An Asynchronous Algorithm on NetSolve Global Computing System

Nahid Emad[†] S. A. Shahzadeh Fazeli [†] Jack Dongarra [‡]

March 30, 2004

Abstract

The Explicitly Restarted Arnoldi Method (ERAM) allows to find a few eigenpairs of a large sparse matrix. The Multiple Explicitly Restarted Arnoldi Method (MERAM) is a technique based upon a multiple use of ERAM and accelerates its convergence [2]. The MERAM allows to update the restarting vector of an ERAM by taking the interesting eigen-information obtained by the other ones into account. This method is particularly well suited to the GRID-type environments. We present an adaptation of the asynchronous MERAM for NetSolve global computing system. We give some results of our experiments and show that we can obtain a good acceleration of the convergence compared to ERAM.

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. The Multiple Explicitly Restarted Arnoldi Method (MERAM) proposed in [2] is a hybrid method which allows to approximate a few eigenpairs of a large sparse non-Hermitian matrix. It makes use of several differently parameterized ERAM for the benefit of the same application.

In this paper we present the application of the asynchronous MERAM to NetSolve global computing system. We show some limitations of this kind of systems to implement the asynchronous algorithms. We give then an adaptation of the algorithm for NetSolve and show that we can obtain a good acceleration of 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, some of its variants and their characteristics. We point out the limitations of NetSolve-type systems to implement the asynchronous algorithm of MERAM and 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 perspective on the problem.

[†]Laboratoire PR*i*SM, Université Versailles St-Quentin, 45, av. des États-Unis, 78035 Versailles, France (`(emad, sas)@prism.uvsq.fr`)

[‡]Computer Science Department, University of Tennessee, Knoxville (`dongarra@cs.utk.edu`)

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 :

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

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 a m-dimensional subspace $\mathbb{K} = span(w_1, Aw_1, \cdots A^{m-1}w_1)$. The well-known Arnoldi process is a projection method which generates an orthogonal basis $w_1, \cdots, w_m$ of the Krylov subspace $\mathbb{K}$ by using the Gram-Schmidt orthogonalization process. Let $\texttt{AR}(input : A, m, v; output : H_m, W_m)$ be such Arnoldi reduction. The $m \times m$ matrix $H_m = (h_{i,j})$ and the $n \times m$ matrix $W_m = [w_1, \cdots, w_m]$ issued from `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 (with largest/smallest real part or largest/smallest magnitude) $\Lambda_m = (\lambda_1^{(m)}, \cdots, \lambda_s^{(m)})$ and their associate Ritz vectors $U_m = (u_1^{(m)}, \cdots, u_s^{(m)})$ can be computed as follows[1]:

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

1. Compute an $\texttt{AR}(input : A, m, v; output : H_m, W_m)$ step.
2. Compute the eigenpairs of H_m and select the s desired ones.
3. Compute the s associate Ritz vectors $u_i^{(m)} = W_m y_i^{(m)}$.
4. Compute $r_s = (\rho_1, \cdots, \rho_s)^t$ 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 method is called the Explicitly Restarted Arnoldi (ERAM). Starting with an initial vector v, it computes `BAA`. If the convergence does't occur, then the starting vector is updated and a `BAA` process is restarted until the accuracy of the approximated solution is satisfactory (using appropriate methods on the computed Ritz vectors). This update is designed to force the vector in the desired invariant subspace. This goal can be reached by some polynomial restarting strategies proposed in [4] and discussed in section 4.3. An algorithm of Explicitly Restarted Arnoldi Method is the following:

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

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

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

[1] 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.

3 Multiple Explicitly Restarted Arnoldi Method

The Multiple Explicitly Restarted Arnoldi Method is a technique based upon a multiple use of ERAM to accelerate its convergence. In this method several differently parameterized ERAM co-operate to efficiently compute a solution of a given eigen-problem. The MERAM allows to update the restarting vector of an ERAM by taking the interesting eigen-information obtained by the other ones into account. The ERAMs begin with several subspaces spanned by a set of initial vectors and a set of subspace sizes. If the convergence does't occur for any of them, then the new subspaces will be defined with initial vectors updated by taking the solutions computed by all the ERAM processes into account. These differently sized subspaces are defined with a parameter m and a new initial vector v. To overcome the storage dependent shortcoming of ERAM, a constraint on the subspace size m is imposed. More precisely, m 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 $m_{min} \leq m_{max} \ll n$. Let $m_1 \leq \cdots \leq m_\ell$ with $m_i \in I_m$ $(1 \leq i \leq \ell)$, $M = (m_1, \cdots, m_\ell)$ and $V^\ell = [v^1, \cdots, v^\ell]$ be the matrix of ℓ starting vectors. A serial algorithm of this method to compute s $(s \leq m_1)$ desired Ritz elements of A is the following:

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

1. `Start.` Choose a starting matrix V^ℓ and a set of subspace sizes $M = (m_1, \cdots, m_\ell)$, $it = 0$.
2. `Iterate.` For $i = 1, \cdots, \ell$ 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 $g(r_s^i) \leq tol$ then stop.
   - `(c)` If ($it \geq \ell$ and (it mod ℓ) $\neq 0$) then use the results produced by the ℓ last `BAA` processes to update v^{i+1}.
3. `Restart.` Use the results produced by the ℓ 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 $\rho_j^p \leq \rho_j^q$, an interesting updating strategy would be to choose v^i as a function f of "the best" Ritz vectors:

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

where $U^{best} = (u_1^{best}, \cdots, u_s^{best})$ and u_j^{best} is "the best" jth Ritz vector. The function f has to be chosen in order to force the vector v^i into the desired invariant subspace. A simple choice for f can be a linear combination of $u_1^{best}, \cdots, u_s^{best}$ vectors. The definition of f can be based onto the techniques proposed by Y. Saad in [4] and will be discussed in section 4.3.

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 these synchronization points are removed. That means each ERAM process, after its `BAA` step, broadcasts 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. Let `Receiv_Eigen_Info` be the task of receiving results from one or more ERAM processes by the current ERAM process. Finaly, let `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 explained in the following:

An Asynchronous MERAM Algorithm.

1. `Start.` Choose a starting matrix V^ℓ and a set of subspace sizes $M = (m_1, \cdots, m_\ell)$.
2. `Iterate.` For $i = 1, \cdots, \ell$ do in parallel:
   - Computation process (`ERAM` process)
     - `(a)` Compute a `BAA`$(input : A, s, m_i, v^i; output : r_s, \Lambda_{m_i}, U_{m_i})$ step.
     - `(b)` If $g(r_s^i) \leq tol$ stop.
     - `(c)` If (`Rcv_Eigen_Info`) then update the initial guess.
   - Communication process
     - `(d)` `Send_Eigen_Info`
     - `(e)` `Receiv_Eigen_Info`

The ℓ ERAM processes defined in step 2 of the above 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 the most recent results of the ERAM processes into account. We recall that in the above serial MERAM algorithm the ℓ last results are necessarily the results of the ℓ ERAM processes.

This 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 ERAM processes of MERAM to the available resources can be done as a function of their subspace size at run time. The heterogeneity of computing supports can be then an optimization factor for this method [2]. Because of all these properties, this algorithm is well suited to the GRID-type environments. In a such environment, the ℓ ERAM processes constituting a MERAM can be dedicated to ℓ different servers. Suppose that the i^{th} ERAM process is dedicated to the computation server S_i. This server keeps the execution control of the i^{th} ERAM process until the convergence which occurs by the fastest process. The figure 1 shows an execution scheme of parallel MERAM with $\ell = 3$ on 3 servers.

4 Asynchronous MERAM on NetSolve System

4.1 NetSolve

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. 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.

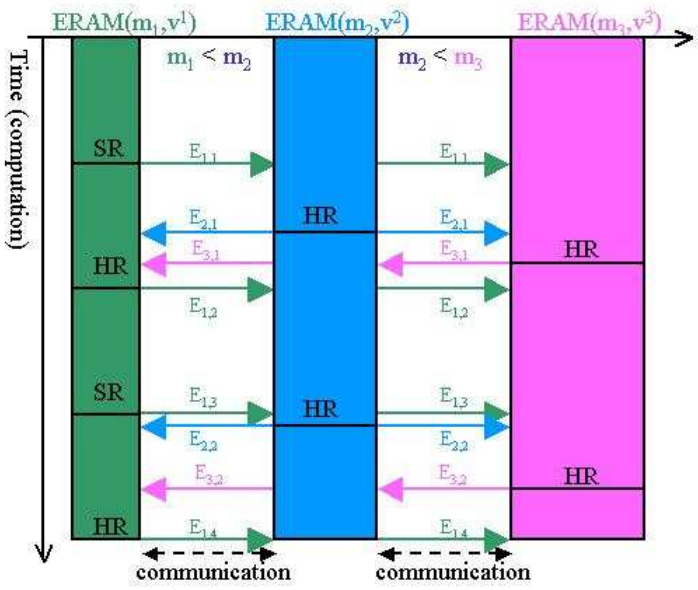

Figure 1: Asynchronous MERAM with $\ell = 3$

There are many advantages to using a system like NetSolve which can provide access to otherwise unavailable software/harware. 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 and the most expeditious route to a problem's solution set. NetSolve currently supports the 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 [8, 9, 10].

4.2 Application of parallel MERAM on NetSolve

The servers of NetSove system can not communicate directly to each other. Consequently, a server can't keep the control of a ERAM process until convergence. The adaptation of parallel MERAM algorithm for NetSolve allows to define a control process corresponding to a client component of NetSolve. This process has to require to system servers the computation of the step(a) of the above algorithm in RPC mode. The reception of the results of the stem (a) occurs asynchronously in respect of... Once this control process receives the results of an BAA step, it runs the steps (b) and (c) of the current process. The updating the restarting vectors will be done synchronously. An adaptation of parallel Multiple Explicitly Restarted Arnoldi Method for NetSolve is the following:

A MERAM Algorithm for NetSolve.

1. `Start.` Choose a starting matrix V^ℓ and a set of subspace sizes $M = (m_1, \cdots, m_\ell)$.

2. `Iterate.` For $i = 1, \cdots, \ell$ do :

   - `(a)` Compute a $\texttt{BAA}(input : A, s, m_i, v^i; output : r_s, \Lambda_{m_i}, U_{m_i})$ step in RPC mod.
   - If `(received_results)` then

     `(b)` If $g(r_s^i) \leq tol$ stop..
     `(c)` If (`Rcv_Eigen_Info`) then update the initial guess.

   `(e) Receiv_Eigen_Info`

Algorithm MERAM on NS: $\texttt{MERAM-NS}(input : A, s, M, V^\ell, tol; output : r_s, \Lambda_m, U_m)$

1. `Start.` Choose a starting matrix V^ℓ and a set of subspace sizes $M = (m_1, \cdots, m_\ell)$.
2. For $i = 1, \cdots, \ell$: send $\texttt{BAA}_i(input : A, s, m_i, v^i; output : r_s^i, \Lambda_{m_i}, U_{m_i})$.
3. `Iterate`. For $i = 1, \cdots, \ell$
   If result ($\texttt{BAA}_i$) is ready then:
4. (a) receive result $\texttt{BAA}_i$
   (b) If $g(r_s^i) \leq tol$ then stop.
   (c) use the results produced by the ℓ last `BAA` processes to update v^i.
   (d) send $\texttt{BAA}_i$ $(input : A, s, m_i, v^i; output : r_s^i, \Lambda_{m_i}, U_{m_i})$.

We notice that in this implementation the step (d) of the parallel MERAM algorithm are not necessary.

Figure 2 presents this implementation of `MERAM3` on a NetSolve system consisting of 3 ERAMs and 21 servers.

Clearly, the MERAM algorithm is equivalent to a particular use of several Explicitly Restarted Arnoldi methods. It allows to update the restarting vector v^i of an ERAM by taking the interesting eigen-information obtained by the other ones into account. Another advantage of this algorithm is its structure with respect to large coarse parallelism. The restarting strategy (**??**) also allows to produce ℓ *different* Krylov subspaces.

4.3 Restarting Strategies

The restarting strategy is a critical part of simple and multiple Explicitly Restarted Arnoldi algorithms. Saad [5] proposed to restart the iteration of ERAM with a vector preconditioning so that it has to be forced in 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 [5, 4]. Another

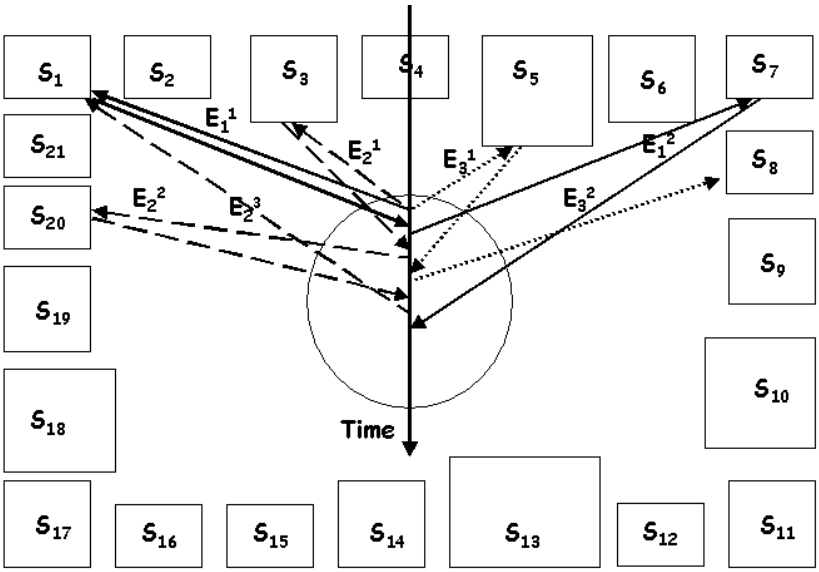

Figure 2: MERAM in NetSolve (Ex. of 3 ERAMs et 21 servers) S_k: k-th server of NetSolve system available on the internet $E_i{}^j$: j-th iteration of $E_i = E(A, m_i, v_i)$

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. This restarting strategy is polynomial because $u_i^{(m)}(k) \in \mathcal{K}_{m,v}$ implies $u_i^{(m)}(k) = \phi_i(A)v$ for some polynomial ϕ_i and then $v(k) = \sum_{i=1}^{s} \alpha_i \phi_i(A)v$. 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 [6] for more details). We presented a new linear combination consists in taking a linear combination between s eigenvalues and eigenvectors, one can calculate the coefficients for a linear combination between the eigenvectors:

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

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

$$l_k(\lambda) = \prod_{\substack{i=1 \\ i \neq k}}^{s} \left(\frac{\lambda - \lambda_i{}^{(m)}}{\lambda_k{}^{(m)} - \lambda_i{}^{(m)}}\right) \qquad k = 1, 2, \cdots, s$$

with: $\lambda = (\lambda_{best} + \bar{\lambda} - \frac{\lambda_{best}}{n})/2$ where $\bar{\lambda} = \frac{\sum_{k=1}^{s} \lambda_k{}^{(m)}}{s}$
et λ_{best} is the eigenvalue with minimum residual norm.

5 Numerical experiments

In this section we describe numerical experiments to analyze the performance of ERAM and MERAM for solving the eigenproblem. We implemented ERAM and MERAM using C and

MATLAB for the real case on NetSolve system. We have used MATLAB programs with stopping criterion $\frac{\|Ax-\lambda x\|}{\|A\|_F} < 10^{-8}$ for the figures 3, 4, 6 and 7, and $\frac{\|Ax-\lambda x\|}{\|A\|_F} < 10^{-14}$ for the figure 5. For all figures the initial vector is $x = z_n = (1, 1, \cdots, 1)$. We used these methods in order to find the 2 and 5 eigenvalues of largest magnitude. The matrices are taken from the matrix market [1].The efficiency of these algorithms can thus be measured in terms of the number of restarts (iterations) *it*. Our matrices are:

Matrix	Size of matrix	nonzero elements
$af23560.mtx$	23560	484256
$mhd4800b.mtx$	4800	16160
$gre\_1107.mtx$	1107	5664
$west2021.mtx$	2021	7353

Table 1: *Matrices*

We run experimentations with 3 ERAMs ($l = 3$)in nonblocking call. The number of iterations of MERAM in all of figures is the number of iterations of the ERAM process which reach convergence. It is generally the ERAM process with the largest subspace size.

5.1 MERAM versus ERAM

In figures 3, $\cdots$, 7 MERAM$(m_1, \cdots, m_l)$ denotes an MERAM with subspaces sizes $(m_1, \cdots, m_l)$, ERAM(m) denotes an ERAM with subspace size m.
Figures 3, $\cdots$, 7 present the number of iteration to reach convergence using MERAM and ERAM on NetSolve system. Experiment on figure 3 is using matrix $af23560.mtx$ and MRERAM$(7, 5, 10)$ and ERAM(10). MERAM is converging in 74 iterations while ERAM is not converging in 240 iterations. Experiment on figure 4 is using matrix $mhd4800b.mtx$ and MRERAM$(7, 5, 10)$ and ERAM(10). MERAM is converging in 6 iterations while ERAM is converging in 41 iterations. Experiment on figure 5 is using matrix $mhd4800b.mtx$ and MRERAM$(10, 15, 20)$ and ERAM(20). MERAM is converging in 4 iterations while ERAM is converging in 19 iterations. Experiment on figure 6 is using matrix $gre\_1107.mtx$ and MRERAM$(30, 10, 5)$ and ERAM(30). MERAM is converging in 32 iterations while ERAM is converging in 46 iterations. Experiment on figure 7 is using matrix $west2021.mtx$ and MRERAM$(7, 5, 10)$ and ERAM(10). MERAM is converging in 14 iterations while ERAM is converging in 18 iterations. Tables 2, 3 and 4 and figures 3, $\cdots$, 7 indicates that our algorithm shows better performance than ERAM. It is seen from tables 2, 3 and 4 that MERAM was considerably more efficient than ERAM, as shown by *it*.

Matrix	m	s	x	it	Res.Norms	Fig
$af23560.mtx$	10	2	z_n	240	No converge	3
$mhd4800b.mtx$	10	2	z_n	41	8.127003e-10	4
$mhd4800b.mtx$	20	5	z_n	19	4.089292e-15	5
$gre\_1107.mtx$	30	2	z_n	46	3.389087e-09	6
$west2021.mtx$	10	2	z_n	18	1.742610e-09	7

Table 2: *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 over-

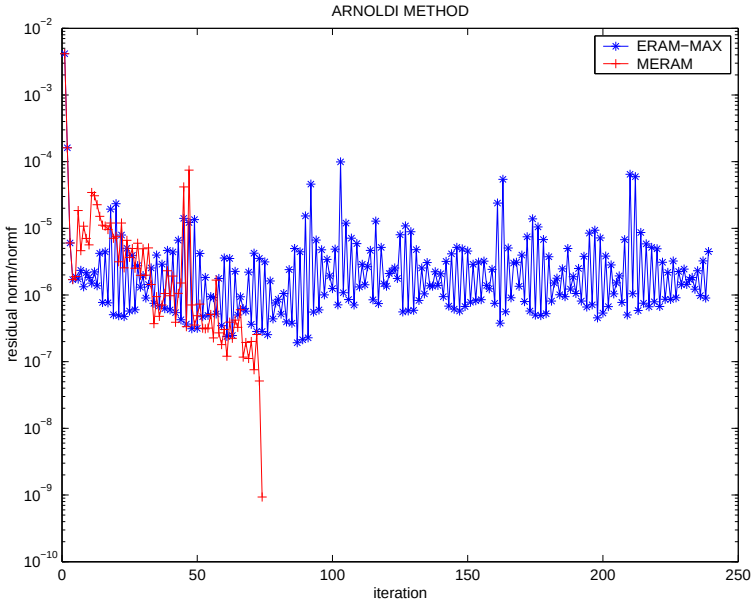

Figure 3: matrix $af23560.mtx$, MERAM$(7, 5, 10)$ conv. in 74 iterations, ERAM(10) no conv. in 240 iterations

Matrix	m_1, m_2, m_3	s	$x_1, x_2, x_3,$	it	Res. Norms	Fig
$af23560.mtx$	7, 5, 10	2	z_n, z_n, z_n	74	9.329017e-10	3
$mhd4800b.mtx$	7, 5, 10	2	z_n, z_n, z_n	6	4.016027e-09	4
$mhd4800b.mtx$	10, 15, 20	5	z_n, z_n, z_n	4	2.999647e-15	5
$gre\_1107.mtx$	30, 15, 10	2	z_n, z_n, z_n	32	6.753314e-09	6
$west2021.mtx$	7, 5, 10	2	z_n, z_n, z_n	14	6.267924e-09	7

Table 3: *MERAM*

all performance of Arnoldi type algorithm, We have proposed a variant of Multiple Explicitly Restartd Arnoldi Method and some of its characteristics restarting strategy on Netsolve system. We have seen that the Multiple Explicitly Restarted Arnoldi Method accelerates the convergence of Explicitly Restarted Arnoldi Method. The numerical experiments have demonstrated that this variant of MERAM are often much more efficient than ERAM. Also, the strategy presented in the paper can be applied to other variants of Arnoldi method this. In addition, this strategy may be used in some Krylov subspace type method for the solution of large sparse unsymmetric eigenproblem.

References

[1] Z.BAI, D. DAY, J. DEMMEL AND J. DONGARA, *A Test Matrix Collection for Non-Hermitian Problems*, http://math.nist.gov/MatrixMarket.

[2] N. EMAD, S. PETITON, AND G. EDJLALI, *Multiple Explicitly Restarted Arnoldi Method for Solving Large Eigenproblems*, SIAM Journal of Scientific Computing, Page 1-20, Submitted in Jan. 2000.

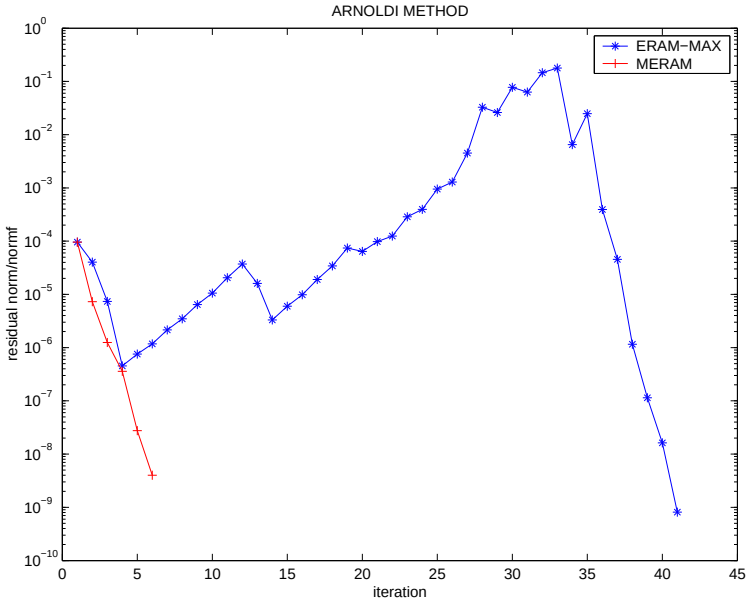

Figure 4: matrix *mhd*4800*b.mtx*, MERAM$(7, 5, 10)$ conv. in 6 iterations, ERAM(10) conv. in 41 iterations

[3] R. Lehoucq, D.C. Sorensen and P.A Vu, *ARPACK: Fortran subroutines for solving large scale eigenvalue problems, Release 2.1*, available form `netlibornl.gov` in the `scalapack` directory, 1994.

[4] Y. Saad, *Numerical Methods for Large Eigenvalue Problems*, Manchester University Press, 1993.

[5] Y. Saad, *Chebyshev acceleration techniques for solving nonsymmetric eigenvalue problems*, Math. Com., 42, 567-588, 1984.

[6] Y. Saad, *Variations on Arnoldi's Method for Computing Eigenelements of Large Unsymmetric Matrices*, Linear Algebra Applications, 34, 269-295, 1980.

[7] D.C. Sorensen, *Implicitly restarted Arnoldi/Lanczos Methods for Large Scale Eigenvalue Calculations*, In D. E. Keyes, A. Sameh, and V. Venkatakrishnan, eds. *Parallel Numerical Algorithms*, pages 119-166, Dordrecht, 1997, Kluwer.

[8] H. Casanova and J. Dongarra , *NetSolve: A Network Server for Solving Computational Science Problems.* The International Journal of Supercomputer Applications and High Performance Computing,1997.

[9] H. Casanova and J. Dongarra , *NetSolve's Network Enabled Server: Examples and Applications*, IEEE Computational Science and Engineering, 57-67, 5(3), 1997, 1998.

[10] H. Casanova and J. Dongarra , *NetSolve version 1.2: Design and Implementation*, UT Department of Computer Science Technical Report,1998.

[11] H., *User's Guide to NetSolve V1.4*, full reference at http://icl.cs.utk.edu/netsolve

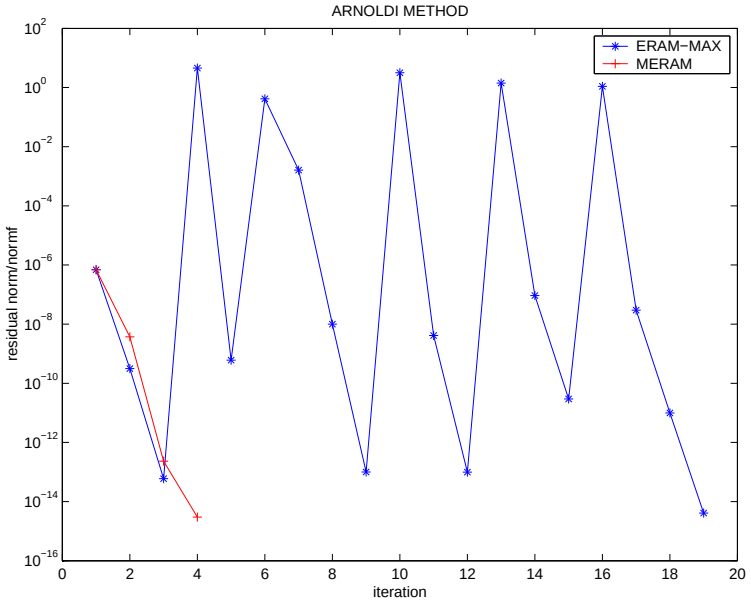

Figure 5: matrix $mhd4800b.mtx$, MERAM$(10, 15, 20)$ conv. in 4 iterations, ERAM(20) conv. in 19 iterations

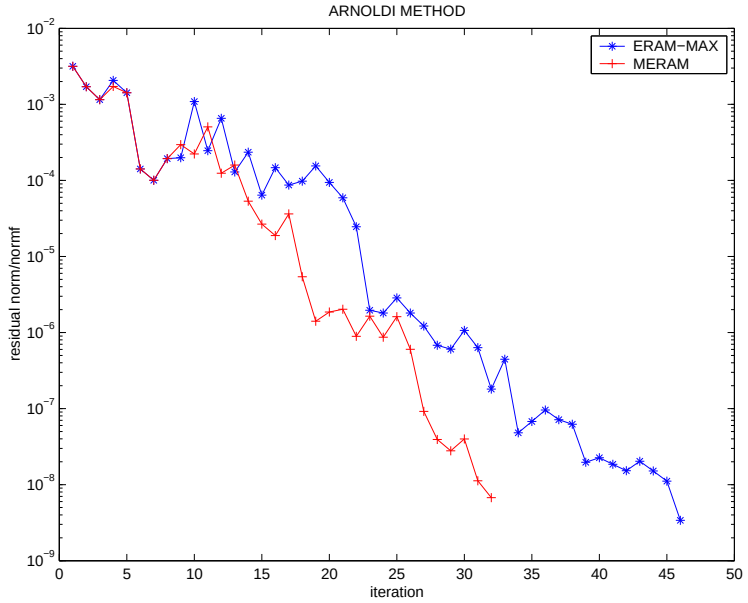

Figure 6: matrix $gre\_1107.mtx$, MERAM$(30, 10, 5)$ conv. in 32 iterations, ERAM(30) conv. in 46 iterations

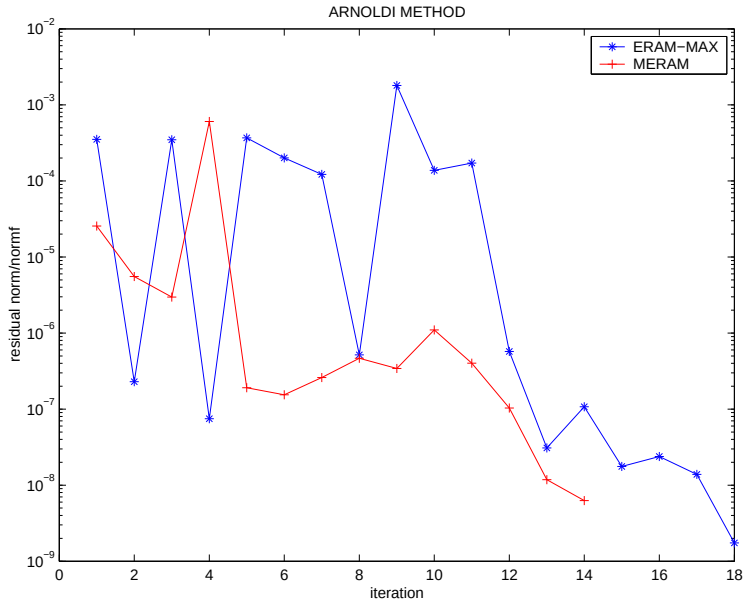

Figure 7: matrix *west*2021.*mtx*, MERAM$(7, 5, 10)$ conv. in 14 iterations, ERAM(10) conv. in 18 iterations

problem	af23560.mtx		mhd4800b.mtx		gre_1107.mtx	
order of matrix	23560		4800		1107	
number of entries	484256		16160		5664	
algorithm	ERAM	MERAM	ERAM	MERAM	ERAM	MERAM
size of subspace	10	7, 5, 10	20	10, 15, 20	30	30, 10, 5
number of restarts	*	74	19	4	46	32

problem	west2021.mtx		mhd4800b.mtx	
order of matrix	2021		4800	
number of entries	7353		16160	
algorithm	ERAM	MERAM	ERAM	MERAM
size of subspace	10	7, 5, 10	10	7, 5, 10
number of restarts	18	14	41	6

Table 4: *ERAM and MERAM*