UNIVERSITY OF LEEDS

Proceedings Paper:
Salman, N, Wilson, M, Neville, A orcid.org/0000-0002-6479-1871 et al. (1 more author) (2017) Implementation of MCA in the framework of LIGGGHTS. In: Wriggers, P, Bischoff, M, Onate, E, Owen, DRJ and Zohdi, T, (eds.) 5th International Conference on Particle-Based Methods - Fundamentals and Applications (PARTICLES 2017). PARTICLES 2017, 26-28 Sep 2017, Hannover, Germany. International Center for Numerical Methods in Engineering (CIMNE) , pp. 767-777. ISBN 9788494690976

V International Conference on Particle-based Methods – Fundamentals and Applications
PARTICLES 2017
P. Wriggers, M. Bischoff, E. Oñate, D.R.J. Owen, & T. Zohdi (Eds)

IMPLEMENTATION OF MCA IN THE FRAMEWORK OF LIGGGHTS

NADIA SALMAN[1], MARK WILSON[1], ANNE NEVILLE[1] AND ALEXEY SMOLIN[2]

[1] University of Leeds
Centre of Doctoral Training in Integrated Tribology
Woodhouse Lane, LS2 9JT, Leeds, UK
email: mnnsa@leeds.ac.uk , webpage: https://www.leeds.ac.uk

[2] Institute of Strength Physics and Materials Science
Siberian Branch of Russian Academy of Sciences
pr. Akademicheskiy 2/4, 634021, Tomsk, Russia
email: asmolin@ispms.tsu.ru , webpage: http://www.ispms.ru

Key words: Movable Cellular Automata, LIGGGHTS, Plastic Deformation, Solid Behaviour.

Abstract. We describe the implementation of the Movable Cellular Automata Method (MCA) within the framework of the open-source code LIGGGHTS to simulate complex solid behaviour; most importantly plastic deformation, on different scales. The developed code extends the capabilities of the MCA method, as well as that of LIGGGHTS software; which simulates granular behaviour and is based on the discrete element method. The main difference between MCA and DEM is that the interaction between the particles is based on a many-body forces form of inter-automata interactions, similar to the embedded atom method used in molecular dynamics, because pair-wise interactions between elements used in DEM are insufficient to simulate irreversible strain accumulation (plasticity) in ductile consolidated materials. We first give an overview of the MCA method and its significance, followed by the implementation approach. The code has been successfully verified against analytical data.

1 INTRODUCTION

The Movable Cellular Automata Method (MCA) method was first introduced by Psakhie, Horie et al in 1995 [1] as a simulation tool within the framework of mesomechanics. MCA is a hybrid particle-based method based on the classical cellular automata (CA), discrete element (DEM) and molecular dynamics (MD) methods; combining their advantages. This method allows the modelling of complex materials behaviour and processes. Many developments in MCA have been made since 1995, and the latest description of the method can be found in [2]; where MCA is presented as a discrete approach to model the behaviour of materials on different length scales and is used as a multi-scale modelling approach.

MCA represents the medium as an ensemble of contacting or linked particles to simulate fracture and material deformation, as in the widely known discrete element methods. However, one of the fundamental problems with some particle-based methods, including DEM, is the correct representation of the inter-particle interaction forces, which is the most sensitive and time consuming part of any particle-based simulation [3]. The forces that describe the particle-particle interactions determine the physical and mechanical response of the system. In DEM, these take an approximated pair-wise form to describe materials on the

microscale. This form assumes that the total energy of the system is just the sum of the pair bonds, the same as in the Lennard-Jones potentials [4], which has been proven to often fail to describe the material on the macroscale and damage generation at scales lower than the size of the discrete element; that is why they are often coupled with continuum approaches.

Research showed that this problem can be solved by using a many-body interaction form which provides an accurate description of highly consolidated solids where elastic-plastic deformation occurs [2,3,5]. Hence, the authors of the MCA method applied the many-body interaction concept of the embedded atom method (EAM) [6,7], widely used in MD, to the MCA equations of motion. This allowed them to connect the average stresses and strains for the volume of each particle with the forces of interaction with its neighbouring particles. Meaning each automaton in the system follows the applied constitutive laws, leading to an accurate mechanical response of the whole system, and the capability of correct simulation of irreversible strain accumulation (plasticity) in ductile materials.

Since MCA is a particle-based method, it can be implemented within an MD or DEM code with some modifications as they have the same main functionalities. We chose LIGGGHTS as a framework for this task. LIGGGHTS [8] stands for LAMMPS Improved for General Granular and Granular Heat Transfer Simulations, which is a DEM open-source code, distributed under the GNU general public licence (GPL). LIGGGHTS was extended from LAMMPS [9], a powerful MD software developed at Sandia National Lab, to include the simulation of granular materials on larger scale levels. They are massive parallel computing software and designed for large scale simulations; by using spatial decomposition techniques to break down the simulation space into smaller 3D sub-domains; where each is assigned to a different processor. Message Passing Interface (MPI) exchange is used for communication between the processors allowing large simulation domains to be scaled in memory and speed.

In this paper we briefly describe the MCA methodology, more details can be found in [2,3,5], then present its implementation into the open-source code LIGGGHTS to enable the simulation of solid behaviour and plasticity in a 3D framework. This is done by adding our own commands and classes to the code, mainly a new atom style, pair style, bond style, and some fix styles, to implement MCA functionalities within LIGGGHTS; which is explained in section 3. LIGGGHTS source code, examples and documentation can be found at [10].

2 THEORETICAL BACKGROUND: THE MCA METHOD

MCA assumes that the simulated system is discretised into a series of small elements of finite size, known as the movable cellular automata. These automata are in contact and/or chemically linked (bonded) or unlinked (unbonded), and when an external load is applied they interact with each other, rotating and moving from one position to another. The interactive state between automata could be changed and they can form new automata pairs; describing mechanical deformation processes [5]. The motion of the automata is simulated according to their inter-automata interactions; using the Newton-Euler equations of motion; including the pair relationship, many-body forces and bond forces.

Hence, the simulation of automata motion is governed by the Newton-Euler equations to simulate the translational and rotational motion of pairs of automata:

$$\begin{cases} \boldsymbol{m_i \frac{d^2\vec{R}_i}{dt^2} = \sum_{j=1}^{N_i} \vec{F}_{ij}^{pair} + \vec{F}_i^{\Omega} = \sum_{j=1}^{N_i} \vec{F}_{ij}^{n} + \vec{F}_{ij}^{t} = \vec{F}_i} \\ \boldsymbol{\hat{J}_i \frac{d\vec{\omega}_i}{dt} = \sum_{j=1}^{N_i} \vec{M}_{ij}} \end{cases} \quad \textbf{(1)}$$

where $m_i, \vec{R_i}, \vec{\omega_i}, \hat{J_i}$ are the mass, location vector, rotation velocity and moment of inertia of automaton i respectively. $\vec{F}_i$ is the total force acting on i from its surrounding neighbours (N_i), and $\vec{M}_{ij}$ is the momentum of the system. This form for the inter-automata interaction forces is borrowed from the embedded atom method (EAM) [6,7]. By analogy, the total force $\vec{F}_i$ consists of a pair-wise component ($\vec{F}_{ij}^{pair}$) which depends on the displacement of i relative to j, and the volume-dependent component ($\vec{F}_i^{\Omega}$) which depends on the combined effects of all the neighbours of automata i. $\vec{F}_i$ can be described as the sum of the normal/central ($\vec{F}_{ij}^{n}$) and tangential/shear ($\vec{F}_{ij}^{t}$) components of interaction forces, and are characterized by the corresponding specific normal (σ_{ij}) and shear (τ_{ij}) response forces:

$$\begin{cases} \boldsymbol{\vec{F}_{ij}^{n} = \sigma_{ij} S_{ij}} \\ \boldsymbol{\vec{F}_{ij}^{t} = \vec{\tau}_{ij} S_{ij}} \end{cases} \quad \textbf{(2)}$$

where S_{ij} is the area of contact between automata i and j. Within the framework of multi-body interactions, forces acting on the automata are calculated using deformation parameters such as: elastic constants (shear modulus G_i and bulk modulus K_i), plastic constants (yield stress σ_y and work hardening ε_h), relative overlap (Δh_{ij}) for translational interaction, and tangential/shear displacement (Δl_{ij}^{shear}) for rotational interaction, as shown in Figure 1.

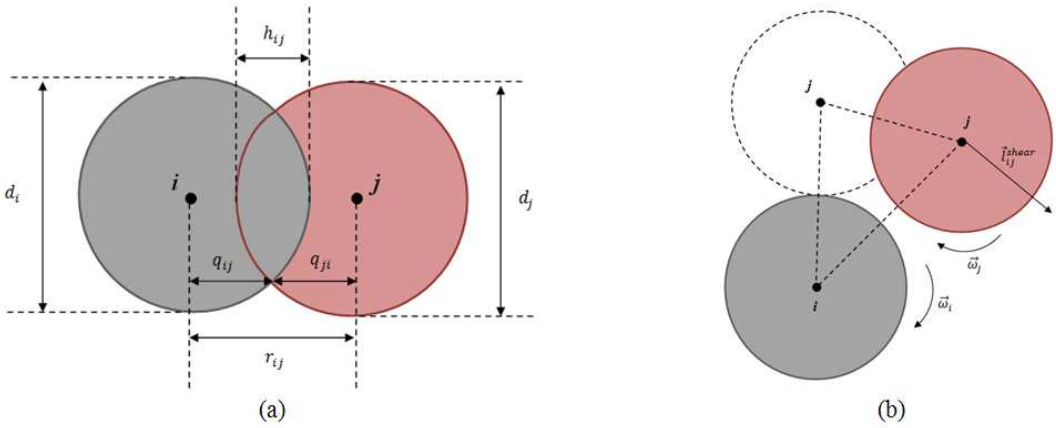


Figure 1: Schematic showing the (a) translational (b) rotational interactions between a pair of automata

$$\begin{cases} \boldsymbol{\Delta h_{ij} = \Delta \xi_{ij} d_i/2 + \Delta \xi_{ji} d_j/2} \\ \boldsymbol{\Delta \vec{l}_{ij}^{shear} = \Delta \vec{V}_{ij}^{shear} \Delta t / r_{ij} = \Delta \gamma_{ij} q_{ij} + \Delta \gamma_{ji} q_{ji}} \end{cases} \quad \textbf{(3)}$$

where normal strain (ξ_{ij}) and shear strain (γ_{ij}) are the parameters of deformation. Hence, the response of an isotropic elastic material can be described by the generalized Hooke's law:

$$\begin{cases} \boldsymbol{\Delta \sigma_{ij} = 2G_i \Delta \xi_{ij} + (1 - 2G_i/K_i) P_i} \\ \boldsymbol{\Delta \vec{\tau}_{ij} = 2G_i (\Delta \vec{\gamma}_{ij})} \end{cases} \quad \textbf{(4)}$$

where P_i is the pressure in volume (Ω_i) of automaton i, also known as the mean stress $\bar{\sigma}_{\text{mean}}^{i}$.

In 3D representation, bending and torsional deformation of the pair of automata also occur due to the difference in automaton rotation:

$$\Delta \boldsymbol{K_{ij}} = -\big(\boldsymbol{G_i} + \boldsymbol{G_j}\big)(\boldsymbol{\omega_j} - \boldsymbol{\omega_i})\Delta \boldsymbol{t} \tag{5}$$

where K_{ij} is the torque of automata pair, which is used to calculate the momentum in Eq. 1.

To calculate these parameters of deformation, in the many-body interaction form it is assumed that the stresses and strains are uniformly distributed in the volume of an automaton i. The average stresses and strains in the volume of i are determined by the average stress tensor ($\bar{\sigma}^i_{\alpha\beta}$) and average strain tensor ($\bar{\varepsilon}^i_{\alpha\beta}$), which are used to calculate mean stress P_i and the tensor invariants such as the equivalent stress ($\bar{\sigma}^i_{eq}$) and equivalent strain ($\bar{\varepsilon}^i_{eq}$) :

$$\begin{cases} \bar{\boldsymbol{\sigma}}^{\boldsymbol{i}}_{\boldsymbol{\alpha\beta}} = \frac{\mathbf{1}}{\Omega_i}\sum_{\boldsymbol{j=1}}^{\boldsymbol{N_i}} \boldsymbol{S_{ij} q_{ij}}(\vec{\boldsymbol{n}}_{\boldsymbol{ij}})_{\boldsymbol{\alpha}}\big[\boldsymbol{\sigma_{ij}}(\vec{\boldsymbol{n}}_{\boldsymbol{ij}})_{\boldsymbol{\beta}} + \boldsymbol{\tau_{ij}}(\vec{\boldsymbol{t}}_{\boldsymbol{ij}})_{\boldsymbol{\beta}}\big] \\ \boldsymbol{P_i} = -\bar{\boldsymbol{\sigma}}^{\boldsymbol{i}}_{\boldsymbol{mean}} = -\frac{\bar{\sigma}^i_{xx} + \bar{\sigma}^i_{yy} + \bar{\sigma}^i_{zz}}{3} \\ \bar{\boldsymbol{\sigma}}^{\boldsymbol{i}}_{\boldsymbol{eq}} = \frac{\mathbf{1}}{\sqrt{2}}\sqrt{(\bar{\boldsymbol{\sigma}}^{\boldsymbol{i}}_{\boldsymbol{xx}} - \bar{\boldsymbol{\sigma}}^{\boldsymbol{i}}_{\boldsymbol{yy}})^{\mathbf{2}} + (\bar{\boldsymbol{\sigma}}^{\boldsymbol{i}}_{\boldsymbol{yy}} - \bar{\boldsymbol{\sigma}}^{\boldsymbol{i}}_{\boldsymbol{zz}})^{\mathbf{2}} + (\bar{\boldsymbol{\sigma}}^{\boldsymbol{i}}_{\boldsymbol{zz}} - \bar{\boldsymbol{\sigma}}^{\boldsymbol{i}}_{\boldsymbol{xx}})^{\mathbf{2}} + \mathbf{6}\big[(\bar{\boldsymbol{\sigma}}^{\boldsymbol{i}}_{\boldsymbol{xy}})^{\mathbf{2}} + (\bar{\boldsymbol{\sigma}}^{\boldsymbol{i}}_{\boldsymbol{yz}})^{\mathbf{2}} + (\bar{\boldsymbol{\sigma}}^{\boldsymbol{i}}_{\boldsymbol{xz}})^{\mathbf{2}}\big]} \end{cases} \tag{6}$$

where $\alpha, \beta = X, Y, Z$ are the coordinates of the system, q_{ij} is the distance between the mass centre of i and its contact point with j, $(\vec{n}_{ij})_{\alpha,\beta}$ and $(\vec{t}_{ij})_{\beta}$ are the projections of unit-normal $(\vec{n}_{ij})$ and unit tangential $(\vec{\tau}_{ij})$, vectors on X,Y,Z coordinates. The average strain tensor in the automaton i can be computed by increments of elastic stress tensor at each time step [3]

The stress/strain tensor components are then used to realize the different elastic and plastic deformation models developed in continuum mechanics. This means that the forces of inter-automata interactions are directly obtained by the constitutive laws of the modelled medium.

For elastic-plastic behaviour, the von Mises model for plastic flow theory is used by adopting the Wilkins algorithm [11,12]. Here, if the stress intensity ($\bar{\sigma}^i_{eq}$) exceeds the plastic stress ($\bar{\sigma}^i_{pl}$) which is the radius of Von Misses yield circle, then the corrected stress for i is:

$$(\bar{\boldsymbol{\sigma}}^{\boldsymbol{i}}_{\boldsymbol{\alpha\beta}})' = (\bar{\boldsymbol{\sigma}}^{\boldsymbol{i}}_{\boldsymbol{\alpha\beta}} - \boldsymbol{P_i})\boldsymbol{M_i} + \boldsymbol{P_i} \tag{7}$$

where $(\bar{\sigma}^i_{\alpha\beta})'$ is the corrected (returned) average stress tensor, $\bar{\sigma}^i_{\alpha\beta}$ is the elastic stress tensor calculated before, and $\boldsymbol{M_i} = \bar{\sigma}^i_{pl}/\bar{\sigma}^i_{eq}$ is the coefficient of stress drop. Thus the corrected normal $(\boldsymbol{\sigma_{ij}})'$ and tangential $(\boldsymbol{\tau_{ij}})'$ forces are:

$$\begin{cases} (\boldsymbol{\sigma_{ij}})' = (\boldsymbol{\sigma_{ij}} - \boldsymbol{P_i})\boldsymbol{M_i} + \boldsymbol{P_i} \\ (\boldsymbol{\tau_{ij}})' = \boldsymbol{\tau_{ij}}\boldsymbol{M_i} \end{cases} \tag{8}$$

For fracture and bonding behaviour, a pair of automata can be considered to be linked or unlinked, and they can switch their state by using a switching criteria defined by the deformation of the material. These switching criteria could be a fracture or bonding criteria, and it is possible to apply any of the well known criteria used in continuum mechanics such as Mohr-Coulomb, Humber-Mises-Hencky, Drucker- Prager and others for this.

3 IMPLEMENTATION OF MCA IN LIGGGHTS: SOFTWARE DEVELOPMENT

Here we describe the implementation of the MCA 3D elastic-plastic model into LIGGGHTS and the relevant code parts that were added to describe MCA functionalities.

3.1 General data structure

LIGGGHTS is written in C++ using an object-oriented structure making it possible to modify and extend. The flowchart in

Figure 2 outlines the general structure of the LIGGGHTS program, which is quite similar to any general particle-based simulation program, showing their relevant functions in the source code in their order of execution.

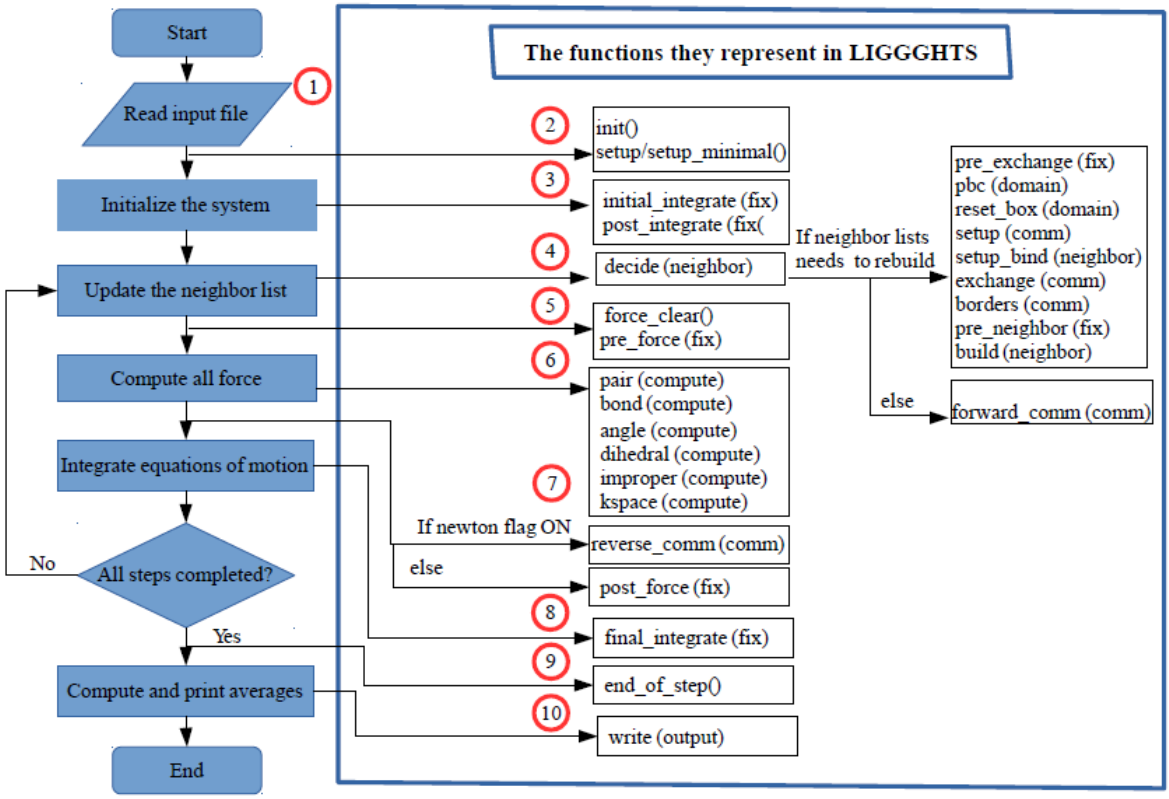


Figure 2: Flow chart of program structure and the relevant functions in LIGGGHTS

The system is first initialised using the input data which is defined in an input script by commands relevant to the acquired simulation. The initial positions and velocities are assigned to the particles. Then the time-stepping starts by integration; we use the Velocity-Verlet integration scheme [13], which is the most commonly used in MD and DEM simulations to calculate position and velocity as a function of time. According to the Velocity-Verlet integration scheme, as shown in Algorithm 1, the first step is to update the velocities by a half time-step and positions by one step. Then compute the interaction forces between the particles and their neighbours. Then update the velocities by another half-step.

Algorithm 1: Velocity-Verlet integration scheme

1- Calculate: $\vec{v}\left(t+\frac{1}{2}\Delta t\right)=\vec{v}(t)+\frac{1}{2}\vec{a}(t)\Delta t$
2- Calculate: $\vec{x}(t+\Delta t)=\vec{x}(t)+\vec{v}\left(t+\frac{1}{2}\Delta t\right)\Delta t$
3- Derive $\vec{a}(t+\Delta t)$ from the interaction forced using $\vec{x}(t+\Delta t)$
4- Calculate: : $\vec{v}(t+\Delta t)=\vec{v}\left(t+\frac{1}{2}\Delta t\right)+\frac{1}{2}\vec{a}(t+\Delta t)\Delta t$

LIGGGHTS works by calling the main functions in the order shown in the flowchart in Figure 2. Steps 1 and 2 in Algorithm 1 occur in the 'initial_integrate' function shown in the flowchart, step 3 by computing the forces (mainly 'pair' and 'bond' computes), and step 4 by 'final_integrate'. The calculation of the inter-particle forces is the most time consuming part of any particle-based simulation. The higher the number of particles, the relative distances and velocities between the neighbouring pairs, the higher the computational time to evaluate the forces between them. The computational time is reduced by using cut-off distance, neighbour lists and linked cell list algorithms to identify the nearby particles and only update and calculate the forces on the particles within the neighbour area within a given time step.

Each command in LIGGGHTS corresponds to a relevant class that defines that specific functionality. The following are the new commands added relevant to MCA implementation and examples of how they are defined in the input script:

```
atom_style    mca radius ${rp} packing fcc n_bondtypes ${bt} bonds_per_atom ${bpa}
pair_style    mca ${skin}
bond_style    mca
fix           integr nve_group nve/mca
fix           bondcr all bond/create/mca 1 1 1 ${cutoff} 1 ${bpa}
fix           topV_fix top mca/setvelocity 0 0 v_vel_up
```

Algorithm 2 shows the functions and the relevant MCA classes that were added to LIGGGHTS in the order of execution as shown in Figure 2, to implement our new commands by adding MCA functionalities to the source-code.

Algorithm 2: Programme structure with relevant new MCA implemented classes

1- init( ) / setup( )	[AtomVecMCA],[FixBondCreateMCA],[FixMCASetvel]
2- initial_integrate( )	[FixNVEMCA]
3- post_integrate( )	[FixBondCreateMCA]
4- pre_exchange( )	[FixBondExchangeMCA]
5- pre_force( )	[FixMeanStressMCA]
6- pair_compute( )	[PairMCA]
7- bond_compute( )	[BondMCA]
8- post_force( )	[FixMCASetvel]
9- final_integrate( )	[FixNVEMCA]

The main features that were added/implemented and their relevant classes according to Algorithm 2 are explained below. For the sake of ease of writing, the terms 'automata' and 'atoms' are used interchangeably having the same meaning.

3.2 Initialization

3.2.1 Initialise atom positions: AtomVecMCA

Before time-stepping begins the atoms are generated, the necessary structures are allocated in their memory locations and each atom is assigned with a position and velocity. In MCA, same as MD, the particles are placed in an appropriate lattice structure (SC, FCC, HCP).

Although the size of an automaton is characterized by a diameter (d_i), the shape of the automaton is not always a sphere. The real shape is determined by the area of its contact with its neighbor (S_{ij}), they have face-face interactions. This equivalent shape is characterized by a new radius parameter which is calculated from the initial volume of the automata.

Thus, some new automata parameters need to be defined in LIGGGHTS, which is done in a new class called 'AtomVecMCA'. These parameters are: mca radius, orientation vector, inertia, contact area, contact distance, volume, mean stress, equivalent stress, equivalent strain and number of bonded automata. Initial contact area and volume of automata are also defined and computed here based on the radius and packing.

3.2.2 Create bonds between atoms: FixBondCreateMCA

Each automaton forms bonds with its neighbours, and the maximum number of bonds of an automaton depends on the coordination number; 6 bonds for SC and 12 bonds for FCC/HCP. Each two neighboring automata form an automata pair, and are considered to be in contact. Initially, if the simulated specimen is a consolidated solid, then the contacts are assumed to be linked (bonded). If there are damages or cracks, then they are assumed to be unlinked (unbonded). This is implemented in the new class 'FixBondCreateMCA'.

3.2.3 Initialise atom velocities: FixMCASetvel

Atoms should also be initialised with a velocity before the time stepping begins. It is also sometimes useful to set boundary conditions or loading via velocity. This can be done in the new class 'FixMCASetvel'.

3.2.4 Initial integration: FixNVEMCA

Here the time stepping (run) begins and, as explained, according to the Velocity Verlet integration scheme, the first step is to update the velocities by a half time-step and positions by one step. In MCA, we also update the rotation velocity $\vec{\omega}$ by half-step.

3.2.5 Post integration: FixBondCreateMCA

After the first half of integration, the bond list and number of bonds need to be updated. This is also done in the class 'FixBondCreateMCA'.

3.3 Neighbour list generation and update

As explained, neighbour lists are produced and /or checked at every time step to exclude computing the interaction forces for any far away automata. This also includes checking the bonds between automata at every time step, erasing any broken bonds from this list and

adding any new bonds created. This is done in a new fix called 'FixExchangeBondMCA' is to exchange data of MCA bonds across processors at every time step.

3.4 Force calculation

3.4.1 Mean stress predictor: FixMeanStressMCA

Before starting the force calculation on each automaton, the predictor for $\bar{\sigma}^i_{mean}$ estimation should be calculated because the specific forces of interaction between the automata at the current time step is calculated using $\bar{\sigma}^i_{mean}$ as shown in Eq. 4. This is done by the 'pre_force' function in the FixMeanStressMCA' class. Usually for elastic behaviour, there is no need for a predictor corrector, while for accurate elasto-plastic behaviour two cycles are usually enough.

3.4.2 Pair and many-body force interactions: PairMCA

Algorithm 3 shows how the forces on the atoms are calculated, which is done in the new class 'PairMCA'. The elastic specific forces are calculated first, then the corrector for plasticity is calculated using the equivalent stress as described before, and the new total forces are obtained. Rotation is also taken into consideration and added to the code.

Algorithm 3: MCA forces computation algorithm - calculation in every time step

1- Calculate σ_{ij}, $\vec{\tau}_{ij}$ and K_{ij} at current time step (n+1), from equations 3,4 and 5.

2- Calculate corresponding values of $\bar{\sigma}^i_{xx}$, $\bar{\sigma}^i_{yy}$, $\bar{\sigma}^i_{xy}$, $\bar{\sigma}^i_{zz}$, $\bar{\sigma}^i_{mean}$ and $\bar{\sigma}^i_{eq}$, from equation 6.

3- Examine the value of $\bar{\sigma}^i_{eq}$, then calculate M_i if required.

4- Calculate σ'_{ij} and τ'_{ij}, from equations 7 and 8.

5- Calculate the forces and torques of automata, from equations 2,4 and 5.

3.4.3 Bond forces: BondMCA

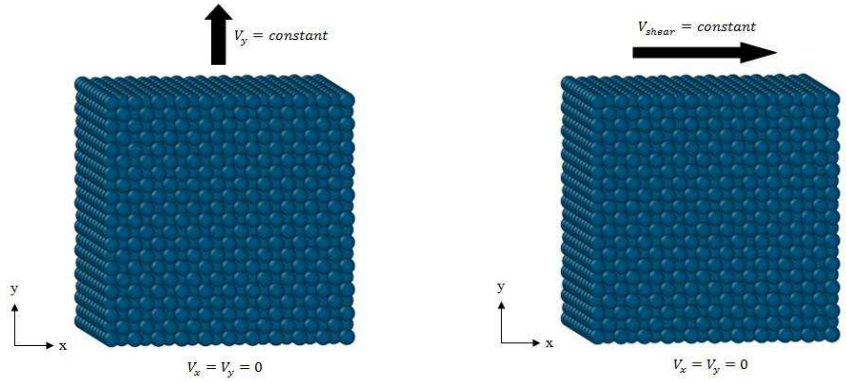
In this class the breaking of bonds is implemented. To do so the contact distances between automata are calculated, then using the equivalent stress criterion it checks if any bonds have broken using the Drucker- Prager criterion or any new bonds have been created.

3.5 Final integration

Before the final integration step, boundary conditions are added if needed which is done in the new 'FixMCASetVel' class. Then according to the velocity Verlet integration, the last step is updating the velocity and rotations for another half step, as shown in Algorithm 1, to obtain the final location and velocities of the automata. This is also implemented in 'FixNVEMCA' and this is then the end of the current time step. If it is not the last time-step of the simulation, the cycle is repeated starting from step 4 in the flowchart of Figure 2.

4 VERIFICATION OF ELASTIC-PLASTIC MODEL

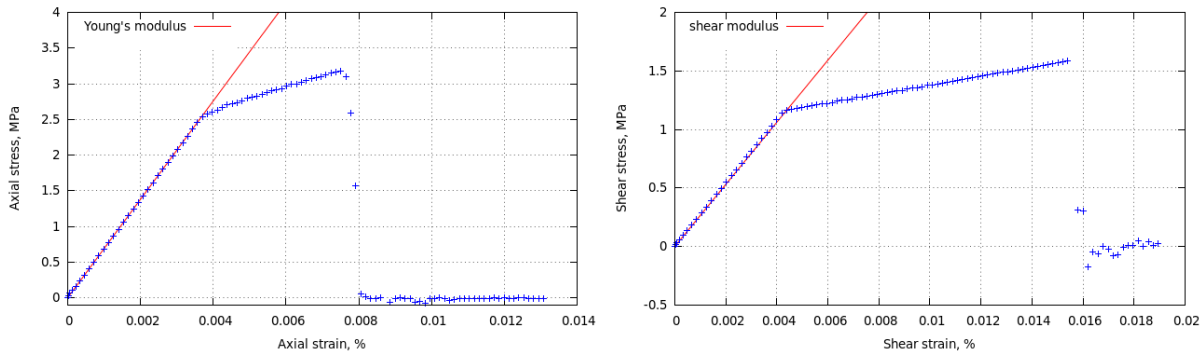
To verify the elastic-plastic MCA developed model in LIGGGHTS, 3D samples under uni-axial compression/tension and shear were modeled. The response function parameters of the material were checked to demonstrate correct macroscopic mechanical response. The simulated samples are 3D Aluminum samples, with equal-sized FCC packed automata as shown in Figure 3. The internal structure is assumed to be homogenous and free of discontinuities (damages or cracks), and all automata are assumed to be initially bonded. The material parameters are: Young's modulus $E = 70\,GPa$, Poisson's ratio $\mu = 0.3$, density $\rho = 2700\,kg/m^3$, yield stress $\sigma_y = 2\,MPa$ and work hardening modulus $\varepsilon_h = 10\,GPa$. The bottom layer of particles are fixed (velocity = 0) and the upper layer moves vertically (tension/compression) or horizontally (shear) with a low constant velocity to simulate quasi-static deformation regime. Both layers are free in the horizontal direction.



(a) (b)

Figure 3: Initial structure of MCA 3D model sample simulated in LIGGGHTS under (a) tension (b) shear

To study fracture behaviour, uni-axial tension/compression and shear tests were simulated with the setup shown in Figure 3. Their stress/strain curves (blue curves) are plotted against the analytical solution (red line) of Young's modulus for tension/compression and shear modulus for shear behaviour in Figure 4.

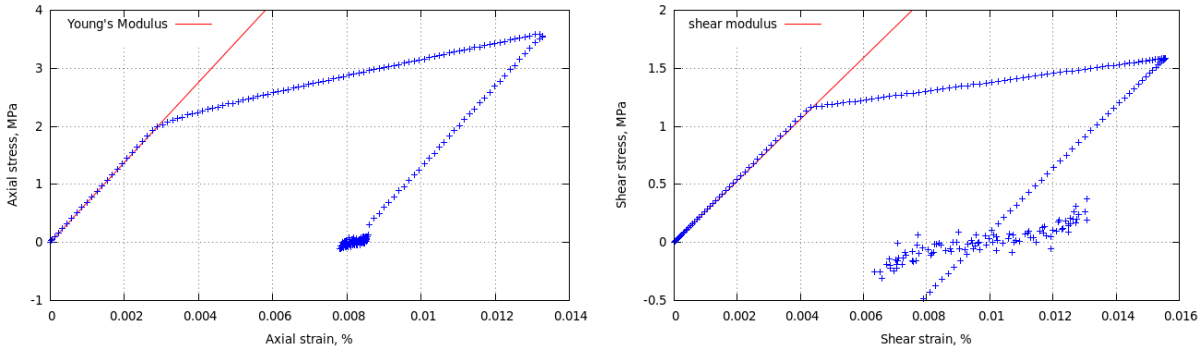


(a) (b)

Figure 4: Stress/strain curves of uni-axial (a) tension and (b) shear compared to analytical solution (red line).

Both tension and compression showed correct macroscopic response and very close agreement with the analytical solution as shown in Figure 4(a). It is important to mention that here we only tested with large automata sizes because MPI exchange has not been implemented yet to allow for parallel processing. More accurate results are obtained by decreasing the automata size relatively to the required behaviour. However, still the results show how the implementation of the many-body interaction forces form resulted in correct macroscopic response; which is not possible when only pair-wise interactions are considered. The same is shown by the shear behaviour in Figure 4(b). The results show the possibility of modelling fracture under tension/compression and shear loading as the material fails after reaching a critical value.

Furthermore, to test correct plasticity behaviour, uni-axial loading/unloading was also simulated for tension and shear as shown in Figure 5. The results show a typical loading-unloading curve, which means the model is capable of modelling correct elasto-plastic behaviour.



(a) (b)

Figure 5: Stress/strain curves of uni-axial loading/unloading by (a) tension and (b) shear compared to analytical solution (red line).

5 CONCLUSIONS

- A new elastic-plastic model was developed within LIGGGHTS open-source code. The model is based on the MCA particle-based method.
- This implementation allows for the modelling of correct fracture and elasto-plastic behaviour, as shown by the verification tests.
- The current limitation of the code is that it does not allow MPI exchange for parallel processing, however this will still be implemented within the code.
- Further work will also include testing the bond breaking of the particles, as well more complex materials systems, with smaller automata size and different boundary conditions.

REFERENCES

[1] S.G. Psakhie, Y. Horie, S.Y. Korostelev, A.Y. Smolin, A.I. Dmitriev, E. V. Shilko, S. V. Alekseev, Method of movable cellular automata as a tool for simulation within the framework of mesomechanics, Russ. Phys. J. 38 (1995) 1157–1168.

[2] E. V. Shilko, S.G. Psakhie, S. Schmauder, V.L. Popov, S. V. Astafurov, A.Y. Smolin, Overcoming the limitations of distinct element method for multiscale modeling of materials with multimodal internal structure, Comput. Mater. Sci. 102 (2015) 267–285.
[3] S.G. Psakhie, E. V Shilko, A.S. Grigoriev, S. V Astafurov, A. V Dimaki, A mathematical model of particle – particle interaction for discrete element based modeling of deformation and fracture of heterogeneous elastic – plastic materials, Eng. Fract. Mech. 130 (2014) 96–115.
[4] S. Plimpton, Fast Parallel Algorithms for Short–Range Molecular Dynamics, J. Comput. Phys. 117 (1995) 1–19.
[5] S. Psakhie, E. Shilko, A. Smolin, S. Astafurov, V. Ovcharenko, Development of a formalism of movable cellular automaton method for numerical modeling of fracture of heterogeneous elastic-plastic materials, Frat. Ed Integrita Strutt. 24 (2013) 26–59.
[6] M.S. Daw, M.I. Baskes, Embedded-atom method: Derivation and application to impurities, surfaces, and other defects in metals, Phys. Rev. B. 29 (1984) 6443–6453.
[7] M.S. Daw, S.M. Foiles, M.I. Baskes, The Embedded-Atom Method: A Review of Theory and Applications, Elsevier Sci. Publ. N/A (1993) 251–310.
[8] C. Kloss, C. Goniva, LIGGGHTS: a new open source discrete element simulation software, in: Proc. Fifth Int. Conf. Discret. Elem. Methods, London, UK, 2010: pp. 25–26.
[9] S.J. Plimpton, Large-scale Atomic/Molecular Massively Parallel Simulator, LAMMPS, (n.d.). http://lammps.sandia.gov.
[10] Christoph Kloss, LIGGGHTS, (n.d.). https://github.com/CFDEMproject/LIGGGHTS-PUBLIC.
[11] M.L. Wilkins, Calculations of elastic-plastic flow, Academic Press, New York, 1964.
[12] M.L. Wilkins, Computer simulation of dynamic phenomena, 1999.
[13] N. Martys, R.D. Mountain, Velocity Verlet algoritm for dissipative-particle-dynamics-based models for suspensions, Phys. Rev. E. 59 (1999) 3733–3736.