Mecánica Computacional Vol. XVI
compilado por G. Etse y B. Luccioni
Tucumán, Septiembre de 1996

SOLVING NONLINEAR TRANSIENT PROBLEMS BY DOMAIN DECOMPOSITION METHODS

V. E. Sonzogni
INTEC, CONICET, UNIVERSIDAD NACIONAL DE LITORAL
Güemes 3450, 3000, Santa Fe, Argentina
e-mail: sonzogni@intec.unl.edu.ar, Tel/Fax: 54(42)556673

RESUMEN

En este trabajo se trata la solución de problemas de respuesta transitoria no lineal de estructuras por métodos de descomposición del dominio. Estas técnicas resultan convenientes para problemas de grandes dimensiones, aún para un procesamiento secuencial. En este trabajo se apunta a la solución en paralelo, y para ésto la descomposición del dominio puede ser un algoritmo interesante de resolución, sobre todo en un entorno de computadoras paralelas de memoria local o para la resolución con redes de computadoras secuenciales y/o paralelas.

ABSTRACT

Development of algorithms for parallel solution of nonlinear problems of transient mechanical response is considered in this work. Domain decomposition techniques are chosen as they result in efficient tools for solving large problems in engineering analysis. They lead to high granularity tasks and result in powerful ways of performing parallel computations. They are also suitable for applications on different hardware architectures and/or different programming models. Algorithms reported for structural mechanics are presented, as well as the characteristics of nonlinear finite element programs with regard to their parallel execution capability.

1. INTRODUCTION

Domain decomposition techniques are efficient tools for solving large problems in engineering analysis. A first division arises whether the analysis domain is decomposed in *overlapping* subdomains (Schwarz methods) or in *non overlapping* subdomains (i.e. Schur Complement method). The latter has been widely used in structural mechanics.
Despite their usefulness in dividing large computational tasks in the framework of traditional computer architecture, these methods are particularly attractive regarding parallel computations. The development of parallel algorithms for non linear structural mechanics is being currently accomplished in the Computational Mechanics Group of INTEC. The present is a progress report where domain decomposition techniques are presented, specially those with proved efficiency in finite element computations. The main characteristics of a non linear finite element code are discussed in connection with parallel computation.

2. THE SCHUR COMPLEMENT

Domain decomposition techniques provide efficient solution methods for mechanical problems. They are based on splitting the analysis domain Ω into a number NSD of non overlapping subdomains (figure 1). Let Ω^s denote each subdomain and Γ_i^s $(i = 1, 3)$ their boundaries. Γ_1 states for boundaries with kinematical conditions, Γ_2 for those with mechanical conditions and Γ_3 for frontiers with other subdomains. With this notation the whole domain is

$$\Omega = \bigcup_{s=1}^{NSD} \Omega^s \bigcup \Gamma_3 \tag{1}$$

where

$$\Gamma_3 = \bigcup_{s=1}^{NSD} \Gamma_3^s \tag{2}$$

is the inter-subdomains frontier.

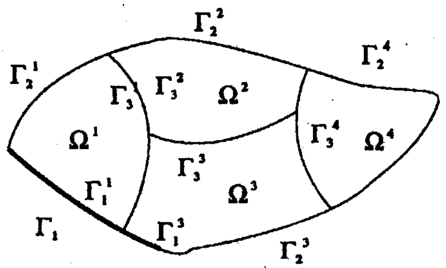

Figure 1: Domain decomposition

Applying the usual displacement based finite element method to solve a static mechanical problem the following system is get

$$\mathbf{K}\mathbf{u} = \mathbf{f} \tag{3}$$

At each subdomain (comprising Ω^s and Γ_3^s) the subdomain stiffness matrix $\mathbf{K}^s$, displacement vector $\mathbf{u}^s$ and force vector $\mathbf{f}^s$ may be constructed. We can realize partitioning these matrices into a group of internal d.o.f. $\mathring{\mathbf{u}}^s$ and a group of interface d.o.f. $\bar{\mathbf{u}}^s$. The subdomain stiffness matrix may be written

$$\mathbf{K}^s = \begin{bmatrix} \mathring{\mathbf{K}}^s & \vec{\mathbf{K}}^s \\ \vec{\mathbf{K}}^{s,T} & \bar{\mathbf{K}}^s \end{bmatrix} \tag{4}$$

and the displacement and force vectors

$$\mathbf{u}^s = \begin{bmatrix} \mathring{\mathbf{u}}^s \\ \bar{\mathbf{u}}^s \end{bmatrix} \quad \text{and} \quad \mathbf{f}^s = \begin{bmatrix} \mathring{\mathbf{f}}^s \\ \bar{\mathbf{f}}^s \end{bmatrix} \tag{5}$$

The upper symbol $\mathring{\Box}$ means *internal* d.o.f., $\bar{\Box}$ means *interface* d.o.f. and $\vec{\Box}$ refers to interaction between internal and interface d.o.f..
On the other hand, if we collect the contribution of all subdomains to the interface d.o.f., we can write

$$\mathbf{K}_I = \mathop{\mathbf{A}}_{s=1}^{NSD} \bar{\mathbf{K}}^s \tag{6}$$

as the assembled stiffness matrix for the whole interface problem, $\mathbf{u}_I$ being the displacement vector associated to it. Matrices with subscript I have the size of the whole interface problem.

Recalling the whole problem, the stiffness matrix may be written

$$\mathbf{K} = \begin{bmatrix} \mathring{\mathbf{K}}^1 & 0 & \dots & 0 & \dots & \vec{\mathbf{K}}_I^1 \\ 0 & \mathring{\mathbf{K}}^2 & \dots & 0 & \dots & \vec{\mathbf{K}}_I^2 \\ \vdots & \vdots & \ddots & \vdots & & \vdots \\ 0 & 0 & \dots & \mathring{\mathbf{K}}^s & \dots & \vec{\mathbf{K}}_I^s \\ \vdots & \vdots & & \vdots & \ddots & \vdots \\ \vec{\mathbf{K}}_I^{1,T} & \vec{\mathbf{K}}_I^{2,T} & \dots & \vec{\mathbf{K}}_I^{s,T} & \dots & \mathbf{K}_I \end{bmatrix} \tag{7}$$

and the vectors

$$\mathbf{u} = \begin{bmatrix} \mathring{\mathbf{u}}^1 \\ \mathring{\mathbf{u}}^2 \\ \dots \\ \mathring{\mathbf{u}}^s \\ \dots \\ \mathbf{u}_I \end{bmatrix} \quad \text{and} \quad \mathbf{f} = \begin{bmatrix} \mathring{\mathbf{f}}^1 \\ \mathring{\mathbf{f}}^2 \\ \dots \\ \mathring{\mathbf{f}}^s \\ \dots \\ \mathbf{f}_I \end{bmatrix} \tag{8}$$

The equilibrium equation (3) may be partitioned into the following systems

$$\begin{cases} \mathring{\mathbf{K}}^s\mathring{\mathbf{u}}^s + \vec{\mathbf{K}}_I^s\mathbf{u}_I = \mathring{\mathbf{f}}^s & , s = 1, NSD \\ \sum_{s=1}^{NSD} \vec{\mathbf{K}}_I^{s,T}\mathring{\mathbf{u}}^s + \mathbf{K}_I\mathbf{u}_I = \mathbf{f}_I & \end{cases} \tag{9}$$

Performing gaussian elimination by blocks:

$$\begin{cases} \mathring{\mathbf{K}}^s\mathring{\mathbf{u}}^s = \mathring{\mathbf{f}}^s - \vec{\mathbf{K}}_I^s\mathbf{u}_I \quad , s = 1, NSD \\ [\mathbf{K}_I - \sum_{s=1}^{NSD} \vec{\mathbf{K}}_I^{s,T}(\mathring{\mathbf{K}}^s)^{-1}\vec{\mathbf{K}}_I^s]\mathbf{u}_I = \mathbf{f}_I - \sum_{s=1}^{NSD} \vec{\mathbf{K}}_I^{s,T}(\mathring{\mathbf{K}}^s)^{-1}\mathring{\mathbf{f}}^s \end{cases} \tag{10}$$

The coefficient matrix of the second equation in (10):

$$\mathbf{S} = \mathbf{K}_I - \sum_{s=1}^{NSD} \vec{\mathbf{K}}_I^{s,T}(\mathring{\mathbf{K}}^s)^{-1}\vec{\mathbf{K}}_I^s \tag{11}$$

is known as the *Schur complement* matrix or *capacitance* matrix.

The first group of equations in (10) represents the uncoupled equilibrium system for the *internal* d.o.f. $\mathring{\mathbf{u}}^s$ at each subdomain, resulting from the non penetration character of the decomposition. This part of the solution is perfectly parallelizable. The size of these problems is given by the granularity of the domain decomposition.

The second part of (10) represents the *interface* problem. The size of the Schur complement **S** is usually much smaller than that of the global matrix **K**, but it is also more dense than **K**. The condition number of **S** is usually much smaller than that of **K**. On the other hand matrix **S** may not be explicitly assembled and the solution performed in a subdomain-wise fashion. This part of the solution is coupled for the whole problem. The efficiency of the global solution is highly tightened to the efficiency of the solution for the interface problem.

We can therefore think as the problem being solved in two steps: an interface problem and an internal one. A popular strategy is to solve the internal problem (10-a) by *direct* methods and the interface problem (10-b) by *iterative* ones. Direct methods are preferred for (10-a) since the lead to close solutions and no error propagation is produced to the interface problem.

Direct methods, however, are not suitable for the interface problems due to their large storage requirements. Matrix **S** is usually full and expensive to construct. Some applications of direct methods are reported in the literature, but they are mainly concerned with special cases (e.g. slender structures) where the interface problem size is limited [1,2]. Iterative methods perform well for the interface problem, in particular conjugate gradient techniques with preconditioning. As it was already said, the condition number for the Schur complement is less than for the whole stiffness matrix. In the case of a Laplace problem, $cond(\mathbf{K}) = O(\frac{1}{h^2})$ while $cond(\mathbf{S}) = O(\frac{1}{h})$. Domain decomposition can be seen as a way of preconditioning the whole problem.

3. SOLUTION OF THE INTERFACE PROBLEM

3.1 The Preconditioned Conjugate Gradient Method

We will focus on the solution of the interface problem (10-b). The interface d.o.f. matrix $\mathbf{K}_I$ may be written

$$\mathbf{K}_I = \sum_{s=1}^{NSD} \mathbf{K}_I^s \tag{12}$$

and the Schur complement

$$\mathbf{S} = \sum_{s=1}^{NSD} \mathbf{S}^s \tag{13}$$

with

$$\mathbf{S}^s = \mathbf{K}_I^s - \vec{\mathbf{K}}_I^{s,T} (\mathring{\mathbf{K}}^s)^{-1} \vec{\mathbf{K}}_I^s \tag{14}$$

Eq. 13 shows that contributions of each subdomain to the matrix **S** may be computed independently. Equation 10-b is rewritten

$$\mathbf{S}\, \mathbf{u}_I = \mathbf{g}_I \tag{15}$$

We will now discuss the solution of (15) via Preconditioned Conjugate Gradient methods. The algorithm is shown in Table I for a generic linear system

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{16}$$

The most time consuming parts of the process are the matrix-vector products at steps A.2 and B.1, and the solution of the LEQ systems at A.3 and B.7 (preconditioning). The rest of the operations on vectors with the size of the global interface problem are dot product and SAXPY operations (Sum of Alpha X Plus Y).

Preconditioning is of paramount importance regarding the performance of a Conjugate Gradient algorithm. Among the popular preconditioners we can mention:

1. Jacobi or diagonal scaling
2. Block Jacobi
3. Incomplete Cholesky factorization
4. Element by Element (EBE)

Domain decomposition results itself in a good preconditioner for a global Conjugate Gradient solution. As limiting cases it performs as a direct global method if the number of subdomains tends to one (i.e. the whole structure as a subdomain), or as a Conjugate Gradient global solution with EBE preconditioner when the number of subdomain tends to the number of elements. Domain

Table I: Preconditioned Conjugate Gradient Algorithm

```
A.      Initialization
  A.1   $\mathbf{x}(0) = \mathbf{x}_{n-1}$
  A.2   $\mathbf{r}(0) = \mathbf{b} - \mathbf{A}\mathbf{x}(0)$                 matrix-vector prod + sum vectors
  A.3   solve $\mathbf{P}\mathbf{z}(0) = \mathbf{r}(0)$                              system solution
  A.4   $\rho(1) = (\mathbf{r}(0), \mathbf{z}(0))$                                 dot product
  A.5   $\mathbf{p}(1) = \mathbf{z}(0)$
  A.6   $i = 1$
B.      Iterations (i=1,2,...)
  B.1   $\mathbf{a}(i) = \mathbf{A}\mathbf{p}(i)$                                  matrix-vector prod.
  B.2   $m(i) = (\mathbf{p}(i), \mathbf{a}(i))$                                dot product
  B.3   $\alpha(i) = \frac{\rho(i)}{m(i)}$
  B.4   $\mathbf{x}(i) = \mathbf{x}(i-1) + \alpha(i)\mathbf{p}(i)$                                  SAXPY
  B.5   $\mathbf{r}(i) = \mathbf{r}(i-1) - \alpha(i)\mathbf{a}(i)$                                  SAXPY
  B.6   Convergence test on $\mathbf{r}(i)$ or $\mathbf{x}(i)$
        if converged go to C,
        otherwise go to B.7
  B.7   solve $\mathbf{P}\mathbf{z}(i) = \mathbf{r}(i)$                              system solution
  B.8   $\rho(i+1) = (\mathbf{r}(i), \mathbf{z}(i))$                               dot product
  B.9   $\beta(i) = \frac{\rho(i+1)}{\rho(i)}$
  B.10  $\mathbf{p}(i+1) = \mathbf{z}(i) + \beta(i)\mathbf{p}(i)$                                   SAXPY
  B.11  $i = i + 1$, go to B.1
C.      End C.G. loop
  C.1   $\mathbf{x}_n = \mathbf{x}(i)$
```

decomposition or substructure-by-substructure preconditioners are reported to be more efficient than EBE preconditioner [3].
Some methods for efficient solution of the interface problem within a domain decomposition frame are referred to in the subsequent sections.

3.2 The Dirichlet-Neumann Method [12]

The most time consuming part in the algorithm of Table I are the matrix-vector products (A.2 and B.3) and the system solutions (A.3 and B.7). We will consider these operations in the frame of a domain decomposition.
The matrix-vector product in B.1 (also in A.3) may be written

$$\mathbf{a} = \mathbf{S}\mathbf{p} \tag{17}$$

S being the Schur complement and, owing to (13),

$$\mathbf{a} = \sum_{s=1}^{NSD} \mathbf{a}^s = \sum_{s=1}^{NSD} \mathbf{S}^s\mathbf{p} \tag{18}$$

that is, the contribution to a is computed separately at each subdomain. At subdomain s we have (see (14))

$$\mathbf{S}^s\mathbf{p} = [\mathbf{K}^s_I - \vec{\mathbf{K}}^{s,T}_I (\mathring{\mathbf{K}}^s)^{-1} \vec{\mathbf{K}}^s_I]\mathbf{p} \tag{19}$$

and considering the subdomain problem:

$$\begin{bmatrix} \mathring{\mathbf{K}}^s & \vec{\mathbf{K}}^s \\ \vec{\mathbf{K}}^{s,T} & \bar{\mathbf{K}}_I^s \end{bmatrix} \begin{bmatrix} \mathbf{v}^s \\ \mathbf{p} \end{bmatrix} = \begin{bmatrix} 0 \\ \mathbf{a}^s \end{bmatrix} \tag{20}$$

The contribution of subdomain s to the vector in (18) is

$$\mathbf{a}^s = \vec{\mathbf{K}}^{s,T}\mathbf{v}^s + \mathbf{K}_i^s\mathbf{p} \tag{21}$$

where $\mathbf{v}^s$ is the solution of

$$\mathring{\mathbf{K}}^s\mathbf{v}^s = -\vec{\mathbf{K}}^s\mathbf{p} \tag{22}$$

Equations 20 to 22 show that to compute the matrix-vector product (18) it suffices to solve at each subdomain a *Dirichlet* problem where prescribed values $\mathbf{p}$ are imposed on the interface Γ_3^s, and the associated force vector $\mathbf{a}^s$ is obtained. Finally the contributions $\mathbf{a}^s$ of each subdomain are added together.
For the preconditioning phase (B.7 and A.3, in Table I) the following procedure is followed [4]. At each subdomain a matrix $\mathbf{D}_I^s$ is defined such that

$$\sum_{s=1}^{NSD} \mathbf{D}_I^s = \mathbf{I}_I \tag{23}$$

That is to say that by assembling the matrices $\mathbf{D}_I^s$ of all subdomains the identity matrix is get on the global interface space. The simplest choice for $\mathbf{D}_I^s$ is a diagonal matrix whose entries are the reciprocal of the number of subdomains that share the current d.o.f. .
Given the residual force computed at each conjugate gradient iteration the projection onto each subdomain is performed:

$$\mathbf{r}^s = \mathbf{D}^{s,T}\mathbf{r} \tag{24}$$

At each subdomain the following system is solved:

$$\mathbf{S}^s\mathbf{z}^s = \mathbf{r}^s \tag{25}$$

Finally subdomain contributions to the kinematical residual are averaged:

$$\mathbf{z} = \sum_{s=1}^{NSD} \mathbf{D}^s\mathbf{z}^s \tag{26}$$

This is equivalent to use a preconditioner of the form

$$\mathbf{P}^{-1} = \sum_{s=1}^{NSD} \mathbf{D}^s(\mathbf{S}^s)^{-1}\mathbf{D}^{sT} \tag{27}$$

Solution of (25) is in turn equivalent to solve a *Neumann* problem on subdomain s where the solution vector $\mathbf{z}^s$ contains displacements of the interface d.o.f.. It can be performed without explicitly forming the Schur complement matrix $\mathbf{S}$, by writing for each subdomain:

$$\begin{bmatrix} \mathring{\mathbf{K}}^s & \vec{\mathbf{K}}^s \\ \vec{\mathbf{K}}^{s,T} & \bar{\mathbf{K}}_I^s \end{bmatrix} \begin{bmatrix} \mathbf{v}^s \\ \mathbf{z}^s \end{bmatrix} = \begin{bmatrix} 0 \\ \mathbf{r}^s \end{bmatrix} \tag{28}$$

The solution of the Neumann problem (28) on subdomain s is

$$\mathbf{v}^s = -(\mathring{\mathbf{K}}^s)^{-1}\vec{\mathbf{K}}^s\mathbf{z}^s \tag{29}$$

$$\mathbf{z}^s = \left(\mathbf{K}_I^s - \vec{\mathbf{K}}^{s,T}(\mathring{\mathbf{K}}^s)^{-1}\vec{\mathbf{K}}^s\right)^{-1}\mathbf{r}^s = \left(\mathbf{S}^s\right)^{-1}\mathbf{r}^s \tag{30}$$

When applying the conjugate gradient iteration to the interface problem, matrix-vector products (A.2 and B.1) and system solutions (A.3 and B.7) are replaced by alternatively solving subdomain problems with Dirichlet and Neumann boundary conditions, respectively. These subdomain solvers are perfectly parallelizable. For Laplace problems it has been reported that while the condition number for the global matrix is $O(\frac{1}{h^2})$ (h being the element size), that for the Schur complement is $O(\frac{1}{h})$, and by using the preconditioner (27) it reduces to $O(1)$. This result have not been proved for other problems but numerical evidence show similar behavior for structural mechanics problems. In general cases equation 25 leads to singular $\mathbf{S}$ matrices unless enough rigid body motions be prevented by proper fixations. Several techniques have been proposed to handle this drawback. De Roek and Le Tallec [4] modified the solution algorithm by eliminating d.o.f. associated with zero pivots. If the rigid body motions are set arbitrarily, the efficiency of the preconditioner deteriorates with the increase in the number of subdomains. A practical limit was found to be 16 subdomains [5]. This is due to the local character of the correction in the residual forces implied in this algorithm. In fact, Widlund has shown that the condition number of a DDM, whithout a mechanism of global transport of the information, grows at least as $\frac{1}{H^2}$, H being the subdomin largest diameter [6]. Some attempts to propagate the corrections between the subdomains have been done by solving, at each iteration, a *coarse problem* with few d.o.f. on each subdomain [7,8,9]. Other efficient techniques have been developed, such as the FETI (Finite Element Tearing and Interpolating), which is based on a Dual Schur complement method [10,11].

4. PARALLEL CHARACTERISTICS OF FINITE ELEMENT CODES

The structure of a non linear finite element code for structural analysis may be written in the general pattern:

```
For each time step:
  A update variables
  B iterate on:
     B.1 compute element stress
     B.2 assemble element forces into global vectors
     B.3 solve linearized equilibrium equations system
     B.4 convergence test:
              if converged go to next time step
              if not converged go to next iteration
```

The algorithm operations may be classified, with regard to their parallel character, in:

a) naturally uncoupled tasks: such as updating of global state vectors (A) or computing element stress (B.1). They are easily parallelizable. Allocating each element to one processor is the most direct way of doing it.

b) loosely uncoupled tasks: such as assembly operations (B.2) or error parameters computation (B.4) Care must be taken in updating global (shared) variables. Parallel accumulation tasks should be synchronized. Reordering strategies (v.g. "coloring") may be used for these tasks.

c) strongly coupled tasks: the solution of the linear equations system. It is the most complex tasks to render parallel. It may be solved either by *algebraic parallelism* where each task of the solution

algorithm is split into the different processors, or by *domain decomposition* in the way described in precedent sections.
Domain decomposition, being a tool for the solution of the equation system, is suitable to deal with the other types of parallel tasks. It requires just to synchronize the accumulation operations while assembling or error computing (b).

5. PARALLEL COMPUTATIONS ON CLUSTERS OF COMPUTERS

Distributed memory architectures are taken as target for our development. In particular the work on clusters of workstations or personal computers is considered. Domain decomposition methods are well suited for these architectures, but may be used also in shared memory computers.
A message passing programming model is suitable for distributed computations. Communication between processors is handled by means of PVM (Parallel Virtual Machine) [13]. This is a software package allowing that heterogeneous or homogeneous clusters of computers be used as a single parallel -distributed memory- computer. It provides the basic routines for communication and runs under Unix on a wide range of computers (including PC).
A software that manages communication between personal computers have been developed [14]. This package runs under DOS and allows to split computations in a fashion similar to PVM. With limited characteristics, however, it allows to conduct parallel computations under DOS.

6. CONCLUSIONS

In this report, domain decomposition techniques for solving large systems of equations have been presented. The aim is to make use of them to perform parallel solution of non linear finite element structural problems.
Parallelization of linear elastic programs have been performed and the work on transient analysis is being conducted.

ACKNOWLEDGEMENTS

This work have been done under support of CONICET (Consejo Nacional de Investigaciones Científicas y Técnicas), Universidad Nacional del Litoral, Argentina, and Fundación Antorchas.

REFERENCES

1. Farhat, Ch., Crivelli, L (1989) - "A General Aproach to NL FE Computations on Shared Memory Multiprocessors", Comp. Meth. in Appl. Mechanics and Engineering, Vol. 72, pp 153-172.
2. Lesoinne, M., Farhat, Ch., Géradin M. (1991) - "Parallel/Vector Improvements of the Frontal Method", Int. J. Numer. Methods in Engineering, Vol. 32, pp 1267-1282.
3. Nour-Omid, B. (1993) - "Solving Large linearized Systems in Mechanics", in Solving Large-Scale Problems in Mechanics (M. Papadrakakis, Ed.), J.Wiley and Sons.
4. De Roeck, Y.-H., Le Tallec, P. (1991) - "Analysis and Test of a Local Domain Decomposition Preconditioner", in IV Intl.Symp. on Domain Decomposition Meth. for Partial Differential Equations (R. Glowinsky, Y. Kuznetsov, G. Meurant, J. Périaux and O. Widlund, Eds.), SIAM, Philadelphia.

5. Le Tallec, P., De Roeck, Y.-H., Vidrascu, M. (1990) - "Domain Decomposition Methods for Large Linearly Elliptic 3D Problems", Tech.Report TR/PA/90/20, Centre Européen de Recherche et de Formation Avancée en Calcul Scientifique, Toulouse, France.
6. Widlund, O.B. (1988) - "Iterative Substructuring Methods: Algorithms and Theory for Elliptic Problems in the Plane", First Int. Symposium on Domain Decomposition Methods, SIAM, R. Glowinsky, G.H. Golub, G.A. Meurant and J. Periaux, eds., Philadelphia, 1988.
7. Bramble, J.H., Pasciak, J.E., Schatz A.H., (1986) - "The Construction of Preconditioners for Elliptic Problems by Substructuring", Math. Computations, Vol. 47, pp 103-134.
8. Mandel J. (1993) - "Balancing Domain Decomposition", Communications in Numerical Methods in Engineering, Vol. 9, pp 233-241.
9. Smith B.F. (1991) - "An Optimal Domain Decomposition Preconditioner for the Finite Element Solution of Linear Elasticity Problems", SIAM J. Sci.Stat.Comput., Vol 13.
10. Farhat, Ch., Roux, F.-X. (1991) - "A Method of Finite Element Tearing and Interconnection and its Parallel Solution Algorithm", Int.J.Numerical Methods in Engineering, Vol.32, pp 1205-1227.
11. Farhat, Ch., Géradin, M. (1990) - "Using a Reduced Number of Lagrange Multipliers for Assembling Parallel Incomplete Field Finite Element Approximations",Report CU-CSSC-90-23, Univ. of Colorado, Boulder.
12. Bjørstad, P.E. and Widlund, O.B., (1986) - "Iterative Methods for the Solution of Elliptic Problems on Regions Partitioned into Substructures", SIAM J.Num.Analysis, Vol. 23, pp. 1097-1120.
13. Beguelin, A., Dongarra, J., Geist, G.A., Manchek, R., Sunderam, V.S., (1992) - "A user's guide to PVM parallel virtual machine", Oak Ridge National Lab. Int. Report, TN, June 1992.
14. Sonzogni, V.E. (1994) - "PCCOM: A Software System to Perform Concurrent Computations with a Message Passing Programming Models under DOS", Report GTM 95-5, INTEC-UNL-CONICET, 1995.