TRANSPORT PROBLEMS 2019 Volume 14 Issue 3
PROBLEMY TRANSPORTU DOI: 10.20858/tp.2019.14.3.15

Keywords: multibody dynamics; parallel computations; co-simulation; railway research

Dmitry POGORELOV, Alexander RODIKOV, Roman KOVALEV*
Bryansk State Technical University, Laboratory of Computational Mechanics
bulv. 50 let Oktyabrya 7, Bryansk, 241035, Russia
**Corresponding author*. E-mail: kovalev@umlab.ru

PARALLEL COMPUTATIONS AND CO-SIMULATION IN UNIVERSAL MECHANISM SOFTWARE. PART I: ALGORITHMS AND IMPLEMENTATION

Summary. Parallel computations speed up simulation of multibody system dynamics, in particular, dynamics of railway vehicles and trains. It is important for reduction of required time at the stage of new railway vehicle design, for increase of complexity of studied problems and for real-time applications. We consider realization of parallel computations in Universal Mechanism software in three different areas: simulation of rail vehicle and train dynamics, evaluation of wheel profile wear and multi-variant computations. The use of clusters for parallel running of multi-variant computations is illustrated. Co-simulation based on the interface between Universal Mechanism and Matlab/Simulink and other software tools is discussed.

1. INTRODUCTION

In this paper, we consider three areas for parallel computations:
- simulation of rail vehicle and train dynamics,
- evolution of railway wheel profile due to wear and
- multi-variant computations.

In the first two cases, the parallelism is realized on multi-core processors with the help of the multithread technique. The third one is based on the multiprocess technique that might use not only the local computer but also the network computational resources.

Parallel computation is a well-known tool for speeding up the analysis of different scientific and engineering tasks. Compared with general multibody system (MBS) analysis, rail vehicle research has several specific features that should be taken into account [1]. For instance, dynamic models along with vehicles often include flexible rails and bridges, which increase the number of degrees of freedom and make parallel computations desired.

Bibliography of parallelism in simulations of MBS dynamics includes hundreds of publications. Main theoretical results in this field are related to parallel treating articulated body systems with long kinematic chains. The divide-and-conquer (DCA) algorithm [2, 3] and the constraint force algorithm (CFA) [4] are the O(log N) algorithms, i.e. they require O(log N) operations in simulation of an articulated MBS with N rigid bodies on N processors. DCA, CFA and further developments of these algorithms [5-10] do not support implicit solver procedures for stiff MBS, which limits their application to simulation of rail vehicle dynamics.

Another important area for application of parallelism is related to molecular dynamics and mathematically similar problems like DEM, SPH and so on [11, 12]. Concerning railway dynamic problems, these methods are actual in simulation of track ballast [13, 14] as well as fluid and granular media sloshing [15].

A promising method of parallelization in the case of limited number of processors consists in dividing an MBS into several subsystems. Subsystems are selected in different manner, either by natural grouping of bodies and force elements [16, 17] or by cutting some joints [18]. Weakly connected subsystems are considered in the article by Getmansky & Gorobtsov [19].

In a particular case, any MBS can be considered as a set of separate bodies after cutting all joints. This method corresponds to the Cartesian MBS formulation. Joints are taken into account by constraint equations [20, 21] or replaced by spring-and-damper force elements (compliant joints) [22-24]. Replacing rigid joints by compliant ones allows a very efficient parallelization of any MBS, but it makes equations of motion stiff. This fact was the main reason why the compliant joints could not be widely used in MBS software codes. To overcome the problems connected with the stiffness of equations, a heterogeneous multiscale method is proposed in Valasek and Mraz, as well as Mraz and Valasek [22, 23], to be used for numerical integration, whereas a special implicit solver is used in Pogorelov [24]. This paper includes an advanced description of the parallel algorithm proposed in Pogorelov [24] and implemented in Universal Mechanism (UM) software [25] for simulation of dynamics of general multibody systems and, in particular, rail vehicles and trains.

The Cartesian formulation provides a good balanced parallel computation of kinematic relations, equations of motion and forces. At the same time, a large system of linear equations with a sparse system matrix must be solved many times during the integration process. In the case of rigid joints, the system matrix is symmetric but not positive definite and includes the mass matrix and the Jacobian matrix of constraint equations as blocks. The size of the matrix is $6N+m$, where N is the number of bodies and m is the number of constraint equations [20]. If the compliant joints are used, the system matrix is a sum of the mass matrix and the Jacobian matrix of stiff forces including compliant joints, the matrix size is $6N$, and it is positive definite [26, 27]. Parallel solving linear equations with a sparse matrix can be executed with the preconditioned conjugate gradient method (PCGM) [28-30]; some papers [31, 32] give examples of PCGM use in parallel MBS simulations. As it is shown in Section 2 of this paper, a block-diagonal preconditioning matrix can be constructed as the dominant one in a definite sense, and a small number of PCGM iterations are required.

In the case of MBS tasks, parallel computations are usually executed on multicore processors, GPU and clusters. In our opinion, simulations of rail vehicle and train dynamics should be implemented on multicore processors. GPU are highly productive in solving problems of molecular dynamics, and clusters are efficient in the case of multi-variant simulations, in particular, when solving optimization problems.

Parallel simulation of MBS dynamics on multicore processors as it is implemented in UM software is considered in Section 2. The section includes general forms of equations of motion for rigid and flexible body systems according to the Cartesian formalism with compliant joints, description of the Park implicit solver and details of a parallel algorithm based on the fork-join method. Examples of parallel simulation of railway vehicle and train dynamics are given in Part II of this paper.

The implementation of parallel approach to predict wear of railway wheel profiles on multicore processors is considered in Section 3. Two different methods are usually used for prediction of profile wear [33]: the sequential method [34, 35] and the parallel one [36, 37]. Both methods consider simulation of the rail vehicle dynamics on track sections with different geometry taking into account various irregularities, rail profiles, vehicle mass, speeds and so on. The methods differ in the strategy of modification of profiles. In the sequential algorithm, the profiles are modified at the end of each simulation. In the parallel algorithm, the modification of profiles due to wear is performed many times in small intervals of the traveling distance. The parallel algorithm is faster than the sequential one, but the latter method is applicable for rail profile wear as well. An example of use of a multicore processor for speeding up the evaluation of wheel profile wear is given in Part II of this paper.

Parallel implementation of multi-variant computations in UM software is considered in Section 4. Some problems connected with co-simulation in MBS dynamics are discussed in Section 5.

Later we will use the following terms and definitions. A process is an executing instance of a program. On a multiprocessor system, multiple processes can be executed in parallel. A thread is a subset of the process. A process has at least one thread. A process may also be made up of multiple

threads of execution that execute instructions concurrently. Multiple threads of control can exploit the true parallelism possible on multiprocessor systems.

2. PARALLEL SIMULATION OF RAIL VEHICLE AND TRAIN DYNAMICS

In this section, we consider an algorithm for parallel simulation of rail vehicles and trains dynamics on computers with multi-core processors.

2.1. Equations of motion

First, consider a rigid body with six degrees of freedom. Position of body *i* is set by three Cartesian coordinates $\mathbf{r}_i = (x_i, y_i, z_i)^T$ for the origin of the body-fixed system of coordinates, and by three orientation coordinates $\mathbf{p}_i$. Kinematics of body *i* includes formulas for linear and angular velocities and accelerations $\mathbf{v}_i, \mathbf{a}_i, \boldsymbol{\omega}_i, \boldsymbol{\varepsilon}_i$ as well as for the direct cosine matrix $\mathbf{A}_{0i}$

$$\mathbf{v}_i = \dot{\mathbf{r}}_i,\ \mathbf{a}_i = \ddot{\mathbf{r}}_i,\ \mathbf{A}_{0i} = \mathbf{A}_{0i}(\mathbf{p}_i),\ \boldsymbol{\omega}_i = \mathbf{B}_i\dot{\mathbf{p}}_i,\ \boldsymbol{\varepsilon}_i = \mathbf{B}_i\ddot{\mathbf{p}}_i + \boldsymbol{\varepsilon}'_i. \tag{1}$$

Newton-Euler dynamic equations in accelerations are as follows:

$$\begin{aligned} &\mathbf{M}_{ir}\mathbf{w}_{ir} = -\mathbf{k}_{ir} + \sum_j \mathbf{G}_{ijr}, \\ &\mathbf{w}_{ir} = (\mathbf{a}_i^T \quad \boldsymbol{\varepsilon}_i^T)^T = \mathbf{W}_{ir}\ddot{\mathbf{q}}_i + \mathbf{w}'_{ir},\ \mathbf{q}_i = (\mathbf{r}_i^T \quad \mathbf{p}_i^T)^T,\ \mathbf{w}'_{ir} = (\mathbf{0} \quad \boldsymbol{\varepsilon}_i'^T)^T, \end{aligned} \tag{2}$$

where $\mathbf{M}_{ir}$ is the 6×6 mass matrix, and $\mathbf{w}_{ir}, \mathbf{k}_{ir}, \mathbf{G}_{ijr}$ are the 6×1 vectors of accelerations, inertia forces and applied forces.

$$\mathbf{M}_{ir} = \begin{pmatrix} m_i\mathbf{E}_3 & -m_i\tilde{\boldsymbol{\rho}}_{ci} \\ m_i\tilde{\boldsymbol{\rho}}_{ci} & \mathbf{I}_i \end{pmatrix},\ \mathbf{W}_{ir} = \begin{pmatrix} \mathbf{E}_3 & \mathbf{0} \\ \mathbf{0} & \mathbf{B}_i \end{pmatrix},\ \mathbf{k}_{ir} = \begin{pmatrix} m_i\tilde{\boldsymbol{\omega}}_i\tilde{\boldsymbol{\omega}}_i\boldsymbol{\rho}_{ci} \\ \tilde{\boldsymbol{\omega}}_i\mathbf{I}_i\boldsymbol{\omega}_i \end{pmatrix}, \mathbf{G}_{ijr} = \begin{pmatrix} \mathbf{F}_{ij} \\ \mathbf{T}_{ij} \end{pmatrix}.$$

Here m_i is the mass, $\mathbf{I}_i$ is the matrix of inertia tensor, $\boldsymbol{\rho}_{ci}$ is the radius-vector of centre of mass relative to the body-fixed frame, $\mathbf{F}_{ij}, \mathbf{T}_{ij}$ are applied force and torque corresponding to interaction between bodies *i* and *j* including forces for compliant joints and $\mathbf{E}_3$ is the 3×3 identity matrix. The symbol 'tilde' over a vector denotes a skew-symmetric matrix generated by the vector and used for the matrix notation of the vector product.

Flexible bodies are used in railway research for modelling both parts of vehicles and elements of track infrastructure like rails, sleepers and bridges. Equations of motion of flexible bodies in multibody system dynamics are often derived with Craig-Bampton method [38].The body movement is split on a gross motion of a reference frame and small elastic deformations [39]. Coordinates and kinematics of the reference frame coincide with aforementioned ones for a rigid body (1). A vector to a separate body node *k* is as follows:

$$\mathbf{r}_{ik} = \mathbf{r}_i + \mathbf{A}_{0i}(\boldsymbol{\rho}_{ik} + \Delta\mathbf{u}_{ik})$$

with elastic displacement $\Delta\mathbf{u}_{ik}$ of the node relative to reference frame. According to the Craig-Bampton methodology, a set of constraint and fixed interface (static and dynamic) nodal modes specify the elastic deformations so that the node displacement $\Delta\mathbf{u}_{ik}$ and rotation $\Delta\boldsymbol{\pi}_{ik}$ are linear functions of generalized elastic coordinates $\mathbf{q}_{if}$

$$\Delta\mathbf{u}_{ik} = \mathbf{H}_{iku}\mathbf{q}_{if},\ \Delta\boldsymbol{\pi}_{ik} = \mathbf{H}_{ik\pi}\mathbf{q}_{if}.$$

The equations of motion of flexible body are as follows:

$$\mathbf{M}_{if}\mathbf{w}_{if} + \mathbf{C}_{if}\dot{\mathbf{q}}_i + \mathbf{K}_{if}\mathbf{q}_i + \mathbf{k}_{if} = \sum_k \mathbf{G}_{ikf}$$

$$\mathbf{w}_{if} = \begin{pmatrix} \mathbf{a}_i \\ \boldsymbol{\varepsilon}_i \\ \ddot{\mathbf{q}}_{if} \end{pmatrix}, \mathbf{G}_{ikf} = \begin{pmatrix} \mathbf{F}_{ik} \\ \mathbf{T}_{ik} + \tilde{\boldsymbol{\rho}}_{ik}\mathbf{F}_{ik} \\ \mathbf{H}_{iku}^T\mathbf{A}_{i0}\mathbf{F}_{ik} + \mathbf{H}_{ik\pi}^T\mathbf{A}_{i0}\mathbf{T}_{ik} \end{pmatrix}, \tag{3}$$

where $\mathbf{C}_{if}, \mathbf{K}_{if}$ are the damping and stiffness matrices, and $\mathbf{F}_{ik}$, $\mathbf{T}_{ik}$ are the force and torque applied to node *k*.

Finally, consider equations of motion for a flexible structure in nodal coordinates, which can be used for modeling flexible rails and sleepers. In contrary to the previous case, the nonlinear gross motion does not take place, and equation of motion can be derived according to the usual finite element method. Both rails and sleepers are considered here as Timoshenko or Euler-Bernoulli beams with the following equations of motion for a beam with n_{ib} nodes:

$$\mathbf{M}_{ib}\ddot{\mathbf{q}}_{ib} + \mathbf{C}_{ib}\dot{\mathbf{q}}_{ib} + \mathbf{K}_{ib}\mathbf{q}_{ib} = \sum_k \mathbf{G}_{ikb}. \tag{4}$$

Here $\mathbf{q}_{ib}$ is vector of $6n_{ib}$ nodal coordinates, $\mathbf{M}_{ib}, \mathbf{C}_{ib}, \mathbf{K}_{ib}$ are block tridiagonal mass, damping and stiffness matrices of size $6n_{ib} \times 6n_{ib}$ and $\mathbf{G}_{ikb}$ is the $6n_{ib} \times 1$ vector of applied forces and torques at node *k*.

The general form of equation of motion for body *i* of types (2-4) is as follows:

$$\mathbf{M}_i\mathbf{w}_i + \mathbf{C}_i\dot{\mathbf{q}}_i + \mathbf{K}_i\mathbf{q}_i + \mathbf{k}_i = \sum_j \mathbf{G}_{ij},\ \mathbf{w}_i = \mathbf{W}_i\ddot{\mathbf{q}}_i + \mathbf{w}_i'. \tag{5}$$

The matrices $\mathbf{C}_i, \mathbf{K}_i$ in (5) should be omitted for rigid bodies.

2.2. Numerical method

Equations of motion (5) are stiff from the numerical point of view [40] by two main reasons. First, if flexible bodies are presented in the model, internal elastic and damping forces are stiff. Second, compliant joints, contact forces and some other force elements make equations stiff as well. The implicit Park numerical method is used in UM software for solving stiff equations of motion [41]. The linear multistep Park method is based on combination of backward differentiation formulas of the second and third orders. Consider some details of the method, which are important for description of the parallel computations.

At the beginning of each integration step, predictor values of the coordinates $\mathbf{q}_i^p$ for body *i* are computed. Here and below we omit the step index. The unknown corrector values $\delta\mathbf{q}_i$ are introduced so that $\mathbf{q}_i = \mathbf{q}_i^p + \delta\mathbf{q}_i$. The Park finite difference formula is used to express the time derivatives of coordinates, velocities and accelerations in term of the corrector value

$$\dot{\mathbf{q}}_i = \dot{\mathbf{q}}_i^p + \delta\mathbf{q}_i/\beta,\ \ddot{\mathbf{q}}_i = \ddot{\mathbf{q}}_i^p + \delta\mathbf{q}_i/\beta^2,\ \mathbf{w}_i = \mathbf{w}_i^p + \mathbf{W}_i\,\delta\mathbf{q}_i/\beta^2,\ \beta = 0.6h. \tag{6}$$

Here *h* is the step size of integration.

Let us introduce new unknown variables

$$\delta\mathbf{q}_i' = \mathbf{W}_i\delta\mathbf{q}_i, \tag{7}$$

in which equations can be written in a more compact form. For example,

$$\mathbf{w}_i = \mathbf{w}_i^p + \delta\mathbf{q}_i'/\beta^2. \tag{8}$$

Now we introduce Jacobian matrices (JM) for stiff force elements. Let the generalized force $\mathbf{G}_{ij}$ in Equations (5) be stiff and i, j are the indices of interacting bodies. Suppose the force depends on

positions and velocities of interacting bodies. Implicit integration scheme requires expansion of stiff forces and torques in series of $\delta\mathbf{q}'_i, \delta\mathbf{q}'_j$ neglecting nonlinear terms

$$\mathbf{G}_{ij} = \mathbf{G}_{ij}^p + \mathbf{J}_{ij,i}\delta\mathbf{q}'_i + \mathbf{J}_{ij,j}\delta\mathbf{q}'_j, \tag{9}$$

where $\mathbf{J}_{ij,i}, \mathbf{J}_{ij,j}$ are the JM of forces. Approximate expressions for JM of different forces and compliant joints are derived in Pogorelov and Pogorelov [26, 27]. It is important that for the stiff forces, used in simulation of rail vehicles, the matrix

$$\begin{pmatrix} \mathbf{J}_{ij,i} & \mathbf{J}_{ij,j} \\ \mathbf{J}_{ji,i} & \mathbf{J}_{ji,j} \end{pmatrix}$$

composed of the approximate JM is negative semidefinite [26, 27].

Substitution of Equations (7-9) in (5) yields linear equations relative to the unknown values $\delta\mathbf{q}'_i$

$$\mathbf{M}'_i\delta\mathbf{q}'_i - \beta^2 \sum_{j\in S_i} \mathbf{J}_{ij,j}\delta\mathbf{q}'_j = \beta^2\mathbf{Q}_i \tag{10}$$

with the set S_i of stiff force indices for body *i*, the positive definite matrix $\mathbf{M}'_i$

$$\mathbf{M}'_i = \mathbf{M}_i^p + \beta\mathbf{C}_i + \beta^2\mathbf{K}_i - \beta^2 \sum_{j\in S_i} \mathbf{J}_{ij,i} \tag{11}$$

and the vector summarizing all forces

$$\mathbf{Q}_i = -\mathbf{M}_i^p\mathbf{w}_i^p - \mathbf{C}_i\dot{\mathbf{q}}_i^p - \mathbf{K}_i\mathbf{q}_i^p - \mathbf{k}_i^p + \sum_j \mathbf{G}_{ij}^p. \tag{12}$$

The PCGM [28, 29] is suitable for parallel solving Equations (10) and the $\mathbf{M}' = \mathrm{diag}\{\mathbf{M}'_i\}$ matrix is highly efficient as the preconditioning matrix in this algorithm. To explain this statement, consider Equations (10) in the full matrix form

$$(\mathbf{M}' - \Delta\mathbf{J})\delta\mathbf{q}' = \mathbf{Q}.$$

According to a theorem proved in Pogorelov [26], the spectral radius of the matrix $\mathbf{M}'^{-1}\Delta\mathbf{J}$ is less than 1,

$$\rho(\mathbf{M}'^{-1}\Delta\mathbf{J}) < 1.$$

In this sense, the preconditioning matrix $\mathbf{M}'$ is a dominant one and a good convergence of the PCGM iterations is observed. If we take into account that the mass matrix is completely included in the preconditioning matrix, the PCGM iterations serves to stabilization of numerical method rather than to increase of the accuracy. As a result, the number of iterations is usually small and rarely exceeds one iteration.

Consider steps of the PCGM for solving Equations (10) in the form suitable for parallel computations.

- Cholesky factorization of blocks of the preconditioning matrix (11)

$$\mathbf{M}'_i = \mathbf{L}_i\mathbf{L}_i^T. \tag{13}$$

Now Equations (10) can be rewritten for the variables $\delta\mathbf{q}_i^L = \mathbf{L}_i^T\delta\mathbf{q}'_i$:

$$\begin{gathered} \delta\mathbf{q}_i^L - \sum_{j\in S_i} \mathbf{J}_{ij,j}^L\delta\mathbf{q}_j^L = \mathbf{Q}_i^L, \\ \mathbf{J}_{ij,j}^L = \beta^2\mathbf{L}_i^{-1}\mathbf{J}_{ij,j}\mathbf{L}_j^{-T},\ \mathbf{Q}_i^L = \beta^2\mathbf{L}_i^{-1}\mathbf{Q}_i. \end{gathered} \tag{14}$$

- PCGM start: computation of initial solution, direction and residual $\delta\mathbf{q}_i^{L0}, \mathbf{d}_i^0, \mathbf{r}_i^0$:

$$\delta\mathbf{q}_i^{L0} = \mathbf{Q}_i^L,\ \mathbf{r}_i^0 = \mathbf{d}_i^0 = -\sum_{j\in S_i} \mathbf{J}_{ij,j}^L\delta\mathbf{q}_j^{L0},\ \gamma = 0. \tag{15}$$

- PCGM iterations include three steps; k=1,2,… is the index of iterations.

CGM1. $\mathbf{d}_i^k = \mathbf{r}_i^{k-1} + \gamma \mathbf{d}_i^{k-1}$; $\mathbf{d}_i^{kL} = \mathbf{L}_i^{-T}\mathbf{d}_i^k$.

CGM2. $\mathbf{c}_i^k = \mathbf{d}_i^k - \beta^2 \mathbf{L}_i^{-1}\sum_j \mathbf{J}_{ij}\mathbf{d}_j^{Lk}$; $\alpha = -\sum \mathbf{d}_i^{kT}\mathbf{r}_i^k \big/ \sum \mathbf{d}_i^{kT}\mathbf{c}_i^k$.

CGM3. $\delta\mathbf{q}_i^{Lk} = \delta\mathbf{q}_i^{Lk-1} + \alpha\mathbf{d}_i^k$; $\mathbf{r}_i^k = \mathbf{r}_i^{k-1} + \alpha\mathbf{c}_i^k$; $\rho_k = \sum \mathbf{r}_i^{kT}\mathbf{r}_i^k$; $\gamma = \rho_k / \rho_{k-1}$.

Iterations stop when $\sqrt{\rho_k} < \varepsilon$, where ε is the error tolerance.

- After the convergence of iterations, the corrector values are computed

$$\delta\mathbf{q}_i' = \mathbf{L}_i^{-T}\delta\mathbf{q}_i^L \tag{16}$$

as well as the final calculation of the model coordinates is done

$$\mathbf{q}_i = \mathbf{q}_i^p + \mathbf{W}_i^{-1}\delta\mathbf{q}_i'. \tag{17}$$

2.3. Parallel solution of equations of motion on multi-core processors

The technique, based on the Windows API threads, is used for practical realization of parallel computations in UM according to the algorithms described in Section 2.2. The fork-join model of parallelism uses a master thread and several parallel sections (PSs) on each integration step [42], Fig. 1. The total number of threads is recommended not to exceed the number of cores on the local computer. There exists a barrier at the end of each PS, and the balancing of computational load in the threads is important.

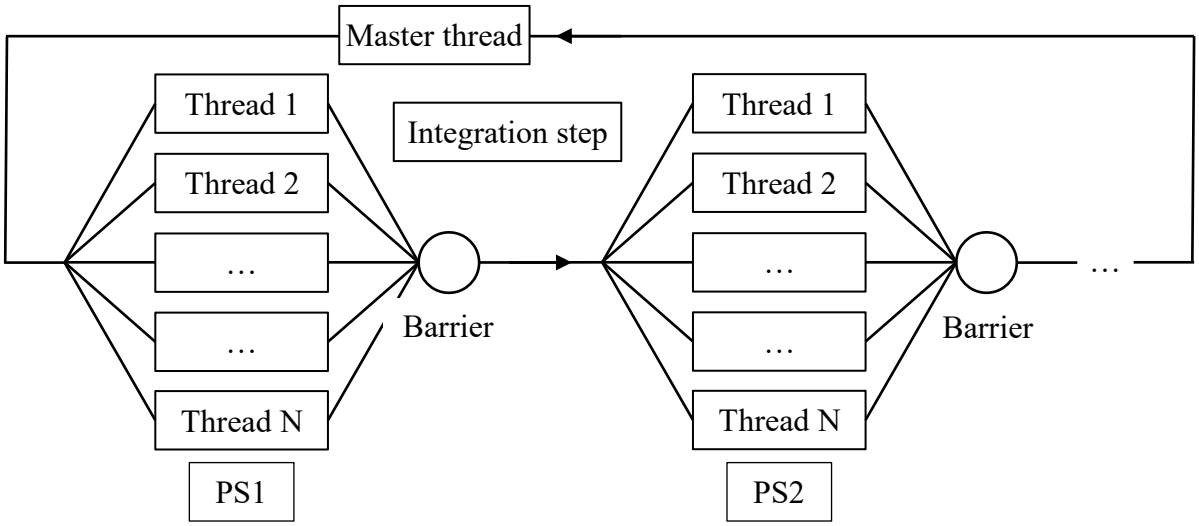

Fig. 1. Fork-join model of parallel computation

Consider the list of PSs according to the algorithm described in Section 2.2.

PS1. Prediction $\mathbf{q}_i^p, \dot{\mathbf{q}}_i^p, \ddot{\mathbf{q}}_i^p$, computation of kinematics (1), mass matrices $\mathbf{M}_i$, internal elastic and damping forces $\mathbf{C}_i\dot{\mathbf{q}}_i^p + \mathbf{K}_i\mathbf{q}_i^p$, inertia $\mathbf{k}_i$ and gravity forces in Equations (5).

PS2. Evaluation of forces and Jacobian matrices $\mathbf{G}_{ij}^p, \mathbf{J}_{ij,i}, \mathbf{J}_{ij,j}$, Equations (9).

PS3. Evaluation (11) and factorization (13) of the preconditioning matrices $\mathbf{M}_i'$, computation of total forces in Equations (12) as well as $\mathbf{Q}_i^L$ in (14).

PS4. Computation of start vectors of the PCGM $\delta\mathbf{q}_i^{L0}, \mathbf{r}_i^0, \mathbf{d}_i^0$ (15).

The next three sections correspond to iterations of PCGM.

PS5. Evaluation of $\mathbf{d}_i^k$, CGM1.

PS6. Evaluation of $\mathbf{c}_i^k$, CGM2.

PS7. Evaluation of $\delta\mathbf{q}_i^{Lk}$, $\mathbf{r}_i^k$, CGM3.

PS8. Computation of the corrector values $\delta\mathbf{q}_i'$ (16).

PS9. Computation of new values of coordinates (17), velocities and accelerations (6).

All PSs make computations independent for each of the bodies except the evaluation of forces and Jacobians in PS2. Thus, a good balancing of the CPU load in the case of a rigid body MBS is usually possible after splitting the total set of bodies into near-equal subsets. To decrease the thread load disbalance in computation of forces in PS2, the force elements are preliminary grouped according to their type and uniformly distributed between the threads.

In the case of a hybrid MBS including flexible bodies, an improvement of CPU load balancing can be achieved by using a simple algorithm for optimal distribution of computations between threads. The algorithm relates to the combinatorial generalized stone problem. Let w_{ij} be the average CPU time, which is necessary to run computational procedure i in PSj for one body or force element. We accept this parameter as a weight of the procedure in the corresponding PS. The expected computation load for thread k is as follows:

$$W_{jk} \approx \sum_{i \in S_{jk}} w_{ij}$$

where S_{jk} is the set of bodies or forces assigned to thread k in PSj. The optimal distribution of computation procedures between threads is formulated as the minimax problem

$$\min \max_k W_{jk},$$

i.e. as minimization of the maximal CPU load by changing the sets S_{jk}. An appropriate approximation of the optimal solution for N threads can be found by the following algorithm:

- computational procedures are sorted in decreasing order of the weight $w_{1j} \geq w_{2j} \ldots$;
- the first N procedures with the maximal weights are assigned to each thread;
- each of the next procedures is assigned to the thread with the minimum aggregate weight.

3. EVOLUTION OF RAILWAY WHEEL PROFILE DUE TO WEAR

UM software has a special tool for prediction of railway wheel profiles evolution due to wear. In this tool, a parallel discrete approach has been implemented. This algorithm supposes a parallel simulation of a railway vehicle model with different configurations which somehow approximate the real operational conditions of the vehicle. Configurations differ in track geometry and irregularities, rail profiles, vehicle mass, speed and so on. The set of configurations should be a representative set of conditions in which the rail vehicle is operated.

A track length travelled by the vehicle during the simulation is divided into a sequence of intervals (wear steps). The number of intervals is the same for all the configurations. The wheel profiles are kept unchanged within each of the interval. The wheel profiles are modified at the end of each wear step according to weighted diagrams of distribution of friction work along the wheel profile. A small quantity of material is removed on each wear step. Large number of steps results in a smooth profile modification. The calculation of material losses is based on the theory proposed by J.F. Archard [43]. Contact forces are computed using the model of W. Kik and J. Piotrowski [33] or the CONTACT library [44].

Multithread parallel computations are realized as follows (Fig. 2):

- at each wear step, the configurations are computed in parallel using a fixed number of threads,
- the computation barrier at the end of the section corresponds to the end of the wear step and

- modification of profiles and rebalancing of threads is executed before the start of the next step.

A weight of a configuration for thread balancing is equal to its CPU time at the previous wear step. Similar to the optimization procedure described in Section 2.3, the list of configurations is sorted in decreasing order of the weights. At the begin of a new step, the parallel computation starts with the configurations of the maximal weight, and when a thread finishes the computation for the assigned configuration it obtains the next one with the maximal weight from the rest of the list.

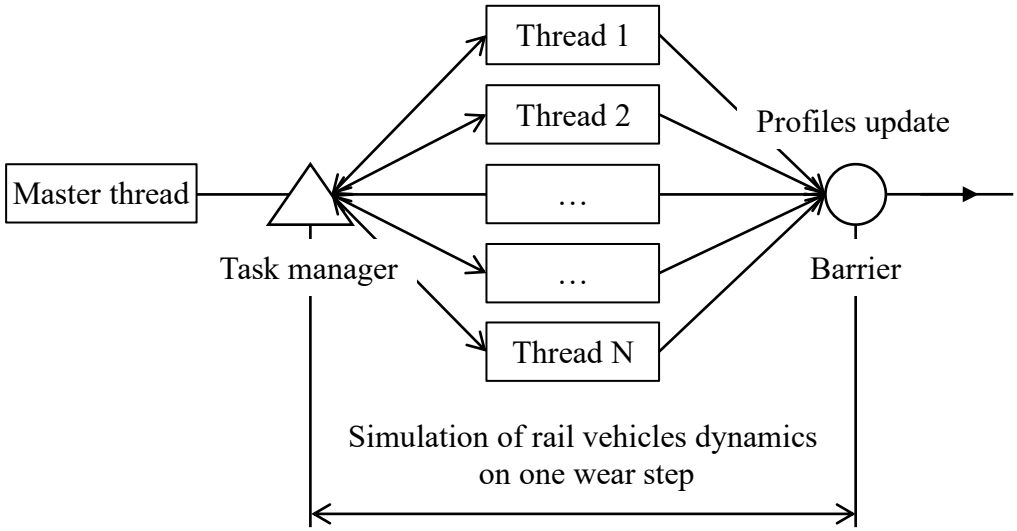

Fig. 2. Fork-join model of parallel computation of wear of railway wheel profiles

4. MULTI-VARIANT PARALLEL COMPUTATIONS

The aforementioned sections are devoted to parallel multithread computing of a numerical experiment within an executing instance of a program that is called a process. UM software also supports another type of parallel computing – parallel running the number of solvers as processes to calculate different models or different configurations of a model. Model configurations differ from each other in parameters (geometrical, inertial, stiffness, damping and other) or experimental conditions like wheel or rail profile, railway track geometry or irregularities and vehicle speed.

Such an approach allows engineer to specify the list of model parameters and experimental conditions, its lower and upper limits and a number of points with the limit for each parameter and then run the automatic parallel computation of the series of numerical experiments. In typical engineering practice, there might be decades and hundreds and even thousands of numerical experiments. In such a case, automatic running a series of numerical experiments saves hours and days of monotonous work.

To compute the series of numerical experiments, a control process runs one instance of solver many times sequentially or several of them simultaneously with feeding them source data for every next numerical experiment. At the same time, every solver may use one or several threads for computations. Thus, we have a combination of the multiprocess and multithread approaches. There are several possible strategies to compute a series of numerical experiments: (1) to run one solver sequentially with allowed maximum of computing threads in it, (2) to run allowed maximum of solvers with the only computing thread in it or (3) to run several solvers with several threads in each. The comparison of the possible strategies mentioned above will be given in Part II of the paper. In UM software, the maximum number of simultaneously running processes (solvers) and threads in one solver is limited by the number of logical processors of the operating system.

UM software includes a special service of distributed calculations named UM Cluster. It allows using all computational power of a local/corporate network for execution of series of numerical

experiments that decreases time efforts correspondingly. This possibility is very easy and effective to use within computer centers and laboratories. Service of distributed calculations is based on using TCP/IP that allows employing any computer not only in local network but also in Intra- and Internet for the needs of a project. It significantly decreases total time for computation of big series of numerical experiments.

The service consists of two parts: server and client ones. The server part works on the head computer and controls the execution – sends jobs and receives results back. The client part is run on the peripheral computers, gets and computes jobs and sends results to the server.

Remote install and uninstall of the client parts on client computers are supported. Remote installation of the software components on the client computer runs automatically without disturbing remote users. There is a built-in tool to determine all available computers in a local network that simplify initial procedures of searching and adding client computers to the computational cluster.

5. CO-SIMULATION

Computer simulation in railway research sometimes requires a combination of purely multibody models that Universal Mechanism deals with and associated models of control systems, power electrical machines, hydraulic and pneumatic elements etc., see, for instance, the article by Wang et al [45]. Such complex systems cannot be described with the help of UM built-in elements and tools and should be simulated with the help of special-purpose codes. UM software supports several such interfaces: interface with Matlab/Simulink [46] and SimInTech [47], interface with dynamic-linked library (DLL) developed by user with the help of any programming language.

Users can then simulate their systems' full-motion behaviour from within the Simulink or SimInTech environment or include their models into the UM environment and visualize the results using animations and plotting.

UM software supports two different co-simulation techniques. The first technique supposes the exporting the Simulink or SimInTech models as a DLL with subsequent loading the DLL within the model context and setting connections between a mechanical part and the model exported as the DLL. Such an approach also allows a user to develop his/her own dynamic-linked library using any program environment that supports generating DLLs. In this sense, it is a universal way to plug in user's code and algorithms into a UM model.

The second technique supposes exporting a UM model from UM for posterior integration into a Simulink or SimInTech model. To import UM models into Simulink, the standard S-Function element is used. S-Function element includes the code that is automatically generated by UM and uses UM as a COM server to send and get signals. To import UM models into SimInTech model, the special 'UM model' block is used.

Universal Mechanism is also distributed as a Component Object Model (COM) library for using in third-party applications including Simulink. Component Object Model (COM) is a binary-interface standard for software components introduced by Microsoft in 1993. It is used to enable inter-process communication object creation in a large range of programming languages [48]. COM interfaces implemented in UM allows a user to load UM models, change their parameters and simulate dynamics of the model in a co-simulation mode with getting kinematical data of the mechanical model for external visualization and sending control signals. Typical way of usage of UM COM interfaces is given in Fig. 3.

Universal Mechanism as a COM server for dynamical analysis was used in a number of projects including two train driving real-time simulators [24, 49].

6. CONCLUSION

A detailed model of parallelization in simulation of multibody systems containing both rigid and flexible bodies is described in the paper. The model includes equations of motion, with the main

feature consists in replacement of rigid joints by compliant ones; formulation of PCGM with a very efficient preconditioning; description of parallel sections for the fork-join parallelism; and the algorithm for improvement of thread balancing in parallel sections, which is important when the simulated model includes flexible bodies along with the rigid ones. As it will be shown in Part II of this publication, the method allows a considerable speed up of simulation on multicore processors, both relatively simple models of rail vehicles with about one hundred DOF and much more complex models with dozens of thousands of DOF. The algorithm efficiency is based on the use of shared memory parallelism, on parallelization of all stages of computations from the prediction up to the solution of linear equations, and on the good balancing of threads within all of the parallel sections.

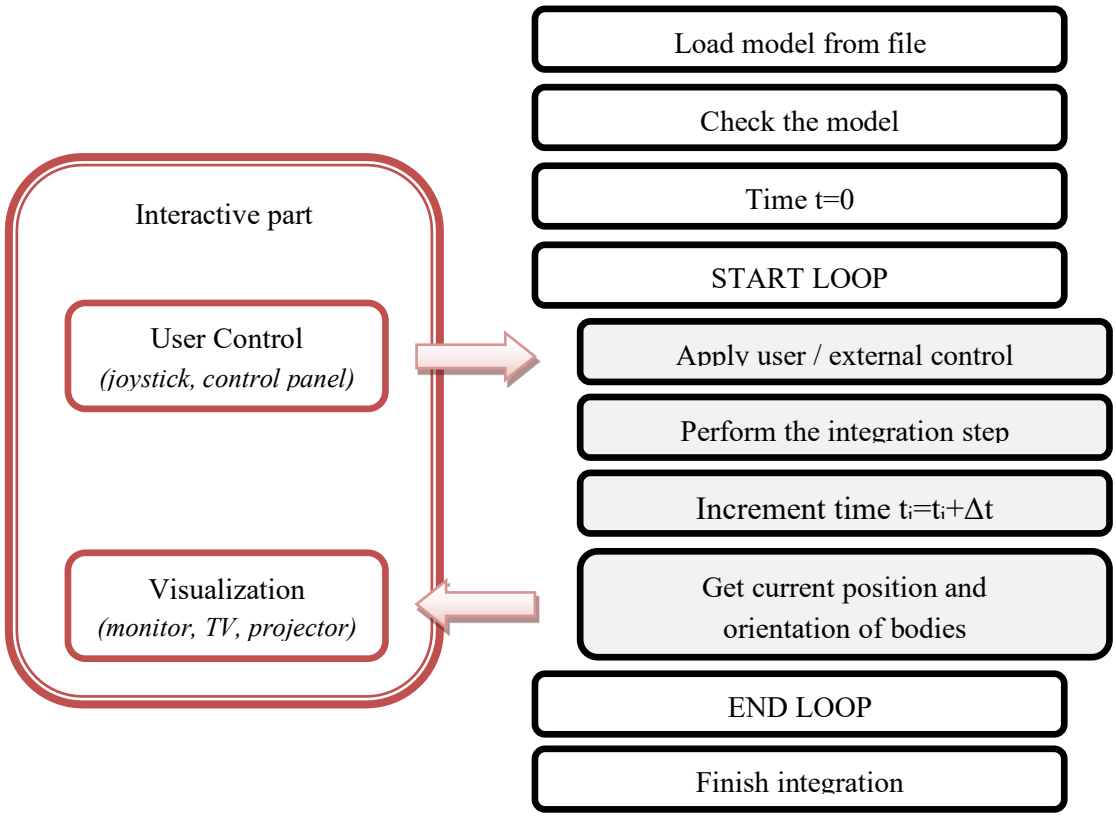

Fig. 3. Using UM COM server in the third-party application

The implementation of the fork-joint model for the parallel approach to predicting wear of railway wheel profiles is described in the paper. The efficiency of this algorithm will be shown in Part II.

Parallel multi-variant computations as well as UM interfaces with Matlab/Simulink and SimInTech are briefly discussed.

ACKNOWLEDGMENTS

The research was supported by the Russian Foundation for Basic Research under grant 17-01-00815a.

References

1. Wu, Q. & Spiryagin, M. & Cole C. & et al. Parallel computing in railway research. *International Journal of Rail Transportation*. 2018.
2. Featherstone, R. A divide-and-conquer articulated body algorithm for parallel O (log (n)) calculation of rigid body dynamics. Part 1: Basic algorithm. *The International Journal of Robotics Research.*1999. Vol. 18. No. 9. P. 867-875.
3. Featherstone, R. A divide-and-conquer articulated body algorithm for parallel O (log (n)) calculation of rigid body dynamics. Part 2: Trees, loops, and accuracy. *The International Journal* of Robotics Research. 1999. Vol. 18. No. 9. P. 876-892.
4. Fijany, A. & Sharf, I. & D'Eleuterio, G.M.T. Parallel O (log N) algorithms for computation of manipulator forward dynamics. *IEEE Transactions on Robotics and Automation*. 1995. Vol. 11. No. 3. P. 389-400.
5. Critchley, J.H. & Anderson, K.S. & Binani, A. An efficient multibody divide and conquer algorithm and implementation. *Journal of Computational and Nonlinear Dynamics*. 2009. Vol. 4. No. 2. 021001.
6. Fijany, A. & Featherstone, R. A new factorization of the mass matrix for optimal serial and parallel calculation of multibody dynamics. *Multibody System Dynamics*. 2013. Vol. 29. No. 2. P. 169-187.
7. Khan, I.M. & Anderson, K.S. A logarithmic complexity divide-and-conquer algorithm for multi-flexible-body dynamics including large deformations. *Multibody System Dynamics*. 2015. Vol. 34. No.1. P. 81-101.
8. Chadaj, K. & Malczyk, P. & Fraczek, J. A parallel Hamiltonian formulation for forward dynamics of closed-loop multibody systems. *Multibody System Dynamics*. 2017. Vol. 39. No. 1-2. P. 51-77.
9. Liu, F. & Zhang, J. & Hu, Q. A modified constraint force algorithm for flexible multibody dynamics with loop constraints. Near Dynamics. 2017. Vol. 90. No. 3. P. 1885-1906.
10. Malczyk, P. & Fraczek, J & González, F. & et al. Index-3 divide-and-conquer algorithm for efficient multibody system dynamics simulations: theory and parallel implementation. *Nonlinear Dynamics*. 2019. Vol. 95. No. 1. P. 727-747.
11. Iglberger, K. & Rüde, U. Massively parallel rigid body dynamics simulations. *Computer Science-Research and Development*. 2009. Vol. 23. No. 3-4. P. 159-167.
12. Negrut, D. & Serban, R. & Mazhar, H. & et al. Parallel computing in multibody system dynamics: why, when and how. *Journal of Computational and Nonlinear Dynamics*. 2014. Vol. 9. No. 4. 041007.
13. Khatibi, F. & Esmaeili, M. & Mohammadzadeh, S. DEM analysis of railway track lateral resistance. *Soils and Foundations*. 2017. Vol. 57. No. 4. P. 587-602.
14. Tutumluer, E. & Qian, Y. & Hashash, Y.M.A. & et al. Discrete element modelling of ballasted track deformation behavior. *International Journal of Rail Transportation*. 2013. Vol. 1. No. 1-2. P. 57-73.
15. Fleissner, F. & Lehnart, A. & Eberhard, P. Dynamic simulation of sloshing fluid and granular cargo in transport vehicles. *Vehicle system dynamics*. 2010. Vol. 48. No. 1. P. 3-15.
16. Fisette, P. & Peterkenne, J.M. Contribution to parallel and vector computation in multibody dynamics. *Parallel Computing*. 1998. Vol. 24. P. 717-728.
17. Postiau, T. & Sass, L. & Fisette, P. & et al. High-performance multibody models of road vehicles: Fully symbolic implementation and parallel computation. 20th International Conference of Theoretical and Applied Mechanics. Chicago, 2000. In: *Vehicle System Dynamics: International Journal of Vehicle Mechanics and Mobility.* 2001. Vol. 35. P. 57-83.
18. Anderson, K.S. & Duan, S. Highly parallelizable low order dynamics simulation algorithm for multi-rigid-body systems. *Journal of Guidance. Control and Dynamics*. 2000. Vol. 23. No. 2. P. 355-364.
19. Гетманский, В.В. & Горобцов, А.С. Решение задач большой размерности в системах моделирования многотельной динамики с использованием параллельных вычислений.

Известия Волгоградского государственного технического университета. 2007. No. 9(35). P. 10-12. [In Russian: Getmansky, V.V. & Gorobtsov, A.S. Large-scale tasks solving in multibody dynamics simulation systems using parallel computing. *Proceedings of Volgograd State Technical University*].
20. Orlandea, N. & Chace, M.A. & Calahan, D.A. A sparsity oriented approach to the dynamic analysis and design of mechanical systems – Part I. *Journal of Engineering for Industry*. 1977. Vol. 99, No. 3. P. 773-784.
21. Haug, E.J. *Computer-Aided Kinematics and Dynamics of Mechanical Systems Volume-I*. Boston: Allyn and Bacon. 1989. 511 p.
22. Valasek, M. & Mraz, L. Parallelization of multibody system dynamics by heterogeneous multiscale method. In: *Proceedings of Multibody Dynamics 2011, ECCOMAS Thematic Conference*. Brussels. 2011.
23. Mraz, L. & Valasek, M. Solution of three key problems for massive parallelization of multibody dynamics. *Multibody System Dynamics*. 2013. Vol. 29. No. 1. P. 21-39.
24. Pogorelov, D. & Yazykov, V. & Lysikov, N. & et al. Train 3D: the technique for inclusion of three dimensional models in longitudinal train dynamics and its application in derailment studies and train simulators. *Vehicle System Dynamics*. 2017. Vol. 55. No. 4. P. 583-600.
25. Universal Mechanism. Bryansk: Laboratory of Computational Mechanics. Available at: https://www.universalmechanism.com/.
26. Pogorelov, D.Y. Jacobian matrices of the motion equations of a system of bodies. *Journal of Computer and Systems Sciences International*. 2007. Vol. 46. No. 4. P. 563-577.
27. Pogorelov, D.Y. Simulation of constraints by compliant joints. *Journal of Computer and Systems Sciences International*. 2011. Vol. 50. No. 1. P. 158-173.
28. Shewchuk, J.R. *An introduction to the conjugate gradient method without the agonizing pain*. Pittsburgh: Carnegie-Mellon University. 1994. 58 p.
29. Eberhard, P. *Kontaktuntersuchungen durch hybride Mehrkörpersystem/Finite Elemente Simulationen*. [In German: *Contact investigations by hybrid multibody system / finite element simulations*]. Aachen: Shaker Verlag; 2000. 318 p.
30. Saad Y. *Iterative methods for sparse linear systems*. 2nd ed. Philadelphia: SIAM. 2003. 528 p.
31. Kurdila, A. & Menon R.G. & Sunkel, J.W. Nonrecursive Order N formulation of multibody dynamics. *Journal of Guidance Control and Dynamics*. 1993. Vol. 16. No. 5. P. 838-844.
32. Sharf, I. & D'Eleuterio, G.M.T. An Iterative Approach to Multibody Simulation Dynamics Suitable for Parallel Implementation. *Journal of Dynamic Systems, Measurement, and Control*. 1993. Vol. 115. No. 4, P. 730-735.
33. Piotrowski, J. & Kik, W. A simplified model of wheel/rail contact mechanics for non-Hertzian problems and its application in rail vehicle dynamic simulations. *Vehicle System Dynamics*. 2008. Vol. 46. No. 1-2. P. 27-48.
34. Zobory, I. Prediction of Wheel/Rail Profile Wear. *Vehicle System Dynamics*. 1997. Vol. 28. No. 2-3. P. 221-259.
35. Goryacheva, I.G. & Zakharov, S.M. & Soshenkov, S.N. & et. al. Tribodynamic simulation of wheel/rail profile evolution and contact-fatigue damage accumulation for some variable track and vehicle parameters. *Vniizht Bulletin (Railway Research Institute Bulletin)*. 2011. No. 1. P. 13-19.
36. Von Dist, K. & Ferrarotti, G. & Kik, W. & et al. Wear analysis of the high-speed-grinding Vehicle HSG-2: validation, simulation and comparison with measurements. In: *25th International Symposium on Dynamics of Vehicles on Roads and Tracks*. Rockhampton, 2017.
37. Auciello, J. & Ignesti, M. & Marini, L. & et al. Development of a model for the analysis of wheel wear in railway vehicles. *Meccanica*. 2013. Vol. 48. No. 3. P. 681-697.
38. Craig, R. & Bampton, M. Coupling of substructures for dynamic analyses. *AIAA Journal*. 1968. Vol. 6. No. 7. P. 1313-1319.
39. Simeon, B. DAEs and PDEs in elastic multibody systems. *Numerical Algorithms*. 1998. Vol. 19. P. 235-246.
40. Hairer, E, Wanner, G. *Solving ordinary differential equations II. Stiff and differential-algebraic problems*. Berlin: Springer. 1996. 614 p.

41. Park, K.C. An improved stiff stable method for direct integration of nonlinear structural dynamic equations. *Journal of Applied Mechanics*. 1975. Vol. 42. No. 2. P. 464-470.
42. Nyman, L. & Laakso, M. Notes on the History of Fork and Join. *IEEE Annals of the History of Computing*. 2016. Vol. 38. No. 3. P. 84-87.
43. Archard, J.F. Contact and Rubbing of Flat Surface. *Journal of Applied Physics*. 1953. Vol. 24. No. 2. P. 981-988.
44. Vollebregt, E.A.H. *User guide for CONTACT, rolling and sliding contact with friction (Technical Report TR09-03)*. Delft: VORtech CMCC. 2018. 141 p.
45. Wang, H. & Deng, Z. & Ma, S. & Sun, R. & Li, H. & Li, J. Dynamic Simulation of the HTS Maglev Vehicle-Bridge Coupled System Based on Levitation Force Experiment. *IEEE Transactions on Applied Superconductivity*. 2019. Vol. 29. No. 5. P. 1-6. Art no. 3601606. DOI: 10.1109/TASC.2019.2895503.
46. *Matlab/Simulink*. Available at: https://www.mathworks.com.
47. *SimInTech*. Available at: simintech.ru.
48. *Wikipedia: Component Object Model*. Available at: https://en.wikipedia.org/wiki/Component_Object_Model.
49. Öztürk, V. & Arar, Ö.F. & Rende, F.Ş. & et al. Validation of railway vehicle dynamic models in training simulators. *Vehicle System Dynamics*. 2017. Vol. 55. No. 1. P. 41-71.

Received 10.05.2018; accepted in revised form 05.09.2019