Trends in Mathematics
Information Center for Mathematical Sciences
Volume 4, Number 1, June 2001, Pages 108–115

SUBSTRUCTURING TECHNIQUE AND PARALLEL COMPUTATIONS

M.P.LEVIN

Abstract. A features analysis of substructuring technique intended for solution of huge systems of linear equations on parallel computers is provided. Some new features increasing the efficiency of this technique are proposed and considered.

1. Introduction

In recent years various types of algorithms intended for solution of huge systems of linear algebra equations arising in finite difference or finite element approximations of boundary value problems for partial differential equations (PDE) have been developed [1-11]. All these known algorithms belong to one of two classes just namely direct and iterative solvers. According to the name, the first class of solvers is based on direct methods and consists of algorithms based on various modifications of Gauss elimination technique. The second class is base on different types iteration methods and are mostly popular in recent years because, at first sight, the parallel realization of these algorithms seems more simple in comparing with the realization of the first class solvers. In this paper, analyzing the realization of direct solvers based on the Gauss elimination technique and considering some new features of this algorithm, we show that it can be also efficiently realized in parallel computations.

The main principle of the decomposition using in the both above mentioned classes is a dividing of the region, where the corresponded boundary value problem is set, into the finite set of subregions or substructures. However the features of the dividing are depended on the type of solvers. For the direct solvers the dividing is usually provided without overlapping of the neighborhood subregions. Otherwise for iterative solvers the dividing is usually provided with overlapping, since its plays very important role in acceleration of convergence of iterative solvers.

Both solvers in comparing with each other have as some advantages and also some deficiencies. The principal advantage of the direct solvers consist in the independence of the number of operations on the spectral properties of the considering systems of the linear algebra equations. The number of operations in the direct solvers depends only on the structure of coefficients matrix and therefore even in case, when we don't know the spectral properties of the considering system, we

2000 *Mathematics Subject Classification.* 34A05, 34A30.
Key words and phrases. linear equations,substructuring technique, Gauss elimination, parallel computation, decomposition.
Received February 10, 2001.

could estimate the time necessary for computations. Sometimes, especially in industrial applications, it is very important. But result of computations by the direct methods depends on spectral properties and especially the rounding error spreading in these methods depends on the spectral properties of coefficients matrices. This means that in case of bad spectral properties, sometimes we could evaluate solution, but this solution should have big rounding errors and sometimes we could not obtain solution in consequence of the rounding errors overflow in computations.

Considering the iterative solvers let us turn our view on the principle advantage of these algorithms consisting in independence of rounding errors of searching variables on the number of variable [3,4]. The rounding error of each variable is defined by only the properties of the neighborhood equations corresponding to this variable. It means, that the rounding errors in the iterative solvers are not spread from the first equation to the last and thus has not been accumulated during the computations. Of course, it is a significant advantage of iterative solvers, but if we don't know the spectral properties of considering system, we could not estimate the time of computations and also warrant the convergence of iterations. Therefore in industrial applications, it is very important to have at disposal both direct and iterative solvers for the effective solution of huge system of linear algebraic equations.

2. Background of Substructuring Technique

The basic idea of the substructuring technique consists in the generation and solution of the linear algebra equations systems approximating the boundary value problems for the partial differntial equations (PDE) by the generation and transformation of the matrices corresponding to the separate subregions composing the considering region. These subregions don't have overlapping and their boundaries consist of two parts: means usual external boundaries and internal or interface boundaries. The interface boundaries are the boundaries between the neighborhood subregions. Further we call the mesh nodes located on the interface boundaries as an interface nodes and appropriate equations as an interface equations.

Now let us consider any complicated region S composed by m subregions S_j, $(j = 1, 2, 3, ..., m)$. We enumerate searching variables and appropriate equations as follows: at first we enumerate all variables correspond to the internal nodes and external boundaries of all considering subregions consecutively going on all subregions and at second we enumerate variables corresponding to the interface nodes going on all subregions in the same order as in the previous case. Then we can present the considering system of linear equations in the arrow-wise form, as follows

$$\begin{bmatrix} A_{ii}^{(1)} & 0 & ... & 0 & A_{iB}^{(1)} \\ 0 & A_{ii}^{(2)} & ... & 0 & A_{iB}^{(2)} \\ ... & ... & ... & ... & ... \\ 0 & 0 & ... & A_{ii}^{(m)} & A_{iB}^{(m)} \\ A_{Bi}^{(1)} & A_{Bi}^{(2)} & ... & A_{Bi}^{(m)} & A_{BB} \end{bmatrix} \begin{Bmatrix} x_i^{(1)} \\ x_i^{(2)} \\ ... \\ x_i^{(m)} \\ x_B \end{Bmatrix} = \begin{Bmatrix} b_i^{(1)} \\ b_i^{(2)} \\ ... \\ b_i^{(m)} \\ b_B \end{Bmatrix} . \tag{1}$$

Here we denote as $A_{ii}^{(j)}$, $A_{iB}^{(j)}$, $A_{Bi}^{(j)}$ and A_{BB} submatrices of the coefficients matrix of the linear algebra equations system, $x_i^{(j)}$ and x_B subvectors of the searching unknown vector, $b_i^{(m)}$ and b_B subvectors of the known right-hand side. Further we consider the case of symmetrical and positive defined systems arising in consideration of the boundary value problems for the PDE with the self-adjoint operators.

In this case we have

$$(A_{iB}^{(j)})^T = A_{Bi}^{(j)} \ .$$

Also let us notice that $A_{iB}^{(j)}$ and $A_{Bi}^{(j)}$ are rectangular matrices and $A_{ii}^{(j)}$, A_{BB} are square matrices. Subvectors $x_i^{(j)}$ and $b_i^{(m)}$ correspond to the internal nodes or nodes located on external boundaries, whereas subvectors x_B and b_B correspond to the interface nodes located on boundaries between subregions. In compact form, we can write (1), as follows

$$Ax = b \ , \tag{2}$$

where

$$A = \begin{bmatrix} A_{ii}^{(1)} & 0 & ... & 0 & A_{iB}^{(1)} \\ 0 & A_{ii}^{(2)} & ... & 0 & A_{iB}^{(2)} \\ ... & ... & ... & ... & ... \\ 0 & 0 & ... & A_{ii}^{(m)} & A_{iB}^{(m)} \\ A_{Bi}^{(1)} & A_{Bi}^{(2)} & ... & A_{Bi}^{(m)} & A_{BB} \end{bmatrix} \ , \quad x = \begin{Bmatrix} x_i^{(1)} \\ x_i^{(2)} \\ ... \\ x_i^{(m)} \\ x_B \end{Bmatrix} \ , \ b = \begin{Bmatrix} b_i^{(1)} \\ b_i^{(2)} \\ ... \\ b_i^{(m)} \\ b_B \end{Bmatrix} \ .$$

Let us take into consideration any matrix H with the arrow structure similar to the matrix A structure, as follows

$$H = \begin{bmatrix} H_{ii}^{(1)} & 0 & ... & 0 & H_{iB}^{(1)} \\ 0 & H_{ii}^{(2)} & ... & 0 & H_{iB}^{(2)} \\ ... & ... & ... & ... & ... \\ 0 & 0 & ... & H_{ii}^{(m)} & H_{iB}^{(m)} \\ H_{Bi}^{(1)} & H_{Bi}^{(2)} & ... & H_{Bi}^{(m)} & H_{BB} \end{bmatrix} \ .$$

Then, multiplying both sides of (1) or (2) by H-matrix, we obtain

$$HAx = Hb \ , \tag{3}$$

where

$$HA = \begin{bmatrix} H_{ii}^{(1)}A_{ii}^{(1)}+ \\ +H_{iB}^{(1)}A_{Bi}^{(1)} & H_{iB}^{(1)}A_{Bi}^{(2)} & ... & H_{iB}^{(1)}A_{Bi}^{(m)} & H_{ii}^{(1)}A_{iB}^{(1)}+ \\ & & & & +H_{iB}^{(1)}A_{BB} \\ H_{iB}^{(2)}A_{Bi}^{(1)} & H_{ii}^{(2)}A_{ii}^{(2)}+ \\ & +H_{iB}^{(2)}A_{Bi}^{(2)} & ... & H_{iB}^{(2)}A_{Bi}^{(m)} & H_{ii}^{(2)}A_{iB}^{(2)}+ \\ & & & & +H_{iB}^{(2)}A_{BB} \\ ... & ... & ... & ... & ... \\ H_{iB}^{(m)}A_{Bi}^{(1)} & H_{iB}^{(m)}A_{Bi}^{(2)} & ... & H_{ii}^{(m)}A_{ii}^{(m)}+ \\ & & & +H_{iB}^{(m)}A_{Bi}^{(m)} & H_{ii}^{(m)}A_{iB}^{(m)}+ \\ & & & & +H_{iB}^{(m)}A_{BB} \\ H_{Bi}^{(1)}A_{ii}^{(1)}+ & H_{Bi}^{(2)}A_{ii}^{(2)}+ \\ +H_{BB}A_{Bi}^{(1)} & +H_{BB}A_{Bi}^{(2)} & ... & H_{Bi}^{(m)}A_{ii}^{(m)}+ \\ & & & +H_{BB}A_{Bi}^{(m)} & H_{BB}A_{BB}+ \\ & & & & +\sum_{j=1}^{m} H_{Bi}^{(j)}A_{iB}^{(j)} \end{bmatrix} \ .$$

Choosing $H_{iB}^{(j)} = 0$ and $H_{Bi}^{(j)} = -H_{BB}A_{Bi}^{(j)}(A_{ii}^{(j)})^{-1}$, for $j = 1, 2, ..., m$ we obtain HA-matrix in the following block form

$$HA = \begin{bmatrix} H_{ii}^{(1)}A_{ii}^{(1)} & 0 & ... & 0 & H_{ii}^{(1)}A_{iB}^{(1)} \\ 0 & H_{ii}^{(2)}A_{ii}^{(2)} & ... & 0 & H_{ii}^{(2)}A_{iB}^{(2)} \\ ... & ... & ... & ... & ... \\ 0 & 0 & ... & H_{ii}^{(m)}A_{ii}^{(m)} & H_{ii}^{(m)}A_{iB}^{(m)} \\ 0 & 0 & ... & 0 & \begin{array}{c} H_{BB}[A_{BB}- \\ -\sum_{j=1}^{m} A_{Bi}^{(j)}(A_{ii}^{(j)})^{-1}A_{iB}^{(j)}] \end{array} \end{bmatrix} .$$

The appropriate right-hand side of (3) is followed

$$Hb = \begin{bmatrix} H_{ii}^{(1)}b_i^{(1)} \\ H_{ii}^{(2)}b_i^{(2)} \\ ... \\ H_{ii}^{(m)}b_i^{(m)} \\ H_{BB}[b_B - \sum_{j=1}^{m} A_{Bi}^{(j)}(A_{ii}^{(j)})^{-1}b_i^{(j)}] \end{bmatrix} .$$

For simplification we can choose $H_{BB} = E_{BB}$, where E_{BB} is the appropriate identical matrix.

Thus, we could see, that the initial problem (1) of the high dimension is reduced to the set of the low dimension problems as follows: at the first stage, we need to solve the following system of the linear equations with respect to the unknown vector x_B corresponding to the interface nodes

$$[A_{BB} - \sum_{j=1}^{m} A_{Bi}^{(j)}(A_{ii}^{(j)})^{-1}A_{iB}^{(j)}]x_B = [b_B - \sum_{j=1}^{m} A_{Bi}^{(j)}(A_{ii}^{(j)})^{-1}b_i^{(j)}] , \tag{4}$$

and, at the second stage, to solve the systems corresponding to subregions with respect to unknown variables $x_i^{(j)}$, $j = 1, 2, ..., m$ corresponding to the internal nodes of the separate subregions

$$H_{ii}^{(j)}A_{ii}^{(j)}x_i^{(j)} = H_{ii}^{(j)}b_i^{(j)} - H_{ii}^{(j)}A_{iB}^{(j)}x_B , \ (j = 1, 2, ..., m) . \tag{5}$$

3. Parallel Realization

The second stage of the above mentioned algorithm (5) can be efficiently realized on the parallel computers both with share and with common memory, and the first stage of this algorithm is a principle subject for the effective parallelization. Usually the system (4) is dense and it's solution needs more times for computations in comparison with systems (5). Also the coefficient matrix in (5) depends on the invert matrices of coefficients for all systems (5). At first sight, it seems that we need to invert matrices $A_{ii}^{(j)}$ for the all subregions to evaluate the coefficient matrix of (4), and then evaluate the following term $A_{Bi}^{(j)}(A_{ii}^{(j)})^{-1}A_{iB}^{(j)}$, known as a Shur compliment, as proposed in [1-3,11]. But now we consider how to joint the evaluation of matrix for system (4) and decomposition of matrices for systems (5) in parallel regime.

It is easy to see, that transformation of all blocks, except the last diagonal block, in the last column of coefficient matrix HA depends only on the matrices

corresponding to the any separate subregion, and therefore this transformation can be parallelized efficiently on all types of parallel computers.

Let us introduce the matrices $A_{ii}^{(j)}$ as a product of the low triangle and upper triangle matrices, as follows

$$A_{ii}^{(j)} = L_{ii}^{(j)} U_{ii}^{(j)}, \ (j = 1, 2, ..., m) \ . \tag{6}$$

This decomposition is valid, if the matrices $A_{ii}^{(j)}$ are positive defined. The classical well-known Gauss algorithm [3] can be applied to evaluate this decomposition and the appropriate matrices $L_{ii}^{(j)}$ and $U_{ii}^{(j)}$. We can note, that these matrices are easy invertible. Let us choose $H_{ii}^{(j)} = (L_{ii}^{(j)})^{-1}$ and take into consideration the following block matrices corresponding to the separate subregions

$$A^{(j)} = \begin{bmatrix} A_{ii}^{(j)} & A_{iB}^{(j)} \\ A_{Bi}^{(j)} & A_{BB}^{(j)} \end{bmatrix} \ ,$$

and the appropriate vectors of right-hand side

$$b^{(j)} = \begin{Bmatrix} b_i^{(j)} \\ b_B^{(j)} \end{Bmatrix} \ .$$

Here

$$\sum_{j=1}^{m} A_{BB}^{(j)} = A_{BB} \ , \tag{7}$$

and

$$\sum_{j=1}^{m} b_i^{(j)} = b_B \ . \tag{8}$$

We apply the Gauss elimination to the transformation of the matrices $A^{(j)}$ into the block upper triangular matrices with upper triangular submatrices in upper left corner. This transformation can be written as follows

$$\begin{aligned} H^{(j)} A^{(j)} &= \begin{bmatrix} (L_{ii}^{(j)})^{-1} & 0 \\ -A_{Bi}^{(j)} (A_{ii}^{(j)})^{-1} & E_{BB}^{(j)} \end{bmatrix} \begin{bmatrix} A_{ii}^{(j)} & A_{iB}^{(j)} \\ A_{Bi}^{(j)} & A_{BB}^{(j)} \end{bmatrix} = \\ &= \begin{bmatrix} U_{ii}^{(j)} & (L_{ii}^{(j)})^{-1} A_{iB}^{(j)} \\ 0 & [A_{BB}^{(j)} - A_{Bi}^{(j)} (A_{ii}^{(j)})^{-1} A_{iB}^{(j)}] \end{bmatrix} \ , \end{aligned} \tag{9}$$

where $E_{BB}^{(j)}$ are the appropriate identical matrices. We call this transformation *an incomplete Gauss elimination* of $A^{(j)}$ matrix. Analyzing HA matrix, in case $H_{ii}^{(j)} = (L_{ii}^{(j)})^{-1}$ and $H_{BB} = E_{BB}$, we can see that this matrix is a composition of $H^{(j)} A^{(j)}$ matrices taken over the all considering subregions and the low right angle submatrix of HA matrix is a sum of the low right angle submatrices of $H^{(j)} A^{(j)}$-matrix. Thus we can realize the transformation of matrices $A_{ii}^{(j)}$ into the upper triangular form and generation of the coefficient matrix and the appropriate right-hand side of the interface equations simultaneously in parallel regime by processing of all separate matrices. The appropriate transformation of the right-hand sides for

the separate subregions is described by the following formula

$$H^{(j)} \begin{bmatrix} b_i^{(j)} \\ b_B^{(j)} \end{bmatrix} = \begin{bmatrix} (L_{ii}^{(j)})^{-1} & 0 \\ -A_{Bi}^{(j)}(A_{ii}^{(j)})^{-1} & E_{BB}^{(j)} \end{bmatrix} \begin{bmatrix} b_i^{(j)} \\ b_B^{(j)} \end{bmatrix} = \tag{10}$$

$$= \begin{bmatrix} (L_{ii}^{(j)})^{-1} b_i^{(j)} \\ b_B^{(j)} - A_{Bi}^{(j)}(A_{ii}^{(j)})^{-1} b_i^{(j)} \end{bmatrix}.$$

It's not easy to see, that the algorithm considered above is more convenient for parallel computations in comparison with algorithm realizing the Shur compliment because it decreases the number of interprocessor communications.

After the generation of equations corresponding to the interface nodes, we need to solve the system of these equations. Its solution is a narrow place of our algorithm, because the next stage of the considering algorithm, means the back substitution of the solution into the splitting systems of the linear equations with upper triangular matrices corresponding to the internal nodes, and solution of these systems, can be simply parallelized by processing on the separate systems.

Now let us consider a problem of the Gauss elimination algorithm realization in the solution of the interface equations in the parallel regime. We restrict our consideration by the case of the symmetrical systems, usually arising in finite element and some other applications in mathematical physics. In this case, it is convenient to use a column oriented modification of the Gauss elimination algorithm [3]. In this modification, the transformation of the coefficient matrix entries $[a_{ij}]$ and the appropriate right-hand side $[b_i]$ due to evaluation of the current pivoting variable with number n from the all other equations with numbers $l > n$ is described by the following algorithm:

Step 1. Let l be a number of the current pivot row of the considering matrix $[a_{ij}]$ and let the width of the matrix band in this row be equal to n_l, where $n_l \geq 1$.

Step 2. If $n_l = 1$, then go to the Step 1 with $l := l + 1$, else go to the next step.

Step 3. Take a diagonal entry in this row $d_1 = a_{ll}$.

Step 4. Start the cycle on evaluation of all columns of the matrix $[a_{ij}]$ in the range $m = (l+1), (l+2), ..., n_l$ and also of the right-hand side $[b_i]$ as follows.

Step 5. Take the entry $d_2 = a_{ln_l}$ and, if $d_2 = 0$, then go to Step 10, else go to the next step.

Step 6. Calculate $d = d_2/d_1$.

Step 7. Read a row-oriented vector

$$v_r = [a_{l,(l+1)}, a_{l,(l+2)}, ..., a_{l,n_l}]^T$$

and a column-oriented vectors

$$v_c = [a_{(l+1)m}, a_{(l+2)m}, ..., a_{m,m}]^T$$

and

$$b_c = [b_{(l+1)}, b_{(l+2)}, ..., b_m]^T.$$

Step 8. Calculate

$$v_c := v_c - d \cdot v_r \ , \qquad b_c := b_c - \frac{b_l}{d_1} \cdot v_r \ , \tag{11}$$

and return all recount components of v_c and b_c vectors on the appropriate positions in the data $[a_{ij}]$-matrix and in the right-hand side $[b_i]$ respectively.

Step 9. Finish the cycle on m.

Step 10. Assign $l := l + 1$ and, if l is not a number of the last equation, return to Step 1, otherwise finish the Gauss elimination process.

Since evaluations of vectors v_c and b_c are not connected between themselves, then they can be executed in different parallel processors with high efficiency. Also evaluation of the components of above mentioned vectors is a subject for parallelization. But sometimes this work can be done by compilers. For example on Hewlett Packard Work Stations, the C++ and Fortran compilers both in the optimization regime implement even not parallelized computations according to the formulas (10) by the substitution of the first level Basic Linear Algebra Subroutines (BLAS) themselves.

4. Multi-Level Substructuring Technique and Equivalent Decomposition.

Above we have considered the mathematical background for the two-level decomposition model in the substructuring technique. This means that only the data region is divided into subregions. However each subregion also can be divided into subregions and so on. In this case we have so called the multi-level substructuring decomposition. All formulas considered above, in multi-level case are also the same. But programming in multi-level case is more complicated than in two-level case. The essential simplification of programming is possible by the using of the recursive functions technique [10]. Also we can add that by using of the multi-level decomposition, we essentially lowering the order of considering systems of linear equations, and therefore decreasing the time and number of operations necessary for the solution of these systems. The significant success can be achieved, if we provide decomposition of initial region by such manner, that in result we obtain many *equivalent* subregions. We call two subregions i and j *equivalent*, if they have identical $A^{(i)}$ and $A^{(j)}$ matrices. In this case, we can essentially reduce the time and the number of operations necessary for the *incomplete Gauss elimination*, because for a few equivalent subregions we need to provide the full *incomplete Gauss elimination* only for one subregion and for all other subregions we need to provide it only for their right-hand sides vectors (10). Of course, if the appropriate right-hand sides also are equivalent, the final effect will be more significant. Thus we can see, that a good decomposition of the initial region with separation of equivalent subregions in the above mentioned sense, can deliver an excellent results in lowering of computational efforts for the solution of huge linear systems arising in the numerical solutions of boundary-value problems for PDE.

References

[1] A.Noor, H.Kamel and R.Fulton, *Substructuring Techniques - Status and Projections*, Computers and Structures, 8 (1978), pp. 621–632.

[2] P.Bjorstad, *A Large Scale, Sparse, Secondary Storage, Direct Linear Equation Solver for Structural Analysis and its Implementation on Vector and Parallel Architectures*, Parallel Computing, 5 (1987), pp. 3–12.

[3] J.M.Ortega, *Introduction to parallel and vector solutions of linear systems*, Plenum Press, New York, 1988.

[4] J.M.Ortega and R.G.Voigt, *Solution of partial differential equations on vector and parallel computers*, SIAM, Philadelphia, 1985.

[5] M.Dryja, B.F.Smith and O.B.Widlund, *Shwarz analysis of iterative substructuring algorithms for elliptic problems in three dimensions*, SIAM Journal of Numerical Analysis, 31 (1994), pp. 1662–1694.

[6] B.F.Smith, *Implementation of an iterative/substructuring algorithm for problems in three dimensions*, SIAM Journal on Scientific Computing, 14 (1993), pp. 406–423.

[7] L.C.Berselli and F.Salery, *New substructuring domain decomposition methods for advection-diffusion equations*, Journal of Computational and Applied Mathematics, 116 (1995), pp. 201–220.

[8] N.Bouhaddi and R.Fillod, *Substructuring by a two level dynamic condensation method*, Computers and Structures, 60 (1996), pp. 403–410.

[9] H.R.Thomas, H.T.Yang, Y.He and A.D.Jefferson, *Solving coupled thermo-mechanical problems in unsaturated sold using a substructuring frontal technique*, Communications in numerical methods in engineering, 14 (1998), pp. 783–792.

[10] M.Saxena and R.Perrucchio, *Parallel FEM algorithms based on recursive spatial decomposition - II. Automatic analysis via hierarchical substructuring*, Computers and Structures, 47 (1993), pp. 143–154.

[11] J.R.Shewchuk and O.Ghattas, *A compiler for finite element methods with domain-decomposed unstructured meshes*, in: Domain Decomposition Methods in Scientific and Engineering Computing, Proceedings of the Seventhe International Conference on Domain Decomposition, October 27-30, 1993, The Pensylvania State University (1994), pp. 445–450.

Department of Mathematics, Korea Advanced Institue of Science and Technology
E-mail address: levin@math.kaist.ac.kr; Mikhail_Levin@hotmail.com