Finite and Discrete Element Modelling of Internal Erosion in Water Retention Structures

by

Seyed Pouyan PIRNIA

MANUSCRIPT-BASED THESIS PRESENTED TO ÉCOLE DE TECHNOLOGIE SUPÉRIEURE IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY
Ph.D.

MONTREAL, JULY 19, 2019

ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

BOARD OF EXAMINERS

THIS THESIS HAS BEEN EVALUATED

BY THE FOLLOWING BOARD OF EXAMINERS

Mr. François Duhaime, Thesis Supervisor
Department of Construction Engineering at École de technologie supérieure

Mr. Yannic A. Ethier, Thesis Co-supervisor
Department of Construction Engineering at École de technologie supérieure

Mr. Philippe Bocher, President of the Board of Examiners
Department of Mechanical Engineering at École de technologie supérieure

Mr. Henri Champliaud, Member of the jury
Department of Mechanical Engineering at École de technologie supérieure

Mr. Mohamed Meguid, External Evaluator
Department of Civil Engineering at McGill University

THIS THESIS WAS PRESENTED AND DEFENDED

IN THE PRESENCE OF A BOARD OF EXAMINERS AND PUBLIC

JULY 3, 2019

AT ECOLE DE TECHNOLOGIE SUPERIEURE

ACKNOWLEDGMENT

My special acknowledgments are for my advisor, Prof. François Duhaime, for his guidance, strong passion, encouragement and flexibility that made this dissertation possible. This dissertation would not have been possible without his scientific contribution. I have had the chance to be accepted in his research group and worked with such a wonderful man. I must thank my co-advisor Prof. Yannic Ethier for his invaluable support and kindly advice during my doctoral works. I am also deeply thankful to Prof. Jean-Sébastien Dubé for his contribution as a co-author in the papers.

I would like to thank the PhD committee members: Dr. Mohamed Meguid, Dr. Henri Champliaud and Dr. Philippe Bocher for their valuable time in reviewing the dissertation. Many thanks go to the experts in Hydro-Québec, especially Annick Bigras and Hugo Longtin. The thesis could not be completed without financial support from Hydro-Québec and NSERC.

I would like to acknowledge all my friends and teachers who supported me during my journey from Iran to Canada, especially Bahram Aghajanzadeh. Thanks to my LG2 colleagues who have enriched my daily life throughout my studies.

I would highly indebt to my dear parents for their endless love, dedication and support in my life. They always got my back and helped me to fulfill my dreams. My special thanks are extended to my brother, Amir, for his support and guidance. Finally, I have to thank my beloved wife, Sadaf, for her love, understanding and encouragement which helped in completion of this thesis.

Modélisation par éléments finis et éléments discrets de l'érosion interne dans les ouvrages de rétention d'eau

Seyed Pouyan PIRNIA

RÉSUMÉ

L'érosion interne implique le transport de particules à l'intérieur d'un sol en raison d'un écoulement en milieu poreux. Ce phénomène est considéré comme une menace sérieuse pour les structures en matériaux granulaires. L'érosion interne est la principale cause de dommage ou de rupture du corps ou de la fondation des barrages en remblai. Par conséquent, il est nécessaire d'avoir une connaissance précise des interactions fluide-particule dans les sols saturés lors de la conception et de l'exploitation des barrages. Le comportement hydrodynamique des milieux poreux en géotechnique est généralement modélisé à l'aide de méthodes qui considèrent le sol comme un milieu continu, telles que la méthode des éléments finis (MEF).

Il est de plus en plus courant de combiner la méthode des éléments discrets (MED) avec des méthodes pour les milieux continus, comme la MEF, afin de fournir des informations microscopiques sur les interactions fluide-solide. Cette thèse a pour but de développer un algorithme MEF-MED hiérarchique permettant d'analyser le processus d'érosion interne dans des milieux poreux pour des applications à grande échelle. Pour atteindre cet objectif, nous avons (i) programmé une interface polyvalente entre deux codes MEF et MED, (ii) développé une méthode macroscopique de calcul des forces de trainée sur les particules (CGM) pour le modèle couplé MEF-MED afin de minimiser le temps de calcul, (iii) développé un algorithme multi-échelle pour l'interface afin de limiter le nombre de particules discrètes impliquées dans la simulation, (iv) évalué la précision de la force de traînée dérivée de CGM, et (v) former un réseau de neurones artificiels (ANN) afin d'améliorer la prédiction de la force de traînée sur les particules.

Le développement de modèles multiméthodes ou hybrides combinant des analyses de type continuum et des éléments discrets est une piste de recherche prometteuse pour combiner les avantages associés aux deux échelles de modélisation. Cette thèse présente tout d'abord ICY, une interface entre COMSOL Multiphysics (code commercial d'éléments finis) et YADE (code d'éléments discrets ouvert). À travers une série de classes JAVA, l'interface associe la modélisation par éléments discrets à l'échelle des particules à la modélisation à grande échelle avec la méthode des éléments finis. ICY a été validé avec un exemple simple basé sur la loi de Stokes. Une comparaison des résultats pour le modèle couplé et la solution analytique montre que l'interface et son algorithme fonctionnent correctement. Le chapitre présente également un exemple d'application pour l'interface. L'interface a utilisé la force de traînée CGM pour modéliser un test d'érosion interne dans un perméamètre.

Le nombre de particules qui peuvent être incluses dans les simulations DEM avec ICY est limité. Cette limitation réduit le volume de sol pouvant être modélisé. La deuxième partie de la thèse propose une approche multiméthode hiérarchique basée sur ICY pour modéliser le comportement couplé hydromécanique de sols granulaires saturés. Un algorithme multiméthode a été développé pour limiter le nombre de particules dans la simulation DEM et permettre à terme la modélisation de l'érosion interne de grandes structures. Le nombre de particules dans les simulations a été limité en utilisant des sous-domaines discontinus le long de l'échantillon. Cette approche évite de générer le domaine complet comme modèle DEM. Les particules dans ces petits sous-domaines ont été soumises à la flottabilité, à la gravité, à la force de traînée et aux forces de contact pendant de courts pas de temps. Les petits sous-domaines fournissent au modèle de continuum des données initiales (par exemple, un flux de particules). Le modèle FEM résout une équation de conservation des particules pour évaluer les changements de porosité sur des intervalles de temps plus longs. L'algorithme multi-échelle a été vérifié en simulant un test numérique d'érosion interne.

Le mouvement des fluides dans les applications géotechniques est généralement résolu avec une forme homogénéisée des équations de Navier-Stokes. La force totale de traînée obtenue de la CGM peut être appliquée aux particules proportionnellement à leur volume (CGM-V) ou à leur surface (CGM-S). Cependant, il existe une certaine incertitude quant à l'application des modèles de traînée CGM aux mélanges polydisperses de particules. La précision de CGM pour la modélisation de n'a pas été systématiquement étudiée en comparant les résultats CGM avec les résultats plus précis obtenus en résolvant les équations de Navier-Stokes à l'échelle des pores. La dernière partie de cette thèse compare les forces de traînée CGM-V et CGM-S avec celles qui sont obtenues avec la résolution des équations de Navier-Stokes à l'échelle des pores avec la MEF. COMSOL Multiphysics a été utilisé pour simuler l'écoulement dans trois cellules unitaires avec différentes valeurs de porosité (0,477, 0,319 et 0,259). Chaque cellule unitaire comportait un squelette monodisperse de grandes particules avec des positions fixes, et une particule plus petite, de taille et de position variables. Les résultats ont montré que les forces de trainée CGM-V et CGM-S sont généralement assez éloignées des forces obtenues à petite échelle avec la MEF. La précision diminue davantage quand le contraste entre les tailles des grandes particules et de la petite particule augmente. Un ANN a été formé pour prédire la force de trainée MEF en utilisant comme données d'entrée les forces de trainée CGM-V et CGM-S, le rapport entre les tailles de particules, et la distance entre la petite particule et les deux grandes particules les plus près. Une très bonne corrélation a été trouvée entre la sortie de l'ANN et les résultats MEF. Ce résultat montre qu'un ANN peut fournir des forces de trainée aussi précises que celles de la MEF, mais avec un temps de calcul comparable à celui des méthodes CGM.

Cette thèse contribue à la littérature en améliorant notre compréhension des méthodes hybrides MED-continuum et des calculs de force de traînée dans les simulations MED. La thèse présente des recommandations aux chercheurs et aux développeurs qui tentent de modéliser l'érosion interne dans des systèmes de sols à l'échelle réelle.

Mots-clés: érosion interne, élément discret, élément fini, COMSOL, YADE, force de traînée, calcul macroscopique des forces de trainée, réseau de neurones artificiels

Finite and Discrete Element Modelling of Internal Erosion in Water Retention Structures

Seyed Pouyan PIRNIA

ABSTRACT

Internal erosion is a process by which particles from a soil mass are transported due to an internal fluid flow. This phenomenon is considered as a serious threat to earthen structures. Internal erosion is the main cause of damage or failure in the body or foundation of embankment dams. Therefore, it is necessary to have an accurate knowledge of fluid-particle interactions in saturated soils during design and operation. The hydrodynamic behaviour of porous media in geotechnical engineering is typically modelled using continuum methods such as the finite element method (FEM).

It has become increasingly common to combine the discrete element method (DEM) with continuum methods such as the FEM to provide microscopic insights into the behaviour of granular materials and fluid–solid interactions. This Ph.D. thesis aims to develop a hierarchical FEM-DEM algorithm to analyze the internal erosion process in large scale earthen structures. To achieve this goal, we (i) programmed a versatile interface between two FEM and DEM codes, (ii) implemented a coarse-grid method (CGM) for the coupled FEM-DEM model to minimize the computations associated with drag force calculation, (iii) developed a multiscale algorithm for the interface to limit the number of discrete particles involved in the simulation, (iv) assessed the accuracy of drag force derived from CGM, and (v) trained an Artificial Neural Network (ANN) to improve the prediction of the drag force on particles.

The development of multimethod or hybrid models combining continuum analyses and discrete elements is a promising research avenue to combine the advantages associated with both modelling scales. This thesis first introduces ICY, an interface between COMSOL Multiphysics (commercial finite-element engine) and YADE (open-source discrete-element code). Through a series of JAVA classes, the interface combines DEM modelling at the particle scale with large scale modelling with the finite element method. ICY was verified with a simple example based on Stokes' law. A comparison of results for the coupled model and the analytical solution shows that the interface and its algorithm work properly. The thesis also presents an application example for the interface. The interface used CGM drag force to model an internal erosion test in a permeameter.

The number of particles that can be included in the DEM simulation of ICY is limited, thus restricting the volume of soil that can be modelled. The second part of the thesis proposes a multimethod hierarchical approach based on ICY to model the coupled hydro-mechanical behaviour for saturated granular soils. A hierarchical algorithm was specifically developed to limit the number of particles in the DEM simulations and to eventually allow the modelling of internal erosion for large structures. The number of discrete bodies in the simulations was

restricted through employing discontinuous subdomains along the sample. This avoids generating the full sample as a DEM model. Particles in these small subdomains were subjected to buoyancy, gravity, drag force and contact forces for small time steps. The small subdomains provide the continuum model with particle flux. The FEM model solves a particle conservation equation to evaluate porosity changes for longer time steps. The multimethod framework was verified by simulating a numerical internal erosion test.

The fluid motion in geotechnical applications is typically solved using CGM. With these methods, an average form of the Navier–Stokes equations is solved. The total drag force derived from CGM can be applied to the particles proportionally to their volume (CGM-V) or surface (CGM-S). However, there is some uncertainty regarding the application of the CGM drag models for polydispersed particle. The accuracy of CGM has not been systematically investigated through comparing CGM results with more precise results obtained from solving the Navier-Stokes equations at the pore scale. The last part of this research investigates the accuracy of CGM-V and CGM-S drag forces in comparison with the pore-scale values obtained by FEM. COMSOL Multiphysics was used to simulate the fluid flow in three unit cells with different porosity values (0.477, 0.319 and 0.259). The unit cell involved a mono-size skeleton of large particles with fixed positions and a smaller particle with variable sizes and positions. The results showed that the CGM-V and CGM-S could not predict precisely the drag force on the small particle. An ANN was trained to predict the drag force on the smaller particle. A very good correlation was found between the ANN output and the FEM results. The ANN could thus provide drag force values with accuracy similar to that obtained using flow simulations at the pore scale, but with computational resources that are comparable to CGM.

This thesis contributes to the literature by improving our understanding of hybrid DEM-continuum methods and drag force computations in DEM simulations. It provides guidelines to researchers and developers who try to model internal erosion in real scale soil systems.

Keywords: Internal erosion, discrete element, finite element, COMSOL, YADE, drag force, coarse-grid method, artificial neural network

TABLE OF CONTENTS

Page

LIST OF FIGURES

LIST OF ABBREVIATIONS

ANN	Artificial Neural Network
AoR	Angle of Repose
BCC	Body-Centered Cubic structure
BVP	Boundary Value Problem
CFD	Computational Fluid Dynamics
CGM	Coarse-Grid Method
DEM	Discrete Element Method
FCC	Face-Centered Cubic structure
FEM	Finite Element Method
GUI	Graphical User Interface
ICY	Interface COMSOL-YADE
iCP	Interface COMSOL-PHREEQC
LBM	Lattice-Boltzmann Method
LM	Levenberg-Marquardt algorithm
MPI	Message-Passing Interface
MSE	Mean Squared Error
ODE	Ordinary Differential Equation
OpenMP	Open Multi-Processing
PDE	Partial Differential Equations
PFV	Pore-scale Finite Volume
REV	Representative Elementary Volume

RMSE	Root Mean Squared Error
R^2	Coefficient of determination
SC	Simple Cubic packing structure
SPH	Smoothed Particle Hydrodynamics

LIST OF SYMBOLS AND UNITS OF MEASUREMENT

A	bulk cross-sectional area of flow
A_c	empirical factor for the Kozney-Carman equation
A_i	surface of solid particles
A_{Pi}	projected area of particle i
b	neuron's bias
c_n	contact damping coefficient
μ_c	Coulomb friction coefficient
D_R	particle specific weight
d	particle or sphere diameter
d_i	diameter of particle I
D_x	grain size corresponding to x% finer (used for coarser fraction or filter layer)
d_x	grain size corresponding to x% finer (used for finer fraction or base layer)
D_T	tangential matrix
D_R	particle specific weight
e	void ratio
E	particle elastic modulus or Young's modulus
F_i	resulting force on particle i
F_i^{ext}	force resulting from external loads on particle i
F_i^c	contact force on particle i at the interacting point
F_n	normal contact force
F_T	tangential contact force

F_B	buoyancy force
F_w	weight force
F_D	drag force
F_{DPi}	drag force on particle i
F_i^{damp}	force resulting from damping in the system
F_n^d	normal damping term
F_T^d	tangential damping term
F_{DPi}	drag force on particle i
F_{sample}	total force on the packing
f	volume flux of small particles per unit surface and time in each subdomain
g	acceleration of gravity
G	elastic shear modulus
h	hydraulic head
Δh	hydraulic head change (m) over length L
i	hydraulic gradient
K	hydraulic conductivity
k_n	normal stiffness
k_t	tangential stiffness
k_i	equivalent stiffness that is evaluated considering all the contacts of one particle
L	flow path length
l_t	straight-line distance between the beginning and ending point of a tortuous flow path

l	length of a tortuous flow path
M_i	mass of solid particles
M_i	moment on each individual particle
m_i	element mass
M_i^{ext}	torque resulting from external loads
M_i^{damp}	torque resulting from damping in the system
n	porosity
n_e	effective porosity
n_s	number of samples
n_p	number of particles in the representative elementary volume or a subdomain
n_c	number of particles in contact
$\vec{n}$	unit vector directed along the line between spheres (i and j) centres at the contact point
P	pore pressure
∇P	pore pressure gradient
P	vector containing the predicted outputs
$\dot{P}$	mean of the predicted values
q_i^c	resultant torques due to rolling
R	large particle radius
r	fine particle radius
Re	Reynolds number
r_i^c	vector connecting the centre of the i-th sphere with the contact point c

S_s	specific surface
S_{large}	sum of large particle surface
S_{fine}	sum of fine particle surface
t	time
Δt	time step
Δt_{crit}	critical time step
t_u	unit vector
T_l	tortuosity
T	vector containing the target outputs
u	fluid velocity
u_n	penetration distance
V_{large}	sum of large particle volume
V_{fine}	sum of fine particle volume
v_z	mean velocity of the small particles
V_{Pi}	volume of particle i
V	particle velocity
v	Darcy, discharge or apparent velocity
v_s	average or seepage velocity
V_v	void volume
V_T	total volume
w	rotational velocity
w_{max}	maximum of the natural period of the mass-spring system

W_i	synaptic weights
$\dot{w}$	rotational accelerations of spherical particles
x_i	inputs for an artificial neural network
$\dot{x}$	translational velocity
$\ddot{x}$	translational accelerations of spherical particles
Z	elevation head
z_i	vertical displacement of small particle i
α	damping constant
γ_w	unit weight of the fluid
γ_s	unit weight of particles
ε	strain
μ	fluid dynamic viscosity
ρ	fluid density
σ	total stress
σ'	effective stress
τ	viscous stress
υ	Poisson's ratio
ϕ	proportion of the small particle volume with respect to the subdomain volume
Ω	volume percentage of large particles

CHAPTER 1

INTRODUCTION

1.1 Background

Embankment dams are structures that impound and control water in upstream reservoirs. They can be erected on almost all foundations or sites which are not proper for building concrete structures, or where suitable soils are available locally (Fell et al., 2005). Soils are transported to the site, dumped, and compacted in layers of required thickness. There are two main kinds of embankment dams: earthfill and rockfill dams.

The chief advantages of earth dams are their adaptability with weak foundations and economic benefits inasmuch as required construction materials are principally supplied near the dam site. Earth dams can be of two main types: homogeneous and zoned dams (Figure 1.1). Since filters are included in zoned embankment dams to control seepage, this type is often preferred.

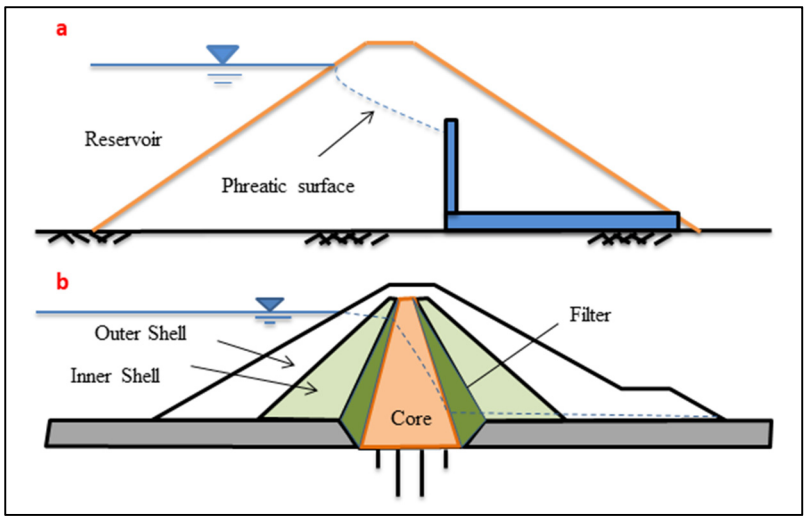


Figure 1.1 Homogeneous earth dam with chimney drain (a) and Zoned earth dam with central vertical core (b)

There are several types of dam failure including collapse and breaching. The resulting damage depends largely on the volume of water stored in the reservoir. For large dams and reservoirs, failure can often cause significant damage and loss of life (Graham & Wayne, 1999). Dam failure can be caused by any one, or a combination of the following factors (Zhang et al., 2009):

- Flood and long period of rainfall;
- Overtopping as a result of spillway design error;
- Internal erosion or piping, specifically in earth dams;
- Improper maintenance for gates, valves, and other mechanical components;
- Sub-standard construction materials or improper design;
- Surges caused by landslides in the reservoir and resulting dam overtopping;
- High winds and associated waves resulting in erosion of the upstream slope;
- Earthquakes.

It is often impossible to determine the exact cause of dam failure inasmuch as failure tends to destroy the evidence that would allow the cause to be identified (Hellström, 2009). Nevertheless, statistics show that overtopping and internal erosion are the two main causes of embankment dam failure while failure by slide is less common (Foster et al., 2000). Most piping failures happen very fast. As a consequence, there is not enough time to take proper actions (Hellström, 2009).

Internal erosion corresponds to the transportation of soil in embankment dams by seepage flow (ICOLD, 2016). It changes the hydraulic and mechanical characteristics of materials in porous media. The property that is most influenced is hydraulic conductivity.

The existence of internal erosion has been known for over 80 years. According to the statistical analysis done by Foster et al. (2000) probing 11 192 dams in the world from 1986 to 2000, 136 dams encountered failure mainly because of internal erosion. These 136 dams represent 46% of the total number of dams that encountered failure. Failure from internal erosion will occur if four conditions are satisfied (Fell et al., 2005):

1) Existence of a seepage flow path and a source of water;
2) Existence of erodible material in the flow path;
3) Existence of an unprotected exit which allows the discharge of eroded materials;
4) Existence of an appropriate material directly above the flow path to support the roof of the pipe.

Nearly all internal erosion failures have occurred when the water level in the reservoir was near its highest level ever (Foster, 2000). Although most internal erosion failures happen on the reservoir first filling as a result of weaknesses in the dam, it is also a threat to existing dams due to (ICOLD, 2016):

- Settlement and cracking because of extreme water levels and earthquakes;
- Deterioration of spillways and hydraulic structures because of aging;
- Ineffective filters or transition zones.

There are four main processes for erosion in an earth dam: backward erosion, concentrated leak, suffusion and contact erosion (Figure 1.2) (Hellström, 2009). Backward erosion begins at the exit point and progresses backward to form a pipe. For concentrated leak, the water source forms a crack or a soft region to an exit point. The erosion hole gets progressively wider since erosion continues along the walls. During suffusion, fine particles of soil are eroded and move between the coarser particles. Suffusion happens in soils that are described as internally unstable. Erosion at the interface between two soils is termed interfacial or contact erosion. Piping is defined in the same sense as internal erosion processes that create and extend an open conduit for flow through the soil.

Piping can occur in different parts of the dam: through the embankment, through the foundation and from the embankment into the foundation (Hellström, 2009; Foster et al., 2000). According to Foster et al. (2000), piping through the embankment dam's body is the most common mode of failure as it is 2 times more probable than piping through the foundation and 20 times more probable than piping from the embankment into the foundation.

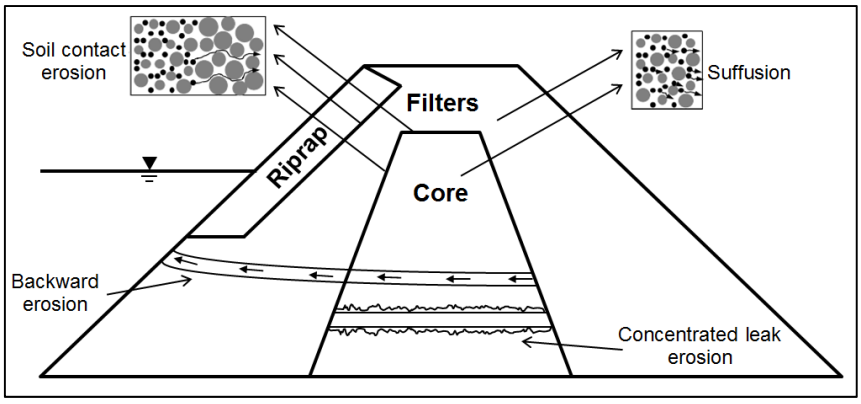


Figure 1.2 Backward erosion, concentrated leak, suffusion and soil contact erosion
Taken from Chang (2012)

Dam overtopping caused 30% of dam failures in the U.S. over the last 75 years (FEMA, 2013). Overtopping causes a breach by erosion of the dam material. Overtopping is caused by inadequate spillway capacity and improper operation of spillway gates. Core overtopping is a different phenomenon during which the water level in the reservoir is situated above the crest of the dam's core and below the dam's crest (Figure 1.3). Core overtopping causes a parallel water flow at the interface of the core (fine soil layer) and its surrounding filter (coarse soil layer) that can initiate contact or interfacial erosion (Dumberry et al., 2017). The shear stress resulting from the hydraulic head gradient at the interface between filter and core materials can cause the erosion of finer materials. Contact erosion may trigger serious damages in embankment dams.

Recently, because of improvement in the analysis of extreme flood events, and better precipitation and watershed information, it has been inferred that several thousand dams in the United States alone do not have sufficient spillway capacity to accommodate the appropriate design floods (FEMA, 2013). As a consequence, there has been a drive to understand the failure mechanisms associated with core and dam overtopping (FEMA, 2007), and to assess the performance of different protective measures in the eventuality of dam overtopping (FEMA, 2014).

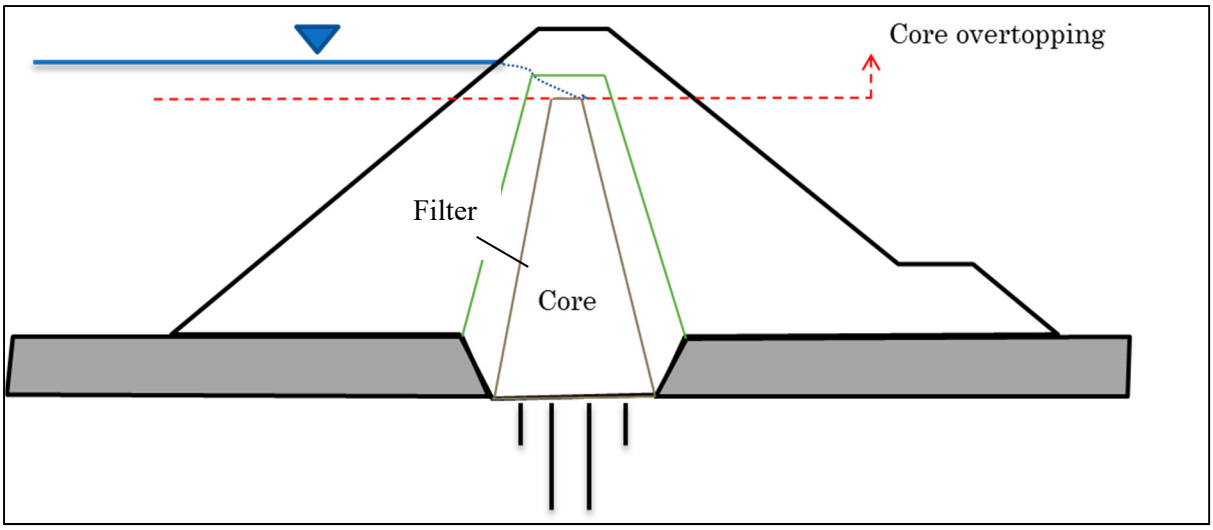


Figure 1.3 Core overtopping and water flow at the interface of core and filter

Canada is among the 10 most important dam builders in the world (CDA, 2015). More than 10 000 dams can be found in Canada. Of these dams, 933 are classified as "large" dams with a reservoir of more than 3 million m^3. The province of Quebec in Canada holds a third of the large dams. There are 6000 dams and dikes in Quebec. Of these, 10% are managed by Hydro-Québec. As can be seen from Figure 1.4, 72% of Hydro-Québec dams are embankment dams.

Internal erosion in embankment dams is a very complex phenomenon that is not well understood. It cannot be detected until it has progressed enough to be visible. As will be shown with the literature review, one of the most promising method to analyze internal erosion is numerical modelling inasmuch as it allows several factors and parameters to be considered in the process. As a result, it can lead to a better insight into the details of the internal erosion happening inside the dam. It can provide an early notion of potential erosion progress inside the dams.

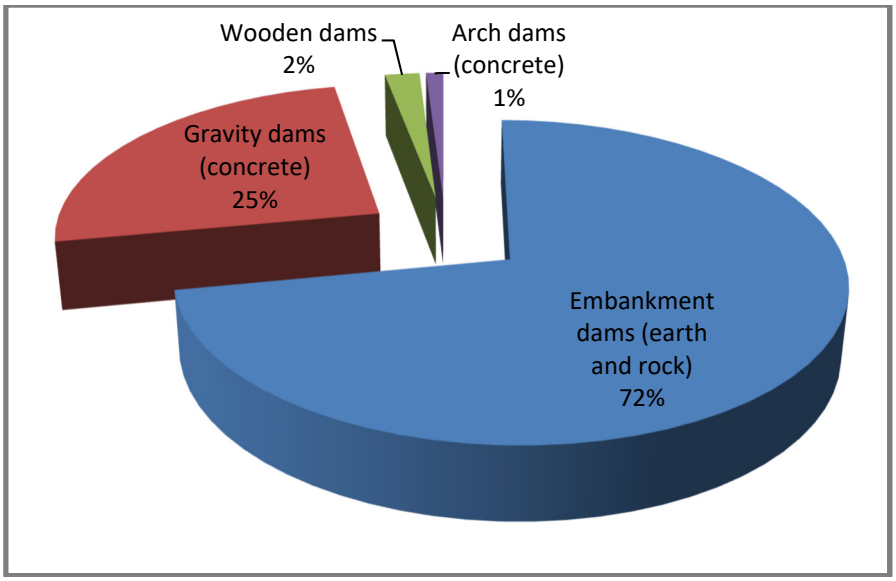


Figure 1.4 Percentage of dams operated by Hydro-Québec according to their type
Taken from Hydro-Québec (2002)

Granular materials, like the body of an earth dam, have conventionally been analyzed within a continuum framework in which the discrete nature of the soil is not taken into account. Continuum models have had particular success in capturing some important aspects of porous media behaviour such as seepage and stress-strain behaviour. Nevertheless, some processes, like internal erosion, derive from complex microstructural mechanisms at the particle level, and are currently difficult to model with continuum models. To understand the macroscopic behaviour, the modelling should be done at the microscopic scale (Guo & Zhao, 2014).

The Discrete Element Method (DEM) is becoming increasingly common in geotechnical engineering (O'Sullivan, 2015). DEM is a numerical method for computing the motion and interaction of a large number of small particles. This approach considers explicitly each particle in a granular material, hence it can simulate finite displacements and rotations of particles(Cundall & Hart, 1993). It has had outstanding success in reproducing the mechanical response of dry granular material at both the particle and continuum scales (e.g., O'Sullivan et al., 2008). A multipurpose interface is needed to allow data to be exchanged between

continuum models based on FEM and particle scale models based on DEM. Current hybrid or multimethod models for soils often are not extensible on both FEM and DEM sides. There is also a need for a hydrodynamic method to calculate drag force in DEM simulations involving a large number of particles. The most precise methods that solve fluid motion at the pore scale are not applicable due to the heavy computational cost. On the other hand, there is not a conclusive study considering the accuracy of methods that solve an averaged form of the Navier–Stokes equations at the continuum scale. For most soil mechanic applications, it is not feasible to model large scale structures, like an earth dam, solely with DEM. As a consequence, to be included in the modelling of large-scale applications, DEM must be coupled with continuum models in a multiscale analysis where small scale DEM simulations are conducted for selected nodes in the model. This type of multiscale hybrid model remains in development and has not seen widespread use in geotechnical practice.

1.2 Objectives

In this thesis, we tried to take some important steps required to achieve a multiscale FEM-DEM model to be capable of simulating the internal erosion process in large structures. The main and specific objectives can be summarized as follows:

1. Development of an interface between COMSOL Multiphysics and discrete element code YADE for the modelling of porous media (paper #1).

The sub-objectives of paper #1 were:

i. Develop an Interface between COMSOL and YADE to exchange data.
ii. Program a YADE interface to apply hydrodynamic forces on particles based on a head loss determined at the macroscopic scale.
iii. Verify the interface using experimental results for contact erosion available in the literature.

2. Development of a multiscale computational algorithm aimed at stimulating fluid-particle interaction for large-scale applications in soil mechanics (paper #2).

The sub-objectives of paper #2 were:

i. Develop a mass flux conservation equation for the COMSOL model.
ii. Implement our multiscale computational algorithm for the ICY and YADE script.
iii. Verify the multiscale model performance for a numerical suffusion test.

3. Assessment of the coarse grid method in computation of drag force and proposing an improved method (paper #3).

The sub-objectives of paper #3 were:

iv. Calculate the drag force at the pore scale on particles inside the unit cells involved a skeleton of large particles and a smaller particle.
v. Generate a data set by changing the smaller particle size and position in the unit cells.
vi. Compare the drag forces derived from Darcy's law and CGM, with the drag forces derived from the Navier-Stokes equations.
vii. Train an artificial neural network using the data set and assess its performance.

1.3 Synopsis and content

Chapter 2 of this dissertation presents a literature review on the following subjects:

- Physics of porous media
- Experimental studies of internal erosion
- Numerical modelling of fluid-particle interaction
- Numerical modelling of internal erosion in embankment dams
- Discrete element methods

The main part of this dissertation includes three manuscripts in Chapters 3, 4 and 5. Two of them were published and one other is submitted. Three conference papers (Pirnia et al., 2016, 2017 and 2018) were also presented during this project.

- Chapter 3: "ICY: An interface between COMSOL Multiphysics and discrete element code YADE for the modelling of porous media" Published in Computers and Geosciences, 2019.

This Chapter presents an interface that allows virtually any PDE to be combined with the DEM. Through a JAVA interface called ICY, a DEM code modelling at the particle scale (open-source code YADE) was combined with large scale modelling with the finite element method (commercial software package COMSOL). The particle–fluid interaction is considered by exchanging such interaction forces as drag force and buoyancy force between the DEM and the FEM model. The interface was developed for a practical application including a relatively small assembly of particles. Small number of particles included in the coupled model simulation was a restricting factor in modelling of large scale soil systems. The development of a multiscale framework combining continuum analyses and discrete elements is a promising research avenue to address this limitation. The presented interface allows multiscale modelling for large scale granular structures. The interface was introduced in Pirnia et al. (2016) and orally presented in the 69th Canadian Geotechnical Conference, Vancouver, Canada.

- Chapter 4: "Hierarchical multiscale numerical modelling of internal erosion with discrete and finite elements" submitted in Acta Geotechnica in March, 2019.

This Chapter presents a multiscale algorithm based on ICY that limits the number of particles in the DEM simulation. It eventually allows the modelling of internal erosion for large structures. With the multiscale algorithm, smaller DEM subdomains are generated to simulate particle displacements and flux at the microscale. The particle flux distribution are set in a 1-D COMSOL model that uses a particle conservation equation to calculate new porosity and drag force values after longer time steps. The multiscale algorithm avoids generating the full

sample as a DEM model. The multiscale model was introduced in Pirnia et al. (2017) and orally presented in the 70th Canadian Geotechnical Conference, Ottawa, Canada.

- Chapter 5: “Drag force calculations in polydisperse DEM simulations with the coarse-grid method: influence of the weighting method and improved predictions through artificial neural networks ” Published in Transport in porous media, 2019.

This Chapter performed a detailed analysis on the accuracy of the coarse-grid method (CGM) which is often used to compute drag force on the particles in geotechnical engineering. The CGM solves an averaged form of the Navier–Stokes equations at the continuum scale. The CGM drag force values were compared with finite element values by solving the Navier-Stokes equations on a small particle variable in size and position inside three different porosity unit cells. It was found that the CGM methods generally did not produce precise drag forces. Hence, applicability of an artificial neural network (ANN) trained using the FEM drag force values was assessed to predict the drag force on the smaller particle.

The final published paper in Chapter 4 may differ from the version presented in the dissertation based on probable reviewers’ requests.

Chapter 6 presents a discussion of the results and recommendations for future works.

CHAPTER 2

LITERATURE REVIEW

The chapter first describes the physics and characteristics of porous media. It, then, briefly presents experimental findings about the internal erosion and core overtopping in embankment dams. The next section concerns the numerical modelling of saturated porous media. The methods to couple fluid flow with DEM are presented. The next section reviews available numerical methods for modelling internal erosion and highlights the fundamental issues that hamper to use the methods for structures such as embankment dams. Two next sections deal respectively with Multiphysics models and the discrete element method. Fundamental concepts for DEM are explained in detail. The last section introduces YADE, a DEM package that will be used in the project.

2.1 Physics of porous media

Fluid-particle interaction in geomechanics needs to be studied in terms of physical and mechanical properties of solid skeleton of porous media and the fluid. The interaction between soil particles and pore water is encountered in many problems in geotechnical engineering such as liquefaction (Chen, 2009). Porous media can be defined as solid bodies that contain void spaces inside (Figure 2.1).

Porous media are typically classified as unconsolidated (dispersed) or consolidated. Gravel and sand are examples of unconsolidated porous media. A porous media is characterized by a variety of geometrical properties (Scheidegger, 1958). Porosity is one of the most important parameters for the characterization of porous media. It is defined as the ratio of void volume (V_v) to total volume (V_T):

$$n = \frac{V_v}{V_T} \tag{2.1}$$

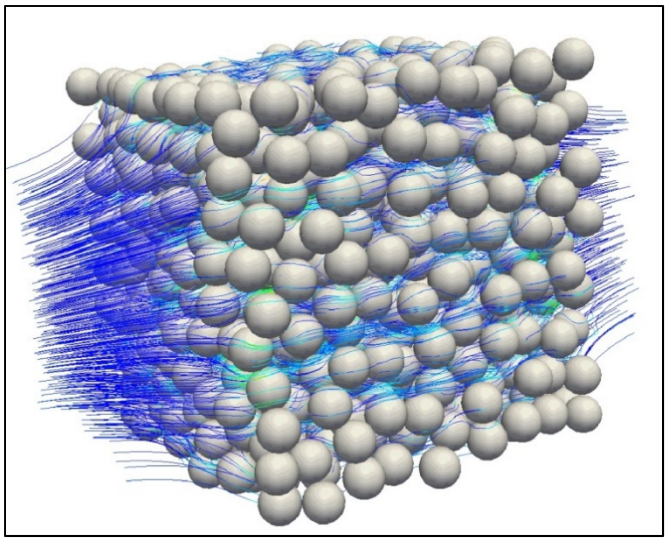


Figure 2.1 Example of a 3D porous medium
Taken from Perovic et al.(2016)

Porosity can be interconnected or non-interconnected (Scheidegger, 1958). The interconnected porosity is also known as the effective porosity (n_e). Fluid flow in porous media occurs in the effective porosity. Non-connected pores may be considered as part of the solid matrix (Bear, 2012).

The specific surface (S_s) is another basic characteristic of a porous medium. This feature determines the fluid flow behaviour through the solid matrix. It is defined as the ratio of the solid phase surface (A_i) to the solid phase mass (M_i):

$$S_s = \frac{A_i}{M_i} \left(\frac{m^2}{g}\right) \tag{2.2}$$

Fine grained porous media have a greater S_s than coarse-grained media. The behaviour of coarse-grained media is governed by the body forces while the behaviour of fine-grained media is controlled by surface forces.

Tortuosity is a dimensionless geometrical property that describes diffusion and fluid flow in porous media. It states the influence of the flow path followed by fluid in a porous media. Tortuosity can be defined as:

$$T_l = \frac{l_t}{l} \tag{2.3}$$

where l is the straight-line distance between the beginning and ending point of a tortuous flow path with length l_t.

2.2 Internal erosion

Internal erosion refers to the migration of particles from the soil matrix as a result of seepage flow. Internal erosion may create open conduit or pipe inside the soil. Suffusion refers to the transport of fine particles through the pores supported by coarser particles without changes in the soil volume (Moffat et al., 2005). It commonly occurs in internally unstable soils with gap graded classification. Suffusion is also described as "internal suffusion" because of the fine particles redistribution inside a layer and changing the local permeability (Kovacs, 1981). It is commonly observed between the core and filter of embankment dams (Garner & Sobkowicz, 2002). The phenomenon is defined as "suffosion" or "external suffusion" by some researchers (Moffat, 2005; Kovacs, 1981) when the coarser fraction of the soil is rearranged by migration of fine particles. The phenomenon suffosion is accompanied by an overall change in the volume of soil. The internal erosion may lead to high seepage velocities and internal instability condition through the soil.

Initiation, continuation, progression and breach are four phases (or mechanisms) of the internal erosion process (ICOLD, 2016). The two first steps are governed at the micro-scale between soil particles and fluid. Initiation occurs when a particle is detached from the soil when the erosive drag force is greater than the resistance forces such as the cohesion, the interlocking effect and the weight of the soil particles. The detached particle is transported in the void

regarding the seepage force. The particle may stop in the layer (suffusion) or be washed out of the layer (suffosion).

Kenney & Lau (1985) defined the term 'internal instability' as "the ability of a granular material to prevent loss of its own small particles due to disturbing agents such as seepage and vibration". They performed laboratory tests on a wide range of gap-graded soils and realized that the initiation and extent of the suffosion process depends on three main criteria (Figure 2.2):

- Mechanical criterion: the fine particles must be under low effective stress and hence transportable under seepage. If the voids between coarser particles not completely occupied by the finer particles, the finer particles carry a relatively low stress. Skempton & Brogan (1994) identified two finer fraction values. The first critical content of the finer particles (blow which do not fill the voids in the coarse component) was estimated between 24-29% of finer fractions by mass for loose and dense samples, respectively. The second critical fraction is an upper limit at 35% at which the finer particles completely separate the coarse particles from one another.
- Geometric criterion: the pore constrictions of the coarser soil must be large enough to allow grains from the finer soil to be transported. Pore size criteria are often based on the ratio between the d_{85} of the finer soil and the D_{15} of the coarser soil (e.g., Sherard et al., 1984), where D_x is the grain size for which x % of the mass is composed of smaller grains.
- Hydraulic criterion: a critical water velocity is needed to induce sufficient seepage forces to move the fine particles through the void space. This family of criteria is closely linked with those developed in the field of sediment transport for fluvial environments (e.g., Cao et al., 2006).

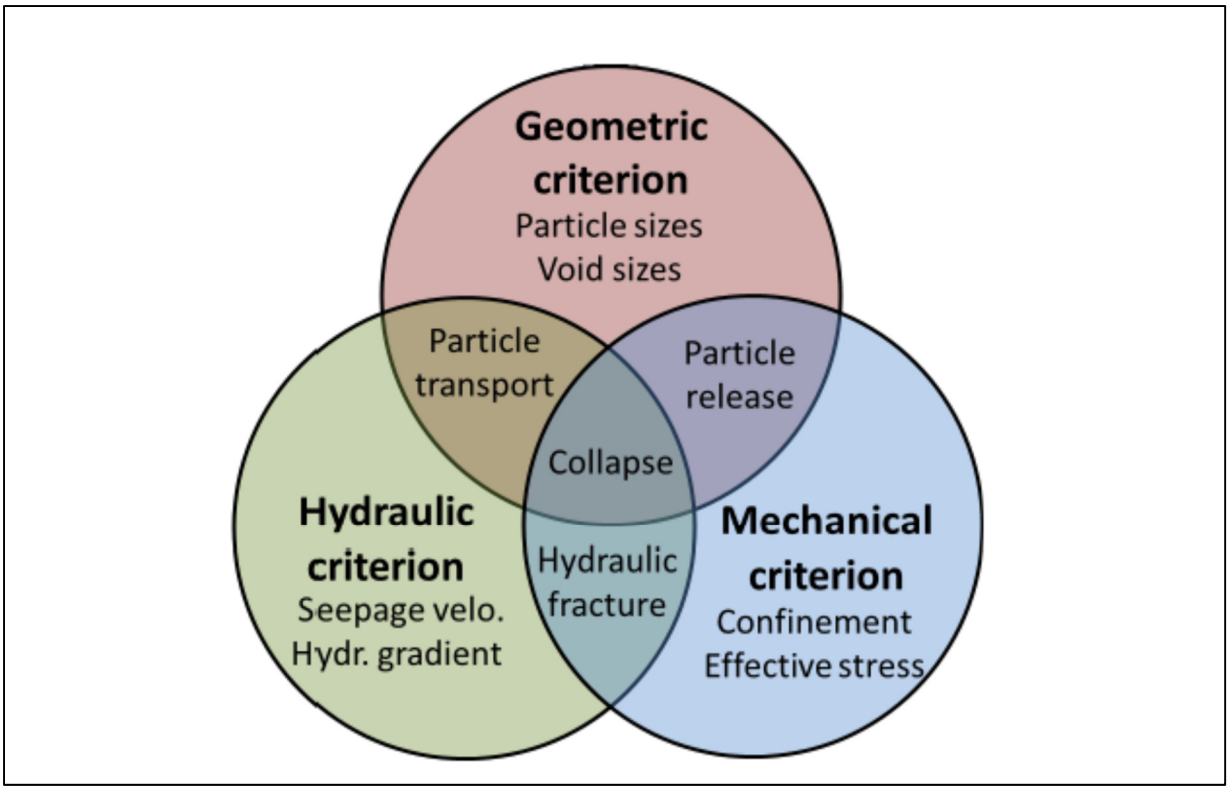


Figure 2.2 Conceptual diagram of internal erosion criteria
Taken from Garner & Fannin (2010)

In a recent review, Philippe et al. (2013) identified geometrical and hydraulic criteria for the erosion of the finer soil at the interface between layers of coarse and fine grained materials. Both the pore size and hydrodynamic families of criteria are fairly well understood for the interface between two uniform sands and simple flow conditions. This is not the case however for widely graded soils (e.g., till), cohesive soils, unsaturated conditions, hydrodynamic transients and more complex geometries (e.g., sloping surfaces and discontinuities). The impact of effective stress on erosion also remains a matter of debate (Philippe et al., 2013; Shire et al., 2014).

Vaughan & Soares (1982) proposed the idea of perfect filter in which a filter will retain the finer particles that could arise during erosion. Filters are used to control internal erosion in embankment dams while allowing seepage flow to exit without causing excessive hydraulic

gradients (Indraratna & Locke, 1999). Two basic functions, retention and permeability, are required of filters in embankment dams. The retention function, also known as stability function, is to prevent the erosion of the base soil through the protecting filter. The permeability function is met when the filter accommodates the seepage flow without the build-up of excess hydrostatic pressure. Permeability ratios of at least 25 are often quoted between the filter and adjacent materials.

Geometric criterion methods are based on particle size distribution. The gradation curve of the dam's core determines the filter grading. The non-erosion filters criteria (Table 1) were developed by Terzaghi and Sherard & Dunnigan (1989) (ICOLD, 2016).

Table 1.1 Criteria for no-erosion filters, Taken from Sherard & Dunnigan (1985, 1989)

Impervious soil group	**Base soil**	***A**** ***(%)***	**Filter criteria**
1	Fine silts and clays	>85	$D_{15} \leq 9\ d_{85}$
2	Sandy silts and clays and silty and clayey sands	40-85	$D_{15} \leq 0.7$ mm
3	Sands and sandy gravels with small content of fines	<15	$D_{15} \leq 4\ d_{85}$
4	Coarse impervious soils intermediate between Group 2 & 3	15-39	$D_{15} \leq (4.d_{85} - 0.7).(40\text{-}A/40\text{-}15) + 0.7$

*A is % of the mass finer than 0.075 mm in the fraction passing the 5 mm sieve

For zoned embankment dams, experimental studies on overtopping involve raising the water level from an operational level below the core to the crest of the dam. In many cases, it has been observed that internal erosion will occur before the water level reaches the crest of the

dam. For example, Wörman & Olafsdottir (1992) progressively increased the water level on the upstream side of a laboratory model of a rockfill dam with a sand filter and a till core. They did not observe internal erosion when the water level reached the interface between core and sand filter. On the other hand, when the water level crossed the interface between sand filter and rockfill, erosion of the filter began at the downstream edge of the interface, a region where water velocities and hydraulic gradients are relatively high. In that manner, Wörman & Olafsdottir (1992) observed erosion for two rockfill materials, both of which did not respect the filter criteria with respect to the sand filter. More recently, Maknoon & Mahdi (2010) confirmed this observation with similar zoned embankment models. The erosion was also observed in the unsaturated portion of the core and filter with zones of low effective stress (Zhang & Chen, 2006).

Most contact erosion experiments have tried to reproduce idealized 1D flow conditions for saturated soils under a constant hydraulic gradient (Guidoux et al., 2010; Wörman & Olafsdottir, 1992). The gradual overtopping of the core of a dam involves time- and space-dependent effective stress, porosity, hydraulic gradient and degree of saturation. Internal erosion also has a feedback on these variables. For example, erosion can induce preferential flow paths and localized deformations that will change permeability, hydraulic gradient and stress tensor (Dumberry et al., 2017).

2.3 Numerical modelling of fluid-particle interaction

Terzaghi's (1925) effective stress principle is the most fundamental theory in soil mechanics. This principle states that the load on a saturated soil is born by the pore pressure and the grain skeleton. When a clay deposit or some other low-permeability soil is loaded, the pore pressure increases as a response to the stress increase. The effective stress is calculated as the difference between the total stress and pore water pressure (or neutral stress).

$$\sigma' = \sigma - P \tag{2.4}$$

Where P, σ' and σ are respectively the pore pressure, the effective stress and the total stress (positive for compression). The pore water pressure and the hydraulic head are related through the following equation:

$$P = (h - z)\gamma_w \tag{2.5}$$

where:

- γ_w is the unit weight of water,
- h is the total hydraulic head, and
- z is a datum head.

The relationship between the effective stress and strain (ε) takes the following generic form (Lewis et al., 1998):

$$\boldsymbol{\sigma'} = D_T \varepsilon \tag{2.6}$$

Where D_T is a tangential matrix.

Deformation and shear strength changes are associated with changes in the effective stress. The effective stress decreases as a result of increasing fluid pressures or increasing the total stress. This can lead to a reduction in shear strength and deformation of the granular material. Increasing pore pressures also cause particle motion in phenomena such as liquefaction and internal erosion in dams. DEM basically models the micromechanical response of granular materials in dry condition so the applied total stress is equal to the effective stresses. There is a need for algorithms to couple particle motion and fluid flow.

In most geotechnical applications, up to three phases (i.e., soil, water and air) typically interact with each other. Independent solutions for one phase or a subsystem is impossible without the

simultaneous response of the others (Zienkiewicz & Taylor, 2000). These kinds of systems are known as coupled models. The coupling can be achieved by overlapping domains or by applying boundary conditions at domain interfaces. Dynamic of fluid structure is a case in point that fluid and structural system cannot be solved independently without considering interface forces. Biot theory of poroelasticity (1941), for instance, considers deformations of the interaction between fluid flow and solid deformations considering the soil particles as a continuum phase. There is mutual influence between the soil structure and the flow of water. The following section introduces basic concepts of fluid-particle interaction in porous media.

2.3.1 Fluid transport equations in porous media

Motion of fluid can be simulated accurately by solving the Navier-Stokes equations on a Eulerian mesh with sub-particle resolution (Goodarzi et al., 2015). The Navier-Stokes equations were developed by Claude-Louis Navier and George Gabriel Stokes in 1822. The Navier-Stokes equations are based on the assumption that the fluid is a continuum. The flow variables (density, velocity and pressure) are also assumed to be continuum functions of space and time.

The Navier-Stokes equations can determine the velocity vector field that applies to a fluid. It can be derived from the application of Newton's second law in combination with a fluid stress and a pressure term. The equations can be obtained from the basic and continuity conservation of mass and momentum, and continuity applied to the fluid properties. The equations are capable of capturing the microscale and macroscale behaviours of the fluid flow and its interaction with the solid bodies.

The equation describing conservation of mass is called the continuity equation. It describes the relation between the fluid velocity (u) and density (ρ) as

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho u) = 0 \tag{2.7}$$

The divergence of density expresses the net rate of mass flux per unit volume. A simpler form of the equation is obtained for an incompressible fluid having a constant density:

$$\nabla \cdot u = 0 \tag{2.8}$$

The differential form of conservation of momentum is given by the Navier-Stokes equation for incompressible Newtonian fluid (equation 2.8) and constant viscosity (μ) would be:

$$\rho g - \nabla P + \mu \nabla^2 u = \rho \frac{\partial v}{\partial t} \tag{2.9}$$

where P is pressure and g is the vector representing the acceleration due to gravity.

Flow through porous media is often modelled by Darcy's law (Darcy, 1856). Henry Darcy performed an experimental study on water flow in a pipe filled with sand. He found that water flow is proportional to the cross-sectional area (A) and head loss (Δh) along the pipe. Darcy's law describes the water flow based on a continuum hypothesis and averaged quantities like permeability.

$$v = -K \frac{\Delta h}{L} \tag{2.10}$$

where L is the flow length (m) and K is the hydraulic conductivity (m/s) which is a measure of porous media's ability to transmit water. Hydraulic conductivity depends on the intrinsic permeability of the porous media, degree of saturation, and water viscosity and density. It can be calculated using the Kozeny-Carman equation (Chapuis & Aubertin, 2003):

$$log[K] = A_c + log\left[\frac{n^3}{(1-n)^2}\,\frac{1}{S_s^2}\,\frac{1}{D_R^2}\right] \tag{2.11}$$

where D_R is the specific weight and A_c is an empirical factor between 0.29-0.51.

The velocity vector in Darcy's law (equation 2.10) is called the apparent velocity and it is different from the real velocity (u) calculated in the Navier-Stokes equation. In Darcy's law, the fluid velocity (v) is calculated by dividing the discharge (m^3/s) by the bulk cross-sectional area of flow (m^2). This value is smaller than the actual velocity because the flow takes place only through the voids. An average velocity in the voids, or seepage velocity (v_s), can be computed as follows:

$$v_s = \frac{v}{n_e} \tag{2.12}$$

The hydraulic gradient describes the water flow direction (seepage) in the soil. It is defined as the relation of Δh (hydraulic head difference) and the length of the flow path (L):

$$i = \frac{\nabla h}{L} \tag{2.13}$$

The hydraulic head in Darcy's law describes the mechanical energy per unit weight of water. It is the sum of the velocity head ($v^2/2g$), which is usually neglected in geotechnical engineering, pressure head ($p/\rho g$) and the elevation with respect to datum (z). The total energy (h) at the flow cross-section ($N.m/N$) is calculated with the following equation:

$$h = \frac{v^2}{2g} + \frac{P}{\gamma} + z \tag{2.14}$$

This equation is called Bernoulli relation and *h* is its constant. If we write Bernoulli's equation between two points (1 and 2) of a fluid volume (in the case of an incompressible and inviscid fluid in a steady irrotational motion):

$$\frac{P_1}{\gamma_w} + \frac{v_1^2}{2g} + z_1 = \frac{P_2}{\gamma_w} + \frac{v_2^2}{2g} + z_2 \tag{2.15}$$

In an anisotropic medium *K* has different values depending on the direction of water flow through the porous media (K_x, K_y, K_z). By combining the continuity consideration (equation 2.8) and Darcy velocity (equation 2.10) in a homogeneity medium, the water flow is shown:

$$\frac{\partial \theta_w}{\partial t} = K_x \frac{\partial^2 h}{\partial x^2} + K_y \frac{\partial^2 h}{\partial y^2} + K_z \frac{\partial^2 h}{\partial z^2} \tag{2.16}$$

Where θ_w is the volumetric water content, and t is time. Equation 2.16 can be expressed as the Laplace equation (equation 2.17) in the case of steady state water flow through an isotropic porous medium ($K_x = K_y = K_z = K$):

$$\frac{\partial^2 h}{\partial x^2} + \frac{\partial^2 h}{\partial y^2} + \frac{\partial^2 h}{\partial z^2} = 0 \tag{2.17}$$

$$\nabla^2 h = 0 \tag{2.18}$$

Although the Navier-Stokes equations describe all forms of flow, Darcy's law is applicable only to laminar flow (O'Sullivan, 2015).

Viscous flow generally is classified as turbulent or laminar. Reynolds (1883) showed the basic difference between categories by injecting a thin stream of dye into the flow through a tube (Kundu et al., 2012). He found that at low flow rate, the fluid moves in parallel layers without overturning motion of the layer. This flow with orderly manner of dye particles is called

laminar (Kundu et al., 2012). In contrast, the dye streak spreads throughout the cross-section of the tube in case of turbulent flow.

The Reynolds number is a dimensionless parameter to determine the flow regime type in a conduit. It is defined as the ratio of the inertial to viscous forces in the flow:

$$Re = \frac{v.D.g}{\mu} \tag{2.19}$$

where v is the average fluid velocity, D is the tube diameter and μ is the fluid dynamic viscosity. In laminar flow, the viscous forces are dominant over the inertial forces. The Reynolds number for flow in porous media can be calculated via equation 2.20 proposed by Tsuji et al., (1993).

$$Re = \frac{n\rho d|V - v|}{\mu} \tag{2.20}$$

where:

- n is the porosity,
- d is the particle diameter,
- V is the average particle velocity,
- v is the average fluid velocity.

Trusel & Chung (1999) observed four regimes of flow in porous media based on Reynolds number. The first regime is limited to $Re < 1$ which is called Darcy regime. There is no inertial effect in laminar creeping.

Forchheimer regime is the second regime. The flow is in strictly steady laminar while the inertial effects become increasingly significant at the upper limit of this regime. The Re is around 100. The third regime is transitional between more or less inertial flows to full inertial

flow. The *Re* is between 100 to 800. Most of this regime is dominated by inertial effects as vortices are shaped at the downstream of the particles. The fourth and final regime is fully turbulent and occurs above *Re* number of 800.

2.3.2 Discrete Element Method

The modelling of internal erosion at the particle scale can be done with the discrete element method (DEM). DEM was first proposed by Cundall (1971; 1974 cited in Cundall and Strack 1979) for the analysis of rock mechanics problems. Finite displacements as well as rotations of particles are simulated (Cundall & Hart, 1993). In DEM system, particles position are automatically updated when they make new contacts or lose them (O'Sullivan, 2015). DEM simulations make possible access to information such as contact forces and contact orientations that are almost impossible to achieve in laboratory tests.

Besides the capability of DEM to simulate complex phenomena in granular materials, the main advantage of DEM compared with other methods is simplicity of governing equations and computational cycle (Figure 2.3). With DEM, Newton's second law of motion (*Force* = *mass* × *acceleration*) is applied individually to each grain of a soil (Cundall & Strack, 1979).

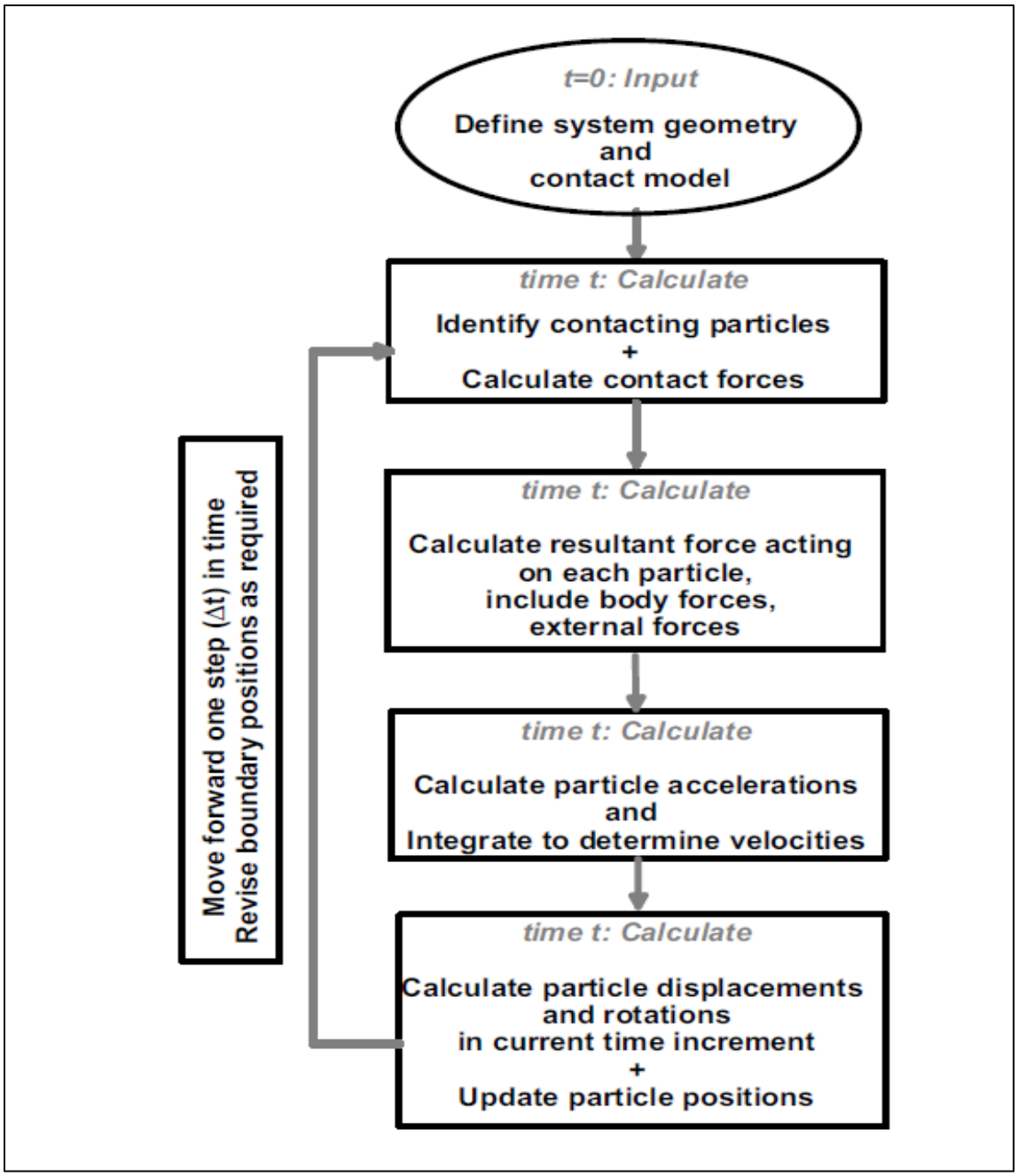


Figure 2.3 Calculation sequences in a DEM simulation
Taken from O'Sullivan (2015)

2.3.2.1 Governing equations of motion

The translational ($\ddot{x}$) and rotational accelerations ($\dot{w}$) of spherical particles are obtained by applying Newton's second law of motion. For the *i-th* element (Figure 2.4) we have:

$$\ddot{x}_i = F_i/m_i \tag{2.21}$$

$$\dot{w}_i = M_i/I_i \tag{2.22}$$

where m_i is the element mass and I_i the moment of inertia. F_i and M_i are the resultant force and moment on each particle:

$$F_i = \sum_{c=1}^{n_c} F_i^c + F_i^{ext} + F_i^{damp} \tag{2.23}$$

$$M_i = \sum_{c=1}^{n_c} (F_i^c \times r_i^c + q_i^c) + M_i^{ext} + M_i^{damp} \tag{2.24}$$

where:

- F_i^{ext} and M_i^{ext} are the external loads,
- F_i^c the contact force at the contact point,
- F_i^{damp} and M_i^{damp} are the force and torque resulting from damping in the system,
- r_i^c is the vector connecting the centre of the i-th sphere with the contact point c,
- q_i^c is the resultant torques due to rolling,
- n_c is the number of particles in contact.

Numerical integration of the accelerations according to a centred finite difference system over a time step (Δt) gives the translational velocity ($\dot{x}$) and rotational velocity (w) of the particle:

$$\dot{x}_i^{[t+\Delta t/2]} = \dot{x}_i^{[t-\Delta t/2]} + \left(\frac{F_i}{m_i}\right)\Delta t \tag{2.25}$$

$$w_i^{[t+\Delta t/2]} = w_i^{[t-\Delta t/2]} + \left(\frac{M_i}{I_i}\right)\Delta t \tag{2.26}$$

The velocities can be numerically integrated to give the new particle positions:

$$x_i^{[t+\Delta t]} = x_i^{[t]} + \dot{x}_i^{[t+\Delta t/2]}\Delta t \tag{2.27}$$

$$w_i^{[t+\Delta t]} = w_i^{[t]} + w_i^{[t+\Delta t/2]}\Delta t \tag{2.28}$$

After obtained new positions, the calculation cycle of updating contact forces and particle locations are repeated to detect new contacts or losing contacts. The forces occurring at the contact point F_i can be decomposed into the normal (F_n) and tangential (F_T) components:

$$F_i = F_n + F_T = f_n.\vec{n} + F_T \tag{2.29}$$

where $\vec{n}$ is the unit vector directed along the line between spheres (i and j) centres at the contact point:

$$\vec{n} = \frac{x_i - x_j}{\|x_i - x_j\|} \tag{2.30}$$

The penetration distance (u_n) is calculated as:

$$u_n = \|(x_i - x_j)\|.\vec{n} - (r_j + r_i) \tag{2.31}$$

$$u_n = \begin{cases} u_n & u_n < 0 \\ 0 & u_n \geq 0 \end{cases} \tag{2.32}$$

The t_u unit vector is defined as:

$$t_u.\vec{n} = 0 \tag{2.33}$$

The simplest constitutive model that can be used to calculate the contact forces described by the normal stiffness k_n, tangential stiffness k_t, the Coulomb friction coefficient μ_c, and the contact damping coefficient c_n (Figure 2.5).

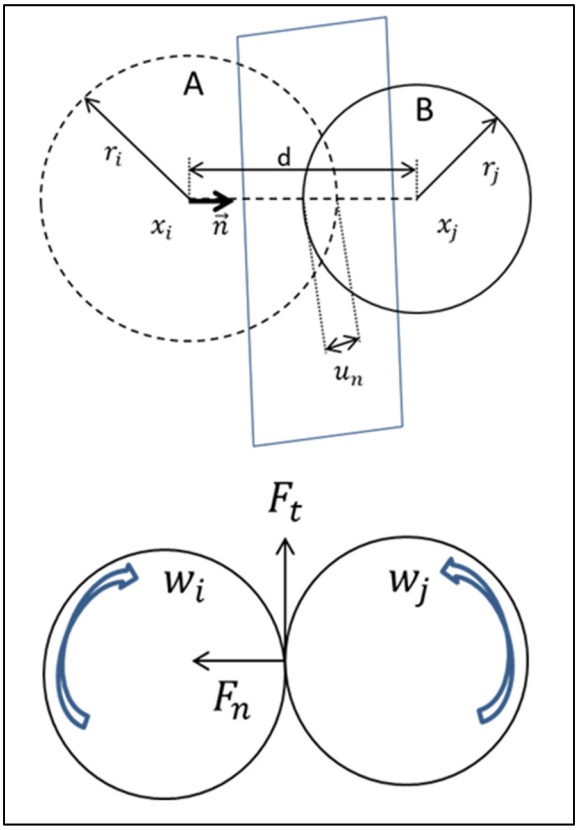


Figure 2.4 Geometry of the contact of two spheres

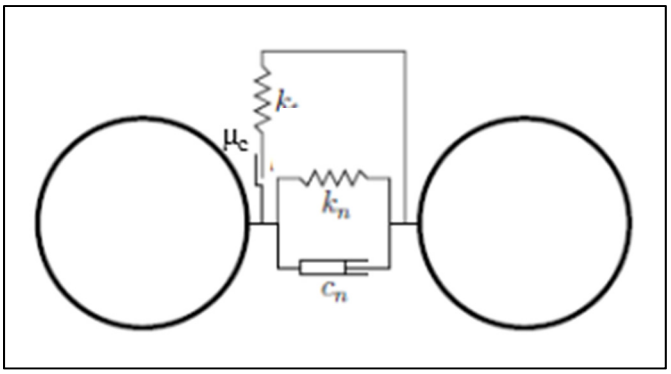


Figure 2.5 Rheological contact model

The normal contact force is defined as:

$$F_n = f_n.\vec{n} = (k_n u_n).\vec{n} \tag{2.34}$$

The tangential force, also known as shear force, acts orthogonal to the contact normal vector. The tangential contact model must be able to describe the material response when the contact is "stuck" (i.e. there is no relative movement at the contact) and when the contact is sliding (O'Sullivan, 2015). The Coulomb friction law is the simplest way to define the contact condition. It is a function of friction coefficient μ_c. If a cohesionless contact is considered for simplicity, the condition for the absence of slippage at the contact can be written as:

$$|F_T| < \mu_c F_n \tag{2.35}$$

When slippage occurs, F_T is given by:

$$|F_T| = \mu_c F_n \tag{2.36}$$

In the absence of slippage, the tangential component is computed as:

$$F_T^{init} = F_T^{t-\Delta t} - k_t\{(\dot{x}_i - \dot{x}_j)t - (w_i r_i + w_j r_j)\}\Delta t \tag{2.37}$$

$$|F_T^t| = F_T^{init} \tag{2.38}$$

otherwise:

$$F_T^t = \mu_c |f_n| \frac{F_T^{init}}{\left\| F_T^{init} \right\|} \tag{2.39}$$

2.3.2.2 Constitutive models

Elastic theory states the relation between the load and deformation at the contact point of two particles. The elastic response in the contact models is typically classified into linear and nonlinear models. The linear elastic models are the simplest kind of contact model to simulate the force-displacement in DEM (O'Sullivan, 2015). The contact normal force is a function of the normal contact stiffness k_n and the overlap at the contact point. Cundall & Strack (1979) defined k_n proportional to the particle size. The model was implemented in the PFC codes (Itasca, 1998).

Two springs stiffnesses k_n^i and k_n^j in the normal direction, one spring for each particle, and two k_T^i and k_T^j in the tangential direction forms the elastic connection between two particles i and j.

$$k_n = \frac{k_n^i k_n^j}{k_n^i + k_n^j} \tag{2.40}$$

$$k_t = \frac{k_t^i k_t^j}{k_n^i + k_t^j} \tag{2.41}$$

Each spring stiffness is a function of the particle elastic modulus E:

$$k_n^i = 4Er_i \tag{2.42}$$

The spring stiffness cannot be directly related to the solid particles' material properties. The Hertzian contact model was developed to address the non-physical nature of the linear spring model. In fact, it relates the spring constant to the material properties. The normal stiffness in the Hertz theory is defined as:

$$k_n = \left(\frac{2}{3}\frac{G\sqrt{2r'}}{(1-v)}\right)\sqrt{u_n} \tag{2.43}$$

where G is the elastic shear modulus, v is Poisson's ratio and r' is defined as:

$$r' = \frac{2r_i r_j}{r_i + r_j} \tag{2.44}$$

Mindlin and Deresiewicz (1953) extended the theory for the tangential force that was not considered in the Hertz model:

$$k_T = \left(\frac{2(G^2 3(1-v)r')^{1/3}}{2-v}\right)|f_n|^{1/3} \tag{2.45}$$

The linear and Hertz-Mindlin contact model developed over the model presented by Mindlin and Deresiewicz are the most common tangential contact models for DEM simulations in geomechanics (O'Sullivan, 2015).

DEM needs an accurate determination of the microscopic properties (e.g., damping, coefficients of friction, etc.) to simulate behaviour of a mass of particles as close to reality as possible. The microscopic properties are determined using macroscopic properties such as Angle of Repose (AoR), particle-size distribution, shear rate, bulk density, triaxial and direct

shear tests. The same AoR can be obtained from a wide range of combinations of rolling and sliding friction coefficients. Thus, AoR alone cannot be considered as a reliable parameter for DEM calibration. Discharging time is another parameter that can be considered along with AoR to determine the microscopic parameters more accurately (Derakhshani et al., 2014). Discharging time is defined as the time that takes the particles leave the top part of the pile.

Derakhshani et al. (2014) determined the microscopic properties of quartz sand through comparing experimental results and DEM results. Particle diameters were in the range between 300 to 600 μm. The Sandglass test (Figure 2.6) was first performed for different amounts of quartz sand to measure the discharging time and AoR. The Sandglass experimental test setup involved the transplant champer which is filled by a certain amount of particles and two Sandglass neck (5 and 8 mm). Sand particles flowed from the upper champer when the plug was pulled from the Sandglass neck. The particle stabilized after a few second in the chamber. The test results were then reproduced by DEM for a varied range of coefficients of rolling and sliding friction. The discharging values were measured along with the AoR in DEM simulations. Rolling and sliding friction coefficients of 0.3 and 0.52 were determined by comparing the AoR and discharging time of materials and DEM results. The DEM model was then validated by a comparison between the experimental results of the conical pile test and DEM simulations. However, the simulations in this study may suffer from the confinement effect of the chamber.

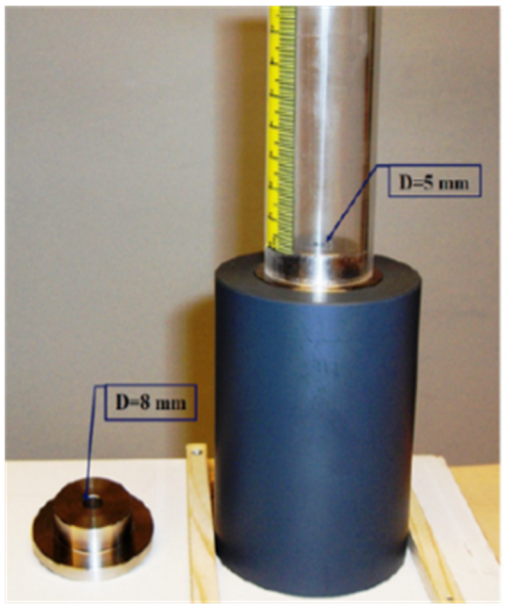


Figure 2.6 Sandglass experimental setup
Taken from Derakhshani et al. (2014)

Dang & Maguid (2018) used heap tests to determine AoR of rock clusters (Figure 2.7). They draw a relationship between the AoR and the inter-particle friction coefficient required for the DEM analysis of the rock particles. Since particle shapes play a key role in DEM simulations, both spherical and clumps were used to better understand the influence of particle sphericity and angularity on the results. Particle irregularity can be introduced in the DEM models by clumping spheres of different sizes. Four clump templates (tetrahedral, cubic and octahedral) were chosen to resemble the shape of rock used in the experiments (Figure 2.7).

The average measured AoR of 35 laboratory heap tests was 25.2°. In DEM simulations, the friction coefficient was incrementally changed to reach the calculated angle of repose of 25.2°. The friction coefficients were estimated at 0.55, 0.35 and 0.25 for tetrahedral, cubical and octahedral clumps, respectively. While the maximum AoR of about 21° was found for the spherical particles which corresponded to a friction coefficient of about 0.9.

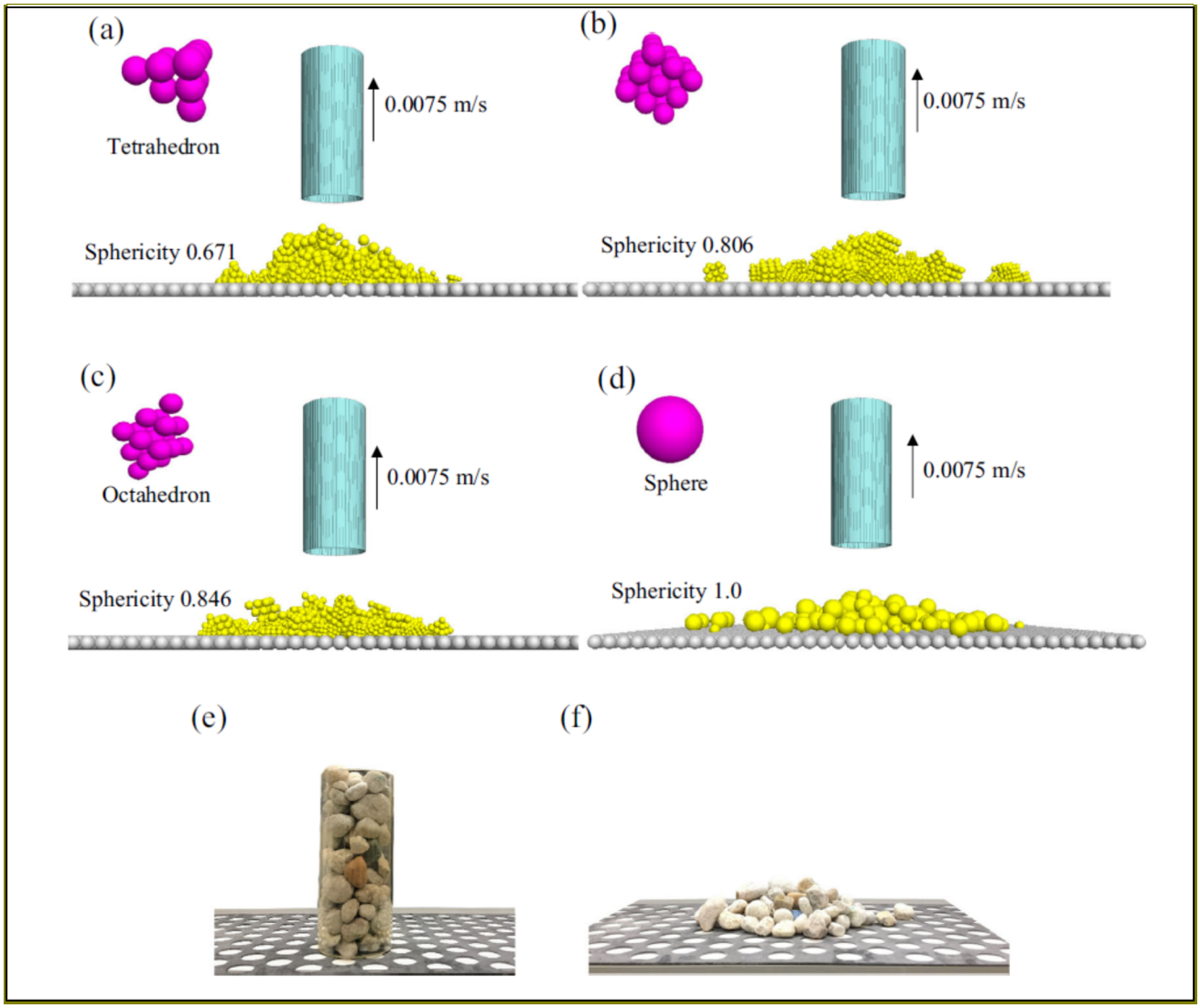


Figure 2.7 Experimental and numerical modeling of the repose angle tests:
(a) tetrahedral clumps; (b) cubic clumps; (c) octahedral clumps; (d) sphere;
(e) & (f) experimental observations
Taken from Dang & Maguid (2018)

The Young's modulus is not the same as the contact stiffness. In DEM code YADE, with FrictMat materials and FrictPhys interactions, a fictitious Young's modulus (E) is used to define the normal contact stiffness K_n once two spheres form an interaction. For two spheres with Young's modulus E and diameters d_1 and d_2, K_n can be calculated as follows (Šmilauer et al., 2015b):

$$K_n = E\left(\frac{d_1 d_2}{d_1 + d_2}\right) \tag{2.46}$$

YADE uses only a static friction angle which defines the microscopic friction coefficient between two individual bodies. The macroscale internal friction angle is the angle of inclination of the Mohr-Coulomb failure criterion with respect to the horizontal axis.

2.3.2.3 Damping

The elastic contact models used in DEM cannot describe the contact normal response realistically due to a lack of energy dissipation as a result of damage and contact separation in the normal contact direction. If we consider the particles in a DEM model as a connected system of springs, the system vibration will never stop. Artificial damping is defined by DEM users to avoid such non-physical phenomenon. Viscous and non-viscous dampings are the two main types of damping in DEM. In both cases, both normal and tangential damping terms (F_n^d and F_T^d) are added to equations 2.27 and 2.28.

In the case of viscous damping, Cundall & Strack (1979) proposed a system of global damping in which damping is considered as the effect of dashpots connecting each particle to the ground. The value of the damping force on each particle is proportional to the magnitude of the translational and rotational velocities:

$$F_n^d = -\alpha\, m_i\, \dot{x}_i \tag{2.47}$$

$$F_T^d = -\alpha\, I_i\, w_i \tag{2.48}$$

where α is the damping constant.

Cundall (1987) presented a non-viscous damping system to overcome some limitation of the viscous damping (e.g., application equal damping to all nodes). The non-viscous damping acts at each node, independent from each particle, to limit the particles vibrations. The damping force is proportional to the magnitude to the resultant force and resultant moment. The damping reduces the driving forces and increases those forces resisting the motion. The damping force acts in the opposite direction of the velocity ($\dot{x}$):

$$F_n^d = -\alpha \left| \sum_{c=1}^{n_c} F_i^c + F_i^{ext} \right| \frac{\dot{x}_i}{\|\dot{x}_i\|} \tag{2.49}$$

$$F_T^d = -\alpha \left| \sum_{c=1}^{n_c} (F_i^c \times r_i^c + q_i^c) + M_i^{ext} \right| \frac{w_i}{\|w_i\|} \tag{2.50}$$

The main feature of non-viscous damping is that it is applied only on accelerating motion. It thus avoids erroneous damping forces derived from steady-state motion (Itasca, 2004).

2.3.2.4 Numerical stability

A limitation on the time step (Δt) is needed to ensure the stability of the explicit integration scheme. Time step cannot be bigger than the critical time step (Δt_{crit}). The critical time step is evaluated as the maximum of the natural period of the mass-spring system (Šmilauer et al., 2015b):

$$\Delta t_{crit} = \frac{2}{w_{max}} \tag{2.51}$$

where w_{max} is defined as:

$$w_{max} = max\sqrt{\frac{k_i}{m_i}} \tag{2.52}$$

where k_i is an equivalent stiffness that is evaluated considering all the contacts of one particle. The critical time step is then:

$$\Delta t_{crit} = min\ 2\sqrt{\frac{m_i}{k}} \tag{2.53}$$

2.3.2.5 DEM applications in geotechnical engineering

Block and particulate DEM are the two main types of discrete element models in geomechanics (O'Sullivan, 2015). Block DEM codes are used to model rock blocks and masonry structures such as stone retaining walls (Basarir et al., 2008). The use of particulate DEM in geotechnical engineering is becoming increasingly common with increasing computational power. DEM has been used for analyzing different aspects of soil mechanics such as granular mechanics (Thornton, 2000), anisotropy of clay (Yao & Anandarajah, 2003; Anandarajah, 2003), particle fracture and crushing (Lu & McDowell, 2006; Cheng et al., 2003), strain localization (Jiang & Yin, 2012; Mohamed & Gutierrez, 2010) and soil-structure interaction (Dang & Meguid, 2013). The DEM is also applied to a wide range of research areas outside of geotechnical engineering in physics, mathematics, chemical engineering, geology, material science, etc. Zhu et al. (2008 and 2007) provided two useful reviews on application of DEM and developed algorithms in chemical engineering fields.

A large number of DEM codes are currently available for applications in geotechnical engineering. Some of them allow different methods to be combined, such as PFC3D and YADE which can respectively be combined with finite difference code FLAC and an in-house finite difference code developed at McGill (Tran et al., 2018). The available continuum-discontinuum interfaces either limit the modification that can be made to the code or require

the researchers to write their own code which can be very time-consuming. Most studies have used commercial codes PFC2D and PFC3D (Itasca, 2004) for elemental studies where different external stress tensors are applied on soil elements to study their behaviour and to reproduce the macroscopic behaviour of soils. The code can be modified using an embedded scripting language named FISH and recently using Python scripts.

Since their source codes can be accessed, open source and in-house DEM software packages like LIGGGHTS (Kloss & Goniva, 2011), YADE (Kozicki & Donzé, 2009) and ESySParticle Simulation (Weatherley, 2009) offer many advantages for research projects where new DEM applications and tools are to be developed (O'Sullivan, 2015). The molecular dynamic code LAMMPS (Plimpton, 1995) is becoming more common for DEM simulations. It is an open-source code for parallel computing on distributed memory machines using message-passing interface (MPI) techniques. The parallel processing is suitable for simulation of large number of particles. However, there is communication cost between cores; so, increasing processor does not reduce running time on a linear scale (O'Sullivan, 2015). The granular contact model LIGGGHTS developed by Kloss and Goniva (2010) is based on LAMMPS. YADE is a very extensible DEM package that lets researchers add plug-ins for new methods and numerical models in a single package. Its preprocessors and post processors are already developed so the researchers mostly need to revise the equations. YADE adopts shared-memory parallel execution environment using OpenMP (Open Multi-Processing) and a shared memory strategy. It increases the calculation speed on multiprocessor systems.

YADE is coupled with open source FEM code Escript (Gross et al., 2007). Escript is a Python library to solve boundary value problems. It does not include a graphical user interface. Users need to work with Python scripts. The only example of a coupled Escript-YADE model that we are aware of is the one presented by Guo & Zhao (2014). This reference concerns the modelling of dry granular material and is discussed in section 2.4.

The geotechnical research group at McGill University (Dang & Meguid, 2013; Tran et al., 2018) coupled YADE with an in-house FEM code. The FEM was implemented in the YADE source code in C++. Users can develop their own analyses using Python scripts.

2.3.2.6 YADE framework

The YADE framework permits changes, extensions and code reuse besides providing many low-level operations through plug-ins and libraries (Kozicki & Donzé, 2009). Figure 2.8 shows the schematic framework of YADE. The framework is divided into several layers and each layer depends on the layers below.

The different layers in figure 2.8 were described by Kozicki & Donzé (2009). The first layer is the library layer. The class factory takes the class name of plug-ins to be loaded or unloaded as a string. It can make the relation between different installed classes (e.g., different plug-ins used to solve contact interactions) possible. The generic layer is the core of YADE and it links the different engines, bodies and interactions which are part of all DEM simulations. The next layer is called the common layer. It embeds components which are commonly applied by different simulation types such as Newton's law of motion or damping methods Cundall & Strack (1979). The common layer could be used to extend YADE, for example by adding an FEM package. The specialized layer is based on the common layer but the code in this layer cannot be shared between different methods. The top layer is graphical user interface. There is an interface that can perform computation remotely.

Figure 2.9 depicts the schematics of a simulation loop in YADE. The loop relies on algorithms which are regarded as *engines* in YADE to detect and process the interactions. The engine outputs can be forces and displacements. The output generally results in a response that influences body state. In total, there are three types of data structures in YADE: bodies, interactions and intermediate data.

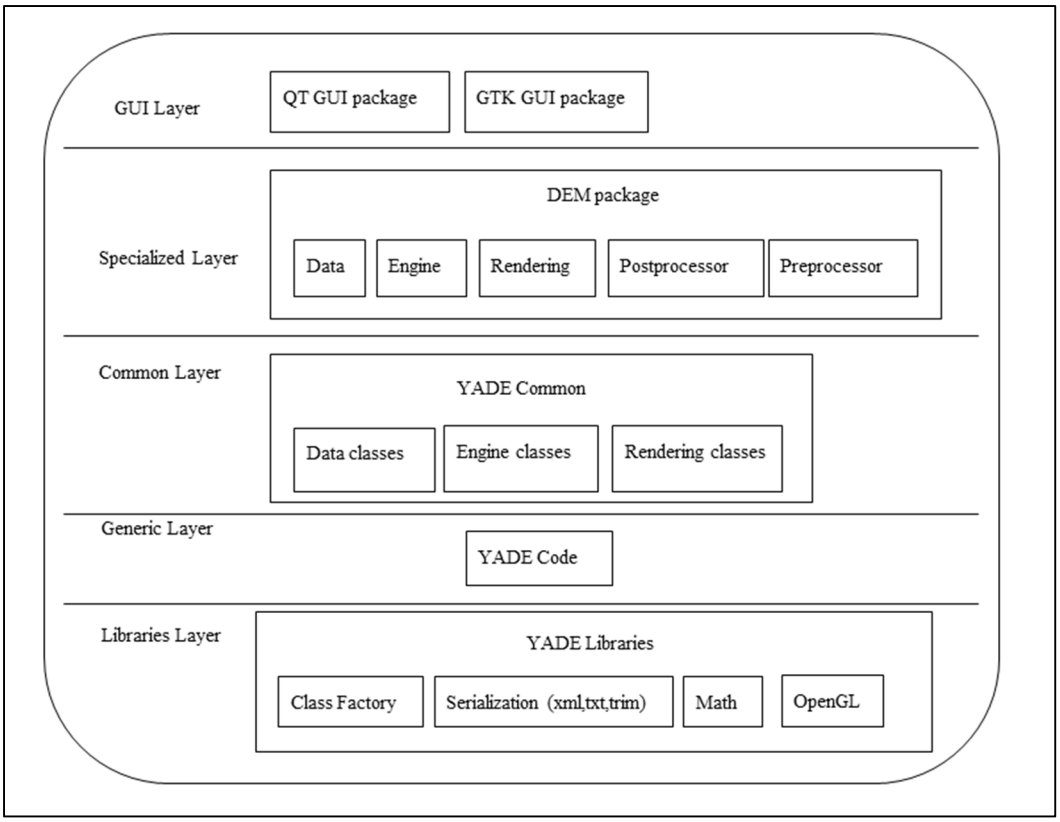


Figure 2.8 Layered structure of YADE framework
Taken from Kozicki and Donzé (2009)

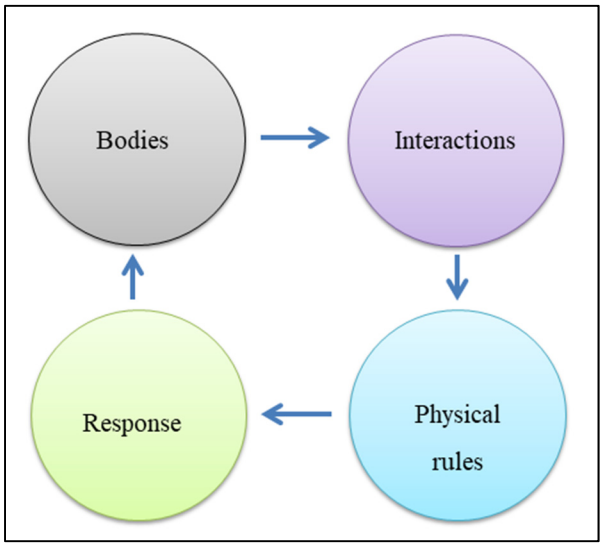


Figure 2.9 Simplified schematics of simulation loop

2.3.3 Coupling DEM-flow schemes

Particles motion in a saturated porous media is influenced by the fluid. Drag force is the dominant fluid interaction force and considered as the most important parameter in particle motion. Thus, the accuracy of fluid-particle interaction models bears a close relationship with drag force calculations. For geotechnical applications, drag force can be calculated at the pore scale or at a coarser continuum scale where individual particles are not taken into account. The pore-scale methods ideally simulate the fluid and a system of particles by solving the Navier-Stokes at the sub-void or pore scale. It is considered the most accurate technique to model fluid flow through porous media.

Computational fluid dynamics (CFD) use numerical methods to analyze and solve problems involving fluid flows. Numerical methods need computers to perform the calculations required to simulate and analyze fluid flows with respect to a surface. Most CFD methods can be used to solve the flow at sub-particle discretization. Mesh-based numerical methods such as finite volume, finite difference and finite element methods have been widely used to solve the flow at the pore scale.

The lattice Boltzmann (LB) method has recently become a popular method to solve the Navier-Stokes equation on a mesh with sub-particle resolution (Rubinstein et al., 2016). In the LB method, the fluid is modelled as fluid pockets that move about a 3-D mesh of nodes (Rubinstein et al., 2106). The Boltzmann equation governs the fluid mass transportation from one node to another one. The particles in the simulation overlap with the mesh and a no-slip condition is considered along the particle boundaries (O'Sullivan, 2015).

Pore network modelling (Chareyre et al., 2012) is another method to solve the fluid interaction with particles using a pore-scale finite volume formulation. The pore network is decomposed using a triangulation method and Voronoi graphs. The flow rate in the pore network is then assumed to be proportional to the pressure gradient and a local conductance value function of

the pore throat geometry. This method requires less computing resources than solving the Navier-Stokes equations at the microscale. Forces on particles obtained from pore network flow have been shown to be comparable to those obtained by solving the Navier-Stokes equation for a 9-sphere assembly with two distinct diameters.

The smoothed particle hydrodynamics (SPH) method has recently received some attention. In this method, a set of points represents the state of the system. The SPH points carry material properties and interact with each other through a weight function or smoothing function. SPH is a mesh-free lagrangian method that is considered a special advantage over the traditional grid-based methods (Liu & Liu, 2010).

Pore scale methods are computationally expensive due to the large number of pores and the complex geometry of soils involving large numbers of particles. Therefore, coarse-grid methods are commonly used in geotechnical engineering to decrease the hydrodynamic calculation costs. The coarse-grid method (CGM) was originally proposed by Tsuji et al. (1993) for the modelling of fluidized bed applications. The drag force is calculated using the average parameters of each cell. Pressure and fluid velocities are calculated on a grid that is coarser than the average particle diameter. The flow velocity and pressure are determined by the averaged Navier-Stokes and momentum equations presented by Zhu et al., (2007), Kafui et al., (2002) and Tsuji et al., (1993). CGM drag force (F_D) is often calculated based on empirical relations in the geotechnical literature. Ergun (1952) is a widely used empirical drag equation for particle systems where porosity (n) is less than 0.8 (O'Sullivan, 2015; Rubinstein et al., 2016):

$$F_D = \beta \frac{v - V}{\rho} \tag{2.54}$$

$$\beta = 150\,\mu \frac{(1-n)^2}{d^2\,n^2} + 1.75 \frac{(1-n)\rho|v - V|}{nd} \tag{2.55}$$

Zeghal & El Shamy (2004) used Ergun equation to compute volumetric average drag force for each finite volume using averaged particle size and fluid velocity. Then, the drag force on particles was calculated by distribution of the drag force among the particles in each finite volume proportionally to their volume. Assuming laminar flow in soil mechanics, drag force can be calculated based on Darcy's law and permeability prediction methods, such as the Kozeny-Carman equation (Pirnia et al., 2019; Chapuis & Aubertin, 2003). Accuracy of coarse-grid methods in modelling monodispersed and polydispersed particle systems will be discussed in more details in Chapter 5.

2.4 Numerical modelling of internal erosion in embankment dams

Numerical modelling is an efficient method to study and analyze internal erosion in existing saturated soil structures. Internal erosion is typically modelled based on continuum or discontinuum methods. FEM is the most common type of continuum model to study seepage and stresses-strain in embankment dams through the past 40 years (Day et al., 1998; Hnang, 1996; Ng & Small, 1999; Sharif et al., 2001; Zhang & Du, 1997). FEM discretize the porous medium body to small elements.The physic of each element is considered continuous. Most seepage analyses are based on FEM and it is typically modelled by solving a water conservation equation (equation 2.16). The continuum-based theory of porous media solves particle and water conservation equations based on average characteristics of the material (e.g., porosity) for a representative elementary volume (REV). Internal erosion can be modelled by the mass exchange between the fluid and the solid continuous phases in a continuum-based theoretical framework. The solid mass is decreased and transferred to the fluid phase as erosion progresses.

The continuum method was first proposed by Vardoulakis et al. (1996) to simulate the sand production problem for radial and axial flow in which sand and fine particles are displaced from the soil matrix due to high fluid flow and stress changes. The sand production happens during pumping fluid from the porous media. The proposed model was based on mass balance

equations of three continuous phases for each REV consisting of a solid skeleton, fluidized or suspended particles and the fluid. A phenomenological erosion law was used for the mass generation term in the conservation equation of the suspended phase. Rotunno et al. (2018) developed a novel formulation based on the mass balance equation of Vardoulakis (1996) for the resolution of the backward piping problem in a multidimensional porous medium. The model was used to simulate both the propagation and enlargement of the pipe at the scale of the hydraulic structure. Flow is assumed as laminar in porous media and turbulent in pipes. The fluid mass and the fluid pressure values are exchanged between two systems. The model could reproduce some features of backward erosion piping observed in different experimental tests. Abdou et al. (2018) recently proposed a numerical approach at a representative elementary volume (REV) scale to model the transient and spatial evolution of the average porosity of a porous medium using a FEM software (COMSOL Multiphysics). The method is based on the erosion model of Vardoulakis (1996) while the mechanisms of erosion and deposition are characterised by flow velocity thresholds. The erosion and deposition terms used in the constitutive law increase and decrease the porosity, respectively. Internal erosion occurs when the forces from the fluid are higher than the forces that keep the particles together.

Continuum models face fundamental limitations when dealing with internal erosion phenomena which are controlled by mechanisms at the particle scale. The discrete element method (DEM) has been used to study micro-mechanical response of granular materials. DEM models suffer from two shortcomings. The first drawback is the excessive computational cost for modelling large scale applications. The second issue is the inability to give real shapes to particles although clumps have been used to model simple particle shapes.

For most soil mechanics applications, the numerical modelling of internal erosion needs a multiscale framework (e.g., Frishfelds et al., 2011). A multiscale method may take advantages from both FEM and DEM. Multiscale methods generally use information at smaller scale to eventually model the response of the material at larger scale. The multiscale methods obtain the continuum response without resorting to phenomenology (Andrade & Tu, 2009). For

granular materials, results from modelling at the micro scale have a feedback on the large scale continuum model by locally changing the permeability and by inducing local volume changes because of particle entrainment. Changes on the continuum model also have an influence on the particle scale model as seepage can be concentrated in areas where erosion has already taken place, thus inducing further erosion.

The existing hybrid or multimethod DEM-flow models (explained in 2.3.3) can only simulate internal erosion in small-scale soil assemblies and often need massive computational resources due to too large number of particles in DEM simulations. There is a need for a multiscale computational algorithm that overcomes the DEM limitation regarding the number of discrete bodies and eventually allows the modelling of internal erosion for large structures.

A promising approach is to couple DEM with continuum models in a multiscale analysis where small scale DEM simulations are conducted for selected nodes in the model. These analyses are described as hierarchical. The large-scale model uses the information from a discrete model as an input to model the material behaviour (Andrade & Tu, 2009). A hierarchical multiscale model aimed at monitoring strain localization problem based on the assumption of a simple plasticity model at the macroscale was developed by Andrade et al. (2011). Their approach evades phenomenological nature because it obtains plasticity parameters directly from DEM. Some improvements have recently been made to reach fully hierarchical multiscale models. Guo & Zhao (2014), Nguyen et al., (2013), Dang & Meguid (2013) and Stransky & Jirasek (2012) have presented hierarchical multiscale discrete-continuum frameworks in which a large scale continuum model based on FEM was coupled with small scale DEM simulations conducted at each Gauss point of the large-scale FEM mesh. The geometric information (strain) from the FEM model is transferred to the microscale model at representative volume element (RVE) to solve the boundary value problem (BVP) using periodic boundary conditions (Figure 2.10). The stress tensor and the constitutive characteristics (stiffness tensor) are sent back to the macro-scale problem to update the strain values. A main drawback with the method is the large number of DEM iteration steps that are needed to reach local convergence (the

Newton–Raphson scheme used to update the finite element solution). Wang & Sun (2016) and Guo & Zhao (2016) extended the same hierarchical multiscale scheme to saturated porous media by solving Darcy's law at the macroscale.

There is no example of hierarchical multiscale model for the modelling of internal erosion in the literature. The examples that were mentioned previously are all centred on the mechanical behaviour of porous media. Chapter 4 of this thesis presents the first multiscale hierarchical framework aimed at modelling internal erosion for large-scale particle assemblies. The model solves fluid flow and particle conservation in the porous media by FEM and calculates particle flux based on particles displacements in small subdomains along the discrete body with DEM. In other words, DEM provides FEM with parameters that depend on microscale behaviour at specific points of the geometry.

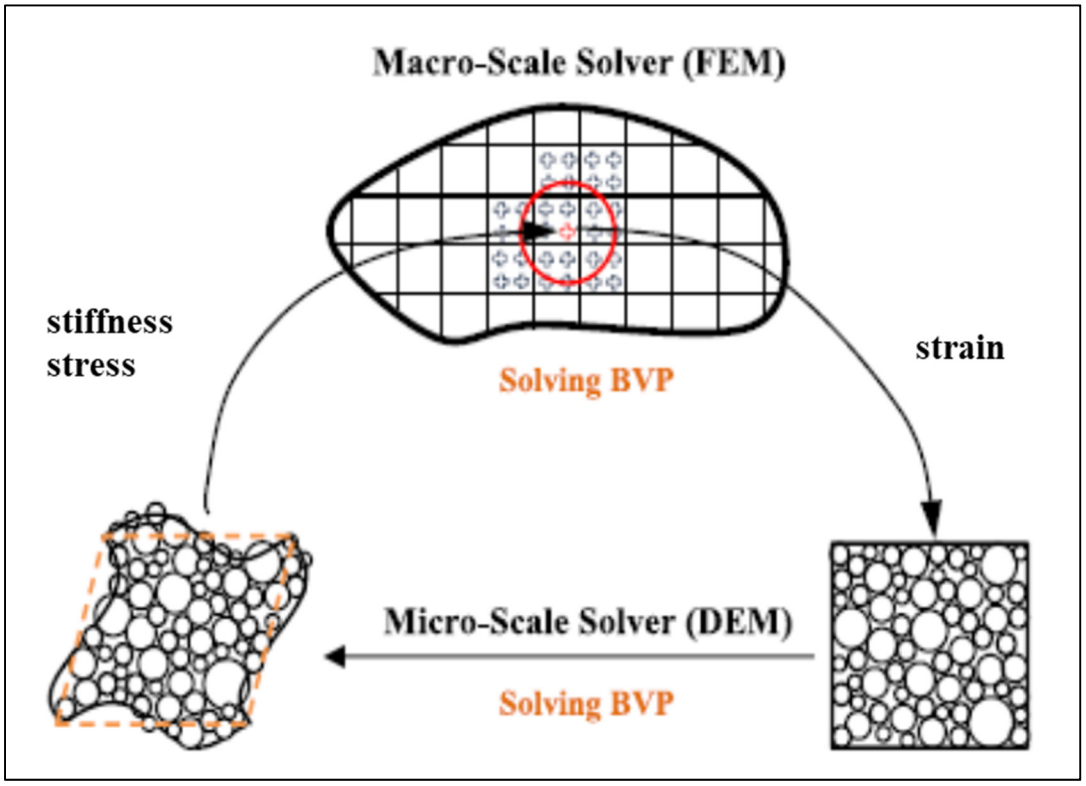


Figure 2.10 Schematic of hierarchical multiscale modelling
Taken from Liu et al. (2015)

2.5 Multiphysics models

Recently, a growing trend has been observed toward the development of Multiphysics and multipurpose software packages. Multiphysics codes involve the coupled simulation of multiple phenomena. It can involve the resolution of sets of partial differential equations or the combination of different model types, for example finite elements and molecular dynamics models. A review by Keyes et al. (2013) showed that there are now numerous Multiphysics models: PETSC (Balay et al., 2013), MUSE (Zwart et al., 2009), OOFEM (Patzák & Bittnar, 2001), Chombo (Trebotich et al., 2008), Fenics (Logg et al., 2012) and COMSOL Multiphysics (COMSOL, 2013). COMSOL Multiphysics is one of the most popular Multiphysics codes in the scientific and industrial communities because of its flexibility and its pre- and post-processing interface.

The physics that can be modelled include fluid flow, seepage, chemical reactions, stress-strain behaviour and heat transfer. COMSOL can also be used to solve custom partial differential equations (PDEs), ordinary differential equations (ODEs) and initial value problems. Two main modules are available for geotechnical engineering applications with COMSOL Multiphysics:

- Subsurface Flow Module: It can be used for simulating fluid flow in porous media, groundwater flow, spread of pollutants through soil and flow of oil and gas to wells. It can link this flow with other phenomena such as chemical reactions and heat transfer.
- Geomechanics module: It can be applied to analyze stress-strain behaviour for geotechnical applications like foundations, tunnels, excavations and slope stability.

The advantages of using COMSOL are stated at the beginning of the methodology section in Chapter 3. Their presentation also highlights COMSOL's main advantages: its user-friendly GUI, its large number of preprogrammed differential equations than can be coupled, its interface allowing users to program their own differential equations (e.g., Duhaime & Chapuis, 2014), and its JAVA API that facilitates the development of new applications for the interface and the programming of add-ons (e.g., the Amphos 21 iMaGe project, Duhaime et al., 2017).

COMSOL's main disadvantage is its cost as it is a commercial software package. However, combination of an open-source DEM code with the JAVA interface of COMSOL remains relatively inexpensive.

CHAPTER 3

ICY: AN INTERERFACE BETWEEN COMSOL MULTIPHYSICS AND YADE

Pouyan Pirnia [a], Francois Duhaime [b], Yannic Ethier [c], Jean-Sebastien Dube [d]

[a, b, c, d] Department of Construction Engineering, École de technologie supérieure, 1100, rue Notre-Dame Ouest, Montréal, Québec, Canada, H3C 1K3

Published in *Computers and Geoscienc,* February 2019

3.1 Abstract

The thermal, mechanical and hydrodynamic behaviour of porous media in geoscience applications is usually modelled through the finite-element (FEM) or finite-difference methods. These continuum models tend to perform poorly when modelling phenomena that are essentially dependent on behaviour at the particle scale or phenomena that are not accurately described by partial differential equations (PDE), such as internal erosion and filtration. The discrete nature of granular materials can be modelled through the discrete-element method (DEM). However, in some instances, DEM models would benefit from an interface with continuum models to solve coupled PDEs or to model phenomena that occur at a different scale. This paper introduces ICY, an interface between COMSOL Multiphysics, a commercial finite-element engine, and YADE, an open-source discrete-element code. The interface is centred on a JAVA class. It was verified using the simple example of a sphere falling in water according to Stokes' law. For this example, the drag force was calculated in COMSOL and body forces (gravity, buoyancy and drag) on the sphere were summed in YADE. The paper also presents an application example for the interface based on the modelling of internal erosion tests.

3.2 Introduction

Flow through porous media, like soil deposits or earth dams, has conventionally been analyzed within a continuum framework. Continuum models have had particular success in capturing some important aspects of porous media behaviour, such as seepage and stress-strain behaviour. Nevertheless, some phenomena, such as internal erosion, derive from complex microstructural mechanisms at the particle scale that cannot currently be upscaled and described by macroscale partial differential equations (PDE). Since continuum models do not explicitly take into account the discrete nature of porous media, phenomena like internal erosion should be modelled at the particle scale (Guo and Zhao, 2014). At the same time, these phenomena often depend on macroscale parameters such as stress and pore pressure. A multiscale approach is thus needed.

The discrete-element method (DEM) is becoming increasingly common in the modelling of porous media (O'Sullivan, 2015). With DEM, the motion and interaction (contact forces) of a large number of small particles are computed. This approach considers explicitly each particle in a granular porous media and the contact forces between them. Hence, it can simulate finite displacements and rotations of particles (Cundall and Hart, 1992). Besides the capability of DEM to simulate complex phenomena in granular materials, the main advantage of DEM compared with other methods is the relative simplicity of governing equations and computational cycle.

The discrete element method has had great success in reproducing the mechanical response of dry granular material at both the particle and continuum scales (e.g., O'Sullivan et al., 2008). However, for field scale applications, such as earth dams, it is not feasible to model structures solely with DEM. The current practical limit on the number of particles in a model using personal computers is around 100 000 (O'Sullivan, 2015). For fine sand with a uniform diameter of 0.10 mm, this translates to a maximum model volume on the order of 70 mm^3 for hexagonal close packing. As a consequence, to be included in the modelling of large-scale applications, DEM must be coupled with continuum models in a multiscale analysis where

small scale DEM simulations are conducted for selected nodes in the model. The continuum model can also be used to calculate boundary conditions for the DEM simulations (e.g., hydraulic gradient and stress). This type of hybrid multiscale models remains in development and has not seen widespread use (Indraratna et al., 2015; Chareyre et al., 2012; Elmekati & El Shamy, 2010; Eberhardt et al., 2004).

A large number of DEM codes are currently available for applications involving granular media. The most common commercial DEM codes are PFC2D and PFC3D (Itasca, 2014). These codes have been used in a significant number of elemental studies where different external stress tensors are applied on soil elements to study their behaviour and to reproduce the macroscopic behaviour of soils (O'Sullivan, 2015; Ding, 2013). Although these commercial software packages allow some modification using an embedded scripting language named FISH, it is not as versatile as some open source codes. Python has recently been integrated directly into PFC 5.0. It allows models to be manipulated from Python scripts.

The molecular dynamic code LAMMPS (Plimpton, 1995) is one example of an open-source code that can be used for DEM simulations. It allows parallel computing on distributed memory machines using message-passing techniques (MPI). Parallel computing makes LAMMPS suitable for the simulation of large numbers of particles. However, increasing the number of processors does not reduce computation time linearly (O'Sullivan, 2015). LAMMPS and its derivative LIGGGHTS, have been used for some applications involving granular materials (e.g., Huang et al., 2013; Bym et al., 2013). LIGGGHTS stands for LAMMPS Improved for General Granular and Granular Heat Transfer Simulations.

YADE ("Yet Another Dynamical Engine") is a highly flexible and extensible open-source DEM package used in geotechnical engineering. The YADE framework permits changes, extensions and code reuse besides providing many low-level operations through plugins and libraries (Kozicki & Donzé, 2009). YADE has been developed for shared-memory parallel execution environment using OpenMP (Open Multi-Processing) increase the calculation speed

on multiprocessor systems. YADE has been shown to require computation times that are similar to PFC3D (Jakob, 2012). A few methods can already be used with YADE to calculate hydrodynamic forces on particles: pore network flow and Lattice-Boltzmann method (Lominé et al., 2013; Chareyre et al., 2012).

There are already a few examples of DEM and FEM codes that can be interfaced. For instance, PFC3D has a Computational Fluid Dynamics (CFD) module that allows fluid-particle interactions to be modelled based on the volume-averaged coarse-grid approach (Furtney et al., 2013). Goniva et al. (2010) developed a CFD-DEM coupling to solve fluid-particle interactions using OpenFOAM, a finite-volume code, with LIGGGHTS. Zhao & Shan (2013) modified the OpenFOAM library to solve the locally averaged Navier–Stokes equation based on the coarse grid approximation method proposed by Tsuji et al. (1993). The interface between the open-source FEM platform Kratos and DEM engine DEMPack is another example. It has made possible the coupling of DEM with fluid dynamic, heat transfer and structural analyses in the same package (Isach, 2013). Finally, Guo and Zhao (2014) have presented a framework to couple a large scale continuum model based on FEM with small scale DEM simulations conducted at each Gauss point of the large-scale FEM mesh. Open source codes Escript (Gross et al., 2007) for FEM and YADE for DEM were used.

Hybrid FEM-DEM models for soils often require computing resources that are not readily available. For instance, the Guo & Zhao (2014) model required approximately 4 hours of computing time with 16 processors to solve a 2D problem involving only 240 elements. Also, existing interfaces are often programmed with specific applications in mind and often require extensive programming to develop new applications.

Recently, a growing trend has been observed toward the development of versatile multiphysics finite difference and finite element software packages. Multiphysics codes involve the coupled simulation of multiple phenomena. They can involve the resolution of sets of PDEs, for example the combined analysis of stresses and strains, heat transfer, seepage and solute

transport in a porous media (e.g., Finsterle et al., 2014). They can also involve the combination of different model types, for example finite elements and molecular dynamics models (Keyes et al., 2013).

Some multiphysics software packages, such as COMSOL Multiphysics (COMSOL, 2016a), can be integrated in scripts or linked with other codes (Duhaime & Chapuis, 2014; Nardi et al., 2014). COMSOL and PHREEQC, a thermodynamic equilibrium code, were successfully coupled by Nardi *et al.* (2014) to create an interface named iCP (Interface COMSOL-PHREEQC). iCP is written in JAVA and uses the COMSOL JAVA-API and the IPhreeqc C++ dynamic library.

This paper presents ICY, a multipurpose interface that allows data to be exchanged between continuum models based on COMSOL Multiphysics (FEM) and particle scale model based on YADE (DEM). Details on the interface code are first given. The interface is then verified with the simple example of a particle falling in water according to Stokes' law. An application example involving the modelling of internal erosion tests in porous media is also presented. In this example, DEM is used to compute particle displacements, while hydrodynamic forces on particles are calculated based on Darcy's law with FEM. Other potential applications for the coupled model are finally presented.

Versatility is a key feature of ICY. The current interface allows virtually any partial differential equations (PDEs) and ordinary differential equations (ODEs) to be coupled with a discrete element simulation. The interface presented in this paper constitutes an important step toward the integration of multiscale modelling in porous media applications.

3.3 Methodology

3.3.1 COMSOL and YADE

COMSOL Multiphysics is a commercial finite element engine that can solve simultaneously a large range of preprogrammed partial differential equations (PDEs). COMSOL has two main interfaces for geoscience and geotechnical applications: the subsurface flow and geomechanics modules. The phenomena that can be modelled include fluid flow, seepage, chemical reactions, stress-strain behaviour and heat transfer. Custom partial differential equations (PDEs), ordinary differential equations (ODEs) and initial value problems can also be specified without programming, through a graphical user interface (GUI).

COMSOL models can be created through a graphical user interface. Each model is described by a model tree which includes a series of nodes that describe the model geometry, material properties, boundary conditions, PDE, solutions, etc. The information associated with these nodes can be accessed and modified by JAVA classes or MATLAB scripts.

The DEM code interfaced with COMSOL, YADE, is an open-source C++ framework with a Python script interface (Kozicki & Donzé, 2009; Šmilauer et al., 2015a). All computation parts (methods and algorithms) are programmed in C++ using object oriented models. This feature allows developers to add new algorithms and plug-ins. The Python interface is used to describe the model, to control the simulation and for post processing. YADE can be installed with Debian and Ubuntu Linux operating systems. With DEM, Newton's second law of motion (Force = mass × acceleration) is applied individually to each grain of a granular material (Cundall & Strack, 1979). Using the resultant forces on each particle, the velocity and position of each particle are calculated at the end of each time step. The changes in contact status (come into contact or lose contact) are automatically determined (O'Sullivan, 2014).

3.3.2 Coupling procedure

The interface between COMSOL (FEM) and YADE (DEM) involves a partially coupled framework. The algorithm is presented in Figure 3.1. The partially coupled approach solves the continuum and discrete element equations separately for each time step (Goodarzi et al., 2015). Results are exchanged between the two models at the end of each time step.

A JAVA class was used to control COMSOL. The JAVA interface was chosen because of its speed, the capability of being combined with Python code (the programming language used with YADE), and the fact that it does not require the MATLAB LiveLink module for COMSOL (COMSOL, 2016b). Through the JAVA class, command lines are sent to COMSOL to change the initial conditions for each time step, to set the parameter values received from YADE, and to run the simulation. The information sent to COMSOL varies for each application. Other JAVA classes are used in ICY. In the current interface, the algorithm includes two subclasses which are controlled by the main class.

The first subclass, named Clientcaller, was written to connect the interface to YADE and to supply the initial values of the required parameters and variables for YADE's interface via the client-server. As can be seen in Figure 3.1, the algorithm features a client-server between the JAVA class controlling COMSOL and the Python script controlling YADE. The server for the client-server connection should not be confused with the COMSOL server.

The client-server acts as an agent between the two interfaces. It saves computing time by allowing the YADE interface to run independently of the JAVA interface and COMSOL. Additionally, statistical functions (e.g., mean, median) can be applied on the YADE input data from COMSOL and on the YADE data from the previous time step directly, on the server. Hence, the Python interface script remains intact. The client and server tasks are as follow:

i. The client receives simulation information (e.g., iteration number) from the JAVA interface via a terminal in Linux.

ii. The client creates a TCP socket and sends the information to the server.

iii. The COMSOL and YADE model results (e.g., drag force and pressure, particle velocity from previous time step) are formatted by the client script as command line arguments to be sent to the Python script that controls the current YADE time step.

The second subclass, called Reader, was programmed to read and organise results files produced by YADE for COMSOL simulations. It reads the YADE result file including the porous media mechanical response (e.g., particle velocity, porosity or permeability values) and sends them to the main class. The main class then assigns these parameters in the COMSOL model. Eventually, the main class runs the COMSOL model and save the results in predefined files. More details about the two subclasses will be presented for the internal erosion example (Figure 3.6).

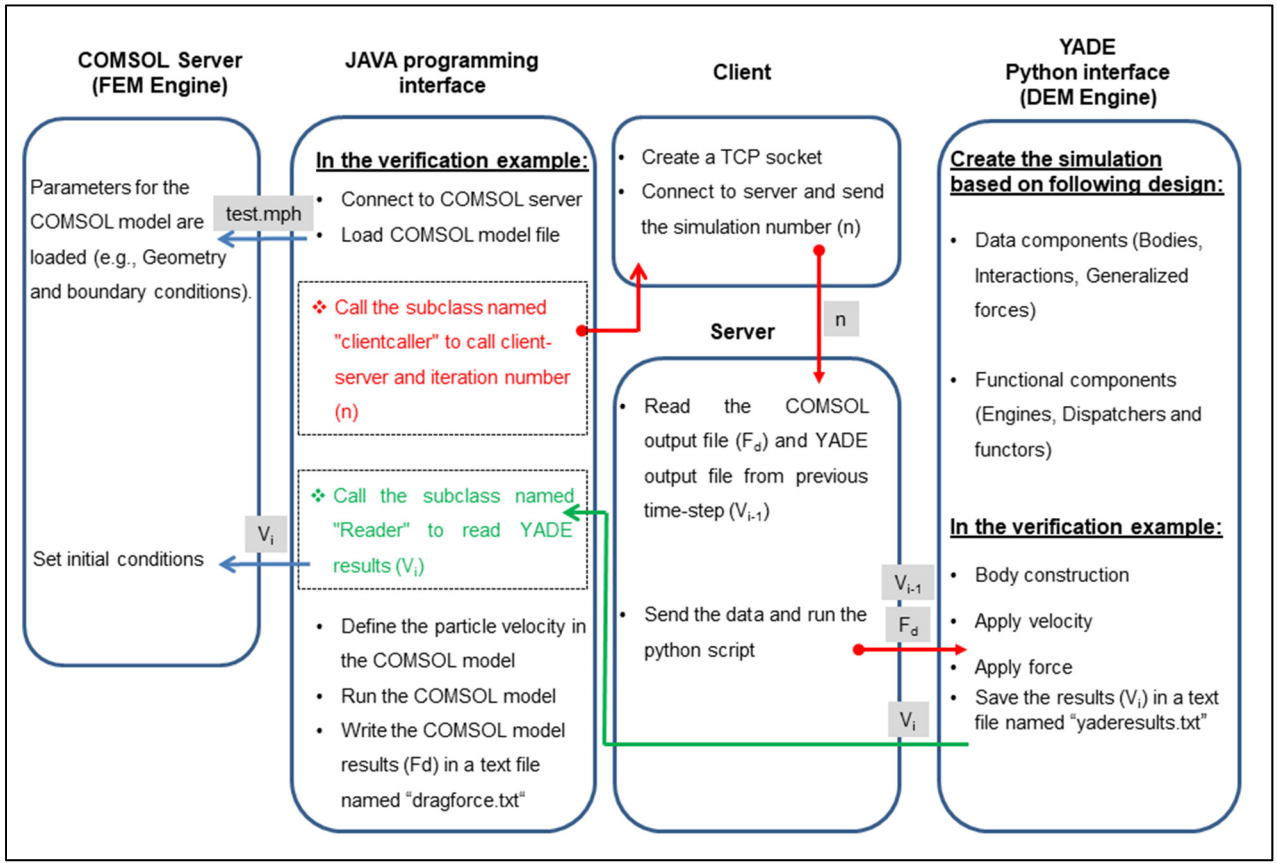


Figure 3.1 Schematic view of the ICY algorithm in the context of the verification example

3.4 Verification

The simple example of a sphere falling in a water column was chosen to demonstrate that data can be successfully exchanged between COMSOL and YADE with ICY. YADE was used to apply Newton's second law on the sphere (discrete element), while COMSOL was used to solve the fluid flow (velocity, pressure and density) around the sphere using the Navier-Stokes equations (continuum). The steady-state incompressible Navier-Stokes equations (neglect inertial term; Stokes flow) with no-slip boundary conditions on the sphere were solved in the COMSOL model.

When a particle falls into a fluid, it accelerates until it reaches a constant (terminal) velocity. In this example, the particle acceleration and terminal velocity were calculated using two methods. First, they were calculated using Stokes' law. Secondly, they were calculated by applying Newton's second law in YADE with a drag force calculated in COMSOL.

3.4.1 Fall velocity and Stokes' equation

Stokes' law expresses the settling velocity of a sphere falling through a viscous fluid. Stokes derived the forces acting on a sphere sinking in a viscous liquid under the effect of gravity. The drag force F_D is expressed as:

$$F_D = 3\pi\mu V d \tag{3.1}$$

where μ is the kinematic viscosity of fluid, V is the sphere velocity and d is the sphere diameter.

Combining equation (3.1) with other forces imposed on the sphere falling in a quiescent and viscous fluid gives the acceleration of the sphere and its terminal velocity. Three forces act on the sphere when it is dropped into a column of liquid: buoyancy (F_B), viscous drag (F_D) and weight (F_w) respectively. Buoyancy and viscous drag are directed upward. Weight is directed

downward. The sphere acceleration (a) and velocity can be calculated by combining these forces:

$$F_B = \left(\frac{\pi}{6} d^3\right) \gamma_w \quad (3.2)$$

$$F_w = \left(\frac{\pi}{6} d^3\right) \gamma_s \quad (3.3)$$

$$a = \frac{F_w - F_B - F_D}{m} \quad (3.4)$$

$$V_{i+1} = V_i + a\,\Delta t \quad (3.5)$$

where:

- m is the particle mass,
- γ_w is the specific weight of water,
- γ_s is the specific weight of the sphere,
- Δt is the time step interval between time t_i and t_{i+1},
- V_i is the particle velocity at t_i,
- V_{i+1} is velocity of the particle in t_{i+1}.

F_B and F_W remain constant while the sphere is falling. Drag force is the only time-dependent force. Since $V = 0$ at the beginning, drag force is initially equal to zero and acceleration is maximum. With time, the drag force increases until it reaches a constant value. From then onward, the particle falls at a constant rate called the terminal velocity.

3.4.2 YADE model

The falling particle model has two main parts. On the YADE side, a sphere with diameter 0.1 mm and zero initial velocity was created. The weight and buoyancy were constant during the simulation. The drag force was calculated in COMSOL. During each time step, COMSOL solved the Navier-Stokes equations based on the velocity received from the previous time step in the YADE model. Then the calculated drag force was sent to YADE by the JAVA interface and client-server to compute the acceleration and the velocity for the next time step. The time step was set to 0.001 s in YADE. The COMSOL model is solved in a steady-state condition. Therefore, the time step in YADE is the ICY or global time step. Note that for other applications, different time steps can be used for ICY (global time step), YADE and COMSOL. This will be the case in the application example. Particles density and gravity acceleration implemented in the YADE model were 2500 kg/m^3 and 9.806 m^2/s, respectively.

3.4.3 COMSOL model

A particle with the same diameter as in the YADE model was created. The particle velocity calculated in YADE was applied as a boundary condition on fluid velocity (inlet) in COMSOL by the JAVA interface. Drag force was calculated in COMSOL and sent back to YADE by the client-server for the next time step. In COMSOL, part of the model tree was defined through the GUI (geometry, materials, fluid properties, boundary conditions, and mesh). In the geometry node, a sphere with a radius of 0.05 mm was created in a box representing the mathematical domain (width, depth and length of 40 mm). These dimensions were sufficiently large to prevent wall effects. Laminar and incompressible flow with a reference pressure of 1 atm was applied at the outlet. The mesh size near the sphere was shown to have a large influence on drag force accuracy. After trying a wide range of element sizes, a maximum element size of 0.0078 mm was chosen with an element growth rate (size ratio for two contiguous elements) of 1.5. The results do not change with a smaller mesh size.

3.4.4 Verification results

Results from Stokes' law and the FEM-DEM model are compared in Figure 3.2. The velocity values for the COMSOL-YADE model are almost equal to those from Stokes' law. According to Stokes' law, it takes 0.007 s for the particle acceleration to decrease to almost zero. After 0.022 s, the particle moves with a velocity of 0.008989 m/s. The FEM-DEM results are almost identical. After 0.022 s, the particle velocity is 0.008990 m/s, a difference of 0.0027%. This shows that the drag force and particle velocity are correctly exchanged between COMSOL and YADE.

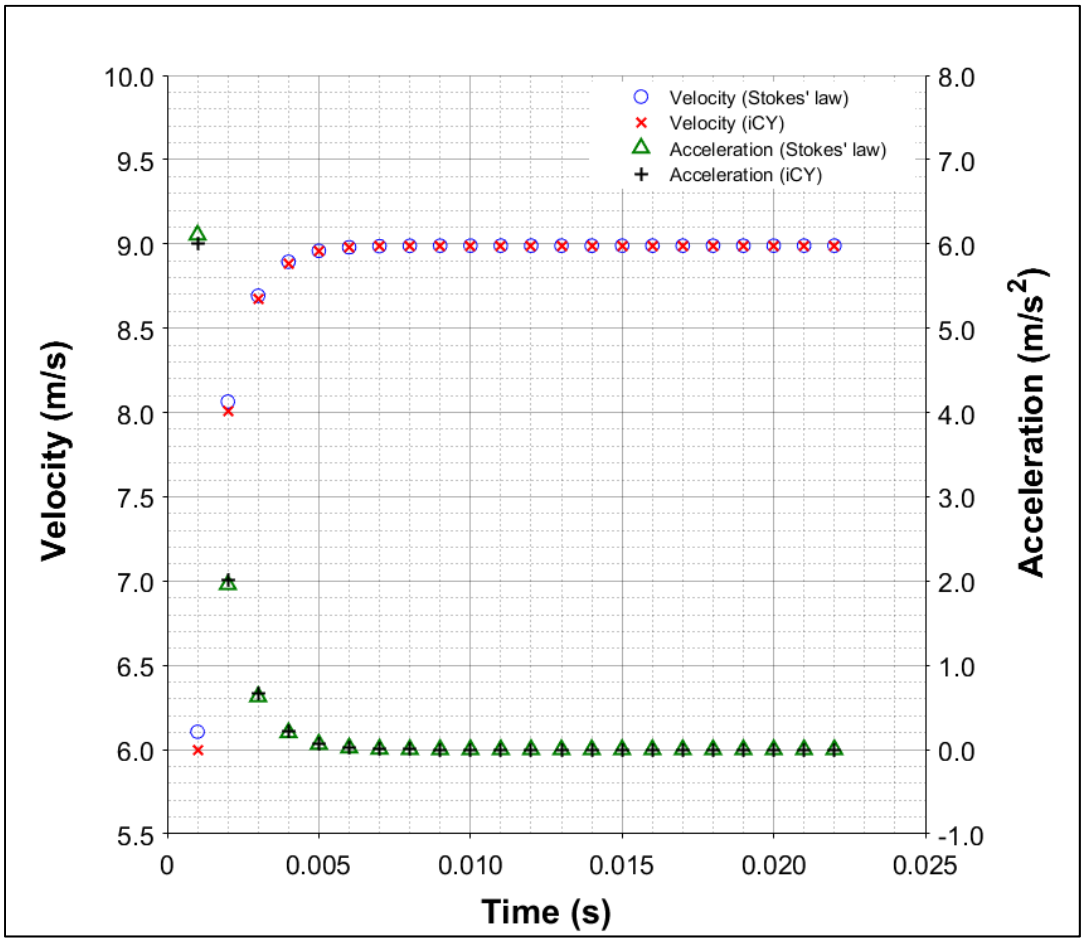


Figure 3.2 Comparison of terminal velocity and accelerations through Stokes' law and FEM-DEM model

3.5 Application example

In this section, the simulation of an internal erosion test first presented by Tomlinson & Vaid (2000) is used as an application example for ICY. This permeameter test was chosen because it involves two layers of monodisperse spherical glass beads and the specimen is relatively small. Thus it can easily be simulated in YADE. In this example, YADE was used to solve the motion and to calculate the contact forces and torques for a large number of particles by means of the DEM, while fluid flow was solved with COMSOL based on the FEM.

3.5.1 Apparatus, testing materials and procedure

The permeameter used by Tomlinson & Vaid (2000) is presented schematically in Figure 3.3. The specimen was composed of two layers of glass beads: a finer layer on top and a coarser layer at the bottom. The bottom and top layers had thicknesses of 3.7 and 1.9 cm respectively. Spherical glass beads with a uniform surface texture were used to exclude the influence of particle shape on the internal erosion results (Tomlinson & Vaid, 2000). The minimum round fraction was 70% and the glass density was 2500 kg/m^3. A test with 3-mm glass beads in the coarse layer and 0.346-mm glass beads in the fine layer (grain-size ratio of 8.7) under a confining pressure of 100 kPa was selected to be reproduced by the coupled model.

The cylindrical permeameter has an inside diameter and a height of 10 cm. The base pedestal has 5-mm holes to allow water and the glass beads to reach the sample collector. The specimen bottom is covered with a mesh with 1.5-mm openings. This mesh can retain the glass beads from the bottom layer inside the permeameter, but it allows water and the finer glass bead to flow out of the specimen.

At the beginning of the test, the confining stress applied by the top platen was gradually increased to 100 kPa over the course of several minutes. The specimen was then left under this

stress for one hour. Thereafter, the desired hydraulic gradient was applied to the specimen through a 5 mm hole in the top platen. A small hydraulic head difference of 2 cm was first applied to initiate flow in the specimen. Finally, the upstream hydraulic head was increased rapidly from 2 to 23 cm in 1 minute.

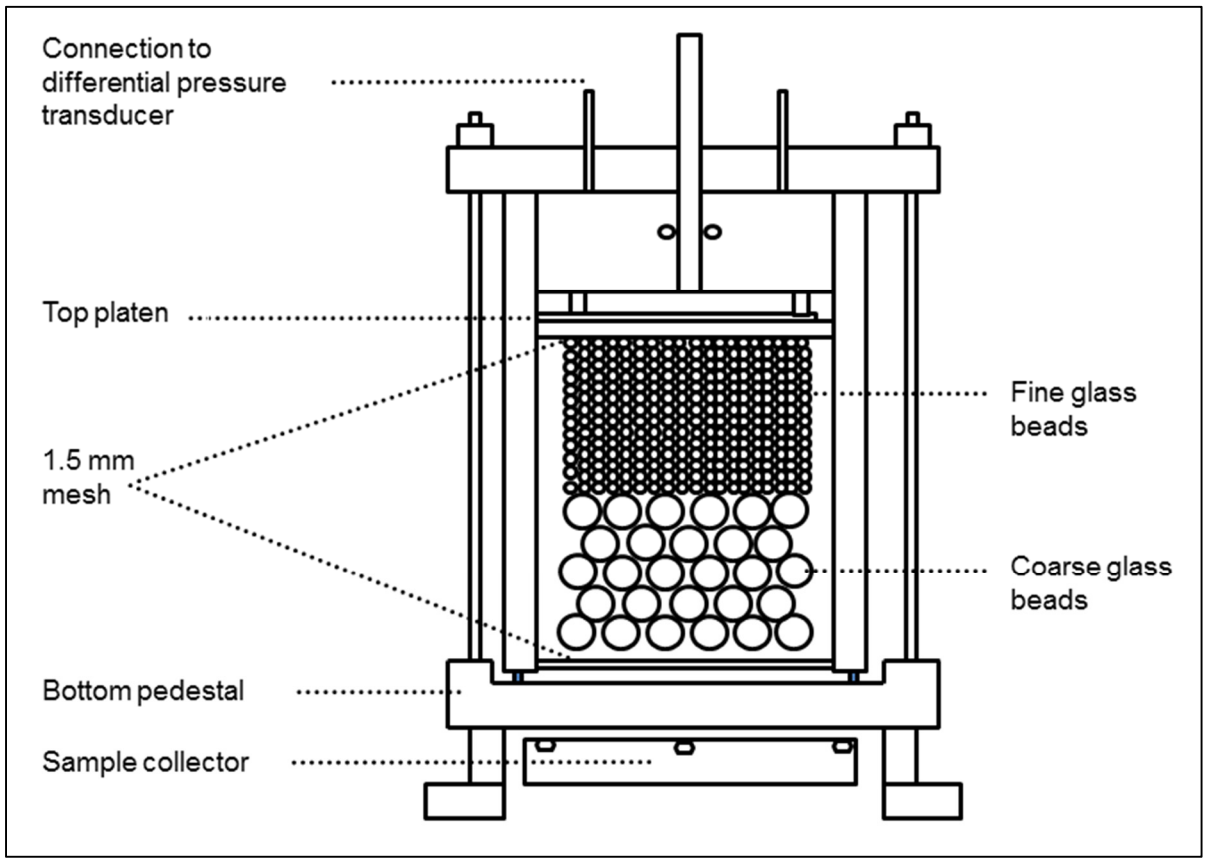


Figure 3.3 Schematic diagram of laboratory permeameter
Taken from Tomlinson and Vaid, 2000

3.5.2 Fluid-DEM coupling theory

For the permeameter test, the same body forces as in the verification example (weight, buoyancy and drag force) were applied to the discrete spheres. Drag force, the only variable force, was directed downward.

There are two main approaches for the computation of hydrodynamic forces. The first approach is the sub-particle scale method. In this method, the Navier-Stokes equations are solved at the pore scale with an appropriate computational fluid dynamics (CFD) method, for example the Lattice-Boltzmann method (LBM) (Lominé et al., 2013). This approach requires computational resources that are not readily available.

The second approach, the coarse-grid method, is less computationally intensive. It was proposed by Tsuji et al. (1993). In this method, the fluid cell embraces several particles. Fluid flow derives from average pressures and velocities in several pores in each cell.
The principal difference between the sub-particle and coarse-grid methods is how the porous media topology is represented. The microscale grain arrangement is not considered explicitly for coarse-grid methods. The frictional losses are calculated based on Darcy's law and macroscale permeability values. With sub-particle methods, frictional losses are calculated by solving the Navier-Stokes equations at the microscale. The grain arrangement from the DEM simulation is considered explicitly.

Goodarzi et al. (2015) developed a coarse-grid framework to model fluid-soil interaction. The fluid was modelled as a continuum on an Eulerian mesh. The equivalent drag force was calculated from the Ergun equation (Ergun, 1949). It was then applied to particles at the microscopic scale in the DEM simulation. In this paper, a coarse-grid method based on Darcy's law was applied to model the Tomlinson & Vaid (2000) experiment.

DEM and coarse-grid calculations of drag force with FEM are the two main components in the DEM-FEM model. These components have a feedback on each other. In the DEM model, the movement of particles is influenced by the drag force calculated with the coarse-grid method. The particle displacements have in return an influence on the permeability and the hydraulic gradient that are calculated with the coarse-grid method. The hydraulic gradients are assessed

in COMSOL for the whole permeameter by solving a water conservation equation based on Darcy's law:

$$\nabla.(K\nabla h) = 0 \tag{3.6}$$

where K is the hydraulic conductivity (m/s) and h is the hydraulic head (m).
Based on Darcy's law, the flow rate in a porous media is related to the hydraulic head difference and the porous media hydraulic conductivity or permeability:

$$v = K\frac{\Delta h}{L} \tag{3.7}$$

where:

- v is the Darcy velocity (m/s),
- L is the flow path length (m), and
- Δh is hydraulic head change (m) over length L.

The influence of drag force on the fluid results in a force acting in the direction opposite to fluid movement. From force equilibrium considerations (Figure 3.4), the drag force on particles (F_D) can be derived based on Darcy's law:

$$F_D = \Delta U.A \tag{3.8}$$

$$F_D = \Delta U.dx.dy \tag{3.9}$$

where $\Delta U = \Delta h.\gamma_w$ is the difference between the real pressure differential (ΔP) and the hydrostatic pressure differential ($dz.\gamma_w$). The hydraulic head (h) is the sum of the pressure head (P/γ_w) and the elevation head (z).

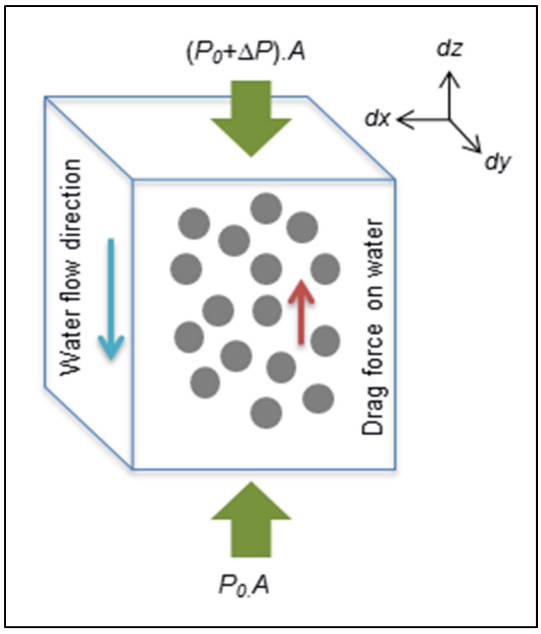


Figure 3.4 Effective forces on water
in a volume of porous media

The total drag force can be applied on each particle proportionally to their volume or surface (Zeghal & El Shamy, 2004). If it is applied proportionally to their volume, the drag force on each particle is given by:

$$F_{DPi} = \frac{F_D}{(1-n).V_T}.V_{Pi} \tag{3.10}$$

where:

- F_{DPi} is drag force on particle i,
- n is porosity,
- V_T is total volume of box, and
- V_{Pi} is the volume of particle i.

If the volume definition (dx dy dz) and equation (3.9) are substituted in equation (3.10), the drag force on each particle can be defined as:

$$F_{DPi} = \frac{\Delta U}{(1-n)dz} . V_{Pi} \tag{3.11}$$

3.5.3 Model implementation

The DEM specimen is presented in Figure 3.5a. Compared to the test set-up, the domain has a smaller horizontal section (1 cm × 1 cm) to reduce the total number of particles. The real height of the coarse-grained layer was used (3.7 cm). To further reduce the number of particles, the fine-grained layer thickness was halved. To compensate for the smaller number of fine particles, the eroded particles gathered in the bottom container were moved to the top of the fine-grained layer at the end of each global time step in the coupled COMSOL-YADE simulation (0.5 s).

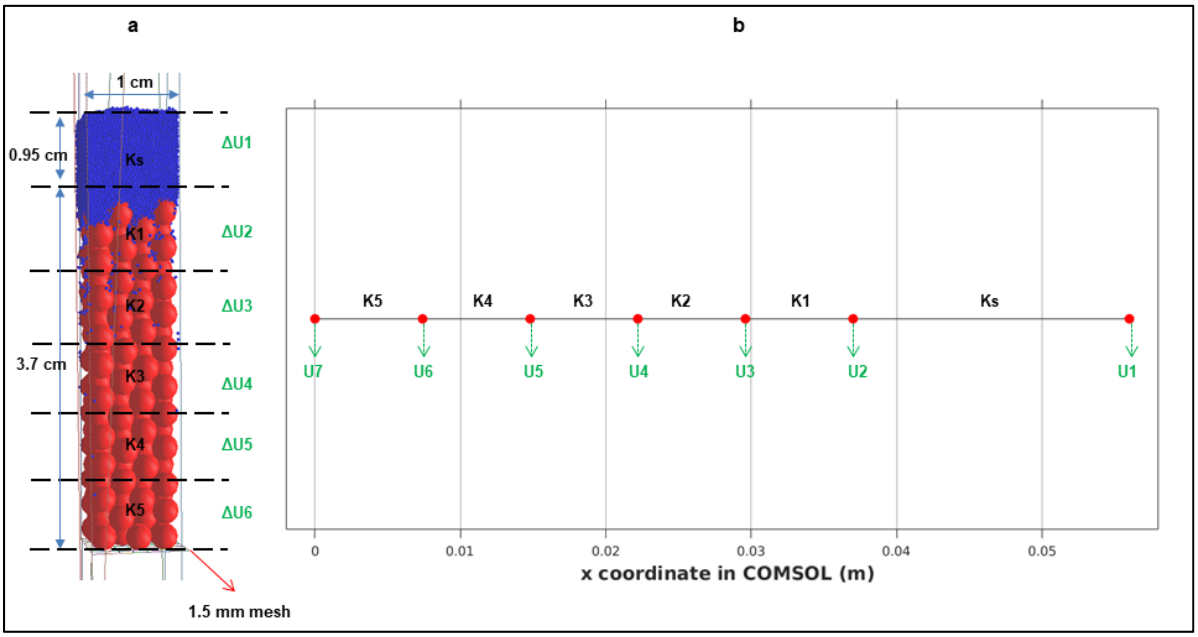


Figure 3.5 Model implementation in COMSOL and YADE. The x coordinate in COMSOL (b) represents the vertical axis in YADE (a)

A 2D mesh with 1.5 mm holes was produced by Gmsh, a finite element mesh generator (Geuzaine & Remacle, 2009). The mesh was located at the bottom of the coarse-grained layer (Figure 3.5a). It allowed the fine particles to reach the container below the mesh.

The DEM specimen contains 160 coarse particles and 25000 fine particles. The DEM specimen was compacted by a wall generated at the top of the fine particles layer. The wall moved downwards at a constant velocity (0.01 m/s). The compaction was stopped when porosity of the layer reached 0.5. After settlement and compaction of the fine-grained layer, a small portion of finer beads (0.17 g), approximately 3000 particles, fell through the specimen. These particles were removed from the container before subjecting the specimen to the hydraulic gradient. This initial segregation was also reported by Tomlinson & Vaid (2000) in the experimental tests. This mass was removed from the container as well. At the end of this stage, the specimen is ready to be submitted to the hydraulic gradient.

The YADE time step was determined based on the P-wave velocity in the spheres as calculated by the PWaveTimeStep function (Šmilauer et al., 2015b). The P-wave velocity is a function of the particles' density and Young's modulus (E). FrictPhys interactions were used for the contact model in YADE. This contact model is based on the classical linear elastic-plastic law of Cundall & Strack (1979).

To have a longer time-step compared to PwaveTimeStep function, the density scaling technique presented by O'Sullivan (2015) was used in this study. The particles' density was multiplied by 100. This increases particle weight by a factor of 100. The buoyancy (equation 3.2) and drag force (equation 3.12) were also multiplied by 100 to maintain the same proportions between forces.

The Young's modulus and Poisson ratio of the particles were set to 0.01 GPa and 0.3 respectively. A small Young's modulus value was assigned to decrease the P-wave velocity and to increase the maximum stable time step as a result. The damping coefficient and friction

angle (φ) were found to be the most influential parameters in this simulation. The damping coefficient dissipates kinetic energy at the particle contacts. A wide range of damping coefficients and friction angles were tested for the YADE model (see application results and discussion for a comparison). The coupled model shows results that are similar to the experimental results with a damping coefficient of 0.4 and a friction angle of 17.19 degrees.

According to Tomlinson & Vaid (2000), confining pressure has a negligible effect on the stability of finer beads, especially for the particle size ratio of 8.7 used in this example. A similar observation made with the numerical model during preliminary tests. Therefore, the 100 kPa confining pressure was not taken into account in the numerical model. Particles density was set to 2500 kg/m^3 in the YADE model. Fluid density and viscosity were set to 1000 kg/m^3 and 0.001 Pa.s in the COMSOL model, respectively.

The COMSOL component of the coupled model was used to calculate the hydraulic gradient and the drag force. It consists in a 1-D domain representing the real thicknesses of the two layers (3.7 and 1.9 cm, Figure 3.5). Based on equation (3.11), the drag force on each particle depends on the hydraulic gradient, porosity and particle volume. The coarse-grained layer was divided into 5 sections to calculate 5 average drag forces for each time step (Figure 3.5a). The highest number of cells that could be used was 5 because of the filter layer thickness (3.7 cm) and the average particle size ((0.3+0.0346)/2 = 0.167 cm). When dividing the filter layer in 5 cells, the thickness of each cell is 0.74 cm. This results in a ratio between cell and average particle size of 4.5. According to O'Sullivan (2014), cell dimensions should be 5 to 10 times larger than the average particle size. The model results with two cells are also compared to those with five cells to verify the sensitivity of the model with respect to the number of cells. The delay might be due to a smoothing effect of the pressure gradient that results in a smoothing of drag forces. Therefore, the number of cells is an important parameter that needs to be chosen carefully.

The hydraulic conductivity for the 5 filter sections were defined as parameters that were modified based on the YADE results as explained in the next section. The hydraulic conductivity values were assigned to the centre of the five cells in the COMSOL model. The hydraulic conductivity (K) in the water conservation equation (equation 3.6) was defined as a linear interpolation of the K values for the five sections. The permeability of the fine-grained layer was set to 0.00134 m/s. It was evaluated with the Kozeny-Carman equation (Chapuis & Aubertin, 2003) considering the porosity of the layer of fine particles as 0.37. The hydraulic head at the top of the fine-grained layer was set to 23 cm as in the experiment.

3.5.4 Calculation sequence

Figure 3.6 illustrates the sequences of calculation used for this simulation. For each global time step, the YADE simulation was first conducted. The hydraulic conductivity of each layer $(K_1, K_2, K_3, K_4, K_5)$ was predicted based on the new particle distribution and the Kozeny-Carman equation. The porosity and specific surface (total grain surface divided by total grain mass) of each layer were calculated in YADE.

The drag force equation programmed in YADE's Python interface requires the average pressure differential (e.g., $\Delta U_2 = U_2 - U_3$) in each cell. The pressure values at the cells' top and bottom boundaries ($U_1, U_2, U_3, U_4, U_5, U_6, U_7$) were calculated in the COMSOL model based on the previously mentioned hydraulic conductivity values (Figure 3.5b). After every global time step in COMSOL, pressures at the subdomains' boundaries are saved in a text file. This file is read by the client-server at the beginning of global time step in YADE to supply 5 average pressure differentials to the Python interface. Based on the pressure gradients, the applied drag forces are updated at the beginning of each new global time steps in YADE. It should be apparent that the COMSOL model is solved as a steady-state (stationary problem).

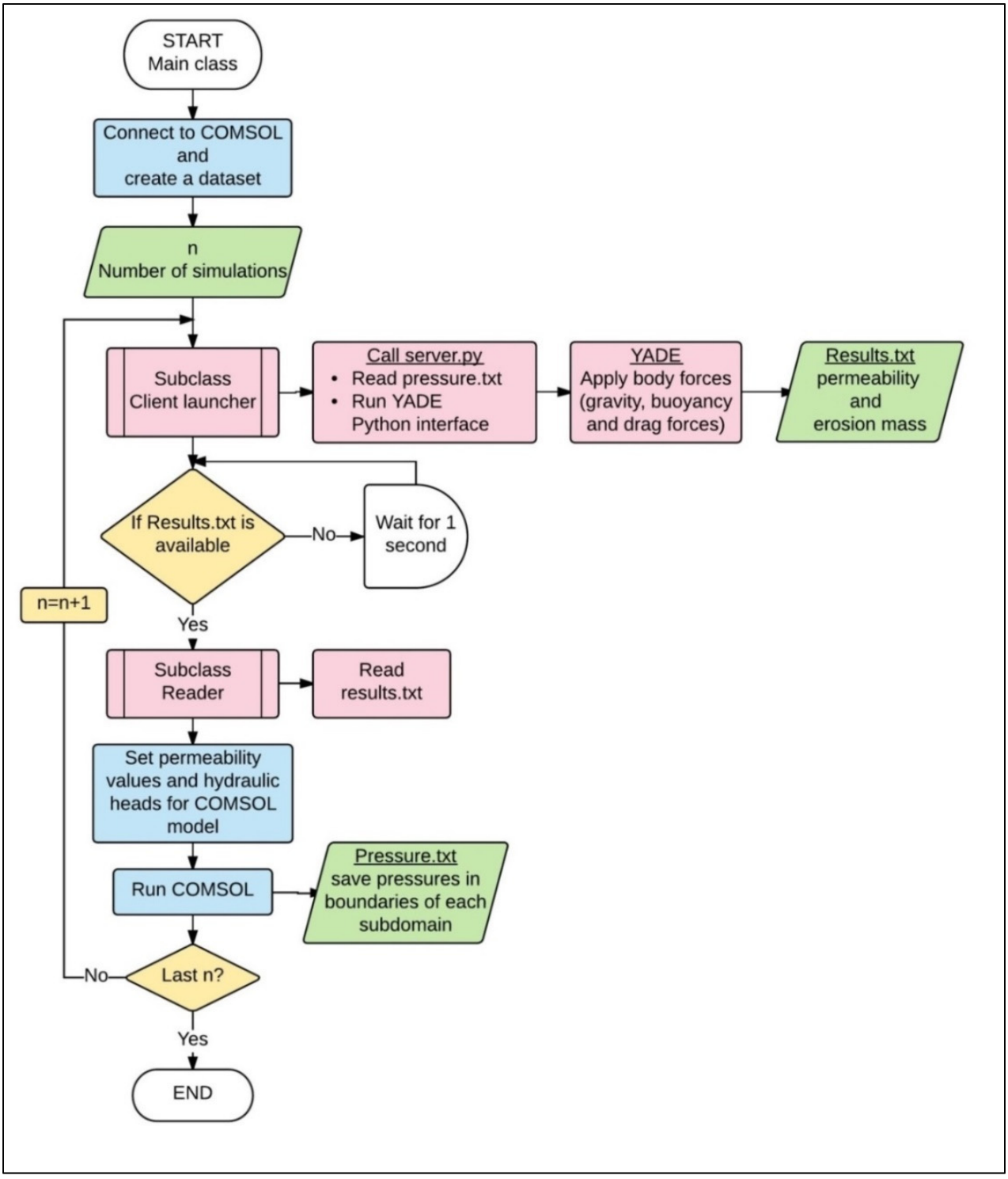


Figure 3.6 Calculation sequence in FEM-DEM simulation of internal erosion

3.6 Application results and discussion

According to Tomlinson & Vaid (2000), the base layer was all eroded in 45 s after the application of the hydraulic head difference of 23 cm for the modelled experiment (Figure 3.7). The mass of eroded particles reported by Tomlinson & Vaid (2000) was 190 g. This corresponds to 2.41 g for a numerical specimen with a 1 x 1 cm section. The coupled model (five fluid cells) with time-steps of 0.5 s, friction angle 17.2 degrees and damping 0.4 in YADE resulted in the complete erosion (2. 41 g) of the base layer in 48 s (circle markers). As shown in Figure 3.7, the coupled model results are dependent on friction angle and damping coefficients in the YADE model. The results also indicate that the number of fluid cells for the coarse-grid approach can influence erosion. A model with the same parameters, but two fluid cells in YADE, reached the same total erosion 10 seconds later (plus markers). It could stem from a smoothing effect of the pressure gradient resulting in a smoothing of drag forces. Therefore, number of cells is an effective parameter that needs to be chosen regarding the case study.

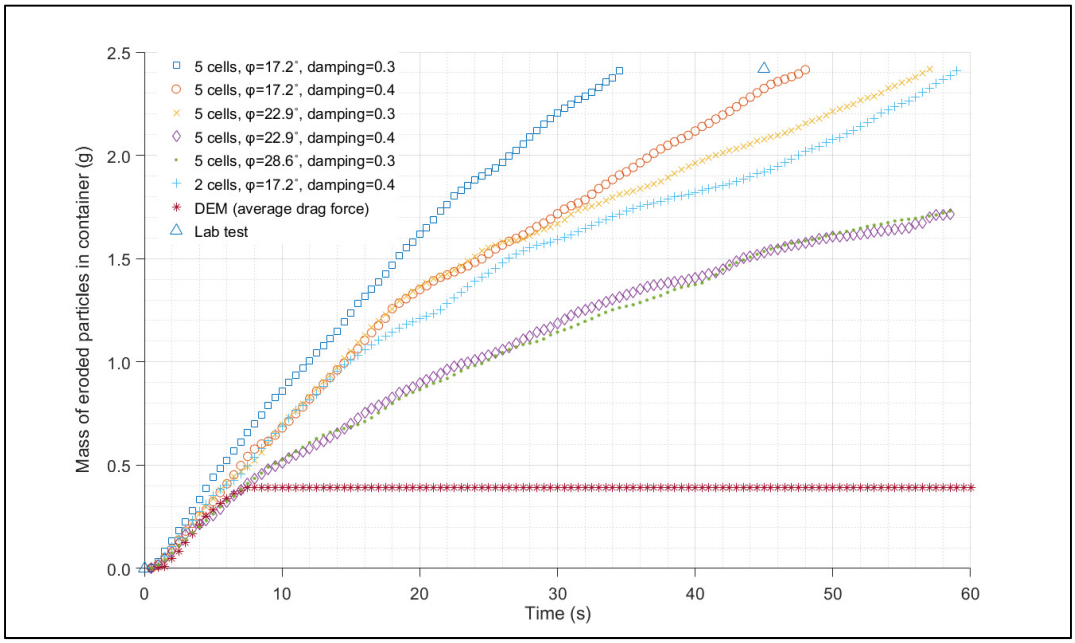


Figure 3.7 Eroded mass for the experimental test, FEM-DEM model with different parameters and DEM under constant drag force

The same test was also simulated exclusively with YADE with a constant drag force corresponding to the initial hydraulic gradient in the coarse-grained layer. In this case, erosion was stopped after 7 seconds with 0.39 g of eroded particles in the container.

A comparison of the FEM-DEM results with the experimental results confirms that using Darcy's law and a continuum model to calculate pressure gradients and drag force for the modelling of internal erosion in granular material can give realistic results. The main part of drag force calculation is done in YADE with a negligible computational cost.

Modelling the same test but under a constant average drag force in YADE reveals the necessity of using a multiscale approach in the modelling of internal erosion. Piping was also stopped after a few seconds under larger but still constant hydraulic head difference (up to 100 cm). The main reason is that finer particles are trapped gradually in the coarse-grained layer. This clogs the coarse-grained layer and eventually stops erosion. In reality, the migration of finer particles to empty spaces in the coarse-grained layer gradually raises the pore pressure and the drag force, thus limiting clogging. This process is considered in the FEM-DEM computational cycle.

3.7 Conclusions

This paper introduced ICY, an interface between COMSOL, a FEM engine and YADE, a DEM code. The interface is based on a series of JAVA classes. The interface was verified with the simple example of a sphere falling in water according to Stokes' law. In this test, the particle motion was simulated using YADE. Drag force on the particle was calculated by solving the Navier-Stokes equations in COMSOL. Comparison between simulation and analytical results showed that the framework could accurately replicate the results obtained from Stokes' law. The coupled model was then applied to reproduce a laboratory erosion test with drag force calculated with a coarse-grid method. The numerical results were in good agreement with the experimental results. The coarse-grid method can be substituted for pore-scale approaches with

a higher computational cost such as LBM and pore network flow methods in case studies with very large number of particles.

Regardless of accuracy of FEM-DEM results, the main objective of this study is developing a versatile interface between DEM and FEM models. A great number of applications in geoscience could benefit from multiscale models that consider both the particle and continuum scales. Multiscale FEM-DEM models could be used for instance in the study of mineral industry applications (e.g., granular material segregation and sedimentation), geotechnical applications (e.g., internal erosion and fluidized bed) and energy extraction (e.g., sand production problem).

The coupled model might be used to simulate fluid-particles interaction for large-scale applications in soil mechanics and geosciences in future. A multiscale scheme based on ICY is already under development to simulate internal erosion tests of a large permeameter.

Computer code availability

Hardware required: recommended 3GHz or more, 8 cores. **Software required:** YADE, COMSOL Multiphysics, JAVA integrated development environment (IDE). **Program language:** Java, Python. **Program size:** 28 MB. **Supplemental file:** ICY instruction guide. **The source code and supplemental file are available at:** https://github.com/pouyanpirnia/ICY-2018.

CHAPTER 4

HIERARCHICAL MULTISCALE NUMERICAL MODELLING OF INTERNAL EROSION WITH DISCRETE AND FINITE ELEMENTS

Pouyan Pirnia [a], Francois Duhaime [b], Yannic Ethier [c], Jean-Sebastien Dube [d]

[a, b, c, d] Department of Construction Engineering, École de technologie supérieure, 1100, rue Notre-Dame Ouest, Montréal, Québec, Canada, H3C 1K3

Paper submitted for publication, February 2019

4.1 Abstract

This paper presents a coupled finite and discrete element model (FEM and DEM) to simulate internal erosion. The model is based on ICY, an interface between COMSOL, an FEM engine, and YADE, a DEM code. With this model, smaller DEM subdomains are generated to simulate particle displacements at the gain scale. Particles in these small subdomains are subjected to buoyancy, gravity, drag and contact forces for small time steps (0.05 second). The DEM subdomains provide the macroscale (continuum) model with a particle flux distribution. Through a mass conservation equation, the flux distribution allows changes in porosity, hydraulic conductivity and hydraulic gradient to be evaluated for longer time steps (up to 0.5 second) and at a larger, continuum scale. The updated hydraulic gradients from the continuum model provide the DEM subdomains with updated hydrodynamic forces based on a coarse-grid method. The number of particles in the DEM subdomains is also updated based on the new porosity distribution. The multiscale model was verified with the simulation of suffusion. Results for the proposed multiscale model were generally consistent with results based on a DEM model incorporating the full sample and simulation duration. The multiscale algorithm could enable the modelling of internal erosion for larger structures (e.g. dams) and the study

of erosion law and homogenization techniques for the modelling of internal erosion with continuum methods.

4.2 Introduction

Internal erosion can be defined as the seepage-induced erosion of soil particles through the pore network of a soil or through larger openings or conduits. It can result in serious damage for water-retaining structures such as embankment dams and levees (Foster et al., 2000). Internal erosion includes four distinct mechanisms: regressive erosion, erosion along a concentrated leak, interfacial erosion, and suffusion. This paper is centred on the numerical modelling of suffusion, the erosion of small particles through a coarser granular skeleton (ICOLD, 2017).

The design of a new dam does not normally require the numerical modelling of internal erosion as robust design criteria are available in the literature (ICOLD, 2017). However, older water retaining structures do not always satisfy these criteria. Dam safety studies could thus benefit from efficient methods for the numerical modelling of internal erosion. The methods that are currently available can be classified into two main groups. Continuum methods combine particle and water conservation equations with phenomenological erosion laws (e.g. Vardoulakis et al. 1996), whereas discontinuum methods consider the behaviour of individual soil particles explicitly through the discrete element method (DEM) (e.g. Lominé et al. 2013).

Continuum methods are routinely employed in geotechnical engineering to model seepage and stress-strain relationships based on phenomenological constitutive models. While less common, continuum methods are also available for the modelling of internal erosion. These methods are centred on particle mass exchanges between fluid and solid phases. Vardoulakis et al. (1996; 2001) proposed a model of sand erosion for radial or axial flow conditions based on mass balance equations for water and suspended solids. The mass generation term in the conservation equation for suspended solid is associated with a phenomenological erosion law

based on filtration theory and attributed to Sakthivadivel (1966). According to this law, the increase in suspended solid mass due to erosion is proportional to the flow rate, the concentration of suspended solids, the volume fraction of solids and an empirical coefficient (Vardoulakis et al. 1996). Modified erosion laws were also presented by Steeb et al. (2005; 2007) to take into account non-erodible fines and the rate law of Wan and Fell (2004). The main drawback of these continuum methods is the current lack of validation with experimental data, especially in the context of real geotechnical applications.

Discontinuum methods based on the discrete element method (DEM) do not require a phenomenological erosion law to model internal erosion. With DEM, finite particle displacements are calculated based on Newton's second law of motion (Cundall & Strack, 1979; O'Sullivan, 2014). Particle-particle and fluid-particle interactions are modelled through contact laws and hydrodynamic forces. Internal erosion models based on DEM differ principally based on the methods used to apply the seepage (drag) force on each particle.

Coarse-grid methods apply a drag force on each particle based on a macroscale head loss calculated through Ergun's equation (Tsuji et al., 1993; Zeghal & El Shamy, 2004) or Darcy's law and the Kozeny-Carman relationship (Pirnia et al. 2019). The drag force can be applied on particles of different sizes proportionally to their volume or surface. Examples were presented by Zhang et al. (2019) and Zeghal & El Shamy (2004) for the modelling of suffusion in a gap-graded silty sand and liquefaction, respectively. The commercial DEM code PFC3D was used for both examples (Itasca, 2004). The main advantage of coarse-grid methods is their computational efficiency. Their main disadvantage is that they fail to resolve the variability of drag force values at the pore scale for small particles in a coarse-grained skeleton.

More accurate drag force values can be obtained by solving the Navier-Stokes equations at the pore scale. This can be done with different numerical methods, the lattice Boltzmann method (LBM) currently being the most common (Galindo-Torres, 2013). This method allows a regular grid to be used and the conservation of linear and angular momentum to be verified.

Several DEM-LBM coupling examples are available in the literature. For example, Lominé et al. (2013) simulated fluid-particle interactions with DEM-LBM for a hole erosion test (Wan & Fell, 2002). Galindo-Torres et al. (2015) used a DEM-LBM coupling to study contact erosion at the interface between two monodisperse layers with contrasting particle sizes. Wang et al. (2018) developed a 3D bonded DEM-LBM model to resolve fluid-particle interactions in cohesive materials.

Intermediate methods have also been developed to provide information on the local drag force variability without solving the Navier-Stokes equations around each particle. The pore scale finite volume formulation (PFV) decomposes the pore network using a triangulation method and Voronoi graphs (Chareyre et al., 2012). The flow rate in the pore network is then assumed to be proportional to the pressure gradient and a local conductance value function of the pore throat geometry. This method was shown to replicate the drag force values obtained by solving the Navier-Stokes equation at the pore scale for assemblies of 8 to 200 spheres. PFV was also used by Wautier et al. (2018) for the modelling of suffusion in well-graded granular materials. A semi-resolved DEM-CFD model was developed by Cheng et al. (2018) to analyze fine particle migration through a gap-graded soil consisting of fine and coarse particles. The semi-resolved model takes advantage of both coarse-grid and pore-scale methods. On the one hand, the flow around the coarse particles is fully resolved using the finite volume method and a mesh which is finer than the particle size. On the other hand, the drag force on the fine particles is solved based on locally averaged Navier-Stokes equations over a mesh that is several times larger than the fine particles.

The previous survey of numerical modelling methods that have been applied to internal erosion shows that modellers are currently facing a choice between continuum models that have seen relatively little experimental validation or microscale methods based on computationally costly DEM models. Even with methods that avoid solving fully resolved Navier-Stokes equations at the pore scale, such as the coarse-grid (Tsuji et al., 1993), PFV or semi-resolved methods

(Chareyre et al., 2012), the number of particles involved in DEM simulations is too large to model large structures, such as embankment dams.

The hierarchical or multiscale modelling methods that have recently been developed in geomechanics show one potential avenue to take advantage of DEM simulations in a continuum framework. Hierarchical multiscale approaches combine two length scales, most often using the finite element method (FEM) and DEM (Andrade et al., 2011; Dang & Meguid, 2013; Guo & Zhao, 2014 and 2016; Wang & Sun, 2016). The FEM is used to discretize the whole domain and solve the governing equations for the boundary value problem, while DEM simulations provide the local material responses at the Gauss points of the FEM mesh. The objective is to bypass phenomenological constitutive laws for the FEM (e.g. linear-elastic stress-strain relationship) and overcome the DEM limitation regarding the number of discrete bodies.

A few examples of hierarchical multiscale model for geomechanical problems have been developed. Andrade et al. (2011) studied a strain localization problem based on a simple plasticity model at the macroscale. The continuum model extracted its plasticity parameters directly from DEM simulations. Guo & Zhao (2004) developed a hierarchical multiscale framework in which a large scale continuum model based on FEM was coupled with small scale DEM simulations at the FEM gauss points. The geometric strain tensors from the FEM model is applied to microscale DEM simulations corresponding to representative volume elements (RVE) with periodic boundary conditions. Stiffness tensors are sent back to the FEM model to update the strain tensors. The work of Guo & Zhao (2004) on dry porous media was later extended to saturated porous media (Guo & Zhao, 2016; Wang & Sun, 2016). The relationship between the effective stress and strain tensors is determined with the same method, but seepage is also modelled at the macroscale with a water conservation equation and Darcy's law. Laminar flow and a negligible drag force on the particle in the DEM simulations is assumed. The applicability of hierarchical multiscale framework is currently limited to the mechanical behaviour of porous media.

Pirnia et al. (2019) simulated internal erosion during a permeameter test using ICY, a multipurpose interface that allows data to be exchanged between continuum models based on FEM (COMSOL) and particle scale models based on DEM (YADE) (Pirnia et al., 2016; 2019; Tomlinson & Vaid, 2000). The YADE model was divided in five cells in which discrete drag force values were applied on the particles based on a coarse-grid method. For each time step, the COMSOL model solved Darcy's law based on permeability values calculated in YADE from the porosity and grain size distribution in each cell. The cell permeability values were estimated based on the Kozeny-Carman equation (Chapuis & Aubertin, 2003). The model results compared favorably with the experimental results. However, the model suffered from the same shortcoming as the microscale models that were presented in the previous paragraphs: it was limited to a small laboratory specimen because of the heavy computational load associated with the large number of particles.

This paper presents a hierarchical FEM-DEM model based on ICY. The model is aimed at simulating internal erosion for large-scale applications (e.g. dams) without a phenomenological erosion law, without generating all particles and considering the total simulation duration in the discrete model. With this method, RVE subdomains use the DEM to calculate particle flux values for short time steps. These particle flux values are sent to a continuum model that predicts the porosity and drag force values of the subdomains for longer time steps through mass conservation equations for water and particles. The paper begins by introducing the algorithm and its implementation in ICY. The model is later validated by comparing its result to the microscale model of Pirnia et al. (2019). To the best of our knowledge, this paper presents the first hierarchical FEM-DEM model for the modelling of internal erosion based on continuum conservation equations for water and particles, and microscale particle flux calculations.

4.3 Methodology

4.3.1 ICY

ICY is an interface between COMSOL Multiphysics and YADE (Pirnia et al., 2019). YADE ("Yet Another Dynamical Engine") is an open-source discrete-element code (Kozicki & Donzé, 2009; Šmilauer et al., 2015). Simulations in YADE are described and controlled by a Python interface. COMSOL Multiphysics (COMSOL, 2016) is a commercial finite element engine. It can treat simultaneously multiple physical phenomena, such as fluid dynamics, seepage, chemical reactions, stress-strain behaviour and heat transfer. ICY is programmed with a Java class. The interface was verified using the example of a sphere falling in water according to Stokes' law (Pirnia et al., 2016; 2019). ICY allows virtually any partial differential equations (PDEs) to be coupled with a discrete element simulation.

4.3.2 Hierarchical multiscale FEM-DEM model

The hierarchical FEM-DEM model was developed to simulate suffusion in the specimen shown in Figure 4.1. The specimen is composed of spheres with two distinct diameters. The finer particles are initially dispersed throughout a coarser-grained skeleton which remains fixed during the simulation.

The main idea behind this hierarchical model is to solve fluid flow and particle conservation in the porous media at the continuum scale with FEM and to calculate particle flux values based on particle displacements in small subdomains along the specimen with DEM. In other words, the DEM provides the FEM with particle fluxes at specific nodes along the geometry. The DEM and FEM calculations have a feedback on each other. The initial small particle distribution in each DEM subdomain for each time step, and the drag force on each particle are respectively controlled by the particle and water conservation equations. The particle displacements have in return an influence on the particle conservation equation through the

flux computations. The DEM and FEM sides of the calculation cycle are described in the following sections.

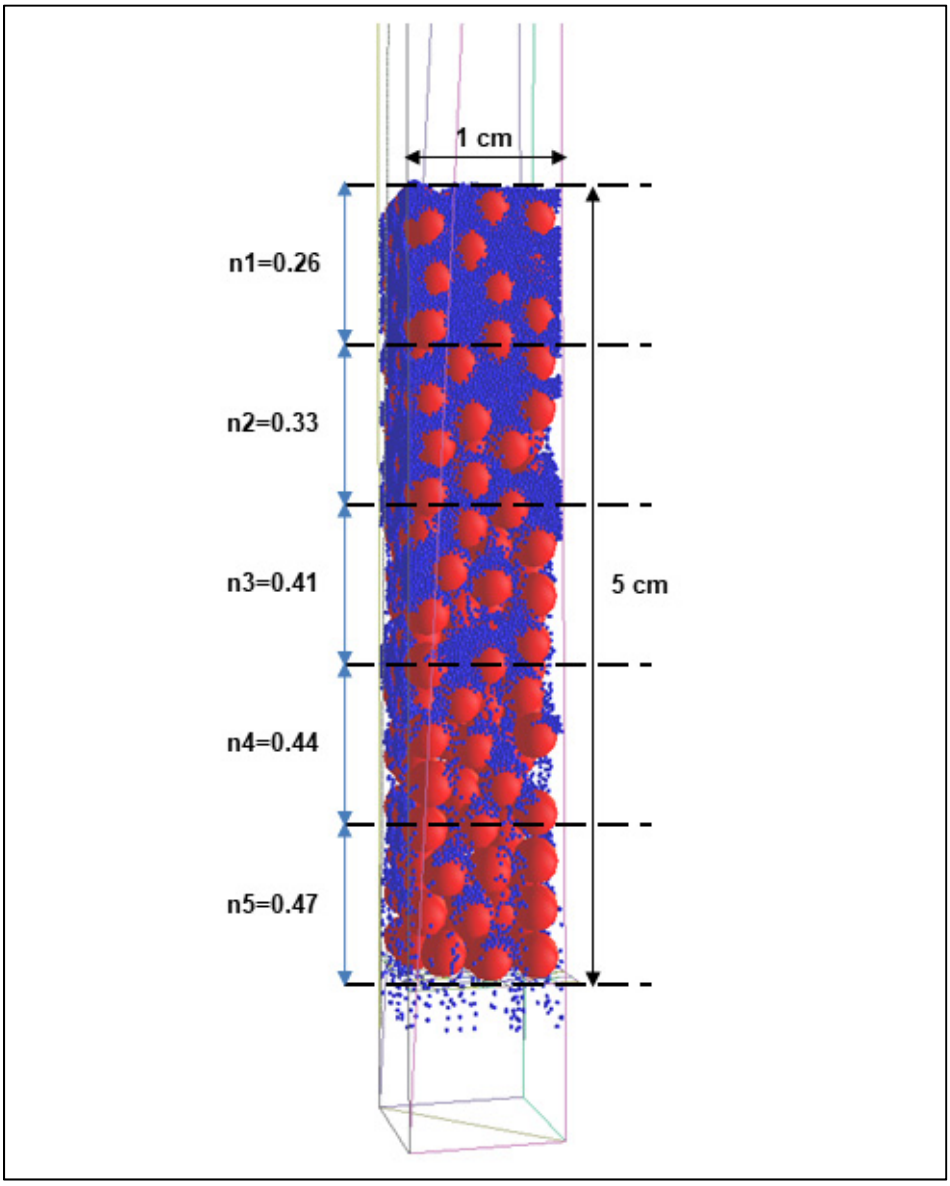


Figure 4.1 Representation of the numerical specimen for the modelling of suffusion

4.3.2.1 DEM

Particles in the subdomains are subjected to contact forces and body forces such as gravity, buoyancy and drag force. Drag force is the only variable body force. It acts on the particles in the direction of fluid movement and on the fluid in the opposite direction (Figure 4.2). In order to minimize computational costs, drag force values were derived based on a coarse-grid approach and Darcy's law using a macroscopic hydraulic gradient calculated with the FEM. Darcy's law relates the Darcy or filtration velocity (v) to the hydraulic gradient (i) and the porous media hydraulic conductivity (K):

$$v = -K\,i \tag{4.1}$$

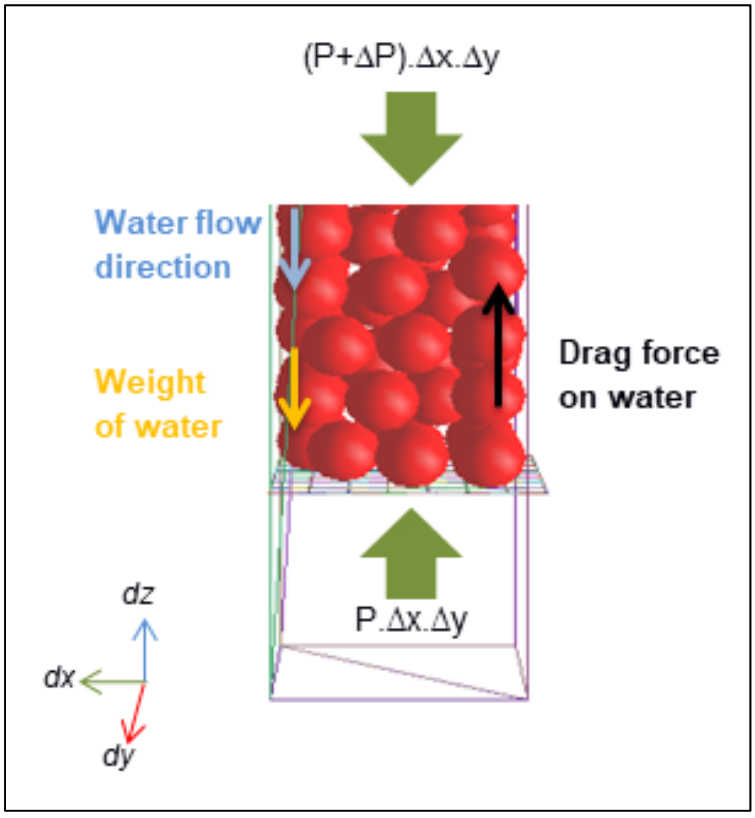


Figure 4.2 Effective forces on water in a DEM subdomain

For the one-dimensional suffusion test presented in Fig 1, $i = \partial h/\partial z$, where z is the elevation and h is the hydraulic head. The hydraulic head is the sum of the elevation head (z), and the pressure head (P/γ_w), where γ_w is the unit weight of water and P is the water pressure. Equation 3.1 can be reformulated by substituting the hydraulic head definition.

$$v = -\frac{K}{\gamma_w}\left(\frac{\partial P}{\partial z} + \gamma_w\right) \tag{4.2}$$

The total drag force (F_D) on the particles of a DEM subdomain can be derived from Darcy's law (equation 4.2) and force equilibrium considerations (Figure 4.2):

$$F_D = -\frac{\partial P}{\partial z}\Delta x \Delta y \Delta z - \gamma_w \Delta x \Delta y \Delta z \tag{4.3}$$

where Δx, Δy and Δz are the side lengths of a DEM subdomain. The first term on the right-hand side represents the force due to the pressure difference on both sides of the subdomain. The second force represents the weight of water. The drag force on each particle is obtained by distributing the total drag force (equation 4.3) among the particles proportionally to their volume (Pirnia et al., 2019; Zeghal & El Shamy, 2004; Tsuji et al., 1993):

$$F_{DPi} = -\frac{V_{Pi}\left(\frac{\partial P}{\partial z} + \gamma_w\right)}{(1-n)} \tag{4.4}$$

where F_{DPi} is the drag force on particle i, n is the subdomain porosity and V_{Pi} is the volume of particle i.

After each time step (Δt) in YADE, the volume flux of small particles per unit surface and time (f) in each subdomain is calculated from the vertical component of the mean velocity of the small particles (v_z):

$$f = v_z \, \phi \tag{4.5}$$

$$v_z = \frac{\sum z_i}{\Delta t \, n_s} \tag{4.6}$$

where ϕ is the proportion of small particles with respect to the subdomain volume, z_i is the vertical displacement of small particle i and n_s is the number of small particles in each subdomain.

4.3.2.2 FEM and computation cycle

The FEM side of the computational cycle for the hierarchical multiscale model is centred on two partial differential equations (PDE) that verify the conservation of water and particles, respectively. Fluid flow in the porous media is modelled by solving a water conservation PDE based on Darcy's law:

$$\frac{\partial}{\partial z}\left(K \frac{\partial h}{\partial z}\right) = 0 \tag{4.7}$$

Hydraulic conductivity values for equation (4.7) are estimated using the Kozeny-Carman equation (Chapuis & Aubertin, 2003):

$$K = \frac{A.n^3}{D_R^2.S_s^2.(1-n)^2} \tag{4.8}$$

Where D_R is the particle specific weight ($D_R = \gamma_s/\gamma_w$, where γ_s is the particle unit weight), A is an empirical factor that varies between 0.29-0.51 and S_s is the specific surface, the particle surface divided by the particle mass.

The second PDE is aimed at verifying particle conservation and calculating porosity changes along the specimen length for future time steps. It is derived from the difference in *f* values at the upper and lower surfaces of a RVE (Figure 4.3).

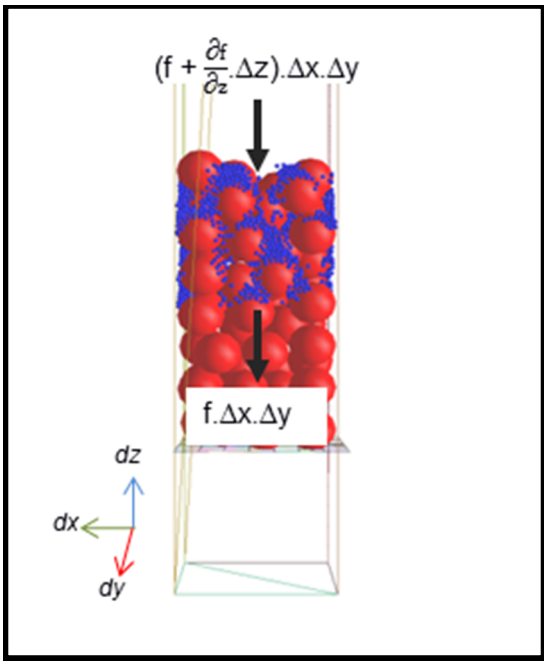


Figure 4.3 Change in small particle flux
for a porous media RVE

The difference between *f* at the upper and lower surfaces must be equal to the change in small particles stored in the box:

$$\frac{\partial f}{\partial z} = \frac{\partial \phi}{\partial t} \tag{4.9}$$

If the large particles are fixed, the change in ϕ is equal in magnitude but opposite to the porosity change:

$$\frac{\partial f}{\partial z} = -\frac{\partial n}{\partial t} \tag{4.10}$$

The computation cycle is presented schematically in Figure 4.4. Each main time step begins with a shorter DEM simulation in YADE to calculate a mean particle flux value for each subdomain with equations (4.5) and (4.6). The flux values are then sent to an interpolation function in COMSOL through the main ICY Java class. The interpolated flux distribution is assumed to be time-independent when solving equation (4.10) for the longer FEM time steps in COMSOL (Figure 4.4). New porosity values are predicted at the midpoint of each DEM subdomain at the end of the COMSOL time steps. These values are sent to the DEM model to update the porosity. For each subdomain, the predicted porosity is first compared with the value at the end of the previous time step. The porosity change is translated to a number of small particles to be added or removed randomly.

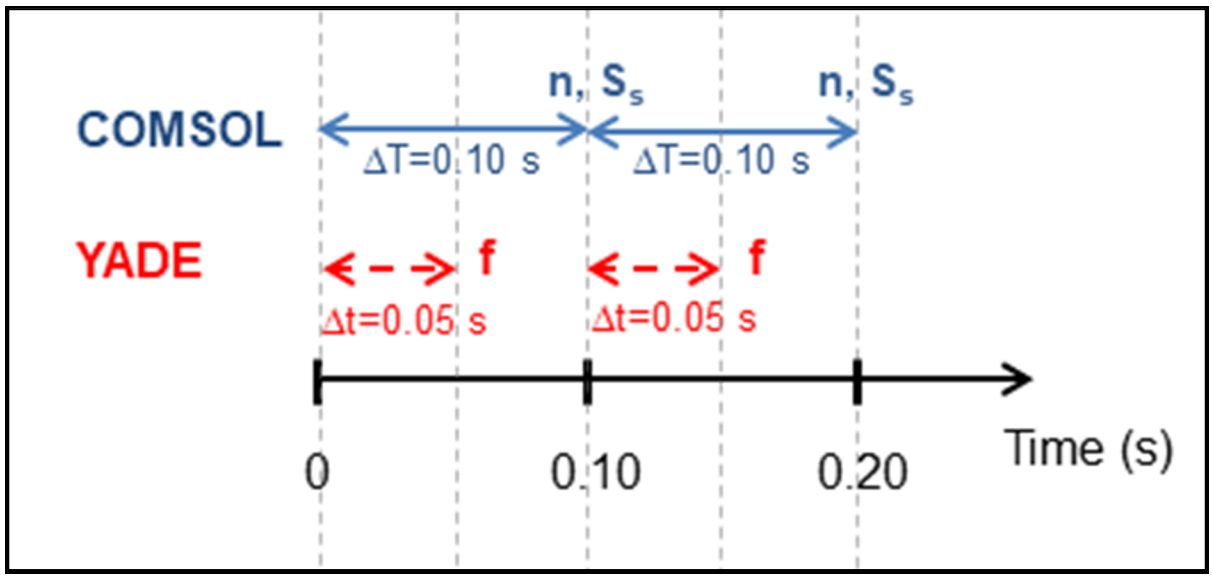


Figure 4.4 Example of time steps for COMSOL and YADE in the hierarchical multiscale model. Both water and particle conservation equations are solved for longer time steps in the COMSOL model. Flux values are computed for shorter time steps in YADE

The change in porosity value can be used to calculate new specific surface values at the end of each FEM time step:

$$S_s = \frac{S_{large} + S_{fine}}{V_{large} \cdot \rho_s + V_{fine} \cdot \rho_s} \tag{4.11}$$

$$S_s = \frac{1}{\rho_s}\left(\frac{\frac{3\Omega}{R} + \frac{3(1-n-\Omega)}{r}}{1-n}\right) \tag{4.12}$$

where:

- S_{large}: sum of large particle surface
- S_{fine}: sum of fine particle surface
- V_{large}: sum of large particle volume
- V_{fine}: sum of fine particle volume
- ρ_s: *density of particles*
- *R: large particle radius*
- *r: fine particle radius*
- Ω *: volume percentage of large particles*

The new porosity and specific surface distributions at the end of the COMSOL time step allow a new *K* profile to be computed through equation (4.8). New pressure values are calculated through equation (4.7) at the upper and lower surface of each DEM subdomain. These pressure values are used to update the drag force values in each subdomain (equation 4).

4.4 Validation example

Suffusion was modelled for the specimen shown in Figure 4.1 with two procedures to validate the hierarchical multiscale model. With the first procedure, the full specimen and simulation duration were modelled with a single DEM domain, without using the particle conservation

equation (equation 4.10). The second procedure is centred on the hierarchical multiscale model introduced in the previous section.

The same initial specimen was used for both procedures. The coarse-grained skeleton has a cross-section of 1 cm × 1 cm and a height of 5 cm. It is composed of spheres with a 3 mm diameter. Their position was fixed after settlement on a 2-D mesh with 1 mm openings. The mesh was produced with the finite element mesh generator Gmsh (Geuzaine & Remacle, 2009). The mesh is located above an empty container with a height of 1 cm. A cloud of 20 000 fine spherical particles with a diameter of 0.3 mm and random positions was generated in the pore space between the coarser particles. A constant hydraulic head difference of 5 cm was applied between both ends of the specimen.

The contact model and parameters used by Pirnia et al. (2019) for their ICY application example was used for both procedures. All particles have a density of 2500 kg/m^3 (i.e. glass beads). The Young's modulus and Poisson's ratio of the particles were set to 0.1 MPa and 0.3, respectively. A relatively small Young's modulus value was assigned to decrease the P-wave velocity and to increase the maximum stable DEM time step as a result (Šmilauer et al., 2015). A damping coefficient of 0.3 and a friction angle (φ) of 17.2 degrees were assigned for the model.

Both procedures consist of a global time stepping scheme and alternating simulations in YADE and COMSOL. Each YADE and COMSOL simulation has its own time stepping scheme based on the default adaptive time stepping in COMSOL and the P-wave velocity in YADE. The computational cycles for the full specimen and hierarchical multiscale procedures are detailed in the following sections.

4.4.1 Full specimen model implementation

The YADE model for the full specimen procedure is presented in Figure 4.5a. It is divided in 5 cells. For each global time step, a constant drag force is applied in each cell. At the end of the YADE simulation, new n and S_s values are calculated for each cell. Using equation (4.8), a new K value is calculated for each cell and sent to the COMSOL model.

The COMSOL model solves equation (4.7) to calculate the drag force values to be applied in YADE for the next global time step. The 1-D COMSOL model represents the total thickness of the specimen (Figure 4.5b). The hydraulic conductivity values obtained in the previous YADE time step (K_1, K_2, K_3, K_4, K_5) are assigned to the center of the five cells in the COMSOL model. Equation 7 is solved using constant-head boundary conditions and based on a linear interpolation of the K values.

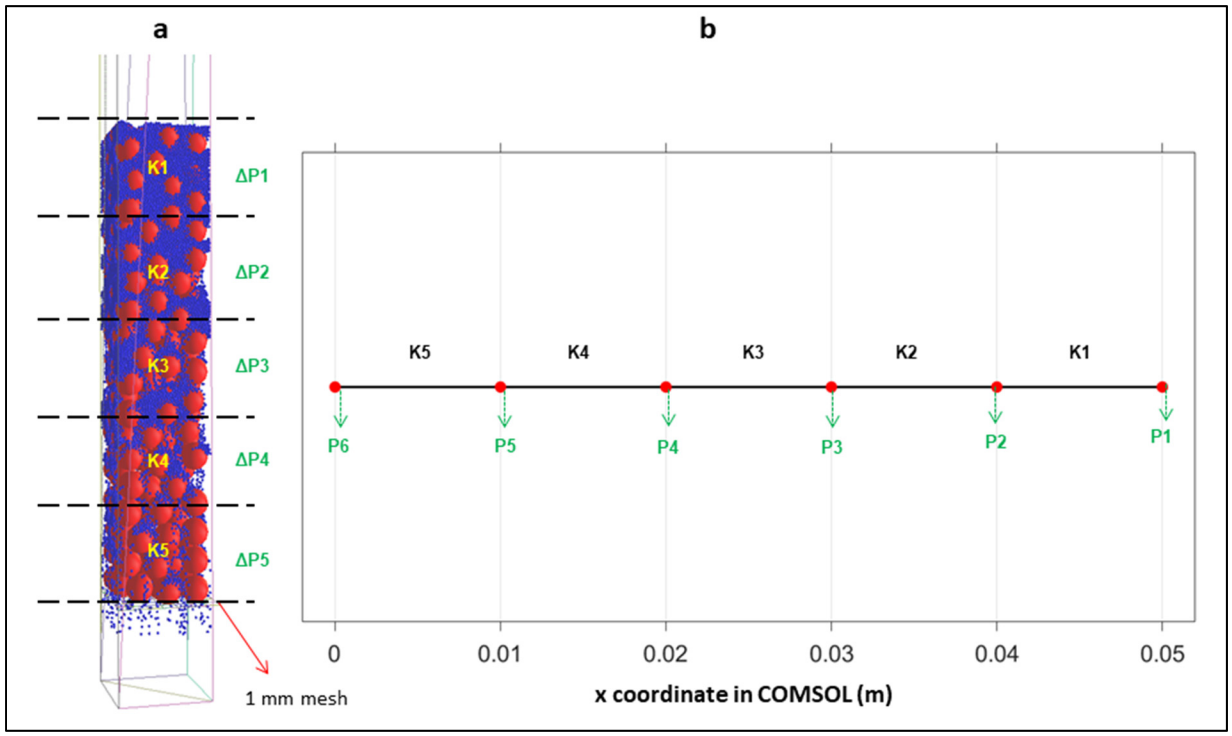


Figure 4.5 Model implementation in YADE and COMSOL for the full specimen procedure. The vertical axis in YADE (a) is represented by the x coordinate in COMSOL (b)

The COMSOL model allows the pressure to be calculated at the upper and lower boundaries of each cell ($P_1, P_2, P_3, P_4, P_5, P_6$ in Figure 4.5b). The pressure difference for each cell is used in YADE to compute the drag force with equation (4) for the next global time step. More information on the calculation sequence with ICY for a similar full specimen model may be found in Pirnia et al. (2019).

4.4.2 Hierarchical multiscale model implementation

With the hierarchical multiscale procedure, particle flux values are calculated along the specimen in 5 YADE subdomains. The particle flux distribution is substituted into the particle conservation equation (4.10), to calculate and extrapolate the time-dependent n and S_s distributions along the specimen. As with the full specimen procedure, pressure values at the subdomain boundaries are calculated by solving equation (4.7) and drag force in the subdomains are calculated with equation (4.4).

The particle flux was calculated in 5 subdomains (1 cm × 1 cm × 1 cm). One of the motivations behind the multiscale hierarchical procedure is to eventually decrease the number of DEM particles in internal erosion simulations. For comparison purposes, for some of the scenarios presented in this paper, the subdomains encompass the complete specimen.

Particles that are located below each subdomain can clog the pore space and influence the migration of small particles in the subdomains above. Three scenarios were defined to verify this influence. In the first (reference) scenario, the finer particles below each subdomain were updated at the beginning of each YADE time step based on the porosity values that were extrapolated in the COMSOL model (Figure 4.6a). In the second scenario, the fine particles below each subdomain were kept but their number was not updated during the simulation. In the third scenario, the finer particles below each subdomain were removed from the initial specimen (Figure 4.6b).

Figure 4.7 shows the computation sequence for the hierarchical multiscale model. Each global time step (loop counter *n*) begins with a shorter YADE simulation for each subdomain (loop counter *i*). These shorter simulations (see also Figure 4.4 for time stepping scheme) allow the particle flux distribution, not the exact particle displacements, to be calculated. Mean particle flux values are calculated for each subdomain using equation 4.6 (f_1, f_2, f_3, f_4, f_5 in Figure 4.6c).

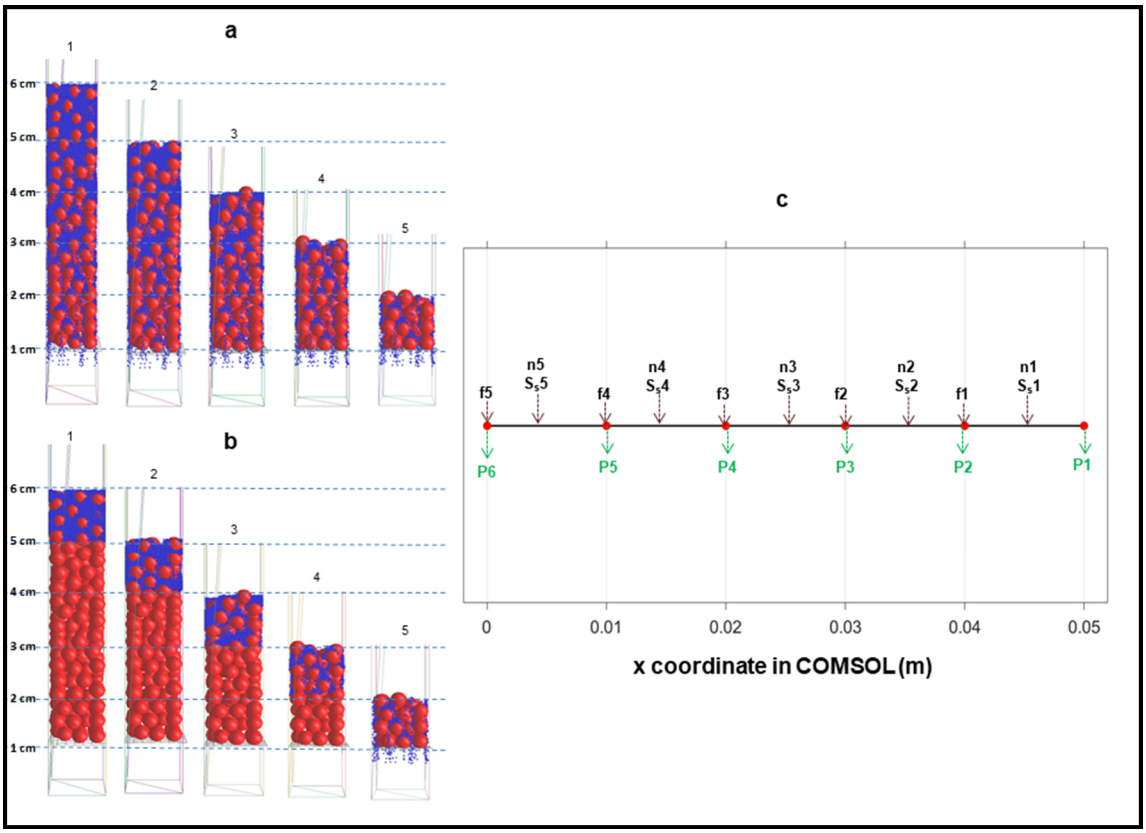


Figure 4.6 Model implementation in YADE and COMSOL for the hierarchical multiscale model. All particles (a) or only the coarse particles (b) are kept below the flux calculation section

The flux values are assigned in the COMSOL model after the loop on the DEM subdomains. It was found that better results are obtained when the flux is assigned to the bottom of each subdomain (Figure 4.6c). The particle conservation equation is then solved for a global time step (Figure 4.4). New average porosity values for each subdomain are calculated at the end of the time step (n_1, n_2, n_3, n_4, n_5). Corresponding S_s and K values are calculated with equations

(4.8) and (4.12). New pressure values are obtained by solving equation (4.7). The pressure and porosity values are written in a text file that is passed on to YADE and used to add or subtract particles from the model and to calculate new drag force values.

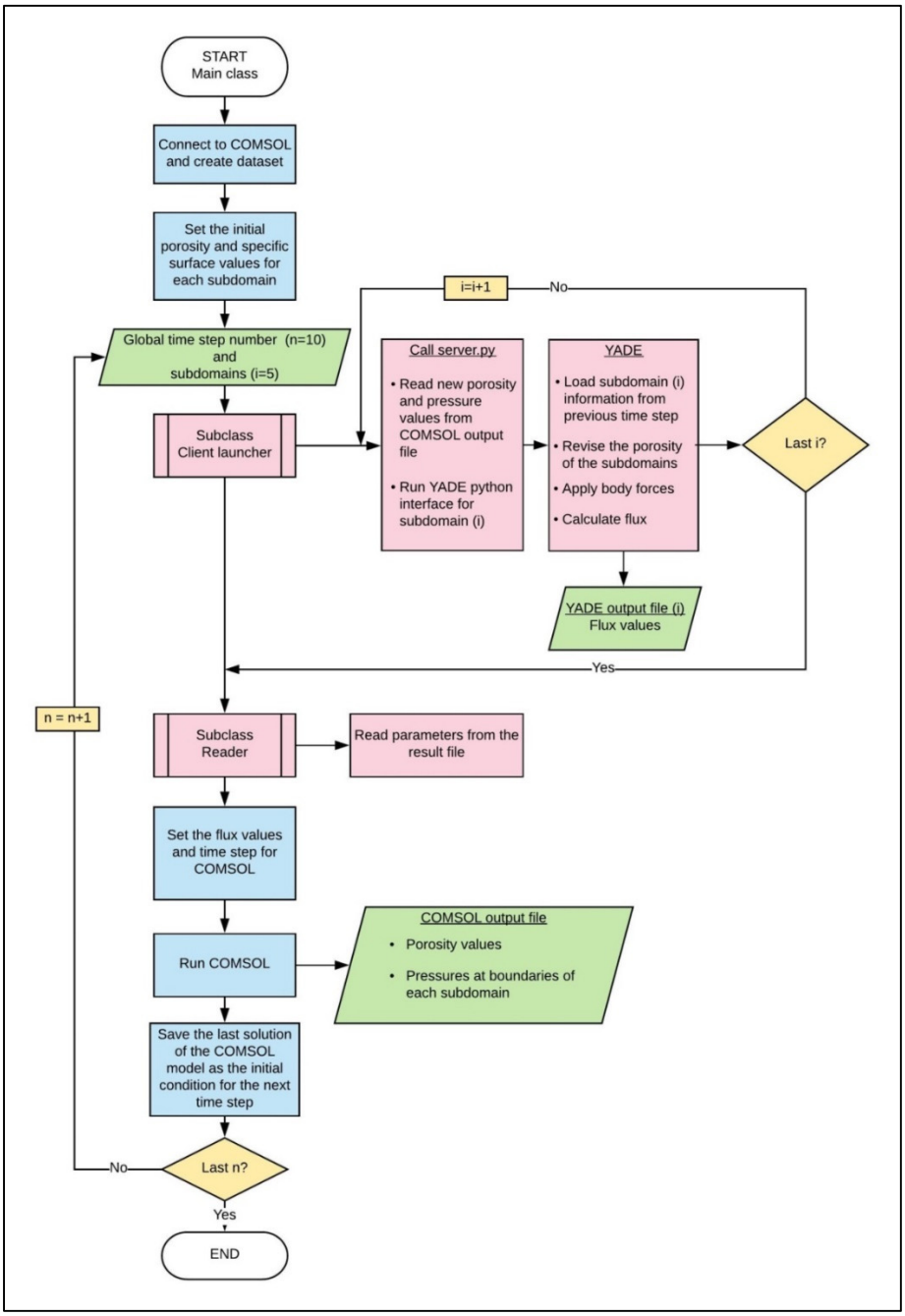


Figure 4.7 Calculation sequence in the hierarchical multiscale simulations

The initial porosity distribution for the first global time step was set using a linear interpolation of the subdomain porosity values. After the first time step, the porosity distribution from the previous global time step is used as the initial value for the next time step in the COMSOL model. A linear interpolation of the flux values was used. The interpolation methods for the flux and the initial porosity were found to have a negligible influence on the results.

Several parameters of the multiscale scheme were noted to have an influence on the results. The influence of the following parameters was studied: length of the global time step, particles under the subdomain, particle removal method, flux assignment location in the COMSOL model, interpolation methods for porosity and specific surface and number of subdomains. Some results are reported in the following sections along with a comparison with the full specimen procedure.

4.5 Results and Discussions

4.5.1 Full specimen model

From a numerical standpoint, the results of the full specimen procedure are mainly influenced by the time step and the number of cells. The influence of the number of cells was studied by Pirnia et al. (2019). The time step must be sufficiently short so that results are independent of the time stepping scheme. On the other hand, it must be sufficiently long for the particles to reach a constant velocity and to obtain representative flux values that are not overly influenced by random velocity fluctuations. For example, Figure 4.8 compares particle flux values for the full specimen procedure for time steps of 0.012 and 0.05 s. The drag force was not updated during this simulation. For the shorter time step, flux values for the five cells show oscillations that can reach 20 % (Figure 4.8a). The first flux value after the simulation beginning is also generally lower than the following flux values. The initially lower flux is caused by particle acceleration at the beginning of a DEM time step following the generation of new particles that are initially static and the loading of the previous DEM simulation. Flux changes are

dampened for the longest time step and do not show the impact of initial particle acceleration (Figure 4.8b). The time step was set to 0.05 s for the comparison with the hierarchical multiscale model.

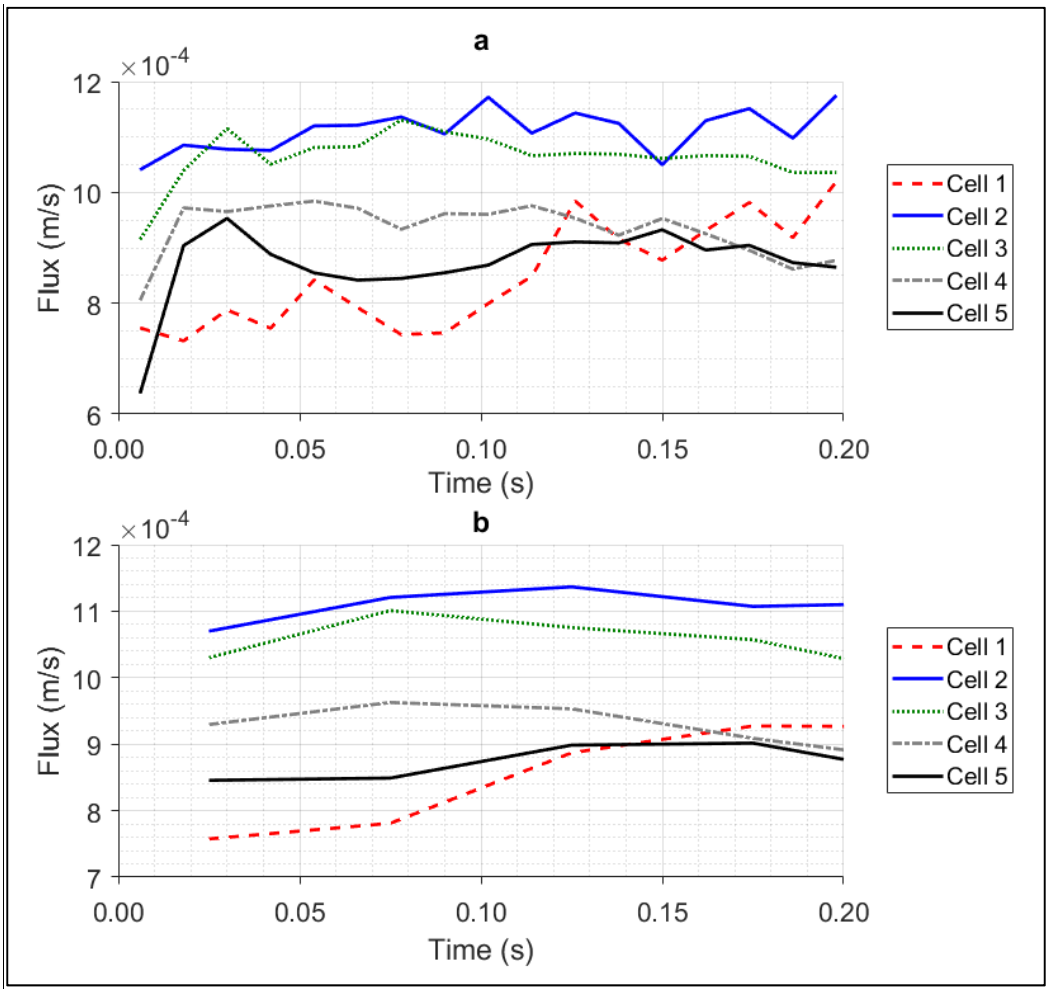


Figure 4.8 Particle flux as a function of time for two time steps of 0.012 s (a) and 0.05 s (b)

Figure 4.9 presents the relationship between the time elapsed since the simulation beginning and the cumulative mass of particles that are eroded and that reach the bottom of the specimen. The specimen contains 1.17 g of fine particles at the beginning of the simulation. Most of the fine particle mass (1.05 g) was eroded in 8 s after application of the constant hydraulic head

difference of 5 cm. Most of the particle mass, i.e. 0.95 g, reached the container in 4 s or less (0.95 g).

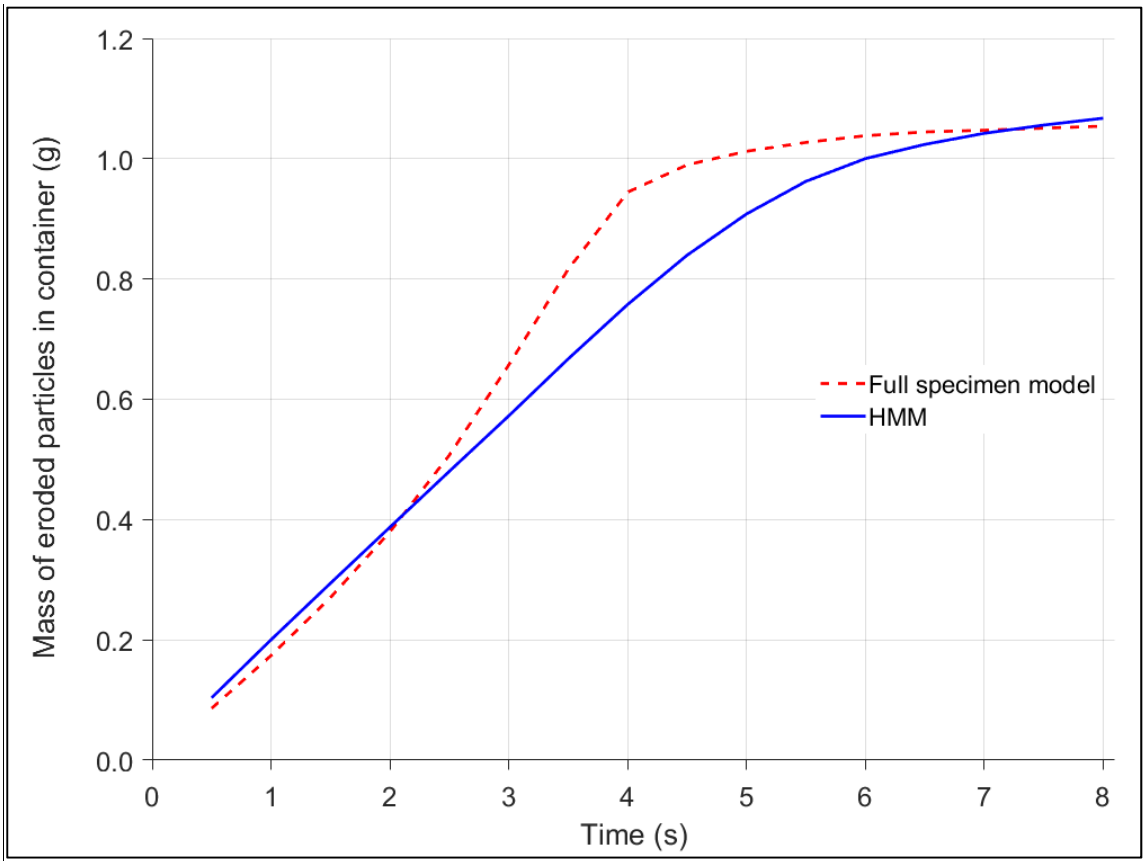


Figure 4.9 Cumulative eroded mass for the full specimen procedure and the reference hierarchical multiscale model (HMM).

Particle flux and porosity values for cells 1, 2 and 4 are illustrated in Figure 4.10. For the top cell (cell 1), the flux primarily decreases due to the entrainment of small particles toward the lower cells. The flux becomes very small as the porosity reaches a maximum value determined by the porosity of the coarse-grained skeleton. A similar behaviour, albeit delayed, can be observed for cell 2. Lower cells, such as cell 4, do not show a marked decrease in flux as they also receive particles from the upper cells. Their porosity is also higher at the beginning of the simulation due to particles falling through the coarse-grained skeleton during specimen preparation (Figure 4.1).

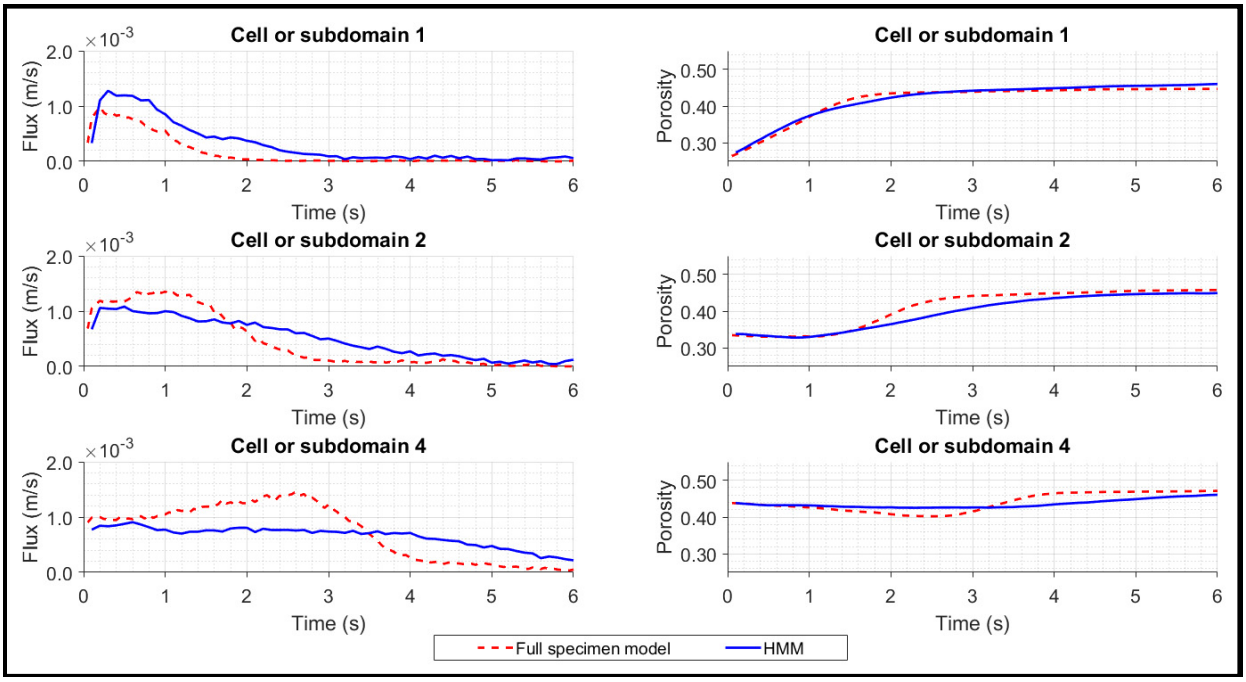


Figure 4.10 Flux and porosity in the DEM cells or subdomains as a function of time elapsed since the simulation beginning for the full specimen procedure and the reference hierarchical multiscale model (HMM)

4.5.2 Hierarchical multiscale model (HMM)

Figure 4.9 compares the cumulative eroded mass for the full specimen procedure and a reference hierarchical multiscale simulation with a global (COMSOL) time steps of 0.1 s and a YADE time step of 0.05 s (Figure 4.4). The cumulative eroded masses for both models are similar. A slight increase in the erosion rate can be observed for the full specimen after 2.5 s. This increase is not observed for the hierarchical multiscale simulation. This difference is probably due to instabilities that are associated with the time extrapolation for the hierarchical multiscale model.

Figure 4.10 compares the flux and porosity values for cells 1, 2 and 4 for the full specimen procedure and the reference hierarchical multiscale model (scenario 1). The results for subdomains 1, 2, 3 and 5 generally agree with those for the same cells in the full specimen

model. On the other hand, results for subdomain 4 show larger differences, especially for the flux which increases in the full specimen model at the beginning of the simulations.

Particles that are located below each subdomain can clog the pore space and influence the migration of small particles in the subdomains above. Three scenarios were defined to verify this influence. In the reference scenario, the finer particles below each subdomain were updated at the beginning of each YADE time step based on the porosity values that were extrapolated in the COMSOL model (Figure 4.6a). In the second scenario, the fine particles below each subdomain were kept but their number was not updated during the simulation. In the third scenario, the finer particles below each subdomain were removed from the initial specimen (Figure 4.6b).

Results for the three scenarios are compared in Figure 4.11. The three scenarios show almost the same temporal and spatial variations of porosity and flux. However, in the case of the third scenario, the porosity of the first subdomain was increased at the beginning of the simulation ($t < 1$ s). The clogging effect due to the fine particles below the subdomain is more important in the first subdomain as, after the first subdomain, the second subdomain has the smallest porosity and the largest content of fine particles at the beginning of the simulation.

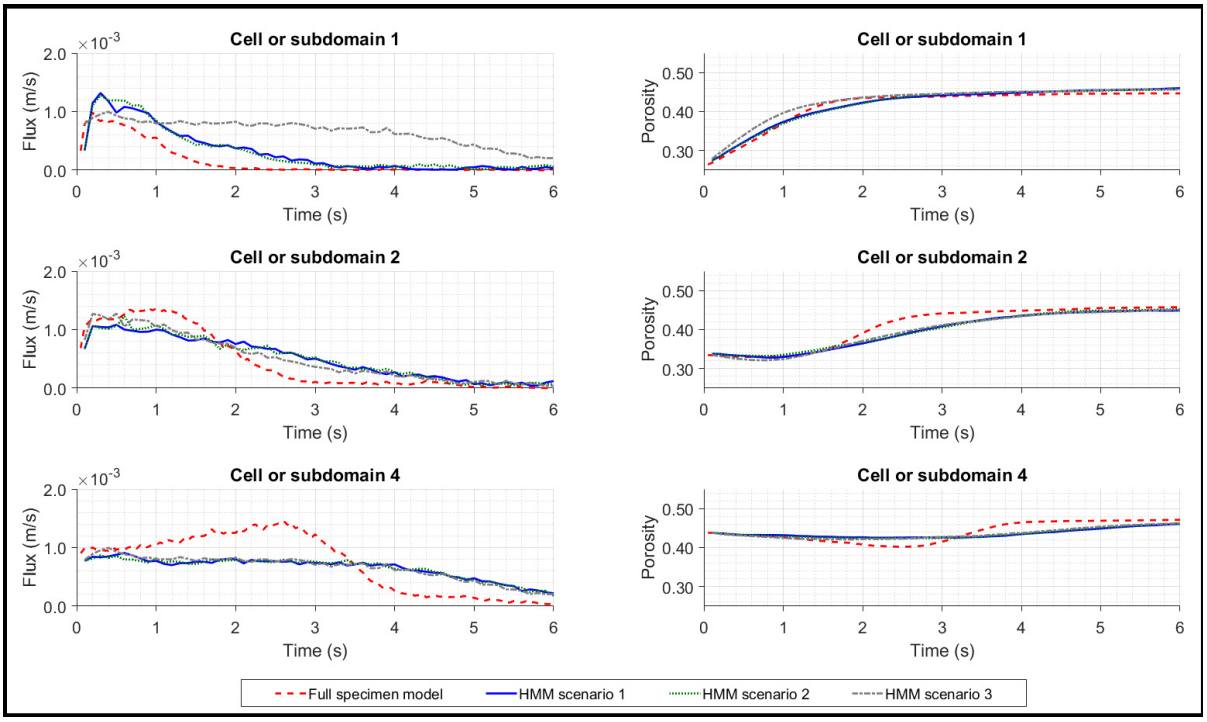


Figure 4.11 Flux and porosity variations for the hierarchical multiscale FEM-DEM model (HMM) with different updating methods for the finer particles below each DEM subdomain

In the reference simulation, the flux values calculated in YADE for each subdomain were assigned to the endpoint of each subdomain in the COMSOL model. To verify the influence of this parameter, the flux values were also assigned to the center of each subdomain. As shown in Figure 4.12, the porosity for the reference method are closer to the full specimen results, especially for subdomains 1 and 2. When the flux is assigned to the center of a subdomain, the flux and porosity derivatives (equation 4.10) in the lower half of the subdomain are influenced by the flux value assigned in the next subdomain.

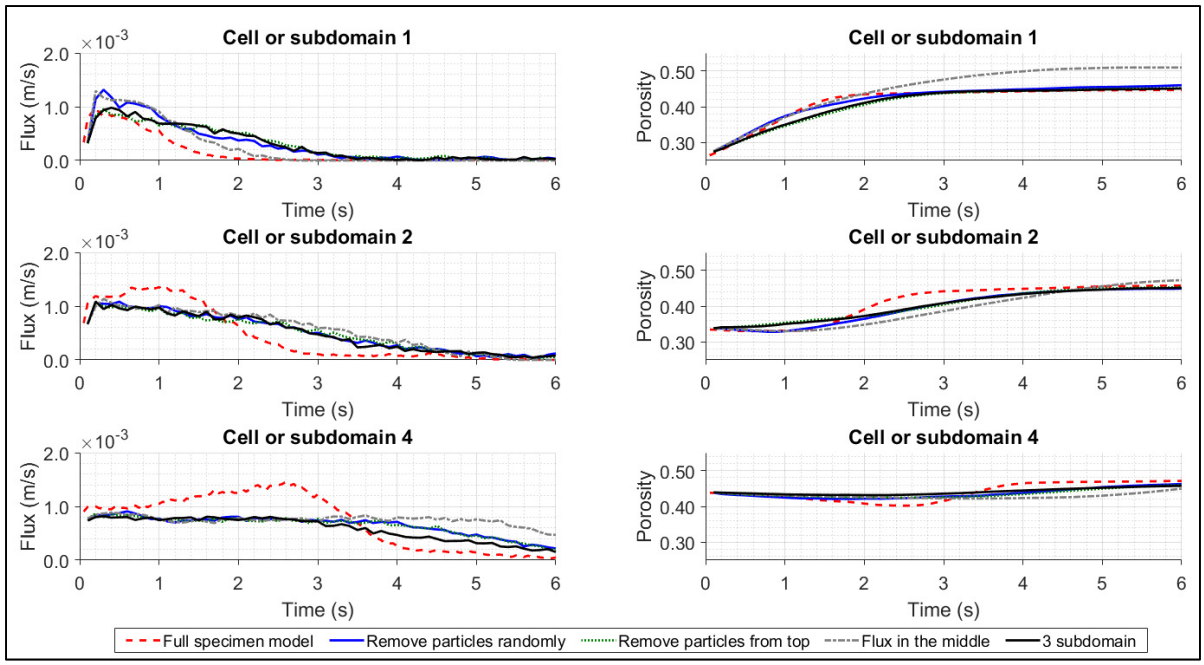


Figure 4.12 Flux and porosity variations for the hierarchical multiscale model under different particle removal methods from DEM subdomains, assigning the flux values at the middle of the subdomains in COMSOL model, and three DEM subdomains

The method applied to remove particles from the YADE subdomain at the end of each COMSOL time step can also have an influence on the results. In the reference simulation, the fine particles were selected randomly to be eliminated from the domain. For a suffusion example, it might also seem reasonable to remove the particles at the top (or upstream side) of each subdomain, especially for the first subdomain in which new particles are not added at the upstream boundary. A new function was thus programmed in the YADE model to remove the particles from top to bottom in each subdomain. This particle removal method only improved the results for the first subdomain at the beginning of the simulation (0.5 s). Otherwise, random particle removal showed a better agreement with the full specimen results.

In the previous simulations, the five cells represented the total length of the specimen because of the need for a minimum subdomain height. One of the objectives of the hierarchical multiscale model is to simulate larger applications (e.g. dams) for which DEM subdomains would provide particle flux values at nodes on a mesh. The performance of the hierarchical

multiscale model was thus verified by simulating the same test, but with three subdomains (1 cm × 1 cm × 1 cm) instead of five. Flux values were calculated with YADE for subdomains 1, 2 and 5 were kept. Three subdomains were enough to reproduce particles transport through the specimen. However, the porosity in the absent subdomains, as calculated from the COMSOL model, were less accurate (Figure 4.12).

Different time step lengths for COMSOL were also compared. Results for COMSOL time steps of 0.05, 0.1, 0.2 and 0.5 s are plotted in Figure 4.13. The smallest time step (0.05 s) is equal to the time step in YADE. The model did not show any significant change for time steps below 0.2 s. The porosity values were often overestimated compared to the full specimen model values when the time step was longer than 0.2 s as it leads to an overestimation of the flux due to removing more particles from the subdomains. It can be inferred that the rate of porosity changes along the full specimen model was slower than 0.2 s.

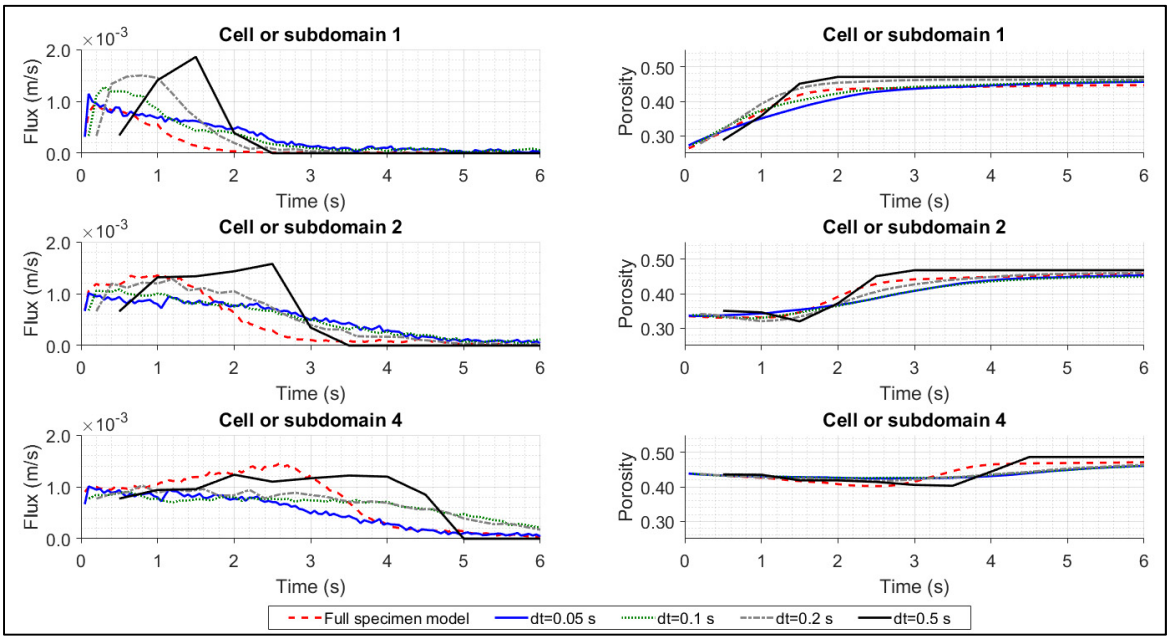


Figure 4.13 Porosity and flux for the hierarchical multiscale model under different COMSOL time steps for a constant YADE time step

4.6 Conclusion

This paper introduced a hierarchical multiscale FEM-DEM model to simulate internal erosion. The model is based on ICY, an interface between COMSOL (FEM) and YADE (DEM). The micromechanical behaviour of the particles is modelled in distinct DEM subdomains along the specimen. Shorter DEM simulations allow the flux of particles to be calculated for each subdomain. These flux values are assigned in a 1-D COMSOL model that uses a particle conservation equation to predict new porosity and drag force values after longer global time steps. At the beginning of a new global time step, these values are sent to the DEM subdomains to update the porosity and drag force. The water flow through the sample is modelled with a continuum method by solving a water conservation equation based on Darcy's law. A coarse-grid method was used to calculate the drag force value for each subdomain.

The method was validated using a suffusion test. The porosity, flux values and cumulative eroded mass calculated with the hierarchical multiscale procedure were generally consistent with results obtained with a full DEM specimen and without a particle conservation equation. The proposed hierarchical multiscale method could be improved and adapted to simulate internal erosion for large structures (e.g. embankment dams) and to act as a stepping stone in the development of continuum methods for the modelling of internal erosion. Many applications in soil mechanics, such as internal erosion and fluidized bed, could benefit from the multiscale model that considers both the discrete and continuum scales. The proposed model could also be used to improve full continuum methods (e.g. Vardoulakis (1996)). The numerical aspect of the hierachical multiscale model could be improved by involving an implicit time-stepping scheme in the COMSOL model.

CHAPTER 5

DRAG FORCE CALCULATIONS IN POLYDISPERSE DEM SIMULATIONS WITH THE COARSE-GRID METHOD: INFLUENCE OF THE WEIGHTING METHOD AND IMPROVED PREDICTIONS THROUGH ARTIFICIAL NEURAL NETWORKS

Pouyan Pirnia [a], Francois Duhaime [b], Yannic Ethier [c], Jean-Sebastien Dube [d]

[a, b, c, d] Department of Construction Engineering, École de technologie supérieure, 1100, rue Notre-Dame Ouest, Montréal, Québec, Canada, H3C 1K3

Paper published in *Transport in Porous Media,* June 2019

5.1 Abstract

Several methods are employed for drag force calculations with the discrete element method depending on the desired accuracy and the number of particles involved. For many applications, the fluid motion cannot be solved at the pore scale due to the heavy computational cost. Instead, the coarse-grid method (CGM) is often used. It involves solving an averaged form of the Navier–Stokes equations at the continuum scale and distributing the total drag force among the particles. For monodisperse assemblages, the total drag force can be uniformly distributed among the particles. For polydisperse assemblages however, the total CGM drag force must be weighted. It can be applied proportionally to the volume (CGM-V) or surface (CGM-S) of each particle. This article compares the CGM-V and CGM-S weighting methods with the weighting obtained by solving the Navier-Stokes equations at the pore scale with the finite element method (FEM). Three unit cells (simple cubic, body-centered cubic and face-centered cubic) corresponding to different porosity values (respectively 0.477, 0.319 and 0.259) were simulated. Each unit cell involved a skeleton of large particles and a smaller particle with variable size and position. It was found that both the CGM-V and CGM-S weighting methods do not generally give accurate drag force values for the smaller particles in

a polydisperse assemblage, especially for large size ratios. An artificial neural network (ANN) was trained using the FEM drag force as the target data to predict the drag force on smaller particles in a granular skeleton. The trained ANN showed a very good agreement with the FEM results, thus presenting ANN as a possible avenue to improve drag force weighting for the coarse-grid method.

5.2 Introduction

Fluid-particle interactions play a fundamental role in most geotechnical problems and applications. For example, fluid flow can trigger particle detachment and transportation during internal erosion in embankment dams. Through the concept of effective stress, fluid flow also influences the shear strength and compressibility of granular materials.

The discrete element method (DEM), originally proposed by Cundall and Strack (1979), has been widely used to model the micromechanical behaviour of granular materials. The success of this method lies in simple governing equations that consider each particle and the interactions between them. The motion of each particle is computed from Newton's second law of motion. A contact model (force–displacement law) describes the forces at the particle contacts. DEM has yielded insights into the mechanical response of granular materials at both the micro- and macroscale.

Many algorithms have been presented in the literature to couple particle motion and fluid flow. The accuracy of fluid-particle interaction models depends on drag force calculations (O'Sullivan, 2015). Drag force models for geotechnical applications can be classified into two main groups. The first group is based on numerical solutions to the Navier-Stokes or discrete Boltzmann equations at the pore scale. The second group includes coarse-grid methods based on Darcy's law (Darcy, 1856) or its extension by Brinkman (1949) and Ergun (1952) for higher flow velocities.

Solving the Navier-Stokes or discrete Boltzmann equations at the pore scale is considered the most accurate technique to model fluid flow through the interconnected voids of granular materials (Holmes et al., 2011; Beestra et al., 2007; Chareyre et al., 2012; Hill et al., 2001). Flow at the pore scale can be solved with most numerical methods encountered in computational fluid dynamics (Liu and Liu, 2010). Mesh-based methods like the finite volume, finite difference and finite element methods (FEM) are relatively common. Recently, the lattice Boltzmann (LB) (Cook et al., 2004; Lominé et al., 2013; Rubinstein et al., 2016) and smoothed particle hydrodynamics (SPH) (Holmes et al., 2011; Liu and Liu, 2010) methods have become increasingly common. The LB method involves the modelling of fluid flow as the displacement of discrete packets on a lattice according to the discrete Boltzmann equation. With the SPH method, the fluid is represented by a set of particles that interact through smoothing functions. Pore scale approaches suffer from large computational costs that limit their application in geomechanics, especially for problems involving large numbers of particles. These methods are more suitable for fundamental research or for industrial applications involving a small number of particles.

The coarse-grid method (CGM) was originally proposed by Tsuji et al. (1993) for the modelling of fluidized bed applications. With this method, pressure and fluid velocities are calculated on a grid that is typically 5-10 times coarser than the average particle diameter. The flow velocity and pressure are calculated based on an averaged form of the Navier-Stokes equations that considers the porous media as a continuum. This approach avoids solving the Navier-Stokes equations in each pore. The averaged Navier-Stokes equations and the associated continuity equation have been presented by Zhu et al. (2007), Kafui et al. (2002) and Tsuji et al. (1993).

Several methods can be used to calculate the total drag force with CGM (O'Sullivan, 2015). The Ergun (1952) and Wen and Yu (1966) equations are two of the most commonly employed empirical drag equations for particle systems (O'Sullivan, 2015; Rubinstein et al., 2016). The Ergun and Wen and Yu relations are valid for cases where porosity is less than 0.8 and more

than 0.8, respectively. For geotechnical applications, fluid flow is generally assumed to be laminar because of the low porosity and flow velocity (Goodarzi et al. 2015). In this case, drag force can be calculated based on Darcy's law and permeability prediction methods, such as the Kozeny-Carman equation (Pirnia et al. 2019; Chapuis and Aubertin 2003).

The geotechnical literature presents several examples of CGM drag force calculations. Zeghal and El Shamy (2004) used the Ergun equation to calculate average drag force values for the modelling of soil liquefaction. Goodarzi et al. (2015) used CGM drag force values based on the Ergun and Kozeny-Carman equations to simulate upward seepage and isotropic compression. Pirnia et al. (2019) used CGM drag force values to model an internal erosion test in a permeameter. Darcy's law and the Kozeny-Carman equation were used to estimate the total drag force (Chapuis and Aubertin, 2003). Zhang et al. (2019) studied the seepage erosion mechanism of soils around tunnels based on a coarse-grid method in PFC3D (Itasca, 2004).

Papers on fluidized beds, such as Tsuji et al. (1993), deal with monodispersed particle systems. In this case, the total CGM drag force can be distributed uniformly on neighbouring particles that have the same size. On the other hand, geotechnical applications usually deal with polydisperse particle systems. In this case, the total CGM drag force must be weighted when applied to neighbouring particles with different sizes. In other words, larger drag force values should be applied on larger particles. The drag force can be applied proportionally to the particle volume (CGM-V) or surface (CGM-S). A survey of the previously cited examples shows that the CGM-V weighting method is more common.

The drag force on neighbouring particles can also vary locally for fluidized beds or other applications involving monodisperse particle systems. The accuracy of CGM for the modelling of fluidized beds has recently been investigated by comparing its results with those obtained from pore-scale simulations. Kriebitzsch et al. (2013) performed a comparative analysis for a gas-solid fluidized bed with uniform particles. They found that the average CGM drag force is about 33 % lower than the reference value from pore-scale methods. Esteghamatian et al.

(2017) also compared pore-scale and CGM results for a homogeneous liquid-solid fluidization of spherical particles. The pore-scale and CGM predictions of pressure drop over the bed and bed height were generally in agreement. However, CGM considerably underestimated the local particle velocity fluctuations, regardless of the applied drag laws (Beetstra et al., 2007; Di Felice, 1994; Huilin and Gidaspow, 2003). The drag force on each particle depends on its size, but also on its position with respect to other particles. Some particles can be hidden behind other particles and, as consequence, be influenced by a smaller drag force.

Although the coarse-grid method has been widely used to simulate fluid-solid interactions for fluidized beds and geotechnical applications, some ambiguity remains regarding the application of the different drag models, especially for systems comprising particles of different sizes. While total drag force are relatively easy to measure (e.g., Chapuis and Aubertin, 2003), it is not possible to observe directly the drag force applied on individual particles. Therefore, no systematic studies of the weighting methods employed to distribute the total CGM drag force on particles with varying sizes have been presented in the literature. There are no clear guidelines on the choice between the CGM-V and CGM-S weighting methods, even if the two approaches can result in drag force values that differ by several orders of magnitude for the same total drag force, especially for large particle size contrasts.

This paper first aims at analyzing the accuracy of the CGM-V and CGM-S weighting methods for simple particle systems comprising two particle sizes. Drag force obtained using the CGM-V and CGM-S weighting were compared with drag force values obtained with pore scale FEM models. The commercial FEM code COMSOL Multiphysics (COMSOL, 2017) was used to conduct simulations for three unit cells: simple cubic, body-centered cubic, and face-centered cubic. This paper only looks at weighting methods: the total drag force is set by the FEM boundary conditions. Using COMSOL's JAVA programming interface, a total of 2712 simulations were conducted by changing the small particle size and position inside the packings outlined by the coarser particles.

The second objective of the paper is to evaluate the capability of a multilayer perceptron artificial neural network (ANN) to predict the drag force weighting in a polydisperse system. The ANN was trained with the database generated by the FEM simulations. The input data for the ANN were the CGM-V and CGM-S drag forces, the particle size ratio, the porosity of the coarse grain skeleton and the distance between the small particle and the nearest large particle. The drag force predicted by the ANN were found to be in very good agreement with the FEM values.

Our results allow for the formulation of practical recommendations regarding the applicability of CGM for different particle size ratios and porous media porosity values. The ANN presented in the paper introduces a new drag force calculation method that could potentially combine the accuracy of microscale methods with the computing efficiency of CGM methods.

5.3 Coarse-grid method

Fluid flow at the subparticle scale is governed by the continuity (equation 5.1) and Navier-Stokes equations (equation 5.2). These equations respectively express fluid mass (equation 5.1) and momentum (equation 5.2) conservation for incompressible flow.

$$\nabla . \boldsymbol{u} = 0 \tag{5.1}$$

$$-\nabla P + \rho_w \mathbf{g} + \nabla \cdot (\mu \nabla \mathbf{u}) = \rho_w \frac{\partial \mathbf{u}}{\partial t} \tag{5.2}$$

where $\mathbf{u}$ is the fluid velocity, P is the pore pressure, ρ_w is the fluid density, $\mathbf{g}$ is the acceleration of gravity, μ is the fluid dynamic viscosity, t is the time variable, $\nabla \cdot$ is the divergence operator and ∇ is the gradient operator. Bold fonts refer to vector quantities.

The total fluid force, or drag force $\mathbf{F_D}$, on a particle can be expressed as the integral of the sum of viscous stress (τ) and pressure on the surface (S):

$$F_D = \int_S (\tau + P)\boldsymbol{n}\, dS \tag{5.3}$$

where $\mathbf{n}$ is the unit normal to a differential surface element *dS*.

Darcy's law was derived experimentally by Henry Darcy in 1856. It relates the effective or Darcy velocity (**v**) to the hydraulic gradient (∇h) and the porous media hydraulic conductivity (*K*):

$$\mathbf{v} = -K\, \nabla h \tag{5.4}$$

The hydraulic head (*h*) is the sum of the pressure head (P/γ_w), where γ_w is the unit weight of water, and the elevation head (*z*). The contribution of the velocity head ($v^2/2g$) is usually neglected for porous media. Equation 5.4 can be reformulated in terms of the elevation and pressure heads.

$$\mathbf{v} = -\frac{K}{\gamma_w}(\nabla P - \rho_w \mathbf{g}) \tag{5.5}$$

Darcy's law can be also derived theoretically by upscaling the incompressible Navier-Stokes equations using a volume averaging procedure (Narsilio et al., 2009; Whitaker, 1985).

Compared to the drag force on the fluid, the total drag force ($\mathbf{F_D}$) on a particle system is equal but of opposite direction. The total drag force on a system of particles can thus be derived based on Darcy's law (equation 5.4) and force equilibrium considerations (Figure 5.1). For example, along the y-axis on Figure 5.1:

$$F_D = \Delta P\, dxdz \tag{5.6}$$

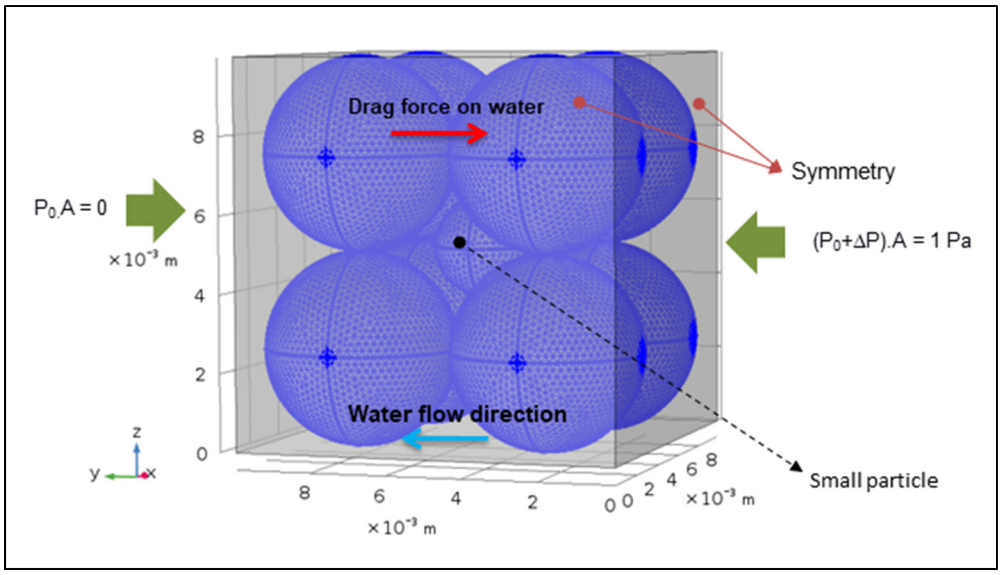


Figure 5.1 Boundary conditions for Simple Cubic packing

where ΔP is the pressure change along the y axis for a representative elementary volume (REV) dxdydz. and F_{DY} is the drag force component along the y axis. Equation 5.6 can be generalized based on equation (5.5):

$$\mathbf{F_D} = (\nabla P - \rho_w \mathbf{g})V = -\frac{\gamma_w V}{K}\mathbf{v} \tag{5.7}$$

where V is the REV volume. $\mathbf{F_D}$ corresponds to a total drag force that must be distributed among the particles. The CGM-V and CGM-S weighting methods allow two different drag force values to be derived for each particle in a representative elementary volume (REV) (Zeghal and El Shamy, 2004). If $\mathbf{F_D}$ is applied proportionally to the volume of each particle in volume V, the drag force on one particle is given by:

$$\mathbf{F_{DPi}} = \frac{\mathbf{F_D}}{(1-n)V} V_{Pi} \tag{5.8}$$

where $\mathbf{F_{DPi}}$ is the drag force on particle *i*, *n* is the porosity (volume of void in *V*/*V*), V_{Pi} is the volume of particle *i*.

The drag force on each particle can be defined by the following relationship if equation (5.7) is substituted in equation (5.8):

$$\mathbf{F_{DPi}} = \frac{(\nabla P - \rho_w \mathbf{g})}{(1-n)} V_{Pi} \tag{5.9}$$

If the total drag force is applied proportionally to the projected surface of each particle, the drag force is defined by:

$$\mathbf{F_{DPi}} = \frac{\mathbf{F_D}}{\sum_{j=1}^{k} \frac{\pi d_j^2}{4}} . A_{Pi} \tag{5.10}$$

where *k* is number of particles in the representative elementary volume, d_j is the diameter of article j, and A_{Pi} is the projected area of particle i.

By substituting equation (5.7) in equation (5.10):

$$F_{DPi} = \frac{\left(\nabla P + \gamma_w \nabla z\right)\, dxdy\, dz}{\sum_{i=1}^{n} \frac{\pi d_i^2}{4}} . A_{Pi} \tag{5.11}$$

By substituting the specific surface (total surface divided by total mass) in equation (5.11), the following relationship is obtained:

$$\mathbf{F_{DPi}} = \frac{(\nabla P - \rho_w \mathbf{g})}{\rho_s (1-n) S_s} A_{Pi} \tag{5.12}$$

where ρ_s is the solid phase density.

5.4 Methodology

5.4.1 FEM model development and drag force calculations

Three COMSOL models were created to simulate the water flow through 3-D packings in response to a pressure difference between the inlet and outlet. For each model, the spheres are contained in a cube of size L = 1 cm. The loosest packing is a Simple Cubic (SC) packing of eight spheres (Figure 5.1). The large particle diameter is D = 0.5 cm (1/2 L). The porosity of the packing is 47.7 %. The second and third packings correspond respectively to the Body-Centered Cubic (BCC, Figure 5.2a) structure with a porosity of 31.9 % and the Face-Centered Cubic (FCC) structure with the porosity of 25.9 % (Figure 5.2b). The particle radiuses in the case of BCC and FCC are respectively √3/4 L and 1/√8 L. For L = 1 cm, the BCC and FCC packings correspond to particles with diameters of 0.43 and 0.35 cm, respectively.

The steady-state incompressible Navier-Stokes equations (equation 5.1 and equation 5.2) were solved in the COMSOL models. The same combination of boundary conditions was used for the three packings. The inlet and outlet flow boundaries are highlighted in Figure 2. A constant pressure difference of 1 Pa was imposed across the length. A symmetry condition was applied on the lateral boundaries. No-slip boundary conditions were imposed on the surface of each particle.

As the first objective of the paper was to look at the weighting of the total drag force for coarse grid methods in systems comprising two particle sizes, a smaller particle was also generated in each unit cell. The possible size and position combinations for the small particle depend on the packing. The drag force on the small particle can be influenced by the pressure gradient, the small particle size and its position inside the box. Preliminary simulations confirmed that the drag force is proportional to the pressure gradient. Different pressure gradients did not

influence the distribution of the drag force between small and large particles. Therefore, only the position and size of the small particle were varied in the final set of simulations.

The pore scale F_D on each particle was computed with equation 5.3 and the FEM simulations. The CGM-V and CGM-S weighting was obtained with equations 5.9 and 5.12. In both cases, the numerator ($\nabla P - \rho_w \mathbf{g}$) was determined by the boundary conditions and domain size ($\Delta P/L$).

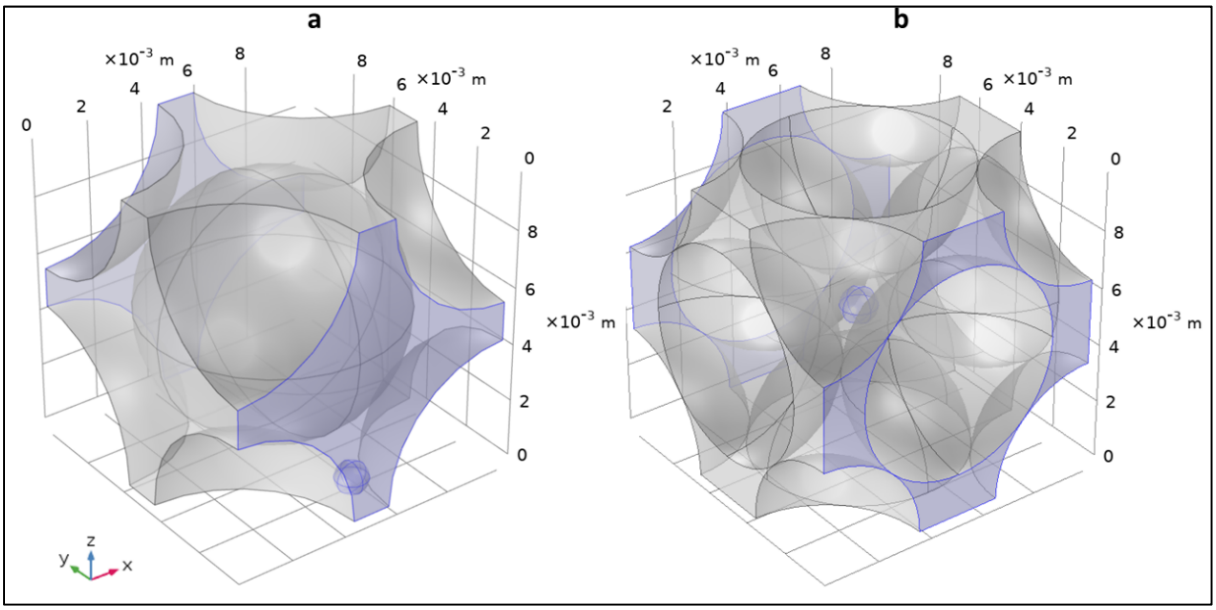


Figure 5.2 Geometry of COMSOL models for Body-Centered Cubic (a) and Face-Centered Cubic (b) packings

The mesh size was shown to have a large influence on drag force accuracy. Three different maximum element sizes were specified on the small particles, large particles and lateral boundaries for the three packings to make the simulations mesh independent. Each mesh consists of approximately 1, 2 and 5 million tetrahedral elements for the SC, BCC and FCC domains, respectively.

For the three packings, the drag force calculations for the smallest particle size (d = 0.0001 m) was more influenced by the mesh size. The accuracy of F_D was estimated from simulations

with decreasing mesh sizes. The error was defined as the difference between the value calculated with the final mesh and the value extrapolated for an infinitely fine mesh (Roache, 1997). The maximum error for all packings and particle sizes was close to 10 %, but the error was generally much smaller, especially for particles with $d > 0.0001$ m. The sum of viscous and pressure forces on all particles was also compared to the total force on the particles (ΔPL^2). The difference was less than 0.6%, 1.2% and 4% for the SC, BCC and FCC packings, respectively.

The simulations were conducted for the small particle diameters (d) and positions presented in Table 1. The range of small particle positions was constrained based on symmetry considerations. COMSOL's JAVA programming interface was used to change the size and position of the small particle in COMSOL models for the three packings and to run the simulations. The JAVA class was also used to reject parameter combinations for which the small particle was overlapping with the large particles, the lateral boundaries, the inlet or the outlet. For each simulation, the drag force on the particles and lateral walls, and the average velocity at the outlet surface were written in a .csv file. A total of 2712 simulations were conducted for the three packings.

Table 5.1 Parameters for the geometry of the small particle in the COMSOL models

Type of Packing	**Small particle diameter (m)**	**Positions**		
		x	**Y**	**Z**
SC	0.0001:L/100:0.0036	L/2:L/20:3L/4	L/4:L/20:3L/4	L/2:L/20:3L/4
BCC	0.0001:L/100:0.0016	0.001:L/10:L/2	0.001:L/10:L	0.001:L/10:L/2
CCP	0.0001:L/100:0.0011	0.001:L/10:L/2	0.001:L/10:L	0.001:L/10:L/2

5.4.2 Artificial Neural Network

An ANN was prepared with the MATLAB neural network toolbox to predict the drag force on small particles based on the dataset obtained from the COMSOL simulations. A neural network receives input data, processes these inputs, and returns output data that must match a target dataset. The input data used to train the neural network consisted of the drag force on the small particles calculated by CGM-V and CGM-S, the particle size ratio, the porosity and the differences in x, y and z coordinates between the center of the small particle and the center of the nearest large particle. The target data are the drag force values derived from the FEM model.

Input, hidden and output layers, in neural networks are made of a number of artificial neurons. Artificial neurons are the fundamental element and computing unit of the neural network (Figure 5.3). The inputs (x_i) are separately multiplied by synaptic weights (W_i). The neuron's bias (b) is added to the summed weighted inputs to generate the net input. The net input is passed through an activation function (f) to produce the output (y).

$$y = f\left(\sum_{i=1}^{n} x_i . w_i + b\right) \tag{5.13}$$

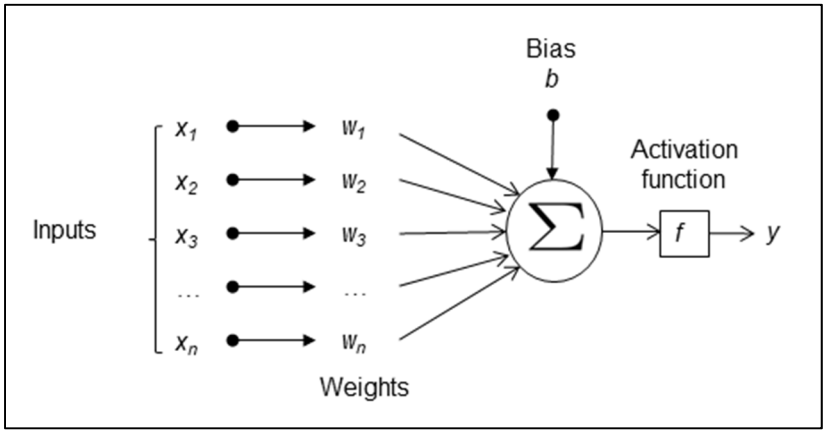


Figure 5.3 A single artificial neuron

The original dataset containing 2712 simulations was split into training, validation and testing datasets. During the training step, the weights are updated to minimize the error and improve the network performance. For supervised learning, the network output is compared with the target values to determine the error. After each training epoch, the validation dataset is used to verify if the network is overfitting the training data and losing its ability to generalize. The testing dataset is not introduced to the network during training. It provides an independent measure of the network performance after training. The main portion of the dataset (70 % chosen randomly) was presented to the network for training. The rest was split equally between the validation and testing datasets (15 % each).

A network with 7 neurons in the input layer, 20 neurons in two hidden layers and 1 neuron in the output layer was chosen in this study (Figure 5.4). Sigmoid activation functions were used to increase the nonlinearity of the neural network output (Karn, 2016). A hyperbolic tangent sigmoid transfer function with range (-1, 1) was used for the hidden layers (Figure 5.5a). A unipolar sigmoid activation function was used in the output layer to constrain the data to the range (0, 1) (Figure 5.5b).

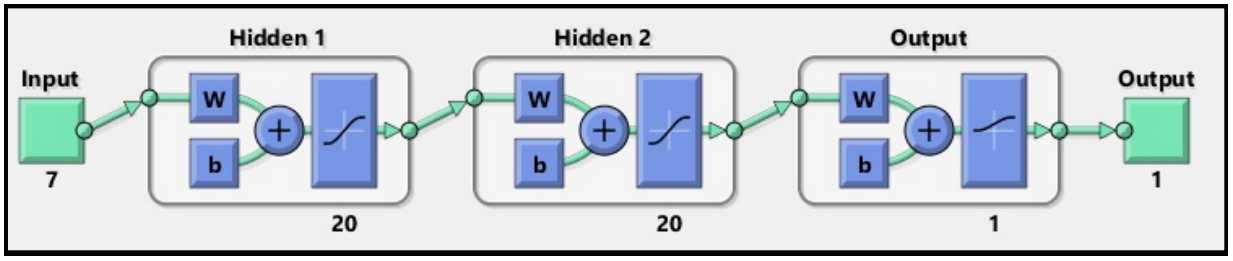


Figure 5.4 Architecture of the ANN trained with MATLAB

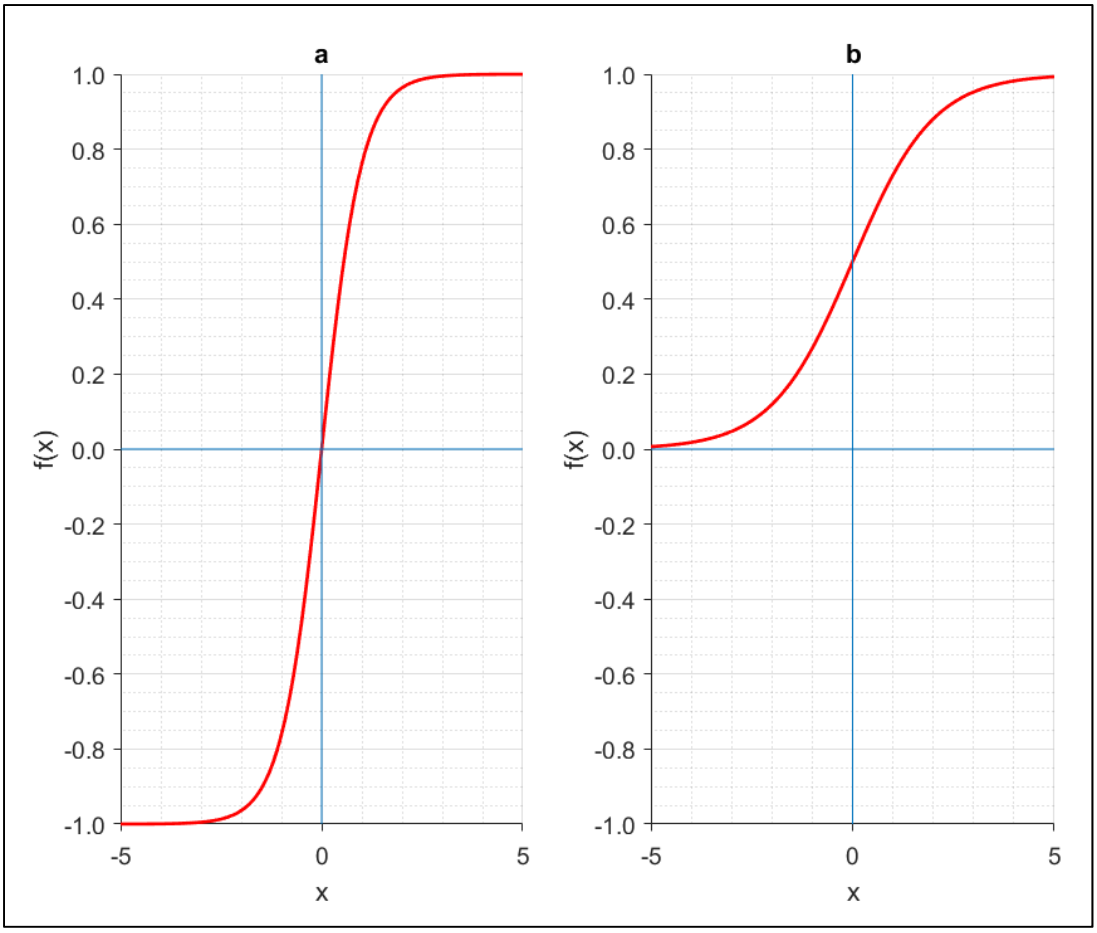


Figure 5.5 Bipolar sigmoid function (a), unipolar sigmoid function (b)

The performance of the network was evaluated using the mean squared error (MSE), the root mean squared error (RMSE) and the coefficient of determination (R^2). The MSE shows the quality of the estimator. It has the following definition:

$$MSE = \frac{\sum_{i=1}^{n}(T_i - P_i)^2}{n} \tag{5.14}$$

where T and P are vectors containing the target and predicted outputs, and n_s is the number of samples. The RMSE is simply the square root of the MSE. The R^2 indicates how well the predicted values fit the target values:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(T_i - P_i)^2}{\sum_{i=1}^{n}\left(T_i - \dot{P}\right)^2} \tag{5.15}$$

where $\dot{P}$ is the mean of the predicted values. The R^2 range between 0 and 1, the latter corresponding to a better fit.

The Levenberg-Marquardt (LM) algorithm (Levenberg, 1944; Marquardt, 1963) is the most commonly used method for training neural networks, especially for small and medium-sized networks (Nawi, 2013; Hagan and Menhaj 1994). The LM algorithm showed the best performance and reached the lowest RMSE and MSE values in the lowest number of epochs. Epochs correspond to the number of times the entire training dataset passes through the network. The early stopping method was used to stop training the network. It improves the generalization and avoids overfitting to the training data. The training was stopped when the error on the validation dataset increased for six iterations. The weights and biases that resulted in the minimum validation error were used for the trained model.

Hyperparameters are variables that determine the model structure. They are set before the training begins (e.g., number of hidden neurons). An optimization of hyperparameters is needed to reach the best prediction results. The parameters are chosen based on trial and error as there is no fixed rule to select them. The optimization for the ANN presented in this paper involved changing the number of hidden layers, the number of neurons in the hidden layer and the learning rate. The learning rate is a coefficient that determines how quickly a network updates its parameters to minimize the prediction error. A small learning rate of 0.0001 was found to produce the best results. The number of neurons in the input and output layers are governed by the number of input variables introduced to the network and the number of expected output from the network. The network was tested with 6, 8, 10 and 20 hidden neurons. The network showed significant improvement in accuracy when two hidden layers were used instead of one.

5.5 Results and discussion

Figure 5.6 compares the FEM drag force on small particles with the results of the CGM-V and CGM-S methods for the SC packing. All forces presented in this section are normalized by the total force on the packing ($F_{particle}$ / F_{sample}). The box plots represent the 3 quartiles of the FEM drag force values that were obtained for different small particle positions. The CGM-S values are systematically higher than the CGM-V values. For d/D>0.52, only the CGM-S values give a drag force that is consistent with the FEM results. For particle size ratios between 0.44-0.52, the drag force values from the CGM-S method correspond to the maximum FEM values, while the CGM-V values fall below or close to the minimum FEM values. The variability of the FEM drag force values on the small particles increases progressively with decreasing d/D ratios. This implies that the acceleration and velocity of small particles vary considerably as they move through the pore space compared to the larger particles. In the case of particle size ratios that are smaller than 0.44, the median FEM drag force is close to the CGM-S value but the maximum and minimum FEM drag force values fall outside of the range defined by the CGM-V and CGM-S values. The CGM-S and CGM-V methods cannot replicate the variability of the FEM drag force.

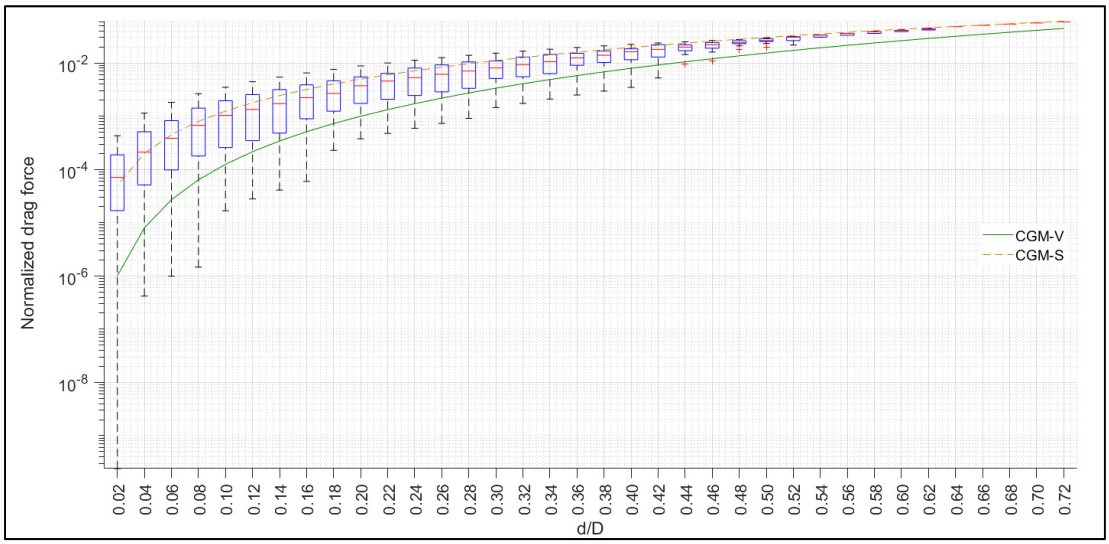


Figure 5.6 Distribution of the drag force values on the small particle in the Simple Cubic. The box plots shows the FEM values

For a given d/D ratio and packing, the drag force depends on the position of the small particle with respect to other particles. Figure 5.7 shows the relationship between the normalized drag force and particle position in the x-y plane for a small particle with a radius of 0.0001 m (d/D = 0.02) and SC packing for the large particles. The drag force can be seen to decrease when the particle is displaced from the center of the box (x = 0.005 m, y = 0.005 m and z = 0.005 m, Figure 5.1) toward areas with a slower fluid velocity between the two particles (x = 0.0075 m, y = 0.005 m and z = 0.005 m). On the other hand, the drag force reaches its maximum values when the small particle is placed in the constriction that faces the flow direction (x = 0.005 m, y = 0.0075 m and z = 0.005 m).

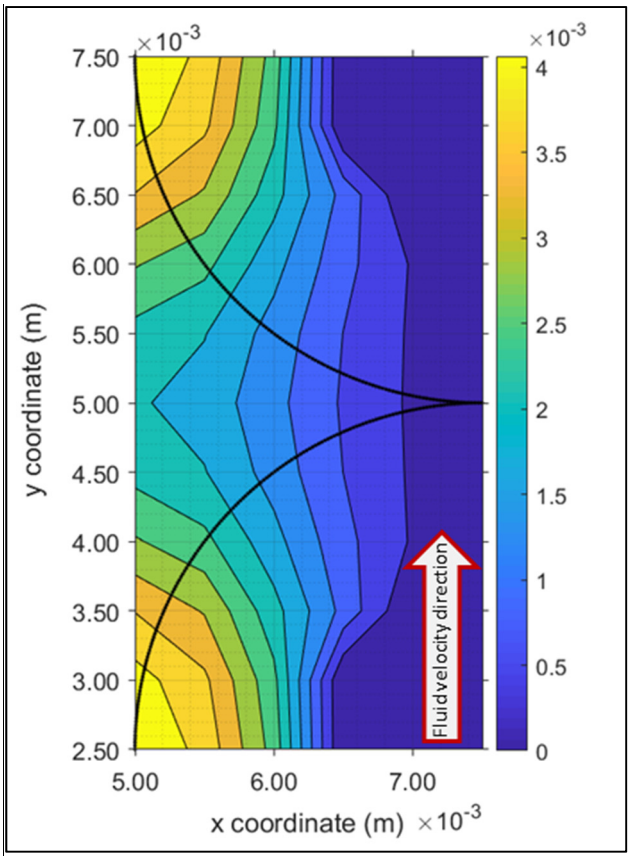


Figure 5.7 Normalized drag force on a small particle (d = 0.0001 m, d/D = 0.02) with respect to its position. The two quadrants correspond to the outline of the large particles (SC packing) on the x-y plane at z = 0.005 m

Fewer particle ratios and positions could be studied for the BCC and CCP packings due to their higher density. For both of these packings, the CGM-V and CGM-S drag force values were mostly lower than the FEM values (Figure 5.8a-b). The CGM-V method systematically underestimated the drag force compared to the FEM results for both BCC and CCP packings. The CGM-S method fared better. Nevertheless, it predicted values that were among the lowest quartile of the FEM results for 6 out of 7 particle size and packing combinations.

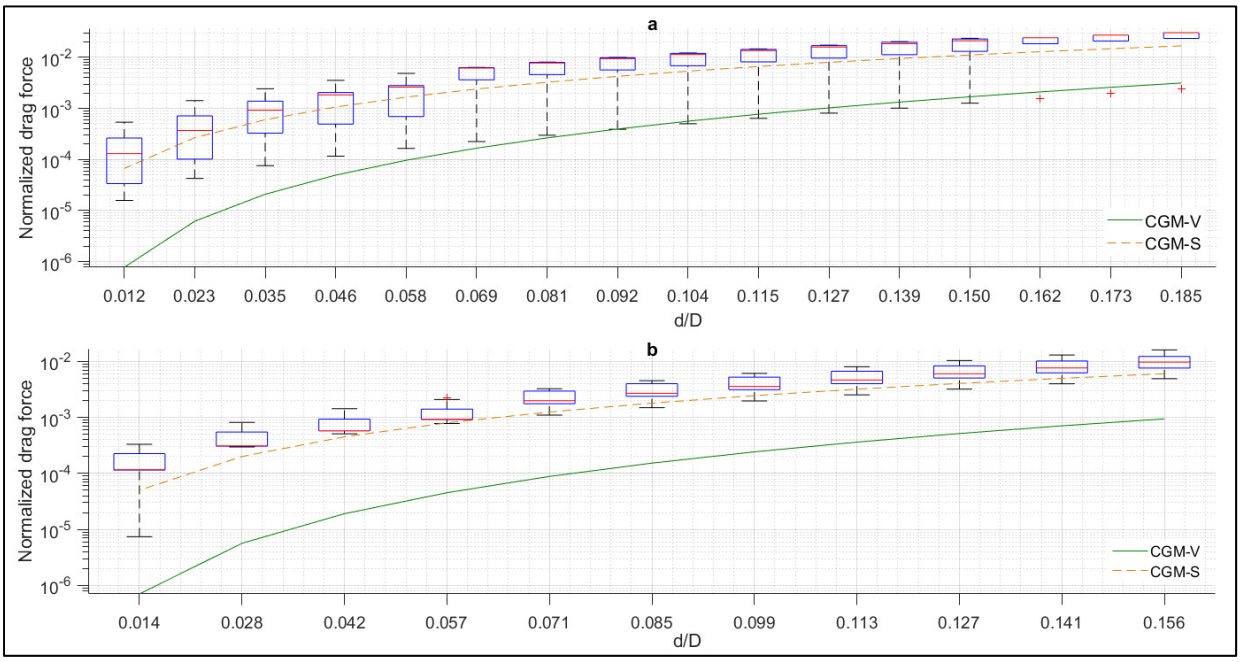


Figure 5.8 Distribution of the drag force values on the small particle in the Body-Centered Cubic (a) and the Face-Centered Cubic (b) packings. The box plots shows the FEM values

The results on Figures 5.6-8 clearly show that coarse grid methods, especially the CGM-V method, tend to underestimate the drag force on small particles in a porous material. For loose packings (e.g., SC structure with 47.7% porosity), the CGM-S values might be used to predict the drag force, as long as the particle size ratio is larger than 0.52. An average between the CGM-V and CGM-S values might be used to predict the drag force for the particle size ratios between 0.44-0.52. The maximum error between CGM-S and the FEM results is 146% while it is 77% between an average of the CGM-V and CGM-S values and the FEM results. The

coarse-grid method should not be considered as an accurate technique to predict the drag force for smaller particle size ratios and in the case of denser assemblies such as BCC and CCP. This observation is particularly important for the modelling of internal erosion and particle transport in porous media as these applications generally involve particle size ratios that are smaller than 0.44.

Performance of the ANN Levenberg-Marquardt algorithm (ANN-LMA) in terms of epoch number is shown in Figure 5.9. The training was stopped after 240 epochs while the training parameters such as weights and biases from iteration number 234 were applied for the final trained model. The best performance based on MSE on the normalized drag force for the validation set is 3.64×10^{-9} at epoch 234.

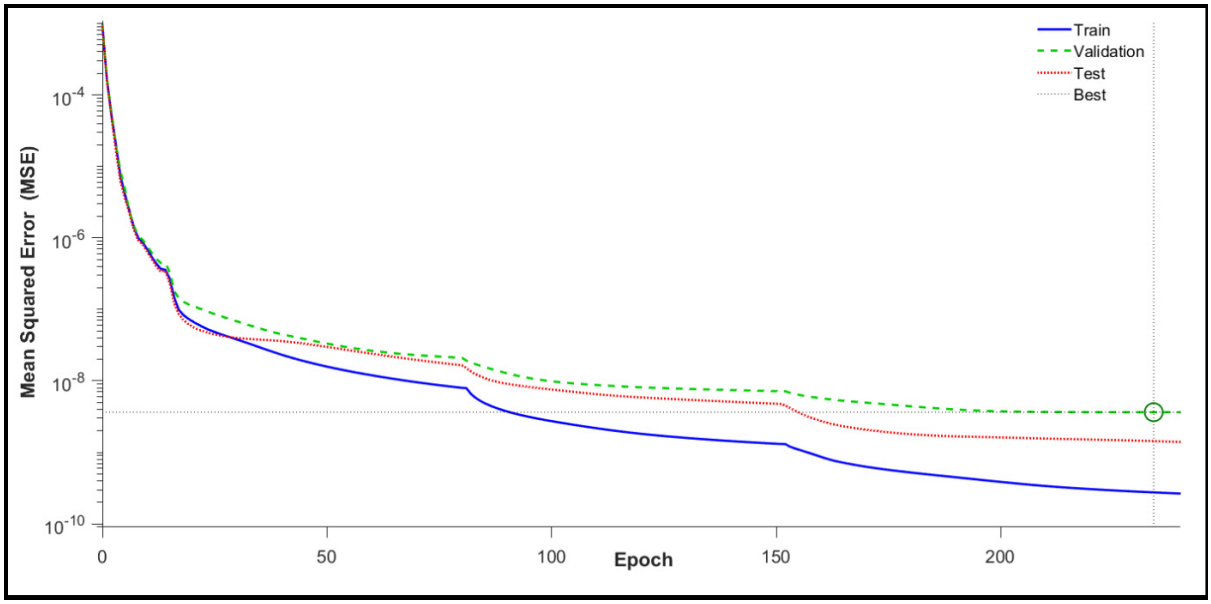


Figure 5.9 Evaluation of training algorithm per epoch

The performance of the trained model on the training, validation and testing sets is summarized in Figure 5.10. Sometimes a trained model shows very convincing performance during training, but it cannot predict properly on testing dataset. The nearly similar correlation

coefficient R^2 of the three datasets and speed of convergence to the best accuracy proves the right choice of the ANN architecture and training algorithm.

A comparative analysis of the ANN-LMA and two surface and volume methods against the FEM results is presented in Figure 5.11. The line of best fit compares the predicted drag forces with the desired values. The trained ANN algorithm could predict the drag force with a high degree of accuracy (RMSE = 0.000038). However, the Mean RMSE for the CGM-S and CGM-V were 0.0028 and 0.0032, respectively. There is an approximately perfect agreement between the ANN predictions and observed data (R^2 = 0.999) while CGM-S (R^2 = 0.8704) and CGM-V (R^2 = 0.8307) showed relatively poor correlations with the best fit line.

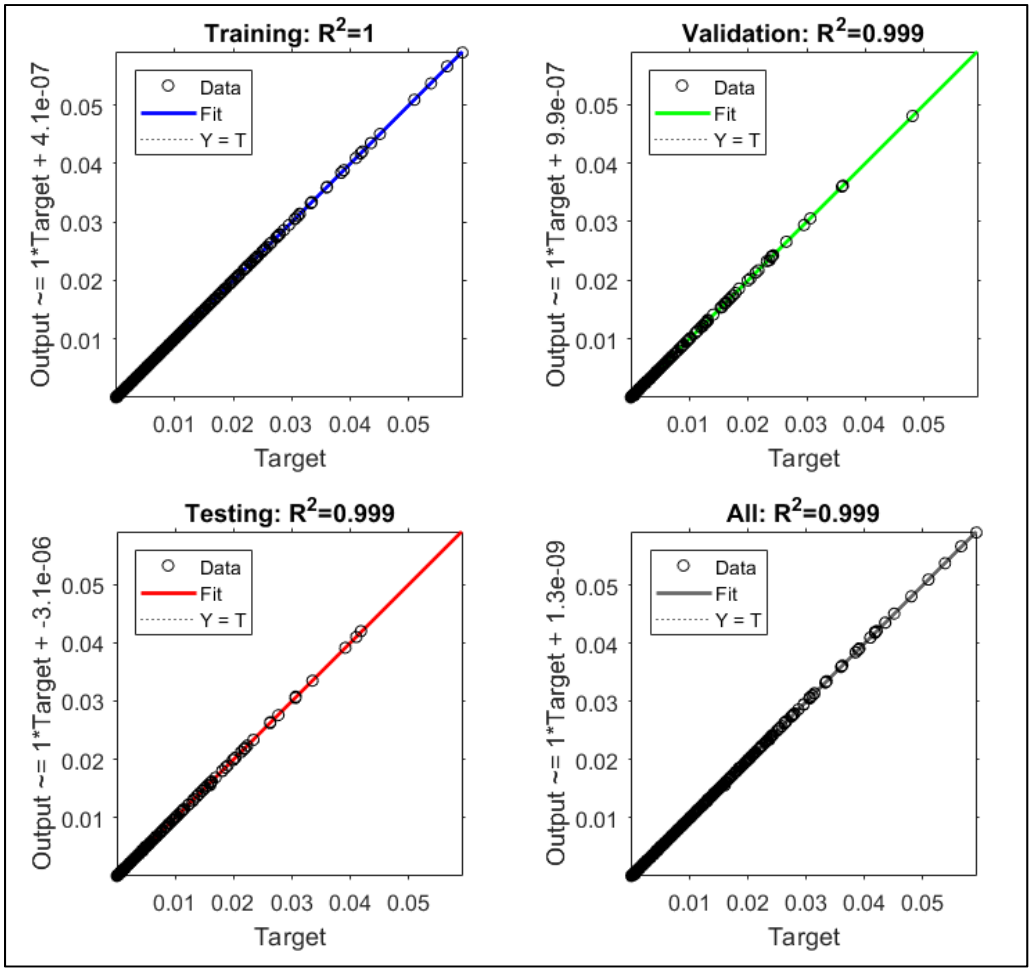


Figure 5.10 Performance of trained ANN on training, validation and testing datasets

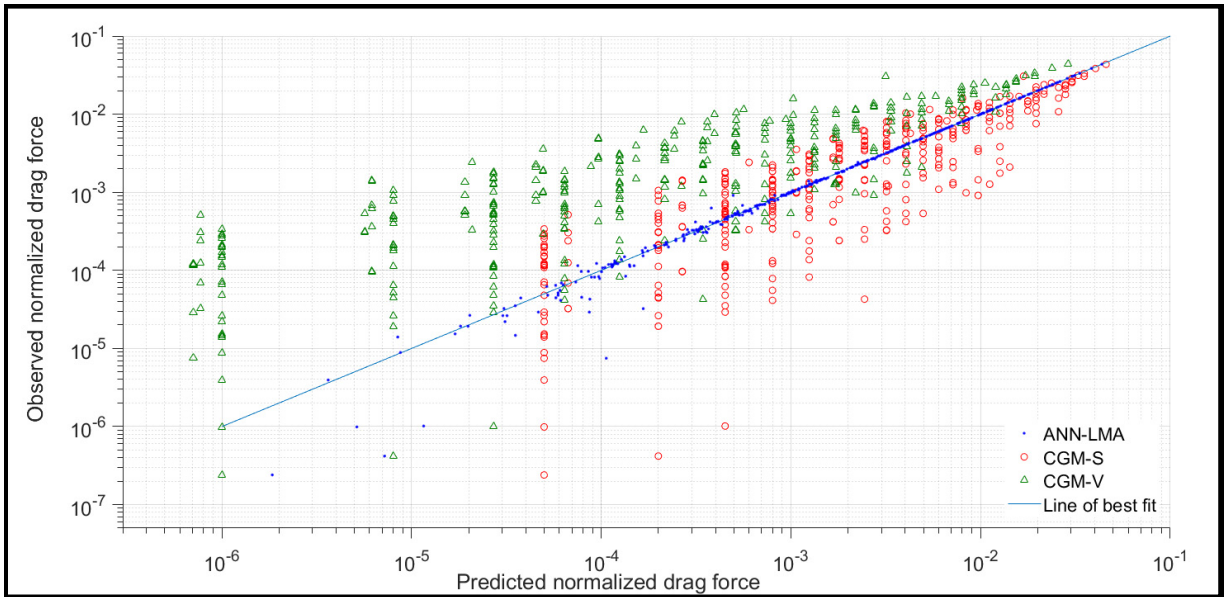


Figure 5.11 Comparison between the FEM and predicted drag forces

The ANN-LMA showed an excellent generalization for the small and noisy dataset. It could successfully estimate wide variations of the force regarding the positions for all particle size ratios. However, the model accuracies smoothly declined as the particle size ratios decreased.

The predicted results from ANN showed that the ANN could provide a better performance than the coarse-grid methods. A trained ANN might be replaced with the costly FEM and inaccurate coarse-grid methods to predict the drag force on particles in porous media. This method could eventually lead to quick and precise hydrodynamic forces for the coupled fluid-DEM codes.

For bidisperse sphere assemblages, the implementation of the ANN method in a DEM-FEM code such as ICY (Pirnia et al. 2019) is relatively straightforward. The total drag force can be calculated with the FEM code based on the Kozeny-Carman (Chapuis and Aubertin 2003), Ergun (1952) or Wen and Yu (1966) relationships. The weighting of the total drag force on smaller particles can be conducted in the DEM code using bounding volumes centred on each small particle. These bounding volumes allow the porosity and the specific surface around each

small particle to be calculated. The distance between the small particle and each larger particle within the bounding volume can be computed to find the smallest distance. These parameters and the total drag force allow the CGM-V and CGM-S drag force values to be calculated with Eqs. 9 and 12. The required input can then be fed to the ANN. The ANN training can be conducted using a dataset similar to the one presented in this paper. Alternatively, a dataset could be built from FEM simulations based on a large number of randomly selected bounding volumes. For the larger particles, based on the results presented in this paper, the CGM-S drag force can be used directly.

5.6 Conclusions

The coarse grid method is commonly used in geotechnical engineering to apply drag forces on polydispersed particle systems. The first objective of the paper was to compare the drag forces derived from Darcy's law and the CGM-V and CGM-S weighting methods for polydisperse assemblages, with the drag forces derived from the Navier-Stokes equations solved at the pore scale using the FEM. Three unit cells of 1 cm corresponding to maximum porosity values of 47.7, 31.9 and 25.9 % were simulated. Each unit cell consisted of a fixed coarse-grained skeleton and a smaller particle in the pore space. The CGM drag forces were applied on the smaller particles proportionally to their volume (CGM-V) or surface (CGM-S). A constant pressure difference of 1 Pa was applied across the unit cells. 2712 combinations of packing, small particle size and small particle position were simulated using COMSOL's JAVA programming interface.

The results show that in a loose particle packing (47.7% porosity), the CGM-S could provide results that were close to the more accurate FEM results for large size ratios (0.72-0.54), in other words for relatively uniform grain size distributions. For intermediate size ratios (0.52-0.44), the mean of the CGM-V and CGM-S values gave drag forces that were close to the median FEM drag force. The CGM-V and CGM-S methods generally did not produce accurate drag forces for smaller particle size ratios, especially since the variability of the FEM drag

forces increased with decreasing size ratios. The CGM results generally underestimated the drag force for the denser packings compared to the FEM results.

The second objective of the paper was to look at the applicability of ANN for drag force predictions. A back-propagation neural network was developed to examine the capability of ANNs to predict the drag force on smaller particles in a coarse-grained skeleton. The database containing the results of the 2712 FEM simulations was used for training, validating and testing the network. The ANN predictions agreed well with the target FEM values. The trained network showed a reliable prediction capability for all particle size ratios and packings. ANNs are thus a promising tool to calculate drag forces in polydispersed materials.

CHAPTER 6

CONCLUSIONS AND RECOMMENDATIONS

6.1 Conclusions

The introduction explained the necessity of analysing internal erosion in embankment dams. Internal erosion and overtopping are considered as two of the main causes of embankment dam failure. Internal erosion changes the hydraulic and mechanical characteristics of materials as a result of particle transportation in porous media. The phenomenon has been studied for about one century.

The literature review has presented experimental and numerical methods for the study of internal erosion. Numerical modelling can lead to a better insight into the internal erosion occurring inside the dam as it allows several factors and parameters to be considered in the process. DEM has been widely used to study the mechanics of the motion and interaction of dry granular and discontinuous materials. Fluid effects are now also included in an increasing number of DEM simulations.

A major drawback of DEM is the limited number of discrete bodies that can be included in a numerical model. As a consequence, DEM cannot be used for the modelling of large scale assemblies and must be coupled with continuum models in a multiscale analysis where small scale DEM simulations are conducted for selected nodes in the model. A multiscale model is inherently needed to model internal erosion because the continuum and particle scale models have a mutual feedback on each other.

There is currently no multiscale algorithm available for modelling internal erosion in soil mechanics. On the other hand, a great number of applications in soil mechanics could benefit from multiscale models that consider both the particle and continuum scales. Contact erosion

at the interface of core and filter in embankment dams is a case in point. Some of the gaps to reach a multiscale FEM-DEM were addressed in this dissertation. The following points list the main contributions for the three papers:

- A versatile and multipurpose interface was developed between COMSOL Multiphysics (FEM) and YADE (DEM) to allow the coupling of any partial differential equations (PDEs) with a discrete element simulation.
- A YADE interface was programmed to apply drag force on particles based on a coarse-grid method (CGM) using Darcy's law.
- A multiscale computational algorithm was implemented for the interface and YADE script.
- In the multiscale scheme, a partial differential equation was defined in the COMSOL model to verify particle conservation and predict porosity along the geometry for future time steps. A second variable was implemented in the permeability equation (Kozeny-Carman) to update the specific surface based on the new value of porosity.
- The accuracy of drag force values derived from Darcy's law and CGM was evaluated based on FEM results obtained by solving the Navier-Stokes equations at the pore scale.
- An improved method was proposed to predict the drag force on particles as quickly as CGM and accurately as FEM.

The first objective of the study was developing an interface based on a series of Java classes to couple the finite element code COMSOL and the discrete element code YADE. The interface enables researchers to utilize the features of both continuum and DEM approaches. The interface, named ICY, is designed to be versatile for different geoscience applications. The nature of the coupling can vary between applications. For example, the verification and application examples involve very different couplings.

The performance of interface was verified with the classic example of a sphere falling in water according to Stokes' law. It involves the exchange of two variables that are respectively used in a boundary condition in the FEM engine and as a force in the DEM engine. The particle

motion was simulated using YADE. Drag force on the particle was calculated by solving the Navier-Stokes equations in COMSOL. The framework could replicate exactly the analytical results obtained from Stokes' law. The validation test has shown that ICY can be employed to model fluid-particle interaction by solving the Navier-Stokes equations. An advantage of ICY is that COMSOL can be used to solve virtually any partial differential equations and that the data exchanged between COMSOL and YADE can vary between applications. Another advantage of the JAVA interface is that more COMSOL users can use the software without purchasing the COMSOL "Matlab LiveLink" add-on.

The simulation of an internal erosion test presented by Tomlinson and Vaid (2000) is then used as an application example for ICY. The test consists in a small permeameter test that involves two layers of monodisperse spherical glass beads. The motion and the contact forces for particles were calculated by means of YADE, while fluid flow was solved with COMSOL based on the FEM. The example involves the exchange of drag force values that are applied as constants in 5 DEM cells and hydraulic conductivity values that are used in a linear interpolation in the FEM model.

There are currently two pore scale approaches available in YADE for the computation of drag force: the lattice Boltzmann method (Sibille et al., 2015) and pore network flow method (Chareyre et al., 2012). With sub-particle methods, frictional losses are calculated by solving the Navier-Stokes equations at the microscale. The grain arrangement from the DEM simulation is considered explicitly. This causes a significant computational cost even for small-scale particle systems. Therefore, to minimize the computational cost, drag force on particles were calculated using the coarse-grid method. In this method, the fluid cell embraces several particles. Fluid flow derives from average pressures and velocities in several pores in each cell. The microscale grain arrangement is not considered explicitly for coarse-grid methods. This approach can significantly reduce the hydrodynamic computational costs.
In this study, the frictional losses were calculated based on Darcy's law and macroscale permeability values. The relationship between outflow (m^3/s) and hydraulic gradient in the

experimental tests is linear. It means that fluid flow in the permeameter test follows Darcy's law. According to the definition presented by Tsuji et al. (1993), the Reynolds numbers were in the 0.7 to 2 range for the five sections during the simulation. This shows that the flow regime is mostly laminar or transitional. If the flow regime was turbulent, different equations could be used in COMSOL to model the flow. The new equations could be selected using the graphical user interface. It is one of the advantages of ICY with respect to other FEM-DEM interfaces.

Tomlinson & Vaid (2000) present the relationship between the mean permeability value for both the base soil and filter layers, and time elapsed for each test. The mean permeability at the beginning of the test can be compared with the mean permeability in our numerical model. For the application example, the initial permeability for Tomlinson & Vaid (2000) and our numerical model are respectively 0.042 and 0.04 cm/s. It is more difficult to compare the permeability later in the test as our COMSOL model assumes a constant thickness for the base soil layer. This thickness decreases during the Tomlinson & Vaid (2000) tests.

The friction angle between particles and walls is the same as for interactions between particles because the numerical sample is representing the middle part of the experimental specimen. A sensitivity analysis was also performed on the wall friction angle. When the wall friction angle was increased to twice the interparticle friction angle (17.2˚), the total erosion mass was constant at 2.41 g in 48 s.

There was a 100 kPa confining pressure in the experimental test that was not taken into account in the numerical model. It had a negligible effect for the stress range observed in the model that is presented in this paper. However, the confining pressure showed a significant effect for higher confining pressures (higher than 300 kPa) and smaller grain size ratios. Pressure can be applied on the specimen using a wall.

A comparison of the coupled model results with the experimental results confirms that drag force values calculated from pressure gradients derived from the continuum model can be

appropriate for a fully saturated medium. The coarse grid method could be an appropriate substitute for costly pore scale approaches.

The objective of the application test was not to reproduce perfectly the permeameter test, but to show the capabilities of the interface in modelling fluid-particle systems. Nevertheless, there are many parameters that could be modified to have a better calibration. For instance, the effect of boundaries should be studied. The number of particles in the horizontal sections is relatively small. This could lead to preferential paths between particles and walls.

The application test presented in this study is a simplified form of internal erosion that considers fine particles movement only in the vertical direction (z). The influence of drag forces in the lateral directions (x, y) on the particles and the total drag force weighting method need to be investigated with the results obtained from a pore-scale model.

The DEM-FEM model does not constitute a multiscale model, but a hybrid model. The interface can be seen as the first step in the development of a multiscale model for simulating complex geotechnical issues, such as internal erosion, to be studied. The number of particles involved in DEM simulations restrict application of the FEM-DEM model for small laboratory tests due to the heavy computational cost associated with the number of particles.
A multiscale algorithm is first needed to limit the number of particles in the DEM simulation. Hierarchical multiscale algorithm can be used to tackle this problem. In this method, the large-scale model uses the information from a discrete model as an input to model macroscopic behaviour of porous media. However, all existing hierarchical multiscale models aimed at monitoring soil deformation problem and there is no example of hierarchical multiscale DEM-FEM models for the internal erosion in the literature.

Chapter 4 presents a multiscale computational algorithm aimed at stimulating fluid-particle interaction for large scale applications in soil mechanics. The multiscale model in this research was developed based on ICY. It aims at limiting the number of particles in the DEM simulation

and to eventually allow the modelling of internal erosion for large structures. With the multiscale algorithm, smaller DEM subdomains are generated to simulate particle displacements at the microscale. Particles in these small subdomains are subjected to body and contact forces for small time steps. Drag force was applied on each discrete body in YADE based on the same coarse-grid method using Darcy's law. The continuum model uses particle flux distributions from the DEM subdomains to evaluate variations in porosity, hydraulic conductivity and hydraulic gradient for longer time steps and at a larger scale. The updated hydraulic gradients from the continuum model provide the DEM subdomains with updated hydrodynamic forces based on a coarse grid method.

The algorithm was verified by simulating a numerical suffusion test with 5 cm height and by comparing the multiscale model results with numerical results based on a DEM model incorporating the full sample. Five small separat subdomains along the full sample represent the microscopic behaviour of the discrete body. The flux of small particles in the subdomains is solved for a short time-step in YADE. These values were set to the centre of subdomains in a 1D COMSOL model at the beginning of each time-step. The COMSOL model extracts porosity and specific surface for longer time-steps using a particle conservation equation. In the next global time-step, these values are sent to DEM to update the porosity of subdomains. The DEM subdomains represent 100% of the total thickness.

A good agreement was found between the multiscale model and the full-scale model in terms of porosity and flux values. The multiscale model results are depended on magnitude of the COMSOL time-step. It needs to be justified based on the time variation of porosity in the specimen. The performance of the multiscale method was also verified with three subdomains. It avoids generating the full sample as a DEM model. The model with three subdomains could successfully predict the porosity and flux changes through the specimen. However, it was not accurate as the five subdomains model in predicting porosity of the absent subdomains.

For simplicity, HMM developed in this study consider the mass flux of particles solely in vertical direction (z). Particles could not leave the subdomains from lateral boundaries. Drag force on the particles are also assumed to be 1D and parallel to the water flow direction.

The assumption of isotropic hydraulic conductivity values ($K_x = K_y = K_z$) is true for a subdomain with spherical particles. Real soils are anisotropic medium with different K values depending on the direction of water flow through the porous media.

YADE might be unable to update the subdomains for very low porosity values because the code adds new particles in voids without interacting with existing objects. Furthermore, choosing a representative subspecimen will be challenging if particles are non-uniform with different properties.

Another restrictive parameter to use the HMM model for large-scale applications is that the porosity of the lower subdomains was simultaneously affected from the porosity changes of upper subdomains in the COMSOL model. The effect of porosity changes in upper subdomains should be transferred with a delay to lower subdomains.

Discrete element simulations are used increasingly often to model phenomena involving fluid particle interactions, such as liquefaction and internal erosion. Drag is often the main force resulting from fluid-particle interactions with the discrete element method (DEM). Several methods are employed for drag calculations depending on the desired accuracy and the number of particles involved. Solving the Navier-Stokes equations at the pore scale is one of these methods. In geotechnical engineering, the fluid motion is not typically solved at the pore scale for large particle assemblies due to the heavy computational cost. Coarse-grid methods are often used to compute the drag force on particles because of their low computational cost. It involves solving an averaged form of the Navier–Stokes equations at the continuum scale. The total drag force derived from CGM can be applied to the particles proportionally to their volume (CGM-V) or surface (CGM-S).

However, the accuracy of coarse-grid methods has not been systematically studied in the literature. The paper presented in Chapter 5 compares the CGM-V and CGM-S drag forces obtained using coarse-grid methods with finite element (FEM) drag forces obtained by solving the Navier-Stokes equations at the pore scale. Three unit cells (simple cubic, body-centered cubic and face-centered cubic) corresponding to different porosity values (respectively 0.477, 0.319 and 0.259) were simulated. Each unit cell consisted of a fixed coarse-grained skeleton and a smaller particle in the pore space. A large number of simulations (2712) were conducted by changing the position and diameter of a small particle in the unit cells. Our comparison clearly shows that coarse grid methods perform very poorly unless the particle size distribution is relatively uniform.

Another objective of this study was to look at the applicability of ANN for drag force predictions. An Artificial Neural Network (ANN) was trained using the FEM results to predict the drag force on small particles based on their size and their location with respect to the closest large particle and the flow direction. An excellent agreement was obtained between the ANN and FEM results for all particle size ratios and packings. The trained ANN is found to be a promising method to improve the computational efficiency of drag force calculations in polydispersed materials.

6.2 Recommendations

This chapter recommends further lines of research in line with the research presented in this thesis.

- Improve the coupling used for the contact erosion example.

There are some parameters that could be modified to have a better calibration for the application of contact erosion tests in Chapter 3. For example, the drag force could be based on polynomial interpolations of the pressure from COMSOL model instead of being assumed constant average pressure differential in each cell. Another important parameter to investigate is the

effect of boundaries and walls although the influence of the horizontal section was verified to some extent by changing the friction angle for interactions between walls and particles.

- Make the interface more user-friendly.

A new version of the interface, specially designed for internal erosion applications, can be made more user-friendly by bypassing the client-server agent between the JAVA interface and YADE Python interface. Moreover, a graphical user interface (GUI) should be added to make ICY simpler for users who are not experienced in JAVA or Python programming.

We propose to employ simple application examples to extent the interface and show the capabilities of ICY in the modelling of internal erosion. An example could be the modelling of volume and permeability changes in a sand boil or during sand filter backwash as a function of the hydraulic gradient. COMSOL would be used to model the water flow based on a water conservation equation. YADE would be used to model the porous media expansion based on the drag force values calculated from the COMSOL model.

- Compare some erosion modelling results with the continuum-based internal erosion theory first proposed by Vardoulakis et al. (2001).

Numerical models of internal erosion tests could be used to compare the results from our hierarchical multiscale model with results obtained solely with COMSOL and the Vardoulakis et al. (2001) theory. For this example, the interface would combine water and particle conservation equations at the continuum scale, with particle flux calculations at the particle scale.

- Applicability of the ANN drag force for large particles systems.

The ANN trained in Chapter 5 was only applied to predict the drag force on particles through a MATLAB script and it is not yet incorporated in a DEM code. In a DEM simulation, particles

position and consequently the particle size ratios of neighboring particles vary with time. The YADE interface could be extended to update the drag force on each particle using the trained ANN at each time step. The sand boil test, for instance, could be employed to evaluate the ANN performance in comparison with pore-scale simulations (e.g., LBM).

- Embedding machine learning algorithms to improve the contact model efficiency in the DEM simulations.

The calculation of contact forces is the most time consuming part of DEM simulations. It comprises almost 90% of the simulation time (Sutmann, 2002). Calculating the contact forces might be more efficient by including machine learning algorithms in the DEM simulations. A large database needs to be generated by DEM simulations in which contact forces vary regarding the physical and mechanical properties of particles in contact. This would allow the number of particles in simulations to be increased.

APPENDIX I

ICY INSTRUCTION GUIDE

Pouyan Pirnia [a], Francois Duhaime [b], Yannic Ethier [c], Jean-Sebastien Dube [d]

[a, b, c, d] Department of Construction Engineering, École de technologie supérieure, 1100, rue Notre-Dame Ouest, Montréal, Québec, Canada, H3C 1K3

Published in *Computers and Geoscienc,* February 2019

Introduction

ICY is an interface between COMSOL Multiphysics and YADE. COMSOL Multiphysics is a finite element engine. It can model multiple physical phenomena simultaneously, such as flow, seepage, chemical reactions, stress-strain behaviour and heat transfer. Solving coupled systems of partial differential equations (PDEs) is the key feature of COMSOL. YADE ("Yet Another Dynamical Engine") is an open-source discrete-element code. Simulations in YADE are described and controlled by Python scripts. ICY is programmed in a JAVA class.
Two project folders named Verification and Applicationtest are provided by the authors (Pirnia et al., 2018). The Verification project simulates a particle falling in water according to Stokes' law. The Applicationtest project simulates an internal erosion test. The project files include JAVA classes (ICY.java, Clientcaller.java, Reader.java), property files (define.properties), YADE interface scripts (test.py), client-server scripts (client.py and server.py), mesh file (4.mesh), test0.yade (including initial specimen composed of two layers of glass beads: a finer layer on top and a coarser layer at the bottom) and COMSOL models (test.mph). The two projects involve the exchange of different data between COMSOL and YADE. The files that differ for the two projects are test.py (YADE model), test.mph (COMSOL model) and the ICY.java class (main interface code). These are also the file that should be modified to create new applications for the interface.

Prerequisites

YADE and ICY are only executable on Linux operating systems. YADE, COMSOL and a JAVA integrated development environment (IDE) need to be installed before using the interface. The codes have been tested under Linux Ubuntu version 14.04 using YADE version 1.14.1 and COMSOL version 5.2. The JAVA classes were compiled and run using the NetBeans IDE version 8.0.2. The JAVA classes and Python scripts could have to be adapted if ICY is run with

different versions of YADE and COMSOL, and under a different JAVA IDE.

Preparing the COMSOL model

The COMSOL file (test.mph) contains information on geometry, materials, fluid properties, boundary conditions, and mesh. The easiest way to edit these parameters or to define new ones is through COMSOL's graphical user interface (GUI). The main JAVA class can also modify the parameters of the COMSOL model, for example the particle velocity in the verification example, and the permeability values (*k1, k2, k3, k4* or *k5*) or hydraulic head at the top of the specimen (Hupstream) in the application example.

Figures 1-4 show how to prepare the GUI for the application test. Figure A I-1 shows how to define parameters using the COMSOL GUI. For the application example, the initial parameter values are arbitrary as they are controlled by the interface. Figure A I-2 shows how to define the points were the pressure values at the top and bottom of each cell will be obtained from the COMSOL model. Figure A I-3 shows how the relationship between hydraulic conductivity and the z coordinate (x in COMSOL) is defined using the parameters defined in Figure A I-1. The GUI can also be used to modify any other parameter on the FEM side of the model, such as the finite-element mesh (Figure A I-4).

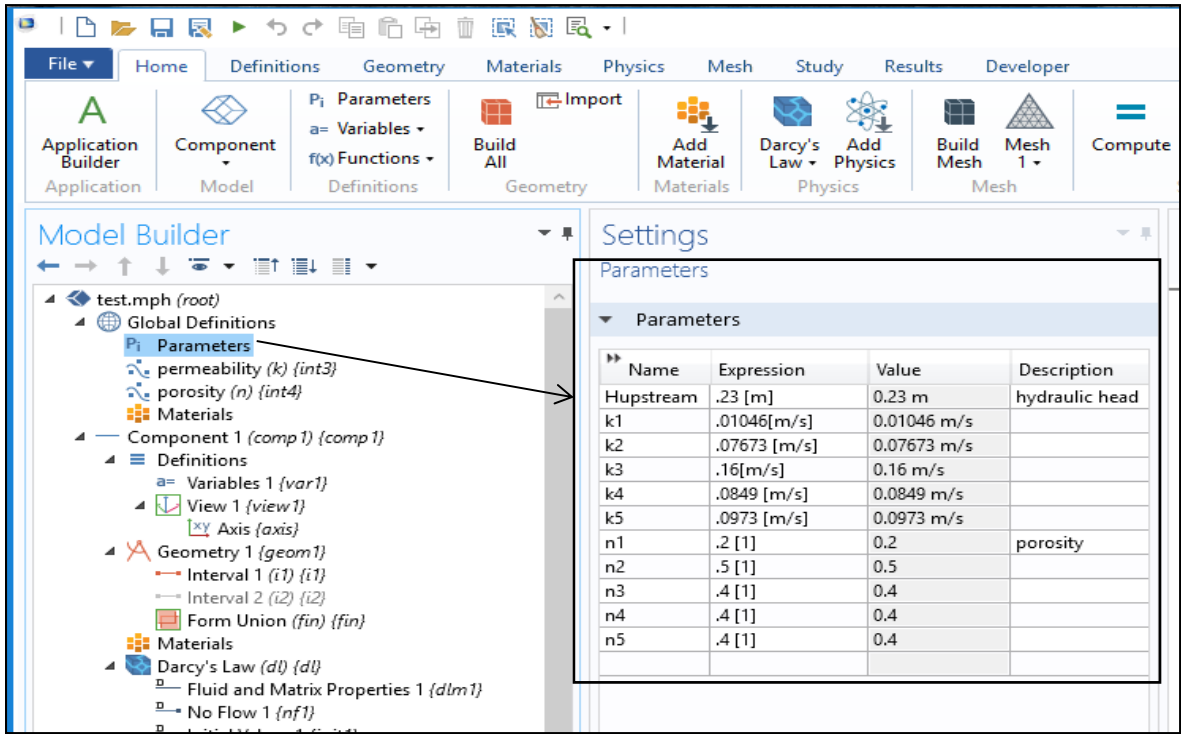


Figure-A I-1 Parameter definition in COMSOL

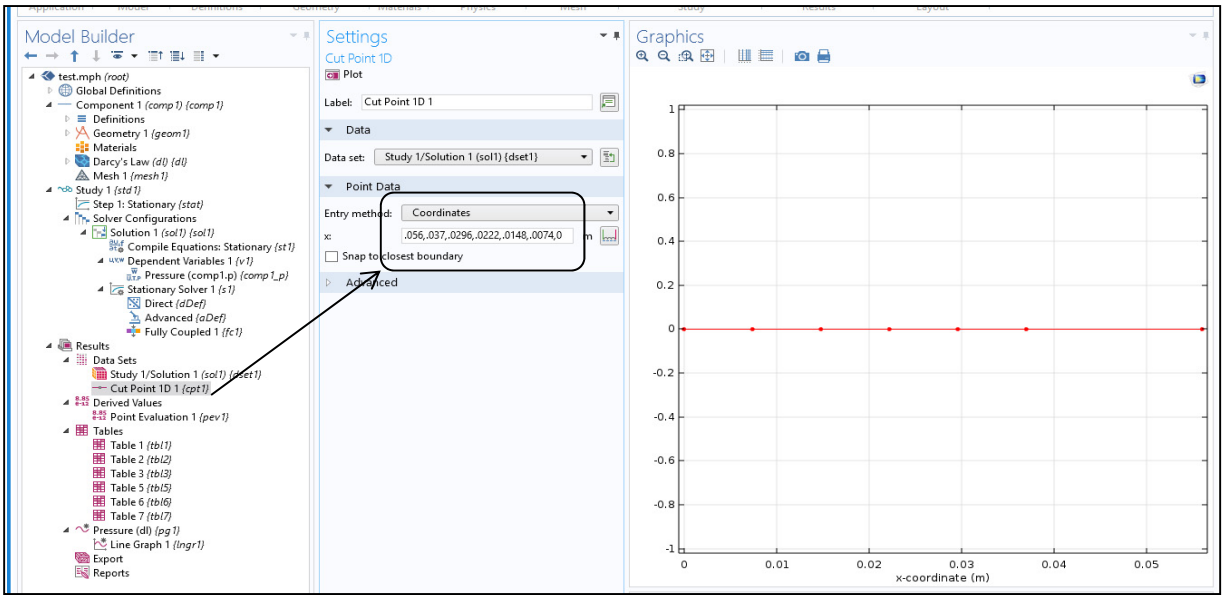


Figure-A I-2 Definition of the points where p values will be obtained

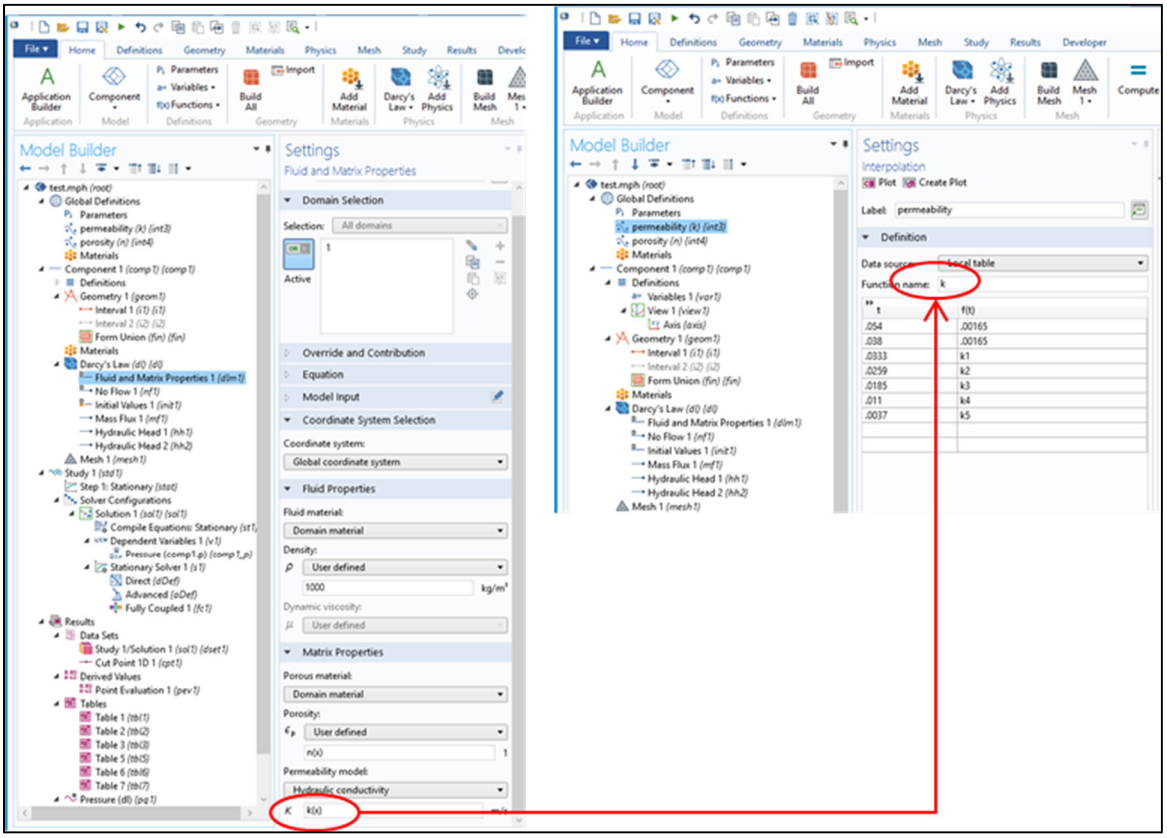


Figure-A I-3 Definition of the hydraulic conductivity function and assignment to the domain

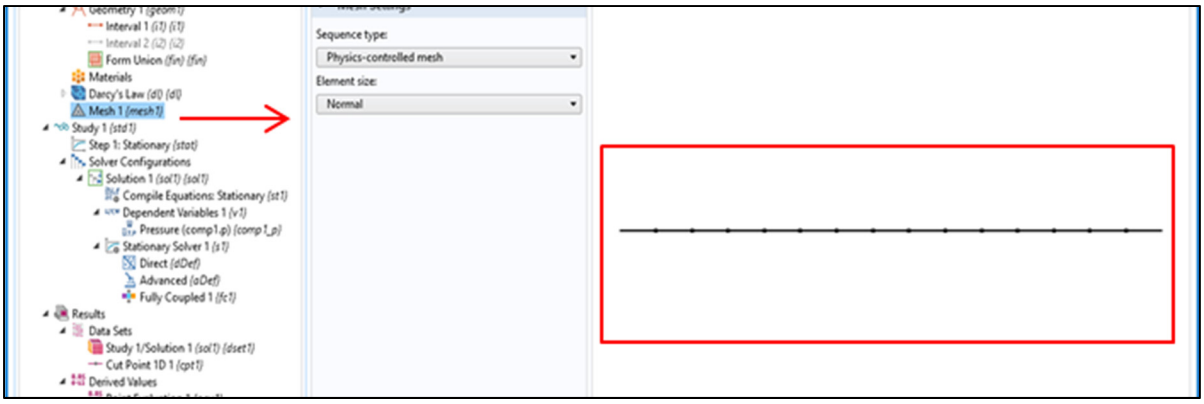
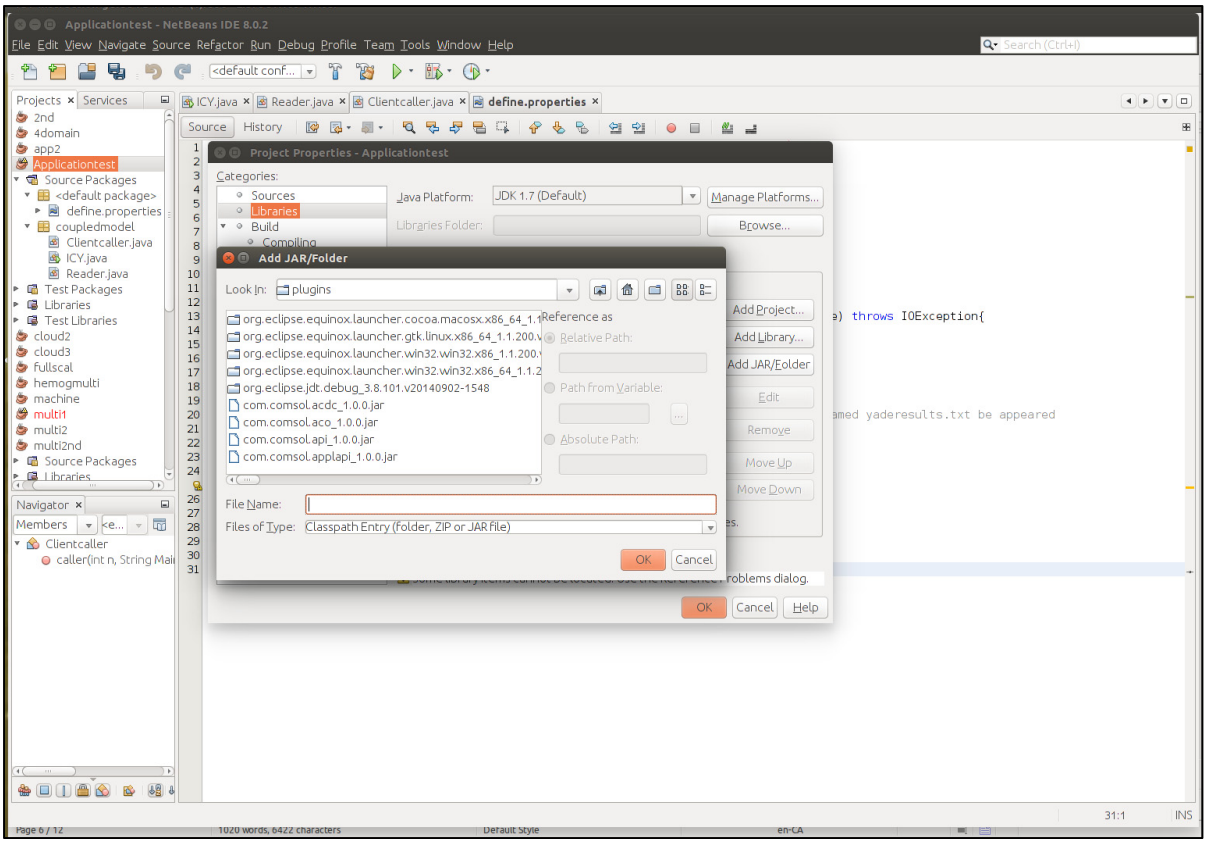


Figure-A I-4 Mesh definition in COMSOL

Preparing the projects in NetBeans

After creating an IDE project folder, the COMSOL plugins have to be added to the project through:

Properties ---> Libraries ---> Add JAR/Folder ---> add all .jar files in plugins folder in COMSOL installation folder



For running the examples, the COMSOL file (test.mph), YADE script (test.py), client-server (client.py and server.py), mesh file (4.mesh), pressure file (pressure.txt including arbitrary initial pressures for the Application test), dragforce.txt (including drag force for the verification test), test0.yade (including initial specimen) and the JAVA source packages (src folder including ICY.java, Clientcaller.java, Reader.java, define.properties) need to be added to the IDE project folder (Figure A I-5).

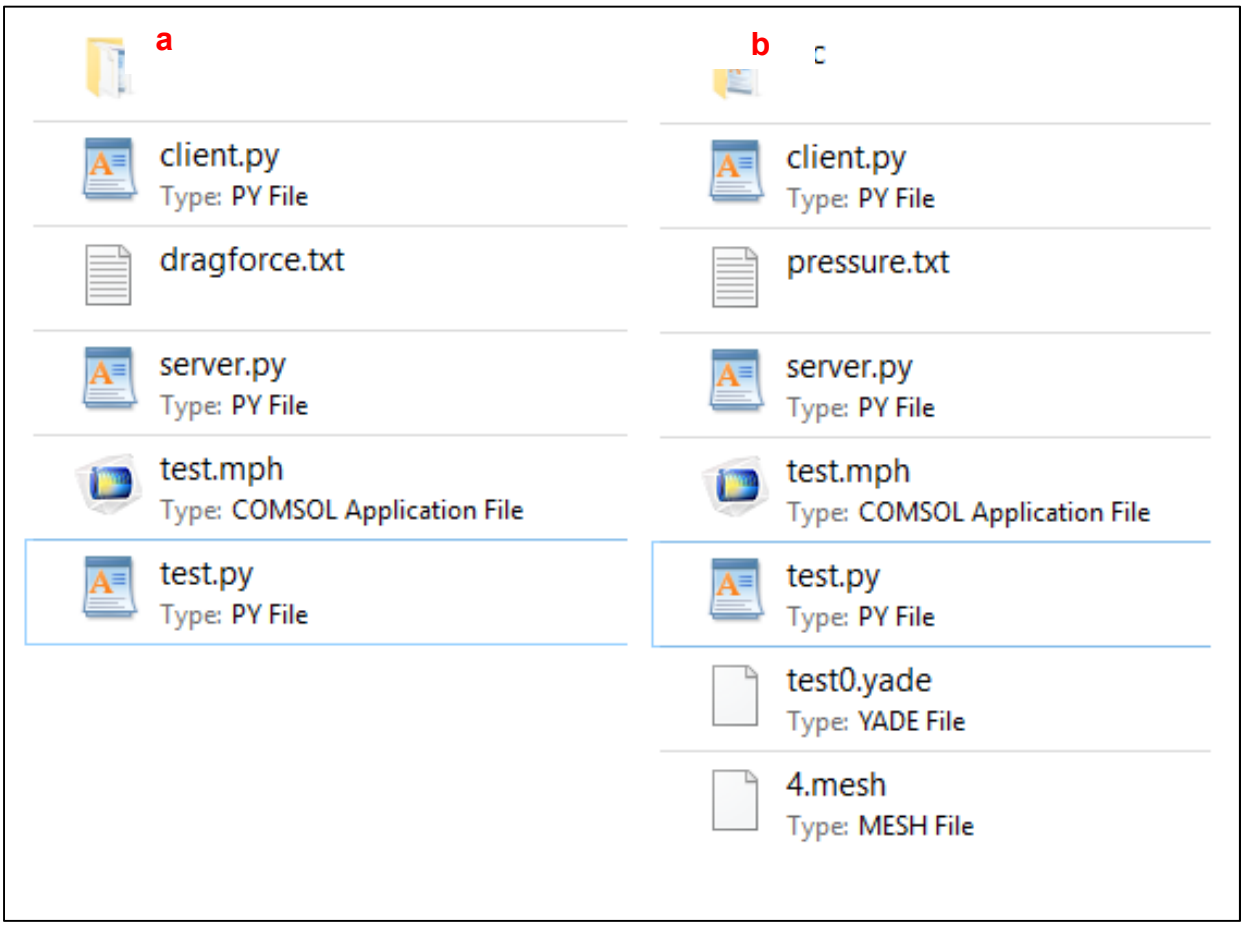


Figure-A I-5 Required files and folder for running the a) Verification example and b) Application test

The project's files ICY.java, Reader.java, Clientcaller.java and define.properties have first to be opened in NetBeans. The directories (MainPath and SavingFolder) in the property file (define.properties) have to be changed to correspond to the project directories on the computer, an example of the Application test was presented in Figure A I-6.

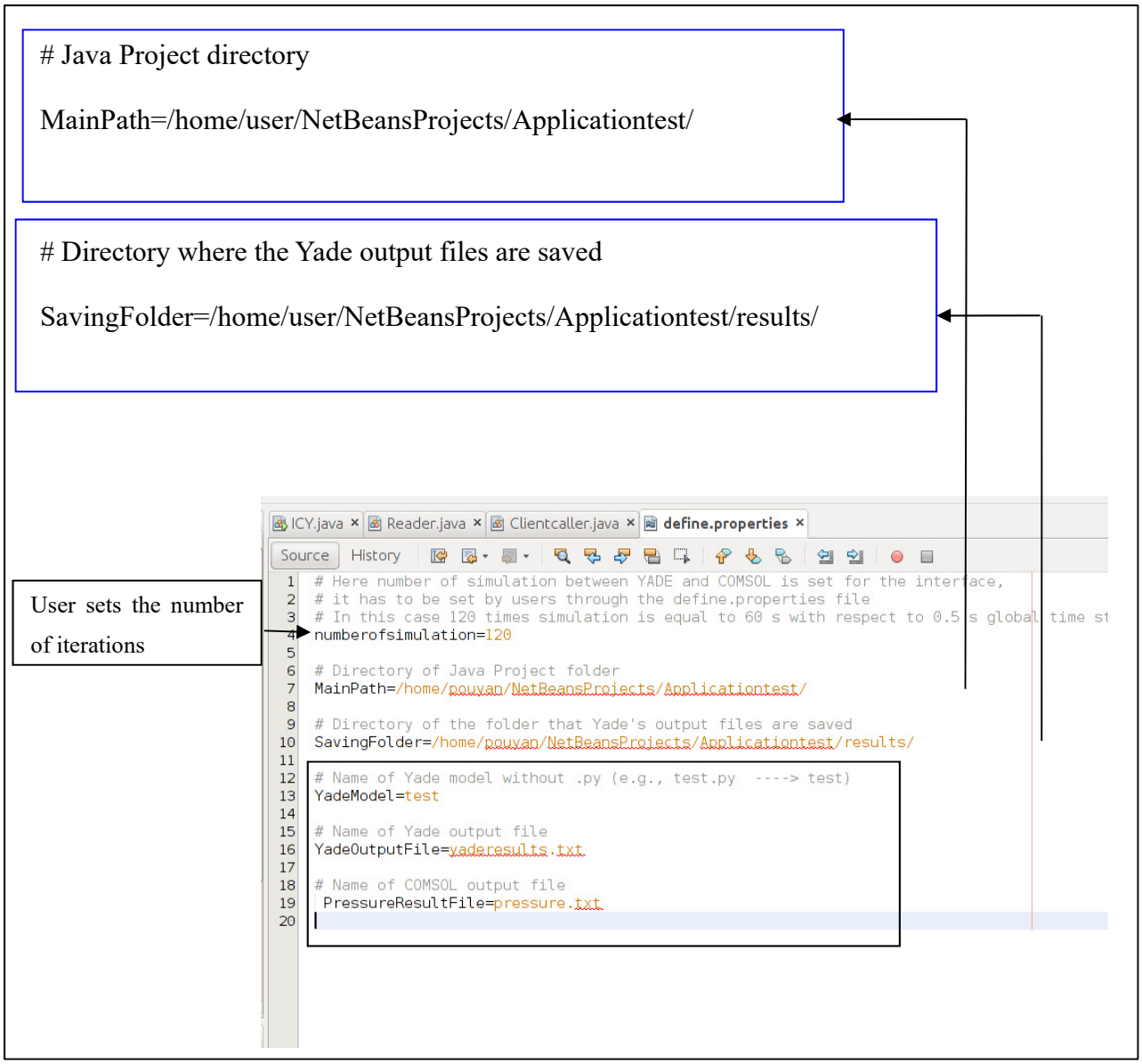


Figure-A I-6 Setting variables and directories in the property file

Users do not need to change anything in Reader.java and clientcaller.java files for the application test. The command lines only need the YADE model and output files names which are taken from the property file (Figure A I-7).

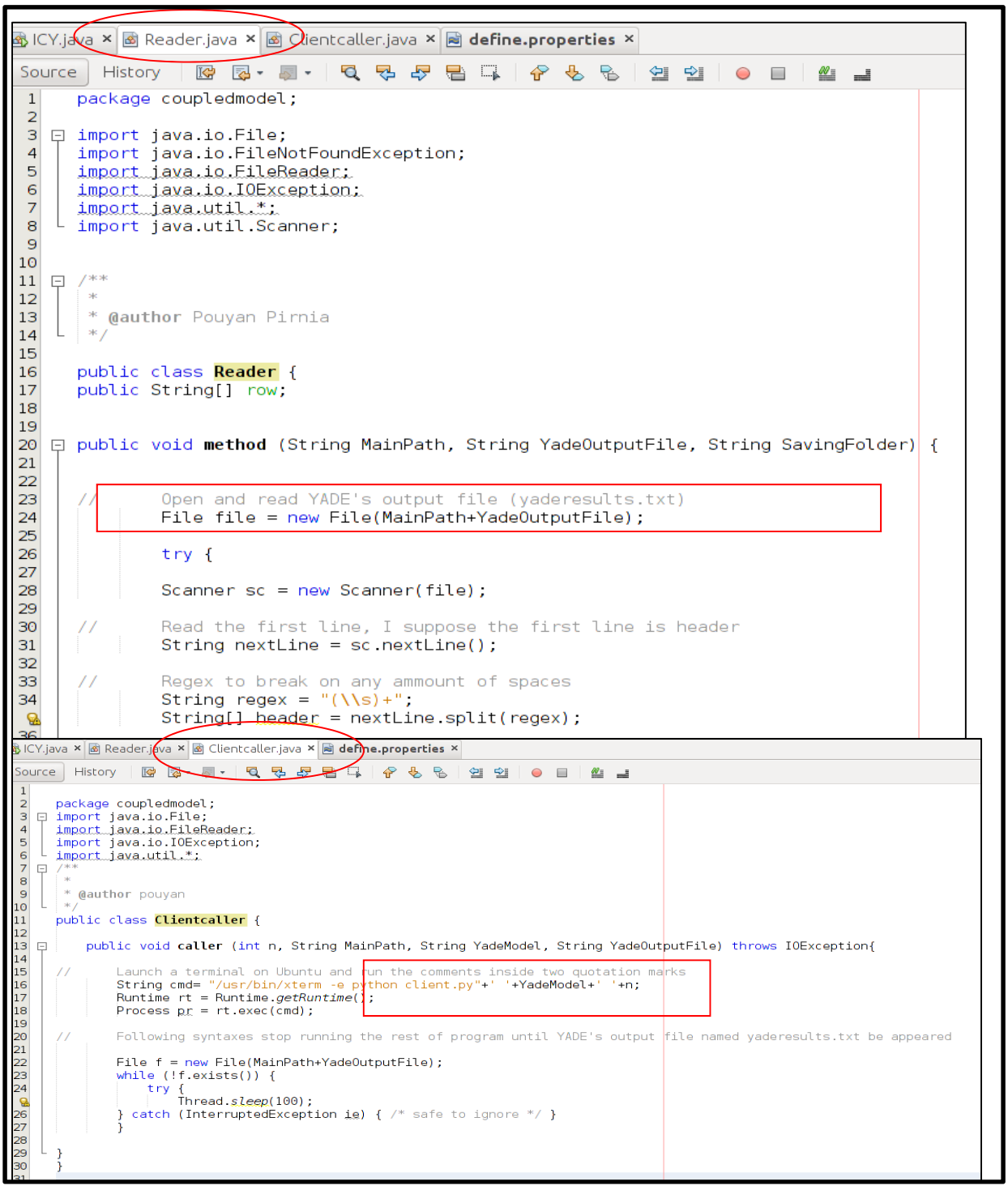


Figure-A I-7 Names in Reader.java and clientcaller.java files taken from the property file

Figure 8 shows the tasks are performed in the ICY.java class for the application test.

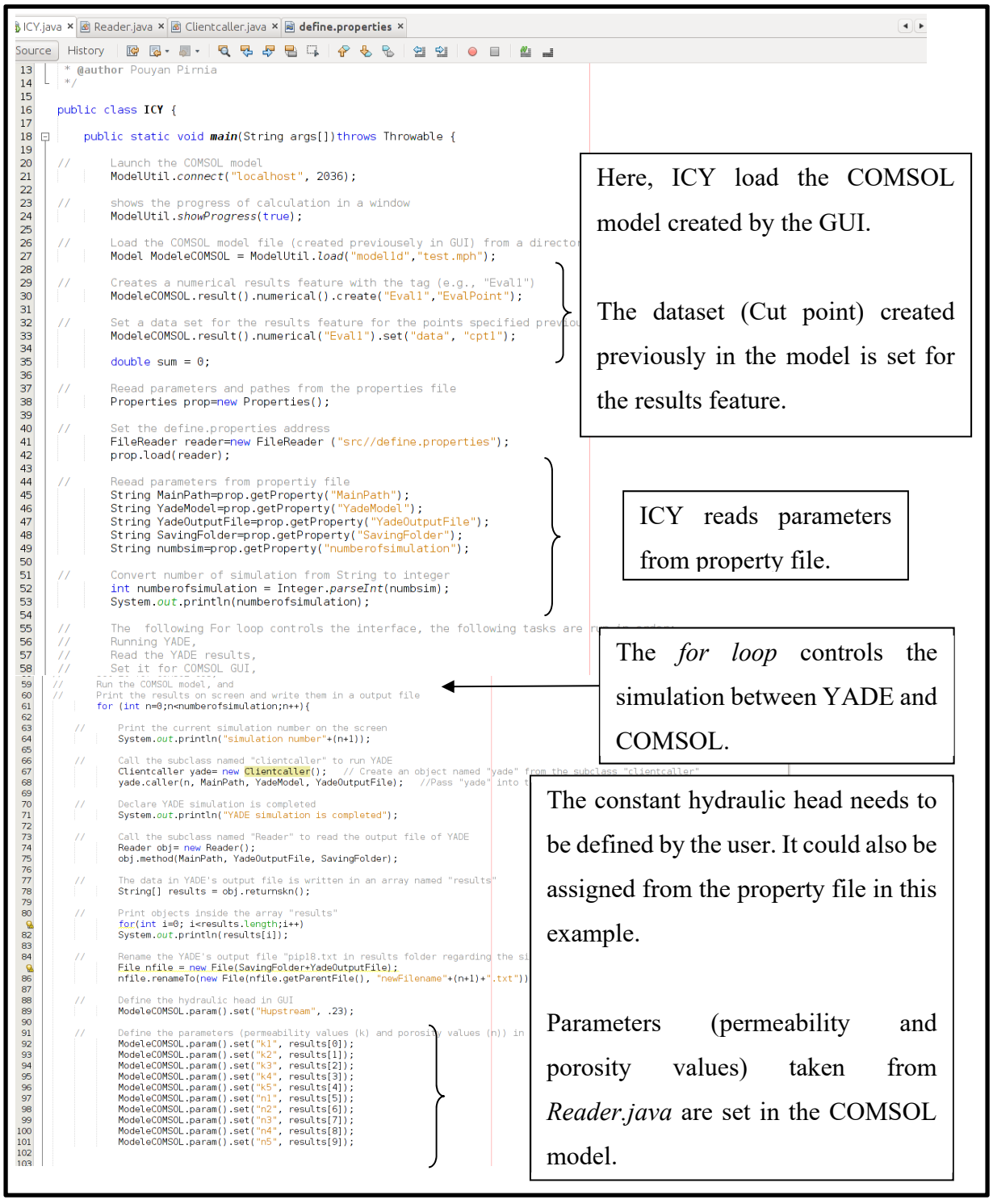


Figure-A I-8 Main tasks in the ICY.java class script

Preparing the YADE script

Parameters could be modified in the YADE script for the application example presented in Figure A I-9.

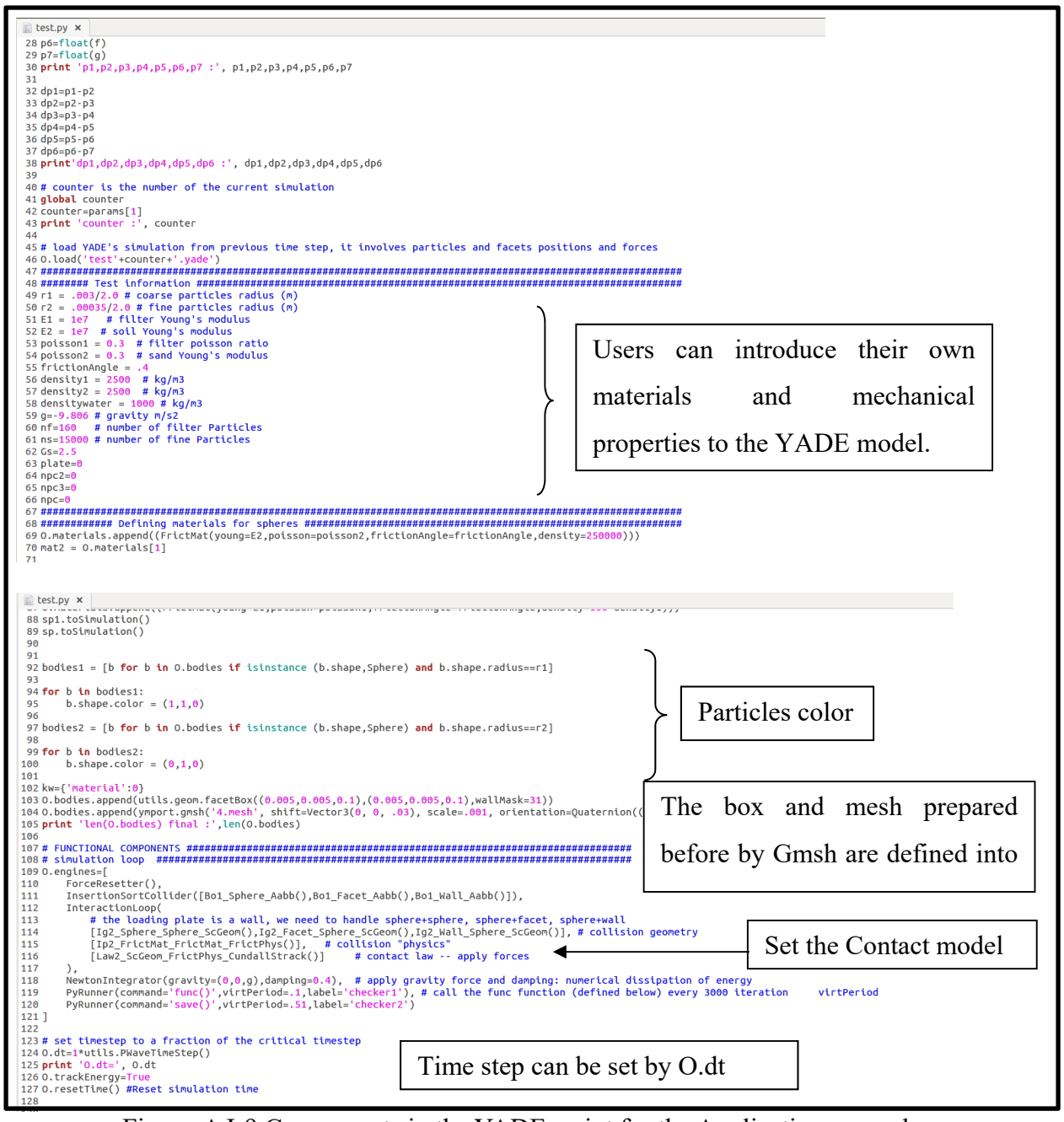


Figure-A I-9 Components in the YADE script for the Application example

Running the interface

Before compiling ICY in NetBeans, the COMSOL server and the python client-server need to be launched. To start the COMSOL server manually, a terminal window is opened and the following command is typed in the COMSOL installation directory:

$. / comsol mphserver

For connecting the client to the server, a second terminal window is opened. The following command is typed in the directory containing the server.py file (MainPath directory):

$ python server.py

At this point, ICY can be compiled and run. The simulation progress is printed step by step on the NetBeans screen. It lets users follow the YADE and COMSOL outputs during the simulation.

APPENDIX II

VERIFICATION CODES

<JAVA codes>

<define.properties>

```
# Here number of simulation between YADE and COMSOL is set for the interface,
# it has to be set by users through the define.properties file
# In this case 120 times simulation is equal to 60 s with respect to 0.5 s global time step in YADE
numberofsimulation=30

# Directory of Java Project folder
MainPath=/home/pouyan/NetBeansProjects/verification/

# Directory of the folder that Yade's output files are saved
SavingFolder=/home/pouyan/NetBeansProjects/verification/results

# Name of Yade model without ".py" (e.g., test.py  ----> test)
YadeModel=test

# Name of Yade's output file
YadeOutputFile=yaderesults.txt

# Name of COMSOL outout file
DragForce=dragforce.txt

# lastvel.txt is a text file that keep particle's velocity from previous time step, this file is generated by the interface
# This value is used by Yade for next time step, in fact, the velocity taken from this file is applied as the initial velocity of falling sphere at each time step
 LastVelocityFile=lastvel.txt

```

<ICY.java>

```
1   package coupledmodel;
2   import com.comsol.model.Model;
3   import com.comsol.model.util.ModelUtil;
4   import java.io.*;
5   import static java.lang.System.*;
6   import java.util.Arrays;
7   import java.util.Properties;
8   import java.util.Scanner;
9   import java.util.concurrent.TimeUnit;
10
11  /**
12   *
13   * @author Pouyan Pirnia 2017
14   * This is the main class of iCY. It calls two other classes named "clientcaller" and
     "Reader".
15   * It controls both YADE and COMSOL models.
16   */
17
18  public class ICY {
19
20
21      public static void main(String args[])throws Throwable {
22
23  //      Connect to a previously launched COMSOL server
24          ModelUtil.connect("localhost", 2036);
25
26  //      Shows the calculation progress in a window
27          ModelUtil.showProgress(true);
28
29  //      Load the COMSOL model file (created previously in GUI) using its directory and
    name it (e.g., test.mph)
30          Model ModeleCOMSOL = ModelUtil.load("model3d","test.mph");
31
32  //      Create a numerical results feature with a tag (e.g., "Eval1")
33          ModeleCOMSOL.result().numerical().create("Eval1","EvalGlobal");
34
35  //      Define a dataset for the results feature
36          ModeleCOMSOL.result().numerical("Eval1").set("data","dset1");
37
38  //      Read parameters and paths from property
    file
39          Properties prop=new Properties();
40
41  //      Set the define.properties address
42          FileReader reader=new FileReader ("src//define.properties");
43          prop.load(reader);
44
45  //      Read parameters from the property file
46          String MainPath=prop.getProperty("MainPath");
47          String YadeModel=prop.getProperty("YadeModel");
48          String YadeOutputFile=prop.getProperty("YadeOutputFile");
49          String SavingFolder=prop.getProperty("SavingFolder");
50          String LastVelocityFile=prop.getProperty("LastVelocityFile");
51          String DragForce=prop.getProperty("DragForce");
52          String numbsim=prop.getProperty("numberofsimulation");
53
54  //      Convert number of simulation from String to integer
55          int numberofsimulation = Integer.parseInt(numbsim);
56          System.out.println(numberofsimulation);
57
58
59  //      The  following For loop controls the interface, the following tasks are run in
    order:
60  //      Running YADE,
61  //      Reading the YADE results,
62  //      Sending the YADE results to COMSOL,
63  //      Running the COMSOL model, and
64  //      Printing the results on screen and write them in a output file
65          for (int n=0;n<numberofsimulation;n++){
```

```
66
67  //      Print the current simulation number on the screen
68          System.out.println("simulation number"+(n+1));
69
70  //      Call the subclass named "clientcaller" to run YADE
71          Clientcaller yade= new Clientcaller();   // Create an object named "yade" from
            the subclass "clientcaller"
72          yade.caller(MainPath, YadeModel, YadeOutputFile);   //Pass "yade" into the
            method "caller"
73
74  //      Declare YADE simulation is completed
75          System.out.println("velserie.txt is written");
76
77
78  //      Call the subclass named "Reader" to read the output file of YADE
79          Reader obj= new Reader();
80          obj.method(MainPath, YadeOutputFile, SavingFolder);
81
82  //      Print the last velocity on screen
83          System.out.println("This is lastvel :"+obj.returnsvel());
84
85  //      The data in YADE's output file is written in an array named "results"
86          String[] results = obj.returnsvel();
87
88
89  //      Print objects inside the array "results"
90          for(int i=0; i<results.length;i++)
91          System.out.println(results[i]);
92
93          String lastvel = results[2];
94
95  //      Write the last velocity in a text file named lastvel. This value is used by
    Yade for the next time step This velocity is applied as the initial velocity of falling
    sphere
96          try (PrintWriter out = new PrintWriter(MainPath+LastVelocityFile)) {
97              out.write(lastvel);
98          }
99              System.out.println("lastvel.txt is written");
100
101 //      Rename YADE's output file "velserie.txt" in results folder with a name
    including the simulation number
102         File nfile = new File(SavingFolder+LastVelocityFile);
103         nfile.renameTo(new File(nfile.getParentFile(),
            "newFilename"+(n+1)+".txt"));
104
105 //      Define the velocity in the GUI
106         ModeleCOMSOL.param().set("Velocity", lastvel);
107
108 //      Run the COMSOL model
109         ModeleCOMSOL.sol("sol1").run();
110
111 //      Integrate Drag force around the sphere in water flow direction (the total
    stress in the y-direction).
112         ModeleCOMSOL.result().numerical("eval1").set("expr", "intop1(spf.T_stressy)");
113
114 //      Save the COMSOL model under a name that includes the simulation number
115         ModeleCOMSOL.save("simulation"+(n+1)+".mph");
116
117 //      Write the COMSOL model results in a text file named "dragforce.txt"
118         double [][] data;
119         data = ModeleCOMSOL.result().numerical("eval1").getReal();
120
121         try{
122             File fac=new File(MainPath+DragForce);
123
124             FileWriter pw=new FileWriter(fac);
125             for (int i = 0; i<data.length; i++){
126             for(int j = 0; j<data[0].length; j++){
127                     pw.write(data[i][j] + ",");
```

```
128                     pw.close();
129                     System.out.println("dragforce is written:"+data[i][j]);
130                     System.out.println("#####################################");
131                 }
132                 }
133
134         }catch (IOException e){
135             out.println("error");
136         }
137
138         }
139
140     ModelUtil.showProgress(false);
141     System.out.println("Simulation is finished");
142     System.exit(0);
143     }
144
145 }
146
147
148
```

<Reader.java>

```
1   package coupledmodel;
2
3   import java.io.File;
4   import java.io.FileNotFoundException;
5   import java.util.Scanner;
6
7   /**
8    *
9    * @author Pouyan Pirnia 2017
10   * This subclass reads YADE's output file (yaderesults.txt) and save it in an array.
     YADE's output file is then moved to a folder named results.
11   */
12
13  public class Reader {
14  public String[] row;
15
16  public void method (String MainPath, String YadeOutputFile, String SavingFolder) {
17
18  //      Open and read YADE's output file (yaderesults.txt)
19          File file = new File(MainPath+YadeOutputFile);
20          try {
21
22          Scanner sc = new Scanner(file);
23
24  //      Move to the second line assuming the first line contains the header.
25          String nextLine = sc.nextLine();
26
27  //      Break on any amount of spaces
28          String reg = "(\\s)+";
29
30  //      This is printing all columns, you can access each column from row using the
    array indexes, example row [0], row [1], row [2]...
31          while (sc.hasNext()) {
32              row = sc.nextLine().split(reg);
33          }
34          sc.close();
35          }
36          catch (FileNotFoundException e) {
37          }
38
39  //      Move YADE's output file "yaderesults.txt" from the first directory to a
    specific folder "results" and save it there
40          try{
41
42              if(file.renameTo(new File(SavingFolder+ file.getName()))){
43              System.out.println("File was moved successfully!");
44              }
45             else{
46              System.out.println("File could not be moved!");
47              }
48
49          }catch(Exception e){
50              e.printStackTrace();
51          }
52
53  }
54
55
56  public String[] returnsvel(){
57
58  //      Save the results in an array
59          return row;
60      }
61
62  }
63
64
```

<Clientcaller.java>

```
1  package coupledmodel;
2  import java.io.File;
3  import java.io.IOException;
4
5  /**
6   *
7   * @author Pouyan Pirnia 2017
8   * This subclass runs the YADE script using the client-server as an agent
9   */
10
11 public class Clientcaller {
12
13     public void caller (String MainPath, String YadeModel, String YadeOutputFile)
       throws IOException{
14
15 //      Launch a terminal on Ubuntu and run the command inside the quotation marks and
   in the YadeModel string
16         String cmd= "/usr/bin/xterm -e python client.py"+' '+YadeModel;
17         Runtime rt = Runtime.getRuntime();
18         Process pr = rt.exec(cmd);
19
20 //      Following syntaxes stop running the rest of program until YADE's output file
   named yaderesults.txt appears
21         File f = new File(MainPath+YadeOutputFile);
22         while (!f.exists()) {
23             try {
24                 Thread.sleep(100);
25             } catch (InterruptedException ie) { /* safe to ignore */ }
26         }
27
28
29 }
30 }
31
```

<YADE Python script – test.py>

```
import re
from yade import pack, plot, geom, utils

# DATA COMPONENTS
################################################################################
######## Receive data from
server#########################################################################

params= sys.argv[1:] #get everything after the test3.py

fd= params[0]   #put the drag force value in fd
t=fd.split(",")  # remove ","
fdz=float(t[0])
fdz=-1*fdz
print fdz

vel=params[1]  #put the velocity value in vel
zvel=float(vel)
print zvel

################################################################################################
##################
######## Test information
#############################################################################

E1 = 1e9   #Young's modulus (Pa)
poisson1 = 0.3   # Poison ratio
frictionAngle = 0.07  #Friction angle
density1 = 2650   #sphere density  kg/m3
densitywater = 1000   #water density
g = 9.80665   #gravity m/s2
r=.00005   #radius of sphere (m)
d=.0001    #diameter of sphere (m)

################################################################################################
##################
############ Defining materials for spheres
#################################################################

O.materials.append(FrictMat(young=E1,poisson=poisson1,frictionAngle=frictionAngle,density
=density1, label='mat_spheres'))

O.bodies.append(utils.sphere([2,0.01,10],.00005,color=[1,1,0], material='mat_spheres'))

# FUNCTIONAL COMPONENTS
################################################################################
# simulation loop
################################################################################

O.engines=[
    ForceResetter(),
    InsertionSortCollider([Bo1_Sphere_Aabb(),Bo1_Facet_Aabb()]),
    InteractionLoop(
        [Ig2_Sphere_Sphere_ScGeom(),Ig2_Facet_Sphere_ScGeom()],
        [Ip2_FrictMat_FrictMat_FrictPhys()],
        [Law2_ScGeom_FrictPhys_CundallStrack()]

    ),
    GravityEngine(gravity=(0,0,-9.80665)),   # apply gravity force to particles
    NewtonIntegrator(damping=0.509),    # damping: numerical dissipation of energy
    PyRunner(command='velocity()',realPeriod=.5),   # call the velocity() function
    (defined below) every .5 second
    PyRunner(command='myAddPlotData()',iterPeriod=1) # call function 'myAddPlotData()'
    every 1 iteration
]
```

```
O.dt=.001   # set time step

O.bodies[0].state.vel=(0,0,zvel)   # apply velocity (calculated in the last iteration of previous simulation) on sphere

bz=(4.0/3.0*3.1416*r**3)*densitywater*g   # buoyancy force

O.forces.addF(0,(0,0,bz+fdz),1)   # apply buoyancy and drag forces on the sphere

w=(4.0/3.0*3.1416*r**3)*density1*g  # weight of sphere
print 'w :', w

F=w-(bz+fdz)   # the balance force
print 'F :', F

def velocity():
    plot.saveDataTxt('yaderesults.txt')  # save results in a text file, this is Yade's outputFile
    O.pause()

def myAddPlotData():
    if O.iter<2:
        plot.addData(t=O.time,F=O.forces.f(0).norm(),Vz=O.bodies[0].state.vel[2],
        Position=O.bodies[0].state.pos[2])  #selected parameters will be written on the text file

plot.plots={'t':('F','Vz')}
plot.plot()
O.saveTmp()
O.run()
O.wait()
quit()

```

<client-server >

Client.py

```
1  import socket
2  import sys
3
4  HOST, PORT = "localhost", 64000  # Dynamic or Private Port number
5  data = " ".join(sys.argv[1:]) # Get the script argument (YadeModel) ["/usr/bin/xterm
   -hold -e python client.py"+' '+YadeModel]. YadeModel is named "test" in the
   define.properties file.
6
7  # Create a socket (SOCK_STREAM means a TCP socket)
8  sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
9
10 try:
11     # Connect to server and send data
12     sock.connect((HOST, PORT))
13     sock.sendall(data + "\n")
14
15     # Receive data from the server and shut down
16     received = sock.recv(1024)
17 finally:
18     sock.close()
19
20 print "Sent:     {}".format(data)
21 print "Received: {}".format(received)
22
```

<Server.py>

```
import SocketServer
import subprocess
import os.path
import signal
import re

class MyTCPHandler(SocketServer.BaseRequestHandler):

    def handle(self):

        self.data = self.request.recv(1024).strip()  # receive the string that was sent by client and remove the extra whitepace from front and back of the string (e.g., test)

        params = self.data.split(' ')    # split the string by space ("test")

        function = params[0]    # put the first parameter in function (function = "test")

        with open('dragforce.txt','r') as f:   #open COMSOL's output file (dragforce.txt) and read it
            fd=f.read(30)
            f.close()

        with open('lastvel.txt','r') as f:   # open lastvel.txt which is generated by the interface, it gives the particle velocity from previous time step
            vel=f.read(15)
            f.close()

        if function == "test":  # if Yade's script (test.py) and Yade's output file (velserie.txt), drag force value from COMSOL (fd) and particle's velocity (vel) are available go to the function doTest
            self.doTest('test.py', 'yaderesults.txt', fd, vel)

        else:
            self.request.sendall("ERROR: Function not found")  # otherwise show the message "Function not found"


    def doTest(self, testFile, outputFile, fd, vel):
        try:

            paramsString=fd.rstrip()   # remove the white space at the end of the string. Without this syntax, the next parameter is written on the next line.

            command = 'python yade -j8 -x '+testFile+' '+paramsString+' '+vel   # this is the command line for terminal to run Yade's script (test3.py)

            print 'command: '+command

            process = subprocess.Popen(command.strip(), shell=True)   # run the command.strip() or open the process

            while not os.path.isfile(outputFile):    # return true if path is an existing regular file
                pass

            content = ''   # content is defined as an empty string

            while not self.isNotEmptyContent(content):    # Open and read the outputFile if the outputFile is not empty based on the function isNotEmptyContent
                with open(outputFile, 'r') as content_file:
                    content = content_file.read()
                    content_file.close()

            os.kill(process.pid, signal.SIGUSR1)    # kill the process
#            os.remove(outputFile)
```

```
55
56              self.request.sendall(content)
57
58          except Exception as e:    # report a message "ERROR" on terminal if doTest is
            not completed
59              self.request.sendall('ERROR: '+str(e))
60
61
62      def isNotEmptyContent(self, content):
63          return re.search("\w+", content)   # searching in the file for defined
            character in content
64
65  if __name__ == "__main__":
66      HOST, PORT = "localhost", 64000
67      server = SocketServer.TCPServer((HOST, PORT), MyTCPHandler)  # create the server,
        binding to localhost on port 64000
68      server.serve_forever()   # activate the server
69
```

APPENDIX III

APPLICATION EXAMPLE CODES

JAVA codes

<define.Properties>

```
# Here number of simulation between YADE and COMSOL is set for the interface,
# it has to be set by users through the define.properties file
# In this case 120 times simulation is equal to 60 s with respect to 0.5 s global time step in YADE
numberofsimulation=120

# Directory of Java Project folder
MainPath=/home/pouyan/NetBeansProjects/Applicationtest/

# Directory of the folder that Yade's output files are saved
SavingFolder=/home/pouyan/NetBeansProjects/Applicationtest/results/

# Name of Yade model without .py (e.g., test.py  ----> test)
YadeModel=test

# Name of Yade output file
YadeOutputFile=yaderesults.txt

# Name of COMSOL output file
 PressureResultFile=pressure.txt

```

<ICY.java>

```java
package coupledmodel;
import com.comsol.model.Model;
import com.comsol.model.util.ModelUtil;
import java.io.*;
import static java.lang.System.*;
import java.util.Arrays;
import java.util.*;
import java.util.Scanner;
import java.util.concurrent.TimeUnit;

/**
 *
 * @author Pouyan Pirnia
 */

public class ICY {

    public static void main(String args[])throws Throwable {

//      Launch the COMSOL model
        ModelUtil.connect("localhost", 2036);

//      shows the progress of calculation in a window
        ModelUtil.showProgress(true);

//      Load the COMSOL model file (created previousely in GUI) from a directory and
names it (e.g., piping.mph)
        Model ModeleCOMSOL = ModelUtil.load("model1d","test.mph");

//      Creates a numerical results feature with the tag (e.g., "Eval1")
        ModeleCOMSOL.result().numerical().create("Eval1","EvalPoint");

//      Set a data set for the results feature for the points specified previously in
GUI with the tag cpt1 (1D point evaluation)
        ModeleCOMSOL.result().numerical("Eval1").set("data", "cpt1");

        double sum = 0;

//      Reead parameters and pathes from the properties
file
        Properties prop=new Properties();

//      Set the define.properties address
        FileReader reader=new FileReader ("src//define.properties");
        prop.load(reader);

//      Reead parameters from propertiy file
        String MainPath=prop.getProperty("MainPath");
        String YadeModel=prop.getProperty("YadeModel");
        String YadeOutputFile=prop.getProperty("YadeOutputFile");
        String SavingFolder=prop.getProperty("SavingFolder");
        String numbsim=prop.getProperty("numberofsimulation");

//      Convert number of simulation from String to integer
        int numberofsimulation = Integer.parseInt(numbsim);
        System.out.println(numberofsimulation);

//      The  following For loop controls the interface, the following tasks are run in
order:
//      Running YADE,
//      Read the YADE results,
//      Set it for COMSOL GUI,
//      Run the COMSOL model, and
//      Print the results on screen and write them in a output file
        for (int n=0;n<numberofsimulation;n++){

    //      Print the current simulation number on the screen
            System.out.println("simulation number"+(n+1));

```

```
66      //      Call the subclass named "clientcaller" to run YADE
67              Clientcaller yade= new Clientcaller();   // Create an object named "yade"
                from the subclass "clientcaller"
68              yade.caller(n, MainPath, YadeModel, YadeOutputFile);   //Pass "yade" into
                the method "caller" with the current number of simulation (n) and the
                property file info
69
70      //      Declare YADE simulation is completed
71              System.out.println("YADE simulation is completed");
72
73      //      Call the subclass named "Reader" to read the output file of YADE
74              Reader obj= new Reader();
75              obj.method(MainPath, YadeOutputFile, SavingFolder);
76
77      //      The data in YADE's output file is written in an array named "results"
78              String[] results = obj.returnskn();
79
80      //      Print objects inside the array "results"
81              for(int i=0; i<results.length;i++)
82              System.out.println(results[i]);
83
84      //      Rename the YADE's output file "pip18.txt in results folder regarding the
        simulation number
85              File nfile = new File(SavingFolder+YadeOutputFile);
86              nfile.renameTo(new File(nfile.getParentFile(),
                "newFilename"+(n+1)+".txt"));
87
88      //      Define the hydraulic head in GUI
89              ModeleCOMSOL.param().set("Hupstream", .23);
90
91      //      Define the parameters (permeability values (k) and porosity values (n)) in
        GUI
92              ModeleCOMSOL.param().set("k1", results[0]);
93              ModeleCOMSOL.param().set("k2", results[1]);
94              ModeleCOMSOL.param().set("k3", results[2]);
95              ModeleCOMSOL.param().set("k4", results[3]);
96              ModeleCOMSOL.param().set("k5", results[4]);
97              ModeleCOMSOL.param().set("n1", results[5]);
98              ModeleCOMSOL.param().set("n2", results[6]);
99              ModeleCOMSOL.param().set("n3", results[7]);
100             ModeleCOMSOL.param().set("n4", results[8]);
101             ModeleCOMSOL.param().set("n5", results[9]);
102
103
104     //      The object number 10 in the results array is number of eroded particles
        accumulated in the container at the end of each global time step
105             String aString=results[10];
106             System.out.println("string result10: "+aString);
107
108     //      convert string to double
109             double aDouble = Double.parseDouble(aString);
110             System.out.println(aDouble);
111
112     //      sum the new amount (aDouble) with the previous value (sum)
113             sum=sum+aDouble;
114             System.out.println("number of eroded particles total: "+sum);
115             System.out.println("cumulative weight of eroded particles total:
                "+sum*0.000000056123);
116
117     //      Run the COMSOL model
118             ModeleCOMSOL.sol("sol1").run();
119             System.out.println("Running COMSOL");
120
121     //      Define the pressure expression for "Eval1"
122             ModeleCOMSOL.result().numerical("Eval1").set("expr", "p");
123
124
125     //      Write the COMSOL model results on a text file named "pressure.txt"
126             double [][] data;
```

```
127             data = ModeleCOMSOL.result().numerical("Eval1").getReal();
128
129             int rows = data.length;
130             System.out.println("number of pressur rows: "+rows);
131             int cols = data[0].length;
132             System.out.println("number of pressure columns: "+cols);
133
134     //      Setting pressure file directory from define.properties
135             String
            PressureFilePath=prop.getProperty("MainPath")+prop.getProperty("PressureResul
            tFile");
136
137             try{
138                 File fac=new File(PressureFilePath);
139
140                 FileWriter pw=new FileWriter(fac);
141
142                 for (int i = 0; i<data.length; i++){
143                     for(int j = 0; j<data[0].length; j++){
144                         pw.write(data[i][j] + ",");
145                         System.out.println("pressure in level"+(i+1)+" is
                            :"+data[i][j]);
146                     }
147                 }
148
149             pw.close();
150             System.out.println("####################################");
151
152
153             }catch (IOException e){
154                 out.println("error");
155             }
156
157         }
158
159     ModelUtil.showProgress(false);
160     System.out.println("Simulation is finished");
161     System.exit(0);
162     }
163
164 }
165
166
167
```

<Clientcaller.java>

```
1
2   package coupledmodel;
3   import java.io.File;
4   import java.io.FileReader;
5   import java.io.IOException;
6   import java.util.*;
7   /**
8    *
9    * @author pouyan
10   */
11  public class Clientcaller {
12
13      public void caller (int n, String MainPath, String YadeModel, String
        YadeOutputFile) throws IOException{
14
15  //      Launch a terminal on Ubuntu and run the comments inside two quotation marks
16          String cmd= "/usr/bin/xterm -e python client.py"+' '+YadeModel+' '+n;
17          Runtime rt = Runtime.getRuntime();
18          Process pr = rt.exec(cmd);
19
20  //      Following syntaxes stop running the rest of program until YADE's output file
    named yaderesults.txt be appeared
21
22          File f = new File(MainPath+YadeOutputFile);
23          while (!f.exists()) {
24              try {
25                  Thread.sleep(100);
26          } catch (InterruptedException ie) { /* safe to ignore */ }
27          }
28
29  }
30  }
31
```

<Reader.java>

```
1   package coupledmodel;
2
3   import java.io.File;
4   import java.io.FileNotFoundException;
5   import java.io.FileReader;
6   import java.io.IOException;
7   import java.util.*;
8   import java.util.Scanner;
9
10
11  /**
12   *
13   * @author Pouyan Pirnia
14   */
15
16  public class Reader {
17  public String[] row;
18
19
20  public void method (String MainPath, String YadeOutputFile, String SavingFolder) {
21
22
23  //      Open and read YADE's output file (yaderesults.txt)
24          File file = new File(MainPath+YadeOutputFile);
25
26          try {
27
28          Scanner sc = new Scanner(file);
29
30  //      Read the first line, I suppose the first line is header
31          String nextLine = sc.nextLine();
32
33  //      Regex to break on any ammount of spaces
34          String regex = "(\\s)+";
35          String[] header = nextLine.split(regex);
36
37
38  //      This is printing all columns, you can
39  //      access each column from row using the array
40  //      indexes, example header[0], header[1], header[2]...
41          while (sc.hasNext()) {
42              row = sc.nextLine().split(regex);
43          }
44
45          sc.close();
46          }
47          catch (FileNotFoundException e) {
48          }
49
50  //      Remove the YADE's output file "yaderesults.txt" from the first directory to a
    specific folder "results" and save it there
51          try{
52
53              if(file.renameTo(new File(SavingFolder + file.getName()))){
54              System.out.println("File is moved successfully!");
55                  }else{
56              System.out.println("File is failed to move!");
57              }
58
59          }catch(Exception e){
60              e.printStackTrace();
61          }
62  }
63  public String[] returnskn(){
64          return row;
65  }
66  }
67
```

<YADE Python script – test.py>

```
# import yade modules that we will use below
import re
from yade import pack, plot, geom, utils, ymport, export, bodiesHandling

# DATA COMPONENTS
################################################################################
######## Receive data from
server#########################################################################

params= sys.argv[1:] #get everything after the test.py

dp= params[0] #put 7 pressure values in string format in dp
print dp
t=dp.split(",") #split dp by "," and set each pressure value in a new character
a=(t[0])
b=(t[1])
c=(t[2])
d=(t[3])
e=(t[4])
f=(t[5])
g=(t[6])
print a,b,c,d,e,f,g

# convert string to float
p1=float(a)
p2=float(b)
p3=float(c)
p4=float(d)
p5=float(e)
p6=float(f)
p7=float(g)
print 'p1,p2,p3,p4,p5,p6,p7 :', p1,p2,p3,p4,p5,p6,p7

dp1=p1-p2
dp2=p2-p3
dp3=p3-p4
dp4=p4-p5
dp5=p5-p6
dp6=p6-p7
print'dp1,dp2,dp3,dp4,dp5,dp6 :', dp1,dp2,dp3,dp4,dp5,dp6

# counter is the number of the current simulation
global counter
counter=params[1]
print 'counter :', counter

# load YADE's simulation from previous time step, it involves particles and facets
positions and forces
O.load('test'+counter+'.yade')
################################################################################################
##################
######## Test information
################################################################################
r1 = .003/2.0 # coarse particles radius (m)
r2 = .00035/2.0 # fine particles radius (m)
E1 = 1e7   # filter Young's modulus
E2 = 1e7  # soil Young's modulus
poisson1 = 0.3  # filter poisson ratio
poisson2 = 0.3  # sand Young's modulus
frictionAngle = .3
density1 = 2500  # kg/m3
density2 = 2500  # kg/m3
densitywater = 1000 # kg/m3
g=-9.806 # gravity m/s2
nf=160   # number of filter Particles
ns=15000 # number of fine Particles
Gs=2.5
```

```
63  plate=0
64  npc2=0
65  npc3=0
66  npc=0
67  ##############################################################################################
    ##################
68  ############ Defining materials for spheres
    ###############################################################
69  #O.materials.append((FrictMat(young=E2,poisson=poisson2,frictionAngle=frictionAngle,densi
    ty=250000, label = 'mat_wal2')))
70  #mat2 = O.materials[1]
71
72  ## In this section eroded particles in the container and their arrays are erased and
    equal amount of the same particles are generated at the top fine particles layer  #####
73  npc2 = sum(b.shape.radius for b in O.bodies if (isinstance(b.shape,Sphere) and
    (b.state.pos[2]<.03) and b.shape.radius == r2))/r2
74  print 'number of small particles in the container :', npc2
75  npc=npc2
76  npc=int(npc)
77  print 'npc:', npc
78
79
80  sp1 = pack.SpherePack()
81  sp1.makeCloud((0,0,.1),(0.01,0.01,.15),rMean=r2,num=npc)
82  for b in O.bodies:
83      if isinstance(b.shape,Sphere) and b.state.pos[2]<0.03: O.bodies.erase(b.id)
84
85  sp=SpherePack(); sp.fromSimulation()
86  O.reset()
87  #mat1 =
    O.materials.append((FrictMat(young=E1,poisson=poisson1,frictionAngle=frictionAngle,densit
    y=100*density1, label = 'mat_wal')))
88  #mat2 =
    O.materials.append((FrictMat(young=E2,poisson=poisson2,frictionAngle=.5,density=250000,
    label = 'mat_wal2')))
89
90  O.materials.append((FrictMat(young=E1,poisson=poisson1,frictionAngle=.6,density=100*densi
    ty1),FrictMat(young=E2,poisson=poisson2,frictionAngle=.3,density=100*density1,label =
    'mat_wal2')))
91
92  mat1,mat2 = [O.materials[i] for i in (0,1)]
93
94
95  sp1.toSimulation()
96  sp.toSimulation()
97
98
99  bodies1 = [b for b in O.bodies if isinstance (b.shape,Sphere) and b.shape.radius==r1]
100
101 for b in bodies1:
102     b.shape.color = (1,1,0)
103     b.mat = mat2
104 bodies2 = [b for b in O.bodies if isinstance (b.shape,Sphere) and b.shape.radius==r2]
105
106 for b in bodies2:
107     b.shape.color = (0,1,0)
108     b.mat = mat2
109
110 kw={'material':0}
111 O.bodies.append(utils.geom.facetBox((0.005,0.005,0.1),(0.005,0.005,0.1),wallMask=31))
112 O.bodies.append(ymport.gmsh('4.mesh', shift=Vector3(0, 0, .03), scale=.001,
    orientation=Quaternion((0, 0, 0), 0), **kw))
113 print 'len(O.bodies) final :',len(O.bodies)
114
115 # FUNCTIONAL COMPONENTS
    ###########################################################################
116 # simulation loop
```

```
    ###########################################################################
117 O.engines=[
118     ForceResetter(),
119     InsertionSortCollider([Bo1_Sphere_Aabb(),Bo1_Facet_Aabb(),Bo1_Wall_Aabb()]),
120     InteractionLoop(
121     # the loading plate is a wall, we need to handle sphere+sphere, sphere+facet,
        sphere+wall
122     [Ig2_Sphere_Sphere_ScGeom(),Ig2_Facet_Sphere_ScGeom(),Ig2_Wall_Sphere_ScGeom()], #
        collision geometry
123     [Ip2_FrictMat_FrictMat_FrictPhys()],   # collision "physics"
124     [Law2_ScGeom_FrictPhys_CundallStrack()]     # contact law -- apply forces
125     ),
126     NewtonIntegrator(gravity=(0,0,g),damping=0.4),  # apply gravity force and damping:
        numerical dissipation of energy
127     PyRunner(command='func()',virtPeriod=.5,label='checker1'), # call the func function
        (defined below) every 3000 iteration     virtPeriod
128     PyRunner(command='save()',virtPeriod=.51,label='checker2')
129 ]
130
131 # set timestep to a fraction of the critical timestep
132 O.dt=1*utils.PWaveTimeStep()
133 print 'O.dt=', O.dt
134 O.trackEnergy=True
135 O.resetTime() #Reset simulation time
136
137
138 ###############################################################################################
    ######################
139 ############ Calculation Section
    ###########################################################################
140 ###############################################################################################
    ######################
141 print 'BOTTOM LAYER INFORMATION %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%'
142 h1=max(b.state.pos[2]+b.shape.radius for b in O.bodies if isinstance(b.shape,Sphere)
    and b.shape.radius==r1)  # h1 is the height of filter layer
143 print 'h1=', h1
144 ParticlesVolume=sum(4.0/3.0*3.1416*b.shape.radius**3 for b in O.bodies if (isinstance
    (b.shape,Sphere) and (b.state.pos[2]<h1 and b.state.pos[2]>.03)))  #volume of filter
    particles
145 print 'ParticlesVolume=', ParticlesVolume
146 boxArea=.01*.01
147 BoxVolume=(h1-.03)*boxArea
148 print 'BoxVolume=', BoxVolume
149 n1=1-(ParticlesVolume/BoxVolume)
150 print 'porosity of bottom layer :', n1
151
152
153 print 'TOP LAYER INFORMATION %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%'
154 h2=max(b.state.pos[2]+b.shape.radius for b in O.bodies if isinstance(b.shape,Sphere)
    and b.shape.radius==r2)-h1    # h2 is the height of top of the fine particle layer
155 print 'h2=', h2
156 soilVolume=sum(4.0/3.0*3.1416*b.shape.radius**3 for b in O.bodies if isinstance
    (b.shape,Sphere) and (b.state.pos[2]<h2 and b.state.pos[2]>h1) and b.shape.radius==r2)
157 print 'soilVolume=', soilVolume
158 soilLayerVolume=h2*boxArea
159 n2=(soilLayerVolume-soilVolume)/soilLayerVolume
160 print 'porosity of top layer:', n2
161
162 print '1st filter layer information ########################################'
163 totalThickness=h1-.03
164 boxvolume1=boxArea*totalThickness*1.0/5.0
165 print 'boxvolume1:', boxvolume1
166 nfp1 = sum(b.shape.radius for b in O.bodies if (isinstance(b.shape,Sphere) and
    (b.state.pos[2]<h1 and b.state.pos[2]>4.0/5.0*totalThickness+.03) and b.shape.radius ==
    r1))/r1
167 print 'number of filter particles in Layer1 :', nfp1
168 tvfp1=(4.0/3.0*3.1416*r1**3)*nfp1
```

```
print 'total volume of filter particles in Layer1 :', tvfp1
#########################################################################
nsp1 = sum(b.shape.radius for b in O.bodies if (isinstance(b.shape,Sphere) and
(b.state.pos[2]<h1 and b.state.pos[2]>4.0/5.0*totalThickness+.03) and b.shape.radius ==
r2))/r2
print 'number of small particles in Layer1 :', nsp1
tvsp1=4.0/3.0*3.1416*(r2**3)*nsp1
print 'total volume of small particles in Layer1 :', tvsp1
#########################################################################
tpv1=tvfp1+tvsp1
print 'total particles volume in Layer1 :', tpv1
pfp1=(tvfp1/tpv1)
print 'proportion of filter particles in layer1 :', pfp1
psp1=(tvsp1/tpv1)
print 'proportion of small particles in layer1 :', psp1
nf1=1-(tpv1/boxvolume1)
print 'porosity of filter layer1:', nf1

S1=3/(r1*density1)  #m2/kg
S2=3/(r2*density2)
Ss1=S1*pfp1+S2*psp1
print 'S1,S2,Ss1:', S1,S2,Ss1

e1=nf1/(1-nf1)
k1=(math.sqrt(10)*(e1**3))/((Gs**2)*(Ss1**2)*(1+e1))
print 'void ratio and permeability in layey1 :', e1, k1

print '2nd filter layer information
#########################################################################'
nfp2 = sum(b.shape.radius for b in O.bodies if (isinstance(b.shape,Sphere) and
(b.state.pos[2]<4.0/5.0*totalThickness+.03 and
b.state.pos[2]>3.0/5.0*totalThickness+.03) and b.shape.radius == r1))/r1
print 'number of filter particles in Layer2 :', nfp2
tvfp2=(4.0/3.0*3.1416*r1**3)*nfp2
print 't1otal volume of filter particles in Layer2 :', tvfp2
#########################################################################
nsp2 = sum(b.shape.radius for b in O.bodies if (isinstance(b.shape,Sphere) and
(b.state.pos[2]<4.0/5.0*totalThickness+.03 and
b.state.pos[2]>3.0/5.0*totalThickness+.03) and b.shape.radius == r2))/r2
print 'number of small particles in Layer2 :', nsp2
tvsp2=4.0/3.0*3.1416*(r2**3)*nsp2
print 'total volume of small particles in Layer2 :', tvsp2
#########################################################################
tpv2=tvfp2+tvsp2
print 'total particles volume in Layer2 :', tpv2
pfp2=(tvfp2/tpv2)
print 'proportion of filter particles in layer2 :', pfp2
psp2=(tvsp2/tpv2)
print 'proportion of small particles in layer2:', psp2
nf2=1-(tpv2/boxvolume1)
print 'porosity of filter layer2:', nf2

Ss2=S1*pfp2+S2*psp2
print 'S1,S2,Ss2 :', S1,S2,Ss2

e2=nf2/(1-nf2)
k2=(math.sqrt(10)*(e2**3))/((Gs**2)*(Ss2**2)*(1+e2))
print 'void ratio and permeability in layer2 :', e2, k2

print '3rd filter layer information
#########################################################################'
nfp3 = sum(b.shape.radius for b in O.bodies if (isinstance(b.shape,Sphere) and
(b.state.pos[2]<3.0/5.0*totalThickness+.03 and
b.state.pos[2]>2.0/5.0*totalThickness+.03) and b.shape.radius == r1))/r1
print 'number of filter particles in Layer3 :', nfp3
tvfp3=(4.0/3.0*3.1416*r1**3)*nfp3
print 'total volume of filter particles in Layer3 :', tvfp3
```

```
226 ##############################################################################
227 nsp3 = sum(b.shape.radius for b in O.bodies if (isinstance(b.shape,Sphere) and
    (b.state.pos[2]<3.0/5.0*totalThickness+.03 and
    b.state.pos[2]>2.0/5.0*totalThickness+.03) and b.shape.radius == r2))/r2
228 print 'number of small particles in Layer3 :', nsp3
229 tvsp3=4.0/3.0*3.1416*(r2**3)*nsp3
230 print 'total volume of small particles in Layer3  :', tvsp3
231 ##############################################################################
232 tpv3=tvfp3+tvsp3
233 print 'total particles volume in Layer3 :', tpv3
234 pfp3=(tvfp3/tpv3)
235 print 'proportion of filter particles in layer3 :', pfp3
236 psp3=(tvsp3/tpv3)
237 print 'proportion of small particles in layer3 :', psp3
238 nf3=1-(tpv3/boxvolume1)
239 print 'porosity of filter layer3 :', nf3
240
241 Ss3=S1*pfp3+S2*psp3
242 print 'S1,S2,Ss3:', S1,S2,Ss3
243
244 e3=nf3/(1-nf3)
245 k3=(math.sqrt(10)*(e3**3))/((Gs**2)*(Ss3**2)*(1+e3))
246 print 'void ratio and permeability in layer3 :', e3, k3
247
248 print '4th filter layer information
    ##############################################################################'
249 nfp4 = sum(b.shape.radius for b in O.bodies if (isinstance(b.shape,Sphere) and
    (b.state.pos[2]<2.0/5.0*totalThickness+.03 and
    b.state.pos[2]>1.0/5.0*totalThickness+.03) and b.shape.radius == r1))/r1
250 print 'number of filter particles in Layer4 :', nfp4
251 tvfp4=(4.0/3.0*3.1416*r1**3)*nfp4
252 print 'total volume of filter particles in Layer4 :', tvfp4
253 ##############################################################################
254 nsp4 = sum(b.shape.radius for b in O.bodies if (isinstance(b.shape,Sphere) and
    (b.state.pos[2]<2.0/5.0*totalThickness+.03 and
    b.state.pos[2]>1.0/5.0*totalThickness+.03) and b.shape.radius == r2))/r2
255 print 'number of small particles in Layer4 :', nsp4
256 tvsp4=4.0/3.0*3.1416*(r2**3)*nsp4
257 print 'total volume of small particles in Layer4 :', tvsp4
258 ##############################################################################
259 tpv4=tvfp4+tvsp4
260 print 'total particles volume in Layer4 :', tpv4
261 pfp4=(tvfp4/tpv4)
262 print 'proportion of filter particles in layer4 :', pfp4
263 psp4=(tvsp4/tpv4)
264 print 'proportion of small particles in layer4 :', psp4
265 nf4=1-(tpv4/boxvolume1)
266 print 'porosity of filter layer4 :', nf4
267
268 Ss4=S1*pfp4+S2*psp4
269 print 'S1,S2,Ss4:', S1,S2,Ss4
270
271 e4=nf4/(1-nf4)
272 k4=(math.sqrt(10)*(e4**3))/((Gs**2)*(Ss4**2)*(1+e4))
273 print 'void ratio and permeability in layer4 :', e4, k4
274
275
276 print '5th filter layer information
    ##############################################################################'
277 nfp5 = sum(b.shape.radius for b in O.bodies if (isinstance(b.shape,Sphere) and
    (b.state.pos[2]<1.0/5.0*totalThickness+.03 and b.state.pos[2]>.03) and b.shape.radius
    == r1))/r1
278 print 'number of filter particles in Layer5 :', nfp5
279 tvfp5=(4.0/3.0*3.1416*r1**3)*nfp5
280 print 'total volume of filter particles in Layer5 :', tvfp5
281 ##############################################################################
282 nsp5 = sum(b.shape.radius for b in O.bodies if (isinstance(b.shape,Sphere) and
```

```
    (b.state.pos[2]<1.0/5.0*totalThickness+.03 and b.state.pos[2]>.03) and b.shape.radius
    == r2))/r2
283 print 'number of small particles in Layer5 :', nsp5
284 tvsp5=4.0/3.0*3.1416*(r2**3)*nsp5
285 print 'total volume of small particles in Layer5 :', tvsp5
286 ########################################################################
287 tpv5=tvfp5+tvsp5
288 print 'total particles volume in Layer5 :', tpv5
289 pfp5=(tvfp5/tpv5)
290 print 'proportion of filter particles in layer5 :', pfp5
291 psp5=(tvsp5/tpv5)
292 print 'proportion of small particles in layer5 :', psp5
293
294 nf5=1-(tpv5/boxvolume1)
295 print 'porosity of filter layer5:', nf5
296
297 Ss5=S1*pfp5+S2*psp5
298 print 'S1,S2,Ss5:', S1,S2,Ss5
299
300 e5=nf5/(1-nf5)
301 k5=(math.sqrt(10)*(e5**3))/((Gs**2)*(Ss5**2)*(1+e5))
302 print 'void ratio and permeability in layer5 :', e5, k5
303
304 print 'specimen information##########################################'
305 nfp=nfp1+nfp2+nfp3+nfp4+nfp5
306 print 'initial porosity in each layer:', nf1,nf2,nf3,nf4,nf5
307 print 'number of filter particles:', nfp
308 nsp=nsp1+nsp2+nsp3+nsp4+nsp5
309 print 'number of small particles:', nsp
310
311 tvfp=(4.0/3.0*3.1416*r1**3)*nfp
312 print 'total volume of filter particles in specimen :', tvfp
313 pfp=(tvfp/ParticlesVolume)
314 print 'proportion of filter particles in specimen :', pfp
315 tvsp=(4.0/3.0*3.1416*r2**3)*nsp
316 psp=(tvsp/ParticlesVolume)
317 print 'proportion of small particles in specimen :', psp
318
319 Ss=S1*pfp+S2*psp
320 print 'S1,S2,Ss:', S1,S2,Ss
321
322 ############### DRAG FORCE calculation
    ############################################################
323 vp1=(4.0/3.0)*3.1416*r1**3 #a filter sphere volume
324 vp2=(4.0/3.0)*3.1416*r2**3 #a fine sphere volume
325 d1=2*r1 #diameter of a coarse sphere
326 d2=2*r2 #diameter of a fine sphere
327 bz1=-.52333*(d1**3)*densitywater*g*100 # buoyancy force on filter particles
328 bz2=-.52333*(d2**3)*densitywater*g*100 # buoyancy force on fine particles
329
330 hl=(h1-.03)/5.0 #height of each filter layer
331
332
333 fdst=100*(dp1/((1-.37)*.019))*-vp2 # drag force on spheres in the fine particles layer
334 print 'fdst=', fdst
335
336 fdf1=100*(dp2/((1-nf1)*hl))*-vp1   # drag force on filter spheres in the 1st filter layer
337 print 'fdf1=', fdf1
338 fds1=100*(dp2/((1-nf1)*hl))*-vp2   # drag force on fine spheres in the 1st filter layer
339 print 'fds1=', fds1
340
341 fdf2=100*(dp3/((1-nf2)*hl))*-vp1
342 print 'fdf2=', fdf2
343 fds2=100*(dp3/((1-nf2)*hl))*-vp2
344 print 'fds2=', fds2
345
346 fdf3=100*(dp4/((1-nf3)*hl))*-vp1
```

```
print 'fdf3=', fdf3
fds3=100*(dp4/((1-nf3)*hl))*-vp2
print 'fds3=', fds3

fdf4=100*(dp5/((1-nf4)*hl))*-vp1
print 'fdf4=', fdf4
fds4=100*(dp5/((1-nf4)*hl))*-vp2
print 'fds4=', fds4

fdf5=100*(dp6/((1-nf5)*hl))*-vp1
print 'fdf5=', fdf5
fds5=100*(dp6/((1-nf5)*hl))*-vp2
print 'fds5=', fds5

########## apply forces on particles
######################################################################################

for b in O.bodies:

    if isinstance(b.shape,Sphere) and (b.state.pos[2]<h1 and
    b.state.pos[2]>4.0/5.0*totalThickness+.03) and b.shape.radius == r1:
    O.forces.addF(b.id,(0,0,bz1+fdf1),1)
    if isinstance(b.shape,Sphere) and (b.state.pos[2]<4.0/5.0*totalThickness+.03 and
    b.state.pos[2]>3.0/5.0*totalThickness+.03) and b.shape.radius == r1:
    O.forces.addF(b.id,(0,0,bz1+fdf2),1)
    if isinstance(b.shape,Sphere) and (b.state.pos[2]<3.0/5.0*totalThickness+.03 and
    b.state.pos[2]>2.0/5.0*totalThickness+.03) and b.shape.radius == r1:
    O.forces.addF(b.id,(0,0,bz1+fdf3),1)
    if isinstance(b.shape,Sphere) and (b.state.pos[2]<2.0/5.0*totalThickness+.03 and
    b.state.pos[2]>1.0/5.0*totalThickness+.03) and b.shape.radius == r1:
    O.forces.addF(b.id,(0,0,bz1+fdf4),1)
    if isinstance(b.shape,Sphere) and (b.state.pos[2]<1.0/5.0*totalThickness+.03 and
    b.state.pos[2]>.03) and b.shape.radius == r1:
    O.forces.addF(b.id,(0,0,bz1+fdf5),1)

    elif isinstance(b.shape,Sphere) and (b.state.pos[2]>h1) and b.shape.radius == r2:
    O.forces.addF(b.id,(0,0,bz2+fdst),1)

    elif isinstance(b.shape,Sphere) and (b.state.pos[2]<h1 and
    b.state.pos[2]>4.0/5.0*totalThickness+.03) and b.shape.radius == r2:
    O.forces.addF(b.id,(0,0,bz2+fds1),1)
    elif isinstance(b.shape,Sphere) and (b.state.pos[2]<4.0/5.0*totalThickness+.03 and
    b.state.pos[2]>3.0/5.0*totalThickness+.03) and b.shape.radius == r2:
    O.forces.addF(b.id,(0,0,bz2+fds2),1)
    elif isinstance(b.shape,Sphere) and (b.state.pos[2]<3.0/5.0*totalThickness+.03 and
    b.state.pos[2]>2.0/5.0*totalThickness+.03) and b.shape.radius == r2:
    O.forces.addF(b.id,(0,0,bz2+fds3),1)
    elif isinstance(b.shape,Sphere) and (b.state.pos[2]<2.0/5.0*totalThickness+.03 and
    b.state.pos[2]>1.0/5.0*totalThickness+.03) and b.shape.radius == r2:
    O.forces.addF(b.id,(0,0,bz2+fds4),1)
    elif isinstance(b.shape,Sphere) and (b.state.pos[2]<1.0/5.0*totalThickness+.03 and
    b.state.pos[2]>.03) and b.shape.radius == r2:
    O.forces.addF(b.id,(0,0,bz2+fds5),1)

#########################################################################################
#####################################
# the function "func" just repeat previous command lines in Calculation Section

def func():
    if O.iter>0:
    print 'BOTTOM LAYER INFORMATION %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%'
    h1=max(b.state.pos[2]+b.shape.radius for b in O.bodies if
    isinstance(b.shape,Sphere) and b.shape.radius==r1)
    print 'h1=', h1
    ParticlesVolume=sum(4.0/3.0*3.1416*b.shape.radius**3 for b in O.bodies if
    (isinstance (b.shape,Sphere) and (b.state.pos[2]<h1 and b.state.pos[2]>.03)))
    print 'ParticlesVolume=', ParticlesVolume
```

```
400       boxArea=.01*.01
401       BoxVolume=(h1-.03)*boxArea
402       print 'BoxVolume=', BoxVolume
403       n1=1-(ParticlesVolume/BoxVolume)
404       print 'porosity of bottom layer :', n1
405
406       print '1st FILTER layer
          information##############################################################################
          '
407       totalThickness=h1-.03
408       boxvolume1=boxArea*totalThickness*1.0/5.0
409       print 'boxvolume1 :', boxvolume1
410       nfp1 = sum(b.shape.radius for b in O.bodies if (isinstance(b.shape,Sphere) and
          (b.state.pos[2]<h1 and b.state.pos[2]>4.0/5.0*totalThickness+.03) and
          b.shape.radius == r1))/r1
411       print 'number of filter particles in Layer1 :', nfp1
412       tvfp1=(4.0/3.0*3.1416*r1**3)*nfp1
413       print 'total volume of filter particles in Layer1 :', tvfp1
414       ###########################################################################
415       nsp1 = sum(b.shape.radius for b in O.bodies if (isinstance(b.shape,Sphere) and
          (b.state.pos[2]<h1 and b.state.pos[2]>4.0/5.0*totalThickness+.03) and
          b.shape.radius == r2))/r2
416       print 'number of small particles in Layer1 :', nsp1
417       tvsp1=4.0/3.0*3.1416*(r2**3)*nsp1
418       print 'total volume of small particles in Layer1 :', tvsp1
419       ###########################################################################
420       tpv1=tvfp1+tvsp1
421       print 'total particles volume in Layer1 :', tpv1
422       pfp1=(tvfp1/tpv1)
423       print 'proportion of filter particles in layer1 :', pfp1
424       psp1=(tvsp1/tpv1)
425       print 'proportion of small particles in layer1 :', psp1
426       global nf1
427       nf1=1-(tpv1/boxvolume1)
428       print 'porosity of filter layer1 :', nf1
429
430       S1=3/(r1*density1)  #m2/kg
431       S2=3/(r2*density2)
432       Ss1=S1*pfp1+S2*psp1
433       print 'S1,S2,Ss1:', S1,S2,Ss1
434
435       e1=nf1/(1-nf1)
436       global k1
437       k1=(math.sqrt(10)*(e1**3))/((Gs**2)*(Ss1**2)*(1+e1))
438       print 'void ratio and permeability in layer1 :', e1, k1
439
440       print '2nd FILTER layer information
          ###########################################################################'
441       nfp2 = sum(b.shape.radius for b in O.bodies if (isinstance(b.shape,Sphere) and
          (b.state.pos[2]<4.0/5.0*totalThickness+.03 and
          b.state.pos[2]>3.0/5.0*totalThickness+.03) and b.shape.radius == r1))/r1
442       print 'number of filter particles in Layer2 :', nfp2
443       tvfp2=(4.0/3.0*3.1416*r1**3)*nfp2
444       print 't1otal volume of filter particles in Layer2 :', tvfp2
445       ###########################################################################
446       nsp2 = sum(b.shape.radius for b in O.bodies if (isinstance(b.shape,Sphere) and
          (b.state.pos[2]<4.0/5.0*totalThickness+.03 and
          b.state.pos[2]>3.0/5.0*totalThickness+.03) and b.shape.radius == r2))/r2
447       print 'number of small particles in Layer2 :', nsp2
448       tvsp2=4.0/3.0*3.1416*(r2**3)*nsp2
449       print 'total volume of small particles in Layer2 :', tvsp2
450       ###########################################################################
451       tpv2=tvfp2+tvsp2
452       print 'total particles volume in Layer2 :', tpv2
453       pfp2=(tvfp2/tpv2)
454       print 'proportion of filter particles in layer2 :', pfp2
455       psp2=(tvsp2/tpv2)
```

```
456       print 'proportion of small particles in laye2 :', psp2
457       global nf2
458       nf2=1-(tpv2/boxvolume1)
459       print 'porosity of filter layer2 :', nf2
460
461       Ss2=S1*pfp2+S2*psp2
462       print 'S1,S2,Ss2:', S1,S2,Ss2
463
464       e2=nf2/(1-nf2)
465       global k2
466       k2=(math.sqrt(10)*(e2**3))/((Gs**2)*(Ss2**2)*(1+e2))
467       print 'void ratio and permeability in layer2 :', e2, k2
468
469       print '3rd FILTER layer information
          ##############################################################################'
470       nfp3 = sum(b.shape.radius for b in O.bodies if (isinstance(b.shape,Sphere) and
          (b.state.pos[2]<3.0/5.0*totalThickness+.03 and
          b.state.pos[2]>2.0/5.0*totalThickness+.03) and b.shape.radius == r1))/r1
471       print 'number of filter particles in Layer3 :', nfp3
472       tvfp3=(4.0/3.0*3.1416*r1**3)*nfp3
473       print 'total volume of filter particles in Layer3 :', tvfp3
474       ##############################################################################
475       nsp3 = sum(b.shape.radius for b in O.bodies if (isinstance(b.shape,Sphere) and
          (b.state.pos[2]<3.0/5.0*totalThickness+.03 and
          b.state.pos[2]>2.0/5.0*totalThickness+.03) and b.shape.radius == r2))/r2
476       print 'number of small particles in Layer3 :', nsp3
477       tvsp3=4.0/3.0*3.1416*(r2**3)*nsp3
478       print 'total volume of small particles in Layer3 :', tvsp3
479       ##############################################################################
480       tpv3=tvfp3+tvsp3
481       print 'total particles volume in Layer3 :', tpv3
482       pfp3=(tvfp3/tpv3)
483       print 'proportion of filter particles in layer3:', pfp3
484       psp3=(tvsp3/tpv3)
485       print 'proportion of small particles in layer3:', psp3
486       global nf3
487       nf3=1-(tpv3/boxvolume1)
488       print 'porosity of filter layer3 :', nf3
489
490       Ss3=S1*pfp3+S2*psp3
491       print 'S1,S2,Ss3:', S1,S2,Ss3
492
493       e3=nf3/(1-nf3)
494       global k3
495       k3=(math.sqrt(10)*(e3**3))/((Gs**2)*(Ss3**2)*(1+e3))
496       print 'void ratio and permeability in layer3 :', e3, k3
497
498       print '4th FILTER layer information
          ##############################################################################'
499       nfp4 = sum(b.shape.radius for b in O.bodies if (isinstance(b.shape,Sphere) and
          (b.state.pos[2]<2.0/5.0*totalThickness+.03 and
          b.state.pos[2]>1.0/5.0*totalThickness+.03) and b.shape.radius == r1))/r1
500       print 'number of filter particles in Layer4 :', nfp4
501       tvfp4=(4.0/3.0*3.1416*r1**3)*nfp4
502       print 'total volume of filter particles in Layer4 :', tvfp4
503       ##############################################################################
504       nsp4 = sum(b.shape.radius for b in O.bodies if (isinstance(b.shape,Sphere) and
          (b.state.pos[2]<2.0/5.0*totalThickness+.03 and
          b.state.pos[2]>1.0/5.0*totalThickness+.03) and b.shape.radius == r2))/r2
505       print 'number of small particles in Layer4 :', nsp4
506       tvsp4=4.0/3.0*3.1416*(r2**3)*nsp4
507       print 'total volume of small particles in Layer4 :', tvsp4
508       ##############################################################################
509       tpv4=tvfp4+tvsp4
510       print 'total particles volume in Layer4 :', tpv4
511       pfp4=(tvfp4/tpv4)
512       print 'proportion of filter particles in layer4 :', pfp4
```

```
513     psp4=(tvsp4/tpv4)
514     print 'proportion of small particles in layer4 :', psp4
515     global nf4
516     nf4=1-(tpv4/boxvolume1)
517     print 'porosity of filter layer4 :', nf4
518
519     Ss4=S1*pfp4+S2*psp4
520     print 'S1,S2,Ss4:', S1,S2,Ss4
521
522     e4=nf4/(1-nf4)
523     global k4
524     k4=(math.sqrt(10)*(e4**3))/((Gs**2)*(Ss4**2)*(1+e4))
525     print 'void ratio and permeability in layer4 :', e4, k4
526
527
528     print '5th FILTER layer information
        ##############################################################################'
529     nfp5 = sum(b.shape.radius for b in O.bodies if (isinstance(b.shape,Sphere) and
        (b.state.pos[2]<1.0/5.0*totalThickness+.03 and b.state.pos[2]>.03) and
        b.shape.radius == r1))/r1
530     print 'number of filter particles in Layer5 :', nfp5
531     tvfp5=(4.0/3.0*3.1416*r1**3)*nfp5
532     print 'total volume of filter particles in Layer5 :', tvfp5
533     ##############################################################################
534     nsp5 = sum(b.shape.radius for b in O.bodies if (isinstance(b.shape,Sphere) and
        (b.state.pos[2]<1.0/5.0*totalThickness+.03 and b.state.pos[2]>.03) and
        b.shape.radius == r2))/r2
535     print 'number of small particles in Layer5 :', nsp5
536     tvsp5=4.0/3.0*3.1416*(r2**3)*nsp5
537     print 'total volume of small particles in Layer5 :', tvsp5
538     ##############################################################################
539     tpv5=tvfp5+tvsp5
540     print 'total particles volume in Layer5 :', tpv5
541     pfp5=(tvfp5/tpv5)
542     print 'proportion of filter particles in layer5 :', pfp5
543     psp5=(tvsp5/tpv5)
544     print 'proportion of small particles in layer5 :', psp5
545     global nf5
546     nf5=1-(tpv5/boxvolume1)
547     print 'porosity of filter layer5:', nf5
548
549     Ss5=S1*pfp5+S2*psp5
550     print 'S1,S2,Ss5:', S1,S2,Ss5
551
552     e5=nf5/(1-nf5)
553     global k5
554     k5=(math.sqrt(10)*(e5**3))/((Gs**2)*(Ss5**2)*(1+e5))
555     print 'void ratio and permeability in layer5 :', e5, k5
556
557     print 'Container information##############################################'
558         global npc2,npc3
559     npc2 = sum(b.shape.radius for b in O.bodies if (isinstance(b.shape,Sphere) and
        (b.state.pos[2]<.03) and b.shape.radius == r2))/r2
560     print 'number of small particles in the container :', npc2
561
562     print 'specimen information##############################################'
563
564     nfp=nfp1+nfp2+nfp3+nfp4+nfp5
565     print 'initial porosity in each layer:', nf1,nf2,nf3,nf4,nf5
566     print 'number of filter particles:', nfp
567     nsp=nsp1+nsp2+nsp3+nsp4+nsp5
568     print 'number of small particles:', nsp
569
570     tvfp=(4.0/3.0*3.1416*r1**3)*nfp
571     print 'total volume of filter particles in specimen :', tvfp
572     pfp=(tvfp/ParticlesVolume)
573     print 'proportion of filter particles in specimen:', pfp
```

```
574       tvsp=(4.0/3.0*3.1416*r2**3)*nsp
575       psp=(tvsp/ParticlesVolume)
576       print 'proportion of small particles in specimen:', psp
577
578       Ss=S1*pfp+S2*psp
579       print 'S1,S2,Ss:', S1,S2,Ss
580
581
582       ############## DRAG FORCE calculation
          ############################################################
583       vp1=(4.0/3.0)*3.1416*r1**3
584       vp2=(4.0/3.0)*3.1416*r2**3
585       d1=2*r1
586       d2=2*r2
587       bz1=-.52333*(d1**3)*densitywater*g*100
588       bz2=-.52333*(d2**3)*densitywater*g*100
589       print 'buoyancy1,2:', bz1, bz2
590
591       hl=(h1-.03)/5.0
592
593       fdst=100*(dp1/((1-.37)*.019))*-vp2
594       print 'fdst=', fdst
595
596       fdf1=100*(dp2/((1-nf1)*hl))*-vp1
597       print 'fdf1=', fdf1
598       fds1=100*(dp2/((1-nf1)*hl))*-vp2
599       print 'fds1=', fds1
600
601       fdf2=100*(dp3/((1-nf2)*hl))*-vp1
602       print 'fdf2=', fdf2
603       fds2=100*(dp3/((1-nf2)*hl))*-vp2
604       print 'fds2=', fds2
605
606       fdf3=100*(dp4/((1-nf3)*hl))*-vp1
607       print 'fdf3=', fdf3
608       fds3=100*(dp4/((1-nf3)*hl))*-vp2
609       print 'fds3=', fds3
610
611       fdf4=100*(dp5/((1-nf4)*hl))*-vp1
612       print 'fdf4=', fdf4
613       fds4=100*(dp5/((1-nf4)*hl))*-vp2
614       print 'fds4=', fds4
615
616       fdf5=100*(dp6/((1-nf5)*hl))*-vp1
617       print 'fdf5=', fdf5
618       fds5=100*(dp6/((1-nf5)*hl))*-vp2
619       print 'fds5=', fds5
620
621           ########## apply forces on particles
              ##############################################################################
              #####
622       for b in O.bodies:
623
624           if isinstance(b.shape,Sphere) and (b.state.pos[2]<h1 and
              b.state.pos[2]>4.0/5.0*totalThickness+.03) and b.shape.radius == r1:
625           O.forces.addF(b.id,(0,0,bz1+fdf1),1)
626           if isinstance(b.shape,Sphere) and (b.state.pos[2]<4.0/5.0*totalThickness+.03
              and b.state.pos[2]>3.0/5.0*totalThickness+.03) and b.shape.radius == r1:
627           O.forces.addF(b.id,(0,0,bz1+fdf2),1)
628           if isinstance(b.shape,Sphere) and (b.state.pos[2]<3.0/5.0*totalThickness+.03
              and b.state.pos[2]>2.0/5.0*totalThickness+.03) and b.shape.radius == r1:
629           O.forces.addF(b.id,(0,0,bz1+fdf3),1)
630           if isinstance(b.shape,Sphere) and (b.state.pos[2]<2.0/5.0*totalThickness+.03
              and b.state.pos[2]>1.0/5.0*totalThickness+.03) and b.shape.radius == r1:
631           O.forces.addF(b.id,(0,0,bz1+fdf4),1)
632           if isinstance(b.shape,Sphere) and (b.state.pos[2]<1.0/5.0*totalThickness+.03
              and b.state.pos[2]>.03) and b.shape.radius == r1:
```

```
633             O.forces.addF(b.id,(0,0,bz1+fdf5),1)
634
635             elif isinstance(b.shape,Sphere) and (b.state.pos[2]>h1) and b.shape.radius == r2:
636             O.forces.addF(b.id,(0,0,bz2+fdst),1)
637
638             elif isinstance(b.shape,Sphere) and (b.state.pos[2]<h1 and
                b.state.pos[2]>4.0/5.0*totalThickness+.03) and b.shape.radius == r2:
639             O.forces.addF(b.id,(0,0,bz2+fds1),1)
640             elif isinstance(b.shape,Sphere) and (b.state.pos[2]<4.0/5.0*totalThickness+.03
                and b.state.pos[2]>3.0/5.0*totalThickness+.03) and b.shape.radius == r2:
641             O.forces.addF(b.id,(0,0,bz2+fds2),1)
642             elif isinstance(b.shape,Sphere) and (b.state.pos[2]<3.0/5.0*totalThickness+.03
                and b.state.pos[2]>2.0/5.0*totalThickness+.03) and b.shape.radius == r2:
643             O.forces.addF(b.id,(0,0,bz2+fds3),1)
644             elif isinstance(b.shape,Sphere) and (b.state.pos[2]<2.0/5.0*totalThickness+.03
                and b.state.pos[2]>1.0/5.0*totalThickness+.03) and b.shape.radius == r2:
645             O.forces.addF(b.id,(0,0,bz2+fds4),1)
646             elif isinstance(b.shape,Sphere) and (b.state.pos[2]<1.0/5.0*totalThickness+.03
                and b.state.pos[2]>.03) and b.shape.radius == r2:
647             O.forces.addF(b.id,(0,0,bz2+fds5),1)
648
649         print '*****************************************'
650             print 'O.time :', O.time
651
652
653     def save():
654         if O.time>.0:
655             global nf1,nf2,nf3,nf4,nf5,k1,k2,k3,k4,k5,npc,npc2, h1, counter
656         npc=npc2+npc3
657             npc=int(npc)
658         print 'h1, npc :', h1, npc
659
                plot.addData(n1=nf1,n2=nf2,n3=nf3,n4=nf4,n5=nf5,k1=k1,k2=k2,k3=k3,k4=k4,k5=k5,nc=
                npc) #selected parameters will be written on the text file
660
661         counterint=int(float(counter))
662         ncint=counterint+1
663         ncstr=str(ncint)
664         print 'countersave:', ncstr
665         O.save("test"+ncstr+".yade")     # save current simulation to file name.yade
666             plot.saveDataTxt('yaderesults.txt')  # save the results in yaderesults.txt
667             O.pause()
668
669     O.saveTmp()
670     O.run()
671     O.wait()
672     quit()
673
674
675
676
677
678
679
```

<Cleinet-server>

<client.py>

```
1  import socket
2  import sys
3
4  HOST, PORT = "localhost", 64000
5  data = " ".join(sys.argv[1:]) #get everything after the client.py (client.py pip18"+'
   '+n)
6
7  # Create a socket (SOCK_STREAM means a TCP socket)
8  sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
9
10 try:
11     # Connect to server and send data
12     sock.connect((HOST, PORT))
13     sock.sendall(data + "\n")
14
15     # Receive data from the server and shut down
16     received = sock.recv(1024)
17 finally:
18     sock.close()
19
20 print "Sent:     {}".format(data)
21 print "Received: {}".format(received)
22
```

<server.py>

```
1   import SocketServer
2   import subprocess
3   import os.path
4   import signal
5   import re
6
7   class MyTCPHandler(SocketServer.BaseRequestHandler):
8
9       def handle(self):
10
11          self.data = self.request.recv(1024).strip() #receive the string were sent by
            client and remove the extra whitepace from front and back of the string (
            test"+' '+n)
12
13          params = self.data.split(' ')  # split the string by space ("test", "n")
14
15          function = params[0]  #put the first parameter in function (function = "test")
16
17          counter = params[1]   # function = "n"
18
19          with open('pressure.txt','r') as f:  #open COMSOL's output file (pressure.txt)
            and read it
20              dp=f.read(300)
21              f.close()
22
23          if function == "test":  #if Yade's script (test.py) and the output file
            (yaderesults.txt), pressure values from COMSOL (dp) and simulation number
            (counter) are available go to function doTest
24              self.doTest('test.py', 'yaderesults.txt', dp, counter)
25
26          else:
27              self.request.sendall("ERROR: Function not found")   # unless show the
                message "Function not found"
28
29
30      def doTest(self, testFile, outputFile, dp, counter):
31          try:
32
33              paramsString=dp.rstrip() #remove the rest of characters in the line without
                this syntax the next parameter written in the next line
34
35              counterstring=counter.rstrip()
36
37              command = 'python yade -j8 -x '+testFile+' '+paramsString+'
                '+counterstring  #This is the command line for terminal to run Yade's
                script (pip18.py)
38
39              print 'command: '+command
40
41              process = subprocess.Popen(command.strip(), shell=True)   #Run the
                command.strip() or open the process
42
43              while not os.path.isfile(outputFile):   #return true if path is an existing
                regular file
44                  pass
45
46              content = ''  # content is defined as space
47
48              while not self.isNotEmptyContent(content):   # Open and read the outputFile
                if the outputFile is not empty based on the function isNotEmptyContent
49                  with open(outputFile, 'r') as content_file:
50                      content = content_file.read()
51                      content_file.close()
52
53              os.kill(process.pid, signal.SIGUSR1)   #kill the process
54
55              self.request.sendall(content)
```

```
56
57              except Exception as e:           # report a message "ERROR" on terminal if
                doTest is not completed
58                  self.request.sendall('ERROR: '+str(e))
59
60
61          def isNotEmptyContent(self, content):
62              return re.search("\w+", content)  # searching in the file for defined
                charachter in content
63
64      if __name__ == "__main__":
65          HOST, PORT = "localhost", 64000
66          server = SocketServer.TCPServer((HOST, PORT), MyTCPHandler)
67          server.serve_forever()
68
```

LIST OF BIBLIOGRAPHICAL REFERENCES

Abdou, H., Emeriault, F., & Plé, O. (2018). New approach to describe hydro-mechanical phenomenon of suffusion: erosion, transport and deposition. *European Journal of Environmental and Civil Engineering*, 1-19.

Anandarajah, A. (2003). Discrete element modelling of leaching-induced apparent overconsolidation in kaolinite. *Soils and foundations*, *43*(6), 1-12.

Andrade, J. E., Avila, C., Hall, S., Lenoir, N., & Viggiani, G. (2011). Multiscale modeling and characterization of granular matter: from grain kinematics to continuum mechanics. *Journal of the Mechanics and Physics of Solids*, *59*(2), 237-250.

Andrade, J. E., & Tu, X. (2009). Multiscale framework for behavior prediction in granular media. *Mechanics of Materials*, *41*(6), 652-669.

Balay, S., Abhyankar, M., Adams, J., Brown, P., Brune, K., Buschelman, L., . . . Zampini, H. (2013). PETSc Users Manual, Argonne National Laboratory.

Basarir, H., Karpuz, C., & Tutluoğlu, L. (2008). 3D modeling of ripping process. *International Journal of Geomechanics*, *8*(1), 11-19.

Bear, J. (2012). *Hydraulics of groundwater*. Vol. 2. Springer Science & Business Media.

Beetstra, R., van der Hoef, M. A., & Kuipers, J. (2007). Drag force of intermediate Reynolds number flow past mono-and bidisperse arrays of spheres. *AIChE journal, 53*(2), 489-501.

Biot, M. A. (1941). General theory of three-dimensional consolidation. *Journal of applied physics, 12*(2), 155-164.

Brinkman, H. (1949). A calculation of the viscous force exerted by a flowing fluid on a dense swarm of particles. Flow, *Turbulence and Combustion, 1*(1), 27.

Bym, T., Marketos, G., Burland, J., & O'sullivan, C. (2013). Use of a two-dimensional discrete-element line-sink model to gain insight into tunnelling-induced deformations. *Geotechnique, 63*(9), 791.

Cao, Z., Pender, G., & Meng, J. (2006). Explicit formulation of the Shields diagram for incipient motion of sediment. *Journal of Hydraulic Engineering, 132*(10), 1097-1099.

CDA. (2015). *Dams in Canada*. Available at http://www.imis100ca1.ca/cda/Main/Dams_in_Canada/Dams_in_Canada/CDA/Dams_In_Canada.aspx

Chapuis, R. P., & Aubertin, M. (2003). On the use of the Kozeny Carman equation to predict the hydraulic conductivity of soils. *Canadian Geotechnical Journal, 40*(3), 616-628.

Chareyre, B., Cortis, A., Catalano, E., & Barthélemy, E. (2012). Pore-scale modeling of viscous flow and induced forces in dense sphere packings. *Transport in porous media, 94(*2), 595-615.

Chen, F. (2009). *Coupled flow discrete element method application in granular porous media using open source codes*. Doctoral Dissertations, University of Tennessee, 2009.

Cheng, K., Wang, Y., & Yang, Q. (2018). A semi-resolved CFD-DEM model for seepage-induced fine particle migration in gap-graded soils. *Computers and Geotechnics, 100*, 30-51.

Cheng, Y., Nakata, Y., & Bolton, M. (2003). Discrete element simulation of crushable soil. *Geotechnique, 53*(7), 633-641.

COMSOL. (2016). *COMSOL MULTIPHYSICS*. Version 5.3a. available at http://www.comsol.com/

Cook, B., Lee, M., DiGiovanni, A., Bronowski, D., Perkins, E., & Williams, J. (2004). Discrete element modeling applied to laboratory simulation of near-wellbore mechanics. *International Journal of Geomechanics, 4*(1), 19-27.

Cundall, P. (1987). Distinct element models of rock and soil structure. *Analytical and computational methods in engineering rock mechanics*, 129-163.

Cundall, P. A., & Hart, R. D. (1992). Numerical modelling of discontinua. *Engineering computations, 9*(2), 101-113.

Cundall, P. A., & Strack, O. D. (1979). A discrete numerical model for granular assemblies. *Geotechnique, 29*(1), 47-65.

Dang, H. K., & Meguid, M. A. (2013). An efficient finite–discrete element method for quasi-static nonlinear soil–structure interaction problems. *International Journal for Numerical and Analytical Methods in Geomechanics*, 37(2), 130-149.

Darcy, H. (1856). *Les fontaines publiques de la ville de Dijon: exposition et application*. Victor Dalmont.

Day, R., Hight, D., & Potts, D. (1998). Finite element analysis of construction stability of Thika Dam. *Computers and Geotechnics, 23*(4), 205-219.

Derakhshani, S. M., Schott, D. L., & Lodewijks, G. (2014). Micro–macro properties of quartz sand: Experimental investigation and DEM simulation. *Powder technology, 269*, 127-138.

Di Felice, R. (1994). The voidage function for fluid-particle interaction systems. *International Journal of Multiphase Flow, 20*(1), 153-159.

Ding, X., Zhang, L., Zhu, H., & Zhang, Q. (2014). Effect of model scale and particle size distribution on PFC3D simulation results. *Rock mechanics and rock engineering, 47*(6), 2139-2156.

Duhaime, F., Ahmed, S., Pirnia, P., Ethier, Y., & Marefat, V. (2017). Stress-based method for slope stability analyses with COMSOL Multiphysics. Paper presented at the GeoOttawa, Ottawa, ON, CANADA.

Duhaime, F., & Chapuis, R. P. (2014). A coupled analysis of cavity and pore volume changes for pulse tests conducted in soft clay deposits. *International Journal for Numerical and Analytical Methods in Geomechanics, 38*(9), 903-924.

Dumberry, K., Duhaime, F., & Éthier, Y. A. (2015). Experimental study of contact erosion during core overtopping. *In Canadian Dam Association Annual Conference, Mississauga, Canada, October 3-8, 2015* (pp. 214-228). Toronto, Canada: Canadian Dam Association.

Dumberry, K., Duhaime, F., & Ethier, Y. A. (2017). Erosion monitoring during core overtopping using a laboratory model with digital image correlation and X-ray microcomputed tomography. *Canadian Geotechnical Journal, 55*(2), 234-245.

Eberhardt, E., Stead, D., & Coggan, J. (2004). Numerical analysis of initiation and progressive failure in natural rock slopes—the 1991 Randa rockslide. *International Journal of Rock Mechanics and Mining Sciences, 41*(1), 69-87.

Elmekati, A., & El Shamy, U. (2010). A practical co-simulation approach for multiscale analysis of geotechnical systems. *Computers and Geotechnics, 37*(4), 494-503.

Ergun, S. (1952). Fluid flow through packed columns. *Chemical Engineering Progress, 48*, 89-94.

Esteghamatian, A., Bernard, M., Lance, M., Hammouti, A., & Wachs, A. (2017). Micro/meso simulation of a fluidized bed in a homogeneous bubbling regime. *International Journal of Multiphase Flow, 92*, 93-111.

Fell, R., MacGregor, P., Stapledon, D., Bell, G., & Foster, M. (2015). *Geotechnical engineering of dams* (2nd edition). Boca Raton, FL: Taylor & Francis.

FEMA. (2013). *Selecting and Accommodating Inflow Design Flood for Dams.* Coll. (FEMA P-94).

FEMA. (2014). *Overtopping protection for dams: Best practices for design, construction, problem identification and evaluation, inspection, maintenance, renovation and repair* (Report n˚ P-1015). Denver, CO: USBR.

FEMA. (2007). *The National Dam Safety Program – Final Report on Coordination and Cooperation with the European Union on Embankment Failure Analysis.* . Coll. (Report FEMA 6002).

Finsterle, S., Sonnenthal, E. L., & Spycher, N. (2014). Advances in subsurface modeling using the TOUGH suite of simulators. *Computers & Geosciences, 65*, 2-12.

Foster, M., Fell, R., & Spannagle, M. (2000). The statistics of embankment dam failures and accidents. *Canadian Geotechnical Journal, 37*(5), 1000-1024.

Frishfelds, V., Hellström, J. G. I., Lundström, T. S., & Mattsson, H. (2011). Fluid flow induced internal erosion within porous media: modelling of the no erosion filter test experiment. *Transport in porous media, 89*(3), 441-457.

Furtney, J., Zhang, F., & Han, Y. (2013). Review of methods and applications for incorporating fluid flow in the discrete element method. In *Proceedings of the 3rd International FLAC/DEM Symposium*, Hangzhou, China.

Galindo-Torres, S. (2013). A coupled Discrete Element Lattice Boltzmann Method for the simulation of fluid–solid interaction with particles of general shapes. *Computer Methods in Applied Mechanics and Engineering, 265*, 107-119.

Galindo-Torres, S., Scheuermann, A., Mühlhaus, H., & Williams, D. (2015). A micro-mechanical approach for the study of contact erosion. *Acta Geotechnica, 10*(3), 357-368.

Gao, G., & Meguid, M. (2018). On the role of sphericity of falling rock clusters—insights from experimental and numerical investigations. *Landslides, 15*(2), 219-232.

Garner, S., & Fannin, R. (2010). Understanding internal erosion: a decade of research following a sinkhole event. *The international journal on hydropower & dams, 17*(3), 93-98.

Garner, S., & Sobkowicz, J. (2002). Internal instability in gap-graded cores and filters. In *Proceedings of the Canadian Dam Association Annual Conference* (pp. 6-10).

Geuzaine, C., & Remacle, J. F. (2009). Gmsh: A 3-D finite element mesh generator with built-in pre-and post-processing facilities. *International Journal for Numerical Methods in Engineering, 79*(11), 1309-1331.

Goniva, C., Kloss, C., Hager, A., & Pirker, S. (2010). An open source CFD-DEM perspective. *In Proceedings of OpenFOAM Workshop*, Göteborg (pp. 1-10).

Goodarzi, M., Kwok, C. Y., & Tham, L. G. (2015). A continuum-discrete model using Darcy's law: formulation and verification. *International Journal for Numerical and Analytical Methods in Geomechanics, 39*(3), 327-342.

Graham, P., & Wayne, J. (1999). *A procedure for estimating loss of life caused by dam failure.* US Department of the Interior, Bureau of Reclamation, Dam Safety Office.

Gross, L., Bourgouin, L., Hale, A. J., & Mühlhaus, H.-B. (2007). Interface modeling in incompressible media using level sets in Escript. *Physics of the Earth and Planetary Interiors, 163*(1), 23-34.

Guidoux, C., Faure, Y.-H., Beguin, R., & Ho, C.-C. (2010). Contact Erosion at the Interface between Granular Coarse Soil and Various Base Soils under Tangential Flow Condition. *Journal of geotechnical and geoenvironmental engineering, 136*(5), 741-750.

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

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

Hagan, M. T., & Menhaj, M. B. (1994). Training feedforward networks with the Marquardt algorithm. *IEEE transactions on Neural Networks, 5*(6), 989-993.

Hellström, J. G. I. (2009). *Internal erosion in embankment dams fluid flow through and deformation of porous media* (Luleå University of Technology, Sweden).

Hill, R. J., Koch, D. L., & Ladd, A. J. (2001). The first effects of fluid inertia on flows in ordered and random arrays of spheres. *Journal of Fluid Mechanics, 448*, 213-241.

Hnang, T.-k. (1996). Stability analysis of an earth dam under steady state seepage. *Computers & structures, 58*(6), 1075-1082.

Holmes, D. W., Williams, J. R., & Tilke, P. (2011). Smooth particle hydrodynamics simulations of low Reynolds number flows through porous media. International *Journal for Numerical and Analytical Methods in Geomechanics, 35*(4), 419-437.

Huang, X., Hanley, K. J., O'Sullivan, C., & Kwok, F. C. (2014). Effect of sample size on the response of DEM samples with a realistic grading. *Particuology, 15*, 107-115.

Huilin, L., & Gidaspow, D. (2003). Hydrodynamics of binary fluidization in a riser: CFD simulation using two granular temperatures. *Chemical Engineering Science, 58*(16), 3777-3792.

ICOLD. (2016). Bulletin 164 Internal erosion of existing dams, levees and dikes, and their foundations. (Volume 2: Case histories, investigations, testing, remediation and surveillance).

ICOLD. (2017). Bulletin 164 Internal erosion of existing dams, levees and dikes, and their foundations.

Indraratna, B., & Locke, M. (1999). Design methods for granular filters—Critical review. *Proceedings of the institution of civil engineers-geotechnical engineering, 137*(3), 137-147.

Indraratna, B., Ngo, N. T., Rujikiatkamjorn, C., & Sloan, S. W. (2015). Coupled discrete element–finite difference method for analysing the load-deformation behaviour of a single stone column in soft soil. *Computers and Geotechnics, 63*, 267-278.

Itasca, C. G. (1999). *PFC 3D-User manual*. Itasca Consulting Group, Minneapolis.

Itasca, C. G. (2004). *PFC3D (particle flow code in 3 dimensions) manual*. (Version 4.0 edn). Minneapolis, Minnesota.

Jiang, M., Zhu, H., & Li, X. (2010). Strain localization analyses of idealized sands in biaxial tests by distinct element method. *Frontiers of Architecture and Civil Engineering in China, 4*(2), 208-222.

Kafui, K., Thornton, C., & Adams, M. (2002). Discrete particle-continuum fluid modelling of gas–solid fluidised beds. *Chemical Engineering Science, 57*(13), 2395-2410.

Karn, U. (2016). *A Quick Introduction to Neural Networks*. Consulted on August 11, 2016 available at https://ujjwalkarn.me/2016/08/09/quick-intro-neural-networks/

Kenney, T., & Lau, D. (1985). Internal stability of granular filters. *Canadian Geotechnical Journal, 22*(2), 215-225.

Keyes, D. E., McInnes, L. C., Woodward, C., Gropp, W., Myra, E., Pernice, M., . . . Connors, J. (2013). Multiphysics simulations Challenges and opportunities. *International Journal of High Performance Computing Applications, 27*(1), 4-83.

Kloss, C., & Goniva, C. (2011). LIGGGHTS–Open Source Discrete Element Simulations of Granular Materials Based on Lammps. *Supplemental Proceedings: Materials Fabrication, Properties, Characterization, and Modeling, Volume 2*, 781-788.

Kovács, G. (2011). Seepage hydraulics (Vol. 10). Elsevier.

Kozicki, J., & Donzé, F. (2009). Yade-open dem: an open-source software using a discrete element method to simulate granular material. *Engineering computations, 26*(7), 786-805.

Kriebitzsch, S., Van der Hoef, M., & Kuipers, J. (2013). Fully resolved simulation of a gas-fluidized bed: a critical test of DEM models. *Chemical Engineering Science, 91*, 1-4.

Kundu, P. K., Cohen, I. M., & Dowling, D. R. (2012). *Fluid Mechanics, 5th Edition*. Academic, Berlin.

Levenberg, K. (1944). A method for the solution of certain non-linear problems in least squares. *Quarterly of applied mathematics*, 2(2), 164-168.

Lewis, R. W., Schrefler, B. A., & Rahman, N. A. (1998). A finite element analysis of multiphase immiscible flow in deforming porous media for subsurface systems. *Communications in numerical methods in engineering, 14*(2), 135-149.

Liu, M., & Liu, G. (2010). Smoothed particle hydrodynamics (SPH): an overview and recent developments. *Archives of computational methods in engineering, 17*(1), 25-76.

Liu, Y., Sun, W., Yuan, Z., & Fish, J. (2016). A nonlocal multiscale discrete-continuum model for predicting mechanical behavior of granular materials. *International Journal for Numerical Methods in Engineering, 106*(2), 129-160.

Logg, A., Mardal, K.-A., & Wells, G. (2012). Automated solution of differential equations by the finite element method: The FEniCS book (Vol. 84). Springer Science & Business Media.

Lominé, F., Scholtès, L., Sibille, L., & Poullain, P. (2013). Modeling of fluid–solid interaction in granular media with coupled lattice Boltzmann/discrete element methods:

application to piping erosion. *International Journal for Numerical and Analytical Methods in Geomechanics, 37*(6), 577-596.

Lu, M., & McDowell, G. (2006). Discrete element modelling of ballast abrasion. *Geotechnique, 56*(9), 651-655.

Maknoon, M., & Mahdi, T.-F. (2010). Experimental investigation into embankment external suffusion. *Natural hazards, 54*(3), 749-763.

Marquardt, D. W. (1963). An algorithm for least-squares estimation of nonlinear parameters. *Journal of the society for Industrial and Applied Mathematics, 11*(2), 431-441.

Mindlin, R., & Deresiewicz, H. (1953). Timoshenko's shear coefficient for flexural vibrations of beams. Columbia university New York.

Moffat, R. (2005). *Experiments on the internal stability of widely graded cohesionless soils* Ph.D. thesis, Department of Civil Engineering, The University of British Columbia, Vancouver, B.C.

Mohamed, A., & Gutierrez, M. (2010). Comprehensive study of the effects of rolling resistance on the stress–strain and strain localization behavior of granular materials. *Granular matter, 12*(5), 527-541.

Nardi, A., Idiart, A., Trinchero, P., de Vries, L. M., & Molinero, J. (2014). Interface Comsol-PHREEQC (iCP), an efficient numerical framework for the solution of coupled multiphysics and geochemistry. *Computers & Geosciences, 69*, 10-21.

Narsilio, G. A., Buzzi, O., Fityus, S., Yun, T. S., & Smith, D. W. (2009). Upscaling of Navier–Stokes equations in porous media: Theoretical, numerical and experimental approach. *Computers and Geotechnics, 36*(7), 1200-1206.

Nawi, N. M., Khan, A., & Rehman, M. Z. (2013). A new back-propagation neural network optimized with cuckoo search algorithm. *In International Conference on Computational Science and Its Applications*, 413-426. Springer.

Ng, A. K., & Small, J. C. (1999). A case study of hydraulic fracturing using finite element methods. *Canadian Geotechnical Journal, 36*(5), 861-875.

Nguyen, T., Combe, G., Caillerie, D., & Desrues, J. (2013). Modeling of a cohesive granular materials by a multi-scale approach. *Powders and Grains, 1542*(1194-1197), 106.

O'Sullivan, C. (2014). *Particulate discrete element modelling: a geomechanics perspective*. London: CRC Press.

O'Sullivan, C. (2015). Advancing geomechanics using DEM. *In International Symposium on Geomechanics from Micro to Macro*, IS-Cambridge 2014, September 1, 2014 - September 3, 2014 (Vol. 1, pp. 21-32). Taylor and Francis - Balkema.

O'Sullivan, C., Cui, L., & O'Neill, S. C. (2008). Discrete element analysis of the response of granular materials during cyclic loading. *Soils and foundations, 48*(4), 511-530.

Patzák, B., & Bittnar, Z. (2001). Design of object oriented finite element code. *Advances in Engineering Software, 32*(10), 759-767.

Perović, N., Frisch, J., Salama, A., Sun, S., Rank, E., & Mundani, R.-P. (2017). Multi-scale high-performance fluid flow: Simulations through porous media. *Advances in Engineering Software, 103*, 85-98.

Philippe, P., Beguin, R., & Faure, J. (2013). Contact Erosion. Bonelli, S. & Nicot, F. (Eds), *In Erosion in Geomechanics Applied to Dams and Levees*. Hoboken, NJ: John Wiley & Sons.

Pirnia, P., Duhaime, F., Ethier, Y., & Dubé, J.-S. (2019). Drag Force Calculations in Polydisperse DEM Simulations with the Coarse-Grid Method: Influence of the Weighting Method and Improved Predictions Through Artificial Neural Networks. *Transport in Porous Media*, 1-17.

Pirnia, P., Duhaime, F., Ethier, Y., & Dubé, J.-S. (2019). ICY: an interface between comsol multiphysics and discrete element code yade for the modelling of porous media. *Computers & Geosciences, 123*, 38-46.

Pirnia, P., Duhaime, F., Éthier, Y., & Dubé, J.-S. (2016). Development of a multiscale numerical modelling tool for granular materials presented at *The 69th Canadian Geotechnical Conference*, Vancouver, BC.

Plimpton, S. (1995). Fast parallel algorithms for short-range molecular dynamics. *Journal of computational physics, 117*(1), 1-19.

Reynolds, O. (1885). LVII. On the dilatancy of media composed of rigid particles in contact. With experimental illustrations. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science, 20*(127), 469-481.

Roache, P.J.: Quantification of uncertainty in computational fluid dynamics. Annu. Rev. Fluid Mech. 29, 123–160 (1997)

Rotunno, A. F., Callari, C., & Froiio, F. (2019). A finite element method for localized erosion in porous media with applications to backward piping in levees. International *Journal for Numerical and Analytical Methods in Geomechanics, 43*(1), 293-316.

Rubinstein, G. J., Derksen, J., & Sundaresan, S. (2016). Lattice Boltzmann simulations of low-Reynolds-number flow past fluidized spheres: effect of Stokes number on drag force. *Journal of Fluid Mechanics, 788*, 576-601.

Sakthivadivel, R. (1966). Theory and mechanism of filtration of non-colloidal fines through a porous medium. Berkeley, University of California.

Santasusana Isach, M. (2013). Kratos Dem, a parallel code for concrete testing simulations using the discrete element method, Universitat Politècnica de Catalunya.

Scheidegger, A. (1958). *The physics of flow through porous media*. University Of Toronto Press: London.

Sharif, N. H., Wiberg, N.-E., & Levenstam, M. (2001). Free surface flow through rock-fill dams analyzed by FEM with level set approach. *Computational mechanics, 27*(3), 233-243.

Sherard, J. L., & Dunnigan, L. P. (1985). Filters and leakage control in embankment dams. Dans Seepage and leakage from dams and impoundments (pp. 1-30). ASCE.

Sherard, J. L., & Dunnigan, L. P. (1989). Critical filters for impervious soils. *Journal of Geotechnical Engineering, 115*(7), 927-947.

Sherard, J. L., Dunnigan, L. P., & Talbot, J. R. (1984). Basic properties of sand and gravel filters. *Journal of Geotechnical Engineering, 110*(6), 684-700.

Shire, T., O'Sullivan, C., Hanley, K., & Fannin, R. (2014). Fabric and effective stress distribution in internally unstable soils. *Journal of geotechnical and geoenvironmental engineering, 140*(12), 04014072.

Skempton, A., & Brogan, J. (1994). Experiments on piping in sandy gravels. *Geotechnique, 44*(3), 449-460.

Šmilauer, V., et al., 2015a. *Yade Documentation*, second ed. The Yade Projecthttps://doi.org/10.5281/zenodo.34073. http://yade-dem.org/doc/.

Šmilauer, V., et al., 2015b. Reference manual. *Yade Documentation*, second ed. The Yade Projecthttps://doi.org/10.5281/zenodo.34045. http://yade-dem.org/doc/.

Šmilauer V, Catalano E, Chareyre B, Dorofeenko S, Duriez J, Gladky A, Kozicki J, Modenese C, Scholtès L, Sibille L (2015) , Reference manual. In Yade Documentation 2nd ed. The Yade Project , DOI 10.5281/zenodo.34045 (http://yade-dem.org/doc/)

Steeb, H., Diebels, S., & Vardoulakis, I. (2005). A multiphase continuum-based model capturing erosion and deposition. *Trends in Applications of Mathematics to Mechanics*, 519–528.

Steeb, H., Diebels, S., & Vardoulakis, I. (2007). Modeling internal erosion in porous media presented at Geo-Denver 2007, New Peaks in Geotechnics, Denver, Colorado.

Stránský, J., & Jirásek, M. (2012). Open source FEM-DEM coupling. *Engineering Mechanics, 18*.

Sutmann, G. (2002). Classical molecular dynamics. In J. Grotendorst, D. Marx, and A. Muramatsu (Eds.), Quantum Simulations of Complex Many-Body Systems: From Theory to Algorithms, Lecture Notes, Volume 10, pp. 211–254. John von Neumann Institute for Computing, Julich, NIC Series.

Terzaghi, K. (1925). Principles of soil mechanics, IV—Settlement and consolidation of clay. *Engineering News-Record, 95*(3), 874-878.

Thornton, C. (2000). Numerical simulations of deviatoric shear deformation of granular media. *Geotechnique, 50*(1), 43-53.

Tomlinson, S. S., & Vaid, Y. (2000). Seepage forces and confining pressure effects on piping erosion. *Canadian Geotechnical Journal, 37*(1), 1-13.

Trebotich, D., Straalen, B., Graves, D., & Colella, P. (2008). Performance of embedded boundary methods for CFD with complex geometry, J. Phys. Conf. Ser., 125, 012083.

Trussell, R. R., & Chang, M. (1999). Review of flow through porous media as applied to head loss in water filters. *Journal of Environmental Engineering, 125*(11), 998-1006.

Tsuji, Y., Kawaguchi, T., & Tanaka, T. (1993). Discrete particle simulation of two-dimensional fluidized bed. *Powder technology, 77*(1), 79-87.

Tran, VDH., Meidani, M., & Meguid, MA. (2018). Coupled 3D Finite-Discrete Element Analysis Tool: User Manual Rev. 2. McGill University.

Vardoulakis, I., Papanastasiou, P., & Stavropoulou, M. (2001). Sand erosion in axial flow conditions. *Transport in porous media, 45*(2), 267-280.

Vardoulakis, I., Stavropoulou, M., & Papanastasiou, P. (1996). Hydro-mechanical aspects of the sand production problem. *Transport in porous media, 22*(2), 225-244.

Vaughan PR, Soares HF 1982. Design of Filters for Clay Cores of Dams. *Journal of Geotechnical Engineering, 108*, No. GT1(January): 17-31.

Wan, C. F., & Fell, R. (2002). Investigation of internal erosion and piping of soils in embankment dams by the soil slot erosion test and the hole erosion test. University of New South Wales, School of Civil and Environmental Engineering.

Wan, C. F., & Fell, R. (2004). Investigation of rate of erosion of soils in embankment dams. *Journal of geotechnical and geoenvironmental engineering, 130*(4), 373-380.

Wang, K., & Sun, W. (2016). A semi-implicit discrete-continuum coupling method for porous media based on the effective stress principle at finite strain. *Computer Methods in Applied Mechanics and Engineering, 304*, 546-583.

Wang, M., Feng, Y., Pande, G., & Zhao, T. (2018). A coupled 3-dimensional bonded discrete element and lattice Boltzmann method for fluid-solid coupling in cohesive geomaterials. *International Journal for Numerical and Analytical Methods in Geomechanics, 42*(12), 1405-1424.

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*(1), 339-352.

Weatherley, D. (2009). *ESyS-Particle v2. 0 user's guide*.

Wen, C. Y. (1966). Mechanics of fluidization. *Chemical Engineering Progress, 62*, 100-111.

Wörman, A., & Olafsdottir, R. (1992). Erosion in a granular medium interface. *Journal of Hydraulic Research, 30*(5), 639-655.

YADE/Christian Jakob, 2012. Comparisons with PFC3D. [ONLINE] Available at: https://yade-dem.org/wiki/Comparisons_with_PFC3D, Accessed date: 24 May 2018.

Yao, M., & Anandarajah, A. (2003). Three-dimensional discrete element method of analysis of clays. *Journal of Engineering Mechanics, 129*(6), 585-596.

Zeghal, M., & El Shamy, U. (2004). A continuum-discrete hydromechanical analysis of granular deposit liquefaction. *International Journal for Numerical and Analytical Methods in Geomechanics, 28*(14), 1361-1383.

Zhang, D.-M., Gao, C.-P., & Yin, Z.-Y. (2019). CFD-DEM modeling of seepage erosion around shield tunnels. *Tunnelling and Underground Space Technology, 83*, 60-72.

Zhang, L., & Chen, Q. (2006). Seepage failure mechanism of the Gouhou rockfill dam during reservoir water infiltration. *Soils and foundations, 46*(5), 557-568.

Zhang, L., & Du, J. (1997). Effects of abutment slopes on the performance of high rockfill dams. *Canadian Geotechnical Journal, 34*(4), 489-497.

Zhang, L., Xu, Y., & Jia, J. (2009). Analysis of earth dam failures: A database approach. *Georisk, 3*(3), 184-189.

Zhao, J., & Shan, T. (2013). Coupled CFD–DEM simulation of fluid–particle interaction in geomechanics. *Powder technology, 239*, 248-258.

Zhu, H., Zhou, Z., Yang, R., & Yu, A. (2007). Discrete particle simulation of particulate systems: theoretical developments. *Chemical Engineering Science, 62*(13), 3378-3396.

Zhu, H., Zhou, Z., Yang, R., & Yu, A. (2008). Discrete particle simulation of particulate systems: a review of major applications and findings. *Chemical Engineering Science, 63*(23), 5728-5770.

Zienkiewicz, O., & Taylor, R. (2000). *The finite element method* (5th edition), volume 1, Oxford: Butterworth-Heinemann.

Zwart, S. P., McMillan, S., Harfst, S., Groen, D., Fujii, M., Nualláin, B. Ó., . . . Hut, P. (2009). A multiphysics and multiscale software environment for modeling astrophysical systems. *New Astronomy, 14*(4), 369-378.