\mathbf{z}_i^{i\prime}, \mathbf{z}_{i-1}^{i\prime}, \dot{\mathbf{z}}_{i-1}^{i\prime}, \ddot{\mathbf{z}}_{i-1}^{i\prime}\right)$
    $\boldsymbol{\psi}'^{\mathrm{T}} \leftarrow integrate\left(\int_{t_0}^{t_F}\left(\mathbf{g}_{\hat{\mathbf{z}}^i}\mathbf{z}^{i\prime} + \mathbf{g}_{\hat{\dot{\mathbf{z}}}^i}\dot{\mathbf{z}}^{i\prime} + \mathbf{g}_{\hat{\ddot{\mathbf{z}}}^i}\ddot{\mathbf{z}}^{i\prime} + \mathbf{g}_{\hat{\boldsymbol{\rho}}}\right)\mathrm{d}t\right)$
end

The basic steps of the direct sensitivity formulation presented, are summarized in the pseudo-code 1. The process begins with the calculation of positions, velocities and accelerations at the initial time, as well as their corresponding sensitivities. Then, the time can be increased and the states

can be computed by solving the dynamic equations. With the new states, the TLM (48) can be evaluated and solved applying a numerical integrator. With the computed sensitivities of the states it is possible to integrate the objective function. This process is executed each time step until the final time is reached.

In the semi-recursive Matrix R direct sensitivity formulation, most of the computational effort is devoted to the calculation of derivatives, while the time required for the solution of the TLM is usually negligible for a low number of parameters. The computational burden associated to the calculation of derivatives can be reduced using a combination of dense and sparse methods (sparse is convenient for constraint derivatives), by means of the storage and reuse of terms (there are many elemental derivatives repeated which can be efficiently computed and stored), by means of the exploitation of symmetry in matrices and hyper-matrices and by means of the use of recursive procedures whenever it is possible (for the evaluation of derivatives in the open-loop model, for instance).

Because the number of sensitivity systems of equations grows with the number of parameters, the computational effort devoted to solve them scales linearly with it, and for a number high enough, it is desirable to resort to other sensitivity schemes in which the computational effort is independent of this dimension, like the adjoint method.

3.2. Adjoint sensitivity analysis

The adjoint variable method applied to Matrix R formulations was presented and discussed in [34], starting from three different constructions of the equations of motion: a first-order explicit ODE system, a first-order implicit ODE system and a second-order implicit ODE system. The first option delivers the simplest possible expressions for the adjoint equations, avoiding the time derivative of the mass matrix (as in the first-order implicit ODE system) and the time derivative of the damping matrix and the second time derivative of the mass matrix (as in the second-order implicit ODE system). In the current work, the developments presented in [34] for the first-order explicit ODE system will be recalled and combined with the semi-recursive formalism.

First of all, the semi-recursive Matrix R equations of motion (42) should be reformulated as a first-order implicit system by means of the definition of a new vector of states $\mathbf{y} = \left[\mathbf{z}^{i\mathrm{T}} \quad \mathbf{v}^{\mathrm{T}}\right]^{\mathrm{T}}$, being $\dot{\mathbf{z}}^i = \mathbf{v}$.

$$\begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \bar{\mathbf{M}} \end{bmatrix} \begin{bmatrix} \dot{\mathbf{z}}^i \\ \dot{\mathbf{v}} \end{bmatrix} = \begin{bmatrix} \mathbf{v} \\ \bar{\mathbf{Q}} \end{bmatrix} \tag{62a}$$

$$\hat{\mathbf{M}}(\mathbf{y}, \boldsymbol{\rho})\, \dot{\mathbf{y}} = \hat{\mathbf{Q}}(t, \mathbf{y}, \boldsymbol{\rho}) \tag{62b}$$

Considering that the leading matrix of (62b) always has an inverse (for a proper selection of DoF), (62b) can be transformed into a first-order explicit system:

$$\dot{\mathbf{y}} = \hat{\mathbf{M}}^{-1}(\mathbf{y}, \boldsymbol{\rho})\, \hat{\mathbf{Q}}(t, \mathbf{y}, \boldsymbol{\rho}) = \mathbf{f}(t, \mathbf{y}, \boldsymbol{\rho}) \tag{63}$$

Let us now consider the following Lagrangian:

$$\mathcal{L}(\boldsymbol{\rho}) = \boldsymbol{\psi} - \int_{t_0}^{t_F} \boldsymbol{\mu}^{\mathrm{T}} \left(\dot{\mathbf{y}} - \mathbf{f}(t, \mathbf{y}, \boldsymbol{\rho})\right) \mathrm{d}t \tag{64}$$

where $\boldsymbol{\psi} = \boldsymbol{\psi}(\mathbf{y}, \dot{\mathbf{y}}, \boldsymbol{\rho})$ is the objective function defined in (30), with all the dependencies considered as implicit.

Since $\dot{\mathbf{y}} - \mathbf{f}(t, \mathbf{y}, \boldsymbol{\rho}) = \mathbf{0}$, the value of the Lagrangian is equal to the value of the objective function, and also its derivatives for any value of $\boldsymbol{\mu}$.

Computing the infinitesimal variations of $\mathcal{L}$,

$$\delta\mathcal{L} = \int_{t_0}^{t_F} \left(\mathbf{g}_{\check{\mathbf{y}}} \delta\mathbf{y} + \mathbf{g}_{\check{\dot{\mathbf{y}}}} \delta\dot{\mathbf{y}} + \mathbf{g}_{\check{\boldsymbol{\rho}}} \delta\boldsymbol{\rho}\right) \mathrm{d}t - \int_{t_0}^{t_F} \delta\boldsymbol{\mu}^{\mathrm{T}} \left(\dot{\mathbf{y}} - \mathbf{f}(t, \mathbf{y}, \boldsymbol{\rho})\right) \mathrm{d}t - \int_{t_0}^{t_F} \boldsymbol{\mu}^{\mathrm{T}} \left(\delta\dot{\mathbf{y}} - \mathbf{f}_{\check{\mathbf{y}}} \delta\mathbf{y} - \mathbf{f}_{\check{\boldsymbol{\rho}}} \delta\boldsymbol{\rho}\right) \mathrm{d}t \tag{65}$$

in which the notation explained in equation (40) has been used to include implicit dependencies on relative coordinates in the partial derivatives with respect to $\mathbf{y}$, $\dot{\mathbf{y}}$ and $\boldsymbol{\rho}$.

Integrating by parts in time and rearranging terms:

$$\delta\mathcal{L} = \left[\left(\mathbf{g}_{\check{\dot{\mathbf{y}}}} - \boldsymbol{\mu}^{\mathrm{T}}\right) \delta\mathbf{y}\right]_{t_0}^{t_F} + \int_{t_0}^{t_F} \left(\mathbf{g}_{\check{\mathbf{y}}} - \dot{\mathbf{g}}_{\check{\dot{\mathbf{y}}}} + \boldsymbol{\mu}^{\mathrm{T}} \mathbf{f}_{\check{\mathbf{y}}} + \dot{\boldsymbol{\mu}}^{\mathrm{T}}\right) \delta\mathbf{y}\, \mathrm{d}t + \int_{t_0}^{t_F} \left(\mathbf{g}_{\check{\boldsymbol{\rho}}} + \boldsymbol{\mu}^{\mathrm{T}} \mathbf{f}_{\check{\boldsymbol{\rho}}}\right) \delta\boldsymbol{\rho}\, \mathrm{d}t \tag{66}$$

The objective of the adjoint approach is to eliminate the need of calculating the derivatives of the states. In this case the objective is to nullify the expression multiplying $\delta\mathbf{y}$, leading to the following adjoint ODE systems:

$$\dot{\boldsymbol{\mu}} = -\mathbf{f}_{\check{\mathbf{y}}}^{\mathrm{T}} \boldsymbol{\mu} - \mathbf{g}_{\check{\mathbf{y}}}^{\mathrm{T}} + \dot{\mathbf{g}}_{\check{\dot{\mathbf{y}}}}^{\mathrm{T}} \tag{67a}$$

$$\boldsymbol{\mu}^{t_F} = \left[\mathbf{g}_{\check{\dot{\mathbf{y}}}}^{\mathrm{T}}\right]^{t_F} \tag{67b}$$

Nevertheless, these equations involve some problems related to the dependencies on $\dot{\mathbf{y}}$ and the derivative $\dot{\mathbf{g}}_{\check{\dot{\mathbf{y}}}}$, which could need the calculation of the jerks $\ddot{\mathbf{z}}^i$. These problems can be solved through the transformation of the dependencies on $\dot{\mathbf{y}}$ to implicit dependencies on $\mathbf{y}$ and $\boldsymbol{\rho}$ using (62b), resulting the final adjoint system:

$$\dot{\boldsymbol{\mu}} = -\mathbf{f}_{\check{\mathbf{y}}}^{\mathrm{T}} \left(\boldsymbol{\mu} + \mathbf{g}_{\check{\dot{\mathbf{y}}}}^{\mathrm{T}}\right) - \mathbf{g}_{\check{\mathbf{y}}}^{\mathrm{T}} \tag{68a}$$

$$\boldsymbol{\mu}^{t_F} = \mathbf{0} \tag{68b}$$

$$\mathbf{f}_{\check{\mathbf{y}}} = \hat{\mathbf{M}}^{-1}\left(\hat{\mathbf{Q}}_{\check{\mathbf{y}}} - \hat{\mathbf{M}}_{\check{\mathbf{y}}}\mathbf{f}\right) = \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ -\bar{\mathbf{M}}^{-1}\left(\bar{\mathbf{K}} + \bar{\mathbf{M}}_{\check{\mathbf{z}}^i}\dot{\mathbf{v}}\right) & -\bar{\mathbf{M}}^{-1}\bar{\mathbf{C}} \end{bmatrix} \tag{68c}$$

$$\mathbf{g}_{\check{\mathbf{y}}} = \begin{bmatrix} \mathbf{g}_{\check{\mathbf{z}}^i} & \mathbf{g}_{\dot{\check{\mathbf{z}}}^i} \end{bmatrix} \tag{68d}$$

$$\mathbf{g}_{\check{\mathbf{y}}} = \begin{bmatrix} \mathbf{0} & \mathbf{g}_{\ddot{\check{\mathbf{z}}}^i} \end{bmatrix} \tag{68e}$$

The objective function gradient can be calculated with the remaining terms:

$$\boldsymbol{\psi}'^{\mathrm{T}} = -\left[\mathbf{y}_{\check{p}}^{\mathrm{T}}\boldsymbol{\mu}\right]_{t_0} + \int_{t_0}^{t_F}\left(\mathbf{f}_{\check{p}}^{\mathrm{T}}\left(\boldsymbol{\mu} + \mathbf{g}_{\check{\mathbf{y}}}^{\mathrm{T}}\right) + \mathbf{g}_{\check{p}}^{\mathrm{T}}\right)\mathrm{d}t \tag{69a}$$

$$\mathbf{f}_{\check{p}} = \hat{\mathbf{M}}^{-1}\left(\hat{\mathbf{Q}}_{\check{p}} - \hat{\mathbf{M}}_{\check{p}}\mathbf{f}\right) = \begin{bmatrix} \mathbf{0} \\ \bar{\mathbf{M}}^{-1}\left(\bar{\mathbf{Q}}_{\check{p}} - \bar{\mathbf{M}}_{\check{p}}\ddot{\mathbf{z}}^i\right) \end{bmatrix} \tag{69b}$$

The implementation of the adjoint variable method requires the storage of information at each time step of the dynamic simulation. Once the simulation is complete, the adjoint equations can be solved backward in time, for which the same partial derivatives needed in the direct method have to be computed. The objective function gradient can be calculated from the values of the adjoint variables at each time step. This process is summarized in the pseudo-code 2.

Comparing (48) and (68), it can be inferred that both systems might be advantageous in different circumstances. The direct method leads to as many systems of equations as parameters, while the adjoint method requires to solve as many systems of equations as objective functions. In highly parameterized problems (like optimal control) the adjoint method would be the preferred option, while in other problems which require the evaluation of many objective functions (or optimization constraints), the direct method is desirable.

Algorithm 2 Adjoint sensitivity.

procedure SOLVEADJOINTSENSITIVITIES
  $t \leftarrow t_0$
  $\mathbf{z}_0, \dot{\mathbf{z}}_0 \leftarrow iniPosVel(\mathbf{z}_0^i, \dot{\mathbf{z}}_0^i)$
  $\mathbf{z}'_0, \dot{\mathbf{z}}'_0 \leftarrow iniSensitPosVel(\mathbf{z}_0^{i\prime}, \dot{\mathbf{z}}_0^{i\prime})$
  $\ddot{\mathbf{z}}_0 \leftarrow iniAcel(\mathbf{z}_0, \dot{\mathbf{z}}_0)$
  $\ddot{\mathbf{z}}'_0 \leftarrow iniSensitAcel(\mathbf{z}'_0, \dot{\mathbf{z}}'_0)$
  while $t <= t_{end}$ **do**
    $t = t + \Delta t$
    $\mathbf{z}, \dot{\mathbf{z}}, \ddot{\mathbf{z}} \leftarrow solveDynamics()$
  $\boldsymbol{\mu}^{t_F} \leftarrow \mathbf{0}$
  $\dot{\boldsymbol{\mu}}^{t_F} \leftarrow -\mathbf{f}_{\check{\mathbf{y}}}^{\mathrm{T}}\left(\boldsymbol{\mu} + \mathbf{g}_{\check{\mathbf{y}}}^{\mathrm{T}}\right) - \mathbf{g}_{\check{\mathbf{y}}}^{\mathrm{T}}$
  while $t >= t_0$ **do**
    $\boldsymbol{\mu}_i, \dot{\boldsymbol{\mu}}_i \leftarrow predictor\left(\boldsymbol{\mu}_{i+1}, \dot{\boldsymbol{\mu}}_{i+1}\right)$
    $\bar{\mathbf{M}}_{\check{\mathbf{z}}^i}\ddot{\mathbf{z}}^i, \bar{\mathbf{K}}, \bar{\mathbf{C}}, \bar{\mathbf{Q}}_{\check{p}} \leftarrow evalDerivatives()$
    $\boldsymbol{\mu}_i \leftarrow \left(\dot{\boldsymbol{\mu}}_i = -\mathbf{f}_{\check{\mathbf{y}}}^{\mathrm{T}}\left(\boldsymbol{\mu}_i + \mathbf{g}_{\check{\mathbf{y}}}^{\mathrm{T}}\right) - \mathbf{g}_{\check{\mathbf{y}}}^{\mathrm{T}}\right)$
    $\dot{\boldsymbol{\mu}}_i \leftarrow corrector\left(\boldsymbol{\mu}_i, \boldsymbol{\mu}_{i+1}, \dot{\boldsymbol{\mu}}_{i+1}\right)$
    $\boldsymbol{\psi}'^{\mathrm{T}} \leftarrow integrate\left(\int_{t_0}^{t_F}\left(\mathbf{f}_{\check{p}}^{\mathrm{T}}\left(\boldsymbol{\mu} + \mathbf{g}_{\check{\mathbf{y}}}^{\mathrm{T}}\right) + \mathbf{g}_{\check{p}}^{\mathrm{T}}\right)\mathrm{d}t\right)$
  $\boldsymbol{\psi}'^{\mathrm{T}} \leftarrow \boldsymbol{\psi}'^{\mathrm{T}} - \left[\mathbf{y}_{\check{p}}^{\mathrm{T}}\boldsymbol{\mu}\right]_{t_0}$
end

Extending the comparison to other direct and adjoint sensitivity formulations [14,16,35,36], the proposed methods result in simpler direct and adjoint equations (systems of ODEs rather than DAEs) in which the size of the system is proportional to the number of DoF rather than the number of dependent coordinates. On the contrary, the derivative calculation is more involved in the Matrix R method.

3.3. Gradient-based optimal design

The semi-recursive sensitivity methods presented in the previous subsections make possible the analytical evaluation of the gradient of one or multiple objective functions. The information provided by the gradients is of paramount importance in the optimization of multibody systems, and it can be used as input in one of the several long-known optimization algorithms that use gradients to calculate the direction and the size of the step at each optimization iteration [37].

Let us consider the following optimization problem as an example of application of the sensitivity methods presented:

$$\min_{\boldsymbol{\rho}} \quad \psi = \int_{t_0}^{t_F} g\left(\mathbf{z}, \dot{\mathbf{z}}, \ddot{\mathbf{z}}, \boldsymbol{\rho}, t\right)\mathrm{d}t \tag{70}$$

$$\text{s.t.} \quad \boldsymbol{\Phi}^e\left(\mathbf{z}, \dot{\mathbf{z}}, \ddot{\mathbf{z}}, \boldsymbol{\rho}, t\right) = \mathbf{0} \tag{71}$$

$$\boldsymbol{\Phi}^i\left(\mathbf{z}, \dot{\mathbf{z}}, \ddot{\mathbf{z}}, \boldsymbol{\rho}, t\right) \geq \mathbf{0} \tag{72}$$

where the superscript e stands for equality and i stands for inequality. For the evaluation of objective function and constraints gradient, it is possible to define an *extended* objective function $\boldsymbol{\psi}^{ext}$ such that:

$$\boldsymbol{\psi}^{ext} = \begin{bmatrix} \psi \\ \boldsymbol{\Phi}^e \\ \boldsymbol{\Phi}^i \end{bmatrix} \quad \Rightarrow \quad \left(\boldsymbol{\psi}^{ext}\right)' = \begin{bmatrix} \psi' \\ (\boldsymbol{\Phi}^e)' \\ \left(\boldsymbol{\Phi}^i\right)' \end{bmatrix} \tag{73}$$

The gradient of the new objective function $\boldsymbol{\psi}^{ext}$ can be computed using the sensitivity methods presented, thus objective function and constraint gradients can be evaluated in a single sensitivity calculation.

With this information, a general gradient-based optimization algorithm can be used for the minimization of (70), like those available in MATLAB in the context of the *fmincon* function, others available in languages like Python or Fortran (LBFGS-B [38]) or by means of in-house implementations of well-known algorithms [37].

4. Numerical experiments

The dynamic and sensitivity semi-recursive matrix R formulations described in this work, have been implemented in the MBSLIM multibody library [29] as general sensitivity formulations. The implementation has been tested with a five-bar linkage and in two maneuvers of a vehicle. The integration method used in all the experiments is the implicit trapezoidal rule, used on both the dynamics and sensitivity equations, and the time step for each numerical experiment has been selected as the maximum time step that delivers a response within the range of accuracy specified in each benchmark problem description [39]. The experiments have been conducted in an Intel Core i7-8700 CPU at 3.20 GHz. The methods have been coded in Fortran using the latest features of the standard Fortran 2018, with the Fortran Intel Parallel Studio XE 2018 as compiler on a Windows 10 operating system. Although the equations presented are valid for a generic reference point, the results included in this section correspond to the RTdyn0 semi-recursive method, with the center of mass of each body as reference point.

The semi-recursive Matrix R sensitivity methods have been implemented according to the following guidelines:

- Recursive procedures: recursive procedures have been exploited in both dynamics and derivative calculations whenever possible in order to minimize the number of algebraic operations.
- Storage and reuse of terms: there are some terms that appear repeatedly in the sensitivity equations and in the derivative expressions, thus the most efficient approach is to compute them once, store and then reuse them when they are needed.
- Combination of sparse and dense algebra: independent coordinate formulations like the semi-recursive Matrix R usually lead to small and very dense problems that can be solved with dense solvers. Here, both dynamic and sensitivity systems of equations are solved using LAPACK (*Linear Algebra PACKage*). On the contrary, sparse algebra is used in kinematic problems like in the evaluation of Matrix $\mathbf{R}^{\boldsymbol{\Phi}}$ and in the storage of sparse structures like the derivatives of kinematic constraints.
- Symmetry: the symmetry of some dynamic and kinematic terms can be harnessed for saving calculations and storing terms. For example, only the derivatives of the diagonal and the upper triangular part of the mass matrix have to be computed and stored, and then it is possible to operate with them using appropriate routines.

4.1. Five-bar mechanism

The sensitivity formulations introduced in this document are firstly tested in the five-bar benchmark problem included in the IFToMM benchmark library [39] under the title “Sensitivity analysis of a five-bar mechanism”. There, the reader can find extensive information about topology, geometry, initial position and velocity, simulation time, objective functions and sensitivity parameters.

The mechanism, displayed in Fig. 1, is composed of five bars (one of them fixed) linked by means of five revolute joints with parallel axes, which converts it in a 2-DoF linkage.

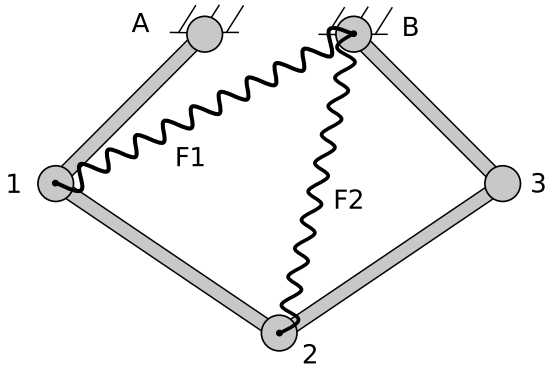

Fig. 1. Five-bar linkage.

In this experiment, the goal is to evaluate the gradient of the following array of objective functions:

$$\boldsymbol{\psi} = \begin{bmatrix} \psi_1 & \psi_2 & \psi_3 \end{bmatrix}^{\mathrm{T}} \tag{74a}$$

$$\psi_1 = \int_{t_0}^{t_F} \left(\mathbf{r}_2 - \mathbf{r}_{20}\right)^{\mathrm{T}} \left(\mathbf{r}_2 - \mathbf{r}_{20}\right) \mathrm{d}t\,, \qquad \psi_2 = \int_{t_0}^{t_F} \dot{\mathbf{r}}_2^{\mathrm{T}} \dot{\mathbf{r}}_2 \mathrm{d}t\,, \qquad \psi_3 = \int_{t_0}^{t_F} \ddot{\mathbf{r}}_2^{\mathrm{T}} \ddot{\mathbf{r}}_2 \mathrm{d}t\,. \tag{74b}$$

wherein $\mathbf{r}_2$ and $\mathbf{r}_{20}$ represent the instant and initial position of the point identified as 2 in Fig. 1, respectively.

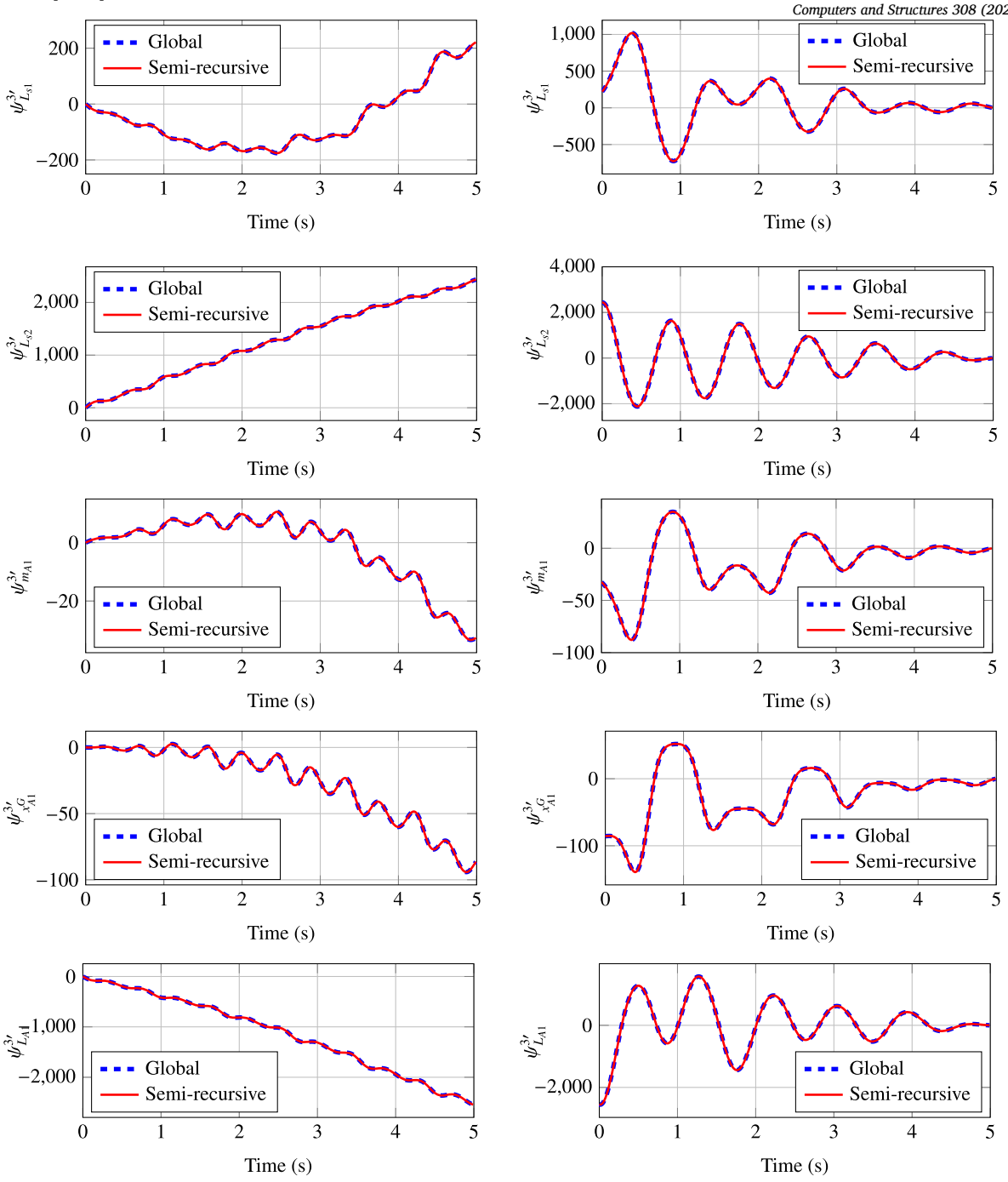

Fig. 2. Evolution in time of the gradient of ψ^3 evaluated with direct (left) and adjoint (right) sensitivity formulations.

The gradient of each objective function is calculated with respect to the parameters below:

$$\boldsymbol{\rho} = \begin{bmatrix} L_{\mathrm{s1}} & L_{\mathrm{s2}} & m_{\mathrm{A1}} & x^{\mathrm{G}}_{\mathrm{A1}} & L_{\mathrm{A1}} \end{bmatrix} \tag{75}$$

being L_{s1} and L_{s2} the natural lengths of the springs, and m_{A1}, $x^{\mathrm{G}}_{\mathrm{A1}}$ and L_{A1} the mass, X-local coordinate of the center of mass and length of the bar between vertices A and 1, respectively.

The sensitivity analysis of a 5-seconds dynamic simulation of this linkage under the action of gravity and spring forces has been performed with the semi-recursive Matrix R sensitivity formulations detailed in this document. The accuracy of the results is compared in Table 1 against the reference provided in the IFToMM benchmark library [39]. Fig. 2 includes a comparison between each component of the gradient of the objective function ψ^3 evaluated with global and semi-recursive Matrix R direct and adjoint sensitivity formulations. It can be observed that, since the set of independent coordinates used is identical in global and semi-recursive methods, the results coincide in both cases. Moreover, Fig. 2 shows how the value of the gradient at time $t = 5s$ in direct methods (left column) matches the results at times $t = 0s$ in adjoint methods (right column) due to the backward integration of the gradient in the adjoint case.

Table 1
Objective functions gradient evaluated with different sensitivity formulations.

	Direct SR-Matrix R	Adjoint SR-Matrix R	Reference
$(\psi^1)'_{L_{s1}}$	-4.2288	-4.2288	-4.2288
$(\psi^1)'_{L_{s2}}$	3.2116	3.2116	3.2116
$(\psi^1)'_{m_{A1}}$	0.31866	0.31866	0.31866
$(\psi^1)'_{x^G_{A1}}$	0.44235	0.44235	0.44235
$(\psi^1)'_{L_{A1}}$	3.3598	3.3599	3.3598
$(\psi^2)'_{L_{s1}}$	-15.452	-15.452	-15.452
$(\psi^2)'_{L_{s2}}$	50.309	50.308	50.309
$(\psi^2)'_{m_{A1}}$	0.97017	0.97016	0.97012
$(\psi^2)'_{x^G_{A1}}$	0.74569	0.74569	0.74560
$(\psi^2)'_{L_{A1}}$	-27.359	-27.358	-27.359
$(\psi^3)'_{L_{s1}}$	221.64	221.65	221.64
$(\psi^3)'_{L_{s2}}$	2436.6	2436.6	2436.6
$(\psi^3)'_{m_{A1}}$	-32.497	-32.498	-32.497
$(\psi^3)'_{x^G_{A1}}$	-85.658	-85.658	-85.657
$(\psi^3)'_{L_{A1}}$	-2546.6	-2546.6	-2546.6

This example aims to check the accuracy of the methods rather than their efficiency, which is assessed in the following more complex numerical experiment.

4.2. Buggy vehicle

The second numerical experiment employed to test the accuracy and efficiency of the sensitivity methods presented is a buggy vehicle with articulated suspensions and with tire-ground interaction modeled by means contact-frictional tire forces, displayed in Fig. 3. The complete model description along with the initial conditions, simulation time, forces description, objective functions and sensitivity parameters is documented in the IFToMM benchmark library [39] in two separate benchmark problems entitled “Sensitivity analysis of a step descent maneuver of a buggy vehicle” and “Sensitivity analysis of a double lane change maneuver of a buggy vehicle”.

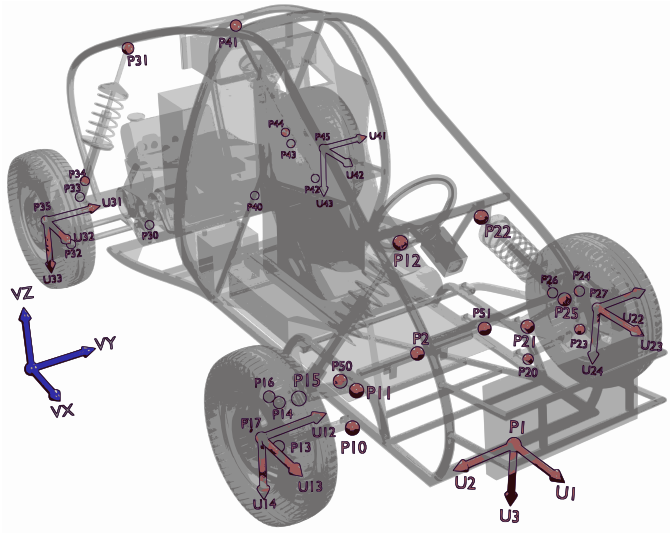

Fig. 3. Buggy vehicle with the points and vectors defining the model.

In the first maneuver, the vehicle moves in a straight line with its steering blocked and descents a step of 1 cm placed at a distance of 5.5 m from the origin. Considering a forward initial linear speed of $3\,\mathrm{m\,s^{-1}}$, the step is reached approximately at $t = 2.0\,\mathrm{s}$. The 4.5-second simulation is executed with the dynamic formulation presented in this document with a time step of 1 ms.

As objective function, a measurement of the chassis accelerations is considered:

$$\psi = \int_{t_0}^{t_F} \ddot{r}_{1_z}^2 \, \mathrm{d}t \tag{76}$$

being $\ddot{r}_{1_z}$ the Z (vertical) component of the acceleration of a point in the front of the chassis.

Table 2
Objective function gradient for the step descent maneuver.

	Semi-recursive methods		Global methods		Reference
	Direct Matrix R	Adjoint Matrix R	Direct Matrix R	Adjoint Matrix R	
ψ'_{k_f}	2.0483×10^{-4}	2.0493×10^{-4}	2.0538×10^{-4}	2.0547×10^{-4}	2.06×10^{-4}
ψ'_{c_f}	9.3358×10^{-4}	9.3358×10^{-4}	9.3357×10^{-4}	9.3357×10^{-4}	9.34×10^{-4}
ψ'_{k_r}	-3.8189×10^{-5}	-3.8252×10^{-5}	-3.8138×10^{-5}	-3.8202×10^{-5}	-3.90×10^{-5}
ψ'_{c_r}	7.5319×10^{-4}	7.5319×10^{-4}	7.5319×10^{-4}	7.5319×10^{-4}	7.53×10^{-4}
ψ'_{m_c}	4.0976×10^{-2}	4.0979×10^{-2}	4.0895×10^{-2}	4.0899×10^{-2}	4.06×10^{-2}

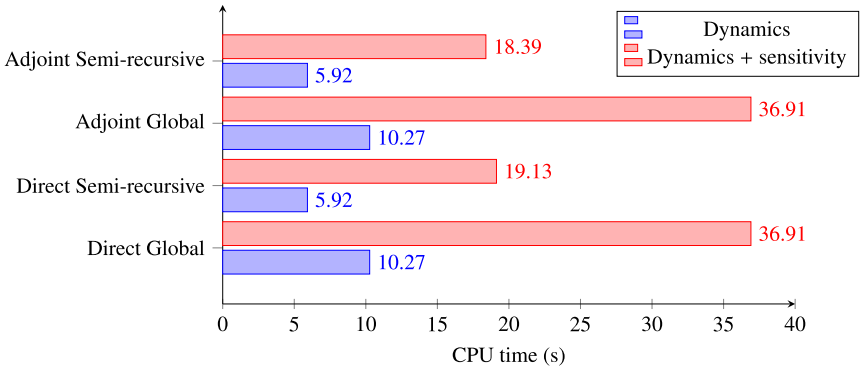

Fig. 4. CPU times of objective function and gradient evaluations for the step descent maneuver.

The sensitivity parameters include:

$$\boldsymbol{\rho} = \begin{bmatrix} k_f & c_f & k_r & c_r & m_c \end{bmatrix} \tag{77}$$

with k_f, c_f, k_r and c_r representing the stiffness and damping coefficients of the frontal and rear suspensions, respectively, and m_c identifying the mass of the chassis.

In order to assess the efficiency of the semi-recursive sensitivity formulations presented in this document, they are compared with equivalent Matrix R sensitivity formulations in natural (or fully-Cartesian) coordinates. It is important to remark that the independent variables are identical in natural and relative coordinate models, thus matching results could be expected (despising numerical errors). Therefore, this experiment allows to evaluate the computational gains of using semi-recursive methods instead of global methods.

In Table 2, the results of the step descent objective function gradient are presented for different formulations. The differences observed are due to the collision in the step descent, which involves impact forces which require a very low time step to be perfectly simulated. In fact, the level of convergence between formulations and reference response can be increased by decreasing the time step. However, no time step reduction has been executed since all the results presented fulfill the error criteria required by the benchmark problem.

Fig. 4 presents the efficiency for the direct and adjoint, global and semi-recursive Matrix R formulations, showing that the semi-recursive formulation saves around 48.2% of CPU-time in the case of the direct method and up to 50.2% in the case of the adjoint, for this particular example and maneuver. This figure also reflects that semi-recursive methods outperform global methods in both dynamics and sensitivities, representing the time devoted to sensitivity calculations around 31% of the total time in semi-recursive methods (32.2% in adjoint and 30.1% in direct sensitivities) and 27.8% in global methods. Apart from the dynamics calculation, almost all the computational time in the four sensitivity formulations compared is spent in the evaluation of derivatives, while the time needed for the solution of the sensitivity equations is almost negligible. This is explained by the fact that direct and adjoint systems of equations are linear and of very small size.

The second maneuver consists in a 12-second double lane change maneuver as described in the benchmark problem "Sensitivity analysis of a double lane change maneuver of a buggy vehicle" (see [39]). In this case, the objective function measures the square of the roll rate of the chassis,

$$\psi = \int_{t_0}^{t_F} \dot{\phi}^2 \mathrm{d}t \tag{78}$$

while the set of parameters is preserved from the step descent maneuver, declared in (77).

The time step for both dynamics and sensitivity analysis is increased in this maneuver to 15 ms due to the smoothness of the motion (no abrupt force variation appears).

In Table 3 and Fig. 5, the accuracy and efficiency of the sensitivity methods are assessed, respectively. These data exhibit the correctness and validity of the expressions presented as well as the higher efficiency of the semi-recursive methods.

Fig. 5 confirms the speed-ups of semi-recursive methods for this maneuver, around 49.3% of CPU-time for the direct semi-recursive compared to the direct global and around 50.1% in the case of the adjoint semi-recursive compared to the adjoint global. It is worth to mention that the joint calculation of dynamics and sensitivities is executed in real time, with a factor of 1.6 in global methods and 3.1 in semi-recursive methods. This highlights the efficiency of the methods presented and makes patent the advantages of analytical differentiation in sensitivity calculations.

It is worth mentioning that, even if the topological semi-recursive Matrix R sensitivity formulations are faster than their global counterparts, the comparison with the global and semi-recursive ALI3-P sensitivity formulations presented in [33], reveals that Matrix R formulations are slower

Table 3
Objective function gradient for the double lane change maneuver.

	Semi-recursive methods		Global methods		Reference
	Direct Matrix R	Adjoint Matrix R	Direct Matrix R	Adjoint Matrix R	
ψ'_{k_t}	3.9538×10^{-8}	3.9538×10^{-8}	3.9533×10^{-8}	3.9533×10^{-8}	3.96×10^{-8}
ψ'_{c_t}	-1.2736×10^{-8}	-1.2736×10^{-8}	-1.2737×10^{-8}	-1.2737×10^{-8}	-1.28×10^{-8}
ψ'_{k_r}	-5.1100×10^{-8}	-5.1100×10^{-8}	-5.1100×10^{-8}	-5.1100×10^{-8}	-5.11×10^{-8}
ψ'_{c_r}	-4.1108×10^{-8}	-4.1108×10^{-8}	-4.1108×10^{-8}	-4.1108×10^{-8}	-4.12×10^{-8}
ψ'_{m_c}	2.5574×10^{-6}	2.5574×10^{-6}	2.5580×10^{-6}	2.5580×10^{-6}	2.56×10^{-6}

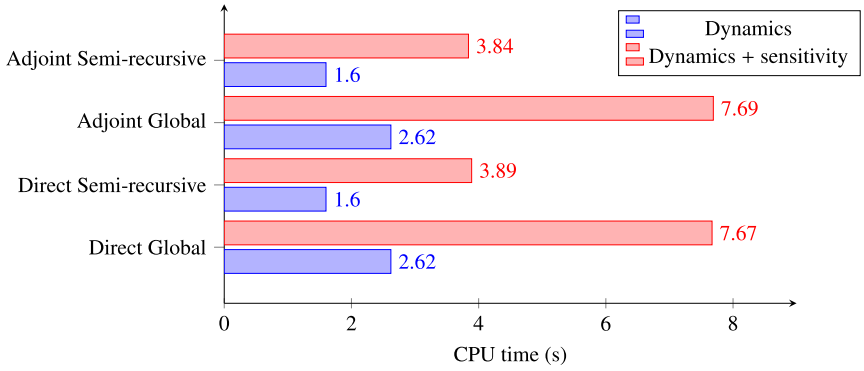

Fig. 5. CPU times of objective function and gradient evaluations for the double lane change maneuver.

than ALI3-P for the examples presented. The same conclusion can likely be extended to other multibody systems with similar sizes, coordinates and degrees of freedom.

5. Conclusions

In this document, the combination of the Matrix R constraint enforcement scheme with topological semi-recursive methods is reviewed. The classical Matrix R formulation is extended to support degrees of freedom which are not a subset of the dependent coordinates, thus the method presented is more general and cannot be considered equivalent to a coordinate partitioning method. Constrained and unconstrained kinematic problems in joint coordinates are revisited as an introductory step for the generation of semi-recursive Matrix R equations of motion.

The sensitivity analysis of the semi-recursive dynamic formulation introduced has been derived using a direct differentiation method and an adjoint variable method, reaching a set of compact expressions involving derivatives with respect to relative coordinates and parameters.

A detailed description of the main derivatives required in both sensitivity methods has been provided. Additionally, some new differentiation rules and notation have been declared with the aim of keeping sensitivity equations as concise and clear as possible.

The sensitivity methods have been implemented in the MBSLIM general purpose multibody library, in which each derivative required has been analytically calculated, programmed and tested. The sensitivity methods have been evaluated in a five-bar linkage and in two maneuvers of a buggy vehicle with articulated suspensions. Results show an important computational gain of semi-recursive methods with respect to the sensitivity analysis of Matrix R in natural coordinates for the same level of accuracy.

CRediT authorship contribution statement

Álvaro López Varela: Writing – original draft, Validation, Investigation, Formal analysis. **Daniel Dopico Dopico:** Writing – review & editing, Supervision, Resources, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Conceptualization. **Alberto Luaces Fernández:** Software, Resources, Methodology, Investigation, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

The support of the Spanish Ministry of Economy and Competitiveness (MINECO) under project DPI2016-81005-P and the support of Spanish Ministry of Science and Innovation (MICINN) under project PID2020-120270GB-C21 are greatly acknowledged. Furthermore, the first author would like to greatly acknowledge the support of MINECO by means of the doctoral research contract BES-2017-080727, co-financed by the European Union through the ESF program, and Centro Mixto de Investigación UDC – Navantia Astillero 5.0. El Astillero del Futuro.

Appendix A. Notation

Consider a model defined by $\mathbf{z} \in \mathbb{R}^n$ relative coordinates, with $\mathbf{q}(\mathbf{z}) \in \mathbb{R}^{n_q}$ natural coordinates referred to points and vectors and with a set $\boldsymbol{\rho} \in \mathbb{R}^p$ of system parameters. Let us define a new differentiation rule of a generic function $f\left(\mathbf{z}, \dot{\mathbf{z}}, \ddot{\mathbf{z}}, \mathbf{q}(\mathbf{z}), \dot{\mathbf{q}}(\mathbf{z}, \dot{\mathbf{z}}), \ddot{\mathbf{q}}(\mathbf{z}, \dot{\mathbf{z}}, \ddot{\mathbf{z}}), \boldsymbol{\rho}\right)$ as:

$$f_{\hat{\mathbf{x}}} = f_{\mathbf{q}}\mathbf{q}_{\mathbf{x}} + f_{\dot{\mathbf{q}}}\dot{\mathbf{q}}_{\mathbf{x}} + f_{\ddot{\mathbf{q}}}\ddot{\mathbf{q}}_{\mathbf{x}} + f_{\mathbf{x}} \tag{A.1}$$

being $\mathbf{x}$ any of the dependencies of function f, $\dot{()}$ a first temporal derivative and $\ddot{()}$ a second time derivative. In brief, the *hat* notation in a subscript indicates that the derivative is evaluated considering all the dependencies of the function on natural coordinates and the dependencies of natural coordinates on $\mathbf{x}$.

As an example, let us consider the vector of kinematic constraints as the function f. The derivatives with respect to the relative coordinates, natural coordinates and system parameters according to (A.1) are:

$$\boldsymbol{\Phi}_{\hat{\mathbf{z}}} = \boldsymbol{\Phi}_{\mathbf{q}}\mathbf{q}_{\mathbf{z}} + \boldsymbol{\Phi}_{\mathbf{z}} \tag{A.2a}$$

$$\boldsymbol{\Phi}_{\hat{\dot{\mathbf{q}}}} = \boldsymbol{\Phi}_{\mathbf{q}}\mathbf{q}_{\dot{\mathbf{q}}} + \boldsymbol{\Phi}_{\dot{\mathbf{q}}} \tag{A.2b}$$

$$\boldsymbol{\Phi}_{\hat{\boldsymbol{\rho}}} = \boldsymbol{\Phi}_{\mathbf{q}}\mathbf{q}_{\boldsymbol{\rho}} + \boldsymbol{\Phi}_{\boldsymbol{\rho}} \tag{A.2c}$$

Observe that the derivatives with respect to $\dot{\mathbf{z}}$, $\ddot{\mathbf{z}}$, $\dot{\mathbf{q}}$ and $\ddot{\mathbf{q}}$ are null in this case and have been removed from (A.2a).

The total derivative of a function f with respect to the set of system parameters can be calculated as:

$$f' = \left(f_{\mathbf{q}}\mathbf{q}_{\mathbf{z}} + f_{\dot{\mathbf{q}}}\dot{\mathbf{q}}_{\mathbf{z}} + f_{\ddot{\mathbf{q}}}\ddot{\mathbf{q}}_{\mathbf{z}} + f_{\mathbf{z}}\right)\mathbf{z}' + \left(f_{\dot{\mathbf{q}}}\dot{\mathbf{q}}_{\dot{\mathbf{z}}} + f_{\ddot{\mathbf{q}}}\ddot{\mathbf{q}}_{\dot{\mathbf{z}}} + f_{\dot{\mathbf{z}}}\right)\dot{\mathbf{z}}' + \left(f_{\ddot{\mathbf{q}}}\ddot{\mathbf{q}}_{\ddot{\mathbf{z}}} + f_{\ddot{\mathbf{z}}}\right)\ddot{\mathbf{z}}' + f_{\hat{\boldsymbol{\rho}}} \tag{A.3}$$

being

$$f' = \frac{\mathrm{d}f}{\mathrm{d}\boldsymbol{\rho}}, \quad \mathbf{z}' = \frac{\mathrm{d}\mathbf{z}}{\mathrm{d}\boldsymbol{\rho}}, \quad \dot{\mathbf{z}}' = \frac{\mathrm{d}\dot{\mathbf{z}}}{\mathrm{d}\boldsymbol{\rho}}, \quad \ddot{\mathbf{z}}' = \frac{\mathrm{d}\ddot{\mathbf{z}}}{\mathrm{d}\boldsymbol{\rho}} \tag{A.4}$$

Using (A.1), it is possible to define it more concisely:

$$f' = f_{\hat{\mathbf{z}}}\mathbf{z}' + f_{\hat{\dot{\mathbf{z}}}}\dot{\mathbf{z}}' + f_{\hat{\ddot{\mathbf{z}}}}\ddot{\mathbf{z}}' + f_{\hat{\boldsymbol{\rho}}} \tag{A.5}$$

as long as the intermediate dependencies on $\mathbf{q}$, $\dot{\mathbf{q}}$ and $\ddot{\mathbf{q}}$ are included in the new differentiation rule.

Data availability

Data will be made available on request.

References

[1] Cuadrado J, Dopico D, Naya M, Gonzalez M. Penalty, semi-recursive and hybrid methods for mbs real-time dynamics in the context of structural integrators. Multibody Syst Dyn 2004;12(2):117–32. https://doi.org/10.1023/B:MUBO.0000044421.04658.de.
[2] García de Jalón J, Bayo E. Kinematic and dynamic simulation of multibody systems: the real-time challenge. New York USA: Springer-Verlag; 1994.
[3] Callejo A, García de Jalón J, Luque P, Mántaras DA. Sensitivity-based, multi-objective design of vehicle suspension systems. J Comput Nonlinear Dyn 2015;10(3):031008. https://doi.org/10.1115/1.4028858.
[4] Gutiérrez-López M, Callejo A, García de Jalón J. Computation of independent sensitivities using Maggi's formulation. 2012.
[5] Martins J, Sturdza P, Alonso J. The complex-step derivative approximation. ACM Trans Math Softw 2003;29(3):245–62. https://doi.org/10.1145/838250.838251.
[6] Maly T, Petzold LR. Numerical methods and software for sensitivity analysis of differential-algebraic systems. Appl Numer Math 1996;20(1):57–79. https://doi.org/10.1016/0168-9274(95)00117-4.
[7] Feehery W, Tolsma J, Barton P. Efficient sensitivity analysis of large-scale differential-algebraic systems. Appl Numer Math 1997;25(1):41–54. https://doi.org/10.1016/S0168-9274(97)00050-0.
[8] Callejo A, Narayanan S, García de Jalón J, Norris B. Performance of automatic differentiation tools in the dynamic simulation of multibody systems. Adv Eng Softw 2014;73:35–44. https://doi.org/10.1016/j.advengsoft.2014.03.002.
[9] Dürrbaum A, Klier W, Hahn H. Comparison of automatic and symbolic differentiation in mathematical modeling and computer simulation of rigid-body systems. Multibody Syst Dyn 2002;7(4):331–55. https://doi.org/10.1023/A:1015523018029.
[10] Ambrósio J, Neto M, Leal R. Optimization of a complex flexible multibody systems with composite materials. Multibody Syst Dyn 2007;18(2):117–44. https://doi.org/10.1007/s11044-007-9086-y.
[11] Callejo A, Dopico D. Direct sensitivity analysis of multibody systems: a vehicle dynamics benchmark. J Comput Nonlinear Dyn 2019;14(2):021004. https://doi.org/10.1115/1.4041960.
[12] Haug EJ, Neel K, Krishnaswami P. Design sensitivity analysis and optimization of dynamically driven systems. In: Computer aided analysis and optimization of mechanical system dynamics, vol. 9. 1984. p. 555–635.
[13] Mani N, Haug E. Singular value decomposition for dynamic system design sensitivity analysis. Eng Comput 1985;1(2):103–9. https://doi.org/10.1007/BF01200068.
[14] Haug EJ. Design sensitivity analysis of dynamic systems. In: Mota Soares CA, editor. Computer aided optimal design: structural and mechanical systems. Berlin, Heidelberg: Springer Berlin Heidelberg; 1987. p. 705–55.
[15] Ashrafiuon H, Mani N. Analysis and optimal design of spatial mechanical systems. J Mech Des, Trans ASME 1990. https://doi.org/10.1115/1.2912593.
[16] Bestle D, Seybold J. Sensitivity analysis of constrained multibody systems. Arch Appl Mech 1992;62(3):181–90. https://doi.org/10.1007/BF00787958.
[17] Callejo A, Sonneville V, Bauchau O. Discrete adjoint method for the sensitivity analysis of flexible multibody systems. J Comput Nonlinear Dyn 2019;14(2). https://doi.org/10.1115/1.4041237.
[18] Dopico Dopico D, López Varela Á, Luaces Fernández A. Augmented Lagrangian index-3 semi-recursive formulations with projections. Multibody Syst Dyn 2020. https://doi.org/10.1007/s11044-020-09771-9.
[19] Kim SS, Vanderploeg MJ. A general and efficient method for dynamic analysis of mechanical systems using velocity transformations. J Mech Des, Trans ASME 1986;108(2):176–82. https://doi.org/10.1115/1.3260799.
[20] Nikravesh P, Gim G. Systematic construction of the equations of motion for multibody systems containing closed kinematic loops. J Mech Des, Trans ASME 1993;115(1):143–9. https://doi.org/10.1115/1.2919310.
[21] Rodriguez JI, Jimenez JM, Funes FJ, de Jalón JG. Recursive and residual algorithms for the efficient numerical integration of multi-body systems. Multibody Syst Dyn 2004;11(4):295–320.

[22] García de Jalón J, Alvarez E, De Ribera F, Rodriguez I, Funes F. A fast and simple semi-recursive formulation for multi-rigid-body systems. Comput Methods Appl Sci 2005;2:1–23. https://doi.org/10.1007/1-4020-3393-1_1.
[23] Funes FJ, García de Jalón J. An efficient dynamic formulation for solving rigid and flexible multibody systems based on semirecursive method and implicit integration. J Comput Nonlinear Dyn 2016;11(5). https://doi.org/10.1115/1.4032246.
[24] Pan Y, Dai W, Xiong Y, Xiang S, Mikkola A. Tree-topology-oriented modeling for the real-time simulation of sedan vehicle dynamics using independent coordinates and the rod-removal technique. Mech Mach Theory 2020;143. https://doi.org/10.1016/j.mechmachtheory.2019.103626.
[25] Avello A, Jiménez J, Bayo E, García de Jalón J. A simple and highly parallelizable method for real-time dynamic simulation based on velocity transformations. Comput Methods Appl Mech Eng 1993;107(3):313–39. https://doi.org/10.1016/0045-7825(93)90072-6.
[26] Cuadrado J, Dopico D. A hybrid global-topological real-time formulation for multibody systems. Volume 5: 19th Biennial Conference on Mechanical Vibration and Noise, Parts A, B, and C of International Design Engineering Technical Conferences and Computers and Information in Engineering Conference. Available from: https://asmedigitalcollection.asme.org/IDETC-CIE/proceedings-pdf/IDETC-CIE2003/37033/115/2597128/115_1.pdf. https://doi.org/10.1115/DETC2003/VIB-48315.
[27] Cuadrado J, Dopico D, Gonzalez M, Naya M. A combined penalty and recursive real-time formulation for multibody dynamics. J Mech Des, Trans ASME 2004;126(4):602–8. https://doi.org/10.1115/1.1758257.
[28] Jaiswal S, Rahikainen J, Khadim Q, Sopanen J, Mikkola A. Comparing double-step and penalty-based semirecursive formulations for hydraulically actuated multibody systems in a monolithic approach. Multibody Syst Dyn 2021;52(2):169–91. https://doi.org/10.1007/s11044-020-09776-4.
[29] Dopico D, Luaces A, Lugrís U, Saura M, González F, Sanjurjo E, et al. MBSLIM: multibody systems in laboratorio de ingeniería mecánica. Available from: http://lim.ii.udc.es/MBSLIM, 2009–2016.
[30] García de Jalón J, Jiménez JM, Avello A, Martín F, Cuadrado J. Real time simulation of complex 3-D multibody systems with realistic graphics. In: Haug EJ, Deyo RC, editors. Real-time integration methods for mechanical system simulation. Berlin, Heidelberg: Springer Berlin Heidelberg; 1991. p. 265–92.
[31] García de Jalón J, Unda J, Avello A. Natural coordinates for the computer analysis of multibody systems. Comput Methods Appl Mech Eng 1986;56(3):309–27. https://doi.org/10.1016/0045-7825(86)90044-7.
[32] García de Jalón J, Callejo A, Hidalgo A, Gutierrez M. Efficient solution of Maggi's equations. In: Proceedings of the ASME design engineering technical conference 4 (PARTS A AND B); 2011. p. 115–24.
[33] López Varela A, Dopico Dopico D, Luaces Fernández A. Augmented Lagrangian index-3 semi-recursive formulations with projections: direct sensitivity analysis. Multibody Syst Dyn 2023. https://doi.org/10.1007/s11044-023-09928-2.
[34] Dopico D, Zhu Y, Sandu A, Sandu C. Direct and adjoint sensitivity analysis of ordinary differential equation multibody formulations. J Comput Nonlinear Dyn 2014;10(1):1–8. https://doi.org/10.1115/1.4026492.
[35] Dopico D, González F, Luaces A, Saura M, García-Vallejo D. Direct sensitivity analysis of multibody systems with holonomic and nonholonomic constraints via an index-3 augmented Lagrangian formulation with projections. Nonlinear Dyn May 2018. https://doi.org/10.1007/s11071-018-4306-y.
[36] Dopico D, Sandu A, Sandu C. Adjoint sensitivity index-3 augmented Lagrangian formulation with projections. Mech Based Des Struct Mach 2021. https://doi.org/10.1080/15397734.2021.1890614.
[37] Nocedal J, Wright S. Numerical optimization. 2nd edition. New York: Springer-Verlag; 2006.
[38] Morales J, Nocedal J. Remark on "Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound constrained optimization". ACM Trans Math Softw 2011;38(1). https://doi.org/10.1145/2049662.2049669.
[39] I. T. C. for Multibody Dynamics. Library of computational benchmark problems. Available from: http://www.iftomm-multibody.org/benchmark, 2014.