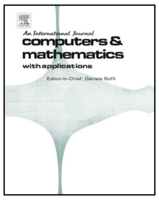
Computers and Mathematics with Applications 148 (2023) 70–80

Contents lists available at ScienceDirect

Computers and Mathematics with Applications

journal homepage: www.elsevier.com/locate/camwa



Efficient strategy for space-time based finite element analysis of vibrating structures

Bartłomiej Dyniewicz [a], Jacek M. Bajkowski [b,*], Czesław I. Bajer [a,*]

[a] Institute of Fundamental Technological Research, Polish Academy of Sciences, Pawińskiego 5b, Warsaw, 02-106, Poland
[b] Faculty of Mechanical and Industrial Engineering, Warsaw University of Technology, Narbutta 85, Warsaw, 02-524, Poland

A R T I C L E   I N F O

Keywords:
Space-time
Finite element
Direct decoupling
Vibration
Moving load
Parallel algorithm

A B S T R A C T

This paper presents an efficient parallel computing strategy to solve large-scale structural vibration problems. The proposed approach utilises a novel direct method that operates using simplex-shaped space-time finite elements and allows for the direct decoupling of variables during the assembly of global matrices. The method uses consistent stiffness, inertia and damping matrices and deals with non-symmetric matrices. One significant advantage of this approach is that the computational cost remains unaffected by the bandwidth of the matrix in the traditional sense because only non-zero coefficients are retained. The speed of computations demonstrates a noticeable increase as the number of nodes and the problem's dimensionality grow. To demonstrate the effectiveness of the parallel space-time approach, a comparison with a sequentially executed code was conducted. The results indicate that the proposed method enables calculations at least 20 times faster than those achieved using the classical finite element method. Furthermore, the parallelisation algorithm was successfully implemented to solve a dynamics problem involving a large-scale, three-dimensional railway structure subjected to a moving load. Remarkably, the problem was solved in a reasonable amount of time using a relatively low-cost personal computer.

1. Introduction

When properly implemented and executed, the classical finite element method (FEM) produces highly accurate results but requires a long solving time because discretising the complex structure results in a system with a large number of degrees of freedom, often reaching millions [1–4].

Even more processing power is required when moving inertial perturbance is considered since the fine discretisation along the trajectory of the movement is necessary. The wave phenomena dominate particularly in the case of high-speed motion, e.g. when a high-speed train is travelling along an embankment [5]. The induced ground vibration negatively influences the surrounding infrastructure, and the comfort of inhabitants. It motivates the development of fast and reliable numerical methods for accurately simulating rail-embankment interactions. The problem of obtaining the numerical response of such a railway system is considered in this paper as a benchmark of the proposed space-time parallelisation strategy using 3-D finite element analysis.

In the early studies of dynamic problems related to high speed trains presented by Krylov [6], the moving force was modelled only as the quasi-static force transmitted by the sleepers in a ballasted track. The investigations of ground vibrations were conducted by analytical means assuming a homogeneous elastic half-space [7,8] or by semi-analytical models for multi-layered ground [9]. Two-dimensional (2-D) models may be attractive for their computational efficiency, but are prone to underestimating the soil damping and ignoring wave propagation characteristics, while transferring a 3-D force into an equivalent 2-D force is problematic [10,11]. Yang and Hung [12] used a two-and-a-half-dimensional finite element method (2.5-D FEM) to study the dynamic response of the train track and ground. The 2.5-D FEM is useful when the applied load and structural response are 3-D while the structure itself is 2-D, which allows for simpler discretisation and reduces the computational effort [13,14]. However, longitudinally invariant geometry is assumed, which requires the ground, tunnels and rails to be homogeneous in the track direction.

The fully 3-D models allow to include irregularities in the geometry and material without compromising the accuracy. They overcome plane stress/strain limitations and allow including the wave propagation behaviour in the load-moving direction [15,16]. In [17] a fully 3-D model

* Corresponding author.
E-mail address: jm.bajkowski@gmail.com (J.M. Bajkowski).

https://doi.org/10.1016/j.camwa.2023.08.002
Received 20 February 2023; Received in revised form 20 June 2023; Accepted 1 August 2023

combining multi-body, FEM and boundary elements was formulated in time to predict vibrations in the environment due to train passage. In [18] a time domain explicit 3-D FEM model with non-linear excitation mechanisms was used to simulate the propagation and transmission of ground vibration in the vicinity of high-speed railways. In [19] Katou et al. analysed a numerical model of the embankment of the railroad under a high-speed train with a 3-D viscoelastic finite difference method. However, the execution of a 3-D analysis is extremely time-consuming and requires high-performance computing resources.

When examining vibration and wave propagation problems described with second-order differential equations of the hyperbolic type, numerical methods, like the finite element method, require a great solving time. Parallel computing techniques can facilitate the acceleration of such bulk tasks.

In [20], authors implemented the explicit nonlinear FEM for sheet forming simulation. The graphics processing unit (GPU) allowed more than 27 times faster computations than the central processing unit (CPU). In [21], the parallel GPU computations of finite elements models were 4 times more efficient when compared with single-core CPU computations. In [22], it was found that GPU can accelerate the computation by a factor of up to 20 compared to the non-parallel CPU code. In [23], authors achieved speedups of the single GPU concerning the single-core CPU larger than 100.

In [24] strategy for the matrix assembly procedure in a Galerkin implementation of local maximum entropy meshfree schemes was structured to exploit the massive parallelism of GPU architecture, demonstrating speedups reaching 1035 times when using a dedicated workstation. A more modest acceleration of up to 91 times was achieved using a mobile workstation, proving the possibility of handling industrially relevant applications on dedicated computing infrastructure and personal computers.

In the present paper, we introduce a novel approach that harnesses parallel computing to reduce the computational effort in a nonlinear time-stepping algorithm significantly. This approach differs from traditional methods used to reduce computational time in stepwise algorithms. The key novelty lies in the type of the matrices obtained directly through discretisation. In conventional approaches, the systems of equations are full (bandwise), and require a separate matrix triangulation process. In contrast, our proposed method, utilising simplex-shaped elements, directly yields systems of equations with already formulated triangular matrices during the global matrix assembly from elemental matrices.

Moreover, the features of the formulation predestine the space-time method to be efficiently mapped into parallel processing units.

To the author's knowledge, no documented efforts have been made thus far to parallelise calculations using the method of space-time elements with simplex shapes. Previous works employing the space-time elements method have described solutions to original problems more straightforwardly and intuitively. However, effective parallelisation of solutions to initial-boundary problems is only feasible using symplectic elements, which require appropriate mesh topology in both time and space.

The main chapters of the work introduce the concept of a single-stage solution for each time step in a linear problem. Non-linear problems are solved iteratively at each time step, following the traditional methods of solving systems of non-linear equations. The presented method offers a significant advantage by allowing for the separation of subsystems and limiting iterations within each time step to a specific spatial area with a considerably smaller number of unknowns compared to the entire problem domain. A demonstration of the method's feasibility in solving non-linear problems is presented in the final chapter.

There are three main advantages of the space-time approximation:

- it allows the non-stationary division of the structure into finite elements and thus adjusting the position of the mesh nodes to the travelling sources of disturbances,
- it enables the relatively easy formulation of phenomena described by differential equations with shifted arguments, using a stationary discretisation,
- it enables the partitioning of time and space into simplex elements for semi-separation of the resulting system of algebraic equations at the stage of its formation, so the size of a semi-infinite or infinite problem can be limited to a small subsystem with few unknowns [25,26].

In Section 2 the fundamentals of space-time discretisation performed with simplex-shaped finite elements are given. In Section 3, the strategy of parallelisation for space-time computations is presented for test problems, and limitations are discussed. In Section 4 the efficiency of the parallel space-time FEM is compared with other approaches. In Section 5 the solution of a 3-D example of an elongated railway embankment structure with a refined mesh is used as a benchmark. Finally, the Conclusions summarise the contribution and perspectives for further development.

2. Essentials of space-time finite element method

2.1. Space-time finite element discretisation

The idea of splitting space-time into various subdomains goes back to 1969 when Oden [27] presented discretised time layers of a physical structure representing a uni-dimensional structure consisting of quadrangles and triangles. The application of simplex-shaped elements was further developed [28,29] to directly obtain a decoupled system of algebraic equations during the formulation and assembly of global matrices. The solution can be carried out joint-by-joint, and thus the efficiency of computations is increased because only a sequence of small systems of equations, each having the same number of equations as the number of nodal degrees of freedom, must be solved, instead of one global system. No triangulation of the global matrix or LU decomposition is required. This concept is the foundation of the idea of the parallelisation strategy presented in this paper.

A vibrating structure defined in space is considered in time. The domain of interest is determined by $\Omega = \{\mathbf{x}, t : \mathbf{x} \in R^3, t \in (0, t_f)\}$. Here, $\mathbf{x} = (x, y, z)$ defines the spatial domain while the time interval defines the period of observation. In the typical approach, the spatial domain is topologically separate from the time domain. The displacements, velocities, accelerations, and their spatial derivatives are assumed at the same time t, at which the equilibrium is evaluated. Therefore, the problem is described in the plane that is one section at a time of space-time (Fig. 1a).

The transfer from a one-time layer to another one at a distance of Δt is not continuous but performed with incremental formulas.

The space-time domain surrounds the physical space and extends its dimensionality to time. The approach used in the algorithm proposed in this work assumes interpolations to be continuous in time. Subdomains are separated from volumes or hyper-volumes by two parallel planes evaluated at times t and $t + \Delta t$ (Fig. 1b). Space-time subdomains can be separated with more complex forms or even evolve in time (Fig. 1c). For simplicity, only space-time subdomains limited by time planes at a given time and creating a stationary set of nodes are considered in this work.

The classical finite element method for time-dependent problems uses the discretisation

$$\mathbf{u}(\mathbf{x}, t) = \tilde{\mathbf{N}}(\mathbf{x})\, \tilde{\mathbf{q}}_e \times \mathbf{T}(t)\,, \tag{1}$$

where $\tilde{\mathbf{q}}_e$ is the vector of element nodal unknowns evaluated at time t like the displacement or velocity vector, $\tilde{\mathbf{N}}$ is the matrix of interpolation (shape) functions in space, while $\mathbf{T}(t)$ is the interpolation function in time.

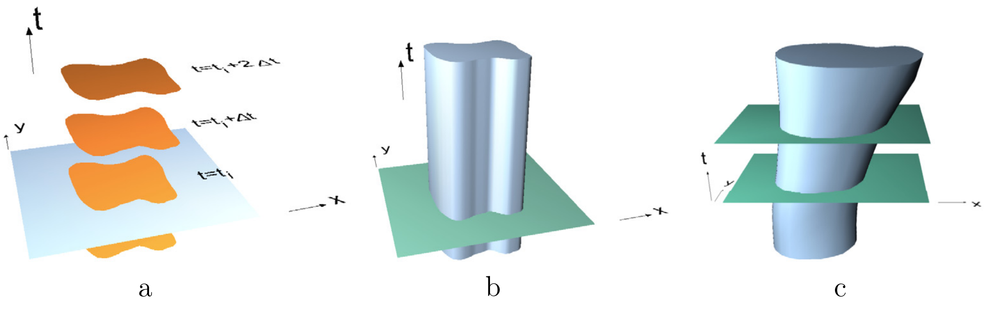


Fig. 1. 2-D object in classical discrete time integration (a), in space-time (b), and evolving in space-time (c).

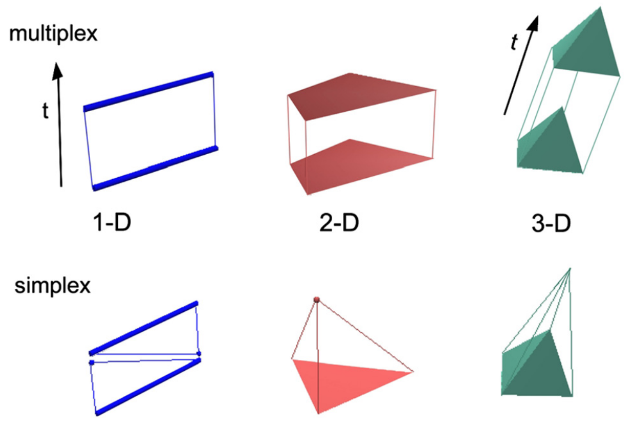


Fig. 2. Multiplex and simplex approximation in space-time for 1, 2 and 3-dimensional objects.

The space-time approach uses the formula

$$\mathbf{u}(\mathbf{x},t)=\mathbf{N}(\mathbf{x},t)\,\mathbf{q}_e \tag{2}$$

where $\mathbf{q}_e$ contains nodal unknowns for both times limiting a space-time layer. Simplex-shaped elements $\mathbf{q}_e$ contain nodal parameters for one node more than their spatial representatives, i.e. triangle instead of a line segment, tetrahedron instead of a triangle, etc. (Fig. 2). Thus $\mathbf{N}$ is the matrix shape function with a larger number of columns than $\tilde{\mathbf{N}}$. The classical approach to time-stepping algorithms can be considered a sequence of elliptic problems, one per time step. In contrast, the space-time approach is a sequence of semi-hyperbolic problems.

2.2. Numerical model in space-time approach

The space-time FEM method used in the paper is based on the topological properties of triangles, tetrahedra, and hyper-tetrahedra, which allow the global matrices of the resulting systems of algebraic equations to be generated in a semi-decoupled form. Classical methods result in band matrices, whereas triangular half-band matrices are directly formed with the simplex-shaped space-time approach. Thus the solution stage can be extremely fast since the system of equations describing the structure can be solved directly joint-by-joint. The classical formulation of the discrete dynamic problem of elasticity is presented below.

A continuum closed in a domain $\overline{V}$, being a subdomain in Euclidean space E^3 is considered. V denotes the interior of this subspace and ∂V its boundary, being the sum of ∂V_t and ∂V_u. Stress and displacement boundary conditions are assumed on ∂V. The motion of the body in time interval $[0,T]$ is considered. Displacement vector $\mathbf{u}$, velocity vector $\mathbf{v}$, inertial forces $\rho\mathbf{f}$, symmetric tensor of stresses $\boldsymbol{\sigma}$ and strains $\boldsymbol{\varepsilon}$ are determined on Cartesian product of sets $V\times[0,T]$. Vector of surface forces $\hat{\mathbf{t}}$ is determined on the product $\partial V\times[0,T]$. A set of kinematic and physical equations with boundary and initial conditions describes the problem. Equations of motion are as follows

$$\operatorname{div}\boldsymbol{\sigma}^T+\rho\mathbf{f}=\rho\frac{\partial\mathbf{v}}{\partial t}\,,\quad (\mathbf{x},t)\in V\times[0,T]. \tag{3}$$

The respective system of equations formulates the problem locally. The transition to the global formulation is obtained by multiplication of (3) by the virtual displacement function $\delta\mathbf{u}(\mathbf{x},t)$. After integration we obtain

$$\int_{t_0}^{t_1}\int_V\left(\operatorname{div}\boldsymbol{\sigma}^T+\rho\mathbf{f}-\rho\dot{\mathbf{v}}\right)\delta\mathbf{u}\,\mathrm{d}V\,\mathrm{d}t+\int_{t_0}^{t_1}\int_{\partial V_t}\hat{\mathbf{t}}\,\delta\mathbf{u}\,\mathrm{d}(\partial V)\,\mathrm{d}t=0\,. \tag{4}$$

Integration by parts yields

$$\int_{t_0}^{t_1}\int_V\rho\,(\mathbf{f}\,\delta\mathbf{u}+\dot{\mathbf{u}}\,\delta\dot{\mathbf{u}})\,\mathrm{d}V\,\mathrm{d}t+\int_{t_0}^{t_1}\int_{\partial V_t}\hat{\mathbf{t}}\,\delta\mathbf{u}\,\mathrm{d}(\partial V)\,\mathrm{d}t=\int_{t_0}^{t_1}\int_V\boldsymbol{\sigma}\delta\boldsymbol{\varepsilon}\,\mathrm{d}V\,\mathrm{d}t\,. \tag{5}$$

The domain $\{\overline{V},0\le t\le T\}$ must be discretised. In this initial-boundary problem, the half-infinite space-time band can be split into various space-time finite elements. The straightforward partition into rectangular elements in time (generally into multiplex shape elements) makes this method similar to the classical FEM, with time integration carried out with the Newmark family method.

The simplest space-time elements can be cut out of the time layer limited by planes $t=t_i$ and $t=t_{i+1}$ in forms of prisms. Thus final objects can be considered as finite spatial elements extended over a time interval (Fig. 1b). The unknown parameters like real and virtual displacements $\mathbf{u}$ and $\delta\mathbf{u}$, respectively, and their derivatives $\dot{\mathbf{u}}$, $\boldsymbol{\varepsilon}$, $\boldsymbol{\sigma}$ etc. are interpolated from nodal displacements $\mathbf{q}$ and $\delta\mathbf{q}$

$$\begin{aligned}&\mathbf{u}(\mathbf{x},t)=\mathbf{N}(\mathbf{x},t)\,\mathbf{q},\ \ \delta\mathbf{u}(\mathbf{x},t)=\mathbf{N}^*(\mathbf{x},t)\,\delta\mathbf{q},\\&\dot{\mathbf{u}}(\mathbf{x},t)=\dot{\mathbf{N}}(\mathbf{x},t)\,\mathbf{q},\ \ \delta\dot{\mathbf{u}}(\mathbf{x},t)=\dot{\mathbf{N}}^*(\mathbf{x},t)\,\delta\mathbf{q},\\&\boldsymbol{\varepsilon}(\mathbf{x},t)=\mathbf{B}(\mathbf{x},t)\,\mathbf{q},\ \ \delta\boldsymbol{\varepsilon}(\mathbf{x},t)=\mathbf{B}^*(\mathbf{x},t)\,\delta\mathbf{q},\\&\boldsymbol{\sigma}(\mathbf{x},t)=\mathbf{E}\,\mathbf{B}(\mathbf{x},t)\,\mathbf{q}.\end{aligned} \tag{6}$$

Matrix $\mathbf{B}$ can be obtained by acting with a differential operator $\mathcal{D}$ on the shape functions $\mathbf{N}$: $\mathbf{B}=\mathcal{D}\mathbf{N}$, where $\mathcal{D}=\frac{1}{2}\left(\mathrm{grad}+\mathrm{grad}^T\right)$. Symbol $(.)^*$ refers to the virtual state. The Kelvin-Voigt model of viscoelasticity defined by the Young modulus E and viscous damping coefficient η_w is assumed. The above interpolation is applied to each space-time subdomain. The set of local equations is then obtained.

Considering (6) in (5) the quadratic form of the equilibrium of the energy in time interval $[t_0,\,t_1]$ can be written

$$\sum_{e=1}^{NE}\left((\boldsymbol{\Pi}_e^T\,\delta\mathbf{q}_e)^T\boldsymbol{\Pi}_e^T\tilde{\mathbf{K}}_e\boldsymbol{\Pi}_e\cdot\boldsymbol{\Pi}_e^T\mathbf{q}_e-(\boldsymbol{\Pi}_e^T\,\delta\mathbf{q}_e)^T\boldsymbol{\Pi}_e^T\mathbf{Q}_e\right)=0. \tag{7}$$

NE is the number of space-time elements in the space-time layer. Matrices $\boldsymbol{\Pi}_e$ are zero-one tables assigning degrees of freedom of the element to the global set of degrees of freedom. These matrices determine the way of summation of local matrices into a global matrix. The same process is carried on in a classical finite element approach. The elemental space-time stiffness matrix $\tilde{\mathbf{K}}_e$ can be considered in a similar way to the equivalent stiffness matrix in the Newmark algorithm

$$\tilde{\mathbf{K}}_e = \mathbf{K}_e + \mathbf{M}_e \ . \tag{8}$$

$\mathbf{K}_e$ contributes the stiffness effect and is proportional to physical stiffness k multiplied by time step h, while $\mathbf{M}_e$ contributes the inertia effect and is proportional to physical mass m divided by time step h.

If Kelvin-Voigt model is assumed and damping forces are included in (3), two additional terms $\mathbf{W}_e$ and $\mathbf{Z}_e$ must be added to (8)

$$\tilde{\mathbf{K}}_e = \mathbf{K}_e + \mathbf{M}_e + \mathbf{W}_e + \mathbf{Z}_e \ . \tag{9}$$

Final forms of stiffness $\mathbf{K}_e$, inertia $\mathbf{M}_e$, internal $\mathbf{W}_e$ and external $\mathbf{Z}_e$ element damping matrices are given below

$$\begin{aligned}
\mathbf{K}_e &= \int_{t_0}^{t_1}\int_V (\mathcal{D}\mathbf{N})^T \mathbf{E}\,\mathcal{D}\mathbf{N}\,\mathrm{d}V\,\mathrm{d}t \ ,\\
\mathbf{M}_e &= -\int_{t_0}^{t_1}\int_V \left(\frac{\partial \mathbf{N}}{\partial t}\right)^T \mathbf{R}\,\frac{\partial \mathbf{N}}{\partial t}\,\mathrm{d}V\,\mathrm{d}t \ ,\\
\mathbf{W}_e &= \int_{t_0}^{t_1}\int_V (\mathcal{D}\mathbf{N})^T \eta_w \mathcal{D}\frac{\partial \mathbf{N}}{\partial t}\,\mathrm{d}V\,\mathrm{d}t \ ,\\
\mathbf{Z}_e &= \int_{t_0}^{t_1}\int_V \mathbf{N}^T \eta_z \frac{\partial}{\partial t}\mathbf{N}\,\mathrm{d}V\,\mathrm{d}t \ ,
\end{aligned} \tag{10}$$

where $\mathbf{R}$ is the matrix of inertia and η_z is the external damping coefficient. The vector of external forces acting on the space-time element is denoted by $\mathbf{Q}_e$

$$\mathbf{Q}_e = \int_{t_0}^{t_1}\int_V \mathbf{N}_e(\mathbf{x},t)\,\hat{\mathbf{t}}(\mathbf{x},t)\,\mathrm{d}V\,\mathrm{d}t \ . \tag{11}$$

Let us denote the initial displacement vector by $\mathbf{q}_0$ and initial velocities by $\dot{\mathbf{q}}_0$

$$\mathbf{q}_0 = \sum_{e=1}^{NE} \mathbf{\Pi}_e^T \int_{V_e} \mathbf{N}_e(\mathbf{x},0)\,\mathbf{u}(\mathbf{x},0)\,\mathrm{d}V_e \ , \quad \dot{\mathbf{q}}_0 = \sum_{e=1}^{NE} \mathbf{\Pi}_e^T \int_{V_e} \mathbf{N}_e(\mathbf{x},0)\,\dot{\mathbf{u}}(\mathbf{x},0)\,\mathrm{d}V_e \ . \tag{12}$$

$\dot{\mathbf{q}}_0$ can be reduced to the displacement vector using a difference rule, for example $\mathbf{q}_{-1} = \mathbf{q}_0 - \dot{\mathbf{q}}_0\,h$.

Finally, the equilibrium of the i-th time layer bounded by t_i and t_{i+1} is described by the equation

$$\begin{bmatrix} \tilde{\mathbf{K}}_{i(1,1)} & \tilde{\mathbf{K}}_{i(1,2)} \\ \tilde{\mathbf{K}}_{i(2,1)} & \tilde{\mathbf{K}}_{i(2,2)} \end{bmatrix} \begin{Bmatrix} \mathbf{q}_i \\ \mathbf{q}_{i+1} \end{Bmatrix} = \begin{Bmatrix} \mathbf{Q}_i \\ \mathbf{Q}_{i+1} \end{Bmatrix} \ . \tag{13}$$

Successive time layers join together and matrices cover one another, resulting in algebraic equations for each time layer

$$\begin{bmatrix}
\tilde{\mathbf{K}}_{0(1,1)} & \tilde{\mathbf{K}}_{0(1,2)} & \mathbf{0} & \mathbf{0} & \dots \\
\tilde{\mathbf{K}}_{0(2,1)} & \tilde{\mathbf{K}}_{0(2,2)} + \tilde{\mathbf{K}}_{1(1,1)} & \tilde{\mathbf{K}}_{1(1,2)} & \mathbf{0} & \dots \\
\mathbf{0} & \tilde{\mathbf{K}}_{1(2,1)} & \tilde{\mathbf{K}}_{1(2,2)} + \tilde{\mathbf{K}}_{2(1,1)} & \tilde{\mathbf{K}}_{2(1,2)} & \dots \\
\mathbf{0} & \mathbf{0} & \tilde{\mathbf{K}}_{2(2,1)} & \tilde{\mathbf{K}}_{2(2,2)} & \dots \\
\dots & \dots & \dots & \dots & \dots
\end{bmatrix}
\begin{Bmatrix} \mathbf{q}_0 \\ \mathbf{q}_1 \\ \mathbf{q}_2 \\ \mathbf{q}_3 \\ \dots \end{Bmatrix}
=
\begin{Bmatrix} \mathbf{F}_0 \\ \mathbf{F}_1 \\ \mathbf{F}_2 \\ \mathbf{F}_3 \\ \dots \end{Bmatrix} . \tag{14}$$

Forces $\mathbf{F}_i$ in the node layer $i = 1, 2, \dots$, contain assembled nodal forces $\mathbf{Q}_i$ defined for two successive time layers $[t_i - h; t_i]$ and $[t_i; t_i + h]$ joining at time t_i. Summing matrices for these time layers gives a semi-infinite matrix tree-diagonal system of algebraic equations. Starting with the known initial conditions $\mathbf{q}_0$, it is solved layer by layer. The main benefit is that the global matrices of coefficients for each time layer $\tilde{\mathbf{K}}_{0(1,2)}$, $\tilde{\mathbf{K}}_{1(1,2)}$ etc. of the system of equations are directly constructed as triangular. Paper [28] elaborates on formulating the triangular matrices and discusses an efficient method of numbering the matrix coefficients. Therefore, building further rows of the triangular matrix $\tilde{\mathbf{K}}_{i(1,2)}$ corresponding to the unknowns with higher indices can be performed simultaneously with the calculation of the unknowns with low indices, because the corresponding rows necessary for their calculation have been created earlier. Thus, it allows to start solving a system of equations before all the lines (understood as blocks) are complete. Therefore, assembling the global matrix of the system of equations and solving it can run almost in parallel, node-by-node.

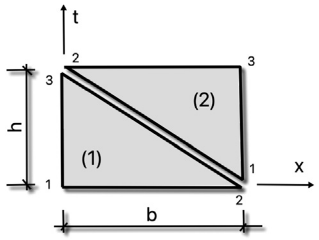


Fig. 3. Space-time simplex-shaped (triangular) elements of a bar.

To conclude the above formulations, some remarks are worth noting:

- element matrices in the case of arbitrary geometry are integrated over space and time,
- in the case of a stationary discretisation only one quarter of every matrix must be derived. The remaining quarters are identical or differ in a multiplier only,
- continuous description, in both space and time, enables the solution of complex linear differential equations with varying coefficients or non-linear equations, as in the dynamics of inertial load travelling over structures.

3. Space-time parallelisation strategy

A simplex-shaped space-time finite element described in Section 2 will be used to discuss the parallelisation algorithm for test cases.

Simplex-shaped space-time element of a bar

One spatial element is modelled as two simplex-shaped space-time triangles, with sides b along x-axis and h along t-axis (Fig. 3). Shape functions $\mathbf{N}(x,t)$ in space and time must be defined separately for each triangle. For the element No. (1) the shape function matrix is as follows

$$\mathbf{N}^{(1)} = \left[1 - \frac{x}{b} - \frac{t}{h};\ \frac{x}{b};\ \frac{t}{h}\right] \ , \tag{15}$$

and for the element No. (2)

$$\mathbf{N}^{(2)} = \left[1 - \frac{t}{h};\ 1 - \frac{x}{b};\ \frac{x}{b} + \frac{t}{h} - 1\right] \ . \tag{16}$$

In this case, shape functions are linear in space and time. The strain is

$$\varepsilon^{(1)} = \frac{\partial}{\partial x}\mathbf{N}\,\mathbf{q} = \left[-\frac{1}{b};\ \frac{1}{b};\ 0\right]\mathbf{q} \ , \tag{17}$$

where $\mathbf{q}$ contains three nodal displacements. Finally, according to (10^1) the integration over a triangle Δ results in the stiffness matrix

$$\mathbf{K}^{(1)} = \int_\Delta \frac{1}{b}\begin{bmatrix} -1 \\ 1 \\ 0 \end{bmatrix} EA\,\frac{1}{b}\,[-1;\ 1;\ 0]\,\mathrm{d}x\mathrm{d}t = \frac{EAh}{2b}\left[\begin{array}{cc|c} 1 & -1 & 0 \\ -1 & 1 & 0 \\ \hline 0 & 0 & 0 \end{array}\right] \tag{18}$$

where A is the cross-sectional area of the bar. The same process must be carried on for the element No. (2), which gives

$$\mathbf{K}^{(2)} = \frac{EAh}{2b}\left[\begin{array}{c|cc} 0 & 0 & 0 \\ \hline 0 & 1 & -1 \\ 0 & -1 & 1 \end{array}\right] . \tag{19}$$

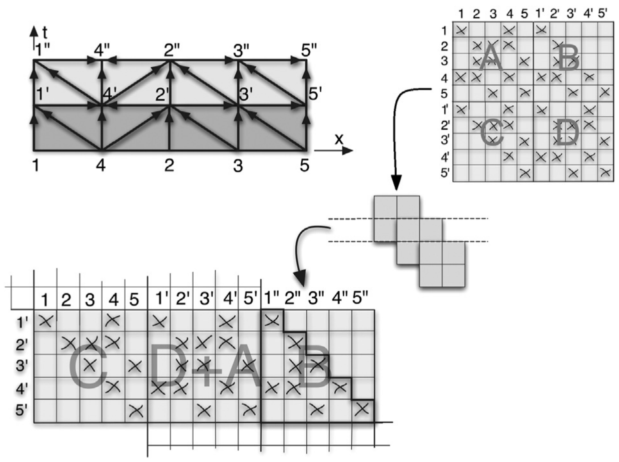


Fig. 4. Matrix assembly and information flow between nodes in exemplary 1-D structure.

Both matrices must be assembled according to the mesh topology. Similar derivation and integration must be performed for the remaining terms in (10). The assembly of matrices in time layer results in global matrices that are overlapping as depicted in Fig. 4, because one node layer determining the end of one time layer states the beginning of the next time layer.

It is convenient to use linear interpolating functions in space-time elements with higher dimensionality. In such cases, the integration of the products of these functions and their derivatives in domains of triangles, tetrahedrons and hyper-tetrahedrons are conducted analytically using simplified formulas. Notably, this integration involves efficient execution of floating-point fused multi-add (FMA) operations responsible for $a * b + c$ computations on GPU cards. The possibility of numerical integration for shape functions of greater complexity also remains viable.

Although elemental matrices require the computation of more coefficients than in the classical finite element method, submatrices are identical in pairs for the stationary partition of a structure. In the non-stationary partition, integrating the elemental space-time domain results in four different submatrices. A more general problem of 1-D bar discretised into four spatial elements is considered to demonstrate the distribution of coefficients in global matrices. The space-time layer is now split into eight space-time triangles. This way, two successive instants are joined. According to the formulation, the solution is expressed in terms of displacements. A three-level time stepping scheme is obtained, where the resulting matrices are assembled from elements neighbouring time t_i, i.e. covering intervals $[t_{i-1}, t_i]$ and $[t_i, t_{i+1}]$. The final equation of the time integration scheme, i.e. one layer of the system (14), can be written in the following matrix form

$$\mathbf{C}\mathbf{q}_{i-1} + (\mathbf{D} + \mathbf{A})\mathbf{q}_i + \mathbf{B}\mathbf{q}_{i+1} = \mathbf{F}_i \tag{20}$$

or as the final system of algebraic equations

$$\mathbf{B}\mathbf{q}_{i+1} = \mathbf{R}_i, \tag{21}$$

where $\mathbf{R}_i$ contains all known terms defined at t_i and t_{i-1}.

The matrices $\mathbf{C}$, $\mathbf{D} + \mathbf{A}$ and $\mathbf{B}$ have dimensions equal to the number of degrees of freedom. $\mathbf{C}$ is an upper triangular matrix, $\mathbf{B}$ is a lower triangular matrix, and in the case of a stationary meshing, one is the transpose of the other, as depicted in Fig. 4. In this particular numerical integration scheme, the information propagates from one node to another in consecutive time steps at a finite speed. Even if the node numbering does not initially result in triangular forms, swapping the equations and re-ordering the variables allows for the transformation of the matrices into pure triangular and band matrices. The structure of $\mathbf{B}$ is particularly important for our purposes, as it serves as the coefficient matrix in the system of equations. The first two products of matrices and known displacement vectors $\mathbf{q}_{i-1}$ and $\mathbf{q}_i$, are shifted to the right-hand side of the equation. Since the system of equations has a triangular band matrix, its solution in successive time steps can be effectively parallelised.

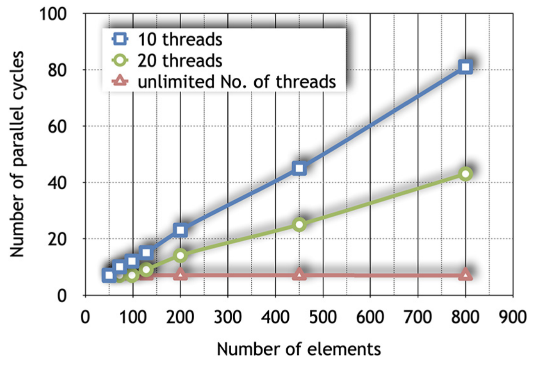


Fig. 5. Number of required cycles for one time step in 2-D mesh increasing in size.

Let the nodes of the tetrahedron in $\mathbf{R}^3$ at time t_i be 1, 2, 3, 4, and the nodes of its representative at time t_{i+1} be 5, 6, 7, 8. Then four hyper-tetrahedra are defined by the nodes: (1, 2, 3, 4, 5), (2, 3, 4, 5, 6), ... (4, 5, 6, 7, 8), which are obtained as subsets of the series of node numbers. This numbering scheme is valid for triangles representing 1-D structures and tetrahedra for 2-D structures and other elements of higher dimensionality.

The optimal parallelisation requires the specific flow of information in the initial stage of solving. In the first clock cycle, the unknowns of only one node can be determined. In successive clock cycles, neighbouring nodes are included one by one in the calculations. Thus the front line in space-time is the slope and after the appropriate number of initial steps in a single cycle, every node treated in the front line corresponds to a different moment.

In the parallel space-time method, finite element characteristic matrices must be added to a global matrix in such a way as to protect concurrent access to the same memory from more than one processor. This stage must be divided into sub-stages generating matrices of elements separated by at least one spatial element, i.e., concurrently computed elements can not have common nodes. In a two-dimensional mesh composed of regularly arranged triangles, it is possible to conceptually separate rosettes formed by triangles clustered around individual nodes. The matrix of elements surrounding a given node is determined in successive computational cycles, and the coefficients are placed in the global matrix. This process must be carried out sequentially for the successive spatial elements within a given rosette. In the mentioned mesh, a maximum of eight cycles is necessary to transition to the subsequent time step. These same operations are then repeated for the nodes within successive rosettes. Importantly, the rosettes occupy the spatial domain entirely and do not overlap. In summary, for arbitrarily large square meshes composed of triangles in two dimensions, a minimum of eight cycles is required when employing a sufficiently large number of processors. On the other hand, regular three-dimensional meshes based on tetrahedra require approximately 20 computational cycles per individual time step. This factor is constant, independent of the mesh size and restricted only by the number of processors. The fewer processors are available, the more cycles are required.

Fig. 5 shows the number of threads required for a solution with maximum parallelisation in the case of an increasing number of elements. A limited number of available threads significantly increases the number of cycles required to solve all unknowns in a single time step of the time integration process. The required number of processors/threads is significant for efficient computations with large meshes. The order of calculating the sub-matrices of the individual nodes affects the order of solving the unknowns assigned to these nodes.

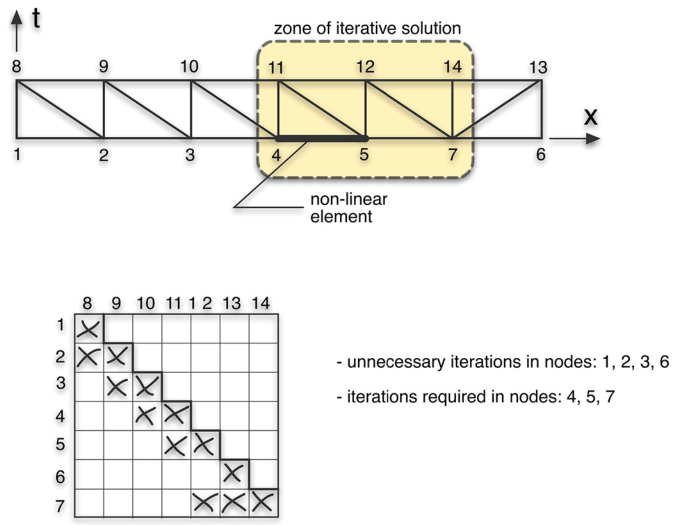

t
zone of iterative solution
8
9
10
11
12
14
13
x
1
2
3
4
5
7
6
non-linear element
- unnecessary iterations in nodes: 1, 2, 3, 6
- iterations required in nodes: 4, 5, 7

Fig. 6. The scheme of limited iteration for 1-D structure with non-linear element.

Space-time 1-D bar with nonlinearities
Important feature of the presented method involves iterations concerning the solution of nonlinear algebraic equations when nonlinearities are limited to a small spatial zone. The problems described by nonlinear equations require multiple iterative solving of systems of algebraic equations because the matrix coefficients and the vectors of the right-hand sides of the matrix equations change. This step in the computational processes is costly. The system of equations and the solution is changed due to the nonlinear influences. In a classical approach, the alteration of a single variable in the iterative solution influences all the remaining variables, and the whole system of equations must be solved. In the space-time approach, only a small subsystem related to the sub-zone representing the change of parameters and a limited surrounding zone of influence must be solved, as depicted in Fig. 6. If the behaviour of the inner element is influenced by plastic flow, only the unknowns of the surrounding nodes must be recomputed, despite the size of the mesh. At the outer parts of the spatial mesh, both triangular (simplex) and rectangular (multiplex) space-time finite elements can be added without the risk of including their nodes into the iterative recomputing. In Fig. 6 the unknowns 1, 2 and 3 (the first three rows of the matrix) do not affect the values of variables 4 and the next ones at time $t + \Delta t$ i.e. on the results at nodes 11 and beyond. Modifying the characteristic matrices in the element covered by nodes 4 and 5 (bold line in the figure) does not affect the solution in nodes with low initial numbers. Obviously, in many time steps, the non-linear disturbance affects the entire system of equations with a delay of few to several dozen time steps due to matrices multiplied by result vectors at previous moments. The impact of the change on the entire area, understood as an elliptical property, takes place in a dynamics problem solved by the space-time method with symplectic elements in a wave-like manner, in accordance with the properties of hyperbolic differential equations. This is the compromise between the full implicit time integration methods and the explicit variable decoupling.

Space-time 2-D plate
The 2-D thick plate is partitioned into 8 triangular spatial elements. These triangles gain a third dimension and become tetrahedral in space and time. The displacements of 10 nodes must be determined in every time step. In a sequential solution using a single core processor, 10 solution cycles are required per time step. The computations performed on a quad-core processor presented in Fig. 7 required only three cycles for each time step. In this example, cycles 13–15 allow the passing of one time step. The numbers of the nodes are placed in the table. Superscript i indicates the time layer for which the unknowns of a given node are solved. Over one cycle various nodes are solved at different times. The more processors are available, the fewer cycles are required. The part of computations corresponding to one and multiple processor cycles in the time-stepping procedure is described as Algorithm 1 in the Appendix.

4. Accuracy and performance verification

A square thick plate of side $L = 12$ m, simply supported at all four edges was considered to estimate the accuracy of the space-time approach. The spacial domain was subdivided into 40×40 elements and subjected to a force moving along the centre line at $v = 100$ m/s. The displacement $w(t)$ was related to the static deflection w_0 of the mid-point of the plate loaded at the centre [30]. The comparison of the deflection line obtained with a standard FEM model integrated with the Newmark method (FEM+Newmark) was compared in Fig. 8a with the results obtained using the space-time method (STFEM). The classical finite element approach with a stationary mesh and integrated with the Newmark method is identical to the multiplex space-time element approach in its particular case (see Eqn. (1)).

Fig. 8b compares curves for the central point of the plate obtained with both methods. Both pairs of curves coincide well, while the space-time approach exhibits more stiff properties than the finite element approach.

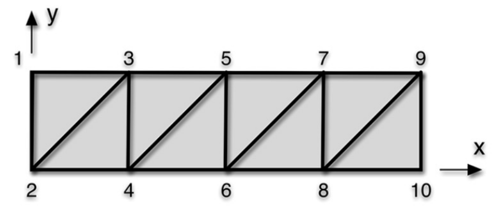
The solution of a vibrating square plate covered with an unstructured triangular mesh refined in successive cases enables an overview of the computational costs and numerical efficiency of the presented approach. Although there are two degrees of freedom in the node and the bandwidth in a 2-D problem is not wide for a direct solution, the case may be considered moderate. The matrices were treated as non-symmetric to imitate the worst-case scenario for estimating the cost of computations.

The efficiency of the parallel space-time approach was compared with relatively similar methods. A sequentially executed program was written using Fortran 90, assuming band matrices with bandwidth minimisation. The commercial FEM software performing sequential computations was used as a second reference application. The Intel Xeon Silver 4208 CPU with 8 cores, 16 threads and a base frequency of 2.1 GHz, set to single thread operations was used for reference code execution. The Nvidia Quadro GV100 card with 65536 blocks with 2048 threads was used to perform parallel computations. A dedicated code implementing the space-time finite element algorithm was written in the CUDA parallel framework for the GPU card and the results were treated as the reference.

Sparse band global stiffness and inertia matrices were generated and held with internal zeros inside the band. The bandwidth was minimised, element matrices were computed and the global matrix was formed in every time step to simulate problems with variable coefficients. Variables in double precision were declared and the Lapack library procedure DGBSV was used to solve the system of equations. This code is denoted as CPU in the tables and figures.

First, in Fig. 9*a*, a comparison of the space-time simplex approach performed on the GPU with a dedicated CPU code and with a single GPU thread is given in the case of one time step solved. The ratio of the CPU solution time (wall-clock time) to a single GPU thread is about 35. The cost is proportional to the number of unknowns and the proportionality in both cases is identical. The parallel solution of the GPU takes the same time independently of the number of nodes in the mesh.

Figs. 9*b*–*d* track the time for 10, 100, and 1000 steps computed using the CPU and GPU. The comparison is held with a simple, non-optimised code dedicated for a rough comparison. In each case, the performance of the CPU linearly depends on the number of nodes. The computation time of the GPU is constant and does not increase with an increase in the task size. Despite its slower clock speed, the GPU performs better for tasks with more than 1000 nodes. This limiting size is characteristic of the considered 2-D problem. In the case of 3-D meshes, the limiting



cycle	core 1	core 2	core 3	core 4
1	1^1			
2	2^1			
3	3^1			
4	4^1	1^2		
5	5^1	2^2		
6	6^1	3^2		
7	7^1	4^2	1^3	
8	8^1	5^2	2^3	
9	9^1	6^2	3^3	
10	10^1	7^2	4^3	1^4
11	*idle*	8^2	5^3	2^4
12	*idle*	9^2	6^3	3^4
13	1^5	10^2	7^3	4^4
14	2^5	*idle*	8^3	5^4
15	3^5	*idle*	9^3	6^4
16	4^5	1^6	10^5	7^4
17	5^5	2^6	*idle*	8^4
18	6^5	3^6	*idle*	9^4

Fig. 7. The idea of parallel computations on a 4-core computer of 2-D mesh with 10 nodes and 8 elements: numbers indicate the node while superscripts indicate the time (e.g. 4^2 expresses displacements of node 4 in time $t = 2\Delta t$).

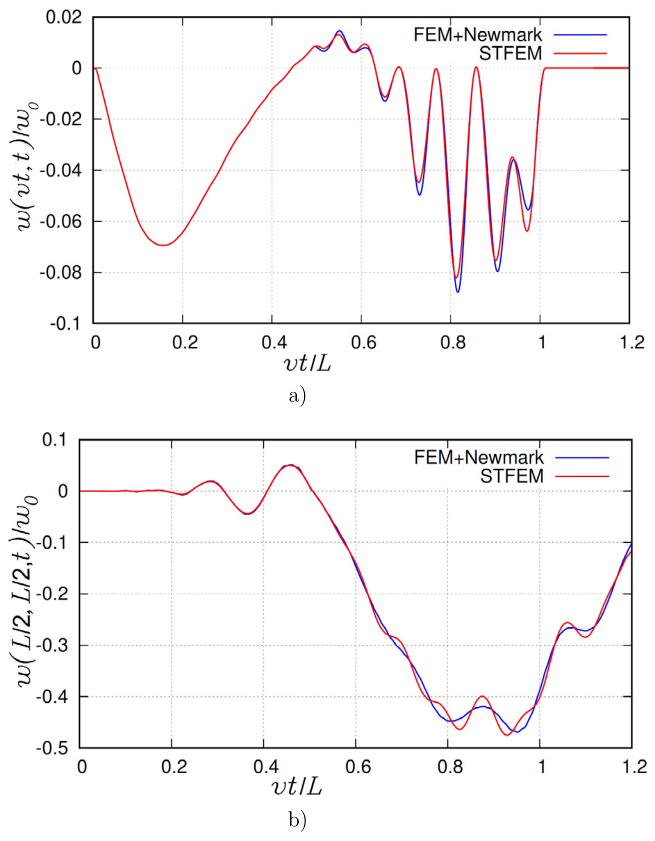


Fig. 8. Vertical displacements of the thick plate subjected to a force moving at 100 m/s: a) contact point, b) centre of the plate.

size favourable for the GPU would be reached for much smaller tasks. Nevertheless, the characteristic trends would be the same. Although the time of computations carried out on a GPU card is constant up to 10,000 nodes, it increases for greater tasks. This can be related to a saturation of the GPU threads, and a tenfold increase of the nodes involves a similar time extension.

Analysis of multiple calculation runs provides valuable insights that can be correlated with the curves depicted in Figs. 9 and 10. Firstly, the limited computational power of the graphics card employed in the tests must be acknowledged. Moreover, two main factors that significantly contribute to the increase in calculation time beyond a constant threshold, regardless of the number of grid nodes, should be mentioned. The first factor relates to the limited speed of data exchange between the computer's main memory and the graphics card. Ideally, all the necessary task data should reside within the GPU card's memory. However, in reality, the basic calculation program and its associated data resources are located in the computer's primary operating memory. Consequently, frequent data exchanges between the main memory and the GPU substantially prolong the computational process.

The second limiting factor lies in the architecture of the GPU card itself. The calculations presented in the study were performed using a card with 640 tensor cores capable of executing 64 floating-point fused multiply-add operations per clock cycle. It translates to a total of 40,000 available threads that the graphics card effectively utilises. While small or medium-sized tasks can be computed within a single cycle (or a group of cycles) on the GPU, larger tasks necessitate multiple cycles (or groups of cycles). This phenomenon becomes evident in Figs. 9 and 10 when the number of task nodes exceeds 60,000, resulting in an extension of calculation time. Significantly, the extended computational time does not simply scale linearly with the time required for a medium-sized task, as the GPU autonomously organises and manages the computational process.

Meaningful results were obtained by comparing the efficiency of the GPU with commercial software. The 2-D square plate was discretised into an increasing number of simple three-node triangular finite elements.

The straight lines in Fig. 10 show the time cost as a function of the task size. Since both dedicated finite element codes (CPU and commercial software) realise the same solution scheme, the time is a linear function of the number of unknowns. Because the commercial software uses a more efficient sparse direct solver, the line in the figure is less steep than the CPU line. The GPU line is flat and becomes advantageous in the case of more than 10^3–10^4 nodes. Despite different architecture of the CPU and GPU used in the test, one can compare the number of computational cycles as a rough measure of the algorithm's effec-

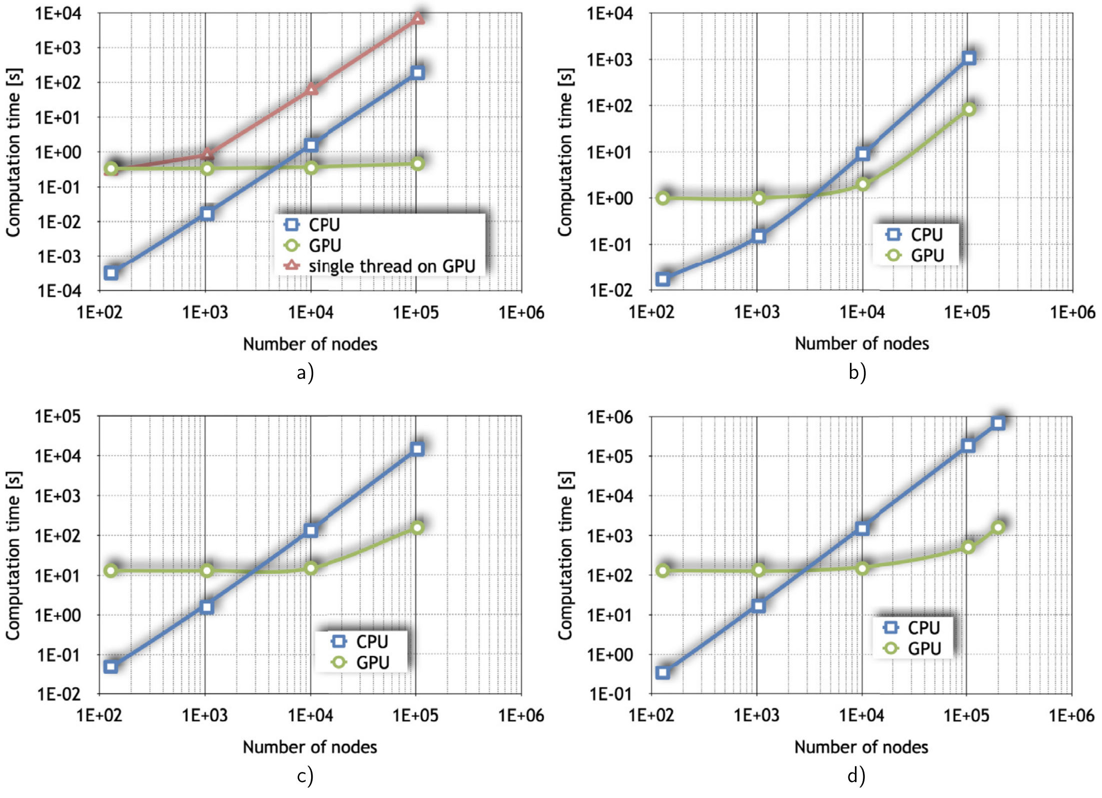


Fig. 9. Wall-clock computation time for various numbers of time steps: a) single step, b) 10 steps, c) 100 steps, d) 1000 steps.

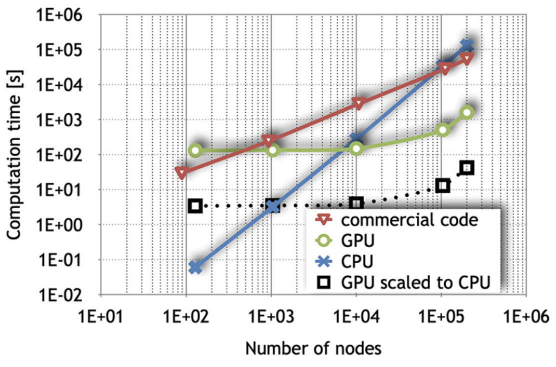


Fig. 10. Comparison with commercial software for 100 time steps.

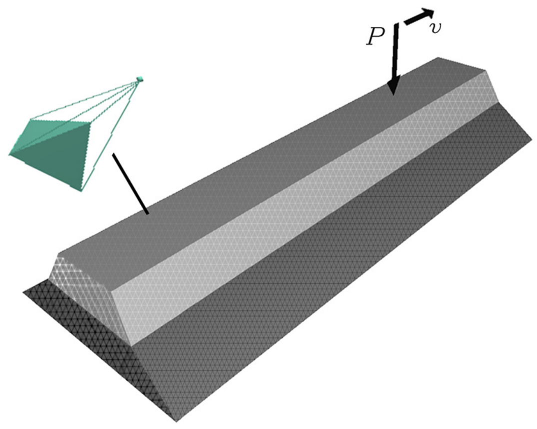


Fig. 11. Two layer railway embankment under a moving load.

tiveness. The computation time was scaled to the processor operating frequency. If the GPU line is scaled by 35 with respect to the clock frequency, the advantage of the GPU is exhibited over the entire range of tested numbers of nodes, which is shown by the dotted line in Fig. 10.

5. Results for real-scale 3-D embankment

The high-speed moving load introduces wave phenomena and a significant increase of displacement and stresses at discontinuities, commonly present in geotechnical, civil engineering and transportation problems. Thus, the inevitable fine homogeneous discretisation becomes computationally expensive. This paper aims to demonstrate the efficiency of a computing strategy using a relatively straightforward example that is easy to replicate so that specialists from different fields of science can easily analyse it or even modify it. Therefore, a problem of a large-size 3-D model of a railway embankment subjected to a load moving at speed $v = 56$ m/s was solved to demonstrate the efficiency of the parallel space-time strategy. It required a fine discretisation grid over a long distance and numerous recalculations. The response of the soil was not considered in detail since the purpose of the example was to demonstrate the numerical performance.

The embankment was modelled as a dual-layer 3-D trapezoid. It was discretised into 82944 tetrahedral spatial elements, with 15625 nodes and 46875 degrees of freedom.

Space discretisation could be performed using any mesh generator suitable for classical methods of 3-D discretisation. The only requirement is to disable using brick elements in favour of 3-D tetrahedra instead. All phenomena occurring in the modelled problems, e.g. friction, unilateral contact, material and geometric nonlinearities, can be realised in the same way as in the classical finite element method.

The upper ballast layer was assumed to be viscoelastic, while the bottom substrate was viscoplastic (Fig. 11). The load P was moving along the embankment with velocity v. Geometric and material nonlinearities forced multiple iterations in every single time step. The formulation of global matrices had to be carried out in every time step and every iteration.

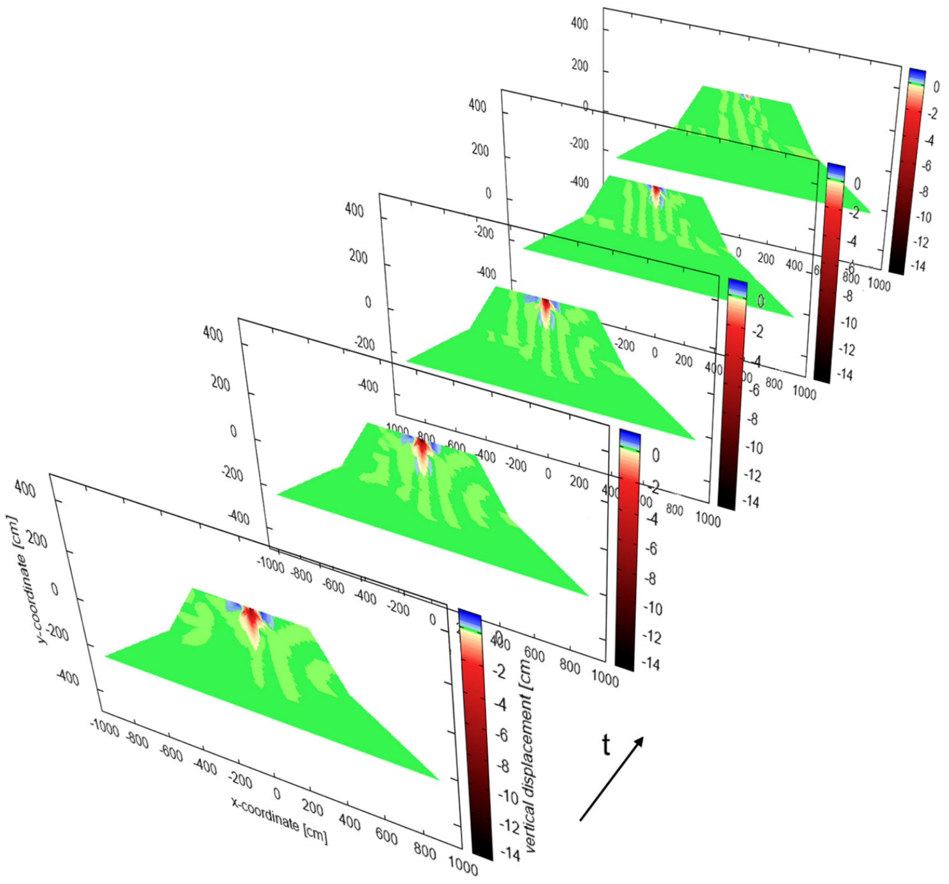


Fig. 12. Distribution of vertical displacements in successive stages.

Table 1
Total computation time in different problems (100 time steps).

problem type	CPU (scaled to GPU frequency)	GPU (single core)	GPU (parallel)	GPU parallel vs. scaled CPU (factor)
linear	105 min	121 min	1 min	0.0095
nonlinear	4631 min	253 min	63 min	0.0136

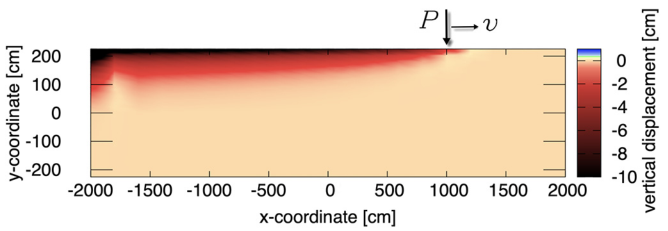


Fig. 13. Distribution of vertical displacements in the vertical section after passing 75% of the embankment's length.

The iteration in a limited number of elements presented in Fig. 6 was not implemented, since the greater part of a domain was treated as a non-linear one. Rather, such a technique would be computationally-effective for locally nested non-linear zones. The entire observation period took 1.2 million time steps multiplied by 2-3 iterations per step. Although the problem could be reduced using symmetry, the size of the task was specifically left unchanged to estimate the efficiency of the method for large-scale problems. The consistent stiffness, inertia, and damping description were used, instead of fast diagonal variable decoupling with explicit methods often applied in impact engineering.

The results of vertical displacements in selected sections of the embankment are presented in Figs. 12 and 13. The displacement maps show the concentrations around the load zone cross-section. Areas placed deeper experience displacements by a few orders smaller than the maximum ones and thus are more prone to rounding errors. In order to visualise them, the colour palette was enhanced, resulting in a seemingly asymmetrical results of a symmetrical problem.

The time needed for one single pass of the simulation on the GPU was shorter by a factor of 24–25 than on the CPU, not taking into account different processor clock frequencies. If the CPU/GPU frequency ratio of 35 was roughly taken into account, the time efficiency factor increased to about 850. This value is consistent with previous estimates presented in Fig. 10.

Table 1 compares the time required to perform the calculations of the embankment in the linear case (with a single formulation of global matrices), and nonlinear case (with matrix formulation in every time step and every iteration). In the last column, the factor shows how many times the parallel computing was shorter than sequential computing. It is consistent with the test presented in Figs. 9 and 10. The operating system decides how to apportion the computation and buffers the communication between the motherboard and the GPU card. However, one can get an overall evaluation of the efficiency of the presented space-time finite element approach. The rough comparison presented in the manuscript is not perfect, but it gives a possible assessment of the proposed parallelisation method's effectiveness.

The presented computational strategy can successfully adapt to more advanced problems of dynamics, considering additional nonlinear phenomena and complex geometries. Fig. 14a presents an example of an

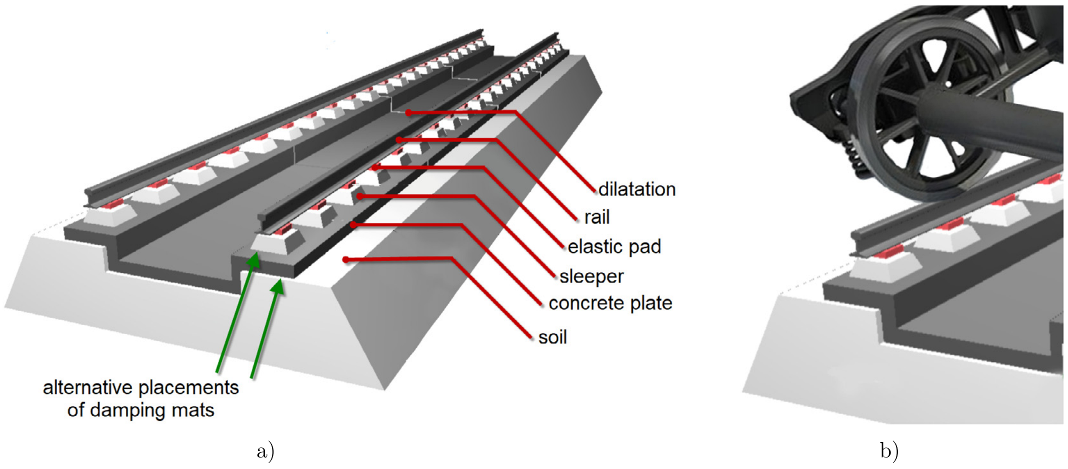


Fig. 14. A physical model of the subway track a), and visualisation of bogie's wheelset on the track -b).

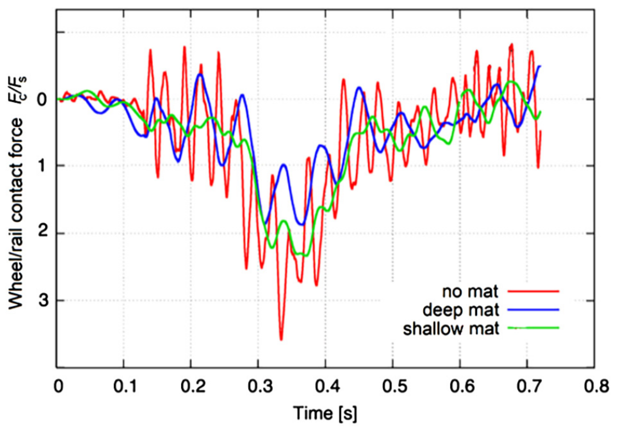


Fig. 15. Relative oscillations of the contact force between the rail and wheel for different damping mats.

underground ballastless subway track loaded with a rail vehicle travelling at a speed of 20 m/s.

A reinforced concrete slab was placed on the hardened soil, and dissipative mats were used to mitigate unwanted vibrations. In the first case, a so-called deep mat was placed underneath the plate, while in the second variant, a shallow damping mat was placed under the sleepers. The concrete slab was divided into sections by dilatations. Elastic pads separated the rails from the sleepers.

Three-dimensional tetrahedral solid elements were used to model the soil layer numerically. The horizontal and vertical parts of the reinforced concrete slab were described with 2D triangular elements of the Mindlin plate, coupled with the elements of the plane stress state. In space-time, they correspond to tetrahedrons. Elastic pads and dissipative mats were treated as Kelvin-Voigt point elastic-damping elements. The rails were modelled as elements of the Timoshenko beam, which are classic two-node sections in real space, but in space-time, they become three-node triangular elements. The rail vehicle was modelled as a structure composed of grid elements, multi-level oscillators and point masses. Compatibility of forces and displacements at the wheel-rail contact points was ensured iteratively.

The first stage of motion, when a rail vehicle bogie's wheelset enters the track was presented in Fig. 14b. The results in Fig. 15 present the oscillations of the contact force F_c between the wheel and the rail related to the static values of the contact force F_s.

The results revealed that the lowest oscillations of contact forces are generated when deep mats are used. According to our analysis, using space-time, FEM allowed for shortening the calculations 18–21 times depending on the set of material parameters and number of iterations compared with classical FEM computed sequentially. Nevertheless, taking advantage of multi-processor parallel computations requires writing dedicated procedures, which might be time-consuming. The proposed strategy of employing parallel computing with space-time elements demonstrates its universality in tackling diverse computational physics problems. It was proven effective even in handling highly intricate geometries in [31] to enhance simulations of a viscoplastic-softening brain model with nearly 47,000 degrees of freedom, simulating the impact with a rigid obstacle.

6. Conclusions

The methodology proposed in this paper introduces a novel approach for parallel bulk computation of dynamic structures. It enables the direct decoupling of the resulting system of equations and the distribution of computations to parallel processing units. The resulting global matrix takes on a triangular form and retains only the non-zero coefficients, highlighting its efficiency.

Proper discretization of space facilitates the separation of zones with nonlinearities that require iterative solutions. Consequently, only small parts of the system of algebraic equation system need to be recomputed iteratively. These characteristics reduce computational effort and allow for the frontal solution of the time-stepping scheme, contributing to increased efficiency.

A significant reduction in computation time was achieved by employing parallel computing on the GPU. This reduction varies from several times in large-scale, non-optimised complex problems to thousands of times in more straightforward test cases. As demonstrated in a full-scale example of a nonlinear dynamic problem, parallel space-time FEM outperforms classical FEM computed sequentially, providing a speed improvement of approximately 20 times.

This method holds potential for further optimisation and facilitates the solution of inverse problems while enabling efficient exploration of structural variations during the development phase of large-scale structural dynamics problems. Moreover, implementation, even on personal computers using a GPU card, ensures time-efficient analysis.

Data availability

Data will be made available on request.

Acknowledgements

This research has been supported within the projects UMO-2017/26/E/ST8/00532 and UMO-2019/33/B/ST8/02686 funded by the Polish National Science Centre, which is gratefully acknowledged by the authors.

Appendix A

Algorithm 1 Distribution of solutions over processor cores.

1. Reading of mesh geometry and other data. a spatial domain must be discretised into simplexes (triangles for 2D, tetrahedra for 3D problems).
2. Distribution of nodal solutions over processor cores (for example as in Fig. 7 scheme). The following parts (a) and (b) are performed in a sequence on a single processor core or simultaneously on parallel cores:
   (a) Calculation of matrices of a small number of space-time elements required to formulate the system of 1, 2, or 3 equations (depending on dimensionality of the physical problem) and to finally solve nodal parameters (displacements or velocities) of a single node.
   (b) Loop over these space-time elements
   - Space-time stiffness matrix is composed of four submatrices: $\tilde{\mathbf{K}}_{(1,1)}$, $\tilde{\mathbf{K}}_{(1,2)}$, $\tilde{\mathbf{K}}_{(2,1)}$, and $\tilde{\mathbf{K}}_{(2,2)}$ (see for example Eqs. (13), (18), and (19)):

     $$\begin{array}{c|c} \tilde{\mathbf{K}}_{(1,1)} & \tilde{\mathbf{K}}_{(1,2)} \\ \hline \tilde{\mathbf{K}}_{(2,1)} & \tilde{\mathbf{K}}_{(2,2)} \end{array}$$
   - Submatrix $\tilde{\mathbf{K}}_{(1,1)}$ is multiplied by $\mathbf{q}_i$ and moved to the right-hand-side vector
   - Submatrix $\tilde{\mathbf{K}}_{(1,2)}$ constitutes the matrix in the system of 2 or 3 equations, depending on the dimensionality of the problem
   - Submatrix $\tilde{\mathbf{K}}_{(2,1)}$ is multiplied by q_i and held in a global vector for the next time step
   - The system of 2 or 3 equations must be solved to determine nodal unknowns. Lower indices $i-1$, i, and $i+1$ denote the time layers

     $$\tilde{\mathbf{K}}_{(2,1)\,i-1}\,\mathbf{q}_{i-1} + \tilde{\mathbf{K}}_{(2,2)\,i-1}\,\mathbf{q}_i + \tilde{\mathbf{K}}_{(1,1)\,i}\,\mathbf{q}_i + \tilde{\mathbf{K}}_{(1,2)\,i}\,\mathbf{q}_{i+1} = \mathbf{F}_i$$

     Only $\tilde{\mathbf{K}}_{(1,2)\,i}\,\mathbf{q}_{i+1}$ remains on the left-hand-side of the system of equations. Remaining terms are moved to the right-hand-side as vectors, together with the external load vector $\mathbf{F}_i$
   - Submatrix $\tilde{\mathbf{K}}_{(2,2)}$ is multiplied by $\mathbf{q}_{i+1}$ and added to the global vector for the next time step
   - End for a single node.
3. Repeat step 2. for following nodes and remaining time steps

References

[1] A. Kużelewski, E. Zieniuk, M. Kapturczak, Acceleration of integration in parametric integral equations system using CUDA, Comput. Struct. 152 (2015) 113–124, https://doi.org/10.1016/j.compstruc.2015.02.019.

[2] G. Belinassi, A. Goldman, M.D. Gubitoso, R. Carrion, Vibration soil isolation analysis based on a 3-D frequency domain direct boundary element implementation: GPGPU acceleration, Eng. Anal. Bound. Elem. 105 (2019) 178–187, https://doi.org/10.1016/j.enganabound.2019.03.037.

[3] S. Wang, C. Wang, Y. Cai, G. Li, A novel parallel finite element procedure for non-linear dynamic problems using GPU and mixed-precision algorithm, Eng. Comput. 37 (6) (2020) 2193–2211, https://doi.org/10.1108/EC-07-2019-0328.

[4] X. Li, Y. Yan, S. Shao, S. Ji, GPU-based simulation of dynamic characteristics of ballasted railway track with coupled discrete-finite element method, Comput. Model. Eng. Sci. 126 (2) (2021) 645–671, https://doi.org/10.32604/cmes.2021.013674.

[5] T. Ekevid, H. Lane, N.-E. Wiberg, Adaptive solid wave propagation—influences of boundary conditions in high-speed train applications, Comput. Methods Appl. Mech. Eng. 195 (4) (2006) 236–250, https://doi.org/10.1016/j.cma.2004.12.030, Adaptive Modeling and Simulation.

[6] V.V. Krylov, Generation of ground vibrations by superfast trains, Appl. Acoust. 44 (2) (1995) 149–164, https://doi.org/10.1016/0003-682X(95)91370-I.

[7] G. Eason, The stresses produced in a semi-infinite solid by a moving surface force, Int. J. Eng. Sci. 2 (6) (1965) 581–609, https://doi.org/10.1016/0020-7225(65)90038-8.

[8] H.A. Dieterman, A.V. Metrikine, Steady-state displacements of a beam on an elastic half-space due to a uniformly moving constant load, Eur. J. Mech. A, Solids 16 (1997) 295–306.

[9] X. Sheng, C. Jones, M. Petyt, Ground vibration generated by a load moving along a railway track, J. Sound Vib. 228 (1) (1999) 129–156, https://doi.org/10.1006/jsvi.1999.2406.

[10] G. Saussine, C. Cholet, P. Gautier, F. Dubois, C. Bohatier, J. Moreau, Modelling ballast behaviour under dynamic loading. Part 1: a 2D polygonal discrete element method approach, Comput. Methods Appl. Mech. Eng. 195 (19) (2006) 2841–2859, https://doi.org/10.1016/j.cma.2005.07.006.

[11] H.R. Nejati, M. Ahmadi, H. Hashemolhosseini, Numerical analysis of ground surface vibration induced by underground train movement, Tunn. Undergr. Space Technol. 29 (2012) 1–9, https://doi.org/10.1016/j.tust.2011.12.006.

[12] H.-H. Hung, Y.-B. Yang, D.-W. Chang, Wave barriers for reduction of train-induced vibrations in soils, J. Geotech. Geoenviron. Eng. 130 (12) (2004) 1283–1291, https://doi.org/10.1061/(ASCE)1090-0241(2004)130:12(1283).

[13] S. Francois, M. Schevenels, P. Galvin, G. Lombaert, G. Degrande, A 2.5D coupled FE–BE methodology for the dynamic interaction between longitudinally invariant structures and a layered halfspace, Comput. Methods Appl. Mech. Eng. 199 (23) (2010) 1536–1548, https://doi.org/10.1016/j.cma.2010.01.001.

[14] P. Coulier, S. Francois, G. Degrande, G. Lombaert, Subgrade stiffening next to the track as a wave impeding barrier for railway induced vibrations, Soil Dyn. Earthq. Eng. 48 (2013) 119–131, https://doi.org/10.1016/j.soildyn.2012.12.009.

[15] A.E. Kacimi, P.K. Woodward, O. Laghrouche, G. Medero, Time domain 3D finite element modelling of train-induced vibration at high speed, Comput. Struct. 118 (2013) 66–73, https://doi.org/10.1016/j.compstruc.2012.07.011.

[16] J.Y. Shih, D.J. Thompson, A. Zervos, The effect of boundary conditions, model size and damping models in the finite element modelling of a moving load on a track/ground system, Soil Dyn. Earthq. Eng. 89 (2016) 12–27, https://doi.org/10.1016/j.soildyn.2016.07.004.

[17] P. Galvin, A. Romero, J. Dominguez, Fully three-dimensional analysis of high-speed train–track–soil-structure dynamic interaction, J. Sound Vib. 329 (24) (2010) 5147–5163, https://doi.org/10.1016/j.jsv.2010.06.016.

[18] D. Connolly, A. Giannopoulos, M. Forde, Numerical modelling of ground borne vibrations from high speed rail lines on embankments, Soil Dyn. Earthq. Eng. 46 (2013) 13–19, https://doi.org/10.1016/j.soildyn.2012.12.003.

[19] M. Katou, T. Matsuoka, O. Yoshioka, Y. Sanada, T. Miyoshi, Numerical simulation study of ground vibrations using forces from wheels of a running high-speed train, J. Sound Vib. 318 (4) (2008) 830–849, https://doi.org/10.1016/j.jsv.2008.04.053.

[20] Y. Cai, G. Li, H. Wang, G. Zheng, S. Lin, Development of parallel explicit finite element sheet forming simulation system based on GPU architecture, Adv. Eng. Softw. 45 (1) (2012) 370–379, https://doi.org/10.1016/j.advengsoft.2011.10.014.

[21] D.-K. Kang, C.-W. Kim, H.-I. Yang, GPU-based parallel computation for structural dynamic response analysis with CUDA, J. Mech. Sci. Technol. 28 (10) (2014) 4155–4162, https://doi.org/10.1007/s12206-014-0928-2.

[22] Q. Ren, C.L. Chan, Natural convection with an array of solid obstacles in an enclosure by lattice Boltzmann method on a CUDA computation platform, Int. J. Heat Mass Transf. 93 (2016) 273–285, https://doi.org/10.1016/j.ijheatmasstransfer.2015.09.059.

[23] F. Bonelli, M. Tuttafesta, G. Colonna, L. Cutrone, G. Pascazio, An MPI-CUDA approach for hypersonic flows with detailed state-to-state air kinetics using a GPU cluster, Comput. Phys. Commun. 219 (6) (2017) 178–195, https://doi.org/10.1016/j.cpc.2017.05.019.

[24] F. Cosco, F. Grecob, W. Desmet, D. Mundo, GPU accelerated initialization of local maximum-entropy meshfree methods for vibrational and acoustic problems, Comput. Methods Appl. Mech. Eng. 366 (2020) 13089, https://doi.org/10.1016/j.cma.2020.113089.

[25] C.I. Bajer, B. Dyniewicz, Virtual functions of the space-time finite element method in moving mass problems, Comput. Struct. 87 (2009) 444–455, https://doi.org/10.1016/j.compstruc.2009.01.007.

[26] B. Dyniewicz, Space-time finite element approach to general description of a moving inertial load, Finite Elem. Anal. Des. 62 (2012) 8–17, https://doi.org/10.1016/j.finel.2012.07.002.

[27] J.T. Oden, A generalized theory of finite elements, II. Applications, Int. J. Numer. Methods Eng. 1 (1969) 247–259.

[28] C.I. Bajer, Triangular and tetrahedral space–time finite elements in vibration analysis, Int. J. Numer. Methods Eng. 23 (1986) 2031–2048.

[29] C.I. Bajer, Adaptive mesh in dynamic problem by the space–time approach, Comput. Struct. 33 (2) (1989) 319–325, https://doi.org/10.1016/0045-7949(89)90002-3.

[30] B. Dyniewicz, D. Pisarski, C. Bajer, Vibrations of a Mindlin plate subjected to a pair of inertial loads moving in opposite directions, J. Sound Vib. 386 (2017) 265–282, https://doi.org/10.1016/j.jsv.2016.09.027.

[31] B. Dyniewicz, J.M. Bajkowski, C. Bajer, Effective viscoplastic-softening model suitable for brain impact modelling, Materials 15 (2) (2022) 2270, https://doi.org/10.3390/ma15062270.