International Journal of Applied Engineering Research ISSN 0973-4562 Volume 12, Number 20 (2017) pp. 9592-9604

Using Multi-Core Architectures to Solve the Wheel-Rail Contact Problem

Nicola Bosso[1] and Nicolò Zampieri[2*]

[1]Department of Mechanical and Aerospace Engineering, Politecnico di Torino, C.so Duca degli Abruzzi 24, 10129 Torino, Italy.

Orcid: 0000-0002-5433-6365

[2]Department of Mechanical and Aerospace Engineering, Politecnico di Torino, C.so Duca degli Abruzzi 24, 10129 Torino, Italy.

Orcid: 0000-0002-9197-1966

[*](Corresponding author)

Abstract

The forces exchanged between wheel and the rail are fundamental in order to study the railway vehicle dynamic. These forces arise from the non linear combination of profile contact and friction and their calculation can be performed using iterative methods which require important calculation time. Simulation of railway vehicle dynamic takes therefore large calculation time or simplified contact models are needed. This becomes critical especially in case of real time simulation. Recent development of CPU based on multi-core processor makes possible to distribute the calculation of contact forces and the vehicle dynamic problem over different cores, reducing the calculation time. This work describes the strategies adopted to perform parallel distributed calculation starting from the RTCONTACT code, developed at Politecnico di Torino during the last years. According to the proposed approach, the contact problem is solved using one core for each wheel, while one or more cores can be used to solve the vehicle dynamics. An example of application of the contact code, distributed on different cores, is shown during evaluation in a real-time environment of the wheel and roller profile evolution on a scaled roller-rig.

Keywords: Wheel-rail contact; multi-core architectures; vehicle dynamics; real-time; roller-rig.

INTRODUCTION

The use of numerical models for the design of railway vehicles is nowadays very common since they allow to optimize the dynamic performance of the vehicle by comparing different constructive solutions. Several design architectures can in fact be simulated in few time and without the necessity of prototypes. This strategy has allowed in the last years to design railway vehicle with always higher performance in terms of stability, safety and ride comfort. The reliability of the process also depends on the use of advanced and accurate contact algorithms, and therefore commercial codes are continuously improving their models. Complex models are however slower than simplified approaches and the cannot be executed in Real Time. The recently adoption of railway hardware in the loop (HIL) simulators requires instead numerical models able to run in real-time (RT). HIL simulators allow to test different vehicle components as if they were installed on the vehicle running on track, that is replaced by an RT numerical model. The HIL simulators reduce the need of experimental tests on track that are expensive and time consuming. The HIL simulator is usually composed by a RT controller executing the dynamic simulation of the vehicle model, actuators, sensors and relative data acquisition connected to the component to be tested. These systems are usually used with the purpose of testing vehicle control systems and electronic control units [1, 2]. A reliable HIL railway simulator requires an accurate and fast vehicle an contact numerical models, but the accuracy of the models is often in contrast with their numerical efficiency. The use of real-time numerical models is not very common in railway applications and few solutions can be found in literature. Those solution, when run on a single core, usually endure the simulation accuracy to achieve numerical efficiency. The approach used by Meli and others [2, 3], is to developed the numerical model in the Simulink environment and compiling it to a DSpace system by using the Real-Time Workshop Toolbox included in Simulink. In this case the mathematical equations, describing the vehicle behavior, are implemented directly in Simulink and this makes complex to modify the numerical model due to design changes or to adapt the same model to other vehicles, because it requires re-compiling.

Another solution is proposed in [4, 5] by the authors, and in this case the numerical model has been developed with a modular approach. Each module has been included in a C language S-function and compiled separately for the DSpace controller. This solution allows simpler adaptation of the same model to different vehicles, because only the modified modules need to be recompiled.

The multibody (MB) commercial codes, such as Simpack and Adams VI-grade, include a proprietary real-time compiler that allows to convert the numerical model for real-time applications after a system reduction [6, 7]. This procedure is necessary to convert the differential algebraic equations (DAE), used to describe the MB model, to ordinary differential equations (ODE) that is the formalism required by fixed time step integrators. The system reduction procedure in order to reduce the CPU loads neglects the higher frequency modes due to low mass bodies and concentrates on the quasi static ones. This approach is currently limited to the automotive modules because wheel–rail contact models are not supported by the current real-time compilers [8, 9]. The calculation of the contact forces is in fact the computational activity that more loads the CPU, requiring more than the 70% of the computational power, while only the 30% is used to solve the vehicle dynamic [10]. The high calculation times are due to the complexity of the contact model which has to evaluate on each wheel the contact point position, the contact area dimension and the friction forces by integrating the tangential stresses. In the last years always more efficient algorithms have been proposed to calculate the friction forces [11, 12], but the process used to evaluate the contact point position still remains the bottleneck of the whole contact code [13, 14].

The CPUs with multi-core technology, developed in the last years, can be used to increase the speed of the contact model taking advantage from the parallel computing technique. Mei and Zhou [10, 15, 16] show the use of FPGA controllers to evaluate the wheel-rail contact forces. In particular it is described the parallelization techniques used to distribute the FASTSIM algorithm exploiting the parallel-processing capability of FPGAs. The work is limited to the parallelization of the algorithm used to evaluate the contact forces while the algorithms for the calculation of the contact point position and the contact area dimension are not taken into account. In case of use of FPGAs for railway dynamic simulation, the actual limitation is the requirement of a specific hardware and the relevant activity required to write and compile the code, which leads to limited flexibility when the model need to be modified. For this reason the an application including an entire vehicle model has not been yet developed.

This paper shows the use of multi-core processors to increase the computational efficiency of the wheel-rail contact code (RTCONTACT) developed by the railway research group at Politecnico di Torino [13, 14]. In particular the work describes the architecture of a single coach model for RT applications that has been distributed on eight cores. The validation of the RTCONTACT code with respect to UIC standards and Simpack MB code is then described. The final part of the work shows the use of the contact code to evaluate in real-time the wheel and roller wear on a scaled roller-rig. In this case a single suspended wheelset is considered and the contact code is executed on a multi-core real-time controller that includes the data acquisition system in order to provide to the contact code the signals generated by the sensors installed on the roller-rig.

PARALLELIZATION OF A TYPICAL RAILWAY VEHICLE MODEL

In order to improve the numerical efficiency of the RTCONTACT code [13] a procedure has been adopted to distribute the problem on multi-core architectures in order to solve each sub task in parallel. The contact code has been modified and integrated into a LabView™ model, which allows parallelizing the calculation in a RT multi-core environment. The code parallelization allows to significantly reduce the calculation time in order to make the contact code suitable for real-time simulations.

The contact problem and the vehicle dynamic have been split in order to improve the contact algorithm efficiency. In particular one core is used to solve the vehicle dynamic, one core is used to define the track geometry, while the contact forces are evaluated using several dedicated cores. The cores used to solve the dynamics and track geometry are also used to calculate the contact forces when idle, in fact contact force calculation is solved in sequence with vehicle dynamic and track management, as shown on Figure 1 for a vehicle with four axles.

The track management module, shown in Figure 1, calculates the section of rail that has to be in contact with each wheel (purple arrow in Figure 1). This section depends on the track geometry and on the position of each wheelset with respect to the track. For this reason the core used to manage the track requires, as input, the state vector of the coach to calculate the rail sections to be considered at each time step. This activity cannot be parallelized since it has to be performed after that the coach dynamic has been solved, and the vehicle position is known, but before the calculation of the contact forces since those requires the track section as an input. However this activity is very fast and also if it is performed in sequence it does not affect very much the calculation time.

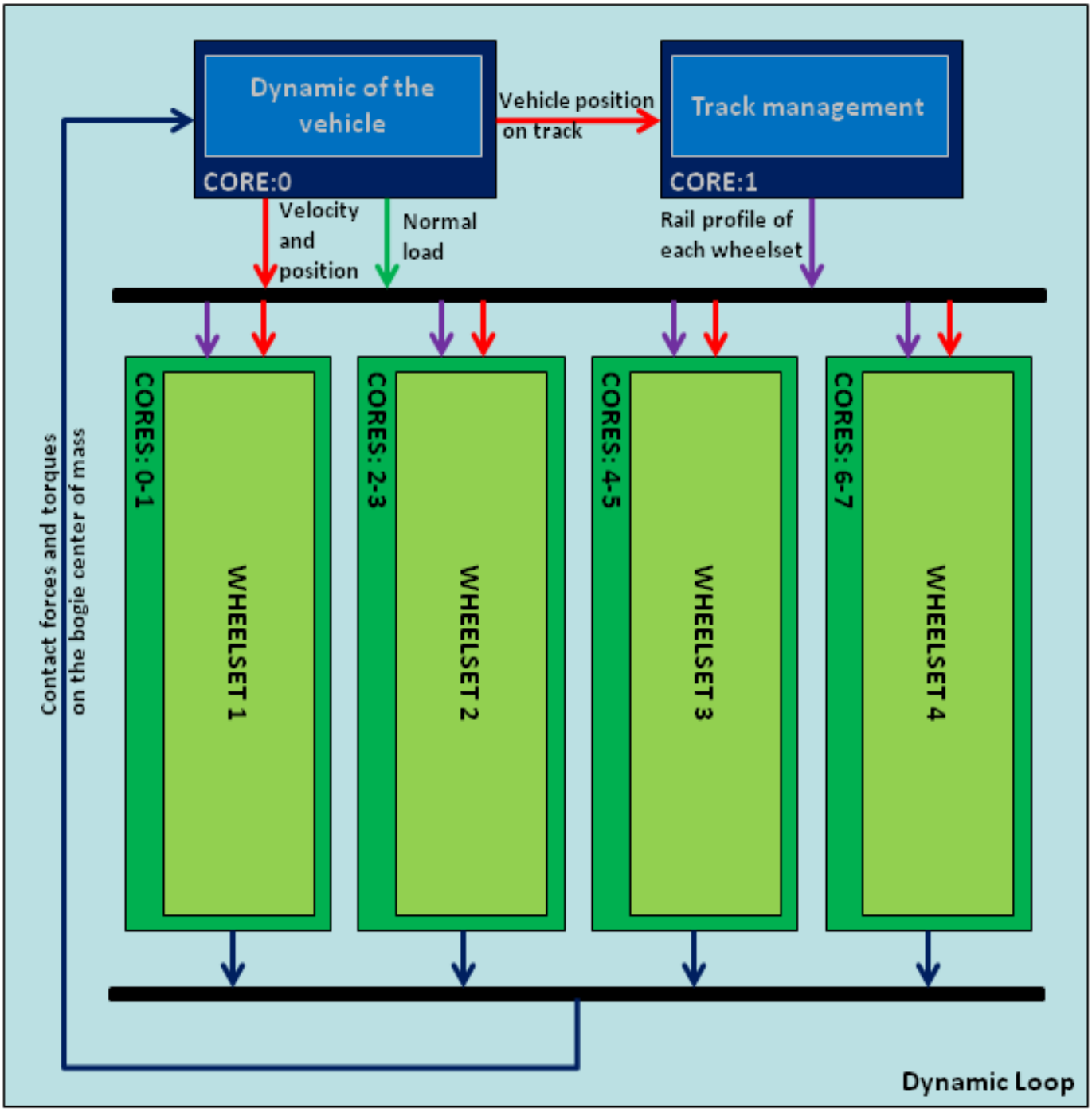

Figure 1: Model of the bogie distributed on the eight cores architecture. The model takes into account a single vehicle composed by four wheelsets.

The contact algorithm is very complex and it involves high computational load due to the strong non linearity of the system. This is the reason why two cores are used for each wheelset, calculating at the same time on different cores the contact forces on the two wheels.

The contact problem on the two wheels cannot be considered independently since the two wheels are rigidly joined to the axis so that the wheelset has to be considered as a single rigid body. For this reason only some of the modules of the contact model can be parallelized and the code has been written in order to maximize the portion of the code which can be executed in parallel, as described in section 3. In this work the dynamic model of the coach has been simplified since the focus of the work is to show the RTCONTACT contact code parallelization and the benefits in terms of computational performance. The typical vehicle model is shown in Figure 2 where are also indicated the cores used to solve the different sub-models.

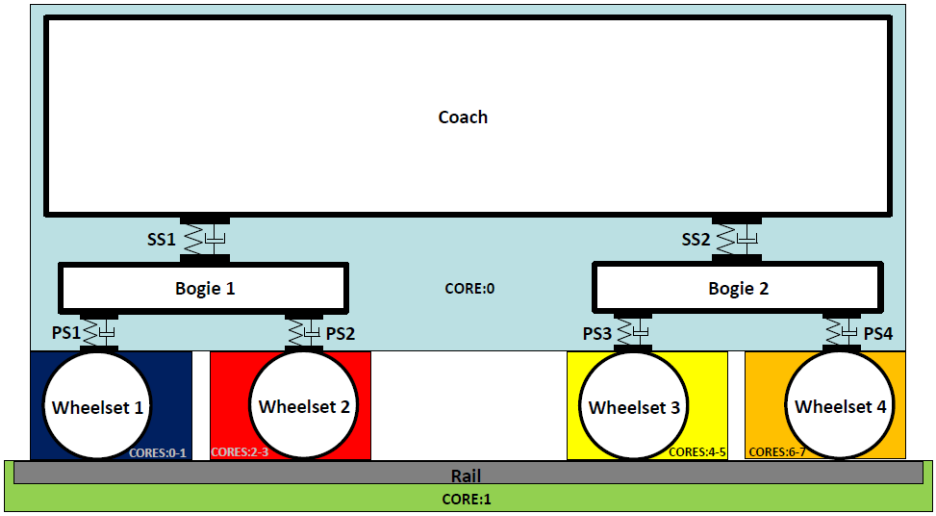

Figure 2: Model of the vehicle distributed on the eight cores.

PARALLELIZED MODEL OF THE CONTACT CODE

The most time consuming activity for the CPU is the execution of the contact algorithm which more affects the computational performance of the whole code. The example of a single coach previously shown in Figures 1 and 2 uses a total of 8 cores in order to solve the contact problem and in particular two cores for each wheelset. The strategy for the parallelization of the RTCONTACT algorithm (used to evaluate the contact forces on the two wheels of the same wheelset) is not trivivial since it is necessary to recognize the parts of the code that can be executed in parallel from the ones that has to be executed in sequence. The contact algorithm will therefore analysed in detail in this section. In Figure 3 it is shown the parallelized contact module for a single wheelset, which has been developed in LabView™ environment.

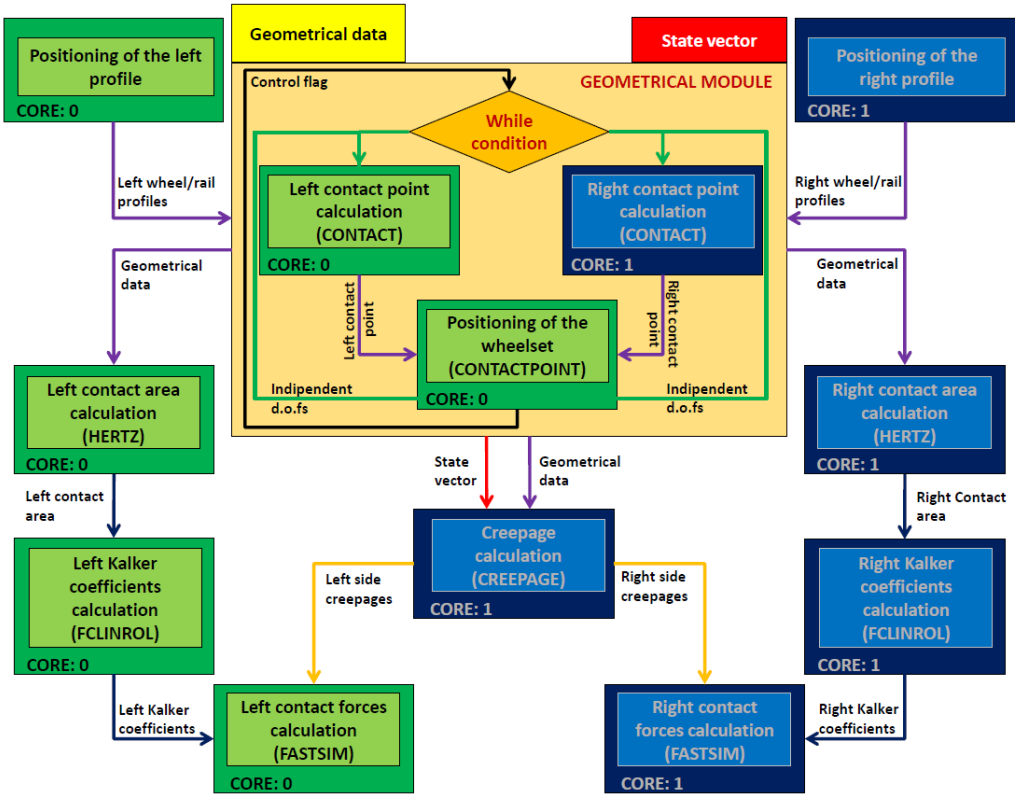

Figure 3: Contact model of the wheelset distributed on two cores.

Each module has been written in C++ language and compiled into a Dynamic Link-Library (.dll) in order to be able to include the C++ code into LabView™. This commercial software has been chosen since it allows to make data input/output using I/O boards and to validate in real-time the whole code on the roller-rig, which have been developed by the authors [17, 18, 19]. Being the routines of the contact code written in C++ these can be easily exported in other environments and be integrated in commercial software. A possible application could be the numerical simulation of long train dynamic which is very time consuming due to the huge number of vontact elements [20]. From Figure 3 it is possible to notice that also the geometrical module has been parallelized; the Contact algorithm is called two times for each time step of the while loop since it has to determine the contact point position on each wheel, while the Contactpoint algorithm is called only once because it has to calculate the dependent d.o.fs of the wheelset (vertical displacement z and roll angle □) which are a function of of the contact points position. It is evident that the Contactpoint algorithm works in sequence to the Contact algorithm and it represents the bottle neck of the RTCONTACT algorithm; for this reason the Contactpoint algorithm has been optimize so that to improve its computational efficiency. The algorithm does not have iterative procedures and so the execution time of this algorithm is more than thousand times less than the time required by the Contact algorithm, which has to interpolate the profile and to find the contact points position. The while loop of the geometrical module stops when the wheelset is in the correct position, i.e. the penetration between the wheel and the rail profiles on both sides is less than the maximum allowed ε, and the control flag changes its state in the Contactpoint algorithm. After that the geometrical quantities in the contact patches have been determined, the Hertz algorithm calculates the contact areas in parallel for the two wheels. Also the Fclinrol algorithm, which calculates the Kalker coefficients, is executed at the same time for the two wheels. This module calculates the coefficients interpolating the values from a look-up table. The module that calculates the creepage is not parallelized because it has only to apply simple and direct formulas and there are no advantages from its parallelization. The last part of the RTCONTACT algorithm calculates the forces using the FASTISIM method and due to the iterative procedures used inside this algorithm it is convenient to calculate the contact forces for the two wheels at the same time on different cores. The contact forces calculated are used to solve the dynamic simulation of the vehicle.

THE CONTACT MODEL

The contact model can be subdivided into four main modules (as shown in Figure 4): the geometrical module that calculates the contact point position, the Hertz module that calculates the shape and the dimension of the contact area, the kinematical module that calculates the creepages in the contact area and the tangential force module that obtains the contact forces by integrating the tangential pressure on the contact area.

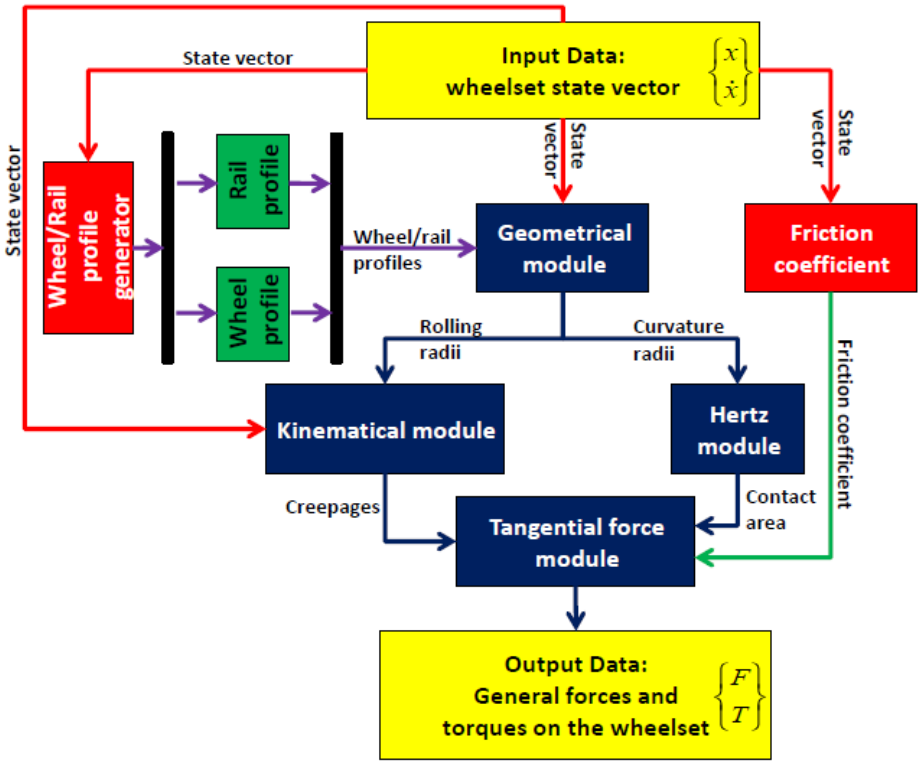

Figure 4: Contact model for a single wheelset.

The geometrical module is the one that more influences the computational efficiency of the code. In literature are present different contact models which are usually classified according to the contact hypothesis adopted: rigid contact, elastic contact and pseudo-elastic contact. The first model, which has been adopted in this work, calculates the contact point minimizing the function of the distance between the wheel and the rail profile. The second one instead evaluates the normal force exchanged between the wheel and rail proportional to the interpenetration, with a proportionality constant equal to the Hertzian stiffness. The third model is instead used to improve the numerical efficiency of the code and the method is based on the minimization of a regularized function that expresses the distance between the profiles.

The scheme shown in Figure 4 shows two additional modules that have not yet been mentioned: the wheel/rail generator and the friction coefficient estimator. The first module is used to update the profiles of the wheel and the rail during the simulation, if the wear phenomenon is considered. As regards the rail in addition to update the profile due to wear it is necessary to calculate the profile corresponding to the section of the track considered. The profile generator module, using the state vector, is able to calculate the wheelset position along the track so that to determine the section of rail in contact with the wheels. Obviously the user has to provide as input the wheel and rail nominal profiles and the track layout. The friction coefficient block deals to calculate the friction coefficient as a function of the slip velocity, that is evaluated from the state vector data, interpolating its value from a look-up table. This table has been experimentally calculated by the authors using the roller-rig device [17,18,19].

Geometrical Module

The geometrical module is designed to determine the position of the contact points and to calculate the most important geometrical quantities in the contact patches. The results of this algorithm are fundamental since they are used as input by the other modules. The geometrical algorithm is the part of the contact code that requires the highest calculation time (about 80% of the calculation time to execute the whole algorithm). This module is based on the rigid contact theory and it considers two-dimensional profiles i.e. the cross sections of the wheel and the rail profiles are used to find the contact point position. The geometrical module needs as input the lateral position (y) and the yaw angle ( ) of the wheelset (these d.o.fs are the independent d.o.f. of the wheelset) and it gets the wheel and rail profiles from the profile generator. The module is composed by two algorithms: the first one (Contact algorithm), deals with determining the position of the contact points; the second one (Contactpoint algorithm) roto-translates the wheelset, so that wheel and rail profiles, on both sides, penetrate less than the maximum allowed penetration. At first, the geometrical module, translates the wheel profiles in lateral direction of a quantity equal to the lateral displacement of the wheelset (y). After this translation the module roto-translates the wheelset in vertical direction and around the roll axis in order to minimize the wheel and rail penetration on both sides; this is an iterative process governed by the Contactpoint algorithm that contains an analytical relation to determine the roto-translation to be adopted for the next step in order to reduce the number of iterations. Each time the wheel profiles are roto-translated, a clipping operation is performed in order to consider only the portion of wheel which is contained in the projection of the rail profile on the transversal plane. After that, wheel and rail profiles are interpolated by using a common abscissa y, in order to be able to easily calculate the vertical distance Δz between the two profiles. The process used to evaluate the contact point position is based on the discrete minimization of the interference between wheel and rail profiles and this method is developed into the Contact algorithm which detects the two points of the profiles (left and right) where the interferences have its maximum or the distance have its minimum (when the relative profile location lead to a no contact). In Equation 1 is summarized the algorithm used in Contact and d is the distance between the wheel and the rail profile, q is the wheelset state vector, y_C is the lateral coordinate of the contact point and is the maximum allowed penetration.

$$\min_{y_C} d(\boldsymbol{q}, y_C) \leq \varepsilon \quad (1)$$

The distance function d is negative if there is penetration and positive otherwise, while is a negative value which has been chosen equal to 10E-12 m in order to be able to assume the contact as rigid. The Contactpoint algorithm, if the interference is greater than the allowed value or in case of no contact, applies a roto-translation to the wheelset in order to reach the desired value of the interferences. In case of linear or regular profiles the target position can be found in a single iteration, in case of non linear profiles, the rotation can cause the selection of a different pair of points as contact points and the process requires more iterations to converge (usually less than 4-5 iterations). After that the wheelset is in the correct position, the code calculates the geometrical quantities, which are necessary for the other modules, as shown in [13].

Hertz Module

This module is used to calculate the dimension of the contact area and the algorithm his based on the Hertz's theory, therefore the shape of the contact area is elliptical. In order to assume the Hertz theory when friction is present between the surfaces the quasi-identity hypothesis has been introduced; this hypothesis states that the tangential pressures due to friction do not influence the shape of the contact area. Another hypothesis introduced is that the contact between the surfaces is non-conformal and this hypothesis is at fault when

localized wear occurs. In this condition would be better to use a non-Hertzian approach, which is able to consider non elliptical contact area; this theory has not been developed in RTCONTACT since the code has to be suitable for real-time simulations, which requires very low calculation time. A non–Hertzian model in fact would introduce an iterative method for the calculation of the contact area shape based on the Pangiotopulos' theory [21], which uses an iterative process for the calculation of the distribution of both the normal pressure and the tangential pressure. The Hertz theory allows the calculation of the contact area without iterative processes, using as input the normal load on the wheel and the local curvatures of the wheel and the rail in the contact zone. The curvatures are calculated in the geometrical module according to Frenet's formula as shown in Equation 2, where z is the vertical coordinate of the wheel or rail profile.

$$\rho = \frac{\ddot{z}}{(1-\dot{z}^2)^{\frac{3}{2}}} \tag{2}$$

The outputs of the Hertz module are the ellipse semi-axes which are determined using look up tables that contain the solution of the elliptical integrals as a function of a parameter that can be calculated from the curvature radii.

Kinematical Module

In this module are calculated the kinematical creepages on the basis of a simplified approach based on a change of the "first order theory" of A. De Pater. The creepages are a function of the relative velocity between the two bodies in contact along three different directions: longitudinal, lateral and normal to the contact surface (spin creepage). Therefore there are three different creepages defined as the ratio between the relative velocity and a reference velocity; the last one is the same for the three creepages and it is the longitudinal velocity of the vehicle. The spin creepage is the only one that is not dimensionless and its unit is [1/m], since it is defined as the ratio between the relative wheel angular velocity, measured around the axis normal to the contact area, and passing through the contact point and the reference velocity. The general definition of the three creepages is shown in Equations 3.

$$\begin{aligned} \xi &= \frac{\Delta V_x}{V_0} \\ \eta &= \frac{\Delta V_y}{V_0} \\ \phi &= \frac{\Delta \omega_N}{V_0} \end{aligned} \tag{3}$$

Tangential Force Module

In this module is performed the calculation of the contact forces using the quantities that have been calculated by the previous modules. Two tangential forces and a torque occur due to the friction phenomena on the contact area. These forces are a non linear function of the contact ellipses semi-axes ratio $(\frac{a}{b})$, of the creepages and of the Kalker's coefficients, as shown in Equation 4.

$$\left(F_{X,}F_Y, M_Z\right) = f\left(\frac{a}{b}, C_X, C_Y, C_{XY}, C_Z, \xi, \eta, \varphi\right) \tag{4}$$

The torque M_Z can be neglected since its value is very low with respect to forces F_x and F_y. In a general form the two forces and the torque can be calculated integrating the pressure distributions in the contact area A as shown in Equations 5. Being the contact area very small the M_Z is a small value since the force terms F_X and F_Y are multiply by the x and y coordinates of the contact area. This remark allows neglecting the term M_Z without affecting the accuracy of the numerical model.

$$\begin{aligned} F_X &= \iint_A p_x dxdy \\ F_Y &= \iint_A p_y dxdy \\ M_Z &= \iint_A \left(xp_y - yp_y\right) dxdy \end{aligned} \tag{5}$$

The Kalker's coefficients have been calculated by Kalker in the linear theory; these coefficients were tabulated and used to calculate the contact forces without using iterative methods. These coefficients are also used in the Kalker's simplify method FASTSIM that allows calculating the contact forces with a good accuracy and with low calculation time. In literature there are different methods that can be used to calculate the contact forces and the most accurate method is the Kalker's CONTACT algorithm that is able to consider also non-elliptical contact areas. However this algorithm is much more slower than FASTSIM and it is not suitable for real-time simulations. Other methods are the Polach's method and the heuristic methods. The RTCONTACT algorithm includes the following methods:

- Kalker's linear theory
- FASTSIM
- Polach
- Heuristic method developed by the authors [22]

The method that allows the best balance between the accuracy and the calculation time is FASTSIM since it is more accurate than the linear theory and it is also able to consider the tangential force saturation, which occurs when the total tangential force is equal to the product of the normal force F_N and the friction coefficient μ, as shown in Equation 6.

$$\sqrt{F_X^2 + F_Y^2} \leq \mu F_N \tag{6}$$

VALIDATION OF THE CONTACT CODE

The contact code has been validated according to the UIC519 leaflet and the commercial multibody code Simpack. The validation with respect to the standard has been performed

considering the quantities shown in the UIC519 leaflet that only refers to the geometrical module. The UIC519 leaflet provides for different sets of wheel and rail profiles the rolling radii difference, the sum of the contact angles tangents and the equivalent conicity as a function of the wheelset lateral position. The wheel and rail profiles considered for the validation are respectively the RUIC519-A and SUIC519-A that are defined in the standard by means of piecewise functions. In order to include these profiles in the RTCONTACT code these functions have been used to generate tables which contain the points of the profiles. The table has been generated considering the same cant angle and profile positioning with respect to the track center line as that used on the UIC 519 leaflet. Figure 5 shows a comparison of the rolling radii difference vs. the wheelset lateral displacement according to the RTCONTACT code and the UIC519 standard.

The two curves are superimposed for little lateral displacements of the wheelset (thread contact), while they are different for displacements larger than 6 mm (flange contact). This difference is caused since the UIC standard does not take into account the wheelset roll angle, which has an important effect in case of flange contact. The rolling radii difference is very important since it is used to calculate the equivalent conicity that affects very much the vehicle running behavior.

In order to validate also the other algorithms of the contact code a comparison has been performed with respect to Simpack. The main task of the comparison is the evaluation of the accuracy of the contact algorithm and for this reason the model of a single wheelset on track has been considered. The two models have been developed using the same geometrical and inertial parameters, including the same description for the wheel and the rail profiles, which have been provided to the two codes using the same number of points and considering a track gauge of 1435 mm and a cant angle of 1/40. A first comparison takes into account the contact geometrical parameters as a function of the wheelset lateral position involving only the geometrical module of the code, which is very important because it is the most complex part of the algorithm. In this case no dynamic effect has been considered and the wheelset is moved statically in different lateral positions. For each lateral position of the wheelset the rolling radii difference, the contact angle and the contact point position have been calculated. Figure 6 shows the rolling radii difference vs. the wheelset lateral displacement obtained with Simpack and RTCONTACT. Figure 7 reports the contact point position of the right wheel with respect to the center of the rail profile, assumed to be located 0.75 m from the track center line.

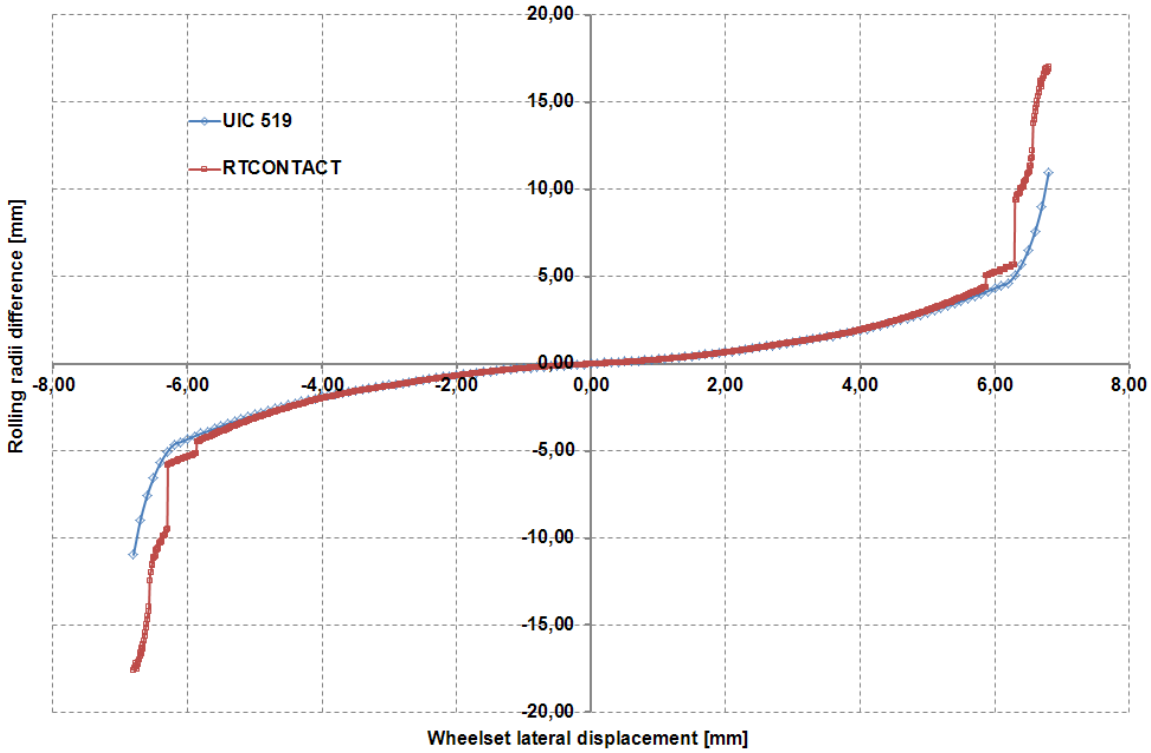

Figure 5: Rolling radii difference vs. wheelset lateral displacement.

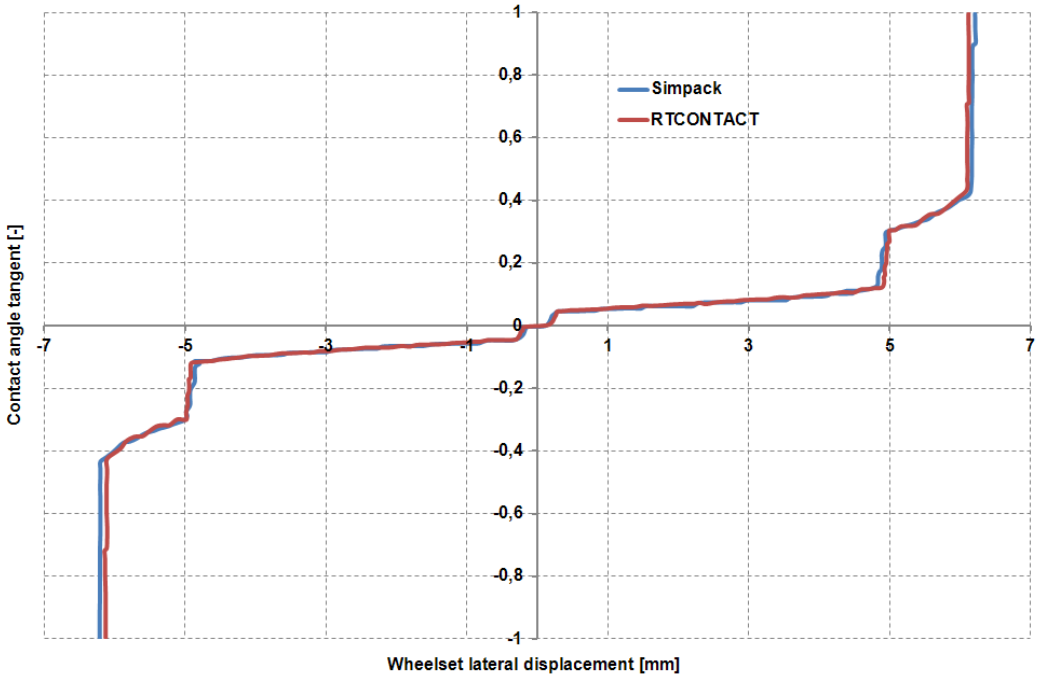

Figure 6: Contact angle tangent vs. wheelset lateral displacement.

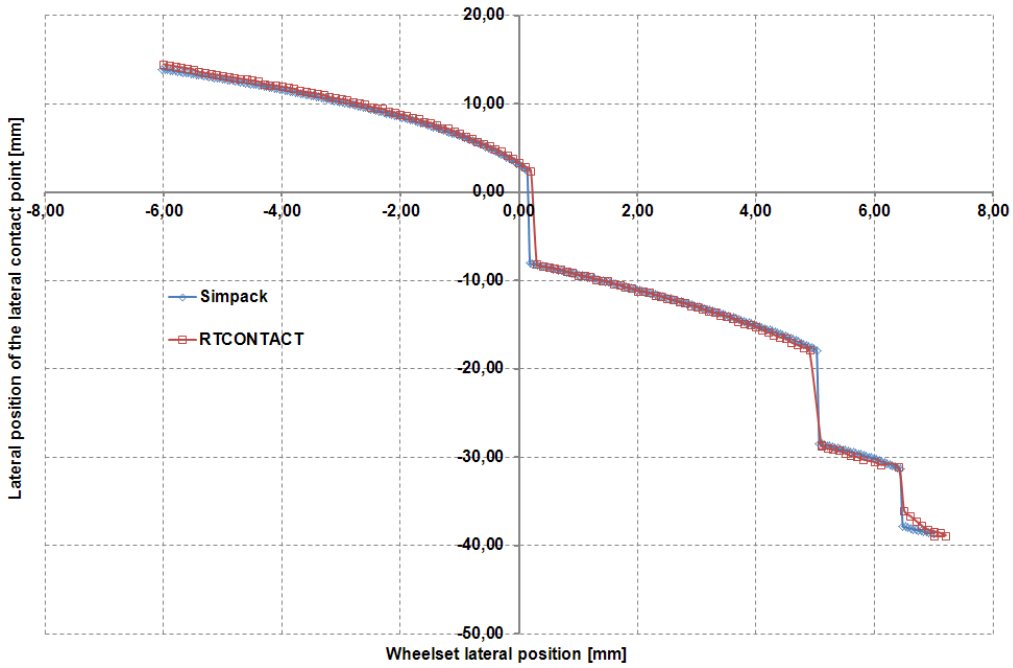

Figure 7: Lateral contact point position on the right wheel.

A second comparison between the RTCONTACT and the Simpack code has been performed considering a single wheelset running on track and taking into account the dynamic effects. The wheelset has an initial longitudinal velocity of 10 m/s and a lateral displacement of 2 mm. Under these conditions two tests have been performed considering a friction coefficient equal to 0.05 and 0.4. This comparison allows to test and validate the whole algorithm when the wheelset dynamic behavior is completely controlled by the contact forces. Figure 8 shows the wheelset lateral position when a friction coefficient of 0.4 is considered and in this case the wheelset has an unstable behavior. Figure 9 considers instead a friction coefficient equal to 0.05 and in this case the wheelset behavior is stable.

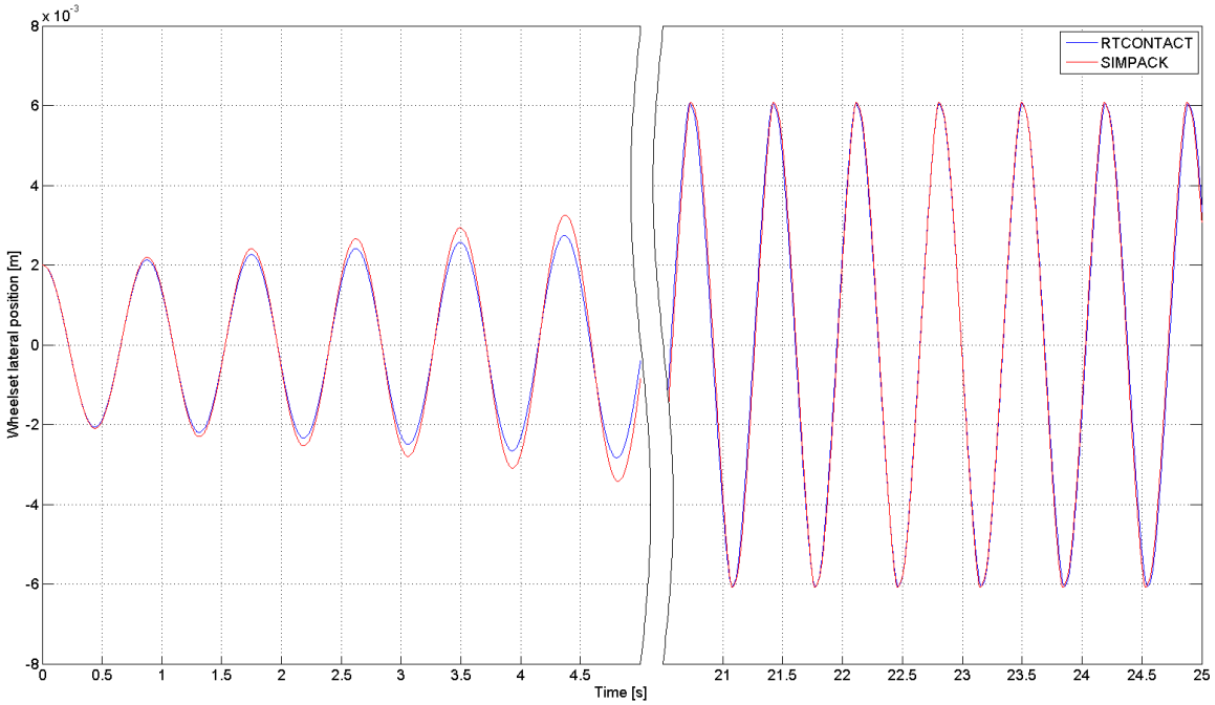

Figure 8: Wheelset lateral position over time when a friction coefficient of 0.4 is considered.

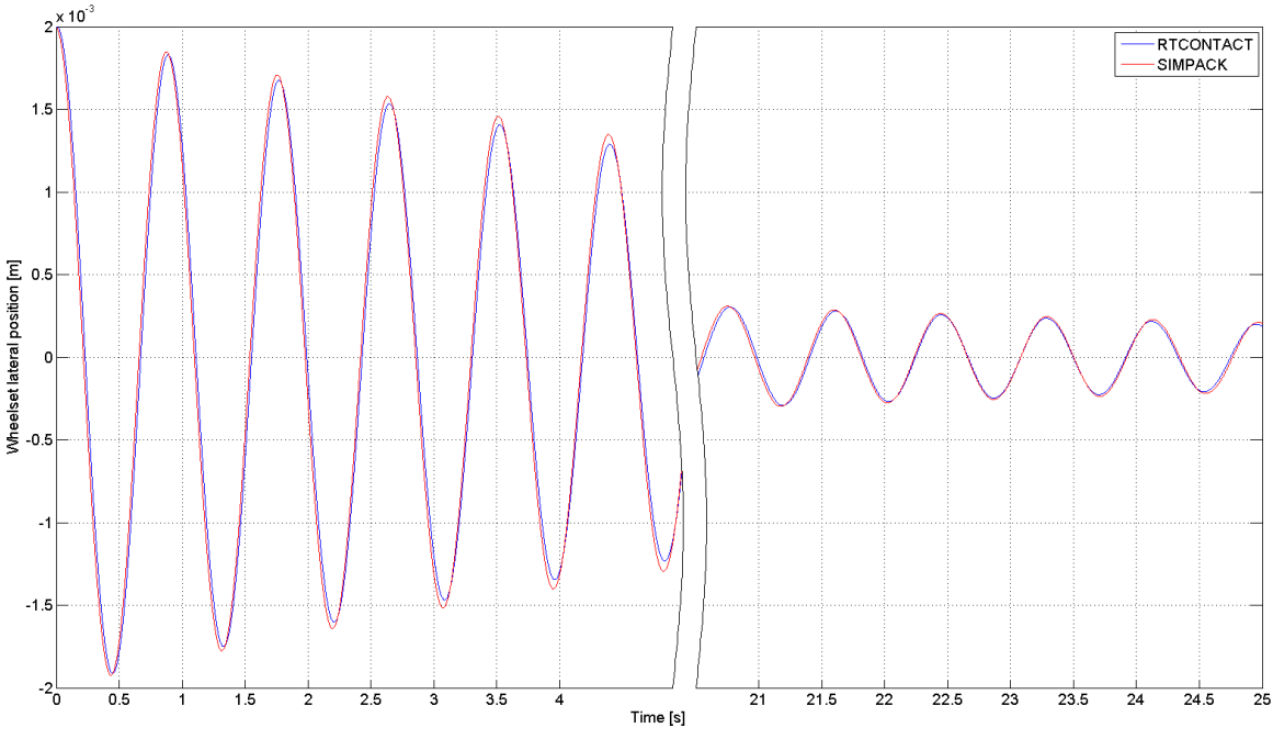

Figure 9: Wheelset lateral position over time when a friction coefficient of 0.05 is considered.

The RTCONTACT code has been experimentally validated on the roller-rig developed by the authors [17,18,19]. The test bench allows to reproduce in scale the wheel-rail contact conditions that occur during the vehicle operation. In particular it is possible to reproduce different scenarios in terms of adhesion condition, contact pressure and slip velocity, under well controlled conditions that can be replicated several times. The experimental device is composed by a 1:5 scaled single suspended wheelset supported by two independent rotating rollers, see Figure 10.

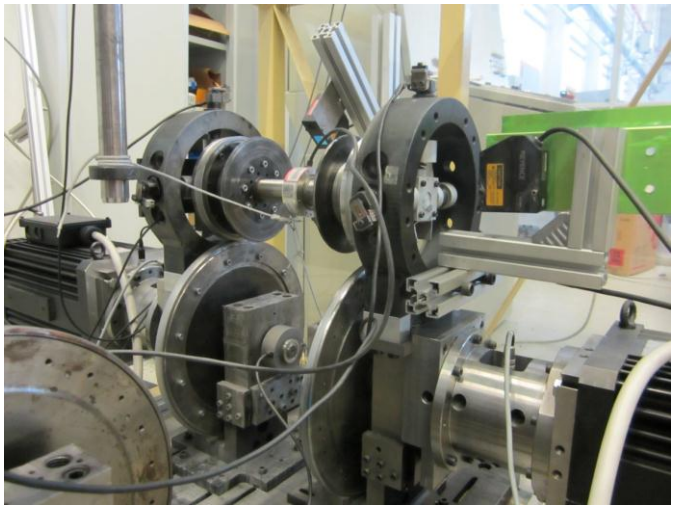

Figure 10: Roller-Rig with single suspended wheelset.

The two motors connected to the rollers are controlled by digital drives which communicates in real-time by means of a deterministic network (CTNET). The RTCONTACT code is executed in real-time on an industrial controller (based on an Intel Core 2 Quad Q9100 2.26 GHz processor) running LabVIEW-RT, which includes a data acquisition board to acquire the signals provided by the sensors installed on the test bench. The sensors installed on the roller rig are the following:

- 6 load cells to measure the forces acting on the axle-boxes;
- 6 accelerometers to measure the accelerations on the axle-boxes;
- 1 encoder to measure the wheelset angular velocity;
- 3 high precision laser sensors to measure the wheelset lateral position and yaw angle.

The yaw angle of the wheelset can be easily calculated according to Equation 7, where x_1 and x_2 are the longitudinal displacements of the two axle-boxes, while l is the distance between the measuring lasers. The angular velocity of the rollers is measured directly from the drives which acquire the signals from the two high precision digital encoders installed on the motors.

$$\psi = \frac{|x_1|+|x_2|}{l} \qquad (7)$$

In this case since a single wheelset is considered the contact code is executed on two cores according to the scheme of Figure 3. Once the contact forces are calculated the wheel and roller profiles are updated according to the algorithm described in [19]. This activity cannot be carried out in parallel with the contact forces calculation since the results of the RTCONTACT code are necessary for the wear calculation. In order to speed up this phase the four profiles (two rollers and two wheels) are updated in parallel using the four cores available.

The tests have been performed in order to experimentally validate the RTCONTACT code and in particular the module used to predict the profile shape evolution due to wear. The wear tests on the roller-rig have been performed imposing slip between the wheel and rail. This operation is performed decreasing the angular velocity of one motor while keeping constant the angular speed of the other one. Being the wheelset not connected to any traction system, the creepage at the left and right contact areas only depends from the velocity of the rollers and the friction force equilibrium at each contact patch. In any case the creepage value on both sides can be easily calculated from the angular speed of the rollers and the wheelset. Regulating the preload of the vertical spring it is possible to modify the normal load acting between the wheel and the roller. In this way the creepage value on the two wheels can be controlled with a good precision.

The roller rig includes a laser profilometer, designed by the authors, that allows to measure the wheel and roller profiles while the test bench is stopped. The system is composed by two high precision lasers (one for the rollers and one for the wheels), that are installed on a frame that is able to move on a slider. The lateral position of the frame is measured by a position transducer. In this way it is possible to acquire the wheel and roller profiles simply moving the frame parallel with the wheelset axis.

The wheel and roller profiles are acquired with the laser profilometer on both sides before running the wear tests. The acquired profiles have been then provided to the RTCONTACT code. When the wear test starts the contact code on the basis of the signal acquired from the sensors evaluates in real-time the evolution of the wheel and roller profiles. The real profiles are acquired several times during the test in order to be compared with the estimated ones. Since the aim of this work is the description of the procedure adopted to parallelize the RTCONTACT code no experimental results are shown in this section but they can be find in [19].

CONCLUSIONS

The work describes the wheel-rail contact code (RTCONTACT) developed by the railway research group at Politecnico di Torino. In particular the strategies adopted to parallelize the code are discussed and described distinguishing the sections of code that can be executed in parallel from the ones that must be executed in sequence. The paper considers the case of a single wheelset and a coach with four axles where the computational load due to the contact forces calculation is distributed on eight cores. The parallelization technique allows to speed-up the contact code taking advantage from the actual CPUs that are normally designed

with multicore technology. The real-time code has been validated with respect to the standards and the commercial MB code Simpack. The RTCONTACT code has been compiled and executed on a real-time controller to estimate the wheel and roller wear on a scaled roller-rig.

Competing Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

REFERENCES

[1] C.G. Kang, H.Y. Kim, M.S. Kim, and B.C. Goo, Real-time simulations of a railroad brake system using a dSPACE board, Proceedings of the ICROS-SICE International Joint Conference 2009, Fukuoka, Japan, 2009, pp. 4073–4078.

[2] B. Allotta, R. Conti, E. Meli, L. Pugi, A. Ridolfi, A. Rindi, Analysis and simulation of a Hardware In the Loop roller-rig model for the study of degraded adhesion condition in railway applications, Proceedings of the XXI AIMETA conference, Torino, Italy, 2013.

[3] P. D'Adamio, J. Escalona, E. Galardi, E. Meli, L. Pugi, A. Rindi, Real Time Modelling of a Railway Multibody Vehicle: Application and Validation on a Scaled Railway Vehicle, Proceedings of the Third International Conference on Railway Technology: Research, Development and Maintenance, Civil-Comp Press, 259, Stirlingshire, UK, 2016, doi: 10.4203/ccp.110.259.

[4] A.S. Abeidi, A. Gugliotta, N. Bosso, A. Somà, (2006) Numerical simulation of wear in railway wheel profiles, Proceedings of 8th Biennial ASME Conference on Engineering Systems Design and Analysis ESDA2006.

[5] N. Bosso, M. Spiryagin, A. Gugliotta, A. Somà (2013) Mechatronic Modeling of Real-Time Wheel-Rail Contact, Springer-Verlag Berlin Heidelberg, Berlin, DOI: 10.1007/978-3-642-36246-0.

[6] A. Eichberger, W. Rulka (2004) Process Save Reduction by Macro Joint Approach: The Key to Real Time and Efficient Vehicle Simulation, Vehicle System Dynamics, 41:5, 401-413.

[7] S.-Soo Kim, W. Jeong (2007) Subsystem synthesis method with approximate function approach for a real-time multibody vehicle model, Multibody System Dynamic, 17, 141–156, DOI: 10.1007/s11044-007-9038-6.

[8] M. Spiryagin , Y. Q. Sun , C. Cole , T. McSweeney , S. Simson and I. Persson (2013) Development of a real-time bogie test rig model based on railway specialised multibody software, Vehicle System Dynamics, 51:2, 236-250, DOI:10.1080/00423114.2012.724176.

[9] M. Spiryagin, S. Simson, C. Cole, I. Persson, (2012) Co-simulation of a mechatronic system using Gensys and Simulink, Vehicle System Dynamics, 50(3), 495–507.

[10] Y.J. Zhou, T.X. Mei, and S. Freer, FPGA Implementation of wheel–rail contact laws, IET Proc. Control Theory Appl. 4, 303–313, 2010.

[11] J. Piotrowski, (2010) Kalker's algorithm Fastsim solves tangential contact problems with slip-dependent friction and friction anisotropy, Veh Syst Dyn., 48:7, 869-889, DOI: 10.1080/00423110903178495

[12] E. A. H. Vollebregt, P. Wilders (2011), FASTSIM2: a second-order accurate frictional rolling contact algorithm, Computational Mechanics, 47:105–116, doi: 10.1007/s00466-010-0536-7.

[13] N. Bosso, A. Gugliotta, and N. Zampieri, RTCONTACT: An efficient wheel–rail contact algorithm for real-time dynamic simulations, Proceedings of the ASME/ASCE/IEEE 2012 Joint Rail Conference (JRC2012), Philadelphia, PA, 2012, DOI:10.1115/JRC2012-74044.

[14] N. Bosso, N. Zampieri (2013) Real-time implementation of a traction control algorithm on a scaled roller rig, Vehicle System Dynamics, 51:4, 517-541, DOI: 10.1080/00423114.2012.750001.

[15] Y. J. Zhou, T. X. Mei, S. Freer, FPGA Implementation of wheel–rail contact laws, IET Proc. Control Theory Appl., 4, 303–313, 2010, doi: 10.1049/iet-cta.2008.0601.

[16] Y. Zhou, T.X. Mei, S. Freear (2010) Real-time modeling of wheel-rail contact laws with system-on-chip, IEEE Trans. Parallel Distrib. Syst., 672–684.

[17] N. Bosso, A. Gugliotta, A. Somà (2004) Dynamic behavior of a railway wheelset on a roller rig versus tangent track, Shock and Vibration, 11, 467-492.

[18] N. Bosso, A. Gugliotta, N. Zampieri (2015) Strategies to simulate wheel–rail adhesion in degraded conditions using a roller-rig, Vehicle System Dynamics, 53:5, 619-634, doi: 10.1080/00423114.2014.981194.

[19] N. Bosso, N. Zampieri (2014) Experimental and numerical simulation of wheel-rail adhesion and wear using a scaled roller rig and a real-time contact

code, Shock and Vibration, vol. 2014, 1-14, doi: 10.1155/2014/385018.

[20] N. Bosso, N. Zampieri (2017) Long train simulation using a multibody code, Vehicle System Dynamics, 55:4, 552-570, doi: 10.1080/00423114 .2016.1267373.

[21] P.D. Panagiotopoulos (1975) A nonlinear programming approach to the unilateral contact and friction-boundary value problem in the theory of elasticity, Ingenieur, 44, 421-432.

[22] N. Bosso, A. Gugliotta, N. Zampieri (2013) Study of Adhesion and Evaluation of the Friction Forces Using a Scaled Roller-Rig, 5th World Tribology Congress, WTC 2013, 3, 2640-2643, Torino, Italy.