ENHANCED THE PERFORMANCE OF LANCZOS-TYPE ALGORITHMS BY RESTARTING FROM THE POINT GENERATED BY EIEMLA FOR THE SOLUTION OF SYSTEMS OF LINEAR EQUATIONS

Maharani Maharani[a] , Abdellah Salhi[b] and Wali Khan Mashwani[c]

[a] Department of Mathematics, University of Jenderal Soedirman, Purwokerto, Indonesia,
Email: maharaniyusro@gmail.com

[b] Department of Mathematical Sciences, University of Essex, Colchester, CO4 3SQ, U.K.
Email : as@essex.ac.uk

[c] Department of Mathematics, Kohat University of Science and Technology, Pakistan
Email: mashwanigr8@gmail.com

ABSTRACT: *Lanczos-type algorithms are well known as effective iterative methods for solving non-symmetric of systems of linear equation (SPL). However, they are fragile when involving a large number of iterations, which is well-known as a breakdown phenomenon. This study introduces modelling Lanczos algorithms through interpolation and extrapolation tools, to avoid the use of a large number of iterations and hence avoiding the breakdown. An iterate generated by embedding this model into Lanczos algorithms (and hence called embedding interpolation and extrapolation model in Lanczos-types Algorithms, or EIEMLA), is then used to restart the new algorithm. The whole procedure is named restarting EIEMLA (REIEMLA). This restarting framework aims to accelerate the convergence of Lanczos-type algorithms. Theoretical and numerical results are presented and are compared with other existing restarting strategies in Lanczos algorithms. Empirically, restarting from the iterate generated by the model function performs better than other existing restarting discussed in [8] and [14].*

Keywords: breakdown, EIEMLA; restarting strategies; EIEMLA

INTRODUCTION

Lanczos-type algorithms are well known as an effective iterative methods for solving non-symmetric systems of linear equations (SPL). These algorithms was first proposed by Brezinski and his team, [1,2,3], by using theory of orthogonal polynomials (FOP's). Theoretically, for solving n dimensions of SPL, we need n number of iterations to get a good solution. Practically, however, we often use more than n iterations. This is because the computational errors are accumulated in every iteration of the algorithms. On the other hand, breakdown in Lanczos algorithms is also an unavoidable. Therefore, a number of strategies are already developed to enhance the performance of Lanczos-types algorithms such as the look-ahead strategy, [4,5], which is also called the Method of Recursive Zoom (MRZ), the look-around strategy, [10, 11], typically try to get over and/or around the non-existing orthogonal polynomials. Other strategies such as switching between Lanczos-type algorithms and restarting them have also been considered, [7, 8, 9]. The later mentioned approaches are performing better than MRZ in terms of robustness, [1, 8]. The improvement of existing restarting in Lanczos-type algorithms, has been investigated in [14], by considering three different points for restarting. The recent work [15], dis cussed modelling in Lanczos-type algorithms for the same purpose. The resulting model is called embedding interpolation and extrapolation in Lanczos-types algorithms (EIEMLA). In this study, we suggest restarting from the point generated by the EIEMLA aiming at to find a better point to restart a Lanczos-type algorithm so that we would obtain a better result. This result is then compared with the existing restarting strategies in Lanczos-type algorithms. The rest of this paper is organized as

follows. In Section 2, we look at some background theories related to the derivation of EIEMLA. Restarting of EIEMLA and its implementation are discussed in Section 3. Some numerical results and comparison of this restarting against the existing restarting strategies, i.e. RLLastIt, RLMinRes and RLMedVal are discussed in Section 4. Lastly, we conclude this study in Section 5.

2. Derivation of EIEMLA

We follow [15] to derive the embedding interpolation and extrapolation in Lanczos-types algorithms (EIEMLA). Suppose we run a Lanczos-type algorithm, [1, 3], for k iterations, where $k \leq n$. Note, it is intentionally and pre-emptively stopped before the algorithm breaks down. We then consider the generated iterates which form a sequence $S = \{x_1, x_2, \ldots, x_k\}$. Let xm, be the iterate with the lowest residual norm, $\|\mathbf{r_m}\|$where $m \leq k$. Assume that some good iterates, namely those with small residual norms, concentrate in interval [m − j, k], for some integer j. Set

$$V_1 = \{\mathbf{x}_{m-j}, \mathbf{x}_{m-j+1}, \cdots, \mathbf{x}_k\}, \quad (1)$$

which is a subset of S. Write the components of each iterate in S as

$$v_1 = \left\{x_{m-j}^{(1)}, x_{m-j+1}^{,(1)}, \cdots, x_k^{(1)}\right\}$$
$$v_2 = \left\{x_{m-j}^{(2)}, x_{m-j+1}^{,(2)}, \cdots, x_k^{(2)}\right\} \quad (2)$$
$$\vdots$$

$v_n = \left\{x_{m-j}^{(n)}, x_{m-j+1}^{\prime(n)}, \cdots, x_k^{(n)}\right\}$- namely, each vi contains all of the it h entries of iterates xl , for l = m − j, m − j + 1, · · · , k, and for i = 1, 2, · · · , n. Thus, we find a function which terpolates each set of vi's using PCHIP interpolant, [10,11]. We assume that each sequence of $\mathbf{x}_{m-j}^{(i)}, \mathbf{x}_{m-j+1}^{(i)}, \cdots, \mathbf{x}_k^{(i)}$ is monotonic and

convergent for some j and i = 1, 2, …, n, to its limit, [13], i.e.

$$\lim_{k\to\infty} x_k^{(i)} = x_*^{(i)} . \qquad (3)$$

Let t be elements in R. Set

$$\begin{aligned} w_1 &= \left\{\left(t_{m-j}, x_{m-j}{}^{(1)}\right), \left(t_{m-j+1}, x_{m-j+1}{}^{(1)}\right), \ldots, \left(t_k, x_k{}^{(1)}\right)\right\} \\ w_2 &= \left\{\left(t_{m-j}, x_{m-j}{}^{(2)}\right), \left(t_{m-j+1}, x_{m-j+1}{}^{(2)}\right), \ldots, \left(t_k, x_k{}^{(2)}\right)\right\} \qquad (4) \\ &\vdots \\ w_n &= \left\{\left(t_{m-j}, x_{m-j}{}^{(n)}\right), \left(t_{m-j+1}, x_{m-j+1}{}^{(n)}\right), \ldots, \left(t_k, x_k{}^{(n)}\right)\right\}. \end{aligned}$$

Using PCHIP to interpolate each wi, for i = 1, 2, . . . , n, yields functions fi. As it is a regular interpolation process in R3 then for some t = m − j, m − j + 1, . . . , k, fi satisfy

$$f_i(t) \approx x_t^{(i)} \qquad \text{for } i = 1, 2, \ldots, n. \qquad (5)$$

For instance,

$$\begin{aligned} f_i(m-j) &\approx x_{m-j}^{(i)} \\ f_i(m-j+1) &\approx x_{m-j+1}^{(i)} \qquad (6) \\ &\vdots \\ f_i(k) &\approx x_k^{(i)} \qquad \text{for } i = 1, 2, \ldots, n. \end{aligned}$$

Since we use an appropriate interpolant to interpolate the data, i.e. the one that preserves the monotonicity of the data, then the extrapolation based on this interpolation process enables us to get the next point outside of the range. It means that if we calculate $fi(t\ *)$ with $t * \in [k+1, s] \subset R$, where $s \geq k + 1$, then we obtain

$$f_i(t^*) \approx x_r^{(i)} \qquad \text{for } i = 1, 2, \ldots, n, \qquad (7)$$

where each $x_r^{(i)}$ has a similar property as $x_t^{(i)}$ in (5). In other words, if the sequence of $x_t^{(i)}$ is monotonically increasing/decreasing, so is $x_r^{(i)}$. Thus arranging vector$\mathbf{x}_r$, with $x_r^{(i)}$ being the ith entries of the vector, yields an approximate solution of the system.

Since PCHIP captures the persistent pattern of the data, then this process also enables us to generate a new sequence of solutions, obviously by considering the weakness of extrapolation method. In other words, we can still choose the integer s such that the residual norms of the iterates gener- ated by this process, x_{k+1}, x_{k+2}, . . ., x_s are small enough. It is expected that these iterates replace the"missing" iterates not generated by the Lanczos- type algorithm due to breakdown. The algorithm of EIEMLA is given in Algorithm 1.

Algorithm 1 The EIEMLA method

Initialization. Choose $\mathbf{x}_0$ and $\mathbf{y}$. Set $\mathbf{r}_0 = \mathbf{b} - A\mathbf{x}_0$, $\mathbf{y}_0 = \mathbf{y}$, and $\mathbf{z}_0 = \mathbf{r}_0$.
Fix the number of iterations to, say, k, and the tolerance, ε, to $E-13$ and run a Lanczos-type algorithm.
if $\|\mathbf{r}_k\| \leq \varepsilon$ **then**
  The solution is obtained
  Stop
else
  Collect all k vector solutions as in S.
  Choose some j such that $m - j \leq k$.
  Set w_i as in (4), for $i = 1, 2, \ldots, n$.
  Interpolate w_i using PCHIP to get f_i.
  Choose $t^* \in [m, s] \subset R$, where $s \geq m \geq k$ is an integer, and calculate $f_i(t^*)$.
  for $q = 1, 2, \ldots, l$ **do**
    Arrange vectors

$$\mathbf{x}_*{}^q = \begin{pmatrix} (f_1{}^q)(t^{*q}) \\ (f_2{}^q)(t^{*q}) \\ \vdots \\ (f_n{}^q)(t^{*q}) \end{pmatrix}, \qquad (8)$$

    where $l = length([m, s])$.
    Calculate the residual norms of (8) as follows

$$\|\mathbf{r}_*{}^q\| = \|\mathbf{b} - A\mathbf{x}_*{}^q\| \qquad (9)$$

  end for
end if
The solutions of the systems are $\mathbf{x}_*{}^{(1)}, \mathbf{x}_*^{(2)}, \ldots, \mathbf{x}_*{}^{(l)}$.
Stop.

2.1. Formal Basis of EIEMLA

As above mentioned, the sequences generated by the Lanczos-type algorithm have the property of monotonicity. Since we consider PCHIP which preserves monotonicity, [6], to interpolate the sequences, we can assume that points returned by the function, are also monotonic. This leads to the theorem below which guarantees the monotonicity property of sequences generated by Lanczos-type algorithms.

Theorem 1. Given a sequence $\{\mathbf{x}_k\}$ of k iterates generated by a Lanczos-type algorithm, sequences of $x_k^{(i)}$, i = 1, 2, … , n, and k = 1, …, namely the entries of k iterates, are monotonic.

The next theorem is to show that the distance between two vectors and $\mathbf{x}_{k+1}$ and $\mathbf{x}_{model}$ is sufficiently small, where $\mathbf{x}_{k+1}$ is iterate generated by Lanczos process, and $\mathbf{x}_{model}$ is iterate generated by EIEMLA. Theorem 2. Let $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_k$ be theiterates generated by Orthodir algorithm. Let $\mathbf{x}_{model}$ be a vector returned by EIEMLA as explained in the previous section. Then, for some $\varepsilon > 0$,

$$\|\mathbf{x}_{k+1} - \mathbf{x}_{model}\| \leq \varepsilon, \qquad (10)$$

where ||·|| is the Euclidean norm.

Finally we have a theorem which guarantees the residual norm of the iterate generated by EIEMLA is always smaller or equal to that of the iterate generated by the Lanczos-type algorithms considered.

Theorem 3. Let $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_k$ be the iterates generated by Orthodir algorithm. Let $\mathbf{r}_{model}$ be a residual vector which corresponds to the iterate generated by EIEMLA.
Then,

$$\|\mathbf{r}_{model}\| \leq (1+|\alpha|)\,\|\mathbf{r}_k\|. \qquad (11)$$

Note here, all of the proofs of theorems above can be seen in [159].

3. Restarting EIEMLA

There are some different points to restart a Lanczos algorithm which lead some different algorithms, as mentioned in [14]. In this particular restarting, called REIEMLA, we take theiterate generated by EIEMLA as a starting point. It is as illustrated in Figure 1. First, a Lanczos-type algorithm generates the sequence$\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_k$, with assuming it breaks after k iterations. The sequence is then regressed to get a model function. Using this function, we generate a solution, $\mathbf{x}^{(1)}_{model}$. Next, we restart the Lanczos-type algorithm from this solution to get another sequence of iterates. We regress again to get $\mathbf{x}^{(2)}_{model}$. It is continued until $\mathbf{x}^{(m)}_{model}$ is achieved, and the corresponding residual norm is less than the given tolerance.

By a theorem in [14], we know that the restarting framework allows Lanczos-type algorithms to generate better iterates. This is expected here, too. This restarting approach is described as Algorithm 2.

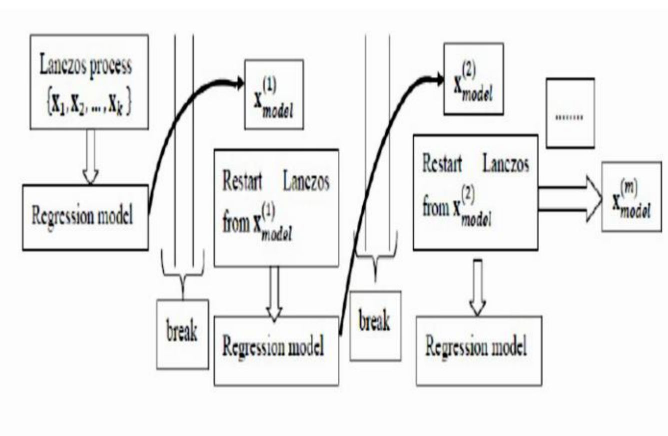


Figure 1: The process of REIEMLA on SLE's

Algorithm 2 REIEMLA

1: Initialization. Choose $\mathbf{x}_0$ and $\mathbf{y}$. Set $\mathbf{r}_0 = \mathbf{b} - A\mathbf{x}_0$, $\mathbf{y}_0 = \mathbf{y}$, and $\mathbf{z}_0 = \mathbf{r}_0$.
2: Fix the number of iterations to, say k, and the tolerance, ε, to $1E-13$.
3: Run EIEMLA for k iterations. Obtain a sequence of iterates $\{\mathbf{x}_{k+1}, \mathbf{x}_{k+2}, \ldots, \mathbf{x}_s\}$, where $s \geq> k+1$, and calculate the residual norms of these iterates.
4: Compute the minimum of the residual norms, name it as $\|\mathbf{r}_{model}\|$.
5: **if** $\|\mathbf{r}_{model}\| \leq \varepsilon$ **then**
6: The solution is obtained, i.e. the iterate which is associated with this residual norm, name it as $\mathbf{x}_{model}$.
7: Stop.
8: **else**
9: Initialize the algorithm with

$$\begin{aligned} \mathbf{x}_0 &= \mathbf{x}_{model} \\ \mathbf{y} &= \mathbf{b} - A\mathbf{x}_0 \end{aligned}$$

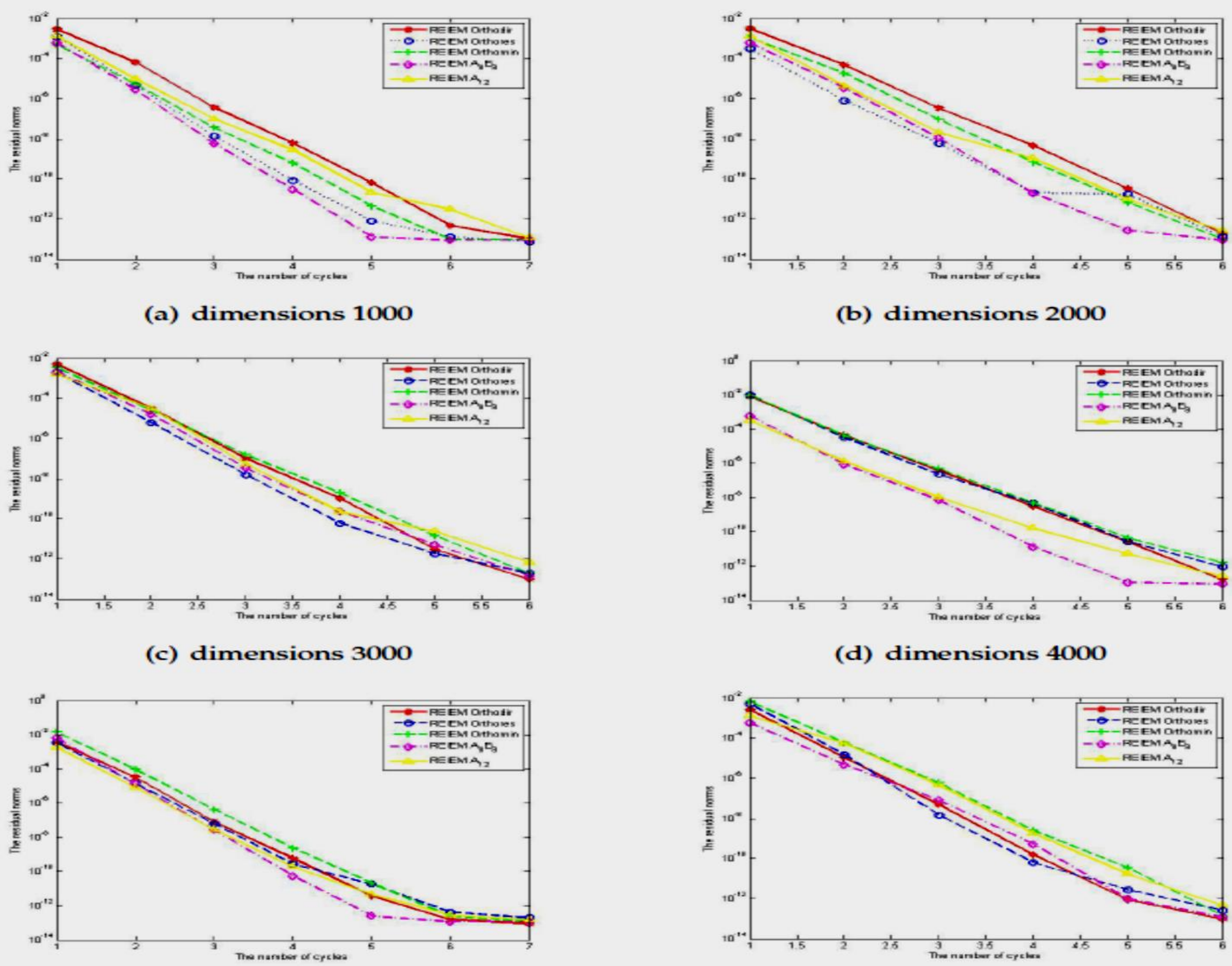
10: Go to 3.
11: **end if**
12: Take $\mathbf{x}_{model}$ as the approximate solution.
13: Stop.

4. Numerical Results Using Sparse Matrix:

Experiments have been carried out using five implementations of REIEMLA's, including restarting EIEM (REIEM) Orthores, REIEM Orthodir, REIEM Orthomin, REIEM A8B8, and REIEM A12. The aim is to look at the performance of these EIEMLA's when they are put on the restarting framework. Particularly, we will look at the robustness and the efficiency of each algorithm. The problems solved range from 1000 to 400000 variables. Overall, REIEM A8B8 found more accurate approximate solutions. This can be seen in Table (1), particularly for dimensions between 60000 and 400000. It also showed the best performance in term of efficiency; it consistently took the shortest time on all problems. The second best performance came from REIEM Orthodir. The rest of methods had mixed performances on both accuracy and efficiency. Figures 2, 3, 4, and 5 represent the residual norms of all solutions generated by all considered algorithms for dimensions ranging from 1000 to 8000, 9000 to 70000, and 80000 to 400000, respectively. We can see there, that most of figures show that the red, blue, pink, and yellow curves, corresponding to REIEM Orthodir, REIEM Orthores, REIEM A8B8, and REIEM A12, respectively, have a similar shape. The green curve, on the other hand, which represents REIEM Orthomin, appears on top of the other curves for some problems, such as on Figures from 4(c) to 5(d). It means that it failed to reach the prescribed convergence tolerance.

Table 1: REIEMLA's results on SLE's of different dimensions ($\delta = 0.2$).

Dim	REIEM Orthodir			REIEM Orthores			REIEM Orthomin			REIEM A_8B_8			REIEM A_{12}		
n	$\|r_{model}\|$	$T(s)$	cycles*	$\|r_{model}\|$	$T(s)$	cycles*	$\|r_{model}\|$	$T(s)$	cycles*	$\|r_{model}\|$	$T(s)$	cycles*	$\|r_{model}\|$	$T(s)$	cycles*
1000	1.0429E−13	2.7754	7	7.4768E−14	2.7204	7	1.0055E−13	2.4289	7	8.8728E−14	2.3128	7	1.1678E−13	2.8407	7
2000	1.8022E−13	4.7902	5	1.3016E−13	4.7256	5	9.7620E−14	4.7388	5	8.9780E−14	4.5694	5	2.6154E−13	4.8302	5
3000	9.5558E−14	8.2847	6	1.9059E−13	8.1635	6	1.8697E−13	8.2978	6	1.3804E−13	7.8779	6	6.6643E−13	8.3374	6
4000	1.5314E−13	11.1859	6	9.0616E−13	11.0619	6	1.5536E−12	11.2805	6	8.7363E−14	10.6054	6	2.4647E−13	11.2323	6
5000	8.8714E−14	13.8305	6	1.7117E−13	13.6183	6	1.1540E−13	13.6849	6	8.9895E−14	13.3218	6	1.5963E−13	13.9249	6
6000	9.3580E−14	16.6379	6	2.5422E−13	16.4647	6	1.3951E−13	16.5377	6	1.1735E−13	14.9176	6	4.5703E−13	16.6322	6
7000	8.8991E−14	21.5419	7	9.4050E−14	21.2915	7	8.7140E−14	21.4752	7	1.0280E−13	16.4627	6	1.1965E−13	21.7140	7
8000	7.0246E−14	26.1215	7	1.7535E−13	26.6733	7	9.9997E−14	27.6451	7	1.0766E−13	24.4211	7	4.0340E−13	28.7450	7
9000	1.2724E−13	34.7815	6	2.3651E−13	34.5213	6	2.7370E−13	36.4677	6	9.3160E−14	32.6551	6	1.3555E−13	34.5217	6
10000	1.1770E−13	43.3406	7	1.8943E−13	42.9775	7	8.5523E−14	42.2791	7	1.1804E−13	36.6519	7	1.3061E−13	43.7932	7
20000	1.3307E−13	78.4202	6	2.5346E−13	75.5837	6	1.3043E−13	76.6608	6	9.6495E−14	74.1202	6	2.1849E−13	77.1374	6
30000	7.0624E−14	1.3807E+02	8	1.3414E−13	1.3828E+02	8	1.2435E−13	1.1271E+02	8	1.3085E−13	1.2985E+02	8	8.3067E−14	1.4636E+02	8
40000	8.7169E−14	1.6587E+02	7	2.0691E−13	1.6405E+02	7	1.1421E−13	1.6513E+02	7	1.5730E−13	1.2543E+02	7	1.3643E−13	1.6427E+02	7
50000	8.8856E−14	2.0350E+02	7	1.9407E−13	2.0388E+02	7	1.1142E−13	2.0585E+02	7	1.1251E−13	1.7703E+02	7	1.4125E−13	2.0489E+02	7
60000	1.0238E−13	2.2516E+03	6	2.5612E−13	2.2548E+03	6	5.7458E−13	1.8015E+03	6	5.7665E−14	1.7265E+03	6	1.0136E−13	2.6891E+03	6
70000	1.1291E−13	2.3504E+03	6	2.5633E−13	2.3568E+03	6	2.3774E−07	2.8314E+03	6	5.4051E−14	2.3329E+03	6	8.5146E−12	9.6349E+02	6
80000	1.1291E−13	3.7782E+02	6	5.5238E−13	3.7811E+02	6	9.5690E−09	3.7915E+02	6	6.9059E−14	3.8115E+02	6	2.5610E−13	3.7799E+02	6
90000	1.2560E−13	4.1922E+02	7	3.5634E−13	4.1968E+02	7	4.4348	4.2066E+02	6	5.5320E−14	4.2106E+02	7	4.6976E−11	4.2059E+02	7
100000	6.7572E−14	1.9632E+03	6	2.7009E−13	1.9678E+03	6	2.2545E−07	2.3319E+03	6	1.8846E−14	2.0884E+03	6	2.0253E−11	1.7952E+03	6
200000	1.5334E−13	3.0293E+03	6	2.4225E−12	3.0356E+03	6	6.3091E−09	2.8280E+03	6	1.8846E−14	2.7299E+03	6	1.8905E−13	3.1738E+03	6
300000	3.5587E−13	1.4159E+03	6	1.3904E−11	1.3917E+03	6	2.9327E−08	1.4026E+03	6	9.1562E−14	1.4166E+03	6	3.4152E−13	1.4245E+03	6
400000	1.0917E−13	3.0293E+03	7	2.8929E−13	2.1188E+03	7	0.9752	2.1174E+03	7	6.3743E−14	2.0909E+03	7	1.7095E−11	2.1298E+03	7

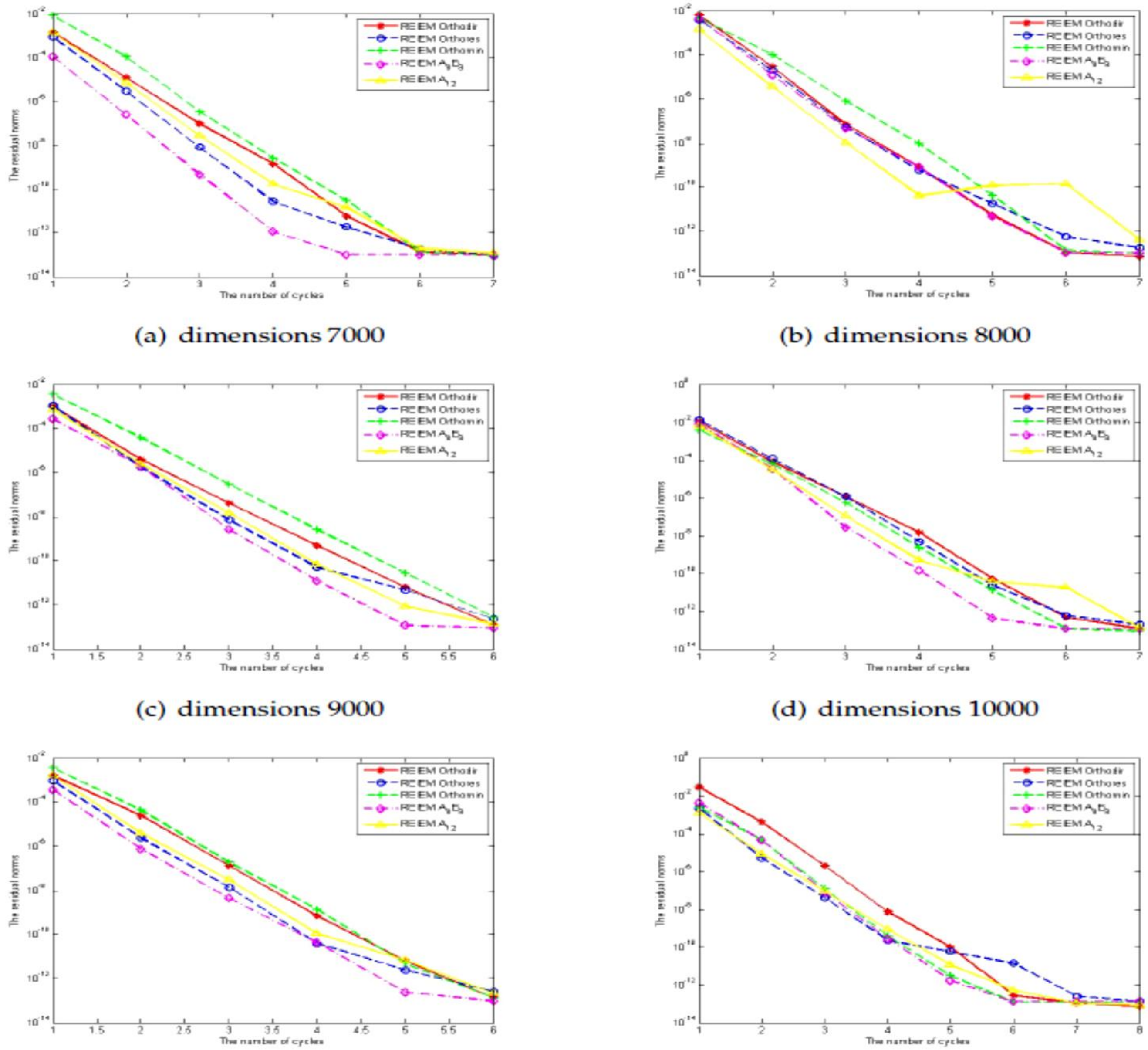


(a) dimensions 1000 (b) dimensions 2000

(c) dimensions 3000 (d) dimensions 4000

(e) dimensions 5000 (f) dimensions 6000

Figure 2: The performance of REIEMLA's for the case of $\delta = 0.2$, dimensions 1000 to 6000



(a) dimensions 7000

(b) dimensions 8000

(c) dimensions 9000

(d) dimensions 10000

(e) dimensions 20000

(f) dimensions 30000

Figure 3: The performance of REIEMLA's for the case of $\delta = 0.2$, dimensions 7000 to 30000

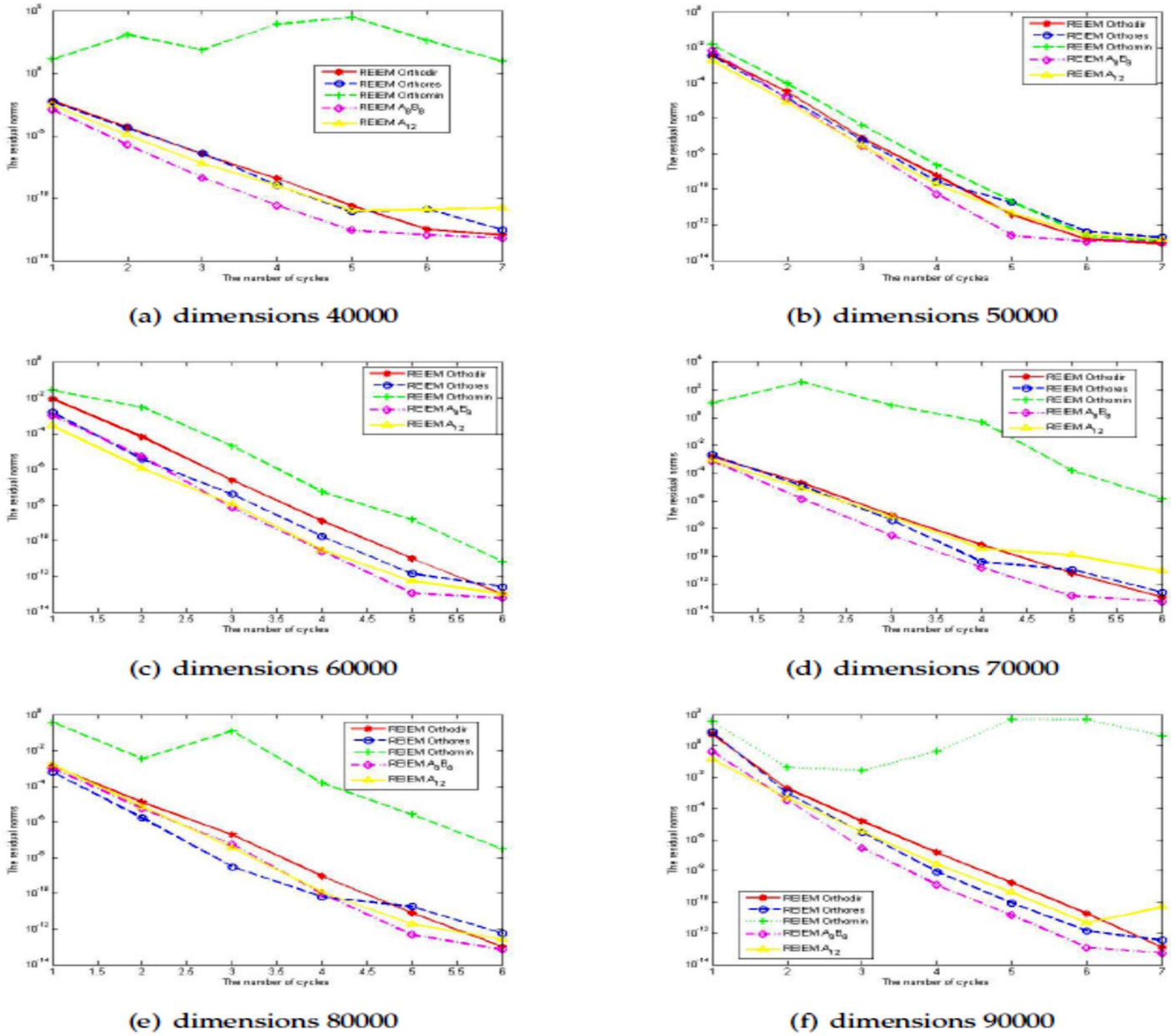


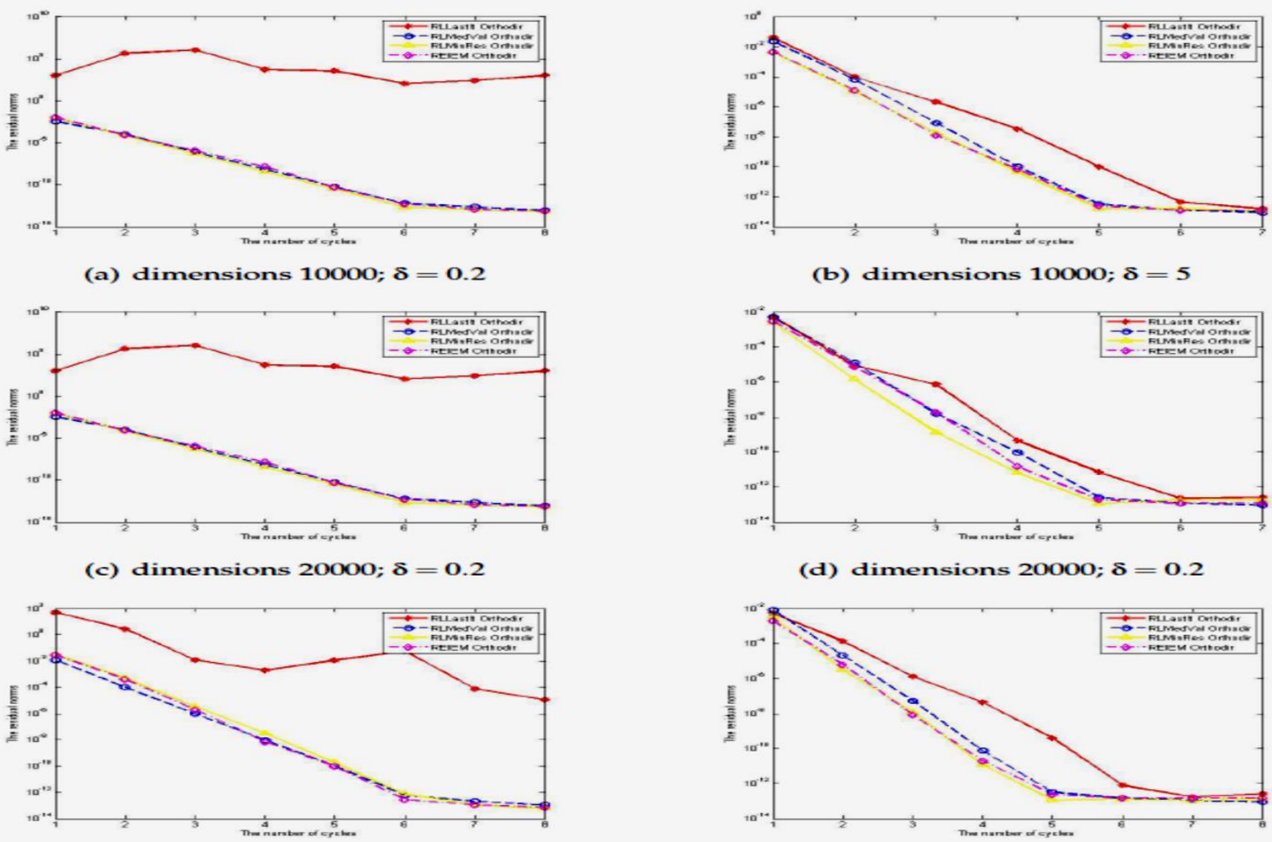
Figure 4: The performance of REIEMLA's for the case of $\delta = 0.2$, dimensions 40000 to 90000

4.1. Comparison of REIEMLA, RLastIt, RLMinRes, and RLMedVal

This section compares REIEMLA and other restarting of Lanczos-type algorithms discussed in [14], including RLLastIt, RLMinRes, and RLMedVal. The aim of this study is to look at which starting point improves the performance of Lanczos-type algorithms, in this case, we used Orthodir algorithm which is one of Lanczos-types. In addition, the stability of RLLastIt, RLMinRes, and RLMedVal will be checked for problems ranging from 10000 to 1000000 dimensions, which are significantly larger than those in [8]. All of the results are recorded in Table 2 and visualized in Figures 6, 7, 8, 9, 10, 11, dan 12. According to the results, REIIEMLA performed the best than other restarting strategies, though it is the slowest. According to Table 2, overall, for $\delta = 0.2$, REIEM Orthodir produced more accurate approximate solutions than RLLastIt Orthodir, RLMedVal Orthodir, and RLMinRes Orthodir. However, it was the slowest. RLMinRes Orthodir was the second best for accuracy and it was the fastest. RLLastIt Orthodir, on the other hand, was the worst overall. It still suffered from breakdown (see dimensions 20000, 60000, 90000 column). Yet, for dimensions 1000000, RLLastIt Orthodir produced an approximate solution with a residual norm of 299.2. RLMedVal Orthodir was the third best on accuracy. The computational time of RLMedVal Orthodir is rather low, compared to that of REIEM Orthodir.

Table 2: Comparison of RLLastIt, RLMinRes, RLMedVal, and REIEMLA on SLE's.

Dim	RLLastIt			RLMedVal			RLMinRes			REIEMLA		
n	$\|r_{last}\|$	$T(s)$	cycles*	$\|r_{medval}\|$	$T(s)$	cycles*	$\|r_{min}\|$	$T(s)$	cycles*	$\|r_{model}\|$	$T(s)$	cycles*
10000	$1.0197E+03$	1.6417	8	$8.2954E-14$	3.4869	8	$6.3812E-14$	1.6030	8	$7.2723E-14$	46.8517	8
20000	NaN	1.7079	7	NaN	8.2981	7	$1.3651E-13$	3.1302	7	$1.3778E-13$	83.4271	7
30000	$1.1431E-05$	4.5129	8	$1.0587E-13$	9.5411	8	$5.6777E-14$	3.36974	8	$7.0624E-14$	$1.3811E+02$	8
40000	$1.0034E-13$	5.1161	7	$1.3915E-13$	11.6754	7	$1.3337E-13$	15.0736	7	$1.0416E-13$	$1.4671E+02$	7
50000	$1.9219E-11$	5.6047	7	$3.0603E-13$	11.9592	7	$1.2119E-13$	4.9212	7	$8.8856E-14$	$1.6158E+02$	7
60000	NaN	2.7684	2	NaN	16.8510	2	$9.3471E-14$	5.7720	8	$9.4736E-14$	$2.1146E+02$	8
70000	$8.6199E-04$	7.1957	7	$5.1839E-13$	16.1976	7	$1.1174E-13$	6.3761	7	$8.8579E-14$	$2.2149E+02$	7
80000	$1.0629E-13$	8.3211	8	$1.3325E-13$	20.0473	8	$1.3564E-13$	6.5941	8	$8.7312E-14$	$2.8178E+02$	8
90000	NaN	15.0871	4	$2.2863E-13$	22.4101	8	$6.7621E-14$	9.1351	8	$9.0221E-14$	$3.1943E+02$	8
100000	$8.4367E-06$	19.8926	6	$2.7951E-12$	38.6984	6	$1.4457E-13$	17.7201	6	$6.7572E-14$	$4.6079E+02$	6
200000	$2.2371E-07$	53.0010	7	$3.0373E-13$	92.9957	7	$1.0271E-13$	42.4940	7	$7.3917E-14$	$1.0452E+03$	7
300000	$3.6456E-05$	84.0492	8	$2.9710E-13$	$1.3644E+02$	8	$9.7195E-14$	69.0920	8	$5.7715E-14$	$1.7206E+03$	8
400000	$1.0269E-06$	$1.2527E+02$	21	$4.0323E-13$	$2.0462E+02$	21	$1.0179E-13$	$1.0265E+02$	21	$6.2753E-14$	$2.2990E+03$	21
500000	4.0095	$1.2553E+02$	7	$4.9181E-13$	$2.2944E+02$	7	$1.1028E-13$	$1.0562E+02$	7	$7.5120E-14$	$2.5773E+03$	7
600000	$2.3792E-08$	$2.1644E+02$	8	$3.3380E-13$	$3.3367E+02$	8	$1.1534E-13$	$1.6817E+02$	8	$6.9701E-14$	$3.5818E+03$	8
700000	$8.8101E-04$	$9.3461E+02$	8	$4.8528E-13$	$1.1483E+03$	8	$1.4392E-13$	$1.3074E+04$	8	$9.6797E-14$	$1.1499E+02$	8
800000	0.0018	$1.0005E+03$	8	$3.1252E-13$	$1.3431E+03$	8	$1.3669E-13$	$8.7973E+02$	8	$9.0274E-14$	$1.3790E+04$	8
900000	$1.7343E-06$	$2.9709E+02$	9	$3.5152E-13$	$4.9551E+02$	9	$1.0190E-13$	$2.0792E+02$	9	$7.8603E-14$	$5.6509E+03$	9
1000000	299.2002	$3.4414E+02$	10	$2.7200E-13$	$5.7312E+02$	10	$1.1225E-13$	$2.7021E+02$	10	$7.8631E-14$	$5.7529E+03$	10

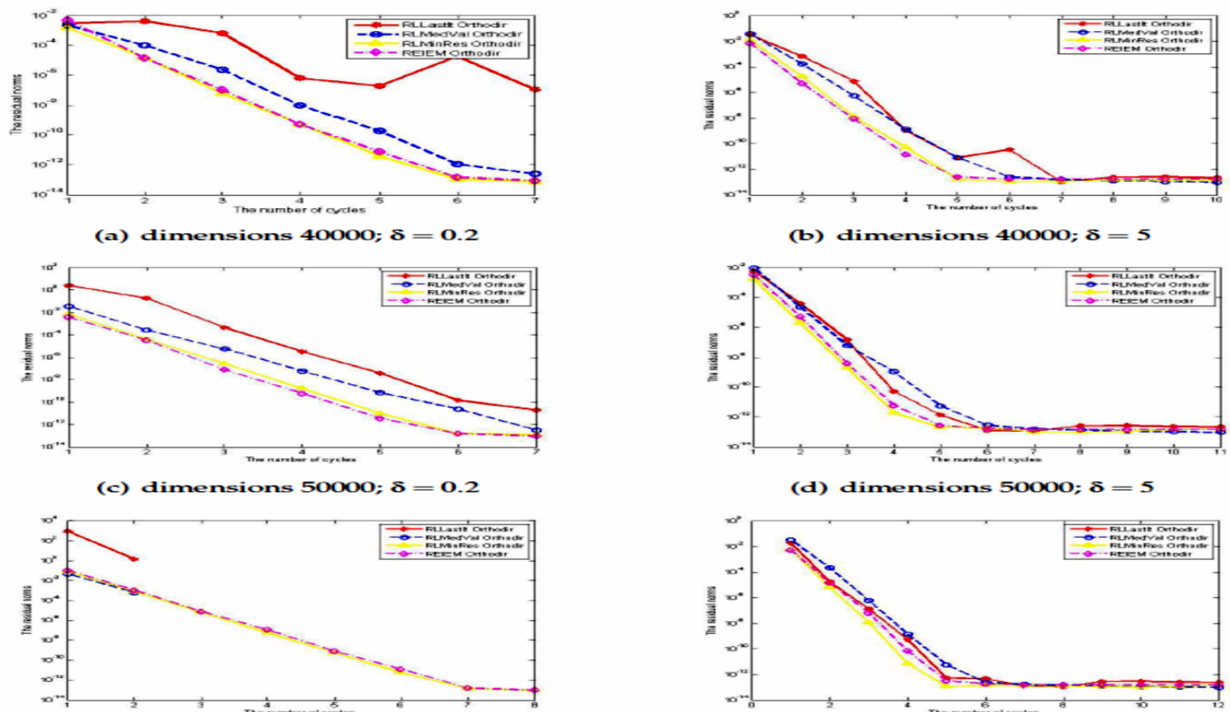


(a) dimensions 10000; $\delta = 0.2$ (b) dimensions 10000; $\delta = 5$

(c) dimensions 20000; $\delta = 0.2$ (d) dimensions 20000; $\delta = 0.2$

(e) dimensions 30000; $\delta = 0.2$ (f) dimensions 30000; $\delta = 5$

Figure 6: The performances of RLLastIt Orthodir, RLMinres Orthodir, RLMedVal Orthodir and REIEM Orthodir on SLE's for $\delta = 0.2$ and $\delta = 5$, dimensions 10000 to 30000



(a) dimensions 40000; $\delta = 0.2$

(b) dimensions 40000; $\delta = 5$

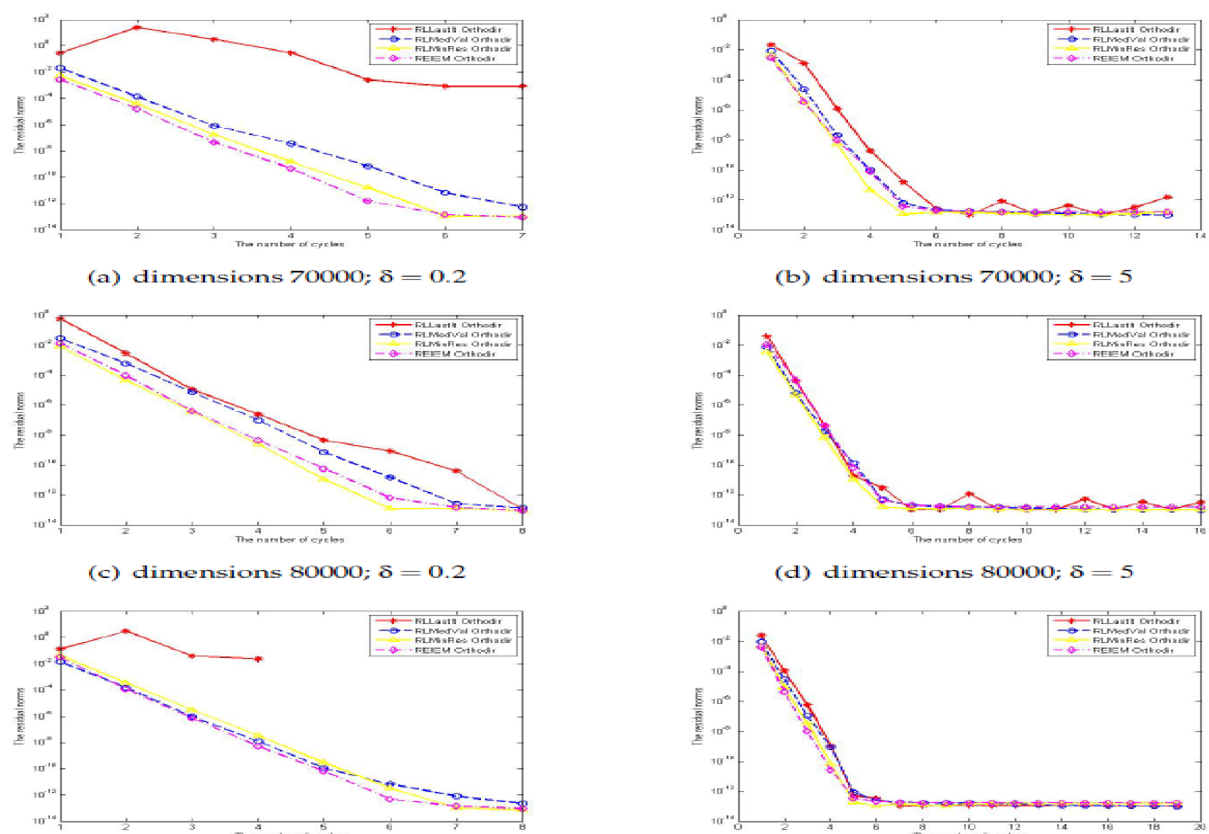
(c) dimensions 50000; $\delta = 0.2$

(d) dimensions 50000; $\delta = 5$

(e) dimensions 60000; $\delta = 0.2$

(f) dimensions 60000; $\delta = 0.2$

Figure 7: The performances of RLLastIt Orthodir, RLMinres Orthodir, RLMedVal Orthodir and REIEM Orthodir on SLE's for $\delta = 0.2$ and $\delta = 5$, dimensions 40000 to 60000



(a) dimensions 70000; $\delta = 0.2$

(b) dimensions 70000; $\delta = 5$

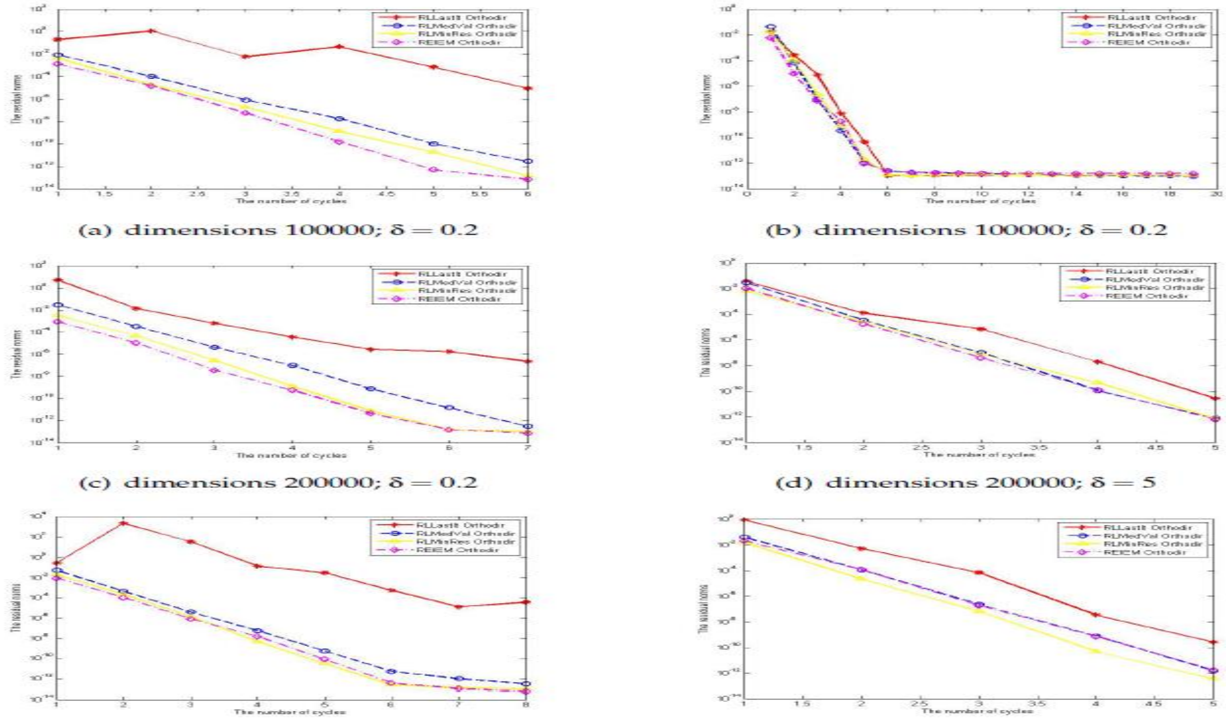
(c) dimensions 80000; $\delta = 0.2$

(d) dimensions 80000; $\delta = 5$

(e) dimensions 90000; $\delta = 0.2$

(f) dimensions 90000; $\delta = 5$

Figure 8: The performances of RLLastIt Orthodir, RLMinres Orthodir, RLMedVal Orthodir and REIEM Orthodir on SLE's for $\delta = 0.2$ and $\delta = 5$, dimensions 70000 to 90000



(a) dimensions 100000; $\delta = 0.2$

(b) dimensions 100000; $\delta = 0.2$

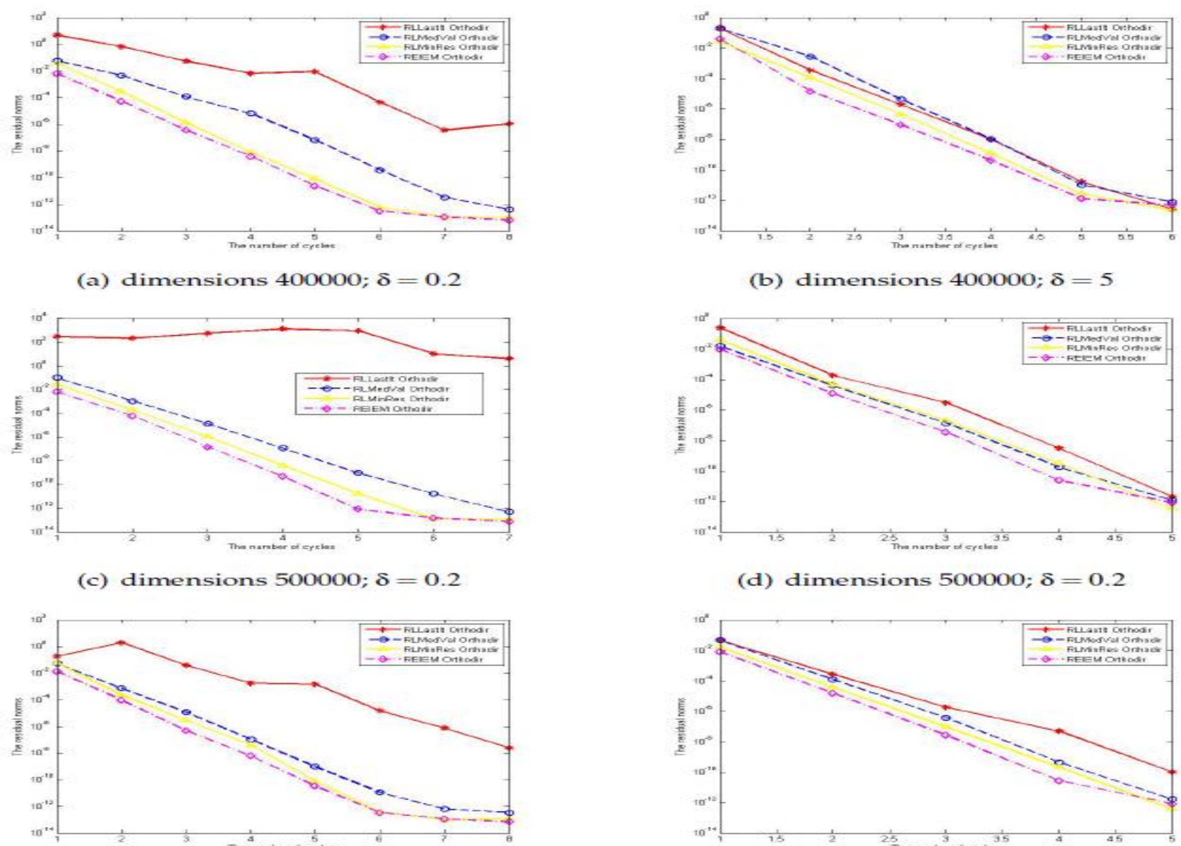
(c) dimensions 200000; $\delta = 0.2$

(d) dimensions 200000; $\delta = 5$

(e) dimensions 300000; $\delta = 0.2$

(f) dimensions 300000; $\delta = 5$

Figure 9: The performances of RLLastIt Orthodir, RLMinres Orthodir, RLMedVal Orthodir and REIEM Orthodir on SLE's for $\delta = 0.2$ and $\delta = 5$, dimensions 100000 to 300000



(a) dimensions 400000; $\delta = 0.2$

(b) dimensions 400000; $\delta = 5$

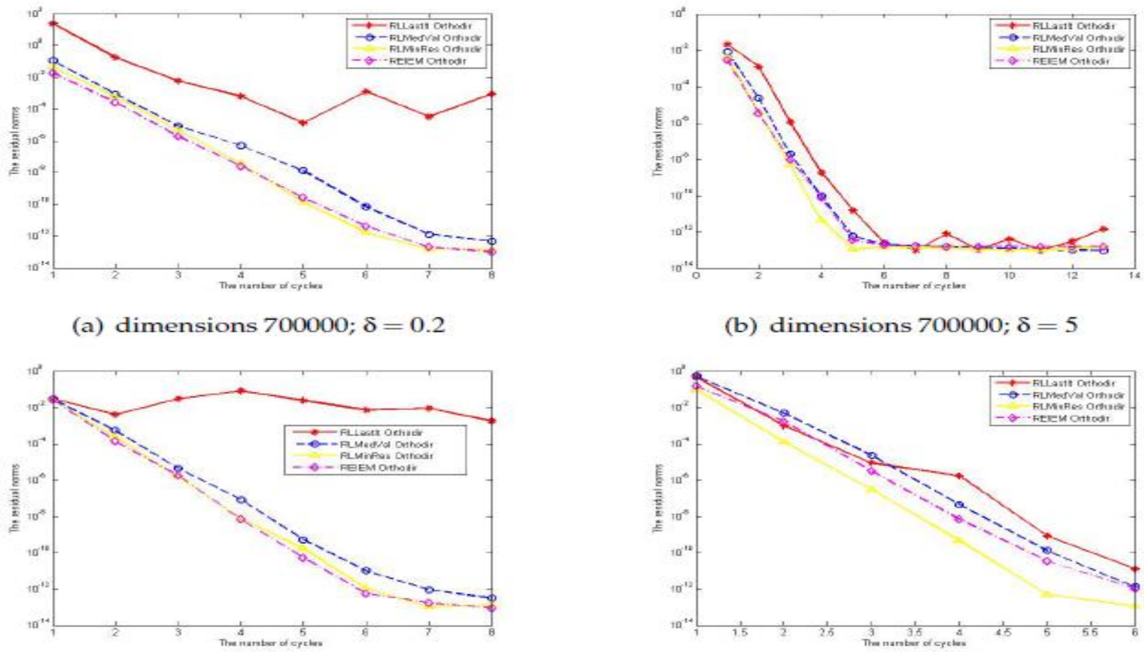
(c) dimensions 500000; $\delta = 0.2$

(d) dimensions 500000; $\delta = 0.2$

(e) dimensions 600000; $\delta = 0.2$

(f) dimensions 600000; $\delta = 5$

Figure 10: The performances of RLLastIt Orthodir, RLMinres Orthodir, RLMedVal Orthodir and REIEM Orthodir on SLE's for $\delta = 0.2$ and $\delta = 5$, dimensions 400000 to 600000



(a) dimensions 700000; $\delta = 0.2$ (b) dimensions 700000; $\delta = 5$

(c) dimensions 800000; $\delta = 0.2$ (d) dimensions 800000; $\delta = 5$

Figure 11: The performances of RLLastIt Orthodir, RLMinres Orthodir, RLMedVal Orthodir and REIEM Orthodir on SLE's for $\delta = 0.2$ and $\delta = 5$, dimensions 700000 and 800000

The comparison of REIEM Orthodir, RLLastIt Orthodir, RLMedVal Orthodir, and RLMinRes Orthodir for different values of δ can be seen in some figures above mentioned. Figures in the first column show the behaviour of 4 restarting for $\delta = 0.2$, while those in the second column show that for $\delta = 5$. We can see in the first column that for most problems, the red curve, which represents RLLastIt Orthodir, is on top. It indicates that this restarting failed to achieve approximate solutions with the required tolerance. On the second column, in contrast, the red curve along with other curves hit small residual norm

5. CONCLUSION

Restarting from the iterate generated by EIEMLA's (REIEMLA's) have been implemented. This kind of restarting uses an iterate generated by EIEMLA as a starting point which is different from either those investigated in [8] or in [14], which is a novelty of our works. We can conclude here that REIEMLA produced the best results as can be seen in Table 2 of Section 4.1, so they do agree with the theory expanded in Section 3. This means that the method is comparatively better in terms of quality of solution. However, given the time overheads required to find the regression model and restarting, both REIEMLA is not efficient in terms of computing time.

REFERENCES

[1] C. Baheux, New Implementations of Lanczos Method, Journal of Computational and Applied Mathematics 57 (1995) 3–15.

[2] C. Brezinski, R. Zaglia, A New Presentation of Orthogonal Polynomials with Applications to their Computation, Numerical Algorithms 1 (1991) 207–221.

[3] C. Brezinski, H. Sadok, Lanczos-type Algorthms for Solving Systems of Linear Equation, Applied Numerical Mathematics 11 (1993) 443–473.

[4] Brezinski, C. and Zaglia, R. and Sadok,H., Avoiding breakdown and near-breakdown in Lanczos type algorithms, Numerical Algorithms, 1, 261–284, 1991.

[5] Brezinski, C. and Zaglia, R. and Sadok, H., A breakdown-free Lanczos type algorithm for solving linear systems, Numerical Mathematics, 63, 29–38, 1992.

[6] R. Delbourgo, J. Gregory, Shape Preserving Piecewise Rational Interpolation, SIAM Journal on Scientific and Statistical Computing 10 (1983).

[7] M. Farooq, New Lanczos-type Algorithms and their Implementation, Ph.D. thesis, Department of Mathematical Sciences, University of Essex, 2011.

[8] M. Farooq, A. Salhi, A Preemptive Restarting Approach to Beating Inherent Instability, Iranian

Journal of Science and Technology Transaction a Science 12 (2012).

[9] M. Farooq, A. Salhi, A Switching Approach to Avoid Breakdown in Lanczos-type Algorithms, Applied Mathematics and Information Sciences 8 (2014) 2161–2169.

[10] F. N. Fritsch, R. Carlson, Monotone Piecewise Cubic Interpolation, SIAM Journal Numerical Analysis 17 (1980) 239–248.

[11] F. N. Fritsch, J. Butland, A Method for Constructing Local Monotone Piecewise Cubic Interpolants, SIAM J. Sci. Stat. Comput. 5 (1984) 300–304.

[12] P. Grave-Morris, A Look-Around Lanczos Algorithm for Solving a System of Linear Equations, Numerical Algorithm 15 (1997) 247–274.

[13] E. Guennouni, A Unified Approach to Some Strategies for the Treatment of Breakdown in Lanczos-type Algorithms, Application Mathematics 26 (1999) 477–488.

[14] M. Maharani, A. Salhi, Restarting from Specific Points to Cure Breakdown in Lanczos-type Algorithms, Journal of Mathematical and Fundamental Sciences 2 (2015) 167–184.

[15] M. Maharani, A. Salhi, Enhancing the Stability of Lanczos-type Algorithms by Embedding Interpolation and Extrapolation for the Solution of Systems of Linear Equations, 2016.

[16] A. Sidi, F.William Ford, A. David Smith, Acceleration of Convergence of Vector Sequences, SIAM Journal on Numerical Analysis 23 (1986) 178–196.