2nd Int. Ph.D. Symposium in Civil Engineering 1998 Budapest

SOME PRELIMINARY RESULTS OF A PARALLEL MODEL FOR NON-CONVENTIONAL FINITE ELEMENT FORMULATIONS

Ildi Cismasiu[1] and J. P. Moitinho de Almeida[2]
Instituto Superior Técnico, Departamento de Engenharia Civil
1096 Lisboa CODEX, Portugal

SUMMARY

In the present paper some implementation aspects of a parallel model for the solution of the large-scale systems deriving from non-conventional finite element formulations are presented. Results obtained when this implementation is used for conventional finite element formulation are also presented. A comparison is made between alternative variants of the Conjugate Gradient algorithm for the parallel solution of the linear system, based on a subdomain division technique.

Keywords: Finite Element Formulations; Parallel Processing; Topological Data Structures; Conjugate Gradient Methods.

1. INTRODUCTION

The non-conventional finite element formulations developed within the Structural Analysis Group at IST, are by their nature, suitable for parallel processing [1]. Most of the formulations lead to systems of large dimension, with a non-overlapping sparse block structure. These formulations use coarse meshes with approximation functions of high degree, which allow to use an element by element parallelisation technique, which makes them attractive when implemented on a distributed platform.

In the numerical implementation process it becomes clear that only by using appropriate data structures it is possible to take full advantage of these characteristics. In this paper we present an implementation of such data structures, which tries to satisfy a set of objectives that were specifically taken into consideration:

- to gain as much as possible in terms of speed up and efficiency,
- to preserve generality and robustness, allowing to use all non-conventional and conventional formulations,
- to efficiently organise the communications required by the parallel computation using a conjugate gradient method.

To achieve these goals appropriate data structures were designed, to describe the finite element mesh, allowing for an efficient determination of the relations between sides, elements and nodes.

[1] *PhD student*
[2] *Associate Professor of Structural Engineering*

A second class of data structures was developed to split up the coefficients matrix in such a way that the identification between the logical division of the solution vector with the corresponding processor is easily done.

2. DATA STRUCTURE FOR TOPOLOGICAL MODELLING

The core of the data structures used for the geometrical or topological manipulation of a two dimensional model, is the winged-edge data structure, proposed by Mantÿla [3, 9]. This data structure fully describes the components of a 2D finite element mesh, elements (faces), edges and nodes and their topological relations. Each `Loop` is an element in the mesh description, the `Node` and the `Edge` have the same meaning as in the finite element mesh.

The data structures were initially developed for the Hybrid Mixed model [5], and redesigned to allow their use irrespective of the formulation. They have been tested successfully for a conventional nodal formulation.

One of the advantage of these data structures is that all the procedures which operate on these components (e.g to identify the edges connected to a node, to identify the nodes/edges of an element or to find the adjacent elements) can be efficiently implemented and the computational cost is proportional to the number of entities locally involved.
These structures can be calculated by processing a "standard" finite element data file or computed and used within the mesh generator [11], which may be coupled with the analysis program.
To handle specific aspects about each finite element formulation e.g. approximation functions or material definitions, data substructures termed `Local`, were developed for each topological entity. One of the functions of these Local structures is to keep the mapping information between the topological entities, structural variables and structural matrices, expressed by pointers.

3. DATA STRUCTURES FOR MATRIX AND VECTOR STORAGE

For conventional finite element models the variables associated with the nodes that belong to shared edges are shared between processors when a domain decomposition is used to split up the data for distributed computation. For non-conventional finite element formulations the shared variables are associated with the shared edges. Two data structures were defined to handle this variety of perspectives about similar problems, the `Part` and the `Block`. Each `Part` corresponds to a logical subdivision in the finite element mesh, keeping the information required for inter processor communication. Each `Block` is the matrix describing the behaviour of the corresponding `Parts`, about which it keeps information.

3.1. Implementation aspects for the `Part` and `Block` structures

The components of the `Part` and `Block` structure are shown in Tab.l. These data structures are created only at subdomain level and they are inserted in chained lists.

```
struct Block
{
   Part *row, *column;
   /* matrix storage */
   int nonzeros, *irow, *icolumn;
   double *vals;
   …
   Block *previous, *next;
};
```

```
struct Part
{
  int id, number;
  int NSubDomains, *SubDomains;
  int Dimension;
  double *x[MAX_VECTORS];
  …
  Part *previous, *next;
};
```

Tab. 1: Data structures for matrix and vector storage.

The coefficients of the matrices are kept in each `Block` using a sparse (row, column, value) and symmetric format.

In the `Part` data structure, information is held about the dimension of the corresponding vectors. The `NSubDomains` field is used to identify if the data is local. The subdomains that share the part are indicated by `*SubDomains`. The variable number holds in a condensed form the same information.

Conventional FEM	Non-conventional FEM
Node$\in$Part	Node$\notin$Part
Edge$\notin$Part	Edge$\in$Part
Loop$\notin$Part	Loop$\in$Part

Tab. 2: Links between topological and `Part` data structures.

The links between the `Local` data structures of the topological entities (which induce structural variables) and the `Part` structures are formulation dependent, as shown in Tab.2 for conventional and non-conventional formulations. Other combinations of these can be carried out.

3.2. Data distribution and mapping

The data distribution scheme established in the experimental implementation is general and applies to any finite element mesh. The subdomain data set includes all `Nodes`, `Edges` and `Loops` of the subdomain and a "halo" copy of all the `Edges` connected to the boundary nodes and all the `Loops` adjacent to the boundary nodes. In this way the information necessary to prepare all communications can be locally computed. The partition of a simple finite element mesh into three subdomains, is exemplified in Fig.l.a.

Each `Part` is responsible for a set of variables. The numbering of the variables in each `Part` starts from zero, thus each `Part` identified by the same number holds the same structural variable set on all processors.

In what follows we will present a simple approach for setting up the Part structures. All mesh entities, which induce structural variables, as defined in Tab.2, are checked and the following variables, which identify the corresponding `Parts` are computed:

- the number of the subdomains which share that entity;
- the sutidomain list;

- a unique number associated to the Bach subdomain list;
- the number of associated variables;

A search of the `Part` fist is done to check if a `Part` with the same unique `number` already exists. If such a `Part` exists, the `Part` pointer is set to the mesh entity in the `Local` structure. The offset value of the variable or variables set is also determined. If the search result is zero, then a new `Part` will be created and inserted in the `Part` list.

The `Parts` for a non-conventional formulation and for a conventional (variables associated with nodes) formulation are identified on Fig.l.a and Fig.l.b, respectively.

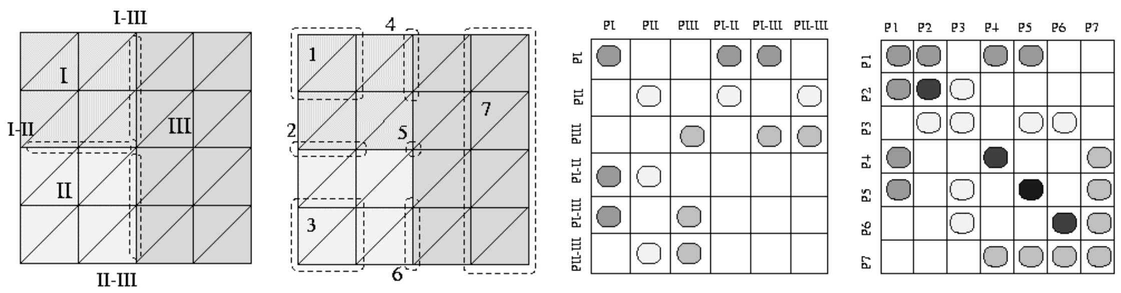


Fig. 1: Parts identification and global structural matrix building blocks for non-conventional and conventional formulations.

The system in each subdomain is described by a matrix of pointers to `Blocks` with *n* x *n* dimension, where *n* is the number of `Parts` in each subdomain. After the structural matrix computation has been completed the empty blocks are eliminated. For non-conventional formulations the contributions of each edge/element do not overlap with other contributions, see Fig.l.c, unlike for conventional formulations where the coefficients are obtained by summing the contributions of adjacent elements, represented by `Part2`, `Part4`, `Part5`, `Part6` in Fig.l.d. This step is performed after inserting the coefficients in each `Block` and reordering them by column and row.

The matrix vector products are performed at subdomain level multiplying each `Block` by the corresponding column Part and adding the result to the row `Part`, as detailed in [5]. After these calculations have been performed the result obtained in each shared `Part` (the ones that belong to more than one subdomain) is sent to the neighbour(s), while expecting to receive as many sets of values as the number of neighbours.

4. DESCRIPTION OF THE FORMULATION

The finite element model used [4] is the implementation of the Stress Hybrid Mixed formulation described in [6] for digital Walsh approximation functions.
The only requirement imposed on the approximation functions by the finite element formulation is that they must span a basis for an L_2 function space. This makes possible the use of these discontinuous functions in the finite element model.

In the resulting governing systems the elementary matrices as well as the structural matrix are very sparse in nature due to the orthogonality of the approximation functions.

Due to the very coarse nature of the digital functions, "accurate" results can only be obtained for meshes with a very high number of degrees of freedom. Normally this is obtained with coarse meshes of elements of high degree.

A condensed form of the system can be obtained by eliminating different sets of variables. Here a condensed form of the governing system is used after an elimination of the internal degrees of freedom of each subdomain. The resulting system is the Schur complement of the shared variables, which is used in the parallel computation.

5. ALTERNATIVE CG METHODS FOR THE PARALLEL SOLUTION OF THE FINITE ELEMENT GOVERNING SYSTEM

The Conjugate Gradient algorithm [8] is widely used to solve linear systems $\boldsymbol{A}\,\boldsymbol{x} = \boldsymbol{b}$, where $\boldsymbol{A}$ is a symmetric positive definite matrix. The method does not require a factorisation of the matrix, and is well suited to parallelisation.

The standard non-preconditioned CG method requires one matrix vector multiplication, two global inner products and three global vector updates at each iteration step. In a distributed environment the two inner products and the matrix vector multiplication require communications between the processors.

In practical applications, to speed up convergence, the CG is used in combination with a preconditioner [10]. If matrix $\boldsymbol{M}$ approximates the coefficient matrix $\boldsymbol{A}$, then system $\boldsymbol{M}^{-1}\boldsymbol{A}\,\boldsymbol{x} = \boldsymbol{M}^{-1}\boldsymbol{b}$ has an improved condition number and, consequently, faster convergence.

Various variants of the CG method were considered, namely, a global CG approach with and without preconditioner and the Schur complement approach on the shared variables without preconditioning.

The parallel implementation of the CG algorithm was designed on a master-slave scheme [5]. The input data is read by the master from a data file, processed and sent to the slave processors. Concurrent elementary stiffness matrix evaluation is done and distributed CG iterations are performed to solve the resulting system. Local and global communications and synchronisation between the processors are performed at each iteration step using the `PVM` message passing library [7]. After each iteration a test of convergence is performed simultaneously by each processor. When convergence is achieved, the slaves can proceed with the pre-processing step. After this step, if there is no more work to be done the slaves exit and the computation is finished on the master.

For the non-conventional formulation tested the intervening matrices are usually very sparse and the synchronisation of the variables is the factor that mostly affects the global performance of the parallel CG solver. In order to avoid unwanted synchronisation points, different alternative CG schemes were tested. Most of them are affected by rounding errors and require the positive definiteness of $\boldsymbol{A}$.

A rearranged version of the standard CG algorithm presented in [2] was chosen. This scheme allows to overlap communication time with useful computations and is as stable as the original algorithm.

Different solvers were tested to solve the internal system. Best results were obtained with a direct solver, namely the `MA47` Harwell routine.

6. NUMERICAL EXAMPLES

To test the performance of the parallel implementation a simple example, consisting in the static analysis of a square cantilever plate is used. The tests were performed on a single machine and on a cluster of 4 personal computers, all equipped with 233MHz Pentium II processors, 132Mb memory, running PVM under Linux. These machines were connected via 100 Mbit Ethernet.

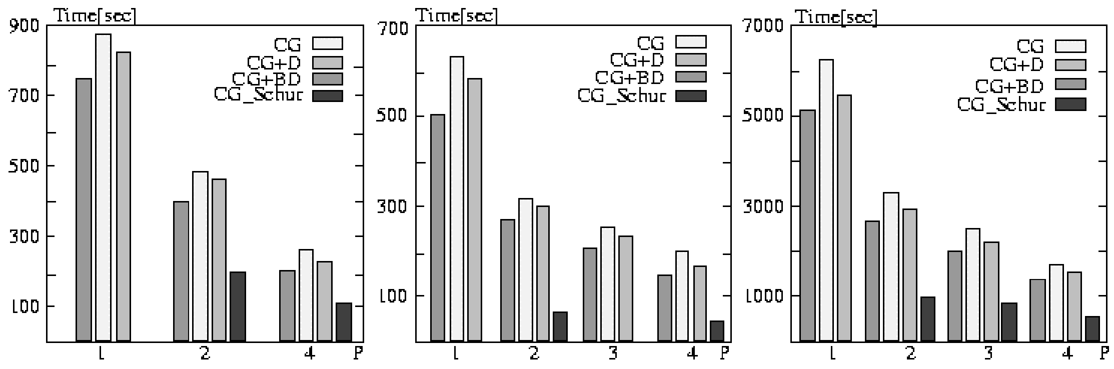


Fig. 2: Execution times for different problem sizes with various CG methods.

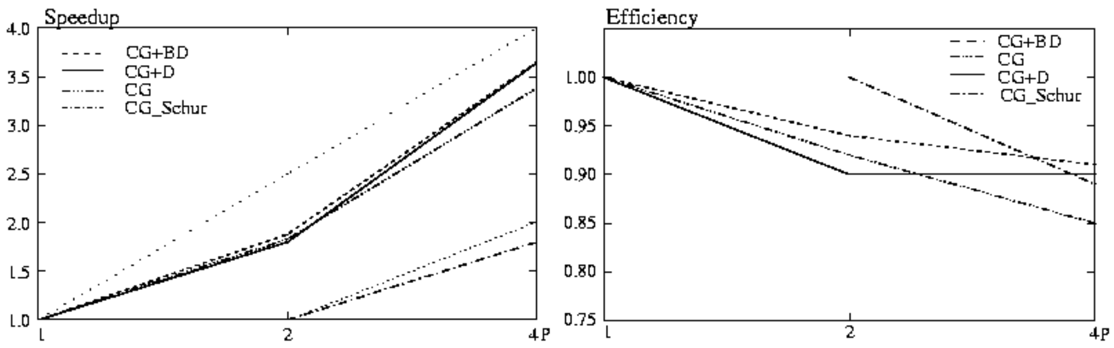


Fig. 3: Speedup and efficiency values for the 4-element mesh with 58604 dof.

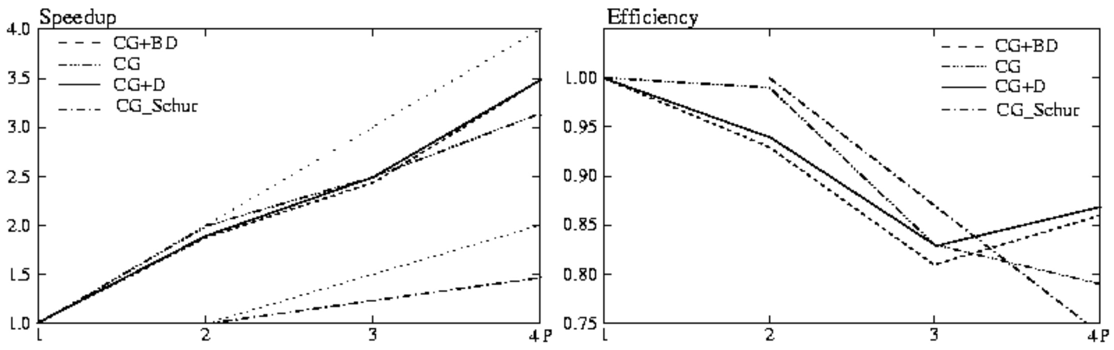
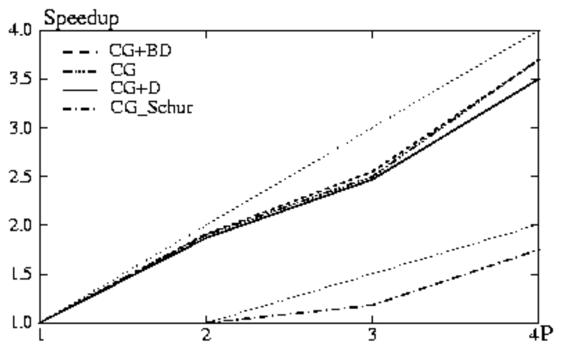


Fig. 4: Speed up and efficiency for the 16-element mesh with 59576 dof.

In Fig.2 the wall-clock times versus number of processors, *P*, are presented for different problem sizes (Fig.2.a for a 4-element mesh with 58604 dof, Fig.2.b and 2.c for a 16-element mesh with 59576 and 233912 dof, respectively) using various CG alternatives. All the procedures perform similarly. The small differences rise from the smaller number of iterations to achieve the convergence for CG with diagonal, CG+D, and block diagonal, CG+BD, preconditioners. The CG on the Schur complement, CG_Schur, is only used for more than one subdomain. This procedure gives the best results in terms of the total execution time.



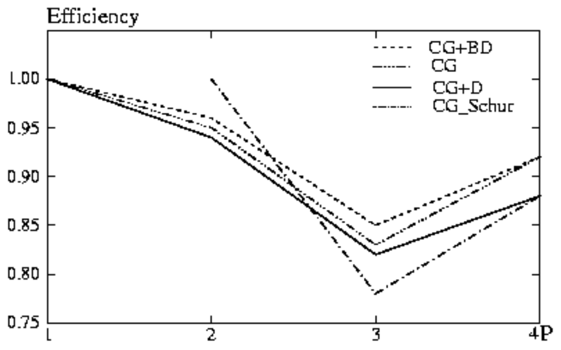


Fig. 5: Speed up and efficiency for the 16-element mesh with 233912 dof.

The speedup and efficiency values obtained from the data represented in Fig.2 are shown in Fig.3 to 5. For the CG_Schur the reference values are taken for P=2. As the number of elements/subdomains is small, the load distribution for three processors is necessarily unbalanced, and in this case, there is a decrease of the parallel performance, (Figs.4 and 5).

A second set of tests was run using the simplest constant stress triangular element for the same square cantilever plate discretized in 5098 (Mesh A) and 20198 (Mesh B) elements, respectively. For these tests, which illustrate the successful use of the data structures described, the total execution wall-clock Limes, *Tt*, the speedup, *Sp*, and efficiency, *Ef*, values obtained are shown in Tab.3.

	Mesh A - 5198 dof			Mesh B - 20396 dof		
P	*Tt [sec]*	*Sp*	*Ef*	*Tt [sec]*	*Sp*	*E f*
1	12.32	1.00	1.00	86.26	1.00	1.00
2	9.70	1.27	0.63	48.04	1.80	0.90
3	8.15	1.51	0.50	41.72	2.07	0.69
4	11.20	1.10	0.28	38.35	2.25	0.56

Tab. 3: Total execution times, speedup and efficiency values for different problem sizes for nodal formulation.

The values obtained show a "normal" behaviour for problems with this dimension. For small sizes there is a decrease of the parallel performance because when the number of processors is increased, most of the time is spent on inter-processor communications.

7. CONCLUSIONS

The results obtained seem to confirm the initial expectations and satisfy the proposed goal of generality, though the performance is not optimal in certain conditions. The use of other message passing interfaces, namely `MPI`, or a "real" `MIMD` environment, may provide a better understanding about the behaviour of the model.

The Schur complement approach provides the best execution times. Various preconditioners proposed for this strategy [10] will further improve its performance. Current work is focused on extending the solver to other non-conventional formulations [6].

8. ACKNOWLEDGMENTS

This work is part of the research activity of ICIST and was supported by the grant PRAXIS XXI - BD/9754/96.

9. REFERENCES

[1] Almeida, J. P. NI. and Freitas, J. A. T. (1995), "Some aspects on the parallel implementation of non conventional finite element formulations" . in D. R. J. Owen and E. Oñate, eds, *Computational Plasticity: Fundamentals and - Applications, Proceedings of the Fourth International Conference, Part.2,* pp.2027-2035.

[2] Barrett, R., Berry, VL, Chan, T. F., et al. (1994), *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods,* SI AKI. Philadelphia, PA.

[3] Baumgart, B. 6.(1975), "A polyhedron representation for computer vision", in *Proceedings of National Computer Conference,* Vo1.44, pp.589-596.

[4] Castro, L. M. S.(1996), *Wavelets e Séries de Walsh em Elementos Finitos,* Ph.D Thesis, Universidade Técnica de Lisboa, Instituto Superior Técnico, Lisbon.

[5] Cismasiu, L. de Almeida, J. M., Castro, L., and Harbis. D. (1998), "Parallel solution techniques for hybrid mixed finite element models"' ; in M. Papadrakakis and B. Topping. eds, *Innovative Computational Methods for Structural Mechanics.* Saxe Coburg Pub.

[6] Freitas, J. A. T., Almeida, J. P. M., and Pereira, E. M. B. R. (1996), "Non-conventional formulations for the finite element method", *Structural Engineering and Mechanics.* Vol.4, pp. 655-678.

[7] Geist, A., Beguelin, A., Dongarra, J., et al. (1994), *PVM: Parallel virtual machine - A users' guide and tutorial for networked parallel computing,* Scientific and Engineering Computation, MIT Press, Cambridge.

[8] Hestenes, M. R. and Stiefel, E.(1952), "Methods of conjugate gradients for solving linear systems", *Journal of Research of the National Bureau of Standards,* Vo1.49, pp. 409-436.

[9] Mantÿla, M. (1988), *An Introduction to Solid Modeling*, Computer Science Press, Rockville, Maryland.

[10] Papadrakakis, M. (1993), "Solving large scale linear problems in solid and structural mechanics", in M. Papadrakakis, ed., *Solving large scale problems in mechanics - The development and application of computational solution methods,* pp.l-37.

[11] Piteri, M. A. and Almeida, J. P. M. (1995), "Hierarchical 2d mesh generation using a topological data structure" in E. R. d. Arantes e Oliveira and J. Bento, eds., *Proceedings of the Fifth EPMESC Conference, Macao; 1-4 August 1995,* Vol.2. pp. 981-986.