POLITECNICO DI TORINO
Repository ISTITUZIONALE

A parallel solver for large scale DFN flow simulations

Original
A parallel solver for large scale DFN flow simulations / Berrone, S., Pieraccini, S., Scialo', S., Vicini, F.. - In: SIAM JOURNAL ON SCIENTIFIC COMPUTING. - ISSN 1064-8275. - STAMPA. - 37:3(2015), pp. C285-C306. [10.1137/140984014]

Availability:
This version is available at: 11583/2561152 since: 2016-09-07T11:43:29Z

Publisher:
SIAM - Society for Industrial and Applied Mathematics

Published
DOI:10.1137/140984014

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright

(Article begins on next page)

14 September 2026

SIAM J. SCI. COMPUT.
Vol. 37, No. 3, pp. C285–C306
© 2015 Society for Industrial and Applied Mathematics

A PARALLEL SOLVER FOR LARGE SCALE DFN FLOW SIMULATIONS*

STEFANO BERRONE†, SANDRA PIERACCINI†, STEFANO SCIALÒ†, AND FABIO VICINI†

Abstract. Flows in fractured media have been modeled using many different approaches in order to get reliable and efficient simulations for many critical applications. The common issues to be tackled are the wide range of scales involved in the phenomenon, the complexity of the domain, and the huge computational cost. In the present paper we propose a parallel implementation of the PDE-constrained optimization method presented in [S. Berrone, S. Pieraccini, and S. Scialò, *SIAM J. Sci. Comput.,* 35 (2013), pp. B487–B510; S. Berrone, S. Pieraccini, and S. Scialò, *SIAM J. Sci. Comput.,* 35 (2013), pp. A908–A935; S. Berrone, S. Pieraccini, and S. Scialò, *J. Comput. Phys.,* 256 (2014), pp. 838–853] for dealing with arbitrary discrete fracture networks (DFNs) on nonconforming grids. We show the scalability performances and the efficiency of the parallel algorithm, and we also test the robustness of the method on complex and strongly connected DFN configurations which would be very difficult to mesh using conventional approaches relying on some kind of mesh conformity at the interfaces.

Key words. message passing interface (MPI), parallel scalability, discrete fracture networks, single-phase flows, PDE-constrained optimization

AMS subject classifications. 65N30, 65N15, 65N50, 65J15, 68U20, 68W10, 68W40, 86-08

DOI. 10.1137/140984014

1. Introduction.

The simulation of flows in fractured media is a challenging issue relevant in several critical applications: Oil and gas enhanced production, nuclear waste geological storage, carbon dioxide geological storage, geothermal applications, and energy and gas storage. The flow mainly takes place in the fractures, and the contribution of the surrounding rock matrix can have, in many cases, a marginal impact on the flow pattern. Intensity and directionality of flow largely depend on the distribution of fractures and on their hydraulic properties.

Several models have been proposed for the simulation of flows in fractured media. Here we consider the discrete fracture network (DFN) model [8, 15, 7], with polygonal planar fractures representing the real fractures of the medium. Since the actual distribution of the fractures in a large scale geological basin cannot be precisely determined in a deterministic way, DFNs are built as representations of natural media starting from stochastic distributions derived from "in situ" measurements [6, 8, 12, 15, 21, 41]. DFN flow simulations are also a starting point for investigating the hydraulic behavior and hydraulic and mechanical interactions [10, 11, 39, 9, 33].

Concerning numerical simulations on DFNs, several discretization approaches, such as the standard finite element method (FEM), mixed hybrid discretizations, and extended finite elements [34, 35, 40, 19], are usually applied. One of the major

*Submitted to the journal's Software and High-Performance Computing section August 27, 2014; accepted for publication (in revised form) March 9, 2015; published electronically May 7, 2015. This work was supported by Italian MIUR through PRIN fund 2012HBLYE4_001, by the CINECA-ISCRA grant "Parallel Implementation of an Optimization Approach to Discrete Fracture Network Simulations," IsC20_ParDFN, HP10CFKT1E, and by the INdAM-GNCS project "Tecniche numeriche per la simulazione di flussi in reti di fratture di grandi dimensioni" (2015).

http://www.siam.org/journals/sisc/37-3/98401.html

†Dipartimento di Scienze Matematiche, Politecnico di Torino, 10129 Torino, Italy (stefano.berrone@polito.it, sandra.pieraccini@polito.it, stefano.scialo@polito.it, fabio.vicini.90@gmail.com).

complexities related to standard discretizations is the construction of good quality meshes on the DFN. Several approaches have been proposed in order to numerically deal with such an issue. A first family of methods is mainly driven by ensuring conformity of meshes at the fracture intersections, called traces, by the introduction of modifications in the fractures or in the traces [30, 22]. Another proposed approach is based on the use of the mortar method [34, 35], which allows the relaxation of the conformity constraints. A different approach consists in modeling the DFN as a distribution of one-dimensional (1D) pipes mutually connecting the centers of fracture intersections [20, 13, 25]. The resulting mesh of pipes should model the DFN topology and hydraulic properties via a suitable definition of hydrogeological parameters. In [31] the solution in the fractures is written in terms of the solution at the fracture intersections. Other interesting approaches can be found in [26, 17, 32].

In recent papers [3, 4, 5] a new optimization approach for flow simulations on arbitrary DFNs was proposed. The approach totally circumvents the problem of mesh generation without any need for geometrical modification tailored on the DFN (e.g., fracture or trace removal or displacement). Furthermore, the method is naturally conceived in a fracture-oriented way, and decoupled computations on the fractures are envisaged. The method has proven to be quite robust on several medium-sized DFNs [1].

Efficient solvers for DFN flow simulations are of crucial importance in several respects. They are clearly needed for very large scale simulations at the basin scale, where the number of fractures involved may realistically be as high as 10^6 fractures. In addition to efficiency, robustness is a crucial issue, especially when a large number of simulations are required and expected to run in a completely automatic way, without the need of ad hoc human intervention. This is the case, for example, of uncertainty quantification analysis on DFNs [2]. Indeed, when a stochastic geometry is considered, the DFNs on which simulations have to be performed are built on the basis of a given probability distribution, and each simulation contributes to the analysis. Robustness is of crucial importance in this case, since either neglecting DFNs on which simulations fail or modifying them would in turn correspond to a change in the probability distribution adopted.

The aim of the present paper is twofold. A parallel implementation of the method is described, and its scalability is analyzed on quite general stochastically generated DFNs. Furthermore, we aim at showing robustness and efficiency of the approach also on rather general DFNs presenting a large heterogeneity in fracture dimensions, distance, and angles formed by intersecting traces. In particular, very close traces on the same fracture and nearly parallel intersecting traces will be treated without any need of tailored mesh generation.

The paper is organized as follows. In section 2 the model and its finite dimensional discretization are briefly sketched. In section 3 the parallel algorithm is described, and numerical tests are presented in section 4.

2. Model description. We consider here a network surrounded by an impervious rock matrix, so that flow occurs only along fractures and across fracture intersections. The flow in the fractures is ruled by the Darcy law, and matching conditions at fracture intersections are imposed in order to ensure conservation and head continuity. The flow entering/exiting through fracture intersections acts as a source/sink on the fracture. In what follows, we briefly describe the model and its discretization, referring the reader to [3, 5] for full details.

Let Ω denote a DFN in three-dimensional (3D) space, composed of connected open

planar fractures F_i for $i = 1, \dots, I$. Intersections between fractures, called traces, are denoted by S_m, with $m = 1, \dots, M$. Each trace is given by the intersection of exactly two fractures, such that $S_m = \bar{F}_i \cap \bar{F}_j$ with $i \neq j$ collected in the set I_{S_m}. The set of traces belonging to fracture F_i is denoted by $\mathcal{S}_i$.

The boundary of Ω, $\partial\Omega$, is divided into a Dirichlet part Γ_D and a Neumann part Γ_N, such that $\Gamma_D \cap \Gamma_N = \emptyset$ and $\Gamma_D \cup \Gamma_N = \partial\Omega$. The boundary of each fracture F_i is denoted by ∂F_i and is also split into a Dirichlet part $\Gamma_{iD} = \partial F_i \cap \Gamma_D$ and a Neumann part $\Gamma_{iN} = \partial F_i \cap \Gamma_N$. Functions b^D and b^N prescribe the Dirichlet and Neumann boundary conditions, and their restrictions to the boundary ∂F_i are indicated by b_i^D and b_i^N, respectively. We remark that while Γ_D should be nonempty, Γ_{iD} can be empty for some i (see [5]).

The hydraulic head in Ω is denoted by H, and H_i is the restriction of H to F_i. The following functional spaces are introduced on each fracture F_i: the space $V_i = \mathrm{H}^1_0(F_i) = \left\{v \in \mathrm{H}^1(F_i) : v_{|\Gamma_{iD}} = 0\right\}$ and $V_i^D = \mathrm{H}^1_{\mathrm{D}}(F_i) = \left\{v \in \mathrm{H}^1(F_i) : v_{|\Gamma_{iD}} = b_i^D\right\}$, such that, on each fracture, the solution H_i belongs to V_i^D. Let $\mathbf{K}_i$ denote the fracture transmissivity tensor on F_i. On each trace S_m, for each fracture F_i, $[\![\frac{\partial H_i}{\partial \hat{\nu}^i_{S_m}}]\!]_{S_m}$ denotes the jump of the conormal derivative along the unit vector normal to S_m on F_i.

The hydraulic head can be expressed by the following system of equations:

$$\int_{F_i} \mathbf{K}_i \nabla H_i \nabla v \mathrm{d}\Omega = \int_{F_i} q_i v \mathrm{d}\Omega + \int_{\Gamma_{iN}} b_i^N v_{|\Gamma_{iN}} \mathrm{d}\gamma + \sum_{m=1}^{M} \int_{S_m} \left[\!\!\left[\frac{\partial H_i}{\partial \hat{\nu}^i_{S_m}} \right]\!\!\right]_{S_m} v_{|S_m} \mathrm{d}\gamma \qquad \forall v \in V_i,\ \forall i = 1, \dots, I, \tag{2.1}$$

$$H_{i|S_m} - H_{j|S_m} = 0 \qquad \text{for } i, j \in I_{S_m},\ \forall m = 1, \dots, M, \tag{2.2}$$

$$\left[\!\!\left[\frac{\partial H_i}{\partial \hat{\nu}^i_{S_m}} \right]\!\!\right]_{S_m} + \left[\!\!\left[\frac{\partial H_j}{\partial \hat{\nu}^j_{S_m}} \right]\!\!\right]_{S_m} = 0 \qquad \text{for } i, j \in I_{S_m},\ \forall m = 1, \dots, M, \tag{2.3}$$

where $q_i \in \mathrm{L}^2(F_i)$ is a source term for fracture F_i. Equation (2.1) is the weak formulation of the Darcy law on the fracture. Equations (2.2)–(2.3) enforce continuity of the hydraulic head and flux balance across the traces, respectively. Note that the last term on the right-hand side of (2.1) accounts for the flux incoming in F_i through its traces.

In order to decouple the resolution of (2.1)–(2.3), the problem has been reformulated in [3] as a minimization problem, in which the exact fulfillment of (2.2)–(2.3) is replaced by the minimization of a cost functional. For the sake of simplicity of notation we assume in this subsection that the traces are disjoint; this simplifying assumption can be removed following [3, Remark 2.1]. Let us define on each trace S_m and for each fracture F_i the quantity $U_i^m \in \mathrm{H}^{-\frac{1}{2}}(S_m)$ as

$$U_i^m = \left[\!\!\left[\frac{\partial H_i}{\partial \hat{\nu}^i_{S_m}} \right]\!\!\right]_{S_m} + \alpha H_{i|S_m},$$

where α is a positive constant. We define the cost functional J as

$$J(H, U) = \sum_{m=1}^{M} \left(\left\| H_{i|S_m} - H_{j|S_m} \right\|^2_{\mathrm{H}^{\frac{1}{2}}(S_m)} + \left\| U_i^{S_m} + U_j^{S_m} - \alpha \left(H_{i|S_m} + H_{j|S_m} \right) \right\|^2_{\mathrm{H}^{-\frac{1}{2}}(S_m)} \right), \tag{2.4}$$

where i, j are such that $i, j \in I_{S_m}$. The unique minimizer of the quadratic functional J constrained by equations

(2.5)
$$\int_{F_i} \mathbf{K}_i \nabla H_i \nabla v \mathrm{d}\Omega + \alpha \sum_{m=1}^{M} \int_{S_m} H_{i|_{S_m}} v_{|_{S_m}} \mathrm{d}\gamma = \int_{F_i} q_i v \mathrm{d}\Omega + \int_{\Gamma_{iN}} b_i^N v_{|_{\Gamma_{iN}}} \mathrm{d}\gamma + \sum_{m=1}^{M} \int_{S_m} U_i^m v_{|_{S_m}} \mathrm{d}\gamma$$

for all $v \in V_i$, for all $i = 1, \ldots, I$, coincides with the solution of the system (2.1)–(2.3) [3, 5]. Thus, the quantities U_i^m on the traces act as the control variables of the minimization process, while the hydraulic head is the observed variable.

2.1. Discrete problem. The finite dimensional counterpart of the problem depicted in the previous section is obtained by means of finite element discretizations. A key point of the approach used is that each fracture is meshed independently of the others. The triangulations introduced are also independent of trace disposition. A finite dimensional discretization is also introduced for the control variables on each trace. The discretization grid can be independent of the meshes on the fractures.

In what follows, a lower-case letter will be used to denote the discrete variables; thus, the discrete hydraulic head on fracture F_i will be denoted h_i, and the discrete control variable on the trace S_m of fracture F_i will be denoted u_i^m. Let us introduce on F_i the set $\{\varphi_{i,k}\}_{k=1,\ldots,N_i}$ of standard linear finite element basis functions on triangular elements, and on S_m the set $\{\psi_{i,k}^m\}_{k=1,\ldots,N_i^m}$ of linear finite element basis functions in one dimension. Hence, the quantities N_i and N_i^m denote the number of degrees of freedom (DOFs) for h_i and u_i^m, respectively. Hence we write $h_i = \sum_{k=1}^{N_i} h_{i,k}\varphi_{i,k}$, $u_i^m = \sum_{k=1}^{N_i^m} u_{i,k}^m \psi_{i,k}^m$. Overloading the notation, the symbol h_i will also be used to denote the N_i-dimensional vector of the DOFs for the hydraulic head, and u_i^m will be used to denote the N_i^m-dimensional vector of DOFs for the corresponding control variable. The difference will always be clear from the context. Furthermore, we introduce the following vectors. We set $h = (h_1^T, \ldots, h_I^T)^T \in \mathbb{R}^{N^F}$, $N^F = \sum_{i=1,\ldots,I} N_i$; for each fracture F_i we define $u_i \in \mathbb{R}^{N_{\mathcal{S}_i}}$, $N_{\mathcal{S}_i} = \sum_{S_m \in \mathcal{S}_i} N_i^m$, as the vector obtained collecting columnwise the DOFs of the control variables u_i^m on all traces $S_m \in \mathcal{S}_i$, and we define $u \in \mathbb{R}^{N^T}$, $N^T = \sum_{i=1,\ldots,I} N_{\mathcal{S}_i}$, as the vector collecting all the DOFs for the control variables.

The discrete equations for the constraints are obtained in a classical way and can be written on each fracture as

$$A_i h_i = \tilde{q}_i + \mathcal{B}_i\, u_i, \tag{2.6}$$

where $\tilde{q}_i \in \mathbb{R}^{N_i}$ accounts for both the source term and boundary conditions, $A_i \in \mathbb{R}^{N_i \times N_i}$ is the stiffness matrix, and matrix $\mathcal{B}_i \in \mathbb{R}^{N_i \times N_{\mathcal{S}_i}}$ collects the integrals of the basis functions $\varphi_{i,k}$, $k = 1, \ldots, N_i$, against the basis functions $\{\psi_{i,k}^m\}$, $k = 1, \ldots, N_i^m$, $S_m \in \mathcal{S}_i$. It is worth noting that, assuming u_i is known, (2.6) can be solved independently on each fracture and only involves data related to fracture F_i.

Replacing the definitions of the discrete variables in (2.4), considering $\mathrm{L}^2(S_m)$ norms, the following expression for the discrete functional is derived:

$$\begin{aligned} J = \frac{1}{2} \sum_{i=1}^{I} \sum_{S_m \in \mathcal{S}_i} \Bigg(& \int_S \Bigg(\sum_{k=1}^{N_i} h_{i,k} \varphi_{i,k|_{S_m}} - \sum_{k=1}^{N_j} h_{j,k} \varphi_{j,k|_{S_m}} \Bigg)^2 \mathrm{d}\gamma \\ & + \int_{S_m} \Bigg(\sum_{k=1}^{N_i^m} u_{i,k}^m \psi_{i,k}^m + \sum_{k=1}^{N_j^m} u_{j,k}^m \psi_{j,k}^m - \alpha \sum_{k=1}^{N_i} h_{i,k} \varphi_{i,k|_{S_m}} - \alpha \sum_{k=1}^{N_j} h_{j,k} \varphi_{j,k|_{S_m}} \Bigg)^2 \mathrm{d}\gamma \Bigg). \end{aligned} \tag{2.7}$$

For each fracture F_i let us introduce the vector h_i^+ obtained appending to h_i all vectors h_j referring to fractures F_j intersecting with F_i; namely, we append to h_i vectors h_j with $j \in I_{S_m}$, $j \neq i$, for all $S_m \in \mathcal{S}_i$; thus,

$$h_i^+ \in \mathbb{R}^{N_i^+}, \qquad N_i^+ = N_i + \sum_{j \in I_{S_m}, j \neq i, S_m \in \mathcal{S}_i} N_j.$$

Similarly, we define the vector $u_i^+ \in \mathbb{R}^{N_{\mathcal{S}_i}^+}$ by collecting all vectors u_i^m and u_j^m with $i, j \in I_{S_m}$ for all $S_m \in \mathcal{S}_i$ (thus, $N_{\mathcal{S}_i}^+ = \sum_{k \in I_{S_m}, S_m \in \mathcal{S}_i} N_k^m$). Rearranging the various integrals in (2.7) into matrices, the discrete functional is written in algebraic form as

$$J = \frac{1}{2} \sum_{i=1}^{I} \left(h_i^T G_i^h h_i^+ + u_i^T G_i^u u_i^+ - \alpha (u_i^T B_i^u h_i^+ + h_i^T B_i^h u_i^+) \right), \tag{2.8}$$

where matrices $G_i^h \in \mathbb{R}^{N_i \times N_i^+}$ collect the integrals involving only basis functions for the hydraulic head; matrices $G_i^u \in \mathbb{R}^{N_{\mathcal{S}_i} \times N_{\mathcal{S}_i}^+}$ collect the integrals involving only basis functions for the control variables; and matrices $B_i^u \in \mathbb{R}^{N_{\mathcal{S}_i} \times N_i^+}$ and $B_i^h \in \mathbb{R}^{N_i \times N_{\mathcal{S}_i}^+}$ collect the integrals involving basis functions both for the hydraulic head and for the control variables.

Grouping (2.6) on the fractures, and further assembling the terms in the functional (2.8), it is possible to derive a compact form of the DFN problem as an equality-constrained quadratic programming problem:

$$\min \;\; J(h,u) := \frac{1}{2} \left(h^T G^h h - \alpha h^T B^h u - \alpha u^T B^u h + u^T G^u u \right) \tag{2.9}$$

$$\text{s.t.} \;\; Ah - \mathcal{B}\, u = q, \tag{2.10}$$

where $G^h \in \mathbb{R}^{N^F \times N^F}$, $G^u \in \mathbb{R}^{N^T \times N^T}$, $A \in \mathbb{R}^{N^F \times N^F}$, $B^h \in \mathbb{R}^{N^F \times N^T}$, $B^u \in \mathbb{R}^{N^T \times N^F}$, $\mathcal{B} \in \mathbb{R}^{N^F \times N^T}$ are built blockwise starting from the corresponding local matrices.

Remark 1. Equations (2.9)–(2.10), defined on the whole DFN, are here formally written in order to highlight the structure of the problem. The problem will be actually treated in a fracture-oriented way. We also highlight that the global matrices B^h and B^u satisfy $B^h = (B^u)^T$, whereas this property is in general not satisfied by the local matrices B_i^h and B_i^u.

Thanks to linearity of the constraints, we can (formally) write $h = A^{-1}(\mathcal{B}\, u + q)$ and eliminate the unknown h from J, thus obtaining the equivalent unconstrained minimization problem

$$\begin{aligned} \min \hat{J}(u) := \frac{1}{2} u^T (\mathcal{B}^T A^{-T} G^h A^{-1} \mathcal{B} + G^u - \alpha \mathcal{B}^T A^{-T} B - \alpha B^T A^{-1} \mathcal{B}) u \\ + q^T A^{-T} (G^h A^{-1} \mathcal{B} - \alpha B) u. \end{aligned} \tag{2.11}$$

We may compactly write the functional $\hat{J}(u)$ as $\hat{J}(u) = \frac{1}{2} u^T \hat{G} u + \hat{q}^T u$. The matrix $\hat{G}$ is symmetric positive definite. Indeed, it is clearly symmetric positive semidefinite by construction. Positive definiteness follows from uniqueness of the solution to the unconstrained optimization problem (2.11), which can be stated using arguments similar to those used in [3].

Problem (2.11) can be solved via an iterative gradient based method, such as the conjugate gradient (CG) method, which calls for the repeated evaluation of the

gradient of $\hat{J}$. This is given by

$$\nabla \hat{J}(u) = (\mathcal{B}^T A^{-T} G^h A^{-1} \mathcal{B} + G^u - \alpha(\mathcal{B}^T A^{-T} B + B^T A^{-1} \mathcal{B}))u + (\mathcal{B}^T A^{-T} G^h - \alpha B^T) A^{-1} q = \hat{G}u + \hat{q}.$$

Our method can be written as a parallel version of the CG method applied to the functional $\hat{J}$ or, equivalently, to the linear system $\hat{G}u + \hat{q} = 0$, without explicitly assembling either the matrix $\hat{G}$ or the vector $\hat{q}$. The core of each iteration of the CG method is the computation of the gradient of $\hat{J}$; within the approach adopted here, this can be easily performed in parallel on each fracture of the DFN.

In order to describe the approach, let us note that the gradient can be written in terms of some auxiliary variables as follows. Recalling that $A^{-1}(\mathcal{B}\, u + q) = h$, we have

$$\nabla \hat{J}(u) = \mathcal{B}^T A^{-T} G^h h + G^u u - \alpha\, \mathcal{B}^T A^{-T} B u - \alpha B^T h,$$

and setting $p := A^{-T}(G^h h - \alpha B u)$, we get

$$\nabla \hat{J}(u) = \mathcal{B}^T p + G^u u - \alpha B^T h. \tag{2.12}$$

As a consequence, for a given u, the gradient of $\hat{J}$ can be computed by (2.12) once the linear systems

$$A h = \mathcal{B}\, u + q, \qquad A^T p = G^h h - \alpha B u \tag{2.13}$$

are solved. These linear systems, which are formally written in (2.13) on the whole DFN, can be split into I smaller "local" systems, defined on the fractures, resorting to the matrix blocks previously introduced and defined locally on the fractures. Namely, recalling the definition of quantities h_i^+ and u_i^+ provided in view of (2.8), equations (2.12) and (2.13) correspond to the following: for $i = 1, \ldots, I$,

$$A_i h_i = \tilde{q}_i + \mathcal{B}_i\, u_i, \tag{2.14}$$

$$A_i^T p_i = G_i^h h_i^+ - \alpha B_i^h u_i^+, \tag{2.15}$$

$$(\nabla \hat{J})_i = \mathcal{B}_i^T\, p_i + G_i^u u_i^+ - \alpha B_i^u h_i^+. \tag{2.16}$$

As shown in (2.14)–(2.16), the gradient $\nabla \hat{J}(u)$ can therefore be computed as a contribution from the various fractures in the DFN. The evaluation of the contribution $(\nabla \hat{J})_i$ only requires, as information additional to that available on fracture F_i itself, information from the fractures intersecting F_i in order to assemble vectors u_i^+ and h_i^+. Furthermore, given the structure of the functional in (2.7), and consequently the sparsity pattern of the matrices involved in the computations, only a small subset of the total DOFs of the hydraulic head need to be shared, i.e., only the DOFs corresponding to values of the hydraulic head on the traces.

Remark 2. The idea of splitting the computations for each CG iteration into subproblems defined on subdomains is in agreement with classical domain decomposition (DD) approaches [37, 38] already applied to DFN flow simulations in [36]. Although there is a similarity between the approach depicted here and DD methods, we highlight that the structure of each CG iteration is not the same, due to the different structure of the reduced Hessian $\hat{G}$. In classical DD methods for elliptic problems, the problem can be formulated as the minimization of an energy defined on the whole domain Ω, given by the sum of a number of contributions relative to subdomains Ω_i. When the

problem is split into subproblems on the subdomains, continuity constraints are imposed on the interfaces through the introduction of auxiliary variables. Here, on the contrary, the functional to be minimized is defined on the interfaces (i.e., the traces) and is subject to constraints defined on the fractures, i.e., both on the subdomains and on the traces. In both methods, one may proceed with a reduction to a problem defined on the interfaces, but while in the DD case the problem is reformulated in terms of the auxiliary variables introduced, in our case it is reformulated in terms of one of the unknowns of the original problem, as the internal variables h and the constraint Lagrange multipliers p are eliminated, leading to a system involving only the control variable u.

3. The parallel algorithm. In this section we describe in detail the parallel algorithm for applying the procedure depicted in the previous section.

Let k be the iteration index, and let g^k denote the gradient $\nabla \hat{J}(u)$ at iteration k, obtained by grouping columnwise vectors $g_i^k := (\nabla \hat{J})_i$ for all $i = 1, \ldots, I$.

Let us introduce the symbol δ to denote the increment of a given quantity within iterations, e.g., $\delta u_i^k := u_i^{k+1} - u_i^k$. From (2.14)–(2.16) we straightforwardly have $\delta h_i^k = A_i^{-1}\, \mathcal{B}_i\, \delta u_i^k$, $\delta p_i^k = A_i^{-T}(G_i^h \delta (h_i^+)^k - \alpha B_i^h \delta (u_i^+)^k)$, and $\delta g_i^k = \mathcal{B}_i^T\, \delta p_i^k + G_i^u \delta (u_i^+)^k - \alpha B_i^u \delta (h_i^+)^k$.

The serial algorithm is detailed in the following. The direction of movement for u is denoted d^k. Vectors d_i^k and $(d_i^+)^k$ are derived from d^k following the same rules used for deriving u_i^k and $(u_i^+)^k$ from u^k.

Serial CG algorithm

1. Choose an initial guess u^0 and set $k = 0$
2. Compute h_i^k and p_i^k solving (2.14) and (2.15) and g_i^k by (2.16) $\forall i = 1, \ldots, I$
3. Compute $\beta_{i,N}^k = (g_i^k)^T g_i^k$ $\forall i = 1, \ldots, I$
4. Set $d_i^k = -g_i^k$ $\forall i = 1, \ldots, I$
5. While $g^k \neq 0$
   - 5.1. Compute δg_i^k:
     - i. Solve $A_i \delta h_i^k = \mathcal{B}_i\, d_i^k$
     - ii. Solve $A_i^T \delta p_i^k = G_i^h \delta (h_i^+)^k - \alpha B_i^h (d_i^+)^k$
     - iii. Compute $\delta g_i^k = \mathcal{B}_i^T\, \delta p_i^k + G_i^u (d_i^+)^k - \alpha B_i^u \delta (h_i^+)^k$
   - 5.2. Compute λ^k with an exact line search along d^k
   - 5.3. Update $u_i^{k+1} = u_i^k + \lambda^k d_i^k$
   - 5.4. Update $g_i^{k+1} = g_i^k + \lambda^k \delta g_i^k$ $\forall i = 1, \ldots, I$
   - 5.5. Set $\beta_{i,D}^{k+1} = \beta_{i,N}^k$ and compute $\beta_{i,N}^{k+1} = \left(g_i^{k+1}\right)^T g_i^{k+1}$ $\forall i = 1, \ldots, I$
   - 5.6. Compute $\beta^{k+1} = \frac{\sum_{i=1}^I \beta_{i,N}^{k+1}}{\sum_{i=1}^I \beta_{i,D}^{k+1}}$
   - 5.7. Update $d_i^{k+1} = -g_i^{k+1} + \beta^{k+1} d_i^k$
   - 5.8. $k = k + 1$

The exact line search at step 5.2 is performed as follows:

1. Compute $\lambda_{i,N}^k = (d_i^k)^T g_i^k$ and $\lambda_{i,D}^k = (d_i^k)^T \delta g_i^k$ $\forall i = 1, \ldots, I$
2. Compute $\lambda^k = \frac{\sum_{i=1}^I \lambda_{i,N}^k}{\sum_{i=1}^I \lambda_{i,D}^k}$

Thus the same quantity δg_i^k computed at step 5.1 is required. It can be noticed

that all the required computations only involve the resolution of local linear systems defined on the fractures. As such, the algorithm is easily parallelizable by splitting the computations among parallel processes.

Remark 3. Step 5.1 requires the solution of two linear systems on each fracture. We point out that matrices A_i are constant within iterations; hence their factorization is computed just once. Moreover, since the method is specifically designed for easily tackling complex geometries without the need of mesh conformity, overrefinement due to mesh generation problems is avoided, and very coarse meshes can be used on the fractures; thus, the local matrices A_i are typically of moderate size.

For parallel computations, we consider a distributed memory architecture, with communications among processors based on the message passing interface (MPI) protocol. Let N_p be the number of concurrent processes P_n, $n = 0, \dots, N_p - 1$. Each process P_n is assumed to manage a set of I_n fractures. Let $\mathcal{F}_n$ denote the set of fractures assigned to process P_n. In the MPI-parallel version four communication phases need to be added to the previous serial algorithm: the communication of the scalar quantities $\lambda_{i,N}^k, \lambda_{i,D}^k$ and $\beta_{i,N}^{k+1}, \beta_{i,D}^{k+1}$ and the communication of the two vector quantities δh_i^k and d_i^k.

The parallel version of the method is described in the following algorithm.

Parallel CG algorithm

1. Open the parallel environment with N_p processes

Process P_0 only:

2. Partition the DFN into N_p subsets of fractures

All processes:

3. Choose an initial guess u_i^0, and set $k = 0$
4. Compute h_i^k and p_i^k solving (2.14)–(2.15) and g_i^k by (2.16) $\forall i$ s.t. $F_i \in \mathcal{F}_n$
5. Compute $\beta_{i,N}^k = (g_i^k)^T g_i^k$ $\forall i$ s.t. $F_i \in \mathcal{F}_n$
6. Set $d_i^k = -g_i^k$ $\forall i$ s.t. $F_i \in \mathcal{F}_n$. *Communication phase:* d_i^k
7. While $g_k \neq 0$
   - 7.1. Compute δg_i^k
     - i. Solve $A_i \delta h_i^k = \mathcal{B}_i\, d_i^k$ $\forall i$ s.t. $F_i \in \mathcal{F}_n$. *Communication phase:* δh_i^k
     - ii. Solve $A_i^T \delta p_i^k = G_i^h \delta (h_i^+)^k - \alpha B_i^h (d_i^+)^k$ $\forall i$ s.t. $F_i \in \mathcal{F}_n$
     - iii. Compute $\delta g_i^k = \mathcal{B}_i^T\, \delta p_i^k + G_i^u (d_i^+)^k - \alpha B_i^u \delta (h_i^+)^k$ $\forall i$ s.t. $F_i \in \mathcal{F}_n$
   - 7.2. Compute $\lambda_{i,N}^k$ and $\lambda_{i,D}^k$ $\forall i$ s.t. $F_i \in \mathcal{F}_n$.
   - 7.3. *Communication of $\lambda_{i,N}^k$ and $\lambda_{i,D}^k$ and computation of λ^k through MPI Reduce operation*
   - 7.4. Update $u_i^{k+1} = u_i^k + \lambda^k d_i^k$ $\forall i$ s.t. $F_i \in \mathcal{F}_n$
   - 7.5. Update $g_i^{k+1} = g_i^k + \lambda^k \delta g_i^k$ $\forall i$ s.t. $F_i \in \mathcal{F}_n$
   - 7.6. Set $\beta_{i,D}^{k+1} = \beta_{i,N}^k$, and compute $\beta_{i,N}^{k+1} = \left(g_i^{k+1}\right)^T g_i^{k+1}$ $\forall i$ s.t. $F_i \in \mathcal{F}_n$
   - 7.7. *Communication of $\beta_{i,N}^k$ and $\beta_{i,D}^k$ and computation of β^{k+1} through MPI Reduce operation*
   - 7.8. Update $d_i^{k+1} = -g_i^{k+1} + \beta^{k+1} d_i^k$ $\forall i$ s.t. $F_i \in \mathcal{F}_n$. *Communication phase:* d_i^k
   - 7.9. $k = k + 1$

The process of DFN partitioning is performed using the code METIS [24, 23]. A graph is associated to the network of fractures: the nodes correspond to the fractures,

and the arcs correspond to the traces. The weight of each node of the graph is given by the number of DOFs of the discretization introduced on the fracture. METIS provides a weight-balanced partitioning of the graph minimizing the number of arc-cuts, i.e., the number of communications, recalling that only trace-related data need be communicated.

After the partitioning, some of the fractures in $\mathcal{F}_n$ will have intersections only with other fractures in $\mathcal{F}_n$ itself, thus having no part in the communication phase. Let $\mathcal{F}_n^{\mathrm{nc}}$ be the subset of the noncommunicating fractures in $\mathcal{F}_n$, with cardinality I_n^{nc}, and let $\mathcal{F}_n^{\mathrm{c}}$ be the subset of the communicating fractures, i.e., the subset of fractures in $\mathcal{F}_n$ forming traces with fractures managed by a process different from P_n. Further, let $\mathcal{S}_n^{\mathrm{c}}$ be the set of these traces and $\mathcal{P}_n$ the set of the processes other than P_n that manage at least one fracture intersecting a fracture in $\mathcal{F}_n^{\mathrm{c}}$. Let I_n^{c} be the cardinality of $\mathcal{F}_n^{\mathrm{c}}$ and $I_{\mathcal{P}_n}$ the cardinality of $\mathcal{P}_n$. At each iteration k of the algorithm, each process P_n is required to send only the DOFs of d_i^k for each trace $S \in \mathcal{S}_n^{\mathrm{c}}$ and the subset of DOFs of δh_i^k that affect the value of δh_i^k itself on the traces in $\mathcal{S}_n^{\mathrm{c}}$. Similarly, P_n needs to receive information related to the traces in $\mathcal{S}_n^{\mathrm{c}}$ from the processes in $\mathcal{P}_n$.

Two different communication strategies, or topologies, can be envisaged for the parallel algorithm: a *MasterSlave* topology or a *Point-to-Point* topology. In the *MasterSlave* communication architecture a hierarchy of processes is established, and the presence of a *Master* process (process P_0) managing a group of *Slave* processes $(P_1, \ldots, P_{N_p-1})$ is considered. The *Slave* processes actually perform the required computations, while the *Master* process is devoted to the partitioning of the DFN and to the communication phases. Thus, at step 2 of the parallel CG algorithm the DFN is actually partitioned into $N_p - 1$ subsets of fractures. The *Slave* processes communicate with each other through the *Master* process, according to the following communication scheme. Each *Slave* process P_n, $n = 1, \ldots, N_p - 1$, packs the required data into a unique array and sends it to the *Master* process. The *Master* process, while receiving information, rearranges the data into $N_p - 1$ new arrays. As soon as the array for the *Slave* P_n is complete with all the data coming from the processes in $\mathcal{P}_n$, the *Master* sends it to the process. As a consequence, with the *MasterSlave* architecture, at each iteration of the algorithm the number of communications is proportional to the number of processes N_p, while the amount of data to be shared strongly depends on the structure of the DFN and on the partitioning process. Further, due to the different loading of the *Slaves*, communications take place at slightly different times, thus avoiding congestion of the *Master*. For very large DFNs it is possible to have more than one *Master* process managing groups of *Slave* processes, and a *MasterofMasters* process in charge of the communications between different *Master* processes.

In the *Point-to-Point* communication scheme, on the contrary, all the available processes contribute to perform the computations. The *Point-to-Point* communication strategy requires that all processes P_n, $n = 0, \ldots, N_p - 1$, send to and receive from all the processes in $\mathcal{P}_n$ the required data. In this case, the number of communications is therefore proportional to $\sum_{n=0}^{N_p-1} I_{\mathcal{P}_n}$, thus strongly depending on the structure of the DFN and on the partitioning. The *MasterSlave* architecture is preferred for the present work, as it allows us to keep the number of required communications as low as possible, independently of the complexity of the DFN. Further, the amount of data to be shared is not considered a bottleneck even for DFNs with a large number of traces, as it is limited to few DOFs related to the traces in the set $\mathcal{S}_n^{\mathrm{c}}$.

Step 7.1 of the parallel CG method is here further detailed as an example of the communication process with the *MasterSlave* topology as described above.

MasterSlave - Communication algorithm

7.1. Compute δg_i^k:

if process P_0:

i. For $n = 1, \ldots, N_p - 1$
   a. Receive information from process P_n
   b. For $m = 1, \ldots, N_p - 1$
      If $P_n \in I_{\mathcal{P}_m}$, extract from P_n information for P_m and store it in the array v_m.

ii. For $n = 1, \ldots, N_p - 1$
   Perform an MPI nonblocking send of v_n to process P_n

if process P_n, $n \neq 0$:

i. Solve $A_i \delta h_i^k = \mathcal{B}_i \, d_i^k \; \forall i$ s.t. $F_i \in \mathcal{F}_n^{\mathrm{c}}$
ii. Extract DOFs of δh_i^k related to the traces in $\mathcal{S}_n^{\mathrm{c}}$ and collect them in an array h_{c}
iii. Perform an MPI nonblocking send of δh_{c} to process P_0
iv. Solve $A_i \delta h_i^k = \mathcal{B}_i \, d_i^k \; \forall i$ s.t. $F_i \in \mathcal{F}_n^{\mathrm{nc}}$
v. Receive information from process P_0 and update vector $\delta(h_i^+)^k$
vi. Solve $A_i^T \delta p_i^k = G_i^h \delta(h_i^+)^k - \alpha B_i^h (d_i^+)^k \; \forall i$ s.t. $F_i \in \mathcal{F}_n$
vii. Compute $\delta g_i^k = \mathcal{B}_i^T \, \delta p_i^k + G_i^u (d_i^+)^k - \alpha B_i^u \delta(h_i^+)^k \; \forall i$ s.t. $F_i \in \mathcal{F}_n$

4. Numerical results. In this section we report numerical results in order to assess the performance of the method. Several DFN configurations with different features have been considered.

4.1. Test problem description. Four different DFN configurations are used for the simulations, obtained as follows. Two networks have been stochastically generated, labeled DFN709 and DFN1425, consisting of 709 and 1425 fractures, respectively. In Table 1 we report the number of fractures and some relevant data about the number of traces of all the DFNs considered: the total number of traces M, the minimum and maximum number of traces on the fractures, and extremal traces densities, i.e., the ratio between the number of traces over the fracture area. The geometrical characteristics of the considered DFNs are described by Figures 1–3. Namely, in Figure 1 a 3D plot of the 709 and 1425 fracture DFNs is reported. The coloring of fractures in these images is proportional to fracture areas: the fracture areas span several orders of magnitude, ranging from $376\,\mathrm{m}^2$ to $10^4\,\mathrm{m}^2$ in the smaller DFN, and from $3.2\,\mathrm{m}^2$ to $10^4\,\mathrm{m}^2$ in the larger DFN. Figures 2–3 report histograms of trace related data. Figure 2 shows the distribution of angles formed by intersecting traces in a single fracture: on the left the full distribution of the angles is reported, with a detail of the lower values on the right. The DFNs have traces intersecting orthogonally as well as traces intersecting with very small angles, down to 0.26 degrees.

TABLE 1
DFN related data.

		Traces			Trace density	
	I	M	Min	Max	Min	Max
DFN709	709	3939	1	42	0.00080	0.042
DFN878	878	3835	1	43	0.00073	0.152
DFN915	915	4318	1	49	0.00073	0.613
DFN1425	1425	13086	1	92	0.00073	0.920

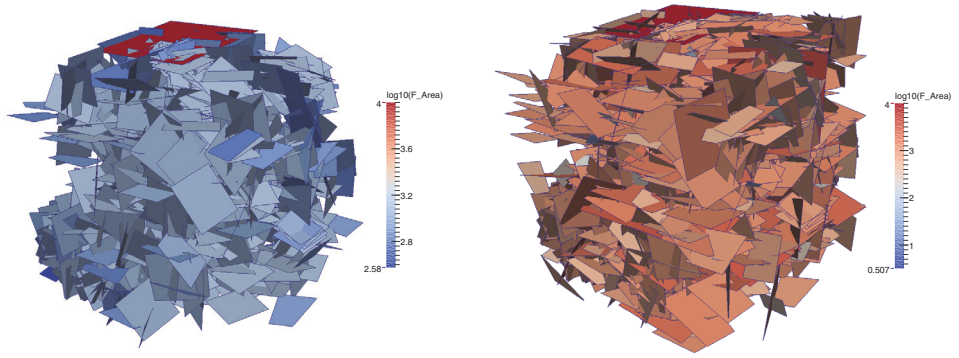

FIG. 1. *3D view of DFN*709 *(left) and DFN*1425 *(right).*

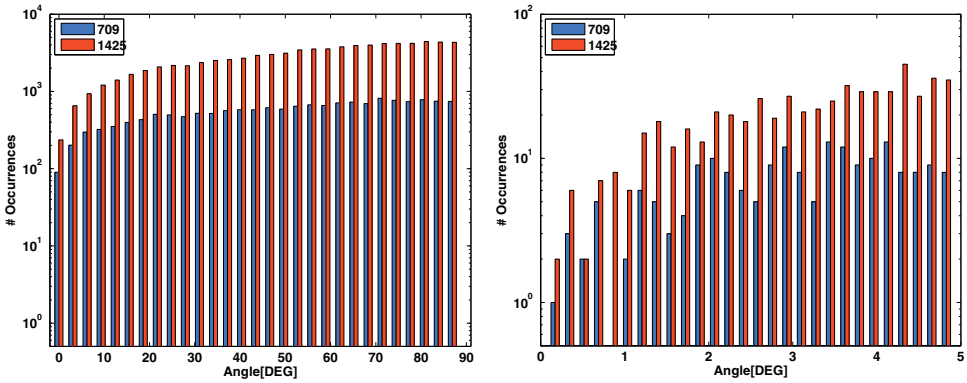

FIG. 2. *DFN*709 *and DFN*1425. *Left: Distribution of angles between pairs of intersecting traces in the same fracture. Right: Zoom of lower values.*

In Figure 3 we report the distribution of trace lengths (left) and of distance between couples of nonintersecting traces in the same fracture (right). The figures highlight the multiscale nature of the DFNs considered, with trace lengths ranging from approximately 1.6 cm to 260 m, and with disjoint trace distances down to 0.12 mm.

Two additional networks have also been generated, starting from DFN1425: they have been obtained by removing some selected fractures in such a way that the new networks do not present geometrical features known in literature to be critical, i.e., narrow angles between intersecting traces and small distance of nonintersecting traces in the same fracture. More precisely (refer to Table 1 for the names), DFN878 does not contain nonintersecting traces with distance smaller than 0.3 m, and DFN915 does not contain intersecting traces forming angles smaller than 18 degrees. These threshold values have been chosen in such a way that the DFNs generated have a similar number of fractures, traces, and DOFs.

Summarizing, the provided data highlight that two of the chosen DFN configurations, namely, DFN709 and DFN1425, display the complexities (frequently present in problems of interest for practical applications) that make DFN simulations particularly challenging: these are a strong multiscale nature and the presence of intricate networks of traces intersecting with narrow angles, which are very difficult to mesh,

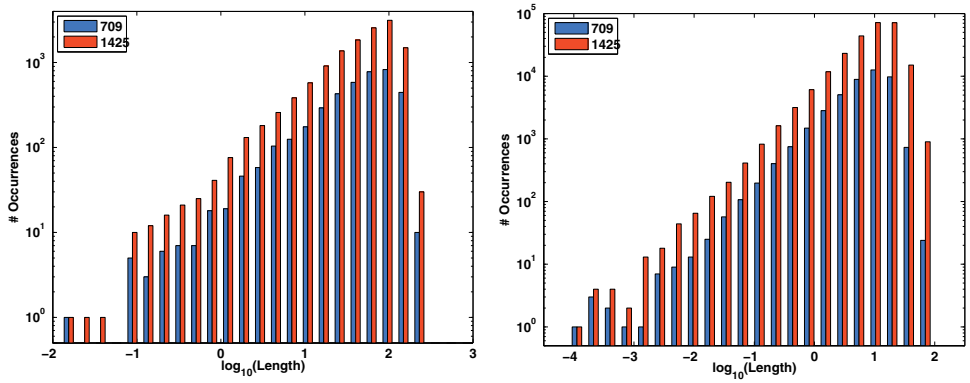

FIG. 3. *DFN*709 *and DFN*1425*: Distribution of trace lengths (left) and of distances between couples of nonintersecting traces in the same fracture (right).*

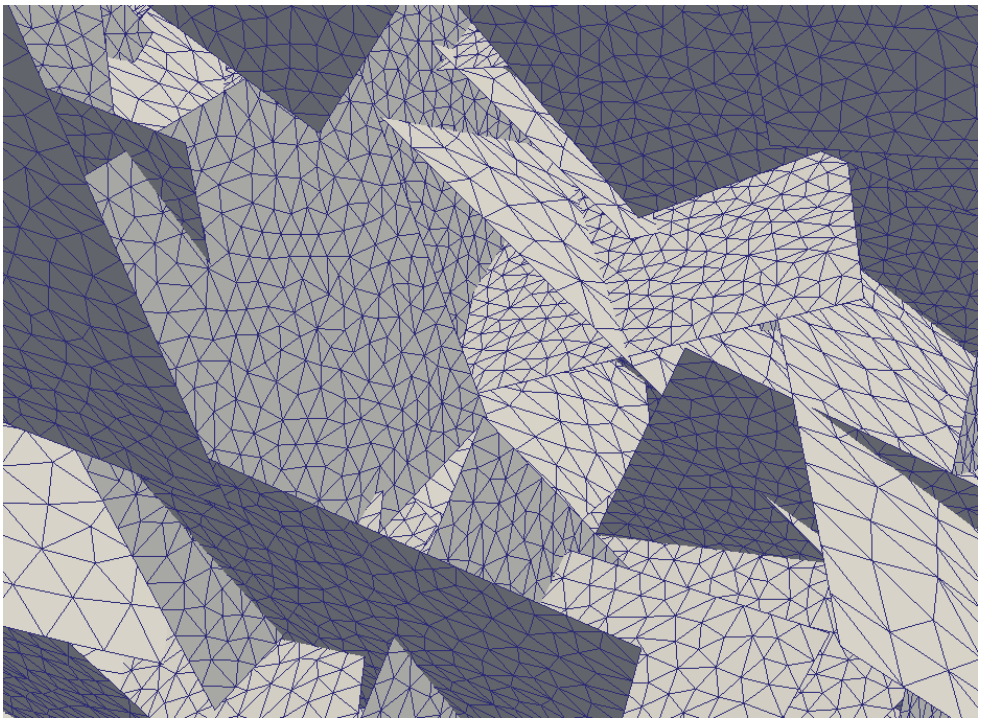

FIG. 4. *DFN*709*: Detail of a computational mesh.*

yielding a huge number of unknowns for the generation of a good quality mesh, if some kind of conformity is required. On the other hand, DFN878 and DFN915 have a less severe geometrical structure, with no extremely narrow angles or very close traces. As an example of the ability of the proposed method to deal with complex networks, a detail of a nonconforming mesh is reported in Figure 4 for DFN709.

Each DFN has been discretized using three different grids for the hydraulic head, with discretization parameter $\delta \in \{0.5, 2, 7\}$. The parameter δ represents the upper bound for the area of triangular elements produced by the triangulator on each frac-

TABLE 2
Number of degrees of freedom for the hydraulic head h and the control variable u.

	N^F			N^T		
DFN	$\delta = 0.5$	$\delta = 2$	$\delta = 7$	$\delta = 0.5$	$\delta = 2$	$\delta = 7$
709	1378796	356527	107467	363007	187645	105817
878	1753469	452856	136549	329584	170733	96759
915	1808368	467474	140799	351899	182739	103701
1425	2838981	733588	220980	1157310	598596	338489

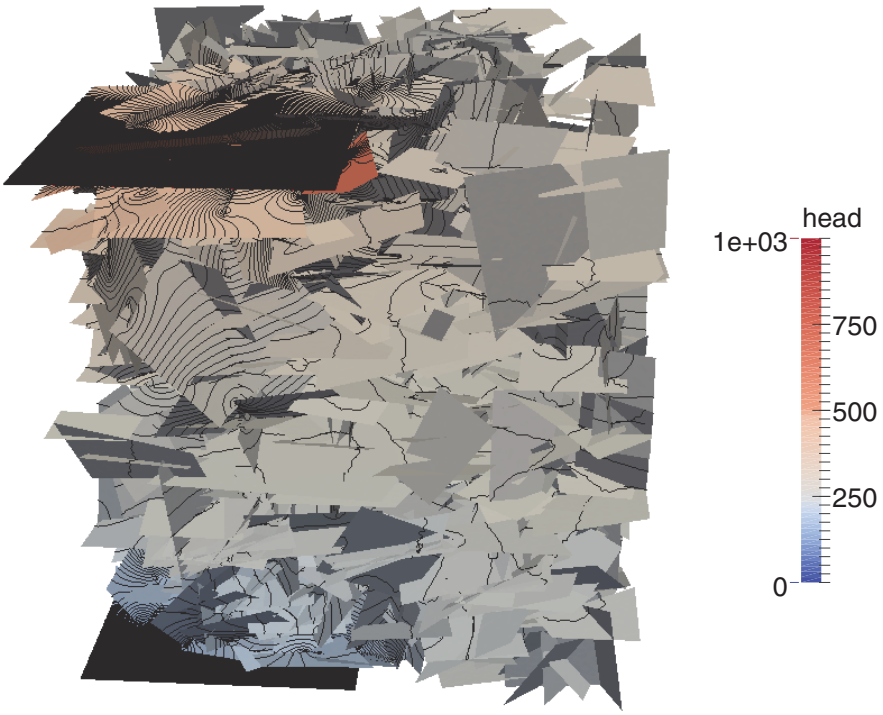

FIG. 5. *Solution for DFN*1425 *with contour lines, $\delta = 2$. The source and sink fractures are the horizontal fractures in the top and bottom left corners, respectively.*

ture. The discretization of the control variables is chosen as induced by the discretization of the hydraulic head; i.e., the grid on the traces is given by the intersection points between the traces and the edges of the triangular elements on the fractures. We remark that other node distributions can be alternatively adopted for the discretization of the control variables, as this choice only affects the quadrature rule used to compute the required integrals. The resulting number of DOFs is reported in Table 2. All the simulations are performed using a constant unitary fracture transmissivity and choosing as the initial guess (see step 1 of the conjugate gradient algorithm) the null vector. Furthermore, the parameter α was set equal to 1. Boundary conditions are fixed as follows: a constant Dirichlet boundary condition $h = 1000$ is prescribed on one edge of a selected fracture (named the *source* fracture), and a homogeneous Dirichlet boundary condition, $h = 0$, is prescribed on one edge of a different fracture

TABLE 3
Relative residuals and flux conservation error Δ_{cons} through iterations.

	relative residual			Δ_{cons}		
# iter	$\delta = 0.5$	$\delta = 2$	$\delta = 7$	$\delta = 0.5$	$\delta = 2$	$\delta = 7$
DFN709						
1000	3.749e-4	4.826e-4	5.767e-4	5.864e-6	8.381e-4	1.056e-3
10000	1.014e-5	1.244e-5	1.845e-5	1.018e-5	1.031e-6	4.497e-5
20000	3.615e-6	4.715e-6	8.539e-6	4.814e-6	2.275e-6	2.177e-5
DFN878						
1000	3.716e-4	3.826e-4	3.944e-4	5.723e-5	1.925e-5	2.685e-5
10000	6.668e-6	9.271e-6	1.462e-5	2.083e-6	1.780e-6	5.027e-7
20000	4.821e-6	3.902e-6	4.705e-6	4.125e-6	2.091e-7	1.096e-6
DFN915						
1000	3.548e-4	3.840e-4	3.989e-4	4.932e-6	4.231e-5	1.962e-4
10000	8.486e-6	9.471e-6	1.636e-5	3.740e-6	8.943e-6	3.462e-6
20000	7.362e-6	5.896e-6	7.442e-6	2.729e-6	5.329e-6	5.204e-6
DFN1425						
1000	4.022e-4	4.209e-4	4.424e-4	7.235e-4	7.678e-4	1.096e-3
10000	7.286e-6	8.687e-6	1.618e-5	6.258e-6	3.426e-5	5.754e-5
20000	2.527e-6	2.891e-6	6.787e-6	6.240e-6	3.683e-5	5.739e-5

TABLE 4
Hydraulic head mismatch Δ_{head} and flux unbalance Δ_{flux} through iterations.

	Δ_{head}			Δ_{flux}		
# iter	$\delta = 0.5$	$\delta = 2$	$\delta = 7$	$\delta = 0.5$	$\delta = 2$	$\delta = 7$
DFN709						
1000	2.116e-6	2.356e-6	2.802e-6	5.683e-10	9.552e-10	1.002e-9
10000	5.138e-7	8.427e-7	1.092e-6	3.278e-10	3.461e-10	4.903e-10
20000	5.002e-7	8.278e-7	1.062e-6	3.202e-10	3.409e-10	4.818e-10
DFN878						
1000	2.892e-6	3.322e-6	3.994e-6	4.863e-10	5.930e-10	7.395e-10
10000	7.318e-7	1.149e-6	1.668e-6	2.382e-10	3.299e-10	4.853e-10
20000	7.030e-7	1.127e-6	1.641e-6	2.400e-10	3.319e-10	4.743e-10
DFN915						
1000	2.774e-6	3.187e-6	3.854e-6	4.397e-10	5.369e-10	6.780e-10
10000	7.595e-7	1.117e-6	1.683e-6	2.196e-10	3.045e-10	4.309e-10
20000	7.109e-7	1.095e-6	1.641e-6	2.211e-10	3.004e-10	4.234e-10
DFN1425						
1000	9.025e-7	1.009e-6	1.231e-6	1.311e-10	1.670e-10	2.102e-10
10000	2.168e-7	3.460e-7	5.397e-7	6.186e-11	9.671e-11	1.168e-10
20000	2.121e-7	3.424e-7	5.284e-7	6.187e-11	9.553e-11	1.146e-10

(the *sink* fracture); all other edges in the DFN are insulated.

4.2. Robustness. In this subsection, we compare the behavior of the method on the four networks considered here. In order to provide an overview of the solution obtained, we report in Figure 5 the computed solution obtained for DFN1425 with the intermediate grid. In the figure the coloring is proportional to the value of the hydraulic head, and contour lines are also displayed to highlight gradients of the solution.

First, we highlight that, as expected, the method was capable of performing the required computations on all the DFNs with all mesh sizes, without any kind of geometrical or mesh adjustment.

Then, we focus on the behavior of the method as far as the fulfillment of head continuity and flux balance is concerned. To this aim, we introduce the following error

measures. Let $\mathcal{I}_{\text{in}}$ and $\mathcal{I}_{\text{out}}$ denote the index sets corresponding to source and sink fractures, respectively. We define the computed average flux through the network as

$$\bar{\Phi} = \frac{1}{2}\left(\sum_{i\in\mathcal{I}_{\text{in}}}\sum_{S_m\in\mathcal{S}_i}\int_{S_m}(u_i^m - \alpha h_{i|S_m}) - \sum_{i\in\mathcal{I}_{\text{out}}}\sum_{S_m\in\mathcal{S}_i}\int_{S_m}(u_i^m - \alpha h_{i|S_m})\right).$$

Then we define the relative flux conservation error per trace unit length as

$$\Delta_{\text{cons}} = \frac{\left|\sum_{i\in\mathcal{I}_{\text{in}}\cup\mathcal{I}_{\text{out}}}\sum_{S_m\in\mathcal{S}_i}\int_{S_m}(u_i^m - \alpha h_{i|S_m})\right|}{\bar{\Phi}};$$

furthermore, we define the relative mismatch in hydraulic head continuity and the relative flux imbalance per trace unit length, respectively, as

$$\Delta_{\text{head}} = \frac{\sqrt{\sum_{m=1}^{M}\left\|h_{i|S_m} - h_{j|S_m}\right\|^2}}{h_{\max}L},\quad \Delta_{\text{flux}} = \frac{\sqrt{\sum_{m=1}^{M}\left\|u_i^m + u_j^m - \alpha\left(h_{i|S_m} + h_{j|S_m}\right)\right\|^2}}{\bar{\Phi}L},$$

L being the cumulative trace length.

In Table 3 we report for all the DFNs and the mesh sizes considered here the values of the relative residual computed through iterations, and of the corresponding flux conservation errors. In Table 4 we report the hydraulic head mismatches and flux imbalances. It is clearly seen that the relative residual and the error measures attained for a fixed number of iterations and mesh are not significantly different, thus resulting in no evidence of influence of geometrical features in the behavior of the method. Namely, the behavior of the method appears to not be affected by the presence of intersecting traces with very narrow angles, or very close nonintersecting traces. We remark that, even if increasing the number of iterations improves the accuracy of the solution of the linear system, the hydraulic head continuity mismatch and the flux imbalance due to the mesh nonconformities reach a *plateau*. Hence, a higher number of iterations would not result in a significant improvement of the mismatches, which could be reduced only by refining the meshes.

Remark 4. The number of iterations required to get acceptable results is quite high, if compared, e.g., with classical preconditioned DD methods. The introduction of a suitable preconditioner in order to reduce the iteration numbers is clearly recommended. On the other hand, since the approach adopted here and classical DD methods yield CG reduced problems with different structures (see Remark 2), classical DD preconditioners [27, 28, 14, 29, 18] cannot be straightforwardly applied to the present context.

4.3. Scalability performance. Simulations for scalability performance assessment have been performed on the *Eurora* cluster, located at the Italian HPC Center Cineca. The cluster has 64 16-core compute nodes. Half of the nodes contain 2 Intel® Xeon® SandyBridge eight-core E5-2658 processors, with a clock rate of about 2.10 GHz, whereas the other half of the cards contain 2 Intel® Xeon® SandyBridge 8-core E5-2687W processors, with a clock of about 3.10 GHz. 58 compute nodes have 16GB of memory. The remaining 6 nodes (with processors at 3 GHz clock rate) have 32 GB RAM. Up to 32 nodes were available for the computations, and a portable batch system (PBS) job-scheduling was used for job submission.

Remark 5. Being the cluster intensively used by many users at the same time, simulation time is strongly influenced by the load status. For this reason, time measurements are performed through repeated simulations for the same configuration,

selecting the minimum time. This procedure, however, can only partially compensate for fluctuations of simulation time due to cluster load.

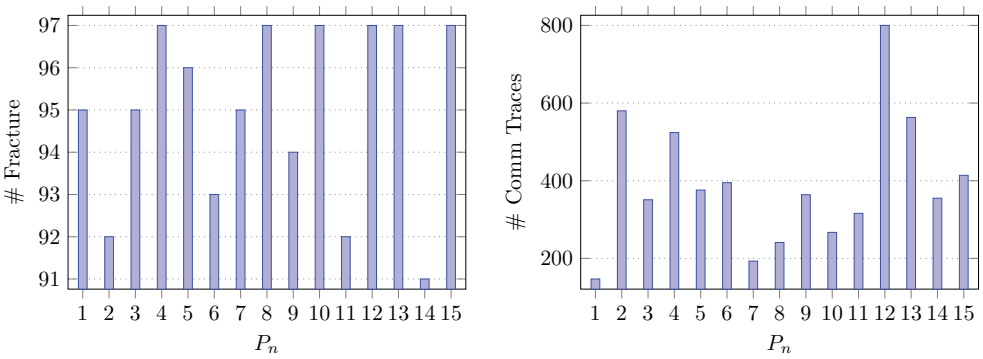

FIG. 6. *DFN*1425*: Number of fractures (left) and communicating traces (right) per process for a* 15 *Slave simulation.*

TABLE 5
Number of communicated h-DOFs and u-DOFs for various numbers of parallel processes.

DFN	Slaves	$\delta = 0.5$	[%]	$\delta = 2$	[%]	$\delta = 7$	[%]
				u DOFs comm			
709	3	6978	[1.9]	3648	[1.0]	2119	[0.6]
	7	17755	[4.9]	9392	[2.6]	5451	[1.5]
	15	36994	[10.2]	19433	[5.4]	11253	[3.1]
	31	114061	[31.4]	59340	[16.3]	33779	[9.3]
1425	3	55096	[4.8]	28694	[2.5]	16488	[1.4]
	7	108229	[9.4]	56773	[4.9]	32788	[2.8]
	15	185821	[16.1]	97148	[8.4]	56057	[4.8]
	31	286723	[24.8]	149710	[12.9]	86405	[7.5]
				h DOFs comm			
709	3	7216	[0.5]	3885	[1.1]	2355	[2.2]
	7	18379	[1.3]	10016	[2.8]	6075	[5.7]
	15	38240	[2.8]	20682	[5.8]	12497	[11.6]
	31	117008	[8.5]	62290	[17.5]	36729	[34.2]
1425	3	56671	[2.0]	30269	[4.1]	18066	[8.2]
	7	111724	[3.9]	60270	[8.2]	36284	[16.4]
	15	191661	[6.8]	103000	[14.0]	61908	[28.0]
	31	295443	[10.4]	158446	[21.6]	95148	[43.1]

The code is implemented using the `C++` language, with the application of the MPI paradigm. Linear algebra computations are performed with the `C++` template library *Eigen* [16], chosen for its reliability on sparse linear algebra. In particular, the SparseCholesky module is used for the computation of the Cholesky factorization for the solution of (2.14), (2.15). The factorization of the matrices A_i is performed only once in the preprocessing step.

Two kind of simulations are performed, labeled MPI and MPI*. In MPI simulations each process runs on a different node, such that all communications occur through the *Infiniband*® network, differing from MPI* simulations, in which eight processes per node are assigned in order to reach a larger number of parallel processes.

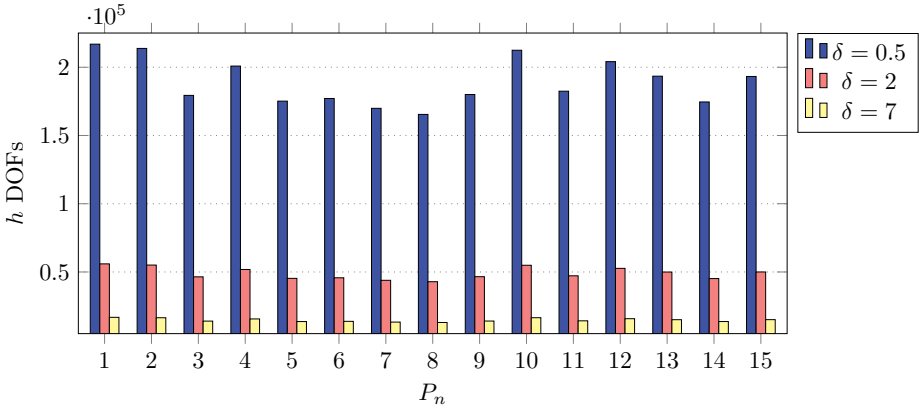

FIG. 7. *DFN1425: Number of total h-DOFs per process for a 15 Slave simulation.*

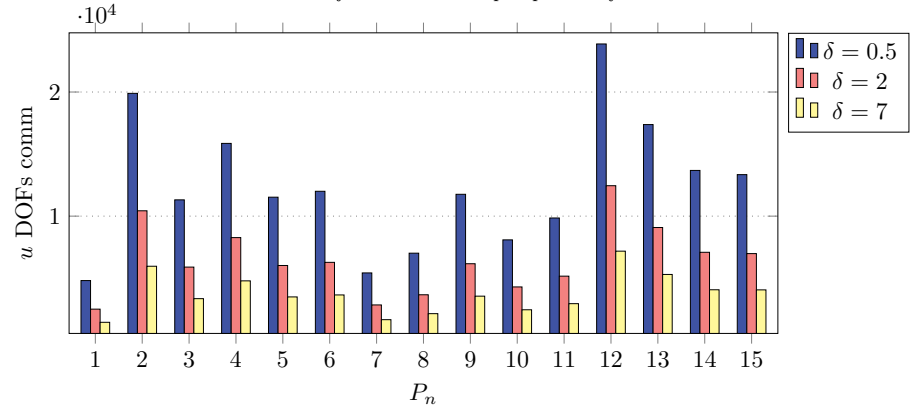

FIG. 8. *DFN1425: Number of communicated u-DOFs per process for a 15 Slave simulation.*

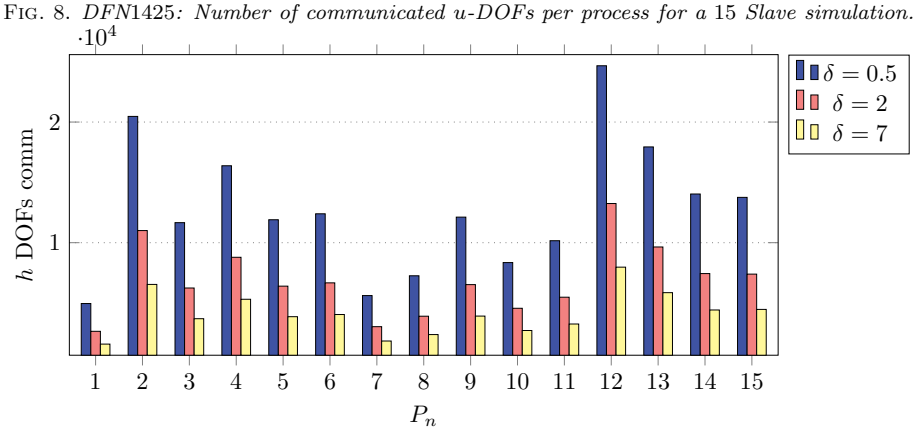

FIG. 9. *DFN1425: Number of communicated h-DOFs per process for a 15 Slave simulation.*

In the following, we focus on the smallest and the largest DFNs here considered, namely, DFN709 and DFN1425. Figures 6–9 give insight into the partitioning process performed by the library METIS, in the case of a 15-*Slave* run for the larger DFN. Figure 6 reports the number of fractures assigned to each *Slave* process and the number of communicating traces. The number of actual DOFs that each process has to manage is shown in Figures 7–9. Figure 7 reports the number of h-DOFs on the fractures handled by each process. Since the number of h-DOFs is related to

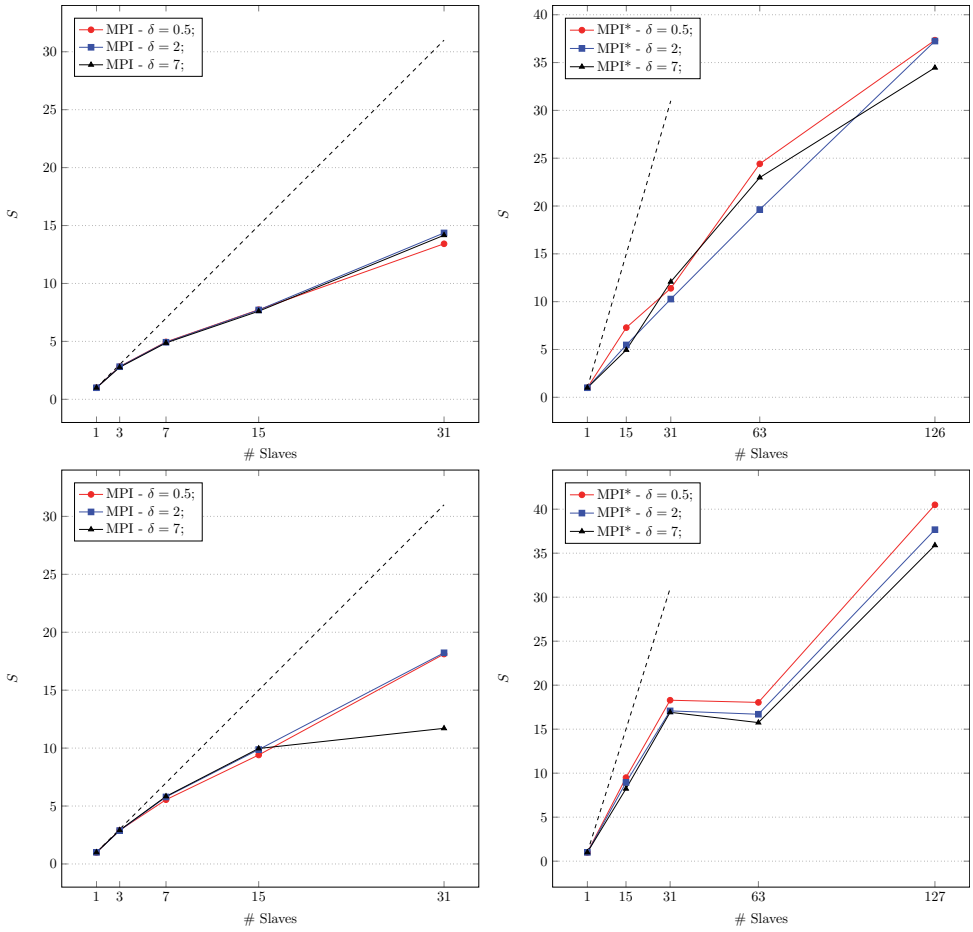

FIG. 10. *Scalability plot. Top: DFN*709*. Bottom: DFN*1425*. MPI (left) and MPI* (right).*

the size of the linear systems to be solved, the figure highlights that the workload in terms of computations is well balanced among the processes. Figures 8 and 9 show the number of h-DOFs and u-DOFs on the communicating traces, which are a subset of all the traces assigned to a process. This provides an indication of the weight of communications. It is worth remarking that, even if both the chosen DFN configurations have a high density of traces (see Table 1), the number of communicated DOFs is small compared to the total number of DOFs, with percentages of about 1% with a few *Slaves*, up to about 40% with 31 *Slaves* for the coarser grids. The percentage of DOFs involved in communications reduces when the grid is refined, whereas it increases when a larger number of *Slaves* is used in the simulations. The general overview is reported in Table 5.

The scalability performances of the algorithm on the selected DFNs are described in terms of the speedup ratio S_p and the efficiency E_p, which are defined as follows. Let t_1 denote the time required by the algorithm with a single *Slave* and t_p the time required with p *Slave* processes. The speedup S_p and efficiency E_p are defined,

TABLE 6
Time per iteration of the parallel CG algorithm and efficiency for DFN709 and DFN1425.

Version	Slaves	Time [*sec*] $\delta = 0.5$	Time [*sec*] $\delta = 2$	Time [*sec*] $\delta = 7$	Efficency % $\delta = 0.5$	Efficency % $\delta = 2$	Efficency % $\delta = 7$
DFN709							
Serial	1	0.83754	0.23432	0.08039			
MPI	3	0.29631	0.08324	0.02909	94.22	93.83	92.11
	7	0.16898	0.04758	0.01651	70.81	70.35	69.55
	15	0.10830	0.03038	0.01057	51.56	51.41	50.69
	31	0.06239	0.01632	0.00568	43.31	46.33	45.68
MPI*	15	0.11496	0.04281	0.01624	48.57	36.49	33.00
	31	0.07350	0.02281	0.00666	36.76	33.13	38.96
	63	0.03431	0.01194	0.00350	38.74	31.16	36.47
	126	0.02243	0.00629	0.00233	29.64	29.56	27.35
DFN1425							
Serial	1	2.67725	0.70857	0.24254			
MPI	3	0.91974	0.24591	0.08328	97.03	96.05	97.08
	7	0.48278	0.12251	0.04159	79.22	82.63	83.30
	15	0.28466	0.07176	0.02434	62.70	65.82	66.44
	31	0.14775	0.03888	0.02072	58.45	58.79	37.76
MPI*	15	0.28194	0.07920	0.02947	63.31	59.64	54.87
	31	0.14638	0.04149	0.01434	59.00	55.09	54.57
	63	0.14838	0.04244	0.01540	28.64	26.50	25.00
	127	0.06611	0.01881	0.00676	31.89	29.66	28.26

respectively, as

$$S_p = \frac{t_1}{t_p}, \qquad E_p = \frac{S_p}{p}.$$

In Figure 10 we report the scalability plot for MPI (left) and MPI* (right) runs, displaying the speedup ratio S_p. For the smaller DFN, and for the larger DFN with a few *Slave* processes, the scalability performances are quite similar for all the grids considered, whereas, for the larger DFN with the higher number of parallel processes, the performances slightly increase as the grid parameter reduces. This is an expected behavior, since on the finer grids the number of total DOFs increases, thus increasing the cost of computations, whereas the percentage of DOFs involved in communications reduces. The efficiency E_p and the time required to perform an iteration of the algorithm are reported in Table 6. The values reported show that, as a whole, acceptable efficiencies are achieved by the algorithm, and, for the higher number of *Slave* processes tested (127 *Slaves*), it is still possible to obtain an efficiency of about 30%. Efficiency values are larger for the larger DFN, as expected, since for more complex problems the weight of communications is counterbalanced by the cost of the required computations. Further, the communication algorithm designed is aimed at shadowing both the communication time and the time required by the *Master* process to arrange and send data to the *Slave* processes. Recalling the structure of the *Master-Slave* communication algorithm, each *Slave* process first solves the linear systems on the communicating fractures, performs nonblocking send operations, and then solves the linear systems on the remaining noncommunicating fractures. Only after these operations are completed are the data from the *Master* received. As shown by the

reported data, this communication algorithm allows us to improve the efficiency of the parallel algorithm with larger DFN configurations. In this case processes have to manage a larger chunk of fractures, and only some of them are communicating. In such a way the range of the higher efficiency values can be extended to an increasing number of parallel processes for larger DFN configurations. Simulations with 15 and 31 *Slave* processes are repeated for both MPI and MPI* configurations. It is possible to see that differences are in general almost negligible and are probably due to simulation time discrepancies for different loading statuses of the cluster.

5. Conclusions and future extensions. The parallel performances of an implementation of the optimization based method presented in [3, 4, 5] have been shown, applied to quite complex DFNs. The considered DFNs are quite challenging as they are affected by a very large heterogeneity in fracture dimensions, trace lengths, trace distances, and trace angles. In spite of these geometrical complexities, the method can perform simulations without requiring any modification of the geometry or introducing a huge number of elements for meshing purposes.

Parallel efficiencies of over 50% are achieved with up to 16 parallel processes, and those of about 30% are obtained with up to 128 parallel processes. The efficiency of the parallel algorithm improves as the size of the DFN increases; hence parallel computing is worthwhile when dealing with huge DFNs.

A possible extension to a 2-level *Master* configuration with several *Master* processes of level 1 managing groups of *Slave* processes and a *Master* process of level 0 managing the communications between the level 1 *Masters* can efficiently extend the approach to large scale DFNs with a number of fractures on the order of $10^4 - 10^5$, preserving the scalability performances of the current parallel implementation on parallel supercomputers with hundreds of nodes. An MPI-OpenMP hybrid implementation is currently under development, in order to split in-node computations over several parallel threads and further reduce the number of MPI communications.

Finally, preconditioning is crucial in order to further improve the overall performances of the algorithm. A preconditioner defined independently on each fracture of the DFN is currently under development with promising results, and two or multiple level preconditioners are under investigation.

REFERENCES

[1] M. F. Benedetto, S. Berrone, S. Pieraccini, and S. Scialò, *The virtual element method for discrete fracture network simulations*, Comput. Methods Appl. Mech. Engrg., 280 (2014), pp. 135–156.

[2] S. Berrone, C. Canuto, S. Pieraccini, and S. Scialò, *Uncertainty quantification in discrete fracture network models: Stochastic fracture transmissivity*, Politecnico di Torino, http://porto.polito.it/id/eprint/2564948, 2014.

[3] S. Berrone, S. Pieraccini, and S. Scialò, *A PDE-constrained optimization formulation for discrete fracture network flows*, SIAM J. Sci. Comput., 35 (2013), pp. B487–B510.

[4] S. Berrone, S. Pieraccini, and S. Scialò, *On simulations of discrete fracture network flows with an optimization-based extended finite element method*, SIAM J. Sci. Comput., 35 (2013), pp. A908–A935.

[5] S. Berrone, S. Pieraccini, and S. Scialò, *An optimization approach for large scale simulations of discrete fracture network flows*, J. Comput. Phys., 256 (2014), pp. 838–853.

[6] D. Billaux, J.P. Chiles, K. Hestir, and J. Long, *Three-dimensional statistical modelling of a fractured rock mass: An example from the Fanay-Augéres mine*, Internat. J. Rock Mech. Mining Sci. Geomech. Abstracts, 26 (1989), pp. 281–299.

[7] E. Bonnet, O. Bour, N. E. Odling, P. Davy, I. Main, P. Cowie, and B. Berkowitz, *Scaling of fracture systems in geological media*, Rev. Geophys., 39 (2001), pp. 347–383.

[8] M. C. Cacas, E. Ledoux, G. de Marsily, B. Tillie, A. Barbreau, E. Durand, B. Feuga, and P. Peaudecerf, *Modeling fracture flow with a stochastic discrete fracture network: Calibration and validation:* 1. *The flow model*, Water Resour. Res., 26 (1990), pp. 479–489.
[9] G. Cammarata, C. Fidelibus, M. Cravero, and G. Barla, *The hydro-mechanically coupled response of rock fractures*, Rock Mech. Rock Engrg., 40 (2007), pp. 41–61.
[10] P. Davy, O. Bour, J.-R. De Dreuzy, and C. Darcel, *Flow in multiscale fractal fracture networks*, Geol. Soc. London Special Publications, 261 (2006), pp. 31–45.
[11] G. de Marsily, F. Delay, J. Gonalvés, Ph. Renard, V. Teles, and S. Violette, *Dealing with spatial heterogeneity*, Hydrogeology J., 13 (2005), pp. 161–183.
[12] W. S. Dershowitz and H. H. Einstein, *Characterizing rock joint geometry with joint system models*, Rock Mech. Rock Engrg., 1 (1988), pp. 21–51.
[13] W. S. Dershowitz and C. Fidelibus, *Derivation of equivalent pipe networks analogues for three-dimensional discrete fracture networks by the boundary element method*, Water Resour. Res., 35 (1999), pp. 2685–2691.
[14] C. R. Dohrmann, *An approximate BDDC preconditioner*, Numer. Linear Algebra Appl., 14 (2007), pp. 149–168.
[15] B. Dverstorp and J. Andersson, *Application of the discrete fracture network concept with field data: Possibilities of model calibration and validation*, Water Resour. Res., 25 (1989), pp. 540–550.
[16] Eigen, *Eigen documentation*, http://eigen.tuxfamily.org/dox/, 2014.
[17] J. Erhel, J.-R. de Dreuzy, and B. Poirriez, *Flow simulation in three-dimensional discrete fracture networks*, SIAM J. Sci. Comput., 31 (2009), pp. 2688–2705.
[18] C. Farhat, M. Lesoinne, P. LeTallec, K. Pierson, and D. Rixen, *FETI-DP: A dual-primal unified FETI method.* I. *A faster alternative to the two-level FETI method*, Internat. J. Numer. Methods Engrg., 50 (2001), pp. 1523–1544.
[19] L. Formaggia, A. Fumagalli, A. Scotti, and P. Ruffo, *A reduced model for Darcy's problem in networks of fractures*, ESAIM Math. Model. Numer. Anal., 48 (2014), pp. 1089–1116.
[20] B. Gylling, L. Birgersson, L. Moreno, and I. Neretnieks, *Analysis of a long-term pumping and tracer test using the channel network model*, J. Contaminant Hydrology, 32 (1998), pp. 203–222.
[21] P. A. Hsieh, *Scale effects in fluid flow through fractured geologic media*, in Scale Dependence and Scale Invariance in Hydrology, G. Sposito, ed., Cambridge University Press, Cambridge, UK, 1998, pp. 335–353.
[22] T. Kalbacher, R. Mettier, C. McDermott, W. Wang, G. Kosakowski, T. Taniguchi, and O. Kolditz, *Geometric modelling and object-oriented software concepts applied to a heterogeneous fractured network from the Grimsel rock laboratory*, Comput. Geosci., 11 (2007), pp. 9–26.
[23] G. Karypis, *METIS software*, http://glaros.dtc.umn.edu/gkhome/metis/metis/overview, 2013.
[24] G. Karypis and V. Kumar, *A fast and high quality multilevel scheme for partitioning irregular graphs*, SIAM J. Sci. Comput., 20 (1998), pp. 359–392.
[25] V. Lenti and C. Fidelibus, *A BEM solution of steady-state flow problems in discrete fracture networks with minimization of core storage*, Comput. Geosci., 29 (2003), pp. 1183–1190.
[26] J. C. S. Long, P. Gilmour, and P. A. Witherspoon, *A model for steady fluid flow in random three-dimensional networks of disc-shaped fractures*, Water Resour. Res., 21 (1985), pp. 1105–1115.
[27] J. Mandel, *Balancing domain decomposition*, Comm. Numer. Methods Engrg., 9 (1993), pp. 233–241.
[28] J. Mandel and M. Brezina, *Balancing domain decomposition for problems with large jumps in coefficients*, Math. Comp., 65 (1996), pp. 1387–1401.
[29] J. Mandel and C. R. Dohrmann, *Convergence of a balancing domain decomposition by constraints and energy minimization*, Numer. Linear Algebra Appl., 10 (2003), pp. 639–659.
[30] H. Mustapha and K. Mustapha, *A new approach to simulating flow in discrete fracture networks with an optimized mesh*, SIAM J. Sci. Comput., 29 (2007), pp. 1439–1459.
[31] B. Nœtinger and N. Jarrige, *A quasi steady state method for solving transient Darcy flow in complex 3D fractured networks*, J. Comput. Phys., 231 (2012), pp. 23–38.
[32] A. W. Nordqvist, Y. W. Tsang, C. F. Tsang, B. Dverstop, and J. Andersson, *A variable aperture fracture network model for flow and transport in fractured rocks*, Water Resour. Res., 28 (1992), pp. 1703–1713.

[33] P. Panfili and A. Cominelli, *Simulation of miscible gas injection in a fractured carbonate reservoir using an embedded discrete fracture model*, in Proceedings of the Abu Dhabi International Petroleum Exhibition and Conference, SPE International, Richardson, TX, 2014, SPE-171830-MS.

[34] G. Pichot, J. Erhel, and J. de Dreuzy, *A mixed hybrid mortar method for solving flow in discrete fracture networks*, Appl. Anal., 89 (2010), pp. 1629–643.

[35] G. Pichot, J. Erhel, and J. de Dreuzy, *A generalized mixed hybrid mortar method for solving flow in stochastic discrete fracture networks*, SIAM J. Sci. Comput., 34 (2012), pp. B86–B105.

[36] G. Pichot, B. Poirriez, J. Erhel, and J.-R. de Dreuzy, *A mortar BDD method for solving flow in stochastic discrete fracture networks*, in Domain Decomposition Methods in Science and Engineering XXI, Lect. Notes Comput. Sci. Eng., Springer, Berlin, 2014, pp. 99–112.

[37] A. Quarteroni and A. Valli, *Domain Decomposition Methods for Partial Differential Equations*, Numer. Math. Sci.c Comput., Clarendon Press, Oxford, UK, 1999.

[38] A. Toselli and O.B. Widlund, *Domain Decomposition Methods—Algorithms and Theory*, Springer Ser. Comput. Math. 34, Springer, Berlin, 2005.

[39] M. Verscheure, A. Fourno, and J.-P. Chilès, *Joint inversion of fracture model properties for CO_2 storage monitoring or oil recovery history matching*, Oil Gas Sci. Tech. Rev. IFP Energies Nouvelles, 67 (2012), pp. 221–235.

[40] M. Vohralík, J. Maryška, and O. Severýn, *Mixed and nonconforming finite element methods on a system of polygons*, Appl. Numer. Math., 51 (2007), pp. 176–193.

[41] C. Xu and P. Dowd, *A new computer code for discrete fracture network modelling*, Computers Geosci., 36 (2010), pp. 292–301.