FE-BUI – FINITE ELEMENT BEOWULF USER INTERFACE: A HOMEMADE PACKAGE FOR AUTOMATED PARALLEL FINITE ELEMENT COMPUTATIONS

Antony N. Spyropoulos, Athanasios G. Papathanasiou, John A. Palyvos, and Andreas G. Boudouvis*

School of Chemical Engineering
National Technical University of Athens
Zografou Campus, 15780 Athens, Greece
*e-mail: boudouvi@chemeng.ntua.gr, web page: http://www.chemeng.ntua.gr/dep/boudouvis

Keywords: Beowulf Clusters, Parallel Computations, Finite Element, Krylov, Preconditioning, Message Passing

Abstract. *A computational package, the FE-BUI, is introduced for automated parallelization of finite element codes. The package has been developed at the Computer Center of the School of Chemical Engineering of NTUA. Its scope is to provide the ability for parallel execution of serial codes on Beowulf clusters, in an easy and efficient way.*

1 INTRODUCTION

The main computational cost of the finite element codes comes from the solution of large linear algebraic equation systems. Direct (e.g. frontal-type) solvers require large memory and computational cost, and most importantly, their serial parts lack the advantage of exhibiting appreciable parallel efficiency. Recent products of the development of parallelization tools for finite element codes are freely available packages such as the partitioner parMETIS[1] and the solvers Aztec[2] and PETSc[3]. Even with these tools, the effort and cost required for parallelization of a serial code might be prohibitively high. The other alternative, the automated parallelization of serial codes with parallel compilers, yields no more than 10% reduction of the computational cost.

In this paper is presented a new, homemade, parallel package for the solution of finite element problems. This package offers convenience and effectiveness in doing large scale computations since there is no need for the user to learn and implement suitable solvers and communication protocols in parallel computer architectures. The user simply calls, from the serial code, the parallel solver, which takes care of the mesh partitioning, of the load assignment to the available processors and of the parallel solution of the resulting linear systems. The package uses parallel iterative solvers that are based on Krylov projection methods[4] and exploits the architecture of Beowulf clusters using the MPI (Message Passing Interface)[5] for the processors communication. Typical runs with 3D finite element problems on a small, 4-processor cluster yield a reduction of the computational cost by a factor of 3.

2 BEOWULF CLUSTERS

The outcome of the evolution, during the last decade, of the hardware of the personal computers (PCs), mainly in processors, driven by the major companies in this field, Intel and AMD, but also in motherboards and memories, is low-cost and high-performance personal computing. Moreover, commodity computer networks offer high bandwidth and low latency, like Fast Ethernet, Gigabit Ethernet and the more advanced networks Myrinet (http://www.myri.com) and SCI (http://www.dolphinics.com). This progress in conjunction with the development of Linux (http://www.linux.org), a freely available, stable and reliable operating system, enables large scale computations on Beowulf clusters (http://www.beowulf.org). These clusters are computational systems that consist of PCs which are interconnected with a private network.

Beowulf clusters are distributed memory parallel computers, where each processor has a private memory and does not have direct access to the memory of the other processors. Thus, a two-processor communication is required when a processor needs data residing in the memory of another processor. This communication can be done with the MPI, which is a library of subroutines that a programmer calls from a C or a Fortran code. In this case, the parallel execution of a serial code on Beowulf clusters needs the explicit programming of the communication between the processors. This is the main obstacle for the user in converting a serial code to a suitable, for Beowulf clusters, parallel code.

3 THE FE-BUI PACKAGE

The FE-BUI is programmed in Fortran 77. Its installation requires the freely available libraries: BLAS[6], LAPACK[7] and MPICH[8] or LAM/MPI[9]. The main components are shown in Figure 1.

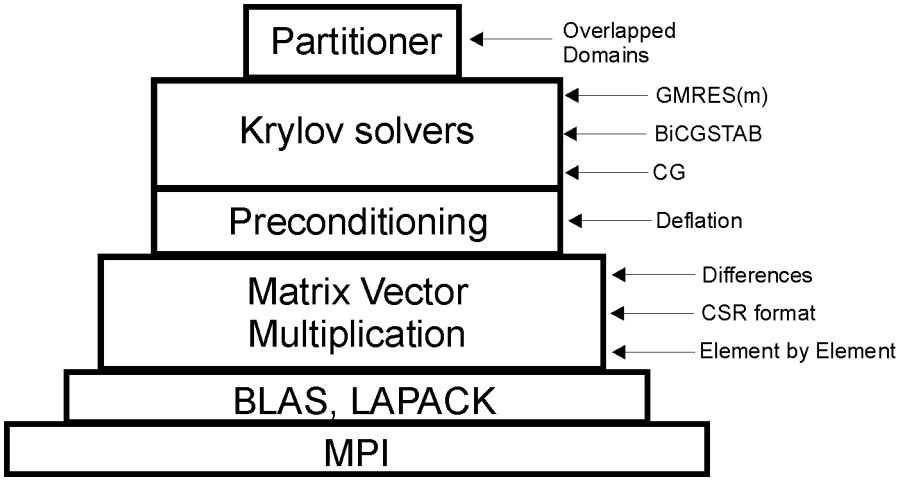

Figure 1. The FE-BUI components

3.1 KRYLOV ITERATIVE SOLVERS

The computational kernel of the FE-BUI is based on the preconditioned Krylov iterative solvers for the solution of large and sparse linear systems, such as the GMERS(m), BiCGSTAB and CG; currently, only the GMRES(m) solver is employed.

The GMRES is preferred for the iterative solution of large algebraic equation sets with non-symmetric matrices, on the basis of its parallel efficiency[10]. Starting from an initial guess, x_o, of the solution of the linear system:

$$Ax = b \tag{1}$$

where $A \in \mathbf{R}^{NxN}$ and $x, b \in \mathbf{R}^{NxN}$, GMRES uses Arnoldi's method[11], combined with an orthogonalization technique – the Modified Gram-Schmidt method is used here – to construct an orthonormal basis $V_m \in \mathbf{R}^{Nxm}$ of the m-dimensional Krylov subspace

$$K_m(A, v) = \text{span}\left\{v, Av, A^2v, ..., A^{m-1}v\right\} \tag{2}$$

where $v \equiv r_o / \|r_o\|_2$, $r_o \equiv b - Ax_o$. The new approximation of the solution is

$$x_m = x_o + V_m y_m \tag{3}$$

where y_m is a vector of size m and it is computed from the solution of the least squares problem

$$y_m = \underset{y}{\text{argmin}} \left\| \beta e_1 - \overline{H}_m y \right\|_2, \; y \in \mathbf{R}^m \tag{4}$$

In eq. (4), $\beta \equiv \|r_o\|_2$, $e_1 = [1,0,...,0]^T$ and $\overline{H}_m \in \mathbf{R}^{(m+1)xm}$ is an upper Hessenberg matrix, such as

$$AV_m = V_{m+1}\overline{H}_m \Rightarrow V_m^T A V_m = H_m \tag{5}$$

$H_m \in \mathbf{R}^{mxm}$ is an upper Hessenberg matrix obtained from the $\overline{H}_m$ by deleting its last row. The least squares problem (4) is solved by transforming $\overline{H}_m$ into an upper triangular matrix $R_m \in \mathbf{R}^{mxm}$ using plane rotations[4].

The storage requirements and the computational cost of Arnoldi's method increase rapidly with m and, thus, a restarting variant of the GMRES – the GMRES(m) – is used in practice. When m reaches a certain preset value, the algorithm restarts, using the last approximation x_m from eq. (3) as a new initial guess. Thus, two iterations are performed: the "inner" m iterations and the "outer" iterations that correspond to the restarts of the

GMRES(m).

3.2 PRECONDITIONING

A preconditioner is essential in enhancing the convergence rate of a Krylov iterative solver. Thus, the original linear system (1) must be transformed to an equivalent one that has better convergence properties. In the FE-BUI the linear system (1) is preconditioned from the right

$$AM^{-1}z = b,\ x = M^{-1}z \tag{6}$$

In eq. (6), z is a vector of size N and $M^{-1} \in \mathbf{R}^{NxN}$ is the preconditioner matrix which is constructed from a deflation technique[12] and it is given by

$$M^{-1} = I_N + U(|\mu|T^{-1} - I_r)U^T \tag{7}$$

where $\mu \in \mathbf{R}$ is the largest eigenvalue of the matrix A, $I_N \in \mathbf{R}^{NxN}, I_r \in \mathbf{R}^{rxr}$ are identity matrices, $U \in \mathbf{R}^{Nxr}$ is an orthonormal basis of the r-dimensional invariant subspace, P_r, corresponding to the r smallest eigenvalues (in terms of the absolute value of their real parts) of the matrix A and $T \in \mathbf{R}^{rxr}$ such as

$$T = U^T AU \tag{8}$$

The largest eigenvalue and the Schur vectors of the matrix A, needed in eq. (7), are approximated by those of the Hessenberg matrix H_m. Thus, at each restart of the GMRES(m), a Schur decomposition of the Hessenberg matrix is performed to approximate the largest eigenvalue and the Schur vectors corresponding to the smallest eigenvalues of the matrix A. These vectors are added to P_r, increasing its dimension. In order to save on computational cost and memory requirements arising from the preconditioning operations, an upper limit, r_{max}, on r is set; when it is reached, the update of the preconditioner stops and the GMRES(m) continues with the same preconditioner. The key idea of this preconditioning technique is to remove by deflation the smallest eigenvalues of the matrix A that cause slow or no convergence of the GMRES(m)[13].

3.3 PARTITIONING

The partitioner of the FE-BUI package is based on an overlapped domains[14] partitioning technique. The original domain, tessellated by the finite element method, is divided into subdomains. A subdomain is defined as a separate group of elements and it is assigned to a processor.

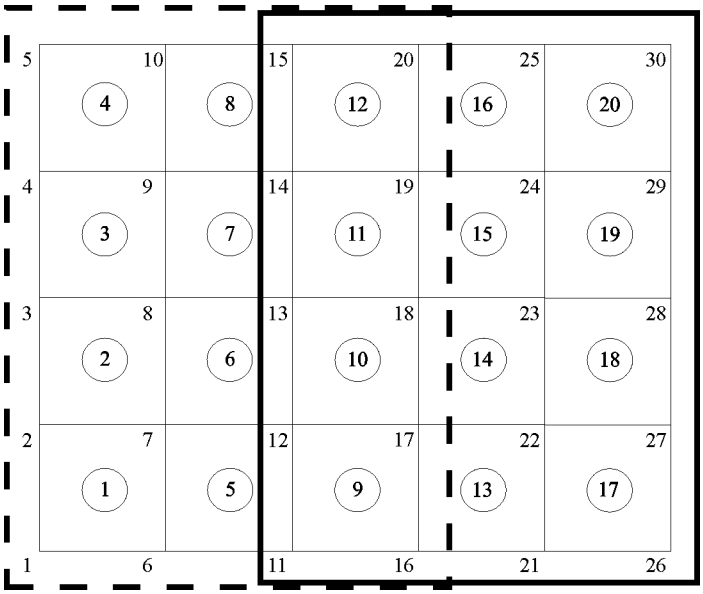

Figure 2. A sample 2D finite element mesh assigned to 2 processors. Dashed and solid lines depict the 2 overlapped subdomains.

In Figure 2 is shown a 2D finite element mesh that is assigned to 2 processors. According to this partitioning

technique, each processor takes 15 nodes, i.e. 15 rows of the matrix A, 1 to 15 for the first processor and 16 to 30 for the second processor. Each processor in order to fully assemble every local node contribution to the matrix A, makes some extra computations to the common elements 9 to 12. The nodes of these elements are called communication nodes.

Thus, in FE-BUI the decomposition of the finite element mesh corresponds to the distribution of the matrix A rows to the processors. It is known that this technique leads to smaller parallel efficiency than that of other domain decomposition techniques[14],[10] (see also section 5), but offers flexibility and usage convenience to the FE-BUI package.

3.4 PARALLEL OPERATIONS IN THE FE-BUI

The basic operations of the GMRES method are: (i) Vector updates, (ii) inner products, (iii) matrix-vector products. Moreover, preconditioning operations are needed for the preconditioned GMRES(m). The performance of these operations depend on the choice of the preconditioner. The deflation preconditioning technique can be analyzed in the same basic operations as the GMRES method. All these operations can be decomposed in tasks and each task can be independently executed on each processor. More details about the parallel implementation of these operations on Beowulf clusters are available in [10]. Briefly, vector updates can be done in parallel without communication between the processors. In order to compute an inner product each processor computes a local inner product. The latter operation is completed through a global communication between the processors to sum up the calculated local inner products. During the global communication, the processors exchange data of length of 8 bytes (the scalar local inner product). For the matrix-vector product, a communication between its processor and its neighbors is required. During the neighboring communication the processors exchange arrays of length equal to the communication nodes. For example, with reference to figure 2, the processors exchange data of length of $5 \cdot 8 = 40$ bytes.

In FE-BUI the matrix-vector product can be done using three methods: a) Compressed Sparse Row Format (CSR) were only the nonzero elements of the matrix A are stored. b) The element-by-element matrix-vector product and c) With a matrix-free approach[15], that is by approximating the elements of A by differencing; this is often the case in nonlinear problems, where A is the Jacobian matrix corresponding to the discretized equations.

4 USAGE OF THE FE-BUI

As a first step with the FE-BUI package, the new user can simply call from his/her serial code the driver subroutine FEBUIdrv replacing the serial solver call. This driver takes care of the solution of the linear system by calling the default partitioning and solver subroutines. At a more advanced level, the user can call selectively the appropriate subroutines.

The required input data of the FEBUIdrv are already computed in most finite element codes. Typical input data are:

a) the total number of the nodes (NN) and the elements (NE) of the mesh,
b) the array NpE of dimension NE – NpE(NE) – that contains the number of nodes of each element,
c) the array NOP(NE,max(NpE)) that associates the local (element level) and global (mesh level) numbering of nodes,
d) the arrays NCOD(NN) and BC(NN), for distinguishing nodes bearing Dirichlet boundary conditions,
e) an approximation to the solution u(NN), in case of nonlinear equation systems.

Also the user must supply the subroutine that computes the element contributions and the right hand side of the linear system.

5 RESULTS

The parallel efficiency of an algorithm is measured by the parallel speedup[14],[16], S, which indicates how faster the algorithm runs using p processors compared to the performance on one processor:

$$S = \frac{\text{Execution time on 1 processor}}{\text{Execution time on p processors}} \tag{9}$$

Ideally, a parallel algorithm must run p times faster when executed on p processors. However, the speedup is limited by the ever-present serial tasks in a parallel algorithm, by the load balancing and by the communication between the processors that is the main factor of a reduced parallel speedup.

The FE-BUI has been tested on the solution of a three-dimensional, nonlinear and free boundary problem of intefacial magnetohydrostatics[17]. The achieved speedup of the preconditioned GMRES(m) is shown in table 1 for two cases:

(i) parallel computations with FE-BUI,

(ii) parallel computations with a parallel code that was developed in [18].

The computations were done on a linear system of 254,857 nodes at a small Beowulf cluster of 4 nodes (http://www.chemeng.ntua.gr/yk/cluster)

CPUs	Case (i)	Case (ii)
1	1	1
2	1.8	1.9
3	2.7	-
4	3.4	3.9

Table 1: Parallel speedup

The achieved speedup is smaller than the ideal, in both cases, due to the communication between the 4 processors. The speedup in (i) is smaller than in (ii) since FE-BUI "ignores" particular aspects[18] of the mesh, resulting in a slightly unbalanced distribution of the mesh to the available processors; in such a case, the under-loaded processors have to remain idle until the over-loaded processors finish their tasks.

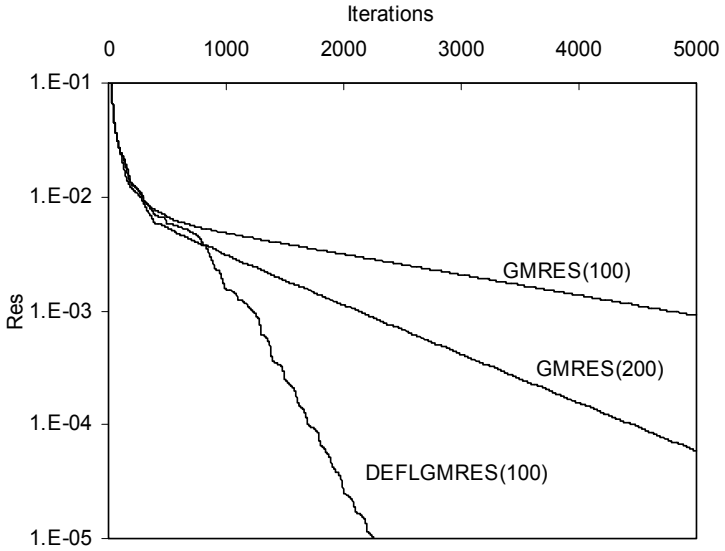

Figure 3. The convergence of the GMRES(m), m = 100, 200 and the DEFLGMRES(100), $r_{max} = 20$.

In Figure 3 is shown the convergence of the GMRES(m) and the preconditioned GMRES(m) by deflation – DEFLGMRES(m). The horizontal axis is the product of m by the restarts of the GMRES(m) or DEFLGMRES(m). The vertical axis is the residual $\text{Res} = \frac{\|b - Ax_m\|_2}{\|b\|_2}$.

Another important feature of the preconditioner of the FE-BUI is that the additional communication load coming from the extra preconditioning operations, has no appreciable effect on the speedup of the GMRES(m) – a significant advantage of the chosen preconditioner compared to a commonly used ILU-type preconditioner[10].

The achieved speedup versus the number of processors for two problem sizes, N=68,377 and N=254,857 is shown in figure 4. The speedup increases with the number of processors significantly faster in big problems than in smaller ones, because in the former case the computational time increases faster than the communication time, as it is noticed also from figure 5. The latter figure shows the relative communication (global and neighbouring) and computational time as percentages of the total execution time of the parallel preconditioned GMRES(m) versus the problem size, when the solver runs on 4 processors.

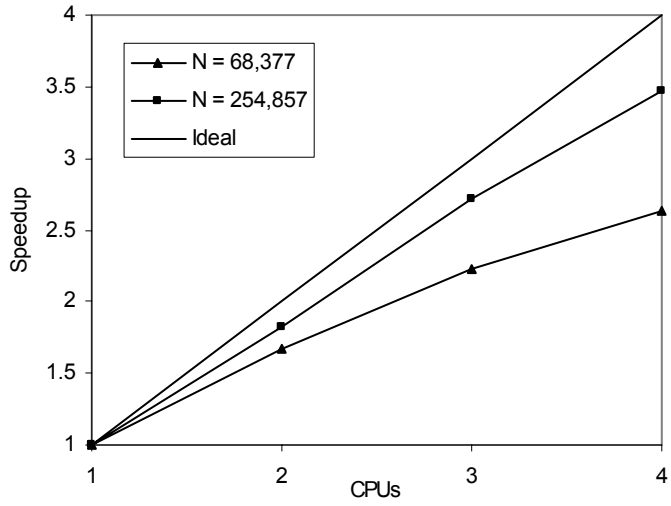

Figure 4. Parallel speedup versus the number of processors for two problem sizes.

The main network overhead comes from the global communication time, although the message length is too small. This is due to the high latency of the Ethernet network of our cluster. The time (t_{comm}) that is spend for a message of n bytes in length is given from:

$$t_{comm} = \mathbf{a} + \mathbf{b} \cdot n \tag{10}$$

where **a** is the latency of the network and **b** is the time for sending 1 byte. Thus, two network related factors limit the communication time: the latency and the bandwidth. Latency limits the exchange of small messages, mainly required in global communication whereas bandwidth limits the exchange of large messages, as happens in the neighboring communication. Thus, for finite element parallel computations with iterative Krylov solvers, a network with small latency is strongly preferred.

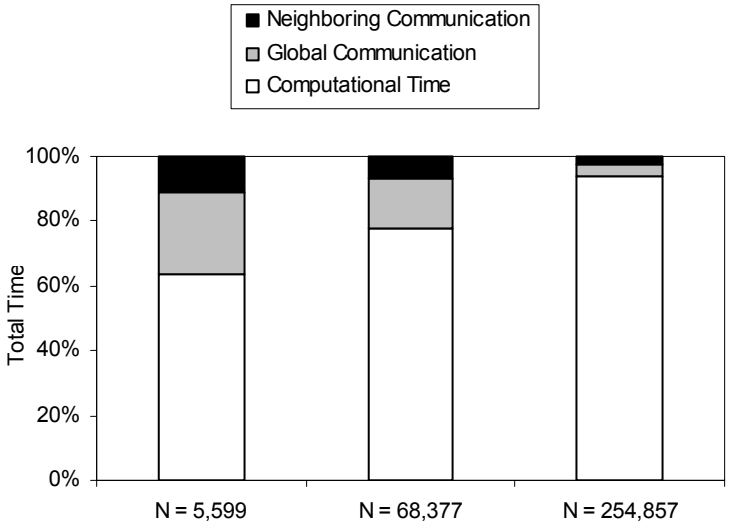

Figure 5. Computational vs Communication time.

More information about the usage and the availability of the FE-BUI, is available at http://www.chemeng.ntua.gr/yk/cluster.

ACKNOWLEDGMENTS

Financial support for this work was provided by the Ministry of Education through the Research Program "Pythagoras" and by the General Secretariat for Research and Technology through the "ENTEP" Program.

REFERENCES

[1] http://www-users.cs.umn.edu/~karypis/metis/index.html
[2] http://www.cs.sandia.gov/CRF/aztec1.html
[3] http://www-unix.mcs.anl.gov/petsc/petsc-2
[4] Saad, Y. (1996), *Iterative methods for sparse linear systems*, PWS Publishing Company. Available to http:// www-users.cs.umn.edu/~saad/books.html
[5] http://www-unix.mcs.anl.gov/mpi/
[6] http://www.netlib.org/blas
[7] http://www.netlib.org/lapack
[8] http://www-unix.mcs.anl.gov/mpi/mpich
[9] http://www.lam-mpi.org
[10] Spyropoulos, A. N., Palyvos, J. A. and Boudouvis, A. G. (2000) "Finite element computations on cluster of PC's and workstations." In *Proceedings of the 8th Euromicro Workshop on Parallel and Distributed Processing – EURO-PDP'2000*, pp. 56-61, Rhodos, Greece, January 2000 (IEEE Computer Society, Los Alamitos, CA, USA).
[11] Arnoldi, W.E. (1951), "The principle of minimized iterations in the solution of the matrix eigenvalue problem", Q. Appl. Math., Vol. 9, pp. 17-29.
[12] Erhel, J., Burrage, K. and Pohl, B. (1996), "Restarted GMRES preconditioned by deflation", J. Comput. Appl. Math., Vol. 69, pp. 303-318.
[13] Van der Vorst, H.A. and Vuik, C. (1993), "The superlinear convergence behaviour of GMRES", J. Comput. Appl. Math., Vol. 48, pp. 327-341.
[14] Smith, B., Bjorstad, P. and Gropp, W. (1996), *Domain Decomposition. Parallel multilevel methods for elliptic partial differential equations*, Cambridge University Press.
[15] Dennis, J. E. and Schnabel, R. B. (1996), Numerical *methods for unconstrained optimization and nonlinear equations*, SIAM, Classics in Applied Mathematics, 16, Philadelphia.
[16] Buyya, R. (ed.) (1999), *High Performance Cluster Computing: Programming and Applications*, Vol. 2, Prentice Hall, NJ, USA.
[17] Spyropoulos, A. N., Palyvos, J. A. and Boudouvis, A. G. (2004), "Bifurcation detection with the (un)preconditioned GMRES(m)", Comput. Methods Appl. Mech. Engrg., Vol. 193, pp. 4707-4716.
[18] Spyropoulos, A. N. (2003), *Large scale computations with parallel processing methods in nonlinear problems of interfacial magnetohydrostatics*, Doctoral Thesis (in Greek), NTUA.