[Article] A parallel solver for large scale DFN flow simulations

Original Citation:
S. Berrone, S. Pieraccini, S. Scialo', F. Vicini A parallel solver for large scale DFN flow simulations. In: SIAM JOURNAL ON SCIENTIFIC COMPUTING. - ISSN 1064-8275
(In Press)
Availability:
This version is available at: http://porto.polito.it/2561152/ since: August 2014
Publisher:
SIAM - Society for Industrial and Applied Mathematics
Terms of use:
This article is made available under terms and conditions applicable to Open Access Policy Article ("Public - All rights reserved") , as described at http://porto.polito.it/terms_and_conditions.html

Porto, the institutional repository of the Politecnico di Torino, is provided by the University Library and the IT-Services. The aim is to enable open access to all the world. Please share with us how this access benefits you. Your story matters.

Publisher copyright claim:
This is the author's post-print version of an article published on SIAM JOURNAL ON SCIENTIFIC COMPUTING, Publisher [pin missing: publisher], Vol UNSPECIFIED , Number UNSPECIFIED Year UNSPECIFIED (ISSN [pin missing: issn]) The present version is accessible on PORTO, the Open Access Repository of the Politecnico of Torino

(Article begins on next page)
A PARALLEL SOLVER FOR LARGE SCALE DFN FLOW SIMULATIONS

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

March 9, 2015

Abstract. Flows in fractured media have been modeled with 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 [3, 4, 5] for dealing with arbitrary Discrete Fracture Networks on non conforming 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, that would result very difficult to mesh with conventional approaches relying on some kind of mesh conformity at the interfaces.

Key words. MPI, Parallel scalability, Discrete Fracture Networks, Single-phase flows, PDE constrained optimization

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

1. Introduction. The simulation of flows in fractured media is a challenging issue relevant in several critical applications: Oil&Gas enhanced production, nuclear waste geological storage, carbon dioxide geological storage, geothermal applications, 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 can not 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 behaviour and hydraulic and mechanical interactions [10, 11, 39, 9, 33].

Concerning numerical simulations on DFNs, several discretization approaches, such as standard FEM, mixed hybrid discretizations, eXtended Finite Elements [34, 35, 40, 19], are usually applied. One of the major 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 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 Mor...
tar method [34, 35], which allows to relax the conformity constraints. A different approach consists in modelling the DFN as a distribution of 1D pipes mutually connecting the centres of fracture intersections [20, 13, 24]. The resulting mesh of pipes should model the DFN topology and hydraulic properties via a suitable definition of hydro-geological 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 [25, 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 of 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 size DFNs [1].

Efficient solvers for DFN flow simulations are of crucial importance on several respects. They are clearly needed for very large scale simulations at basin scale, where the number of fractures involved may realistically count even $10^6$ fractures. Besides efficiency, also 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 two-fold. A parallel implementation of the method is described, and its scalability is analysed 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 only occurs 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 the sequel, we briefly describe the model and its discretization, referring the reader to [3, 5] for full details.

Let $\Omega$ denote a DFN in the 3D space, composed of connected open planar fractures $F_i$, for $i = 1, \ldots, J$. Intersections between fractures, called traces, are denoted by $S_m$, with $m = 1, \ldots, M$. Each trace is given by the intersection of exactly two fractures, such that $S_m = F_i \cap F_j$ with $i \neq j$ collected in the set $I_{S_m}$. The set of traces belonging to fracture $F_i$ is denoted by $S_i$.

The boundary of $\Omega$, $\partial \Omega$, is divided in a Dirichlet part $\Gamma_D$ and a Neumann part
\( \Gamma_N \), 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 \( \partial F_i \) and is also split in a Dirichlet part \( \Gamma_D = \partial F_i \cap \Gamma_D \) and a Neumann part \( \Gamma_N = \partial F_i \cap \Gamma_N \). Functions \( b^D \) and \( b^N \) prescribe the Dirichlet and Neumann boundary conditions, and their restrictions to the boundary \( \partial F_i \) are indicated by \( b^D_i \) and \( b^N_i \), respectively. We remark that while \( \Gamma_D \) should be non-empty, \( \Gamma_iD \) can be empty for some \( i \) (see [5]).

The hydraulic head in \( \Omega \) 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 \( V_i = H^1_0(F_i) = \{ v \in H^1(F_i) : v|_{\Gamma_{\partial F_i}} = 0 \} \) and \( V^D_i = H^1_0(F_i) = \{ v \in H^1(F_i) : v|_{\Gamma_{\partial F_i}} = b^D_i \} \), such that, on each fracture, the solution \( H_i \) belongs to \( V^D_i \). Let \( K_i \) denote the fracture transmissivity tensor on \( F_i \). On each trace \( S_m \), for each fracture \( F_i \), \[ \left[ \partial H_i \over \partial v_{S_m}^i \right]_{S_m} \] denotes the jump of the co-normal derivative along the unit vector normal to \( S_m \) on \( F_i \).

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

\[
\begin{align*}
\int_{F_i} K_i \nabla H_i \nabla v_{\Omega} \, d\Omega &= \int_{F_i} q_i v_{\Omega} \, d\Omega + \int_{\Gamma_{\partial F_i}^N} b_i^N v_{\Gamma_{\partial F_i}^N} \, d\gamma \\
+ \sum_{m=1}^{M} \int_{S_m} \left[ \frac{\partial H_i}{\partial v_{S_m}^i} \right]_{S_m} v_{|S_m} \, d\gamma, & \quad \forall v \in V_i, \forall i = 1, \ldots, I \quad (2.1) \\
H_i|_{S_m} - H_{\partial F_i|S_m} = 0, & \quad \text{for } i, j \in I_{S_m}, \forall m = 1, \ldots, M, \quad (2.2) \\
\left[ \frac{\partial H_i}{\partial v_{S_m}^i} \right]_{S_m} + \left[ \frac{\partial H_j}{\partial v_{S_m}^j} \right]_{S_m} = 0, & \quad \text{for } i, j \in I_{S_m}, \forall m = 1, \ldots, M, \quad (2.3)
\end{align*}
\]

where \( q_i \in L^2(F_i) \) 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 Equation (2.1) accounts for the flux incoming in \( F_i \) through its traces.

In order to decouple the resolution of Equations (2.1)-(2.3), the problem has been reformulated in [3] as a minimization problem, in which the exact fulfillment of Equations (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 \( U^m_i \in H^{-\frac{1}{2}}(S_m) \) as:

\[
U^m_i = \left[ \frac{\partial H_i}{\partial v_{S_m}^i} \right]_{S_m} + \alpha H_{\partial F_i|S_m},
\]

where \( \alpha \) is a positive constant. We define the cost functional \( J \) as

\[
J(H, U) = \sum_{m=1}^{M} \left( \| H_{|S_m} - H_{\partial F_i|S_m} \|_{H^1(S_m)}^2 + \| U^m_i + U^m_{\partial S_m} - \alpha (H_{|S_m} + H_{\partial F_i|S_m}) \|_{H^{-\frac{1}{2}}(S_m)}^2 \right),
\]

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

\[
\begin{align*}
\int_{F_i} K_i \nabla H_i \nabla v_{\Omega} \, d\Omega + \alpha \sum_{m=1}^{M} \int_{S_m} H_{|S_m} v_{|S_m} \, d\gamma &= \int_{F_i} q_i v_{\Omega} \, d\Omega + \int_{\Gamma_{\partial F_i}^N} b_i^N v_{\Gamma_{\partial F_i}^N} \, d\gamma + \sum_{m=1}^{M} \int_{S_m} U^m_i v_{|S_m} \, d\gamma \\
&= \int_{F_i} q_i v_{\Omega} \, d\Omega + \int_{\Gamma_{\partial F_i}^N} b_i^N v_{\Gamma_{\partial F_i}^N} \, d\gamma + \sum_{m=1}^{M} \int_{S_m} U^m_i v_{|S_m} \, d\gamma \quad (2.5)
\end{align*}
\]
∀v ∈ V, ∀i = 1, . . . , I, coincides with the solution of the system (2.1)-(2.3), [3, 5]. Thus, the quantities $U^m$ 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 fracture. The discretization grid can be independent of the meshes on the fractures.

In the sequel, a small 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 $u^m_i$. Let us introduce on $F_i$ the set $\{\varphi_{i,k}\}_{k=1}^{N_i}$ of standard linear finite element basis functions on triangular elements, and on $S_m$ the set $\{\psi_{i,k}^m\}_{k=1}^{N^m}$ of linear finite element basis functions in 1D. Hence, the quantities $N_i$ and $N^m_i$ denote the number of degrees of freedom (DOFs) for $h_i$ and $u^m_i$, respectively. Hence we write $h_i = \sum_{k=1}^{N_i} h_{i,k} \varphi_{i,k}$, $u^m_i = \sum_{k=1}^{N^m_i} u_{i,k}^m \psi_{i,k}^m$. 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 $u^m_i$ to denote the $N^m_i$-dimensional vector of DOFs for the corresponding control variable. The difference will be always clear from the context. Furthermore, we introduce the following vectors. We set $h = (h_1^T, \ldots, h_I^T)^T \in \mathbb{R}^{N_F}$, $N_F = \sum_{i=1}^I N_i$; for each fracture $F_i$ we define $u_i \in \mathbb{R}^{N_S_i}$, $N_S_i = \sum_{m \in S_i} N^m_i$, as the vector obtained collecting column-wise the DOFs of the control variables $u^m_i$ on all traces $S_m \in S_i$, and $u \in \mathbb{R}^{N_T}$, $N_T = \sum_{i=1}^I N_{S_i}$, 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:

$$A_i h_i = q_i + B_i u_i,$$

where $q_i \in \mathbb{R}^{N_i}$ 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 $B_i \in \mathbb{R}^{N_i \times N_S_i}$ collects the integrals of the basis functions $\varphi_{i,k}$, $k = 1, \ldots, N_i$, against the basis functions $\{\psi_{i,k}^m\}$, $k = 1, \ldots, N^m_i$, $S_m \in S_i$. It is worthwhile noting that, assuming $u_i$ is known, Equation (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 Equation (2.4), considering $L^2(S_m)$ norms, the following expression for the discrete functional is derived:

$$J = \frac{1}{2} \sum_{i=1}^I \sum_{S_m \in S_i} \left( \int_S \sum_{k=1}^{N_i} h_{i,k} \varphi_{i,k} |_{S_m} - \sum_{k=1}^{N_i} h_{j,k} \varphi_{j,k} |_{S_m} \right)^2 d\gamma$$

$$+ \sum_{S_m \in S_i} \sum_{k=1}^{N^m_i} u_{i,k}^m \psi_{i,k}^m - \sum_{k=1}^{N^m_i} u_{j,k}^m \psi_{j,k}^m - \alpha \sum_{k=1}^{N_i} h_{i,k} \varphi_{i,k} |_{S_m} - \alpha \sum_{k=1}^{N_i} h_{j,k} \varphi_{j,k} |_{S_m} \right)^2 d\gamma \right).$$

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 $S_m \in S_i$; thus,

$$h_i^+ \in \mathbb{R}^{N_i^+}, \quad N_i^+ = N_i + \sum_{j \in I_{S_m}, j \neq i, S_m \in S_i} N_j.$$
Similarly, we define the vector \( u^+_i \in \mathbb{R}^{N_S^i} \) by collecting all vectors \( u^+_i \) with \( i, j \in I_{S_m} \), for all \( S_m \in S_i \) (thus, \( N_S^i = \sum_{k \in I_{S_m}, S_m \in S_i} N^+_K \)). Rearranging the various integrals in (2.7) into matrices, the discrete functional is written in algebraic form as:

\[
B \in \mathbb{R}^{N_S \times N_S}, \quad \text{treated in a fracture-oriented way. We also highlight that the global matrices}
\]

\[
\text{written in order to highlight the structure of the problem. The problem will be actually}
\]

\[
\text{an equality constrained quadratic programming problem:}
\]

\[
\min \quad J(h, u) := \frac{1}{2} (h^T G^h h - \alpha h^T B^h u - \alpha u^T B^u h + u^T G^u u)
\]

\[
s.t. \quad Ah - Bu = q,
\]

where matrices \( G^h \in \mathbb{R}^{N_T \times N_T} \) collect the integrals involving only basis functions for the hydraulic head; matrices \( G^u \in \mathbb{R}^{N_S \times N_S} \) collect the integrals involving only basis functions for the control variables; matrices \( B^h \in \mathbb{R}^{N_S \times N_T} \) and \( B^u \in \mathbb{R}^{N_S \times N_S} \) collect the integrals involving basis functions both for the hydraulic head and for the control variables.

Grouping Equations (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:

\[
\min \quad J(h, u) := \frac{1}{2} (h^T G^h h - \alpha h^T B^h u - \alpha u^T B^u h + u^T G^u u)
\]

\[
s.t. \quad Ah - Bu = q,
\]

where \( G^h \in \mathbb{R}^{N_T \times N_T}, G^u \in \mathbb{R}^{N_S \times N_S}, A \in \mathbb{R}^{N_T \times N_T}, B^h \in \mathbb{R}^{N_S \times N_T}, B^u \in \mathbb{R}^{N_S \times N_S}, B \in \mathbb{R}^{N_T \times N_T} \) 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 \( B^h_i \) and \( B^u_i \).

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

\[
\min \quad \tilde{J}(u) := \frac{1}{2} u^T (B^T A^{-T} G^h A^{-1} B + G^u - \alpha B^T A^{-T} B - \alpha B^T A^{-1} B) u
\]

\[
+ q^T A^{-T} (G^h A^{-1} B - \alpha B) u.
\]

We may compactly write the functional \( \tilde{J}(u) \) as \( \tilde{J}(u) = \frac{1}{2} u^T \tilde{G} u + q^T u \). The matrix \( \tilde{G} \) 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 \( \tilde{J} \). This is given by:

\[
\nabla \tilde{J}(u) = (B^T A^{-T} G^h A^{-1} B + G^u - \alpha (B^T A^{-T} B + B^T A^{-1} B) u
\]

\[
+ (B^T A^{-T} G^h - \alpha B^T) A^{-1} q = \tilde{G} u + \tilde{q}.
\]

Our method can be written as a parallel version of CG method applied to the functional \( \tilde{J} \), or, equivalently, to the linear system \( \tilde{G} u + \tilde{q} = 0 \), without explicitly assembling
neither the matrix $\hat{G}$, nor 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 here adopted, 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 $A^{-1}(Bu + q) = h$, we have

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

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

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

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

$$ Ah = Bu + q, \quad A^T p = G h - \alpha Bu \quad (2.13) $$

are solved. These linear systems, which are formally written in (2.13) on the whole DFN, can be split in $I$ 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) corresponds to: for $i = 1, \ldots, I$

$$ A_i h_i = \tilde{q}_i + B_i u_i \quad (2.14) $$

$$ A_i^T p_i = G^h h_i^+ - \alpha B^h u_i^+ \quad (2.15) $$

$$ (\nabla \hat{J})^i = B_i^T p_i + G^u u_i^+ - \alpha B^u h_i^+. \quad (2.16) $$

As shown in Equations (2.14)-(2.16), the gradient $\nabla \hat{J}(u)$ can therefore be computed as a contribute from the various fractures in the DFN. The evaluation of the contribute $\nabla \hat{J}_i$ only requires, as additional information besides those 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 Equation (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 in subproblems defined on subdomains is in common with classical Domain Decomposition (DD) approaches [37, 38], already applied to DFN flow simulations in [36]. Although there is a similarity between the approach here depicted 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 $\Omega$, given by the sum of a number of contribution relative to subdomains $\Omega_i$. 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 details 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 $\nabla J(u)$ at iteration $k$,
often obtained by grouping column-wise vectors $g_i^k := (\nabla J)_i$, $\forall i=1,\ldots,I$.

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

The serial algorithm is detailed in the following. The direction of movement for
$u$ is denoted $d^k$. Vectors $d_i^k$ and $(d_i^k)^T$ are derived from $d^k$ following the same rules
used for deriving $u_i^k$ and $(u_i^k)^T$ 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 $\beta_{i,D}^k = (g_i^k)^T g_i^k$, $\forall i=1,\ldots,I$
4. Set $d_i^k = -g_i^k \forall i=1,\ldots,I$
5. While $g^k \neq 0$
   5.1. Compute $\delta g^k_i$:
      i. Solve $A_i \delta h_i^k = B_i d_i^k$
      ii. Solve $A_i^T \delta p_i^k = G_i^T \delta(h_i^+)^k - \alpha B_i^u (d_i^k)^k$
      iii. Compute $\delta g_i^k = B_i^T \delta p_i^k + G_i^u (d_i^k)^k - \alpha B_i^u \delta(h_i^+)^k$
   5.2. Compute $\lambda^k$ 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 $\beta_{i,D}^{k+1} = \beta_{i,D}^k$ and compute $\beta_{i,N}^{k+1} = (g_i^{k+1})^T g_i^{k+1}$, $\forall i=1,\ldots,I$
   5.6. Compute $\beta^{k+1} = \frac{\sum_{i=1}^I \beta_{i,D}^{k+1}}{\sum_{i=1}^I \beta_{i,N}^{k+1}}$
   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 requiring the same quantity $\delta g_i^k$ computed at step 5.1. 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, over-refinement 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 MPI protocol. Let $N_p$ be the number of concurrent processes $P_n$, $n=0,\ldots,N_p-1$. 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 $\delta h_i^k$ 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 $u_0^k$ 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 $\delta g_i^k$

      i. Solve $A_i \delta h_i^k = B_i d_i^k \forall i$ s.t. $F_i \in \mathcal{F}_n$. Communication phase: $\delta h_i^k$

      ii. Solve $A_i^T \delta p_i^k = G_i \delta (h_i^+)^k - \alpha B_i^k (d_i^+)^k, \forall i$ s.t. $F_i \in \mathcal{F}_n$

      iii. Compute $\delta g_i^k = B_i^T \delta p_i^k + G_i^T (d_i^+)^k - \alpha B_i^k \delta (h_i^+)^k, \forall i$ 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 $\lambda^k$ through MPI

   Reduce operation

   7.4. Update $u_{i+1}^k = u_i^k + \lambda^k d_i^k, \forall i$ s.t. $F_i \in \mathcal{F}_n$

   7.5. Update $g_{i+1}^k = g_i^k + \lambda^k \delta g_i^k, \forall i$ s.t. $F_i \in \mathcal{F}_n$

   7.6. Set $\beta_{i,D}^{k+1} = \beta_{i,N}^{k+1}$ and compute $\beta_{i,N}^{k+1} = (g_i^{k+1})^T g_i^{k+1}, \forall i$ s.t. $F_i \in \mathcal{F}_n$

   7.7. Communication of $\beta_{i,N}^k$ and $\beta_{i,D}^k$ and computation of $\beta^{k+1}$ through MPI

   Reduce operation

   7.8. Update $d_{i+1}^k = -g_{i+1}^k + \beta^{k+1} d_i^k, \forall i$ 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 [23, 29]. 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 amount of communications, recalling that only trace-related data need to 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 $\mathcal{F}_n^{\text{nc}}$ be the subset of the non-communicating fractures in $\mathcal{F}_n$, with cardinality $I_n^{\text{nc}}$.
and let $\mathcal{F}_n^c$ 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 $\mathcal{S}_n^c$ 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 $\mathcal{F}_n^c$. Let $I_n^c$ be the cardinality of $\mathcal{F}_n^c$ and $I_{P_n}$ 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 $d_k^n$ for each trace $S \in \mathcal{S}_n^c$ and the subset of DOFs of $\delta h_k^n$ that affect the value of $\delta h_k^n$ itself on the traces in $\mathcal{S}_n^c$. Similarly, $P_n$ needs to receive information related to the traces in $\mathcal{S}_n^c$ 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 in $N_p-1$ 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 in a unique array and sends it to the Master process. The Master process, while receiving information, re-arranges the data in $N_p-1$ 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 Master of Masters process, in charge for 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 each process $P_n$, $n = 0, \ldots, N_p - 1$ sends to and receives from all the processes in $\mathcal{P}_n$ the required data. In this case, the number of communications is therefore proportional to $\sum_{n=0}^{N_p-1} I_{P_n}$, 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 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 fractures, as it is limited to few DOFs related to the traces in the set $\mathcal{S}_n^c$.

The 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 $\delta g_i^k$:

*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 $P_n \in I_{P_m}$, 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 non-blocking send of $v_n$ to Process $P_n$

if Process $P_n$, $n \neq 0$:

i. Solve $A_\delta h_k^i = B_\delta d_k^i$, $\forall i \in \mathcal{F}_n^c$

ii. Extract DOFs of $\delta h_k^i$ related to the traces in $\mathcal{S}_n^c$ and collect them in an array $h_c$

iii. Perform an MPI non-blocking send of $\delta h_c$ to Process $P_0$

iv. Solve $A_\delta h_k^i = B_\delta d_k^i$, $\forall i \in \mathcal{F}_n^c$

v. Receive information from Process $P_0$ and update vector $\delta(h^+)^h_i$

vi. Solve $A_\delta p_k^i = G_\delta(\delta h^{+})^k_i - \alpha B_\delta d_k^i$, $\forall i \in \mathcal{F}_n$

vii. Compute $\delta g_k^i = B_\delta G_\delta(\delta h^{+})^k_i + G_\delta(\delta d^{+})^k_i - \alpha B_\delta(\delta h^{+})^k_i$, $\forall i \in \mathcal{F}_n$

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 4.1 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 4.1-4.3. Namely, in Figure 4.1 a 3D plot of the 709 and 1425 fracture DFNs is reported. The colouring of fractures in these images is proportional to fracture areas: the fracture areas span several orders of magnitude, ranging from 376 m$^2$ to $10^4$ m$^2$ in the smaller DFN, and from 3.2 m$^2$ to $10^4$ m$^2$ in the larger DFN. Figures 4.2-4.3 report histograms of trace related data: Figure 4.2 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>
<thead>
<tr>
<th>Table 4.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFN related data</td>
</tr>
<tr>
<td>-------------</td>
</tr>
<tr>
<td>Traces</td>
</tr>
<tr>
<td>$I$</td>
</tr>
<tr>
<td>DFN709</td>
</tr>
<tr>
<td>DFN878</td>
</tr>
<tr>
<td>DFN915</td>
</tr>
<tr>
<td>DFN1425</td>
</tr>
</tbody>
</table>

In Figure 4.3 we report the distribution of trace lengths (left) and of distance between couples of non-intersecting 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 form DFN1425: they have been obtained by removing some selected fractures, in such a way that the
new networks do not present some geometrical features known in literature to be critical, i.e., narrow angles between intersecting traces and small distance of non-intersecting traces in the same fracture. More precisely, referring to Table 4.1 for the names: DFN878 does not contain non-intersecting 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 degrees of freedom.

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: a strong multiscale nature, and the presence of intricate networks of traces intersecting with narrow angles, very difficult to mesh, and 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, without extremely narrow angles and/or very close traces. As an example of the ability of the proposed method in dealing with complex networks, a detail of a non-conforming mesh is reported in Figure 4.4 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 $\delta$ represents the up-

---

**Figure 4.1.** SD view of DFN709 (left) and DFN1425 (right).

**Figure 4.2.** DFN709 and DFN1425. Left: distribution of angles between pairs of intersecting traces in the same fracture; right: zoom of lower values.
per bound for the area of triangular elements produced by the triangulator on each fracture. 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 also other node distributions can be indifferently 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 degrees of freedom is reported in Table 4.2. All the simulations are performed using a constant unitary
Table 4.2
Number of degrees of freedom for the hydraulic head \( h \) and the control variable \( u \)

<table>
<thead>
<tr>
<th></th>
<th>( N^F )</th>
<th>( N^T )</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFN</td>
<td>( \delta = 0.5 )</td>
<td>( \delta = 2 )</td>
</tr>
<tr>
<td>709</td>
<td>1378796</td>
<td>356527</td>
</tr>
<tr>
<td>878</td>
<td>1753469</td>
<td>452856</td>
</tr>
<tr>
<td>915</td>
<td>1808368</td>
<td>467474</td>
</tr>
<tr>
<td>1425</td>
<td>2838981</td>
<td>733588</td>
</tr>
</tbody>
</table>

Table 4.3
Relative residuals and flux conservation error \( \Delta_{\text{cons}} \) through iterations

<table>
<thead>
<tr>
<th></th>
<th>( \delta = 0.5 )</th>
<th>( \delta = 2 )</th>
<th>( \delta = 7 )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( \delta = 0.5 )</td>
<td>( \delta = 2 )</td>
<td>( \delta = 7 )</td>
</tr>
<tr>
<td>DFN709</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1000</td>
<td>3.749e-4</td>
<td>4.826e-4</td>
<td>5.767e-4</td>
</tr>
<tr>
<td>10000</td>
<td>1.014e-5</td>
<td>1.244e-5</td>
<td>1.845e-5</td>
</tr>
<tr>
<td>20000</td>
<td>3.615e-6</td>
<td>4.715e-6</td>
<td>8.539e-6</td>
</tr>
<tr>
<td>DFN878</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1000</td>
<td>3.716e-4</td>
<td>3.826e-4</td>
<td>3.944e-4</td>
</tr>
<tr>
<td>10000</td>
<td>6.668e-6</td>
<td>9.271e-6</td>
<td>1.462e-5</td>
</tr>
<tr>
<td>20000</td>
<td>4.821e-6</td>
<td>3.902e-6</td>
<td>4.705e-6</td>
</tr>
<tr>
<td>DFN915</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1000</td>
<td>3.548e-4</td>
<td>3.840e-4</td>
<td>3.989e-4</td>
</tr>
<tr>
<td>10000</td>
<td>8.486e-6</td>
<td>9.471e-6</td>
<td>1.636e-5</td>
</tr>
<tr>
<td>20000</td>
<td>7.362e-6</td>
<td>5.896e-6</td>
<td>7.442e-6</td>
</tr>
<tr>
<td>DFN1425</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1000</td>
<td>4.022e-4</td>
<td>4.209e-4</td>
<td>4.424e-4</td>
</tr>
<tr>
<td>10000</td>
<td>7.286e-6</td>
<td>8.687e-6</td>
<td>1.618e-5</td>
</tr>
<tr>
<td>20000</td>
<td>2.527e-6</td>
<td>2.891e-6</td>
<td>6.787e-6</td>
</tr>
</tbody>
</table>

fracture transmissivity and choosing as initial guess (see step 1 of the conjugate gradient algorithm) the null vector. Furthermore, the parameter \( \alpha \) 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 source fracture) and an homogeneous Dirichlet boundary condition, \( h = 0 \), on one edge of a different fracture (the sink fracture); all other edges in the DFN are insulated.

4.2. Robustness. In this subsection, we compare the behaviour of the method on the four networks here considered. In order to provide an overview of the solution obtained, we report in Figure 4.5 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 of all, we highlight that, as expected, the method was capable to perform 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 $I_{in}$ and $I_{out}$ denote the index sets corresponding to source and sink fractures, respectively. We define the computed average flux through the network as

$$\bar{\Phi} = \frac{1}{2} \left( \sum_{i \in I_{in}} \sum_{S_m \in S_i} \int_{S_m} (u^m_i - \alpha h^i_{|S_m}) - \sum_{i \in I_{out}} \sum_{S_m \in S_i} \int_{S_m} (u^m_i - \alpha h^i_{|S_m}) \right).$$

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

$$\Delta_{\text{cons}} = \frac{\left| \sum_{i \in I_{in} \cup I_{out}} \sum_{S_m \in S_i} \int_{S_m} (u^m_i - \alpha h^i_{|S_m}) \right|}{\Phi}.$$

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

$$\Delta_{\text{head}} = \sqrt{\frac{\sum_{m=1}^{M} ||h^i_{|S_m} - h^j_{|S_m}||^2}{h_{\text{max}} L}}, \quad \Delta_{\text{flux}} = \sqrt{\frac{\sum_{m=1}^{M} (u^m_{i} + u^m_{j} - \alpha (h^i_{|S_m} + h^j_{|S_m}))^2}{\Phi L}},$$

being $L$ the cumulative trace length.

In Table 4.3 we report for all the DFNs and the mesh sizes here considered the values of the relative residual computed through iterations, and of the corresponding flux conservation errors. In Table 4.4 we report the hydraulic head mismatches and flux unbalances. It is clearly seen that the relative residual and the error measures attained for 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 be not affected by the presence of

### Table 4.4

**Hydraulic head mismatch $\Delta_{\text{head}}$ and flux unbalance $\Delta_{\text{flux}}$ through iterations**

<table>
<thead>
<tr>
<th># iter</th>
<th>$\Delta_{\text{head}}$</th>
<th>$\Delta_{\text{flux}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\delta = 0.5$</td>
<td>$\delta = 2$</td>
</tr>
<tr>
<td>DFN709</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1000</td>
<td>2.116e-6</td>
<td>2.356e-6</td>
</tr>
<tr>
<td>10000</td>
<td>5.138e-7</td>
<td>8.427e-7</td>
</tr>
<tr>
<td>20000</td>
<td>5.002e-7</td>
<td>8.278e-7</td>
</tr>
<tr>
<td>DFN878</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1000</td>
<td>2.892e-6</td>
<td>3.322e-6</td>
</tr>
<tr>
<td>10000</td>
<td>7.318e-7</td>
<td>1.149e-6</td>
</tr>
<tr>
<td>20000</td>
<td>7.030e-7</td>
<td>1.127e-6</td>
</tr>
<tr>
<td>DFN915</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1000</td>
<td>2.774e-6</td>
<td>3.187e-6</td>
</tr>
<tr>
<td>10000</td>
<td>7.595e-7</td>
<td>1.117e-6</td>
</tr>
<tr>
<td>20000</td>
<td>7.109e-7</td>
<td>1.095e-6</td>
</tr>
<tr>
<td>DFN1425</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1000</td>
<td>9.025e-7</td>
<td>1.009e-6</td>
</tr>
<tr>
<td>10000</td>
<td>2.168e-7</td>
<td>3.460e-7</td>
</tr>
<tr>
<td>20000</td>
<td>2.121e-7</td>
<td>3.424e-7</td>
</tr>
</tbody>
</table>
intersecting traces with very narrow angles, or very close non-intersecting traces. We remark that, even if increasing the number of iterations performed the accuracy of the solution of the linear system gets improved, the hydraulic head continuity mismatch and the flux unbalance due to the mesh non-conformities reach a plateau. Hence, an higher number of iterations would not result in a significant improvement of the mismatches, which could be reduced only refining the meshes.

Remark 4. The number of iterations required to get acceptable results is quite high, if compared, e.g., with classical preconditioned Domain Decomposition methods. The introduction of a suitable preconditioner in order to reduce the iteration numbers is clearly recommended. On the other hand, since the approach here adopted and classical DD methods yield CG reduced problems with different structure (see Remark 2), classical DD preconditioners [26, 27, 14, 28, 18] can not 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 eight-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 PBS job.
scheduling system 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, and selecting the minimum time. This procedure, however, can only partially compensate fluctuations of simulation time due to cluster load.

Table 4.5
Number of communicated $h$-DOFs and $u$-DOFs for various numbers of parallel processes

<table>
<thead>
<tr>
<th>DFN Slaves</th>
<th>$\delta = 0.5$ [%]</th>
<th>$\delta = 2$ [%]</th>
<th>$\delta = 7$ [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>$h$ DOFs comm</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>709</td>
<td>6978 [1.9]</td>
<td>3648 [1.0]</td>
<td>2119 [0.6]</td>
</tr>
<tr>
<td>$u$ DOFs comm</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 4.6. DFN1425: number of fractures (left) and communicating traces (right) per process for a 15 Slave simulation.

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 pro-
In the following, we focus on the smallest and the largest DFNs here considered, namely DFN709 and DFN1425. Figures 4.6-4.9 give an insight of the partitioning process performed by the library METIS, in the case of a 15-Slave run for the larger DFN. Figure 4.6 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 4.7-4.9. Figure 4.7 reports the number of $h$-DOFs on the fractures handled by each process. Since the number of $h$-DOFs is related to 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 4.8 and 4.9 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 on the weight of communications. It is to remark that, even if both the chosen DFN configurations have an high density of traces (see Table 4.1) the number of communicated DOFs is small compared to the total number of degrees of freedom, with percentages of about 1% with 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 4.5.

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 respectively as

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

In Figure 4.10 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 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 behaviour, 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 4.6. The values reported show that, as a whole, acceptable efficiencies are achieved by the algorithm, and, for the higher number of

<table>
<thead>
<tr>
<th>Version</th>
<th>Slaves</th>
<th>Time [sec]</th>
<th>Efficiency %</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>$\delta = 0.5$</td>
<td>$\delta = 2$</td>
</tr>
<tr>
<td>DFN709</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Serial</td>
<td>1</td>
<td>0.83754</td>
<td>0.23432</td>
</tr>
<tr>
<td>MPI</td>
<td>3</td>
<td>0.29631</td>
<td>0.08324</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>0.16898</td>
<td>0.04758</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>0.10830</td>
<td>0.03038</td>
</tr>
<tr>
<td></td>
<td>31</td>
<td>0.06239</td>
<td>0.01632</td>
</tr>
<tr>
<td>MPI*</td>
<td>15</td>
<td>0.11496</td>
<td>0.04281</td>
</tr>
<tr>
<td></td>
<td>31</td>
<td>0.07350</td>
<td>0.02281</td>
</tr>
<tr>
<td></td>
<td>63</td>
<td>0.03431</td>
<td>0.01194</td>
</tr>
<tr>
<td></td>
<td>126</td>
<td>0.02243</td>
<td>0.00629</td>
</tr>
<tr>
<td>DFN1425</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Serial</td>
<td>1</td>
<td>2.67725</td>
<td>0.70857</td>
</tr>
<tr>
<td>MPI</td>
<td>3</td>
<td>0.91974</td>
<td>0.24591</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>0.48278</td>
<td>0.12251</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>0.28466</td>
<td>0.07176</td>
</tr>
<tr>
<td></td>
<td>31</td>
<td>0.14775</td>
<td>0.03888</td>
</tr>
<tr>
<td>MPI*</td>
<td>15</td>
<td>0.28194</td>
<td>0.07920</td>
</tr>
<tr>
<td></td>
<td>31</td>
<td>0.14638</td>
<td>0.04149</td>
</tr>
<tr>
<td></td>
<td>63</td>
<td>0.14838</td>
<td>0.04244</td>
</tr>
<tr>
<td></td>
<td>127</td>
<td>0.06611</td>
<td>0.01881</td>
</tr>
</tbody>
</table>
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 MasterSlave communication algorithm, each Slave process first solves the linear systems on the communicating fractures, performs non-blocking send operations and then solves the linear systems on the remaining non-communicating fractures. Only after these operations are completed the data from the Master are received. As shown by the reported data, this communication algorithm allows 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 a part of them is 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 status 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 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 the 50% are achieved with up to 16 parallel processes, and 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 in 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 reducing 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


