A preemptive restarting approach to beating the inherent instability of Lanczos-type algorithms

M. Farooq1* and A. Salhi2

1Department of Mathematics, University of Peshawar, 25120, Khyber Pakhtunkhwa, Pakistan
2Department of Mathematical Sciences, University of Essex, Wivenhoe Park, Colchester, CO4 3SQ, UK
E-mails: mfarooq@upesh.edu.pk & as@essex.ac.uk

Abstract

Lanczos-type algorithms are well known for their inherent instability. They typically breakdown occurs when relevant orthogonal polynomials do not exist. Current approaches to curing breakdown rely on jumping over the non-existent polynomials to resume computation. This may have to be used many times during the solution process. We suggest an alternative to jumping, which consists of restarting the algorithms that fail. Three different strategies can be taken: (ST1) Restarting following breakdown of the algorithm in use; (ST2) pre-emptive restarting after a fixed number of iterations; (ST3) restarting when near breakdown is detected through monitoring. We describe a restarting framework with a generic algorithm that invokes one or the other of the three strategies suggested. Four of the most prominent recently developed Lanczos-type algorithms namely, \( A_{12}, A_5/B_{10} \) and \( A_k/B_{10} \) will be presented and then deployed in the restarting framework. However, we will only report on results obtained with strategy ST2 as it is the only viable one at the moment.

Keywords: Lanczos algorithm; Systems of Linear Equations; Formal Orthogonal Polynomials, Restarting, Switching, Breakdown

1. Introduction

The Lanczos-type methods for solving Systems of Linear Equations (SLEs) are based on the theory of Formal Orthogonal Polynomials (FOPs). All such methods are implemented via some recurrence relationships between polynomials \( P_k(x) \) represented by \( A_i \), or between two adjacent families of orthogonal polynomials \( P_k(x) \) and \( P^{(i)}_k(x) \) represented by \( A_i \) and \( B_j \) as described in [1, 2]. The coefficients of the various recurrence relationships for orthogonal polynomials are given as ratios of scalar products. When a scalar product in a denominator vanishes, then a breakdown occurs in the algorithm and the process normally has to be stopped. Equivalently, the breakdown is due to the non-existence of some orthogonal polynomials. So quite an important problem is how to be able to continue the solution process in such a situation and arrive at a usable result. Several procedures for that purpose have appeared in the literature in the last few years. It has been proved that it is possible to jump over non-existing orthogonal polynomials [3], and breakdown-free algorithms were thus obtained. The first attempt in this regard was the look-ahead Lanczos algorithm [4]. Other procedures for avoiding breakdown are also proposed in [5-12]. The procedure for jumping over non-existing orthogonal polynomials can be described as follows:

1. Recognize the occurrence of breakdown;
2. Determine the degree of the next existing (regular) orthogonal polynomial;
3. Find the recurrence relationship to compute it, i.e. jump over the non-existing orthogonal polynomials.

This idea of the degree of the next regular polynomial or the length of the jump is explained in [13-19]. Note that it is not our purpose here to compare the advantages of these various methods.

2. The Lanczos approach

We consider a linear system of equations,

\[ Ax = b, \tag{1} \]

where \( A \in \mathbb{R}^{n \times n} \), \( b \in \mathbb{R}^{n \times 1} \) and \( x \in \mathbb{R}^{n \times 1} \).

Let \( x_0 \) and \( y \) be two arbitrary vectors in \( \mathbb{R}^n \) such that \( y \neq 0 \). The Lanczos method [20] consists of constructing a sequence of vectors \( x_k \in \mathbb{R}^n \) defined as follows [21],

*Corresponding author
Received: 23 July 2012 / Accepted: 1 December 2012
where $A^T$ denotes the transpose of $A$.

Equation (2) leads to,

$$x_k - x_0 = -a_1 x_0 - a_2 x_0 - \cdots - a_k A^{-1} r_0, \quad (4)$$

Multiplying both sides by $A$ and adding and subtracting $b$ on the left hand side gives

$$r_k = r_0 + a_1 A r_0 + a_2 A^2 r_0 + \cdots + a_k A^k r_0, \quad (5)$$

From (3), the orthogonality condition gives

$$(A^T y, r_k) = 0, \quad \text{for } i = 0, \ldots, k - 1,$$

and, by (5), we obtain the following system of linear equations

$$\begin{cases}
\alpha_i (y, r_0) + \cdots + \alpha_i (y, A^i r_0) = (y, r_i) \\
\alpha_i ((A^i)^T y, r_0) + \cdots + \alpha_i ((A^i)^T y, A^i r_0) = ((A^i)^T y, r_i)
\end{cases} \quad (6)$$

If the determinant of the above system is different from zero then its solution exists and allows us to obtain $x_k$ and $r_k$. Obviously, in practice, solving the above system directly for the increasing value of $k$ is not feasible. We shall now see how to solve this system for increasing values of $k$ recursively. If we set

$$P_k(x) = 1 + x + \cdots + \alpha_k x^k, \quad (7)$$

then we can write from (5)

$$r_k = P_k(A) r_0. \quad (8)$$

The polynomials $P_k$ are commonly known as the residual polynomials.

Another interpretation of the $P_k$ can be found in [22]. Moreover, if we set

$$c_i = (A^T y, r_0) = (y, A^i r_0), \quad i = 0, 1, \ldots$$

and if we define the linear functional $c$ on the space of polynomials by

$$c(x^i) = c_i, \quad i = 0, 1, \ldots \quad (9)$$

$c$ is completely determined by the sequence $\{c_k\}$ and $c_k$ is said to be the moment of order $k$, [23].

Now, the system (6) can be written as

$$c(x^i P_k(x)) = 0, \quad \text{for } i = 0, \ldots, k - 1. \quad (10)$$

These conditions show that $P_k$ is the polynomial of degree at most $k$, normalized by the condition $P_k(0) = 1$, belonging to POP’s with respect to the linear functional $c$ [23], which is briefly reviewed in Section 3. For more details see [24]. Such a family is called family of orthogonal polynomials with respect to the functional $c$ and the preceding relations are called the orthogonality relations.

Since the constant term of $P_k$ in (7) is 1, it can be written as

$$P_k(x) = 1 + x R_{k-1}(x)$$

where $R_{k-1}(x) = \alpha_0 + \alpha_2 x + \cdots + \alpha_k x^{k-1}$. Replacing $x$ by $A$ in the expression of $P_k$ and multiplying both sides by $r_0$ and using (8), we get

$$r_k = r_0 + AR_{k-1}(A) r_0,$$

which can be written as

$$b - Ax_k = b - Ax_0 + AR_{k-1}(A) r_0,$$

or

$$-Ax_k = -Ax_0 + AR_{k-1}(A) r_0.$$ 

Multiplying both sides by $-A^{-1}$, we get

$$x_k = x_0 - R_{k-1}(A) r_0,$$

which shows that $x_k$ can be computed from $r_k$ without inverting $A$.

3. Formal orthogonal polynomials

The orthogonal polynomials $P_k$ defined in the previous section are given by the determinantal formula,

$$\det \begin{pmatrix} 1 & \cdots & x^i \\ c_0 & \cdots & c_i \\ \vdots & \ddots & \vdots \\ c_0 & \cdots & c_i \\ \vdots & \ddots & \vdots \\ c_0 & \cdots & c_i \\ c_k & \cdots & c_{k+i-1} \\ \vdots & \ddots & \vdots \\ c_k & \cdots & c_{k+i-1} \end{pmatrix} \quad (11)$$

$$p_k(x) = \frac{1}{\det} \begin{pmatrix} x & \cdots & 1 \\ c_0 & \cdots & c_i \\ \vdots & \ddots & \vdots \\ c_0 & \cdots & c_i \\ \vdots & \ddots & \vdots \\ c_0 & \cdots & c_i \\ c_k & \cdots & c_{k+i-1} \\ \vdots & \ddots & \vdots \\ c_k & \cdots & c_{k+i-1} \end{pmatrix}$$
where the denominator of this polynomial is $H_k^{(1)}$ [25]. Obviously, $P_k$ exists if and only if the Hankel determinant $H_k^{(1)} \neq 0$, thus, $P_{k+1}$ exists if and only if $H_k^{(1)} \neq 0$. We assume that for all $k$, $H_k^{(1)} \neq 0$, if for some $k$, $H_k^{(1)} = 0$, then $P_k$ does not exist and breakdown occurs in the algorithm (in numerical calculations the breakdown can occur even if $H_k^{(1)} \approx 0$).

Let us now define a linear functional $c^{(1)}$ [25] on the space of real polynomials as $c^{(1)}(x) = c(x^{(i)}) = c_{i+1}$ and let $P_k^{(1)}$ be a family of orthogonal polynomials with respect to $c^{(1)}$. These polynomials are called monic polynomials [25] (The polynomial in which the highest degree coefficient is always 1) and are given by the following formula:

$$
P_k^{(1)}(x) = \begin{bmatrix}
eq 1 & \ldots & \neq i+1 & \ldots & \neq n+m\neq 1 & \ldots & \neq i+1 & \ldots & \neq n+m\neq 1 & \ldots & \neq i+1 & \ldots & \neq n+m\neq 1 & \ldots & \neq i+1 & \ldots & \neq n+m\neq 1 & \ldots & \neq i+1 & \ldots & \neq n+m\neq 1 & \ldots & \neq i+1 & \ldots & \neq n+m\end{bmatrix}$$

$P_k^{(1)}(x)$ also exists if and only if the Hankel determinant $H_k^{(1)} \neq 0$ [25], which is also a condition for the existence of $P_k(x)$. There exist many recurrence relations between the two adjacent families of polynomials $P_k$ and $P_k^{(1)}$ [25]. Some of these recurrence relations have been reviewed by Magnus, Gragg, Draux, and Gutknecht describing the recursions for formal orthogonal polynomials (FOPs) and, on the other hand, an investigation by Parlett and his student Taylor. See, e.g., Gutknecht’s survey paper [26] for a description of the historic development and for the exact references. See also [27, 28] for more details. Some more recurrence relations between the two families of orthogonal polynomials have been studied in [29], leading to new Lanczos-type algorithms.

A Lanczos-type method consists of computing $P_k$ recursively, then $r_k$ and finally $x_k$ such that $r_k = b - Ax_k$, without inverting $A$, which gives the solution of the system $Ax = b$ in at most $n$ steps, in exact arithmetic, where $n$ is the dimension of the linear system.

4. Recalling some existing algorithms

In the following we will recall some of the most recent and efficient Lanczos-type Algorithms to be used in the restarting framework. The reader should consult the relevant literature for more details.

4.1. Algorithm A12

Algorithm A12 is based on relation $A_{12}$. For details on the derivation of relation $A_{12}$, its coefficients and the algorithm itself, please refer to [29]. The pseudo-code of Algorithm A12 can be described as follows.

**Algorithm 1. Algorithm A12**

1: Choose $x_0$ and $y$ such that $y \neq 0$,
2: Choose $\varepsilon$ small and positive, as a tolerance,
3: Set $r_0 = b - Ax_0, y_0 = y, p = Ar_0, p_1 = Ap, c_0 = (y, r_0)$,
4: $c_1 = (y, p), c_2 = (y, p_1), c_3 = (y, Ap_1), \delta = c_1 c_3 - c_2^2, 
\alpha = c_0 c_3 - c_2 c_1, \beta = \frac{c_0 c_2 - c_1^2}{\delta}$,
5: $r_1 = r_0 - c_0 c_1, x_1 = x_0 + c_0 r_0$,
6: $r_2 = r_0 - \alpha p + \beta p_1, x_2 = x_0 + \alpha r_0 - \beta p$,
7: $y_1 = A^T y_0, y_2 = A^T y_1, y_3 = A^T y_2$.
8: for $k = 3, 4, \ldots, n$ do
9: $y_{k+1} = A^T y_k, q_1 = Ar_{k-1}, q_2 = Ap_1, q_3 = Ar_{k-2}$,
10: $a_{i1} = (y_{k-2}, r_{k-2}), a_{i3} = (y_{k-3}, r_{k-3}), a_{i2} = (y_{k-1}, r_{k-2}), a_{i3} = a_{i2}$,
11: $a_{i1} = (y_{k-1}, r_{k-3}), a_{i1} = (y_{k-2}, r_{k-2}), a_{i3} = a_{i2}$,
12: $a_{i3} = (y_{k-1}, r_{k-3})$,
13: $b_1 = -a_{i1} - a_{i2} F_k, b_2 = -a_{i1} - a_{i3} F_k, b_3 = -s - t F_k$,
14: $\Delta_k = a_{i1} a_{i3} (a_{i2} a_{i3} - a_{i2} a_{i2}) + a_{i1} (b_2 a_{i2} - b_3 a_{i2})$,
16: \[ G_i = \frac{b_i - a_i B_i}{a_{i1}}, \quad C_i = \frac{b_i - a_i B_i - a_i C_i}{a_{i2}}, \quad A_i = \frac{1}{C_i + G_i}, \]

17: \[ r_k = A_i(q_{2i} + B_i q_{1i} + C_i r_{k-2} + F_i r_{k-3}), \]

18: \[ x_k = A_i(C_i x_{k-2} + G_i x_{k-3} - (q_1 + B_i r_{k-2} + F_i r_{k-3})), \]

19: if \( \| r_k \| \leq \varepsilon \), then

20: \( x = x_k \), stop.

21: end if

22: end for

4.2. Algorithm A4

Algorithm A4 is based on relation A4 which is already considered by C. Bahuex in her PhD thesis. Its pseudo-code is as follows.

Algorithm 2. Algorithm A4

1: Choose \( x_0 \) and \( y \) such that \( y \neq 0 \),
2: Choose \( \varepsilon \) small and positive as a tolerance,
3: Set \( r_0 = b - A x_0, y_0 = y \),
4: for \( k = 0, 1, \ldots, n \) do
5: \( E_{k+1} = -\frac{(y_k, r_k)}{(y_{k-1}, r_{k-1})} \), for \( k \geq 1 \) and \( E_1 = 0 \)
6: \( B_{k+1} = -\frac{(y_k, Ar_k) - E_{k+1}(y_k, r_{k-1})}{(y_k, r_k)} \)
7: \( A_{k+1} = -\frac{1}{B_{k+1} + E_{k+1}} \)
8: \( x_{k+1} = A_{k+1}\{B_{k+1} x_k + E_{k+1} x_{k-1} - r_k\} \)
9: \( r_{k+1} = A_{k+1}\{Ar_k + B_{k+1} r_k - E_{k+1} r_{k-1}\} \)
10: if \( \| r_{k+1} \| \leq \varepsilon \), then
11: \( y_{k+1} = A^T y_k \)
12: end if
13: end for

4.3. Algorithm A5 / B_{10}

Algorithm A5 / B_{10} is based on relations A5 and B_{10}, first investigated by C. Bahuex in her PhD thesis. Its pseudo-code is as follows.

Algorithm 3. Algorithm A5 / B_{10}

1: Choose \( x_0, y \) and tolerance \( \varepsilon \geq 0 \),
2: Set \( r_0 = b - A x_0, p_0 = r_0, y_0 = y \),
3: \( A_i = -\frac{(y_0, r_0)}{(y_0, Ar_0)} \), \( C_{10} = 1 \)
4: \( r_1 = r_0 + A_i A p_0, x_1 = x_0 - A_i p_0 \)
5: for \( k = 1, 2, 3, \ldots, n \) do
6: \( y_k = A^T y_{k-1} \)
7: \( D_{k+1} = -\frac{(y_k, r_k)}{C_{k-1}^t(y_k, p_{k-1})} \)
8: \( p_k = r_k + D_{k+1} C_{k-1}^t p_{k-1} \)
9: \( A_{k+1} = -\frac{(y_k, r_k)}{(y_k, A p_k)} \)
10: \( r_{k+1} = r_k + A_{k+1} p_k \)
11: \( x_{k+1} = x_k - A_{k+1} p_k \)
12: if \( \| r_{k+1} \| \leq \varepsilon \), then if \( A_k \leq \varepsilon \), then
13: \( C_{k} = C_{k-1} \)
14: end if
15: end for

4.4 Algorithm A8 / B_{10}: The pseudo-code of A8 / B_{10} is as follows.

Algorithm 4. Algorithm A8 / B_{10}

1: Choose \( x_0 \) and \( y \) such that \( y \neq 0 \),
2: Set \( r_0 = b - A x_0 \)
3: \( z_0 = r_0 \)
4: \( y_0 = y \)

5: for \( k = 0, 1, 2, \ldots, n \) do

6: \( A_{k+1} = - \frac{(y_k, r_k)}{(y_k, A z_k)} \),

7: \( r_{k+1} = r_k + A_{k+1} A z_k \),

8: \( x_{k+1} = x_k - A_{k+1} z_k \),

9: if \( \| r_{k+1} \| \neq \varepsilon \) then

10: \( y_{k+1} = A^T y_k \),

11: \( C_{k+1} = \frac{1}{A_{k+1}} \),

12: \( B_{k+1} = - \frac{C_{k+1} (y_{k+1}, r_{k+1})}{(y_{k+1}, A z_k)} \),

13: \( z_{k+1} = B_{k+1} z_k + C_{k+1} r_{k+1} \).

14: end if

15: end for

5.1. Restarting strategies

They follow the same pattern as switching [30], except that here, the alternative algorithm to switch to is the same as the one we started with. This means that, by re-initializing the Lanczos process with another iterate, the last one found starts working with a different Krylov space base. This is enough to fix any numerical difficulties that have occurred and avert any that might occur. Different strategies can be adopted for restarting different algorithms. These are as follows.

1. Restarting after breakdown: Start a particular Lanczos algorithm until a breakdown occurs, then restart the same Lanczos algorithm, initializing it with the last iterate of the failed algorithm. We call this strategy ST1.

2. Pre-emptive restarting: Run a Lanczos-type algorithm for a fixed number of iterations, halt it and then restart it, initializing it with the last iterate. Note that there is no way to guarantee that breakdown would not occur before the end of the interval. This strategy is called ST2.

3. Breakdown monitoring: Provided monotonicity of reduction in the absolute value of denominators involved in the coefficients of the polynomials involved can be established, breakdown can be monitored as follows. Evaluate regularly those coefficients with denominators that are likely to become zero. Restart the algorithm when the absolute value of any of these denominators drops below a certain level. This is strategy ST3.

5.2. A generic restarting algorithm

Suppose we have a set of Lanczos-type algorithms and we want to use one of these algorithms in the restarting framework using one of the above mentioned strategies ST1, ST2 or ST3. Then the following algorithm can be used.

Algorithm 5 Generic restarting algorithm
1: Start the most stable algorithm, if known.

2: Choose a restarting strategy from \{ST1, ST2, ST3\}.

3: if ST1 then

4: Continue with current algorithm until it halts;

5: if solution is obtained then

6: Stop.

7: else

8: restart the same algorithm;
9: initialize it with current iterate,
10: Go to 4.
11: end if
12: else if ST2 then
13: Continue with current algorithm for a fixed number of iterations until it stops;
14: if solution is obtained then
15: Stop.
16: else
17: restart the same algorithm,
18: initialize it with the current iterate,
19: Go to 13.
20: end if
21: end if
22: Continue with current algorithm and monitor certain parameters for breakdown, until it halts
23: if solution is obtained then
24: Stop.
25: else
26: restart the same algorithm,
27: initialize it with current iterate,
28: Go to 22.
29: end if
30: end if

Note that we have only considered strategy ST2 in this paper. Strategy ST1 is not going to be efficient because a breakdown causes the programme to halt. This requires restarting it by hand. Since this occurs rather frequently, particularly in high dimensions, it is not practical to carry out experiments in this case. Strategy ST3 is not implemented since it is clear that it will not be efficient, at least in the sequential environment we are operating in. Looking at algorithms $A_4$, $A_{12}$, $A_{8}/B_{10}$, and $A_{8}/B_{10}$ given earlier, one notices easily that they involve identifying the coefficients of FOP’s used in the Lanczos process.

Many of these coefficients are ratios. It is the denominator quantities of these ratios that have to be monitored in ST2. There may be a lot of them and there is no obvious way to see which one is decreasing alarmingly and threatening to cause breakdown. There is a serious amount of work yet to be done to understand how these coefficients behave in order to find which ones should be monitored closely. It is also reasonable to assume that, in a parallel environment many can be monitored simultaneously, not in a sequential environment. For these reasons, we have only investigated strategy ST2. In the following, a generic algorithm implementing ST2 is given. Note that a convergence tolerance $\epsilon = 1.0 \times 10^{-3}$ and 20 iterations per cycle are used in all experiments.

5.3. Implementing ST2

ST2 takes as input a given algorithm from a pre-specified list. Here, these algorithms are the ones already listed above, i.e. $A_4$, $A_{12}$, $A_{8}/B_{10}$, and $A_{8}/B_{10}$. Depending on whether the algorithms are of the $A_i$-type (i.e. Lanczos-type algorithm based on a single recurrence relation) or $A_i/B_j$-type (i.e. Lanczos-type algorithm based on two recurrence relations), initialisation has to be done differently; $A_i$-type requires $x_0$, $y_0 = y$, and $A_i/B_j$-type requires $x_0$, $r_0 = b - Ax_0$ and $y_0 = y$ as well as $z_0 = r_0$. The general ST2 algorithm can be described, therefore, as follows.

5.3.1. Algorithm ST2

**Algorithm 6 ST2 restarting algorithm**

1: Choose algorithm $ALG$ from $\{A_4, A_{12}, A_{8}/B_{10}, A_{8}/B_{10}\}$
2: Choose $x_0$ and $y$ such that $y \neq 0$,
3: set $r_0 = b - Ax_0$, $y_0 = y$,
4: if $ALG \neq A_4$ and $ALG \neq A_{12}$ then
5: $z_0 = r_0$;
6: end if
7: run $ALG$ for a fixed number of iterations (a cycle) or until it halts;
8: if solution is obtained then
9: stop;
10: else
11: initialize it with the current iterate;
12: go to 3;
13: end if

5.3.2. Numerical results

Algorithms 1, 2, 3, 4, and algorithm ST2 restarting each one of them [17] has been implemented in Matlab and applied to a number of small to medium size problems for different values of \( \delta \). The test problems we have used arise in the 5-point discretisation of the operator

\[
-\frac{\partial^2}{\partial x^2} - \frac{\partial^2}{\partial y^2} + \gamma \frac{\partial}{\partial x}
\]

on a rectangular region. We refer to them as the Baheux-type problems since we borrowed them from her PhD thesis. These problems are important because they represent the typical SLE’s that crop up in the real world. They are also used by Baheux herself to test three of the algorithms used here. Comparative results are carried out on instances of size ranging from \( n = 20 \) to \( n = 4000 \) of the problem \( Ax = b \) with \( A \) and \( b \) as follows

\[
A = \begin{pmatrix}
B & -I & \ldots & 0 \\
-I & B & : \\
& & \ddots & \ddots \\
& & B & -I \\
0 & \ldots & \ldots & 0
\end{pmatrix},
\]

with

\[
B = \begin{pmatrix}
4 & \alpha & \ldots & \ldots & 0 \\
\beta & 4 & \alpha & : \\
& \ddots & \ddots \\
& \beta & 4 & \alpha \\
0 & \ldots & \ldots & \beta & 4
\end{pmatrix}
\]

and \( \alpha = -1 + \delta \), \( \beta = -1 - \delta \). The parameter \( \delta \) takes values 0.0, 0.2, 5 and 8 respectively. The right hand side \( b \) is taken to be \( b = AX \), where \( X = (1,1,...,1)^T \) the solution of the system is. The dimension of \( B \) is 10. When \( \delta = 0 \), the coefficient matrix \( A \) is symmetric and the problem is easy to solve because the region is a regular mesh [31]. For all other values of \( \delta \), the matrix \( A \) is non-symmetric and the problem is comparatively hard to solve as the region is not a regular mesh.

The results obtained with algorithms \( A_4, A_{12}, A_5 / B_{10} \) and \( A_6 / B_{10} \) executed with the ST2 restarting each one of them, for different values of \( \delta \) on Baheux type problems, are recorded in Tables 1, 2, 3 and 4 below. The results show that the restarting algorithm ST2 is far superior to any one of the algorithms considered individually.

5.3.3. Comments on the numerical evidence

We have implemented \( A_4, A_{12}, A_5 / B_{10} \) and \( A_6 / B_{10} \) to solve a number of problems of the type described in Section 5.3.2 with dimensions ranging from 20 to 4000. The results are compared against those obtained by the restarting strategy ST2 used on each one of the algorithms \( A_4, A_{12}, A_5 / B_{10} \) and \( A_6 / B_{10} \), on the same problems. They show that algorithms \( A_4, A_{12}, A_5 / B_{10} \) and \( A_6 / B_{10} \) are not as robust as when they are restarted in a preemptive fashion (under ST2). Outside the restarting framework, the algorithms failed to solve any of the problems for dimensions greater than 40. Within the restarting framework, all problems have been solved with high precision. This is recorded in Table 1 through Table 4. The results, undoubtedly point to restarting as another way of curing breakdown in Lanczos-type algorithms.
### Table 1. Numerical results for $\delta = 0$

<table>
<thead>
<tr>
<th>Dim of Prob</th>
<th>$ST_2(A_2)$</th>
<th>$ST_2(A_{12})$</th>
<th>$ST_2(A_{2}/B_{10})$</th>
<th>$ST_2(A_{2}/B_{10})$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n$</td>
<td>$r_1$</td>
<td>$T(s)$</td>
<td>$r_1$</td>
<td>$T(s)$</td>
</tr>
<tr>
<td>20</td>
<td>3.8545e-04</td>
<td>0.0010</td>
<td>8.4317e-04</td>
<td>0.0075</td>
</tr>
<tr>
<td>40</td>
<td>6.0376e-04</td>
<td>0.0043</td>
<td>7.7873e-04</td>
<td>0.0126</td>
</tr>
<tr>
<td>60</td>
<td>8.0401e-04</td>
<td>0.0125</td>
<td>3.8138e-04</td>
<td>0.0161</td>
</tr>
<tr>
<td>80</td>
<td>9.8201e-04</td>
<td>0.0140</td>
<td>6.1219e-04</td>
<td>0.0205</td>
</tr>
<tr>
<td>100</td>
<td>2.6273e-04</td>
<td>0.0193</td>
<td>9.4670e-04</td>
<td>0.0216</td>
</tr>
<tr>
<td>200</td>
<td>7.5105e-04</td>
<td>0.0803</td>
<td>6.1319e-04</td>
<td>0.0591</td>
</tr>
<tr>
<td>400</td>
<td>8.2317e-04</td>
<td>0.5170</td>
<td>9.0236e-04</td>
<td>0.4105</td>
</tr>
<tr>
<td>600</td>
<td>8.9464e-04</td>
<td>0.9418</td>
<td>9.6683e-04</td>
<td>2.0031</td>
</tr>
<tr>
<td>800</td>
<td>8.5604e-04</td>
<td>1.5466</td>
<td>8.8581e-04</td>
<td>3.2419</td>
</tr>
<tr>
<td>1000</td>
<td>8.7519e-04</td>
<td>2.4005</td>
<td>8.4853e-04</td>
<td>4.7680</td>
</tr>
<tr>
<td>4000</td>
<td>9.2511e-04</td>
<td>48.0754</td>
<td>6.0811e-04</td>
<td>113.4374</td>
</tr>
</tbody>
</table>

### Table 2. Numerical results for $\delta = 0.2$

<table>
<thead>
<tr>
<th>Dim of Prob</th>
<th>$ST_2(A_2)$</th>
<th>$ST_2(A_{12})$</th>
<th>$ST_2(A_{2}/B_{10})$</th>
<th>$ST_2(A_{2}/B_{10})$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n$</td>
<td>$r_1$</td>
<td>$T(s)$</td>
<td>$r_1$</td>
<td>$T(s)$</td>
</tr>
<tr>
<td>20</td>
<td>9.9896e-04</td>
<td>0.0064</td>
<td>8.9978e-04</td>
<td>0.0130</td>
</tr>
<tr>
<td>40</td>
<td>4.9256e-04</td>
<td>0.0191</td>
<td>5.1552e-04</td>
<td>0.0292</td>
</tr>
<tr>
<td>60</td>
<td>6.9057e-04</td>
<td>0.0210</td>
<td>5.5319e-04</td>
<td>0.0328</td>
</tr>
<tr>
<td>80</td>
<td>8.7129e-04</td>
<td>0.0250</td>
<td>9.2631e-04</td>
<td>0.0388</td>
</tr>
<tr>
<td>100</td>
<td>5.3007e-04</td>
<td>0.0305</td>
<td>9.4369e-04</td>
<td>0.0440</td>
</tr>
<tr>
<td>200</td>
<td>2.2236e-04</td>
<td>0.0980</td>
<td>9.9659e-04</td>
<td>0.1485</td>
</tr>
<tr>
<td>400</td>
<td>9.0428e-04</td>
<td>0.4163</td>
<td>8.5500e-04</td>
<td>0.3117</td>
</tr>
<tr>
<td>600</td>
<td>6.6026e-04</td>
<td>1.0532</td>
<td>9.2957e-04</td>
<td>1.1802</td>
</tr>
<tr>
<td>800</td>
<td>8.6281e-04</td>
<td>1.5170</td>
<td>4.8311e-04</td>
<td>1.8420</td>
</tr>
<tr>
<td>1000</td>
<td>9.0165e-04</td>
<td>2.1245</td>
<td>8.1326e-04</td>
<td>3.3795</td>
</tr>
<tr>
<td>4000</td>
<td>8.9239e-04</td>
<td>36.0361</td>
<td>9.8884e-04</td>
<td>74.3756</td>
</tr>
</tbody>
</table>

### Table 3. Numerical results for $\delta = 5$

<table>
<thead>
<tr>
<th>Dim of Prob</th>
<th>$ST_2(A_2)$</th>
<th>$ST_2(A_{12})$</th>
<th>$ST_2(A_{2}/B_{10})$</th>
<th>$ST_2(A_{2}/B_{10})$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N$</td>
<td>$r_1$</td>
<td>$T(s)$</td>
<td>$r_1$</td>
<td>$T(s)$</td>
</tr>
<tr>
<td>20</td>
<td>1.2543e-04</td>
<td>0.0110</td>
<td>2.1773e-04</td>
<td>0.0168</td>
</tr>
<tr>
<td>40</td>
<td>6.7523e-04</td>
<td>0.0442</td>
<td>4.5708e-04</td>
<td>0.0691</td>
</tr>
<tr>
<td>60</td>
<td>2.9307e-04</td>
<td>0.0524</td>
<td>9.5280e-04</td>
<td>0.1127</td>
</tr>
<tr>
<td>80</td>
<td>6.9125e-04</td>
<td>0.0634</td>
<td>4.0822e-04</td>
<td>0.0909</td>
</tr>
<tr>
<td>100</td>
<td>6.9001e-04</td>
<td>0.0707</td>
<td>6.7233e-04</td>
<td>0.1195</td>
</tr>
<tr>
<td>200</td>
<td>5.4021e-04</td>
<td>0.1661</td>
<td>7.2644e-04</td>
<td>0.1957</td>
</tr>
<tr>
<td>400</td>
<td>6.9291e-04</td>
<td>0.4889</td>
<td>9.8208e-04</td>
<td>1.0148</td>
</tr>
<tr>
<td>600</td>
<td>2.6118e-04</td>
<td>0.9507</td>
<td>9.4438e-04</td>
<td>1.2438</td>
</tr>
<tr>
<td>800</td>
<td>4.5923e-04</td>
<td>1.9086</td>
<td>8.6243e-04</td>
<td>1.9261</td>
</tr>
<tr>
<td>1000</td>
<td>4.5632e-04</td>
<td>3.0028</td>
<td>9.1161e-04</td>
<td>5.6501</td>
</tr>
<tr>
<td>3000</td>
<td>7.1675e-04</td>
<td>37.2835</td>
<td>7.1249e-04</td>
<td>66.2288</td>
</tr>
<tr>
<td>4000</td>
<td>7.9284e-04</td>
<td>60.0409</td>
<td>6.0671e-04</td>
<td>118.7937</td>
</tr>
</tbody>
</table>
Table 4. Numerical results for $\delta = 8$

<table>
<thead>
<tr>
<th>Dim of Prob</th>
<th>ST2($A_4$)</th>
<th>ST2($A_{12}$)</th>
<th>ST2($A_2/B_{10}$)</th>
<th>ST2($A_2/B_{10}$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>n</td>
<td></td>
<td></td>
<td>$r_1$</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>1.217e-01</td>
<td>0.0116</td>
<td>9.972e-01</td>
<td>0.0162</td>
</tr>
<tr>
<td>40</td>
<td>9.161e-01</td>
<td>0.0403</td>
<td>8.814e-01</td>
<td>0.0277</td>
</tr>
<tr>
<td>60</td>
<td>6.218e-01</td>
<td>0.0782</td>
<td>8.220e-01</td>
<td>0.0928</td>
</tr>
<tr>
<td>80</td>
<td>9.577e-01</td>
<td>0.0750</td>
<td>6.825e-01</td>
<td>0.0893</td>
</tr>
<tr>
<td>100</td>
<td>8.971e-01</td>
<td>0.0720</td>
<td>9.260e-01</td>
<td>0.1520</td>
</tr>
<tr>
<td>200</td>
<td>9.792e-01</td>
<td>0.1528</td>
<td>5.784e-01</td>
<td>0.2396</td>
</tr>
<tr>
<td>400</td>
<td>6.309e-01</td>
<td>0.3592</td>
<td>9.000e-01</td>
<td>0.5326</td>
</tr>
<tr>
<td>3000</td>
<td>7.802e-01</td>
<td>42.6996</td>
<td>7.528e-01</td>
<td>73.6804</td>
</tr>
</tbody>
</table>

6. Conclusion

The restarting strategy ST2 considered seems to be very successful at remedying and avoiding breakdown in Lanczos-type algorithms. The supporting numerical evidence is very strong in this respect. Indeed, restarting solved all problems up to dimension of 4000 while individual algorithms only managed to solve them for lower dimensions ($\leq 40$). The cost of preemptive restarting is not substantial. In the case of monitoring the coefficients that can vanish, although the cost is only that of a test of the form $|f|$ [denominator value] $\leq$ tolerance then stop, many such tests may be carried out. We have not measured its impact on the overall computing time. Positive experimental results point to restarting as a significant approach to avoiding breakdown in solving SLE’s by Lanczos-type algorithms. This idea is not only different from existing strategies for dealing with breakdown, [32-34], but also quite easy to understand and follow. However, it remains for some more extensive testing to be done on both large real and randomly generated problems to fully understand the behavior and cost of the restarting approach compared to state-of-the-art Lanczos-type algorithms with in-built precautions to avoid breakdown such as MRZ and BSMRZ. Further work is being expanded in this direction.


References


