Václav Rek

Explicit integration of equations of motion solved on computer cluster

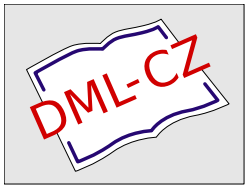
In: Jan Chleboun and Pavel Kůs and Petr Přikryl and Miroslav Rozložník and Karel Segeth and Jakub Šístek and Tomáš Vejchodský (eds.): Programs and Algorithms of Numerical Mathematics, Proceedings of Seminar. Hejnice, June 24-29, 2018. Institute of Mathematics CAS, Prague, 2019. pp. 119–131.

Persistent URL: http://dml.cz/dmlcz/703085

**Terms of use:**

© Institute of Mathematics CAS, 2019

Institute of Mathematics of the Czech Academy of Sciences provides access to digitized documents strictly for personal use. Each copy of any part of this document must contain these *Terms of use*.




This document has been digitized, optimized for electronic delivery and stamped with digital signature within the project *DML-CZ: The Czech Digital Mathematics Library* http://dml.cz

*Programs and Algorithms of Numerical Mathematics 19*
J. Chleboun, P. Kůs, P. Přikryl, M. Rozložník, K. Segeth, J. Šístek, T. Vejchodský (Eds.)
Institute of Mathematics CAS, Prague 2019

# EXPLICIT INTEGRATION OF EQUATIONS OF MOTION SOLVED ON COMPUTER CLUSTER

Václav Rek

Brno University of Technology, Faculty of Civil Engineering,
Institute of Structural Mechanics,
602 00 Brno, Veveří 331/95, Czech Republic
rek@fem.cz

**Abstract:** In the last decade the dramatic onset of multicore and multiprocessor systems in combination with the possibilities which now provide modern computer networks have risen. The complexity and size of the investigated models are constantly increasing due to the high computational complexity of computational tasks in dynamics and statics of structures, mainly because of the nonlinear character of the solved models. Any possibility to speed up such calculation procedures is more than desirable. This is a relatively new branch of science, therefore specific algorithms and parallel implementation are still in the stage of research and development which is attributed to the latest advances in computer hardware, which is growing rapidly. More questions are raised on how best to utilize the available computing power. The proposed parallel model is based on the explicit form of the finite element method, which naturaly provides the possibility of efficient parallelization. The possibilities of multicore processors, as well as parallel hybrid model combining both the possibilities of multicore processors, and the form of the parallelism in a computer network are investigated. The designed approaches are then examined in addressing of the numerical analysis regarding contact/impact phenomena of shell structures.

**Keywords:** explicit form of finite element method, dynamics of structures, parallel computing

**MSC:** 68U20, 74H15, 35L53

## 1. Introduction

Explicit algorithms are highly suitable for a solution of short time highly nonlinear computations mainly for numerical simulation of the processes of forming casts or the simulation of crash tests in the automotive and aviation industry or for form finding of thin membranes in civil engineering, etc. This method facilitates the consideration of a variety of nonlinearities in an easy and explicit manner.

DOI: 10.21136/panm.2018.13

At this time, the technologies of multi-core CPU processors, programmable graphical multi-core GPU processors, as well as their combinations, are commonly available on the market. In view of the affordability of solid computer assemblies, an interconnection of aforementioned parallel machines (CPU+GPU) in a generally heterogeneous computer cluster through the computer network is another level of parallelism. The proposed approach is related to a somewhat different view of distribution of numerical computations in nonlinear dynamics of structures, which explicit integration provides.

A different approach is meant here in comparison to those commonly applied approaches in the field of the parallelization of explicit numerical computations primarily focused on powerful single workstations as for example in the open-source finite element toolkit NiftySim [7] or in the open-source SPH (abbreviation of the Smoothed Particle Hydrodynamics) toolkit DualSPHysics [3]. Most of the approaches deal with the network type of computations related to the FETI method (abbreviation of the Finite Element Tearing and Interconnecting) [5], e.g. in the open-source finite element toolkit SIFEL [9]. The FETI method is well suited for the implicit integration of the equations of motion. In addition, it is also strongly dependent on a robust algorithm providing decomposition of the finite element mesh to particular subdomains. It applies especially to the methods of the Newmark family and other methods frequently used in civil engineering practice [19].

## 2. Mathematical-physical model

Consider a body $\mathscr{B}$ in a three-dimensional Euclidian space $\mathbb{R}^3$, which is composed of an infinite number of material elements. Under the influence of external forces, the body $\mathscr{B}$ will undergo macroscopic geometric changes. If the applied loads are time dependent, the deformation and geometry of the body $\mathscr{B}$ will be a function of time. A material body $\mathscr{B}$ in motion starts from the so-called *initial configuration* $\mathscr{B}_{t_0}$ at time $t_0$. As time proceeds with the application of external forces, the body will occupy a different region $\mathscr{B}_t$ at time $t$, which is called the *current configuration*. A particle of the body $\mathscr{B}$ in the initial configuration $\mathscr{B}_{t_0}$ occupies a position $\mathbf{X}$, which is referred to a reference Cartesian frame.

The deformed configuration is characterized by the mapping, which represents the bijective function $\phi_{\mathscr{B}}\colon \mathscr{B}_{t_0} \to \mathscr{B}_t$. Mapping $\phi_{\mathscr{B}}$ takes the position vector $\mathbf{X}$ from the reference configuration $\mathscr{B}_{t_0}$ and places the same point in the deformed configuration $\mathbf{x} = \mathbf{X} + \mathbf{u}$. Motion of a continuous medium is also denoted by deformation, which is characterized by the rigid body motion, where the original shape of the body after the motion preserves the distance between particles, and by the motion with deformation, which is characterized by changes of distance between particles.

Now let's consider the *Principle of Balance of Linear Momentum*, which is based on the Newton's second law of motion. This law expresses the meaning that the rate of change of linear momentum $\mathbf{p}$ of an arbitrary part of a continuum body $\mathscr{B}_t$

is proportional to the sum of volume $\mathbf{b}$ and surface forces

$$\frac{\mathrm{D}}{\mathrm{D}t}\left[\int_{\mathscr{B}_t}\mathbf{p}\,\mathrm{d}v\right]=\frac{\mathrm{D}}{\mathrm{D}t}\left[\int_{\mathscr{B}_t}\varrho\mathbf{v}\,\mathrm{d}v\right]=\int_{\partial\mathscr{B}_\sigma}\mathbf{t}\,\mathrm{d}a+\int_{\mathscr{B}_t}\varrho\mathbf{b}\,\mathrm{d}v, \tag{1}$$

where $\mathrm{D}/\mathrm{D}t\,[\bullet]$ denotes the material derivative, $\varrho$ the material density, $\mathbf{v}$ the velocity field and $\mathbf{t}$ the traction forces on boundary $\partial\mathscr{B}_\sigma$. The left part of eq. (1) is expressed by the *Reynolds transport theorem* in sense of a body acceleration $\mathbf{a}$ as $\int_{\mathscr{B}_t}\varrho\mathbf{a}\,\mathrm{d}v$. It then finally leads to the *Cauchy's equations of motion* with boundary and initial conditions expressed in their locally valid strong form as follows:

$$\operatorname{div}\boldsymbol{\sigma}^T+\varrho\mathbf{b}=\varrho\mathbf{a}\text{ in }\mathscr{B}\times[0,\tau], \tag{2}$$

$$\mathbf{u}(\mathbf{x},t)=\overline{\mathbf{u}}\text{ on }\partial\mathscr{B}_D\times[0,\tau],\quad\mathbf{t}=\boldsymbol{\sigma}\cdot\hat{\mathbf{n}}\text{ on }\partial\mathscr{B}_\sigma\times[0,\tau], \tag{3}$$

$$\mathbf{v}(t_0)=\overline{\mathbf{v}}_0\text{ in }\mathscr{B},\quad\mathbf{u}(t_0)=\overline{\mathbf{u}}_0\text{ in }\mathscr{B}. \tag{4}$$

Connections between kinematic, thermal and mechanical variables is established by the constitutive equations, which are here considered as locally valid standard material model defined by the differentiable potential, which is here reduced to specifying *elastic energy density* $\Psi$ called Helmholtz's free energy. Consider strain-like variable $\varepsilon\in X_\varepsilon$, where $X_\varepsilon$ is a real Banach space and a stress-like variable $\sigma\in X_\sigma^*$, both are subset of a linear space $X_\sigma^*\times X_\varepsilon$, where $X_\sigma^*$ is the continuous dual space. Then we can write $\boldsymbol{\sigma}\equiv\delta\Psi(\boldsymbol{\varepsilon})$, where $\boldsymbol{\sigma}$ and $\boldsymbol{\varepsilon}$ denote Cauchy's stresses and elastic strains, respectively. The Helmholtz's free energy has the form

$$\Psi(\boldsymbol{\varepsilon})=\frac{\lambda}{2}\left(\operatorname{tr}\boldsymbol{\varepsilon}\right)^2+\mu\boldsymbol{\varepsilon}:\boldsymbol{\varepsilon},\quad\boldsymbol{\sigma}\equiv\varrho\partial_{\boldsymbol{\varepsilon}}\Psi(\boldsymbol{\varepsilon})=\mathbb{C}_{ijkl}:\boldsymbol{\varepsilon}, \tag{5}$$

where $\lambda$ and $\mu$ are Lamé constants, respectively, and $\mathbb{C}_{ijkl}\equiv\lambda\delta_{ij}\delta_{kl}+\mu\left(\delta_{ik}\delta_{jl}+\delta_{il}\delta_{jk}\right)$ denotes fourth order constitutive tensor. It can be viewed as a linearized form of the Saint Venant-Kirchhoff material model.

The critical part of the developed algorithm is the consideration of nonlinear boundary conditions represented by their change caused by the contact of surface portion $\partial\mathscr{B}_C$ of the body $\mathscr{B}$ with the neighboring barrier or with the other body and optionally also by contact of body $\mathscr{B}$ with itself. The contact causes the formation of additional forces, which at the time of their creation are suitably applied into the process of numerical integration of the equations of motion. Here it leads to the conditions providing the basis to treat frictionless contact problems in the context of constraint optimization known as Hertz–Signorini–Moreau or Kuhn–Tucker–Karush condition expressed for flat shells [20]

$$g_{N,C}\geq 0,\quad p_N\leq 0,\quad p_N g_{N,C}\leq 0, \tag{6}$$

where $p_N$ is the associated normal component of the stress traction vector $\mathbf{t}=\boldsymbol{\sigma}\cdot\hat{\mathbf{n}}$ in the current configuration, $g_{N,C}$ denotes the normal gap and $h$ denotes the shell thickness, thus

$$g_{N,C}=\begin{cases}\left|(\mathbf{x}_2-\hat{\mathbf{x}}_1)\cdot\hat{\mathbf{n}}_1-\frac{h}{2}\right| & \text{if }\ (\mathbf{x}_2-\hat{\mathbf{x}}_1)\cdot\hat{\mathbf{n}}_1-\frac{h}{2}<0,\\ 0 & \text{otherwise.}\end{cases} \tag{7}$$

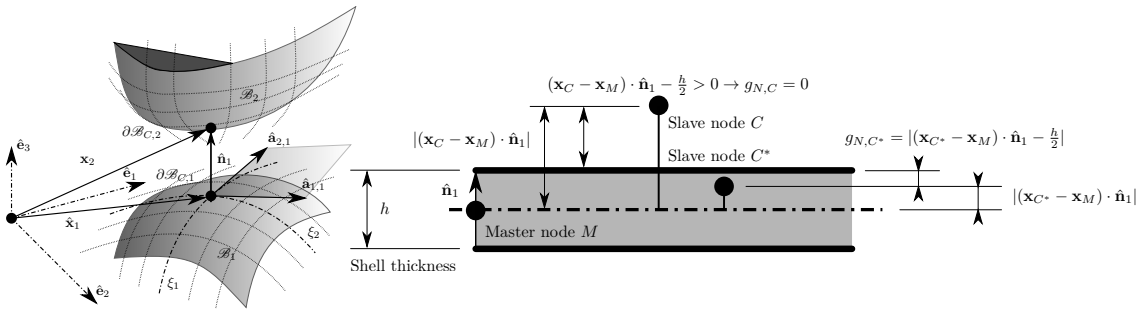



Figure 1: Determination of penetration normal gap $g_{N,C}$ for flat shell finite element

The most commonly used constitutive equation simply representing the complex micromechanical behaviour on the contact surface leads to a standard penalty method $p_N = -\epsilon_N g_N$, where $\epsilon_N$ is a penalty parameter. This approach is considered here.

## 3. Model numerics

For obtaining a semi-discrete form of the equations of motion, the tools of variational calculus are used. These involve finding a local minima of some relevant energy functional $J(\mathbf{q}(t))$, thus $\delta J(\mathbf{q}(t), t) \equiv \mathcal{G}J(\mathbf{q}(t), t)(\boldsymbol{\eta}(t)) = 0$, where $\mathcal{G}J(\mathbf{q}(t), t)(\boldsymbol{\eta}(t))$ denotes Gâteaux derivative of the functional in direction of function $\boldsymbol{\eta}(t) \in X$. It finally leads to the form of Euler-Lagrange equations

$$\delta J(\mathbf{q}(t), t) \equiv \partial_{\mathbf{q}} \mathcal{L}(\mathbf{q}(t), \dot{\mathbf{q}}(t), t) - \frac{\mathrm{d}}{\mathrm{d}t} \left( \partial_{\dot{\mathbf{q}}} \mathcal{L}(\mathbf{q}(t), \dot{\mathbf{q}}(t), t) \right), \tag{8}$$

where $\mathcal{L}(\mathbf{q}(t), \dot{\mathbf{q}}(t), t) \in C^2[a, b]$ is the associated Lagrangian. The particular approach used in this relation is the Hamilton's variational principle in elastodynamics deduced from an analogy to the Lagrangian form of D'Alembert's principle. It then leads to the expressions

$$\mathcal{L} = \mathcal{K} - (\mathcal{U} + \mathcal{U}_c), \tag{9}$$

$$\delta \int_{\tau - \Delta\tau}^{\tau} \mathcal{L} \, \mathrm{d}\tau + \delta \int_{\tau - \Delta\tau}^{\tau} \mathcal{W} \, \mathrm{d}\tau = 0, \quad \delta\mathbf{u}(\tau - \Delta\tau) = \delta\mathbf{u}(\tau) = 0, \tag{10}$$

where functional $\mathcal{K} \colon \mathcal{K} \rightarrow \mathbb{R}$ represents *kinetic energy* and functionals $\mathcal{U} \colon \mathcal{U} \rightarrow \mathbb{R}$ and $\mathcal{W} \colon \mathcal{W} \rightarrow \mathbb{R}$ represent energy of internal (*deformative energy*) and external forces, respectively, and $\mathcal{U}_c \colon \mathcal{U}_c \rightarrow \mathbb{R}$ represents energy of constraint contact forces. A particular composition of the respective energy functionals can be found in e.g. [6], [15] or [20]. The resulting set of nonlinear ODEs of the second order is then used for an explicit time integration. For the approximation of the field variables and respective geometry of the body $\mathscr{B}$, the finite element method is used. Thus $\mathscr{B} \approx \Omega = \bigcup_{e=1}^{n_e} \Omega_e$, where $\Omega$ denotes discretized body $\mathscr{B}$ by $n_e$ finite element non-overlapping subregions $\Omega_e$. The boundary of the region $\partial\Omega$ is composed of the curves or areas $\partial\Omega_e$ of

the elements $\Omega_e$: $\partial\Omega = \bigcup_{e=1}^{n_r} \partial\Omega_e$, which generally approximate the real geometry of the boundary $\partial\mathscr{B}$ without an overlap of finite elements. An effective triangular flat shell finite element developed by T. Belytschko et al. [1] is applied as a suitable type of finite element for explicit computations [18]. The resulting form of semidiscrete set of nonlinear ordinary differential equations

$$\mathbb{M}\ddot{\mathbf{u}} + \mathbf{f}_{\text{int}}(\mathbf{u}) = \mathbf{f}_{\text{ext}}(\mathbf{u}) \in \Omega, \tag{11}$$

where $\mathbb{M}$, $\mathbf{f}_{\text{int}}(\mathbf{u})$ and $\mathbf{f}_{\text{ext}}(\mathbf{u})$ represent the mass matrix, internal and external forces, respectively. Those are then used for subsequent explicit integration. The particular explicit integration algorithm is applied according to the half time step form mentioned in [2]. In an explicit approach, the mass matrix $\mathbb{M}$ is considered in its diagonalized form (i.e. the lumped mass matrix).

The nonlinear contact conditions are involved in the developed parallel computing model for the modeling of the contact/impact phenomena with geometrically nonlinear behavior of a solid continua represented by a large rotational kinematics solved in the small strain regime [18].

## 4. Hybrid-parallel computing model

The hybrid-parallel computing model is based on a physically logical decomposition according to the specific nature of addressed impact problems. Thus the general model of contact is here the source of the domain decomposition algorithm. This approach is applicable primarily to the simulation of the impact tasks of a wider group of separate entities (bodies) interacting with each other through contact forces. This of course also applies to the so-called *self-contact* problem. In this sense of generality it is the most computationally demanding process. The algorithm used must be applied to all the finite element nodes of the finite element model and also to all discretized separate domains $\Omega_i$ (macro entities).

The solution to the problem of contact of solids then theoretically falls within the area of a so-called *nearest neighbor* (NN) search as it is suggested in [20]. The *kd*-tree data structure falls into the set of data structures providing desired properties. This type of data structure represents a special case of binary trees. The detailed mathematical study of the related algorithm provides [10].

The core of such an algorithm is defined here as a collection of $n$ objects (nodes of finite element mesh) that build a data structure, which provides those objects in the time as fast as possible based on the NN query represented by the bounding box encapsulating respective finite element. Appropriate searches in the tree can then be processed in parallel on the set of current finite elements mapped to the available cores of the multicore CPU in the pre-processing phase. It does not require any synchronization procedures causing slowing down the performance by sequential execution of this part of the code. This algorithm belongs to the cornerstones of the entire parallel model.

The rest of the code related to parallel processing on a single workstation is applied similarly to the algorithm proposed for GPGPU technology [16] for multicore CPU environment [17]. This applies to the integration of respective finite elements and explicit integration of equations of motion applied to all the finite element nodes.

## 5. Analysis of the macro entity interaction multigraph

The proposed algorithm for the purpose of hybrid-parallel computational processing strongly relates to the analysis of the motion of the individual macro elements in a space representing the structures interacting with each other through the contact forces as ilustrated in Fig. 2.

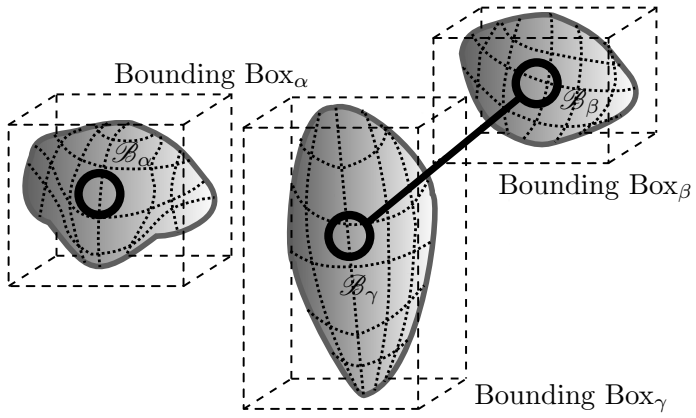



Figure 2: Ilustrative example of the macro entity interaction multigraph (macro body contact interaction)

Nonlinear dynamics is often demonstrated by the chaotic behavior. The source of chaotic behaviour is provided here by mutual interaction of deformable bodies, which generates special types of combinatorial sequences. Such a combinatorial sequence can be defined by so-called *unoriented multi-graph*. The topology of such a multigraph can be defined by the spatial distribution of the interacting bodies. The nodes in the multi-graph represent all the contained bodies (macro entities) and the graph edges represent contact interaction between the bodies.

The MEIM (Macro Entity Interaction Multigraph) assembly is performed through the range searching queries to the kd-tree data structure used for mapping the spatial data. The MEIM analysis itself is further based on the *depth-first search* (DFS) algorithm to find *connected subgraphs* [13] representing individual clusters of touching bodies.

The assembly and analysis of the MEIM is only required while performing a numerical simulation in the scope of a computer network where it is no longer possible to take advantage of the shared memory address space. The algorithm for the data distribution within a computer network starts primarily from the basic DFS analysis of the MEIM. Based on the DFS algorithm, all connected subgraphs are obtained,

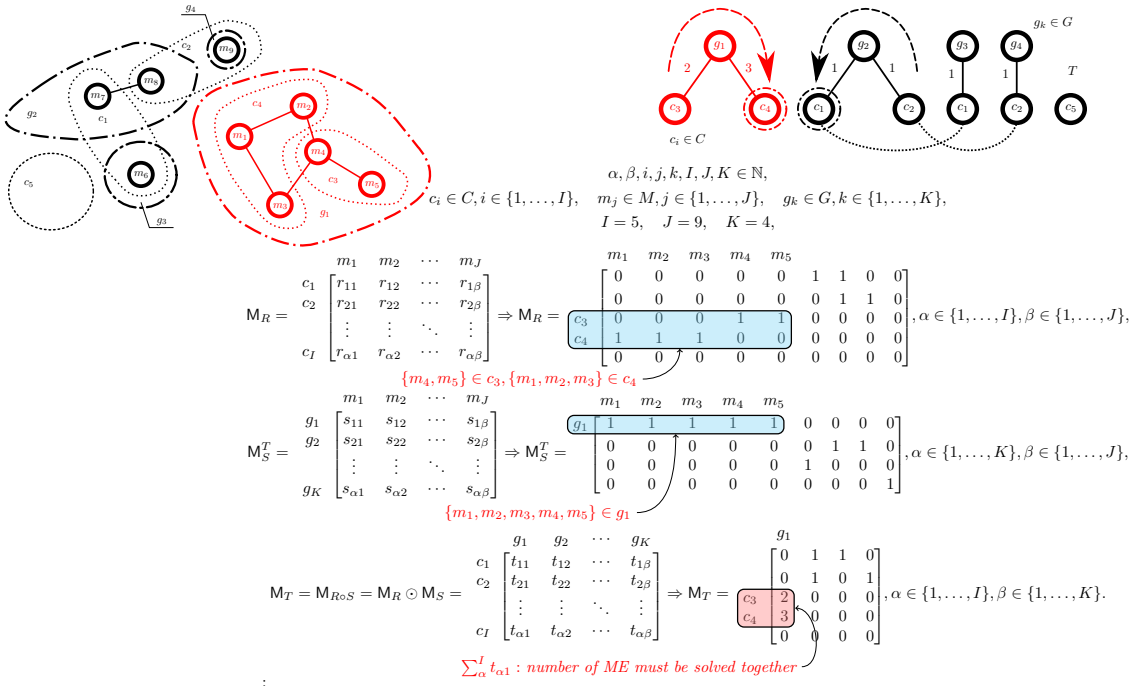



Figure 3: The MEIM model example, where $c_i \in C$ (workstations connected to computer network), $m_j \in M$ (macro entities) and $g_k \in G$ (connected subgraphs)

even if they represent isolated nodes only. The simple example of MEIM is presented in Fig. 3.

As a result of the process of the MEIM analysis, a new state of the data distribution within the computer network is gained. The process of the MEIM analysis is performed at the end of each time step on the side of the central server.

## 6. The new open-source project

Since the implementation of the discussed algorithms and technologies into existing open-source projects mentioned earlier would require considerable time for their own analysis, a hybrid-parallel numerical testing solver called FEXP (abbreviation of the Finite [E]lement [E]XPlicit solver) was composed. The solver has a testing purpose, which means testing the effectiveness of both the algorithms and the technologies used. It combines the possibility of using multi-core processors (CPU) with parallelization in computer heterogeneous cluster interconnected in a LAN (abbreviation of the Local Area Network). In the case of a local workstation within the LAN, it concerns the use of multicore CPUs and the usage of GPGPU technology Nvidia CUDA or OpenCL refering to the use of Graphics Processing Units (GPU) for some parts of the computations. With respect to generality, availability and support of a wide range of software technologies and software modularity, the programming language C++ in current version 14 was chosen as the main programming language for the composition of the FEXP solver.

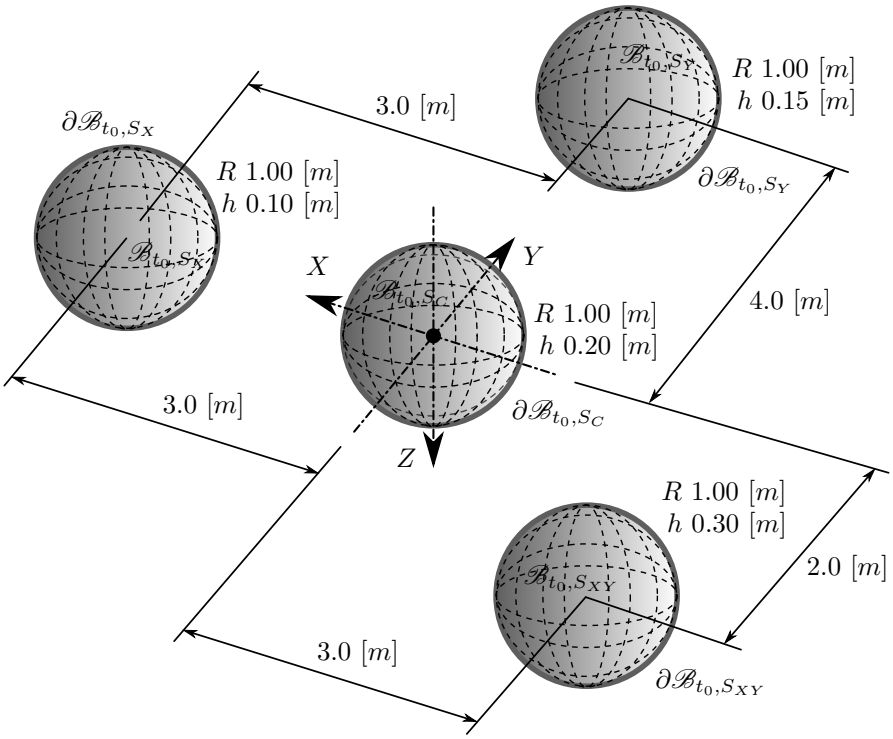



Figure 4: Initial configuration of spheres

As a development environment, the Windows OS was chosen, primarily due to the availability of advanced development tools required to develop such applications. Thus for the project management and code compilation, Microsoft Visual Studio 2015/2017 Community IDE (Integrated Development Environment) was used.

## 7. Simulation test

Verification of the designed solution was performed on a model containing impact of four spheres. This type of example was chosen primarily due to the expected fluctuation of the individual parts of the model within the simulated computer network during the solution process.

The movement of bodies is initiated by the initial conditions represented by the velocity constraints (see Fig. 5), where small velocities in $Z$ direction are introduced primarily due to the applied type of contact detection algorithm which is represented by node-to-element contact. This artificial numerical impurity avoids the state represented by element-to-element contact.

The whole simulation process can be decomposed into three main stages, namely into the initial configuration Fig. 4 and 6, intermediate data fluctuation process due to the mutual contact interaction Fig. 7 and into the final data balance Fig. 8, respectively. During the simulation process another workstation was connected and subsequently it was considered into the simulation process.

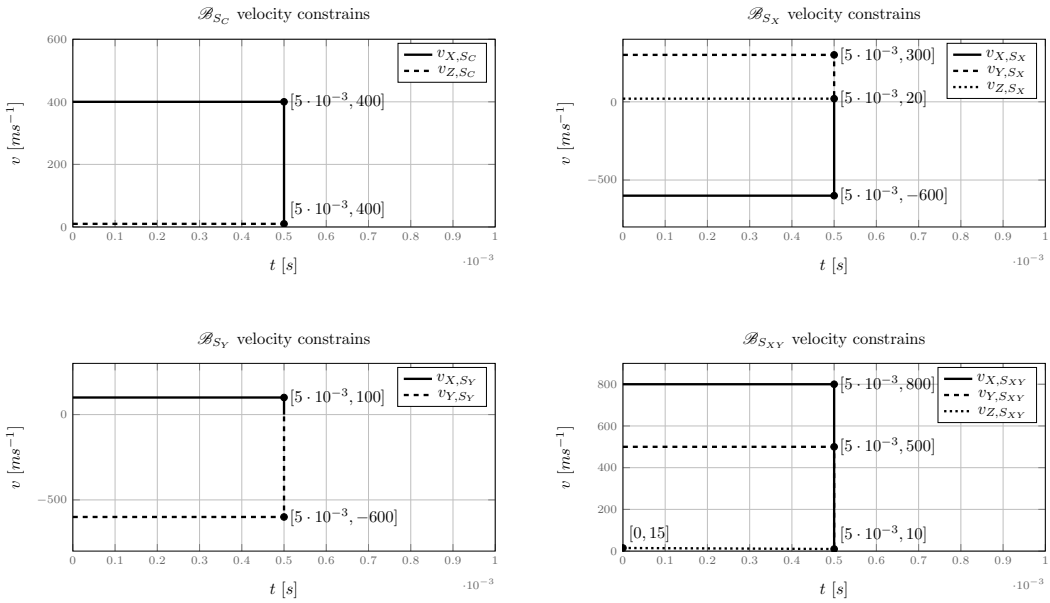



Figure 5: Velocity initial conditions of individual macro entities (spheres)

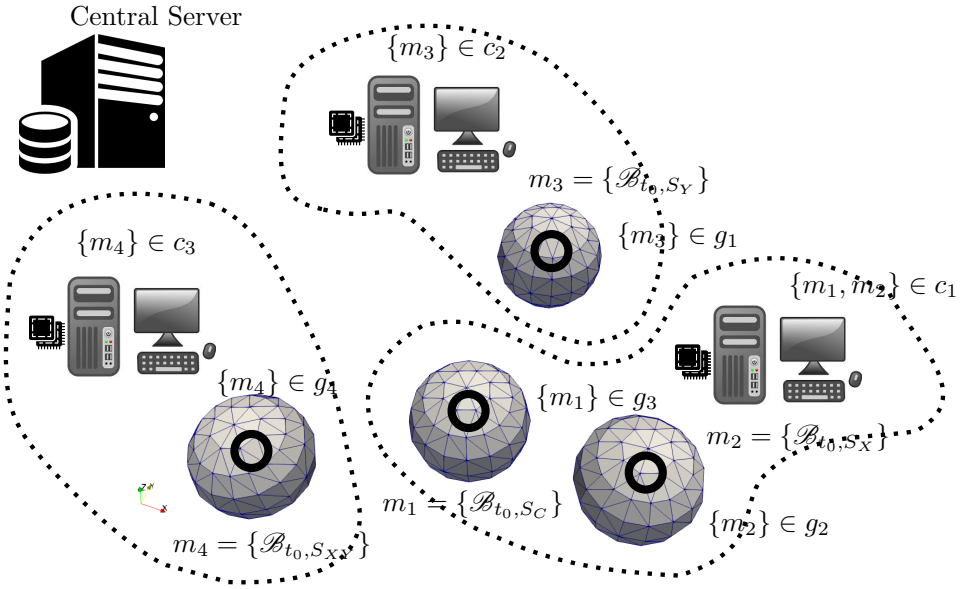



Figure 6: Macro entity distribution over the network at time $t_0$.

The process of numerical computation within the simulated computer cluster can be characterized by Table 1. It shows the amount of data and their distribution over the computer cluster for the aforementioned time frames. The number 1.25 indicates that the data of the entire model were transferred more than once during the computation within the computer cluster.

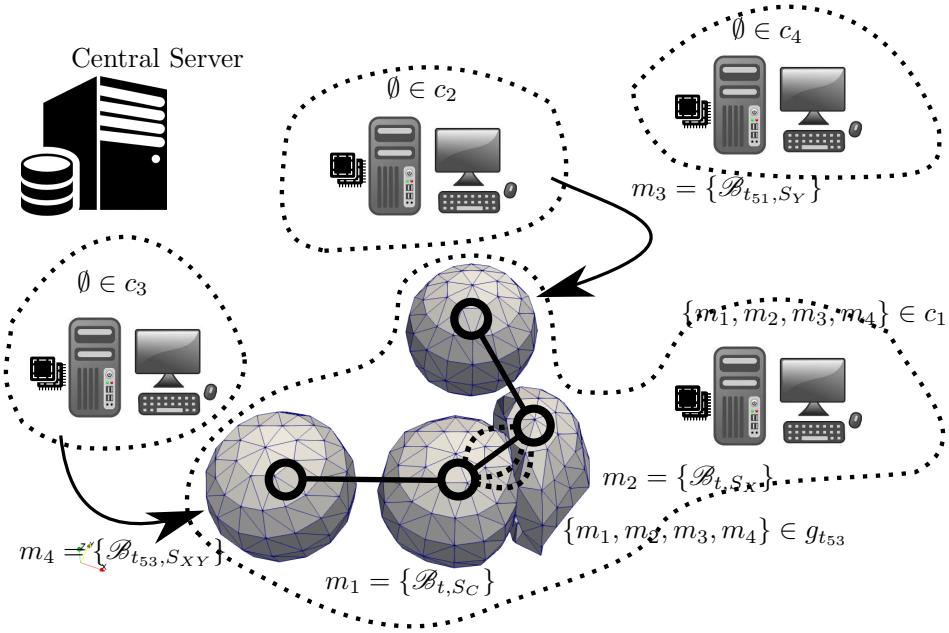



Figure 7: Macro entity distribution over the network at times $t_{51}$ and $t_{53}$.

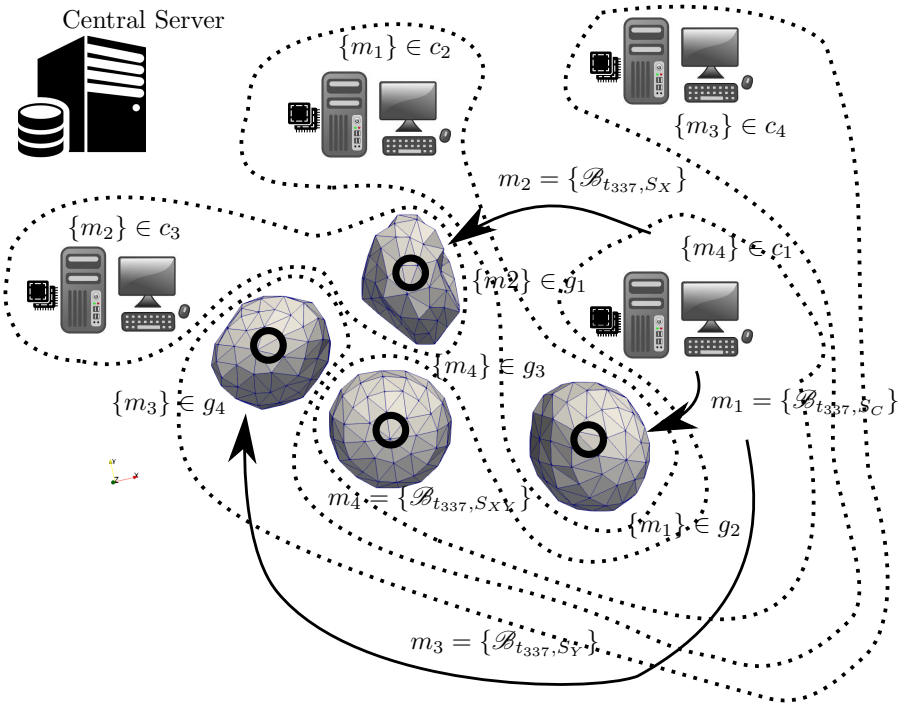



Figure 8: Macro entity distribution over the network at time $t_{337}$.

| Time | Workstation | Elements | Nodes | El./Thread | Nd./Thread |
|---|---|---|---|---|---|
| $t_0$ | $c_1$ | 384 | 196 | 128 | 65.3 |
| | $c_2$ | 192 | 98 | 64 | 32.7 |
| | $c_3$ | 192 | 98 | 64 | 32.7 |
| $t_{51}$ | $c_1$ | 576 | 294 | 192 | 98 |
| | $c_2$ | 0 | 0 | 0 | 0 |
| | $c_3$ | 192 | 98 | 64 | 32.7 |
| | $c_4$ | 0 | 0 | 0 | 0 |
| $t_{53}$ | $c_1$ | 768 | 392 | 256 | 130.7 |
| | $c_2$ | 0 | 0 | 0 | 0 |
| | $c_3$ | 0 | 0 | 0 | 0 |
| | $c_4$ | 0 | 0 | 0 | 0 |
| $t_{337}$ | $c_1$ | 192 | 98 | 64 | 32.7 |
| | $c_2$ | 192 | 98 | 64 | 32.7 |
| | $c_3$ | 192 | 98 | 64 | 32.7 |
| | $c_4$ | 192 | 98 | 64 | 32.7 |
| ∑ Element transfer | **960** | **1.25** | | | |
| ∑ Node transfer | **490** | | | | |

Table 1: The statistics of data transfer.

| Type of parallel element | Volume | Settings |
|---|---|---|
| Computing Workstation | 3/4 | 3–workstation as initial state,<br>$4^{th}$ workstation connected at runtime, |
| Server Workstation | 1 | It start as first, always present. |
| Computing Workstation CPU Thread | 16 | 3–computing threads per one worsktation,<br>1–thread always for communication with server.<br>Number of used threads was determined<br>automatically by number of native threads per CPU. |
| Server CPU Thread | 5/6 | 4–for communication with client workstations,<br>1–main loop thread,<br>1–service thread. |
| ∑ CPU Running Thread | **17/22** | |
| ∑ CPU Computing Thread | **9/12** | |
| ∑ Workstation | **4/5** | |

Table 2: The statistics of parallelism (and concurrency).

Table 2 shows the amount of threads used and their purpose. The numbers separated by backslash represent a specific value before and after connecting workstation $c_4$ to the cluster.

## 8. Conclusions

The first and foremost scientific contribution is related to a somewhat different view of the distribution of numerical computations in nonlinear dynamics of structures. Is is provided by the explicit integration of equations of motion as opposed to those commonly applied approaches in the field of the parallelization of explicit numerical computations primarily focused on powerful single workstations.

Due to the complicated and lengthy development of the FEXP solver and its basic tuning, the performance tests have not been performed yet. In view of these tests, it will be necessary to make partial optimizations which primarily concern the type of communication protocol (replace the current plain text with binary data). As a next step, it is necessary to select a group of tasks for which solution within a computer cluster represents a reduction of the time required for the numerical computation. However, it assumes that the aforementioned tasks will exhibit behaviors similar to the presented test task.

The presented solution represents a starting point in the process of the further code development related both to the currently contained algorithms, but also to other algorithms, which are already in an elaborated phase. The same applies even to other software technologies especially focused on a the field of cloud computing.

## Acknowledgements

This study has been supported by the company FEM consulting s.r.o., Czech Republic.

## References

[1] Belytschko, T., Stolarski, H., and Carpenter, N.: A $C^0$ triangular plate element with onepoint quadrature. *International Journal for Numerical Methods in Engineering* **20** (1984), 787–802.

[2] Belytschko, T., Liu, W. K., and Moran, B.: *Nonlinear Finite Elements for Continua and Structures*. John Wiley & Sons, 2000.

[3] Crespo, A. J. C., Domínguez, J. M., Rogers, B. D., Gómez-Gesteira, M., Longshaw, S., Canelas, R., Vacondio, R., Barreiro, A., García-Fea, O.: DualSPHysics: Open-source parallel CFD solver based on Smoothed Particle Hydrodynamics (SPH). *Computer Physics Communications* **187** (2015), 204–216.

[4] Chaves, E. W. V.: *Notes on Continuum Mechanics (Lecture Notes on Numerical Methods in Engineering and Sciences)*. Springer, 2013.

[5] Farhat, C., Roux, F. X.: A method of finite element tearing and interconnecting and its parallel solution algorithm. *International Journal for Numerical Methods in Engineering* **32** (1991), 1205–1227.

[6] Har, J., and Tamma, K. K.: *Advances in Computational Dynamics of Particles, Materials and Structures*. John Wiley & Sons, 2012.

[7] Johnsen, S. F., Taylor, Z. A., Clarkson, M. J., et al.: NiftySim: A GPU-based nonlinear finite element package for simulation of soft tissue biomechanics. *International Journal of Computer Assisted Radiology and Surgery* **10** (2015), 1077–1095.

[8] Kolář, V., Kratochvíl, J., Leither, F., and Ženíšek, A.: *Výpočet plošných a prostorových konstrukcí metodou konečných prvků*. SNTL - Státní nakladatelství technické literatury, 1972. (In Czech.)

[9] Kruis, J., Koudelka, T., Krejčí, T., Maděra, J., Šejnoha, M., and Sýkora, J.: Parallel computing in selected civil engineering problems. *Developments in Parallel, Distributed, Grid and Cloud Computing for Engineering* **31** (2013), 67–94.

[10] Mehlhorn, K.: *Data Structures and Algorithms 3: Multi-dimensional Searching and Computational Geometry*. Springer, 1990.

[11] Mielke, A.: A mathematical framework for generalized standard materials in the rate-independent case. *Lecture Notes in Applied and Computational Mechanics* **28** (2006), 399–428.

[12] Mielke, A., and Roubíček, T.: *Rate-Independent Systems: Theory and Application (Applied Mathematical Sciences)*. Springer, 2015.

[13] Matoušek, J., and Nešetřil, J.: *Kapitoly z diskrétní matematiky*. Karolinum, Praha, 2010. (In Czech.)

[14] Němec, I., Kolář, V., Ševčík, I., Vlk, Z., Blaauwendraat, J., Buček, J., Teplý, B., Novák, D., and Štembera, V.: *Finite Element Analysis of Structures: Principles and Praxis*. Shaker Verlag, 2010.

[15] Oller, S.: *Nonlinear Dynamics of Structures*. Springer, 2015.

[16] Rek, V., and Němec, I.: Parallel computing procedure for dynamic relaxation method on GPU using NVIDIA's CUDA. *Applied Mechanics and Materials* **821** (2016), 331–337.

[17] Rek, V., and Němec, I.: Parallel computation on multicore processors using explicit form of the finite element method and C++ standard libraries. *Strojnícky casopis – Journal of Mechanical Engineering* **66** (2016), 67–78.

[18] Shen, R. W., and Gu, L.: *Introduction to the Explicit Finite Element Method for Nonlinear Transient Dynamics*. John Wiley & Sons, 2012.

[19] Vala, J.: On some composite schemes of time integration in structural dynamics. *Programs and Algorithms of Numerical Mathematics (PANM)* 19 (2018) in Hejnice (Czech Rep.), Institute of Mathematics CAS, Prague, 2019, submitted.

[20] Wriggers, P.: *Computational Contact Mechanics*. Springer, 2006.