*Journal of Prime Research in Mathematics Vol.* **10***(2015), 104-119.*

# A NEW LANCZOS-TYPE ALGORITHM FOR SYSTEMS OF LINEAR EQUATIONS

MUHAMMAD FAROOQ[1] AND ABDELLAH SALHI[2]

Abstract. Lanczos-type algorithms are efficient and easy to implement. Unfortunately they breakdown frequently and well before convergence has been achieved. These algorithms are typically based on recurrence relations which involve formal orthogonal polynomials of low degree. In this paper, we consider a recurrence relation that has not been studied before and which involves a relatively higher degree polynomial. Interestingly, it leads to an algorithm that shows superior stability when compared to existing Lanczos-type algorithms. This new algorithm is derived and described. It is then compared to the best known algorithms of this type, namely $A_5/B_{10}$, $A_8/B_{10}$, as well as Arnoldi's algorithm, on a set of standard test problems. Numerical results are included.

*Key words* : Lanczos algorithm; Arnoldi algorithm; Systems of Linear Equations; Formal Orthogonal Polynomials

*AMS SUBJECT* : Primary 65F10.

## 1. Introduction and background

The Lanczos algorithm, [28, 29, 14], is an iterative process that has been primarily designed to calculate the eigenvalues of a matrix. However, it has found a wide application in the area of Systems of Linear Equations (SLE's) where it now is a well established solver. Its attraction resides in its efficiency as it only involves vector-to-vector and matrix-to-vector products. Moreover, in exact arithmetic, it converges to the exact solution in at most $n$ steps, where $n$ is the dimension

[1]Department of Mathematics, University of Peshawar, Khyber Pakhtunkhwa, 25120, Pakistan. Email: mfarooq@upesh.edu.pk

[2]Department of Mathematical Sciences, University of Essex, Wivenhoe Park, Colchester, CO4 3SQ, UK. E-mail: as@essex.ac.uk.

of the problem, [29]. While efficiency is its strong point, stability is not. Indeed, it is well known to breakdown as orthogonality of the so called Lanczos vectors, generated during the solution process, is lost. Efforts to avoid this breakdown led to a flurry of papers particularly from Brezinski and his team, [2, 5, 6, 7, 10, 12, 11, 13], and others, [4, 8, 15, 17, 20, 21, 23, 24, 31, 32, 33, 35, 38, 39].

Several Lanczos-type algorithms have been designed and among them, the famous conjugate gradient algorithm of Hestenes and Stiefel, [25], when the matrix is Hermitian and the bi-conjugate gradient algorithm of Fletcher, [22], and the algorithm of Arnoldi, [1, 36], in the general case.

Lanczos-type algorithms are commonly derived from recurrence relations typically using Formal Orthogonal Polynomials (FOP's) of low degree, [29, 6, 16, 37]. Recurrence relations using relatively higher degree FOP's have not been investigated. Here, we set out to design an algorithm that is based on such recurrence relations and FOP's, and study its properties and, in particular, its stability.

1.1. **The Lanczos Process.** Consider the SLE

$$A\mathbf{x} = \mathbf{b}, \tag{1}$$

where $A \in R^{n\times n}$, $\mathbf{b} \in R^n$ and $\mathbf{x} \in R^n$.

Let $\mathbf{x}_0$ and $\mathbf{y}$ be two arbitrary vectors in $R^n$ such that $\mathbf{y} \neq 0$ then Lanczos method [29] consists in constructing a sequence of vectors $\mathbf{x}_k \in R^n$ defined as follows

$$\mathbf{x}_k - \mathbf{x}_0 \in K_k(A, \mathbf{r}_0) = span(\mathbf{r}_0, A\mathbf{r}_0, \ldots, A^{k-1}\mathbf{r}_0), \tag{2}$$

$$\mathbf{r}_k = \mathbf{b} - A\mathbf{x}_k \perp L_k(A^T, \mathbf{y}) = span(\mathbf{y}, A^T\mathbf{y}, \ldots, A^{T^{k-1}}\mathbf{y}), \tag{3}$$

where $A^T$ denotes the transpose of $A$.

Equation (2) gives,

$$\mathbf{x}_k - \mathbf{x}_0 = -\alpha_1\mathbf{r}_0 - \alpha_2 A\mathbf{r}_0 - \cdots - \alpha_k A^{k-1}\mathbf{r}_0. \tag{4}$$

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

$$\mathbf{r}_k = \mathbf{r}_0 + \alpha_1\mathbf{r}_0 + \alpha_2 A\mathbf{r}_0 + \cdots + \alpha_k A^{k-1}\mathbf{r}_0. \tag{5}$$

If we set

$$P_k(x) = 1 + \alpha_1 x + ... + \alpha_k x,$$

then we can write from (5)

$$\mathbf{r}_k = P_k(A)\mathbf{r}_0. \tag{6}$$

From (3), the orthogonality condition gives

$$(A^{T^i}\mathbf{y}, \mathbf{r}_k) = (\mathbf{y}, A^i\mathbf{r}_k) = (\mathbf{y}, A^i\mathbf{P}_k(A)\mathbf{r}_0) = 0, \text{ for } i = 0, ..., k-1.$$

Thus, the coefficients $\alpha_1$,...,$\alpha_k$ form a solution of SLE's,

$$\alpha_1(\mathbf{y}, A^{i+1}\mathbf{r}_0) + ... + \alpha_k(\mathbf{y}, A^{i+k}\mathbf{r}_0) = -(\mathbf{y}, A^i\mathbf{r}_0), \text{ for } i = 0, \ldots, k-1. \quad (7)$$

If the determinant of the above system is not zero then its solution exists and allows to obtain $\mathbf{x}_k$ and $\mathbf{r}_k$. Obviously, in practice, solving the above system directly for increasing values of $k$ is not feasible; $k$ is the order of the iterate in the solution process. We shall see now how to solve this system for increasing values of $k$ recursively, that is, if polynomials $P_k$ can be computed recursively. Such computation is feasible, since polynomials $P_k$ form a family of FOP's which will briefly be explained below.

1.2. **Formal Orthogonal Polynomials.** Define a linear functional $c$ on the space of reel polynomials by

$$c(x^i) = c_i \quad \text{for} \quad i = 0, 1, \ldots$$

where

$$c_i = (A^{T^i}\mathbf{y}, \mathbf{r}_k) = (\mathbf{y}, A^i\mathbf{r}_k) \quad \text{for} \quad i = 0, 1, \ldots$$

Write the orthogonality condition as,

$$c(x^i P_k) = 0 \text{ for } i = 0, \ldots, k-1. \quad (8)$$

The above condition shows that $P_k$ is the polynomial of degree at most $k$ and is a FOP with respect to the functional $c$, [5]. The normalization condition for these polynomials is $P_k(0) = 1$; $P_k$ exists and is unique if the following Hankel determinant

$$H_k^{(1)} = \begin{vmatrix} c_1 & c_2 & \cdots & c_k \\ c_2 & c_3 & \cdots & c_{k+1} \\ \vdots & \vdots & & \vdots \\ c_k & c_{k+1} & \cdots & c_{2k-1} \end{vmatrix}$$

is not zero. In that case we can write $P_k(x)$ as follows.

$$P_k(x) = \frac{\begin{vmatrix} 1 & x & \cdots & x^k \\ c_0 & c_1 & \cdots & c_k \\ \vdots & \vdots & & \vdots \\ c_{k-1} & c_k & \cdots & c_{2k-1} \end{vmatrix}}{\begin{vmatrix} c_1 & \cdots & c_k \\ \vdots & & \vdots \\ c_k & \cdots & c_{2k-1} \end{vmatrix}}, \tag{9}$$

where the denominator of this polynomial is $H_k^{(1)}$, the determinant of the system (7). We assume that $\forall\ k,\ H_k^{(1)} \neq 0$ and therefore all the polynomials $P_k$ exist for all $k$. If for some $k$, $H_k^{(1)} = 0$, then $P_k$ does not exist and breakdown occurs in the algorithm, [6, 10, 11, 13, 8].

A Lanczos-type method consists in computing $P_k$ recursively, then $\mathbf{r}_k$ and finally $\mathbf{x}_k$ such that $\mathbf{r}_k = \mathbf{b} - A\mathbf{x}_k$, without inverting matrix $A$. In exact arithmetic, this gives the solution of the system (1) in at most $n$ steps, where $n$ is the dimension of the SLE, [6, 12].

1.3. **Notation and organization.** The notation introduced by Baheux, in [2, 3], for recurrence relations with three terms is adopted here. It puts recurrence relations involving FOP's $P_k(x)$ (the polynomials of degree at most $k$ with regard to the linear functional $c$) and/or FOP's $P_k^{(1)}(x)$ (the polynomials of degree at most $k$ with regard to linear functional $c^{(1)}$, [9]) into two groups: $A_i$ and $B_j$. Although relations $A_i$, when they exist, rarely lead to Lanczos-type algorithms on their own (the exceptions being $A_4$, [2, 3], and $A_{12}$, [17], so far), relations $B_j$ never lead to such algorithms for obvious reasons. It is the combination of recurrence relations $A_i$ and $B_j$, denoted as $A_i/B_j$, when both exist, that lead to Lanczos-type algorithms. In the following we will refer to algorithms by the relation(s) that lead to them. Hence, there are, potentially, algorithms $A_i$ and algorithms $A_i/B_j$, for some $i = 1, 2, \ldots$ and some $j = 1, 2, \ldots$

In this paper, a new algorithm based on a recurrence relation that has not been studied before, is derived. It is then compared to three other algorithms, one of which is the Arnoldi algorithm.

The rest of the paper is organized as follows. In the next section, the Lanczos-type algorithm $A_8/B_{10}$, [2], and the estimation of the coefficients of the recurrence relations $A_8$ and $B_{10}$ used to derive it are given. Section 3 is on the estimation of the coefficients of recurrence relation

$A_{12}$, [17], used to derive the new algorithm of the same name. Section 4 describes the test problems and reports numerical results. Section 5 is the conclusion and further work.

## 2. Baheux algorithm $A_8/B_{10}$

The choice of algorithm $A_8/B_{10}$, for comparison with our own is dictated by the fact that this is the most robust of the algorithms considered in [2, 3] on some of the problems considered here. So, outperforming this algorithm implies outperforming the rest of the algorithms considered therein.

For completeness, we recall the relevant relations between adjacent FOPs that lead to $A_8/B_{10}$ and their coefficients estimates. These are $A_8$ and $B_{10}$. The details of algorithm $A_5/B_{10}$ are given in [3].

### 2.1. Recurrence relation $A_8$.

Relation $A_8$ is

$$P_k(x) = (A_k x + B_k)P_{k-1}^{(1)}(x) + (C_k x + D_k)P_{k-1}(x), \tag{10}$$

first investigated in [2, 3]. Its coefficients are estimated as

$$B_k = 0, \tag{11}$$

$$C_k = 0, \tag{12}$$

$$D_k = 1, \tag{13}$$

and

$$A_k = -\frac{c(x^{k-1}P_{k-1}(x))}{c(x^k P_{k-1}^{(1)}(x))}. \tag{14}$$

As we know

$$\begin{cases} c(x^k P_k) = (A^{T^k}\mathbf{y}, P_k(A)\mathbf{r}_0) = (\mathbf{y}_k, \mathbf{r}_k), \\ c(x^k P_k^{(1)}) = (A^{T^k}\mathbf{y}, P_k^{(1)}(A)\mathbf{r}_0) = (\mathbf{y}_k, \mathbf{z}_k), \end{cases} \tag{15}$$

with $\mathbf{y}_k = A^T\mathbf{y}_{k-1}$ and $\mathbf{z}_k$ is defined in (21). Using (15), equation (14) becomes

$$A_k = -\frac{(\mathbf{y}_{k-1}, \mathbf{r}_{k-1})}{(\mathbf{y}_k, \mathbf{z}_{k-1})} = -\frac{(\mathbf{y}_{k-1}, \mathbf{r}_{k-1})}{(\mathbf{y}_{k-1}, A\mathbf{z}_{k-1})}. \tag{16}$$

2.2. **Recurrence relation $B_{10}$.** This relation, first investigated in [2, 3], is

$$P_k^{(1)}(x) = (A_k^1 x + B_k^1)P_{k-1}^{(1)}(x) + C_k^1 P_k(x), \tag{17}$$

Its coefficients are estimated as

$$A_k^1 = 0, \tag{18}$$

$$C_k^1 a_k = 1, \tag{19}$$

where $a_k$ is the coefficient of $x^k$ in $P_k(x)$ defined in (10) and

$$B_k^1 = -\frac{C_k^1 c(\mathbf{y}_k, \mathbf{r}_k)}{c(\mathbf{y}_k, \mathbf{z}_{k-1})}. \tag{20}$$

Equation (17) gives

$$\mathbf{z}_k = B_k^1 \mathbf{z}_{k-1} + C_k^1 \mathbf{r}_k. \tag{21}$$

2.3. **Algorithm $A_8/B_{10}$.** The pseudo-code of $A_8/B_{10}$, due to Baheux, [2, 3], is as follows.

**Algorithm 1** Algorithm $A_8/B_{10}$

Choose $\mathbf{x}_0$ and $\mathbf{y}$ such that $\mathbf{y} \neq 0$, and $\epsilon$ arbitrarily small and positive.
Set $\mathbf{r}_0 = \mathbf{b} - A\mathbf{x}_0$,
$\mathbf{z}_0 = \mathbf{r}_0$,
$\mathbf{y}_0 = \mathbf{y}$,
**for** $k = 0, 1, 2, \ldots,$ **do**
  $A_{k+1} = -\frac{(\mathbf{y}_k, \mathbf{r}_k)}{(\mathbf{y}_k, A\mathbf{z}_k)}$,
  $\mathbf{r}_{k+1} = \mathbf{r}_k + A_{k+1} A\mathbf{z}_k$,
  $\mathbf{x}_{k+1} = \mathbf{x}_k - A_{k+1}\mathbf{z}_k$.
  **if** $||\mathbf{r}_{k+1}|| \geq \epsilon$, **then**
    $\mathbf{y}_{k+1} = A^T \mathbf{y}_k$,
    $C_{k+1}^1 = \frac{1}{A_{k+1}}$,
    $B_{k+1}^1 = -\frac{C_{k+1}^1 (\mathbf{y}_{k+1}, \mathbf{r}_{k+1})}{(\mathbf{y}_k, A\mathbf{z}_k)}$,
    $\mathbf{z}_{k+1} = B_{k+1}^1 \mathbf{z}_k + C_{k+1}^1 \mathbf{r}_{k+1}$.
  **else**
    Stop; solution found.
  **end if**
**end for**

## 3. A new Lanczos-type algorithm

In the following, a new recurrence relation which leads to a new variant of the Lanczos algorithm is considered.

3.1. **Recurrence relation $A_{12}$.** Consider the following recurrence relation, [17, 19]

$$P_k(x) = A_k[(x^2 + B_k x + C_k)P_{k-2}(x) + (D_k x^3 + E_k x^2 + F_k x + G_k)P_{k-3}(x)], \tag{22}$$

for $k \geq 3$, where $A_k$, $B_k$, $C_k$, $D_k$, $E_k$, $F_k$ and $G_k$ are constants to be determined using the normalization condition $P_k(0) = 1$ and the orthogonality conditions $c(x^i P_k) = 0$, $\forall i = 0, \ldots, k-1$, $x^i$ being a monic polynomial of exact degree $i$. To find these coefficients, we proceed as follows. Since $\forall k$, $P_k(0) = 1$, equation (22) gives

$$1 = A_k[C_k + G_k].$$

Multiplying both sides of (22) by $x^i$ and then applying the linear functional $c$, we get

$$\begin{aligned} c(x^i P_k) = A_k\{&c(x^{i+2}P_{k-2}) + B_k c(x^{i+1}P_{k-2}) + C_k c(x^i P_{k-2}) + D_k c(x^{i+3}P_{k-3}) \\ &+E_k c(x^{i+2}P_{k-3}) + F_k c(x^{i+1}P_{k-3}) + G_k c(x^i P_{k-3})\}. \end{aligned} \tag{23}$$

Equation (23) is always true for $i = 0, ..., k-7$.
For $i = k-6$, we have

$$0 = D_k c(x^{k-3}P_{k-3}).$$

Since $c(x^{k-3}P_{k-3}) \neq 0$, we have

$$D_k = 0.$$

For $i = k-5$, we get

$$0 = E_k c(x^{k-3}P_{k-3}).$$

But $c(x^{k-3}P_{k-3}) \neq 0$; therefore

$$E_k = 0.$$

For $i = k-4$, (23) gives

$$F_k = -\frac{c(x^{k-2}P_{k-2})}{c(x^{k-3}P_{k-3})}. \tag{24}$$

For $i = k-3$, $i = k-2$ and $i = k-1$ we get the following equations respectively

$$B_k c(x^{k-2}P_{k-2}) + G_k c(x^{k-3}P_{k-3}) = -c(x^{k-1}P_{k-2}) - F_k c(x^{k-2}P_{k-3}), \quad (25)$$

$$\begin{aligned} &B_k c(x^{k-1}P_{k-2}) + C_k c(x^{k-2}P_{k-2}) + G_k c(x^{k-2}P_{k-3}) \\ &= -c(x^k P_{k-2}) - F_k c(x^{k-1}P_{k-3}), \end{aligned} \quad (26)$$

and

$$\begin{aligned} &B_k c(x^k P_{k-2}) + C_k c(x^{k-1}P_{k-2}) + G_k c(x^{k-1}P_{k-3}) \\ &= -c(x^{k+1}P_{k-2}) - F_k c(x^k P_{k-3}). \end{aligned} \quad (27)$$

Let $a_{11}$, $a_{12}$, $a_{13}$, $a_{21}$, $a_{22}$, $a_{23}$, $a_{31}$, $a_{32}$, and $a_{33}$ be the coefficients of $B_k$, $C_k$ and $G_k$ in equations (25), (26) and (27) respectively and let $b_1$, $b_2$ and $b_3$ be the corresponding right sides of these equations. If $\Delta_k$ represents the determinant of the coefficients matrix of the above mentioned system of equations then, we have

$a_{11} = c(x^{k-2}P_{k-2})$, $a_{12} = 0$, $a_{13} = c(x^{k-3}P_{k-3})$,

$a_{21} = c(x^{k-1}P_{k-2})$, $a_{22} = c(x^{k-2}P_{k-2})$, $a_{23} = c(x^{k-2}P_{k-3})$,

$a_{31} = c(x^k P_{k-2})$, $a_{32} = c(x^{k-1}P_{k-2})$, $a_{33} = c(x^{k-1}P_{k-3})$,

$b_1 = -c(x^{k-1}P_{k-2}) - F_k c(x^{k-2}P_{k-3}) = -a_{21} - F_k a_{23}$,

$b_2 = -c(x^k P_{k-2}) - F_k c(x^{k-1}P_{k-3}) = -a_{31} - F_k a_{33}$,

$b_3 = -c(x^{k+1}P_{k-2}) - F_k c(x^k P_{k-3}) = -s - F_k t$, where $s = c(x^{k+1}P_{k-2})$ and $t = c(x^k P_{k-3})$.

Therefore, equations (25), (26) and (27) can be written as

$$a_{11}B_k + 0C_k + a_{13}G_k = b_1, \quad (28)$$

$$a_{21}B_k + a_{22}C_k + a_{23}G_k = b_2, \quad (29)$$

$$a_{31}B_k + a_{32}C_k + a_{33}G_k = b_3. \quad (30)$$

To solve for $B_k$, $C_k$ and $G_k$, Cramer's rule requires

$$\Delta_k = a_{11}(a_{22}a_{33} - a_{32}a_{23}) + a_{13}(a_{21}a_{32} - a_{31}a_{22}).$$

If $\Delta_k \neq 0$, then

$$B_k = \frac{b_1(a_{22}a_{33} - a_{32}a_{23}) + a_{13}(b_2 a_{32} - b_3 a_{22})}{\Delta_k}, \quad (31)$$

$$G_k = \frac{b_1 - a_{11}B_k}{a_{13}}, \quad (32)$$

$$C_k = \frac{b_2 - a_{21}B_k - a_{23}G_k}{a_{22}}, \quad (33)$$

and

$$1 = A_k[C_k + G_k]. \tag{34}$$

With all the necessary coefficients now determined, the expression of the polynomials $P_k(x)$ becomes

$$P_k(x) = A_k\{(x^2 + B_k x + C_k)P_{k-2}(x) + (F_k x + G_k)P_{k-3}(x)\}. \tag{35}$$

Let us now use the relation (35) to compute $P_k(x)$, necessary for the computation of the residual $\mathbf{r}_k = \mathbf{b} - A\mathbf{x}_k = P_k(A)\mathbf{r}_0$ and the corresponding vector $\mathbf{x}_k$.

Assume that $P_k$ has exact degree $k$ and the 3-term recurrence relationship (35) holds. To move to the Krylov space, replace $x$ by $A$ and multiply both side of (35) by $\mathbf{r}_0$ to get,

$$P_k(A)\mathbf{r}_0 = A_k[(A^2+B_kA+C_kI)P_{k-2}(A)\mathbf{r}_0+(F_kA+G_kI)P_{k-3}(A)\mathbf{r}_0]. \tag{36}$$

Using equation (6), gives

$$\mathbf{r}_k = A_k\{(A^2 + B_kA + C_kI)\mathbf{r}_{k-2} + (F_kA + G_kI)\mathbf{r}_{k-3}\}. \tag{37}$$

And using $\mathbf{r}_k = \mathbf{b} - A\mathbf{x}_k$, gives

$$\mathbf{x}_k = A_k\{C_k\mathbf{x}_{k-2} + G_k\mathbf{x}_{k-3} - (A\mathbf{r}_{k-2} + B_k\mathbf{r}_{k-2} + F_k\mathbf{r}_{k-3})\}, \tag{38}$$

with $F_k$ as in equation (24). Using (15), $F_k$ can be written as

$$F_k = -\frac{(\mathbf{y}_{k-2}, \mathbf{r}_{k-2})}{(\mathbf{y}_{k-3}, \mathbf{r}_{k-3})}.$$

Condition (15) can be used equally to rewrite the expressions of $a_{11}$, through $a_{33}$, $b_1$ to $b_3$ as follows.

$a_{11} = (\mathbf{y}_{k-2}, \mathbf{r}_{k-2})$, $a_{12} = 0$, $a_{13} = (\mathbf{y}_{k-3}, \mathbf{r}_{k-3})$,

$a_{21} = (\mathbf{y}_{k-1}, \mathbf{r}_{k-2})$, $a_{22} = (\mathbf{y}_{k-2}, \mathbf{r}_{k-2})$, $a_{23} = (\mathbf{y}_{k-2}, \mathbf{r}_{k-3})$,

$a_{31} = (\mathbf{y}_k, \mathbf{r}_{k-2})$, $a_{32} = (\mathbf{y}_{k-1}, \mathbf{r}_{k-2})$, $a_{33} = (\mathbf{y}_{k-1}, \mathbf{r}_{k-3})$,

$b_1 = -a_{21} - F_k a_{23}$, $b_2 = -a_{31} - F_k a_{33}$, $b_3 = -s - F_k t$,
where $s = (\mathbf{y}_{k+1}, \mathbf{r}_{k-2})$ and $t = (\mathbf{y}_k, \mathbf{r}_{k-3})$.

These parameters allow the explicit computation of $B_k$, $G_k$, $C_k$, and $A_k$ as is given by equations 31, 32, 33 and 34 respectively. Equations (37) and (38) define the new Lanczos-type algorithm.

Now, since all previous formulae are only valid for $k \geq 3$, it is necessary to find the expressions of the polynomials of degrees 1 and 2. From (9), we can write

$$P_1(x) = \frac{\begin{vmatrix} 1 & x \\ c_0 & c_1 \end{vmatrix}}{c_1},$$

$$P_1(x) = 1 - \frac{c_0}{c_1}x,$$

$\mathbf{r}_1 = \mathbf{r}_0 - \frac{c_0}{c_1}A\mathbf{r}_0$, and $\mathbf{x}_1 = \mathbf{x}_0 + \frac{c_0}{c_1}\mathbf{r}_0$, where $c_i = (\mathbf{y}, A^i\mathbf{r}_0)$.

Using (9) again, we can write

$$P_2(x) = \frac{\begin{vmatrix} 1 & x & x^2 \\ c_0 & c_1 & c_2 \\ c_1 & c_2 & c_3 \end{vmatrix}}{\begin{vmatrix} c_1 & c_2 \\ c_2 & c_3 \end{vmatrix}},$$

$$P_2(x) = 1 - \frac{c_0c_3 - c_1c_2}{c_1c_3 - c_2^2}x + \frac{c_0c_2 - c_1^2}{c_1c_3 - c_2^2}x^2,$$

$\mathbf{r}_2 = \mathbf{r}_0 - \alpha A\mathbf{r}_0 + \beta A^2\mathbf{r}_0$, and $\mathbf{x}_2 = \mathbf{x}_0 + \alpha\mathbf{r}_0 - \beta A\mathbf{r}_0$, where $\alpha = \frac{c_0c_3 - c_1c_2}{\delta}$, $\beta = \frac{c_0c_2 - c_1^2}{\delta}$ and $\delta = c_1c_3 - c_2^2$.

3.2. **Algorithm** $A_{12}$**.** Putting together the various steps given in the above section, the new algorithm can be described as follows.

**Algorithm 2** Algorithm $A_{12}$

Choose $\mathbf{x}_0$ and $\mathbf{y}$ such that $\mathbf{y} \neq 0$, and choose $\epsilon$ arbitrarily small and positive.
Set $\mathbf{r}_0 = \mathbf{b} - A\mathbf{x}_0$, $\mathbf{y}_0 = \mathbf{y}$, $\mathbf{p} = A\mathbf{r}_0$, $\mathbf{p}_1 = A\mathbf{p}$, $c_0 = (\mathbf{y}, \mathbf{r}_0)$,
$c_1 = (\mathbf{y}, \mathbf{p})$, $c_2 = (\mathbf{y}, \mathbf{p}_1)$, $c_3 = (\mathbf{y}, A\mathbf{p}_1)$, $\delta = c_1c_3 - c_2^2$,
$\alpha = \frac{c_0c_3 - c_1c_2}{\delta}$, $\beta = \frac{c_0c_2 - c_1^2}{\delta}$,
$\mathbf{r}_1 = \mathbf{r}_0 - \frac{c_0}{c_1}\mathbf{p}$, $\mathbf{x}_1 = \mathbf{x}_0 + \frac{c_0}{c_1}\mathbf{r}_0$,
$\mathbf{r}_2 = \mathbf{r}_0 - \alpha p + \beta\mathbf{p}_1$, $\mathbf{x}_2 = \mathbf{x}_0 + \alpha\mathbf{r}_0 - \beta\mathbf{p}$,
$\mathbf{y}_1 = A^T\mathbf{y}_0$, $\mathbf{y}_2 = A^T\mathbf{y}_1$, $\mathbf{y}_3 = A^T\mathbf{y}_2$, $k = 3$.
**while** $||r_k|| \geq \epsilon$ **do**
  $\mathbf{y}_{k+1} = A^T\mathbf{y}_k$, $\mathbf{q}_1 = A\mathbf{r}_{k-1}$, $\mathbf{q}_2 = A\mathbf{q}_1$, $\mathbf{q}_3 = A\mathbf{r}_{k-2}$,
  $a_{11} = (\mathbf{y}_{k-2}, \mathbf{r}_{k-2})$, $a_{13} = (\mathbf{y}_{k-3}, \mathbf{r}_{k-3})$, $a_{21} = (\mathbf{y}_{k-1}, \mathbf{r}_{k-2})$, $a_{22} = a_{11}$,
  $a_{23} = (\mathbf{y}_{k-2}, \mathbf{r}_{k-3})$, $a_{31} = (\mathbf{y}_k, \mathbf{r}_{k-2})$, $a_{32} = a_{21}$, $a_{33} = (\mathbf{y}_{k-1}, \mathbf{r}_{k-3})$,
  $s = (\mathbf{y}_{k+1}, \mathbf{r}_{k-2})$, $t = (\mathbf{y}_k, \mathbf{r}_{k-3})$, $F_k = -\frac{a_{11}}{a_{13}}$,
  $b_1 = -a_{21} - a_{23}F_k$, $b_2 = -a_{31} - a_{33}F_k$, $b_3 = -s - tF_k$,
  $\Delta_k = a_{11}(a_{22}a_{33} - a_{32}a_{23}) + a_{13}(a_{21}a_{32} - a_{31}a_{22})$,
  $B_k = \frac{b_1(a_{22}a_{33} - a_{32}a_{23}) + a_{13}(b_2a_{32} - b_3a_{22})}{\Delta_k}$,
  $G_k = \frac{b_1 - a_{11}B_k}{a_{13}}$,
  $C_k = \frac{b_2 - a_{21}B_k - a_{23}G_k}{a_{22}}$,
  $A_k = \frac{1}{C_k + G_k}$,
  $\mathbf{r}_k = A_k\{\mathbf{q}_2 + B_k\mathbf{q}_1 + C_k\mathbf{r}_{k-2} + F_k\mathbf{q}_3 + G_k\mathbf{r}_{k-3}\}$,
  $\mathbf{x}_k = A_k\{C_k\mathbf{x}_{k-2} + G_k\mathbf{x}_{k-3} - (\mathbf{q}_1 + B_k\mathbf{r}_{k-2} + F_k\mathbf{r}_{k-3})\}$,
  $k = k + 1$.
**end while**
Stop; solution found.

## 4. Numerical results

$A_{12}$, [17], the algorithm described in the above section, has been tested against algorithms $A_5/B_{10}$ and $A_8/B_{10}$, the best algorithms according to [3, 2], as well as against the established Arnoldi algorithm, [1, 36].

4.1. **Test problems I.** The test problems considered here arise in the 5-point discretisation of the operator $\frac{-d^2}{dx^2} - \frac{d^2}{dy^2} + \gamma\frac{d}{dx}$ on a rectangular region. Comparative results on instances of the following problem ranging from dimension 10 to 100 for parameter $\delta$ taking values 0.0 and 0.2 respectively, are recorded in Table 1 and Table 2.

$$A = \begin{pmatrix} B & -I & \cdots & \cdots & 0 \\ -I & B & -I & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & -I & B & -I \\ 0 & \cdots & \cdots & -I & B \end{pmatrix}, \text{with } B = \begin{pmatrix} 4 & \alpha & \cdots & \cdots & 0 \\ \beta & 4 & \alpha & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \beta & 4 & \alpha \\ 0 & \cdots & & \beta & 4 \end{pmatrix}.$$

and $\alpha = -1 + \delta$, and $\beta = -1 - \delta$. The right-hand side is $\mathbf{b} = A\mathbf{x}$, where $\mathbf{x} = (1, 1, \ldots, 1)^T$, is the solution of the system. The dimension of $B$ is 10.

TABLE 1. Experimental results for problems when $\delta = 0$

| | Arnoldi | | $A_5/B_{10}$ | | $A_8/B_{10}$ | | $A_{12}$ | |
|---|---|---|---|---|---|---|---|---|
| $n$ | $\|\|r_k\|\|$ | t(sec) | $\|\|r_k\|\|$ | t(sec) | $\|\|r_k\|\|$ | t(sec) | $\|\|r_k\|\|$ | t(sec) |
| 10 | $1.2514E^{-10}$ | 0.001450 | $2.2940E^{-13}$ | 0.001892 | $1.7704E^{-13}$ | 0.002770 | $4.9623E^{-13}$ | 0.002819 |
| 20 | $1.7733E^{-11}$ | 0.002207 | $2.5256E^{-14}$ | 0.001842 | $1.7489E^{-13}$ | 0.002654 | $1.7536E^{-13}$ | 0.002904 |
| 30 | $1.2990E^{-14}$ | 0.003602 | $3.9026E^{-09}$ | 0.002220 | $4.9472E^{-09}$ | 0.003179 | $5.4705E^{-08}$ | 0.003370 |
| 40 | $3.5434E^{-11}$ | 0.006071 | $1.4770E^{-10}$ | 0.002416 | $8.4658E^{-10}$ | 0.003095 | $1.4776E^{-08}$ | 0.003526 |
| 50 | $6.1827E^{-08}$ | 0.008870 | $1.9959E^{-06}$ | 0.002962 | $1.3598E^{-06}$ | 0.003696 | $4.7994E^{-06}$ | 0.003980 |
| 60 | $2.9843E^{-14}$ | 0.012282 | $9.1910E^{-06}$ | 0.003001 | $3.7470E^{-06}$ | 0.003776 | $5.0010E^{-06}$ | 0.004354 |
| 70 | $4.2642E^{-13}$ | 0.017151 | $4.9035E^{-06}$ | 0.003622 | $4.2579E^{-06}$ | 0.004194 | $1.3781E^{-06}$ | 0.005658 |
| 80 | $5.0951E^{-08}$ | 0.021938 | $4.4311E^{-06}$ | 0.004498 | $7.7199E^{-06}$ | 0.005504 | $7.5581E^{-06}$ | 0.005271 |
| 90 | $9.6960E^{-13}$ | 0.029083 | $NaN$ | | $8.5560E^{-06}$ | 0.007900 | $3.7301E^{-06}$ | 0.006541 |
| 100 | $1.1397E^{-13}$ | 0.036462 | $1.1889E^{-06}$ | 0.003849 | $3.1695E^{-06}$ | 0.004499 | $8.9530E^{-07}$ | 0.005084 |

TABLE 2. Experimental results for problems when $\delta = 0.2$

| | Arnoldi | | $A_5/B_{10}$ | | $A_8/B_{10}$ | | $A_{12}$ | |
|---|---|---|---|---|---|---|---|---|
| $n$ | $\|\|r_k\|\|$ | t(sec) | $\|\|r_k\|\|$ | t(sec) | $\|\|r_k\|\|$ | t(sec) | $\|\|r_k\|\|$ | t(sec) |
| 10 | $2.3499E^{-15}$ | 0.001377 | $5.2347E^{-04}$ | 0.002339 | $5.2347E^{-04}$ | 0.002948 | $5.2347E^{-04}$ | 0.003231 |
| 20 | $5.6622E^{-11}$ | 0.002149 | $4.1778E^{-11}$ | 0.001842 | $5.8526E^{-11}$ | 0.003090 | $6.3915E^{-10}$ | 0.003372 |
| 30 | $6.8771E^{-15}$ | 0.003573 | $8.9881E^{-04}$ | 0.002220 | $8.9880E^{-04}$ | 0.003580 | $8.9880E^{-04}$ | 0.003847 |
| 40 | $1.8106E^{-10}$ | 0.006137 | $8.7583E^{-04}$ | 0.002830 | $9.3988E^{-04}$ | 0.003620 | $9.1261E^{-04}$ | 0.003977 |
| 50 | $3.5345E^{-08}$ | 0.008552 | $6.2669E^{-04}$ | 0.003360 | $5.7269E^{-04}$ | 0.004055 | $2.5040E^{-04}$ | 0.004964 |
| 60 | $2.8757E^{-13}$ | 0.012544 | $6.3670E^{-04}$ | 0.003877 | $8.4915E^{-04}$ | 0.004885 | $7.3489E^{-04}$ | 0.005345 |
| 70 | $4.2552E^{-13}$ | 0.017352 | $8.5670E^{-04}$ | 0.003902 | $7.0703E^{-04}$ | 0.006158 | $9.9086E^{-04}$ | 0.005052 |
| 80 | $1.7785E^{-04}$ | 0.021629 | $NaN$ | | $NaN$ | | $6.5602E^{-04}$ | 0.012131 |
| 90 | $1.4837E^{-04}$ | 0.029332 | $NaN$ | | $7.5451E^{-04}$ | 0.011230 | $9.5294E^{-04}$ | 0.011842 |
| 100 | $5.8942E^{-13}$ | 0.037067 | $NaN$ | | $NaN$ | | $9.9710E^{-04}$ | 0.018899 |

4.2. **Test problems II.** The coefficient matrix here is taken as the Hilbert matrix, i.e. $A = hilb(n)$, where $hilb(n)$ is a Matlab function,

$n$ being the dimension of $A$. The right-hand side $\mathbf{b}$ and the solution $\mathbf{x}$, are defined in the same way as in test problems I. The Hilbert matrix is notoriously ill-conditioned. Ill-conditioned systems of linear equations are notoriously difficult to solve to any useful accuracy, [18, 27, 34]

TABLE 3. Experimental results when $A$ is a Hilbert matrix.

| | Arnoldi | | $A_5/B_{10}$ | | $A_8/B_{10}$ | | $A_{12}$ | |
|---|---|---|---|---|---|---|---|---|
| $n$ | $\|\|r_k\|\|$ | t(sec) | $\|\|r_k\|\|$ | t(sec) | $\|\|r_k\|\|$ | t(sec) | $\|\|r_k\|\|$ | t(sec) |
| 10 | $3.2101E^{-15}$ | 0.002364 | $4.4809E^{-06}$ | 0.001945 | $4.4809E^{-06}$ | 0.002834 | $4.4809E^{-06}$ | 0.002706 |
| 20 | $2.2288E^{-15}$ | 0.003285 | $9.6002E^{-05}$ | 0.001772 | $9.6002E^{-05}$ | 0.002839 | $9.6002E^{-05}$ | 0.002787 |
| 30 | $3.8953E^{-15}$ | 0.004148 | $1.3341E^{-05}$ | 0.001929 | $1.3376E^{-05}$ | 0.003050 | $1.3340E^{-05}$ | 0.002791 |
| 40 | $3.2251E^{-14}$ | 0.005521 | $3.8228E^{-05}$ | 0.001880 | $3.8276E^{-05}$ | 0.003327 | $3.8229E^{-05}$ | 0.002772 |
| 50 | $2.8673E^{-15}$ | 0.007068 | $7.9457E^{-05}$ | 0.001997 | $7.9463E^{-05}$ | 0.003475 | $7.9457E^{-05}$ | 0.003334 |

All algorithms have been implemented in Matlab version 7.8.0 and run on a PC, under Microsoft Windows XP Professional Operating System, with 3.2GB RAM, and 2.40 GHz Intel(R) Core(TM) 2 CPU 6600. The problems are solved as dense problems, i.e. no sparsity has been exploited. The results point to the Arnoldi algorithm being the most robust overall, but also the slowest overall. Algorithms $A_5/B_{10}$ and $A_8/B_{10}$ are the fastest overall, but the least robust overall; in fact they have failed to solve some problems in high dimension due to breakdown, of course, which is endemic in Lanczos-type algorithms. Algorithm $A_{12}$, like Arnoldi, solved all problems but faster and not as accurately. It is also more robust than $A_5/B_{10}$ and $A_8/B_{10}$ overall, but slower than both of them overall. Its lower speed compared to that of $A_5/B_{10}$ and $A_8/B_{10}$ is expected since the recurrence relation $A_{12}$ involves more coefficients than both recurrence relations $A_5/B_{10}$ and $A_8/B_{10}$. Note that on the Hilbert-type problems, Table 3, the algorithms could not cope with dimensions higher than 50. Arnoldi is again the most stable overall and the slowest as the dimension grows. The other three algorithms have similar performances.

## 5. Conclusion and further work

The way Lanczos-type algorithms are derived using recurrence relations involving FOP's means that many such algorithms can be created, each based on a different set of relations. The choice of recurrence relations to use is dictated by the degree of FOP's to be involved; high degrees mean a large number of coefficients have to be calculated in the

concerned Lanczos process. This, consequently, dictates the computational complexity of the resulting Lanczos-type algorithm. However, it is well known, [26, 30], that computational complexity does not always imply efficiency, or indeed robustness. Moreover, robustness and accuracy are often more important. It is therefore worthwhile to look beyond complexity issues sometimes, like we did here.

In this paper we have shown that, indeed, there are recurrence relations worth exploring since they lead to more robust algorithms. As a result, a new Lanczos-type algorithm, $A_{12}$ has been designed. The numerical performance of this algorithm is compared to that of two existing Lanczos-type algorithms, which were found to be the best among a number of Lanczos-type algorithms, [2, 3], on the same set of problems as considered here. It is also compared to the well established Arnoldi algorithm. It is interesting to find that algorithm $A_{12}$ is overall more robust than $A_5/B_{10}$ and $A_8/B_{10}$ and faster than Arnoldi's. This makes it occupy, at least on the set of problems used here and elsewhere, a happy medium position. It is therefore the ideal candidate for time-limited applications which do not require high accuracy.

## References


[1] W. E. Arnoldi. The principal of minimized iterations in the solution of the matrix eigenvalue problem. *Quarterly of Applied Mathematics*, 9 (1951):17–29.

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

[3] C. Baheux. *Algorithmes d'implementation de la méthode de Lanczos*. PhD thesis, University of Lille 1, France, 1994.

[4] A. Bjôrck, T. Elfving, and Z. Strakos. Stability of Conjugate Gradient and Lanczos Methods for Linear Least Squares Problems. *SIAM Journal of Matrix Analysis and Application*, 19 (1998):720–736.

[5] C. Brezinski. *Padé-Type Approximation and General Orthogonal Polynomials, Internat. Ser. Nuner. Math. 50*. Birkhäuser, Basel, 1980.

[6] C. Brezinski and H. Sadok. Lanczos-type algorithms for solving systems of linear equations. *Applied Numerical Mathematics*, 11 (1993):443–473.

[7] C. Brezinski and M. R. Zaglia. Hybird procedures for solving linear systems. *Numerische Mathematik*, 67 (1994):1–19.

[8] C. Brezinski and M. R. Zaglia. A New Presentation of Orthogonal Polynomials with Applications to their Computation. *Numerical Algorithms*, 1 (1991):207–222.

[9] C. Brezinski, M. R. Zaglia, and H. Sadok. Avoiding breakdown and near-breakdown in Lanczos type algorithms. *Numerical Algorithms*, 1 (1991):261–284.

[10] C. Brezinski, M. R. Zaglia, and H. Sadok. A Breakdown-free Lanczos type algorithm for solving linear systems. *Numerische Mathematik*, 63 (1992):29–38.

[11] C. Brezinski, M. R. Zaglia, and H. Sadok. The matrix and polynomial approaches to Lanczos-type algorithms. *Journal of Computational and Applied Mathematics*, 123 (2000):241–260.

[12] C. Brezinski, M. R. Zaglia, and H. Sadok. New look-ahead Lanczos-type algorithms for linear systems. *Numerische Mathematik*, 83 (1999):53–85.

[13] C. Brezinski, M. R. Zaglia, and H. Sadok. A review of formal orthogonality in Lanczos-based methods. *Journal of Computational and Applied Mathematics*, 140 (2002):81–98.

[14] C. G. Broyden and M. T. Vespucci. *Krylov Solvers For Linear Algebraic Systems*. Elsevier, Amsterdam, The Netherlands, 2004.

[15] D. Calvetti, L. Reichel, F. Sgallari, and G. Spaletta. A Regularizing Lanczos iteration method for underdetermined linear systems. *Jouranl of Computational and Applied Mathematics*, 115 (2000):101–120.

[16] A. Draux. *Polynômes Orthogonaux Formels. Application, LNM 974*. Springer-Verlag, Berlin, 1983.

[17] M. Farooq. *New Lanczos-type Algorithms and their Implementation*. PhD thesis, University of Essex, UK, 2011. `http://serlib0.essex.ac.uk/record=b1754556`.

[18] M. Farooq and A. Salhi. Improving the solvability of ill-conditioned systems of linear equations by reducing the condition number of their matrices. *J. Korean Math. Soc.*, 48 (5) (2011):939–952. http://dx.doi.org/10.4134/JKMS.2011.48.5.939.

[19] M. Farooq and A. Salhi. New Recurrence Relationships Between Orthogonal Polynomials Which Lead to New Lanczos-type Algorithms. *Journal of Prime Research in Mathematics*, 8 (2012):61–75. http://www.sms.edu.pk/journals/jprm/jprmvol8/09.pdf.

[20] M. Farooq and A. Salhi. A Preemptive Restarting Approach to Beating the Inherent Instability of Lanczos-type Algorithms. *Iranian Journal of Sceince and Technology, Transaction A: Science*, 37 (3.1) (2013):349–358. `http://ijsts.shirazu.ac.ir/?_action=articleInfo&article=1634&vol=142`.

[21] M. Farooq and A. Salhi. A Switching Approach to Avoid Breakdown in Lanczos-type Algorithms. *Applied Mathematics and Information Sciences*, 8 (5) (2014):2161–2169. `http://naturalspublishing.com/ContIss.asp?IssID=190`.

[22] R. Fletcher. Conjugate gradient methods for indefinite systems. In G.Alistair Watson, editor, *Numerical Analysis*, volume 506 of *Lecture Notes in Mathematics*, pages 73–89. Springer Berlin Heidelberg, 1976.

[23] A. Greenbaum. *Iterative Methods for Solving Linear System*. Society for Industrial and Applied Mathematics, Philadelphia, 1997.

[24] A. El Guennouni. A unified approach to some strategies for the treatment of breakdown in Lanczos-type algorithms. *Applicationes Mathematicae*, 26 (1999):477–488.

[25] M.R. Hestenes and E. Stiefel. Mehtods of conjugate gradients for solving linear systems. *Journal of the National Bureau of Standards*, 49 (1952):409–436.

[26] L. G. Khachyan. A polynomial algorithm in linear programming. *Soviet Mathematics Doklady (translated)*, 20 (1979):191–194.
[27] H. J. Kim, K. Choi, H. B. Lee, H. K. Jung, and S. Y. Hahn. A new algorithm for solving ill-conditioned linear system. *IEEE Transactions on Magnetics*, 32(3) (1996):1373–1376.
[28] C. Lanczos. An Iteration Method for the Solution of the Eigenvalue Problem of Linear Differential and Integeral Operators. *Journal of Research of the National Bureau of Standards*, 45 (1950):255–282.
[29] C. Lanczos. Solution of systems of linear equations by minimized iteration. *Journal of the National Bureau of Standards*, 49 (1952):33–53.
[30] L. Lovasz. The Ellipsoid Algorithm: Better or Worse than the Simplex? *Mathematical Intelligencer*, 2 (1980):141–146.
[31] G. Meurant. *The Lanczos and conjugate gradient algorithms, From Theory to Finite Precision Computations*. SIAM, Philadelphia, 2006.
[32] B. N. Parlett and D. S. Scott. The Lanczos Algorithm With Selective Orthogonaliztion. *Mathematics of Computation*, 33 (1979):217–238.
[33] B. N. Parlett, D. R. Taylor, and Z. A. Liu. A Look-Ahead Lanczos Algorithm for Unsymmetric Matrices. *Mathematics of Computation*, 44 (1985):105–124.
[34] J. R. Rice. *Matrix Computations and Mathematical Software*. McGraw-Hill, New York, 1981.
[35] Y. Saad. On the Lanczos method for solving linear system with several right-hand sides. *Mathematics of Computation*, 48 (1987):651–662.
[36] Y. Saad. *Iterative methods for sparse linear systems*. SIAM, Philadelphia, 2003.
[37] G. Szegö. *Orthogonal Polynomials*. American Mathematical Society, Providence, Rhode Island, 1939.
[38] H. A. Van der Vorst. An iterative solution method for solving f(A)$\mathbf{x}$=$\mathbf{b}$, using Krylov subspace information obtained for the symmetric positive definite matrix A. *Journal of Computational and Applied Mathematics*, 18(2) (1987):249–263.
[39] Q. Ye. A Breakdown-Free Variation of the Nonsymmetric Lanczos Algorithms. *Mathematics of Computation*, 62 (1994):179–207.