Symbolic Multibody Methods for Real-Time Simulation of Railway Vehicles.

Javier Ros · Aitor Plaza · Xabier Iriarte · Jesús María Pintor

Abstract In this work, recently developed state-of-the-art *symbolic multibody* methods are tested to accurately model a complex railway vehicle. The model is generated using a symbolic implementation of the principle of the virtual power. Creep forces are modeled using a direct symbolic implementation of the standard linear Kalker model. No simplifications, such as base parameter reduction, partial-linearization or look-up tables for contact kinematics, are used. An Implicit-Explicit integration scheme is proposed to efficiently deal with the stiff creep dynamics. Hard real-time performance is achieved: the CPU time required for a very stable 1 ms integration time step is 256 μs.

J. Ros
Department of Mechanical Engineering
Public University of Navarre
Campus de Arrosadia, 31006 Pamplona, Navarre, Spain
Tel.: +34-948166215
Fax: +34-948169099
E-mail: jros@unavarra.es

X. Iriarte
Department of Mechanical Engineering
Public University of Navarre
Campus de Arrosadia, 31006 Pamplona, Navarre, Spain
E-mail: Xabier.Iriarte@unavarra.es

A. Plaza
Department of Mechanical Engineering
Public University of Navarre
Campus de Arrosadia, 31006 Pamplona, Navarre, Spain
E-mail: Aitor.Plaza@unavarra.es

J.M. Pintor
Department of Mechanical Engineering
Public University of Navarre
Campus de Arrosadia, 31006 Pamplona, Navarre, Spain
E-mail: txma@unavarra.es

1 Introduction

Multibody system dynamics is a well established discipline in the context of railway vehicle design. It is used for new concept performance evaluation, stability, lifetime, wear prediction, etc. In general it is desirable to be able to do these analyses as fast as possible. In particular, due to the huge number of computations required, computational performance can be very important when dealing with design optimization. Nevertheless, these tasks do not demand strict real-time performance.

The computational power available on today's off-the-shelf computers is getting closer to allowing real-time direct numerical simulation of complex railway vehicle models. This in turn opens up new possibilities that can greatly benefit the design, safety, and model based predictive maintenance in the railway field. Most important applications can be considered to be HiL (Hardware in the Loop) on the design side, and on-line filtering techniques (Kalman filter alike) in the context of safety, and model based predictive maintenance. These developments usually run on the heels of previous work done in the context of vehicle dynamics.

Symbolic multibody models have been demonstrated to be an effective tool for the modeling of general multibody systems. In particular they have been shown to be very fast when using recursive $O(n^3)$ formulations [1], that in turn require a parametrization based on relative coordinates. The main challenges are related to the enormous size of the expressions that the symbolic processor needs to deal with as this can limit the size of the problem to be analyzed. Recently, in [2] the authors presented a symbolic multibody library in which the concept of recursivity is extended so that it is no longer based on the formulation but, instead, on the parametrization level. This is achieved by the definition of an algebra that includes the typical mechanics operators (position vector, velocity,...) and that deals with the recursivity that might be embedded into the parametrization. The typical tree-shaped body structure is replaced by a tree structure for points and another one for bases. This gives a fine grained control of the recursivity that, in this way, can be different for both tree structures. No limitation is imposed on the parametrization of the system. As a consequence the library allows the implementation of arbitrary dynamics formulations. Atomization (optimization of symbolic expression representation) is embedded into the library from the very bottom upwards. This alleviates the symbolic manipulation of expressions and lowers their complexity to a minimum. This in turn allows one to obtain optimal atomizations that minimize the computational complexity and increases the size of the problems that is possible to analyze.

This article aims to evaluate the feasibility of real-time numerical simulation of a complex locomotive multibody model using state of the art symbolic modeling techniques referred to above. For this study, the FEVE 3000 locomotive was used. A generic (spline based) definition for the contact surfaces of the wheels and rails, including irregularities, is used. Based on these, creep forces are modeled using a direct symbolic implementation of the standard linear Kalker model without simplifications of any kind. Bodies and rail are considered rigid with three-dimensional kinematics. No further simplifications such as contact coordinate removal [3], pre-calculated tables [4], partial linearization [5] or base parameter reduction [6] are presented.

To that end, the modeling is done based on the multibody system symbolic library `lib_3D_MEC_GiNaC` [2], using a relative parametrization with respect to the inertial reference.

The paper is structured as follows: In section 2 the symbolic methods used in this work are briefly described. In section 3 the description of the modeled system is presented. In section 4 the most interesting details of the multibody modeling are presented. In section 5 the results of the simulations are shown and discussed. Finally in section 6 the main conclusions of this work are presented.

2 Symbolic modeling procedures

Simply stated, the main goal of the symbolic modeling of multibody systems can be defined as:

"*to obtain a set of functions that allow for the determination of the position, velocities and accelerations of all the bodies of the system*".

Special symbolic procedures are required if a real-time-capable fast multibody model is desired. The main features of these procedures, as proposed in [7], are summarized below.

2.1 Parametrization and system topology.

In order to model the multibody system a set of geometric parameters $\mathbf{p}$ and generalized coordinates $\mathbf{q}$, along with their associated velocity $\dot{\mathbf{q}}$ and generalized accelerations $\ddot{\mathbf{q}}$ are defined. The authors propose to split up the classical tree-shaped body structure into two different tree-shaped structures: 1) the bases structure and 2) the points structure, see Fig. 1.

In this approach, bases $\mathcal{B}_j$ and points P_j are defined in terms of other bases $\mathcal{B}_i$ and points P_i by the way of relative base-change or rotation matrices $\mathbf{R}_{\mathcal{B}_i}^{\mathcal{B}_j}$ and positions vectors $\mathbf{r}_{P_i}^{P_j}$. The functions used by the symbolic library [8] can be schematically represented as

$$\mathcal{B}_i \xrightarrow{\mathbf{R}_{\mathcal{B}_i}^{\mathcal{B}_j}(exp_x(\mathbf{q},t,\mathbf{p}),exp_y(\mathbf{q},t,\mathbf{p}),exp_z(\mathbf{q},t,\mathbf{p}),\ exp_\phi(\mathbf{q},t,\mathbf{p}))} \mathcal{B}_j \qquad (1)$$

$$P_i \xrightarrow{\mathbf{r}_{P_i}^{P_j}(exp_x(\mathbf{q},t,\mathbf{p}),\ exp_y(\mathbf{q},t,\mathbf{p}),\ exp_z(\mathbf{q},t,\mathbf{p}),\ \mathcal{B}_k)} P_j, \qquad (2)$$

where $exp_*(\mathbf{q}, t, \mathbf{p})$ represent arbitrary symbolic expressions in terms of which vectors and base-change matrices are defined. This in turn confers physical meaning to the defined coordinates and parameters. Note that, in order to illustrate the procedure, the rotation matrix appearing in Eq. (1) is parametrized using Euler parameters.

This splitting of the body structure into the bases and points structures confers complete flexibility to the choice of the parametrization. A body position and orientation no longer needs to be defined with respect to the preceding body in the tree-shaped bodies structure. Instead, the body position is given by a point in the points structure and an orientation by a base in the bases structure.
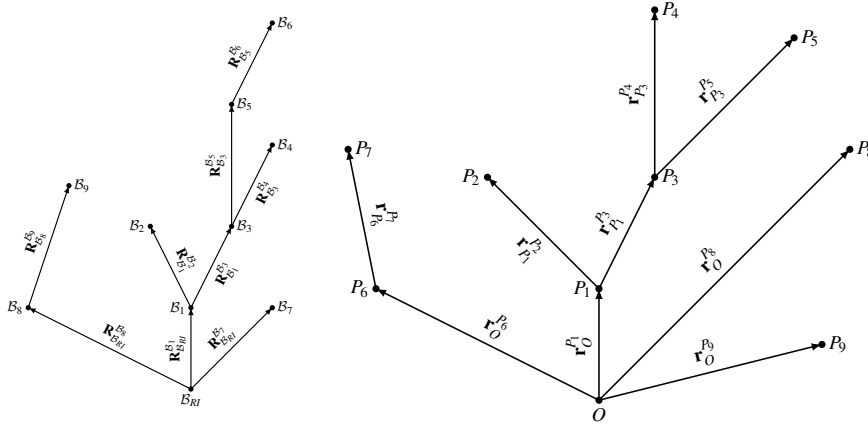
Fig. 1: Illustrative examples of bases (left) and points (right) structures

It should be noted that the bases structure is independent of the points structure. Conversely, the points structure is dependent on the bases structure as the relative position vector components are given using arbitrary bases (note the $\mathcal{B}_k$ parameter in Eq. (2)). Finally, it should be understood that the nature of the parameterization depends on the definition of the rotation matrices and position vectors. For example, if they are defined with respect to another point or orientation the coordinates will be "relative", but if they are defined with respect to an absolute point or orientation they will be "absolute".

2.2 "On-the-way" atomization

The symbolic expressions that need to be dealt with can be huge. The successive multiplication of symbolic expressions leads to an explosive growth in the size of the expressions that can limit the maximum size of the multibody systems that can be analyzed. In order to deal with this problem, a standard technique in the context of symbolic computations called *atomization* is used.

Atomization is a technique that condenses a symbolic expression set by splitting their expressions into several elemental sub-expressions. These elemental sub-expressions are called "atoms". They can be defined in terms of binary operations between symbols, numbers and/or other atoms, or as transcendental functions of atoms.

This technique is beneficial when repeated sub-expressions appear and the same atom is used to represent them. Symbolically, this means less memory—as the sub-expression is allocated in memory once—and faster symbolic manipulations. Numerically, this implies that the repeated sub-expression is only computed once, leading to computational cost savings.

In the present context, the sets of expressions related to the functions used to computationally implement a given MSD formalism are discussed. This atomized representation leads directly to the exportation of these functions in a way that benefits

```
atom27 = sin(theta2);                          atom271 =  m3*l1*cg3x*atom46+I3yy+m3*atom49*l1*cg3z
atom0 = cos(theta3);                                      +atom229;
atom1 = sin(theta3);                           atom273 =  I3yy+atom229;
atom26 = cos(theta2);
atom49 =  atom1*atom26+atom27*atom0;           _M[0] = -m3*( atom197+2.0*atom214+atom215)-atom197*m2
atom46 = -atom27*atom1+atom26*atom0;                     -I3yy-I2yy+-2.0*atom253-I1yy
atom237 = m3*l2*( cg3x*atom0+atom1*cg3z);                +-2.0*( atom27*cg2z+atom26*cg2x)*l1*m2;
atom253 =  m3*l1*( cg3x*atom46+atom49*cg3z)+atom237;     _M[1] = -atom270;
atom200 = -l1*atom26;                          _M[2] = -atom271;
atom214 = -l2*atom200;                         _M[3] = -atom270;
atom197 = (l1*l1);                             _M[4] = -m3*atom215-I3yy+-2.0*atom237-I2yy;
atom215 = (l2*l2);                             _M[5] = -atom273;
atom270 =  atom27*l1*cg2z*m2+m3*( atom214+atom215)       _M[6] = -atom271;
          -m2*cg2x*atom200+I3yy+atom237+I2yy+atom253;    _M[7] = -atom273;
atom229 =  m3*l2*cg3x*atom0+m3*l2*atom1*cg3z;  _M[8] = -I3yy;
```

Fig. 2: Exported C code for atomized mass matrix.

from the referred computational cost savings. To get a less abstract idea, Fig. 2 shows the exported C code for the mass matrix of a simple four-bar linkage mechanism.

The atomization process should ideally be done "on-the-way", meaning that every time a new algebraic operation is performed a new atom is created or replaced by an existing matching atom. Thus, the symbolic method takes advantage of the memory savings and the associated complexity reduction as soon as possible in the problem setup. This means that the symbolic algebra system works internally with atomized expressions, a feature that is not obvious for the standard user but that is widespread in computer algebra systems. See Fig. 3 for an elemental example.

In the same line, it is important to remember that the fundamental symbolic differentiation and substitution operations should be implemented to work directly on atomized expressions. This maximizes atom recycling and limits enormously the time and memory requirements of the algorithms.

In this context, to take advantage of the atomization, care should be taken when choosing the way and order in which the required operations are performed. The operation number should be minimized and atom recycling maximized. A general purpose algorithm aiming at finding an absolute minimum number of operations would require an exhaustive search that is beyond the reach of reasonable computational resources. Therefore, appropriate heuristics must be defined. Recursive dynamics formulations are usually taken as the starting point to define such heuristics. In this work, these heuristics are partly implemented by the way of mechanics operators, as will be explained in the next section.

2.3 Recursive kinematic operators

Recursive formulations represent the state-of-the-art on symbolic MSD [1]. These formulations use relative coordinates to parametrize the system leading to a tree-shaped body structure[1]. This allows the recursive determination positions, velocities and accelerations of points as well as orientations, angular velocities and accelerations of bodies, by the way of the well known "motion composition laws". When different elements -points and orientations- share a common path towards the tree

---

[1] Closed loops are opened to parametrize and closed through constraint equations.

The addition of vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ given their components represented in bases $\mathcal{B}_1$ and $\mathcal{B}_2$

$$\{\boldsymbol{u}\}_{\mathcal{B}_1} = \begin{Bmatrix} u_x \\ u_y \\ u_z \end{Bmatrix}_{\mathcal{B}_1} \quad \text{and} \quad \{\boldsymbol{v}\}_{\mathcal{B}_2} = \begin{Bmatrix} v_x \\ v_y \\ v_z \end{Bmatrix}_{\mathcal{B}_2}$$

is sought. Let the rotation matrix be

$$\mathbf{R}_{\mathcal{B}_1}^{\mathcal{B}_2} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{bmatrix}.$$

The addition of the two vectors represented in $\mathcal{B}_1$ base is performed as follows:

$$\{\boldsymbol{u}+\boldsymbol{v}\}_{\mathcal{B}_1} = \{\boldsymbol{u}\}_{\mathcal{B}_1} + \mathbf{R}_{\mathcal{B}_1}^{\mathcal{B}_2}\{\boldsymbol{v}\}_{\mathcal{B}_2} = \begin{Bmatrix} u_x \\ u_y \\ u_z \end{Bmatrix}_{\mathcal{B}_1} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{bmatrix}\begin{Bmatrix} v_x \\ v_y \\ v_z \end{Bmatrix}_{\mathcal{B}_2} =$$

$$\begin{Bmatrix} u_x \\ u_y \\ u_z \end{Bmatrix}_{\mathcal{B}_1} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & \alpha_1 & -\alpha_2 \\ 0 & \alpha_2 & \alpha_1 \end{bmatrix}\begin{Bmatrix} v_x \\ v_y \\ v_z \end{Bmatrix}_{\mathcal{B}_2} = \begin{Bmatrix} u_x \\ u_y \\ u_z \end{Bmatrix}_{\mathcal{B}_1} + \begin{Bmatrix} v_x \\ \alpha_1 v_y - \alpha_2 v_z \\ \alpha_2 v_y + \alpha_1 v_z \end{Bmatrix}_{\mathcal{B}_1} =$$

$$\begin{Bmatrix} u_x \\ u_y \\ u_z \end{Bmatrix}_{\mathcal{B}_1} + \begin{Bmatrix} v_x \\ \alpha_3 - \alpha_4 \\ \alpha_5 + \alpha_6 \end{Bmatrix}_{\mathcal{B}_1} = \begin{Bmatrix} u_x \\ u_y \\ u_z \end{Bmatrix}_{\mathcal{B}_1} + \begin{Bmatrix} v_x \\ \alpha_7 \\ \alpha_8 \end{Bmatrix}_{\mathcal{B}_1} = \begin{Bmatrix} \alpha_9 \\ \alpha_{10} \\ \alpha_{11} \end{Bmatrix}_{\mathcal{B}_1}$$

where α_i are the atoms:

$\alpha_1 = \cos(\theta)$	$\alpha_2 = \sin(\theta)$	$\alpha_3 = \alpha_1 v_y$	$\alpha_4 = \alpha_2 v_z$	$\alpha_5 = \alpha_2 v_y$	$\alpha_6 = \alpha_1 v_z$
$\alpha_7 = \alpha_3 - \alpha_4$	$\alpha_8 = \alpha_5 - \alpha_6$	$\alpha_9 = u_x + v_x$	$\alpha_{10} = u_y + \alpha_7$	$\alpha_{11} = u_z + \alpha_8$	

Fig. 3: "On-the-way" atomization example.

root, this implies the sharing of common sub-expressions. If applied symbolically, this recursive computation produces nearly good optimal "on-the-way" atomizations. This sharing of expressions is the main feature on which the kinematic forward recursion step, found in recursive formulations, is based.

For example, for a serial multibody system the angular velocity of body S_{i+1} with respect to S_{i-1} could be expressed as follows:

$$\boldsymbol{\omega}_{S_{i-1}}^{S_{i+1}} = \boldsymbol{\omega}_{S_{i-1}}^{S_i} + \boldsymbol{\omega}_{S_i}^{S_{i+1}} \tag{3}$$

In the same way the angular velocity of body S_{i+2} with respect to S_{i-1} is expressed as:

$$\boldsymbol{\omega}_{S_{i-1}}^{S_{i+2}} = \boldsymbol{\omega}_{S_{i-1}}^{S_i} + \boldsymbol{\omega}_{S_i}^{S_{i+1}} + \boldsymbol{\omega}_{S_{i+1}}^{S_{i+2}} = \boldsymbol{\omega}_{S_{i-1}}^{S_{i+1}} + \boldsymbol{\omega}_{S_{i+1}}^{S_{i+2}} \tag{4}$$

So, when computing magnitudes related to a given element it can be appreciated how computations related to elements down in the same chain can be reused. Other kinematic entities like position vectors, base-change matrices, linear velocities, accelerations and angular accelerations can be dealt with analogously.

In correspondence with the substitution of the bodies structure by the bases and points structures proposed in this work, the recursivity at the level of bodies is now

dealt with at the bases and points structure levels. This allows one not only to use arbitrary parametrizations, as commented before, but also a better use of any degree of recursivity that may be implicit when using arbitrary parametrizations.

To that end, kinematic operators that take advantage of any recursivity present in the parameterization are defined: Position vector between two points, velocity of a point with respect to a given frame (point plus orientation), angular velocity, base-change matrix, and so on. Basically the typical *recursivity* found in recursive algorithms is translated to the operator algebra.

To support this, an algebra of 3D vectors and tensors is defined. This algebra relieves the user of dealing with base-changes that are internally dealt with. The full system works using "on-the-way" atomization and the operators are implemented taking advantage of the aforementioned recursivity. In this way, the number of operations is minimized and the reuse of atoms is maximized. As a consequence, an optimal implementation of the given formalism for any parameterization chosen by the user is obtained.

The backward recursion of $O(n^3)$ algorithms can be considered a particular implementation of the principle the virtual power. The inertia forces and moments of the bodies affected by a given virtual movement appear added together in the contribution of this virtual movement to the system dynamic equations. Recursive formulations take advantage of this grouping so that they minimize the required operation count. Taking advantage of this when symbolically applying the principle of virtual power produces atomizations as efficient as state-of-the-art $O(n^3)$ formulations. This is the approach followed by the symbolic implementation of the virtual power principle used in this work.

As an illustration of the achievements of these symbolic methods, nearly optimal atomized equations for standard multibody systems using for example the principle of virtual power and relative coordinates can be obtained. Some authors [1] claim to be unable to do the same unless a direct symbolic implementation of a recursive formulation is used[2].

2.4 Other symbolic methods

There are other symbolic methods that can be applied to reduce even further the complexity of the resulting model: *"trigonometricaly simplifiable expression removal"* [7], *"base parameter formulation of the system inertias"*[9–11], *"base parameter elimination"*[6], etc... This methods can be applied directly on top of the presented modeling techniques. However, they are not considered in this work.

3 Multibody model description

The *FEVE 3000* [12] locomotive multibody model developed in this work is depicted in Fig. 4 with an expanded view of the main parts shown in Fig. 5.

---

[2] In comparison, this method presents the overhead of having to deal with the common atom search. Even if using hash tables to do the search the symbolic processing phase seems to take longer.
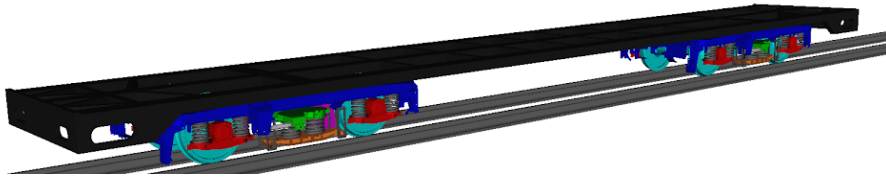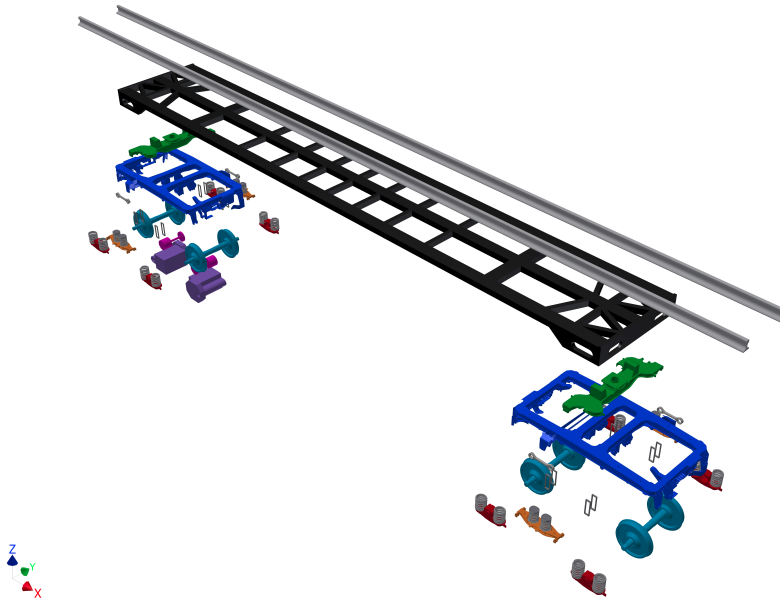
Fig. 4: Multibody model

Fig. 5: Expanded View

The *Vehicle Body* (dark grey) is attached to the front and rear bogies through two *Slider* (green) bodies. The front bogie consists of a *Slider* that rests on a couple of *Suspender* (orange) bodies hanging from the *Bogie Frame* (dark blue), and two *Wheelset* bodies (light blue), each of them with two *Axle Box* (red) bodies. There are a couple of anti-yaw links (grey) between each *Suspender* and the *Bogie Frame*. The rear bogie is identical to the front one but it includes two motors (one per each *Wheelset*). The motor *Housing* (mauve) rotates around the relative *Wheelset* and is attached to the *Bogie Frame* using a bushing. The motor includes a *Rotor* (pink). The transmission of motion from the *Rotor* to the *Wheelset* is done by the way of a gear pair.

The *Slider* is connected by four identical spring-dampers to the *Suspender* part of the *Bogie Frame*. In a similar way, each *Axle Box* is connected to the *Bogie Frame* by two identical spring-dampers. The *Housing* is also attached to the *Bogie Frame* using a bushing. Compliance is considered in the gearing contacts. Linear stiffness

and damping is assumed for spring-dampers, bushings and gear compliance. Braking on the wheels and traction on the rotors is modeled considering externally applied torques.

The wheel-rail interaction model considers a fully three-dimensional rolling contact considering a single contact point per wheel. Normal contact is enforced through the use of constraints, while the tangential forces are determined based on the standard Kalker linear constitutive model. Note that generic wheel and rail profiles are considered. The rails can present general irregularities along the track.

3.1 Parametrization

Multibody

The *Vehicle Body* is positioned relative to the track using absolute coordinates (3 translations followed by 3 Euler rotations). Each *Slider* is attached to the *Vehicle Body* by a revolute joint. A rotation relative to the *Vehicle Body*, in the vertical direction, is used to position the *Slider*.

To simplify the modeling, the effect of the anti-yaw bar is accounted for by removing the relative yaw motion between the *Bogie Frame* and the *Slider*. With the same purpose, the *Suspender* is considered fixed to the *Bogie Frame*. A vertical translation followed by two successive horizontal rotations (roll and tilt) to position the *Bogie Frame* relative to the *Slider* is used. Each *Axle Box* is positioned fixed to the "non-spinning wheelset" frame (NSWHS), a frame that follows the relative *Wheelset* but that does not spin with it. Each *Wheelset* is positioned relative to the *Bogie Frame* using a vertical translation and two horizontal rotations (roll and spin). Other relative degrees of freedom between these bodies are removed by the particular configuration of the spring-dampers. A rotation around the *Wheelset* axis, relative to the NSWHS frame, is used to place each motor *Housing*. A rotation in the same direction, also relative to the NSWHS frame, is introduced to give the angular position of the *Rotor* of each motor.

A total number of 60 generalized coordinates, $\mathbf{q}$, is used in this parametrization.

Contact

The rails and wheel surfaces are described using cubic splines [13],

$$f^*(u^*) = (((a^*(u^* - u^*_{bp}) + b^*)(u^* - u^*_{bp}) + c^*)(u^* - u^*_{bp}) + d^*), \qquad (5)$$

defined based on a set of control points that approximate their geometry.

Figure 6 schematically shows the parametrization for a wheel-rail pair. The wheel and rail profiles are given respectively by $f^w(u^w)$ and $f^r(u^r)$, while the shape of the center line of the base of the rail along the track is given by $f^t_x(s^r)$, $f^t_y(s^r)$ and $f^t_z(s^r)$. Wheel surfaces are assumed to have cylindrical symmetry. Not represented in the figure is the spline used to represent the camber of the rail $f^t_\theta(s^r)$.

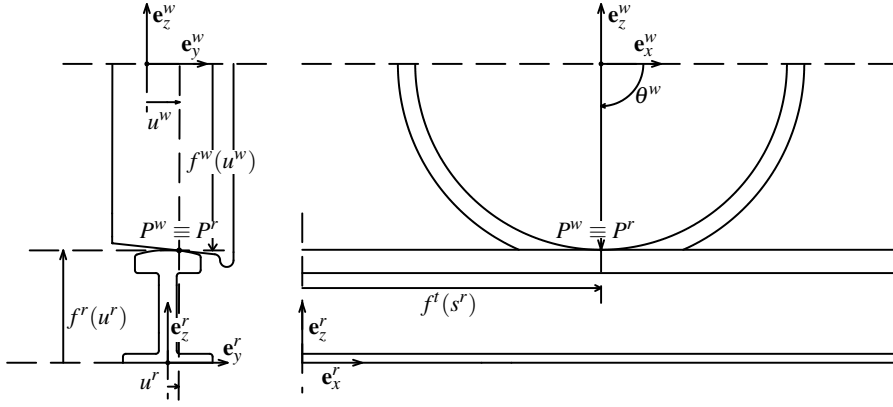Fig. 6: Surfaces and railway parametrization

For the contact point at each wheel-rail pair the parameters θ^w, u^w and s^r, u^r play the role of "generalized coordinates" used to position the contact points, P^w and P^r, respectively onto the wheel and rail surfaces. When the contact is materialized, the point P^w and the point P^r -defined as an arbitrary points in the surfaces of the wheel and rail respectively- are coincident.

The position of P^w is given relative to a reference point in the wheel axis, O^w, as

$$\mathbf{r}_{Ow}^{P^w}(u^w, \theta^w) = f^w(u^w)\cos(\theta^w)\mathbf{e}_x^w + u^w\mathbf{e}_y^w - f^w(u^w)\sin(\theta^w)\mathbf{e}_z^w. \qquad (6)$$

For numerical reasons, the position of this point is given relative to a NSWHS base, $\mathbf{e}_x^w, \mathbf{e}_y^w, \mathbf{e}_z^w$. Analogously, the position of P^r is given relative to a reference point in the ground or track reference, O^r

$$\mathbf{r}_{Or}^{P^r}(u^r, s^r) = f_x^t(s^r)\mathbf{e}_x^r + f_y^t(s^r)\mathbf{e}_y^r + f_z^t(s^r)\mathbf{e}_z^r + u^r\hat{\mathbf{c}} + f^r(u^r)\hat{\mathbf{n}} \qquad (7)$$

The base $\mathbf{e}_x^r, \mathbf{e}_y^r, \mathbf{e}_z^r$ is fixed at the ground. Defining $\hat{\mathbf{t}}$ as a unit vector tangent to the center line of the rail base, $\hat{\mathbf{c}}$ is defined as a unit vector perpendicular to $\hat{\mathbf{t}}$ with an angle $f_\theta^t(s^r)$ with the ground measured in the positive direction of $\hat{\mathbf{t}}$. $\hat{\mathbf{n}}$ completes the base so that it is dexterous.

The whole set of $8 \times 4 = 32$ generalized coordinates required to position the $i = 1...8$ contact points used in the analyzed example is referred to as $\mathbf{s} = [..., \mathbf{s}_i^\mathsf{T}, ...]^\mathsf{T}$, where $\mathbf{s}_i = [\theta_i^w, u_i^w, s_i^r, u_i^r]^\mathsf{T}$ is the set of coordinates required to position points P_i^w and P_i^r at the i-th contact point.

From the symbolic modeling point of view, functions $f^*(u^*)$ are modeled as if a single 3rd order polynomial completely represents the whole profile. At "show-time", the coefficients a^*, b^*, c^*, d^* and break-points u_{bp}^* are updated depending on the position of the contact point.

As commented before, for the purposes of this paper only a single point of contact is considered. Note that flange contact rarely occurs when the train runs along straight tracks or huge radii curved tracks, unless the train velocity is close to its critical speed [14]. Nevertheless the parametrization proposed here is compatible with some multiple-point-of-contact approaches [14].

3.2 Constraint equations

The only constraints that are present in the analyzed problem are those related to the contact points between wheel and rail. At a given contact, it should be enforced that the points P^w and P^r are coincident and that the surfaces at these points are tangent.

Defining the tangent and normal vectors to the wheel at point P^w as $\mathbf{t}_x^w$, $\mathbf{t}_y^w$, $\mathbf{n}^w$, and the tangent and normal vectors to the rail at point P^r as $\mathbf{t}_x^r$, $\mathbf{t}_y^r$, $\mathbf{n}^r$. These conditions can be written [3] as:

$$\phi^n(\mathbf{q},\mathbf{s}) = \mathbf{n}^r \cdot \mathbf{r}_{Pr}^{P^w} = \mathbf{0} \tag{8}$$

and

$$\phi^d(\mathbf{q},\mathbf{s}) = \begin{bmatrix} \mathbf{t}_x^r \cdot \mathbf{r}_{Pr}^{P^w} \\ \mathbf{t}_y^r \cdot \mathbf{r}_{Pr}^{P^w} \\ \mathbf{t}_x^w \cdot \mathbf{n}^r \\ \mathbf{t}_y^w \cdot \mathbf{n}^r \end{bmatrix} = \mathbf{0}, \tag{9}$$

where ϕ^n is the so called normal constraint, and ϕ^d are the so called tangent constraints. For each contact point, the tangent and normal vectors can be defined as:

$$\mathbf{t}_x^r \;\; = \frac{\partial \mathbf{r}_{O^r}^{P^r}/\partial s^r}{\left|\partial \mathbf{r}_{O^r}^{P^r}/\partial s^r\right|}, \;\; \mathbf{t}_y^r = \frac{\partial \mathbf{r}_{O^r}^{P^r}/\partial u^r}{\left|\partial \mathbf{r}_{O^r}^{P^r}/\partial u^r\right|}, \text{ and } \;\; \mathbf{n}^r = \mathbf{t}_x^r \times \mathbf{t}_y^r \tag{10}$$

$$\mathbf{t}_x^w = \frac{\partial \mathbf{r}_{O^w}^{P^w r}/\partial \theta^w}{\left|\partial \mathbf{r}_{O^w}^{P^w}/\partial \theta^w\right|}, \;\; \mathbf{t}_y^w = \frac{\partial \mathbf{r}_{O^w}^{P^w}/\partial u^w}{\left|\partial \mathbf{r}_{O^w}^{P^w}/\partial u^w\right|}, \text{ and } \;\; \mathbf{n}^w = \mathbf{t}_x^w \times \mathbf{t}_y^w. \tag{11}$$

These vectors and constraint equations can easily be defined using the symbolic procedures previously discussed. It will be seen that this symbolic implementation will be very efficient as well.

Now, the subindex $i = 1...8$ is used to refer to the constraint equations relative to each of the 8 contact points. The set of all the normal constraints is referred as $\phi^n(\mathbf{q},\mathbf{s}) = [..., \phi^n{}_i(\mathbf{q},\mathbf{s}),...]^\mathsf{T}$. Analogously, the set of all the tangent constraints is referred as $\phi^d(\mathbf{q},\mathbf{s}) = [..., \phi^d{}_i(\mathbf{q},\mathbf{s}),...]^\mathsf{T}$.

3.3 Dynamic equations

As commented previously, the dynamic equations are obtained based on the direct application of the principle of virtual power. Using the vector $[\mathbf{q},\mathbf{s}]$ as the set of generalized coordinates, the mass matrix $\mathbf{M}$ is obtained by differentiation of the equations motion with respect to the the generalized accelerations and the generalized force $\boldsymbol{\delta}$ vector is obtained by substitution of the generalized accelerations by zero in the equations of motion. These equations should be complemented by the second derivative of the constraint equations to have a determined set of equations

$$\ddot{\phi}^n(\mathbf{q},\mathbf{s}) = \mathbf{0} \tag{12}$$

$$\ddot{\phi}^d(\mathbf{q},\mathbf{s}) = \mathbf{0} \tag{13}$$

This set of equations shows the following structure:

$$
\begin{bmatrix}
\mathbf{M_{qq}(q,\dot q)} & \mathbf{0} & \dot\phi_{\dot q}^{n\mathsf{T}}(\mathbf{q,s}) & \dot\phi_{\dot q}^{d\mathsf{T}}(\mathbf{q,s}) \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \dot\phi_{\dot s}^{d\mathsf{T}}(\mathbf{q,s}) \\
\dot\phi_{\dot q}^{n}(\mathbf{q,s}) & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\dot\phi_{\dot q}^{d}(\mathbf{q,s}) & \dot\phi_{\dot s}^{d}(\mathbf{q,s}) & \mathbf{0} & \mathbf{0}
\end{bmatrix}
\begin{bmatrix}
\ddot{\mathbf q} \\ \ddot{\mathbf s} \\ \boldsymbol\lambda^n \\ \boldsymbol\lambda^d
\end{bmatrix}
=
\begin{bmatrix}
\boldsymbol\delta_{\mathbf q}(\mathbf{q,\dot q}) \\ \mathbf{0} \\ \boldsymbol\gamma^n(\mathbf{q,s,\dot q,\dot s}) \\ \boldsymbol\gamma^d(\mathbf{q,s,\dot q,\dot s})
\end{bmatrix}, \quad (14)
$$

where $\mathbf{M_{qq}}$ and $\boldsymbol\delta_{\mathbf q}$ are the blocks of the mass matrix $\mathbf{M}$ and vector $\boldsymbol\delta$ related to the set of coordinates $\mathbf{q}$. In reference [3], the authors refer this formulation as the Augmented Contact Constraint Formulation (ACCF).

The particular structure of the dynamic equations for the problem analyzed can appreciated in Fig. 7. There, the nonzero entries for matrix $\left[[\mathbf{M}, \dot\phi_{\dot q}{}^{\mathsf{T}}; \dot\phi_{\dot q}, \mathbf{0}], [\boldsymbol\delta; \boldsymbol\gamma]\right]$ are shown as dots. It should be noted that $\dot\phi_{\dot s}^{n}$ is zero numerically, even if symbolically a few nonzero expressions can be seen. As accelerations $\ddot{\mathbf s}$ are not needed, the
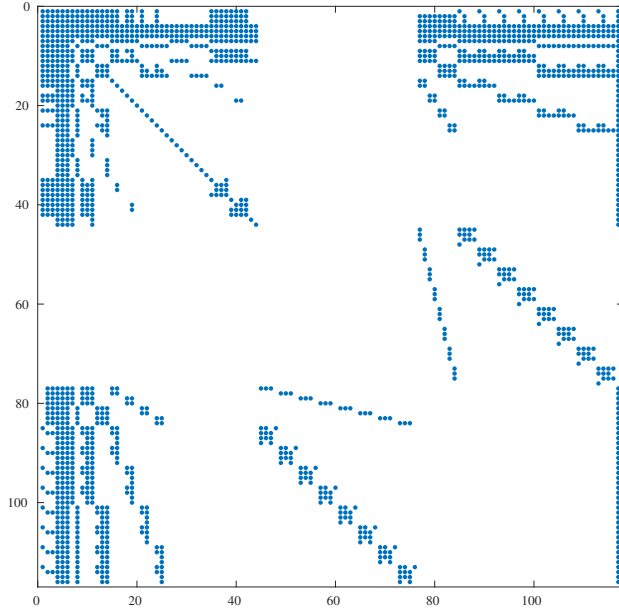

Fig. 7: Dynamic model structure $\left[[\mathbf{M}, \dot\phi_{\dot q}{}^{\mathsf{T}}; \dot\phi_{\dot q}, \mathbf{0}], [\boldsymbol\delta; \boldsymbol\gamma]\right]$

previous system of equations can be reduced to

$$
\begin{bmatrix}
\mathbf{M_{qq}(q,\dot q)} & \dot\phi_{\dot q}^{n\mathsf{T}}(\mathbf{q,s}) \\
\dot\phi_{\dot q}^{n}(\mathbf{q,s}) & \mathbf{0}
\end{bmatrix}
\begin{bmatrix}
\ddot{\mathbf q} \\ \boldsymbol\lambda^n
\end{bmatrix}
=
\begin{bmatrix}
\boldsymbol\delta_{\mathbf q}(\mathbf{q,\dot q}) \\ \boldsymbol\gamma^n(\mathbf{q,s,\dot q,\dot s})
\end{bmatrix}, \quad (15)
$$

The reduced structure of the dynamic equations can appreciated on Fig. 8. Obviously, standard linear solution procedures are going to perform much more efficiently in this

case. There, the nonzero entries for matrix $\left[[\mathbf{M_{qq}}, \dot{\boldsymbol{\phi}}_{\dot{\mathbf{q}}}^{n\top}; \dot{\boldsymbol{\phi}}_{\dot{\mathbf{q}}}^{n}, \mathbf{0}], [\boldsymbol{\delta_q}; \boldsymbol{\gamma}^n]\right]$ are shown as dots. In reference [3], the authors refer this formulation as the Embedded Contact

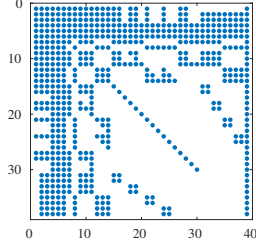Fig. 8: Dynamic model structure $\left[[\mathbf{M_{qq}}, \dot{\boldsymbol{\phi}}_{\dot{\mathbf{q}}}^{n\top}; \dot{\boldsymbol{\phi}}_{\dot{\mathbf{q}}}^{n}, \mathbf{0}], [\boldsymbol{\delta_q}; \boldsymbol{\gamma}^n]\right]$

Constraint Formulation (ECCF) [3].

Constraint stabilization is performed based on projection on the coordinate and velocity manifolds. That requires the solution of the following equations

$$\boldsymbol{\phi}^n(\mathbf{q}, \mathbf{s}) = \mathbf{0} \tag{16}$$

$$\boldsymbol{\phi}^d(\mathbf{q}, \mathbf{s}) = \mathbf{0}, \tag{17}$$

at the coordinate level, and of

$$\dot{\boldsymbol{\phi}}_{\dot{\mathbf{q}}}^n(\mathbf{q}, \mathbf{s})\dot{\mathbf{q}} = \boldsymbol{\beta}^n(\mathbf{q}, \mathbf{s}) = \mathbf{0} \tag{18}$$

$$\dot{\boldsymbol{\phi}}_{\dot{\mathbf{q}}}^d(\mathbf{q}, \mathbf{s})\dot{\mathbf{q}} + \dot{\boldsymbol{\phi}}_{\dot{\mathbf{s}}}^d(\mathbf{q}, \mathbf{s})\dot{\mathbf{s}} = \boldsymbol{\beta}^d(\mathbf{q}, \mathbf{s}) = \mathbf{0} \tag{19}$$

at the velocity level. Note that there are no rheonomous equations in the problem analyzed, an therefore $\boldsymbol{\beta}^n(\mathbf{q}, \mathbf{s})$ and $\boldsymbol{\beta}^d(\mathbf{q}, \mathbf{s})$ are $\mathbf{0}$. For example, partitioning [15, 16] into dependent and independent coordinates is done frequently. The dependent coordinates and velocities are obtained in terms of the independent ones.

It should be remarked that in the ECCF context, it is better not to consider $\mathbf{s}$ and $\dot{\mathbf{s}}$ as standard generalized coordinates, but rather as a set of auxiliary variables that must be known in order to solve equation system (15). After the integration, Eq. (17) and Eq. (19) can be used to obtain the auxiliary variables $\mathbf{s}$ and $\dot{\mathbf{s}}$ in terms of $\mathbf{q}$ and $\dot{\mathbf{q}}$. In this context the position and velocity projection would be performed after this step, based on Eq. (16) and Eq. (18), respectively. These inertia-less coordinates find different names in the literature, such as "surface parameters" [17, 3, 18–20], "non-generalized coordinates" [21] or "auxiliary variables" [22], but their main feature is that they do not participate in the system dynamics.

3.4 Contact model.

In the context of railway dynamic simulation, it is very important to correctly determine the values of the creep forces between the wheel and the rail. To that end, in this

work the well known linear contact theory of Kalker [23] is used. This theory requires the determination of several data: location of the contact point, the creepages, normal loads at this contact point, wheel and rail surface curvatures at the contact point, tangent and normal vectors at the contact patch. These computations are big and must be done for every contact. To speed up the computations, several authors [24–26] propose the use of pre-calculated look-up tables to determine the required data. This procedure is tedious and usually requires the introduction of some modeling simplifications. The authors propose to compute these quantities without simplifications, on line, based on functions exported using the proposed symbolic methods. This is a simpler and more general procedure to apply. The results will confirm that this is a very fast procedure.

Contact patch geometry determination.

Based on classical Hertzian contact theory, the contact patch is a flat ellipse [27]. The semi-axes of this ellipse in the longitudinal and transversal directions, a and b respectively, are determined as follows:

$$a = \left(\frac{3}{2} \frac{1 - \nu^2}{E} \frac{1}{A + B} N \right)^{\frac{1}{3}} m(\theta) \quad \text{and} \quad b = \left(\frac{3}{2} \frac{1 - \nu^2}{E} \frac{1}{A + B} N \right)^{\frac{1}{3}} n(\theta). \quad (20)$$

In these expressions, N is the normal contact force acting on the wheel, E is the Young's modulus and ν the Poisson's ratio. $m(\theta)$ and $n(\theta)$ are adimensional functions proposed by Hertz. On-line interpolation in Table 4.1 in Ref. [27] is used to evaluate these functions. $\theta = \cos^{-1} \left(\frac{|A-B|}{A+B} \right)$, where A and B are determined as

$$A = \frac{1}{2} \left(\frac{1}{R_x^w} + \frac{1}{R_x^r} \right) \quad \text{and} \quad B = \frac{1}{2} \left(\frac{1}{R_y^w} + \frac{1}{R_y^r} \right). \quad (21)$$

R_x^w, R_y^w, R_x^r, R_y^r are the curvature radii of wheel and rail surfaces at the contact point. For the case studied, these are computed as:

$$R_x^w = \frac{\sqrt{(\mathbf{r}_{O^w}^{P^w} \mathbf{e}_x^w)^2 + (\mathbf{r}_{O^w}^{P^w} \mathbf{e}_y^w)^2}}{\sqrt{1 - (\mathbf{n}^r \mathbf{e}_y^w)^2}} \quad 3 \quad R_y^w = \left| \frac{(1 + \frac{\partial f^w}{\partial u_w})^{3/2}}{\frac{\partial^2 f^w}{\partial u_w^2}} \right|$$

$$R_y^r = \left| \frac{(1 + \frac{\partial f^r}{\partial u_r})^{3/2}}{\frac{\partial^2 f^r}{\partial u_r^2}} \right| \quad R_x^r = \infty \quad (22)$$

These curvature radii are obtained and exported based on the symbolic methods presented preciously in this paper. The normal force N is the Lagrange multiplier associated with the normal constraint of the relative contact point. It is obtained directly from the solution of the dynamic system of equations. To keep the dynamic problem linear, avoiding a nonlinear iteration, the normal force used is the one obtained in the previous integration step.

---

[3] Distance from the rotation axis to the contact point along the normal at the contact point.

Creep forces and moments

From the modeling perspective, creep forces and moments are considered external actions, so their effect is contained in the vector $\boldsymbol{\delta}_{\mathbf{q}}$, along with all the contributions due to other inertial constitutive and external forces. For each wheel, these forces and moments are symbolically expressed as

$$\mathbf{f} = f_x\,\mathbf{t}_x^r + f_y\,\mathbf{t}_y^r \quad \text{and} \quad \mathbf{m} = m_z\,\mathbf{n}^r, \tag{23}$$

where f_x, f_y and m_z are symbols. These represents the tangent contact force and spin contact moment acting on the wheel at the contact point P^w. They are defined at the contact base $\mathbf{t}_x^r, \mathbf{t}_y^r, \mathbf{n}^r$. The components of these vectors are numerically computed as

$$\begin{bmatrix} f_x \\ f_y \\ m_z \end{bmatrix} = -G \begin{bmatrix} ab\,c_{11} & 0 & 0 \\ 0 & ab\,c_{22} & \sqrt{ab}\,c_{23} \\ 0 & -\sqrt{ab}\,c_{23} & (ab)^2\,c_{33} \end{bmatrix} \begin{bmatrix} \xi_x \\ \xi_y \\ \varphi_z \end{bmatrix} \tag{24}$$

[27]. The parameter G is the material shear modulus and $c_{ij}(a/b, \nu)$ are the coefficients determined by Kalker, tabulated in Ref. [28]. On-line interpolation in these tables is used. The creepages ξ_x, ξ_y and φ are defined as:

$$\xi_x = \frac{\mathbf{v}_{Gr.}^{P^w}}{\frac{1}{2}\left(\left|\mathbf{v}_{Gr.}^{O^w}\right| + \left|\boldsymbol{\omega}_{Gr.}^{w} \wedge \mathbf{r}_{O^w}^{P^w}\right|\right)}\,\mathbf{t}_x^r \tag{25}$$

$$\xi_y = \frac{\mathbf{v}_{Gr.}^{P^w}}{\frac{1}{2}\left(\left|\mathbf{v}_{Gr.}^{O^w}\right| + \left|\boldsymbol{\omega}_{Gr.}^{w} \wedge \mathbf{r}_{O^w}^{P^w}\right|\right)}\,\mathbf{t}_y^r \tag{26}$$

$$\varphi_z = \frac{\boldsymbol{\omega}_{Gr.}^{w}}{\frac{1}{2}\left(\left|\mathbf{v}_{Gr.}^{O^w}\right| + \left|\boldsymbol{\omega}_{Gr.}^{w} \wedge \mathbf{r}_{O^w}^{P^w}\right|\right)}\,\mathbf{n}^r \tag{27}$$

where $\mathbf{v}_{Gr.}^{P^w}$ is the velocity with respect to the ground $(Gr.)$ of the P^w contact point when moves "attached" to the wheel and $\boldsymbol{\omega}_{Gr.}^{w}$ is the wheel-set angular velocity with respect to the ground. $\mathbf{v}_{Gr.}^{O^w}$ in the velocity with respect to the ground of the center of the wheel-set and $\mathbf{r}_{O^w}^{P^w}$ is the position vector form O^w to P^w as shown in Eq. (6). The creepages are determined numerically based on exported functions for the numerators and denominators of these expressions. In this way, division by zero can be dealt with in the numerical solver.

All the symbolic functions required for the implementation of the Kalker model for all the different contact points, referred previously, are computed in a single function call. In this way recycling of atoms is maximized.

4 Numerical integration

For this case study the Linear Kalker model used to model the contact forces has been a major source of problems. To some extent, these forces can be considered viscous friction forces with a huge value for the equivalent viscous constant, making the system dynamics stiff. In this context, the use of an explicit integration scheme is going to require a very small time step, spoiling real-time performance.

In order to solve this problem, an Implicit-Explicit (IMEX) [29] integration schema has been devised and adjusted to overcome the problem associated with the contact forces without penalizing the computational cost. The use of these schemes is not new. It has appeared in the bibliography under other names: semi-implicit [30], additive or combined methods [31], etc. These methods use different types of discretization for the different terms in the dynamic equations. Those terms that are not related to the stiff behavior of the equations are discretized using a low-cost explicit scheme, while the stiff terms are discretized using an implicit scheme.

As commented before, creep forces and moments are introduced in the model as external actions, and their contribution is embedded in the vector $\delta_{\mathbf{q}}$. To make this contribution explicit, δ is split into two Kalker (K) and non-Kalker (NK) contributions. In this way, the first equation in Eq. (15) can be rewritten as:

$$\begin{bmatrix} \mathbf{M_{qq}} & \dot{\phi}_{\dot{\mathbf{q}}}^{n\mathsf{T}} \\ \dot{\phi}_{\dot{\mathbf{q}}}^{n} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{q}} \\ \boldsymbol{\lambda}^n \end{bmatrix} = \begin{bmatrix} \delta_{\mathbf{q}}^K + \delta_{\mathbf{q}}^{NK} \\ \boldsymbol{\gamma}^n \end{bmatrix} \tag{28}$$

This new set of equations can be integrated using an IMEX method. The terms related to creep forces will be integrated using an Implicit scheme and the rest using an Explicit scheme.

Eq. (24) can be expressed as a typical viscous contribution

$$\mathbf{f}^K = -\frac{1}{V}\mathbf{C}^K\boldsymbol{\nu} \tag{29}$$

where

$$\mathbf{f}^K = [f_x, f_y, m_z]^\mathsf{T} \tag{30}$$

$$V = \frac{1}{2}\left(\left|\mathbf{v}_{Gr.}^{O^w}\right| + \left|\boldsymbol{\omega}_{Gr.}^w \wedge \mathbf{r}_{O^w}^{P^w}\right|\right) \tag{31}$$

$$\boldsymbol{\nu} = [\mathbf{v}_{Gr.}^{P^w}\mathbf{t}_x^r \ \ \mathbf{v}_{Gr.}^{P^w}\mathbf{t}_y^r \ \ \boldsymbol{\omega}_{Gr.}^w\mathbf{n}^r]^\mathsf{T} \tag{32}$$

and

$$\mathbf{C}^K = G \begin{bmatrix} ab\,c_{11} & 0 & 0 \\ 0 & ab\,c_{22} & \sqrt{ab}\,c_{23} \\ 0 & -\sqrt{ab}\,c_{23} & (ab)^2\,c_{33} \end{bmatrix} \tag{33}$$

Adding a subindex i to refer to a particular contact point, the contribution δ^K can be obtained as:

$$\delta_{\mathbf{q}}^K = \sum_{i=1}^{8} \frac{\partial \delta_{\mathbf{q}}^K}{\partial \mathbf{f_i}^K}\mathbf{f_i}^K = -\sum_{i=1}^{8} \frac{\partial \delta_{\mathbf{q}}^K}{\partial \mathbf{f_i}^K}\frac{1}{V_i}\mathbf{C}_i^K\frac{\partial \boldsymbol{\nu}_i}{\partial \dot{\mathbf{q}}}\dot{\mathbf{q}} = -\mathbf{C}_{\mathbf{qq}}^K\dot{\mathbf{q}} \tag{34}$$

In order to determine matrix $\mathbf{C}_{\mathbf{qq}}^K$, matrices

$$\frac{\partial \delta_{\mathbf{q}}^K}{\partial \mathbf{f_i}^K}\mathbf{C}_i^K\frac{\partial \boldsymbol{\nu}_i}{\partial \dot{\mathbf{q}}}, \quad i = 1 \ldots 8, \tag{35}$$

are symbolically exported and numerically assembled matrix as

$$\mathbf{C}_{\mathbf{qq}}^{K} = \sum_{i=1}^{8} \frac{1}{V_i} \frac{\partial \boldsymbol{\delta}_{\mathbf{q}}^{K}}{\partial \mathbf{f_i}^{K}} \mathbf{C}_i^{K} \frac{\partial \boldsymbol{\nu}_i}{\partial \dot{\mathbf{q}}}, \tag{36}$$

where V_i and $\mathbf{C}_i^{K}$ are determined using the same procedures described in the previous section.

The contribution $\boldsymbol{\delta}^{NK}$, can be obtained symbolically substituting zero in $\boldsymbol{\delta}_{\mathbf{q}}$ the symbols associated with the external forces f_x, f_y, m_z for every contact point.

Now the dynamic equation set can be expressed as follows:

$$\begin{bmatrix} \mathbf{M}_{\mathbf{qq}} & \dot{\boldsymbol{\phi}}_{\dot{\mathbf{q}}}^{n\mathsf{T}} \\ \dot{\boldsymbol{\phi}}_{\dot{\mathbf{q}}}^{n} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{q}} \\ \boldsymbol{\lambda}^{n} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\delta}_{\mathbf{q}}^{NK} - \mathbf{C}_{\mathbf{qq}}^{K}\dot{\mathbf{q}} \\ \boldsymbol{\gamma}^{n} \end{bmatrix} \tag{37}$$

The IMEX integration procedure proposed follows directly from this equation.

It must be observed that in Kalker's Linear Theory when saturation occurs this method is also valid, because in this case force can be also written as the product of a constant matrix and the creepages. The numerical solver must determine with which matrix to use at each moment.

Discretization

The contribution $\mathbf{C^K}_{\mathbf{qq}}\dot{\mathbf{q}}$ is discretized using an implicit Euler. To that end, it is evaluated at the next time step $t + \Delta t$,

$$\mathbf{C}_{\mathbf{qq}}^{K}\dot{\mathbf{q}}_{t+\Delta t} \tag{38}$$

An explicit Euler scheme for the remaining terms requires acceleration to be discretized as

$$\ddot{\mathbf{q}}_{t+\Delta t} = \frac{\dot{\mathbf{q}}_{t+\Delta t} - \dot{\mathbf{q}}_t}{\Delta t} \tag{39}$$

and $\boldsymbol{\delta}_{\mathbf{q}}^{K}$ to be evaluated at t.

Substituting this into Eq. (37), the final discretization of the system takes the form:

$$\begin{bmatrix} \mathbf{M}_{\mathbf{qq}} + \mathbf{C}_{\mathbf{qq}}^{K}\Delta t & \dot{\boldsymbol{\phi}}_{\dot{\mathbf{q}}}^{n\mathsf{T}}\Delta t \\ \dot{\boldsymbol{\phi}}_{\dot{\mathbf{q}}}^{n} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \dot{\mathbf{q}}_{t+\Delta t} \\ \boldsymbol{\lambda}^{n} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\delta}_{\mathbf{q}}^{NK}\Delta t + \mathbf{M}_{\mathbf{qq}}\dot{\mathbf{q}}_t \\ \boldsymbol{\gamma}^{n}\Delta t + \dot{\boldsymbol{\phi}}_{\dot{\mathbf{q}}}^{n}\dot{\mathbf{q}}_t \end{bmatrix} \tag{40}$$

where all the functions are computed at time t. Note that in order to keep the equation solution linear, $\mathbf{C}^{K}$ is evaluated at t instead of $t + \Delta t$. The structure of this system of equations can be observed in Fig. 9. It is noticeable that the sparsity structure is very similar to the one seen in Fig. 8.

This problem has the same mathematical structure as the standard full dynamic set, so it can be solved using the same procedures. Coordinate partitioning is used [15, 16]. This is a good performing strategy that is also used by other practitioners in the symbolic multibody field. The authors use a LU procedure with full pivoting on
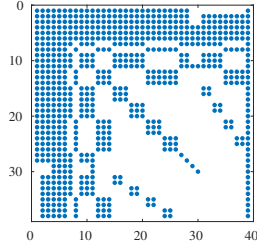
Fig. 9: Dynamic model structure $\left[[\mathbf{M_{qq}} + \mathbf{C_{qq}^K}\Delta t, \dot{\phi}_{\dot{\mathbf{q}}}^{n\top}\Delta t; \dot{\phi}_{\dot{\mathbf{q}}}^{n}, \mathbf{0}], [\boldsymbol{\delta}_{\mathbf{q}}^{NK}\Delta t + \mathbf{M_{qq}}\dot{\mathbf{q}}_t; \boldsymbol{\gamma}^n\Delta t + \dot{\phi}_{\dot{\mathbf{q}}}^{n}\dot{\mathbf{q}}_t] \right]$

the non-tangent constraint Jacobian in the generalized velocities $\dot{\mathbf{q}}$, so the set of independent coordinates can be chosen at each iteration step. This way, no conditions are enforced on the parameterization $\mathbf{q}$ used. Thus, using this IMEX scheme comes for free, as the evaluation of the functions appearing in Eq. (40) has the same complexity as the functions in Eqs. (37) or (15).

It should be noted that the matrix $\mathbf{M_{qq}} + \mathbf{C_{qq}^K}\Delta t$ is not symmetric, so LU decomposition should be used in place of $LDL^\top$, incurring a small penalty in performance.

The solution of this system will give the value of the generalized velocities at $t + \Delta t$, $\dot{\mathbf{q}}_{t+\Delta t}$. To obtain the coordinates at $\mathbf{q}_{t+\Delta t}$ the following explicit mid-point rule is used:

$$\mathbf{q}_{t+\Delta t} = \mathbf{q}_t + \frac{\dot{\mathbf{q}}_{t+\Delta t} + \dot{\mathbf{q}}_t}{2}\Delta t. \tag{41}$$

Note that it is second order and comes at no cost.

Next coordinate projection is performed. First Eq. (17) is used to obtain the contact coordinates $\mathbf{s}_{t+\Delta t}$ in terms of the $\mathbf{q}_{t+\Delta t}$. As $\mathbf{q}_{t+\Delta t}$ is accurate to second order, this procedure gives an error of the same order. To this end the following iterative Newton-Raphson procedure is used:

$$\dot{\phi}_{\dot{\mathbf{s}}}^{d}(\mathbf{q}, \mathbf{s})(\mathbf{s}_{k+1} - \mathbf{s}_k) = -\phi^d(\mathbf{q}, \mathbf{s}) \tag{42}$$

This usually involves a single iteration[4]. After the update of $\mathbf{s}$, Eq. (16) is solved for $\mathbf{q}$ using the same iterative procedure:

$$\dot{\phi}_{\dot{\mathbf{q}}}^{n}(\mathbf{q}, \mathbf{s})(\mathbf{q}_{k+1} - \mathbf{q}_k) = -\phi^n(\mathbf{q}, \mathbf{s}) \tag{43}$$

This procedure usually converges in a single iteration. Note that $\mathbf{q}_{t+\Delta t}$ is accurate to second order after the integration step and the LU decomposition of the previous Jacobians, $\dot{\phi}_{\dot{\mathbf{q}}}^{n}(\mathbf{q}, \mathbf{s})$ and $\dot{\phi}_{\dot{\mathbf{q}}}^{d}(\mathbf{q}, \mathbf{s})$, is known as they have computed at the previous velocity projection step (described latter), so the Jacobian and its decomposition is not updated in this step.

In the velocity projection step, first Eq. (18) is solved for $\dot{\mathbf{q}}$. To that end, the Jacobian $\dot{\phi}_{\dot{\mathbf{q}}}^{n}(\mathbf{q}, \mathbf{s})$ and its LU decomposition are updated. Then, Eq. (19) is solved for $\dot{\mathbf{s}}$. for this purpose, the Jacobians $\dot{\phi}_{\dot{\mathbf{q}}}^{d}(\mathbf{q}, \mathbf{s})$ and $\dot{\phi}_{\dot{\mathbf{s}}}^{d}(\mathbf{q}, \mathbf{s})$ are updated and the LU decomposition of $\dot{\phi}_{\dot{\mathbf{s}}}^{d}(\mathbf{q}, \mathbf{s})$ is computed.

---

[4] This is related to the NSWHS frame used to define the contact point in the wheel.
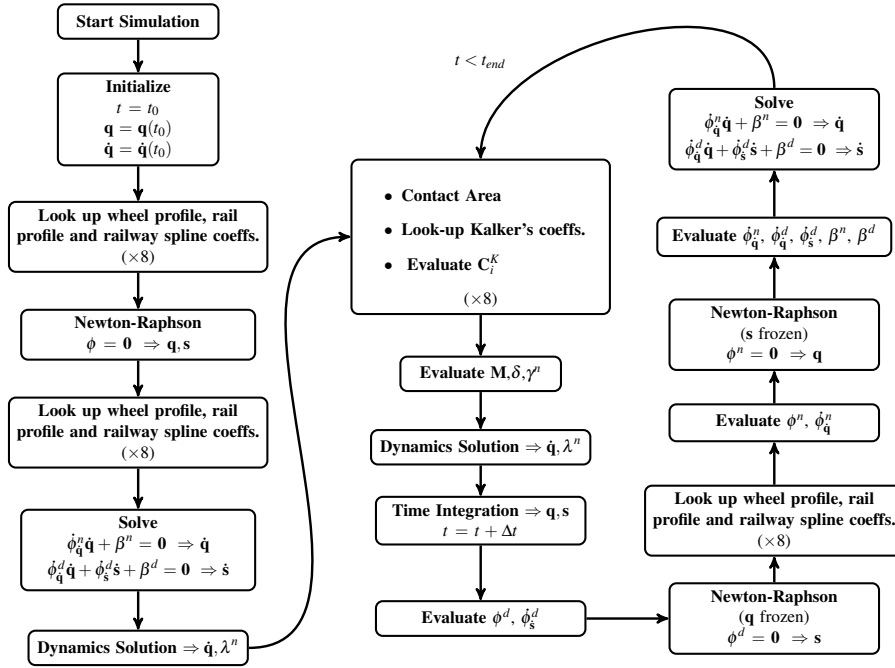
Fig. 10: Integration procedure

In Fig. 10, a schematic representation of the integration procedure described here is presented. To get a clearer picture, the steps related to the determination of the creep forces has been represented.

5 Results

Simulation description

The track used in the simulation starts and ends with two straight and parallel segments running in the x direction and separated by 50 m. Both stretches are joined by a symmetric and smooth double transition curve 270 m long in direction x. On top of the defined geometry, two harmonic vertical irregularities with an amplitude of 10 mm are added. These irregularities are defined using a sine wave that runs in the direction of x with a wavelength of 10 m. Right and left rail irregularities present a phase difference of $\pi/2$. As commented earlier, third order splines are used to discretize the whole track, including the irregularity.

The simulation starts with an initial forward speed of 23.7 m/s with the *Vehicle Body* centered at $x = 0$ m and with a lateral misalignment of 5 mm with respect to to the track center. Vehicle motors are actuated with a constant 200 Nm torque.

Computational results

Fig. 11 shows the trajectory followed by the $Vehicle\ Body$ center. Note that the the given initial state is not in dynamic equilibrium and therefore, the oscillations at the beginning of the simulation are in part due to this. This is related to the sudden application of torque at the simulation start. By the time that the vehicle center enters the track, the oscillations seen are no longer related to the initial condition.
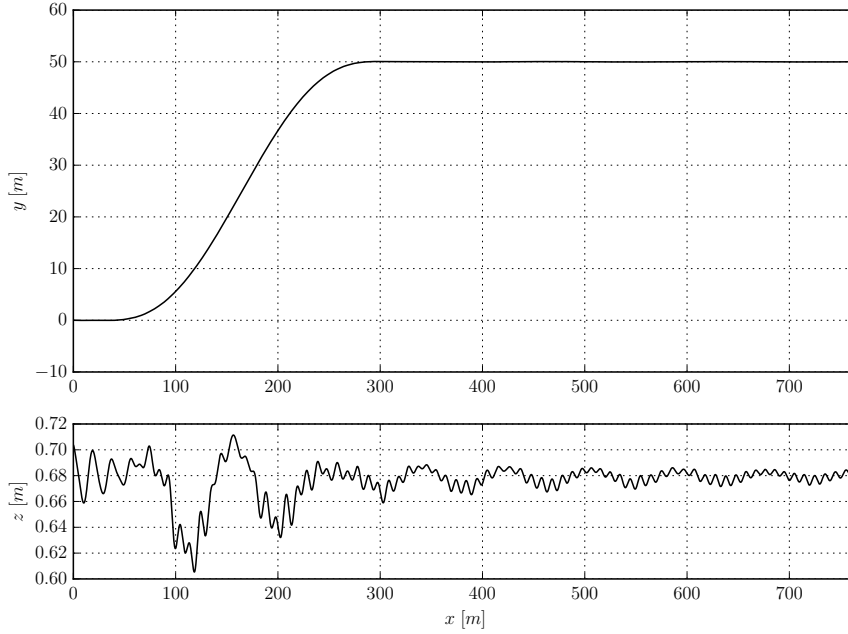

Fig. 11: Trajectories followed by the main body

Fig. 12 shows a zoom of the first graph in Fig. 11. This is done to make the oscillations in that plane visible. The zone in which the vehicle exits the second curve is shown. Two different oscillations can be seen. Two oscillations are clearly distinguishable: The hunting oscillation is the one with the largest wave length, while the shorter one is related to the irregularities of the track.

Creep velocities and creep forces and moments are presented in Fig. 13. It can be observed that creep velocities are higher when the vehicle is at the middle of the curved tracks ($t \approx 5\ s$ and $t \approx 10\ s$). The same behavior is seen for the forces and moments. Small fluctuations on the creepages and forces in the second straight track ($t > 15\ s$) are due to the vertical irregularities.

Using the proposed IMEX integrator with a $\Delta t = 1$ ms, a stable integration is achieved by a generous margin. In the same conditions, when using an explicit Euler method for velocities and the explicit trapezoidal rule for accelerations, time steps
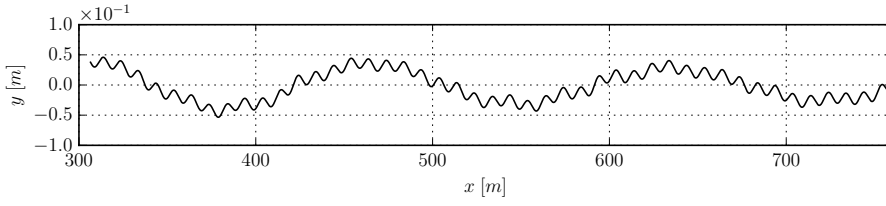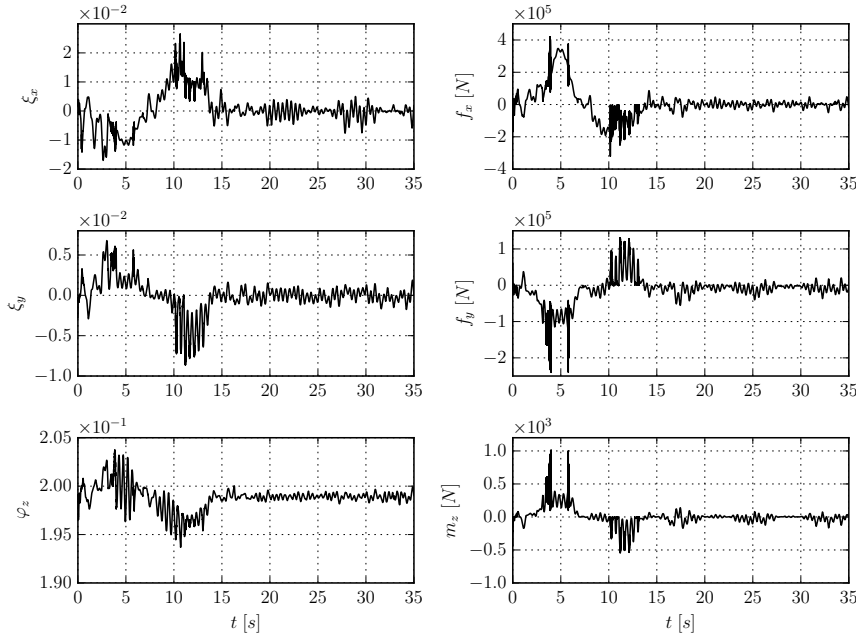
Fig. 12: Hunting oscillation in detail.

Fig. 13: Creep velocities and forces of the Rear Bogie, Front-Right wheelset

smaller than 10^{-4} ms, not compatible with a real-time performance, are required. the implicit trapezoidal rule has also been used, with showing a stable behavior around $\Delta t \approx 1$ ms with not such a generous margin.

Using the IMEX Euler method, a fine grained analysis of the computation times required by the different steps of the proposed algorithm is done. The results are given in Table 1. A seven years old *Intel Core vPro i5* @ 3500 MHz has been used for the test. From this data, it can be seen that it takes 256 μs of CPU time to complete one integration step. That is, *hard* real-time performance is achieved by a wide margin using the proposed procedures. In comparison, using the trapezoidal rule *soft* real-time performance can be achieved by a short margin.

In Fig. 14, the number of iterations required by the **q**-projection and **s**-projection steps are shown. It is noticeable that the **q**-projection only requires a single Newton-

Table 1: Results per time step (1 ms)

Task	CPU Time μs
Contact Area, Look up Kalker coeffs. and evaluate $\frac{\partial \boldsymbol{\delta}_{\dot{\mathbf{q}}}^K}{\partial \mathbf{f_i}^K} \mathbf{C}_i^K \frac{\partial \boldsymbol{\nu}_i}{\partial \dot{\mathbf{q}}}$ ($\times 8$)	30
Look up Wheel profile, rail profile and railway spline coeffs. ($\times 8$)	1
Evaluate $\mathbf{M}, \boldsymbol{\delta}$ and $\boldsymbol{\gamma}^n$	55
Dynamics Solution $\Rightarrow \dot{\mathbf{q}}, \boldsymbol{\lambda}$	103
Time Integration $\Rightarrow \mathbf{q}$	1
Evaluate $\phi^n, \phi^d, \dot{\phi}_{\dot{\mathbf{q}}}^n, \dot{\phi}_{\dot{\mathbf{q}}}^d, \dot{\phi}_{\dot{\mathbf{s}}}^d$ and β^n	29
Projection Solution $\Rightarrow \mathbf{q}, \dot{\mathbf{q}}, \mathbf{s}, \dot{\mathbf{s}}$	37
Total Time	256

Raphson[5] iteration. The same is true for the s-projection. This has required the integration of s after the integration step using an explicit Euler procedure $\mathbf{s}_{t+\Delta t} = \mathbf{s}_t + \dot{\mathbf{s}}_t \, \Delta t$ leading to an to a smaller error ($O(\Delta t^2)$) at the start of the Newton-Raphson iteration. Clearly, the increased number of iterations is coincident with the curved stretches. This result justifies the approach adopted, in which the s-projection is performed before the q-projection.
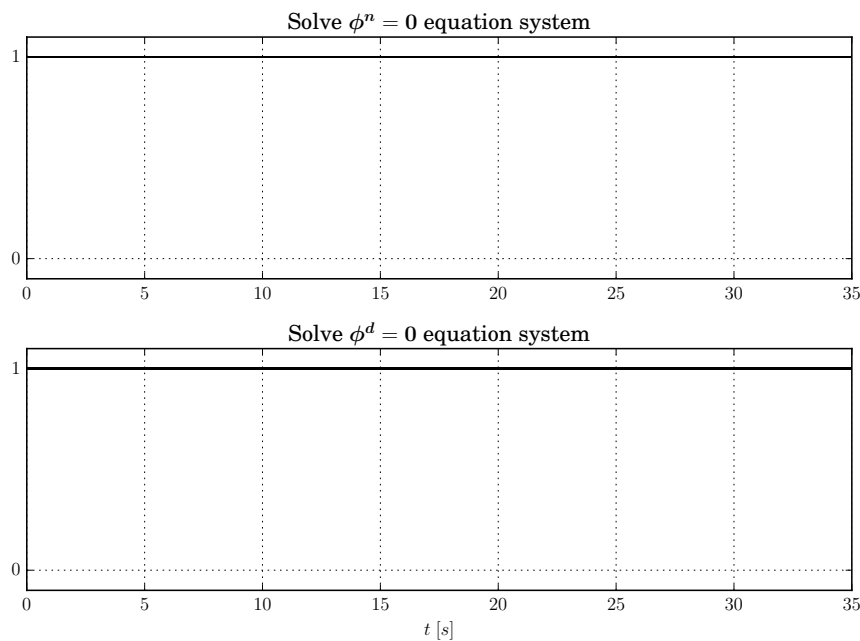

Fig. 14: Iterative steps needed by the Newton-Rapshon algorithm

---

[5] the tolerance used is 10^{-6} amounting to a negligible error of $\approx 10^{-3}$ mm for lengths

Table 2: Atoms and operation for the evaluation of the model functions

Function	Atoms	Operations
$\mathbf{M_{qq}}$	1795	10910
$\boldsymbol{\delta_q}$	3648	19015
ϕ^n	425	1709
ϕ^d	541	2150
$\dot{\phi}^n_{\dot{\mathbf{q}}}$	784	4116
$\dot{\phi}^n_{\dot{\mathbf{s}}}$	846	6668
$\dot{\phi}^d_{\dot{\mathbf{q}}}$	964	6574
$\dot{\phi}^d_{\dot{\mathbf{s}}}$	1113	8437
β^n	0	0
γ^n	3779	25870
$\frac{\partial \boldsymbol{\delta}^K_{\mathbf{q}}}{\partial \mathbf{f_i}^K} \mathbf{C}^K_i \frac{\partial \boldsymbol{\nu}_i}{\partial \dot{\mathbf{q}}}$	350	1917

In Table 2 the number of operations required for the evaluation of the different functions used by the proposed dynamic formalism are presented. The results show a correlation between the time for function evaluations and the number of operations. As a major result of this study, it can be seen that using the symbolic procedures proposed, the penalties incurred for using an exact treatment of the linear Kalker contact model are barely noticeable. Note that this is a fair comparison, as the operation count related to other dynamic and kinematic computations are very optimized, showing numbers compatible with state-of-the-art recursive formulations. This puts into perspective the relevance of symbolic methods proposed in achieving hard real-time performance in the railway dynamics simulation context.

Still, there are still some possibilities to further improve the results given in this article.

1.-The dynamic system structure shown in Fig. 8 shows a decent amount of sparsity. This sparsity is shared with the IMEX discretized dynamic matrix. Important savings can therefore be obtained using a sparse LU algorithm.

2.- In the s-projection and $\dot{\text{s}}$-projection problems, $\dot{\phi}^d_{\dot{\mathbf{s}}}$ is a maximum rank block-diagonal matrix with 4×4 blocks [1]. Therefore, its computation can be speed up by big integer factor. The solution could be easily implemented symbolically or even in parallel.

3.- Removal of the repeated evaluation of constant atoms from the symbolic functions and reuse of atoms common to different exported functions.

As commented in the introduction, at the expense of some accuracy, partial linearization [5] or base parameter reduction [6], can be used to further improve the computational performance of the model.

6 Conclusions

The purpose of the article was to test state-of-the-art methods for the symbolic modeling in the railway context. A complex locomotive running on a track with a complex and general surface geometry has been modeled and tested.

Main aspects of the symbolic methods proposed are summarized: atomization, recursive operators, points and bases structures, general parameterization, etc. Based on this methods the model is obtained using a direct implementation of the principle of virtual work. Creep forces and moments are modeled using a direct symbolic implementation of the linear Kalker model without simplifications. An Implicit-Explicit (IMEX) integrator has been proposed to cope with the contact model while attaining real-time performance. The resulting equations are solved using coordinate partitioning MSD procedures.

A very stable hard-real-time-compatible performance is achieved with a time step of 1 ms. A CPU time of 256 μs per time step is required in a seven year old *Intel Core vPro i5* @ 3500 MHz. The small time required for the determination of the creep forces when using an exact implementation of the linear Kalker model is notable. Also, the efficiency/robustness compromise of the IMEX integrator proposed is remarkable.

The results obtained show the relevance of the methods proposed for the real-time simulation of railway vehicles.

There are still obvious possibilities to improve on the results presented in this work: better sharing of atoms, constant atom revaluation, sparse linear solver implementation and parallelization are the most obvious. On top of this, with a small accuracy penalty, techniques such as partial-linearization and parameter reduction can be used to improve the results presented even further.

7 Acnowledgements

References

1. J.C. Samin and P. Fisette. *Symbolic Modeling of Multibody Systems*. Springer Netherlands, 2003.
2. A. Plaza, J. Ros, and X. Iriarte. Symbolic multibody modeling based on recursive multibody operators and expression atomization. In *ECCOMAS Thematic Conference on Multibody Dynamics 2015*, June 29 - July 2 2015.
3. AhmedA. Shabana, Mahmoud Tobaa, Hiroyuki Sugiyama, and KhaledE. Zaazaa. On the computer formulations of the wheel/rail contact problem. *Nonlinear Dynamics*, 40(2):169–193, 2005.
4. Monica Malvezzi, Enrico Meli, Stefano Falomi, and Andrea Rindi. Determination of wheel–rail contact points with semianalytic methods. *Multibody System Dynamics*, 20(4):327–358, 2008.
5. J. F. Aceituno, J. L. Escalona, and Daniel García-Vallejo. Partially-linearized multibody equations of railroad vehicles on arbitrary tracks for on-board applications. In *ECCOMAS Thematic Conference on Multibody Dynamics 2015*, June 29 - July 2 2015.
6. X. Iriarte, J. Ros, V. Mata, and A. Plaza. Multibody model reduction by parameter elimination. In *ECCOMAS Thematic Conference on Multibody Dynamics 2015*, June 29 - July 2 2015.
7. Aitor Plaza. *Symbolic modeling of multibody systems for real-time*. PhD thesis, Public Universisty of Navarre, 2016.
8. Javier Ros, Luis Arrondo, Javier Gil, and Xabier Iriarte. Lib_3d_mec–ginac, a library for symbolic multibody dynamics.
9. W. Khalil and J.F. Kleinfinger. Minimum operations and minimum parameters of the dynamic models of tree structure robots. *Journal of Robotics and Automation*, RA-3(6):517–526, 1987.

10. J. Ros, X. Iriarte, and V. Mata. 3d inertia transfer concept and symbolic determination of the base inertial parameters. *Mechanism and Machine Theory*, 49:284–297, 2012.
11. Javier Ros, Aitor Plaza, Xabier Iriarte, and Jokin Aginaga. Inertia transfer concept based general method for the determination of the base inertial parameters. *Multibody System Dynamics*, 34(4):327–347, 2015.
12. Eduardo Diez Santa Coloma. Nociones de ferrocarril. Breve historia de FEVE. `http://www.feve.es`, 2009.
13. João Pombo and Jorge AC Ambrósio. General spatial curve joint for rail guided vehicles: kinematics and dynamics. *Multibody system dynamics*, 9(3):237–264, 2003.
14. Everts Andersson, Mats Berg, and Sebastian Stichel. *Rail vehicle dynamics-fundamentals and guidelines*. KTH Railway Technology, 1999.
15. R. A. Wehage and E. J. Haug. Generalized coordinate partitioning for dimension reduction in analysis of constrained dynamic systems. *Journal of Mechanical Design*, 104(4):247–255, 1982.
16. E. J. Haug. *Computer Aided Kinematics and Dynamics of Mechanical Systems. Vol. 1: Basic Methods*. Allyn & Bacon, Inc., Needham Heights, MA, USA, 1989.
17. Ahmed A Shabana, Khaled E Zaazaa, José L Escalona, and Jalil R Sany. Development of elastic force model for wheel/rail contact problems. *Journal of sound and vibration*, 269(1):295–325, 2004.
18. João Pombo, Jorge Ambrósio, and Miguel Silva. A new wheel–rail contact model for railway dynamics. *Vehicle System Dynamics*, 45(2):165–189, 2007.
19. Jury Auciello, Enrico Meli, Stefano Falomi, and Monica Malvezzi. Dynamic simulation of railway vehicles: wheel/rail contact analysis. *Vehicle system dynamics*, 47(7):867–899, 2009.
20. Martin Arnold. *Simulation algorithms in vehicle system dynamics*. Univ., Fachbereich Mathematik und Informatik, 2004.
21. Ahmed A Shabana and Jalil R Sany. An augmented formulation for mechanical systems with non-generalized coordinates: application to rigid body contact problems. *Nonlinear dynamics*, 24(2):183–204, 2001.
22. P Fisette, JM Péterkenne, B Vaneghem, and JC Samin. A multibody loop constraints approach for modelling cam/follower devices. *Nonlinear Dynamics*, 22(4):335–359, 2000.
23. Joost Jacques Kalker. *On the rolling contact of two elastic bodies in the presence of dry friction*. PhD thesis, TU Delft, Delft University of Technology, 1967.
24. M Bozzone, E Pennestrì, and P Salvini. A lookup table-based method for wheel–rail contact analysis. *Proceedings of the Institution of Mechanical Engineers, Part K: Journal of Multi-body Dynamics*, 225(2):127–138, 2011.
25. Hiroyuki Sugiyama, Yoshimitsu Tanii, Yoshihiro Suda, Minoru Nishina, Hisanao Komine, Takefumi Miyamoto, Hisayo Doi, and Hua Chen. Wheel/rail contact geometry on tight radius curved track: simulation and experimental validation. *Multibody System Dynamics*, 25(2):117–130, 2011.
26. José L Escalona and Javier F Aceituno. Modeling wheel-rail contact with pre-calculated lookup tables in arbitrary-geometry tracks with irregularities. In *ASME 2015 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, pages V006T10A078–V006T10A078. American Society of Mechanical Engineers, 2015.
27. S. Iwnicki, editor. *Handbook of Railway Vehicle Dynamics*. Taylor & Francis, 2006.
28. Joost J Kalker. The tangential force transmitted by two elastic bodies rolling over each other with pure creepage. *Wear*, 11(6):421–430, 1968.
29. Uri M Ascher, Steven J Ruuth, and Brian TR Wetton. Implicit-explicit methods for time-dependent partial differential equations. *SIAM Journal on Numerical Analysis*, 32(3):797–823, 1995.
30. Martin Arnold, Bernhard Burgermeister, and Alexander Eichberger. Linearly implicit time integration methods in real-time applications: Daes and stiff odes. *Multibody System Dynamics*, 17(2):99–117, 2007.
31. Scott R Fulton. Semi-implicit time differencing. *Technical report*, 2004.