PARALLEL SOLUTION OF EIGENPROBLEMS IN STRUCTURAL DYNAMICS USING THE IMPLICITLY RESTARTED LANCZOS METHOD

GEORGE O. AINSWORTH JR.* FERNANDO L. B. RIBEIRO†
CARLOS MAGLUTA†

ABSTRACT: This paper presents a parallel implementation of the implicitly restarted Lanczos method for the solution of large and sparse eigenproblems that occur in modal analysis of complex structures using the finite element method. The implicitly restarted technique improves convergence of the desired eigenvalues without the penalty of lost of orthogonality keeping the number of factorization steps in a modest size. In the parallel solution, a subdomain by subdomain approach was implemented and overlapping and non-overlapping mesh partitions were used. Compressed data structures in the formats CSRC and CSRC/CSR were employed to store the global matrices coefficients. The parallelization of numerical linear algebra operations presented in both Krylov and implicitly restarted methods are discussed.

Keywords:generalized eigenvalue problem, Lanczos method, finite elements

1 INTRODUCTION

A standard finite element discretization of ellipitic eigenvalue problems leads to the algebraic form:

$$Ku^h = \lambda^h M u^h \tag{1}$$

where K is the stiffness matrix, M is the mass matrix, the parameter h is the characteristic mesh size, λ^h is the approximate eigenvalue and u^h is the finite-dimensional eigenvector approximation. Due to the nature of fem approximations, the matrices K and M are real, sparse, symmetric and positive definite.

Krylov subspace based methods, such as Lanczos and Arnoldi [1, 2] algorithms, have been widely used for solving eigenproblems associated with large sparse systems. The Krylov subspace associated with a matrix A and a nonzero starting vector v_1 is defined as:

$$\mathcal{K}(A, v_1, k) = span\left\{v_1, Av_1, A^2v_1, \cdots, A^{k-1}v_1\right\} \tag{2}$$

where k is the order of Krylov subspace. Both Lanczos and Arnoldi methods are intended to generate an orthogonal basis for Eq. (2) in such way that the projection of the original matrix into this basis gives a smaller and easier problem to solve (the projected matrix is tridiagonal

*Programa de Engenharia Civil, Universidade Federal do Rio de Janeiro, COPPE, Caixa Postal 68506, Rio de Janeiro, RJ, 21945-970, Brazil. Email: george@coc.ufrj.br,fernando@coc.ufrj.br,magluta@coc.ufrj.br

or Hessenberg if Lanczos or Arnoldi methods is respectively used). It's well known that for orthogonal projections of symmetric problems, Arnoldi and Lanczos factorizations are, mathematically, the same method. Nowadays, the implicitly restarted Lanczos method with spectral transformation such as

$$(K - \sigma M)^{-1} M u^h = \theta u^h \tag{3a}$$

where,

$$\theta = \frac{1}{\lambda^h - \sigma}, \sigma \neq \lambda^h \tag{3b}$$

represents the state of art of numerical solution of eigenproblems [3], [4] and [5].

Algorithm (1) performs a k-step Lanczos factorization:

Input: k_0, j, K, M, $V_{k_0} = [v_1, v_2, \cdots, v_{k_0}]$, T_{k_0}, f_{k_0}

$k = k_0 + j$;

if $(k_0 = 0)$ **then**

Initial vector generation: v_1, with $||v_1|| = v_1^T M v_1 = 1$; $w_1 = K^{-1} M v_1$; $T_1 = v_1^T M w_1$;

$f_1 = w_1 - T_1 v_1$; $k_0 = k_0 + 1$;

end

for $i = k_0, \cdots, k-1$ **do**

$\beta_i = ||f_i||_M$;

$v_{i+1} = f_i / \beta_i$;

$V_{i+1} = [V_i, v_{i+1}]$;

$w_{i+1} = K^{-1} M v_{i+1}$;

$\hat{T}_i = \begin{bmatrix} T_i \\ \beta_i e_i^T \end{bmatrix}$;

$t = V_{i+1}^T M w_{i+1}$;

$T_i = \begin{bmatrix} \hat{T}_i & t \end{bmatrix}$;

$f_{i+1} = w_{i+1} - V_{i+1} h$;

Reorthogonalization Process.

end

$k_0 = k$

Algorithm 1: k-steps of Lanczos factorization ($\sigma = 0$)

Some important details should be mentioned: in the spectral transformation an algebraic linear system was solved with PCG (preconditioned conjugate gradient) and the DGKS [6] method was used to maintain the numerical orthogonality of the basis vectors.

A disadvantage of Lanczos/Arnoldi methods is that one can't know, a priori, how many steps of Algorithm (1) must be performed until the desired eigenvalues and eigenvectors become well approximated by Ritz values and vectors. The increasing of the number of steps leads to a numerical difficulty: the lost of orthogonality of basis vectors V_k, especially for large systems. Also, memory requirements to store V_k begin to be an issue because to recover the eigenvectors of original problem, one must perform a dense matrix-vector operation involving V_k and each eigenvector p of the reduced problem.

To overcome these difficulties, restarting techniques [7, 8, 9, 10] are employed to keep k in a modest size by applying polynomial filters and then, restarting the factorization from a chosen m-th step. Once the $k - m$ steps are rebuild, one obtains an updated k-th step factorization.

The IRLM (Algorithm 2) combines the k-step Lanczos factorization with a implicitly shifted QR iteration [11]. The spectrum of T_k is evaluated and separated in two disjoints sets, one formed with the m approximations of wanted eigenvalues and another with the unwanted portion of the spectrum. The $k-m$ unwanted eigenvalues are sorted out and used as shifts in the QR process. This process is repeated until the contribution of the unwanted modes is damped and then, the m approximations of the wanted eigenvalues fulfill a convergence criteria.

Input: K, M, V_k, T_k and f_k obtained after a k-step Lanczos Factorization
while *convergence = false* **do**
  Compute the spectrum of T_k and $k-m$ "shifts" $\mu_1, \mu_2, \cdots, \mu_{k-m}$;
  $q^H = e_k^T$;
  for $j = 1, 2, \cdots, k-m$ **do**
    Factor $[Q, R] = qr\,(T_k - \mu_j I)$; $T_k = Q^H T_k Q$; $V_k = V_K Q$; $q = q^H Q$;
  end
  $f_k = \beta_k v_{k+1} + ||q||_M f_k$;
  $k_0 = k - m$;
  $V_{k_0} = V_k\,(1:n, 1:k_0)$;
  Beginning with the k_0-step Lanczos Factorization and apply m shifts to obtain a new k-step factorization;
end

Algorithm 2: The implicitly restarted Lanczos algorithm

The convergence criteria adopted for Algorithm (2) was:

$$||f_k|||e_k^T p| \leq tol \cdot |\lambda^h| \tag{4}$$

where tol is a user supplied tolerance. For a more detailed description of the implicitly restarted Lanczos method, the authors recommend [12] and [4].

In this work, we present a parallel implementation of implicitly restarted Lanczos Method for distributed memory architectures. The proposed parallel implementation is based on a subdomain by subdomain (SBS for short) approach and it is suitable for finite element codes using compressed data structures to store the matrix coefficients. The SBS method is an alternative to domain decomposition schemes and the main advantage of SBS method is that both sequential and parallel codes are the same and thus the convergence remains unchanged. Compressed data structures in the formats CSRC and CSRC/CSR [13] were employed for storing global matrices of the finite element method. An example of structural dynamics is presented in order to point out the applicability of the implementation.

2 PARALLEL IMPLEMENTATION

An efficient parallel FEM code should demand communication only in the solution phase. The two mesh partitioning schemes that will be shown demand no communication during the assemblage of global matrices. In the non-overlapping scheme, this is acquainted keeping all coefficients referring to the nodes lying in the internal boundary partially computed, while in the overlapping partitions, all coefficients are global because all information needed to compute the coefficients are available.

In the implicitly restarted Lanczos method, redundant work regarding to the tridiagonal matrix eliminate the needed of communication during the restarting scheme. In practical situations,

the total number of equations must be much greater than the order of the Krylov subspace and in this case the redundant work is irrelevant. The generalized eigenvalue problem requires the solution of a system of algebraic equations. In the symmetric case, the solution was obtained with a parallel version of the PCG (Preconditioned Conjugate Gradient) method.

2.1 NON-OVERLAPPING PARTITIONING

In the non-overlapping partitioning scheme, the original domain Ω is sub-divided in *n* sub-domains $\Omega^i, (i = 0, \ldots, n-1)$, where n denotes the number of processes, in such way that,

$$\Omega = \bigcup_{i=0}^{n-1} \Omega^i; \tag{5}$$

$$\Omega^i \cap \Omega^j = \emptyset \; \forall i \neq j \tag{6}$$

Let $\partial\Omega$ the original boundary of Ω. $\partial\Omega^i$ is the boundary of Ω^i. The closure of a partition is given by $\bar{\Omega}^i = \Omega^i \cup \partial\Omega^i$. The internal boundaries of Ω is represented by $\partial\Omega_{int}$.

Considering the mesh $M_{\bar{\Omega}}$ as a discretization of $\bar{\Omega}$ and supposing that it is possible to obtain a mesh of the non-overlapping partition $M_{\bar{\Omega}^i} \subset M_{\bar{\Omega}}$, each subdomain Ω^i will contain a set formed by nel^i element E^i and a set N^i formed by $nnode^i$ nodes,

$$N^i = \{n_1^i, n_2^i, \ldots, n_{nnode^i}^i\} \tag{7}$$

$$E^i = \{e_1^i, e_2^i, \ldots, e_{nel^i}^i\} \tag{8}$$

Each node located at $\partial\Omega_{int}^i$ belongs to a single partition, but it may appears in others partitions. So, the set N^i will be formed by three subsets: a subset $N^{i,1}$ formed by all nodes belonging to the interior of Ω^i or placed at $\partial\Omega$, a subset $N^{i,2}$ of node placed at $\partial\Omega_{int}^i$ and belonging to the partition i and the last subset $N^{i,3}$ formed by the nodes placed at $\partial\Omega_{int}^i$ but each node belongs to a partition j for all $j \neq i$.

$$N^i = \{N^{i,1}, N^{i,2}, N^{i,3}\} \tag{9}$$

The set formed by the nodes of the original mesh is given by:

$$N = \bigcup_{i=0}^{n-1} \{N^{i,1}, N^{i,2}\} \tag{10}$$

and the set of all nodes placed at $\partial\Omega_{int}$ is

$$N^{\partial\Omega_{int}} = \bigcup_{i=0}^{n-1} N^{i,2} \tag{11}$$

Figure (1) shows schematically four non-overlapping partitions of a domain Ω.

The resulting global matrices obtained from a non-overlapping partition is given by:

$$A^i = \begin{bmatrix} A_{11}^i & A_{12}^i & A_{13}^i \\ A_{21}^i & A_{22}^i & A_{23}^i \\ A_{31}^i & A_{32}^i & A_{33}^i \end{bmatrix} \tag{12}$$

where the blocks $A^i_{11}, A^i_{12}, A^i_{13}, A^i_{21}$ and A^i_{31} are globally computed, while the remaining blocks have partially computed coefficients.

2.2 OVERLAPPING PARTITIONS

To define the overlapping partitioning scheme, a new set, namely $N^{i,4}$ formed by nodes of other partitions is defined. Finally, the overlapping partition is obtained by the addition to the non-overlapping partition mesh $M_{\bar{\Omega}}$ of a set $E^{i,*} = \{e^j_k\}$, $j \neq i$, formed by elements that connect nodes belonging to $N^{i,2}$ and $N^{i,4}$, the elements of an overlapping partition is given by:

$$\widetilde{E}^i = \{E^i \cup E^{i,*}\} \tag{13}$$

$$\widetilde{N}^i = \{N^{i,1}, N^{i,2}, N^{i,3}, N^{i,4}\} \tag{14}$$

For simplicity reasons, it should define a set $N^{i,1a} \subset N^{i,1}$ with nodes belonging to $N^{j,4}$, for some $j \neq i$. A overlapping partition is shown in Figure (2).

Global matrices resulting from overlapping partitions are rectangular and all its coefficients are globally computed,

$$A^i = \begin{bmatrix} A^i_{11} & A^i_{12} & A^i_{13} & 0 \\ A^i_{21} & A^i_{22} & A^i_{23} & A^i_{24} \end{bmatrix} \tag{15}$$

2.3 PARALLELIZATION OF NUMERICAL LINEAR ALGEBRA OPERATIONS

The orthogonal projection procedure used in Krylov subspace methods may be implemented using the Gram-Schmidt orthogonalization process. The main operations present in the implicitly restarted Lanczos method can be classified, according to the amount of floating point operations, in three level: inner products and vector updates belong to level one, matrix-vector (matvec for short) operations are classified as level two and matrix-matrix product operations belong to level three. Since all operations of the numerical linear algebra are reduced to the three levels previously discussed, each level was parallelized as follows.

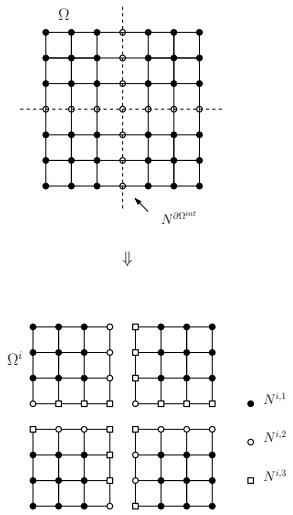


FIGURE 1: Example of a non-overlapping mesh partitioning scheme

2.3.1 LEVEL 1

In both non-overlapping and overlapping partitions, the inner product needed in the orthogonalization process is performed in parallel with coefficients of type 1 and 2:

$$\alpha^i = u^i \cdot p^i = u^{i,1} \cdot p^{i,1} + u^{i,2} \cdot p^{i,2} \tag{16}$$

The local products α^i are accumulated in all processes after an execution of the MPI collective communication routine *MPI_ALLREDUCE*:

$$\alpha = \sum_{i=0}^{n-1} \alpha^i \tag{17}$$

In non-overlapping partitions, vector updates are performed over coefficients of types 1,2 and 3 at the cost of some of some redundancy in terms of floating point operations due to coefficients of type 3:

$$\begin{bmatrix} u_1^i \\ u_2^i \\ u_3^i \end{bmatrix} = \begin{bmatrix} u_1^i \\ u_2^i \\ u_3^i \end{bmatrix} + \alpha \begin{bmatrix} p_1^i \\ p_2^i \\ p_3^i \end{bmatrix} \tag{18}$$

In the overlapping case, the same operation is carried out over the equations of type 1 and 2:

$$\begin{bmatrix} u_1^i \\ u_2^i \end{bmatrix} = \begin{bmatrix} u_1^i \\ u_2^i \end{bmatrix} + \alpha \begin{bmatrix} p_1^i \\ p_2^i \end{bmatrix} \tag{19}$$

In both cases, no communication is needed.

2.3.2 LEVEL 2

Level 2 operations can be divided in two cases: the first case involves a sparse global coefficient matrix and a second case involves a dense matvec product present in the Krylov basis

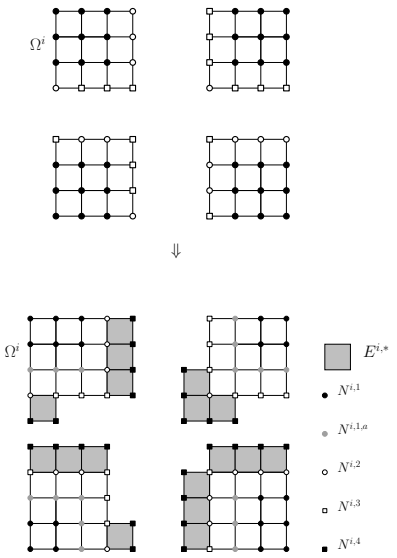


FIGURE 2: Example of an overlapping mesh partitioning scheme

generation and in the matrices updates needed to restart the factorization.

In the non-overlapping method, the sparse matvec product is performed as follows:

$$\begin{bmatrix} p^{i,1} \\ p^{i,2} \\ p^{i,3} \end{bmatrix} = \begin{bmatrix} A^i_{11} & A^i_{12} & A^i_{13} \\ A^i_{21} & A^i_{22} & A^i_{23} \\ A^i_{31} & A^i_{32} & A^i_{33} \end{bmatrix} \begin{bmatrix} u^{i,1} \\ u^{i,2} \\ u^{i,3} \end{bmatrix} \tag{20}$$

Once this matvec product is performed, $p^{i,2}$ and $p^{i,3}$ are partially computed. In order to advance in the itereative process, $p^{i,2}$ and $p^{i,3}$ must be globally computed. Since neighbors coefficients are complementary, communication must be carried out. The adopted point-to-point communication scheme [14] is based in non-blocking MPI subroutines *MPI_ISEND* and *MPI_IRECV* and the received coefficients are mapped and accumulated in their correct positions.

In the overlapping case, the matvec product is performed as follows:

$$\begin{bmatrix} p^{i,1} \\ p^{i,2} \end{bmatrix} = \begin{bmatrix} A^i_{11} & A^i_{12} & A^i_{13} & 0 \\ A^i_{21} & A^i_{22} & A^i_{23} & A^i_{24} \end{bmatrix} \begin{bmatrix} u^{i,1} \\ u^{i,2} \\ u^{i,3} \\ u^{i,4} \end{bmatrix} \tag{21}$$

The point-to-point communication in this case is very similar to the communication of the non-overlapping case. Before the matvec product, the coefficients $u^{i,3}$ and $u^{i,4}$ must be retrieved form the neighbor partitions where they are coefficients of type $1a$ and 2. After communication, a the received coefficients are mapped to their correct positions in vector u and then the matvec operation is performed.

All level 2 operations involving the dense matrix V_k need no communication because the columns of V_k are obtained after the sparse matvec product, this applies to both non-overlapping and overlapping cases.

2.3.3 LEVEL 3

In the restart scheme, level 3 operations must be performed in order to update T_k and V_k:

$$T = Q^T T Q \tag{22}$$

$$V = VQ \tag{23}$$

Recall that the matrix Q is a factor of T_k and since those matrices are replicated, no communication is needed. As a penalty, redundant work is done but while the total number of steps remains in a modest size, the amount of redundant work is irrelevant.

3 COMPRESSED DATA STRUCTURES

As mentioned before, the matrices K and M are symmetric and sparse. To take advantage of sparsity, a compressed data structure was used to store the coefficients. The chosen CSRC scheme [13] is a modified version of CSR [15], in which the lower part of both K and M (since

K and M have the same graph) are stored by rows and the upper part is stored by columns. For symmetric matrices only one array is used for storing the lower part. The diagonal elements are stored in an array ad. The array ja of size nad (number of non-zero coefficients) contains the column indices and the array ia of size $neq + 1$ (neq indicates the number of equations) points to the first coefficient of each row. For symmetric matrices the matvec algorithm is shown in Algorithm (3):

Input: ad, al, ja, ia, p, u
`/* Loop over the equations: */`
for $i = 1, neq$ **do**
  $ui = u(i)$;
  $t = ad(i) * ui$;
  `/* Loop over the lower non-zero entries of equation` i`: */`
  for $k = ia(i), ..., ia(i + 1) - 1$ **do**
    $jak = ja(k)$;
    $s = al(k)$;
    $t = t + s * u(jak)$;
    `/* Upper column: */`
    $p(jak) = p(jak) + s * ui$;
  end
  $p(i) = t$;
end

Algorithm 3: Matvec product ($p = Au$) using CSRC

As shown in the previous section, overlapping partitioning results in rectangular global matrices. For these cases, in order to perform the matvec product, the coefficient matrix is split and stored in two formats: the square matrix is stored in CSRC format and the rightmost matrix is stored in the classical CSR format. Algorithm (4) shows the matvec product performed in this format,

Input: ad, al, a, ja, $ja1$ ia, $ia1$, p, u
`/* Loop over the equations: */`
for $i = 1, neq$ **do**
  $ui = u(i)$;
  $t = ad(i) * ui$;
  `/* Loop over the lower non-zero entries of equation` i`: */`
  for $k = ia(i), ..., ia(i + 1) - 1$ **do**
    $jak = ja(k)$;
    $s = al(k)$;
    $t = t + s * u(jak)$;
    `/* Upper column: */`
    $p(jak) = p(jak) + s * ui$;
  end
  for $k = ia1(i), ia1(i + 1) - 1$ **do**
    $jak = ja1(k)$;
    $t = t + a(k) * u(jak)$;
  end
  $p(i) = t$;
end

Algorithm 4: Matvec product ($p = Au$) using CSRC/CSR

4 RESULTS

In order to point out the potentialities of the presented implementation of IRLM, we solve two generalized eigenvalue problems within the context of structural dynamics. This example was simulated in a cluster with 32 nodes with Intel Core2Duo processors and 8Gb of RAM. The nodes are connected by a 1 Gigabit switch and all machines have Fedora Linux as operating system.

The 15 lowest eigenvalues and eigenvectors of a numerical model of a hydroelectric power plant were calculated. A 60 steps Lanczos factorization was built and 2 implicitly restarts were performed with a tolerance of 1×10^{-13} . Figure (3) shows, in detail, the finite element mesh.

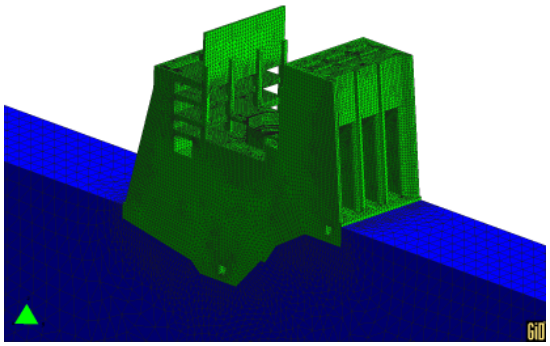


FIGURE 3: Mesh used in the symmetric example

Table (1) has a summary of the numerical properties of the model.

TABLE 1: Main information of the symmetric example

Number of elements	899,104
Number of equations	546,587
Number of iterations of the Implicitly Restarted Lanczos method	2
Number of solutions of the algebraic system of equations (PCG)	115

The speedups curves obtained for the global matrices assembling phase and the implicitly restarted Lanczos method using non-overlapping and overlapping partitions are shown in Fig. (4) . As mentioned in the previous section, in this phase no communication is needed and superlinear speedups were obtained. Non-overlapping partitions obtained better speedups because in the overlapping scheme there are redundant elements. For higher number of nodes, superspeedups appear due to the fit of data in cache memory.

Figure 5 and 6 show the speedups curves for the PCG solver and the total speedup respectively. First, one can note the solver speedups behaviour governs the implicitly restarted Lanczos method and the total speedups because it is the most time consuming stage. It should be mentioned that the number of iterations of the solver and the implicitly restarted Lanczos method were independent of the number of processes. In this example, non-overlapping partitions were more efficient than overlapping partitions. Table (2) shows the highest speedups obtained in each phase.

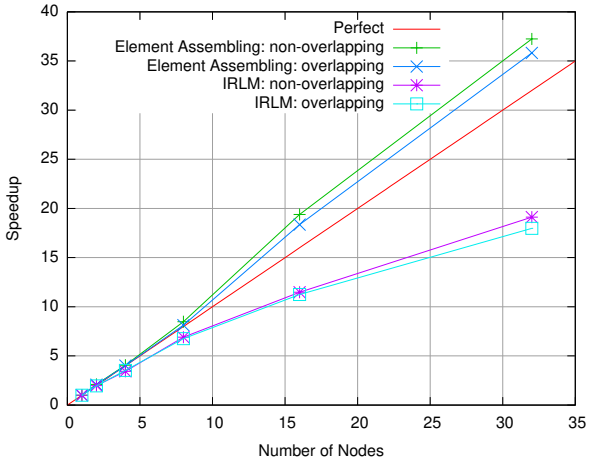


FIGURE 4: Speedups curves for elements calculations and IRLM

TABLE 2: *Maximum speedups* obtained

Phase	*Speedup*	Partitioning Scheme (number of processes)
Global matrix assembling	37.41	Non-overlapping (32 nodes)
PCG	19.13	Non-overlapping (32 nodes)
Impl. restarted Lanczos.	19.12	Non-overlapping (32 nodes)
Total	19.13	Non-overlapping (32 nodes)

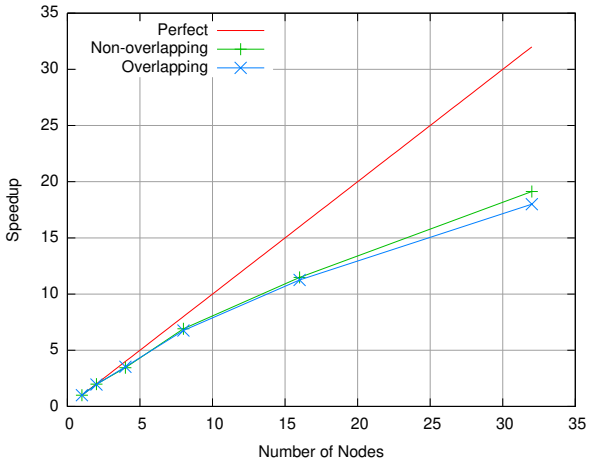


FIGURE 5: Speedups for the PCG method

5 CONCLUSIONS

In this paper we presented a parallel implementation of the implicitly restarted Lanczos method for the solution of generalized eigenproblems discretized by the finite element method. The proposed parallel implementation is based in a subdomain-by-subdomain approach using compressed data structures to store the global coefficients. As expected, the results have shown that convergence of the implicitly restarted Lanczos method is independent of the number of processes.The present implementation of the IRLM includes a shift invert spectral transformation suitable to enforce the convergence of the numerical eigenvalues/eigenvectors to the desired spectrum as shown in the simulated example. The implicitly restarting technique maintained the number of steps in a modest size avoiding problems with memory requirements and

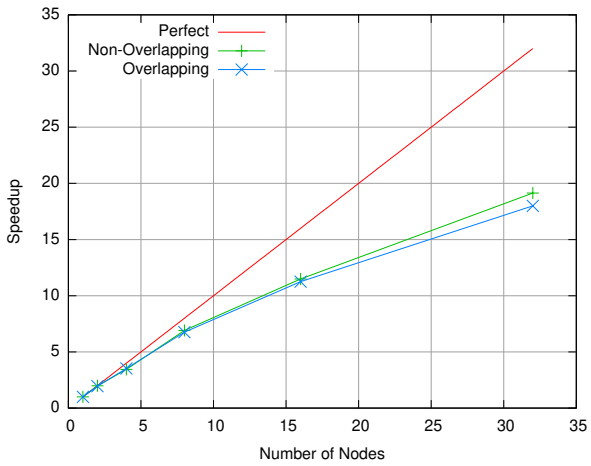


FIGURE 6: Total speedups

orthogonality issues.

The adopted communication scheme, based in the MPI standard, is needed only during the solution phase. In matvec operation, point-to-point communication procedure involving nearest neighbor processes are performed. The examples have shown that non-overlapping and overlapping partitioning schemes can be implemented with equivalent performance and in the simulated examples, non-overlapping partitions had a slight better performance. Compressed data structures in the formats CSRC and CSRC/CSR are very efficient in terms of memory usage allowing the solution of large sparse eigensystems in high performance computing. From the timing results, it might be concluded that the present formulation is suitable for the solution of large symmetric generalized eigensystems.

References

[1] C. Lanczos, “An iteration method for the solution of the eigenvalue problem of linear differential and integral operators,” *Journal of Research, Nat Bureau of Standards*, vol. 45, pp. 255–282, 1950.

[2] W. E. Arnoldi, “The principle of minimized iterations in the solution of the matrix eigenvalue problem,” *Quart. appl. Math*, vol. 9, pp. 17–25, 1951.

[3] B. Nour-Omid, B. N. Parlett, T. Ericsson, and P. S. Jensen, “How to implement the spectral transformation,” *Mathematics of Computation*, vol. 48, no. 178, pp. 663–673, 1987.

[4] P. Arbenz, U. Hetmaniuk, R. B. Lehoucq, and R. Tuminaro, “A comparison of eigensolvers for large-scale 3D modal analysis using AMG-preconditioned iterative solvers,” *International Journal for Numerical Methods in Engineering*, vol. 1, pp. 1–21, 2005.

[5] J. C. Tremblay and T. C. C. Jr., “A refined unsymmetric lanczos eigensolver for computing accurate eigentriplets of a real unsymetric matrix,” *Eletronic Transaction on Numerical Analysis*, vol. 28, pp. 95–113, 2007.

[6] J. Daniel, L. K. W. B. Gragg, and G. W. Stewart, “Reorthogonalization and stable algorithms for updating the Gram-Schmidt QR factorization.” *Mathematics of Computation*, vol. 30, pp. 772–795, 1976.

[7] Y. Saad, “Chebyshev acceleration techniques for solving nonsymmetric eigenvalue problems,” *Mathematics of Computation*, vol. 42, pp. 567–588, 1984.

[8] D. C. Sorensen, “Implicit application of polynomial filters in a k-step Arnoldi method,” *SIAM Journal on Matrix Analysis and Applications*, vol. 13, no. 1, pp. 357–385, 1992.

[9] R. B. Lehoucq, “Analysis and implementation of an implicitly restarted Arnoldi iteration,” Ph.D. dissertation, Rice University, Houston, TX, 1995.

[10] K. Meerbergen and A. Spence, “Implicitly restarted Arnoldi and purification for the shift-invert transformation.” *Mathematics of Computation*, vol. 66, 1997.

[11] G. H. Golub and C. F. V. Loan, *Matrix Computations*, 3rd ed. Baltimore ML: The Jonh Hopkins University Press, 1996.

[12] D. C. Sorensen, “Implicitly restarted arnoldi/lanczos methods for large scale eigenvalue calculations,” Institute for Computer Applications in Science and Engineering (ICASE), Tech. Rep., 1996.

[13] F. L. B. Ribeiro and I. A. Ferreira, “Parallel implementation of the finite element method using compressed data structures.” *Computational Mechanics*, vol. 1, pp. 187–204, December 2007.

[14] I. A. Ferreira, “Solução em paralelo de um modelo termo-químico-mecânico para concreto jovem,” Ph.D. dissertation, Universidade Federal do Rio de Janeiro - COPPE, 2008.

[15] Y. Saad, *Numerical Methods for Large Eigenvalue Problems*, 1st ed. New York, NY: John Wiley & Sons Inc., 1992.

Acknowledgements

The authors are in debt for the financial of the Brazilian research agency CNPq and Coppetec Foundation.