Asociación Argentina
de Mecánica Computacional

Mecánica Computacional Vol XXXIV, págs. 1235-1244 (artículo completo)
Sebastián Giusti, Martín Pucheta y Mario Storti (Eds.)
Córdoba, 8-11 Noviembre 2016

ABOUT THE PARALLEL VERSATILE IMPLEMENTATION OF FINITE ELEMENT TEARING AND INTERCONNECT METHODS

Alejandro Cosimo[a,b], **Alberto Cardona**[a] **and Daniel Rixen**[b]

[a]*CIMEC-Centro de Investigación de Métodos Computacionales (UNL/Conicet), ruta 168 s/n, Predio Conicet "Dr A. Cassano", 3000 Santa Fe, Argentina*

[b]*Institute of Applied Mechanics , Technische Universität München, Boltzmannstr. 15, 85748 Garching bei München, Germany*

Keywords: FETI, Parallel Implementation, PETSc, Object-Oriented Design

Abstract. The FETI (Finite Element Tearing and Interconnect) method is a well-established domain decomposition technique for solving large systems of equations which usually result from the discretisation of partial differential equations. Nowadays, many different versions of FETI are available, each one of them trying to exploit certain characteristics of the physical problem being solved. This broad family of FETI solvers makes necessary to think in informatic solutions which take care not only of the computational performance, but also of the flexibility of the code. Hard-coded solutions must be clearly avoided in order to give support to existing and future FETI versions which have different requirements concerning coarse spaces, preconditioners, direct and iterative solvers and projections. In this work, the issues related to satisfying those requirements in the parallel implementation of FETI methods are studied. An object-oriented design is adopted for ensuring the flexibility needed for the future development of additional FETI versions. A particular emphasis is placed in the implementation of these methods as extensions to the PETSc (Portable, Extensible Toolkit for Scientific Computation) library. A performance test is conducted in order to show the capabilities of the developed library.

1 INTRODUCTION

The FETI (Finite Element Tearing and Interconnect) method is a well-established domain decomposition technique for solving large systems of equations which usually result from the discretisation of partial differential equations. Among the various versions of the FETI method, the one-level FETI (FETI-1) (Farhat and Roux, 1991; Rixen, 2002), the second-level FETI (FETI-2) (Farhat et al., 2000) and the Dual-Primal FETI (FETI-DP) (Farhat et al., 2001) are the main versions from which further improvements have been proposed. In this work, we restrict our study to the family of the FETI-1 and FETI-2 methodologies which are characterised by enforcing the continuity of the solution between subdomains using Lagrange multipliers.

When applying the FETI-1 method to static or steady problems, a generalised inverse must be computed for floating subdomains. At a first glance, floating subdomains would seem to be an unwanted complication. However, the rigid body modes associated to floating subdomains conform a *natural* coarse problem which has the property of propagating the information globally, which makes the iterative solver for the interface problem to converge faster. But most importantly, thanks to this feature, it can be ensured that the FETI-1 method is *numerically scalable* (the number of iterations grows at most weakly with the number of degrees of freedom (DOFs) per subdomain (Farhat et al., 1995)). The FETI-2 methods are based on augmenting the FETI-1 method with an additional *optional* constraint that must be satisfied at each iteration when solving the interface problem (Farhat et al., 2000). It is shown by Farhat et al. (1995) that such additional constraint improves the scalability of elastodynamics problems.

Numerical scalability is needed in order to ensure parallel scalability. Another aspect of the parallel scalability is the correct implementation of these methods. Therefore, the aim of this work is to study the computational implementation of FETI methods in parallel environments. More specifically, distributed memory architectures will be considered. The broad family of FETI solvers makes necessary to think in informatic solutions which take care not only of the computational performance, but also of the flexibility of the code. Hard-coded solutions must be clearly avoided in order to give support to existing and future FETI versions, which have different requirements concerning coarse spaces, preconditioners, direct and iterative solvers and projections. An object-oriented design is adopted for ensuring the flexibility and extensibility needed for the future development of additional FETI versions. In this work, it is proposed to implement the different FETI methods as extensions to the PETSc library (Balay et al., 2015), thus inheriting many of the powerful features that this library for scientific computing has.

The work is organised as follows. In Section 2, the basic concepts of the FETI method are presented. In the following Section, the design of a flexible computational framework for implementing FETI methods is introduced. Different aspects of the assembling and solution of the FETI-1 coarse problem in parallel environments are discussed in Section 4. In Section 5, a performance test is conducted in order to show the capabilities of the developed library. Finally, in the last Section the conclusions of this work are given.

2 BASIC CONCEPTS OF THE FETI METHOD

The basic concepts of the FETI method are here introduced. A more exhaustive presentation of these topics can be found in the state-of-the-art (Farhat et al., 1998; Rixen and Farhat, 1999; Rixen et al., 1999; Spillane and Rixen, 2013). It is assumed that the matrix of the system to be solved is Symmetric Positive Definite (SPD), which is provided, for instance, by a Finite Element Code. Then, we deal with the following problem:

$$Au = f, \tag{1}$$

where A is the system matrix, u is the unknown and f is the right hand side. In the FETI-1 method this problem is solved by decomposing the domain into N_s subdomains and using Lagrange multipliers λ for imposing the continuity of the solution, which writes

$$A^{(s)}u^{(s)} = f^{(s)} - {B^{(s)}}^T\lambda, \quad s = 1 \dots N_s \tag{2}$$

$$\sum_{s=1}^{N_s} B^{(s)}u^{(s)} = 0, \tag{3}$$

where $B^{(s)}$ is a signed Boolean matrix such that $B^{(s)}u^{(s)}$ is the signed restriction of $u^{(s)}$ to the interface of subdomain s. Then, $A^{(s)}$, $u^{(s)}$ and $f^{(s)}$ are the restrictions to subdomain s of A, u and f, respectively. Assuming N_f floating subdomains, the interface problem is given by

$$\begin{bmatrix} F & -G \\ G^T & 0 \end{bmatrix} \begin{bmatrix} \lambda \\ \alpha \end{bmatrix} = \begin{bmatrix} d \\ e \end{bmatrix} \tag{4}$$

where

$$F = \sum_{s=1}^{N_s} B^{(s)}{A^{(s)}}^+{B^{(s)}}^T, \quad G = [B^{(1)}R^{(1)} \dots B^{(N_f)}R^{(N_f)}]^T, \quad \alpha = [{\alpha^{(1)}}^T \dots {\alpha^{(N_s)}}^T]^T$$

$$d = \sum_{s=1}^{N_s} B^{(s)}{A^{(s)}}^+f^{(s)}, \quad e = [{f^{(1)}}^T R^{(1)} \dots {f^{(N_f)}}^T R^{(N_f)}]^T$$

with $R^{(s)}$ as the rigid body modes of subdomain s and ${A^{(s)}}^+$ as the generalised inverse of $A^{(s)}$, which is computed using a direct solver. The restriction of the displacement field to subdomain s, $u^{(s)}$, is computed as

$$u^{(s)} = {A^{(s)}}^+(f^{(s)} - {B^{(s)}}^T\lambda) + R^{(s)}\alpha^{(s)}. \tag{5}$$

The unknowns λ and α are obtained by solving the interface problem iteratively using the Conjugate Gradient (CG) method. However, due to the restriction $G^T\lambda = e$, the CG method can not be directly applied to compute λ. In order to circumvent this problem, see (Farhat et al., 1994b), a projection P is introduced:

$$P = I - QG(G^TQG)^{-1}G^T, \tag{6}$$

where Q is a SPD matrix. Then, λ is expressed as $\lambda = \lambda_0 + P\bar{\lambda}$ where $\lambda_0 = QG(G^TQG)^{-1}e$, and the interface problem becomes

$$P^TFP\bar{\lambda} = P^T(d - F\lambda_0), \tag{7}$$

$$\alpha = (G^TQG)^{-1}G^TQ(F\lambda - d). \tag{8}$$

With the interface problem written like that, the CG method can be applied in order to compute $\bar{\lambda}$. However, in practice the Preconditioned CG must be used and the first equation of the problem writes

$$P\bar{F}^{-1}P^TFP\bar{\lambda} = P\bar{F}^{-1}P^T(d - F\lambda_0), \tag{9}$$

where $\bar{F}^{-1}$ is the preconditioner.

Two preconditioners can be mentioned (Farhat et al., 1994b). The mathematically optimal Dirichlet preconditioner $\bar{F}^{D^{-1}}$, which ensures numerical scalability, and the Lumped preconditioner $\bar{F}^{L^{-1}}$, which is a cheaper approximation of the previous one but it is not optimal. These preconditioners are given by

$$\bar{F}^{D^{-1}} = \sum_{s=1}^{N_s} W^{(s)} B^{(s)} \begin{bmatrix} 0 & 0 \\ 0 & S_{bb}^{(s)} \end{bmatrix} B^{(s)^T} W^{(s)}, \text{ and } \bar{F}^{L^{-1}} = \sum_{s=1}^{N_s} W^{(s)} B^{(s)} \begin{bmatrix} 0 & 0 \\ 0 & A_{bb}^{(s)} \end{bmatrix} B^{(s)^T} W^{(s)},$$

where subindices i and b denote internal and interface DOFs, respectively, $S_{bb}^{(s)} = A_{bb}^{(s)} - A_{ib}^{(s)^T} A_{ii}^{(s)^{-1}} A_{ib}^{(s)}$ is the Schur complement of subdomain s, and $W^{(s)}$ is a diagonal scaling matrix. Examples of scaling techniques are the multiplicity scaling and the K-scaling or super-lumped scaling (Rixen and Farhat, 1999).

In the expression of the projection P, Eq. (6), matrix Q is generally taken as the identity in order to avoid extra-computations. However, other expressions for Q are possible, for instance, Q can be taken as the Dirichlet or Lumped preconditioner (Bhardwaj et al., 2000). It is important to note that the application of the projection P involves solving for vector ξ the *global* problem $G^T Q G \xi = \eta$, which is known as *coarse problem*.

A FETI methodology for solving time dependent problems is presented by Farhat et al. (1994a). Due to the fact that in this case the system matrix A is a linear combination of the mass and stiffness matrices, no projection or coarse problem is involved when extending FETI-1 to such problems. As pointed out by Farhat et al. (1995), by losing the coarse problem the property of numerical scalability is also lost. In order to recover this property, they proposed to build a coarse problem by introducing an *optional* constraint that must be satisfied at each iteration k of the iterative solver. This leads to a projection of the form

$$P_C = I - C(C^T F C)^{-1} C^T F, \tag{10}$$

whose application involves solving the coarse problem $C^T F C \xi = \eta$. In the work of Farhat et al. (1998), it is argued that a good choice for C are the rigid body modes associated to each subdomain when considering only the static response of the structure, *i.e.*, by taking C as $C = G$. However, other alternatives for C are possible. For instance, by following the same reasoning presented by Spillane and Rixen (2013) and as investigated by Leistner et al. (2016), C may be built from the computation of the GENEO (Schwarz-Generalised Eigenvalues in the Overlaps) modes of each of the subdomains.

It should be observed that introducing an optional constraint leads to a second-level algorithm, better known as FETI-2 (Farhat et al., 2000). For dynamics problems, the matrix A is a linear combination of the mass and stiffness matrices, and the interface problem reduces to $F\lambda = d$. Therefore, for this kind of problem which does not involve any natural coarse problem, the main differences with the FETI-1 algorithm are that the projection P is now given by the projection P_C and that λ_0 takes the expression $\lambda_0 = C(C^T F C)^{-1} C^T d$.

3 DESIGN OF A FLEXIBLE COMPUTATIONAL FRAMEWORK FOR FETI

In Section 2, the basic concepts behind the FETI-1 and FETI-2 methodologies were introduced with the main purpose of identifying a minimum set of requirements that a computational framework implementing these methods must satisfy. In order to ensure the flexibility and extensibility required by such computational framework, an object-oriented design is adopted. Many of the basic requirements of the current project are already satisfied by the PETSc library

(Balay et al., 2015). Therefore, it is proposed to implement the FETI computational framework as extensions to PETSc. PETSc is a High Performance library for scientific computing which is written in C using an object-oriented approach.

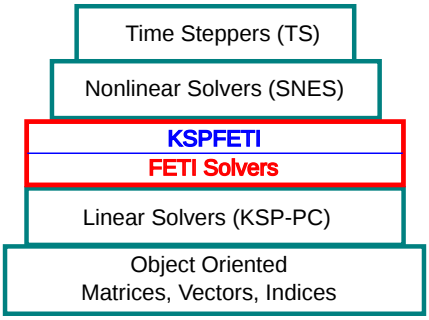


Figure 1: Graphical representation of the PETSc structure and the location of the FETI extensions within this structure. In our library, FETI methods are implemented as linear equations solvers. From a general point of view, every version of FETI inherit from the base class FETI. The KSP solver type KSPFETI is incorporated in order to interface the implemented FETI solvers to the existing PETSc structure and algorithms.

From the concepts presented in Section 2, the following requirements are inferred:

1. Direct solvers are needed for solving the coarse and local problems, and for computing the Schur complements, when required.

2. Iterative solvers are needed for solving the interface problem.

3. Three different types of projections can be identified from the implementation point of view. From the expression for P, Eq. (6), two cases can be distinguished. One with Q as the identity and the other in which Q has another expression. The third case is given by the projection P_C, Eq. (10). Generally, the adopted projection will determine how to compute λ_0, therefore this must be also taken into account in the design.

4. Support for the construction of different coarse spaces must be implemented.

5. Support for scaling operations must be given.

6. Preconditioners for the interface problem are needed. More specifically, at least support for the Dirichlet and Lumped preconditioners must be given.

7. In the case of conforming subdomains, the user performs the domain decomposition of the problem and provides the local to global mapping of the numbering of the subdomains' DOFs. With this information, the library must be able to automatically build the data structures needed by the implementation, such as, for example, the Lagrange multipliers connecting the subdomains.

An implementation of the FETI-DP method as a preconditioner is available in PETSc, which is based on the BDDC PETSc preconditioner. It must be mentioned that part of the code of the FETI-DP preconditioner of PETSc was re-used in the implementation of the design that will be presented in what follows. For instance, the last point mentioned in the list of requirements is based on the implementation of the FETI-DP PETSc preconditioner.

In Fig. 1, it can be observed how it is proposed to extend the PETSc structure in order to give support to FETI solvers. From a high level point of view, all FETI solvers inherit from the base class FETI which is interfaced to the PETSc structure as a Krylov Subspace Solver (KSP) solver, named KSPFETI. In this way, the developed FETI solvers can be used from higher level components of PETSc which are already available, such as Scalable Non-Linear Solvers (SNES) and Time Steppers (TS). From the adopted structure, it should be observed that direct and iterative solvers are available to the FETI object as needed. That is why the FETI object is placed on top of the PETSc KSP and Preconditioner (PC) objects.

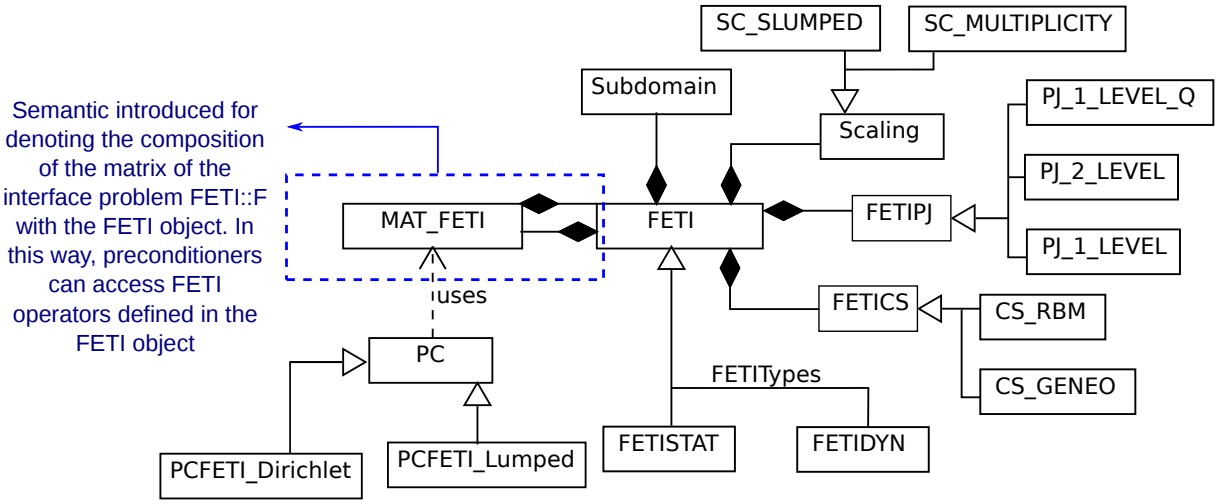


Figure 2: Simplified class diagram of the design of the FETI computational framework.

In Fig. 2, a simplified class diagram of the design proposed for implementing a computational framework for developing FETI methods is shown. As it can be observed, the base class FETI is composed by objects whose main purpose is to satisfy the requirements introduced before. The class Subdomain has no great importance from the design point of view, it is mainly intended to have as data members the local system matrix and right hand side, and an index set with the local to global mapping of the numbering of the subdomains' DOFs. The different FETI scaling methodologies are modelled by the base class Scaling and its derived classes. Support for FETI preconditioners, mainly the lumped and Dirichlet preconditioners, are implemented as new PETSc preconditioners (PC). The class FETI Projection (FETIPJ) is introduced to give support to the different types of projections needed by FETI. The base class FETIPJ defines interfaces to operations dealing with the coarse problem, such as its assembling and its resolution, and with the application of the projection to a vector. There are other operations, such as computing the initial value λ_0, whose expression is strictly linked to the type of projection. That is why, the base class FETIPJ also defines interfaces for this kind of operations. Specific implementations of these methods' interfaces are found in class inheriting from FETIPJ. Three types of projections are considered in the design, PJ_1_LEVEL_Q which implements projection P, Eq. (6), PJ_1_LEVEL which implements the same projection P taking Q as the identity, and PJ_2_LEVEL which implements the projection P_C given by Eq. (10). The FETI Coarse Space (FETICS) class is introduced in order to be able to handle coarse spaces in a flexible manner. This is a base class which defines interfaces to operations for building the coarse space. Currently, coarse spaces which are built from Rigid Body Modes (CS_RBM) and from the GENEO modes (CS_GENEO) are considered. However, there is no limitation regarding the coarse space type, and, therefore, new coarse spaces can be added to

the implementation.

Specific versions of the FETI method inherit from the base class FETI. In the diagram, the classes FETISTAT and FETIDYN are shown, which implement the FETI method for static and dynamic problems, respectively. Therefore, new FETI methods are modelled by extending the behaviour of the base class FETI. However, the classes FETISTAT and FETIDYN are not so complex from the design point of view, because they merely configure the considered FETI method with the corresponding default projections and coarse spaces, which eventually can be modified at run-time. For instance, when the user specifies the FETIDYN method, the projection type is set to a second level projection (PJ_2_LEVEL) and the coarse space is set to rigid body modes (CS_RBM). On the other hand, if the user specifies the FETISTAT method, the projection type is set to a first level projection (PJ_1_LEVEL).

4 ASSEMBLING AND SOLUTION OF THE FETI-1 COARSE PROBLEM

It should be noted that the existence of a coarse problem is the key ingredient to guarantee the numerical scalability of FETI methods. Therefore, an important aspect of the parallel implementation of FETI methods is the assembling and solution of the involved coarse problem. The strategy adopted in this paper for assembling and solving the coarse problem is based in the work of Roux and Farhat (1998), where the coarse problem is solved redundantly with a direct solver in each of the subdomains. In what follows the parallel implementation of the application of the projection P to a vector λ_g is explained. It is supposed that P is given by Eq. (6) with Q taken as the identity, and it is assumed that rigid body modes are used as basis for the coarse space. The MPI standard (Message Passing Interface Forum, 2015) is used for implementing the algorithms here introduced.

It must be noted that the coarse problem $(G^T G)\xi = \eta$ needs to be solved when applying the projection P to a global vector λ_g, *i.e.* $P\lambda_g = \lambda_g - G(G^T G)^{-1} G^T \lambda_g$. As proposed by Roux and Farhat (1998), this problem is redundantly solved in each subdomain using a direct solver. Therefore, the assembling of the coarse problem is done collaboratively between every processor in the system. The result is that, at the end, every processor will have a copy of the matrix of the coarse problem. The central problem in the assembling process is how to assemble the products ${G^{(s)}}^T G^{(k)}$, for floating subdomains s and k. In the proposed algorithm subdomain s assembles the products ${G^{(s)}}^T G^{(k)}$ for every subdomain k. However, it should be noted that the products ${G^{(s)}}^T G^{(k)}$ are different from zero only if subdomains s and k are neighbours. This last observation drives to conclude that the coarse problem has a sparse structure. Then, from the perspective of processor or subdomain s the steps to follow in order to assemble the coarse problem are the following:

1. Compute local $G^{(s)}$.
2. Send $G^{(s)}$ to neighbours and receive from them theirs local $G^{(k)}$. Save a local copy of every received $G^{(k)}$.
3. Multiply the $G^{(k)}$ received from your neighbour and the subdomains' local $G^{(s)}$, by ${G^{(s)}}^T$.
4. Exchange between every processor in the system the computed and assembled rows which are defined by the product ${G^{(s)}}^T G^{(k)}$.

Once the coarse problem has been assembled, it is factorised. Then, the application of projection P to the vector λ_g is obtained by following these steps:

1. Compute $\beta = G^T \lambda_g$. In each floating subdomain s the product between the local ${G^{(s)}}^T$ and λ_g is computed. Then, β is built in each subdomain by gathering the results of all subdomains.

2. Compute $\gamma = (G^T G)^{-1} \beta$ redundantly in each subdomain.

3. Compute $\xi^{(s)} = G\gamma$. This takes the form $\xi^{(s)} = G^{(s)}\gamma^{(s)} + G^{(k_1)}\gamma^{(k_1)} + \cdots + G^{(k_i)}\gamma^{(k_i)}$, where $k_1 \cdots k_i$ are neighbours of subdomain s.

4. Compute in each subdomain s, $(P\lambda_g)^{(s)} = \lambda_g^{(s)} - \xi^{(s)}$.

5 APPLICATION EXAMPLE

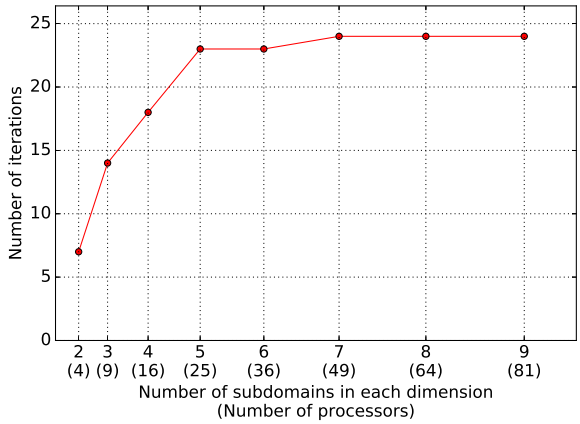
In this section, the performance of the developed library is studied by solving the 2D Poisson equation using Finite Differences with $f = -2$ as source term, and the temperature field imposed to $T = -5$ on the boundary defined by $x = 0$. The domain of analysis is decomposed into square subdomains with a side length $H = 1$. Each of these subdomains have equal mesh size h, which is equal to $1/610$ as 610 divisions in x and y dimensions are specified for the discretisation of each subdomain. In this test, we analyse problems of increasing sizes by adding subdomains simultaneously in x and y dimensions.

The numerical scalability and the parallel efficiency are the parameters to be measured in this study. The FETI method for static or steady problems is adopted, which is identified in the developed library by the FETI object class FETISTAT. Rigid body modes are used to build the coarse space, and the Projected Conjugate Gradient without full-reorthogonalisation is adopted as iterative solver for solving the interface problem, where Eq. (6) with Q as the identity is used as the projection operator. The generalised inverse of the local matrices and the rigid body modes are computed with the MUMPS library (Amestoy et al., 2000). Redundant Lagrange multipliers and the Dirichlet preconditioner with multiplicity scaling are used. The criterion for checking the convergence is based on the projected residual $w^k = P^T r^k$. It is considered that at iteration k the solver converged if $||w^k|| > \epsilon ||w^0||$, where $\epsilon = 10^{-8}$ in this example. The involved tests are run in the cluster Seshat (Cluster SESHAT, 2016).

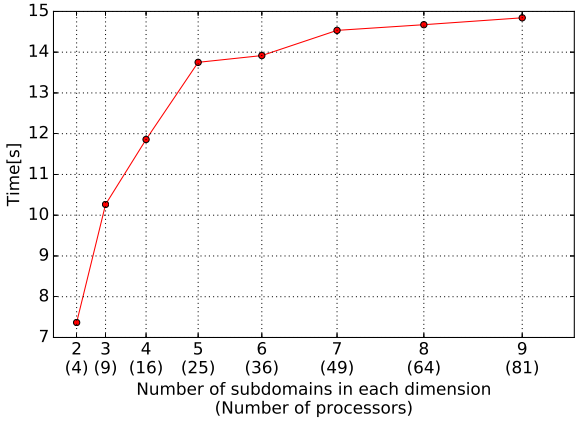
In Fig. 3a, the numerical scalability for the current example can be observed. As expected by theoretical studies, the number of iterations of the iterative solver for the interface problem increases only weakly with the number of subdomains. In Fig. 3c, the running times for the complete simulation of the different cases are shown. It should be noted that these times include the times for assembling the system matrix and the solving time of FETI. From the results, it can be considered that the number of iterations stabilises for a number of subdomains per dimension greater or equal to 5. This is why, the time taken for the problem where 5 subdomains per dimension are used is adopted as reference time for computing the parallel efficiency of the implementation. In Fig. 3b, the obtained results are shown. As it can be appreciated, for the largest problem the efficiency is close to 0.93, factor which contributes to the reliability of the implementation.

6 CONCLUSIONS

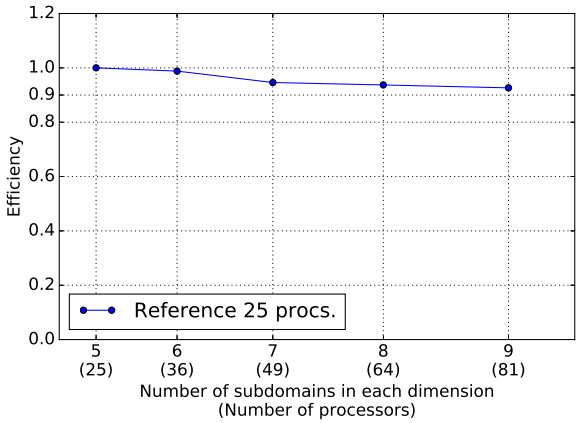
This work begins by introducing the basics notions of the FETI method with the purpose of understanding which are the requirements that a flexible computational framework must satisfy. From that introduction, it is made evident that an object-oriented approach is necessary to ensure the flexibility and extensibility needed by such a computational framework. It is therefore



(a) Numerical scalability.



(b) Running times.



(c) Parallel efficiency.

Figure 3: Results obtained with the FETI method for the 2D Poisson equation for problems of increasing sizes.

proposed to implement FETI methods as extensions to the PETSc library, thus inheriting many of the powerful features that this library for scientific computing has. A design for the library is proposed, where the main components of the FETI method, such as the projections, coarse spaces and preconditioners, are modelled by classes which ensure the re-usability and flexibility of the code. An application example is solved in order to test the correctness of the implementation. It is shown that the implemented method is numerically and parallel scalable, with an efficiency close to 0.93 for the largest problem considered in this test. Future work will be focused on improving the design that is proposed in this paper. Other versions of FETI need to be analysed in order to identify new requirements, and new features must be implemented, such as using the primal residual for checking the convergence of the solver used for the interface problem.

REFERENCES

Amestoy P., Duff I., and L'Excellent J.Y. Multifrontal parallel distributed symmetric and unsymmetric solvers. *Computer Methods in Applied Mechanics and Engineering*, 184(2-4):501 – 520, 2000.

Balay S., Abhyankar S., Adams M.F., Brown J., Brune P., Buschelman K., Dalcin L., Eijkhout V., Gropp W.D., Kaushik D., Knepley M.G., McInnes L.C., Rupp K., Smith B.F., Zampini S., and Zhang H. PETSc Web page. http://www.mcs.anl.gov/petsc, 2015.

Bhardwaj M., Day D., Farhat C., Lesoinne M., Pierson K., and Rixen D. Application of the FETI method to ASCI problems: Scalability results on a thousand-processor and discussion of highly heterogeneous problems. 47(1-3):513–536, 2000.

Cluster SESHAT. http://www.cimec.org.ar/c3/seshat/equipos.php, 2016.

Farhat C., Chen P.S., and Mandel J. A scalable Lagrange multiplier based domain decomposition method for time-dependent problems. *International Journal for Numerical Methods in Engineering*, 38(22):3831–3853, 1995.

Farhat C., Chen P.S., Risler F., and Roux F.X. A unified framework for accelerating the convergence of iterative substructuring methods with Lagrange multipliers. *International Journal for Numerical Methods in Engineering*, 42(2):257–288, 1998.

Farhat C., Crivelli L., and Roux F.X. A transient FETI methodology for large-scale parallel implicit computations in structural mechanics. *International Journal for Numerical Methods in Engineering*, 37(11):1945–1975, 1994a.

Farhat C., Lesoinne M., Le Tallec P., Pierson K., and Rixen D. FETI-DP: A dual-primal unified FETI method part I: A faster alternative to the two-level FETI method. *International Journal for Numerical Methods in Engineering*, 50(7):1523–1544, 2001.

Farhat C., Mandel J., and Roux F.X. Optimal convergence properties of the FETI domain decomposition method. *Computer Methods in Applied Mechanics and Engineering*, 115(3-4):365–385, 1994b.

Farhat C., Pierson K., and Lesoinne M. The second generation FETI methods and their application to the parallel solution of large-scale linear and geometrically non-linear structural analysis problems. *Computer Methods in Applied Mechanics and Engineering*, 184:333–374, 2000.

Farhat C. and Roux F.X. A method of finite element tearing and interconnecting and its parallel solution algorithm. *International Journal for Numerical Methods in Engineering*, 32(6):1205–1227, 1991.

Leistner M., Cosimo A., and Rixen D. Performance and scalability of FETI methods for heterogeneous dynamic problems with different coarse-grids. 2016. Oral presentation at the VII European Congress on Computational Methods in Applied Sciences and Engineering.

Message Passing Interface Forum. MPI: A Message-Passing Interface Standard. Version 3.1. Technical Report, Knoxville, TN, USA, 2015.

Rixen D. Extended preconditioners for the FETI method applied to constrained problems. *International Journal for Numerical Methods in Engineering*, 54(1):1–26, 2002.

Rixen D.J. and Farhat C. A simple and efficient extension of a class of substructure based preconditioners to heterogeneous structural mechanics problems. *International Journal for Numerical Methods in Engineering*, 44(4):489–516, 1999.

Rixen D.J., Farhat C., Tezaur R., and Mandel J. Theoretical comparison of the FETI and algebraically partitioned FETI methods, and performance comparisons with a direct sparse solver. *International Journal for Numerical Methods in Engineering*, 46(4):501–533, 1999.

Roux F.X. and Farhat C. Parallel implementation of direct solution strategies for the coarse grid solvers in 2-level FETI method. *Contemporary Mathematics*, 218:158–173, 1998.

Spillane N. and Rixen D. Automatic spectral coarse spaces for robust Finite Element Tearing and Interconnecting and Balanced Domain Decomposition algorithms. *International Journal for Numerical Methods in Engineering*, 95(11):953–990, 2013.