Programming the micro-mechanical model of granular materials in Julia

Hao Xiong[a], Zhen-Yu Yin[b,*], François Nicot[c]

[a] *College of Civil and Transportation Engineering, Shenzhen University, Shenzhen, China.*
[b] *Department of Civil and Environmental Engineering, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong.*
[c] *Université Grenoble Alpes, IRSTEA, Geomechanics Group, ETNA, Grenoble, France*

Abstract

Modelling the mechanical behaviour of granular materials using the insight of physics, such as discrete element method (DEM), usually costs a lot of computing resources as a result of the storing and transferring of a large amount of particle and contact information. Unlike DEM, the micro-mechanical (MM) model, based on statistics of directional inter-particle contacts of a representative volume of an element, imposes a much lower computational demand while retaining granular physics. This paper presents such a kinematic hypothesis-based MM modelling framework, programmed by a dynamic coding language, Julia. The directional local law of a recently developed 3D-H model is selected as an example of the implementation. The entire code of the MM model programmed by Julia is structured into several functions by which multilevel loops are called in an order. Moreover, a global mixed-loading control method is proposed in this study by which the stress control and strain control can be achieved simultaneously. Using this method, conventional triaxial tests and proportional strain tests are simulated to calibrate the model according to experimental

*Corresponding author.
Email address: zhenyu.yin@polyu.edu.hk; zhenyu.yin@gmail.com (Zhen-Yu Yin)

data. The same experiments are also simulated by DEM for comparison with the MM model to estimate the computational efficiency and accuracy, which demonstrates a significant advantage of the MM model. This study can be directly used for modelling other materials by changing the directional local law and provides helpful guidance for programming of similar multiscale approaches.
Keywords: Julia language, High-performance dynamic programming, Micromechanics, Granular materials, Multiscale, Microstructure

1. Introduction

For several decades, granular materials (soils, pills and rice) have been considered as continuous media with the development of continuum mechanics theory. A series of phenomenological models have been proposed and successfully used in modelling large-scale boundary value problems through numerical methods such as the finite elements and finite differences methods (FEM and FDM) [29, 32, 34, 36, 44]. However, an increasing number of issues involve difficulty in their proper consideration, such as (i) large displacement/deformation, (ii) particle trajectory descriptions, (iii) particle breakage, (iv) crucial arching effects, (v) particle-fluid-gas interaction and so on. Although phenomenological models could partly solve these issues by introducing new parameters, the latter lead to complex mathematical formulations and are usually hard to understand, precisely because phenomenological models attempt to express, in global terms, mechanisms that occur on a highly local scale.

The consideration of these kinds of issues via a discrete mechanics framework is certainly the most convenient approach, especially because the discrete variables and their related governing laws at the particle scale provide clear physical meaning and have been well established. The most popular method belonging to this framework is the so-called "discrete element method" (DEM) [8], based on the double integration

of the particle acceleration expressed by the second Newton's law. The remarkable predictive capacities of various DEM models are illustrated in many studies [11, 26, 35]. Because of its discreteness, DEM requires large computer memory and a high-level CPU core when a great number of particles is considered, especially in large-scale boundary value problems.

To optimise computational resources, two aspects can be improved. On one hand, suitable numerical/analytical homogenisation techniques for upscaling from the grain scale to the macroscopic level are desirable. In the literature, two types of micro-mechanical (MM) models, based on either a kinematic or a static hypothesis, can be increasingly found [5, 17, 28, 43]. For example, the CH-model proposed by [41] is derived based on the static hypothesis; the microdirectional model and H-model proposed by [27, 28, 37] are considered under kinematic hypothesis. Even though the $\mu -$ GM model (developed by [30]) could be derived based on both hypotheses, the static hypothesis is still recommended. Theoretically speaking, both could be used. The major difference between the two is that the derivation direction of the constitutive relationship is different. The kinematically based MM derives from strain to stress, whereas the statically based MM is the opposite.

On the other hand, a convenient programming language also could help a lot. For scientific researchers, the Julia language [3, 4] is a perfect choice. The Julia language is an open source dynamic language publicly released in 2012. Hundreds of individuals have contributed to its development since its release. One of the main benefits to using Julia is its computational efficiency which exceeds other dynamic languages, such as MATLAB [20] or Python [31]. It also exhibits equivalencies of static languages, such as FORTRAN [24] or C++ [33]. With the approximate computational efficiency of static languages and functions of dynamic languages, Julia is a unique combination of both coding classes, giving the user a high-performance

level while still providing an interactive and constructive interface. Other advantages of Julia include its flexibility and open source availability. As a relatively new language, Julia has undergone significant changes between versions, with version 1.0 announced to be released in late 2017. Multiple publications and advances in Julia have been made since its release in 2012. It is regret that no such MM model is available in Julia which limits the high efficient analysis of boundary value problems starting with grain scale.

That said, a convenient programming language could also help a great deal. For scientific researchers, the Julia language [3, 4] is a perfect choice. Julia is an open-source dynamic language that was publicly released in 2012. Hundreds of individuals have contributed to its development since its release. One of the main benefits of using Julia is its computational efficiency, which exceeds that of other dynamic languages such as MATLAB [20] or Python [31]. It also exhibits the equivalencies of static languages such as FORTRAN [24] or C++ [33]. With the approximate computational efficiency of static languages and functions of dynamic languages, Julia is a unique combination of both coding classes, giving the user a high-level performance while still providing an interactive and constructive interface. Other advantages include its flexibility and open source availability. As a relatively new language, Julia has undergone significant changes between versions, with version 1.0 released in late 2017. Multiple publications and advances in Julia have been made since its release in 2012. Unfortunately, no such MM model is available in Julia, which limits the high-efficiency analysis of boundary value problems, starting with grain scale.

The aim of this paper is to answer two questions: why do we develop the MM model, and how do we program an MM model in Julia? The 3D-H model, as an example of an MM model, is first reviewed. The basic formulation and multiscale

framework are introduced. Then, the entire code of the 3D-H model, which is readable with a clear flow chart, is shown and illustrated step by step. After calibrating the model parameters, the model performance could be acknowledged by showing the numerical results on both micro and macroscales under various loading paths. Finally, the calculation efficiency and accuracy of the presented code is compared with the DEM model.

2. A brief introduction to the 3D-H model

Generally, two types of MM models, based on either a static or a kinematic hypothesis, can be found in the literature. The models based on the former have been widely discussed in [43]. Those based on the latter have proven to be suitable in modelling the mechanical behaviour of granular material [27, 28]. The 3D-H model [37], one of the models based on the kinematic hypothesis, was initially proposed for granular materials. It was then implemented within a finite element code to simulate engineering problems [38, 39].

Based on the kinematic hypothesis, the 3D-H model enables the derivation of the macrostress tensor from the macrostrain tensor according to the following steps (Figure 1):

(1) Kinematic localisation: The mesostructure (shown in Figure 3) is a connection between the macro- and mesoscale. The dimension of the mesostructure can be characterised by the vector: $\vec{L} = [l_1, l_2, l_3]^T$, wherein l_1, l_2, l_3 represent the lengths along directions $\vec{n}$, $\vec{t}$, $\vec{w}$, respectively (see Figure 4a and Figure 5a). Thus the kinematic localisation assumption gives:

$$\delta\vec{L} = \bar{\bar{P}}\delta\bar{\bar{\varepsilon}}\bar{\bar{P}}^{-1}\vec{L} \tag{1}$$

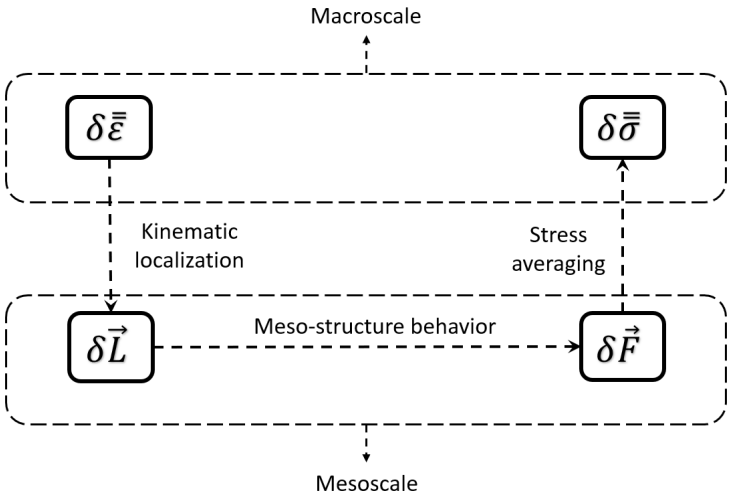


Figure 1: General homogenization scheme of 3D-H model [6].

where $\delta\bar{\bar{\varepsilon}}$ is the incremental macrostrain tensor, and $\bar{\bar{P}}$ is the rotation matrix from global frame $(\vec{x_1}, \vec{x_2}, \vec{x_3})$ to local frame $(\vec{n}, \vec{t}, \vec{w})$ (see Figure 2).

The kinematic localisation defined by Equation 1 is a homogenisation process. It is analogous to the usual Voigt approximation in the field of continuous media. The reader should note that the localisation process goes from the macro- to the mesoscale, not to the microscale, as assumed in the Voigt approximation. Moreover, it has been widely used in granular materials, such as in [6, 27, 28].

In the 3D-H model, the hexagon strain described by the vector $\delta\vec{L} = [\delta l_1, \delta l_2, \delta l_3]^T$ is derived from the macroscopic strain tensor. Then, the term $\delta\vec{L}$ is used as the known variable to compute the relative displacement at each contact and to compute contact forces. $\delta l_1, \delta l_2, \delta l_3$ are independent because 10 particles are involved in a mesostructure, rather than simply a single contact between particles.

(2) Mesostructure behaviour: The mesostructure (Figure 3) can be decomposed into two independent hexagon patterns: Hexagon A (Figure 4) and Hexagon B (Figure 5), both being similar. The geometrical configuration and external forces applied

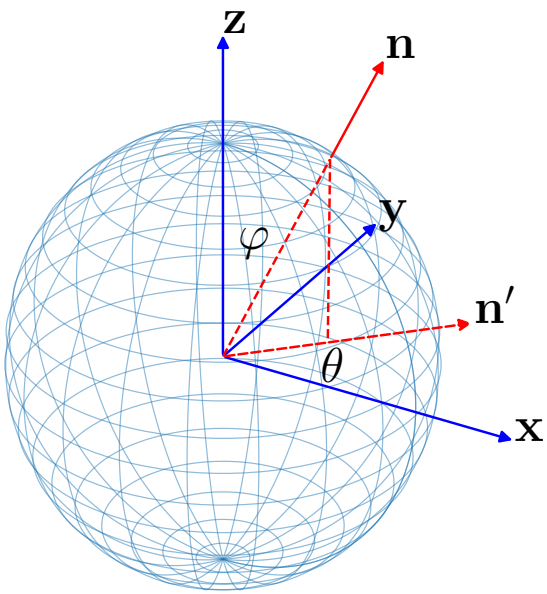


Figure 2: Global and local coordinate system transformation by employing Euler angles in 3D conditions.

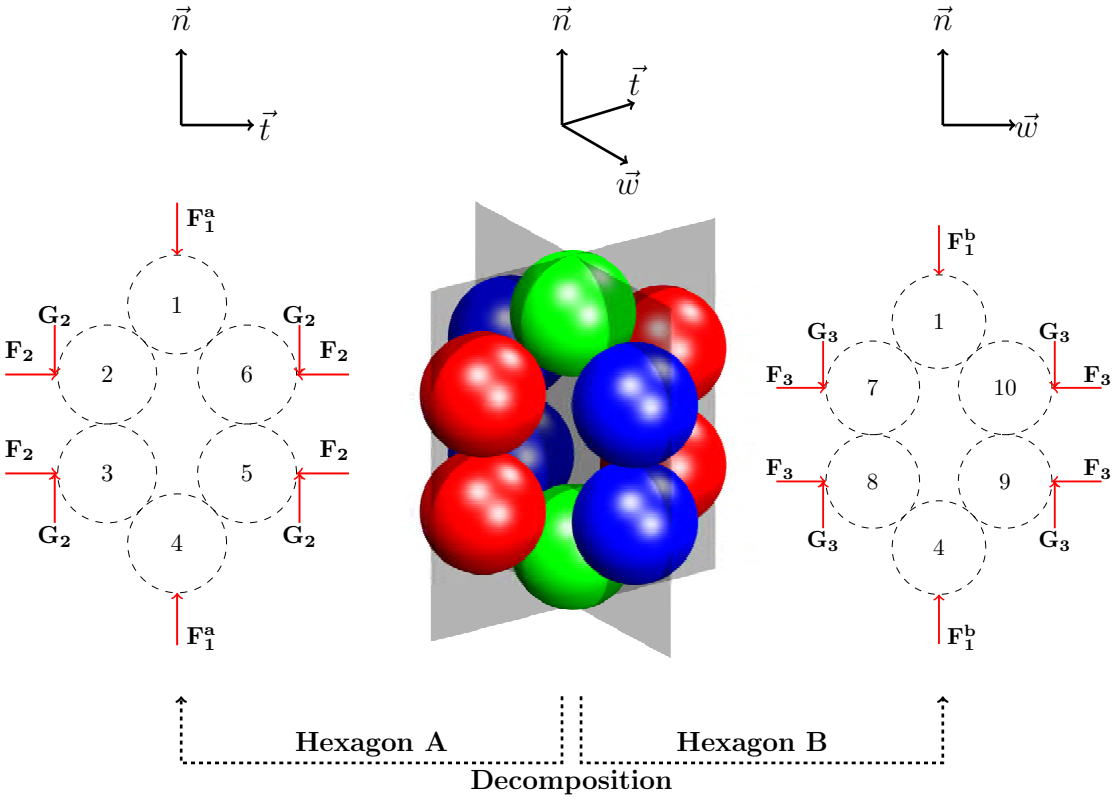


Figure 3: The 3D mesostructure and its decomposition procedure in the 3D-H model.

to the mesostructure are symmetrical; thus for each hexagon, only two grains need to be analysed. For Hexagon A, as shown in Figure 4, only grains 1 and 2 are analysed. The contact between grains 1 and 2 is denoted by contact 1, whereas the contact between grains 2 and 3 is denoted by contact 2. Then, the kinematic relations read (for Hexagon A):

$$\begin{aligned} \delta u_n^1 &= \delta d_1 \\ \delta u_t^1 &= d_1 \delta \alpha_1 \\ \delta u_n^2 &= \delta d_2 \end{aligned} \tag{2}$$

where u_n^i and u_t^i represent the normal and tangential relative displacements at contact i (For Hexagon A, $i = 1$ or 2, for Hexagon B, $i = 3$ or 4), respectively. As depicted in Figure 4a, the geometrical description for Hexagon A gives:

$$\begin{aligned} l_1 &= d_2 + 2d_1 \cos\alpha_1 \\ l_2 &= 2d_1 \sin\alpha_1 \end{aligned} \tag{3}$$

The force balance of grain 1 along direction $\vec{n}$ and of grain 2 along directions $\vec{w}$ and $\vec{n}$, together with the moment balance of grain 2, reads:

$$F_1^a = 2(N_1 \cos\alpha_1 + T_1 \sin\alpha_1) \tag{4a}$$

$$F_2 = N_1 \sin\alpha_1 - T_1 \cos\alpha_1 \tag{4b}$$

$$N_2 = N_1 \cos\alpha_1 + T_1 \sin\alpha_1 + G_2 \tag{4c}$$

$$G_2 = T_2 \tag{4d}$$

where N_i and T_i represent the normal and tangential contact forces of contact i, respectively. The elastic-perfect plastic inter-particle contact law reads, for a given

contact i:

$$\begin{aligned}
\delta N_i &= k_n \delta u_n^i \\
\delta \vec{T_i} &= \min\left\{ \left\| \vec{T_i} + k_t \delta \vec{u_t^i} \right\|, \tan\varphi_g \left(N_i + \delta N_i \right) \right\} \times \frac{\vec{T_i} + k_t \delta \vec{u_t^i}}{\left\| \vec{T_i} + k_t \delta \vec{u_t^i} \right\|} - \vec{T_i}
\end{aligned} \tag{5}$$

After simplifying (see more details in Appendix A), the contact law (for Hexagon A) can be rewritten as follows:

$$\begin{aligned}
\delta N_1 &= -k_n \delta d_1 \\
\delta N_2 &= -k_n \delta d_2 \\
\delta T_1 &= B_1 \delta \alpha_1 - A_1 \delta d_1 + C_1
\end{aligned} \tag{6}$$

Term C_1 differs from zero only during a transition from an elastic to a plastic regime. Except in this situation, it is zero. For very small strain increments, as considered throughout this paper, term C_1 can therefore be neglected.

To obtain the incremental evolution of the external forces, δd_1, δd_2 and $\delta \alpha_1$ need to be expressed as a function of the mesostrain. Three equations are therefore required. Compatibilities (Equations 8) provide two relations. The third is the balance equation of grain 2 along direction $\vec{n}$ (Equation 4c). By taking the inter-particle contact law (Equations 6) into account, we reach the following algebraic system expressing the incremental changes in δd_1, δd_2 and $\delta \alpha_1$ with respect to the incremental changes in δl_1 and δl_2:

$$\begin{bmatrix} 2\cos\alpha_1 & 1 & -2d_1\sin\alpha_1 \\ 2\sin\alpha_1 & 0 & 2d_1\cos\alpha_1 \\ \cos\alpha_1 + \frac{A_1}{k_n}(\sin\alpha_1 + 1) & -1 & \frac{F_2 - B_1(\sin\alpha_1+1)}{k_n} \end{bmatrix} \begin{bmatrix} \delta d_1 \\ \delta d_2 \\ \delta \alpha_1 \end{bmatrix} = \begin{bmatrix} \delta l_1 \\ \delta l_2 \\ 0 \end{bmatrix} \tag{7}$$

where A_1, B_1 are given in Appendix A.

Differentiating Equation 4a, Equation 4b and combining with Equations 6 gives:

$$\begin{aligned}\delta F_1^a &= -k_n \cos\alpha_1 \delta u_n^1 + k_t \sin\alpha_1 \delta u_t^1 - F_2 \delta\alpha_1 \\ \delta F_2 &= -k_t \cos\alpha_1 \delta u_t^1 - k_n \sin\alpha_1 \delta u_n^1 + F_1^a \delta\alpha_1\end{aligned} \tag{8}$$

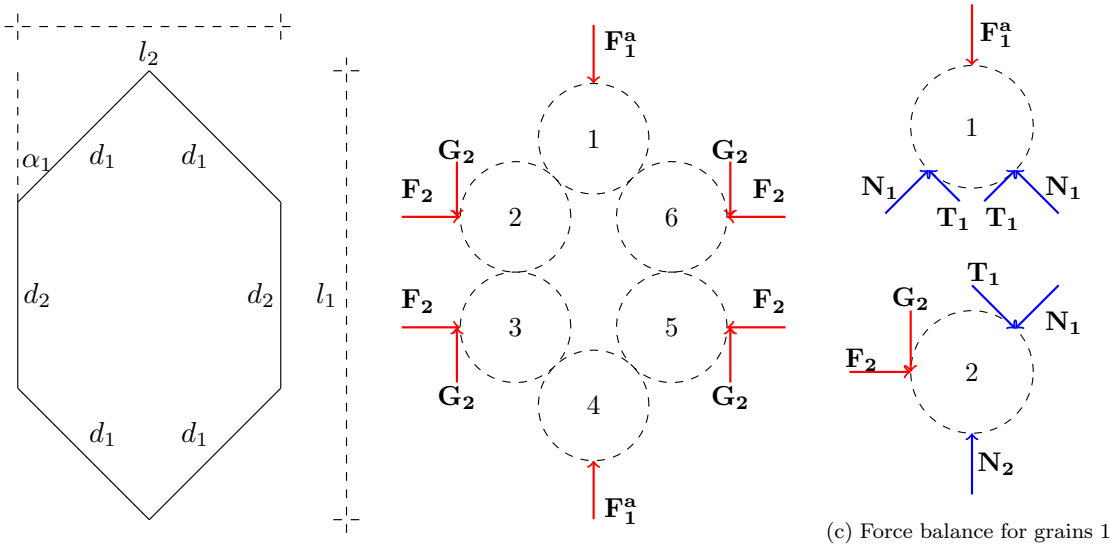
Thus, combining Equations 2, Equations 7 and Equation 8, the incremental constitutive relation for Hexagon A can be expressed as follows:

$$\frac{1}{|D|^a}\begin{bmatrix} K_{11}^a & K_{12}^a \\ K_{21}^a & K_{22}^a \end{bmatrix}\begin{bmatrix} \delta l_1 \\ \delta l_2 \end{bmatrix} = \begin{bmatrix} \delta F_1^a \\ \delta F_2 \end{bmatrix} \tag{9}$$

where:

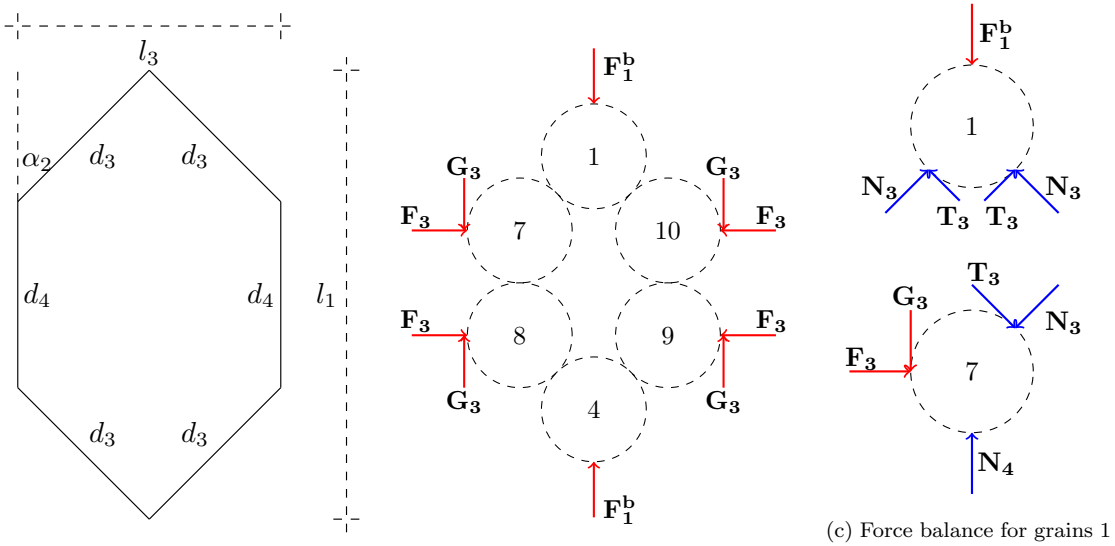
$$\begin{cases} K_{11}^a &= 2\left(F_2 \sin\alpha_1 - k_n d_1 \cos^2\alpha_1 - k_t d_1 \sin^2\alpha_1\right) \\ K_{12}^a &= \left(k_t d_1 \sin\alpha_1 - F_2\right)\left(\dfrac{A_1}{k_n}\sin\alpha_1 + \dfrac{A_1}{k_n} + 3\cos\alpha_1\right) \\ & \quad - \cos\alpha_1\left(B_1 \sin\alpha_1 + B_1 - F_2 + 2k_n d_1 \sin\alpha_1\right) \\ K_{21}^a &= 2\left(k_t - k_n\right) d_1 \sin\alpha_1 \cos\alpha_1 - 2F_1^a \sin\alpha_1 \\ K_{22}^a &= \left(F_1^a - k_t d_1 \cos\alpha_1\right)\left(\dfrac{A_1}{k_n}\sin\alpha_1 + \dfrac{A_1}{k_n} + 3\cos\alpha_1\right) \\ & \quad - \sin\alpha_1\left(B_1 \sin\alpha_1 + B_1 - F_2 + 2k_n d_1 \sin\alpha_1\right) \\ |D|^a &= \dfrac{2}{k_n}(B_1 \sin\alpha_1 + A_1 d_1 \cos\alpha_1)(\sin\alpha_1 + 1) \\ & \quad - \dfrac{2}{k_n}(F_2 \sin\alpha_1 + k_n d_1 \cos^2\alpha_1 + 2k_n d_1) \end{cases} \tag{10}$$

Similarly, the incremental constitutive relation for Hexagon B can also be obtained. Consequently, superimposing Hexagon A and Hexagon B, the total incremental force along direction $\vec{n}$ is $\delta\vec{F_1} = \delta\vec{F_1^a} + \delta\vec{F_1^b}$. The incremental constitutive relation of the 3D mesostructure is finally obtained.



(a) Dimension of Hexagon A (b) External forces (c) Force balance for grains 1 and 2

Figure 4: Mechanical description of Hexagon A.



(a) Dimension of Hexagon B (b) External forces (c) Force balance for grains 1 and 7

Figure 5: Mechanical description of Hexagon B.

(3) Stress averaging: Averaging the mesostress $\bar{\bar{\tilde{\sigma}}}$ taking place within all the mesostructures in the specimen of volume V can be performed as follows:

$$\bar{\bar{\sigma}} = \frac{1}{V} \iiint \omega(\theta, \varphi, \psi) \bar{\bar{P}}^{-1} \bar{\bar{\tilde{\sigma}}}(\vec{n}, \vec{t}, \vec{w}) \bar{\bar{P}} \sin\varphi d\varphi d\theta d\psi \tag{11}$$

where $\bar{\bar{\sigma}}$ is the macro-stress tensor operating on the specimen scale. For an isotropic specimen, the distribution function $\omega(\theta, \varphi, \psi)$ is uniform with $\theta \in [0, 2\pi[, \varphi \in [0, \pi], \psi \in [0, 2\pi[$ (θ, φ, ψ are the Euler angles). The mesostress $\bar{\bar{\tilde{\sigma}}}(\vec{n}, \vec{t}, \vec{w})$ with respect to the local frame can be computed from the local variables (Figure 4 and Figure 5) using the Love-Weber formula [7, 10, 19, 21]:

$$\begin{aligned}
\widetilde{\sigma}_{11}(\vec{n}, \vec{t}, \vec{w}) &= 4N_1 d_1 \cos^2\alpha_1 + 4T_1 d_1 \cos\alpha_1 \sin\alpha_1 + 2N_2 d_2 \\
&\quad + 4N_3 d_3 \cos^2\alpha_2 + 4T_3 d_3 \cos\alpha_2 \sin\alpha_2 + 2N_4 d_4 \\
\widetilde{\sigma}_{22}(\vec{n}, \vec{t}, \vec{w}) &= 4N_1 d_1 \sin^2\alpha_1 - 4T_1 d_1 \cos\alpha_1 \sin\alpha_1 \\
\widetilde{\sigma}_{33}(\vec{n}, \vec{t}, \vec{w}) &= 4N_3 d_3 \sin^2\alpha_2 - 4T_3 d_3 \cos\alpha_2 \sin\alpha_2 \\
\widetilde{\sigma}_{ij}(\vec{n}, \vec{t}, \vec{w}) &= 0 \quad \text{when} \quad i \neq j
\end{aligned} \tag{12}$$

The principal components of the mesostress tensor are calculated from the internal forces acting within the mesostructure. Besides, off-diagonal components can be simply considered as nil, because the mesostructure with respect to $(\vec{n}, \vec{t}, \vec{w})$ always offsets the one with respect to $(-\vec{n}, -\vec{t}, -\vec{w})$ in off-diagonal components when integrated.

Notably, the local void ratio is related to the opening angle, which is not related to local anisotropy. The opening angle $\alpha_{1(2)}$ is a geometrical parameter (Figure 4 and Figure 5). The opening angle, together with the components l_1, l_2, l_3, determine the initial shape of the hexagons as well as the local void ratio of the mesostructure. For a virgin specimen, the initial opening angle is denoted as α_0. Then, $\alpha_1 = \alpha_2 = \alpha_0$.

The initial void ratio $e_0(\vec{n},\vec{t},\vec{w})$ of each mesostructure belonging to the local frame $(\vec{n},\vec{t},\vec{w})$ can be estimated using the initial opening angle α_0 as follows:

$$e_0 = -\frac{4}{\pi}\cos^3\alpha_0 - \frac{6}{\pi}\cos^2\alpha_0 + \frac{4}{\pi}\cos\alpha_0 + \frac{6}{\pi} - 1 \tag{13}$$

3. Julia-based implementation of the 3D-H model

This section illustrates some implementation aspects of the 3D-H model based on the Julia programming language. The entire code of 3D-H model is shown in the form of some listings. An overall flow chart for the solution process is presented, covering all functions.

3.1. Global mixed loading control

The numerical implementation of the 3D-H model is based on a global mixed loading control framework, in which stress, strain or a mixture of both can be used as input. The mixed loading control adopts a strategy which combines the implicit algorithms with a general loading control. To achieve this goal conveniently, the loading control could be expressed by means of linearized constraints. A formula suggested by [1] can be employed, in which the loading condition can be expressed as:

$$\mathbf{S}\boldsymbol{\Delta\sigma}_{n+1} + \mathbf{E}\boldsymbol{\Delta\varepsilon}_{n+1} = \boldsymbol{\Delta}\mathbf{X}_{n+1} \tag{14}$$

where $\mathbf{S}$ and $\mathbf{E}$ are constraint matrices given by elementary tests, as shown in Table 1 for typical loading paths, whereas $\boldsymbol{\Delta}\mathbf{X}_{n+1}=[\Delta x_1, \Delta x_2, \cdots, \Delta x_i]^T_{n+1}$ is the imposed driven vector, and i is the number of degrees of freedom. If $i = 6$ (as shown in Table 1), there are 12 unknowns in Equation 14 but only 6 equations at each loading

step n, the relation between strain and stress increments has to be added, expressed as:

$$\mathbf{\Delta\sigma}_{n+1} = \mathbf{D}^{ep}_{n+1}\mathbf{\Delta}\varepsilon_{n+1} \tag{15}$$

where $\mathbf{D}^{ep}_{n+1}$ is the elastoplastic matrix. To solve Equation 14 and Equation 15, two separate levels of the Newton iteration scheme have been proposed by [42]; the first one solves the loading condition (Equation 14), whereas the other solves the constitutive equations (Equation 15). However, this procedure is not well adapted to a micro-mechanical model, because of the difficulty in obtaining a consistent matrix. This study suggests two different levels of predictor-corrector algorithms to solve Equation 14 and Equation 15. To solve the equation of the linearized constraints, Equation 14 can be rewritten as:

$$\mathbf{R}_{n+1} = \mathbf{S\Delta\sigma}_{n+1} + \mathbf{E\Delta\varepsilon}_{n+1} - \mathbf{\Delta X}_{n+1} \tag{16}$$

where the residual $\mathbf{R}_{n+1} = 0$ contains the same solution as in Equation 14. For each increment, 6 unknown stress increments and 6 unknown strain increments need to be solved by Equation 16. Combining Equation 15 and Equation 16, the strain or stress increments can first be predicted through the use of the elastic matrix $\mathbf{D}^e$. The obtained stress increments, used to calculate the force increments, are then corrected to take into account the plastic condition by the local corrector which will be presented in the following section. At the end of the kth elastic prediction, the residuals need to be calculated:

$$\mathbf{R}^{(k+1)}_{n+1} = \Delta\mathbf{X}^{(0)}_{n+1} - \left[\mathbf{S}\Delta\boldsymbol{\sigma}^{(k+1)}_{n+1} + \mathbf{E}\Delta\boldsymbol{\varepsilon}^{(k+1)}_{n+1}\right] \tag{17}$$

The relative error $\left\|\mathbf{R}^{(k+1)}_{n+1}\right\| \Big/ \left\|\boldsymbol{\sigma}^{(k+1)}_{n+1}\right\| \leq RTOL$ is computed at the end of each iteration to guarantee that the constraints are fully imposed. The relative error

should satisfy the given tolerance $RTOL$ set as 10^{-4}. If this is not the case, the residuals, viewed as correctors, are imposed as new constraints:

$$\mathbf{\Delta X}_{n+1}^{(k+1)} = \mathbf{R}_{n+1}^{(k+1)} \tag{18}$$

Stress and strain are also updated after each iteration. For a displacement driven finite element code, the strain increments at each Gauss point are given. Under this condition, $\mathbf{S}$ is null in Equation 14, the constraints are strain increments; thus, no iteration is needed to solve Equation 14.

3.2. Flow chart

This section shows the entire program of the 3D-H model implemented in the Julia language. As an example, the drained triaxial compression test with 200 kPa of confining pressure is considered. The program includes a main program (main.jl shown in Listing 1) and some subroutines (such as mixload.jl, hexagon.jl and so on). The flow chart shown in Figure 6 can be illustrated in the following steps:

1. The main program main.jl (shown in Listing 1) first reads model parameters from the sample() function. Nt and Np are the numbers of angle subdivisions of θ and φ, respectively. Note that even if the initial states of all mesostructures are the same, their subsequent evolution might be different. Thus the dimension of microstate variable matrices (such as state_F, state_d and so on) needs to be consistent with the number of angle subdivisions. Then, the mixload() function is invoked.

2. As shown in Listing 2, this function is mainly used to control the mixed loading, as introduced in subsection 3.1. The loading() and stiffCC() functions are called to generate the $\mathbf{S}$, $\mathbf{E}$, and $\mathbf{D}^{ep}$ matrices (mentioned in Equation 14 and

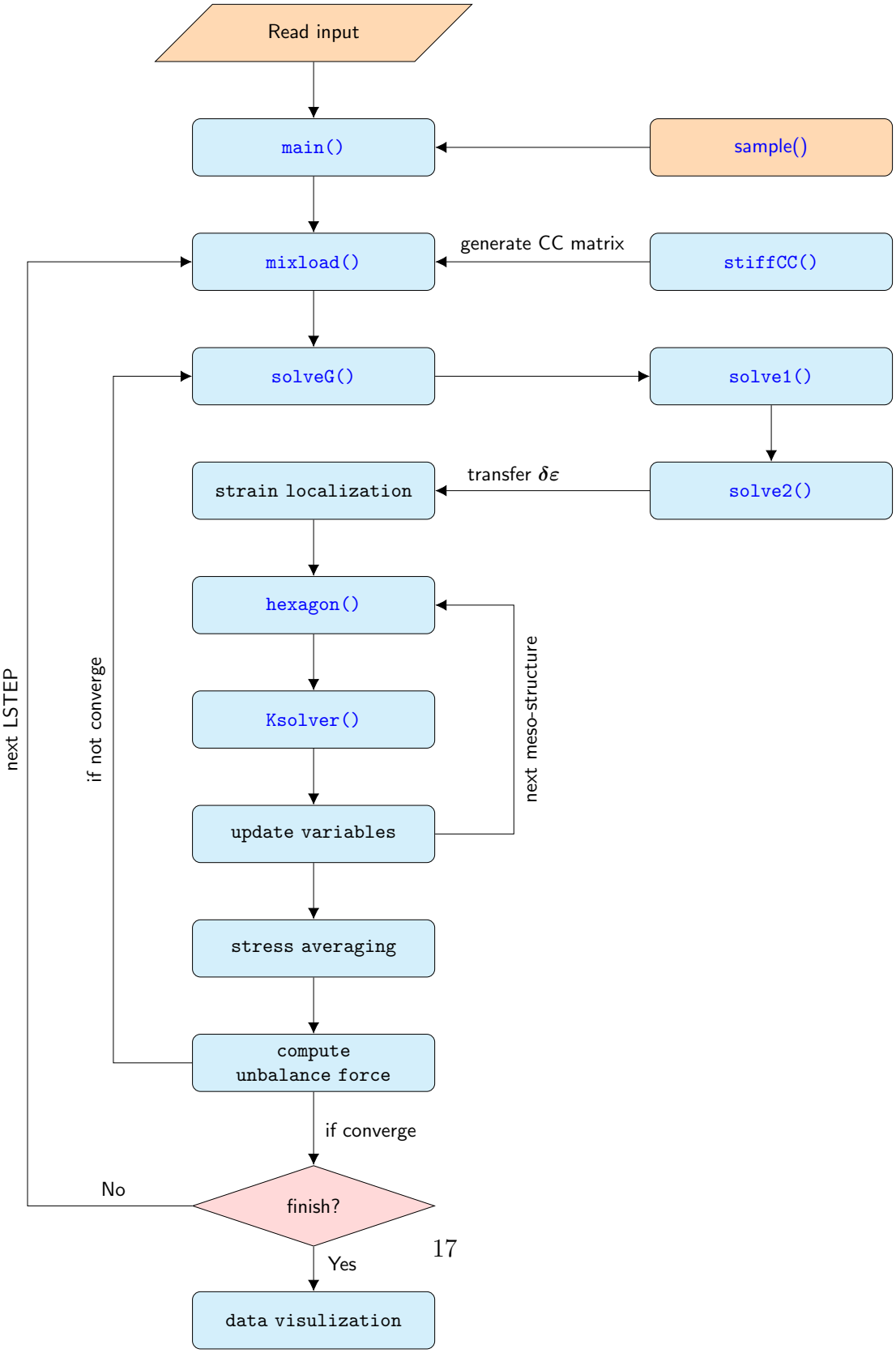


Figure 6: Flow chart of 3D-H model in Julia language.

Equation 15), respectively. Thereafter, the target strain, stress or a mixture of both is loaded by Nstep.

3. For each Nstep, solveG(), solve1() and solve2() are employed to estimate a suitable incremental strain tensor. Then, the kinematic localisation (Equation 1) is used to compute the deformation of mesostructures $\delta\mathbf{L}$ on the mesoscale.

4. For a certain direction, $\delta\mathbf{L}$ is considered as an input of function hexagon() (shown in Listing 7). The function Ksolver() is used to compute variables $\delta d_1, \delta d_2, \delta\alpha_2$ for Hexagon A and $\delta d_3, \delta d_4, \delta\alpha_3$ for Hexagon B (depicted in Figure 4 and Figure 5). These variables are then used to compute incremental local forces $\delta N_1, \delta N_2, \delta T_1$ for Hexagon A and $\delta N_3, \delta N_4, \delta T_3$ for Hexagon B in terms of contact law (see detail in Appendix A). Then, all local variables are updated.

5. When all state variables are updated, the macrostress tensor can be obtained using stress averaging (Equation 11). Thus the residuals $\mathbf{R}_{n+1}^{k+1}$ in Equation 17 are computed. If $\left\|\mathbf{R}_{n+1}^{(k+1)}\right\| \Big/ \left\|\boldsymbol{\sigma}_{n+1}^{(k+1)}\right\| \geq 10^{-4}$, steps 3-5 are repeated until the unbalanced force converges, given a small enough value.

6. Finally, the program goes to the next LSTEP until the end of loading. The resultant data are stored for visualisation at the end of the program.

Listing 1: main

```
include("sample.jl"); include("hexagon.jl"); include("Ksolver.jl")
include("loading.jl"); include("CC.jl"); include("solveG.jl")
include("solve1.jl"); include("solve2.jl"); include("mixload.jl")
global vol=1.5316e-5; const Nt=30; const Np=90; niter_max=100
(kn,kt,al0,phig,d0) = sample()
```

```
state_F=zeros(Nt,Np, 6)
state_d=zeros(Nt, Np, 4)
state_al=zeros(Nt, Np, 2)
state_broken=zeros(Nt, Np)
for i in 1:Nt
    for j in 1:Np
        for k in 1:2; state_al[i,j,k]=al0; end
        for k in 1:4; state_d[i,j,k]=d0; end
        state_broken[i,j]=0; end; end
# loading
iso_data = mixload("iso")
tri_data = mixload("tri")
```

Listing 2: mixload

```
function mixload(method)
    vmix0, vmix1, code, Nstep=loading(method)
    dvmix = (vmix1 - vmix0) / Nstep[1]
    dvmixit=zeros(6)
    eps=zeros(6); deps=zeros(6); sigunb=zeros(6); sigr=zeros(6)
    sigk=zeros(6); sigtotal=zeros(6); stiffCC=CC()
    sig11=[]; sig22=[]; sig33=[]; eps11=[]; eps22=[]; eps33=[]
    for LSTEP in 1:Nstep
        for i in eachindex(dvmix); dvmixit[i]=dvmix[i]; end
        for niter in 1:niter_max
            stiffCC, dvmixit ,deps=solveG(6,code,stiffCC,dvmixit,deps,6)
            sigk=zeros(6)
            for i in 1:Nt; for j in 1:Np
                sigr=hexagon(LSTEP,deps,i,j); sigk=sigk+sigr/vol
            end; end
            global vol=vol*(1-(sum(deps)))
            eps=eps+deps
            for i in 1:6
                sigunb[i]=0; dvmixit[i]=0
                if code[i]<0.5
```

```
                    sigunb[i] = (vmix0[i]+ dvmix[i]*LSTEP) - sigk[i]
                    dvmixit[i]=sigunb[i]; end; end
            sum1=0.; sum2=0.
            for i in 1:6
                sum2 = sum2+sigunb[i]^2
                sum1 = sum1+ sigk[i]^2; end
            sum2=sqrt(sum2)/6.; sum1=sqrt(sum1)/6.
            if niter == niter_max
                global niter_last=niter_max; end
            if sum2 < 0.005*sum1; global niter_last=niter; break; end
        end
        global data=[sig11,sig22,sig33,eps11,eps22,eps33]
    end
    return data
end
```

Listing 3: solveG

```
function solveG(NE,NU,stiff_old,F,u,mband)
    F_sol=zeros(6); F1=zeros(6); a1=zeros(6,6)
    for i in 1:NE
        F_sol[i]=F[i]; F1[i]=F[i]
        for j in 1:NE; a1[i,j]=stiff_old[i,j]; end; end
    for i in 1:NE
        if NU[i]>0.5
            for j in 1:NE; a1[i,j]=0; a1[j,i]=0; end
            a1[i,i]=1
            for j in 1:NE; F1[j]=F1[j] - F[i] * stiff_old[j,i]; end; end; end
    for i in 1:NE; if NU[i]>0.5; F1[i]=F[i]; end; end
    NE, a1, F_sol, mband= solve1(NE, a1, F_sol, mband)
    NE, a1, F_sol, u, F1, mband = solve2(NE, a1, F_sol, u, F1, mband)
    F_sol=F
    for i in 1:NE; if NU[i]>0.5; F_sol[i]=0
    for j in 1:NE; F_sol[i]=F_sol[i]+stiff_old[i,j]*u[j]; end; end; end
    return stiff_old, F,u; end
```

Listing 4: loading

```
function loading(method)
    # stress control=0 ; strain control =1
    sigc=2e5
    epsa=0.1
    if method == "iso"
        step=500
        control=[0, 0, 0, 1., 1., 1.]
        initial=[0.,  0.,  0.,  0.,  0.,  0.]
        final=[sigc,    sigc,   sigc,    0.,  0.,  0.]
    elseif method == "tri"
        step=3000
        control=[1., 0., 0., 1., 1., 1.]
        initial=[0.,  sigc,  sigc,  0.,  0.,  0.]
        final=[epsa,    sigc,   sigc,    0.,  0.,  0.]
    end
    return initial, final, control, step
end
```

Listing 5: solve1

```
function solve1(n, A, p, mband)
    for i in 1:n; jj=n
        if (i+mband-1)<n; jj=i+mband-1; end
        for j in i:jj
            summ=A[i,j]; jj=1
            if (j-mband+1)>1; jj=j-mband+1; end
            for k in i-1:-1:jj; summ=summ-A[i,k]*A[j,k]; end
            if i==j; p[i]=sqrt(summ); else; A[j,i]=summ/p[i]; end; end; end
    return n, A, p, mband; end
```

Listing 6: solve2

```
function solve2(n, A, p, x, b, mband)
    for i in 1:n
        summ=b[i]; jj=1
```

```
        if (i-mband+1)>1; jj=i-mband+1; end
        for k in i-1:-1:jj; summ=summ-A[i,k]*x[k]; end
        x[i]=summ/p[i]; end
    for i in n:-1:1
        summ=x[i]; jj=n
        if (i+mband-1) < n; jj=i+mband-1; end
        for k in i+1:jj; summ=summ-A[k,i]*x[k]; end
        x[i]=summ/p[i]; end
    return n, A, p, x, b, mband; end
```

Listing 7: hexagon

```
function hexagon(LSTEP,deps,i,j)
    (kn,kt,al0,phig,d0) = sample()
    al2=state_al[i,j,1]; al3=state_al[i,j,2]
    d1=state_d[i,j,1]; d2=state_d[i,j,2]
    d3=state_d[i,j,3]; d4=state_d[i,j,4]
    Fn1=state_F[i,j,1]; Ft1=state_F[i,j,2]; Fn2=state_F[i,j,3]
    Fn3=state_F[i,j,4]; Ft3=state_F[i,j,5]; Fn4=state_F[i,j,6]
    ten=state_broken[i,j]
    sigr = zeros(6)
    l1=d2+d1*2*cos(al2); l2=d1*2*sin(al2); l3=d3*2*sin(al3)
    if abs(Ft1)<=tan(phig)* Fn1; ela2=1; else; ela2=0; end
    if abs(Ft3)<=tan(phig)* Fn3; ela3=1; else; ela3=0; end
    if Ft1<0; cp2=-1; else; cp2=1; end
    if Ft3<0; cp3=-1; else; cp3=1; end
    theta=(i-1)*pi/Nt/2.; phi=(j-1)*pi/Np/2.
    Pn=zeros(3,3); depsro=zeros(3,3)
    if ten<.5
        Pn=[cos(phi)                    0.          -sin(phi);
        sin(theta)*sin(phi)         cos(theta)   sin(theta)*cos(phi);
        sin(phi)*cos(theta)         -sin(theta)  cos(theta)*cos(phi);]
        depsro=[deps[1]     deps[4]     deps[5];
        deps[4]     deps[2]     deps[6];
        deps[5]     deps[6]     deps[3];]
```

```
        meps=Pn'*depsro*Pn
        dl1 = -l1 *meps[1,1]; dl2 = -l2 *meps[2,2]; dl3 = -l3 *meps[3,3]
        (dd1,dd2,dal2)=Ksolver(al2,d1,Fn1,Ft1,kn,kt,phig,cp2,dl1,dl2,ela2)
        (dd3,dd4,dal3)=Ksolver(al2,d1,Fn3,Ft3,kn,kt,phig,cp3,dl1,dl3,ela3)
        dFn1=-kn*dd1; dFt1=kt*d1*dal2; dFn2=-kn*dd2
        dFn3=-kn*dd3; dFt3=kt*d3*dal3; dFn4=-kn*dd4
    if abs(Ft1+dFt1) < abs(tan(phig)*(Fn1-kn*dd1))
        ela2=1; d1+=dd1; d2+=dd2; al2+=dal2
        l1+=dl1; l2+=dl2; Fn1+=dFn1; Ft1+=dFt1; Fn2+=dFn2
    else; ela2=0
        (dd1,dd2,dal2)=Ksolver(al2,d1,Fn1,Ft1,kn,kt,phig,cp2,dl1,dl2,ela2)
        dFn1=-kn*dd1; dFt1=cp2*tan(phig)*(Fn1+dFn1)-Ft1
        dFn2=-kn*dd2; d1+=dd1; d2+=dd2; al2+=dal2
        l1+=dl1; l2+=dl2; Fn1+=dFn1; Ft1+=dFt1; Fn2+=dFn2
        if abs(Ft1+dFt1) < abs(tan(phig)*(Fn1-kn*dd1)); ela2=1; end; end
    if abs(Ft3+dFt3) < abs(tan(phig)*(Fn3-kn*dd3))
        ela3=1; d3+=dd3; d4+=dd4; al3+=dal3
        l1+=dl1; l3+=dl3; Fn3+=dFn3; Ft3+=dFt3; Fn4+=dFn4
    else; ela3=0
        (dd3,dd4,dal3)=Ksolver(al3,d3,Fn3,Ft3,kn,kt,phig,cp3,dl1,dl3,ela3)
        dFn3=-kn*dd3; dFt3=cp3*tan(phig)*(Fn3+dFn3)-Ft3
        dFn4=-kn*dd4; d3+=dd3; d4+=dd4; al3+=dal3
        l1+=dl1; l3+=dl3; Fn3+=dFn3; Ft3+=dFt3; Fn4+=dFn4
        if abs(Ft3+dFt3) < abs(tan(phig)*(Fn3-kn*dd3)); ela3=1; end; end
    else; sigr[1:3]=[0., 0., 0.]; end
    sigt1a=4*Fn1*d1*cos(al2)^2+4*Ft1*d1*cos(al2)*sin(al2)+2*Fn2*d2
    sigt1b=4*Fn3*d3*cos(al3)^2+4*Ft3*d3*cos(al3)*sin(al3)+2*Fn4*d4
    sigt1=sigt1a+sigt1b
    sigt2=4*Fn1*d1*sin(al2)^2-4*Ft1*d1*cos(al2)*sin(al2)
    sigt3=4*Fn3*d3*sin(al3)^2-4*Ft3*d3*cos(al3)*sin(al3)
    sigtt=[sigt1  0  0; 0  sigt2  0; 0  0  sigt3]
    msig=Pn*sigtt*Pn'
    sigr[1]=msig[1,1]*sin(phi)
    sigr[2]=msig[2,2]*sin(phi)
    sigr[3]=msig[3,3]*sin(phi)
```

```
    sigr[4:6]=[0., 0., 0.]
    if LSTEP >1
        if (Fn1<0)|(Fn2<0)|(Fn3<0)|(Fn4<0)
            ela2=1.; ela3=1.; sigt2=0; sigt3=0
            sigt1a=0.; sigt1b=0.; sigr=0.5*sigr; ten=1; end; end
    state_al[i,j,1]=al2; state_al[i,j,2]=al3; state_d[i,j,1]=d1
    state_d[i,j,2]=d2; state_d[i,j,3]=d3; state_d[i,j,4]=d4
    state_F[i,j,1]=Fn1; state_F[i,j,2]=Ft1; state_F[i,j,3]=Fn2
    state_F[i,j,4]=Fn3; state_F[i,j,5]=Ft3; state_F[i,j,6]=Fn4
    state_broken[i,j]=ten
    return sigr; end
```

Listing 8: Ksolver

```
function Ksolver(al,d,Fn1,Ft1,kn,kt,phig,cp,dl1,dl2,ela)
x=[0, 0, 0]
if ela>0.5
K=[ 2*cos(al)       1       -2*d*sin(al);
        2*sin(al)       0       2*d*cos(al);
cos(al)         -1      ((Fn1-kt*d)*sin(al)-Ft1*cos(al)-kt*d)/kn]
L=[dl1,   dl2,  0]; x=inv(K)*L
else
K=[ 2*cos(al)       1       -2*d*sin(al);
        2*sin(al)       0       2*d*cos(al);
cos(al)+cp*tan(phig)*(sin(al)+1)  -1  (Fn1*sin(al)-Ft1*cos(al))/kn]
L=[dl1,  dl2,  sin(al)*(cp*tan(phig)*Fn1-Ft1)/kn]; x=inv(K)*L; end
return x; end
```

4. Numerical examples

This section presents some numerical examples demonstrating the performance advantage of Julia as a development tool for MM models. As illustrated in section 3, all simulations were carried out using in-house 3D-H model codes written in Julia. For

tpye of tests	$\mathbf{S}$	$\mathbf{E}$	$\mathbf{\Delta X}_{n+1}$
ISO	$S_{ij}=1, \quad i=j$ $S_{ij}=0, \quad i\neq j$	$E_{ij}=0$	$[\Delta x_1, \Delta x_1, \Delta x_1, 0, 0, 0]^T$
ASO	$S_{ij}=1, \quad i=j$ $S_{ij}=0, \quad i\neq j$	$E_{ij}=0$	$[\Delta x_1, \Delta x_2, \Delta x_3, 0, 0, 0]^T$
TX-CD	$S_{ij}=1, \quad j=i+1$ $S_{ij}=0, \quad \text{else}$	$E_{ij}=1, \quad i=6 \quad j=1$ $E_{ij}=0, \quad \text{else}$	$[0, 0, 0, 0, 0, \Delta x_1]^T$
TX-CU	$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \Delta x_1 \end{bmatrix}$

Table 1: Constraint matrices for mixed controls

comparison purposes, the DEM was considered as a benchmark. The computational accuracy and efficiency of both methods are compared in this section.

4.1. DEM model setup

The DEM employed in this study uses the soft-sphere approach originally developed by [8]. The equilibrium of individual particles is determined by the equations shown in Equation 19. The forces and torques applied to a particle i in contact with particle j are considered to be as follows:

$$\begin{aligned} m_i \frac{d\boldsymbol{v}_i}{dt} &= \sum(\mathbf{F}_{ij}^n + \mathbf{F}_{ij}^t) \\ I_i \frac{d\boldsymbol{\omega}_i}{dt} &= \sum(\mathbf{R}_i \times \mathbf{F}_{ij}^t - \boldsymbol{\tau}_{ij}^r) \end{aligned} \tag{19}$$

where m_i, I_i, $\boldsymbol{v}_i$ and $\boldsymbol{\omega}_i$ are, respectively, the mass, moment of inertia, translational velocity and rotational velocity of particle i. $\mathbf{F}_{ij}^n$ and $\mathbf{F}_{ij}^t$ are the normal and tangential forces caused by interaction between particles i and j at the current time-step. $\mathbf{R}_i$ is the vector between the centre of particle i and the contact point where force $\mathbf{F}_{ij}^t$ is applied. $\boldsymbol{\tau}_{ij}^r$ is the torque caused by rolling friction.

An adapted Hertz-Mindlin contact model was employed. The contact force could be decomposed into normal and tangential non-linear contact forces. They consist of two terms, the former standing for the non-linear elastic Hertz model in the normal direction and the linear elastic Mindlin model in the tangential direction [12, 25], whereas the latter represents a dissipative term. It is added to account for energy dissipation during collisions through inelastic deformation and friction. Thus the normal and tangential forces, $\mathbf{F}_{ij}^n$ and $\mathbf{F}_{ij}^t$, are given by:

$$\begin{aligned} \mathbf{F}_{ij}^n &= -\tfrac{4}{3}E^*\sqrt{R^*\delta_{ij}^n}\boldsymbol{\delta}_{ij}^n - 2\sqrt{\tfrac{5}{6}}\psi\sqrt{C_n m^*}\boldsymbol{v}_{ij}^n \\ \mathbf{F}_{ij}^t &= -8G^*\sqrt{R^*\delta_{ij}^n}\boldsymbol{\delta}_{ij}^t - 2\sqrt{\tfrac{5}{6}}\psi\sqrt{C_t m^*}\boldsymbol{v}_{ij}^t \end{aligned} \tag{20}$$

where E^* is the equivalent Young's modulus of the two colliding particles, defined by $\frac{1}{E^*} = \frac{1-\mathcal{V}_i^2}{E_i} + \frac{1-\mathcal{V}_j^2}{E_j}$, where $\mathcal{V}_i$ and $\mathcal{V}_j$ are the Poisson's ratios; R^* is the equivalent radius, defined by $\frac{1}{R^*} = \frac{1}{R_i} + \frac{1}{R_j}$; m^* is the equivalent mass, defined by $\frac{1}{m^*} = \frac{1}{m_i} + \frac{1}{m_j}$, $\boldsymbol{v}_{ij}^n$ and $\boldsymbol{v}_{ij}^t$ are the normal and tangential components of the relative velocity at the contact; $\boldsymbol{\delta}_{ij}^n$ is the normal contact overlap, given by $|\boldsymbol{\delta}_{ij}^n| = R_i + R_j - d_{ij}$, where d_{ij} is the distance between the centres of particles; $\boldsymbol{\delta}_{ij}^t$ is the tangential contact overlap, given by the integral of the tangential relative velocity through the collision time from the collision's beginning, i.e., $|\boldsymbol{\delta}_{ij}^t| = \int_n^t |\boldsymbol{v}_{ij}^t| dt$, $C_n = 2E^*\sqrt{R^*\delta_{ij}^n}$ and $C_t = 8G^*\sqrt{R^*\delta_{ij}^n}$ are the normal and tangential contact stiffness, where G^* is the equivalent shear modulus, defined as $\frac{1}{G^*} = \frac{2(2-\mathcal{V}_i)(1+\mathcal{V}_i)}{E_i} + \frac{2(2-\mathcal{V}_j)(1+\mathcal{V}_j)}{E_j}$, ψ; and ψ is the damping ratio coefficient that is a function of the coefficient of restitution, ε, given by $\psi = \ln(\varepsilon)/\sqrt{\ln^2(\varepsilon) + \pi^2}$.

4.2. Parameter calibration and model prediction

Consider a representative volume element (RVE) of granular assembly consisting of spheres for the 3D-H model and the DEM model. For the latter, the sample is considered as a cube with the size of $22 \times 22 \times 22\text{mm}^3$. The d_{50} is 1.5mm. The total particle number is controlled to 27000. For comparison purposes, the same particle number is selected for the 3D-H model. Although the concept of particle number does not exist as such in this model, the concept of angle subdivision of θ and φ (shown in Figure 2) could still be considered as an equivalent. For example, assuming only one mesostructure (consisting of 10 particles) exists along each direction, 27000 particles means that the angle subdivision of θ and φ can be $30 \times 90 = 2700$ or $45 \times 60 = 2700$, and so on.

An automatic parameter identification method proposed by [14–16, 40] is applied, in order to provide the best fit to a single isotropically compressed drained triaxial

test confined at 200 kPa, based on the experimental data from [6, 22, 23]. This sand (called Labenne sand) is well characterised from a geotechnical point of view and has been adopted in many studies [2, 13, 18]. The best fit of mechanical and volumetrical responses is shown in Figure 7. The parameters used in the 3D-H and DEM models are reported in Table 2 and Table 3. Then, the same set of parameters is used to predict the experimental curves at 100kPa and 300kPa, providing the performance of the 3D-H and DEM models. It should be noted that p is mean stress defined as $p = \mathrm{tr}(\bar{\bar{\sigma}})$, q is deviatoric stress defined as $q = \sqrt{\frac{1}{3}\mathrm{tr}(\bar{\bar{\sigma}})}$, ε_a is axial strain, ε_v is volumetic strain defined as $\varepsilon_v = \mathrm{tr}(\bar{\bar{\varepsilon}})$.

Table 2: Parameters used in the 3D-H model

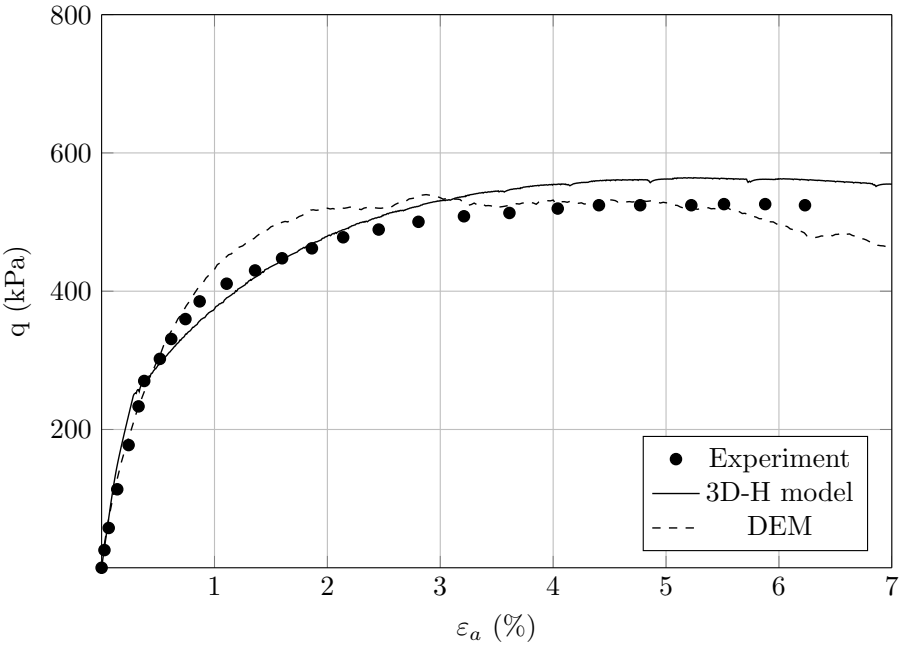
k_n(N/m)	k_t/k_n	e_0	$\varphi_g(^\circ)$	d(mm)
8.0×10^5	0.6	0.45	15	1.25

Table 3: Parameters used in DEM model

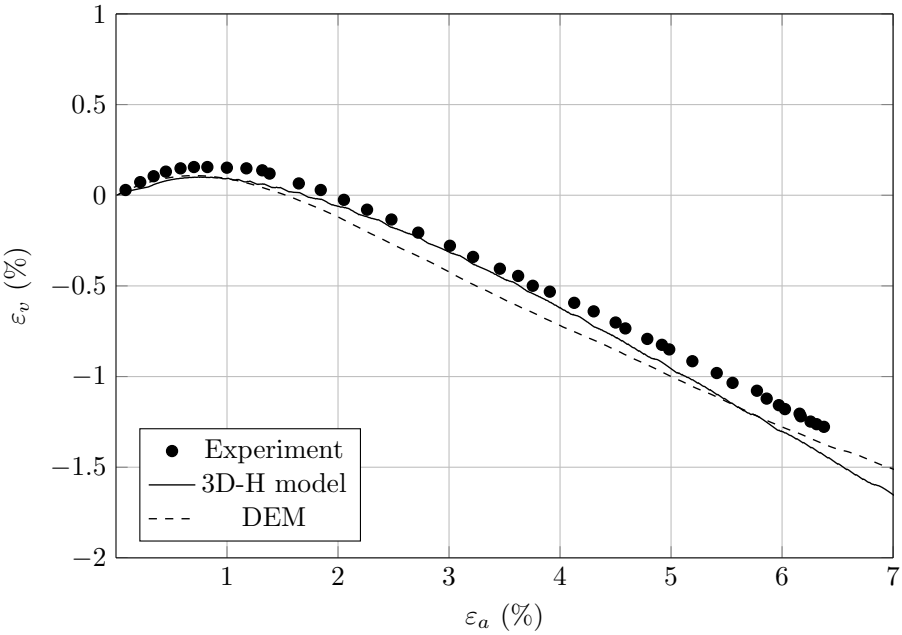
E(Pa)	$\mathcal{V}$	μ_s	μ_r	ρ	ε	d_{50}(mm)
1.0×10^8	0.25	0.8	0.15	2630	0.5	1.25

4.3. Computation efficiency and accuracy

In most MM models, the orientation subdivision significantly influences the computation efficiency and accuracy. In this section, five kinds of different orientation subdivision are selected (listed in Table 4), which corresponds to five different REV samples with the equivalent number of particles in the DEM model.

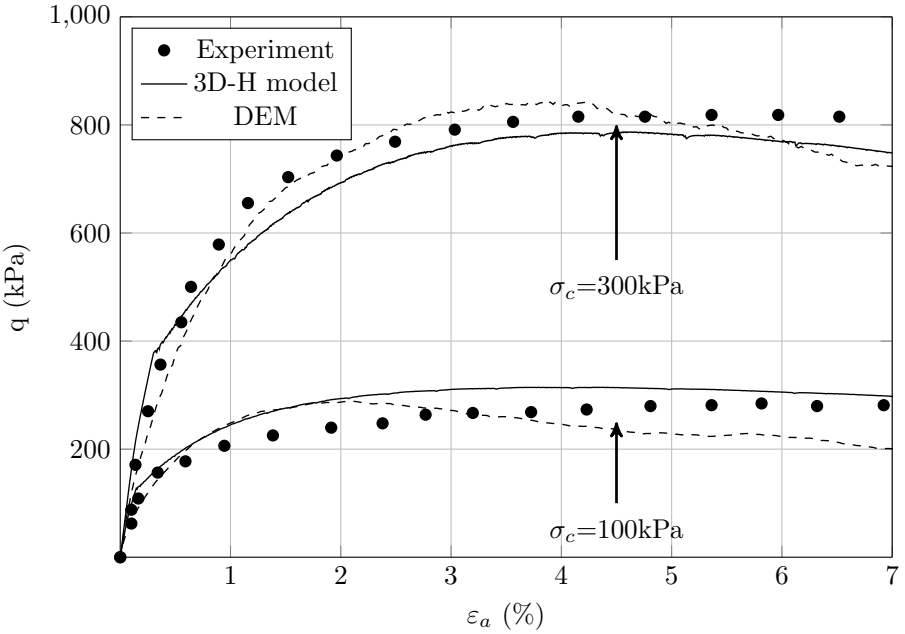


(a) Mechanical response

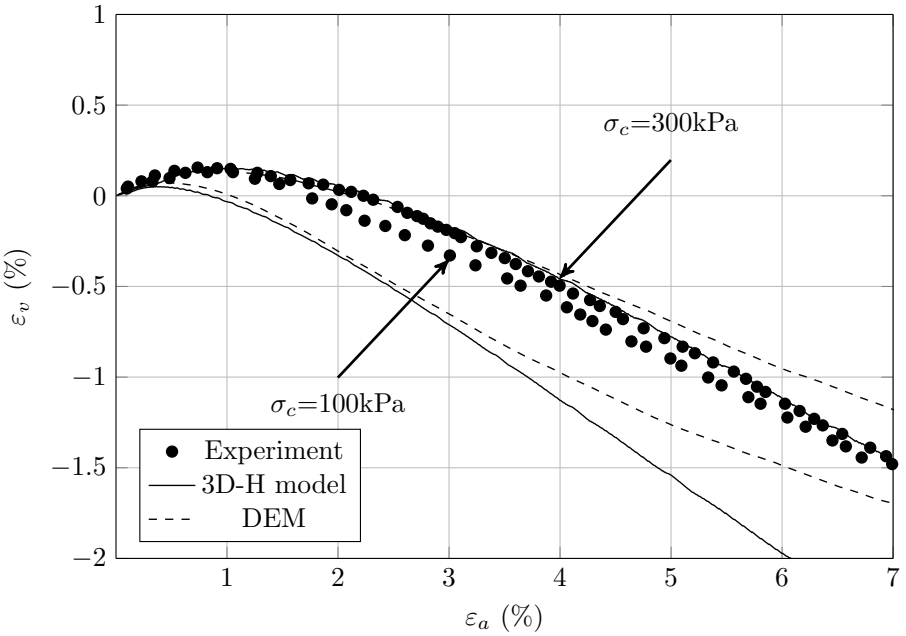


(b) Volumetric response

Figure 7: Parameters calibration on Labenne sand along triaxial loading path confined at 200kPa.



(a) Mechanical response



(b) Volumetric response

Figure 8: Model prediction on Labenne sand along triaxial loading path confined at 100kPa and 300kPa.

Table 4: Sample information summary. θ and φ are Euler angles shown in Figure 2.

DEM	DEM-9K	DEM-18K	DEM-27K	DEM-36K	DEM-54K
3D-H model	H-9K	H-18K	H-27K	H-36K	H-54K
Number of particles	9000	18000	27000	36000	54000
Number of dφ	30	60	90	60	60
Number of dθ	30	30	30	60	90

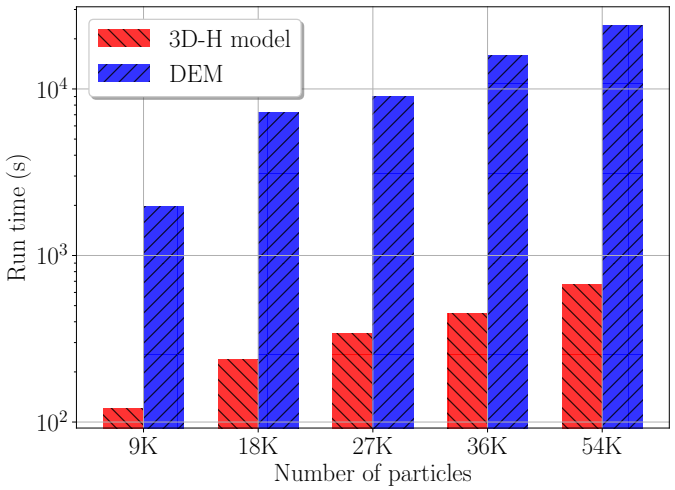


Figure 9: Comparison of computational efficiency between 3D-H model and DEM model.

All samples summarized in Table 4 are first subjected to isotropic compression up to 200kPa; then, triaxial loading is applied until 10% of axial strain. This simulation refers to the actual time taken to complete the analysis using a single thread on an Intel Core Xeon E5-2697A @2.60GHz. Figure 9 shows the comparison of computational efficiency between the two methods. Obviously, the computation efficiency of the MM model is much higher than that of the DEM model, which is a great advantage of the former. The reason for the significant difference in run-time is that the MM model uses a multiscale approach. In DEM, all contact information is stored and transferred step by step. Unlike DEM, MM contains statistical ideas, which means that contact information is expressed as a spatial distribution. For example, if some contacts are located in the same range (which is sufficiently small), these contacts could be represented by one contact, while the macroscopic constitutive behaviour does not change.

4.4. Proportional loading path

The performance of the 3D-H model can be illustrated from other loading paths. For the sake of illustration, a proportional loading path is then considered. After the isotropic confining stage, the specimen is subsequently subjected to a proportional strain loading path [9] defined as:

$$\begin{aligned} &\delta\varepsilon_1 = \text{positive constant} \\ &\delta\varepsilon_2 = \delta\varepsilon_3 \quad \text{(axisymmetric condition)} \\ &\delta\varepsilon_1 + 2R\delta\varepsilon_3 = 0 \quad (R \text{ constant for a given path}) \end{aligned} \tag{21}$$

This set of loading paths is a pure strain-controlled loading, with value R varying in different tests. Note that when value $R = 1$, the volumetric change of the specimen is imposed to be zero, corresponding to the well-known undrained (isochoric) triaxial

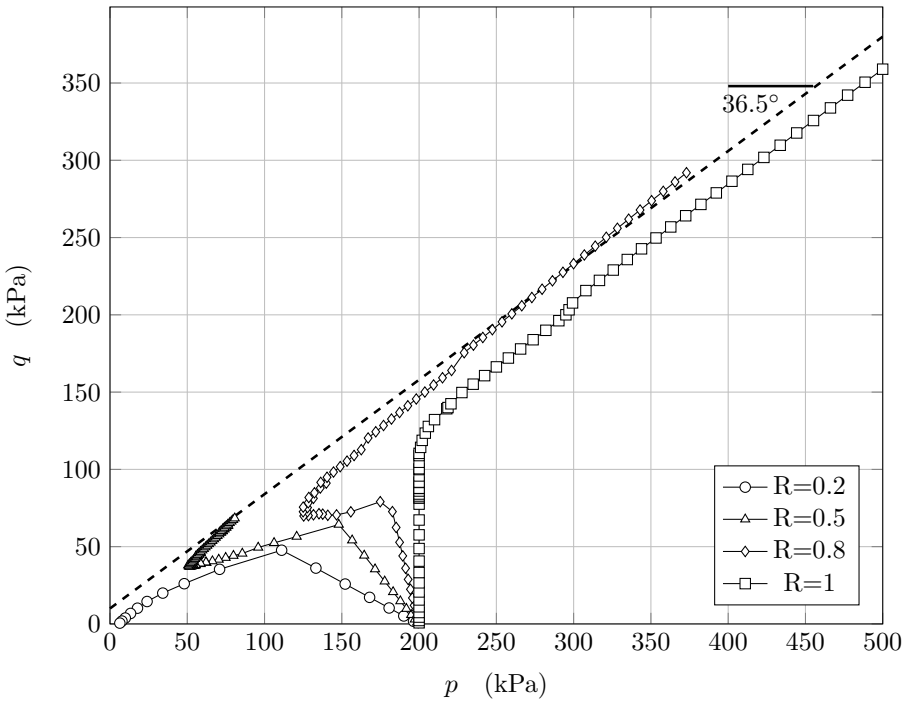


Figure 10: Mechanical response over a proportional loading path with different R values ($\delta\varepsilon_2 = \delta\varepsilon_3 = -\frac{1}{2R}\delta\varepsilon_1$).

test. The test can either follow a dilatant loading path with $0 < R < 1$ or a contractant loading path with $R > 1$, giving advantages to the test that enables the exploration of the model response along any direction in the incremental strain space.

Four different R values are considered, as shown in Figure 10. The mechanical response is presented by analysing the relation between the deviatoric stress variable $q = \sigma_1 - \sigma_2$ and the mean stress $p = (\sigma_1 + \sigma_2 + \sigma_3)/3$ by employing the parameters reported in Table 2. As observed in Figure 10, most of the tests can approach the Mohr-Coulomb line (the dashed line in Figure 10) and continuously increase the stress until reaching the critical state. However, when the R value decreases until a critical

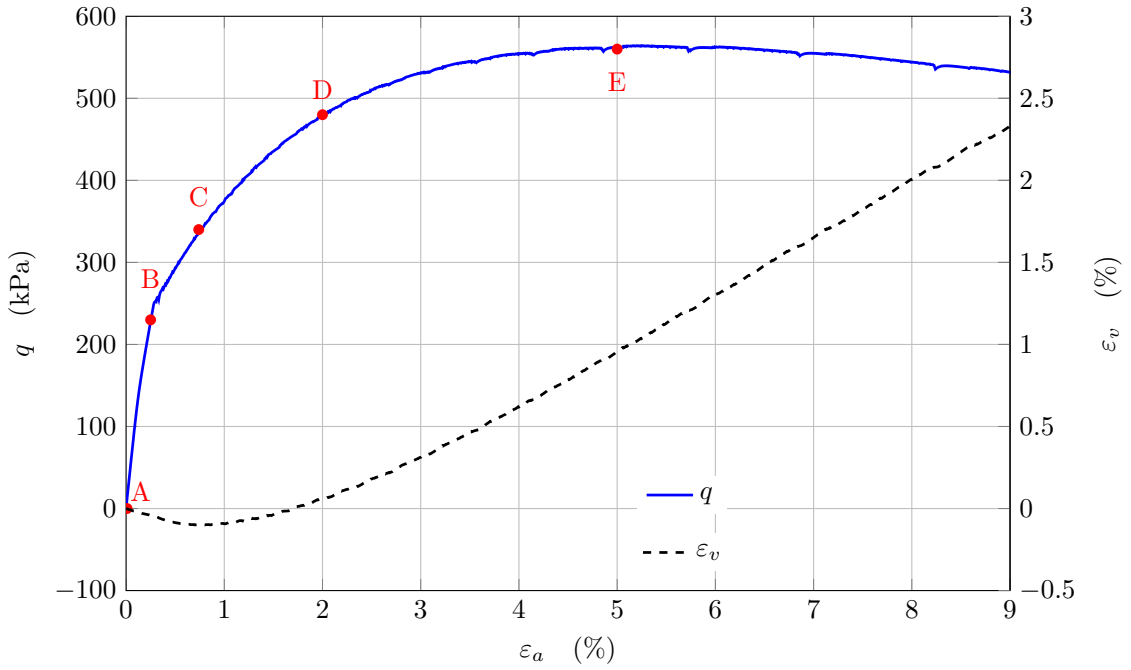


Figure 11: Mechanical and volumetric responses of 3D-H model along triaxial loading paths, confined at 200kPa

value $\bar{R} = 0.2$, liquefaction can be observed whereby stresses eventually vanish. This is a generalisation of what is observed along isochoric biaxial or triaxial paths, where a liquefaction mechanism occurs for loose specimens. Thus the investigation of proportional strain paths facilitates the checking of whether a certain failure mode may occur even for dense specimens. The ability of the microdirectional model and H-model to characterise the liquefaction has been proved [27, 28]. As expected, this ability has been inherited by the 3D-H model.

4.5. Linking between DEM and MM: Microscopic information

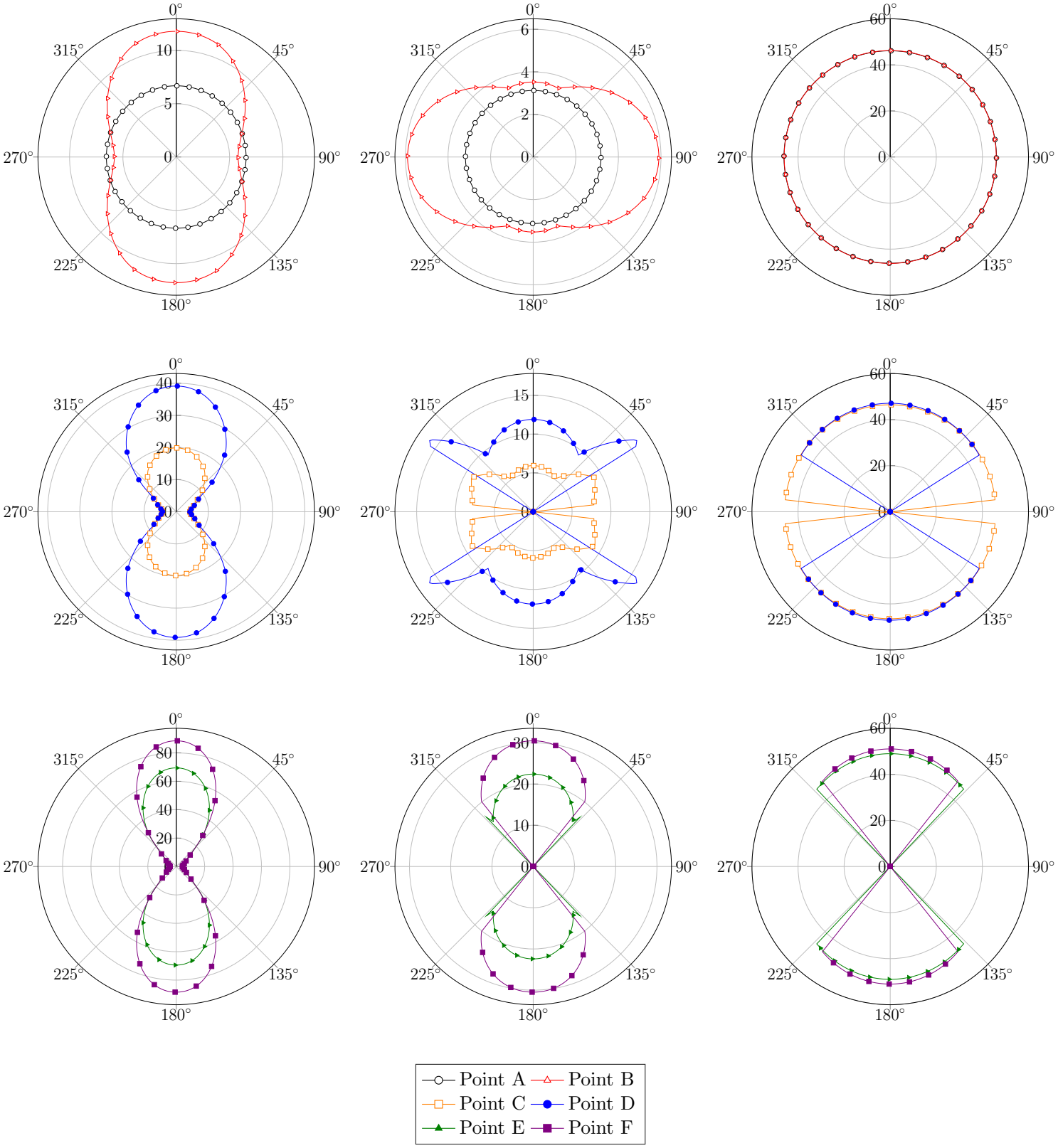


Figure 12: Angular distributions of mesoscopic variables by fixing $\theta = 0°$ and varying φ in polar coordinates at strain states, corresponding to points A-E shown in Figure 11. In each sub-figures, three figures are shown: the angular distribution of external force of mesostructure F_1 (left column); the angular distribution of external force of mesostructure F_3 (center column); the angular distribution of opening angle α_3 (right column).

As illustrated in the previous section, the MM model has a great advantage in terms of saving the computation resources. This means that this model has the potential to simulate large-scale engineering problems with millions of particles by using the current level of computer technology, which is very difficult in DEM based on the limitations of RAM and CPU. The question may arise: Why can DEM reproduce complicated macroscopic behaviour with a few parameters? The answer is: because the geometrical disorder is accounted for. This is one of the two sides that create the complexity on the macroscopic scale. The second side (which is less important) is the contact law. Therefore, one way to improve the MM model is for scientific researchers to analyse and compare the microscopic behaviour of DEM, which is impossible in phenomenological models. In this study, the triaxial loading path with 200 kPa is selected as an example to illustrate the analysis of microscopic information. Figure 11 shows the evolution of deviatoric stress (q) and volumetric strain (ε_v) versus axial strain (ε_a). Six states (corresponding points A-F) are selected: the state at the beginning of shear loading (A); the state at the end of the elastic phase (B); the maximum contraction state (C); the state before the stress peak (D); the state at the stress peak (E); and the state after the stress peak (F).

For the 3D-H model, the microscopic information should present a 3D sphere (for isotropic distribution). However, as a result of the symmetric confining stresses along σ_2 and σ_3, the angular distributions of mesoscopic variables are obviously symmetric about the $\vec{x_1}$ direction. In other words, they are absolutely the same by varying θ. Figure 12 shows angular distributions of mesoscopic variables by varying φ with $\theta = 0°$ at the selected six strain states A-F. The first remark is that angular distributions of F_1, F_3 and α_3 are isotropic (shown as circles). This is a prerequisite to verify that an MM model is correct. As the deviatoric strain loading that is applied (state B), F_1 becomes "peanut-like" in shape, where the direction of the maximum

value of F_1 coincides with the loading direction. F_3 shows a similar tendency but located along the perpendicular direction. When the deviatoric strain reaches state C, some mesostructures near $\varphi = 90^\circ$ begin to fail as empty values of α_3. This will lead to a plastic regime on the macroscale. From states D to F, the failure area becomes larger and larger, resulting in the stress peak and even the softening regime on the macroscopic scale. The reader should noted that a slight increase is found in α_3. This is because the mesostructure is compressed in the principal strain direction. The nature of the change in α_3 is actually the rearrangement of particles.

5. Conclusions

In this paper, the programming method of the MM model using Julia language, in the context of computational granular materials, was presented. The following conclusions can be drawn from the results and discussions:

1. The entire code of the 3D-H model was clearly formulated and programmed. Each function and object is expected to be easily understood by illustrating with a flow chart.

2. A global mixed method was presented for solving linearised mixed control constraint equations. The effectiveness of this method has been validated by simulating drained triaxial compression tests, in which the boundary conditions consisted of imposing the vertical strain and the lateral stresses.

3. Both the 3D-H and DEM models can reproduce the experimental results on Labenne sand, involving only a few parameters. In contrast to the DEM model, where all contact information is transferred and stored, the MM model runs faster using a multiscale approach.

4. The Julia language should be one of the most appropriate languages for programming the MM model because a lot of loops need to be calculated. The easier understanding of how objects are accessed and passed through functions in Julia allows some implementation techniques to be used to make further improvements to the program.

The 3D-H model belongs to the family of MM models, which use the multiscale approach. This study could, therefore, provide guidance for similar multiscale approach programming.

Acknowledgment

The financial support for this research came from the National Natural Science Foundation of China (No. 51579179).

Appendix A. Contact law

This elastic-perfect plastic model includes a Mohr-Coulomb criterion and can be expressed under the following incremental formalism:

$$\begin{cases} \delta N_i = k_n \delta u_n^i \\ \delta T_i = \min\left\{\left\|T_i + k_t \delta u_t^i\right\|, \tan\varphi_g \left(N_i + k_n \delta u_n^i\right)\right\} \times \frac{T_i + k_t \delta u_t^i}{\left\|T_i + k_t \delta u_t^i\right\|} - T_i \end{cases} \tag{A.1}$$

where: $i = 1, 2, 3, 4$ denotes the identifier of contact number.

According to Equations 2, Equations A.1 can be rewritten as follows:

$$\begin{cases} \delta N_i = -k_n \delta d_i \\ \delta T_i = k_t d_i \delta \alpha_j & \text{elastic regime} \\ \delta T_i = \tan\varphi_g \left(N_i - k_n \delta d_i\right) \xi_i - T_i & \text{plastic regime} \end{cases} \tag{A.2}$$

where: ξ_i is the sign of $T_i + k_t d_i \delta\alpha_j$; $j = 1$ when $i = 1, 2$; $j = 2$ when $i = 3, 4$; plastic regime is reached when $\parallel k_t d_i \delta\alpha_j + T_i \parallel \geqslant \tan\varphi_g \left(N_i - k_n \delta d_i\right)$, otherwise it is in elastic regime.

To facilitate the derivation, I_i^p and I_i^e are introduced as indicator functions of the contact state, expressed as follow:

$$I_i^p = \begin{cases} 1 & \text{in plastic regime} \\ 0 & \text{in elastic regime} \end{cases} \quad ; \quad I_i^e = 1 - I_i^p \tag{A.3}$$

Thus, the constitutive relations can be expressed as:

$$\begin{cases} \delta N_i = -k_n \delta d_i \\ \delta T_i = B_i \delta\alpha_j - A_i \delta d_i + C_i \end{cases} \tag{A.4}$$

$$\text{where:} \begin{cases} A_i = I_i^p k_n \xi_i \tan\varphi_g \\ B_i = I_i^e k_t d_i \\ C_i = I_i^p (\xi_i \tan\varphi_g N_i - T_i) \end{cases}$$

References

References

[1] Bardet, J., Choucair, W., 1991. A linearized integration technique for incremental constitutive equations. International Journal for Numerical and Analytical Methods in Geomechanics 15, 1–19.

[2] Belheine, N., Plassiard, J.P., Donzé, F.V., Darve, F., Seridi, A., 2009. Numerical simulation of drained triaxial test using 3d discrete element modeling. Computers and Geotechnics 36, 320–331.

[3] Bezanson, J., Edelman, A., Karpinski, S., Shah, V.B., 2017. Julia: A fresh approach to numerical computing. SIAM review 59, 65–98.

[4] Bezanson, J., Karpinski, S., Shah, V.B., Edelman, A., 2012. Julia: A fast dynamic language for technical computing. arXiv preprint arXiv:1209.5145 .

[5] Bignonnet, F., Dormieux, L., Kondo, D., 2016. A micro-mechanical model for the plasticity of porous granular media and link with the cam clay model. International Journal of Plasticity 79, 259–274.

[6] Cambou, B., Dubujet, P., Emeriault, F., Sidoroff, F., 1995. Homogenization for granular materials. European journal of mechanics. A. Solids 14, 255–276.

[7] Christoffersen, J., Mehrabadi, M.M., Nemat-Nasser, S., 1981. A micromechanical description of granular material behavior. Journal of Applied Mechanics 48, 339–344.

[8] Cundall, P.A., Strack, O.D., 1979. A discrete numerical model for granular assemblies. geotechnique 29, 47–65.

[9] Darve, F., Servant, G., Laouafa, F., Khoa, H.D.V., 2004. Failure in geomaterials: continuous and discrete analyses. Computer methods in applied mechanics and engineering 193, 3057–3085.

[10] De Saxcé, G., Fortin, J., Millet, O., 2004. About the numerical simulation of the dynamics of granular media and the definition of the mean stress tensor. Mechanics of Materials 36, 1175–1184.

[11] Guo, N., Zhao, J., 2016. 3d multiscale modeling of strain localization in granular media. Computers and Geotechnics 80, 360–372.

[12] Hertz, H., 1882. Ueber die beruhrung fester elastischer korper. J reine angew Math 92, 156–171.

[13] Jardine, R., Overy, R., et al., 1996. Axial capacity of offshore piles driven in dense sand, in: Offshore Technology Conference, Offshore Technology Conference.

[14] Jin, Y.F., Yin, Z.Y., Shen, S.L., Hicher, P.Y., 2016. Selection of sand models and identification of parameters using an enhanced genetic algorithm. International Journal for Numerical and Analytical Methods in Geomechanics 40, 1219–1240.

[15] Jin, Y.F., Yin, Z.Y., Wu, Z.X., Zhou, W.H., 2018. Identifying parameters of easily crushable sand and application to offshore pile driving. Ocean Engineering 154, 416–429.

[16] Jin, Y.F., Yin, Z.Y., Zhou, W.H., Huang, H.W., 2019. Multi-objective optimization-based updating of predictions during excavation. Engineering Applications of Artificial Intelligence 78, 102–123.

[17] Kruyt, N.P., Millet, O., Nicot, F., 2014. Macroscopic strains in granular materials accounting for grain rotations. Granular matter 16, 933–944.

[18] Lehane, B., Jardine, R., Bond, A.J., Frank, R., 1993. Mechanisms of shaft friction in sand from instrumented pile tests. Journal of Geotechnical Engineering 119, 19–35.

[19] Love, A.E.H., 2013. A treatise on the mathematical theory of elasticity. volume 1. Cambridge University Press.

[20] MATLAB, 2010. version 7.10.0 (R2010a). The MathWorks Inc., Natick, Massachusetts.

[21] Mehrabadi, M.M., Nemat-Nasser, S., Oda, M., 1982. On statistical description of stress and fabric in granular materials. International Journal for Numerical and Analytical Methods in Geomechanics 6, 95–108.

[22] Mestat, P., 1992. Caractérisation du comportement du sable de labenne. détermination des paramètres des lois de nova et de vermeer à partir d'essais de laboratoire. Laboratoire Central des Ponts et Chaussées, Division MSGI, Rapport interne, Thème GEO 7, 110.

[23] Mestat, P., Riou, Y., 2001. Methodologie de determination des parametres pour la loi de comportement elastoplastique de vermeer et simulations d'essais de mecanique des sols. Bulletin des laboratoires des Ponts et Chaussées .

[24] Metcalf, M., Reid, J.K., Cohen, M., 2004. Fortran 95/2003 Explained. volume 416. Oxford University Press Oxford.

[25] Mindlin, R., 1949. Compliance of elastic bodies in contact. J. Appl. Mech., ASME 16, 259–268.

[26] Muir Wood, D., Maeda, K., 2008. Changing grading of soil: effect on critical states. Acta Geotechnica 3, 3.

[27] Nicot, F., Darve, F., 2011. The H-microdirectional model: accounting for a mesoscopic scale. Mechanics of Materials 43, 918–929.

[28] Nicot, F., Darve, F., Group, R., 2005. A multi-scale approach to granular materials. Mechanics of materials 37, 980–1006.

[29] Njock, P.G.A., Shen, S.L., Zhou, A., Lyu, H.M., 2020. Evaluation of soil liquefaction using ai technology incorporating a coupled enn/t-sne model. Soil Dynamics and Earthquake Engineering 130, 105988.

[30] Pouragha, M., Wan, R., 2018. μ-gm: A purely micromechanical constitutive model for granular materials. Mechanics of Materials 126, 57–74.

[31] Rossum, G., 1995. Python reference manual .

[32] Smith, G.D., 1985. Numerical solution of partial differential equations: finite difference methods. Oxford university press.

[33] Stroustrup, B., 2000. The C++ programming language. Pearson Education India.

[34] Wang, X.W., Yang, T.L., Xu, Y.S., Shen, S.L., 2019. Evaluation of optimized depth of waterproof curtain to mitigate negative impacts during dewatering. Journal of Hydrology 577, 123969.

[35] Wautier, A., Bonelli, S., Nicot, F., 2019. Dem investigations of internal erosion: Grain transport in the light of micromechanics. International Journal for Numerical and Analytical Methods in Geomechanics 43, 339–352.

[36] Wu, Y.X., Shen, S.L., Lyu, H.M., Zhou, A., 2020. Analyses of leakage effect of waterproof curtain during excavation dewatering. Journal of Hydrology 583, 124582.

[37] Xiong, H., Nicot, F., Yin, Z., 2017. A three-dimensional micromechanically based model. International Journal for Numerical and Analytical Methods in Geomechanics 41, 1669–1686.

[38] Xiong, H., Nicot, F., Yin, Z., 2018. From micro scale to boundary value problem: using a micromechanically based model. Acta Geotechnica , 1–17.

[39] Xiong, H., Yin, Z.Y., Nicot, F., 2019. A multiscale work-analysis approach for geotechnical structures. International Journal for Numerical and Analytical Methods in Geomechanics .

[40] Yin, Z.Y., Jin, Y.F., Shen, J.S., Hicher, P.Y., 2018. Optimization techniques for identifying soil parameters in geotechnical engineering: Comparative study and enhancement. International Journal for Numerical and Analytical Methods in Geomechanics 42, 70–94.

[41] Yin, Z.Y., Zhao, J., Hicher, P.Y., 2014. A micromechanics-based model for sand-silt mixtures. International journal of solids and structures 51, 1350–1363.

[42] Zhang, Y., Buscarnera, G., 2016. Implicit integration under mixed controls of a breakage model for unsaturated crushable soils. International Journal for Numerical and Analytical Methods in Geomechanics 40, 887–918.

[43] Zhao, C.F., Yin, Z.Y., Hicher, P.Y., 2018. Integrating a micromechanical model for multiscale analyses. International Journal for Numerical Methods in Engineering 114, 105–127.

[44] Zienkiewicz, O.C., Taylor, R.L., Nithiarasu, P., Zhu, J., 1977. The finite element method. volume 3. McGraw-hill London.