Development of a GPGPU-Parallelized Hybrid Finite-Discrete Element Method for Modelling Rock Fracture

D. Fukuda [a,b], M. Mohammadnejad [b,c], H.Y. Liu [b,*], S. Dehkhoda[c], A. Chan [b], S.H. Cho[d], G.J. Min[d], H. Han [b], J. Kodama[a], Y. Fujii[a]

[a] *Faculty of Engineering, Hokkaido University, Hokkaido 060-8628, Japan*

[b] *College of Sciences and Engineering, University of Tasmania, TAS 7001, Australia*

[c] *CSIRO Minerals Resources Business Unit, Queensland Centre for Advanced Technologies, Brisbane, QLD 4069, Australia*

[d] *Department of Mineral Resources and Energy Engineering, Chonbuk National University, Jeollabuk−do 561−756, South Korea*

*Corresponding author: H.Y. Liu

E-mail: Hong.Liu@utas.edu.au

Tel/Fax: + 03-6226-2113

*Footnote

If you are interested in using our GPGPU-parallelized hybrid finite-discrete element code (Y-HFDEM IDE) for your research or verification, please write an email to the corresponding author. However, we do not accept the use of the code for military or commercial purposes.

Abstract

The hybrid finite–discrete element method (FDEM) is widely used for engineering applications, which, however, is computationally expensive and needs further development, especially when rock fracture process is modelled. This study aims to further develop a sequential hybrid FDEM code formerly proposed by the authors and parallelize it using compute unified device architecture (CUDA) C/C++ on the basis of a general purpose graphics processing unit (GPGPU) for rock engineering applications. Because the contact detection algorithm in the sequential code is not suitable for GPGPU parallelization, a different contact detection algorithm is implemented in the GPGPU-parallelized hybrid FDEM. Moreover, a number of new features are implemented in the hybrid FDEM code, including the local damping technique for efficient geostatic stress analysis, contact damping, contact friction and the absorbing boundary. Then, a number of simulations with both quasi-static and dynamic loading conditions are conducted using the GPGPU-parallelized hybrid FDEM, and the obtained results are compared both quantitatively and qualitatively with those from either theoretical analysis or the literature to calibrate the implementations. Finally, the speed-up performance of the hybrid FDEM is discussed in terms of its performance on various GPGPU accelerators and a comparison with the sequential code, which reveals that the GPGPU-parallelized hybrid FDEM can run more than 128 times faster than the sequential code if it is run on appropriate GPGPU accelerators, such as the Quadro GP100. It is concluded that the GPGPU-parallelized hybrid FDEM developed in this study is a valuable and powerful numerical tool for rock engineering applications.

Keywords: Rocks, Fracture process analysis, Hybrid FDEM, Quasi-static loading, Impact loading, GPGPU, and CUDA C/C++

1. Introduction

Understanding the fracture process mechanism of rocks is significantly important in the field of civil and mining engineering as well as other fields, such as geothermal, hydraulic, oil and gas engineering, in which rock fracture plays an important role. Recently, numerical methods have been increasingly applied to analyze the fracture process of rocks (e.g., [1]). Recent advances in computational mechanics have realized a better understanding of complex fracture processes. Generally, approaches based on computational mechanics can be classified into continuum and discontinuum formulations. In the framework of the fracture process analysis of rocks, continuum-based methods include the finite element method (FEM), the finite difference method (FDM), the boundary element method (BEM), the scaled-boundary finite element method (SBFEM), the extended finite element method (XFEM), meshless methods, methods based on peridynamics and phase-field methods, while discontinuum-based methods include the distinct element method (DEM), the lattice model (LM) method, and molecular dynamics (MD). More detailed information on recent advances in the computational fracture mechanics of rocks can be found in recently published review articles [1-3]. To realistically simulate the fracture process of rock, numerical techniques must be capable of capturing crack onset, arbitrary crack growth, the correct crack length within a given time interval and the propagating directions. In recent years, increasing attention has been paid to these techniques, which can unify the advantages of the aforementioned continuum-based and discontinuum-based methods. Attempts in this direction lead to the development of coupled methods, hybrid/hybrid methods and multiscale coupled methods [1].

The hybrid finite–discrete element method (FDEM) proposed by Munjiza [4] has been employed successfully to model problems that address the transition process from continuum to discontinuum such as rock fracturing and fragmentation. The hybrid FDEM incorporates the advantages of both continuum and discontinuum methods and can realistically simulate the transition from continuum to discontinuum caused by rock fracture. Two main implementations of the hybrid FDEM include Y code [5-8] and the commercial code ELFEN [9-12]. Several attempts have been made to actively extend the Y code, such as Y-GEO [13-19], IRAZU [20-22], Solidity [23-29], HOSS with MUNROU [30,31] and Y-Flow [32-36]. In addition, the authors also developed the Y-HFDEM IDE (integrated development environment) [37-40]. The principles of all of the hybrid FDEM codes are based on continuum mechanics, the cohesive zone model (CZM) and contact mechanics, which make the codes very computationally expensive. Therefore, developing a capable parallel computation scheme is important for handling large-scale problems with a massive number of nodes, elements and contact interactions.

To date, some successful parallel implementations of the hybrid FDEM codes using MPI (message-passing interface) (e.g., [8,9,11,30,31,41,42]) and shared-memory programming such as OpenMP (e.g., [23]) have been reported. Among these, Lukas et al. [8] proposed a novel approach for the parallelization of 2D hybrid FDEM using MPI and parallelization solvers based on dynamic domain decomposition and successfully applied the parallelized Y code to a large-scale 2D problem on a PC cluster. Meanwhile, Lei et al. [30] successfully developed the concept of a virtual parallel machine for the hybrid FDEM using MPI, which can be adapted to various computer architectures ranging from few to thousands of CPU (central processing unit) cores. Furthermore, Rougier et al. [31] introduced the HOSS with MUNROU code, which notably used 208 processors controlled by MPI and developed novel contact detection and contact force calculation algorithms. The developed code was applied successfully to 3D simulation of a dynamic Brazilian test of rock with a split Hopkinson pressure bar apparatus. ELFEN uses MPI in its parallelization scheme and has been employed successfully in 2D and 3D simulations of the rock fracture process. For example, 3D fracture process analysis of a conventional laboratory test using up to 3 million elements has been reported [11]. Additionally, Xiang et al. [23] optimized the contact detection algorithm in their Solidity code and parallelized the code using OpenMP; they modeled a packing system with 288 rock-like boulders and showed that a speedup of 9 times on 12 CPU threads can be achieved (although the details of the applied algorithm and its implementation were not provided). Relevant to the MPI-based parallelization, Guo and Zhao [43,44] developed a hierarchical FEM/DEM coupling, which had been extended for modeling of granular rocks more recently [45,46]. In general, MPI requires a large and expensive PC cluster to achieve the best performance. Meanwhile, the application of shared-memory programming such as OpenMP is limited by the total number of multi-processors that can reside in a single computer; thus, MPI is still required for large-scale problems in which each computer uses both OpenMP and MPI to transfer data among multiple computers. This means that the hybrid MPI/OpenMP is necessary.

In addition to CPU-based parallelization schemes, the GPGPU (general purpose graphics processing unit) accelerator controlled by either OpenCL (open computing language) [47] or CUDA (compute unified device architecture) [48] can be considered another promising method for the parallelization of hybrid FDEM codes. Thousands of GPU-core processors can reside and concurrently work in a small GPGPU accelerator within an ordinary laptop/desktop PC or workstation with lower energy consumption than the CPU-based PC cluster. Moreover, a GPGPU cluster with a massive number of GPGPU accelerators is also possible.

Zhang et al. [49] developed a CUDA-based GPGPU parallel version of Y code (2D) without considering the fracture process and the contact friction. Batinić et al. [50] implemented GPGPU-based parallel hybrid FDEM based on the Y code for analysis of cable structures using CUDA. However, neither of the above implementations have been employed in the simulation of rock fracture. In this regard, a GPGPU-based hybrid FDEM commercial code, namely, IRAZU [20-22], has just been developed with OpenCL and was used successfully in rock fracture simulations. Additionally, some novel GPGPU-based DEM modeling methods coupled with FEM, i.e., coupled FEM/DEM, have been proposed. For example, Nishiura et al. [40] developed the quadruple DEM (QDEM) code using GPGPU and successfully applied the code to the investigation of ballasted railway track dynamics. In addition, Ma et al.[51] and Wang et al. [52] developed a GPGPU-based implementation of continuum-based DEM. These studies [40,51,52] showed that the performance of coupled FEM/DEM can be significantly improved using GPGPU parallelization. However, the coupled FEM/DEM is a physical coupling of two methods (DEM and FEM), unlike the hybrid FDEM, in which the transition from continuum to discontinuum is modeled by CZM under the explicit FEM formulation. In other words, there are three essential differences between the coupled FEM/DEM and the hybrid FDEM: 1) the continuous behavior of the coupled FEM/DEM is modelled through springs with normal and tangential stiffnesses while that of the hybrid FDEM is through intact cohesive elements with high penalty parameters; 2) the transition from continuum to discontinuum of the coupled FEM/DEM is implemented through removing the springs while that of the hybrid FDEM is through the softening of the cohesive elements; and 3) the contact interaction between the discrete elements of the coupled FEM/DEM is calculated through contact springs while that of the hybrid FDEM is through potential contact force calculation method in the framework of explicit FEM. In this sense, the hybrid FDEM parallelized in this study is called the hybrid FDEM to distinguish it from the coupled FEM/DEM. Therefore, the hybrid FDEM is computationally more demanding than the coupled FEM/DEM.

Overall, it can be concluded that IRAZU, which is parallelized using OpenCL, is the only available GPGPU-based hybrid FDEM code to date that is capable of modeling the rock fracture process [20-22]. Thus, further studies are required to develop the GPGPU-based hybrid FDEM. In this regard, we have developed free research code*, Y-HFDEM IDE [37-40], and recently parallelized it using GPGPU with CUDA C/C++. Moreover, it is desirable to fully describe any newly implemented GPGPU-based code because the implementation of GPGPU-based codes differs from that of CPU-based sequential codes. In addition, while there is no freely available GPGPU-based hybrid FDEM code, our code is free to use and may significantly contribute to many studies in the field of rock engineering.

Based on the above background, this paper aims to explain the recently developed GPGPU-based hybrid FDEM algorithm implemented in 2D Y-HFDEM IDE, along with newly implemented functions, to increase the applicability of the algorithm in the field of rock engineering. The capability of the code in realistic modeling is demonstrated by providing examples. Thus, this paper will provide a useful basis for further improvement and development of the hybrid FDEM codes based on GPGPU.

The paper is arranged as follows. First, the theory used in 2D Y-HFDEM IDE is introduced, and then its implementation in the GPGPU parallel computation framework is explained in detail. Section 3 investigates the accuracy and capability of the developed code through several examples such as 2D fracture process analyses of rock fracture in conventional laboratory testing and blasting. In section 4, the performance of the developed GPGPU-based code is compared against that of the original CPU-based sequential code. Finally, section 5 concludes and highlights future work.

2. Hybrid finite–discrete element method implemented in Y-HFDEM IDE

The Y-HFDEM IDE was originally developed using object-oriented programming in visual C++ [37] on the basis of the CPU-based sequential open-source Y-code [5,7]. The Y-HFDEM IDE can not only significantly simplify the process of building and manipulating the input models and greatly reduce the possibility of erroneous model setup but can also display calculated results graphically in real-time with OpenGL. The preprocessor of the Y-HFDEM IDE can even generate simple FDEM mesh and specify initial conditions, physical properties, contact properties, boundary conditions, fracture criteria, and explosive charges if necessary. More complex FDEM meshes are usually generated using third-party software such as ABAQUS®, ANSYS LS-DYNA® and various open-source mesh generators such as Netgen and Gmsh. Then, the generated mesh data can be easily imported into the Y-HFDEM IDE for hybrid FDEM analyses. The postprocessor can visually display the calculated stress, displacement, velocity, force, damage, fracture and fragmentation in real-time pictures or query calculated results in specified locations and graphically display them. In addition, a number of operations such as pan, rotation, zoom, various viewports in perspective and/or orthographic modes, and slideshow are developed to manipulate the numerical models and calculated results (see [37] for further detail). The code has been successfully employed in the simulation of the fracture process in various geotechnical engineering problems [37-40]. Because of the nature of sequential programming, its main application was limited to small-scale 2D problems using a relatively rough mesh. To overcome this limitation, the parallel programming scheme using the GPGPU accelerator controlled by CUDA C/C++ is implemented into the Y-HFDEM 2D IDE through this study. Because the various hybrid FDEM-based codes have been independently developed in different research institute/organization and have different features, the fundamental features of Y-HFDEM 2D IDE along with its GPGPU-based parallelization scheme are explained in detail through the following subsections.

2.1. Fundamental theory of 2D Y-HFDEM IDE

The principles of the hybrid FDEM are based on continuum mechanics, cohesive zone modeling and contact mechanics, all of which are formulated in the framework of explicit FEM [5].

The continuum behavior of materials including rocks is modeled by an assembly of continuum 3-node triangular finite elements (TRI3s) (Fig. 1(a)). Two types of isotropic elastic constitutive models have been implemented. In the first type, which is implemented in the original Y-code and has been widely used, the isotropic elastic solid obeys Eq. (1) of the Neo-Hookean elastic model:

$$\sigma_{ij} = \frac{\lambda}{2}\left(J - \frac{1}{J}\right)\delta_{ij} + \frac{\mu}{J}\left(B_{ij} - \delta_{ij}\right) + \eta D_{ij} \ (i, j = 1, 2, 3) \tag{1}$$

where σ_{ij} denotes the Cauchy stress tensor, B_{ij} is the left Cauchy–Green strain, λ and μ are the Lame's constants, J is the determinant of the deformation gradient, η is the damping coefficient, δ_{ij} is the Kronecker's delta and D_{ij} is the rate of the deformation tensor. However, Eq. (1) cannot model anisotropic elasticity and the plane strain problem, which are also very important in the field of rock engineering. Thus, in the second type, a hyper elastic solid obeying Eqs. (2) and (3) is also implemented:

$$S_{KL} = C_{KLMN} E_{MN} \ (K, L, M, N = 1, 2) \tag{2}$$

$$\sigma_{ij} = \frac{1}{J} F_{iK} S_{KL} F_{jL} + \eta D_{ij} \ (i, j, K, L = 1, 2) \tag{3}$$

where S_{KL} denotes the 2nd Piola–Kirchhoff stress tensor, C_{KLMN} is the effective elastic stiffness tensor, E_{MN} is the Green–Lagrange strain tensor and F_{iK} is the deformation gradient. Note that the Einstein's summation convention applies in Eqs. (2) and (3). By setting C_{KLMN} in Eq. (2) properly, isotropic and anisotropic elastic behaviors can be simulated. In Eqs. (1) and (2), the infinitesimal strain tensor is not used and thus a large displacement and a large rotation can be simulated. For the simulation of the fracture process of materials under quasi-static loading, $\eta = \eta_{crit} = 2h\sqrt{(\rho E)}$ is used to achieve critical damping [5], where h, ρ and E are the element length, density and Young's modulus of the target material, respectively. Accordingly, the hybrid FDEM can address both dynamic and quasi-static problems. The σ_{ij} within each TRI3 is converted to the equivalent nodal force $\mathbf{f}_{\text{int}}$ (e.g., [52]).

The fracture of rock under mode I and mode II loading conditions, i.e., the opening and sliding of cracks, is modeled via CZM using the smeared crack model [53]. To model the behavior of the fracture process zone in front of the crack tips, tensile and shear softening is applied using an assembly of 4-node cohesive elements with an initial thickness of zero (CE4s) (Fig. 1(a)) as a function of crack opening and sliding displacements, (o, s) (Fig. 1(b)). Two methods can be used for CE4 insertion. One method involves inserting the CE4s into all of the boundaries of the TRI3s at the beginning of the analysis; this method is known as the intrinsic cohesive zone model (ICZM) [54]. The second method involves adaptively inserting the CE4s into the particular boundaries of the TRI3s with the help of adaptive remeshing techniques where a given failure criterion is met; this method is referred as the extrinsic cohesive zone model (ECZM) [54]. Many existing hybrid FDEM codes, e.g., Y code including Y-HFDEM IDE, have employed the ICZM, while some codes such as ELFEN have used the ECZM. One of the advantages of the ICZM is that the implementation and application of the parallel computing algorithm is straightforward since adaptive remeshing is unnecessary in this case. However, an "*artificial*" elastic behavior of CE4s before the onset of fracturing must be specified, which requires the introduction and correct estimation of penalty terms and the careful selection of the time step increment Δt to avoid numerical instability.

In the GPGPU-based 2D Y-HFDEM code, normal and shear cohesive tractions (σ^{coh} and τ^{coh}, respectively) acting on each face of CE4 are computed using Eqs. (2) and (3) assuming tensile and shear softening behaviors, respectively:

$$\sigma^{coh} = \begin{cases} \dfrac{2o}{o_{\text{overlap}}} T_{\text{s}} \ \text{ if } o < 0 \\ \left[\dfrac{2o}{o_{\text{p}}} - \left(\dfrac{o}{o_{\text{p}}} \right)^2 \right] f(D) T_{\text{s}} \ \text{ if } 0 \le o \le o_{\text{p}} \\ f(D) T_{\text{s}} \ \text{ if } o_{\text{p}} < o \end{cases} \tag{4}$$

$$\tau^{coh} = \begin{cases} \left[\dfrac{2|s|}{s_{\mathrm{p}}} - \left(\dfrac{|s|}{s_{\mathrm{p}}} \right)^2 \right] \left(-\sigma^{coh} \tan(\phi) + f(D)c \right) & \text{if } 0 \le |s| \le s_{\mathrm{p}} \\ -\sigma^{coh} \tan(\phi) + f(D)c & \text{if } s_{\mathrm{p}} < |s| \end{cases} \qquad (5)$$

where o_{p} and s_{p} are the elastic limits of o and s, respectively, o_{overlap} is the representative overlapping when o is negative, T_{s} is the tensile strength of CE4, c is the cohesion of CE4 and ϕ is the internal friction angle of CE4. Positive o and σ^{coh} are crack opening and tensile cohesive traction, respectively. Eq. (5) corresponds to the Mohr–Coulomb (MC) shear strength model with the tension cut-off. When ICZM is used, the "*artificial*" elastic behavior of each CE4 characterized by o_{p} and s_{p} along with o_{overlap} is necessary to connect the TRI3s to express the intact deformation process, which is given as follows [53]:

$$o_p = 2hT_s / P_f \qquad (6)$$

$$s_p = 2hc / P_{\tan} \qquad (7)$$

$$o_{overlap} = 2hT_s / P_{overlap} \qquad (8)$$

where P_{f}, $P_{\tan}$ and P_{overlap} are the penalty terms of CE4s for opening in the normal direction, sliding in the tangential direction and overlapping in the normal direction, respectively, and h is the element length; the values of the penalty terms can be considered the stiffness of the CE4 for its opening, sliding and overlapping, respectively. Ideally, these penalty values should be infinity to satisfy elastic behavior of rocks according to Eq. (1), but this condition would require an infinitesimal Δt. Therefore, a reasonably large value of the penalty terms, compared to the Young's modulus or Lame's constants, is required, as it is impossible to use infinity in actual numerical simulations. Otherwise, the intact behavior of the bulk rock shows significantly different behavior from that specified by Eqs. (1) and (2), and the elastic constants used in Eqs. (1) and (2) lose their meaning. This consideration is very important, especially for problems in which the speed of the stress wave is important (see section 3.4). The function *f(D)* in Eqs. (4) and (5) is the characteristic function for the tensile and shear softening curves (Fig. 1(b)) and depends on the damage value D of the CE4. The following definitions of D and *f(D)* are used to consider not only mode I and II fracturing but also mixed mode I–II fracturing [13,53]:

$$D = \text{Minimum}\left(1, \sqrt{\left(\frac{o-o_p}{o_t}\right)^2 + \left(\frac{|s|-s_p}{s_t}\right)^2}\right) \text{ if } o \geq o_p \text{ or } |s| > s_p \text{ , otherwise } 0 \tag{9}$$

$$f(D) = \left[1 - \frac{A+B-1}{A+B}\exp\left(D\frac{A+CB}{(A+B)(1-A-B)}\right)\right]\left[A(1-D)+B(1-D)^C\right] \;\; (0 \leq D \leq 1) \tag{10}$$

where A, B and C are the intrinsic rock properties that determine the shape of softening curves, and o_t and s_t are the critical values at which a CE4 breaks and turns into a macro/explicit fracture of o and s, respectively. If tensile loading condition is presented only (i.e. $o>o_p$ and $|s|\leq s_p$), the damage variable D in Eq. 9 becomes the pure mode I damage D_I (= minimum (1, $(o-o_p)/o_t$)). If shear loading condition is presented only (i.e. $o\leq o_p$ and $|s|>s_p$), it becomes the pure mode II damage D_{II} (= minimum (1, $(|s|-s_p)/s_t$)). If both the tensile and shear loading conditions are presented (i.e. $o\leq o_p$ and $|s|>s_p$), it becomes the mixed mode I-II damage D_{I-II} defined in Eq. 9. The true damage is defined as the sum of the pure mode I, pure mode II and mixed mode I-II damages, which is $D = D_I + D_{II} + D_{I-II}$. To avoid unrealistic damage recovery, if the trial f computed from Eq. (10) at the current time step becomes larger than that at the previous time step f_{pre}, a condition of $f = f_{pre}$ is assigned. The o_t and s_t in Eq. (9) satisfy the mode I and II fracture energies G_{fI} and G_{fII} specified in Eqs. (11) and (12), respectively:

$$G_{fI} = \int_{o_p}^{o_t} \sigma^{coh}(o)\mathrm{d}o \tag{11}$$

$$G_{fII} + W_{res} = \int_{s_p}^{s_t} \left\{\tau^{coh}(|s|)\right\}\mathrm{d}|s| \tag{12}$$

where W_{res} is the amount of work per area of CE4 done by the residual stress term in the MC shear strength model. This paper uses the same $f(D)$ with A, B and C equal to 0.63, 1.8 and 6.0 [53], respectively, for both mode I and II fracture processes because of the lack of experimental evidence. In addition, unloading, i.e., the decrease of o or $|s|$, can also occur during the softening regime, i.e., $o > o_{\mathrm{P}}$ or $|s| > s_{\mathrm{P}}$ (see Fig. 1(b)), which is modeled based on Eqs. (13) and (14) [55]:

$$\sigma^{coh} = f(D_{\max})T_s \frac{o}{o_{\max}} \text{ if } 0 < o < o_{\max} \text{ and } o_{\max} > o_{\mathrm{P}} \tag{13}$$

$$\tau^{coh} = \left\{-\sigma^{coh}\tan(\phi) + f(D_{\max})c\right\}\frac{|s|}{s_{\max}} \text{ if } |s| < s_{\max} \text{ and } \mathrm{s}_{\max} > s_{\mathrm{P}} \tag{14}$$

In each CE4, the computed σ^{coh} and τ^{coh} are converted to the equivalent nodal force $\mathbf{f}_{\mathrm{coh}}$ using a 3-point Gaussian integration scheme [53]. When either o_{t} or s_{t} is achieved in a CE4, the element becomes deactivated and its surfaces can be considered new macro-fracture surfaces.

The contact processes between the material surfaces, including the newly created macro fractures by the separation of each CE4, are modeled by the penalty method [4]; a complete and excellent explanation of the method is given in the literature [4]. As a brief explanation, when any two TRI3 elements subjected to contact detection (see the next subsection for the implementation of contact detection in the framework of GPGPU) are found to overlapped, the contact potential due to the overlapping of the two TRI3s, i.e., the contacting couple, is exactly computed. The normal contact force $f_{\mathrm{con\_n}}$, which is normally acting on the contact surface and is proportional to the contact potential, is then computed for each contacting couple. The proportional factor is called the normal contact penalty $P_{\mathrm{n\_con}}$. After the normal contact force $f_{\mathrm{con\_n}}$ and its acting point are obtained, the nominal normal overlap o_n and the relative displacement $\Delta u_{\mathrm{slide}}$ at the acting point of $f_{\mathrm{con\_n}}$ are readily computed. The contact damping model proposed by An and Tannant [56] (Fig. 2) can also be applied if the role of contact damping is very important. When this scheme is applied, the normal contact force $f_{\mathrm{con\_n}}$ mentioned above is regarded as a trial contact force $(f_{\mathrm{con\_n}})^{\mathrm{try}}$, and a trial contact stress $(\sigma_{\mathrm{con\_n}})^{\mathrm{try}}$ is then computed by dividing $(f_{\mathrm{con\_n}})^{\mathrm{try}}$ by the contact area A_{con}. Then, Eq. (15) is used to determine the contact stress $\sigma_{\mathrm{con\_n}}$:

$$\sigma_{\mathrm{con\_n}} = \begin{cases} Minimum\left(\left(\sigma_{\mathrm{con\_n}}\right)^{\mathrm{try}}, T\right) \text{ during the increase of } o_{\mathrm{n}} \text{ (Loading)} \\ T\left(o_{\mathrm{n}} / o_{n\_\max}\right)^{b} \text{ during the decrease of } o_{\mathrm{n}} \text{ (Unloading)} \end{cases} \tag{15}$$

where T is the transition force, b is the exponent, and o_{n_max} is the maximum value of o_n experienced during the loading process at the contact. T limits $\sigma_{\mathrm{con\_n}}$ and defines the transition between a linear elastic stress–displacement relationship and a 'recoverable' displacement at a constant contact stress. The values of T may be related to the physical properties of the rocks being simulated, such as the uniaxial compressive strength. The exponent b adjusts the power of the damping function that is applied to the rebound or extension phase of the contact. The value of the exponent controls the energy loss during an impact event. A very similar contact damping model is implemented in the 2D Y-Geo code in the framework of the hybrid FDEM, in which only b is modeled [13]. After $\sigma_{\mathrm{con\_n}}$ is computed using Eq. (15), it is converted to $f_{\mathrm{con\_n}}$ ($= A_{\mathrm{con}} \times \sigma_{\mathrm{con\_n}}$). The verification of the implemented contact damping is discussed in subsection 3.2. After $f_{\mathrm{con\_n}}$ is determined, the tangential contact force $f_{\mathrm{con\_tan}}$ is computed according to the classical quasi-static Coulomb friction law. First, the trial tangential contact force is computed by $(f_{\mathrm{con\_tan}})^{\mathrm{trial}} =(P_{\mathrm{tan\_con}} \times f_{\mathrm{con\_tan}} \times \Delta u_{\mathrm{slide}})$, where $P_{\mathrm{tan\_con}}$ is a tangential contact penalty. The actual $f_{\mathrm{con\_tan}}$ is computed based on Eq. (16):

$$f_{\mathrm{con\_tan}} = \begin{cases} \left(f_{\mathrm{con\_tan}}\right)^{\mathrm{trial}} & \text{if } \left(f_{\mathrm{con\_tan}}\right)^{\mathrm{trial}} \le \mu_{\mathrm{fric}} f_{\mathrm{con\_n}} \\ \mu_{\mathrm{fric}} f_{\mathrm{con\_n}} & \text{if } \left(f_{\mathrm{con\_tan}}\right)^{\mathrm{trial}} > \mu_{\mathrm{fric}} f_{\mathrm{con\_n}} \end{cases} \tag{16}$$

where μ_{fric} is the friction coefficient between the contact surfaces. The tangential contact force $f_{\mathrm{con\_tan}}$ is applied parallel to the contact surface against the direction of the relative sliding of the contact faces. The verification of the implementation of the contact friction is discussed in subsection 3.2. In each contacting couple, the contact force is converted to the equivalent nodal force $\mathbf{f}_{\mathrm{con}}$ [4].

By computing the nodal forces mentioned above, the following equation of motion, Eq. (17), is obtained and solved in the framework of explicit FEM [4]:

$$\mathbf{M}\partial^2\mathbf{u} / \partial t^2 = \mathbf{f}_{\mathrm{ext}} + \mathbf{f}_{\mathrm{int}} + \mathbf{f}_{\mathrm{coh}} + \mathbf{f}_{\mathrm{con}} \tag{17}$$

where $\mathbf{M}$ is a lumped nodal mass, $\mathbf{u}$ is the nodal displacement and $\mathbf{f}_{\mathrm{ext}}$ is the nodal force corresponding to the external load. The application of fluid-driven pressure due to detonation phenomena such as that described by An et al. [39] can also be considered. The central difference scheme is employed for the explicit time integration to solve Eq. (17). Careful selection of the time step Δt is necessary to avoid numerical instability. An excellent explanation of the reasonable selection of Δt in the hybrid FDEM can be found in section 2.3.5.2 of the literature [29].

2.2. GPGPU-based parallelization of 2D Y-HFDEM IDE by CUDA C/C++

To speed−up the simulation process of the 2D Y-HFDEM IDE, a parallel computation scheme based on the NVIDIA® GPGPU accelerator is incorporated. In our case, the computation on the GPGPU device is controlled through the NVIDIA's CUDA C/C++ [48], which is essentially an ordinary C/C++ programming language with several extensions that make it possible to leverage the power of the GPGPU in the computations. The CUDA programming model uses the abstractions of "*threads",* *"blocks"* and *"grids"*[48] (Fig. 3). A greater degree of parallelism occurs within the GPGPU device itself. Functions also known as "*kernels*" are launched on the GPGPU device and are executed by many "*threads"* in parallel. A "*thread*" is just an execution unit of a "*kernel"* that has a given "*thread index*" within a particular "*block*". As shown in Fig. 3, a "*block*" is a group of the threads, and a unique "*block index*" is given to each "*block*". The "*block index"* and *"thread index*" enable each thread to use its unique "*index*" to globally access elements in the GPGPU data array such that the collection of all threads processes the entire data set in parallel. A "*grid*" is just a group of "*blocks*". A system with a single "*grid*" is used in this study. The "*blocks"* can execute concurrently or serially depending on the number of streaming processors available in a GPGPU accelerator. Synchronization between "*threads"* within a "*block"* is possible, while no synchronization is possible between "*blocks"*. At each "*thread"* level, the corresponding code that "*threads*" execute is very similar to the CPU-based sequential code (see Appendix 1 and Fig. A1), which is one of the advantages of the application of CUDA

C/C++. For example, in the Quadro GP100 accelerator (Pascal generation [48]) used in this paper, the number of streaming processors and CUDA cores [48] are 56 and 3584, respectively. Thus, a high computational performance of the GPGPU parallelized code run on the GPU accelerator can be achieved, compared to that of ordinary CPU-based sequential code. The number of *"blocks"* per "*grid*" (N_{BpG}) and the number of *"threads"* per *"block"* (N_{TpB}) can be changed for the speed-up using GPGPU (Fig. 3). The current version of 2D Y-HFDEM IDE normally sets N_{TpB} to either 256 or 512, and N_{BpG} is automatically computed by dividing the total number of threads (N_{thread}) in each "*kernel*" by N_{TpB}, in which an additional block is needed if $N_{\mathrm{thread}}/N_{TpB}$ is not the multiple of N_{TpB}. The value of N_{thread} is set to be equal to the total number of TRI3s, CE4s, contact couples or nodes depending on the purpose of each "*kernel*".

In the GPGPU implementation of 2D Y-HFDEM IDE, the computation of each TRI3 ($\mathbf{f}_{\mathrm{int}}$ and $\mathbf{M}$), CE4 ($\mathbf{f}_{\mathrm{coh}}$), contact couple ($\mathbf{f}_{\mathrm{con}}$) or nodal equation of motion (Eq. (17)) is assigned to each GPGPU "*kernel*", as shown in Fig. 4, and processed in a massively parallel manner. In Appendix **A.1**, an exemplary code of the GPGPU "*kernel*" programmed to solve Eq. (17) is shown. It is evident from this example that the CUDA code used in the "*kernel*" is very similar to the CPU-based sequential code, which also holds true for all of the computations shown in Fig. 4. Thus, most parts of the original sequential CPU-based code can be used with minimal modifications. To compute the contact force $\mathbf{f}_{\mathrm{con}}$, a "triangle-to-triangle" (TtoT) contact interaction is used in the earliest versions of the Y-2D code[5]. This TtoT approach exactly considers the geometries of both contactor and target TRI3s, and the integration of the contact force distributed along the edges of the TRI3s is done analytically. Because this approach integrates the contact force exactly, it is precise although quite time consuming. As pointed out in the literature [30], the contact interaction in 2D can be further simplified by "triangle-to-point" (TtoP) contact interaction kinematics, which makes the implementation simpler and more time efficient. However, the precision of the computed contact force using the TtoP approach is low unless a sufficient number of target points per TRI3 is used. Thus, in the Y-

HFDEM 2D IDE, the TtoT approach is applied (instead of the TtoP approach) to ensure the precision of the computed contact force.

A flowchart of the 2D Y-HFDEM IDE is shown in Fig. 5. However, one of the challenging problems in Fig. 5 is the achievement of efficient contact detection for identifying each contacting couple only through the GPGPU without any sequential computation. For example, in the case of a sequential CPU implementation, there are powerful and efficient contact detection algorithms, such as the NBS (no binary search) contact detection algorithm proposed by Munjiza [4], which can achieve the linear search in which the required computation for contact detection is proportional to the number of TRI3 candidates subjected to contact detection. However, such contact detection algorithms are not straightforward to implement in the GPGPU-based code. In the 2D Y-HFDEM IDE, considering that hybrid FDEM modeling requires a fine mesh that often consists of TRI3s of a similar size, the following contact detection algorithm is implemented. In this algorithm, the analysis domain comprising a massive number of TRI3s is subdivided into multiple equal-sized (n_x, n_y) square subcells in the x and y directions (Fig. 6) so that the largest TRI3 in the analysis domain is completely included in a single subcell. In this way, the center of every TRI3 can have a single subcell to which it belongs (Fig. 6). By using an integer coordinate (ix, iy) (ix=0,$\cdots$,n_x-1, iy=0,$\cdots$,n_y-1) in the x and y directions for the location of each subcell, unique hash values h (=$iy\times n_x+ix$) are assigned to each subcell. For example, the five TRI3s shown in red in Fig. 6 are included in the blue subcell and have the same hash value. The subsequent contact detection procedure is explained using a simplified example shown in Fig. 7, where there are ten TRI3 candidates of similar size subjected to contact detection. First, all of the TRI3s are mapped into the integer coordinate (ix=0, 1 and 2, and iy=0,1 and 2) with n_x= n_y=3 along with each hash value. In this way, the list L-1 is readily constructed. Then, the IDs of the TRI3s in the list L-1 are sorted from smallest to largest according to the hash values as keys, which generates the list L-2 in Fig. 7. For the key sorting by hash, the radix sorting algorithm optimized for CUDA [57] and implemented in the open-source "*thrust*" library is used; thus, this procedure can also be processed in a massively parallel manner. Utilizing the list L-2 and GPGPU device shared memory [48], the list L-3 is further constructed in a GPGPU "*kernel*", which makes it possible to identify the first and last indices of the particular hash value in the List L-2. As an example, let us consider the index = 2 in the arrays of L-2, i.e. "Sorted TRI3 ID" = 0 and "Sorted_*h*" = 3. By comparing the hash values *h* in the immediately left and right neighbor indices with that in the index = 2 (i.e., indices =1 and 3 of "Sorted_*h*" array) in L-2, it can be found that the hash value *h* in the index=1 (i.e. $h = 0$) is different from that in the index = 2 (i.e. $h = 3$). Thus, the indices = 1 and 2 in L-2 correspond to the last and first TRI3s in the subcells with $h = 0$ and 3, respectively. Thus, the "last index" for $h = 0$ and the "first index" for $h = 3$ in L-3 are set to "1" and "2", respectively, as indicated using the green dash curves in L-2 and L-3 of Fig. 7. At the same time, for the hash values $h = 1$ and 2, these subcells are clearly empty with no TRI3 elements, which are filled in "E" (i.e. empty) in L-3. The same explanation is applicable to all the other indices in the arrays of L-2. In this way, L-3 is constructed. Therefore, the "first index" and the "last index" in L-3 indicate the first and last indices, respectively, in L-2 for each hash. Since this operation only requires to check the hash values of two adjacent array indices in L-2, the construction of L-3 can also be processed in a massively parallel manner. To further explain how the efficient contact detection is achieved using L-2 and L-3, let us consider a TRI3 with ID =8 in Fig. 7. It is evident that the TRI3 with ID = 8 belongs to a sub-cell with h=4. Thus, the neighboring sub-cells around the subcell $h = 4$ are: "h-1-n_x", "h -n_x", "h+1-n_x", "h-1", "h", "h+1", "h-1+n_x", "h +n_x" and "h+1+n_x", i.e. h = 0, 1, 2, 3, 4, 5, 6, 7 and 8, respectively, in this example. According to L-3, it is unnecessary to search the neighboring cells with h = 1, 2, 6 and 8 since these are the empty subcells. Then, for the remaining non-empty subcells with h = 0, 3, 4, 5 and 7, TRI3 with ID = 8 of "Sorted TRI3 ID List" in L-2 may contact with the TRI3s included in these non-empty subcells. As an example, for a subcell with $h = 0$, L-3 indicates that the "first index" = 0 and the "last index" = 1. In this case, it is only necessary to trace the indices from the "first index" to the "last index" and then process contacts for the TRI3s with IDs corresponding to each index of "Sorted TRI3 ID List". Since no sequential CPU procedure is involved in the above procedure, it becomes possible to achieve efficient contact detection using the GPGPU device only.

Therefore, the GPGPU-parallelized Y-HFDEM IDE can run completely parallel on the GPGPU device and no sequential processing is necessary (except for the input and output procedures). The data transfer from the GPGPU device to the host computer is always necessary for outputting the analysis results, the time of which is often negligible in the entire simulation time for most of the Y-HFDEM IDE simulations. The obtained results can be visualized in either OpenGL implemented in the Y-HFDEM IDE [37] or the open-source visualization software, Paraview [58].

Finally, it is worth mentioning that an efficient contact detection activation approach has been proposed and applied in many publications on the hybrid FDEM in the framework of ICZM (e.g., section 2.3.3.2 and Fig. 2.14 in [29]). In the efficient contact detection activation approach, along with the contact candidates prescribed by the user, only TRI3s in the vicinity of a newly created explicit fracture become contact candidates and are added into the contact detection list. One advantage of this approach is that the contact detection and contact force calculation are necessary only for initial material surfaces by the time the broken/failed CE4s are generated; thus, drastic time savings for the contact detection computation are possible. However, in the hybrid FDEM simulation of rocks under compressive-dominant loading conditions such as uniaxial compression, most TRI3s can overlap during compression even before the generation of explicit/macro fractures. In this case, if the amount of overlap is not negligible when broken CE4s are generated, the efficient contact detection activation approach results in the sudden application of a contact force such as a step function, i.e., a shock, which causes spurious numerical instability and results in unrealistic fragmentation. To avoid spurious numerical instability, an infinitesimally small Δt must be used, which makes the simulation impracticable. Although the efficient contact detection activation approach is implemented in the Y-HFDEM IDE, for the numerical simulation of uniaxial compression discussed in subsection 3.2, almost all TRI3s are added into the contact detection at the early stage of simulation, which makes the computation become rather demanding but is essential for avoiding spurious numerical instability. Thus, the demanding contact detection activation approach is also implemented in the current code in addition to the efficient contact detection activation approach. It should be noted that every ICZM-based hybrid FDEM code needs to address this problem carefully to avoid any inaccurate simulation (although it has not been reported in the literature).

3. Numerical tests and code validation

This section aims to conduct verifications and validations via several numerical simulations. All numerical simulations in this section are conducted using the GPGPU-parallelized Y-HFDEM IDE developed in this study. When the numerical simulation results are presented, compressive stress is regarded as negative (cold color) while tensive stress is taken as positive (warm color).

3.1. Verification of the continuum deformation of GPGPU-parallelized hybrid FDEM and the implementation of the local damping scheme

Without the insertion of CE4s, the hybrid FDEM acts as an explicit FEM to solve continuum deformation. To verify the ability of the developed code in the simulation of continuum deformations, a simple example of geostatic stress analysis under gravity is modeled using the GPGPU-parallelized Y-HFDEM IDE, and the numerical model is shown in Fig. 8. The computation of the geostatic stress field is very important in many rock engineering applications, such as the stability testing of underground excavations and slopes. Normally, geostatic stress analysis is conducted before the insertion of CE4s [13,19]. In most explicit FEM schemes, a dynamic relaxation scheme based on artificial damping is applied to compute the geostatic stress field. The critical damping technique is one of the simplest approaches that has been used in many existing applications in the literature [13,19] to compute the *in situ*/initial stress. However, it was noted during our code development that the convergence rate of the critical damping technique is very poor. To solve this problem, we implemented local damping with a mass scaling technique into the GPGPU-parallelized Y-HFDEM IDE, which was initially proposed by Cundall [59] and implemented in Itasca's commercial FLAC software [60]. In the local damping with mass scaling technique, the following Eq. (18) is used instead of the aforementioned Eq. (17):

$$\mathbf{M}^{\text{scale}}\partial^2\mathbf{u}/\partial t^2 = \mathbf{f}_{\text{tot}} + \alpha\left\|\mathbf{f}_{\text{tot}}\right\|\text{sgn}(\mathbf{v})\mathbf{1} \tag{18}$$

where $\mathbf{M}^{\text{scale}}$ is the scaled lumped mass, $\mathbf{f}_{\text{tot}}$ is the nodal out-of-balance-force, i.e., the right-hand side of Eq. (17), $\mathbf{v}$ is the nodal velocity, $\|\mathbf{f}_{\text{tot}}\|$ is the absolute value of each component of $\mathbf{f}_{\text{tot}}$, sgn(·) is the sign function automatically determined by the sign of (·) [59,60] and α is the local damping coefficient[59,60]. The comprehensive details of the local damping scheme can be found in the literature [59,60].

By applying the gravitational acceleration g = 9.806 (kg.m/s^2) downward with the boundary conditions shown in Fig. 8, initial stress analyses under the plane strain condition are conducted with an overburden pressure of 0. Two analyses are conducted using critical damping and local damping schemes. The mechanical properties of the rock mass are listed in Table 1, where the density ρ =1800 (kg/m3), Young's modulus E=12.2 (GPa) and Poisson's ration v =0.25 (correspondingly Lame constants $\lambda = \mu$ = 4.88 (GPa). For the critical damping scheme, the critical viscous damping factor η_{crit} = 3.0 (MPa·s) and the largest time step Δt = 25 (μs) for the stable simulation are selected, while α = 0.8 [59,60] and Δt =1 (s) are used for the local damping scheme with mass scaling. The number of TRI3s and nodes in the model (Fig. 8) are 98314 and 49516, respectively.

Fig. 9(a) depicts the distribution of the vertical stresses σ_{yy} simulated using both the critical damping scheme and the local damping scheme with mass scaling. Because no differences can be identified between the final resultant stresses simulated using these two schemes, only the result of one scheme is displayed in Fig. 9(a). In addition, Fig. 9(b) shows the profiles of σ_{xx} and σ_{yy} along the vertical direction of the model from the top to the bottom. Considering theoretical stresses as $\sigma_{yy}= -\rho g(100-y)$ and $\sigma_{xx}= \nu/(1-\nu)\sigma_{yy}$, the obtained results are in good agreement with the theoretical values. Therefore, the computations of each TRI3 ($\mathbf{f}_{\text{int}}$ and $\mathbf{M}$, $\mathbf{M}^{\text{scale}}$) in Eqs. (1)–(3), (17) and (18) based on the GPGPU computation can be considered to be accurate.

In the case of explicit FEMs with artificial damping schemes, static stress equilibrium is achieved when both the total kinetic energy of the system and the maximum value of $\|\mathbf{f}_{\text{tot}}\|$ among all nodes converge to zero. Fig. 10 illustrates the variations of the total kinetic energy of the system with respect to the calculation time steps during the simulation of the initial stress analyses. Evidently, the simulation with the local damping scheme with mass scaling can achieve the equilibrium stress state significantly faster than that with the critical damping scheme. To reach the final resultant stress state in Fig. 9, the local damping scheme with mass scaling requires approximately 6400 calculation steps, while the critical damping scheme requires approximately 1000000 calculation steps if the convergence criterion suggested in the literature [59,60] is applied.

3.2. Verifications of the implementation of contact damping and contact friction

To assess the accuracy of the contact damping model implemented in section 2.1, a simple impact test is modeled (Fig. 11(a)) using the GPGPU-parallelized Y-HFDEM IDE. The model is same as that reported by Mahabadi et al.[13], and the obtained results are compared with those reported in their work. The model consists of a circular elastic body with a radius of 0.1 (m) vertically impacting a fixed rigid surface. The elastic body is not allowed to fracture in this model. Following the literature [13], gravitational acceleration is neglected, the density of the elastic body is 2,700 (kg/m^3), and the kinetic energy of the elastic body before the impact event is 4.1 (kJ). Because the Lame constants λ and μ for the elastic body are not available in the literature [13], it is simply assumed that $\lambda = \mu = 5.0$ (GPa) and the damping coefficient η =0. Thus, energy dissipation is only due to contact damping. Parametric analyses are conducted by changing the exponent b and the transition force T in Eq. (15), and the normal contact penalty $P_{\text{con\_n}}$ between the elastic body and the rigid surface. The kinetic energy of the elastic body as a function of time is monitored during the parametric analyses.

Fig. 11(b) compares five cases with b values equal to 1, 2, 5, 20 and 30 when $T=\infty$ and $P_{\mathrm{con\_n}} = 0.1$ (GPa). The case with $b = 1$ corresponds to elastic contact, and thus no energy dissipation due to contact occurs. The small decrease in the kinetic energy occurs after the impact event because a small amount of the kinetic energy is converted to the strain energy of the elastic body. By changing the values of b, the amount of kinetic energy dissipated from the system increases. This behavior is the same as that reported in the literature [13] using a sequential hybrid FDEM. Cases with different values of $P_{\mathrm{con\_n}}$ (= 0.1 (GPa) and 10 (GPa)) when $b = 2$ and $T=\infty$ show that the same b does not result in the same energy dissipation when $P_{\mathrm{con\_n}}$ differs, which is a reasonable outcome because the maximum value of the nominal normal overlap o_{n_max} in Eq. (15) during the impact event does change when $P_{\mathrm{con\_n}}$ changes. However, this detail is not reported or explained in the literature [13]. Likewise, cases with different values of T (= ∞ and 1 MPa) when $b = 2$ and $P_{\mathrm{con\_n}} = 0.1$ GPa show a different amount of energy dissipation, which can also be explained by the change in o_{n_max}. Thus, it is verified that the contact detection and computation of $\mathbf{f}_{\mathrm{con}}$ are properly processed in the GPGPU code; however, this paper does not consider any contact damping in the following numerical simulations because the calibration of these parameters is beyond the scope of this paper.

To assess the accuracy of the contact friction model implemented in the GPGPU-parallelized Y-HFDEM IDE, a simple sliding test, suggested by Xiang et al. [61], is modeled, and the obtained results are compared against those from theoretical analyses. The model includes a simple sliding rock square with a length of 5 cm and a fixed rigid base as shown in Fig. 12(a). The material parameters of the rock square are assigned according to those listed in Table 1. The rock square can slip along the horizontal plane with a friction coefficient of $\mu_{\mathrm{fric}} = 0.5$. The sliding distance L is a function of initial velocity (v_i), gravity acceleration (g) and μ_{fric}, which can be determined theoretically through Eq. (19):

$$L = v_i^2 / \left(2\mu_{\mathrm{fric}} g\right) \tag{19}$$

In the numerical modeling, the sliding rock square, which is assigned various initial velocities from 2 m/s to 6 m/s, slows and stops at a distance due to the friction between the rock square and the rigid base. As illustrated in Fig. 12(b), the obtained results from the numerical simulation are in good agreement with those from the theoretical analyses.

3.3. Simulation of the rock failure process in standard laboratory rock mechanics tests

To verify the capabilities of the code in simulating the rock fracture process and associated failure mechanism, this section models two standard laboratory rock mechanics tests, i.e., the Brazilian test and the uniaxial compression test, using the GPGPU-parallelized Y-HFDEM IDE. For this purpose, the failure processes of a relatively soft limestone in these two tests are modeled. The physicomechanical properties of the limestone and the numerical input parameters used in the numerical simulation are listed in Table 1. The numerical input parameters are determined based on the methodology suggested by Tatone and Grasselli [18] while the physicomechanical properties of the limestone are obtained from laboratory tests. The diameter and thickness of the Brazilian disc (BD) are 51.7 (mm) and 25.95 (mm), respectively, and the numerical model consists of 10,520 unstructured TRI3s (Fig. 13(a)). In the numerical model of the uniaxial compression test, the height and diameter of the rock specimen are 129.5 (mm) and 51.7 (mm), respectively. The numerical model consists of 44,214 unstructured TRI3s (Fig. 13(b)). The average edge length h_{ave} of TRI3s in both models is 0.7 (mm). The rock specimens are placed between two moving rigid platens with a constant velocity of 0.05 (m/s) to satisfy quasi-static loading conditions [18]. The friction coefficient μ_{fric} between platens/rock and rock/rock are assumed to be 0.1 and 0.5, respectively [62]. Hereafter, compressive stress is shown as negative (cold color) while tension stress is regarded as positive (warm color).

Fig. 14 illustrates the screenshots of the rock failure process (Fig. 14 (a)-(c)) and the associated indirect tensile stress versus axial strain curve (Fig. 14 (d)) obtained from the numerical simulation of the Brazilian test using the GPGPU-parallelized Y-HFDEM IDE. It should be noted that the screenshots of the rock failure process shown in Fig. 14 (a), (b) and (c) are taken at the stress levels labeled using A, B and C, respectively, in the indirect tensile stress versus axial strain curve depicted in Fig. 14 (d). It can be seen from Fig. 14(a) that the horizontal stress distribution σ_{xx} in the rock specimen is almost uniform along its loading diameter. When the concentrated stress reaches the critical value (i.e., the tensile strength of CE4s for the rock), tensile failure develops in the model (Fig. 14(b)). With the two loading plates further moving toward each other, the splitting failure of the BD occurs due to the propagation of the formed macroscopic fractures, which coalesce with the microcracks initiated due to the resultant tensile stresses during the stress propagation process.

As seen from Fig. 14, at the stress labeled by point A (Fig. 14 (d)), i.e., before the peak stress, microcracks/damages are initiated and propagate near the loading areas (Fig. 14 (a)). Once the resultant stress (point B in Fig. 14 (d)) reaches the peak strength of the rock, the macro-crack, i.e., the splitting crack, then appears around the central line of the model due to the coalescence of microcracks (Fig. 14 (b)). Finally, the stress–strain curve decreases rapidly during the postfailure stage (e.g., point C in Fig. 14 (d)) when the splitting crack propagates along the sub-central line of the rock specimen dividing it into two halves (Fig. 14 (c)). The modelled rock fracture process and the obtained indirect tensile stress versus axial strain curve are in good agreement with those observed in laboratory testing of a Brazilian disc specimen under a quasi-static load. Therefore, the GPGPU-parallelized Y-HFDEM IDE can realistically model the splitting/tensile failure process of rock in the Brazilian test.

Fig. 15 illustrates the stages of the rock fracture process in the uniaxial compression test modeled using the GPGPU-parallelized Y-HFDEM IDE. Fig. 15 (a) illustrates the stress build-up and evolution process before the onset of nonlinearity in the stress-axial strain curve, i.e., point A in Fig. 16(a). As loading continues, unstable microcrack growth commences and continues to the peak stress point of the stress-strain curve, i.e., point B in Fig. 16(a). Then, the microcracks coalesce to form macroscopic cracks, which results in the rock specimen losing its bearing capacity, and correspondingly, the observed stress decreases with the strain increase. Finally, the formed macroscopic cracks propagate, resulting in the rock specimen completely losing its bearing capacity at point C in Fig. 16(a). Fig. 16 (b) compares the final fracture patterns obtained from the numerical simulation and the experimental tests. Clearly, the developed GPGPU-parallelized Y-HFDEM IDE can realistically model the failure process of rock under a uniaxial compression test in which shear failure is the dominant mechanism.

3.4. Simulation of dynamic rock fracture process

In this section, the dynamic fracture process in rock blasting is modeled using the GPGPU-parallelized Y-HFDEM IDE. Because the code is formulated based on the explicit FEM, the hybrid FDEM is also a powerful tool for the simulation of a dynamic rock fracture process and the rock fracture process. By default, the hybrid FDEM considers all boundaries reflection boundaries, i.e., free faces. However, the simulation of dynamic problems such as rock blasting often require a nonreflection, i.e., absorbing, boundary to satisfy the infinity condition. Correspondingly, the absorbing boundary is implemented into the GPGPU-parallelized Y-HFDEM IDE by viscous boundary tractions using an approach similar to that used in the literature [63,64]. The normal and tangential boundary tractions (t_n, t_s) acting perpendicularly and tangentially, respectively, to the boundary are given by Eq. (20):

$$\begin{bmatrix} t_n \\ t_s \end{bmatrix} = -\rho \begin{bmatrix} V_p v_n \\ V_s v_s \end{bmatrix} \tag{20}$$

where V_p and V_s are the P- and S-wave speeds of the target boundary, respectively, and v_n and v_s are the particle velocities that are perpendicular and tangential to the boundary, respectively. The computation of the viscous boundary tractions in each edge of the target boundary is assigned to each GPGPU "*thread*" following the same concept shown in Fig. 4. In addition, the same features as those in the literature [39] are implemented in the

GPGPU-parallelized Y-HFDEM IDE, which includes the modeling of blast-induced pressure due to detonation phenomena and the effect of the loading rate on the fracture behaviors of rock. The blast-induced pressure at time t according to the literature [39] is applied to multiple surface edges of TRI3s corresponding to the surfaces of blast holes and broken CE4s connected to each blast hole. The computations in each edge are also assigned to each GPGPU "*thread*" (following the same concept shown in Fig. 4) and processed completely in a parallel manner. However, the authors have recognized that the gas pressure model used in the literature [39] should be improved further, which is beyond the scope of this paper, and thus not used in the following verification.

The same model used in the literature [64] is utilized with the GPGPU-parallelized Y-HFDEM IDE to simulate the dynamic rock fracture analysis process (Fig. 17) because of its simplicity and validity. The model consists of a single blast hole (0.05 m in radius) within a rock disk (5 m in radius). The model includes 45,086 unstructured TRI3s, 135,258 CE4s and 135,258 nodes. The same mechanical properties of rock as those in the literature [64] are used (see Table 1 in [64]), and the simulation is conducted under the plane strain condition. It should be noted that the 2D DFPA code applied in the literature [64] is developed based on the implicit FEM using the ECZM model, and its numerical model is constructed using structured mesh. Thus, the intact stress wave propagation exactly follows the constitutive behavior of an isotropic elastic body. However, the GPGPU-parallelized Y-HFDEM IDE is developed based on the ICZM, and thus the values of the penalty numbers (P_f, P_tan and P_overlap) of the CE4s need to be set carefully. For the dynamic rock fracture modeling reported in this study, the penalty numbers of $P_\mathrm{f} = P_\mathrm{tan} = P_\mathrm{overlap}$ = 50 times the Young modulus of rock (E_rock=60 GPa) is used. Accordingly, a smaller Δt = 0.4 (ns) is used in the modeling to avoid numerical instability. The following pressure function $P(t)$ of Eq. (21), which is used in the literature [64], is applied to the surface of the blast hole:

$$P(t) = P_0\left(e^{-\alpha t} - e^{-\beta t}\right) / \left(e^{-\alpha t_0} - e^{-\beta t_0}\right) \tag{21}$$

where t_0 is the rise time of the pressure and given by $t_0 = [1/(\beta-\alpha)]\log(\beta/\alpha)$, and β/α is the controlling parameter of the pressure decay. Here, only the case of $\beta/\alpha = 1.5$ and $t_0 = 100$ (μs) is simulated. $P(t)$ is only applied to the initial surface of the blast hole following that in the literature [64], and thus the gas flow into the fractures is not considered. Two cases are considered to demonstrate the effect of the implementation of the absorbing boundary on the exterior boundary of the model shown in Fig. 17. The first case (case 1) considers the exterior boundary an absorbing boundary, while the other case (case 2) considers it a free boundary.

Fig. 18(a) compares the spatial distribution of the maximum principal stress (PS1) and broken CE4s at selected time intervals for both case 1 and case 2. Because the results of both cases are exactly the same by the time t_{arrive}, i.e., when the stress wave front reaches the exterior boundary, only one case is shown before t_{arrive}. After the stress wave front reaches the exterior boundary of the model, the results of both cases are illustrated and compared with each other. The positive value (warm color) of PS1 corresponds to tension, and broken CE4s are shown by black lines. Just after the commencement of the pressure application to the blast hole, the stress wave starts to propagate radially from the blast hole (t = 200 μs in Fig. 18 (a)). The front of the PS1 wave shows compression (cold color), and this means that both the radial and circumferential stress components are in compression. This stress state results in shear fracturing in the vicinity of the blast hole, i.e., the crushed zone. The tensile PS1 wave shown by the warm color follows this compressive stress wave front, which causes the radially propagating tensile cracks around the crushed zone. These tensile cracks extend further with the propagation of the tensile PS1 wave (t = 600 μs) and then arrest because the gas flows in the cracks are not considered. The obtained result is in good agreement with the reported results in the literature [64]. After the compressive stress wave front reaches the external boundary, as expected, no wave reflection is shown in case 1 at t = 1300 μs. However, the wave reflection is shown in case 2 at t = 1300 μs, with the tensile stress wave propagating back to the blast hole. It should be noted that the sign reversal of the reflective stress waves occurs for both the compressive and tensile stress waves impinging on the free face. Thus, the implemented absorbing boundaries show the expected behavior during rock blasting modeling. Correspondingly, it can be concluded that the dynamic fracture process analysis by the GPGPU-parallelized Y-HFDEM IDE has been verified.

To investigate the effect of the penalty numbers (P_{f}, P_{tan} and P_{overlap}) of CE4 on the intact stress wave propagations, Fig. 19(b) shows the spatial distribution of PS1 and broken CE4s at t = 900 μs for three cases with various penalty numbers (i.e., $P_{\mathrm{overlap}} = P_{\mathrm{f}} = P_{\mathrm{tan}}$ = 50 E_{rock}, 10 E_{rock} and E_{rock}). Evidently, the stress distributions clearly differ among the three cases. The condition with the penalty numbers of E_{rock} even shows a different fracture pattern than those with the penalty numbers of 50 E_{rock} and 10 E_{rock}. By computing the smallest distance between the stress wave front and the original blast-hole surface, the apparent P-wave velocities for cases with the penalty numbers of 50 E_{rock}, 10 E_{rock} and E_{rock} are found to be 4910 m/s, 4858 m/s and 4660 m/s, respectively, while the expected (theoretical) P-wave velocity is 5000 m/s according to the theory of elastodynamics. In addition, the apparent wave length of the stress wave becomes longer when the values of the penalty numbers of CE4s decrease. In dynamic fracture process analysis, stress wave propagation is the most important factor because it determines the fracturing process, and many previous publications using the ICZM-based hybrid FDEM for dynamic fracture process analyses simply set the penalty numbers of CE4s to E_{rock} or close to E_{rock}. In the hybrid FDEM, the intact behavior of rocks should only be governed by Eqs. (1) or (2) because either of these equations is the constitutive equation for "continuum behavior"; moreover, the artificial behavior of CE4s controlled by the penalty numbers should not affect the continuum behavior described by Eq. (1) or Eq. (2). Otherwise, there is no meaning in specifying the elastic parameters as input parameters. Hence, the condition of the penalty numbers of CE4s close to E_{rock} *must not be used* for any quantitatively meaningful dynamic fracture process analysis. Therefore, any ICZM-based hybrid FDEM simulations used for quantitative evaluation/prediction should set penalty numbers for CE4s with the utmost care, and the applied penalty numbers of CE4s must be validated before any dynamic fracture process analysis. The ECZM-based method does not suffer from this artificial behavior of CE4s, and our future task includes the implementation of the ECZM-based hybrid FDEM.

4. Performance

This section discusses the performance of the GPGPU-parallelized Y-HFDEM IDE, mainly in terms of its improvement compared with the sequential implementation of the Y-HFDEM IDE and its performance on several GPGPU accelerators. To accomplish this goal, the modeling of the rock failure process in the uniaxial compression test, as discussed in subsection 3.3**,** is selected as a benchmark because it is a computationally demanding simulation. The model shown in Fig. 13 (b) comprises 44,214 unstructured TRI3s, and the numbers of CE4s, nodes and initial contact couples are 66,810, 134,442 and 362,043, respectively. Because the performance of GPGPU-parallelized code is significantly dependent on the applied GPGPU accelerators [48], the GPGPU-parallelized Y-HFDEM IDE is run using several NVIDIA® GPGPU accelerators, i.e., Quadro GP100, GTX 1060, GTX 1050Ti, GTX 830M, TESLA K80(K40) and TESLA K20, to investigate its performance. Each of the NVIDIA® GPGPU accelerators can be categorized based on its generation. Sorting by date from newest to oldest, the Quadro GP100, GTX 1060 and GTX 1050Ti belong to the "Pascal" generation, whereas GTX 830M is in the "Maxwell" generation, and TESLA K80(K40) and TESLA K20 are in the "Kepler" generation. The new "Volta" and "Turing" generations have been released recently, but this paper does not consider these generations. The developed GPGPU-parallelized Y-HFDEM IDE can be run in all these GPGPU accelerators without any modifications. At the same time, an Intel® Core i7-440 CPU (3.40 GHz) is used to run our sequential CPU-based Y-HFDEM IDE.

Fig. 19 shows the speed-up times of the GPGPU-based code relative to the sequential CPU-based code running on a single thread. In Fig. 19, the vertical axis shows the quotients of the total run time using each GPGPU accelerator divided by that using the sequential CPU-based code (= 138.17 (hours)), which, thus, correspond to the speed-up times of the GPGPU-based code relative to the sequential CPU-based code. Clearly, all the GPGPU-based codes can achieve quicker times than the sequential CPU-based code, and the Quadro GP100 accelerator in the "Pascal" generation shows the best performance among them. However, the performance of GTX 830M in the "Maxwell" generation is very poor because the computational capability to perform double precision arithmetic in the GPGPU accelerator of this generation is very limited compared with that of the "Kepler" and "Pascal" generations [48]. Therefore, among the investigated GPGPU accelerators, the application of "Kepler" and "Pascal" generations are suited to achieve better performance, considering the given specifications of each accelerator [48]. In addition, because the performance of GPGPU accelerators have continued to significantly improve by generation, the GPGPU-parallelized Y-HFDEM IDE can easily achieve better speed-up times without requiring any changes if it is run on GPGPU accelerators of the newest generation, such as the Volta generation (e.g., GTX TITAN V, Quadro GV100 and Tesla V100) [48]. This finding is very important because the selection of the proper GPGPU accelerator tends to be difficult for many researchers.

In addition, the relative speed-up times between GPGPU-based and sequential CPU-based codes can further increase when many more elements and nodes with more significant contact detections/force calculations are involved. In other words, keeping all the GPGPU cores busy is the most important factor in achieving the best performance of the GPGPU-based code. For example, the uniaxial compression model shown in Fig. 13 (b) is simulated using six different average element lengths (h_{ave}), whose values are 0.15 (mm), 0.2 (mm), 0.3 (mm), 0.4 (mm), 0.5 (mm) and 0.6 (mm). To make the condition of each case closer, the value of Δt is fixed to 0.5 (ns), with other conditions remaining the same as those in Table 1, excluding the critical damping coefficient η_{crit}, which is selected based on element size. Table 2 shows the number of TRI3s, CE4s and nodes, and the initial number of contact couples in each model. It is evident that mesh discretization with $h_{\mathrm{ave}} = 0.15$ (mm) results in tremendously massive computation. The runtime required for 10,000 calculation time-steps is monitored here.

The list of actual runtimes required for 10,000 calculation time steps is shown at the bottom of Table 2 for several values of h_{ave}, for the cases of the GPGPU-based code using the Quadro GP100 accelerator and the sequential CPU-based code. The results show that 108,767 s (30.2 (hours)) are required to solve the 10,000 time steps in the sequential CPU-based code for $h_{\mathrm{ave}} = 0.15$ (mm), which means that solving the problem with this level of fine discretization is too computationally expensive using the sequential code. Based on the list in Table 2, Fig. 20 shows the speed-up times of the GPGPU-based code using the Quadro GP100 accelerator relative to the sequential CPU-based code, in which the horizontal axis represents the number of TRI3s for each h_{ave}. The relative speed-up time increases when the mesh becomes finer, i.e., when the GPGPU becomes busier. Notably, a relative speed-up time of 128.6 times is achieved for the finest mesh ($h_{\mathrm{ave}} =$ 0.15 (mm)). Thus, the GPGPU accelerator must be kept busy to achieve its best performance. Even if different models and different architectures of the GPGPU accelerators are used to run the CUDA-based GPGPU-parallelized Y-HFDEM IDE discussed in this paper, the obtained speed-up time is quite competent compared with the performance of the OpenCL-based GPGPU code "IRAZU" reported by Lisjak et al. [22]. Finally, considering that the Quadro GP100 can be installed in an ordinary workstation, the demonstrated speed-up performances indicate that less space and time are required to solve large-scale hybrid FDEM simulations by applying the GPGPU-parallelized Y-HFDEM IDE. The presented speed-up list here can provide useful information for the application of GPGPU parallelization to the hybrid FDEM simulations.

5. Conclusion and future work

This paper developed a GPGPU-parallelized Y-HFDEM IDE based on the authors' formal CPU-based sequential hybrid FDEM code. The algorithm of the GPGPU-parallelized hybrid FDEM was first given in detail so that this paper can provide a basis for further improvement and progress of any hybrid FDEM codes that were reviewed in the introduction section on the basis of GPGPU parallelization. It should be noted that a new contact detection algorithm that differs from that in the sequential CPU code was implemented in the GPGPU-parallelized Y-HFDEM IDE because the contact detection algorithm in the sequential code is not suitable for GPGPU parallelization. A number of new features that were unavailable in the original CPU-based sequential code were implemented into the GPGPU-parallelized Y-HFDEM IDE to achieve improvements in rock engineering applications, which include the implementation of efficient geostatic stress analysis through the local damping scheme with mass scaling, contact damping, contact friction and the absorbing boundary. Then, a series of numerical simulations were conducted using the GPGPU-parallelized Y-HFDEM IDE, and the obtained results were compared with those from either theoretical analysis or the literature to calibrate the implementations. Finally, GPGPU-parallelized Y-HFDEM IDE was applied in modeling the rock failure process in the Brazilian test, the uniaxial compression test and rock blasting to demonstrate its application in rock engineering. Through this study, the following conclusions can be drawn:

− The developed GPGPU-parallelized Y-HFDEM IDE can work well with the implementation of the aforementioned algorithm, and its precision is successfully verified through a series of numerical simulations.

− By conducting the fracture process analyses of rock due to quasi-static loading (the uniaxial compression test and Brazilian test) and dynamic loading (blasting), the obtained results successfully demonstrated the capability of the developed GPGPU-parallelized Y-HFDEM IDE for various types of loading configurations. From the obtained results, it can be concluded that for dynamic simulation including stress wave propagation in rock, the correct selection of penalty terms for CE4s using the ICZM-based approach is very important.

− The GPGPU-parallelized Y-HFDEM IDE can run on various GPGPU accelerators of different generations. However, the comparison of the runtimes from the simulation of the uniaxial compression test concludes that GPGPU accelerators of different generations perform in drastically different ways, and thus the proper selection of the GPGPU accelerators is suggested. Remarkably, the GPGPU-parallelized Y-HFDEM IDE running on the Quadro GP100 GPGPU accelerator achieves a speed-up of 128.6 times compared with the authors' formal sequential CPU-based code. It must be emphasized that this performance is obtained using a single GPGPU accelerator.

Therefore, the GPGPU-parallelized Y-HFDEM IDE developed in this study is a valuable and powerful numerical tool for rock engineering applications. However, further work is needed. The following are the highlights of our future tasks.

− The main purpose of this paper was to explain the newly developed GPGPU-parallelized Y-HFDEM IDE, and we intentionally selected the simpler examples for the verifications and validations of the developed code because complex problems make the verifications very challenging. Thus, we will apply the developed GPGPU-based code in wide range of rock engineering problems, such as mechanical rock cutting and rock blasting, in the next phase.

− This study focused on the development and verification of 2D GPGPU-based Y-HFDEM IDE; however, the authors have already developed the prototype of the 3D version of the GPGPU-based hybrid FDEM based on both ICZM and ECZM. The verification and validation of the 3D GPGPU-parallelized Y-HFDEM IDE are in active progress and will be presented in a separate study.

− Because multiple GPGPU accelerators can reside in a single ordinary workstation or even in a GPGPU cloud/cluster, another important task, which is in active progress, is to implement the code using multiple GPGPU accelerators to solve much larger problems using the GPGPU-based hybrid FDEM. In this case, the application of MPI is indispensable for the multiple GPGPU accelerators to communicate with each other.

− For the blasting simulation, although we implemented the approach used in An et al. [39] to model the blast-induced pressure, this approach must be improved to realize a more precise blasting simulation. To achieve a better blasting simulation, we are currently working on coupling the GPGPU-based hybrid FDEM with GPGPU-based smoothed particle hydrodynamics [65] to model the expansion of the blast-induced gas and its interaction with rock, including newly created fracture surfaces. The development is still in active progress.

Acknowledgments

The first author of this work was supported by JSPS KAKENHI (Grant Number JP18K14165) for Grant-in-Aid for Young Scientists. The corresponding author is thankful for the support of the AJF (No. 17/20470), AAS (No. RI8) and IRGS (No. L0018929) grants of Australia and the National Science Foundation grants of China (No. 51574060 and No. 51079017). All authors would like to thank the anonymous reviewers for their helpful and constructive comments that significantly contributed to improving the final version of the paper.

Appendix A.1

Fig. A.1 shows an example GPGPU "*kernel*" for solving Eq. (17), i.e., the 2D mechanical solver for nodes with *x* and *y* degrees of freedom. To enhance the readability of the code, the assignment of the boundary condition is not shown and only the update of the nodal velocity and the coordinate in the *x* direction is shown in this example. The name of the "*kernel*", i.e., the name of this function, is "Ysd2MEC_GPU". As is evident from this example, the appearance of the CUDA C/C++ code is very similar to the sequential CPU-based C/C++ code. Variables with names ending in "DEV" are global data stored on GPGPU global memory [48] and their meanings are given in the example code. The speed of GPGPU global memory access is relatively slow, and thus access should be minimized to improve performance. This "*kernel*" is launched by "Ysd2MEC_GPU<<< N_{BpG}, N_{TpB}>>>" from the host C/C++ code, which is again very similar to ordinary C/C++ except that the specification of (N_{BpG}, N_{TpB}) is necessary (see subsection **2.2**). When the "*kernel*" is launched, each CUDA thread is assigned its unique thread ID (*threadIdx.x*) and block ID (*blockIdx.x*) and N_{TpB} is automatically stored in "*blockDim.x"* (see Figs. 3 and 4). Thus, "*threadIdx.x + blockIdx.x * blockDim.x*" in the code can be considered a unique node ID "*inopo*". If each node ID is within the range of the total number of existing nodes (current_num_nodes_CST), the mechanical solver is processed for the node IDs in completely parallel manner. Note that variables with names ending in "CST" are stored in the GPGPU constant memory [48], which can achieve faster memory access than the GPGPU global memory. However, because memory size is limited, GPGPU constant memory is used to store the constant values, such as mechanical properties and topological data, that do not change throughout the simulation. Finally, using the "*register*" [48] for the declaration of local variables in each "*thread"* can achieve the fastest memory access. However, the total size of the register memory in each "*thread"* is quite limited, and defining too many local variables using the "*register*" keyword results in poor performance. In the developed code, we minimized the variables declared with the "*register*" keyword and used them repeatedly for several meanings. Thus, the readability of the actual code was slightly compromised.

References

1. Mohammadnejad M, Liu H, Chan A, Dehkhoda S, Fukuda D. An overview on advances in computational fracture mechanics of rock. *Geosystem Engineering.* 2018:1-24.
2. Lisjak A, Grasselli G. A review of discrete modeling techniques for fracturing processes in discontinuous rock masses. *Journal of Rock Mechanics and Geotechnical Engineering.* 2014;6(4):301-314.
3. Lei Q, Latham J-P, Tsang C-F. The use of discrete fracture networks for modelling coupled geomechanical and hydrological behaviour of fractured rocks. *Computers and Geotechnics.* 2017;85:151-176.
4. Munjiza AA. *The combined finite-discrete element method.* John Wiley & Sons; 2004.
5. Munjiza A. *The Combined Finite-Discrete Element Method.* Wiley; 2004.
6. Xiang J, Munjiza A, Latham JP. Finite strain, finite rotation quadratic tetrahedral element for the combined finite–discrete element method. *International Journal for Numerical Methods in Engineering.* 2009;79(8):946-978.
7. Munjiza A, Xiang J, Garcia X, Latham JP, D'Albano GGS, John NWM. The Virtual Geoscience Workbench, VGW: Open Source tools for discontinuous systems. *Particuology.* 2010;8(2):100-105.

8. Lukas T, Schiava D'Albano GG, Munjiza A. Space decomposition based parallelization solutions for the combined finite–discrete element method in 2D. *Journal of Rock Mechanics and Geotechnical Engineering.* 2014;6(6):607-615.

9. Rockfield. Rockfield Software Ltd. 2005; http://www.rockfield.co.uk/elfen.htm.

10. Elmo D, Stead D. An Integrated Numerical Modelling–Discrete Fracture Network Approach Applied to the Characterisation of Rock Mass Strength of Naturally Fractured Pillars. *Rock Mechanics and Rock Engineering.* 2010;43(1):3-19.

11. Hamdi P, Stead D, Elmo D. Damage characterization during laboratory strength testing: A 3D-finite-discrete element approach. *Computers and Geotechnics.* 2014;60:33-46.

12. Rogers S, Elmo D, Webb G, Catalan A. Volumetric Fracture Intensity Measurement for Improved Rock Mass Characterisation and Fragmentation Assessment in Block Caving Operations. *Rock Mechanics and Rock Engineering.* 2015;48(2):633-649.

13. Mahabadi OK, Lisjak A, Munjiza A, Grasselli G. Y-Geo: New Combined Finite-Discrete Element Numerical Code for Geomechanical Applications. *International Journal of Geomechanics.* 2012;12(6):676-688.

14. Lisjak A. *Investigating the influence of mechanical anisotropy on the fracturing behaviour of brittle clay shales with application to deep geological repositories*: Civil Engineering, University of Toronto; 2014.

15. Lisjak A, Figi D, Grasselli G. Fracture development around deep underground excavations: Insights from FDEM modelling. *Journal of Rock Mechanics and Geotechnical Engineering.* 2014;6(6):493-505.

16. Lisjak A, Grasselli G, Vietor T. Continuum–discontinuum analysis of failure mechanisms around unsupported circular excavations in anisotropic clay shales. *International Journal of Rock Mechanics and Mining Sciences.* 2014;65:96-115.

17. Mahabadi O, Kaifosh P, Marschall P, Vietor T. Three-dimensional FDEM numerical simulation of failure processes observed in Opalinus Clay laboratory samples. *Journal of Rock Mechanics and Geotechnical Engineering.* 2014;6(6):591-606.

18. Tatone BSA, Grasselli G. A calibration procedure for two-dimensional laboratory-scale hybrid finite–discrete element simulations. *International Journal of Rock Mechanics and Mining Sciences.* 2015;75:56-72.

19. Lisjak A, Liu Q, Zhao Q, Mahabadi OK, Grasselli G. Numerical simulation of acoustic emission in brittle rocks by two-dimensional finite-discrete element analysis. *Geophysical Journal International.* 2013;195(1):423-443.

20. Mahabadi O, Lisjak A, He L, Tatone B, Kaifosh P, Grasselli G. Development of a new fully-parallel finite-discrete element code: Irazu. Paper presented at: 50th US Rock Mechanics/Geomechanics Symposium2016.

21. Lisjak A, Kaifosh P, He L, Tatone BSA, Mahabadi OK, Grasselli G. A 2D, fully-coupled, hydro-mechanical, FDEM formulation for modelling fracturing processes in discontinuous, porous rock masses. *Computers and Geotechnics.* 2017;81:1-18.

22. Lisjak A, Mahabadi OK, He L, et al. Acceleration of a 2D/3D finite-discrete element code for geomechanical simulations using General Purpose GPU computing. *Computers and Geotechnics.* 2018;100:84-96.
23. Xiang J, Latham J-P, Farsi A. Algorithms and Capabilities of Solidity to Simulate Interactions and Packing of Complex Shapes. 2016; Singapore.
24. Latham J, Guo L, Wang X, Xiang J. Modelling the evolution of fractures using a combined FEM-DEM numerical method. 2012.
25. Guo L, Xiang J, Latham J-P, Izzuddin B. A numerical investigation of mesh sensitivity for a new three-dimensional fracture model within the combined finite-discrete element method. *Engineering Fracture Mechanics.* 2016;151:70-91.
26. Lei Q, Latham J-P, Xiang J. Implementation of an Empirical Joint Constitutive Model into Finite-Discrete Element Analysis of the Geomechanical Behaviour of Fractured Rocks. *Rock Mechanics and Rock Engineering.* 2016;49(12):4799-4816.
27. Guo L, Latham J-P, Xiang J. A numerical study of fracture spacing and through-going fracture formation in layered rocks. *International Journal of Solids and Structures.* 2017;110:44-57.
28. Solidity. Solidity Project. 2017; http://solidityproject.com/.
29. Guo L. *Development of a three-dimensional fracture model for the combined finite-discrete element method*, Imperial College London; 2014.
30. Lei Z, Rougier E, Knight EE, Munjiza A. A framework for grand scale parallelization of the combined finite discrete element method in 2d. *Computational Particle Mechanics.* 2014;1(3):307-319.
31. Rougier E, Knight EE, Broome ST, Sussman AJ, Munjiza A. Validation of a three-dimensional Finite-Discrete Element Method using experimental results of the Split Hopkinson Pressure Bar test. *International Journal of Rock Mechanics and Mining Sciences.* 2014;70:101-108.
32. Yan C, Zheng H. A coupled thermo-mechanical model based on the combined finite-discrete element method for simulating thermal cracking of rock. *International Journal of Rock Mechanics and Mining Sciences.* 2017;91:170-178.
33. Yan C, Zheng H. FDEM-flow3D: A 3D hydro-mechanical coupled model considering the pore seepage of rock matrix for simulating three-dimensional hydraulic fracturing. *Computers and Geotechnics.* 2017;81:212-228.
34. Yan C, Jiao Y-Y. A 2D fully coupled hydro-mechanical finite-discrete element model with real pore seepage for simulating the deformation and fracture of porous medium driven by fluid. *Computers & Structures.* 2018;196:311-326.
35. Yan C, Jiao Y-Y, Zheng H. A fully coupled three-dimensional hydro-mechanical finite discrete element approach with real porous seepage for simulating 3D hydraulic fracturing. *Computers and Geotechnics.* 2018;96:73-89.
36. Yan C, Jiao Y-Y, Yang S. A 2D coupled hydro-thermal model for the combined finite-discrete element method. *Acta Geotechnica.* 2018.
37. Liu HY, Kang YM, Lin P. Hybrid finite–discrete element modeling of geomaterials fracture and fragment muck-piling. *International Journal of Geotechnical Engineering.* 2015;9(2):115-131.
38. Liu HY, Han H, An HM, Shi JJ. Hybrid finite-discrete element modelling of asperity degradation and gouge grinding during direct shearing of rough

rock joints. *International Journal of Coal Science & Technology.* 2016;3(3):295-310.

39. An HM, Liu HY, Han H, Zheng X, Wang XG. Hybrid finite-discrete element modelling of dynamic fracture and resultant fragment casting and muck-piling by rock blast. *Computers and Geotechnics.* 2017;81:322-345.

40. Mohammadnejad M, Liu H, Dehkhoda S, Chan A. Numerical Investigation of Dynamic Rock Fragmentation in Mechanical Cutting Using Combined FEM/DEM. 3rd Nordic Rock Mechanics Symposium - NRMS 2017; 2017/11/3/, 2017; Helsinki, Finland.

41. Richard T. *Determination of rock strength from cutting tests. Master's thesis*: University of Minnesota, Minneapolis, MN, USA Google Scholar1999.

42. Zhou Y, Lin J-S. Modeling the ductile–brittle failure mode transition in rock cutting. *Engineering Fracture Mechanics.* 2014;127:135-147.

43. Guo N, Zhao J. Parallel hierarchical multiscale modelling of hydro-mechanical problems for saturated granular soils. *Computer Methods in Applied Mechanics and Engineering.* 2016;305:37-61.

44. Guo N, Zhao J. A coupled FEM/DEM approach for hierarchical multiscale modelling of granular media. *International Journal for Numerical Methods in Engineering.* 2014;99(11):789-818.

45. Wu H, Guo N, Zhao J. Multiscale modeling and analysis of compaction bands in high-porosity sandstones. *Acta Geotechnica.* 2018;13(3):575-599.

46. Wu H, Zhao J, Guo N. Multiscale Insights Into Borehole Instabilities in High-Porosity Sandstones. *Journal of Geophysical Research: Solid Earth.* 2018;123(5):3450-3473.

47. Zhou Y, Lin J-S. On the critical failure mode transition depth for rock cutting. *International Journal of Rock Mechanics and Mining Sciences.* 2013;62:131-137.

48. NVIDIA. Cuda C Programming Guide. 2018; http://docs.nvidia.com/cuda/.

49. Zhang L, Quigley SF, Chan AHC. A fast scalable implementation of the two-dimensional triangular Discrete Element Method on a GPU platform. *Advances in Engineering Software.* 2013;60-61:70-80.

50. Batinić M, Smoljanović H, Munjiza A, Mihanović A. GPU based parallel FDEM for analysis of cable structures. *Građevinar.* 2018;69(12.):1085-1092.

51. Liu Q, Jiang Y, Wu Z, Xu X, Liu Q. Investigation of the Rock Fragmentation Process by a Single TBM Cutter Using a Voronoi Element-Based Numerical Manifold Method. *Rock Mechanics and Rock Engineering.* 2018;51(4):1137-1152.

52. Li X, Zhang Q, Li J, Zhao J. A numerical study of rock scratch tests using the particle-based numerical manifold method. *Tunnelling and Underground Space Technology.* 2018;78:106-114.

53. Liu W, Zhu X, Jing J. The analysis of ductile-brittle failure mode transition in rock cutting. *Journal of Petroleum Science and Engineering.* 2018;163:311-319.

54. Zhang Z, Paulino G, Celes W. Extrinsic cohesive modelling of dynamic fracture and microbranching instability in brittle materials. *International Journal for Numerical Methods in Engineering.* 2007;72(8):893-923.

55. Xianqun H, Chaoshui X. Discrete element modelling of rock cutting: from ductile to brittle transition. *International Journal for Numerical and Analytical Methods in Geomechanics.* 2015;39(12):1331-1351.
56. An B, Tannant DD. Discrete element method contact model for dynamic simulation of inelastic rock impact. *Computers & Geosciences.* 2007;33(4):513-521.
57. Satish N, Harris M, Garland M. Designing efficient sorting algorithms for manycore GPUs. Paper presented at: Parallel & Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on2009.
58. Ayachit U. The paraview guide: a parallel visualization application. 2015.
59. Cundall PA. Distinct element models of rock and soil structure. *Analytical and Computational Methods in Engineering Rock Mechanics.* 1987:129-163.
60. Karekal S. Modeling Rock Chipping Process In Linear Drag Cutting Mode. ISRM International Symposium - EUROCK 2012; 2012/5/28/, 2012; Stockholm, Sweden.
61. Xiang J, Munjiza A, Latham J-P, Guises R. On the validation of DEM and FEM/DEM models in 2D and 3D. *Engineering Computations.* 2009;26(6):673-687.
62. Mahabadi OK. *Investigating the Influence of Micro-scale Heterogeneity and Microstructure on the Failure and Mechanical Behaviour of Geomaterials*: Civil Engineering, University of Toronto; 2012.
63. Lysmer J, Kuhlemeyer RL, Institute of T, et al. *Finite dynamic model for infinite media.* Berkeley, Calif.: Dept. of Civil Engineering, Univ. of California, Institute of Transportation and Traffic Engineering, Soil Mechanics Laboratory; 1969.
64. Cho SH, Kaneko K. Influence of the applied pressure waveform on the dynamic fracture processes in rock. *International Journal of Rock Mechanics and Mining Sciences.* 2004;41(5):771-784.
65. Benz W, Asphaug E. Simulations of brittle solids using smooth particle hydrodynamics. *Computer physics communications.* 1995;87(1-2):253-265.

Table Titles

Table 1. Rock mechanical properties and numerical parameters

Table 2. Model details for several h_{ave} values.

Figure Captions

Fig. 1 Modeling of transition from continuum to discontinuum behavior of rock in 2D Y-HFDEM IDE. (a) Assembly of TRI3s and CE4s and (b) tensile/shear softening curves in ICZM.

Fig. 2. Elastic–inelastic power function model implemented in 2D Y-HFDEM IDE (after [56] with modification).

Fig. 3. The concept of the CUDA programming model using the abstractions of "*threads*", "*blocks*" and "*grid*".

Fig. 4. The concept of massively parallel computation for each CUDA "*kernel*" for a particular purpose.

Fig. 5. Flowchart of GPGPU-based 2D Y-HFDEM IDE.

Fig. 6. The concept of hash values assigned to each square subcell (left) and TRI3s included in a particular subcell (right).

Fig. 7. Efficient contact detection algorithm used in the developed GPGPU-based 2D Y-HFDEM IDE.

Fig. 8. A model of initial stress analysis under gravity.

Fig. 9. The result of initial stress analysis using critical and local damping schemes. (a) Stress distribution computed and (b) the profiles of σ_{xx} and σ_{yy} along the vertical direction of the model from the top to the bottom.

Fig. 10. Comparison of change in total kinetic energy of the system with respect to calculation time-steps between local and critical damping schemes.

Fig. 11. A simple test for verification of contact damping model. (a) Numerical model similar to [13] and (b) time history of total kinetic energy.

Fig. 12. Contact friction verification: (a) model configuration, (b) comparison between numerical and theoretical results.

Fig. 13. Verification models for fracture process of rock under quasi-static loading. (a) Brazilian indirect tensile test and (b) uniaxial compression test.

Fig. 14. The results of the numerical simulation of the Brazilian test. (a) Extension of shear microcracks before the peak stress and corresponding horizontal stress distribution, (b) tensile microcracks at the peak stress and corresponding horizontal stress distribution, (c) postfailure fracture pattern and corresponding horizontal stress distribution and (d) Brazilian indirect tensile stress versus axial strain.

Fig. 15. Fracture process in uniaxial compression test. (a) Initiation and propagation of shear microcracks into model before the peak stress, (b) unstable crack propagation at the peak stress and (c) postfailure fracture pattern.

Fig. 16. Comparison of numerical simulation and experiment for uniaxial compression test. (a) Plot of axial stress versus axial strain and (b) final fracture patterns from numerical simulation and experimental test.

Fig. 17. A numerical model with a single blast-hole used in the dynamic fracture process analysis (after [64]).

Fig. 18. The results of dynamic fracture process analysis for a single blast-hole model. (a) Spatial distribution of PS1 and broken CE4s at selected time intervals and (b) the results of dynamic fracture process analysis using different penalty numbers for CE4s.

Fig. 19. Relative speed-up of the GPGPU-based code using different GPGPU accelerators to the sequential CPU-based code.

Fig. 20. Change in relative speed-up with respect to the number of TRI3s used in FDEM mesh for the simulation of uniaxial compression test.

Fig. A.1. An example of GPGPU "*kernel*" in 2D Y-HFDEM IDE.

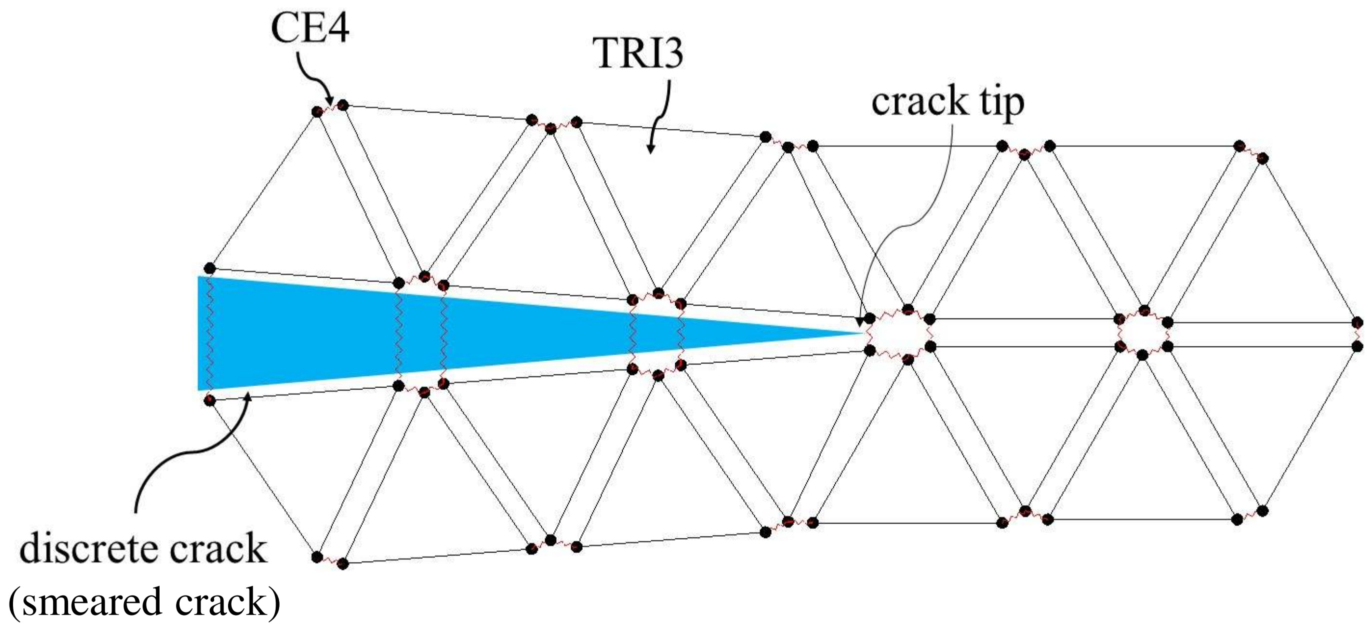
Table 1

Parameter	***Unit***	***Value***
Density (ρ)	kg/m3	1800
Young's modulus (E)	GPa	12.2
Poisson's ratio (v)	-	0.25
Tensile strength (T_s)	MPa	1.77
Cohesion (c)	MPa	5
Internal Friction angle of intact rock (φ)	°	25
Mode I fracture energy (G_{fI})	J/m^2	16
Mode II fracture energy (G_{fII})	J/m^2	160
Normal contact penalty number (P_{n_con})	GPa	1220

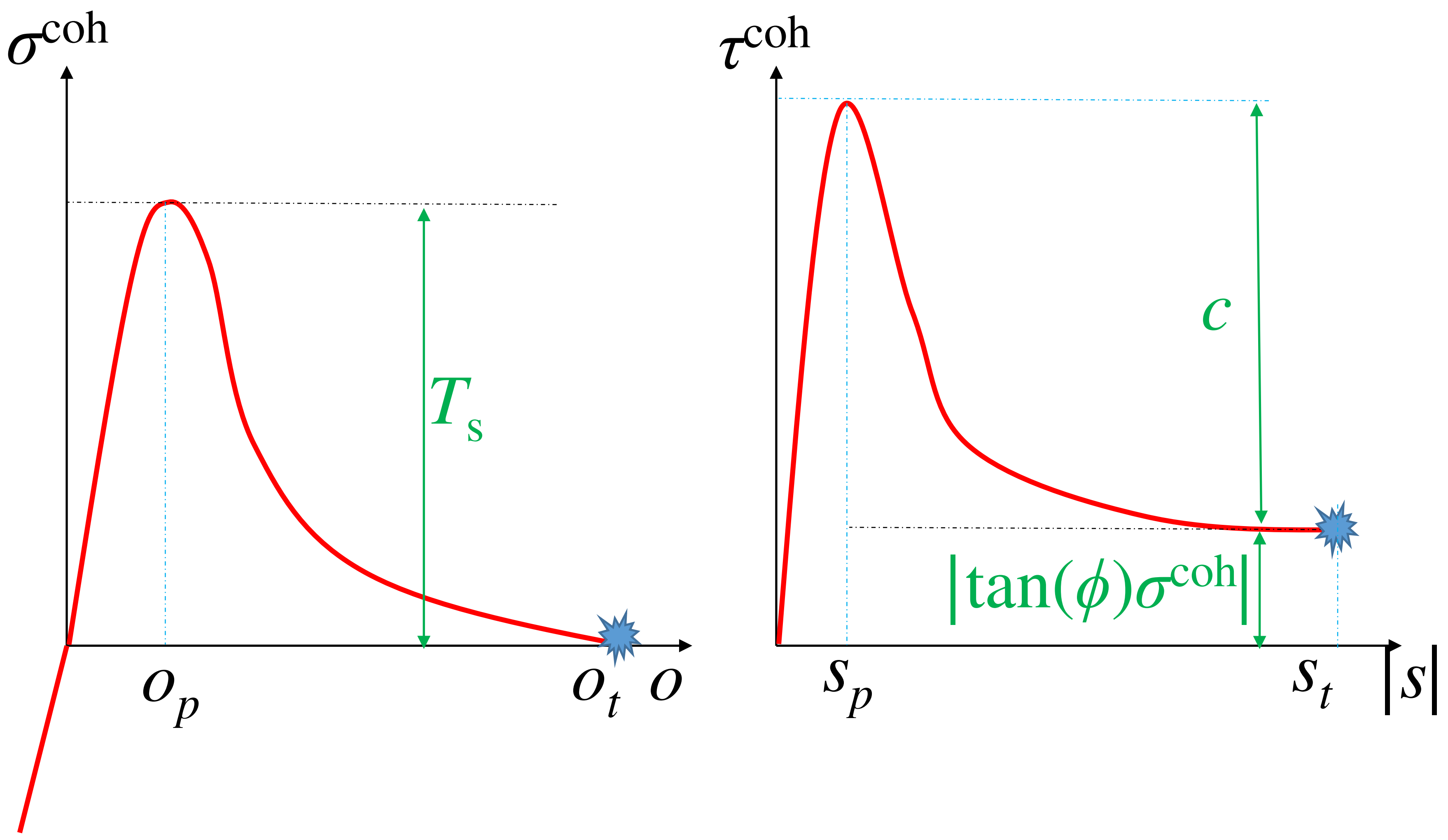
Tangent contact penalty number ($P_{\mathrm{tan\_con}}$)	GPa/m	1220
Fracture penalty numbers (P_{f}, P_{tan}, P_{overlap})	GPa	12200
Average element size (h_{ave})	mm	0.7
Critical viscous damping factor (η)	kg/m.s	5.60E+03

Table 2

h_{ave} / (mm)	0.15	0.2	0.3	0.4	0.5	0.6
The number of nodes	3,253,194	1,893,901	723,894	410,322	303,537	181,038
The number of TRI3s	1,084,398	540,754	241,298	136,774	86,570	60,346
The number of CE4s	1,624,686	809,872	306,990	204,355	129,316	90,040
The initial number of contact couples.	6,970,705	4,348,700	1,943,205	1,096,737	686,954	476,433
Simulation time (sec) for sequential CPU-based code /10000 steps	108,767	42,021	19,343	10,515	9,146	6,231
Simulation time (sec) for GPUGU-based code (Quadro GP100) /10000 steps	846	392	214	146	134	126

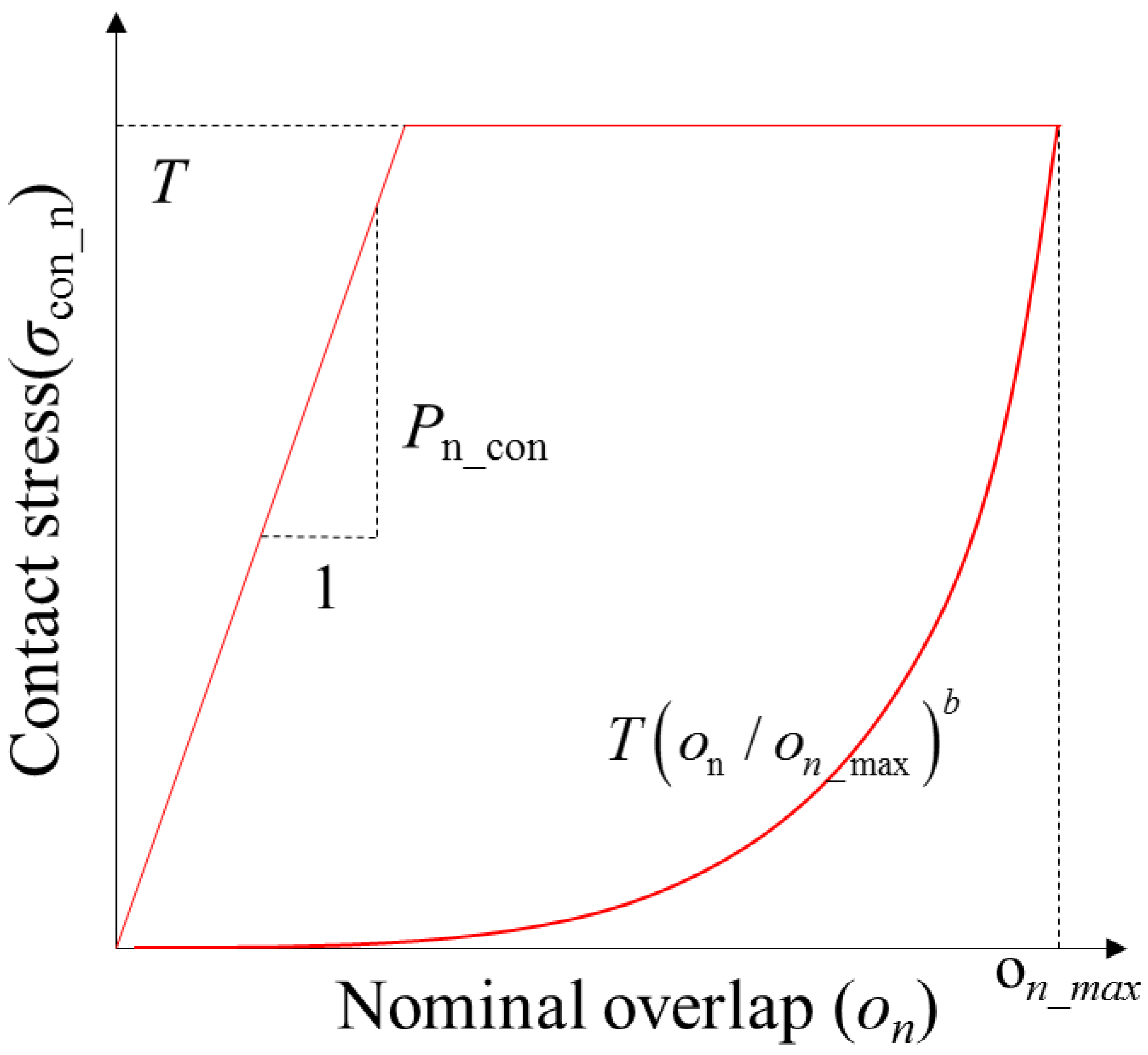

CE4
TRI3
crack tip
discrete crack
(smeared crack)

(a)


σ^{coh}
T_s
o_p
o_t
o
τ^{coh}
c
$|\tan(\phi)\sigma^{\mathrm{coh}}|$
s_p
s_t
$|s|$

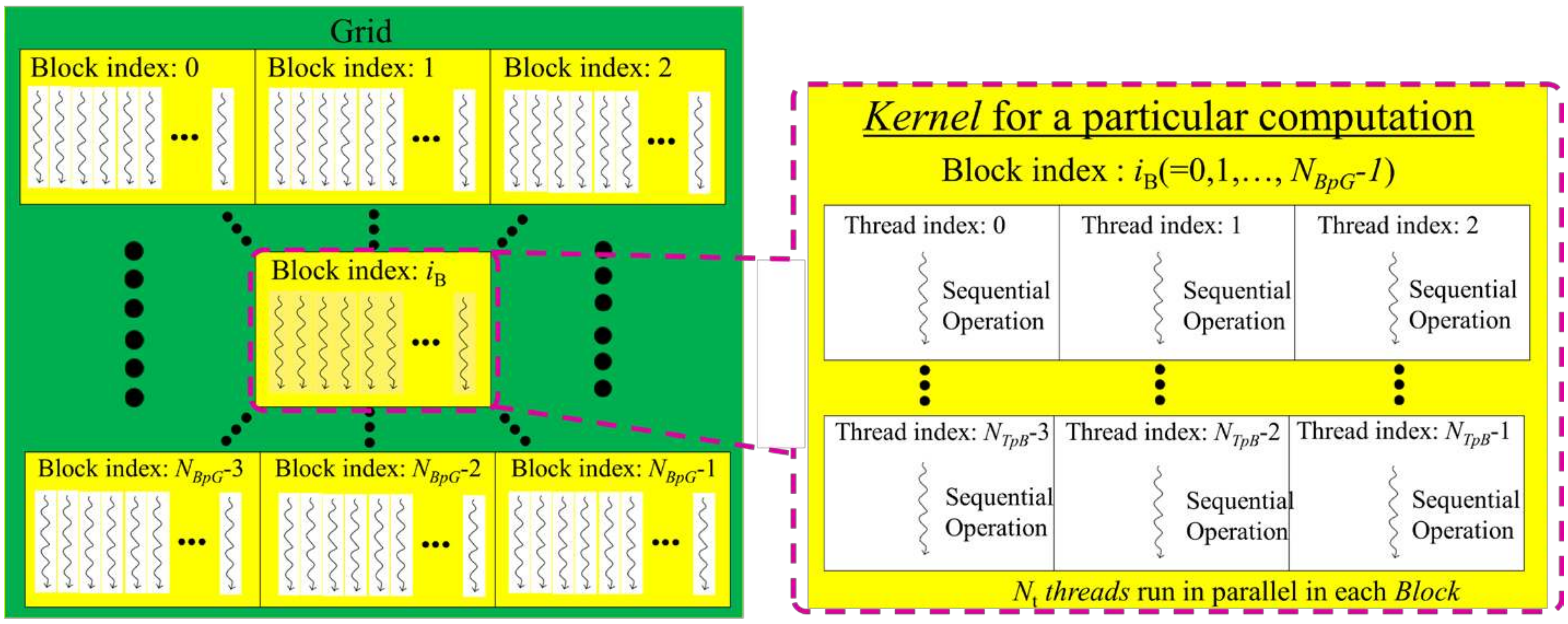
(b)

Fig. 1.



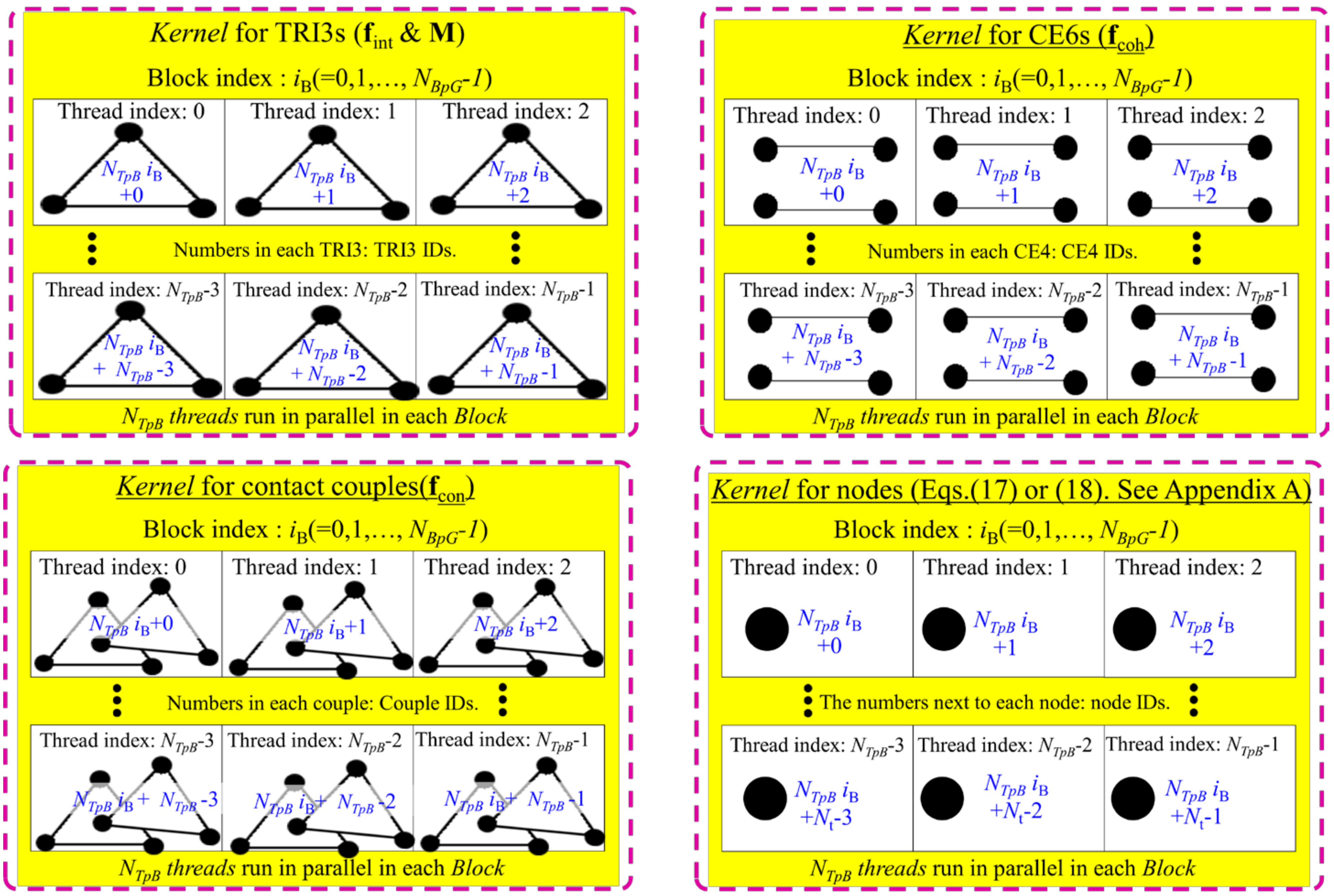
Contact stress($\sigma_{\mathrm{con\_n}}$)
T
$P_{\mathrm{n\_con}}$
1
$T\left(o_{\mathrm{n}} / o_{n\_\mathrm{max}}\right)^{b}$
Nominal overlap (o_n)
o_{n_max}

Fig. 2.



Grid
Block index: 0
Block index: 1
Block index: 2
Block index: i_B
Block index: N_{BpG}-3
Block index: N_{BpG}-2
Block index: N_{BpG}-1
Kernel for a particular computation
Block index : i_B(=0,1,…, N_{BpG}-*1*)
Thread index: 0
Thread index: 1
Thread index: 2
Sequential Operation
Thread index: N_{TpB}-3
Thread index: N_{TpB}-2
Thread index: N_{TpB}-1
N_t *threads* run in parallel in each *Block*

Fig. 3.



Kernel for TRI3s ($\mathbf{f}_{int}$ & $\mathbf{M}$)
Block index : i_B(=0,1,…, N_{BpG}-1)
Thread index: 0
Thread index: 1
Thread index: 2
N_{TpB} i_B +0
N_{TpB} i_B +1
N_{TpB} i_B +2
Numbers in each TRI3: TRI3 IDs.
Thread index: N_{TpB}-3
Thread index: N_{TpB}-2
Thread index: N_{TpB}-1
N_{TpB} i_B + N_{TpB} -3
N_{TpB} i_B + N_{TpB} -2
N_{TpB} i_B + N_{TpB} -1
N_{TpB} *threads* run in parallel in each *Block*
Kernel for CE6s ($\mathbf{f}_{coh}$)
Block index : i_B(=0,1,…, N_{BpG}-1)
Numbers in each CE4: CE4 IDs.
Kernel for contact couples($\mathbf{f}_{con}$)
Block index : i_B(=0,1,…, N_{BpG}-1)
Numbers in each couple: Couple IDs.
Kernel for nodes (Eqs.(17) or (18). See Appendix A)
Block index : i_B(=0,1,…, N_{BpG}-1)
The numbers next to each node: node IDs.
N_{TpB} i_B +N_t-3
N_{TpB} i_B +N_t-2
N_{TpB} i_B +N_t-1

Fig. 4.

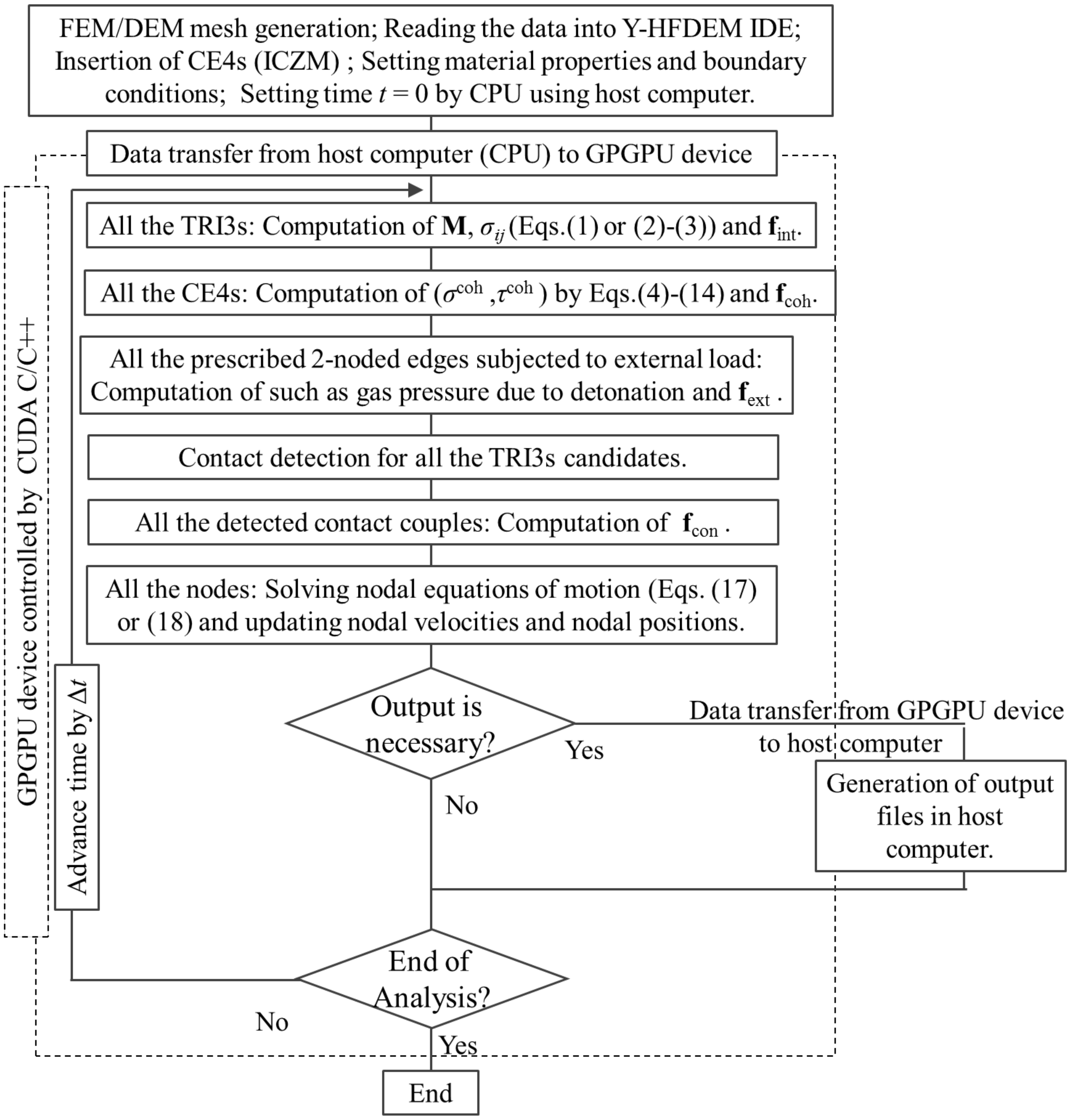
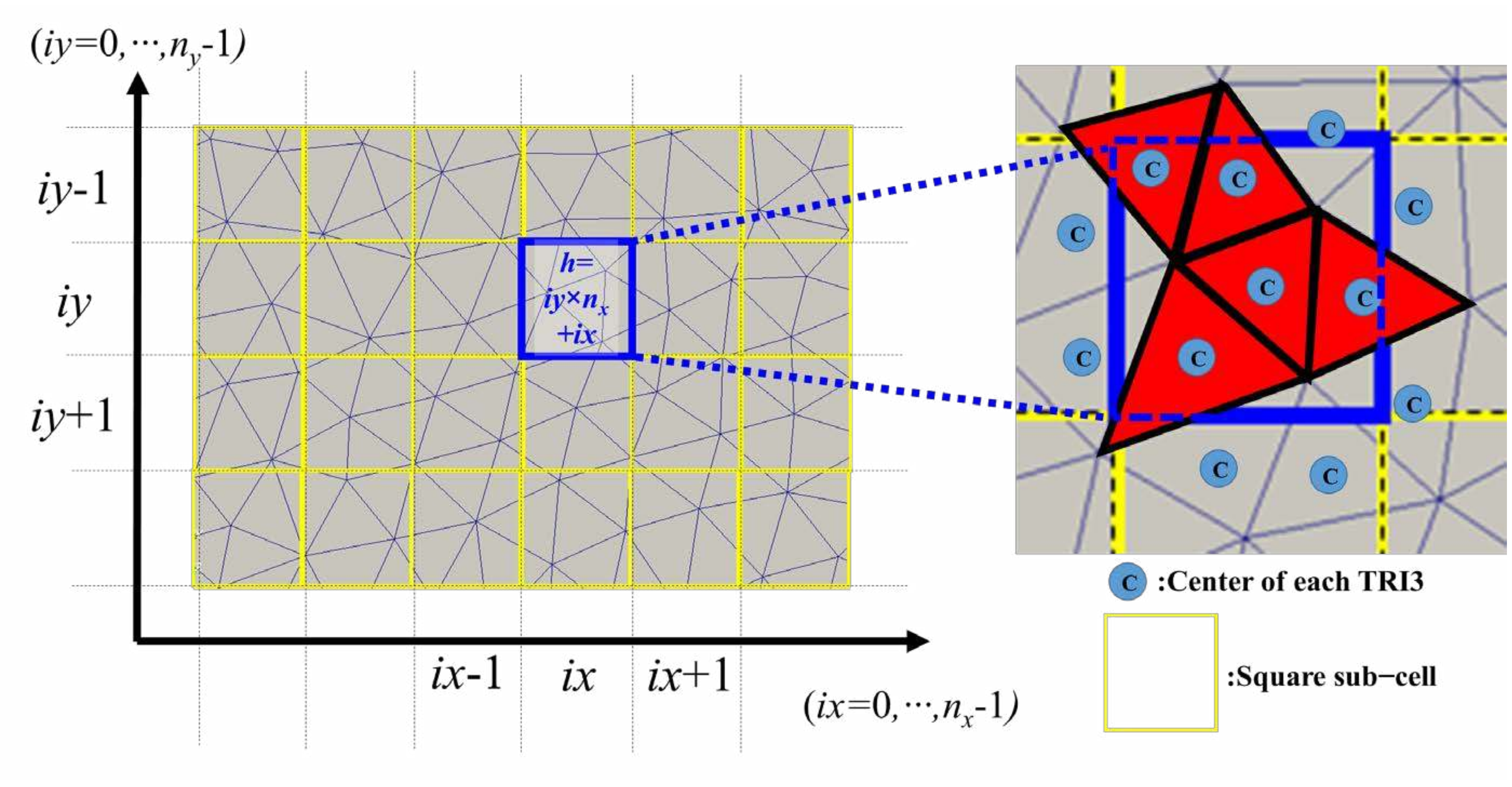
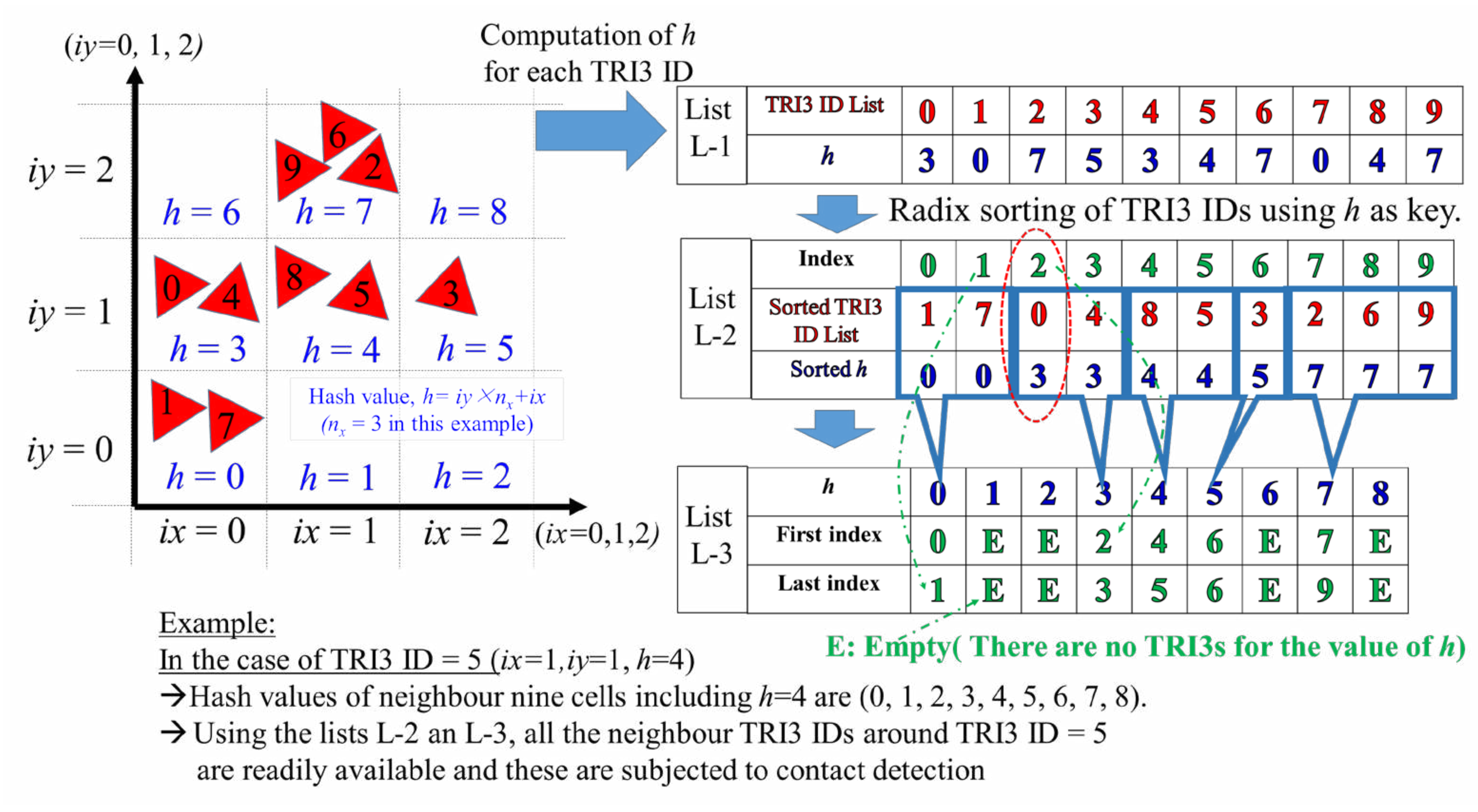

FEM/DEM mesh generation; Reading the data into Y-HFDEM IDE; Insertion of CE4s (ICZM) ; Setting material properties and boundary conditions; Setting time $t = 0$ by CPU using host computer.
Data transfer from host computer (CPU) to GPGPU device
All the TRI3s: Computation of $\mathbf{M}$, σ_{ij} (Eqs.(1) or (2)-(3)) and $\mathbf{f}_{\text{int}}$.
All the CE4s: Computation of (σ^{coh}, τ^{coh}) by Eqs.(4)-(14) and $\mathbf{f}_{\text{coh}}$.
All the prescribed 2-noded edges subjected to external load: Computation of such as gas pressure due to detonation and $\mathbf{f}_{\text{ext}}$.
Contact detection for all the TRI3s candidates.
All the detected contact couples: Computation of $\mathbf{f}_{\text{con}}$.
All the nodes: Solving nodal equations of motion (Eqs. (17) or (18) and updating nodal velocities and nodal positions.
Output is necessary?
Yes
Data transfer from GPGPU device to host computer
Generation of output files in host computer.
No
End of Analysis?
No
Yes
End
Advance time by Δt
GPGPU device controlled by CUDA C/C++

Fig. 5.



$(iy=0,\cdots,n_y\text{-}1)$
iy-1
iy
iy+1
$h=$ $iy\times n_x$ $+ix$
ix-1
ix
ix+1
$(ix=0,\cdots,n_x\text{-}1)$
C
C :Center of each TRI3
:Square sub−cell

Fig. 6.



$(iy=0, 1, 2)$
$iy = 2$
$iy = 1$
$iy = 0$
$h = 6$
$h = 7$
$h = 8$
$h = 3$
$h = 4$
$h = 5$
$h = 0$
$h = 1$
$h = 2$
$ix = 0$
$ix = 1$
$ix = 2$
$(ix=0,1,2)$
Hash value, $h = iy \times n_x + ix$
$(n_x = 3$ in this example)
Computation of h for each TRI3 ID
Radix sorting of TRI3 IDs using h as key.
E: Empty(There are no TRI3s for the value of h)

List L-1

TRI3 ID List	0	1	2	3	4	5	6	7	8	9
h	3	0	7	5	3	4	7	0	4	7

List L-2

Index	0	1	2	3	4	5	6	7	8	9
Sorted TRI3 ID List	1	7	0	4	8	5	3	2	6	9
Sorted h	0	0	3	3	4	4	5	7	7	7

List L-3

h	0	1	2	3	4	5	6	7	8
First index	0	E	E	2	4	6	E	7	E
Last index	1	E	E	3	5	6	E	9	E

Example:
In the case of TRI3 ID = 5 (ix=1, iy=1, h=4)
→Hash values of neighbour nine cells including h=4 are (0, 1, 2, 3, 4, 5, 6, 7, 8).
→ Using the lists L-2 an L-3, all the neighbour TRI3 IDs around TRI3 ID = 5
are readily available and these are subjected to contact detection

Fig. 7.

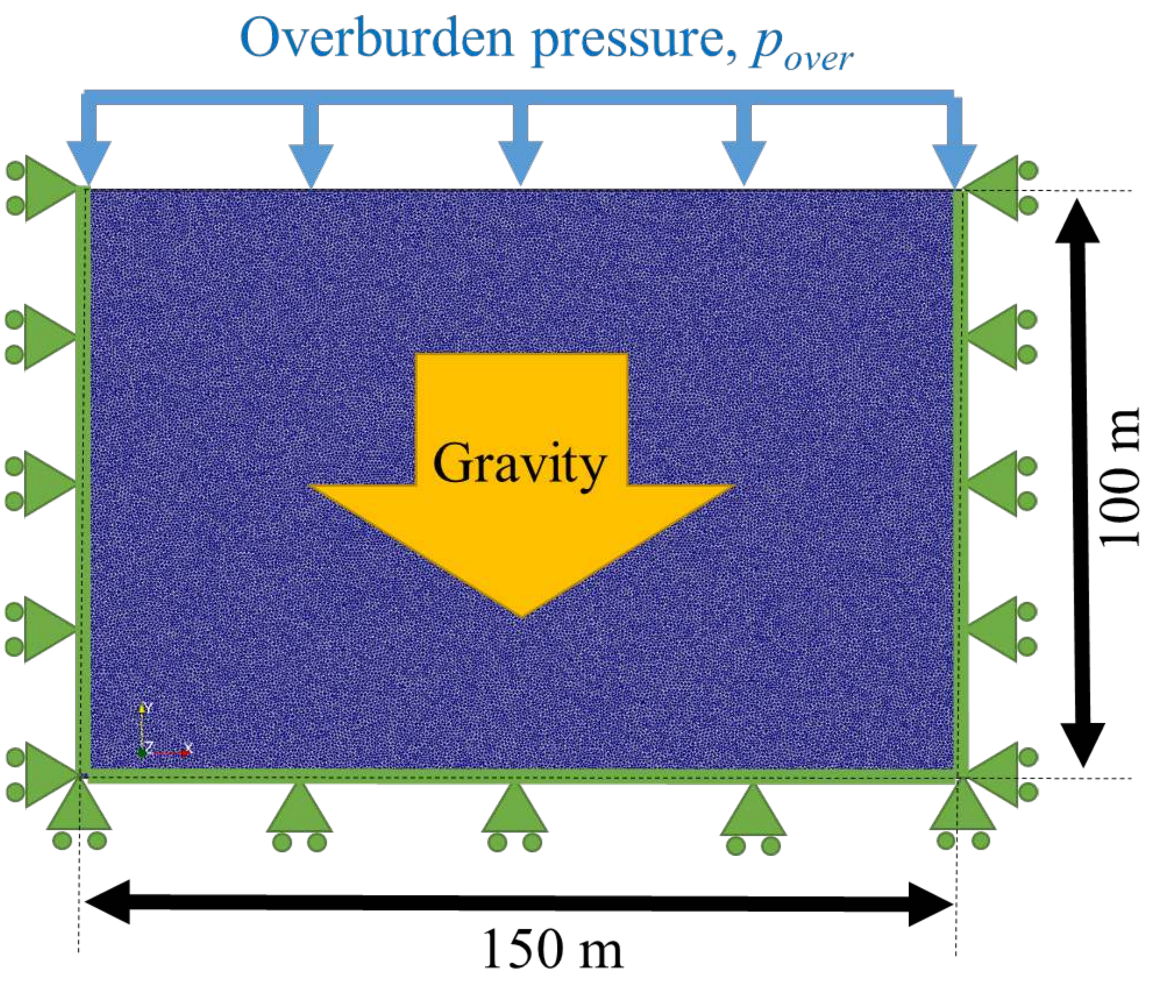
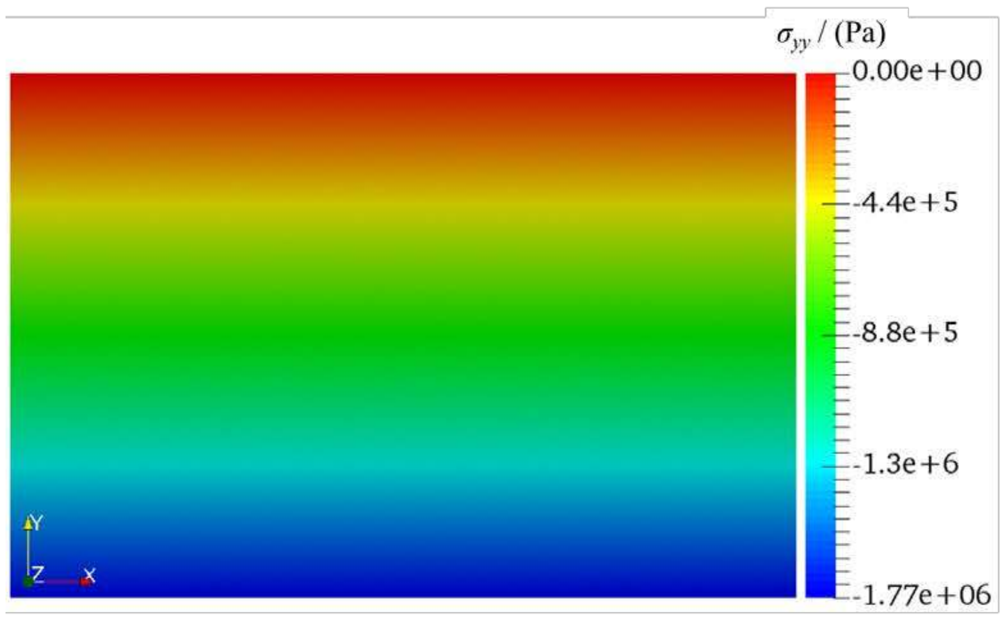
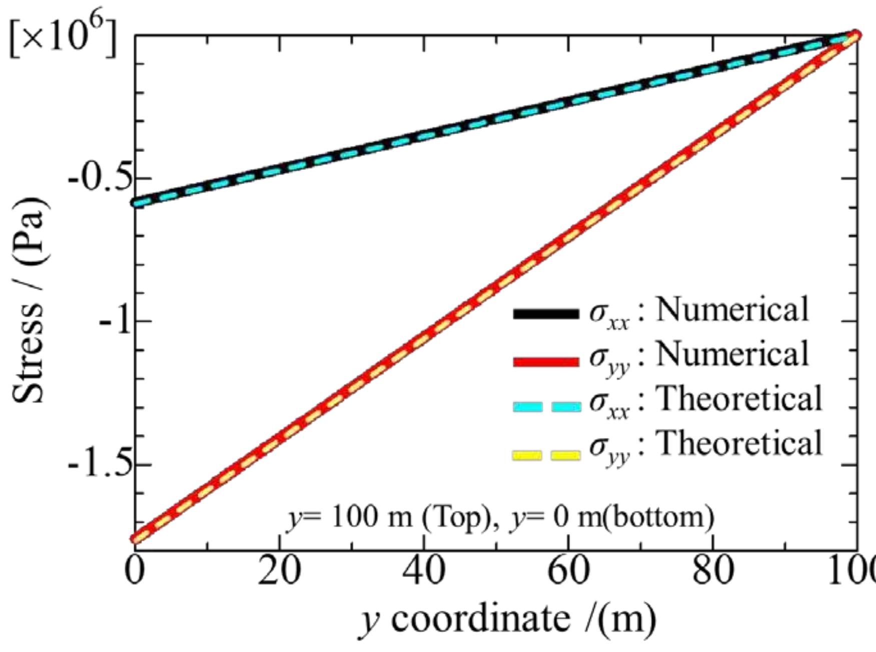


Fig. 8.



σ_{yy} / (Pa)
0.00e+00
-4.4e+5
-8.8e+5
-1.3e+6
-1.77e+06
Y
Z
X

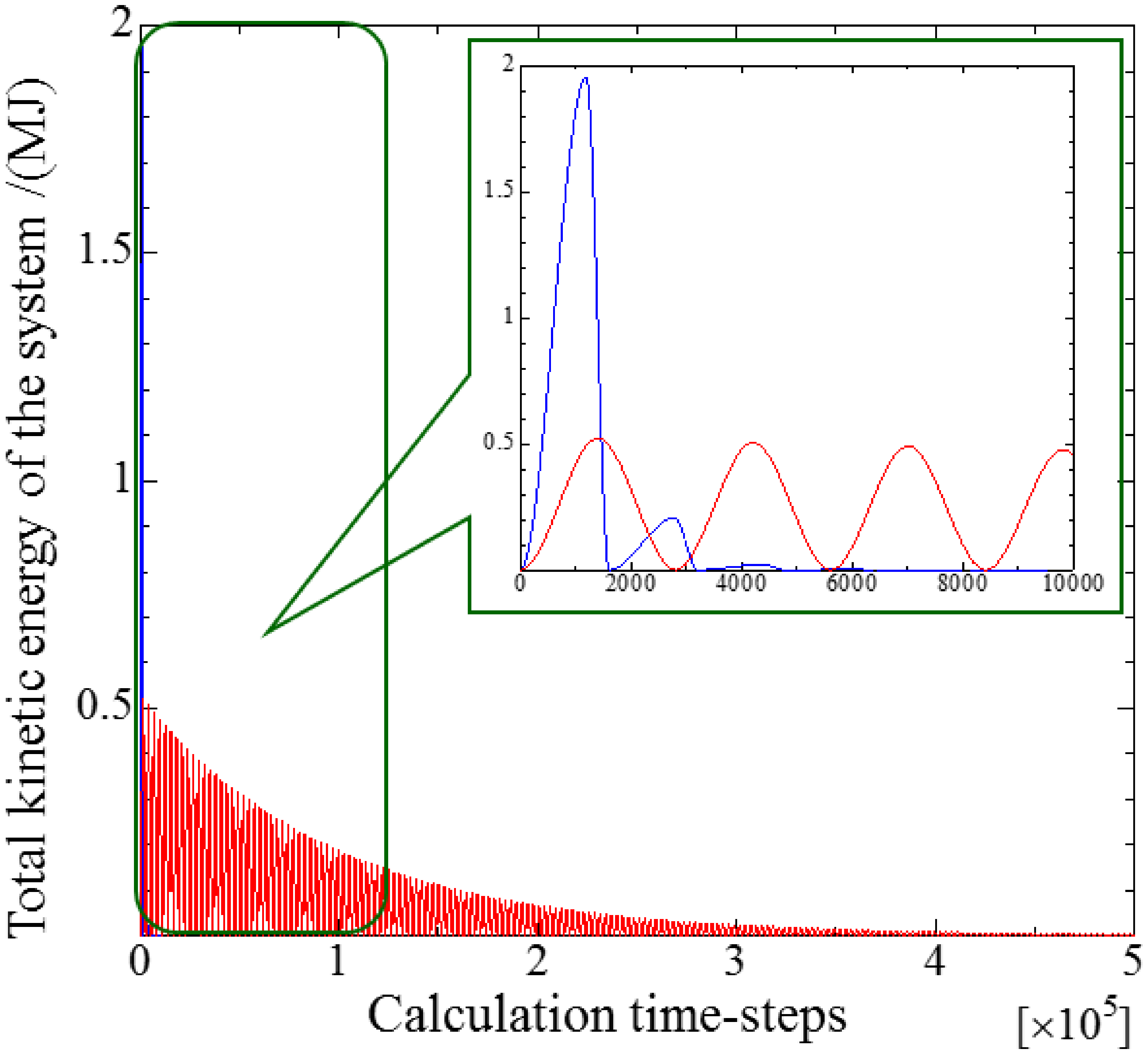
(a)



[×10^6]
Stress / (Pa)
-0.5
-1
-1.5
0
20
40
60
80
100
y coordinate /(m)
σ_{xx} : Numerical
σ_{yy} : Numerical
σ_{xx} : Theoretical
σ_{yy} : Theoretical
y= 100 m (Top), y= 0 m(bottom)

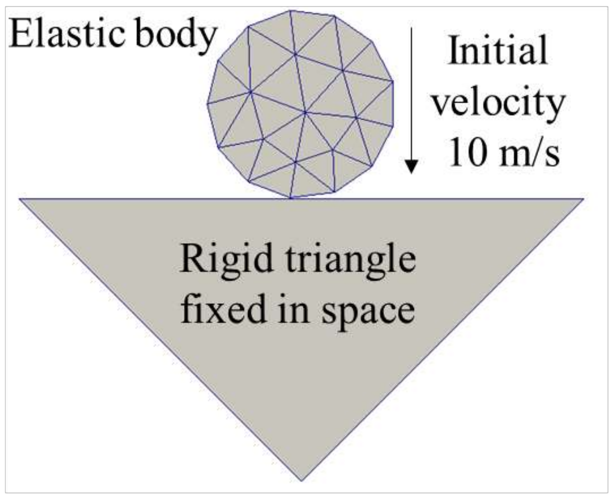
(b)

Fig. 9.

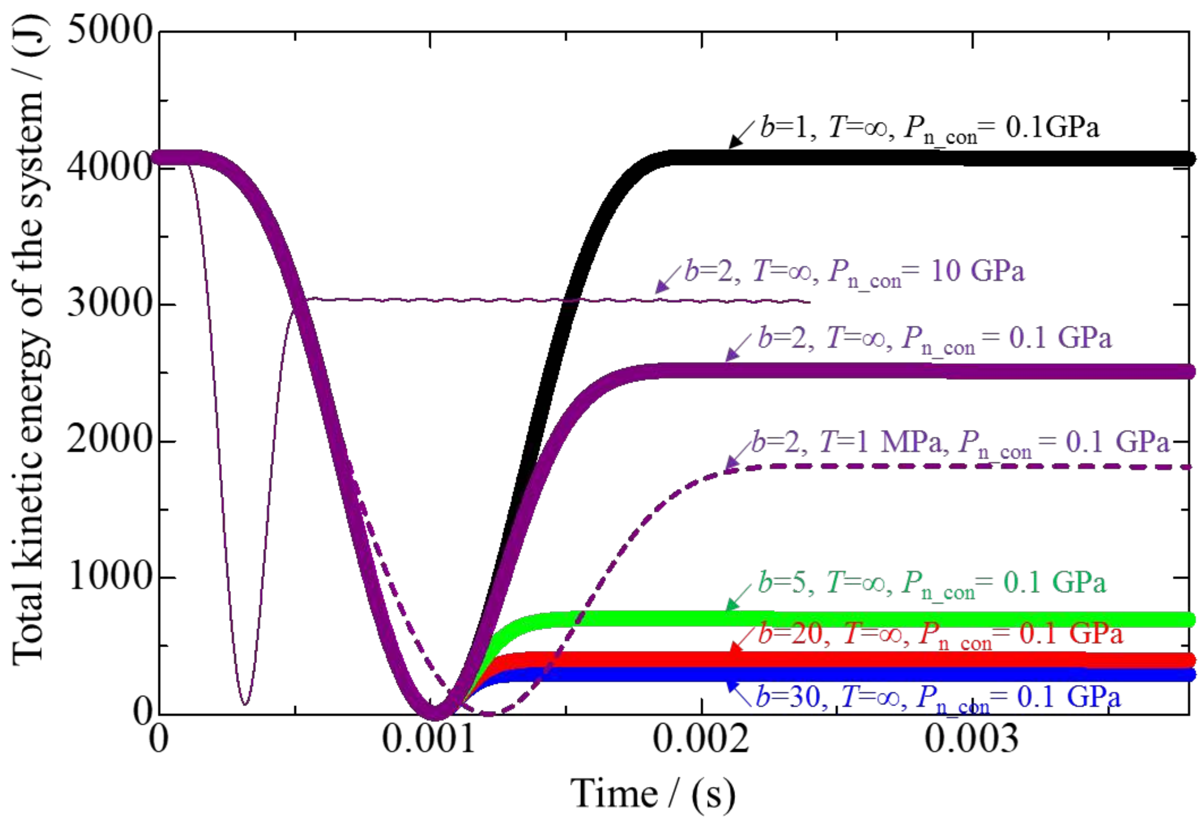


Total kinetic energy of the system /(MJ)
Calculation time-steps
[×10^5]
0
0.5
1
1.5
2
0
1
2
3
4
5
0
2000
4000
6000
8000
10000

Fig. 10.

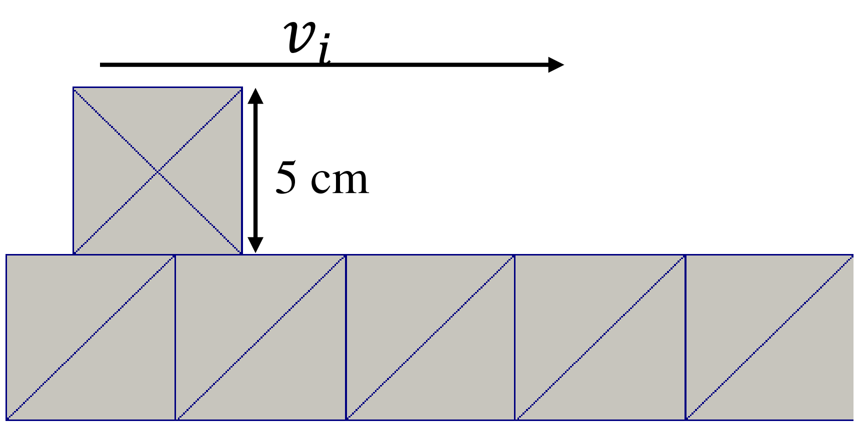

Elastic body
Initial velocity 10 m/s
Rigid triangle fixed in space

(a)


Total kinetic energy of the system / (J)
5000
4000
3000
2000
1000
0
0
0.001
0.002
0.003
Time / (s)
b=1, T=∞, P_n_con= 0.1GPa
b=2, T=∞, P_n_con= 10 GPa
b=2, T=∞, P_n_con = 0.1 GPa
b=2, T=1 MPa, P_n_con = 0.1 GPa
b=5, T=∞, P_n_con= 0.1 GPa
b=20, T=∞, P_n_con= 0.1 GPa
b=30, T=∞, P_n_con= 0.1 GPa

(b)

Fig. 11.


v_i
5 cm
(a)

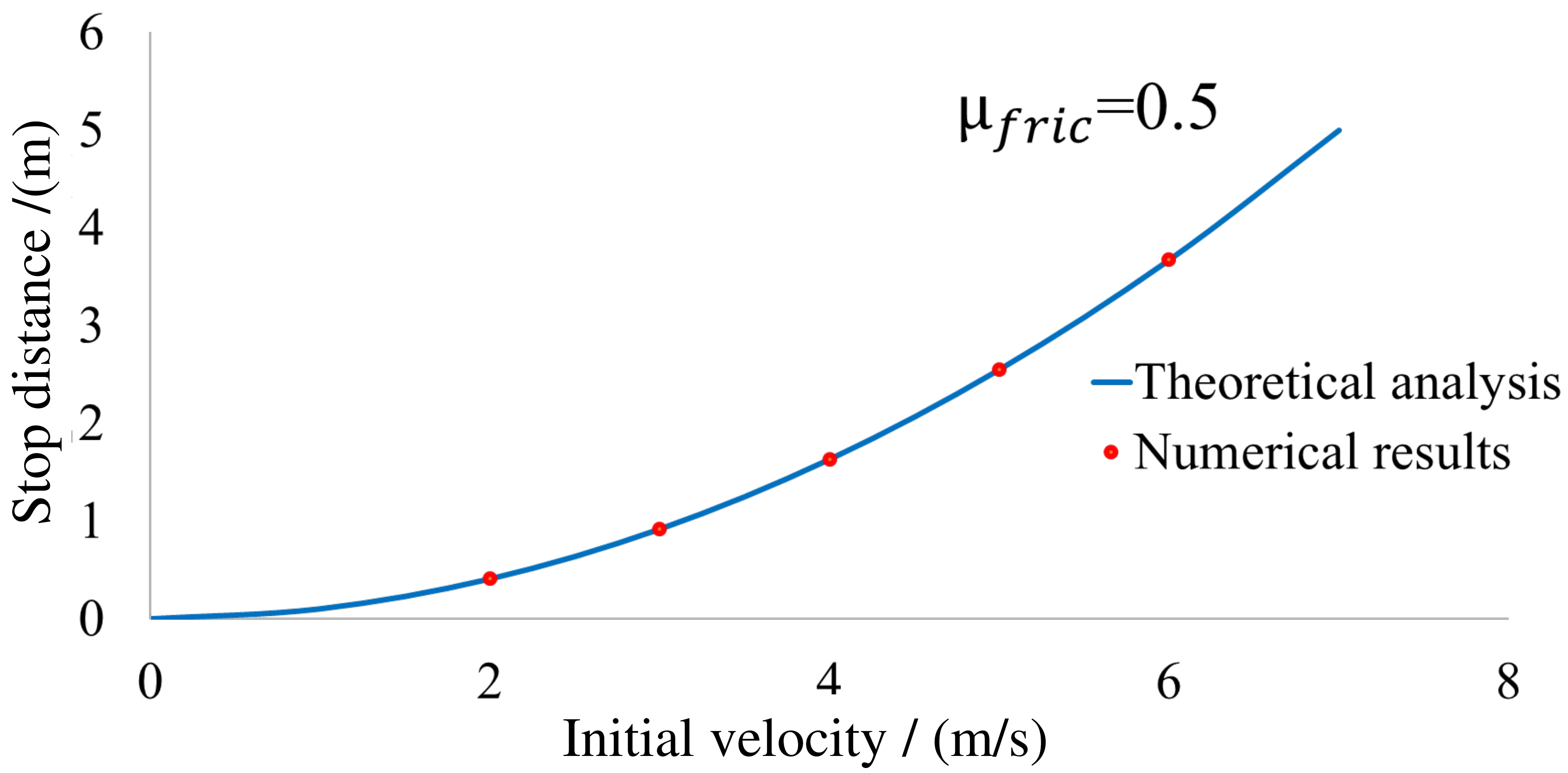

μ_{fric}=0.5
Stop distance /(m)
Initial velocity / (m/s)
Theoretical analysis
Numerical results
0
1
2
3
4
5
6
0
2
4
6
8
(b)

Fig. 12.

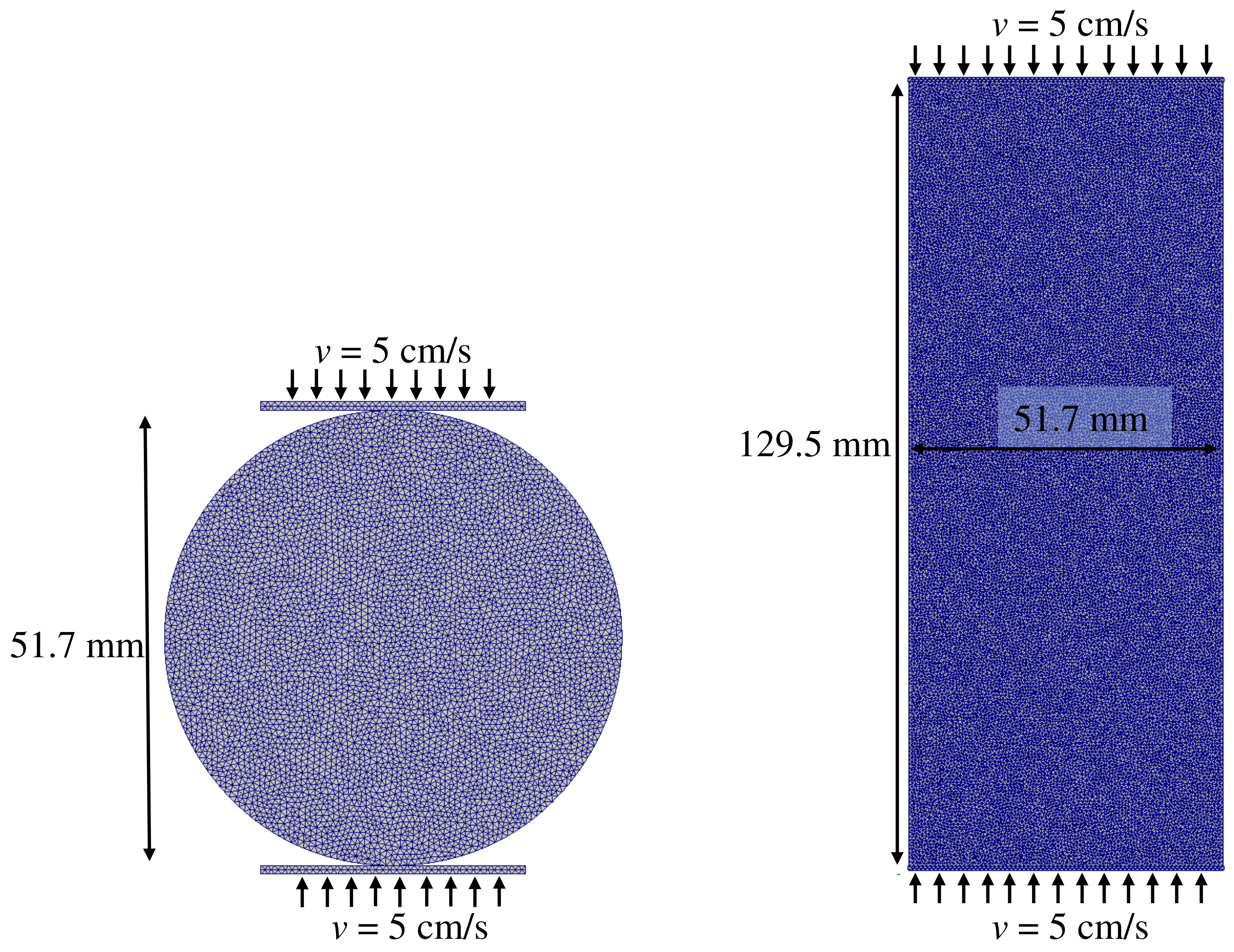


Fig. 13.

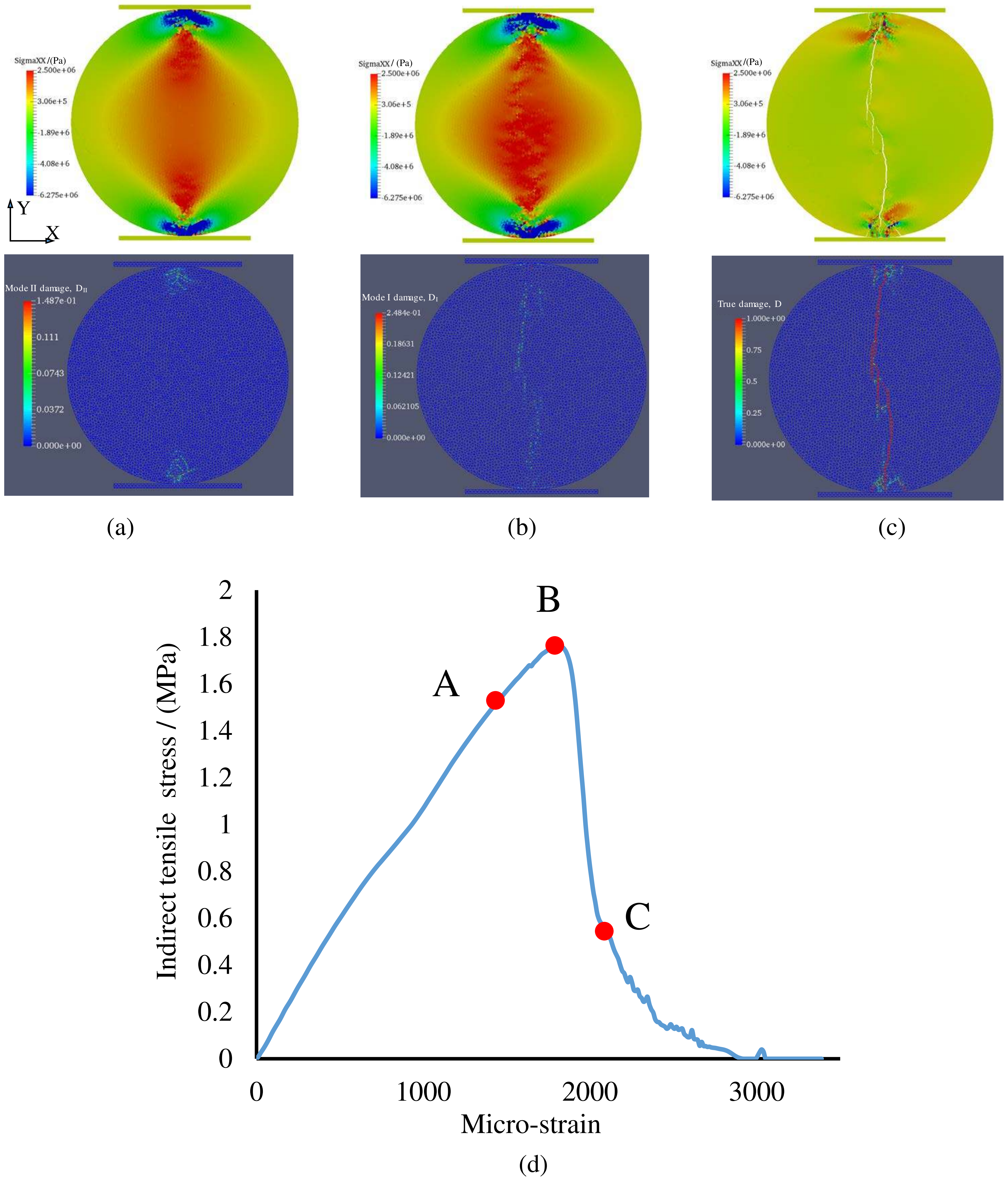
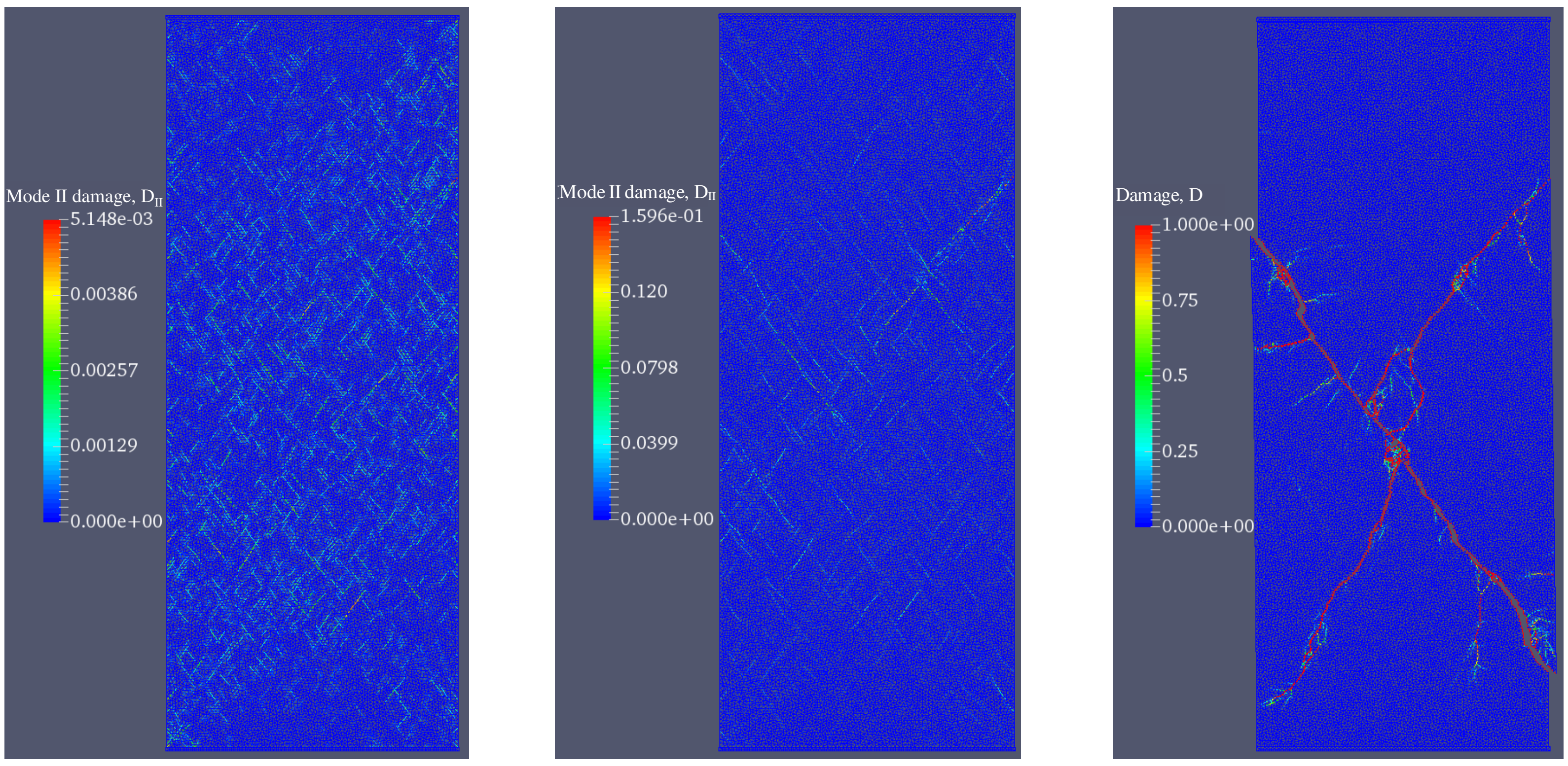
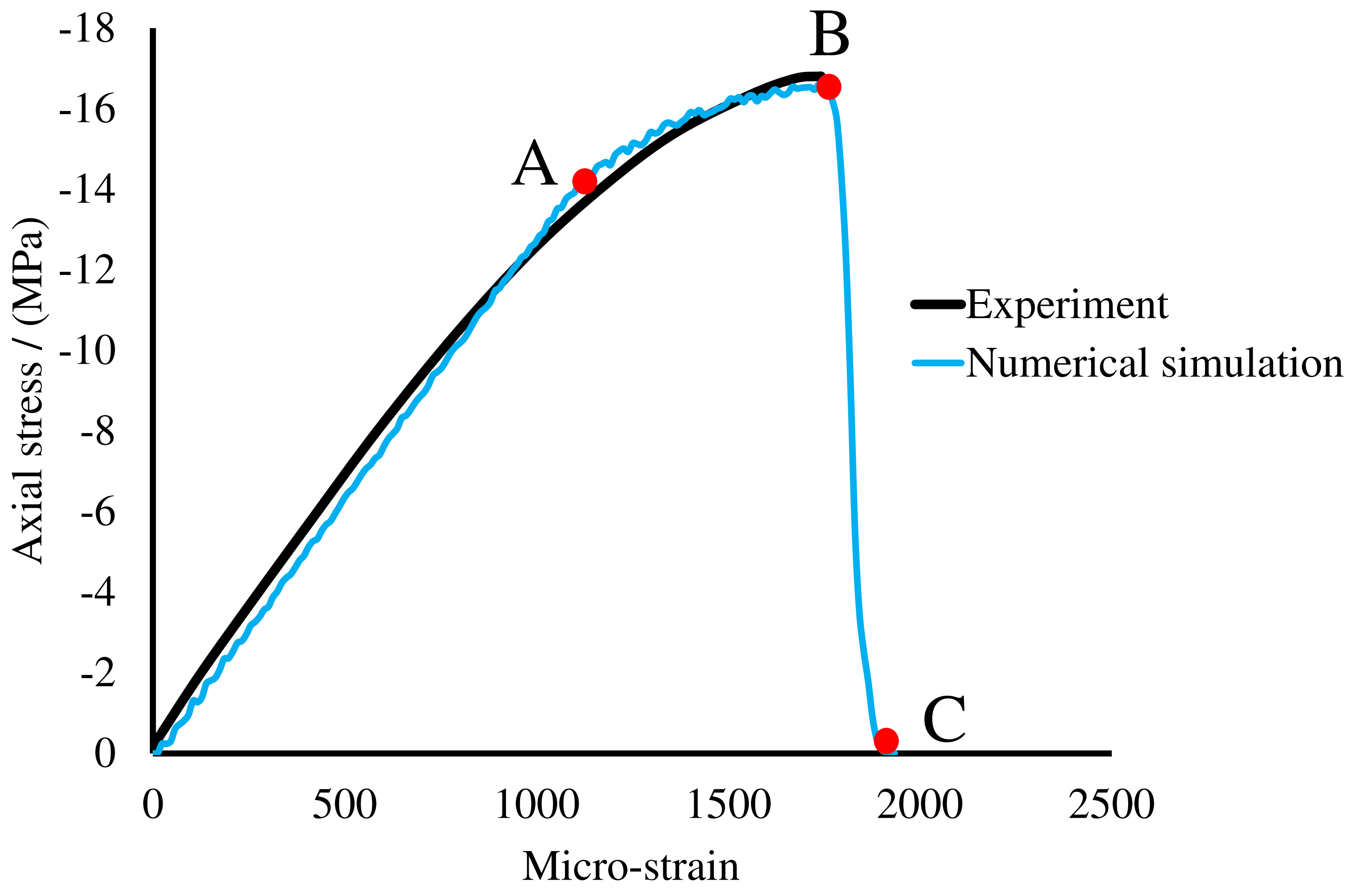

SigmaXX /(Pa)
2.500e+06
3.06e+5
-1.89e+6
-4.08e+6
-6.275e+06
Y
X
Mode II damage, D_{II}
1.487e-01
0.111
0.0743
0.0372
0.000e+00
Mode I damage, D_I
2.484e-01
0.18631
0.12421
0.062105
0.000e+00
True damage, D
1.000e+00
0.75
0.5
0.25
0.000e+00
(a)
(b)
(c)
Indirect tensile stress / (MPa)
Micro-strain
A
B
C
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
1000
2000
3000
(d)

Fig. 14.


Mode II damage, D_{II}
5.148e-03
0.00386
0.00257
0.00129
0.000e+00
Mode II damage, D_{II}
1.596e-01
0.120
0.0798
0.0399
0.000e+00
Damage, D
1.000e+00
0.75
0.5
0.25
0.000e+00

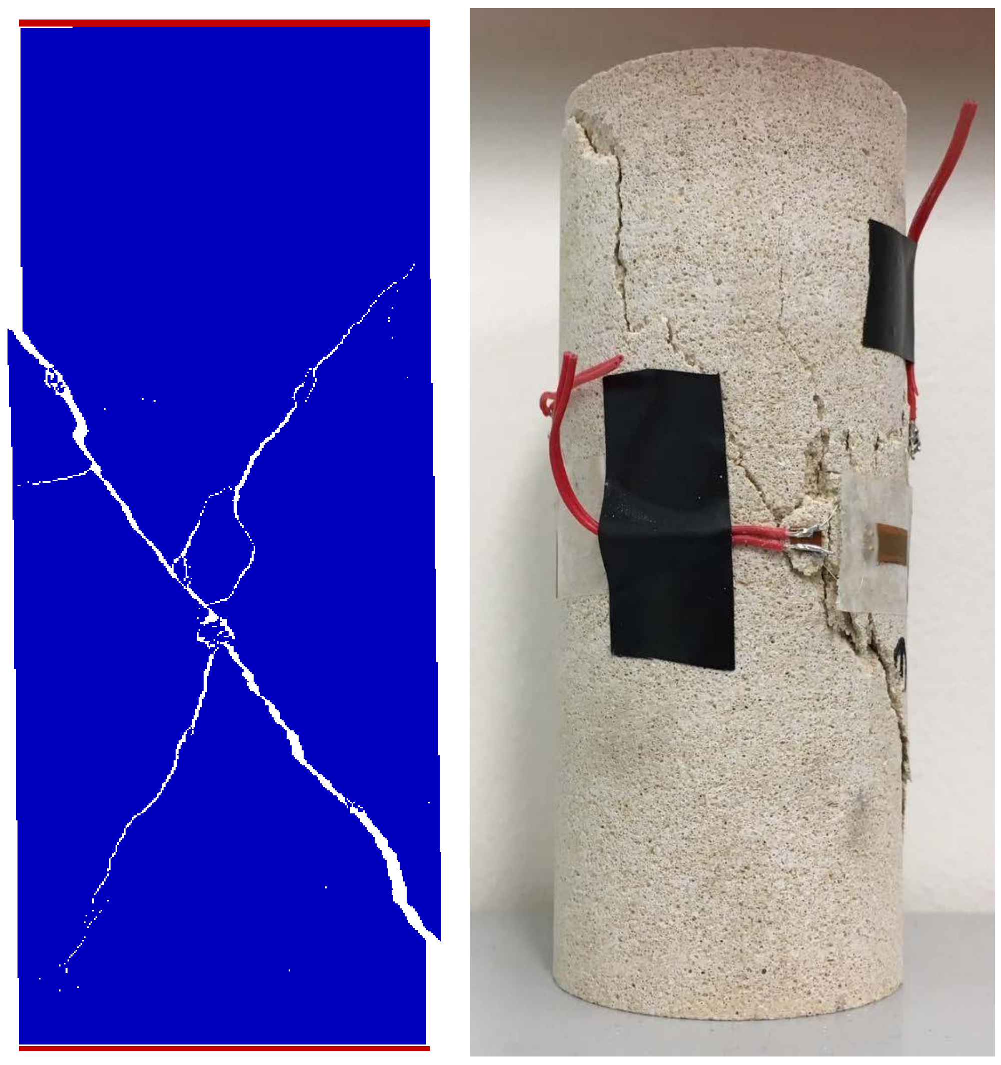
(a) (b) (c)

Fig. 15.



-18
-16
-14
-12
-10
-8
-6
-4
-2
0
Axial stress / (MPa)
A
B
C
Experiment
Numerical simulation
0
500
1000
1500
2000
2500
Micro-strain

(a)



(b)

Fig. 16.

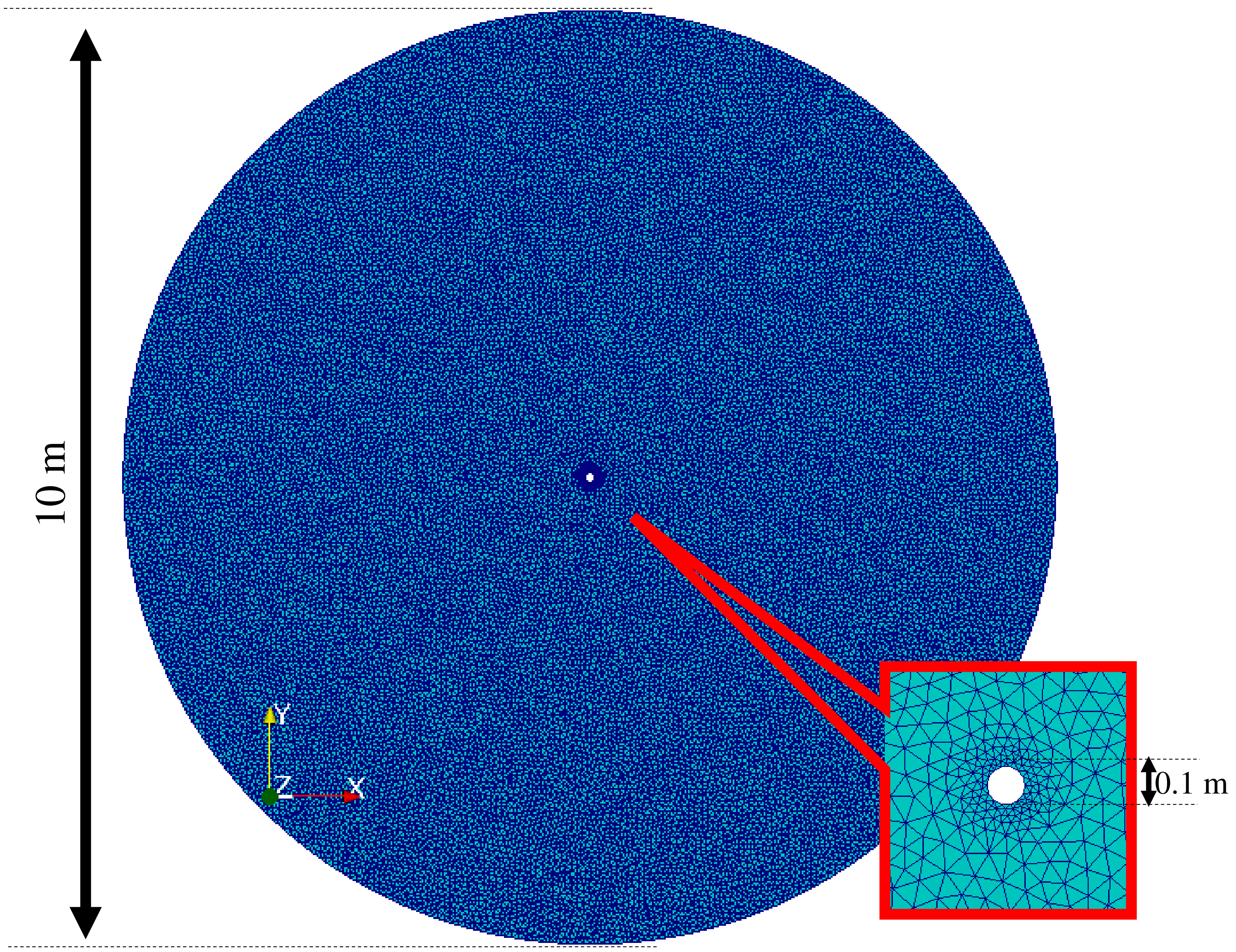
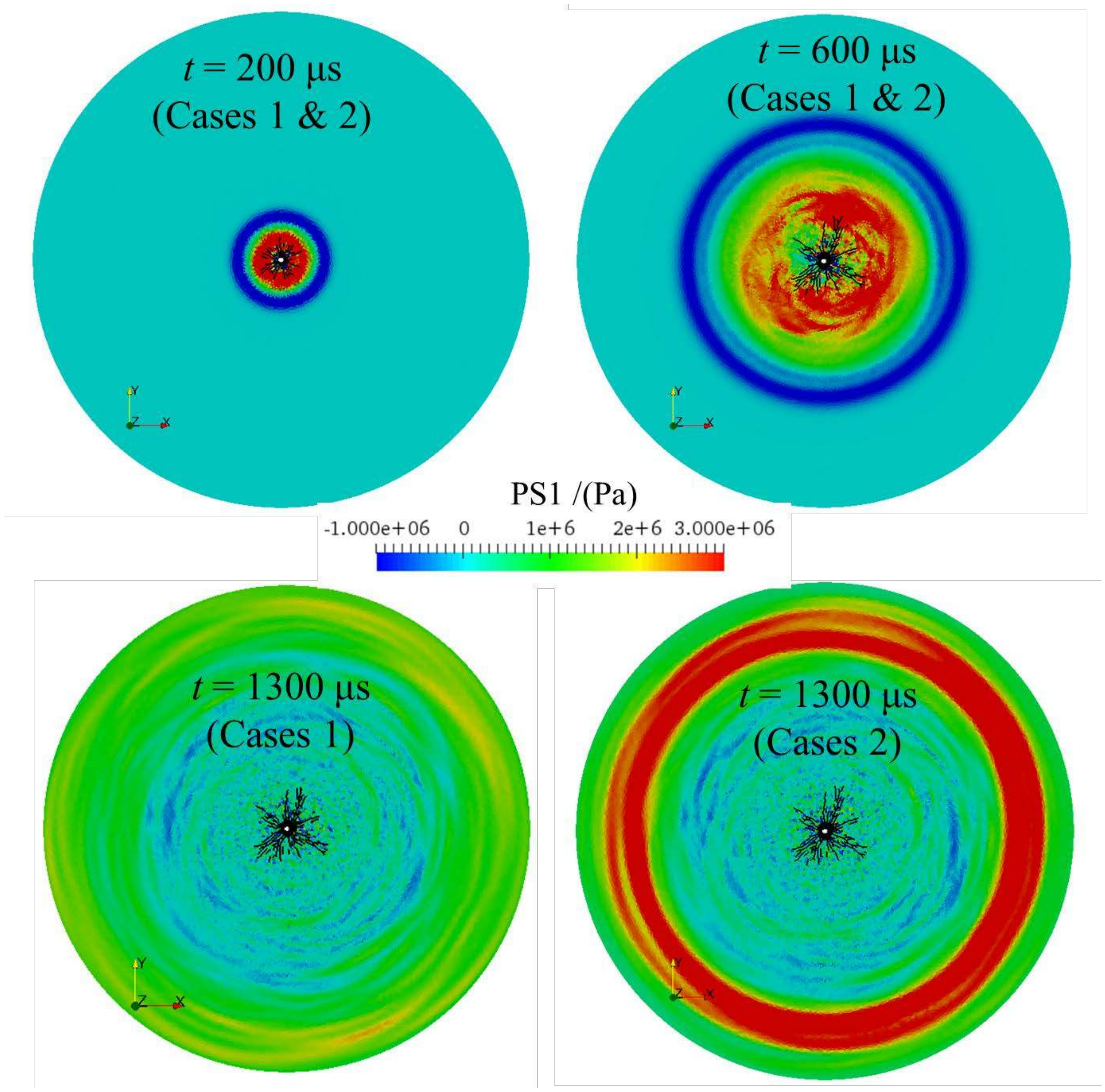
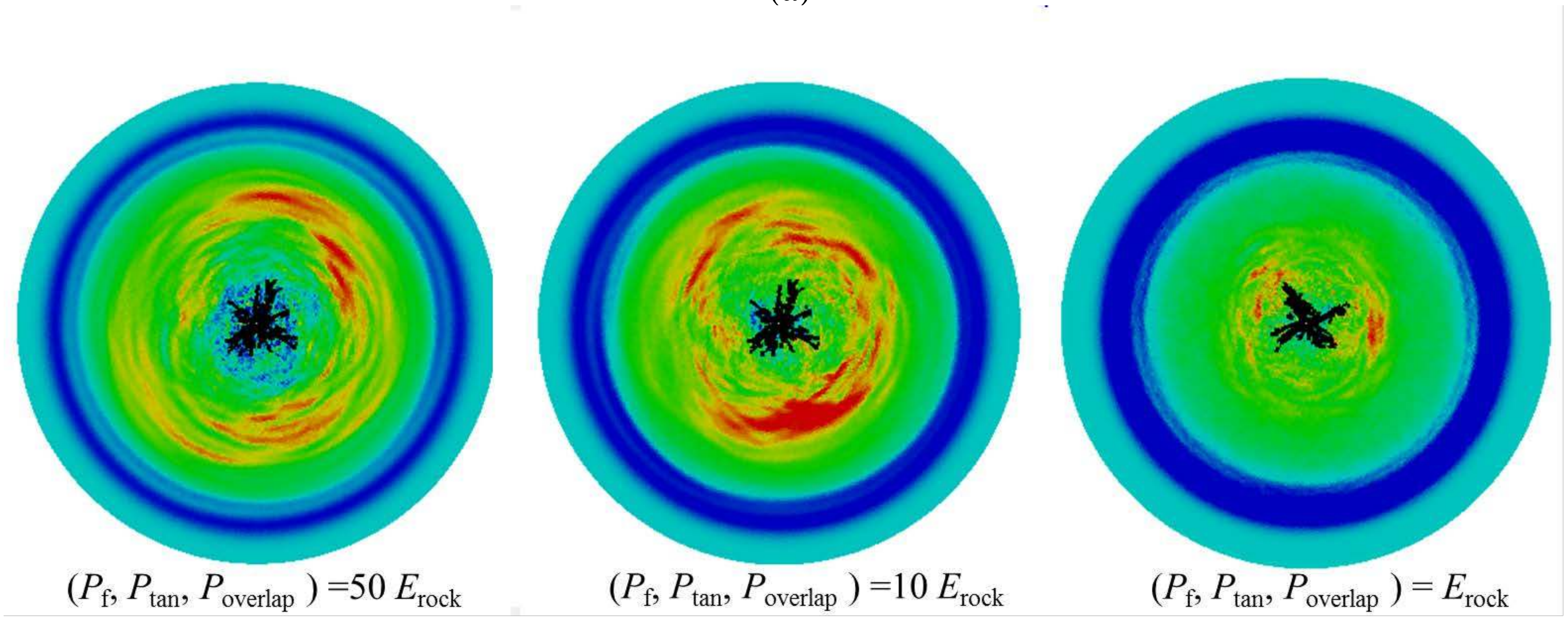


Fig. 17.



t = 200 μs (Cases 1 & 2)
t = 600 μs (Cases 1 & 2)
PS1 /(Pa)
-1.000e+06
0
1e+6
2e+6
3.000e+06
t = 1300 μs (Cases 1)
t = 1300 μs (Cases 2)
Y
Z
X

(a)



$(P_\mathrm{f}, P_\mathrm{tan}, P_\mathrm{overlap}) = 50\, E_\mathrm{rock}$
$(P_\mathrm{f}, P_\mathrm{tan}, P_\mathrm{overlap}) = 10\, E_\mathrm{rock}$
$(P_\mathrm{f}, P_\mathrm{tan}, P_\mathrm{overlap}) = E_\mathrm{rock}$

(b)

Fig. 18.

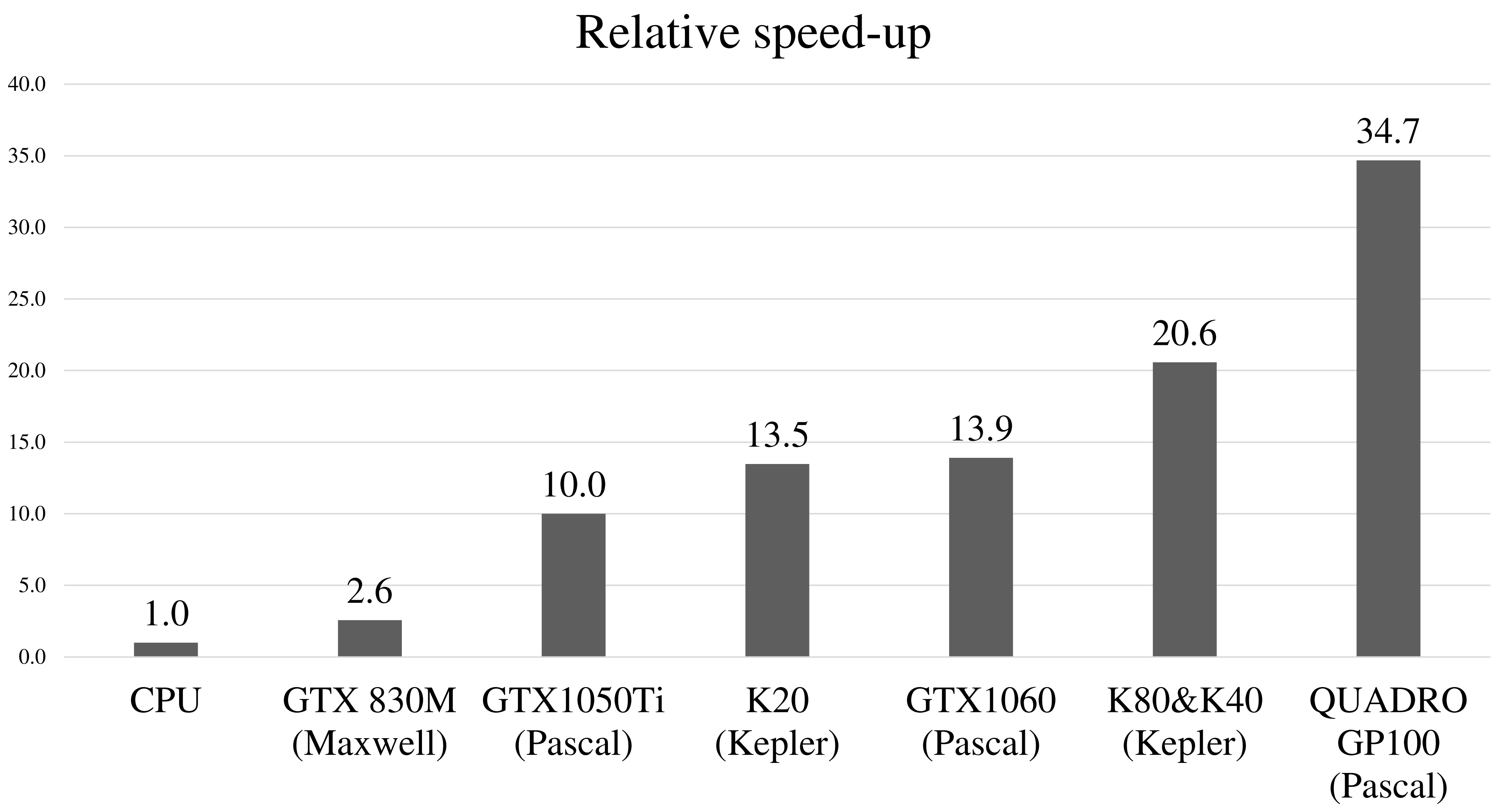

Relative speed-up
40.0
35.0
30.0
25.0
20.0
15.0
10.0
5.0
0.0
1.0
2.6
10.0
13.5
13.9
20.6
34.7
CPU
GTX 830M (Maxwell)
GTX1050Ti (Pascal)
K20 (Kepler)
GTX1060 (Pascal)
K80&K40 (Kepler)
QUADRO GP100 (Pascal)

Fig. 19.

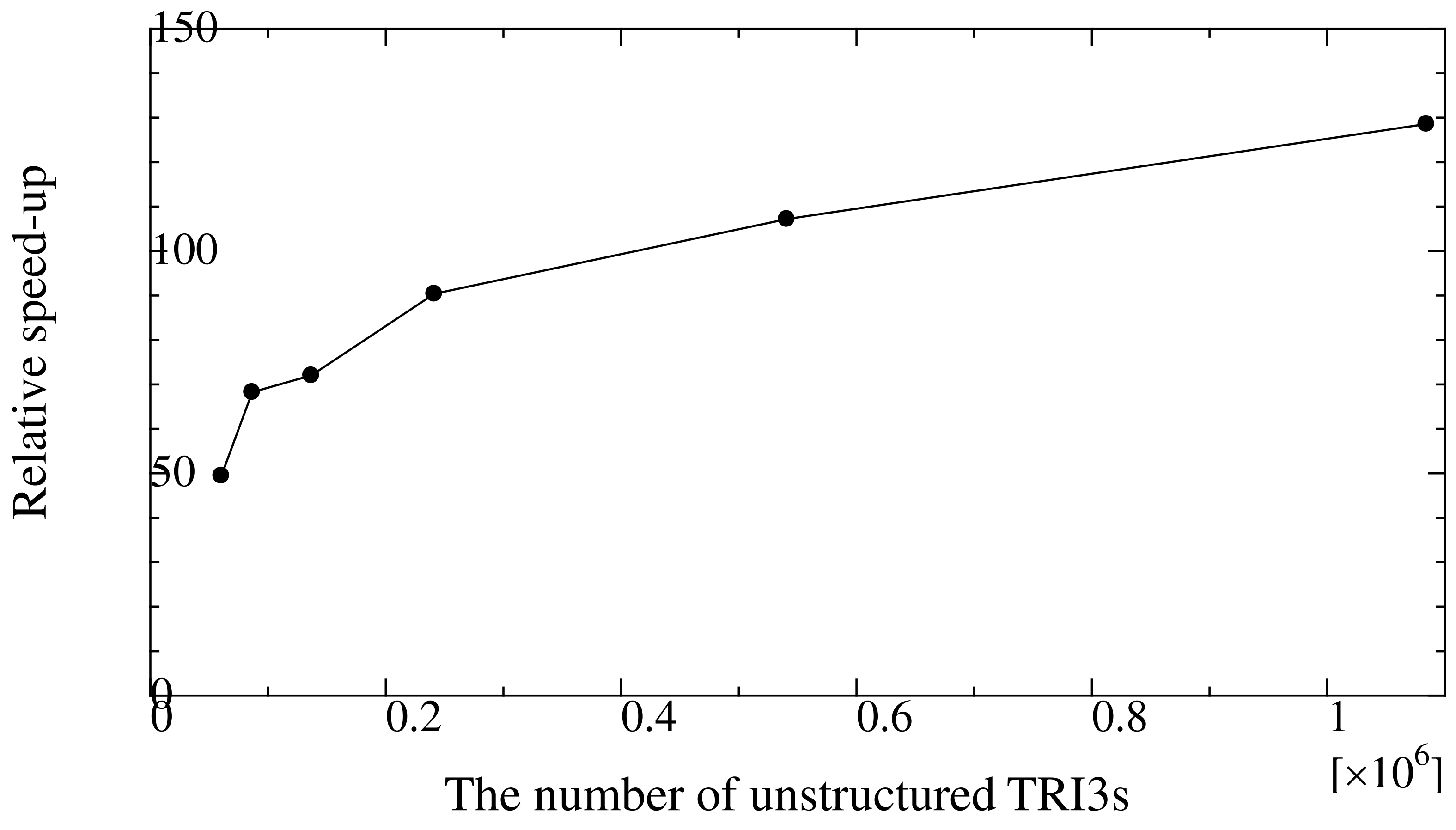

Relative speed-up
150
100
50
0
0
0.2
0.4
0.6
0.8
1
[×10^6]
The number of unstructured TRI3s

Fig. 20.

```
/* An example GPGPU kernel in 2-D Y-HFDEM IDE using CUDA C/C++.
 for 2D mechanical solver for nodes with x & y degrees of freedom.
 Assignment of boundary condition is not shown in this example.
 Access speed to GPGPU device local memory shown by register is faster,
 while that to GPGPU device global memory shown by variable names with
 _DEV is much slower                                                   */

__global__ void Ysd2MEC_GPU( double *d1nmct_DEV, // nodal mass
                             double *d1nvcx_DEV, // nodal velocity for x
                             double *d1nvcy_DEV, // nodal velocity for y
                             double *d1nfcx_DEV, // current nodal force for x
                             double *d1nfcy_DEV, // current nodal force for y
                             double *d1nccx_DEV, // nodal coordinate for x
                             double *d1nccy_DEV  // nodal coordinate for y )
{     register int    inopo;
      register double dt, nodal_mass;
      register double aX,aY,fx,fy,vXnew,vYnew,vXpre,vYpre,;
      // threadIdx.x:thread id, blockIdx.x:block id, blockDim.x :number of threads per block
      // Thus, inopo in the next line corresponds to node ID.
      inopo = threadIdx.x + blockIdx.x * blockDim.x;
      // current_num_TRI3_contact_candidate_CST: The total number of nodes in the system
      if(inopo <  current_num_nodes_CST)
      {      // Get nodal mass from GPGPU device global memory
            nodal_mass = d1nmct_DEV[inopo];
            // As long as nodal_mass is positive, update the nodal positions and velocities
            if(nodal_mass>EPSILON) // EPSILON: Very small value
            {     //////////////////////////////////////////////////
                  // x-direction
                  //////////////////////////////////////////////////
                  // Get x nodal velocity in previous step from GPGPU device global memory
                  vXpre = d1nvcx_DEV[inopo];
                  // Get x nodal force in the previous step from GPGPU device global memory
                  fx    = d1nfcx_DEV[inopo];
                  // Get current time step increment from GPGPU device constant memory
                  dt = dcstec_CST;
                  // Compute nodal acceleration along x direction based on Eq.(15)
                  aX = fx /nodal_mass;
                  // Compute new nodal velocity along x direction
                  vXnew = vXpre + aX * dt;
                  // Update x coordinate of this node and store it into GPGPU device global memory
                  d1nccx_DEV[inopo]  += vXnew * dt;
                  // Store new nodal velocity along x direction to  GPGPU device global memory
                  d1nvcx_DEV[inopo] = vXnew;
                  //////////////////////////////////////////////////
                  // y-direction
                  //////////////////////////////////////////////////
                  /*   Similar to x-direction. → Omitted. */
            }// end if(nodal_mass>EPSILON)
      }// end if(inopo <  current_num_TRI3_contact_candidate_CST)
}// end of Ysd2MEC_GPU
```

Fig. A.1