ECCOMAS Congress 2016
VII European Congress on Computational Methods in Applied Sciences and Engineering
M. Papadrakakis, V. Papadopoulos, G. Stefanou, V. Plevris (eds.)
Crete Island, Greece, 5–10 June 2016

VIRTUAL TESTBENCH FOR THE OMNI WHEEL DYNAMICS SIMULATION: NEW CONTACT TRACKING ALGORITHM

Ivan Kosenko[1], **Sergey Stepanov**[2], **Kirill Gerasimov**[3], **and Mikhail Stavrovskiy**[4]

[1]Moscow Aviation Institute (National Research University)
Volokolamskoe shosse 4, 125993, Moscow, Russia
e-mail: kosenko@ccas.ru

[2]Dorodnicyn Computing Centre of Russian Academy of Sciences
Vavilov street 40, 119333, Moscow, Russia
e-mail: stepsj@ccas.ru

[3]Lomonosov Moscow State University
GSP-1, 1-52, Leninskie Gory, 119991, Moscow, Russia
e-mail: kiriger@gmail.com

[4]STAN Group
1 Vadkovsky lane, 127994, Moscow, Russia
e-mail: stavrov@list.ru

Keywords: Object-Oriented Modeling, Modeling of Mechanical Constraint, Omni Wheel, Contact Tracking, Unilateral Constraint, Roller Inclination, Model of Friction.

Abstract. *General approach for constructing the computer models of the multibody system dynamics with unilateral constraints and friction is under consideration. Base dynamical abstractions, classes, are described which provide a model building using object-oriented paradigm for the design of dynamic objects virtual prototypes.*

Contact tracking block is the most important one of any algorithm for simulation the mechanical contact. This block provides permanent computations of the contact point/patch where the process of bodies mechanical interaction takes place. A general formulation of contact in algebraic and/or differential forms is under analysis.

The omni directional wheel dynamical testbench is under construction then. This model is built in a way such that the wheel disc equipped by rollers along its rim keeps its vertical orientation permanently. Such an arrangement of the virtual testbench provides indeed the simulation of the wheel motion like it is a part of any virtual omni vehicle.

Using the omni wheel permanent vertical orientation we are able to build up an efficient algorithm for contact tracking between the roller and the floor. The case of so-called angled (being preliminarily rotated) rollers is also included. The angle of roller inclination to the wheel plane is possible to be varied, as a parameter of the dynamical problem, in frame of wide range of its values. This approach is a natural generalization of its previous version which was implemented for the case of zero angle of the roller prerotation.

1 INTRODUCTION

A construct of the omni vehicle dynamical model has been presented in [1], see also papers [2, 3]. Concept of an omni wheel was proposed in [4]. Simplified model for mounting of roller on the wheel disk has been considered there: the roller axis of symmetry assumed to belong to the disk, or equivalently an angle of inclination, denote it by ψ, for the roller axis to the wheel plane is equal to zero.

In engineering applications nevertheless one may encounter frequently a situation with $\psi > 0$. We proposed in [1] fast algorithm for tracking a contact provided the omni wheel keeps vertical orientation of its plane (in frame of the whole vehicle construct). Thus actual is a task for building up the contact tracking algorithm also for the case of $\psi > 0$. This task has been completed in this paper. To reach this goal we accept the working model of a virtual testbench consisting of one wheel equipped by rollers along its rim. One can see easily this simplification has mostly methodical nature and does not prevent us to integrate all the construct back into the whole vehicle having generally several omni wheels already analyzed previously [1].

2 GENERAL DESCRIPTION OF THE MULTIBODY DYNAMICS MODEL ARCHITECTURE

2.1 Object-oriented concept

First of all, in frame of the general approach let us build up the multibody system (MBS) dynamical model using object-oriented paradigm with Modelica language [5]. Consider the MBS consisting of $m+1$ bodies $B_0, \ldots, B_m$. Represent it as a set $\mathcal{B} = \{B_0, \ldots, B_m\}$. Here B_0 assumed to be a base body. We suppose B_0 to be connected with an inertial frame of reference, or to have a known motion with respect to the inertial frame of reference. For example one can imagine the base body as a rotating platform, or as a vehicle performing its motion according to a given law. For definiteness and simplicity we suppose in the following text that all state variables describing the rigid bodies motion always refer to one fixed inertial coordinate system connected to the base body by default.

Some bodies of the MBS assumed connected by mechanical constraints. Suppose all constraints compose the set $\mathcal{C} = \{C_1, \ldots, C_n\}$. We include in our considerations constraints of the following types: holonomic/nonholonomic, scleronomic/rheonomic, bilateral/unilateral.

Thus one can uniquely represent a structure of the MBS via a undirected graph $G = (\mathcal{B}, \mathcal{C}, \mathcal{I})$. Here $\mathcal{I} \subset \mathcal{C} \times \mathcal{B}$ is an incidence relation setting in a correspondence the vertex incident to every edge $C_i \in \mathcal{C}$ of the graph. According to physical reasons it is easy to see that for any mechanical constraint C_i there exist exactly two bodies $B_k, B_l \in \mathcal{B}$ connected by this constraint.

It is clear that consideration of the graph G does not provide a structural information sufficient for the MBS dynamics description. Indeed, in addition to the force interaction represented usually by wrenches between bodies B_k, B_l through the constraint C_i there exist kinematic conditions specific for different kinds of constraints. Wrenches themselves can be represented in turn by constraint forces and constraint torques couples. These forces and couples are connected by virtue of Newton's third law of dynamics.

Thus if the system of ODEs for translatory-rotary motion can be associated with the object of a model corresponding to rigid body, then the system of the algebraic equations can be naturally associated with the object of a model corresponding to constraint. Note that according to above consideration the set of algebraic equations comprises relations for constraint wrenches, and kinematic relations depending on the certain type of constraints.

Thus all the "population" of any MBS model is reduced to objects of two classes: *RigidBody*

(objects $B_0, \ldots, B_m$), *Constraint* (objects $C_1, \ldots, C_n$). According to this approach simulation of the whole system behavior reduces to permanent information interaction between the objects of two considered types. Within the frame of Newton's laws of dynamics one can construct the MBS as a communicative network for this interaction. In this case the objects of bodies "feel" the action of other ones through corresponding objects of constraints.

Physical interactions are conducted in models due to objects splitted also in two classes of ports: *WrenchPort, KinematicPort.* The first one is to be used to transfer wrench. In addition, *WrenchPort* has to be used for transferring the information about current location of the point constraint force acts upon.

In our idealized model the force interaction between bodies supposed exactly at a geometric point. Its coordinates are fed outside constraint object through *WrenchPort* permanently in time.

Now it is possible to describe an architecture of information interactions within the particular constraint C_i corresponding to an individual edge of graph G, see Figure 1.

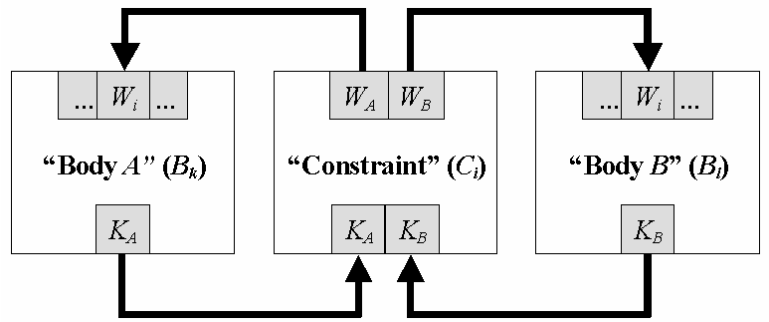

Figure 1: Architecture of constraint

KinematicPort is to be used to transfer the data of rigid body kinematics: configuration (position of center of mass, orientation), velocity (velocity of the center of mass, angular rate), and acceleration (acceleration of the center of mass, angular acceleration) containing in particular information about twist. When getting force information through ports $W_1, \ldots, W_s$ from the incident objects of class *Constraint* the object of class *RigidBody* simultaneously generates, due to an integrator, kinematic information being fed outside through the port K. On the other hand every object of class *Constraint* gets kinematic data from the objects corresponding to bodies connected by the constraint under consideration through its two "input" ports K_A, K_B. Simultaneously using the system of algebraic equations this object generates information concerning wrenches, and transmits the data to "output" ports W_A, W_B for the further transfer to objects of bodies under constraint.

In base class *RigidBody* dynamics of rigid body is described here by means of Newton's differential equations for the body mass center, and by Euler's differential equations for the rotary motion. Note that to be able to have an invariant description of the rotary motion one can use an excellent tool: quaternion algebra $\mathbf{H}$. In this case we "lift" the configuration manifold from $SO(3)$ to $S_3 \subset \mathbf{H}$ and then implement dynamics of rotation in flat space $\mathbf{H} \cong \mathbf{R}^4$ taking into account that S_3 is an invariant manifold of the rotary dynamics redefined on $\mathbf{H}$. In this way we have only one flat chart $\mathbf{H}$ for the underlying due to double covering configuration manifold $SO(3)$, and need not in any special choices of the configuration angles or anything like that.

The double covering $S_3 \to SO(3)$, $\mathbf{q} \mapsto T$ mentioned implemented inside the *RigidBody*

class by the known formula

$$T = \frac{1}{|\mathbf{q}|^2} \cdot \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1 q_2 - q_0 q_3) & 2(q_1 q_3 + q_0 q_2) \\ 2(q_0 q_3 + q_1 q_2) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_2 q_3 - q_0 q_1) \\ 2(q_1 q_3 - q_0 q_2) & 2(q_0 q_1 + q_2 q_3) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix},$$

where q_0, q_1, q_2, q_3 are the Euclidean coordinates of the quaternion $\mathbf{q}$ and $|\mathbf{q}|$ is its Euclidean norm. The rotation matrix T is fed outside the object through the *KinematicPort* permanently in time. The Euler equations are constructed using quaternion algebra in a way described in [6].

Note that according to our technology of the constraint construction two connected bodies are identified by convention with the letters A and B fixed for each body. All kinematic and dynamic variables and parameters concerned one of the bodies are equipped with the corresponding letter as a subscript.

All objects of the class *Constraint* must have classes-inheritors as subtypes of a corresponding base class. According to Newton's third law this superclass must contain the equations of the form

$$\mathbf{F}_A + \mathbf{F}_B = \mathbf{0}, \quad \mathbf{M}_A + \mathbf{M}_B = \mathbf{0}. \tag{1}$$

in its behavioral section. Here arrays $\mathbf{F}_A$, $\mathbf{M}_A$ and $\mathbf{F}_B$, $\mathbf{M}_B$ represent constraint forces and torques "acting in directions" of bodies A and B correspondingly. Kinematic equations for different types of constraints are to be added to Eqs. (1) in different classes-inheritors corresponding to these particular types of constraints.

2.2 Example of a joint constraint

Class *Joint* plays an important auxiliary role in the future model of an omni wheel we will build. *Joint* is a model derived from the base class *Constraint*. Remind [7] that in order to make a complete definition of the constraint object behavior for the case of rigid bodies one has to compose a system of twelve algebraic equations with respect to (w. r. t.) twelve coordinates of vectors $\mathbf{F}_A$, $\mathbf{M}_A$, $\mathbf{F}_B$, $\mathbf{M}_B$ constituting the wrenches acting upon the connected bodies.

First six equations (1) always present in the base model *Constraint* due to Newton's third law. For definiteness suppose these six equations are used to express six components of $\mathbf{F}_B$, $\mathbf{M}_B$ depending on $\mathbf{F}_A$, $\mathbf{M}_A$. Thus six components of $\mathbf{F}_A$, $\mathbf{M}_A$ remain unknown. To determine them each constraint of rigid bodies need in six additional independent algebraic equations. These equations can include components of force and torque directly, or be derived from the kinematic relations corresponding to specific type of the constraint.

In the case of the joint constraint being investigated here let us represent the motion of the body B as a compound one consisting of the body A convective motion w. r. t. an inertial frame of reference, and a relative motion w. r. t. the body A. An absolute motion is one of the body B w. r. t. the inertial system.

Define the joint constraint with help of the following parameters: (a) a unit vector $\mathbf{n}_A$ defining in the body A an axis of the joint; (b) a vector $\mathbf{r}_A$ fixed in the body A and defining a point which constantly stays on the axis of the joint; (c) a vector $\mathbf{r}_B$ fixed in the body B and defining a point which also constantly stays on the axis of the joint. The main task of the base joint class is to keep always in coincidence the geometric axes fixed in each of the bodies.

First of all one has to compute the radii vectors of the points fixed in the bodies w. r. t. inertial system

$$\mathbf{R}_\alpha = \mathbf{r}_{O_\alpha} + T_\alpha \mathbf{r}_\alpha \quad (\alpha = A, B),$$

where [7] $\mathbf{r}_{O_\alpha}$ is the position of the α-th body center of mass, T_α is its current matrix of rotation. The joint axis has the following components

$$\mathbf{n}_{Ai} = T_A \mathbf{n}_A$$

in the inertial frame of reference. According to the equation for relative velocity for the marked point of the body B defined by the position $\mathbf{R}_B$ we have

$$\mathbf{v}_{Ba} = \mathbf{v}_{Be} + \mathbf{v}_{Br}, \quad \mathbf{v}_{Ba} = \mathbf{v}_{O_B} + [\boldsymbol{\omega}_B, T_B \mathbf{r}_B], \quad \mathbf{v}_{Be} = \mathbf{v}_{O_A} + [\boldsymbol{\omega}_A, \mathbf{R}_B - \mathbf{r}_{O_A}], \tag{2}$$

where $\mathbf{v}_{Ba}$, $\mathbf{v}_{Be}$, $\mathbf{v}_{Br}$ are an absolute, convective, and relative velocities of the body B marked point, $\boldsymbol{\omega}_A$, $\boldsymbol{\omega}_B$ are the bodies angular velocities.

Furthermore, according to the computational experience of the dynamical problems simulation the precompiler work is more regular if the kinematic equations are expressed directly through accelerations. Indeed, otherwise the compiler tries to perform the formal differentiation of equations for the velocities when reducing an index of the total DAE system. Frequently this leads to the problems either in time of translation or when running the model.

Thus using the known Euler formulae for the rigid body kinematics and the Coriolis theorem we obtain an equations for the relative linear acceleration in the form

$$\begin{aligned} &\mathbf{a}_{Ba} = \mathbf{a}_{O_B} + [\boldsymbol{\varepsilon}_B, T_B \mathbf{r}_B] + [\boldsymbol{\omega}_B, [\boldsymbol{\omega}_B, T_B \mathbf{r}_B]], && \mathbf{a}_{Ba} = \mathbf{a}_{Be} + 2[\boldsymbol{\omega}_A, \mathbf{v}_{Br}] + \mathbf{a}_{Br}, \\ &\mathbf{a}_{Be} = \mathbf{a}_{O_A} + [\boldsymbol{\varepsilon}_A, \mathbf{R}_B - \mathbf{r}_{O_A}] + [\boldsymbol{\omega}_A, [\boldsymbol{\omega}_A, \mathbf{R}_B - \mathbf{r}_{O_A}]], && \mathbf{a}_{Br} = \mu \mathbf{n}_{Ai}, \end{aligned} \tag{3}$$

where $\mathbf{a}_{Ba}$, $\mathbf{a}_{Be}$, $\mathbf{a}_{Br}$ are an absolute, convective, and relative accelerations of the body B marked point, $\boldsymbol{\varepsilon}_A$, $\boldsymbol{\varepsilon}_B$ are the bodies angular accelerations, $\mathbf{v}_{Br}$ is a relative velocity of the body B marked point, $\boldsymbol{\omega}_A$, $\boldsymbol{\omega}_B$ are the bodies angular velocities.

We also need in an analytic representation of the conditions that the only projections of the bodies angular velocities and accelerations having a differences are ones onto the joint axis. Corresponding equations have the form

$$\boldsymbol{\omega}_B = \boldsymbol{\omega}_A + \boldsymbol{\omega}_r, \quad \boldsymbol{\varepsilon}_B = \boldsymbol{\varepsilon}_A + [\boldsymbol{\omega}_A, \boldsymbol{\omega}_r] + \boldsymbol{\varepsilon}_r, \quad \boldsymbol{\varepsilon}_r = \lambda \mathbf{n}_{Ai}, \tag{4}$$

where $\boldsymbol{\omega}_r$, $\boldsymbol{\varepsilon}_r$ are the relative angular velocities and accelerations.

Besides the kinematic scalars μ, λ we will need in their reciprocal values $F = (\mathbf{F}_A, \mathbf{n}_{Ai})$, $M = (\mathbf{M}_A, \mathbf{n}_{Ai})$ correspondingly. Note that the class described above is a partial one (doesn't yet complete the constraint definition) and can be used to produce any imaginable model of the joint type constraint. To obtain a complete description of the joint model one has to add to the behavioral section exactly two equations. One of them is to define one of the values μ, F (translatory case). Other equation is intended to compute one of the values λ, M (rotary case).

Consider one example of joint classes among others being derived from the *Joint* model for the particular, revolute, type of joint being applied in the omni wheel construct. This model *FixedIdealJoint* is defined by the equations

$$\mu = 0, \quad M = 0$$

and prevents the relative motion along the joint axis but allows free rotation about it. It is exactly a revolute joint without any control for the rotary motion.

It is clear one can create a lot of other different combinations of equations to construct the joint constraints of any desirable type needed in engineering applications. The only issue one

has to take into account when constructing the kinematic pair that if there exists an obstacle for the relative motion eliminating one DOF then it causes an additional scalar kinematic equation "lifted" to the level of accelerations. Reciprocally if the corresponding motion is possible then the developer has to include the scalar equation instead imposing the condition on the matching generalized effort, force or torque what applicable.

2.3 Bond graph concept

Referring for the details to the papers [8, 9] let us give here a brief outline of the multibond graph scheme concerning the general constraint architecture under consideration, see Figure 2. Remind that each multibond is a pair comprising twist as its flow component, while the wrench plays a role of the effort component.

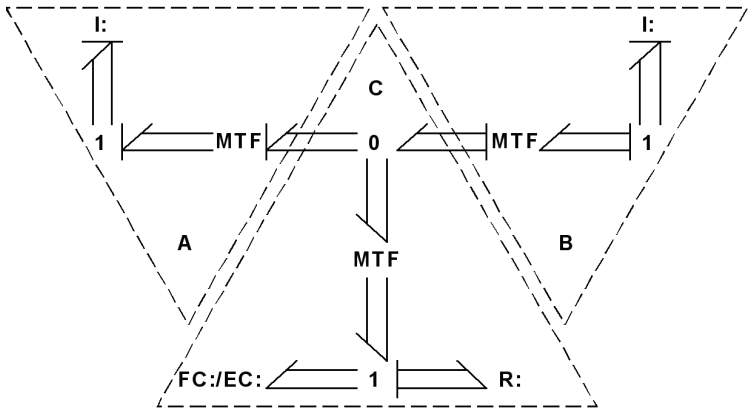

Figure 2: Architecture of constraint: bond graph representation

Describing Figure 2 first of all one has to highlight the multibond true nature. Considering the rigid body as a mechanical system with six degrees of freedom (DOFs) we note that the flow part, twist, of any multibond represents an object tangent to configuration manifold $\mathcal{D} = \mathbf{R}^3 \times SO(3)$ which is in turn the rigid body displacement group [10]. In such a way the multibond effort component representing the wrench of reaction forces is indeed the cotangent object, 1–form, at the same point of $\mathcal{D}$. Thus the multibond power is a natural pairing, the value of 1–form, wrench, evaluated on the configuration manifold tangent vector, twist.

In the way under description we follow the ideas of the paper [10] leaving the rigid body kinematics unsplitted to its translatory and rotary parts. To this end we go farther than the usual approach for the bond graph applications when frequently the developer first keeps in his mind the equations and then builds up the corresponding bond graph. Unlike to this we first construct the bond graph connecting it to the physical nature of the problem, and then the system of DAEs is to be assembled mostly by compiler without any participation of the developer. In this latter case physical invariance of the model independent on the choice of (generalized) coordinates is maintained.

Regarding the general scheme depicted in Figure 2 with its connection to the family of the joint constraint classes we can conclude that the equations (3), (4) together implement implicitly the constraint transformer, central modulated transformer in triangle C of Figure 2, to the joint local coordinate system and four scalar flow constraints forbidding relative translatory and rotary motions in the direction orthogonal to the joint axis. For derived classes only two free scalar bonds remain.

Here we encounter the known complementarity rules [11] in a way similar to one described in [8, 7]. In our context the variables in the pairs (μ, F), (λ, M) are mutually complement, where one of μ, λ is to be utilized for the flow constraint and one of F, M is used to compose the effort constraint. All the variables mentioned complete the set of constraints for the remaining yet unused joint axis creating thus two final scalar constraint elements in the bond graph of Figure 2.

Namely, equations (3) implementing the Coriolis theorem for accelerations simultaneously implement, in an implicit manner, two scalar flow constraints, FC-elements, from the bottom left corner of the multibond graph model in Figure 2. These flow constraints are constructed, due to compiler restrictions, using accelerations instead of velocities being used in a classic bond graph approach. The constraints have an obvious kinematic sense: they prevent the relative motion of the body B marked point in two directions normal to the joint axis fixed in the body A.

In addition, the equations (4) implement two other scalar flow constraints, this time for the rotary motion. These constraints forbid the relative rotation of the body B w. r. t. body A about two axes each normal to the joint axis mentioned above which is rigidly connected with the body A.

Note, that the construct of equations (3) and (4) is such that they allow the body B relative motion along and about the joint axis of the body A thus implementing the kinematic pair with two DOFs. Returning to Figure 2 of the general constraint multi-bondgraph we can conclude that the vertical multibond attached to 0-junction implements flow variables corresponding to the relative body B motion w. r. t. body A in inertial coordinates. Such a description supposes an existence of the special coordinates reference frame connected with the body A at its joint constraint marked point. The transformation to these coordinates is implemented exactly via corresponding transformer, central in the triangle block C. The transformer itself nests in formulae of equations (3) and (4).

The derived joint class example described above is to close the system of kinematic equations (3) and (4) completing them by two scalar additional equations, each playing a role of an either FC-element, like $\mu = 0$, or EC-element, like $M = 0$. Any time to be able to construct a consistent system of equations for the total model we have to follow the guidelines similar to ones of the complementarity rules.

These latter correspond to the notions of the bond graph theory in a natural way. Indeed, the theory of bond graphs is based on the energy interactions. Every our multibond being an energy/power conductor reflects complementarity by its twist/wrench duality. To close the total DAE system for the model under development we have to "close" or rather to "seal" each free scalar bond in EC/FC-element of the block C in Figure 2 by the corresponding one scalar equation for flow or effort variable. Thus here we outline the main rule to compose equations for the models of constraints for MBS of any type in a consistent way when applying the object-oriented approach.

3 GENERAL DESCRIPTION OF THE CONTACT TRACKING ALGORITHM

We outline here general approach for implementing contact tracking algorithm in case of the unilateral constraint. As it was already mentioned we use the so called complementarity rules [11] as a base for the unified description of the unilateral constraint. Taking into account complementarity rules one can see easily that any constraint always is defined by the three scalar equations. To derive these equations consider first local geometry of the problem, see Figure 3.

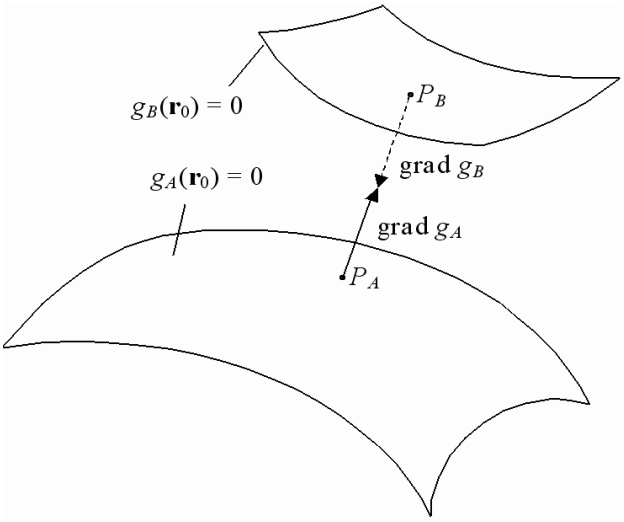

Figure 3: Area of Constraint

Outer surfaces supposed to be defined with respect to principal central axes of corresponding bodies by the equations

$$f_\alpha(\mathbf{r}_\alpha) = 0 \quad (\alpha = A, B).$$

Then in inertial frame of reference for the whole MBS these equations will take the form

$$g_\alpha(\mathbf{r}_0) = 0 \quad (\alpha = A, B),$$

where

$$g_\alpha(\mathbf{r}_0) = f_\alpha\left[T_\alpha^*(\mathbf{r}_0 - \mathbf{r}_{O_\alpha})\right] \quad (\alpha = A, B),$$

with $\mathbf{r}_{O_A}$, $\mathbf{r}_{O_B}$ being a vectors of masscenters positions O_A, O_B for the bodies A and B, and T_A, T_B mean an orthogonal matrices for current bodies orientations. An asterisk denotes conjugating what equivalent to inverting of matrix for the case of orthogonality. Thus the functions $g_A(\mathbf{r}_0)$, $g_B(\mathbf{r}_0)$ depend upon the time indirectly through the variables $\mathbf{r}_A$, $\mathbf{r}_B$, T_A, T_B.

Constraint object of our model is to compute at each current instant positions of the points P_A and P_B which are the nearest ones for interacting bodies A and B. By virtue of above assumptions such points are to be evaluated in a unique way. Denote the radii vectors of these points with respect to inertial frame of reference by $\mathbf{r}_{P_A}$, $\mathbf{r}_{P_B}$. Then using simple geometric considerations for the coordinates of the mentioned vectors one can derive the following system of algebraic equations

$$\begin{aligned} \operatorname{grad} g_A(\mathbf{r}_{P_A}) &= \lambda \cdot \operatorname{grad} g_B(\mathbf{r}_{P_B}), \\ \mathbf{r}_{P_A} - \mathbf{r}_{P_B} &= \mu \cdot \operatorname{grad} g_B(\mathbf{r}_{P_B}), \\ g_A(\mathbf{r}_{P_A}) &= 0, \\ g_B(\mathbf{r}_{P_B}) &= 0. \end{aligned} \tag{5}$$

Here the first equation is a condition of the collinearity for the normals to outer surfaces at the points being computed. The second condition requies the points P_A, P_B to be relocated on the straight line collinear to the normals above.

One can verify easily that one can compute the gradients using the formulae

$$\operatorname{grad} g_\alpha(\mathbf{r}_{P_\alpha}) = T_\alpha \operatorname{grad} f_\alpha\left[T_\alpha^*(\mathbf{r}_{P_\alpha} - \mathbf{r}_{O_\alpha})\right], \tag{6}$$

where $\alpha = A, B$. It easy to see the system (5) consists of eight scalar equations and has eight scalar unknown variables: x_{P_A}, y_{P_A}, z_{P_A}, x_{P_B}, y_{P_B}, z_{P_B}, λ, μ. Variables λ, μ are an auxiliary ones. And the variable μ plays a role of contact detector at that: if $\mu > 0$ then bodies are disconnected, otherwise if $\mu \leq 0$ then bodies are in contact. Equations (5) one uses either without or with a presence of the contact of bodies A, B. In a latter case one uses the equation $\mu = 0$ instead of one of the surfaces equations.

According to computational experience it is more reliable and convenient if one use equations of constraint in a differential form than in an algebraic one (5) instead. Such an approach is used frequently also when analyzing the properties of mechanical systems.

Analytical computations show that one can substitute the equations (5) by the differential ones of the form

$$\frac{d\mathbf{r}_{P_A}}{dt} = \mathbf{v}_A, \quad \frac{d\mathbf{r}_{P_B}}{dt} = \mathbf{v}_B, \quad \frac{d\lambda}{dt} = v_\lambda, \quad \frac{d\mu}{dt} = v_\mu, \tag{7}$$

where the variables of right hand sides are calculated implicitly by help of system of linear algebraic equations of the form

$$\begin{aligned}
[\boldsymbol{\omega}_A, \operatorname{grad} g_A] + T_A \operatorname{Hess} f_A T_A^* (\mathbf{v}_A - \mathbf{v}_{P_A}) - \\
\lambda \{[\boldsymbol{\omega}_B, \operatorname{grad} g_B] + T_B \operatorname{Hess} f_B T_B^* (\mathbf{v}_B - \mathbf{v}_{P_B})\} - \\
v_\lambda \operatorname{grad} g_B &= \mathbf{0}, \\
\mathbf{v}_A - \mathbf{v}_B - \\
\mu \{[\boldsymbol{\omega}_B, \operatorname{grad} g_B] + T_B \operatorname{Hess} f_B T_B^* (\mathbf{v}_B - \mathbf{v}_{P_B})\} - \\
v_\mu \operatorname{grad} g_B &= \mathbf{0}, \\
\operatorname{grad} g_A \cdot \mathbf{v}_A - \operatorname{grad} f_A T_A^* \mathbf{v}_{P_A} &= 0, \\
\operatorname{grad} g_B \cdot \mathbf{v}_B - \operatorname{grad} f_B T_B^* \mathbf{v}_{P_B} &= 0.
\end{aligned} \tag{8}$$

When using this latter, i. e. differential form of a constraint one needs to set a consistent initial values for the variables $\mathbf{r}_{P_A}$, $\mathbf{r}_{P_B}$, λ, μ at the start time instant of simulation. Vectors $\mathbf{v}_{P_A}$ and $\mathbf{v}_{P_B}$ in the system (8) of algebraic linear equations are the velocities of a nearest points of the bodies and are computed according to Euler's formula

$$\mathbf{v}_{P_\alpha} = \mathbf{v}_{O_\alpha} + [\boldsymbol{\omega}_\alpha, \mathbf{r}_{P_\alpha} - \mathbf{r}_{O_\alpha}] \quad (\alpha = A, B). \tag{9}$$

Matrices $\operatorname{Hess} f_A$, $\operatorname{Hess} f_B$ are ones of Hesse for corresponding functions and have the form

$$\operatorname{Hess} f_\alpha = \begin{pmatrix} \dfrac{\partial^2 f_\alpha}{\partial x_\alpha^2} & \dfrac{\partial^2 f_\alpha}{\partial x_\alpha \partial y_\alpha} & \dfrac{\partial^2 f_\alpha}{\partial x_\alpha \partial x_\alpha} \\ \dfrac{\partial^2 f_\alpha}{\partial y_\alpha \partial y_\alpha} & \dfrac{\partial^2 f_\alpha}{\partial y_\alpha^2} & \dfrac{\partial^2 f_\alpha}{\partial y_\alpha \partial z_\alpha} \\ \dfrac{\partial^2 f_\alpha}{\partial z_\alpha \partial x_\alpha} & \dfrac{\partial^2 f_\alpha}{\partial z_\alpha \partial y_\alpha} & \dfrac{\partial^2 f_\alpha}{\partial z_\alpha^2} \end{pmatrix},$$

with $\alpha = A, B$. Normal vector

$$\mathbf{n}_A = \frac{\operatorname{grad} g_A}{|\operatorname{grad} g_A|} \tag{10}$$

will play an important role in the further course. Normal for an outer surface of the body A is chosen here for definiteness. One can use the vector $\mathbf{n}_B$ as well.

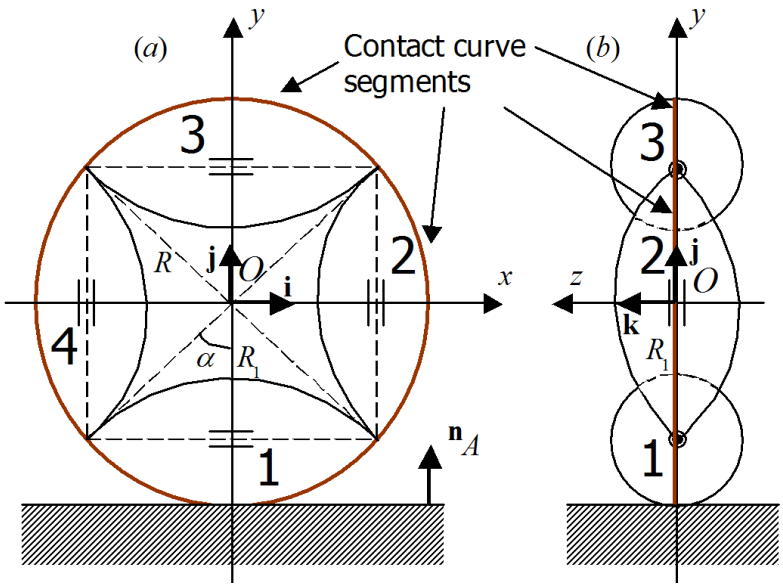

Figure 4: The omni wheel vertically aligned: (a) lateral view; (b) front view.

4 VIRTUAL OMNI WHEEL TESTBENCH ARRANGEMENT

4.1 Preliminaries

Let us consider an omni wheel now. Omni wheel for the case of $\psi = 0$ is shown in Figure 4. There one can see the lateral view, fragment (a), of the wheel being equipped by four axisymmetrical rollers, each having a shape of the circular spindle. These rollers have been enumerated by their numbers. Each roller is connected to the wheel by a joint which axis coincides with the roller axis of rotation. These latter axes both are orthogonal to the wheel radius exiting from the central point O and passing through the the roller central point. So it is possible for the wheel to have a free rolling in direction perpendicular to its plane. Corresponding contacting curve with respect to the wheel coordinate system, being a circle in the case shown, has a coloured highlighting. This curve has a circular shape provided the wheel plane keeps its vertical orientation. Front view of the omni wheel is shown in fragment (b).

For the case of $\psi = 0$ being shown in Figure 4 a roller outer profile, generatrix, along its axis of rotation has evidently a circular shape, see Figure 4, fragment (a), again. This shape provides smooth transfer from one roller to another while the motion occurs. Evidently if $\psi \neq 0$ then it is not the case. Thus, the contact tracking algorithm for the case of $\psi = 0$ implemented in [1] turned out to be simple enough. In the case of $\psi > 0$ it becomes visibly complicated. And its implementation on Modelica language is the main goal of this paper.

Other details of Figure 4 are the following: R is the omni wheel radius, R_1 is the distance between the wheel central point O and the roller central point, α is the half roller angular length from the viewpoint O. Unit vectors $\{\mathbf{i}, \mathbf{j}, \mathbf{k}\}$ of the base being connected with the wheel are shown in their initial positions.

In engineering applications one may encounter frequently a situation with $\psi > 0$. We proposed in [1] fast algorithm for tracking a contact provided the omni wheel keeps vertical orientation of its plane (in frame of the whole vehicle construct). Thus the task for building up the contact tracking algorithm also for the case of $\psi > 0$ is of interest. This task has been completed

in this paper. To reach this goal we accept the working model of a virtual testbench consisting of one wheel equipped by rollers along its rim. One can see easily that this simplification has mostly methodical nature and does not prevent us from integrating all the construct back into the whole vehicle having generally several omni wheels previously analyzed [1].

So let us consider an omni wheel, see Figure 4 its lateral and front views with four rollers, which is able to keep vertical orientation of its plane. We will see later how to arrange an implementation of such a servo-constraint. Note in addition, that in the case of $\psi > 0$ a generatrix of the roller outer surface will not be a segment of the circle anymore. It is represented by a more complicated curve. Moreover, point break of contact on the roller surface does not correspond to the surface tip for the case of $\psi > 0$ as it took place for the simple case of $\psi = 0$. To arrange correct simulation on event of the contact exchange between rollers one has to truncate the roller surface properly.

4.2 Model of the omni wheel dynamics

Vehicle equipped by omni wheels might be replaced by a wrench consisting of force and torque in the multibody, rigid, representation. The force supposed to act at the wheel center. Thus approximately we can analyze the omni wheel dynamics with the wrench applied instead of a remainder of the vehicle.

Moreover, the vehicle, or a separated wheel, performs in our example motion on the horizontal floor for simplicity. Thus, the wheel being embedded into the vehicle in the simplest case should be aligned vertically. To express such an alignment analytically we can connect with the wheel the base $\{\mathbf{i}, \mathbf{j}, \mathbf{k}\}$ originating from the wheel center. Both unit vectors $\mathbf{i}, \mathbf{j}$ lie in the wheel plane, and unit vector $\mathbf{k}$ is normal to it. Thus the vertical alignment of the wheel is equivalent to horizontal alignment for the vector $\mathbf{k}$. Analytical condition for this is

$$\mathbf{k} \cdot \mathbf{n}_A = 0,$$

where unit vector $\mathbf{n}_A$ is vertical, or normal to the floor. In other words, let $T \in SO(3)$ be the matrix of transformation from base $\{\mathbf{i}, \mathbf{j}, \mathbf{k}\}$ to the inertial absolute coordinate system. Then components of vector $\mathbf{k}$ are exactly the components of the matrix $T = (t_{ij})_{i,j=1}^{i,j=3}$ third column. Thus one can express condition of the wheel vertical alignment in the form

$$t_{23} = 0.$$

This latter equation shows that the omni wheel multibody system undergoes the geometrical servo constraint. It is easy to see that this constraint may be implemented via control effort, rotating torque $\mathbf{M}$ directed such as to prevent rotation of the wheel plane w. r. t. horizontal line belonging to this plane.

For details of the torque vector $\mathbf{M}$ computation note that this vector has to be directed along horizontal line passing through the wheel center and belonging to its plane. Directing unit vector $\mathbf{l}$ for this line has to satisfy the equation

$$\mathbf{l} = \mathbf{k} \times \mathbf{n}_A / |\mathbf{k} \times \mathbf{n}_A|.$$

Hence

$$\mathbf{M} = \lambda \mathbf{l}$$

and the multiplier λ is simply a value of torque balancing the wheel vertical orientation. In the wheel model torque $\mathbf{M}$ has to be added to other torques applied to the wheel under simulation. The value λ is exactly the Lagrange multiplier corresponding to the servo-constraint above.

It is easy to see the servo-constraint plays here a role of the virtual testbench for investigating the omni wheel dynamics. The remainder of the whole vehicle model is replaced simplistically by the wrench being applied to the wheel. The whole omni wheel dynamics visual model is seen in Figure 5

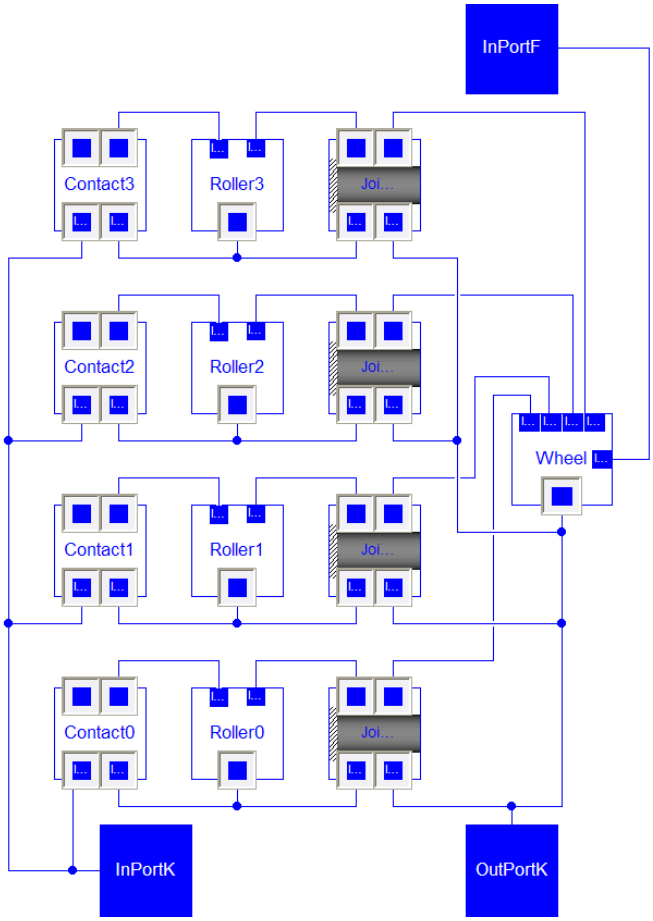

Figure 5: The omni wheel dynamics visual model.

As one can detect here the model of the omni wheel multibody system has been implemented using original multibody dynamics class library developed previously [7, 9]. One can use this library independently or with help of the knowm Modelica Standard Multibody class library or with any other Modelica library. The better way being recommended for such use is the following one. Firstly, one can implement mechanical subsystems of the whole system under implementation. For instance, mechanisms having tree structure are modeled in a better way using Modelica Standard Multibody Library while mechanical subsystems including unilateral constraints with friction are better implemented using the aforementioned library of classes. Secondly, the only issue remained is to implement proper interfaces using models of ports mapping corresponding signals being tranferred from one subsystems to another.

5 IMPROVED CONTACT TRACKING ALGORITHM

5.1 Implicit contact tracking algorithm

We will assume in the further course that the wheel plane keeps its vertical orientation permanently. We have to introduce auxiliary orthonormal bases: $b_1 = \{\mathbf{i}_1, \mathbf{j}_1, \mathbf{k}_1\}$ and $b_2 = \{\mathbf{i}_2, \mathbf{j}_2, \mathbf{k}_2\}$. Intermediate base b_2 characterises partially position and orientation of the roller, while the base b_1 relates to the omni wheel.

The base b_2 coordinate system has its origin O_B at the roller central point. The unit vector $\mathbf{i}_2$ is directed along the roller axis of rotation, see Figure 6, fragment (a). The unit vector $\mathbf{j}_2$ is directed orthogonally to $\mathbf{i}_2$ and lies simultaneously in the vertical plane. The third unit vector $\mathbf{k}_2$ of the base b_2 is defined in a natural way as

$$\mathbf{k}_2 = \mathbf{i}_2 \times \mathbf{j}_2.$$

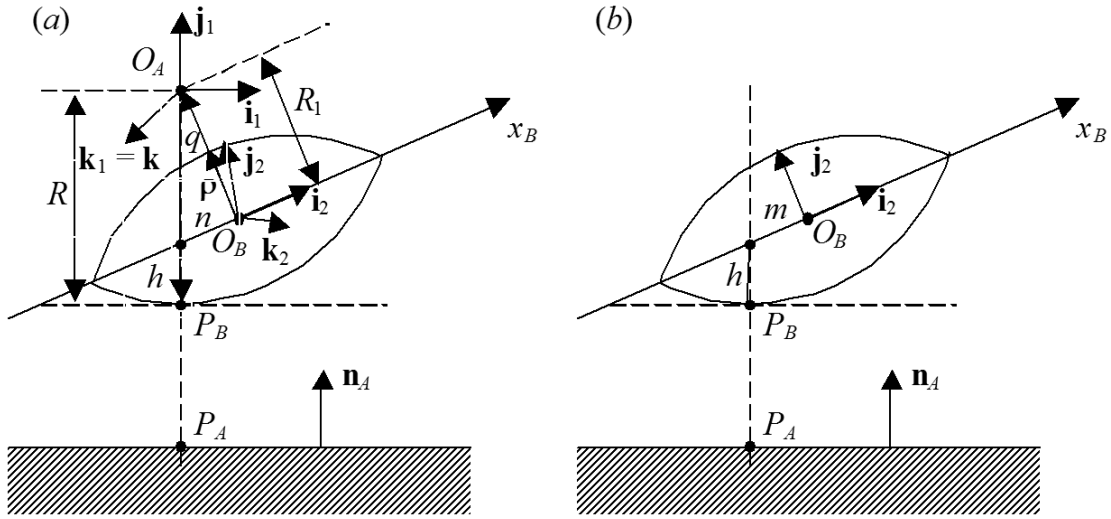

Figure 6: Contact tracking scheme: (a) lateral view of the omni wheel with a roller has been rotated about line O_AO_B by the angle ψ, (b) lateral view of the individual roller.

Remind here that all unit vectors are computed w. r. t. given fixed (absolute) coordinate system. We assume that positions and orientations are known for all bodies belonging to the multibody system for any instant $t \in [t_0, t_1]$ of simulation process. Therefore, we have

$$\mathbf{i}_2 = T_B \cdot (1,0,0)^T, \quad \boldsymbol{\rho} = (\mathbf{r}_{O_A} - \mathbf{r}_{O_B}) \,/\, |\mathbf{r}_{O_A} - \mathbf{r}_{O_B}|\,,$$

where T_B is the roller current orientation matrix.

Origin of the base b_1 coordinate system is located at the point O_A ($= O$ in Figure 4) of the wheel center. The unit vector $\mathbf{i}_1$ is oriented horizontally and belongs to the wheel plane. The unit vector $\mathbf{k}_1$ is orthogonal to the wheel plane and is identical to one of the wheel connected base vectors. We assume that using a controller the vector $\mathbf{k}_1$ permanently maintains its horizontal state. Supposing vector $\mathbf{k}_1(t)$ known we also have $\mathbf{j}_1(t) = (0,1,0)^T$ and $\mathbf{i}_1(t) = \mathbf{j}_1(t) \times \mathbf{k}_1(t)$.

Consider now relations providing base b_2 construction. Unit vector $\mathbf{i}_2$ has been built above. During roller and the floor contact the vector $\mathbf{i}_2(t)$ can not become vertical. Moreover, if the

roller distortion takes place, its angle of rotation $\psi > 0$ about axis $O_A O_B$ is fixed non-zero, then the condition $\mathbf{i}_2 \neq (0,1,0)^T$ is fulfilled permanently. So we can assume that the condition

$$\mathbf{c} = \mathbf{i}_2 \times (0,1,0)^T \neq \mathbf{0}$$

is also fulfilled.

Thus, we can define $\mathbf{k}_2 = \mathbf{c}/|\mathbf{c}|$. And after this we can set $\mathbf{j}_2 = \mathbf{k}_2 \times \mathbf{i}_2$. Geometrical constraints, conditions of orthogonality to be exact, play important role in the omni wheel kinematics

$$\boldsymbol{\rho} \cdot \mathbf{i}_2 = 0, \quad \boldsymbol{\rho} \cdot \mathbf{k}_1 = 0.$$

These equations actually apply to computing the unit vector $\boldsymbol{\rho}$ and we have their differential versions

$$\frac{d}{dt}\boldsymbol{\rho} \cdot \mathbf{i}_2 + \boldsymbol{\rho} \cdot \frac{d}{dt}\mathbf{i}_2 = 0, \quad \frac{d}{dt}\boldsymbol{\rho} \cdot \mathbf{k}_1 + \boldsymbol{\rho} \cdot \frac{d}{dt}\mathbf{k}_1 = 0.$$

The value $c_\beta = \cos\beta = \mathbf{i}_2 \cdot (0,1,0)^T$ of cosine for the angle β of the roller axis inclination to vertical $(0,1,0)^T$ plays also an important role in the contact tracking algorithm. If current value of the variable c_β is less than some limiting parameter $c_{\beta\max}$, and simultaneously if an altitude of the point O_B defining position of the roller center is less than value R of the wheel radius then the contact takes place. Otherwise no contact occurs.

Note here that in order to arrange the unilateral constraint in the multibody system dynamics model the developer usually has to implement something like hybrid automata construct. In our omni wheel model, on the contrary, this is not the case. It turned out sufficient to implement "simple" "`if`" construct to switch states "contact" and "no contact" for each individual roller, and simultaneously to advance forward "contact" state from one roller to its neighbour. The whole picture looks like from time to time neighbouring rollers mutually exchange their states. One can find details of the unilateral constraint implementation in [1]. Merely note that "`if`"-alternatives are the following: (a) "contact" state corresponds to zero-valued relative acceleration of two contacting surfaces at the point of contact, (b) "no contact" branch corresponds to the zero-valued reaction mutual for both bodies at contact. All this is according to the Signorini rule. "`if`"-condition depends on the roller orientation variables.

Essential role in all these computations plays a contact tracking algorithm. Generally, its implementation reduces to computation of the contact point/patch which enables computing forces at contact. Usually, one considers contact of two surfaces participating in rigid/elastic interaction of two massive bodies. As a rule, such algotithms are pretty expensive and noticeably slow the whole simulation process. Fortunately, in case of omni wheels we found here the simplest way to make this computation as fast as possible using "elementary" geometric considerations.

We can also easily see from the Figure 6 that the point P_B of contact between roller and floor is obtained using formula

$$\mathbf{r}_{P_B} = \mathbf{r}_{O_B} + R_1\boldsymbol{\rho} - R\mathbf{j}_1 + \mu\mathbf{k}_1,$$

where the scalar μ is to be computed. Here the value R_1 is the distance between points O_A and O_B. The scalar μ can be computed if we multiply the last equation by $\mathbf{k}_2$ using dot-product. Thus we have

$$\mu = \left[R\mathbf{j}_1 \cdot \mathbf{k}_2 - R_1\boldsymbol{\rho} \cdot \mathbf{k}_2\right] / \mathbf{k}_1 \cdot \mathbf{k}_2$$

since the vector $\mathbf{r}_{P_B} - \mathbf{r}_{O_B}$ lies in vertical section of axis-symmetrical surface of the roller, and the vector $\mathbf{k}_2$ by construction is orthogonal to this section. As a result the position $\mathbf{r}_{P_B}$ of the contact point P_B is uniquely computed.

5.2 Explicit contact tracking algorithm

Yet another way to obtain current position $\mathbf{r}_{P_B}$ of the contact point P_B, or more accurately: the roller point closest to the floor, is an application of the following chain of equations. This chain is simply understood from geometrical scheme shown in Figure 6, (a) and (b).

$$\mathbf{r}_{P_B} = \mathbf{r}_{O_B} - m\mathbf{i}_2 - h\mathbf{j}_1,$$

where $m = R_1 \sin q / \cos q / \cos\psi$, $h = R - R_1 / \cos q$, q is the current value for angle of deviation of the vector $\boldsymbol{\rho}$ from direction of the vector $\mathbf{j}_1$. So we have

$$\cos q = \boldsymbol{\rho} \cdot \mathbf{n}_A, \quad \sin q = (\mathbf{n}_A \times \boldsymbol{\rho}) \cdot \mathbf{k}_1.$$

Here we give explanations of some details of Figure 6. Fragment (a) corresponds to the lateral projection of the wheel and likewise the distorted roller projection. This latter object is shown here in a general position. Furthermore, P_B is the current contact point between the roller and the horizontal floor, n is a projection of the roller axial line segment onto the wheel plane. We can see easily that this projection is computed by the formula $n = m\cos\psi$ because the roller axis is turned about $O_A O_B$ by the angle ψ, see fragment (b) for the roller axial vertical lateral section. Thus, we have to pass two straight line segments from the roller center O_B to reach the point P_B: (a) the segment of the roller axis of length m; (b) the segment down the vertical of length h. As we already mentioned above all variables needed are computed through known variables using explicit formulae.

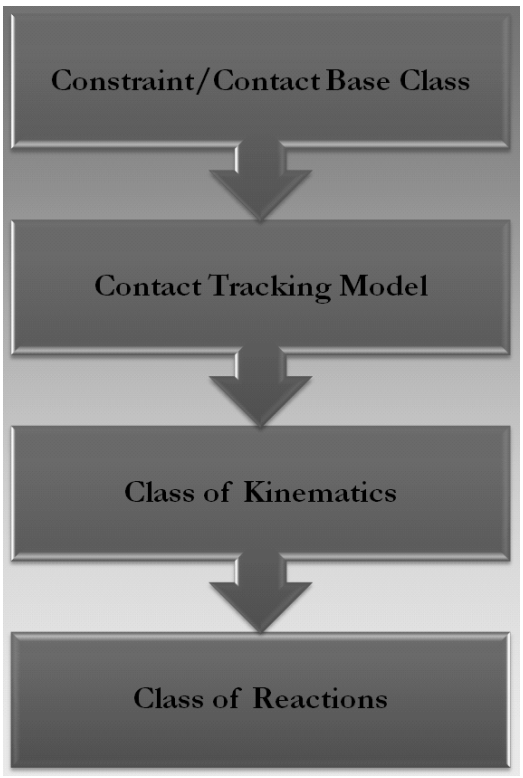

Figure 7: Contact model by stages of inheritance.

In case of $\psi > 0$, distortion exists, for both implicit and explicit algorithms not all the length of the roller surface generatrix is necessarily in contact. So really we have to cut tips of rollers

to provide regular simulation process. Length of the tip to be cut we can obtain for instance empirically or compute it explicitly. Indeed, one can easily see from Figure 6 that the real roller length should be computed by the formula

$$L = 2R \sin\alpha / \cos\psi.$$

"Ideal" switching of contact takes place in this case: exactly at the instant of contact loss for current roller a contact immediately arises for the "next" roller in direction of the wheel rolling.

5.3 Computer implementation

Model of the omni wheel testbench virtual prototype is a container class including the following objects instantiated: (a) disk of the wheel; (b) objects of rollers mounted along the wheel rim; (c) objects of joints connecting rollers and the wheel disk; (d) objects of contacts connecting objects of rollers and the object of the horizontal floor surface; (e) model of base body as a horizontal floor.

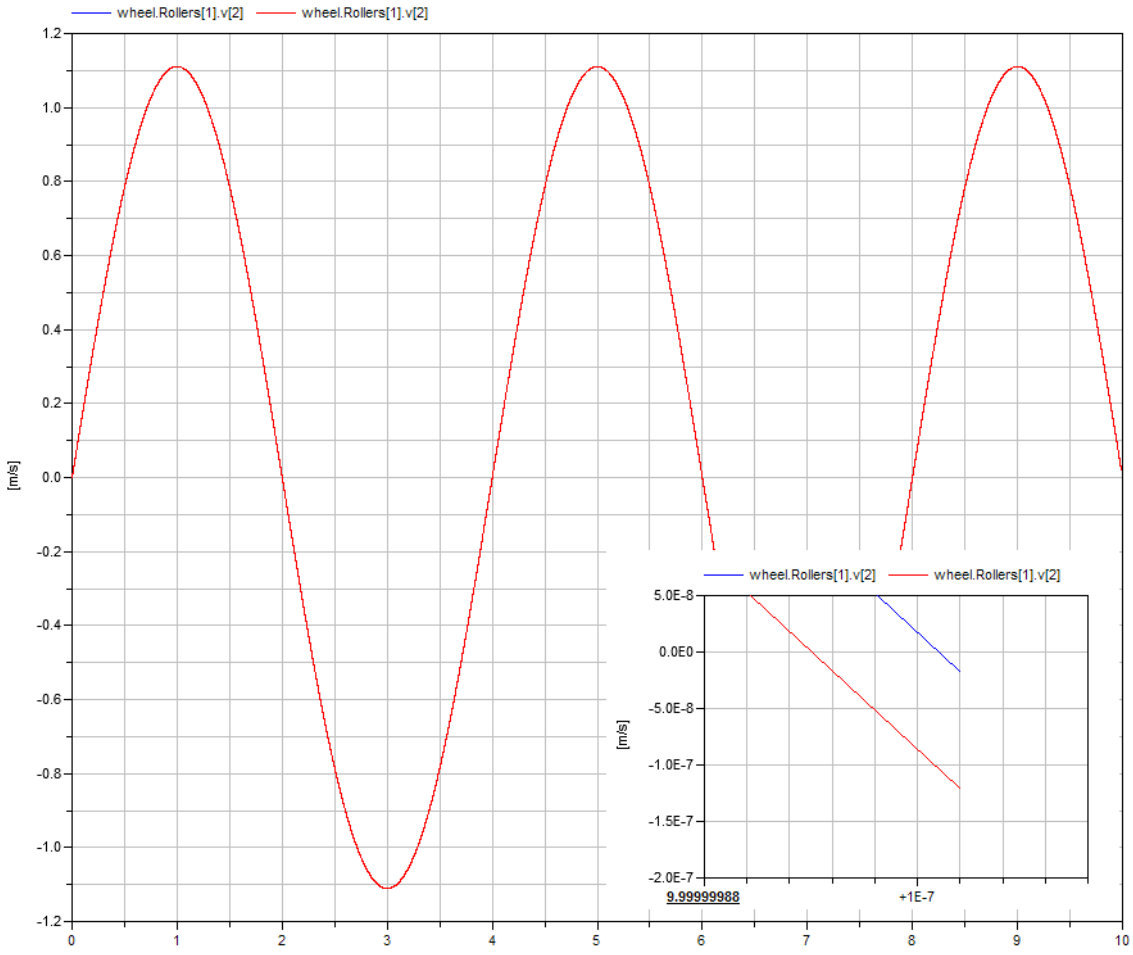

Figure 8: Comparison of dynamics for the roller No. 1 central point, its velocity y-coordinate, for cases of: explicit (blue curve) and implicit (red curve) algorithm of contact tracking.

Let us analyse in more details a structure of contact model. This model has many similarities with contact models previously considered [7]. Nevertheless important differencies exist. One of them mentioned above with regard to organization of the contact class using simple and

efficient construct [1]. Note that in case of $\psi > 0$ the point of contact creates a curve with discontinuities at instances of rollers changes. However, this circumstance does not prevent the process of regular simulation.

Finally, we apply rigid point contact model as part of the simplest omni wheel model. For this we use the base class for constraint/contact models having only equations of Newton's third law as a behavioral section [9].

We use class of the contact tracking model on the second stage of inheritance, see Figure 7. Cases of this class organization have been analysed above. Coordinates of nearest points P_A and P_B at contact for each pair (floor, roller) are computed as a result for this class functionality.

Class for computing all kinematical characteristics at contact needed "works" in case of contact existence on the next stage of inheritance. On the third stage class for computing the reactions at contact is "turned on". Reactions are the following: (a) normal reaction; (b) tangent force of friction; (c) torque of reactions (zero in the current consideration though it is not difficult to compute torque for several contact models).

To verify an approach for building up the models under analysis we compare the omni wheel dynamics in cases of implicit and explicit algorithms. The wheel performs free motion (combining rotation and sliding) with the only restriction: keep vertical alignment of the wheel disk.

Roller No. 1 central point, its mass center, altitude was analyzed and verified. More accurately we examine y-coordinate of the point velocity. Both models turned out almost identical: in the worst case we have a divergence: in accelerations of order 10^{-8}, in velocities of order 10^{-7}, in position of order 10^{-6} over the time span being equal to 10 units of time. Results of simulation for velocities are shown in Figure 8. Other divergencies for the roller No. 1 central point acceleration and position at time = 10units are shown in Figure 9 and 10 respectively. As expected the model with explicit contact tracking algorithm is faster approximately in 1.5 times.

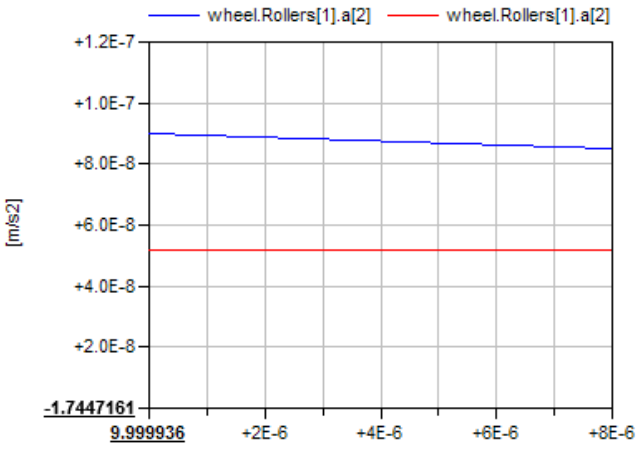

Figure 9: Divergence for y-components of acceleration.

6 CONCLUSIONS

- Two contact tracking algorithms were proposed: implicit and explicit. As expected the second algorithm turned out to be faster almost in 1.5 times. Both algorithms are simple (and efficient) even in simpler case of rollers without any distortion.

- In case of distorted rollers contact curve becomes discontinuous at instants of rollers

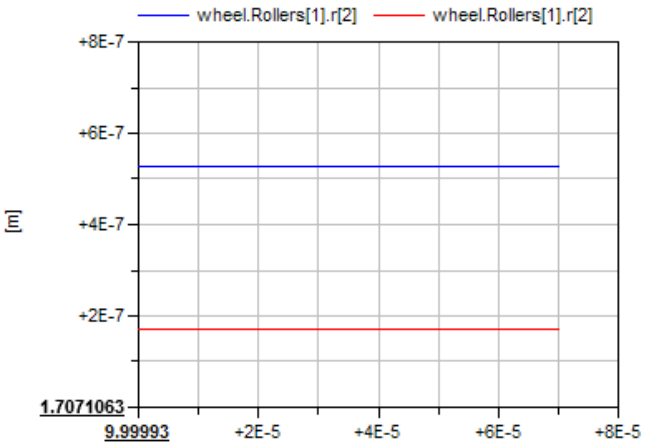

Figure 10: Divergence for y-components of position.

change. But simulation process maintains its regularity.

- Both algorithms generate identical dynamics.
- Process of the contact model design using technology of "vertical separation" outlined above has an evident motivation and allows a simple generalization both for the normal force computation and for the tangent friction force model.

7 ACKNOWLEDGMENT

The investigation was performed in MAI under financial support provided by RSF, project 14-21-00068.

REFERENCES

[1] I. Kosenko, K. Gerasimov, Object-oriented implementation of a unilateral point-contact constraint model with friction in frame of the omni vehicle multibody system. *Proceedings of the jointly organized 11th World Congress on Computational Mechanics – WCCM XI, 5th European Congress on Computational Mechanics – ECCM V, 6th European Congress on Computational Fluid Dynamics – ECFD VI*, Barcelona, Spain, July 20–25, 2014, pp. 4452–4463.

[2] V. Kálmán, Controlled braking for omnidirectional wheels. *International Journal of Control Science and Engineering*, **3**, 48–57, 2013.

[3] J. Tobolár, F. Herrmann, and T. Bünte. Object-oriented modelling and control of vehicles with omni-directional wheels. *Computational Mechanics 2009*, November 2009.

[4] B. E. Ilon. Wheels for a course stable selfpropelling vehicle movable in any desired direction on the ground or some other base. *Technical report, US Patents and Trademarks office*, Patent 3,876,255, 1975.

[5] I. I. Kosenko, Physically oriented approach to construct multibody system dynamics models using Modelica language. *Proceedings of Multibody 2007, Multibody Dynamics 2007.*

An ECCOMAS Thematic Conference, Politecnico di Milano, Milano, Italy, June 25–28, 2007, 20 pp.

[6] I. I. Kosenko, Integration of the equations of a rotational motion of a rigid body in quaternion algebra. The Euler case. *Journal of Applied Mathematics and Mechanics*, **62**:193–200, 1998.

[7] I. I. Kosenko, Implementation of unilateral multibody dynamics on Modelica. *Proceedings of the 4th International Modelica Conference*, Hamburg University of Technology, Hamburg–Harburg, Germany, March 7–8, 2005, pp. 13–23.

[8] I. Kosenko, Implementation of unilateral constraint model for multibody systems dynamics on Modelica language. *Proceedings of ACMD2006, The Third Asian Conference on Multibody Dynamics 2006*, Institute of Industrial Science, The University of Tokyo, Tokyo, Japan, August 1–4, 2006.

[9] I. I. Kosenko, M. S. Loginova, Ya. P. Obraztsov, M. S. Stavrovskaya, Multibody systems dynamics: Modelica implementation and bond graph representation. *Proceedings of the 5th International Modelica Conference*, arsenal research, Vienna, Austria, September 4–5, 2006, pp. 213–223.

[10] D. P. Chevallier, On acceleration analysis in multibody dynamics. C. Soize and G. Schueller eds. *Structural Dynamics. EURODYN 2005. Vol. 1.*, Millpress, Rotterdam, The Netherlands, 2005, pp. 587–592.

[11] F. Pfeiffer. Unilateral multibody dynamics. *Meccanica*, **34**:437–451, 1999.