Available online at www.sciencedirect.com

SCIENCE DIRECT®

Future Generation Computer Systems 22 (2006) 279–290

FGCS
FUTURE GENERATION COMPUTER SYSTEMS

www.elsevier.com/locate/fgcs

An asynchronous algorithm on the NetSolve global computing system

Nahid Emad[a], S.-A. Shahzadeh-Fazeli[a], Jack Dongarra[b,*]

[a] *Prism laboratory, University of Versailles, 45, av. des États-Unis, 78035 Versailles Cedex, France*
[b] *Computer Science Department, University of Tennessee, Innovative Computing Laboratory, 1122 Volunteer Blvd., Suite 203, Knoxville, TN 37996-3450, USA*

Received 7 September 2004; received in revised form 21 October 2005; accepted 24 October 2005

Abstract

The explicitly restarted Arnoldi method (ERAM) allows one to find a few eigenpairs of a large sparse matrix. The multiple explicitly restarted Arnoldi method (MERAM) is a technique based upon a multiple projection of ERAM and accelerates its convergence [N. Emamad, S. Petiton, G. Edjlali, Multiple explicitly restarted Arnoldi method for solving large eigenproblems, SIAM J. Sci. Comput. SJSC 27 (1) (2005) 253-277]. MERAM allows one to update the restarting vector of an ERAM by taking into account the interesting eigen-information obtained by its other ERAM processes. This method is particularly well suited to the GRID-type environments. We present an adaptation of the asynchronous version of MERAM for the NetSolve global computing system. We point out some advantages and limitations of this kind of system to implement the asynchronous hybrid algorithms. We give some results of our experiments and show that we can obtain a good acceleration of the convergence compared to ERAM. These results also show the potential of the MERAM-like hybrid methods for the GRID computing environments.
© 2005 Elsevier B.V. All rights reserved.

Keywords: Large eigenproblem; Arnoldi method; Explicit restarting; Global computing; NetSolve

1. Introduction

The hybrid methods were proposed to accelerate the convergence and/or to improve the accuracy of the solution of some linear algebra problems. These methods combine several different numerical methods or several differently parameterized copies of the same method to solve these problems efficiently [2,7,6,13,14]. The multiple explicitly restarted Arnoldi method proposed in [3] is a hybrid method which allows one to approximate a few eigenpairs of a large sparse non-Hermitian matrix. This technique is based on the projection of the problem on a set of subspaces and thus creates a whole range of differently parameterized ERAM processes. Each ERAM improves its subspace by taking into account all interesting intermediary eigen-information

* Corresponding author. Tel.: +1 865 974 8295; fax: +1 865 974 8296.
E-mail addresses: emad@prism.uvsq.fr (N. Emad), sas@prism.uvsq.fr (S.-A. Shahzadeh-Fazeli), dongarra@cs.utk.edu (J. Dongarra).

0167-739X/$ – see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.future.2005.10.003

obtained by itself as well as by the other ERAM processes. In this method, the communications between ERAM processes can be completely asynchronous.

In this paper we present the application of the asynchronous version of MERAM to the NetSolve global computing system. We show some advantages and limitations of this kind of system to implement the asynchronous hybrid algorithms. We also give an adaptation of the algorithm for NetSolve and show that we can obtain a good acceleration of the convergence with respect to the explicitly restarted Arnoldi method.

Section 2 describes the basic Arnoldi algorithm and explicitly restarted Arnoldi method. Section 3 introduces MERAM and some of its algorithms. We point out the limitations of NetSolve-type systems to implement the asynchronous algorithm of MERAM and present an adaptation of this algorithm for NetSolve in Section 4. This algorithm is evaluated in Section 5 by a set of test matrices coming from various application problems. The concluding remarks in Section 6 will contain our perspectives on the problem.

2. Explicitly restarted Arnoldi method

Let A be a large non-Hermitian matrix of dimension $n \times n$. We consider the problem of finding a few eigenpairs (λ, u) of A:

$$Au = \lambda u \text{ with } \lambda \in \mathbb{C} \text{ and } u \in \mathbb{C}^n. \tag{1}$$

Let $w_1 = v/\|v\|_2$ be an initial guess, m be an integer with $m \ll n$. A Krylov subspace method allows to project the problem (1) onto an m-dimensional subspace $\mathbb{K} = span(w_1, Aw_1, \ldots, A^{m-1}w_1)$. The well-known Arnoldi process is a projection method which generates an orthogonal basis $w_1, \ldots, w_m$ of the Krylov subspace $\mathbb{K}$ by using the Gram-Schmidt orthogonalization process. Let $\mathtt{AR}(input : A, m, v; output : H_m, W_m)$ be such an Arnoldi reduction. The $m \times m$ matrix $H_m = (h_{i,j})$ and the $n \times m$ matrix $W_m = [w_1, \ldots, w_m]$ issued from the AR algorithm and the matrix A satisfy the equation:

$$AW_m = W_m H_m + f_m e_m^H \tag{2}$$

where $f_m = h_{m+1,m} w_{m+1}$ and e_m is the mth vector of the canonical basis of $\mathbb{C}^m$. The s desired Ritz values[1] (with largest/smallest real part or largest/smallest magnitude) $\Lambda_m = (\lambda_1^{(m)}, \ldots, \lambda_s^{(m)})$ and their associate Ritz vectors $U_m = (u_1^{(m)}, \ldots, u_s^{(m)})$ can be computed as follows[2]:

Basic Arnoldi Algorithm. $\mathtt{BAA}(input : A, s, m, v; output : r_s, \Lambda_m, U_m)$.

```
(1) Compute an AR($input : A, m, v; output : H_m, W_m$) step.
(2) Compute the eigenpairs of $H_m$ and select the $s$ desired ones: $(\lambda_i^{(m)}, y_i^{(m)})_{i=1}^{i=s}$.
(3) Compute the $s$ associate Ritz vectors $u_i^{(m)} = W_m y_i^{(m)}$ (for $i = 1, \ldots, s$).
(4) Compute $r_s = (\rho_1, \ldots, \rho_s)^t$ with $\rho_i = \|(A - \lambda_i^{(m)} I)u_i^{(m)}\|_2$.
```

If the accuracy of the computed Ritz elements is not satisfactory the projection can be restarted onto a new $\mathbb{K}$. This new subspace can be defined with the same subspace size and a new initial guess. The technique is called the explicitly restarted Arnoldi method. Starting with an initial vector v, it computes BAA. If the convergence does not occur, then the starting vector is updated (using appropriate methods on the computed Ritz vectors) and a BAA process is restarted until the accuracy of the approximated solution is satisfactory. This update is designed to force the vector into the desired invariant subspace. This goal can be reached by some polynomial restarting strategies proposed in [6,8] and discussed in Section 3.1. An algorithm of the explicitly restarted Arnoldi method is the following:

ERAM algorithm. $\mathtt{ERAM}(input : A, s, m, v, tol; output : r_s, \Lambda_m, U_m)$.

```
(1) Start. Choose a parameter $m$ and an initial vector $v$.
(2) Iterate. Compute a BAA($input : A, s, m, v; output : r_s, \Lambda_m, U_m$) step.
(3) Restart. If $g(r_s) > tol$ then use $\Lambda_m$ and $U_m$ to update the starting vector $v$ and go to 2.
```

[1] A Ritz value/vector of a matrix is an approximated eigenvalue/eigenvector.

[2] We suppose that the eigenvalues and corresponding eigenvectors of H_m are re-indexed so that the first s Ritz pairs are the desired ones.

tol is a tolerance value and the function g defines the stopping criterion of iterations. Some typical examples are: $g(r_s) = \|r_s\|_\infty$ and $g(r_s) = \sum_{j=1}^{s} \alpha_j \rho_j$ where α_j are scalar values.

3. Multiple explicitly restarted Arnoldi method

The multiple explicitly restarted Arnoldi method is a technique based upon an ERAM with multiple projection. This method projects an eigenproblem on a set of subspaces and thus creates a whole range of differently parameterized ERAM processes which cooperate to efficiently compute a solution of this problem. In MERAM the restarting vector of an ERAM is updated by taking into account the interesting eigen-information obtained by the other ones. In other words, the ERAM processes of a MERAM begin with several subspaces spanned by a set of initial vectors and a set of subspace sizes. If the convergence does not occur for any of them, then the new subspaces will be defined with initial vectors updated by taking into account the intermediary solutions computed by all the ERAM processes. Each of these differently sized subspaces is defined with a new initial vector v. To overcome the storage dependent shortcoming of ERAM, a constraint on the subspace size of each ERAM is imposed. More precisely, the size of a projection subspace has to belong to the discrete interval $I_m = [m_{\min}, m_{\max}]$. The bounds $m_{\min}$ and $m_{\max}$ may be chosen in function of the available computation and storage resources and have to fulfill $m_{\min} \le m_{\max} \ll n$. Let $m_1 \le \cdots \le m_\ell$ be a set of ℓ subspace sizes with $m_i \in I_m$ $(1 \le i \le \ell)$, $M = (m_1, \ldots, m_\ell)$ and $V^\ell = [v^1, \ldots, v^\ell]$ be the matrix of ℓ starting vectors. An algorithm of this method to compute s $(s \le m_1)$ desired Ritz elements of A is the following:

MERAM algorithm. MERAM(*input* : A, s, M, V^ℓ, tol; *output* : r_s, Λ_m, U_m)

```
(1) Start. Choose a starting matrix $V^\ell$ and a set of
    subspace sizes $M = (m_1, \ldots, m_\ell)$. Let $it = 0$.
(2) Iterate. For $i = 1, \ldots, \ell$ do: $it = it + 1$.
    (a) Compute a BAA(input : $A, s, m_i, v^i$; output :
        $r_s^i, \Lambda_{m_i}, U_{m_i}$) step.
    (b) If $g(r_s^i) \le tol$ then stop all processes.
    (c) If ($it \ge \ell$ and ($it$ mod $\ell$) $\ne 0$) then use the re-
        sults produced by the $\ell$ last BAA processes to
        update $v^{i+1}$.
(3) Restart. Use the results produced by the $\ell$ last
    BAA processes to update $v^1$ and go to 2.
```

where r_s^i is the vector of the residual norms at the ith iteration.

With the hypothesis that $u_j^{(m_p)}$ is "better" than $u_j^{(m_q)}$ if $\rho_j^p \le \rho_j^q$, an interesting updating strategy would be to choose v^i as a function f of "the best" Ritz vectors:

$$v^i = f(U^{\text{best}}), \tag{3}$$

where $U^{\text{best}} = (u_1^{\text{best}}, \ldots, u_s^{\text{best}})$ and u_j^{best} is "the best" jth Ritz vector. The definition of the function f can be based on the techniques proposed by Saad in [6] and will be discussed in Section 3.1.

The problem of the above algorithm is that there is no parallelism between the BAA processes. This is because of the existence of the synchronization points 2.(c) and 3 in the algorithm. In the following algorithm, proposed in [3], these synchronization points are removed. That means each ERAM process, after its BAA step, sends its results to all other processes. Let Send_Eigen_Info represents the task of sending results from an ERAM process to all other ERAM processes, Receiv_Eigen_Info be the task of receiving results from one or more ERAM processes by the current ERAM process and finally, Rcv_Eigen_Info be a boolean variable that is true if the current ERAM process has received results from the other ERAM processes. A parallel asynchronous version of MERAM is the following:

Asynchronous MERAM algorithm.

```
(1) Start. Choose a starting matrix $V^\ell$ and a set of
    subspace sizes $M = (m_1, \ldots, m_\ell)$.
(2) Iterate. For $i = 1, \ldots, \ell$ do in parallel (ERAM
    process):
    • Computation process
        (a) Compute a BAA(input : $A, s, m_i, v^i$;
            output : $r_s, \Lambda_{m_i}, U_{m_i}$) step.
        (b) If $g(r_s^i) \le tol$ stop all processes.
        (c) Update the initial guess (if (Rcv_Eigen_
            Info) then use the hybrid restart strategy,
            otherwise use the simple restart strategy).
```

- Communication process
  (d) `Send_Eigen_Info`
  (e) `Receiv_Eigen_Info`

The ℓ ERAM processes defined in step 2 of the above algorithm are all independents and can be run in parallel. Each of them is constituted by a computation part and a communication part. The computation and the communication can be overlapped inside of an ERAM process. The updating of the initial vector v^i can be done by taking into account the most recent results of the ERAM processes. We recall that, in the above MERAM algorithm, the ℓ last results are necessarily the results of the ℓ ERAM processes.

The above algorithm is fault tolerant. A loss of an ERAM process during MERAM execution does not interfere with its termination. It has a great potential for dynamic load balancing; the attribution of the ERAM processes of MERAM to the available resources can be done as a function of their subspace sizes at run time. The heterogeneity of computing supports can be then an optimization factor for this method [3]. Because of all these properties, this algorithm is well suited to the GRID-type environments. In such environments, the ℓ ERAM processes constituting a MERAM can be dedicated to ℓ different servers. Suppose that the ith ERAM process is dedicated to the server S_i. This server keeps the execution control of the ith ERAM process until the convergence which occurs, in general, by the fastest server. The Fig. 1 shows an execution scheme of the asynchronous MERAM with $\ell = 3$ on three servers. We notice that the computation and communication parts are overlapped.

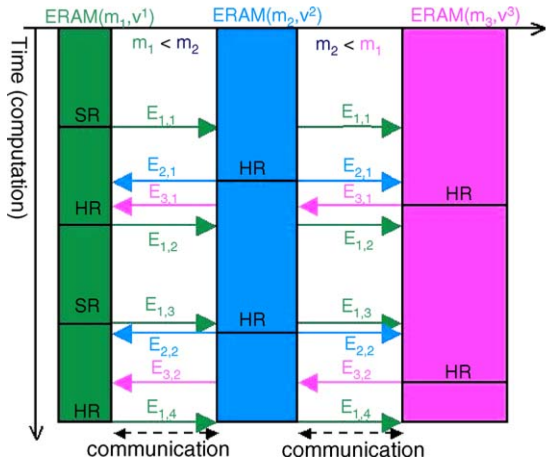

Fig. 1. Asynchronous MERAM with $\ell = 3$.

3.1. Restarting strategies

Saad [7] proposed to restart an iteration of ERAM with a vector preconditioning so that it has to be forced into the desired invariant subspace. It concerns a polynomial preconditioning applied to the starting vector of ERAM. This preconditioning aims at computing the restarting vector so that its components are nonzero in the desired invariant subspace and zero in the unwanted invariant subspace:

$$v(k) = p(A)v \tag{4}$$

where $v(k)$ is kth restarting vector of ERAM and p is a polynomial in the space of polynomials of degree $<m$. One appropriate possibility to define p is a Chebyshev polynomial determined from some knowledge on the distribution of the eigenvalues of A. This restarting strategy is very efficient to accelerate the convergence of ERAM and is discussed in detail in [7,6]. Another possibility to define the polynomial p is to compute the restarting vector with a linear combination of s desired Ritz vectors:

$$v(k) = \sum_{i=1}^{s} \alpha_i u_i^{(m)}(k) \tag{5}$$

where $u_i^{(m)}(k)$ denotes ith Ritz vector computed at the iteration k. There are several ways to choose the scalar values α_i in (5). One choice can be α_i equal to the ith residual norm. Some other choices can be $\alpha_i = 1$, $\alpha_i = i$ or $\alpha_i = s - i + 1$ for $1 \leq i \leq s$ (see [8] for more details). We propose to make use of the following linear combination of the s wanted eigenvectors:

$$v = \sum_{k=1}^{s} l_k(\lambda) u_k{}^{(m)} \tag{6}$$

where s coefficients $l_k(\lambda)$ are defined by:

$$l_k(\lambda) = \prod_{j=1;\ j\neq k}^{s} \left(\frac{\lambda - \lambda_j{}^{(m)}}{\lambda_k{}^{(m)} - \lambda_j{}^{(m)}} \right), \text{ with } \lambda$$
$$= \frac{\lambda_{\min} + \bar{\lambda} - (\lambda_{\min}/n)}{2}, \bar{\lambda} = \frac{\sum_{k=1}^{s} \lambda_k{}^{(m)}}{s}$$

and $\lambda_{\min}$ is the eigenvalue with the smallest residual norm. In the experiments of the next section, we made use of this strategy (i.e., Eq. (6)) to update the initial vector of the ERAM as well as the ones of the ERAM processes of MERAM. For MERAM this equation becomes

$$v^i = \sum_{k=1}^{s} l_k^{(\text{best})}(\lambda) u_k^{(\text{best})} \qquad (7)$$

where $u_k^{(\text{best})}$ is "the best" kth eigenvector computed by the ERAM processes of MERAM and $l_k^{(\text{best})}$ is its associate coefficient.

4. Asynchronous MERAM on a global computing system

4.1. NetSolve global computing system

The NetSolve system is a grid middleware based on the concepts of `Remote Procedure Call (RPC)` that allows users to access both hardware and software computational resources distributed across a network. NetSolve provides an environment that monitors and manages computational resources and allocates the services they provide to NetSolve enabled client programs. NetSolve uses a load-balancing strategy to improve the use of the computational resources available [5]. Three chief components of NetSolve are clients, agents and servers. The semantics of a NetSolve client request are:

(1) Client contacts the agent for a list of capable servers.
(2) Client contacts server and sends input parameters.
(3) Server runs appropriate service.
(4) Server returns output parameters or error status to client.

There are many advantages to using a system like NetSolve which can provide access to otherwise unavailable software/hardware. In cases where the software is in hand, it can make the power of supercomputers accessible from low-end machines like laptop computers. Furthermore, NetSolve adds heuristics that attempt to find the most expeditious route to a problem's solution set. NetSolve currently supports C, FORTRAN, MATLAB, and Mathematica as languages of implementation for client programs. To solve a problem using NetSolve, a problem description file (PDF) corresponding to the problem has to be defined [15,10–12].

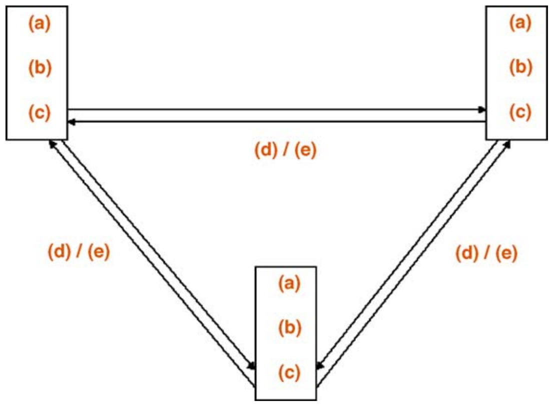

Fig. 2. Asynchronous MERAM on three communicating servers (with $\ell = 3$).

4.2. Asynchronous MERAM on NetSolve system

The servers of the NetSolve system cannot communicate directly to each other. Consequently, contrarily to the MERAM running schemes presented in Figs. 1 and 2, a server cannot keep the control of an ERAM process until the convergence. Fig. 2 shows the asynchronous MERAM algorithm on three servers which communicate directly. Each server runs the steps 2.(a), 2.(b) and 2.(c) of an ERAM and communicates with the other servers by running the steps 2.(d) and 2.(e) of the algorithm. While Fig. 3 shows the asynchronous MERAM algorithm on three servers of a system such as NetSolve where they cannot communicate directly.

Indeed, to adapt the asynchronous MERAM algorithm to NetSolve system a control process centralizing the information and corresponding to a client component of NetSolve has to be defined. This process has to request to the computation servers of the system to run the step 2.(a) of the ERAM processes of MERAM in RPC mode. The running of the step 2.(a) of an ERAM occurs asynchronously with respect to the execution of the same step of the other ERAMs as well as with the execution of the rest of the client algorithm. Once the control process receives the results of a BAA step, it tests the convergence by running the step 2.(b) of

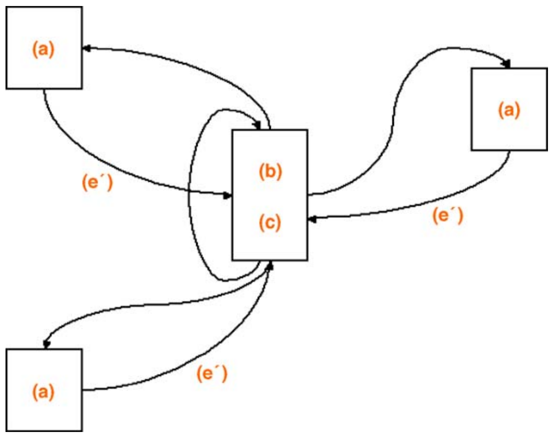

Fig. 3. Asynchronous MERAM on three non-communicating servers (with $\ell = 3$).

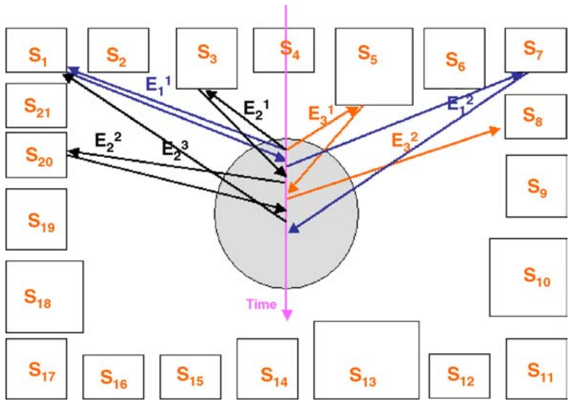

Fig. 4. MERAM-NS on NetSolve (Ex. of three ERAMs processes and 21 servers). S_k is the kth server of the NetSolve system and $E_i{}^j$ is the jth restart of the ith ERAM process of MERAM.

the algorithm. If the convergence is not reached then it updates the initial guess with the available eigen-information on this control/client server. An adaptation of the asynchronous multiple explicitly restarted Arnoldi method for NetSolve is the following:

```
MERAM-NS($input : A, s, M, V^\ell, tol; output : r_s, \Lambda_m, U_m$)

(1) Start. Choose a starting matrix $V^\ell$ and a set of subspace sizes $M = (m_1, \dots, m_\ell)$.
        Let $it_i = 0$ (for $i = 1, \ell$).
(2) For $i = 1, \dots, \ell$ do :
      (a) Compute a BAA($input : A, s, m_i, v^i; output : r_s, \Lambda_{m_i}, U_{m_i}$) step in RPC mode.
(3) Iterate. For $i = 1, \dots, \ell$ do :
    • If (ready_results) then $it_i = it_i + 1$
        (e') Receive results.
        (b) If $g(r_s^i) \le tol$ stop all processes.
        (c) Update the initial guess in function of the available eigen-information.
        (a) Compute a BAA($input : A, s, m_i, v^i; output : r_s, \Lambda_{m_i}, U_{m_i}$) step in RPC mode.
    • End if
(4) End. $it = \max(it_1, \dots, it_\ell)$
```

`ready_results` is a boolean variable which is true if the outputs of the current BAA algorithm are ready. In other words, if the server computing the ith BAA in RPC mode is ready to send its outputs. We notice that in this implementation the step 2.(d) of the asynchronous MERAM algorithm is not necessary and the step 2.(e) is replaced by 3.(e$'$) which consists to receive *all* eigen-information on the control process. Instead, we notice that in each computation request in RPC mode, the client program has to send all inputs to the computation server which accepts this task. That means, in the MERAM-NS algorithm, for each restart (i.e., iteration) of every ERAM process, the client program has to send the n-order matrix A, and an n-size initial vector to a computation server. This engenders an intense communication between the client and computation servers. But this communication is overlapped by the running of the rest of the algorithm. We can notice that when a computational server finishes the step 2.(a) or 3.(a), it has to return $s + 2$ n-size output vectors to the client process. Fig. 4 presents the implementation

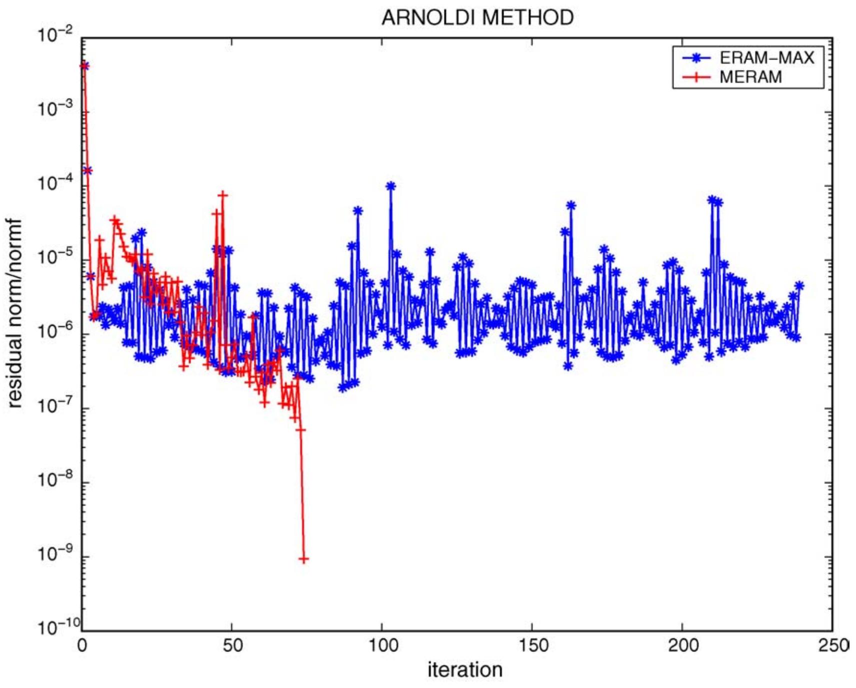

Fig. 5. MERAM(5,7,10) vs. ERAM(10) with $af23560.mtx$ matrix. MERAM converges in 74 restarts, ERAM does not converge after 240 restarts.

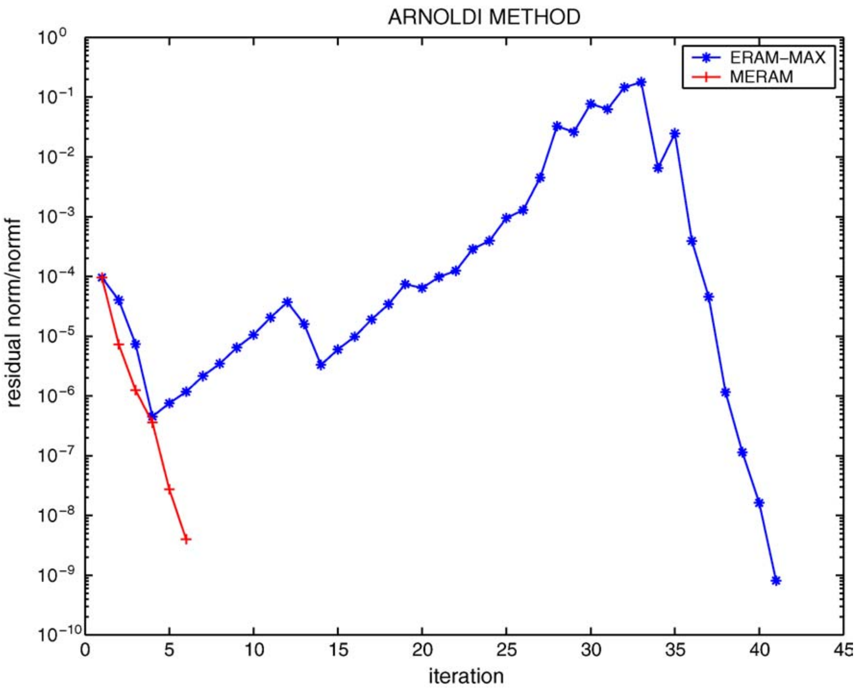

Fig. 6. MERAM(5,7,10) vs. ERAM(10) with $mhd4800b.mtx$ matrix. MERAM converges in six restarts, ERAM converges in 41 restarts.

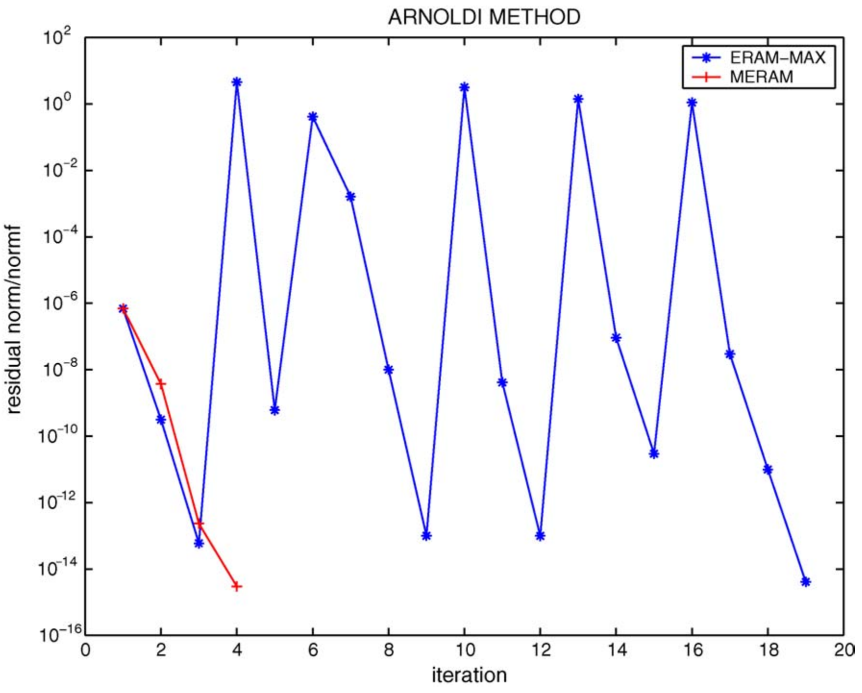

Fig. 7. MERAM(10,15,20) vs. ERAM(20) with $mhd4800b.mtx$ matrix. MERAM converges in four restarts, ERAM converges in 19 restarts.

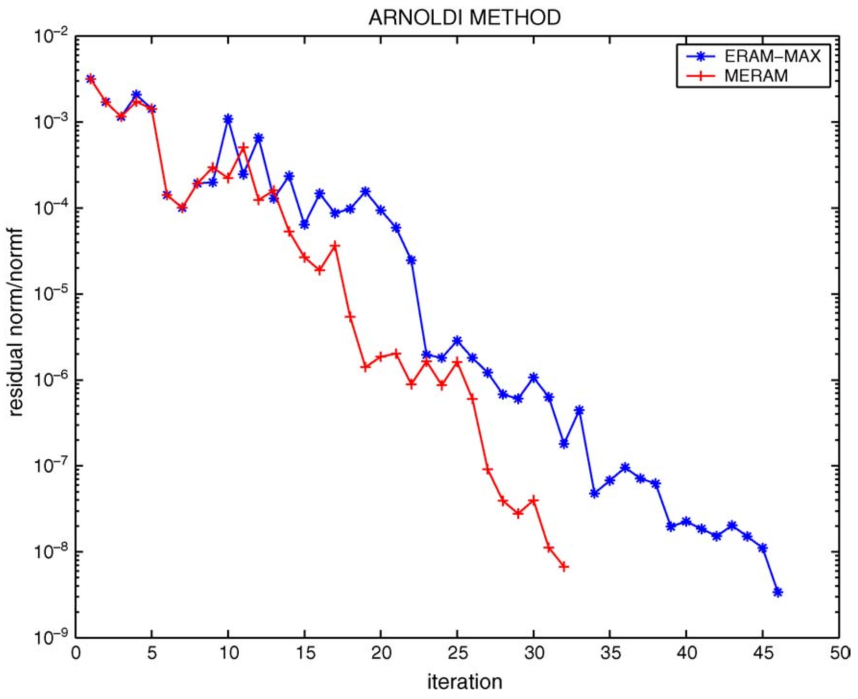

Fig. 8. MERAM(5,10,30) vs. ERAM(30) with $gre\_1107.mtx$ matrix. MERAM converges in 32 restarts, ERAM converges in 46 restarts.

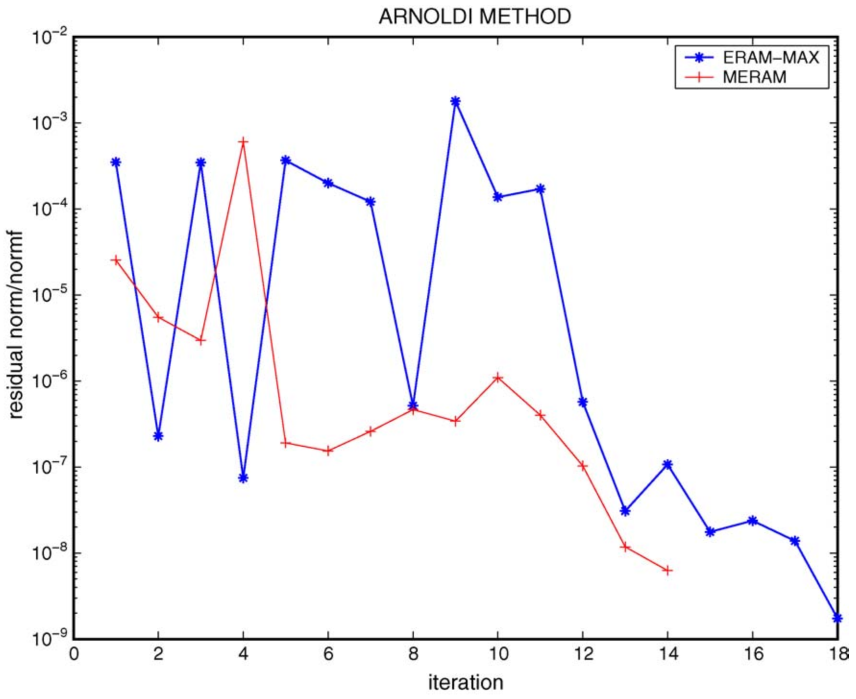

Fig. 9. MERAM(5,7,10) vs. ERAM(10) with $west2021.mtx$ matrix. MERAM converges in 14 restarts, ERAM converges in 18 restarts.

of the MERAM-NS algorithm on a NetSolve system with 21 servers[3] and $\ell = 3$.

In the asynchronous MERAM algorithm, at the end of an iteration each ERAM sends $s + 2$ n-size vectors to $\ell - 1$ other processes. That means, each ERAM has to communicate $(\ell - 1) \times (s + 2) \times n$ data to other processes. The reception of $s + 2$ n-size vectors by a process is not determinism and not quantifiable.

5. Numerical experiments

The experiments presented in this section have been done on a NetSolve system whose computation servers have been located in France (at the University of Versailles and the Institute of Technology of Vélizy sites) and in the USA and interconnected by the internet. We implemented ERAM and MERAM (i.e., MERAM-NS) algorithms using C and MATLAB for some real matrices on the NetSolve system. The client applications are written in MATLAB while the programs having to run in RPC mode (i.e., ERAM processes) are written in C. The stopping criterion is $g(r_s^i) = \|r_s^i\|_\infty$ where $r_s^i = (\rho_1^i, \ldots, \rho_s^i)$ and ρ_j^i is normalized by $\rho_j^i = \rho_j^i / \|A\|_F$ for all $j \in [1, \ldots, s]$ and $i \in [1, \ldots, \ell]$. The tolerance value tol is 10^{-8} in the Figs. 5, 6, 8 and 10^{-14} in the Fig. 7. For all figures the initial vector is $v = z_n = (1, \ldots, 1)/\sqrt{n}$ and the initial matrix is $V^\ell = [v^1 = z_n, \ldots, v^\ell = z_n]$. We search a number $s = 2$ or 5 of the eigenvalues with the largest magnitude. The used matrices are taken from the matrix market [1] and presented in the Table 1. The number of nonzero elements of a matrix is denoted by NNZ. In our experiments, we run MERAM-NS with $\ell = 3$

Table 1
The matrix market used matrices

Matrix	Matrix size	NNZ
$af23560.mtx$	23560	484256
$mhd4800b.mtx$	4800	16160
$gre\_1107.mtx$	1107	5664
$west2021.mtx$	2021	7353

[3] This is an image of the system at a given instant. Indeed, since the servers are volatile, the number of the servers in NetSolve system can change at any moment.

Table 2
ERAM results on NetSolve

Matrix	m	s	v	Iteration	Residual norms	Figures
$af23560.mtx$	10	2	z_n	240	No converge	Fig. 5
$mhd4800b.mtx$	10	2	z_n	41	8.127003e−10	Fig. 6
$mhd4800b.mtx$	20	5	z_n	19	4.089292e−15	Fig. 7
$gre\_1107.mtx$	30	2	z_n	46	3.389087e−09	Fig. 8
$west2021.mtx$	10	2	z_n	18	1.742610e−09	Fig. 9

Table 3
MERAM results on NetSolve

Matrix	m_1, m_2, m_3	s	v^1, v^2, v^3	Iteration	Residual norms	Figures
$af23560.mtx$	5, 7, 10	2	z_n, z_n, z_n	74	9.329017e−10	Fig. 5
$mhd4800b.mtx$	5, 7, 10	2	z_n, z_n, z_n	6	4.016027e−09	Fig. 6
$mhd4800b.mtx$	10, 15, 20	5	z_n, z_n, z_n	4	2.999647e−15	Fig. 7
$gre\_1107.mtx$	5, 10, 30	2	z_n, z_n, z_n	32	6.753314e−09	Fig. 8
$west2021.mtx$	5, 7, 10	2	z_n, z_n, z_n	14	6.267924e−09	Fig. 9

Table 4
Comparison of ERAM(m) and ERAM($m_1, \ldots, m_\ell$) on NetSolve

Matrix	Figures	ℓ	ERAM		MERAM	
			m	Iteration	$m_1, \ldots, m_\ell$	Iteration
af23560.mtx	Fig. 5	3	10	*	5, 7, 10	74
mhd4800b.mtx	Fig. 6	3	20	19	10, 15, 20	4
mhd4800b.mtx	Fig. 7	3	10	41	5, 7, 10	6
gre_1107.mtx	Fig. 8	3	30	46	5, 10, 30	32
west2021.mtx	Fig. 9	3	10	18	5, 7, 10	14

ERAM processes where the steps 2 and 3.(a) are computed in RPC nonblocking mode. The efficiency of our algorithms on NetSolve are measured in terms of the number *it* of the restarts. The number of iterations of MERAM in all of the figures is the number of iterations of the ERAM process which reaches convergence. It is generally the ERAM process with the largest subspace size.

5.1. MERAM-NS versus ERAM

In the following figures, we denote by MERAM ($m_1, \ldots, m_l$) a MERAM with subspaces sizes $m_1, \ldots, m_l$ and by ERAM(m) an ERAM with subspace size m. The Tables 2 and 3 present the results obtained with ERAM and MERAM algorithms on NetSolve and Table 4 presents a comparison between the results obtained by ERAM and MERAM in terms of the number of restarts. We show graphically in Figs. 5–9 the residual norm as a function of iteration number to reach convergence using ERAM and MERAM on NetSolve. The results of our experiments presented in the Tables 2–4 and in the Figs. 5–9 indicate that our MERAM-NS algorithm has better performance than ERAM. We notice from these tables that in terms of the number of the restarts MERAM is considerably more efficient than ERAM.

6. Conclusion

The standard restarted Arnoldi algorithm and its variants may not be efficient for computing a few selected eigenpairs of large sparse non-Hermitian matrices. In order to improve the overall performance of the Arnoldi type algorithm, we propose an adaptation of the multiple explicitly restarted Arnoldi method for NetSolve system. We have seen that this method accelerates the convergence of explicitly restarted Arnoldi method. The numerical experiments have demonstrated that this variant of MERAM is often much more efficient than ERAM. In addition, this concept may be used in some Krylov subspace type methods for the solution of large sparse non symmetric eigenproblems, such as the multiple implicitly restarted method based on IRAM [4,9].

We have shown that the MERAM-type asynchronous algorithms are very well adapted to the global computing systems such as NetSolve. Meanwhile, one of the major problems remains the transfer of the matrix from the client server towards the computation servers. For example, the order of magnitude of the transferred data between client and computation servers is $O((it_1 + \cdots + it_\ell) \times NNZ)$ for MERAM-NS algorithm. Moreover, the classical evaluation of performance is no longer valid in this kind of system. For example, the execution response time cannot be a good measure of performance for MERAM nor for a comparison between MERAM and ERAM. This is for the following reasons:

1. the execution time is dependent on the internet load,
2. the servers are volatile; a server can take part in a portion of calculation and disappear afterwards,
3. the servers are transparent; that means, we do not know the server on which a specific process (such as 2.(a) or 3.(a) steps of `MERAM-NS` algorithm) will be run,
4. the implementation of the ERAM on NetSolve introduces some artificial communications.

One might think that to have a rapid response time it would be better to make use of a classical parallel supercomputer. But the supercomputers are not easily accessible and moreover, the use of a global computing system allows one to take advantage of the otherwise unavailable software and/or hardware resources. In addition, the asynchronous version of the hybrid methods of type MERAM or MIRAM are interesting primarily within the framework of global computing environments.

Acknowledgement

The two first authors were supported in part by French ACI-GRID ANCG project.

References

[1] Z. Bai, D. Day, J. Demmel, J.J. Dongarra, A Test Matrix Collection for Non-Hermitian Eigenvalue Problems, http://math.nist.gov/MatrixMarket, October 1996.
[2] C. Brézinski, M. Redivo-Zaglia, A hybrid procedure for solving linear systems, Numer. Math. 67 (1994) 1–19.
[3] N. Emad, S. Petiton, G. Edjlali, Multiple explicitly restarted Arnoldi method for solving large eigenproblems, SIAM J. Sci. Comput. SJSC 27 (1) (2005) 253–277.
[4] R.B. Lehoucq, D.C. Sorensen, C. Yang, ARPACK User's Guide. Solution of Large-Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods, Software Environ. Tools, SIAM, Philadelphia, 1998.
[5] J.S. Plank, H. Casanova, M. Beck, J.J. Dongarra, Deploying fault tolerance and task migration with NetSolve, Future Gen. Comput. Syst. 15 (5–6) (1999) 745–755.
[6] Y. Saad, Numerical Methods for Large Eigenvalue Problems, Manchester University Press, Manchester, UK, 1992.
[7] Y. Saad, Chebyshev acceleration techniques for solving nonsymmetric eigenvalue problems, Math. Comp. 42 (1994) 567–588.
[8] Y. Saad, Variations on Arnoldi's method for computing eigenelements of large unsymmetric matrices, Linear Algebra Appl. 34 (1980) 269–295.
[9] D.C. Sorensen, Implicitly restarted Arnoldi/Lanczos methods for large scale eigenvalue calculations, in: D.E. Keyes, A. Sameh, V. Venkatakrishnan (Eds.), Parallel Numerical Algorithms, Kluwer Academic Publishers, Dordrecht, The Netherlands, 1997, pp. 119–165.
[10] H. Casanova, J. Dongarra, NetSolve: a network server for solving computational science problems, Int. J. Supercomput. Appl. High Perform. Comput. (1997).
[11] H. Casanova, J. Dongarra, NetSolve's network enabled server: examples and applications, IEEE Computat. Sci. Eng. 5 (3) (1998) 57–67.
[12] H. Casanova, J. Dongarra, NetSolve version 1.2: Design and Implementation, UT Department of Computer Science Technical Report, 1998.
[13] G.L.G. Sleijpen, H.A. Van der Vorst, A Jacobi–Davidson iteration method for linear eigenvalue problems, SIAM Rev. 42 (2000) 267–293.
[14] G.L.G. Sleijpen, H.A. Van der Vorst, Jacobi–Davidson methods, in: Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, H. van der Vorst (Eds.), Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide, Software Eviron. Tools 11, SIAM, Philadelphia, 2000, pp. 88–105.
[15] H., User's Guide to NetSolve V1.4, full reference at http://icl.cs.utk.edu/netsolve.

Nahid Emad received the Habitation á Diriger des Recherches (HDR) in computer science from University of Versailles in 2001, the PhD and MS in applied mathematic from Pierre and Marie Currie University (Paris VI) in 1989 and 1983 and BS from University of Arak (Iran) in 1980. She is associate professor and leads the Intensif Numerical Computation group of the Computer Science Department at the university of Versailles.

Seyed Abolfazl Shahzadeh-Fazeli is assistant professor at the university of Yazd in Iran. He received the PhD in computer science from University of Versailles in 2005 and MS from University of Technologies of Ispahan (Iran) in 1991. His research interests include both theoretical and practical aspects of numerical linear algebra algorithms.

Jack Dongarra holds an appointment as university distinguished professor of computer science in the Computer Science Department at the University of Tennessee and holds the title of Distinguished Research Staff in the Computer Science and Mathematics Division at Oak Ridge National Laboratory (ORNL), and an adjunct professor in the Computer Science Department at Rice University. He specializes in numerical algorithms in linear algebra, parallel computing, use of advanced-computer architectures, programming methodology, and tools for parallel computers. His research includes the development, testing and documentation of high quality mathematical software. He has contributed to the design and implementation of the following open source software packages and systems: EISPACK, LINPACK, the BLAS, LAPACK, ScaLAPACK, Netlib, PVM, MPI, NetSolve, Top500, ATLAS, and PAPI. He has published approximately 200 articles, papers, reports and technical memoranda and he is coauthor of several books. He is a Fellow of the AAAS, ACM, and the IEEE and a member of the National Academy of Engineering.