INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
Int. J. Numer. Meth. Fluids **28**: 23–46 (1998)

A PARALLEL SOLVER BASED ON THE DUAL SCHUR DECOMPOSITION OF GENERAL FINITE ELEMENT MATRICES

DENIS VANDERSTRAETEN AND ROLAND KEUNINGS*

Unité de Mécanique Appliquée, Université catholique de Louvain, Av. G. Lemaître, 4–6, B-1348 Louvain-la-Neuve, Belgium

SUMMARY

A parallel solver based on domain decomposition is presented for the solution of large algebraic systems arising in the finite element discretization of mechanical problems. It is hybrid in the sense that it combines a direct factorization of the local subdomain problems with an iterative treatment of the interface system by a parallel GMRES algorithm. An important feature of the proposed solver is the use of a set of Lagrange multipliers to enforce continuity of the finite element unknowns at the interface. A projection step and a preconditioner are proposed to control the conditioning of the interface matrix.

The decomposition of the finite element mesh is formulated as a graph partitioning problem. A two-step approach is used where an initial decomposition is optimized by non-deterministic heuristics to increase the quality of the decomposition.

Parallel simulations of a Navier–Stokes flow problem carried out on a Convex Exemplar SPP system with 16 processors show that the use of optimized decompositions and the preconditioning step are keys to obtaining high parallel efficiencies. Typical parallel efficiencies range above 80%. © 1998 John Wiley & Sons, Ltd.

KEY WORDS: domain decomposition; parallel algorithms; finite element; Lagrange multipliers; projected GMRES

1. INTRODUCTION

In the finite element discretization of non-linear partial differential equations (PDE's), it is usual to solve large sparse matrix systems. The speed and memory size of a single workstation are often insufficient, so powerful parallel computers are usually used. Among the various techniques available for the parallel solution of algebraic systems, the domain decomposition technique has become very popular because of its intrinsic parallelism [1–3]. Indeed, the solution of the problems local to each subdomain is naturally parallel while the communication phase is restricted to the evaluation of interface quantities.

In this paper, a hybrid domain decomposition solver for general finite element computations is proposed, which combines a direct solution of the subdomain problems with an iterative treatment of the interface system. This work generalizes the dual Schur technique (FETI method) proposed by Farhat and Roux [4,5] for the solution of symmetric positive definite matrices, and extends it to the solution of general matrices.

* Correspondence to: Unité de Mécanique Appliquée, Université catholique de Louvain. Av. G. Lemaître, 4–6, B-1348 Louvain-la-Neuve, Belgium.

CCC 0271–2091/98/110023–24$17.50

Received 5 December 1996
Revised 17 June 1997

Briefly, the main features of the dual Schur approach are as follows. First, the finite element mesh is partitioned in non-overlapping subdomains. In each subdomain, a local problem is defined and solved by a direct Gaussian elimination. The continuity of the unknowns at both sides of the interface are enforced by a set of discrete Lagrange multipliers. In other words, interactions between the subdomains are expressed by Neumann boundary conditions applied at the interface between adjacent subdomains. This approach can be considered as dual to the Schur complement approach in the sense that Neumann conditions are used rather than the usual Dirichlet conditions. An arbitrary mesh partition usually contains floating subdomains where no Dirichlet conditions are applied. Therefore, the set of Lagrange multipliers must be completed by 'rigid modes' of the floating subdomains in order to uniquely determine the solution of the local problems[1]. The interface system is expressed in terms of Lagrange multipliers and rigid modes. In this paper, a projection theorem is proposed to decouple the computation of both sets of variables. The projected system is solved by a parallel GMRES algorithm [6,7] where the preconditioner proposed by Farhat and Roux [4,5] is used to obtain good scalability.

The quality of the partition plays an important role as far as parallel efficiency of the hybrid solver is concerned. A two-step approach is used [8–10], wherein an initial partition is created and further optimized by non-deterministic heuristics such as the Simulated Annealing algorithm [11]. The goals of the optimization step are (a) the reduction of the interface size, (b) the modification of the aspect ratio of the subdomains to control the conditioning of the interface matrix, and finally (c) the adjustment of the number of elements in the subdomains to balance the cost of the local factorizations.

The MIMD programming model in a distributed memory environment was chosen. Communications between processors are performed by means of explicit message passing using the parallel virtual machine (PVM) library [12]. Unless stated otherwise, each processor is assigned a single subdomain.

Using the decompositions generated by the two-step approach, the hybrid solver for Navier–Stokes flow problems has been tested on the Convex Exemplar SPP system with 16 processors. Typical parallel efficiencies range above 80% and the use of optimized decompositions and of the preconditioner lead to good scalability properties.

The remainder of this paper is organized as follows. In Section 2, the formulation of the dual Schur method is derived. The projection theorem for decoupling the computation of multipliers and rigid modes is also presented. Section 3 contains a description of the parallel hybrid algorithm. The parallel issues are emphasized as well as the design of the preconditioner. The generation of almost optimal decompositions adapted to the parallel solver is described in Section 4. Finally, Section 5 presents results for Navier–Stokes flow problems obtained on the Convex Exemplar SPP system with 16 processors.

2. PROBLEM FORMULATION

The formulation is based on the previous work of Farhat and Roux [5] which is extended here to general (*i.e. non-symmetric*, *non-definite*) matrices.

Consider a physical problem defined on a geometry Ω. This problem can be modeled by a set of PDE's and some appropriate boundary conditions. The finite element discretization of the problem leads to a non-linear algebraic system at every time step. This non-linear system

[1] A rigid mode is any solution that satisfies the homogeneous equations associated to the local subdomain problem.

can be solved by a standard iterative Newton–Raphson algorithm which requires, at every iteration, the solution of the linear algebraic system:

$$\boldsymbol{K}\boldsymbol{u} = \boldsymbol{f}, \tag{1}$$

where $\boldsymbol{K}$ is the stiffness matrix of size $N_v \times N_v$, $\boldsymbol{f}$ is the load vector and $\boldsymbol{u}$ is the vector of the nodal unknowns. With the exception of sparsity, we do not assume any particular property of the stiffness matrix, i.e. it can be non-symmetric and indefinite.

In the dual Schur domain decomposition approach to the solution of Equation (1), the domain Ω is decomposed into P non-overlapping subdomains $\Omega_1 \ldots \Omega_P$. The interface between the subdomains is denoted by Γ. Given a particular decomposition, the unknowns can be sorted as *interface* unknowns and *internal* unknowns. Let $\boldsymbol{u}_1 \ldots \boldsymbol{u}_P$ denote the restriction of the solution vector $\boldsymbol{u}$ to the unknowns internal to $\Omega_1 \ldots \Omega_P$ and $\boldsymbol{u}_\mathrm{I}$ be the restriction of the solution vector to the interface unknowns (Figure 1). If the nodal unknowns of Ω_1 are numbered first, next those of $\Omega_2, \ldots$ and finally those of Γ, the finite element system (1) reads

$$\begin{pmatrix} \boldsymbol{K}_{11} & \ldots & 0 & \boldsymbol{K}_{1\mathrm{I}} \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \ldots & \boldsymbol{K}_{PP} & \boldsymbol{K}_{P\mathrm{I}} \\ \boldsymbol{K}_{\mathrm{I}1} & \ldots & \boldsymbol{K}_{\mathrm{I}P} & \boldsymbol{K}_{\mathrm{II}} \end{pmatrix} \begin{pmatrix} \boldsymbol{u}_1 \\ \vdots \\ \boldsymbol{u}_P \\ \boldsymbol{u}_\mathrm{I} \end{pmatrix} = \begin{pmatrix} \boldsymbol{f}_1 \\ \vdots \\ \boldsymbol{f}_P \\ \boldsymbol{f}_\mathrm{I} \end{pmatrix}. \tag{2}$$

Here, the subscript I refers to the interface. The nodal values $\boldsymbol{u}_s$ ($s = 1, 2, \ldots P$) internal to the subdomains are coupled indirectly through the interface unknowns $\boldsymbol{u}_\mathrm{I}$. Note also that most of the $\boldsymbol{K}$-submatrices appearing in Equation (2) can be computed locally within one subdomain. Indeed, the only exception is $\boldsymbol{K}_\mathrm{II}$, which is the sum of the contributions $\boldsymbol{K}_\mathrm{II}^{(s)}$ computed in the different subdomains Ω_s ($s = 1, 2, \ldots P$).

2.1. Local systems

The stiffness matrix and load vector *local to* Ω_s read

$$\boldsymbol{K}^{(s)} = \begin{pmatrix} \boldsymbol{K}_{ss} & \boldsymbol{K}_{s\mathrm{I}} \\ \boldsymbol{K}_{\mathrm{I}s} & \boldsymbol{K}_\mathrm{II}^{(s)} \end{pmatrix}, \tag{3}$$

and

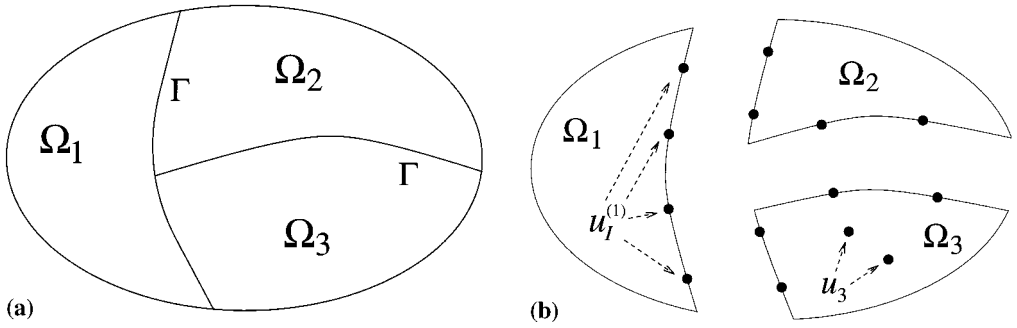

Figure 1. (a) Domain Ω decomposed in three subdomains Ω_1, Ω_2 and Ω_3. The interface between the subdomains is noted Γ; (b) interface variables $\boldsymbol{u}_\mathrm{I}^{(1)}$ of Ω_1 and interior variables $\boldsymbol{u}_3$ of Ω_3.

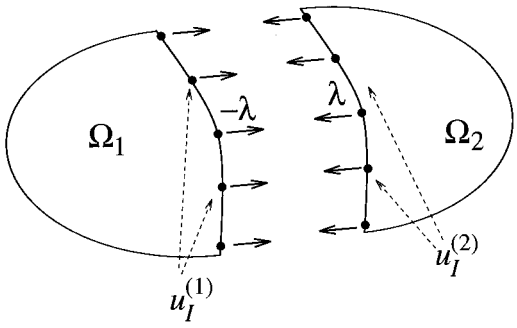

Figure 2. Domain Ω decomposed into two subdomains Ω_1 and Ω_2. The vector $\boldsymbol{\lambda}$ represents the 'tractions' needed to 'join' the two subdomains.

$$\boldsymbol{f}^{(s)} = \begin{pmatrix} \boldsymbol{f}_s \\ \boldsymbol{f}_{\mathrm{I}}^{(s)} \end{pmatrix}. \tag{4}$$

The variables associated to subdomain Ω_s are sorted as internal variable $\boldsymbol{u}_s$ and interface variables $\boldsymbol{u}_{\mathrm{I}}^{(s)}$,

$$\boldsymbol{u}^{(s)} = \begin{pmatrix} \boldsymbol{u}_s \\ \boldsymbol{u}_{\mathrm{I}}^{(s)} \end{pmatrix}. \tag{5}$$

In the following, the dimension of $\boldsymbol{K}^{(s)}$ is denoted by N_s. The total number of interface variables is N_{I}, and $N_{\mathrm{I}}^{(s)}$ is the size of the interface between Ω_s and its adjacent subdomain(s).

If there is no interaction between the subdomains, the systems defined by Equations (3), (4) and (5) can be solved independently in all subdomains. In general, however, some interactions exist on the interfaces between the subdomains and they can be modeled by Neumann boundary conditions applied on the interface. Mathematically, this corresponds to the fact that, in a subdomain, the local equations related to the interface variables are not completely assembled.

Let us define a vector of Lagrange multipliers $\boldsymbol{\lambda}$ of size N_{I} that represents these interactions. This vector affects only the interface equations and, in Ω_s, it is defined as the contribution of neighboring subdomains to the interface equations of Ω_s. For example, for a heat transfer problem, the vector $\boldsymbol{\lambda}$ can be seen as the heat flux that goes through the interface.

For illustrative purposes, let us assume that there are only two subdomains (Figure 2). In the dual Schur approach, the systems local to Ω_1 and Ω_2 read

$$\boldsymbol{K}^{(1)}\boldsymbol{u}^{(1)} = \boldsymbol{f}^{(1)} - \begin{pmatrix} \boldsymbol{0} \\ \boldsymbol{\lambda} \end{pmatrix}, \qquad \boldsymbol{K}^{(2)}\boldsymbol{u}^{(2)} = \boldsymbol{f}^{(2)} + \begin{pmatrix} \boldsymbol{0} \\ \boldsymbol{\lambda} \end{pmatrix}. \tag{6}$$

This set of equations is closed by imposing the continuity of the interface variables,

$$\boldsymbol{u}_{\mathrm{I}}^{(1)} = \boldsymbol{u}_{\mathrm{I}}^{(2)}. \tag{7}$$

In the presence of crosspoints, a multiplier is defined between any pair of connected subdomains (Figure 3). This choice introduces redundant multipliers, but the overdetermination is easily removed by the relation between these local variables. In the example of Figure 3, the multipliers satisfy:

$$\lambda_{12} + \lambda_{23} + \lambda_{31} = 0. \tag{8}$$

In the general case of P subdomains, the system (6) local to Ω_s becomes

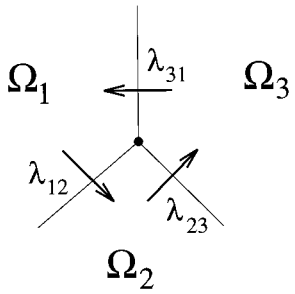

Figure 3. Interface point (crosspoint) between three subdomains, Ω_1, Ω_2, and Ω_3 and related set of Lagrange multipliers.

$$\boldsymbol{K}^{(s)}\boldsymbol{u}^{(s)} = \boldsymbol{f}^{(2)} - \boldsymbol{B}^{(s)^T}\boldsymbol{\lambda}, \tag{9}$$

where the matrix $\boldsymbol{B}^{(s)}$ is a signed Boolean matrix that extracts the interface components of a vector of Ω_s. In the two-subdomain example given above, we have $\boldsymbol{B}^{(1)} = (\mathbf{0}\ \mathrm{I})$ and $\boldsymbol{B}^{(2)} = (\mathbf{0}\ -\mathrm{I})$. The set of systems (9) is closed by imposing the equality of the interface variables:

$$\sum_{s=1}^{P} \boldsymbol{B}^{(s)}\boldsymbol{u}^{(s)} = 0. \tag{10}$$

2.2. Floating subdomains

In general, the formulation of the global finite element problem may be such that some of the subdomains may not have enough Dirichlet boundary conditions to ensure a unique local solution of system (9). Those subdomains are called *floating subdomains*. Their stiffness matrix $\boldsymbol{K}^{(s)}$ is singular and the solution of Equation (9) can be computed up to a constant vector located in the null space of $\boldsymbol{K}^{(s)}$. In the case of an incompressible 2D steady state Stokes flow problem, if no Dirichlet boundary conditions are prescribed, there exists a family of velocity fields, the solution of the problem and such that two velocity fields differ only by a constant translation velocity $(\boldsymbol{u}_x, \boldsymbol{u}_y)$.

A general expression of $\boldsymbol{u}^{(s)}$ in Equation (9) can be written as the sum of a particular solution and a translation in the null space of $\boldsymbol{K}^{(s)}$. More precisely,

$$\boldsymbol{u}^{(s)} = \boldsymbol{K}^{(s)^+}(\boldsymbol{f}^{(s)} - \boldsymbol{B}^{(s)^T}\boldsymbol{\lambda}) + \boldsymbol{N}^{(s)}\boldsymbol{\alpha}^{(s)}, \tag{11}$$

where the matrix $\boldsymbol{K}^{(s)^+}$ is the *generalized inverse*[2] of $\boldsymbol{K}^{(s)}$, the matrix $\boldsymbol{N}^{(s)}$ is made of the basis vectors of the null space of $\boldsymbol{K}^{(s)}$ and the vector $\boldsymbol{\alpha}^{(s)}$ specifies any linear combination of the vectors of the null space. The vector $\boldsymbol{\alpha}^{(s)}$ serves to define a *rigid mode* of the subdomain. Appendix A provides expressions for $\boldsymbol{K}^{(s)^+}$ and $\boldsymbol{N}^{(s)}$.

It is worth pointing out that for systems arising in the discretization of well-posed physical problems, the rank of $\boldsymbol{K}^{(s)}$ is close to N_s while the size of the null space is small. For example, in the case of an incompressible 2D Stokes flow problem, only three parameters are sufficient to uniquely determine the solution in a floating subdomain, whatever the size of the local stiffness matrix.

An additional equation is now needed to determine $\boldsymbol{\alpha}^{(s)}$. If $\boldsymbol{K}^{(s)}$ is singular and if the system (1) has a unique solution (this usually holds if the physical problem is well-posed), the

[2] The generalized inverse of $\boldsymbol{K}$ is the matrix $\boldsymbol{K}^+$ verifying the Moore–Penrose conditions: $\boldsymbol{K}^+\boldsymbol{K}\boldsymbol{K}^+ = \boldsymbol{K}^+$, $\boldsymbol{K}\boldsymbol{K}^+\boldsymbol{K} = \boldsymbol{K}$, $(\boldsymbol{K}^+\mathbf{K})^T = \boldsymbol{K}^+\boldsymbol{K}$, and $(\boldsymbol{K}\boldsymbol{K}^+)^T = \boldsymbol{K}\boldsymbol{K}^+$. If $\boldsymbol{K}$ is a square matrix of full rank, the generalized inverse is $\boldsymbol{K}^{-1}$ [13].

right-hand-side of Equation (9) must belong to the range of $\boldsymbol{K}^{(s)}$. Equivalently, it must be orthogonal to the null space of $\boldsymbol{K}^{(s)^T}$. This condition is expressed as

$$\boldsymbol{M}^{(s)^T}(\boldsymbol{f}^{(s)} - \boldsymbol{B}^{(s)^T}\boldsymbol{\lambda}) = 0, \tag{12}$$

where $\boldsymbol{M}^{(s)}$ contains the vector basis of the null space of $\boldsymbol{K}^{(s)^T}$. For symmetric stiffness matrices, the expression of $\boldsymbol{M}^{(s)}$ is identical to the expression of $\boldsymbol{N}^{(s)}$.

2.3. Interface system

Finally, Equation (11) is substituted into (10), and (12) is used for the Q floating subdomains to obtain the *interface system*, whose unknowns are the Lagrange multipliers and the rigid modes. We obtain

$$\begin{pmatrix} \boldsymbol{F}_\mathrm{I} & -\boldsymbol{G}_\mathrm{I} \\ -\boldsymbol{H}_\mathrm{I}^\mathrm{T} & 0 \end{pmatrix}\begin{pmatrix} \boldsymbol{\lambda} \\ \boldsymbol{\alpha} \end{pmatrix} = \begin{pmatrix} \boldsymbol{d} \\ -\boldsymbol{e} \end{pmatrix}, \tag{13}$$

where

$$\begin{aligned}
&\boldsymbol{F}_\mathrm{I} = \sum_{s=1}^{P} \boldsymbol{B}^{(s)}\boldsymbol{K}^{(s)^+}\boldsymbol{B}^{(s)^T},\\
&\boldsymbol{G}_\mathrm{I} = (\boldsymbol{B}^{(1)}\boldsymbol{N}^{(1)}\ldots\boldsymbol{B}^{(Q)}\boldsymbol{N}^{(Q)}),\\
&\boldsymbol{H}_\mathrm{I} = (\boldsymbol{B}^{(1)}\boldsymbol{M}^{(1)}\ldots\boldsymbol{B}^{(Q)}\boldsymbol{M}^{(Q)}),\\
&\boldsymbol{d} = \sum_{s=1}^{P} \boldsymbol{B}^{(s)}\boldsymbol{K}^{(s)^+}\boldsymbol{f}^{(s)},\\
&\boldsymbol{e} = (\boldsymbol{M}^{(1)^T}\boldsymbol{f}^{(1)}\ldots\boldsymbol{M}^{(Q)^T}\boldsymbol{f}^{(Q)})^T,\\
&\boldsymbol{\alpha} = (\boldsymbol{\alpha}^{(1)^T}\ldots\boldsymbol{\alpha}^{(Q)^T})^T.
\end{aligned} \tag{14}$$

The system (13) can be solved using any appropriate technique. However, it is useful to decouple the computation of $\boldsymbol{\lambda}$ and $\boldsymbol{\alpha}$. The following proposition enables the decoupled evaluation of the Lagrange multipliers and the rigid modes of the floating subdomains.

Proposition 1 *With the above notations, the solution of the system*

$$\begin{pmatrix} \boldsymbol{F}_\mathrm{I} & -\boldsymbol{G}_\mathrm{I} \\ -\boldsymbol{H}_\mathrm{I}^T & 0 \end{pmatrix}\begin{pmatrix} \boldsymbol{\lambda} \\ \boldsymbol{\alpha} \end{pmatrix} = \begin{pmatrix} \boldsymbol{d} \\ -\boldsymbol{e} \end{pmatrix} \tag{15}$$

is given by

$$\begin{cases} \boldsymbol{\lambda} &= \boldsymbol{\lambda}_0 + \boldsymbol{P}_H\boldsymbol{\lambda}' \\ \boldsymbol{\alpha} &= -(\boldsymbol{G}_\mathrm{I}^T\boldsymbol{G}_\mathrm{I})^{-1}\boldsymbol{G}_\mathrm{I}^T(\boldsymbol{d} - \boldsymbol{F}_\mathrm{I}\boldsymbol{\lambda}), \end{cases} \tag{16}$$

where

$$\boldsymbol{\lambda}_0 = \boldsymbol{H}_\mathrm{I}(\boldsymbol{H}_\mathrm{I}^T\boldsymbol{H}_\mathrm{I})^{-1}\boldsymbol{e}, \tag{17}$$

and $\boldsymbol{\lambda}'$ of size N_I is the solution of the projected interface system

$$(\boldsymbol{P}_G^T \boldsymbol{F}_\mathrm{I} \boldsymbol{P}_H)\boldsymbol{\lambda}' = \boldsymbol{P}_G^T(\boldsymbol{d} - \boldsymbol{F}_\mathrm{I}\boldsymbol{\lambda}_0), \tag{18}$$

with $\boldsymbol{P}_H$ *(resp.* $\boldsymbol{P}_G$*) representing any projector onto the null space of* $\boldsymbol{H}_\mathrm{I}^T$ (resp. $\boldsymbol{G}_\mathrm{I}^T$).
Proof: See Appendix B.

In Proposition 1, the two projectors $\boldsymbol{P}_H$ and $\boldsymbol{P}_G$ are not uniquely defined. Any projection of the form

$$\boldsymbol{P}_H = \boldsymbol{I} - \boldsymbol{Q}_H(\boldsymbol{H}_\mathrm{I}^T \boldsymbol{Q}_H)^{-1}\boldsymbol{H}_\mathrm{I}^T \tag{19}$$

is a projection onto the null space of $\boldsymbol{H}_\mathrm{I}^T$ (assuming that the product $\boldsymbol{H}_\mathrm{I}^T\,\boldsymbol{Q}_H$ is non-singular). The projection onto the null space of $\boldsymbol{G}_\mathrm{I}^T$ has a similar form and it makes use of a matrix $\boldsymbol{Q}_G$. If we choose $\boldsymbol{Q}_H = \boldsymbol{G}_\mathrm{I}$ and $\boldsymbol{Q}_G = \boldsymbol{H}_\mathrm{I}$, the two projectors $\boldsymbol{P}_H$ and $\boldsymbol{P}_G^T$ are identical which reduces the computational requirements of Equation (18). However, this single projector entails the solution of a non-symmetric system with the matrix $\boldsymbol{H}_\mathrm{I}^T\,\boldsymbol{G}_\mathrm{I}$ in the evaluation of $\boldsymbol{P}_H$ in (19). In this work, we have preferred the definition of two orthogonal projectors for which the matrix $\boldsymbol{H}_\mathrm{I}^T\,\boldsymbol{Q}_H$ appearing in (19) is symmetric and positive definite. This is achieved with $\boldsymbol{Q}_H = \boldsymbol{H}_\mathrm{I}$ and $\boldsymbol{Q}_G = \boldsymbol{G}_\mathrm{I}$.

3. PARALLEL HYBRID SOLVER

A parallel *hybrid* solver is proposed in this paper, where the internal systems (9) are solved by a direct method and the interface system (13) uses an iterative method. It is assumed that the parallel computer has P processors available and that every processor is assigned one single subdomain.

This choice results from several considerations: First, the interface matrix $\boldsymbol{F}_\mathrm{I}$ is sparse and the various terms $\boldsymbol{B}^{(s)}\boldsymbol{K}^{(s)^+}\boldsymbol{B}^{(s)^T}$are located in different subdomains. Therefore, it is essential to avoid their assembly and the factorization of $\boldsymbol{F}_\mathrm{I}$. Since iterative methods only require matrix–vector products, their use avoids the explicit construction of $\boldsymbol{F}_\mathrm{I}$.

3.1. Local subdomain systems

A matrix–vector product $\boldsymbol{F}_\mathrm{I}\boldsymbol{\lambda}$ is necessary at every iteration of the interface solver. This product involves the evaluation of $\boldsymbol{K}^{(s)^+}\boldsymbol{u}^{(s)}$ (or, equivalently, the solution of a system with matrix $\boldsymbol{K}^{(s)}$). Next, the null space of every floating subdomain must be computed. Therefore, we have chosen to perform an explicit $\boldsymbol{LU}$ factorization of $\boldsymbol{K}^{(s)}$. The products $\boldsymbol{K}^{(s)^+}\boldsymbol{u}^{(s)}$ reduce to a pair of local forward–backward substitutions. During the factorization, a column where a zero pivot is encountered is linearly dependent from the previous columns. This gives a partition of $\boldsymbol{K}^{(s)}$ into subblocks that are further used to compute the null space (see Appendix A for details). In practice, a pivot is said to be zero if it is small compared with the previous pivot.

In this particular implementation, a skyline format is used to store the local matrices. The cost of the $\boldsymbol{LU}$ factorization is thus proportional to the number of variables and to the square of the average bandwidth. It is clear that a more robust implementation could be achieved with advanced techniques using partial or complete pivoting.

Table I. Pseudo-code for the solution of the interface problem with the projected GMRES(m) algorithm

(1) Initialize: $\boldsymbol{\lambda}_0 = \boldsymbol{H}_I(\boldsymbol{H}_I{}^T\boldsymbol{H}_I)^{-1}\boldsymbol{e}$
(2) compute residual: $\boldsymbol{r}_0 = \boldsymbol{P}_G{}^T(\boldsymbol{d} - \boldsymbol{F}_I\boldsymbol{\lambda}_0)$
(3) Iterate $k = 1 \ldots$ until $\|\boldsymbol{r}_{k-1}\|_2 \leq tol$
  (3.1) Project: $\boldsymbol{q}_0 = \boldsymbol{P}_H\boldsymbol{r}_{k-1}$
  (3.2) Set descent direction: $\boldsymbol{s} = \|\boldsymbol{q}_0\|_2\boldsymbol{e}_1$
    $\boldsymbol{p}_1 = \boldsymbol{q}_0/\|\boldsymbol{q}_0\|_2$
  (3.3) For $i = 1$ to m
    (3.3.1) Compute descent direction: $\boldsymbol{z}_i = \boldsymbol{P}_G{}^T\boldsymbol{F}_I\boldsymbol{p}_i$
      $\boldsymbol{q}_i = \boldsymbol{P}_H\boldsymbol{z}_i$
    (3.3.2) Orthogonalize $\boldsymbol{q}_i$: For $j = 1$ to $i-1$
      (3.3.2.1) $\boldsymbol{He}_{j,i} = \boldsymbol{q}_i^T\boldsymbol{p}_j$
      (3.3.2.2) $\boldsymbol{q}_i = \boldsymbol{q}_i - \boldsymbol{He}_{j,i}\boldsymbol{p}_j$
    (3.3.3) Update descent direction: $\boldsymbol{He}_{i+1,i} = \|\boldsymbol{q}_i\|_2$
      $\boldsymbol{p}_{i+1} = \boldsymbol{q}_i/\|\boldsymbol{q}_i\|_2$
    (3.3.4) Update residual: Compute Givens rotation $\boldsymbol{J}_i$
      $\boldsymbol{He}_{\cdot,i} = \boldsymbol{J}_1 \ldots \boldsymbol{J}_i\boldsymbol{He}_{\cdot,i}$
      $\boldsymbol{s} = \boldsymbol{J}_i\boldsymbol{s}$
    (3.3.5) If ($\boldsymbol{s}_{i+1}$ is small enough) then Break
  (3.4) Solve $\boldsymbol{He}\boldsymbol{\beta} = \boldsymbol{s}$
  (3.5) Update solution: $\boldsymbol{\lambda}_k = \boldsymbol{\lambda}_{k-1} + \sum_{l=1}^{i}\beta_l\boldsymbol{p}_l$
  (3.6) Update residual: $\boldsymbol{r}_k = (\boldsymbol{d} - \boldsymbol{F}_I\boldsymbol{\lambda}_k)$
(4) Compute $\boldsymbol{\alpha} = -(\boldsymbol{G}_I{}^T\boldsymbol{G}_I)^{-1}\boldsymbol{G}_I{}^T(\boldsymbol{d} - \boldsymbol{F}_I\boldsymbol{\lambda}_k)$
(5) Return $\boldsymbol{\lambda}_k$, $\boldsymbol{\alpha}$

3.2. Interface solver: parallel projected GMRES

Instead of solving Equation (13), the projected interface system (18) is solved, ensuring that at each iteration, the descent direction $\boldsymbol{p}_k$ belongs to the null space of $\boldsymbol{H}_\mathrm{I}^T$ and that the initial solution $\boldsymbol{\lambda}_0$ satisfies the constraint (17). The matrix $\boldsymbol{F}_\mathrm{I}$ is generally non-symmetric and indefinite. Several iterative algorithms have been proposed to solve the non-symmetric problem (see Reference [14] and the references therein). Here, a parallel GMRES algorithm was implemented [6,7,15]. The pseudo-code for the algorithm is given in Table I.

3.3. Parallel issues

It is important to understand that every component of an interface vector[3] is stored in the two subdomains it belongs to. In the GMRES algorithm, the parallel issues are addressed in

the dot products: If $\boldsymbol{v}$ is an interface quantity of the GMRES algorithm and $\boldsymbol{v}^{(s)}$ is the restriction of $\boldsymbol{v}$ to the interface of Ω_s, the product $\boldsymbol{v}^T\boldsymbol{v}$ is computed by

$$\boldsymbol{v}^T\boldsymbol{v} = \frac{1}{2}\sum_{s=1}^{P}\boldsymbol{v}^{(s)T}\boldsymbol{v}^{(s)}. \tag{20}$$

[3] 'Interface vector', refers to the parallel vectors used in the GMRES algorithm for the evaluation of the Lagrange multipliers and not to the primal interface quantities $\boldsymbol{u}_\mathrm{I}$. Therefore, every interface vector is related to exactly two subdomains (Figure 3).

The factor 1/2 results from the fact that every portion of an interface vector is stored in two different subdomains. In Equation (20), the local dot products are performed independently while the summation consists of a global parallel function over all processors;

the matrix–vector products: The product by $\boldsymbol{F}_{\mathrm{I}}$ involves four steps: (a) the mapping of an interface vector onto a local vector (multiplication by $\boldsymbol{B}^{(s)^T}$), (b) the solution of a system with the matrix $\boldsymbol{K}^{(s)}$, (c) the extraction of an interface quantity from a local vector (multiplication by $\boldsymbol{B}^{(s)}$), and (d) the assembly of the local interface quantities (summation). The first three steps are performed entirely in parallel, while the summation requires the exchange of a portion of the interface vectors between neighboring processors. Since the manipulations with matrix $\boldsymbol{B}^{(s)}$ do not involve any floating point operations, the main computational part is performed during the solution of the local systems;

the projection by $\boldsymbol{P}_{\boldsymbol{H}}$ and $\boldsymbol{P}_{\boldsymbol{G}}$: The assembly and factorization of the matrix $\boldsymbol{H}_{\mathrm{I}}^T\boldsymbol{H}_{\mathrm{I}}$ should be avoided since the data are located in different subdomains. We use a parallel CG algorithm where every multiplication by $\boldsymbol{H}_{\mathrm{I}}$ requires a communication step (the multiplication by $\boldsymbol{H}_{\mathrm{I}}$ is defined as the summation of the products $\boldsymbol{B}^{(s)}\boldsymbol{M}^{(s)}$). Note that the size of $\boldsymbol{H}_{\mathrm{I}}^T\boldsymbol{H}_{\mathrm{I}}$ is small compared with the size of $\boldsymbol{F}_{\mathrm{I}}$, and the solution of the system does not incur excessive additional work.

3.4. Preconditioning

It is always useful to perform a preconditioning step to reduce the number of iterations. This is especially true for the GMRES algorithm, since the algebraic complexity and the amount of communication grow linearly with the iteration number (Point 3.2.2 of Table I).

The preconditioner described by Farhat and Roux [5] has been implemented. In the absence of floating subdomains, the interface matrix reads

$$\boldsymbol{F}_{\mathrm{I}} = \sum_{s=1}^{P} \boldsymbol{B}^{(s)}\boldsymbol{K}^{(s)-1}\boldsymbol{B}^{(s)^T}. \tag{21}$$

Every term of the above equation can be seen as the extraction and assembly of the interface subblock of $\boldsymbol{K}^{(s)-1}$. If the interface variables are numbered last in every subdomain, it can be shown that

$$\boldsymbol{B}^{(s)}\boldsymbol{K}^{(s)-1}\boldsymbol{B}^{(s)^T} = \boldsymbol{S}^{(s)-1}, \tag{22}$$

where the matrix $\boldsymbol{S}^{(s)}$ is the Schur complement matrix of $\boldsymbol{K}^{(s)}$ defined by

$$\boldsymbol{S}^{(s)} = \boldsymbol{K}_{\mathrm{II}}^{(s)} - \boldsymbol{K}_{\mathrm{I}s}\boldsymbol{K}_{ss}^{-1}\boldsymbol{K}_{s\mathrm{I}}. \tag{23}$$

The inverse of the sum $\boldsymbol{F}_{\mathrm{I}}^{-1}$ is approximated by the sum of the inverses and the following is obtained:

$$\tilde{\boldsymbol{F}}_{\mathrm{I}}^{-1} = \sum_{s=1}^{P} \boldsymbol{B}^{(s)}\begin{pmatrix}\boldsymbol{0} & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{S}^{(s)}\end{pmatrix}\boldsymbol{B}^{(s)^T}. \tag{24}$$

This expression provides an efficient preconditioner. It has also been used in Reference [5], where optimality is discussed for symmetric matrices.

The evaluation of the preconditioner is not a trivial task. In general, the numbering of the local variables is determined such as to minimize the bandwidth of $\boldsymbol{K}^{(s)}$. The interface variables are not necessarily numbered last and the Schur complement is not a by-product of the factorization. Two methods have been considered for the evaluation of the product $\tilde{\boldsymbol{F}}_{\mathrm{I}}^{-1}\boldsymbol{\lambda}$:

I1 The explicit construction of $\boldsymbol{S}^{(s)}$. This method requires the factorization of $\boldsymbol{K}_{ss}$ and the solution of $N_{\mathrm{I}}^{(s)}$ systems (evaluation of $\boldsymbol{K}_{ss}^{-1}\boldsymbol{K}_{s\mathrm{I}}$). During the iterative algorithm, every matrix–vector product is performed without additional solution of local systems.
I2 The evaluation of the product $\boldsymbol{S}^{(s)}\boldsymbol{v}$ without explicitly computing $\boldsymbol{S}^{(s)}$. The matrix $\boldsymbol{K}_{ss}$ is factorized but the product $\boldsymbol{K}_{\mathrm{I}s}\boldsymbol{K}_{ss}^{-1}\boldsymbol{K}_{s\mathrm{I}}$ is not computed. Every multiplication by $\boldsymbol{S}^{(s)}$ during the iterative GMRES involves the solution of a system and the multiplication by three subblocks of $\boldsymbol{K}^{(s)}$. This cost is close to that of the evaluation of $\boldsymbol{F}_{\mathrm{I}}\boldsymbol{\lambda}$.

Both implementations are to be considered, depending on the problem. Method **I1** requires less memory. In fact, the size of the Schur matrix is smaller than that of the factors of $\boldsymbol{K}_{ss}$. However, it is not clear which method is computationally better. There is a compromise between the preprocessing step and the cost per iteration. Section 5.5 presents some results and draws some guidelines to decide in favor of a particular method depending on the application.

4. AUTOMATIC PARTITIONING OF FINITE ELEMENT MESHES

The optimal (or at least near-optimal) partitioning of the finite element mesh is crucial to obtaining reasonable efficiency in the hybrid solver. Ideally, the load in the subdomains must be balanced while communications (and, by extension, the solution cost of the interface system) must be minimized. In this section, the features of a 'good' decomposition are analyzed, and the two-step methodology developed recently for the generation of quasi-optimal decompositions is described [8–10].

4.1. Algebraic complexity

Estimation of the complexity of the algorithm highlights the sources of parallel inefficiency and emphasizes the influence of the partition quality.

- The first loss of efficiency comes from an imperfect load balance of the local factorizations. The number of operations required for the factorization of a local stiffness matrix, L_s, is roughly proportional to the number of variables and to the bandwidth:

$$L_s \propto N_s \times (F_s)^2, \tag{25}$$

  where F_s is the bandwidth and N_s denotes the number of variables in Ω_s. The bandwidth depends upon the ordering of the variables. In this implementation, the variables are numbered consecutively as the finite elements are being processed. This means that there is a relationship between the frontal width of the finite element mesh and the bandwidth of the stiffness matrix. Therefore, minimizing the frontal width also amounts to decreasing the bandwidth.
- Communications decrease the parallel efficiency of the parallel GMRES algorithm. They occur in the parallel global summations and in the transfer of interface quantities between neighboring processors. The number of global summations grows as the square of the number of iterations (Point 3.2.2). The size of the transfers of interface quantities depends on the local interface sizes and the number of transfers is proportional to the average number of neighboring subdomains.
- The number of iterations needed to achieve convergence increases with the condition number of the matrix $\boldsymbol{F}_{\mathrm{I}}$ [15,16]. Therefore, the interface matrix should have a condition number as small as possible. Let $h_{\max}$ and $h_{\min}$ denote respectively the largest and smallest

distance between two nodal points belonging to the same element in a finite element mesh. For elliptic self-adjoint operators, Fried [17] has proven that the condition number of the stiffness matrix is a function of the aspect ratio $h_{\min}/h_{\max}$ [17]. The concept of aspect ratio can be extended to meshes decomposed into subdomains. Indeed, in a domain decomposition method, the interface matrix can be seen as the condensation of the stiffness matrix along the interfaces. From this perspective, the subdomains are treated like super-elements. The aspect ratio (AR) of a 2D domain has been defined here as the ratio of the surface of this domain over the surface of its circumscribed circle[4]. By analogy, it can be reasonably predicted that the condition number will depend on the aspect ratio of the subdomains, i.e. long subdomains will generate interface systems that are ill-conditioned and difficult to solve iteratively.

4.2. Generation of decompositions

Considering the above discussion, an optimal decomposition must have subdomains with circle-like or sphere-like shapes and with a small interface size. In addition, the number of elements and their ordering in each subdomain must be adapted to ensure a low local frontal width and a cost of the factorization that is almost identical in all the subdomains.

The problem of finding the optimal decomposition can be formulated as a *graph partitioning problem*, where the computational domain is represented by a graph and where the features of the parallel hybrid solver are modeled by a cost function to be minimized. The cost function used in this work is defined as the weighted sum of three terms: (a) the number of interface nodes, (b) the load imbalance where the load is estimated by Equation (25), and (c) the average aspect ratio of the subdomains. Note that the third requirement makes use of the co-ordinates of the nodes of the finite element mesh. Hence, our problem cannot be formulated, strictly speaking, as a pure graph problem.

In general, the above partitioning problem, being *NP*-complete, is hopeless for seeking optimal solutions. In References [8–10], a *two-step strategy* has been developed to compute suboptimal decompositions:

- In the first step, a direct algorithm generates an initial decomposition. Representative direct algorithms are the Greedy, RSB, or RGB algorithms [8,18,19]. These algorithms are relatively fast but they are unable to systematically decrease the cost function. In addition, all the subdomains have the same number of elements, which leads to load imbalance due to different frontal widths.
- In the second step, this initial decomposition is optimized by minimizing the cost function. This is performed by non-deterministic heuristics that transfer interface elements onto neighboring subdomains. Various heuristics similar to the Simulated Annealing algorithm [11] have been implemented. In addition, a graph contraction procedure speeds up the optimization step.

The reordering of the elements after the optimization step should be avoided because this would cause variations of bandwidth which would result in load imbalance. In our implementation, a reordering occurs at some critical points during the optimization and serves to re-estimate the load. This enables the reduction in bandwidth while preserving an acceptable load balance. The generation of optimized decompositions is described in References [9,20,21].

[4] For 3D objects, the aspect ratio is defined as the ratio of the volume of the object over the volume of its circumscribed sphere [21].

The influence of the decomposition quality on the performance of the hybrid solver is discussed in Section 5.4.

5. PERFORMANCE OF THE HYBRID SOLVER

The hybrid solver has been tested on the contraction channel (STEP) and the square cavity (SQUARE) presented in Figure 4. For the STEP mesh, discretizations in 1200, 2700 and 4800 elements have been considered while the number of elements in the SQUARE mesh ranges from 16×16 to 80×80 elements. The steady state incompressible Navier–Stokes equations ($Re = 100$) have been solved on these meshes, using a classical Galerkin formulation with nine-node elements for the velocity field and four-node elements for the pressure field. The various discretizations of the STEP mesh generate 11 213, 24 918 and 44 023 unknowns, respectively. The number of variables of the SQUARE mesh ranges from 2467 to 58 403. In the following, timing results obtained for the solution of one linear system arising in the Newton–Raphson iterations are presented. The implementation **I2** of the preconditioner was used and the convergence criterion of the GMRES algorithm was set to $tol = 10^{-7}$.

The tests have been carried out on a 16-processor Convex Exemplar SPP system with a global memory of 2 Gbytes. This machine allows the use of either a shared memory or a distributed memory programming model. In the present work, the code is implemented using the PVM message passing environment. The local configuration of the Exemplar dedicates one processor to system administration; therefore, results with 1, 2, 4, 8 and 15 processors are presented.

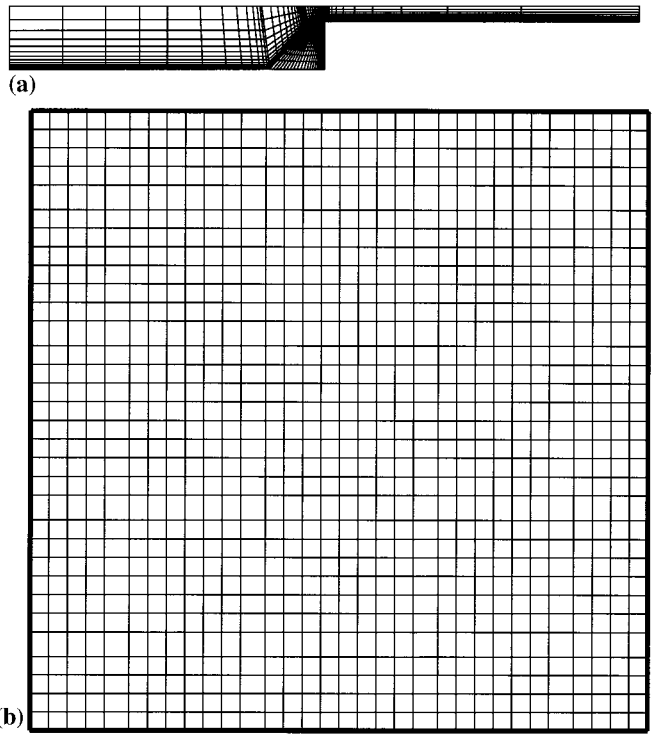

Figure 4. (a) STEP (1200 elements) and (b) SQUARE (1024 elements) finite element meshes.

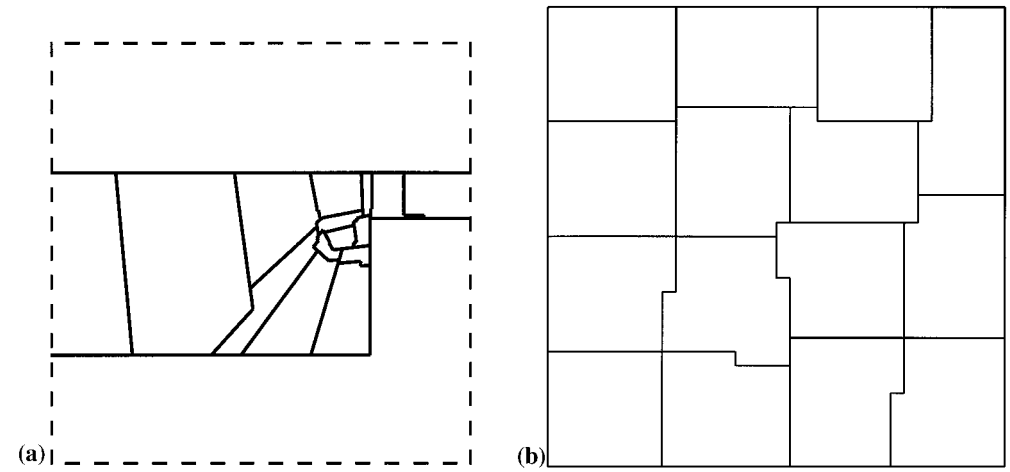

Figure 5. (a) Decomposition of the STEP mesh in 15 domains (zoom of the central part), and (b) decomposition of the SQUARE mesh in 15 domains using the Greedy algorithm followed by the Simulated Annealing optimizer.

The decompositions of the STEP mesh have been produced by the two-step procedure described in Section 4, using the Greedy algorithm followed by the Simulated Annealing algorithm. The SQUARE mesh is decomposed by 2×1, 2×2 and 4×2 rectangles while the decompositions in 15 domains are generated by our classical two-step approach. Typical decompositions in 15 domains are presented in Figure 5.

5.1. Typical result

Figure 6 presents the typical evolution of the maximum subdomain factorization time T_{fact} and the interface solution time T_{interf} as a function of the number of subdomains. We observe that T_{fact} decreases super-linearly with the number of subdomains P. This fact results from the reduction of both the number of variables and the bandwidth of the local problems as the number of subdomains increases. The interface size increases with the number of subdomains,

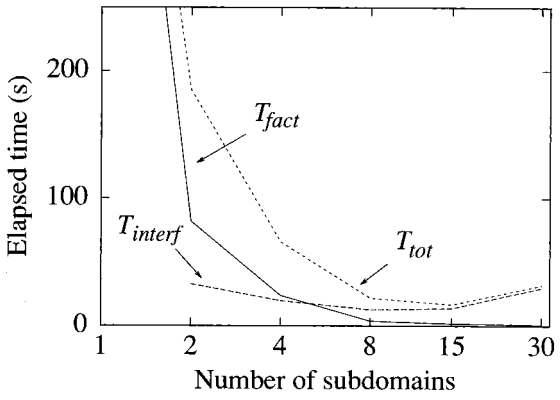

Figure 6. Evolution of T_{fact}, T_{interf}, and T_{tot} as a function of the number of subdomains for the Navier–Stokes flow problem on the 1200-element STEP mesh. The 30-domain problem has been run with 15 processors and the value of T_{interf} is estimated by the maximum local CPU time required for the solution of the interface system (communications and idle time due to swapping are discarded).

Table II. Parallel efficiencies obtained on the Convex Exemplar for the solution of a Navier–Stokes flow problem with the hybrid solver; (a) STEP and (b) SQUARE meshes

N_e	$P=2$	4	8	15
(a)				
1200	0.77	0.81	0.93	0.83
2700	0.91	0.86	0.89	0.88
4800	0.97	0.92	0.85	0.89
(b)				
16×16	0.98	0.94	0.92	0.89
32×32	0.99	0.98	0.96	0.86
48×48	0.99	0.99	0.97	0.91
64×64	0.99	0.99	0.94	0.91
80×80	–	0.99	0.94	0.83

which increases the number of iterations taken by the interface solver. However, a decrease of T_{interf} up to eight subdomains can be seen. The reduction in T_{interf} is a consequence of the reduction in the cost of the local matrix–vector products (they require the solution of a system with matrix $\boldsymbol{K}^{(s)}$) which overcomes the cost of the additional iterations. When the number of subdomains is large, however, the gain due to the matrix–vector products becomes negligible and the value of T_{interf} grows. As a consequence, for any given problem (of fixed size), the total solution time T_{tot} is minimized for some finite value of P. This value tends to increase with the size of the finite element problems.

These results have been obtained with optimized decompositions. With the initial decompositions, the load of the factorization is not well balanced and the matrix–vector product requires more computations. Therefore, the curve of T_{tot} is higher and the minimum appears for lower values of P.

5.2. Parallel efficiency

It is not worth comparing the performance of the hybrid solver with that of the sequential LU factorization of the global system (1). Indeed, a speedup figure would mainly measure the difference between the algebraic complexities of the two solvers rather than an actual parallel speedup. Consider the SQUARE mesh decomposed in two subdomains. The size of the local stiffness matrices is cut by two and the bandwidths are reduced by 50% compared with the global problem. This leads to a reduction of T_{fact} by a factor of eight. If the solution time of the interface problem remains small, it is thus possible to achieve a super-linear speedup close to eight.

For this reason, an alternative definition of parallel efficiency was used, given by

$$\epsilon_P = \frac{T_P^1}{P \times T_P^P}\,, \tag{26}$$

where T_P^Q refers to the P-domain run on Q processors. Since the number of algebraic operations needed to obtain T_P^1 and T_P^P are identical, the efficiency ϵ_P measures only the communication overheads and the load imbalance.

Table II presents the parallel efficiencies obtained for the contraction channel and the square cavity with $P=2$–15 processors and for various discretizations in N_e elements. For the STEP

mesh, the efficiencies exceed 0.77, which expresses a high ratio of computations over communications, and a good load balance. Better efficiencies are obtained for the SQUARE mesh. Indeed, the load is identical in every subdomain and the loss due to communications is small.

A decrease in efficiency is expected when the number of subdomains grows. For the 1200-element STEP mesh, however, ϵ_P increases first. For a particular problem, the performance of the hybrid solver depends on the decomposition quality. For this test case, a better load balancing is achieved with the partition in eight subdomains than with two subdomains and this leads to a better efficiency. In general, comparing performance results obtained with different number of subdomains is hazardous because of the different decomposition quality.

For this solver, communication speed may be a limiting factor to the parallel efficiency. Since the orthogonalization step of the parallel GMRES algorithm requires a number of parallel dot products proportional to N_{iter}^2, the communication latency must be very small. It has been measured to 90 μs on the Convex Exemplar. Therefore, the ratio of computations over communications is large and high efficiencies can be obtained. On the other hand, networks of workstations have larger latencies and resulting efficiencies may thus be very low (an efficiency of 60% was obtained with the 1200-element STEP mesh on a network of four Silicon-Graphics workstations (R3000 and R4000)).

5.3. Scalability

Tables III and IV illustrate various timing results for the STEP and SQUARE meshes as a function of the number of processors. Again, the focus is on the maximum subdomain factorization time T_{fact}, the interface solution time T_{interf}, and the total elapsed time T_{tot}. The value of T_{tot} includes the factorization and the interface times as well as the cost of preconditioning and some idle time due to synchronization requirements.

The factorization time T_{fact} decreases super-linearly with the number of subdomains due to the conjugate reduction in the bandwidth of the subdomain matrices and number of local variables. For a small number of subdomains, T_{fact} dominates the total solution time T_{tot}. It is reasonable to think that the interface solution time increases with the number of subdomains; however, T_{interf} goes to a minimum. The reduction in T_{interf} results from the reduction in the time for local matrix–vector products that overcomes the cost of the additional iterations when P increases. When the number of subdomains becomes large, the value of T_{fact} becomes negligible and the total solution time is governed by T_{interf}. The value of T_{tot} goes to a minimum, the location of which depends on the mesh size and the decomposition quality.

The number of iterations grows relatively slowly with the number of subdomains. This is crucial as far as the parallel scalability is concerned. In the symmetric case, Farhat and Roux [5] observe that the preconditioned interface problem is numerically scalable, i.e. the number of iterations grows extremely slowly as P increases.

Finally, the number of iterations remains almost constant for a fixed number of subdomains as the discretization of the domain becomes increasingly finer.

Table III, shows that N_{iter} grows from 31 iterations (first discretization with $h = 1/16$) to 33 ($h = 1/80$) for the solution of the interface problem with $P = 4$. This looks promising but a theoretical analysis is still needed to confirm the excellent scalability behavior of the preconditioned hybrid solver.

5.4. Influence of the decomposition

Performance results of the hybrid solver with various decompositions in four subdomains of the STEP mesh with 4800 elements are presented. The results illustrate the importance of the optimization step. Similar results have been obtained in other test cases.

Three different decompositions were used: (a) the initial decomposition provided by the Greedy algorithm (GR), (b) the decomposition optimized by the Simulated Annealing algorithm where all the subdomains have an equal number of elements (GR + SA/E), and (c) the decomposition created by our two-step strategy (GR + SA/F) in which the number of elements is adapted according to the frontal width of the subdomains.

The number of iterations for the interface problem depends on the interface size and the conditioning of the interface matrix. The latter is a function of the aspect ratio of the subdomains. Table V relates the performance of the parallel GMRES algorithm to the average aspect ratio ($\overline{\mathrm{AR}}$) and the interface size (N_I) of the decomposition. AR ranges between 0 and 1 (AR = 1 for a disk).

It can be observed that the use of optimized decompositions clearly reduces the interface solution time by a significant factor due to the reduction of N_I and the increase of $\overline{\mathrm{AR}}$. As expected, for equal interface sizes, the decomposition with the highest aspect ratio requires the smallest number of iterations. This improves interface solution time by 20%.

The value of $T_{\mathrm{interf}}/N_{\mathrm{iter}}$ is not influenced by the interface size nor by the communications between neighboring processors. Indeed, the time per iteration is governed by the computational cost of the matrix–vector product and by the global parallel operations needed to perform the dot products. In Table V, the reduction of the time per iteration is the result of

Table III. Performance of the parallel hybrid solver with the SQUARE mesh for the solution of a Navier–Stokes flow on the Convex Exemplar

N_e	P	$\max_s N_s$	N_I	T_{fact} (s)	N_{iter}	T_{jnterf} (s)	T_{tot} (s)
16×16	1	2467	0	13	0	0	14
	2	1275	83	2	19	4	8
	4	659	178	0.5	31	4	6
	8	351	368	0.2	49	5	5
	15	215	593	0.1	99	18	18
32×32	1	9536	0	503	0	0	512
	2	4851	163	42	19	25	109
	4	2467	338	12	28	15	40
	8	1275	688	2	51	13	19
	15	736	1062	1	70	15	19
48×48	1	21 219	0	2899	0	0	2930
	2	10 731	243	425	19	81	876
	4	5427	498	130	32	54	304
	8	2775	1008	10	53	34	58
	15	1533	1526	4	72	31	41
64×64	1	37 507	0	–	–	–	–
	2	18 915	323	1651	19	188	3460
	4	9539	658	544	32	124	1185
	8	4851	1328	69	53	78	217
	15	2658	2003	15	70	58	89
80×80	1	58 403	0	–	–	–	–
	2	29 492	403	–	–	–	–
	4	14 903	818	1600	33	289	3436
	8	7503	1648	206	54	150	603
	15	4082	2558	81	87	115	270

The local memory is insufficient to run the mono-domain problems on the 64×64 and 80×80 meshes as well as the two-domain problem on the 80×80 mesh.

Table IV. Performance of the parallel hybrid solver with the STEP mesh for the solution of a Navier–Stokes flow on the Convex Exemplar

N_e	P	$\max_s N_s$	N_I	T_{fact} (s)	N_{iter}	T_{interf} (s)	T_{tot} (s)
1200	1	11 213	0	657	0	0	669
	2	6878	203	82	20	33	185
	4	3885	345	24	26	20	66
	8	1733	666	4	41	13	22
	15	978	1124	2	58	14	17
2700	1	24 918	0	3686	0	0	3713
	2	15 594	308	681	22	130	1437
	4	8856	575	177	30	81	447
	8	3625	1016	30	63	56	117
	15	2032	1677	5	84	42	54
4800	1	44 023	0	–	–	–	–
	2	27 539	408	2013	22	299	4283
	4	16 136	825	733	32	220	1709
	8	6884	1396	168	78	157	496
	15	3485	2156	27	102	97	144

The local memory is insufficient to run the mono-domain problem with 4800 elements.

a better load balancing of the local matrix–vector products. Every local product requires the solution of a local system with matrix $\boldsymbol{K}^{(s)}$. Balancing the computational load of the factorizations serves to balance the loads for the local forward–backward substitutions.

A proper load balancing of the local factorizations is crucial to reducing the synchronization time before solving the interface system. In Table VI, the cost of the factorization is evaluated. The measured load balance factor $\mathrm{Lbf}_{\mathrm{eff}}$ is compared with the predicted load balance factor $\mathrm{Lbf}_{\mathrm{pred}}$, defined as

$$\mathrm{Lbf}_{\mathrm{eff}} = \frac{\overline{T_{\mathrm{fact}}}}{\max T_{\mathrm{fact}}}, \qquad \mathrm{Lbf}_{\mathrm{pred}} = \frac{\overline{L_s}}{\max L_s}, \tag{27}$$

Table V. Performance of the parallel GMRES algorithm for Navier–Stokes flow in the STEP mesh, obtained for various decompositions into four subdomains

	$\overline{AR}$	N_I	N_{iter}	T_{interf} (s)	T_{interf}/N_{iter} (s)
GR	0.24	830	40	282	7.05
GR+SA/E	0.24	323	34	267	7.85
GR+SA/F	0.32	325	32	220	6.87

Table VI. Statistics of the local factorizations for a Navier–Stokes flow in the STEP mesh, observed for various decompositions into four subdomains

	$\mathrm{Lbf}_{\mathrm{pred}}$	$\max F_s$	$\max L_s$	$\mathrm{Lbf}_{\mathrm{eff}}$	T_{fact} (s)
GR	0.74	71	6.0×10^6	0.76	1284
GR+SA/E	0.78	67	5.4×10^6	0.79	1167
GR+SA/F	0.92	54	3.5×10^6	0.95	733

Table VII. Timing results of the preconditioning step for two implementations obtained for the solution of Navier–Stokes flow with the SQUARE mesh (32×32 elements)

P	Impl.	max N_s	N_I	max $N_\mathrm{I}^{(s)}$	N_iter	$\boldsymbol{F}_\mathrm{I}\boldsymbol{\lambda}$ (s)	$\tilde{\boldsymbol{F}}_\mathrm{I}^{-1}\boldsymbol{\lambda}$ (s)	T_prepro (s)
2	**I1**	4851	163	163	19	5.7×10^{-1}	6.1×10^{-3}	132.5
	I2	id.	id.	id.	id.	id.	5.3×10^{-1}	42.2
4	**I1**	2467	338	169	28	2.4×10^{-1}	5.5×10^{-3}	49.2
	I2	id.	id.	id.	id.	id.	2.2×10^{-1}	12.8
8	**I1**	1275	688	215	51	8.9×10^{-2}	7.6×10^{-3}	17.2
	I2	id.	id.	id.	id.	id.	7.2×10^{-2}	3.9
15	**I1**	736	1062	201	70	3.4×10^{-2}	1.1×10^{-2}	3.5
	I2	id.	id.	id.	id.	id.	1.7×10^{-2}	1.3

where the load L_s is computed using Equation (25) and a bar denotes the average over all subdomains.

This result confirms the importance of the decomposition quality on the performance of the hybrid solver. Indeed, with the two-step strategy, the measured load balance increases from 0.76 to 0.95 while the factorization time is reduced by a factor of 1.7.

5.5. *Effect of preconditioning*

In Table VII, the computational requirements for the use of the preconditioner are presented, with the two implementations **I1** and **I2** proposed in Section 3.4. The elapsed time for the preprocessing step (T_prepro) and the matrix–vector products is shown, obtained with the SQUARE mesh (9539 unknowns).

For the first implementation **I1**, the preprocessing time is large while the cost of preconditioning $\tilde{\boldsymbol{F}}_\mathrm{I}^{-1}\boldsymbol{\lambda}$ is small compared with the matrix–vector product $\boldsymbol{F}_\mathrm{I}\boldsymbol{\lambda}$. The opposite holds for the second implementation. In fact, the preprocessing step includes the solution of $N_\mathrm{I}^{(s)}$ systems in **I1** that are not performed in **I2**. On the other hand, with **I2**, N_iter systems must be solved during the GMRES algorithm. As a consequence, the first implementation is preferable when

Table VIII. Timing results with or without preconditioner obtained for the solution of a Navier–Stokes flow problem with the SQUARE mesh (32×32 elements)

		Precond.				No precond.		
N_e	P	T_prepro (s)	T_interf (s)	N_iter	T_tot (s)	T_interf (s)	N_iter	T_tot (s)
1200	2	69	33	20	185	62	73	148
	4	22	20	26	66	53	131	80
	8	6	13	41	22	47	203	54
	15	2	14	58	17	126	347	128
2700	2	625	130	22	1437	298	105	984
	4	190	81	30	447	254	194	448
	8	31	56	63	117	169	313	210
	15	7	42	84	54	348	546	357
4800	2	1971	299	22	4283	1218	191	3268
	4	770	220	32	1709	925	290	1708
	8	170	157	78	496	560	485	756
	15	24	97	102	144	723	729	757

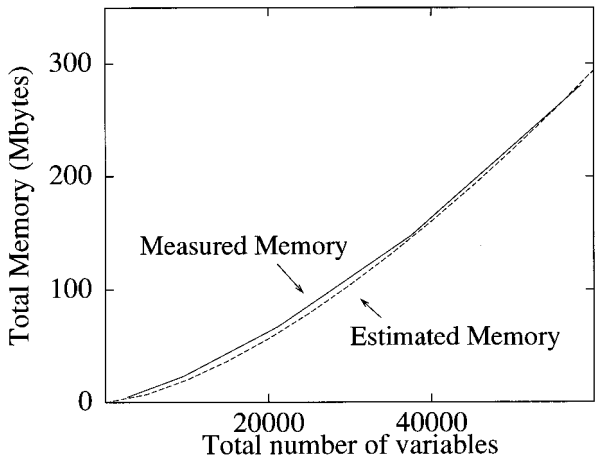

Figure 7. Evolution of the measured (——) and estimated (----) memory requirements for a Navier–Stokes flow defined on a SQUARE mesh decomposed 15 domains.

the number of iterations is expected to be large compared with the local number of interface nodes $N_{\mathrm{I}}^{(s)}$.

Table VIII shows the reduction in the number of iterations of the GMRES algorithm when the preconditioner is used. The preconditioner is not recommended for a small number of processors because T_{prepro} is too high. The situation is different for a larger number of processors because the computational cost of GMRES grows like the square of N_{iter}. Indeed, if P is large, the preprocessing time is relatively small and preconditioning becomes necessary.

5.6. Memory requirements

In the current implementation, the assembly of the local stiffness matrices is performed sequentially before the matrices are broadcast to the processors. Clearly, this constitutes a memory bottleneck that limits the size of the problems that can be run.

In Figure 7, the measured memory requirements is presented for the solution of the Navier–Stokes flow on the SQUARE mesh with 15 domains, as a function of the total number of variables N_v. For medium and large problems, the memory is driven by the size of the L and U factors of the local stiffness matrices while the size of the interface problem remains small (about N_{iter} descent directions of size N_{I} must be stored).

Every subdomain of the SQUARE mesh decomposed in 15 subdomains has about $N_v/15$ variables and the bandwidths of the local stiffness matrices are proportional to $\sqrt{N_v/15}$. Therefore, the total memory M_{est} needed for the local factorizations can be estimated by

$$M_{\mathrm{est}} \simeq \alpha N_v \times \sqrt{N_v}, \tag{28}$$

where $\alpha = 2 \times 10^{-5}$ is taken to fit the data of Figure 7. The comparison of the actual memory required (plain line) and the estimated memory (dash line) of Figure 7 shows a good agreement.

From Equation (28), it is seen that the largest problems to be solved on our Convex Exemplar SPP system with 2 Gbytes of memory have about 260 000 variables (2D Navier–Stokes problem on a square mesh). Note that in 3D, the size of the interface problem cannot be neglected any longer because the interface size becomes close to that of the local problems. In addition, a larger number of iterations are required before convergence is achieved.

6. DISCUSSION

Our experience with the proposed hybrid solver opens some questions about the stability and the accuracy of the formulation of the dual Schur method.

The construction of the null spaces of the local matrices requires special care. In this implementation, the size of the null space is determined by the number of zero pivots encountered during the factorization (or the number of pivots that are relatively small compared with the value of the previous pivots). For stiff problems, or when the entries in the matrix have different orders of magnitude, this test is unsatisfactory. On the contraction channel divided into 15 domains, some parameters had to be tuned by hand to obtain the proper null spaces. A more robust test using the singular value decomposition of the local stiffness matrix could be investigated. However, this would substantially increase the cost of the algorithm.

The next question deals with the iterative accuracy of the solution. We have no theoretical relation between the accuracy of the GMRES algorithm and that of the global finite element system. The accuracy of the global solution is one or two orders of magnitude lower. In addition, the presence of floating subdomains seems to limit the maximum accuracy of the solution of the global problem. Also, the accuracy of the factorization limits that of the matrix–vector products and of the GMRES algorithm itself. Here again, there is no test to provide a bound on the accuracy of the GMRES algorithm.

The preconditioner has been designed for the matrix $\boldsymbol{F}_{\mathrm{I}}$. Neither its validity nor its optimality have been proved for the *projected* interface matrix. Our experience suggests that this preconditioner is still useful with floating subdomains. The expression of a projected precondition interface matrix is under way.

Finally, we note that the hybrid solver may itself be used in a recursive fashion for the solution of the subdomain problems. This would lead to a scalable multilevel approach suitable for very large problem sizes on massively parallel computers. A different approach could also use an iterative method for the solution of the local subdomain problems, assuming that the null spaces can be correctly evaluated.

The goal of this work was to present an efficient parallel solver and to discuss its scalability. The comparison with other state-of-the-art parallel solvers (such as a parallel GMRES preconditioned with ILU) remains to be performed.

7. CONCLUSION

A parallel solver based on domain decomposition for the solution of general algebraic systems that arise from the finite element discretization has been presented. This solver is hybrid, in the sense that it combines a direct factorization of the local problems with an iterative treatment of the interface system by a parallel GMRES algorithm. An important feature of the solver is that it uses a set of Lagrange multipliers to enforce the continuity at the interface.

The decomposition of the finite element mesh has been formulated as a graph partitioning problem. A two-step approach is used where an initial decomposition is optimized by non-deterministic heuristics.

Parallel simulations of a Navier–Stokes flow problem carried out on the Convex Exemplar SPP system with 16 processors show that the use of optimized decompositions and a preconditioning step are the key issues to obtain high parallel efficiencies.

ACKNOWLEDGMENTS

The work of Denis Vanderstraeten is supported by the *Fonds National de la Recherche Scientifique* of the Belgian State. This paper presents research results of the Belgian Programme on Interuniversity Poles of Attraction, initiated by the Belgian State, Prime Minister's Office for Science, Technology and Culture. The scientific responsibility rests with its authors. The authors are thankful to Charbel Farhat (University of Colorado at Boulder) and F.-X. Roux (O.N.E.R.A., Paris) for helpful discussions and comments.

APPENDIX A. SOLUTION OF A CONSISTENT RANK DEFICIENT SYSTEM

In this section, a general expression is derived for the solution of

$$\boldsymbol{K}\boldsymbol{u} = \boldsymbol{f}, \tag{29}$$

when the matrix $\boldsymbol{K}$ of size $N \times N$ has rank $r < N$ and $\boldsymbol{f} \in \mathrm{Range}(\boldsymbol{K})$. Without loss of generality, $\boldsymbol{K}$ can be partitioned as

$$\boldsymbol{K} = \begin{pmatrix} \boldsymbol{K}_{11} & \boldsymbol{K}_{12} \\ \boldsymbol{K}_{21} & \boldsymbol{K}_{22} \end{pmatrix}, \tag{30}$$

where the submatrix $\boldsymbol{K}_{11}$ is of size r and has full rank. A block-Gaussian elimination of $\boldsymbol{K}_{21}$ reads

$$\begin{pmatrix} \boldsymbol{I} & \boldsymbol{0} \\ -\boldsymbol{K}_{21}\boldsymbol{K}_{11}^{-1} & \boldsymbol{I} \end{pmatrix} \begin{pmatrix} \boldsymbol{K}_{11} & \boldsymbol{K}_{12} \\ \boldsymbol{K}_{21} & \boldsymbol{K}_{22} \end{pmatrix} = \begin{pmatrix} \boldsymbol{K}_{11} & \boldsymbol{K}_{12} \\ \boldsymbol{0} & \boldsymbol{K}_{22} - \boldsymbol{K}_{21}\boldsymbol{K}_{11}^{-1}\boldsymbol{K}_{12} \end{pmatrix}. \tag{31}$$

Since rank $(\boldsymbol{K}) = \mathrm{rank}(\boldsymbol{K}_{11})$, it follows that

$$\boldsymbol{K}_{22} - \boldsymbol{K}_{21}\boldsymbol{K}_{11}^{-1}\boldsymbol{K}_{12} = 0. \tag{32}$$

Therefore, the generalized inverse of $\boldsymbol{K}$ is given by [13]:

$$\boldsymbol{K}^{+} = \begin{pmatrix} \boldsymbol{K}_{11}^{-1} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{pmatrix}. \tag{33}$$

A general expression for the solution $\boldsymbol{u}$ can be written as the sum of a particular solution of Equation (29) and a vector in the null space of $\boldsymbol{K}$. More precisely, we have

$$\boldsymbol{u} = \boldsymbol{K}^{+}\boldsymbol{f} + \boldsymbol{N}\boldsymbol{\alpha}, \tag{34}$$

where the matrix $\boldsymbol{N}$ contains the basis vectors of the null space and the vector $\boldsymbol{\alpha}$ specifies any linear combination of these vectors. Since the matrix $\boldsymbol{N}$ contains vectors of the null space, it must verify $\boldsymbol{K}\boldsymbol{N} = \boldsymbol{0}$. This holds for

$$\boldsymbol{N} = \begin{pmatrix} -\boldsymbol{K}_{11}^{-1}\boldsymbol{K}_{12} \\ \boldsymbol{I} \end{pmatrix}. \tag{35}$$

With the expression of $\boldsymbol{K}^{+}$ and $\boldsymbol{N}$, it is easy to verify that any $\boldsymbol{u}$ verifying Equation (34) is solution of (29).

APPENDIX B. PROJECTED INTERFACE SYSTEM

This appendix gives a new formulation of the interface system (13) that decouples the Lagrange multipliers $\boldsymbol{\lambda}$ from the rigid modes $\boldsymbol{\alpha}$. Equation (13) is rewritten as:

$$\begin{cases} \boldsymbol{F}_{\mathrm{I}}\boldsymbol{\lambda} - \boldsymbol{d} = \boldsymbol{G}_{\mathrm{I}}\boldsymbol{\alpha}, \\ \text{subject to } \boldsymbol{H}_{\mathrm{I}}^{T}\boldsymbol{\lambda} = \boldsymbol{e}. \end{cases} \tag{36}$$

Equation (36) expresses the fact that the quantity $\boldsymbol{F}_{\mathrm{I}}\boldsymbol{\lambda} - \boldsymbol{d}$ belongs to the range of $\boldsymbol{G}_{\mathrm{I}}$. This is reformulated by using the space orthogonal to Range($\boldsymbol{G}_I$)—i.e. for all $\boldsymbol{\mu}$ in Ker($\boldsymbol{G}_{\mathrm{I}}^{T}$), the vector $\boldsymbol{\lambda}$ verifies

$$(\boldsymbol{F}_{\mathrm{I}}\boldsymbol{\lambda} - \boldsymbol{d}, \boldsymbol{\mu}) = 0. \tag{37}$$

The symbol (., .) stands for the dot product. By decomposing the vector $\boldsymbol{\lambda}$ in a component in Ker($\boldsymbol{H}_{\mathrm{I}}^{T}$) and in a component in Range($\boldsymbol{H}_{\mathrm{I}}$), we write

$$\boldsymbol{\lambda} = \boldsymbol{\lambda}_0 \oplus \boldsymbol{\lambda}_1, \tag{38}$$

with

$$\boldsymbol{H}_{\mathrm{I}}^{T}\boldsymbol{\lambda}_0 = \boldsymbol{e}, \qquad \boldsymbol{H}_{\mathrm{I}}^{T}\boldsymbol{\lambda}_1 = 0. \tag{39}$$

Equation (37) then reads

$$(\boldsymbol{F}_{\mathrm{I}}\boldsymbol{\lambda}_1 - (\boldsymbol{d} - \boldsymbol{F}_{\mathrm{I}}\boldsymbol{\lambda}_0), \boldsymbol{\mu}) = 0. \tag{40}$$

At this point, we define a projector $\boldsymbol{P}_H$ onto the null space of $\boldsymbol{H}_{\mathrm{I}}^{T}$ and a projector $\boldsymbol{P}_G$ onto the null space of $\boldsymbol{G}_{\mathrm{I}}^{T}$. Since the vector $\boldsymbol{\lambda}_1$ (resp. $\boldsymbol{\mu}$) belongs to the null space of $\boldsymbol{H}_{\mathrm{I}}^{T}$ (resp. $\boldsymbol{G}_{\mathrm{I}}^{T}$), we have

$$\boldsymbol{\lambda}_1 = \boldsymbol{P}_H\boldsymbol{\lambda}', \qquad \boldsymbol{\mu} = \boldsymbol{P}_G\boldsymbol{\mu}', \tag{41}$$

where $\boldsymbol{\lambda}'$ and $\boldsymbol{\mu}'$ represent any vector in R^{N_I} Inserting Equation (41) into (40) leads to

$$(\boldsymbol{F}_{\mathrm{I}}\boldsymbol{P}_H\boldsymbol{\lambda}' - (\boldsymbol{d} - \boldsymbol{F}_{\mathrm{I}}\boldsymbol{\lambda}_0), \boldsymbol{P}_G\boldsymbol{\mu}') = 0, \tag{42}$$

which is transformed into

$$(\boldsymbol{P}_G^{T}\boldsymbol{F}_{\mathrm{I}}\boldsymbol{P}_H\boldsymbol{\lambda}' - \boldsymbol{P}_G^{T}(\boldsymbol{d} - \boldsymbol{F}_{\mathrm{I}}\boldsymbol{\lambda}_0), \boldsymbol{\mu}') = 0. \tag{43}$$

Since Equation (43) is verified for any vector $\boldsymbol{\mu}'$ in R^{N_I}, we must have

$$\boldsymbol{P}_G^{T}\boldsymbol{F}_{\mathrm{I}}\boldsymbol{P}_H\boldsymbol{\lambda}' = \boldsymbol{P}_G^{T}(\boldsymbol{d} - \boldsymbol{F}_{\mathrm{I}}\boldsymbol{\lambda}_0), \tag{44}$$

and the vector $\boldsymbol{\lambda}$ solution of (36) is given by

$$\boldsymbol{\lambda} = \boldsymbol{\lambda}_0 + \boldsymbol{P}_H\boldsymbol{\lambda}', \tag{45}$$

$$\boldsymbol{H}_{\mathrm{I}}^{T}\boldsymbol{\lambda}_0 = \boldsymbol{e}. \tag{46}$$

Finally, an expression for $\boldsymbol{\lambda}_0$ and $\boldsymbol{\alpha}$ is derived. Since $\boldsymbol{H}_{\mathrm{I}}$ has full column rank (see Reference [5] for details), the matrix $\boldsymbol{H}_{\mathrm{I}}^{T}\boldsymbol{H}_{\mathrm{I}}$ has full rank and is invertible. If $\boldsymbol{\lambda}_0$ is expressed as:

$$\boldsymbol{\lambda}_0 = \boldsymbol{H}_{\mathrm{I}}\boldsymbol{\beta}, \tag{47}$$

(recall that $\boldsymbol{\lambda}_0 \in$ Range($\boldsymbol{H}_{\mathrm{I}}$)), we have

$$(\boldsymbol{H}_{\mathrm{I}}^{T}\boldsymbol{H}_{\mathrm{I}})\boldsymbol{\beta} = \boldsymbol{e}, \tag{48}$$

or

$$\boldsymbol{\beta} - (\boldsymbol{H}_{\mathrm{I}}^T \boldsymbol{H}_{\mathrm{I}})^{-1} \boldsymbol{e}. \tag{49}$$

Given this last expression and (47), we have

$$\boldsymbol{\lambda}_0 = \boldsymbol{H}_{\mathrm{I}} (\boldsymbol{H}_{\mathrm{I}}^T \boldsymbol{H}_{\mathrm{I}})^{-1} \boldsymbol{e}. \tag{50}$$

The first equation of (36) can be written as

$$\boldsymbol{G}_{\mathrm{I}}^T \boldsymbol{G}_{\mathrm{I}} \boldsymbol{\alpha} = \boldsymbol{G}_{\mathrm{I}}^T (\boldsymbol{F}_{\mathrm{I}} \boldsymbol{\lambda} - \boldsymbol{d}). \tag{51}$$

Since the matrix $\boldsymbol{G}_{\mathrm{I}}^T \boldsymbol{G}_{\mathrm{I}}$ has also full rank, an expression for $\boldsymbol{\alpha}$ becomes

$$\boldsymbol{\alpha} = (\boldsymbol{G}_{\mathrm{I}}^T \boldsymbol{G}_{\mathrm{I}})^{-1} \boldsymbol{G}_{\mathrm{I}}^T (\boldsymbol{F}_{\mathrm{I}} \boldsymbol{\lambda} - \boldsymbol{d}). \tag{52}$$

The results of these algebraic manipulations have been summarized in Proposition 1 of Section 2.

REFERENCES

1. M.T. Heath, E. Ng and B.W. Peyton, 'Parallel algorithms for sparse linear systems', *SIAM Review*, **33**, 420–460 (1991).
2. P.E. Bjørdstad and O.B. Wildlund, 'Iterative methods for solving elliptic problems on regions partitioned into substructures', *SIAM J. Num. Anal.*, **23**, 1097–1120 (1986).
3. P. Le Tallec, 'Domain decomposition methods in computational mechanics', *Comput. Mech. Adv.*, **1**, 121–220 (1994).
4. C. Farhat and F.-X. Roux, 'A method of finite element tearing and interconnecting and its parallel solution algorithm', *Int. j. numer. methods eng.*, **32**, 1205–1227 (1991).
5. C. Farhat and F.-X. Roux. 'Implicit parallel processing in structural mechanics', *Comput. Mech. Adv.*, **2**, 1–124 (1994).
6. C.C. Paige and M.A. Saunders, 'Solution of sparse indefinite systems of linear equations', *SIAM J. Num. Anal.*, **12**, 617–629 (1975).
7. Y. Saad and M.H. Schultz, 'Conjugate gradient-like algorithms for solving nonsymmetric linear systems', *Math. Comput.*, **44**, 417–424 (1985).
8. D. Vanderstraeten, O. Zone, R. Keunings and L.A. Wolsey, 'Nondeterministic heuristics for automatic domain decomposition in direct finite element calculations', in R.F. Sincovec *et al.* (eds), *Proc. 6th SIAM Conf. on Parallel Proc. for Sci. Comput.*, 1993, pp. 929–932.
9. D. Vanderstraeten and R. Keunings, 'Optimized partitioning of unstructured finite element meshes', *Int. j. numer. methods eng.*, **38**, 433–450 (1995).
10. D. Vanderstraeten, C. Farhat, P.S. Chen, R. Keunings and O. Zone, 'A retrofit based methodology for the fast generation and optimization of large-scale mesh partitions: Beyond the minimum interface size criterion', *Comput. Methods Appl. Mech. Eng.*, **133**, 25–45 (1996).
11. S. Kirkpatrick, C.D. Gelatt and M.P. Vecchi, 'Optimization by simulated annealing', *Science*, **220**, 671–680 (1983).
12. A. Beguelin, J.J. Dongarra, G.A. Geist, R. Manchek and V.S. Sunderam, 'A user's guide to pvm parallel virtual machine', *Technical Report ORNL/TM-11826*, Oak Ridge National Laboratory, 1991.
13. C.R. Rao and S.K. Mitra, *Generalized Inverse of Matrices and its Applications*, Wiley, New York, 1971.
14. R. Barrett *et al.*, *Templates for the Solution of Linear Systems: Building Blocks for Iterative Solvers*, SIAM, Philadelphia, 1994.
15. Y. Saad and M.H. Schultz, 'GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems', *SIAM J. Sci. Comput.*, **7**, 856–869 (1986).
16. A. van der Sluis and H.A. van der Vorst, 'The rate of convergence of conjugate gradients', *Numerische Mathematik*, **48**, 543–560 (1986).
17. I. Fried, 'Condition of finite element matrices generated from nonuniform meshes', *AIAA J.*, **10**, 219–221 (1972).
18. C. Farhat, 'A simple and efficient automatic domain decomposer', *Comput. Struct.*, **28**, 579–602 (1988).
19. H.D. Simon, 'Partitioning of unstructured problems for parallel processing', *Comput. Syst. Eng.*, **2**, 135–148 (1991).

20. D. Vanderstraeten, R. Keunings and C. Farhat, 'Beyond conventional mesh partitioning algorithms and the minimum edge cut criterion: Impact on realistic applications', in D.H. Bailey *et al.* (eds), *Proc. 7th SIAM Conf. On Parallel Proc. for Sci. Comput.*, 1995, pp. 611–614.
21. D. Vanderstraeten, 'Algorithms for parallel finite element computations: mesh partitioning and hybrid solver,' *Ph.D. Thesis*, Université catholique de Louvain, Louvain-la-Neuve, Belgium, 1996.

