A pore-scale thermo–hydro-mechanical model for particulate systems

Robert Caulk, Luc Scholtes, Marek Krzaczek, Bruno Chareyre

Version of Record: https://www.sciencedirect.com/science/article/pii/S0045782520304771
Manuscript_86458b225ecbfaf5460489dda0af7ee9

A pore-scale Thermo-Hydro-Mechanical coupled model for particulate systems

Robert Caulk[1,*], **Luc Sholtès**[2], **Marek Krzaczek**[3], **and Bruno Chareyre**[1]

[1]Univ. Grenoble Alpes, Grenoble INP, Laboratoire 3SR, Grenoble, France
[2]Université de Lorraine, CNRS, GeoRessources, Nancy, France
[3]Faculty of Civil and Environmental Engineering, Gdansk University of Technology, Gdansk, Poland
[*]rob.caulk@gmail.com

ABSTRACT

A pore scale numerical method dedicated to the simulation of heat transfer and associated thermo-hydro-mechanical couplings in granular media is described. The proposed thermo-hydro-mechanical approach is based on an existing hydro-mechanical model that combines the discrete element method for simulating the mechanical behavior of dense sphere packings with the finite volume method for simulating pore space fluid flow and the hydro-mechanical coupling. Within the hydro-mechanical framework, the pore space is discretized as a tetrahedral network defined by the triangulation of discrete element method (DEM) particle centers. It is this discretization of DEM particle contacts and tetrahedral pore spaces that enables the efficient conductive and advective heat transfer models proposed herein. In particular, conductive heat transfer is modeled explicitly between and within solid and fluid phases: across DEM particle contacts, between adjacent tetrahedral pores, and between pores and incident particles. Meanwhile, advective heat transfer is added to the existing implicit fluid flow scheme by estimating mass energy flux from pressure induced fluid fluxes. In addition to the heat transfer model, a thermo-mechanical coupling is implemented by considering volume changes based on the thermal expansion of particles and fluid. The conduction and advection models are verified by presenting comparisons to an analytical solution for conduction and a fully resolved numerical solution for conduction and advection. Finally, the relevance of the fully coupled thermo-hydro-mechanical model is illustrated by simulating an experiment where a saturated porous rock sample is subjected to a cyclic temperature loading.

To be submitted to: Computer Methods in Applied Mechanics and Engineering

Keywords: DEM, pore network, heat transfer, fluid flow, thermo-hydro-mechanical couplings

1 Introduction

Understanding heat transfer throughout particulate systems is of great interest for many scientific disciplines and engineering applications such as environmental sciences, chemical and food processing, powder metallurgy or energy management. Whether the granular medium is packed or fluidized, dry or

List of symbols

$\mathbf{M}$	Diagonal matrix of particle masses
$\mathbf{f}$	Particle forces
$\ddot{\mathbf{x}}$	Particle accelerations
$k_{ij}^{n/s}$	Normal/shear stiffness for particle interaction
$\partial\Theta_i$	Pore contour for tetrahedra i
Θ_i	Domain of tetrahedron i
K	Fluid compressibility
S_{ij}^{f}	Fluid area of facet shared by pores i and j
S_{ij}^{p}	Contact area of particles i and j.
$\mathbf{G}$	Symmetric conductivity matrix for flow solution
$\mathbf{p}$	Vector containing the pressure within each pore
$\dot{\mathbf{V}}$	Vector of rate of volume changes
μ	Mass-energy-flux
Φ	Heat-flux through the pore boundary
Φ_f	Conductive heat flux within the fluid phase
Φ_s	Conductive heat flux between solid and fluid phases
Φ_p	Conductive heat flux between particles
c_f	Heat capacity of fluid
c_p	Heat capacity of particle
T	Temperature of particle or pore
$\dot{U}_i$	Change of internal energy for pore i
A_{ik}	Spherical triangle area shared by particle i and pore k
V_k	Volume of pore k
α	Thermal diffusivity

saturated, heat transfer can occur by conduction within each phase, and across their common interfaces, by advection if the fluid flows, and by radiation (Kunii and Smith (1960); Vargas and McCarthy (2001)). All these processes often act together, in a combined manner, and generally involve volumetric changes of the constituent phases which inevitably produce complex coupled thermo-hydro-mechanical (THM) responses. Despite substantial efforts in recent years, the diversity of THM applications combined with the computational expense associated with general THM solutions demonstrates the need for a computationally efficient particulate THM model.

Traditionally, THM models are set up through continuum approaches based on mathematical frameworks combining sets of equations describing thermodynamics, solid mechanics and hydraulics principles (see for instance the finite element implementations of such concepts by Olivella et al. (1996) and Kolditz et al. (2012), or the finite difference scheme proposed by Rutqvist et al. (2002)). Nonetheless, even though attractive for macro-scale applications, continuum modeling approaches based on the finite element method

(FEM) or the finite difference method (FDM) suffer critical computational and continuity limitations when applied to discontinuous and highly deformable media such as packed or fluidized beds, granular or fractured materials. On the other hand, discrete approaches like, for instance, the discrete element method (DEM, Cundall and Strack (1979)), have proven successful at modeling the behavior of these discrete systems. The strength of DEM for modelling particulate systems has opened up recent efforts to extend its predictive capabilities to THM processes.

Many studies have proposed hydraulic or thermal couplings with DEM (see for instance the HM schemes proposed in Zeghal and El Shamy (2004), Shimizu (2011), Lominé et al. (2013) or Catalano et al. (2014), and the TM schemes proposed in Feng et al. (2008), Tsory et al. (2013), André et al. (2017), Chen et al. (2018), or Joulin et al. (2020)). However, approaches including both thermal and hydraulic components into their formulation remain scarce. Most of these studies combine computational fluid dynamics (CFD) schemes with the DEM. Generally, CFD-DEM schemes are based on computations involving volume average of TH quantities evaluated on fixed/Eulerian coarse grids built over the domain covered by the particles. The grids define subdomains over which the porosity and velocity of the solid phase are averaged and introduced as field variables in the continuum formulation. Most CFD-DEM schemes for THM computations rely on finite difference approximations of the equations governing fluid flow and heat transfers (see, *e.g.*, Al-Arkawazi (2018); Kloss et al. (2014); Shimizu (2006); Zhou et al. (2009)). The interaction between the solid and fluid phases rely on empirical closures provided in the form of correlations required to depict the momentum exchange as well as heat and mass transfers (Deen et al. (2007)). These closure correlations are often empirical but can also be derived from direct numerical simulations (DNS) (Kruggel-Emden et al. (2016). Although DNS-DEM models can be used to study THM processes in particulate systems (Deen et al. (2012)), the small mesh size to particle size ratio results in high computational effort. Thus, DNS-DEM models are restricted to systems comprised of a smaller number of particles than CFD. An alternative particulate THM coupling is based on the lattice Boltzmann method (LBM) (Yang et al. (2017); Zhang et al. (2016)). However, the coupling strategies rely on computations of distribution functions that require an accurate representation of solid-fluid boundaries which can be both numerically and computationally challenging (Peng and Luo (2008)).

An example of a DEM based THM model, presented by Tomac and Gutierrez (2015), includes

convective and conductive heat transfer processes in 2-D. Although the scheme neglects heat induced fluid expansion and heat conduction within the fluid phase, it applies a well tested pipe network model initially proposed by Cheng et al. (1999) and thoroughly verified by Wanne and Young (2008) and Feng et al. (2009).

The 3-D THM coupled model presented here is based on the framework of the pore-scale finite volume (PFV) scheme initially proposed by Chareyre et al. (2012) for up-scaling incompressible viscous flow and later extended to compressible flow by Scholtès et al. (2015). The scheme is derived from the pore network (PN) models, which are applied widely for simulating a variety of porous media processes (Blunt, 2001). The model is implemented in the Yade DEM open source software (Šmilauer V. et al., 2015) and is oriented toward dense grain packing applications as encountered in geomechanics. The proposed THM scheme combines four heat transfer models: a particle-particle conduction model, a particle-fluid conduction model, a fluid-fluid conduction model and a heat advection model. In addition, thermo-mechanical couplings are considered through both effects of fluid thermal expansion and particle thermal expansion.

In summary, a set of equations governing the hydraulic and thermal schemes are presented for the derivation of the geometrical considerations and numerical couplings. Next, a validation exercise is provided where each component of the proposed THM model is challenged. First, the solid-solid conduction scheme is verified against the 1-D analytical solution of the classic heat conduction equation. Second, the convective heat transfer model (encompassing conduction combined with advection) is compared to a fully resolved CFD solution considering the flow of a hot fluid through a cold particle assembly. Third, the full THM scheme is used to simulate an experiment where a saturated rock sample is subjected to thermal loading.

2 Methods

2.1 Mechanical scheme

The Lagrangian discrete element method (DEM) represents the mechanical behavior of a particulate system as a collection of interacting masses, where interactions between masses follow predefined force-displacement laws. Similar to the original DEM scheme (Cundall and Strack, 1979), the present scheme

implemented in Yade open DEM integrates particle positions through time according to Newton's second law of motion, which can be written for the whole system as:

$$\mathbf{M}\ddot{\mathbf{x}} = \mathbf{f} \tag{1}$$

with $\ddot{\mathbf{x}}$ the vector containing each particle acceleration, $\mathbf{M}$ the diagonal matrix of particle masses, and $\mathbf{f}$ the vector containing the total forces applied on the particles. The explicit central finite difference time stepping scheme integrates the particle acceleration from the current step to update the particle position at the next step (see (Šmilauer V. et al., 2015) for details of the implementation). The inter-particle forces, $\mathbf{f}_{ij}$, depend on a contact model, F_{ij}, such that:

$$\frac{\partial \mathbf{f}_{ij}}{\partial t} = \mathrm{F}_{ij}(\mathbf{x}_i, \mathbf{x}_j, \dot{\mathbf{x}}_i, \dot{\mathbf{x}}_j) \tag{2}$$

2.2 Compressible flow scheme

The pore-finite volume (PFV) scheme is used to model compressible fluid flow between solid particles (Scholtès et al., 2015). As presented in Chareyre et al. (2012), the PFV-DEM coupling involves a weighted Delaunay triangulation of particle centers to form a tetrahedral mesh (Fig. 1). The solid volume within each tetrahedron is defined by the intersection of each tetrahedron with its vertex DEM spheres. Meanwhile, the fluid fraction consumes the remainder of each tetrahedron to form individual pores. Each pore is connected to four neighboring pores to constitute a pore network where a Stokes-flow is established based on an integral form of the continuity equation:

$$\int_{\Theta_i} \frac{\partial \rho_f}{\partial t} \mathrm{d}V = -\int_{\Theta_i} \nabla \cdot (\rho_f \mathbf{u}) \mathrm{d}V \tag{3}$$

where Θ_i is the domain of pore i, ρ_f is the fluid density, and $\mathbf{u}$ is the fluid velocity. Application of the divergence theorem reduces the volume integral to a contour integral:

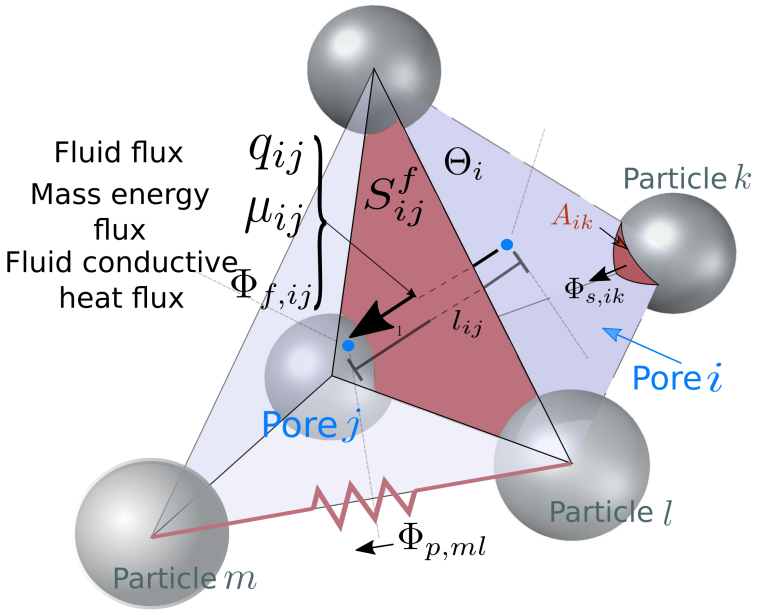

Figure 1. Heat transfer model notations and geometric considerations for fluid flow, advection, and conduction models.

$$\int_{\Theta_i} \frac{\partial \rho_f}{\partial t} \mathrm{d}V = -\int_{\partial\Theta_i} \rho_f (\mathbf{u}-\mathbf{v}) \cdot \mathbf{n} \mathrm{d}S - \int_{\partial\Theta_i} \rho_f \mathbf{v} \cdot \mathbf{n} \mathrm{d}S \tag{4}$$

where $\partial\Theta_i$ is the pore contour, $\mathbf{v}$ is the contour velocity, and $\mathbf{n}$ is the outward pointing unit vector. The consideration of fluid compressibility follows the relation of fluid bulk modulus, K, to the change of fluid pressure with respect to density:

$$K = \rho_f \frac{\partial P_i}{\partial \rho_f} \tag{5}$$

where K can be a function of fluid pressure and air fraction as highlighted in Eq. 13. Finally, reducing $\partial\Theta_i$ to only the fluid fractions, S_{ij}^f, of the pore contour and assuming small Mach numbers, Eq. 4 becomes:

$$\int_{\Theta_i} \frac{1}{K} \frac{\partial p_i}{\partial t} \mathrm{d}V = \sum_{j=1}^{4} \int_{S_{ij}^f} (\mathbf{u}-\mathbf{v}) \cdot \mathbf{n} \, \mathrm{d}S - \dot{V}_i \tag{6}$$

where $\dot{V}_i$ is the rate of pore volume change and the integral on the right hand side represents the sum of fluid fluxes exchanged by each pore and its four neighbors (j=1 to 4):

$$\sum_{j=1}^{4} \int_{S_{ij}^f} (\mathbf{u} - \mathbf{v}) \cdot \mathbf{n} \, \mathrm{d}S = \sum_{j=1}^{4} q_{ij} \tag{7}$$

At low Reynolds numbers, the flux q_{ij} through the pore throat connecting pore i and j is proportional to a local pressure gradient, with a coefficient k_{ij} reflecting the local conductivity g_{ij}:

$$q_{ij} = g_{ij}(p_i - p_j) \tag{8}$$

Hence,

$$\sum_{j=1}^{4} g_{ij}(p_i - p_j) = \dot{V}_i + \frac{\dot{p}_i V_i}{K} \tag{9}$$

which is the discrete form of a diffusion equation. Within Eq. 9, the pressures in the neighboring pores are represented by p_i and p_j and the length of the pore throat is l_{ij}. The conductivity, g_{ij}, was defined in Chareyre et al. (2012):

$$g_{ij} = k_d \frac{S_{ij}^f}{\mu(T) l_{ij}} \tag{10}$$

where k_d is the permeability between pores i and j, S_{ij}^f is the area of the facet shared by pore i and j, and $\mu(T)$ is the temperature dependent fluid viscosity.

The matrix representation of the full linear system representing Eq. 9 for all pores can be expressed as:

$$\mathbf{G}\mathbf{p} = \dot{\mathbf{V}} \tag{11}$$

where $\mathbf{G}$ is the conductivity matrix, $\mathbf{p}$ is the vector containing the pressure within each pore, and $\dot{\mathbf{V}}$ is the vector of rate of volume changes, which is composed of the volume changes due to: particle movements ($\dot{V}_G$), particle thermal expansion ($\dot{V}_{tm}$), and fluid thermal expansion($\dot{V}_{th}$):

$$\dot{\mathbf{V}} = \dot{\mathbf{V}}_G + \dot{\mathbf{V}}_{tm} + \dot{\mathbf{V}}_{th} \tag{12}$$

$\dot{\mathbf{V}}_\mathbf{G}$ depends linearly on the particle velocities, $\dot{\mathbf{x}}$, which can be expressed such that $\dot{\mathbf{V}}_G = \mathbf{E}\dot{\mathbf{x}}$, where $\mathbf{E}$ is essentially a tensor with projected surface area.

In the case of gas-liquid mixtures, an equivalent fluid compressibility C_{eq} is defined by following Najari and Selvadurai (2014):

$$C_{eq} = \phi C_a + (1-\phi)C_w \tag{13}$$

$$C_a = \frac{1}{P_a} \tag{14}$$

$$\phi = \frac{P_{a,0}}{P_a}\phi_0 \tag{15}$$

where ϕ is the fraction of gas within the mixture, C_a is the compressibility of gas, P_a is the absolute pressure of the gas and $P_{a,0}$ and ϕ_0 are the initial absolute gas pressure and gas fraction respectively. C_{eq} is updated at each time-step and used within the compressible flow scheme, Eq. 9, $K = 1/C_{eq}$.

2.3 Heat transfer

Advection is simulated by treating each pore as an open thermodynamic system assuming that:

- radiation can be neglected (T<700 K)
- fluid kinetic energy is negligible (0<Re<1000)
- fluid compression does not generate heat

Thus the integral form of the first law of thermodynamics becomes:

$$\Delta U_o = Q + \sum_{m=1}^{n} \Delta U_m \tag{16}$$

where U_o is the internal energy of the open system (one pore in the present pore network), Q is a source term representing the quantity of heat supplied to the system, and ΔU_m is the change of internal energy of the mth connected system (in the present pore network, these are neighboring pores and neighboring particles, Fig. 1). Considering the assumptions above, $\sum_{m=1}^{n} \Delta U_m$ is limited to mass energy transfer, μ, and boundary heat flux, Φ. Thus, ΔU_o can be expressed as a summation of surface integrals:

$$\Delta U_o = Q + \int_{\partial\Theta_i} \Phi \cdot \mathbf{n} \, \mathrm{d}S + \int_{\partial\Theta_i} \mu \cdot \mathbf{n} \, \mathrm{d}S \tag{17}$$

where $\mathbf{n}$ is the outward pointing unit vector.

2.3.1 Advective heat transfer

Starting with the mass energy flux integral, each pore of the present pore network abuts four neighboring pores (j=1 to 4), which means the integral can be reduced to a summation:

$$\int_S \mu \cdot \mathbf{n} \, \mathrm{d}S = \sum_{j=1}^{4} \int_{S_{ij}^f} \mu_{ij} \cdot \mathbf{n} \, \mathrm{d}S = \sum_{j=1}^{4} \mu_{ij} \tag{18}$$

where the pore-pore mass-energy-flux between home pore i and neighbor pore j, μ_{ij}, is the average over the shared facet and depends on the volumetric flow q_{ij} (see Sec. 2.2):

$$\mu_{ij} = q_{ij} c_f \rho_f T_u \tag{19}$$

$$T_u = \begin{cases} T_i \text{ if } q_{ij} > 0 \\ T_j \text{ if } q_{ij} < 0 \end{cases} \tag{20}$$

where T_u is the temperature of the home or neighbor pore depending on the flow direction, c_f, is the fluid heat capacity at constant pressure, and ρ_f is the fluid density.

As presented in Sec. 2.2, the fluid flux q_{ij}, is already solved (thus, the flow direction is known apriori) as described in Sec. 2.2 as **Gp**. Therefore, the change of internal energy due to mass-energy-flux for all pores can be computed as:

$$\frac{U_i^{t+\Delta t} - U_i^t}{\Delta t} = \left(c_f \rho_f \sum_{j=1}^{4} (T_u^t g_{ij}) \right) \left(p_i^{t+\Delta t} - p_j^{t+\Delta t} \right) \tag{21}$$

which, in matrix form, can be written as:

$$\Delta \mathbf{U} = \mathbf{HP} \tag{22}$$

with

$$\mathbf{H} = c_f \rho_f \Delta t \sum_{j=1}^{4} (T_u^t g_{ij}) \tag{23}$$

The internal energy, $\mathbf{U}^{t+\Delta t}$, is computed for all pores:

$$\mathbf{U}^{t+\Delta t} = \mathbf{U}^t + \mathbf{Hp} \tag{24}$$

and the temperature of the pores at $t + \Delta t$ is updated as:

$$\mathbf{T}^{t+\Delta t} = \frac{\mathbf{U}^{t+\Delta t}}{c_f \rho_f \mathbf{V}^{t+\Delta t}}. \tag{25}$$

2.3.2 Conductive heat transfer

Conductive heat transfer is simulated between interacting particles, between neighboring pores, and between pores and particles assuming:

- radiation is neglected (T$<$700 K)
- the resistance to heat transfer inside the particle is significantly smaller than between particles (Biot number = $\frac{h_i d_i}{k_i} \ll 1$)

Given these assumptions, the heat conduction equation for a single particle k follows:

$$m_k c_p \frac{\partial T_k}{\partial t} = \nabla \cdot \Phi_k + Q \tag{26}$$

where m_k is the mass of particle k, c_p is the particle heat capacity at constant pressure, T_k is the temperature of the particle, Φ_k is the boundary heat-flux into the particle and Q is the heat source supplied to the system. $\nabla \cdot \Phi_k$ is equivalent to the volume integral of the heat-flux, which can be reduced to a surface integral using the divergence theorem:

$$\nabla \cdot \Phi_k = \int_{\partial \Gamma} \Phi_k \cdot \mathbf{n} \mathrm{d}S \tag{27}$$

where $\partial\Gamma$ is the contour of particle k and $\mathbf{n}$ is the outward pointing unit vector. The surface integral is reduced to a summation along all pores and particles incident to particle k:

$$m_k c_p \frac{\partial T_k}{\partial t} = \sum_{w=1}^{N} \int_{S^p_{wk}} \Phi_{p,wk} \cdot \mathbf{n}_{wk} \mathrm{d}S + \sum_{u=1}^{M} \int_{A_{uk}} \Phi_{s,uk} \cdot \mathbf{n}_{uk} \mathrm{d}S \tag{28}$$

where S^p_{wk} and $\Phi_{p,wk}$ are the contact interface and average heat-flux, respectively, between particle w and k. and M and N are the number of pores and particles interacting with particle k, respectively.

Conductive heat-flux between contacting particles, $\Phi_{p,lm}$, follows existing methods by assuming the heat flux is linearly related to the inter particle temperature gradient by a thermal resistance (η) (Feng et al., 2008; Liang and Li, 2014):

$$\int_{S^p_{lm}} \Phi_{p,lm} \cdot \mathrm{n}_{lm} \mathrm{d}S = \eta (T_l - T_m) \tag{29}$$

where T_l is a unique temperature value for particle l and can be interpreted as the average particle temperature. η can be defined in various ways depending on the application of interest, as outlined in Sec. 3 and Sec. 4.

The first integral of Eq. 17, representing conductive heat-flux into the pore, is reduced to a summation by applying the divergence theorem:

$$\int_{\partial \Theta_i} \Phi \mathrm{d}S = \sum_{j=1}^{4} \int_{S^f_{ij}} \Phi_{f,ij} \cdot \mathbf{n}_{ij} \mathrm{d}S + \sum_{k=1}^{4} \int_{A_{ik}} \Phi_{s,ik} \cdot \mathbf{n}_{ik} \mathrm{d}S \tag{30}$$

where Φ_f is the conductive heat flux within the fluid phase, $\mathbf{n}_{ij}$ is the unit vector connecting pores i and j. Φ_s is the conductive heat flux between solid and fluid phases with $\mathbf{n}_{ik}$ being the unit vector connecting pore i center to particle k. The second term of Eq. 30, representing the heat-flux between particles and pores ($\Phi_{s,ik}$), matches the second term of the conservation of energy for each particle in Eq. 28. Therefore, the estimate is made once for each particle pore pair and follows a traditional spherical heat transfer approach (Incropera et al., 2007), reducing the surface integral as follows:

$$\int_{A_{ik}} \Phi_{s,ik} \cdot \mathbf{n}_{ik} \mathrm{d}S = h_{ik} A_{ik} (T_k - T_i) \tag{31}$$

where A_{ik} is the surface of particle k interacting with the pore i (spherical triangle shown in Fig. 1), T_i is the temperature of the pore k, T_k is the temperature of particle k, and h_{ik} the heat transfer coefficient. h_{ik}, is computed based on the Nusselt number Nu which can be empirically estimated using the macroscopic porosity of the particle assembly ($0.35< \varepsilon <1$) and Reynolds number ($0<\mathrm{Re}<10^2$) as proposed by Tavassoli et al. (2015):

$$Nu_k = (7 - 10\varepsilon + 5\varepsilon^2)(1 + 0.1 Re_k^{0.2} Pr_k^{1/3}) + (1.33 - 2.19\varepsilon + 1.15\varepsilon^2) Re_k^{0.7} Pr_k^{1/3} \tag{32}$$

with Pr Prandtl's number and Re the Reynolds number based on the volume average of the pore k fluid velocity. The heat transfer coefficient then becomes $h_{ik} = Nu_k \cdot k_f/(2r_i)$, with k_f as the thermal conductivity of the fluid.

Given the particle-fluid ($\Phi_{s,ik}$, Eq. 31) and particle-particle ($\Phi_{p,ij}$, Eq. 59) heat flux approximations, Eq. 28 is approximated using a forward Euler scheme to estimate the particle temperature change:

$$T_k^{t+\Delta t} = \frac{\Delta t}{m_i c_p} \left[\sum_{w=1}^{N} \frac{2r_c^2 (k_w + k_k)}{d_{wk}} (T_w^t - T_k^t) + \sum_{u=1}^{M} \frac{Nu_k k_f A_k}{2r_i} (T_u^t - T_k^t) \right] + T_k^t \tag{33}$$

where M is the number of incident pores and N is the number of contacting particles.

Similarly to the particle conductive heat transfer, pores conduct heat between one another through their interface. Each interface is characterized by a thermal resistance, which determines the heat flux based on the temperature difference between pores, reducing the surface integral in Eq. 30 to:

$$\int_{S_{ij}^f} \Phi_{f,ij} \cdot \mathbf{n}_{ij} \mathrm{d}S = \frac{k_f S_{ij}^f}{l_{ij}} (T_j - T_i) \tag{34}$$

$$\tag{35}$$

where l_{ij} is the distance between the centers of pores i and j, T_i and T_j are the temperatures of the respective pores, and S_{ij}^f is the fluid area shared by both pores (Fig. 1).

In the present implementation, conservation of energy is ensured by computing pore-pore conduction using mid-step pore temperatures based on the advective heat flux. In other words, pore temperatures are updated twice per time step, first by advection, then by conduction.

2.4 Thermo-Hydro-Mechanical coupling

2.4.1 Hydro-Mechanical coupling

The two way hydro-mechanical coupling is only briefly described here - a full description and validation can be found in Chareyre et al. (2012) and Scholtès et al. (2015). The hydraulic force exerted by the pore fluid on the particles is decomposed into two contour integrals of the hydrostatic pressure P and viscous shear stress τ:

$$\mathbf{F}^p = \int_{\Gamma_p} P \cdot \mathbf{n} \mathrm{d}S + \int_{\Gamma_p} \tau \cdot \mathbf{n} d\mathrm{S} = \mathbf{F}^{P,p} + \mathbf{F}^{\tau,p} \tag{36}$$

where Γ_p is the solid surface of the particle p. Hydrostatic pressure forces, $\mathbf{F}^{P,p}$, and viscous forces $\mathbf{F}^{\tau,p}$ are both added to the interparticle force estimates (Eq. 1). Finally, the effect of particles movement on the pore network morphology is taken into account through Eq. 12.

2.4.2 Thermo-mechanical coupling

The thermo-mechanical coupling results from thermal expansion/contraction of both fluid and particles. The associated pore volume changes are added to the existing rate of volume changes (Eq. 12) used within the flow solution (Eq. 11). In this way, the compressibility effects are handled inherently within the

flow solver. The pore volume change contribution by the expansion/contraction of the solid phase ($\dot{\mathbf{V}}_{tm}$) depends entirely on solid temperature change (Sec. 2.3.2) and is expressed as:

$$\Delta r_k^{t+\Delta t} = r_k^t \beta_p (T_k^{t+\Delta t} - T_k^t) \tag{37}$$

where β_p is the coefficient of thermal expansion of the particles, r_k is the radius of particle k, and T_k its temperature. The particle volume changes, $\sum_{k=1}^{4} \Delta V_i$, are used to compute $\dot{V}_{tm,i}$:

$$\Delta V_i = -\sum_{k=1}^{4} \frac{A_{ik}}{A_k} \frac{4}{3} \pi (r_{k,t+\Delta t}^3 - r_{k,t}^3) \tag{38}$$

$$\dot{V}_{tm,i} = \frac{\Delta V_i}{\Delta t} \tag{39}$$

where Δt is the thermal time step, A_{ik} is the spherical triangle area of particle k shared by pore i (Fig.¿1), and and A_k is the total surface area of particle k.

Meanwhile, the pore fluid thermal expansion/contraction is computed as:

$$\dot{V}_{th,i} = \frac{V_i \beta_f(T) \Delta T}{\Delta t} \tag{40}$$

where ΔT is the temperature change and $\beta_f(T)$ is the temperature dependent fluid volumetric coefficient of thermal expansion. Finally, $\dot{\mathbf{V}}_{th}$ and $\dot{\mathbf{V}}_{tm}$ contribute to the solution of fluid pressures (Eq. 12).

2.5 Porosity scaling

Various attributes can be scaled if the desired porosity is not equivalent to the exact packing porosity. For example, heat capacity, c_p, is scaled such that that the total heat storage is representative of the desired material:

$$c_p = c_{p,o}\frac{1-\phi_d}{1-\phi_p} \tag{41}$$

where $c_{p,o}$ is non-scaled heat capacity of the desired material, ϕ_d is the desired porosity, and ϕ_p is the DEM sphere packing porosity.

Similarly, the pore space can be scaled such that such that the volume of fluid, V_k, per tetrahedral matches the desired porosity:

$$V_k = V_{k,i}\frac{\phi_d}{\phi_p} \tag{42}$$

where $V_{k,i}$ is the geometric pore volume, ϕ_p is the porosity of the DEM sphere packing, ϕ_d is the desired porosity.

3 Analytical verification of particle-particle conduction scheme

The numerical conduction scheme for particle-particle heat conduction (Sec 2.3.2) was verified by comparing various thermal DEM simulations (without pore fluid and neglecting mechanical interactions between particles) to the 1D heat equation:

$$\frac{\partial T}{\partial t} = \alpha\frac{\partial^2 T}{\partial x^2} \tag{43}$$

The analytical solution to the 1D heat equation is constrained by the following initial and boundary conditions:

$$T(x,0) = 120\; x \in [0,L] \tag{44}$$

$$T(0,t) = T(L,t) = 0\; t \in [t > 0] \tag{45}$$

Using Fourier series, the unsteady solution becomes:

$$T(x,t) = \sum_{n=1}^{\infty} D_n \sin\left(\frac{n\pi x}{L}\right) \exp\left(\alpha \frac{n^2\pi^2 t}{L^2}\right) \tag{46}$$

$$D_n = \frac{2}{L}\int_0^L 120 \sin\left(\frac{n\pi x}{L}\right) \mathrm{d}x \tag{47}$$

where $L \approx 1$ is the domain length, α is the effective thermal diffusivity, and t is the time.

Within the DEM conduction scheme, thermal resistivity estimate (η) emulates a continuum by modeling heat flux through a wall with depth d and area A depending on the interacting particles radii:

$$\int_{S_{lm}^p} \Phi_{p,lm} \cdot \mathrm{n}_{lm} \mathrm{d}S = \eta (T_l - T_m) \tag{48}$$

$$\eta = \frac{(k_l + k_m)/2}{d} A \tag{49}$$

$$d = r_l + r_m \tag{50}$$

$$A = 4 r_l r_m \tag{51}$$

$$\tag{52}$$

Thermal micro-parameters of the DEM model are listed in Table 1. Readers can also find the practical input script for conduction in a single row of spheres titled "conductionVerification.py" in the supplementary material included with this paper.

Table 1. DEM Microparameters

Parameter	Value	units
k_p	2.0	W/(m K)
C_p	710	J/(kg K)
ρ_p	2600	kg/m^3
r	0.003	m

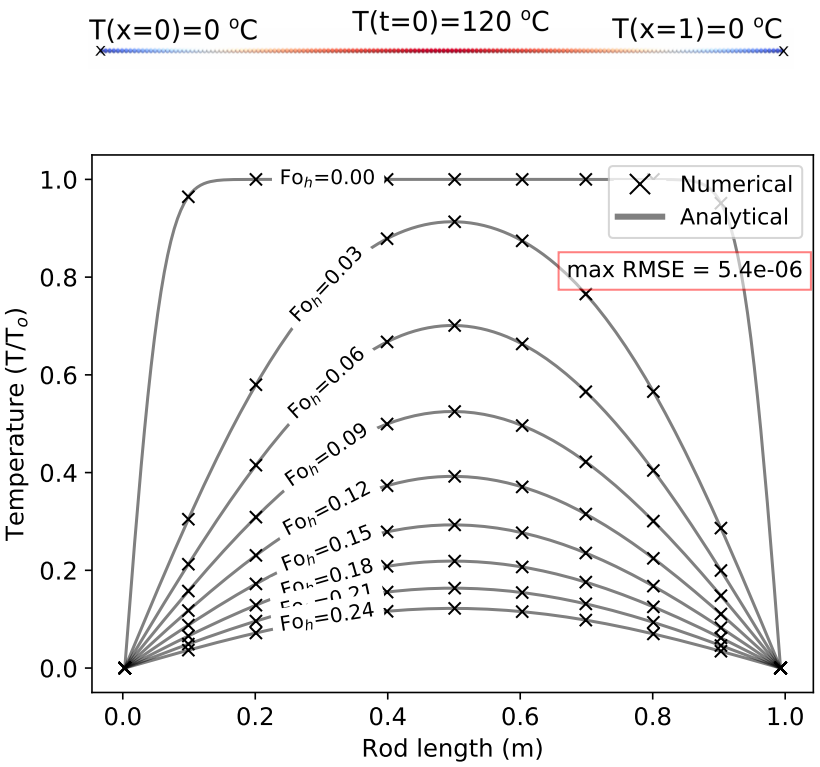

Figure 2. Evolution of temperature distribution for a single row of DEM particles compared to the 1D analytical solution, with $\mathrm{Fo}_h = \frac{\alpha t}{L^2}$.

3.1 Single row of spherical particles

A single row of DEM particles comprised of 166 particles was compared to the analytical solution for the 1D heat equation. The analytical solution was computed using an effective thermal diffusivity computed by scaling the density of DEM spherical elements to cubical continuum elements ($\rho_p \pi/6$):

$$\alpha = \frac{6k_p}{\pi C_p \rho_p}. \tag{53}$$

The boundary particles were set to a constant temperature (T=0°C) while the remaining particles were set to an initial temperature of 120°C. The results show that the numerical temperature distributions match the analytical solution with space and time as shown in Fig. 2, with a maximum RMSE of 5.4e-06 for all Fourier numbers.

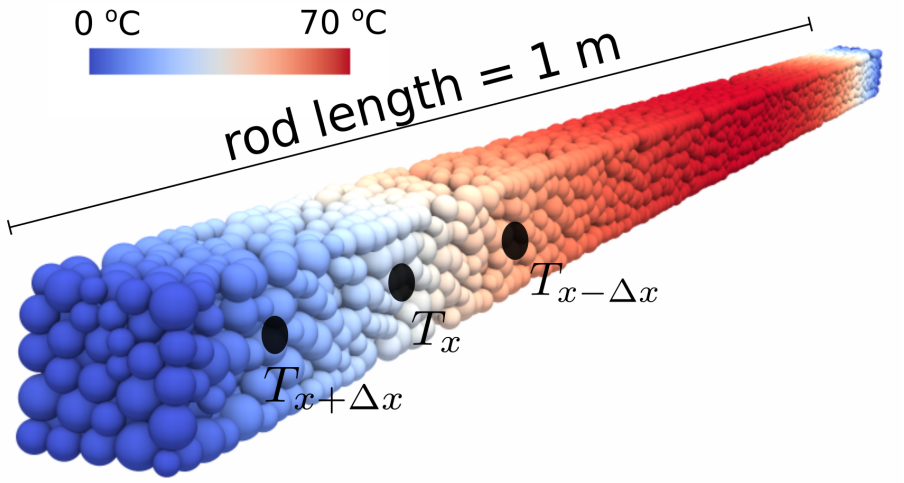

Figure 3. Temperature distribution within a random packing subjected to cooling at Fo_h=0.06. Effective thermal diffusivity is estimated considering the temperature evolution at three points along the axis: $T_{x+\Delta x}$, T_x, $T_{x-\Delta x}$

3.2 Random packings of spherical particles

The thermal diffusivity of random sphere packings cannot be estimated analytically. It can however be estimated numerically by simulating the 1D cooling scenario described in Sec. 3.1. Specifically, the thermal diffusivity α of the assembly is estimated considering the temperature evolution of three particles located along the axis of the packing (Fig. 3) as:

$$\alpha = \frac{\mathrm{d}T/\mathrm{d}t}{\mathrm{d}^2T/\mathrm{d}x^2} \tag{54}$$

where the first and second derivatives are approximated by:

$$\frac{\mathrm{d}T}{\mathrm{d}t} \approx \frac{T_x^t - T_x^{t-\Delta t}}{\Delta t} \tag{55}$$

$$\frac{\mathrm{d}^2T}{\mathrm{d}x^2} \approx \frac{T_{x+\Delta x}^t - 2T_x^t + T_{x-\Delta x}^t}{\Delta x^2} \tag{56}$$

The effective thermal diffusivity is monitored while the packing cools from an initial temperature of 120°C down to the boundary temperatures of 0°C as shown in Fig 4. The final plateaued value of thermal diffusivity can then be used in the 1D heat equation for comparison. The variation of effective thermal

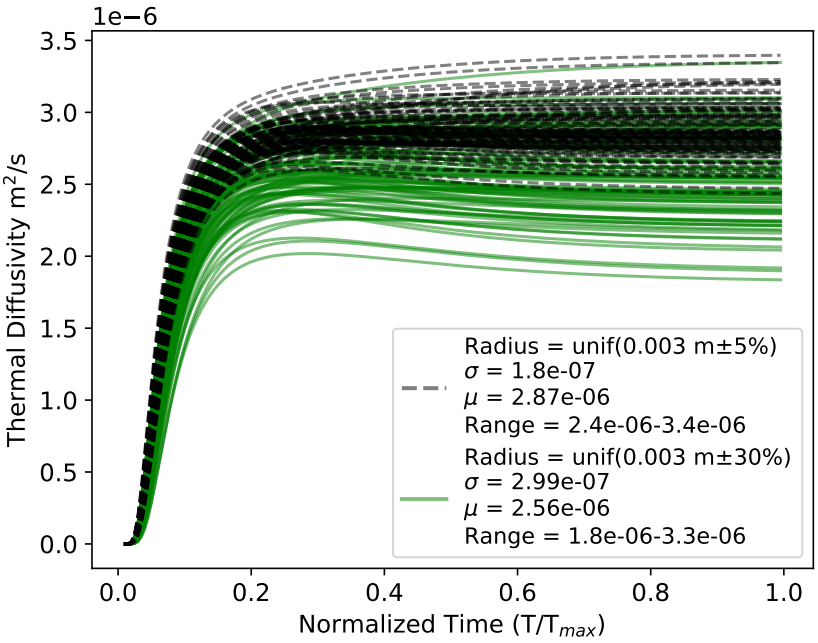

Figure 4. 100 realizations of effective thermal diffusivity estimation associated for random packings generated with two different particle size distributions.

diffusivity for random packings was investigated by running 100 realizations of the thermal diffusivity test as shown in Figures 4 and 5. Each packing realization contained the same uniform particle size distribution (ca. 970 particles, unif(0.003m±5%) or ca. 880 unif(0.003±30%)) but particle positions were randomly generated prior to each thermal diffusivity test. Following the 100 realizations, an estimated mean and standard deviation were assigned to the effective thermal diffusivity distribution of the material. As shown in Figures 4 and 5, increasing the range of particle sizes corresponds to an increase of effective thermal diffusivity spread. These parameter estimates also enabled an illustration of temperature distribution variance with space and time (Figure 6). As shown, the variance is greatest where and when the heat flux is highest (i.e. at center of the specimen during the middle of the simulation). In summary, using random sphere packings reproduce analytical solutions within an expected statistical variation that depends on the sphere size distribution. Further, the thermal diffusivity estimate process presented is also the calibration process of the particle thermal conductivity according to the desired macroscopic thermal diffusivity.

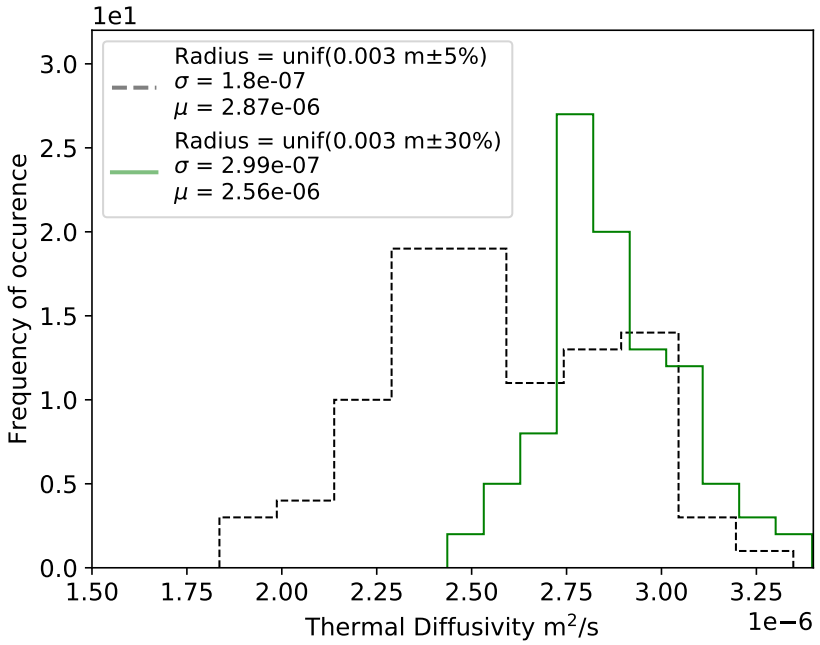

Figure 5. Distribution of effective thermal diffusivity estimate for two random packings generated with two different particle size distributions over 100 realizations.

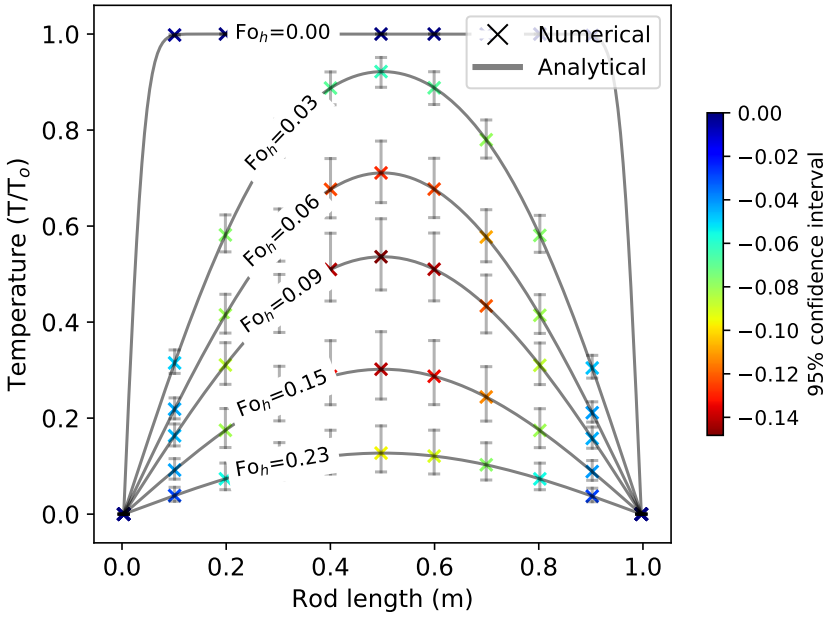

Figure 6. Evolution of temperature distribution for a random packing of spheres compared to 1D heat equation (α=2.8e-6 m^2/s), where confidence intervals are based on the standard deviations of 100 realizations. $\mathrm{Fo}_h = \frac{\alpha t}{L^2}$.

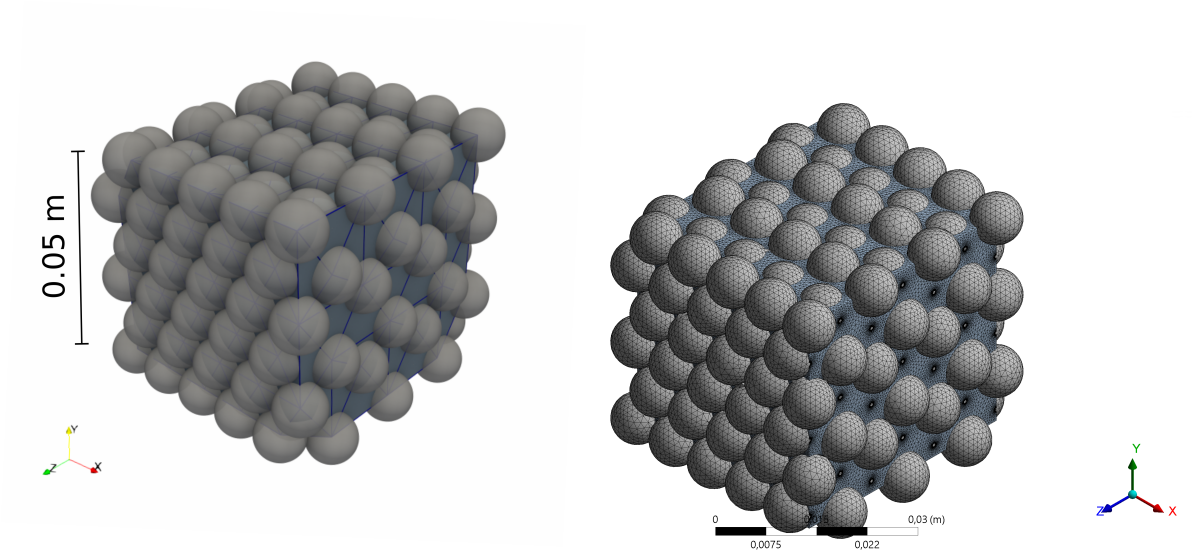

Figure 7. Comparison of left) DEM Yade sphere packing based pore network with right) ANSYS CFX mesh

4 Numerical verification of advection & conduction

A direct comparison was performed between the presently implemented THM-DEM model and a fully resolved thermo-hydraulic CFD FVM (ANSYS CFX) model. Two validation scenarios were performed on 5x5x5 cm sphere packings comprised of 211 spheres (Fig. 7). The first validation scenario, called "no-flow", started with spheres at an initial temperature of 60 °C and still-fluid at 70 °C (Fig. 8a). The second validation scenario, called "constant-flow", started with spheres and fluid at an initial temperature of 60 °C, while 70 °C warm fluid was flushed along the Y axis using a pressure gradient of 2 Pa/m (Fig. 8b)(Table 2). For both scenarios, the center body and center pore transient temperatures were compared. The THM-DEM and CFD models used identical material parameter values reported in Table 2. Readers can also find the practical input scripts for this section, titled "flowScenario.py" and "noFlowScenario.py", in Appendix C as well as the supplementary material included with this paper.

4.1 THM-DEM Thermal Resistivity

The conductive heat flux between two contacting DEM particles, $\Phi_{p,lm}$, was defined using the contact area, as presented in (Norouzi et al., 2016):

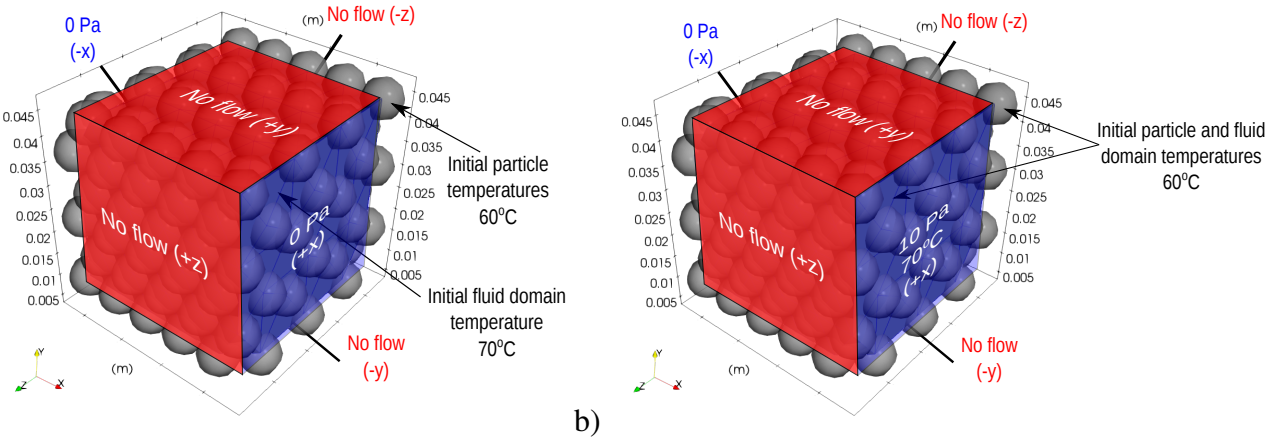

Figure 8. Specimen dimensions and boundary conditions for Yade DEM and ANSYS CFX a)no-flow scenario and b) constant-flow scenario.

$$\int_{S_{lm}^{p}} \Phi_{p,lm} \cdot \mathrm{n}_{lm} \mathrm{d}S = \eta (T_i - T_m) \tag{57}$$

$$\eta = \frac{2r_c^2(k_l + k_m)}{d_{lm}} \tag{58}$$

$$d_{lm} = r_l + r_m - p_d \tag{59}$$

$$r_c = \frac{\sqrt{4d_{lm}^2 r_m^2 - (d_{lm}^2 - r_l^2 + r_m^2)^2}}{2d_{lm}} \tag{60}$$

$$\tag{61}$$

where d is the distance between particles l and m less the overlap, p_d, r is the particle contact radius, k is the particle thermal conductivity, and $T_{l/m}$ are the particle l and m temperatures.

4.2 Fully resolved solution

The fully resolved thermo-hydraulic CFD model was implemented in the commercial software package ANSYS CFX (CFX, 2019) which uses the Finite Volume Method to solve the governing equations presented below. Assuming laminar fluid flow (Re<2300), incompressible fluid, and no buoyancy forces the continuity equation follows:

Table 2. DEM and CFD parameters

Solid Parameter	Value
k W/(m K)	2
C_p J/(kg K)	710
ρ kg/m^3	4976
T_0 °C	60
Fluid parameter	**Value**
k W/(m K)	0.65
C_f J/(kg K)	4184
ρ kg/m^3	1000
T_0 °C	60
viscosity Pa · s	0.001
Reynolds	0-530
Incompressible	
Boundary conditions	
Fluid -Y dirichlet °C	70
Fluid -Y dirichlet Pa	10
Fluid +Y dirichlet Pa	0
Fluid dirichlet ±X±y±z Pa	70
Simulation duration (seconds)	30
Body temp comparison location	(0.25,0.25,0.25)
Flux (kg/s) DEM	0.00394
Flux (kg/s) CFD	0.00472
begin	

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{v}) = S_m \tag{62}$$

where ρ is density, $\mathbf{v}$ is velocity, t is time and S_m is the mass source (e.g. due to vaporization of liquid droplets). The internal mass source was defined as null in the current model. Conservation of momentum in an inertial (non-accelerating) reference frame is described by

$$\frac{\partial}{\partial t}(\rho \mathbf{v}) + \nabla \cdot (\rho \mathbf{v}\mathbf{v}) = -\nabla p + \nabla \cdot (\bar{\bar{\tau}}) + \rho \mathbf{g} + \mathbf{F} \tag{63}$$

where p is the static pressure and $\bar{\bar{\tau}}$ is the stress tensor. The gravitational body force, $\rho\mathbf{g}$ and external body force, $\mathbf{F}$ were both neglected for the present comparison. The stress tensor $\bar{\bar{\tau}}$ is given by:

$$\bar{\bar{\tau}}\mu\left[(\nabla\cdot\mathbf{v}+\nabla\cdot\mathbf{v}^T)-\frac{2}{3}\nabla\cdot\mathbf{v}\mathrm{I}\right] \tag{64}$$

where μ is the molecular viscosity, I is the unit tensor, and the second term on the right-hand side is the effect of volume dilation.

The conservation of energy equation follows a low-speed flow variant:

$$\frac{\partial(\rho e)}{\partial t}+\nabla\cdot(\rho\mathbf{v}e)=\nabla\cdot(\lambda\nabla T)-p\nabla\cdot\mathbf{v}+\nabla\cdot(\bar{\bar{\tau}}\cdot\mathbf{v})+S_h \tag{65}$$

where e is the internal energy and S_h is an internal energy source. Due to very low fluid velocity and small pressure changes, terms $p\nabla\cdot\mathbf{v}$ and $\nabla\cdot(\bar{\bar{\tau}}\cdot\mathbf{v})$ (viscous dissipation) were neglected as well as internal energy source S_h. In consequence, energy change in time depends only on advection and conduction.

The transport equations were augmented with constitutive equations of state for density and enthalpy to form a closed system. It was assumed that fluid is incompressible and there are no buoyancy forces in the fluid. Hence, specific heat (at constant pressure) c_p and density ρ are constant and the incompressible equation of state can be written

$$dH=c_f dT+\frac{dp}{\rho} \tag{66}$$

where h is enthalpy.

The coupling of pressure and velocity follows the high-resolution scheme based on discretization methods presented by Rhie and Chow (1983) and modified by Majumdar (1988).

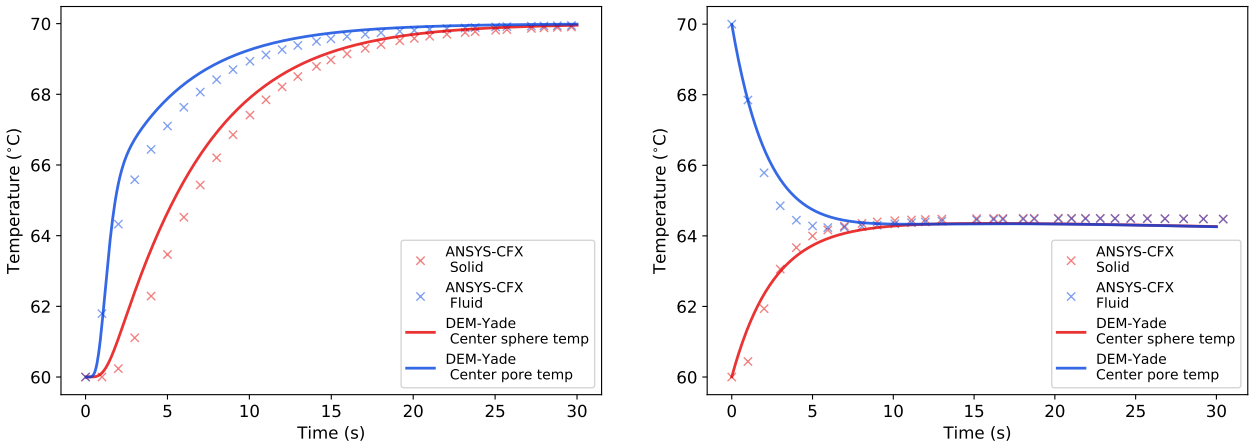

Figure 9. Temperature comparison for ANSYS CFX and Yade DEM in left) flow condition and right) no flow condition for body located at (0.024,0.028,0.026) and pore center located at (0.024,0.023,0.02545)

4.3 Comparison of results

Fig. 9 shows the final comparison of body average temperatures at location (0.25,0.25,0.25) m. For the flow scenario, the THM-DEM model transfers heat more quickly to the center body and center pore which is due to the coarse fluid discretization of pores and particles. Meanwhile, the CFD model captures intricate movements of fluid around particles and complex heat transfer gradients within particles (Fig. 10). Although the THM-DEM model is undoubtedly less accurate during loading (0-15 seconds), both solutions tend toward the same steady state solution. Computational comparison shows that THM-DEM is 100X faster than the ANSYS CFX solution. (Table 3).

Table 3. Performance comparison for THM-DEM and ANSYS CFX

	No-flow (hrs)	Flow (hrs)	Cells
Pore scale THM-DEM (Yade git-dc2ecaec (10-core))	0.04	0.06	1000
Fully resolved FVM (ANSYS CFX v.19.2 (12-core))	3.78	5.38	3.5 million

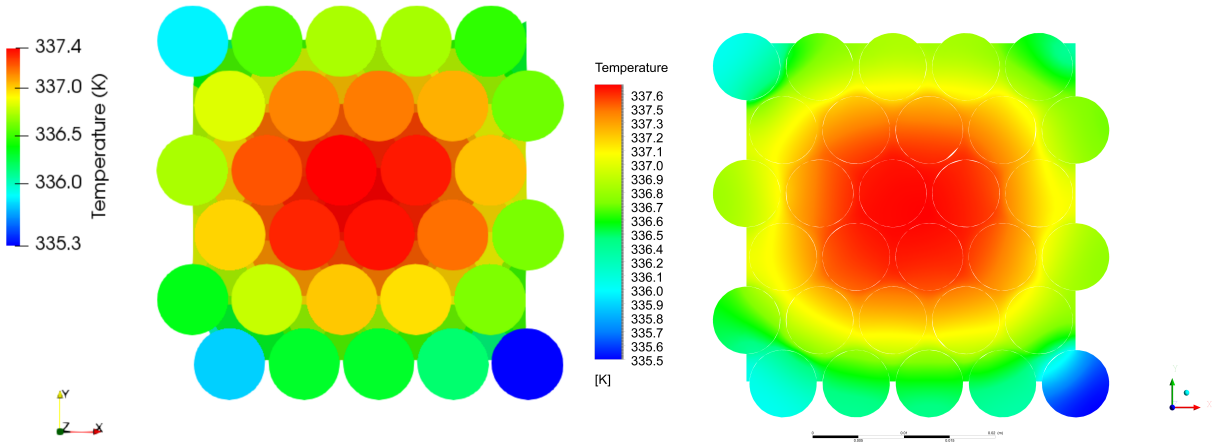

Figure 10. Cross sectional temperature distribution for left) Yade DEM and right) ANSYS CFX at t=30s

5 Experimental and numerical verification of the fully coupled THM-DEM model

Najari and Selvadurai (2014) performed two experiments on a 30 cm tall x 15 cm diameter cylindrical granite specimen containing an inner cylindrical cavity measuring 15 cm tall x 2.4 cm diameter. These experiments were also reproduced numerically by Najari and Selvadurai (2014) using a fully coupled THM finite-element (COMSOL) model. The present study compares these experimental and numerical models to the present DEM-THM model exhibiting the same cylindrical dimensions, boundary conditions, mechanical properties, and thermal properties as Najari and Selvadurai (2014)'s granite specimen (Fig. 12).

5.1 DEM Contact Model

The DEM contact model (Eq. 2, Sec. 2.1) follows a linear elastic model. The normal force, $\mathbf{f}^n$, between two interacting particles i and j is evaluated according to:

$$\mathbf{f}_{ij}^n = k_{ij}^n \Delta D_{ij} \cdot \mathbf{n}_{ij}^n \tag{67}$$

where k_{ij}^n is the normal stiffness, $\mathbf{n}_{ij}^n$ is the unit vector parallel to the branch vector joining the centers of i and j, and $\Delta D_{ij} = D_{ij} - D_{ij}^{eq}$ is the displacement between i and j computed from the equilibrium distance D_{eq} and the actual distance D.

323 k_n is computed assuming two springs are in serial with lengths equal to the interacting particle radii:

$$k_n = \frac{E_a R_a E_b R_b}{E_a R_a + E_b R_b} \tag{68}$$

324 where E is a calibrated particle microparameter referred to as "micro Young's modulus" and R is the radius
325 of particles a and b.

326 Since the shear force, $\mathbf{f}^s$, depends on the orientation of both particles, it is updated incrementally in a
327 local coordinate system according to:

$$\mathbf{f}^s_{ij} = \mathbf{f}^s_{ij,prev} + \Delta\mathbf{f}^s_{ij} \tag{69}$$

328 with

$$\Delta\mathbf{f}^s_{ij} = k^s_{ij}\Delta u^s_{ij} \cdot \mathbf{n}^s_{ij} \tag{70}$$

329 where k^s_{ij} is the shear stiffness, $\mathbf{n}^s_{ij}$ is the unit vector perpendicular to the branch vector, and Δu_s is the
330 incremental tangential displacement.

331 5.2 DEM parameter calibration

332 The micro DEM properties reported in Table 4 were calibrated using typical DEM compression tests and
333 the thermal diffusivity test discussed in Sec. 3. The thermal expansivity of the particles was calibrated by
334 matching the macroscopic specimen thermal expansion to the experimentally observed thermal expansion
335 for a given temperature change. Finally, a density scaling was used to increase the time-step of the
336 explicit motion integration and thereby reduce the computational time to solution (Itasca, 1999; O'Sullivan,
337 2011; Sheng et al., 2003; Thornton and Antony, 2000). In the current quasi-static thermal-hydraulic-
338 mechanical heat transfer simulation, the propagation of elastic waves is the result of temperature changes

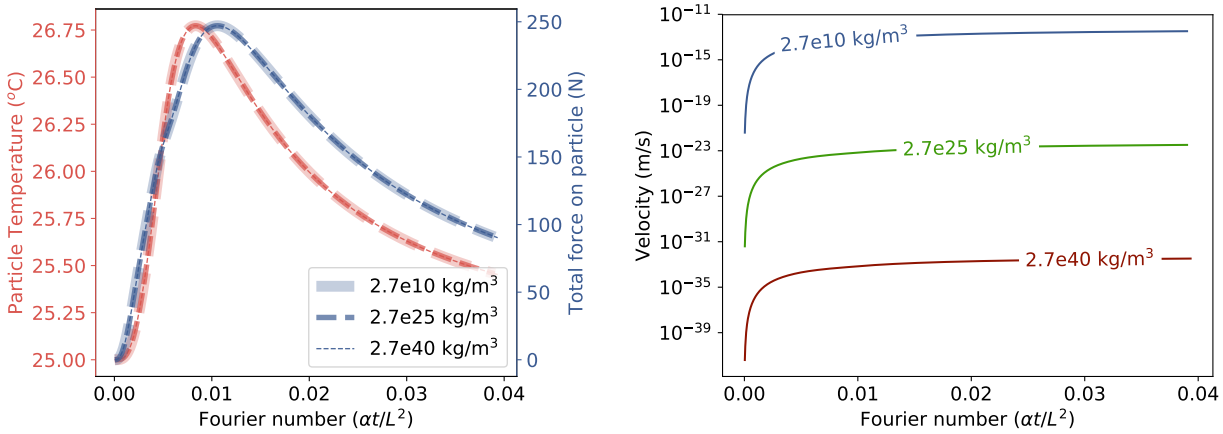

Figure 11. Application of heat flux to particle packing yielding left) temperature and force on particle within packing and right) velocity of particle within packing for various particle densities

only (expansion/contraction of DEM particles). Thus, it is ensured that the density scaling does not affect the development of particle forces. In the context of the present work, a density scaling analysis was performed by applying a ΔT=20°C to one end of the specimen (Fig. 12) for 25 s while monitoring particle temperature, force (magnitude), and velocity. As shown in Fig 11, the density scaling affects the particle velocity, without affecting particle force. Thus, in the quasi-static problem presented here, a density scaling of 10^{40} with a time step of 0.005 s remains stable without affecting the mechanical behavior of the system.

5.3 Hydro-Mechanical verification

The first experiment focused on deriving a permeability estimate based on the HM response of the granite specimen by pumping 3.333e-10 m^3/s of water into the fluid filled cavity. Cavity pressure was monitored during cavity pressurization until steady state was achieved. After one hour at steady state, flow was terminated and depressurization within the cavity was monitored. As shown in Fig. 13, air fractions (ϕ_0) between 2.5e-5 and 3.75e-5 yielded the best match to experimental data. Similar to Najari and Selvadurai (2014)'s FEM-HM model, higher ϕ_0 yields better accuracy during pressurization, and the lower ϕ_0 yields better accuracy during depressurization. Unlike Najari and Selvadurai (2014), the air-fraction magnitude is roughly 2 orders of magnitude lower for the HM-DEM model. In both models, ϕ is a calibration parameter that accounts for more underlying complex physical interactions, so it is expected that the parameter does not match.

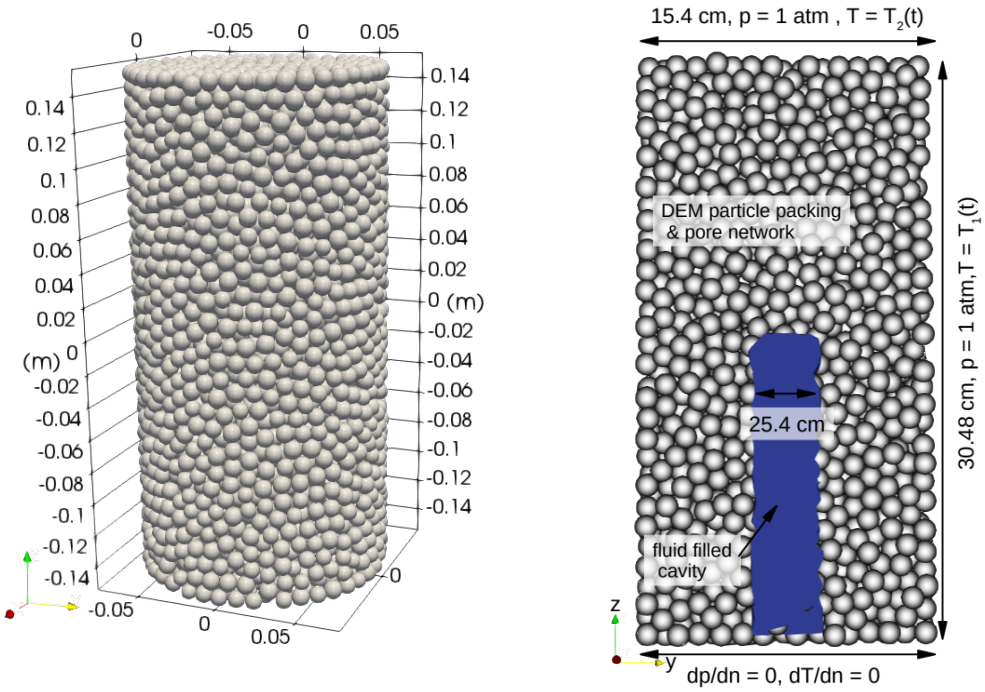

Figure 12. Yade DEM specimen left) Isometric b) split to show fluid filled inner cavity

5.4 Thermo-Hydro-Mechanical verification

The second experiment presented by Najari and Selvadurai (2014) focused on elucidating the THM response of the same granite specimen. During this experiment, the fully saturated granite rock specimen's surface was heated from 25 to 70 °C over the course of one hour (Fig. 14a). The temperature was held at 70 °C for three hours, after which the surface temperature was reduced to 25 °C and held for eight additional hours. During the course of this 12 hour thermal response test, the cavity pressure and temperature were monitored. As shown in Fig. 13, application of the same experimental boundary temperatures to the DEM-THM results in the same experimental and numerical pressurization and depressurization trends reported by Najari and Selvadurai (2014). Specifically, cavity pressure increases with increasing temperature followed by a decrease to 0 kPa once the maximum temperature is reached and steady-state is achieved. As soon as the temperature begins to drop, the cavity pressure decreases below 0 kPa and follows the reverse trend observed during heating. Numerical cavity temperatures also follow closely with experimental and numerical temperatures reported by Najari and Selvadurai (2014), as shown in Fig. 14b.

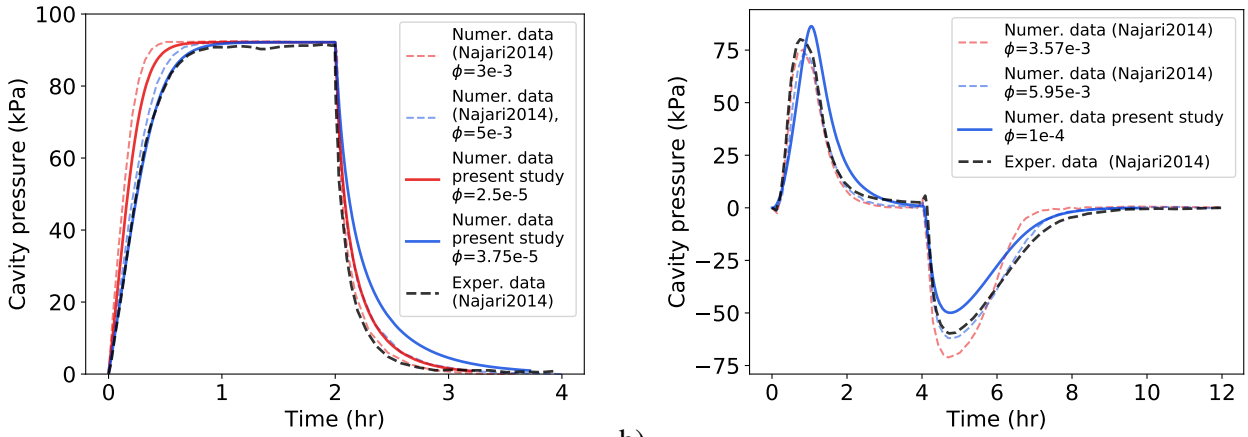

Figure 13. Experimental and numerical cavity pressure curves a) permeability test b) thermal response test

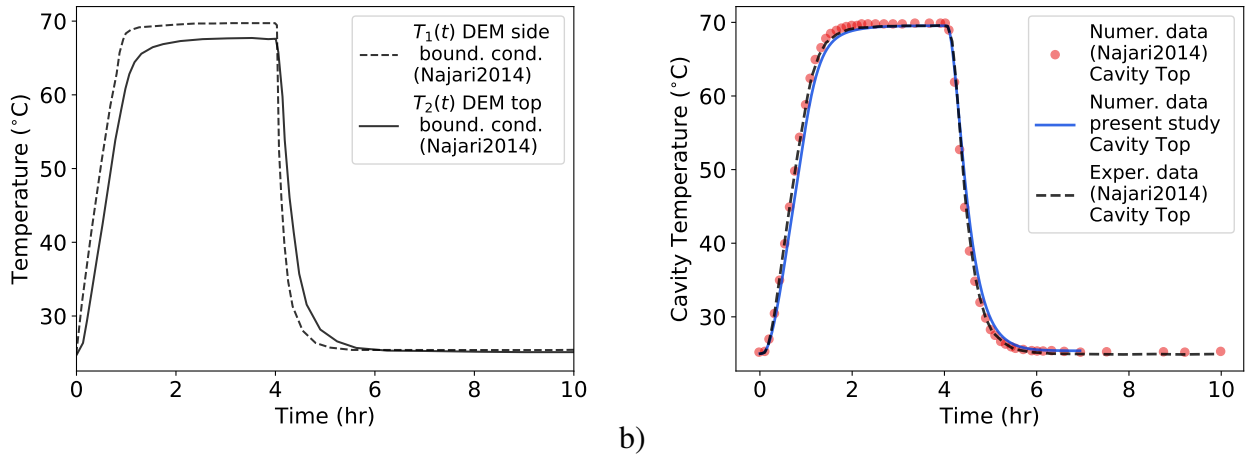

Figure 14. a) Experimentally collected top and circumference specimen temperatures used as variable Dirichlet boundary conditions in FEM-HM Najari and Selvadurai (2014) and the present study DEM-THM. b) Experimental and numerical cavity temperatures during thermal response test.

Table 4. DEM Calibrated microparameters and emergent macroparameters

Parameter	DEM micro	DEM macro	FEM/Experimental
k_{solid} W m^{-1}K^{-1}	30	3.0	3.0
C_p J kg^{-1}K^{-1}	1454	790	790
ρ kg m^{-3}	2750e40	ca. 2750	2750
α m^2 s^{-1}	-	1.43e-6	1.433e-6
$\beta_{particle}$ K^{-1}	1.1e-4	3.0e-5	3.0e-5
β_{fluid} K^{-1}	β(T)	-	β(T)
ϕ_0	2.5e-5	-	3e-3
E GPa	100	60	56
ν	0.2	0.1	0.1
Porosity (%)	46	-	0.6
Permeability m^2	5.25e-18	5.25e-18	5.25e-18
μ Pa·s	μ(T)		μ(T)
Fluid bulk modulus GPa	2.2e9	-	2.2e9
ρ_{solid} kg m^{-3} (ThermalEngine)	2750	-	2750

6 Conclusion

A pore scale model for the simulation of heat transfer and associated thermo-hydro-mechanical couplings in particulate systems has been developed within the open source software Yade DEM. The proposed THM scheme enhances the capabilities of an existing and proven HM scheme by simulating the following heat transfer mechanisms:

- conduction between contacting solid particles
- conduction between pores
- conduction between solid particles and pore fluid
- advection through pore fluid flow

Each mechanism can be simulated independently or collectively, as demonstrated through dedicated verification exercises. In particular, the successful comparison of the present THM-DEM model against a fully resolved CFD model proves that the geometrical considerations and heat flux models for the advection and conduction schemes are physically realistic despite their coarse treatment of the two phase domain. In

addition to the verified conduction and advection schemes presented, thermo-mechanical couplings arise from temperature induced volumetric changes in both solid and fluid phases of the THM-DEM model. The thermo-mechanical effect is exhibited by verifying the full THM-DEM against a THM experiment involving the heating of a saturated rock sample. The proposed model reproduced the key internal fluid pressure trends observed during the THM experiment. First, fluid pressures increase within the rock material during surface temperature heating. Next, the pressure falls as steady state is achieved. Finally, the surface temperature is cooled and the fluid pressure decreases below initial pressures while steady-state is regained. Since the scheme is based on a triangulation of sphere packing poral space, the model applies more specifically to denser particle assemblies, such as those encountered in geomechanics. Thus, it opens up new possibilities for describing and understanding THM processes in porous media from a micromechanical viewpoint such as thermally induced microcracking in geothermal systems or transport process in rock and fractured rock masses.

7 Acknowledgments

The material presented is based upon research supported by the Chateaubriand Fellowship of the Office for Science & Technology of the Embassy of France in the United States. Support was also provided by Veronica Eliasson. The fourth author would like to thank support from Tech21. The third author would like to thank support from the project “Fracture propagation in rocks during hydro-fracking - experiments and discrete element method coupled with fluid flow and heat transport” (years 2019-2022) financed by the National Science Centre (NCN) (UMO-2018/29/B/ST8/00255).

References

Al-Arkawazi, S. (2018). Modeling the heat transfer between fluid-granular medium. *Applied Thermal Engineering*, 128:696 – 705.

André, D., Levraut, B., Tessier-Doyen, N., and Huger, M. (2017). A discrete element thermo-mechanical modelling of diffuse damage induced by thermal expansion mismatch of two-phase materials. *Computer Methods in Applied Mechanics and Engineering*, 318:898 – 916.

Blunt, M. J. (2001). Flow in porous media — pore-network models and multiphase flow. *Current Opinion in Colloid and Interface Science*, 6(3):197 – 207.

Catalano, E., Chareyre, B., and Barthelemy, E. (2014). Pore-scale modeling of fluid-particles interaction and emerging poromechanical effects. *International Journal for Numerical and Analytical Methods in Geomechanics*, 38(1):51–71.

CFX, A. (2019). Ansys CFX Documentation. *ANSYS Inc., USA*.

Chareyre, B., Cortis, A., Catalano, E., and 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, Z., Jin, X., and Wang, M. (2018). A new thermo-mechanical coupled dem model with non-spherical grains for thermally induced damage of rocks. *Journal of the Mechanics and Physics of Solids*, 116:54 – 69.

Cheng, G. J., Yu, A. B., and Zulli, P. (1999). Evaluation of effective thermal conductivity from the structure of a packed bed. *Chemical Engineering Science*, 54(19):4199–4209.

Cundall, P. and Strack, O. (1979). A discrete numerical model for granular assemblies. *Géotechnique*, 29(1):47–65.

Deen, N., Annaland, M. V. S., der Hoef, M. V., and Kuipers, J. (2007). Review of discrete particle modeling of fluidized beds. *Chemical Engineering Science*, 62(1):28 – 44. Fluidized Bed Applications.

Deen, N. G., Kriebitzsch, S. H., van der Hoef, M. A., and Kuipers, J. (2012). Direct numerical simulation of flow and heat transfer in dense fluid–particle systems. *Chemical Engineering Science*, 81:329 – 344.

Feng, Y. T., Han, K., Li, C. F., and Owen, D. R. (2008). Discrete thermal element modelling of heat conduction in particle systems: Basic formulations. *Journal of Computational Physics*, 227(10):5072–5089.

Feng, Y. T., Han, K., and Owen, D. R. (2009). Discrete thermal element modelling of heat conduction in particle systems: Pipe-network model and transient analysis. *Powder Technology*, 193(3):248–256.

Incropera, F. P., DeWitt, D. P., Bergman, T. L., and Lavine, A. S. (2007). *Fundamentals of Heat and Mass Transfer*.

Itasca, C. (1999). *PFC 2D-user manual*.

Joulin, C., Xiang, J., Latham, J. P., Pain, C., and Salinas, P. (2020). Capturing heat transfer for complex-shaped multibody contact problems, a new FDEM approach. *Computational Particle Mechanics*.

Kloss, C., Goniva, C., Hager, A., Amberger, S., and Pirker, S. (2014). Models, algorithms and validation for opensource DEM and CFD-DEM. *Progress in Computational Fluid Dynamics, An International Journal*, 12(2/3):140.

Kolditz, O., Bauer, S., Bilke, L., Böttcher, N., Delfs, J. O., Fischer, T., Görke, U. J., Kalbacher, T., Kosakowski, G., McDermott, C. I., Park, C. H., Radu, F., Rink, K., Shao, H., Shao, H. B., Sun, F., Sun, Y. Y., Singh, A. K., Taron, J., Walther, M., Wang, W., Watanabe, N., Wu, Y., Xie, M., Xu, W., and Zehner, B. (2012). Opengeosys: an open-source initiative for numerical simulation of thermo-hydro-mechanical/chemical (thm/c) processes in porous media. *Environmental Earth Sciences*, 67(2):589–599.

Kruggel-Emden, H., Kravets, B., Suryanarayana, M., and Jasevicius, R. (2016). Direct numerical simulation of coupled fluid flow and heat transfer for single particles and particle packings by a lbm-approach. *Powder Technology*, 294:236 – 251.

Kunii, D. and Smith, J. M. (1960). Heat transfer characteristics of porous rocks. *AIChE Journal*, 6(1):71–78.

Liang, Y. and Li, X. (2014). A new model for heat transfer through the contact network of randomly packed granular material. *Applied Thermal Engineering*, 73(1):982–990.

Lominé, F., Scholtès, L., Sibille, L., and 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.

Majumdar, S. (1988). Role of underrelaxation in momentum interpolation for calculation of flow with nonstaggered grids. *Numerical Heat Transfer*.

457 Najari, M. and Selvadurai, A. P. (2014). Thermo-hydro-mechanical response of granite to temperature
458 changes. *Environmental Earth Sciences*, 72(1):189–198.

459 Norouzi, H. R., Zarghami, R., Sotudeh-Gharebagh, R., and Mostoufi, N. (2016). *Coupled CFD-DEM*
460 *Modeling*. Wiley.

461 Olivella, S., Gens, A., Carrera, J., and Alonso, E. (1996). Numerical formulation for a simulator (code
462 bright) for the coupled analysis of saline media. *Engineering Computations*, 13(7):87–112.

463 O'Sullivan, C. (2011). *Particulate discrete element modelling*.

464 Peng, Y. and Luo, L.-S. (2008). Acomparative study of immersed-boundary and interpolated bounce-back
465 methods in lbe. *Progress in Computational Fluid Dynamics*, 8(1-4):156 – 167.

466 Rhie, C. M. and Chow, W. L. (1983). Numerical study of the turbulent flow past an airfoil with trailing
467 edge separation. *AIAA Journal*.

468 Rutqvist, J., Wu, Y.-S., Tsang, C.-F., and Bodvarsson, G. (2002). A modeling approach for analysis of
469 coupled multiphase fluid flow, heat transfer, and deformation in fractured porous rock. *International*
470 *Journal of Rock Mechanics and Mining Sciences*, 39(4):429 – 442. Numerical Methods in Rock
471 Mechanics.

472 Scholtès, L., Chareyre, B., Michallet, H., Catalano, E., and Marzougui, D. (2015). Modeling wave-induced
473 pore pressure and effective stress in a granular seabed. *Continuum Mechanics and Thermodynamics*,
474 27(1):305–323.

475 Sheng, Y., Lawrence, C. J., Briscoe, B. J., and Thornton, C. (2003). Numerical studies of uniaxial powder
476 compaction process by 3D DEM. *Engineering Computations*, 21(2/3/4):304–317.

477 Shimizu, Y. (2006). Three-dimensional simulation using fixed coarse-grid thermal-fluid scheme and
478 conduction heat transfer scheme in distinct element method. *Powder Technology*, 165(3):140 – 152.

479 Shimizu, Y. (2011). The distinct element analysis for hydraulic fracturing in hard rock considering fluid

viscosity and particle size distribution. *International Journal of Rock Mechanics and Mining Sciences*, 48(5):712 – 727.

Tavassoli, H., Peters, E. A., and Kuipers, J. A. (2015). Direct numerical simulation of fluid-particle heat transfer in fixed random arrays of non-spherical particles. *Chemical Engineering Science*, 129:42–48.

Thornton, C. and Antony, S. J. (2000). Quasi-static shear deformation of a soft particle system. *World*, (109):179–191.

Tomac, I. and Gutierrez, M. (2015). Formulation and implementation of coupled forced heat convection and heat conduction in DEM. *Acta Geotechnica*, 10(4):421–433.

Tsory, T., Ben-Jacob, N., Brosh, T., and Levy, A. (2013). Thermal dem–cfd modeling and simulation of heat transfer through packed bed. *Powder Technology*, 244:52 – 60.

Vargas, W. L. and McCarthy, J. J. (2001). Heat conduction in granular materials. *AIChE Journal*, 47(5):1052–1059.

Wanne, T. S. and Young, R. P. (2008). Bonded-particle modeling of thermally fractured granite. *International Journal of Rock Mechanics and Mining Sciences*, 45(5):789–799.

Yang, B., Chen, S., and Liu, K. (2017). Direct numerical simulations of particle sedimentation with heat transfer using the lattice boltzmann method. *International Journal of Heat and Mass Transfer*, 104:419 – 437.

Zeghal, M. and 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, H., Yuan, H., Trias, F. X., Yu, A., Tan, Y., and Oliva, A. (2016). Particulate immersed boundary method for complex fluid–particle interaction problems with heat transfer. *Computers and Mathematics with Applications*, 71(1):391 – 407.

Zhou, Z. Y., Yu, A. B., and Zulli, P. (2009). Particle scale study of heat transfer in packed and bubbling fluidized beds. *AIChE Journal*, 55(4):868–884.

Šmilauer V. et al. (2015). Yade documentation 2nd ed. the yade project. *Transport in Porous Media*.

A Source code and installation

Source code is freely available as part of Yade DEM on gitlab.com. Installation on a Ubuntu linux (`https://yade-dem.org/doc/installation.html`) requires a single command:

```
sudo apt-get install yade
```

However, the source code for Yade DEM and the included thermal components presented here are available online for review and modification:

- Full source code:`https://gitlab.com/yade-dev/trunk`
- Thermal component:`https://gitlab.com/yade-dev/trunk/-/blob/master/pkg/pfv/Thermal.cpp`

Readers are encouraged to contact the Yade community at `https://answers.launchpad.net/yade` with questions regarding installation, usage, modifications, and theory.

B Setting up a Thermo-Hydro-Mechanical simulation in Yade

Find the full list of available parameters and their full descriptions at `https://yade-dem.org/doc/yade.wrapper.html#yade.wrapper.ThermalEngine`. Otherwise, a standard ThermalEngine simulation in Yade should be setup as follows:

```
# add FlowEngine and ThermalEngine to typical engine list:
O.engines=[
... ,
FlowEngine(label="flow"),
ThermalEngine(label="thermal"),
VTKRecorder(recorders=[..., 'thermal', ... ]), # save thermal
 ↪  quantities for particles
...
]

# Set flow parameters of the simulation:
pZero = 1.01325e5
flow.pZero = pZero
flow.fluidBulkModulus=2.2e9
flow.meshUpdateInterval=200
flow.useSolver=4
flow.permeabilityFactor= -1.5e-17
flow.viscosity=0.001
```

```
flow.bndCondIsPressure=[1,1,0,0,0,0]  # pressure gradient on X axis
flow.bndCondValue=[10,0,0,0,0,0]

# Set thermal flow parameters
flow.tempDependentViscosity=True
flow.thermalEngine=True
flow.fluidRho = 1000.
flow.fluidCp = 4184.
flow.bndCondIsTemperature=[0,0,1,1,0,0] # temperature gradient on Y
 ↪  axis (advection)
flow.thermalBndCondValue=[0,0,70,20,0,0]
flow.phiZero=1e-4 # air fraction in cavities
flow.cavityFluidDensity = 1000.
flow.tZero=20  # initial fluid temperatures

# Set thermal engine parameters
thermal.fluidConduction=True
thermal.conduction=True
thermal.thermoMech=True
thermal.solidThermoMech = True
thermal.fluidThermoMech = True
thermal.advection=True
thermal.bndCondIsTemperature=[0,0,0,0,1,1] # temp gradient on Z axis
 ↪  (conduction)Script
thermal.thermalBndCondValue=[0,0,0,0,20,70]
thermal.fluidK = 0.58
thermal.unboundCavityBodies=True  # enable cavity model
thermal.particleT0 = 20
thermal.particleK = 30.
thermal.particleCp = heatCap
thermal.particleAlpha = 3.0e-6 # solid expansion coeff
thermal.particleDensity = density
thermal.tsSafetyFactor = 0
thermal.uniformReynolds =10  # set a uniform reynolds number (only
 ↪  for very low vel.)
thermal.porosityFactor = 0.006/utils.porosity()
thermal.tempDependentFluidBeta = True  # fluid expansion coeff

# impose cavity at desired locations:
flow.imposeCavity((x,y,z))

# check temperature at desired locations:
flow.getPoreTemperature((x,y,z))

# visualize fluid thermal quantities (temp, RE, etc.)
flow.saveVtk('VTK/')
```

C Scripts

Conduction example script:

```
#**************************************************************************
#  Copyright (C) 2019 by Robert Caulk                                     *
#  rob.caulk@gmail.com                                                    *
#                                                                         *
#  This program is free software; it is licensed under the terms of the  *
#  GNU General Public License v2 or later. See file LICENSE for details. *
#**************************************************************************/
#
# Script demonstrating the use of ThermalEngine by comparing conduction
# scheme to analytical solution to Fourier (rod cooling with constant
# boundary conditions). See details in:
#
# Caulk, R., Scholtes, L., Kraczek, M., Chareyre, B.  A
# pore-scale Thermo-Hydro-Mechanical coupled model for particulate systems.
#

from yade import pack
from yade import timing
import numpy as np
import shutil
timeStr = time.strftime('%m-%d-%Y')
num_spheres=1000
young=1e6
rad=0.003

mn,mx=Vector3(0,0,0),Vector3(1.0,0.008,0.008) # corners of the initial packing

thermalCond = 2. #W/(mK)
heatCap = 710. #J(kg K)
t0 = 400. #K

r = rad
k = 2*2.0*r  # 2*k*r
Cp = 710.
```

```
rho = 2600.
D = 2.*r
m = 4./3.*np.pi*r**2/rho

# macro diffusivity
thermalDiff = 6.*k/(D*np.pi*Cp*rho)

identifier = '-conductionVerification'

if not os.path.exists('VTK'+timeStr+identifier):
        os.mkdir('VTK'+timeStr+identifier)
else:
        shutil.rmtree('VTK'+timeStr+identifier)
        os.mkdir('VTK'+timeStr+identifier)

if not os.path.exists('txt'+timeStr+identifier):
        os.mkdir('txt'+timeStr+identifier)
else:
        shutil.rmtree('txt'+timeStr+identifier)
        os.mkdir('txt'+timeStr+identifier)

shutil.copyfile(sys.argv[0],'txt'+timeStr+identifier+'/'+sys.argv[0])

O.materials.append(FrictMat(young=young,poisson=0.5,frictionAngle=radians(3),c
O.materials.append(FrictMat(young=young,poisson=0.5,frictionAngle=0,density=0,
walls=aabbWalls([mn,mx],thickness=0,material='walls')
wallIds=O.bodies.append(walls)

O.bodies.append(pack.regularOrtho(pack.inAlignedBox(mn,mx),radius=rad,gap=-1e-

print('num bodies ', len(O.bodies))

ThermalEngine = ThermalEngine(dead=1,label='thermal');

newton=NewtonIntegrator(damping=0.2)
intRadius = 1
O.engines=[
        ForceResetter(),
        InsertionSortCollider([Bo1_Sphere_Aabb(aabbEnlargeFactor=intRadius),Bo
        InteractionLoop(
                [Ig2_Sphere_Sphere_ScGeom(interactionDetectionFactor=intRadius
                [Ip2_FrictMat_FrictMat_FrictPhys()],
                [Law2_ScGeom_FrictPhys_CundallStrack()],label="iloop"
        ),
        FlowEngine(dead=1,label="flow",multithread=False),
        ThermalEngine,        GlobalStiffnessTimeStepper(active=1,timeStepUpda
```

```
        #triax,
        VTKRecorder(iterPeriod=500,fileName='VTK'+timeStr+identifier+'/spheres
        newton
]

for b in O.bodies:
        if isinstance(b.shape, Sphere):
                b.dynamic=False

# we only need flow engine to detect boundaries, there is no flow
 ↪  computed for this
flow.dead=0
flow.bndCondIsPressure=[0,0,0,0,0,0]
flow.bndCondValue=[0,0,0,0,0,0]
flow.boundaryUseMaxMin=[0,0,0,0,0,0]
flow.thermalEngine=True
flow.bndCondIsTemperature=[1,1,0,0,0,0]
flow.thermalEngine=True
flow.thermalBndCondValue=[0,0,0,0,0,0]

flow.tZero=t0
flow.pZero=0
thermal.dead=0
thermal.conduction=True
thermal.thermoMech=False
thermal.advection=False
thermal.fluidThermoMech = False
thermal.solidThermoMech = False
thermal.fluidConduction= False

thermal.bndCondIsTemperature=[1,1,0,0,0,0]
thermal.thermalBndCondValue=[0,0,0,0,0,0]
thermal.tsSafetyFactor=0
thermal.particleDensity=2600
thermal.particleT0=t0
thermal.particleCp=heatCap
thermal.particleK=thermalCond
thermal.particleAlpha =11.6e-3
thermal.useKernMethod=False

timing.reset()

flow.updateTriangulation=True
O.dt=1.
O.dynDt=False
```

```
O.run(1,1)
flow.dead=1

def bodyByPos(x,y,z):
        cBody = O.bodies[1]
        cDist = Vector3(100,100,100)
        for b in O.bodies:
                if isinstance(b.shape, Sphere):
                        dist = b.state.pos - Vector3(x,y,z)
                        if np.linalg.norm(dist) <
                         ↪  np.linalg.norm(cDist):
                                cDist = dist
                                cBody = b
        print('found closest body ', cBody.id, ' at ',
         ↪  cBody.state.pos)
        return cBody

# solution to the heat equation for constant initial condition ,
 ↪  BCs=0, and using series for approx
def analyticalHeatSolution(x,t,u0,L,k):
        ns = np.linspace(1,1000,1000)
        solution = 0
        for i,n in enumerate(ns):
                integral = (-2./L)*u0*L*(np.cos(n*np.pi)-1.) /
                 ↪  (n*np.pi)
                solution += integral *
                 ↪  np.sin(n*np.pi*x/L)*np.exp((-k*(n*np.pi/L)**2)*t)
        return solution

# find 10 bodies along x axis
axis = np.linspace(mn[0], mx[0], num=11)
axisBodies = [None] * len(axis)
axisTrue = np.zeros(len(axis))
for i,x in enumerate(axis):
        axisBodies[i] = bodyByPos(x, mx[1]/2, mx[2]/2)
        axisTrue[i] = axisBodies[i].state.pos[0]
np.savetxt('txt'+timeStr+identifier+'/xdata.txt',axisTrue)
print("Axis length used for analy ", max(axisTrue)-min(axisTrue))
from yade import plot

## a function saving variables
def history():
        plot.addData(
                t=O.time,
                i = O.iter,
                temp1 = axisBodies[0].state.temp,
```

```
                temp2 = axisBodies[1].state.temp,
                temp3 = axisBodies[2].state.temp,
                temp4 = axisBodies[3].state.temp,
                temp5 = axisBodies[4].state.temp,
                temp6 = axisBodies[5].state.temp,
                temp7 = axisBodies[6].state.temp,
                temp8 = axisBodies[7].state.temp,
                temp9 = axisBodies[8].state.temp,
                temp10 = axisBodies[9].state.temp,
                temp11 = axisBodies[10].state.temp,
                AnalyTemp1 =
                 ↪ analyticalHeatSolution(0,O.time,t0,mx[0],thermalDiff),
                AnalyTemp2 =
                 ↪ analyticalHeatSolution(axisBodies[1].state.pos[0],O.time,t
                AnalyTemp3 =
                 ↪ analyticalHeatSolution(axisBodies[2].state.pos[0]-min(axis
                AnalyTemp4 =
                 ↪ analyticalHeatSolution(axisBodies[3].state.pos[0]-min(axis
                AnalyTemp5 =
                 ↪ analyticalHeatSolution(mx[0],O.time,t0,mx[0],thermalDiff)
)
        plot.saveDataTxt('txt'+timeStr+identifier+'/conductionAnalyticalCompar

O.engines=O.engines+[PyRunner(iterPeriod=500,command='history()',label='recor

##make nice animations:
VTKrec.dead=0
from yade import plot

plot.plots={'t':(('temp4','k-'),('temp3','r-'),('AnalyTemp4','k--'),('AnalyTem
 ↪ #

plot.plot()
O.saveTmp()
O.timingEnabled=1
from yade import timing
print("starting oedometer simulation")
O.run(200,1)
timing.stats()
```

No flow example script:

```
#**************************************************************************
#  Copyright (C) 2019 by Robert Caulk
 ↪  *
#  rob.caulk@gmail.com
 ↪  *
#
 ↪  *
#  This program is free software; it is licensed under the terms of
 ↪  the  *
#  GNU General Public License v2 or later. See file LICENSE for
 ↪  details. *
#**************************************************************************/
#
# Script demonstrating the use of ThermalEngine by monitoring still
 ↪  fluid
# temperature changes in a sphere packing.
# Also serves as a validation script for comparison
# with ANSYS CFD. See details in
# Caulk, R., Scholtes, L., Kraczek, M., Chareyre, B. A
# pore-scale Thermo-Hydro-Mechanical coupled model for particulate
 ↪  systems.
#
# note: warnings for inifiniteK and Reynolds numbers = nan for
 ↪  boundary
# cells in regular packings are expected. It does not interfere with
 ↪  the
# physics.

from yade import pack, ymport
from yade import timing
import numpy as np
import shutil
timeStr = time.strftime('%m-%d-%Y')
num_spheres=1000# number of spheres
young=1e9
rad=0.003

mn,mx=Vector3(0,0,0),Vector3(0.05,0.05,0.05) # corners of the initial
 ↪  packing

thermalCond = 2. #W/(mK)
heatCap = 710. #J(kg K)
t0 = 333.15 #K
```

```
# micro properties
r = rad
k = 2.0
Cp = 710.
rho = 2600.
D = 2.*r
m = 4./3.*np.pi*r**2/rho

identifier = '-noFlowScenario'

if not os.path.exists('VTK'+timeStr+identifier):
        os.mkdir('VTK'+timeStr+identifier)
else:
        shutil.rmtree('VTK'+timeStr+identifier)
        os.mkdir('VTK'+timeStr+identifier)

if not os.path.exists('txt'+timeStr+identifier):
        os.mkdir('txt'+timeStr+identifier)
else:
        shutil.rmtree('txt'+timeStr+identifier)
        os.mkdir('txt'+timeStr+identifier)

shutil.copyfile(sys.argv[0],'txt'+timeStr+identifier+'/'+sys.argv[0])

O.materials.append(FrictMat(young=young,poisson=0.5,frictionAngle=radians(3),c
O.materials.append(FrictMat(young=young,poisson=0.5,frictionAngle=0,density=0,
walls=aabbWalls([mn,mx],thickness=0,material='walls')
wallIds=O.bodies.append(walls)

sp = O.bodies.append(ymport.textExt('5cmEdge_1mm.spheres',
 ↪  'x_y_z_r',color=(0.1,0.1,0.9), material='spheres'))

print('num bodies ', len(O.bodies))

triax=TriaxialStressController(
        maxMultiplier=1.+2e4/young,
        finalMaxMultiplier=1.+2e3/young,
        thickness = 0,
        stressMask = 7,
        internalCompaction=True,
)

ThermalEngine = ThermalEngine(dead=1,label='thermal');

newton=NewtonIntegrator(damping=0.2)
intRadius = 1
```

```
O.engines=[
        ForceResetter(),
        InsertionSortCollider([Bo1_Sphere_Aabb(aabbEnlargeFactor=intRadius),Bo
        InteractionLoop(
                [Ig2_Sphere_Sphere_ScGeom(interactionDetectionFactor=intRadius
                [Ip2_FrictMat_FrictMat_FrictPhys()],
                [Law2_ScGeom_FrictPhys_CundallStrack()],label="iloop"
        ),
        FlowEngine(dead=1,label="flow",multithread=False),
        ThermalEngine,        GlobalStiffnessTimeStepper(active=1,timeStepUpda
        triax,
        VTKRecorder(iterPeriod=2000,fileName='VTK'+timeStr+identifier+'/sphere
        newton
]

for b in O.bodies:
        if isinstance(b.shape, Sphere):
                b.dynamic=False # mechanically static

flow.dead=0
flow.defTolerance=-1
flow.meshUpdateInterval=-1
flow.useSolver=4
flow.permeabilityFactor= 1
flow.viscosity= 0.001
flow.bndCondIsPressure=[0,0,0,0,0,0]
flow.bndCondValue=[0,0,0,0,0,0]
flow.thermalEngine=True
flow.debug=False
flow.fluidRho = 997
flow.fluidCp = 4181.7
flow.bndCondIsTemperature=[0,0,0,0,0,0]
flow.thermalEngine=True
flow.thermalBndCondValue=[0,0,0,0,0,0]
flow.tZero=343.15
flow.pZero=0

thermal.dead=0
thermal.debug=False
thermal.fluidConduction=True
thermal.ignoreFictiousConduction=True
thermal.conduction=True
thermal.thermoMech=False
thermal.solidThermoMech = False
thermal.fluidThermoMech = False
thermal.advection=True
```

```
thermal.bndCondIsTemperature=[0,0,0,0,0,0]
thermal.thermalBndCondValue=[0,0,0,0,0,0]
thermal.fluidK = 0.6069
thermal.fluidConductionAreaFactor=1.
thermal.uniformReynolds=10
thermal.particleT0 = 333.15
thermal.particleDensity=2600.
thermal.particleK = 2.
thermal.particleCp = 710.
thermal.tsSafetyFactor=0
thermal.useKernMethod=True
thermal.useHertzMethod=False
timing.reset()

O.dt=0.1e-4
O.dynDt=False

O.run(1,1)
flow.dead=0

#triax.goal2=-11000

def bodyByPos(x,y,z):
        cBody = O.bodies[1]
        cDist = Vector3(100,100,100)
        for b in O.bodies:
                if isinstance(b.shape, Sphere):
                        dist = b.state.pos - Vector3(x,y,z)
                        if np.linalg.norm(dist) <
                         ↪  np.linalg.norm(cDist):
                                cDist = dist
                                cBody = b
        return cBody

bodyOfInterest = bodyByPos(0.025,0.025,0.025)

# find 10 bodies along x axis
axis = np.linspace(mn[0], mx[0], num=5)
axisBodies = [None] * len(axis)
axisTrue = np.zeros(len(axis))
for i,x in enumerate(axis):
        axisBodies[i] = bodyByPos(x, mx[1]/2, mx[2]/2)
        axisTrue[i] = axisBodies[i].state.pos[0]

from yade import plot
```

```
def history():
        plot.addData(
                ftemp1=flow.getPoreTemperature((0.025,0.025,0.025)),
                p=flow.getPorePressure((0.025,0.025,0.025)),
                t=O.time,
                i = O.iter,
                temp1 = axisBodies[0].state.temp,
                temp2 = axisBodies[1].state.temp,
                temp3 = axisBodies[2].state.temp,
                temp4 = axisBodies[3].state.temp,
                temp5 = axisBodies[4].state.temp,
                bodyOfIntTemp =
                 ↪  O.bodies[bodyOfInterest.id].state.temp
)
        plot.saveDataTxt('txt'+timeStr+identifier+'/temps'+identifier+'.txt',v
         ↪  'temp1','temp2','temp3','bodyOfIntTemp'))

O.engines=O.engines+[PyRunner(iterPeriod=500,command='history()',label='recor

def pressureField():
        flow.saveVtk('VTK'+timeStr+identifier+'/',withBoundaries=False)
O.engines=O.engines+[PyRunner(iterPeriod=2000,command='pressureField()')]

def endFlux():
        if O.time >= 30:
                O.pause()
O.engines=O.engines+[PyRunner(iterPeriod=10,command='endFlux()')]
VTKrec.dead=0
from yade import plot

plot.plots={'t':(('ftemp1','k-'),('bodyOfIntTemp','r-'))} #
plot.plot()
O.saveTmp()

O.run()
```

Flow example script:

```
#**************************************************************************
#  Copyright (C) 2019 by Robert Caulk
 ↪  *
#  rob.caulk@gmail.com
 ↪  *
#
 ↪  *
#  This program is free software; it is licensed under the terms of
 ↪  the  *
#  GNU General Public License v2 or later. See file LICENSE for
 ↪  details. *
#**************************************************************************/
#
# Script demonstrating the use of ThermalEngine by permeating warm
 ↪  fluid
# through a cold packing. Also serves as a validation script for
 ↪  comparison
# with ANSYS CFD. See details in
# Caulk, R., Scholtes, L., Kraczek, M., Chareyre, B. A
# pore-scale Thermo-Hydro-Mechanical coupled model for particulate
 ↪  systems.
#
# note: warnings for inifiniteK and Reynolds numbers = nan for
 ↪  boundary
# cells in regular packings are expected. It does not interfere with
 ↪  the
# physics


from yade import pack, ymport
from yade import timing
import numpy as np
import shutil
timeStr = time.strftime('%m-%d-%Y')
num_spheres=1000# number of spheres
young=1e9
rad=0.003

mn,mx=Vector3(0,0,0),Vector3(0.05,0.05,0.05) # corners of the initial
 ↪  packing

thermalCond = 2. #W/(mK)
heatCap = 710. #J(kg K)
t0 = 333.15 #K
```

```
# micro properties
r = rad
k = 2.0
Cp = 710.
rho = 2600.
D = 2.*r
m = 4./3.*np.pi*r**2/rho

identifier = '-flowScenario'

if not os.path.exists('VTK'+timeStr+identifier):
        os.mkdir('VTK'+timeStr+identifier)
else:
        shutil.rmtree('VTK'+timeStr+identifier)
        os.mkdir('VTK'+timeStr+identifier)

if not os.path.exists('txt'+timeStr+identifier):
        os.mkdir('txt'+timeStr+identifier)
else:
        shutil.rmtree('txt'+timeStr+identifier)
        os.mkdir('txt'+timeStr+identifier)

O.materials.append(FrictMat(young=young,poisson=0.5,frictionAngle=radians(3),c
O.materials.append(FrictMat(young=young,poisson=0.5,frictionAngle=0,density=0,
walls=aabbWalls([mn,mx],thickness=0,material='walls')
wallIds=O.bodies.append(walls)

sp = O.bodies.append(ymport.textExt('5cmEdge_1mm.spheres',
 ↪  'x_y_z_r',color=(0.1,0.1,0.9), material='spheres'))

print('num bodies ', len(O.bodies))

triax=TriaxialStressController(
        maxMultiplier=1.+2e4/young,
        finalMaxMultiplier=1.+2e3/young,
        thickness = 0,
        stressMask = 7,
        internalCompaction=True,
)

ThermalEngine = ThermalEngine(dead=1,label='thermal');

newton=NewtonIntegrator(damping=0.2)
intRadius = 1
O.engines=[
```

```
        ForceResetter(),
        InsertionSortCollider([Bo1_Sphere_Aabb(aabbEnlargeFactor=intRadius),Bo
        InteractionLoop(
                [Ig2_Sphere_Sphere_ScGeom(interactionDetectionFactor=intRadius
                [Ip2_FrictMat_FrictMat_FrictPhys()],
                [Law2_ScGeom_FrictPhys_CundallStrack()],label="iloop"
        ),
        FlowEngine(dead=1,label="flow",multithread=False),
        ThermalEngine,        GlobalStiffnessTimeStepper(active=1,timeStepUpda
        triax,
        VTKRecorder(iterPeriod=500,fileName='VTK'+timeStr+identifier+'/spheres
        newton
]

for b in O.bodies:
        if isinstance(b.shape, Sphere):
                b.dynamic=False # mechanically static

flow.dead=0
flow.defTolerance=-1
flow.meshUpdateInterval=-1
flow.useSolver=4
flow.permeabilityFactor= 1
flow.viscosity= 0.001
flow.bndCondIsPressure=[1,1,0,0,0,0]
flow.bndCondValue=[10,0,0,0,0,0]
flow.thermalEngine=True
flow.fluidRho = 997
flow.fluidCp = 4181.7
flow.bndCondIsTemperature=[1,0,0,0,0,0]
flow.thermalEngine=True
flow.thermalBndCondValue=[343.15,0,0,0,0,0]
flow.tZero=t0
flow.pZero=0
flow.maxKdivKmean=1
flow.minKdivmean=0.0001;

thermal.dead=0
thermal.fluidConduction=True
thermal.ignoreFictiousConduction=True
thermal.conduction=True
thermal.thermoMech=False
thermal.solidThermoMech = False
thermal.fluidThermoMech = False
thermal.advection=True
thermal.bndCondIsTemperature=[0,0,0,0,0,0]
```

```
thermal.thermalBndCondValue=[0,0,0,0,0,0]
thermal.fluidK = 0.6069
thermal.fluidConductionAreaFactor=1.
thermal.particleT0 = t0
thermal.particleDensity=2600.
thermal.particleK = thermalCond
thermal.particleCp = heatCap
thermal.useKernMethod=True

timing.reset()

O.dt=0.1e-3
O.dynDt=False

O.run(1,1)
flow.dead=0

def bodyByPos(x,y,z):
        cBody = O.bodies[1]
        cDist = Vector3(100,100,100)
        for b in O.bodies:
                if isinstance(b.shape, Sphere):
                        dist = b.state.pos - Vector3(x,y,z)
                        if np.linalg.norm(dist) <
                         ↪  np.linalg.norm(cDist):
                                cDist = dist
                                cBody = b
        print('found closest body ', cBody.id, ' at ',
         ↪  cBody.state.pos)
        return cBody

bodyOfInterest = bodyByPos(0.025,0.025,0.025)

# find 10 bodies along x axis
axis = np.linspace(mn[0], mx[0], num=5)
axisBodies = [None] * len(axis)
axisTrue = np.zeros(len(axis))
for i,x in enumerate(axis):
        axisBodies[i] = bodyByPos(x, mx[1]/2, mx[2]/2)
        axisTrue[i] = axisBodies[i].state.pos[0]

print("found body of interest at", bodyOfInterest.state.pos)

from yade import plot

## a function saving variables
```

```
def history():
        plot.addData(
                ftemp1=flow.getPoreTemperature((0.024,0.023,0.02545)),
                p=flow.getPorePressure((0.025,0.025,0.025)),
                t=O.time,
                i = O.iter,
                temp1 = axisBodies[0].state.temp,
                temp2 = axisBodies[1].state.temp,
                temp3 = axisBodies[2].state.temp,
                temp4 = axisBodies[3].state.temp,
                temp5 = axisBodies[4].state.temp,
                bodyOfIntTemp =
                 ↪  O.bodies[bodyOfInterest.id].state.temp
)
        plot.saveDataTxt('txt'+timeStr+identifier+'/temps'+identifier+'.txt',v
         ↪  'temp1','temp2','temp3','bodyOfIntTemp'))

O.engines=O.engines+[PyRunner(iterPeriod=500,command='history()',label='recor

def pressureField():
        flow.saveVtk('VTK'+timeStr+identifier+'/',withBoundaries=False)
O.engines=O.engines+[PyRunner(iterPeriod=2000,command='pressureField()')]

def endFlux():
        if O.time >= 30:
                flux = 0
                n=utils.porosity()
                for i in flow.getBoundaryVel(1):
                        flux +=i[0]*i[3]/n  # area * velocity /
                         ↪  porosity  (dividing by porosity because
                         ↪  flow engine is computing the darcy
                         ↪  velocity)
                massFlux = flux * 997

                K =
                 ↪  abs(flow.getBoundaryFlux(1))*(flow.viscosity*0.5)/(0.5**2.
                d=8e-3 # sphere diameter
                Kc = d**2/180. * (n**3.)/(1.-n)**2

                print('Permeability', K, 'kozeny', Kc)
                print('outlet flux(with vels
                 ↪  only):',massFlux,'compared to CFD = 0.004724
                 ↪  kg/s')
                print('sim paused')
                O.pause()
```

```
O.engines=O.engines+[PyRunner(iterPeriod=10,command='endFlux()')]
VTKrec.dead=0
from yade import plot

plot.plots={'t':(('ftemp1','k-'),('bodyOfIntTemp','r-'))} #
plot.plot()
O.saveTmp()

print("starting thermal sim")
O.run()
```