J Eng Math (2015) 93:3–19
DOI 10.1007/s10665-013-9656-3

# Convergence acceleration of Kaczmarz's method

**Claude Brezinski · Michela Redivo-Zaglia**

Received: 18 January 2013 / Accepted: 11 June 2013 / Published online: 28 August 2013


**Abstract** The method of alternating projections (MAP) is an iterative procedure for finding the projection of a point on the intersection of closed subspaces of a Hilbert space. The convergence of this method is usually slow, and several methods for its acceleration have already been proposed. In this work, we consider a special MAP, namely Kaczmarz's method for solving consistent systems of linear equations. The convergence of this method is discussed. After giving its matrix formulation and its projection properties, we consider several procedures for accelerating its convergence. They are based on sequence transformations whose kernels contain sequences of the same form as the sequence of vectors generated by Kaczmarz's method. Acceleration can be achieved either directly, that is without modifying the sequence obtained by the method, or by restarting it from the vector obtained by acceleration. Numerical examples show the effectiveness of both procedures.

**Keywords** Acceleration · Extrapolation · Kaczmarz's method · Linear systems · Projection method

## 1 Introduction

The solutions of many problems in numerical analysis and in applied mathematics make use of an iterative method. Quite often, the convergence is slow, thus diminishing the interest of the method. In such a case, if possible, the iterative method can be modified to obtain a better convergence. But if the iterates of the method are produced by a black box with no access, one can transform them, by a sequence transformation, into a new sequence that, under some assumptions, converges faster to the same limit (see [1] for a review of such transformations). In this paper, we want to give an illustration of such a procedure. For that purpose, we chose the method of Kaczmarz for solving a system of linear equations. The reason for this choice will be clearly explained subsequently. Kaczmarz' method

This paper is dedicated to the memory of Bernard Germain-Bonne (1940–2012).

C. Brezinski (✉)
Laboratoire Paul Painlevé, UMR CNRS 8524, UFR de Mathématiques Pures et Appliquées,
Université des Sciences et Technologies de Lille, 59655 Villeneuve d'Ascq Cedex, France
e-mail: claude.brezinski@univ-lille1.fr

M. Redivo-Zaglia
Dipartimento di Matematica, Università degli Studi di Padova, Via Trieste 63, 35121 Padova, Italy
e-mail: Michela.RedivoZaglia@unipd.it

belongs to the important class of methods, called the *methods of alternating projections*, for finding the projection of a point on the intersection of several subspaces of a vector space. As written in the introduction of [2], alternating projection methods have proved to be useful for the solution of important problems in various areas of mathematics, and also in many real applications. Let us begin by a short presentation of these methods.

Let $Q_i$ denote the orthogonal projection on a closed subspace $M_i$ of a Hilbert space $H$, and let $Q_M$ be the composition of the $r$ projecting operators $Q_i$, that is, $Q_M = Q_r \cdots Q_1$. Let $P_M$ be the projection on $M$, the intersection of the subspaces $M_i$.

We are looking for the projection of a given point $\bar{\mathbf{x}}$ on $M$. It holds that

$$\lim_{n\to\infty} Q_M^n \bar{\mathbf{x}} = P_M \bar{\mathbf{x}} \quad \forall \bar{\mathbf{x}} \in H.$$

The *method of alternating projections* (MAP) consists in the iterations

$$\mathbf{x}_{n+1} = Q_M \mathbf{x}_n, \quad n = 0, 1, \ldots, \quad \mathbf{x}_0 = \bar{\mathbf{x}}.$$

This method often converges quite slowly and needs to be accelerated [2]. For this purpose, the projection operator $Q_M$ can be replaced by another (not necessarily linear) operator $T$ that can depend on $n$.

There are two ways of using $T$:

1. Keep the sequence $(\mathbf{x}_n)$ as given by MAP unchanged, and consider the new sequence $(\mathbf{y}_n)$ built by $\mathbf{y}_n = T\mathbf{x}_n$.
2. Iterate the operator $T$, that is, consider the new iterations $\mathbf{x}_{n+1} = T\mathbf{x}_n$.

There exist many choices for $T$. For example, the choice

$$T\mathbf{x}_n = \mathbf{x}_n - t_n \left(Q_M \mathbf{x}_n - \mathbf{x}_n\right),$$

with $t_n = (\mathbf{x}_n, \mathbf{x}_n - Q_M\mathbf{x}_n)/\|\mathbf{x}_n - Q_M\mathbf{x}_n\|^2$, was proposed in [2, p. 44], while $t_n = (\mathbf{r}_n, \mathbf{v}_n)/(\mathbf{v}_n, \mathbf{v}_n)$, where $\mathbf{r}_n = Q_M\mathbf{x}_n - \mathbf{x}_n$ and $\mathbf{v}_n = Q_M^2\mathbf{x}_n - 2Q_M\mathbf{x}_n + \mathbf{x}_n$, was discussed in [3] (the norms are the Euclidean ones). More choices are presented in [4].

Kaczmarz's method [5] for solving a system of linear equations was proposed in 1937. Later, it was rediscovered by Gordon et al. [6] and applied in medical imaging. It was called the *Algebraic Reconstruction Technique* (ART), and its original version, or some variants, continues to be used for tomographic imaging. It is a particular case of row projection methods that received much attention (see, for example, [4,7–10]), and it enters into the framework of MAP. It is also well suited for parallel computations and large-scale problems since each step only requires one row of matrix $A$ (or several rows simultaneously in its block version) and no matrix–vector products. For an impressive list of publications on Kaczmarz's method, see [11]. Kaczmarz's method is also often used for solving an overdetermined consistent $M \times N$ linear system with $M \geq N$, but in this paper, we restrict ourselves to the case of a square regular system. It is easy to extend the algorithms and the theory by properly replacing $N$ by $M$. Let us mention that there exist methods consisting of a decomposition of a matrix into blocks of columns instead of rows and that such methods can also be conveniently implemented on parallel computers [12].

A drawback of Kaczmarz's method, as in general of all projection iterative methods, is its often slow convergence. Thanks to its matrix analysis, we will be able to show how its convergence can be accelerated by some particular choices of the operator $T$ corresponding to the two ways of using it as described earlier.

For definitions and properties of projections, see, for example, [9]. In the sequel, the same notation will be used for a matrix and the projection it represents.

## 2 Kaczmarz's method

We consider a consistent $N \times N$ linear system $A\mathbf{x} = \mathbf{b}$. One single step of Kaczmarz's method consists in

$$\mathbf{p}_{n+1} = \mathbf{p}_n + \frac{(\mathbf{b} - A\mathbf{p}_n, \mathbf{e}_i)}{\left(A^{\mathrm{T}}\mathbf{e}_i, A^{\mathrm{T}}\mathbf{e}_i\right)} A^{\mathrm{T}}\mathbf{e}_i, \tag{1}$$

where $\mathbf{e}_i$ is the $i$th vector of the canonical basis of $\mathbb{R}^N$. There exist several strategies for choosing the index $i$ at each step. The most common one is $i = n \pmod{N} + 1$. In this case, the method is called the *cyclic Kaczmarz method* or, simply, *Kaczmarz's method*, since it was originally proposed by Kaczmarz in this form [5]. One iteration of Kaczmarz's method consists in a complete cycle of steps in their natural order, that is,

$$\left.\begin{aligned} &\mathbf{p}_0 = \mathbf{x}_n, \\ &\mathbf{p}_i = \mathbf{p}_{i-1} + \frac{(\mathbf{b} - A\mathbf{p}_{i-1}, \mathbf{e}_i)}{\left(A^{\mathrm{T}}\mathbf{e}_i, A^{\mathrm{T}}\mathbf{e}_i\right)} A^{\mathrm{T}}\mathbf{e}_i, \qquad i = 1, \ldots, N, \\ &\mathbf{x}_{n+1} = \mathbf{p}_N. \end{aligned}\right\} \tag{2}$$

Denote by $\mathbf{a}_i = A^{\mathrm{T}}\mathbf{e}_i$ the column vector formed by the $i$th row of $A$ and by $b_i$ the $i$th component of the right-hand side $\mathbf{b}$. The computation of each vector $\mathbf{p}_i$ in (2) does not require any matrix–vector product, and we have

$$\mathbf{p}_i = \mathbf{p}_{i-1} + \frac{b_i - (\mathbf{p}_{i-1}, \mathbf{a}_i)}{\|\mathbf{a}_i\|^2}\, \mathbf{a}_i. \tag{3}$$

This remark shows one of the main advantages of Kaczmarz's method, and it allows an easy parallel implementation.

*Remark 1* Setting $\lambda_i = (\mathbf{b} - A\mathbf{p}_{i-1}, \mathbf{e}_i)/(A^{\mathrm{T}}\mathbf{e}_i, A^{\mathrm{T}}\mathbf{e}_i)$, one iteration of Kaczmarz's method (2) can be written as $\mathbf{x}_{n+1} = \mathbf{x}_n + A^{\mathrm{T}}\Lambda_n$, with $\Lambda_n = (\lambda_1, \ldots, \lambda_N)^{\mathrm{T}} \in \mathbb{R}^N$.

*Remark 2* Let $\mathbf{y}_n$ be the iterates obtained by applying the Gauss–Seidel method to the system $AA^{\mathrm{T}}\mathbf{y} = \mathbf{b}$. Then $\mathbf{x} = A^{\mathrm{T}}\mathbf{y}$ and $\mathbf{x}_n = A^{\mathrm{T}}\mathbf{y}_n$ [13].

Inside one iteration, we have the following orthogonality properties for $i = 1, \ldots, N$ [14,15]:

$$(\mathbf{b} - A\mathbf{p}_i, \mathbf{e}_i) = 0, \qquad (\mathbf{p}_i - \mathbf{p}_{i-1}, \mathbf{x} - \mathbf{p}_i) = 0.$$

Then $\mathbf{p}_i$ is the orthogonal projection of $\mathbf{p}_{i-1}$ on $M_i = \{\mathbf{y} \mid (\mathbf{b} - A\mathbf{y}, \mathbf{e}_i) = 0\}$ along $A^{\mathrm{T}}\mathbf{e}_i$.

Let us now analyze each step of (2). Since $(\mathbf{b} - A\mathbf{p}_i, \mathbf{e}_i) = 0$, it holds that

$$\|\mathbf{x} - \mathbf{p}_{i-1}\|^2 = \|\mathbf{x} - \mathbf{p}_i\|^2 + \|\mathbf{p}_i - \mathbf{p}_{i-1}\|^2,$$

which shows that $\|\mathbf{x} - \mathbf{p}_i\| \le \|\mathbf{x} - \mathbf{p}_{i-1}\|$. Therefore, $\|\mathbf{x} - \mathbf{x}_{n+1}\| \le \|\mathbf{x} - \mathbf{x}_n\|$. In fact, as proved in [15], these inequalities are strict, and Kaczmarz's method is always converging to the solution of the system.

Obviously $(\mathbf{b} - A\mathbf{p}_{i-1}, \mathbf{e}_i)^2 \le (\mathbf{b} - A\mathbf{p}_{i-1}, \mathbf{b} - A\mathbf{p}_{i-1})$. In the case where $(\mathbf{b} - A\mathbf{p}_{i-1}, \mathbf{e}_i)^2 \ge (\mathbf{b} - A\mathbf{p}_{i-1}, \mathbf{b} - A\mathbf{p}_{i-1})/N$, one can prove, by an analysis similar to that in [14, p. 122], that the following result holds.

**Proposition 1** *If* $(\mathbf{b} - A\mathbf{p}_{i-1}, \mathbf{e}_i)^2 \ge (\mathbf{b} - A\mathbf{p}_{i-1}, \mathbf{b} - A\mathbf{p}_{i-1})/N$, *then*

$$\|\mathbf{x} - \mathbf{p}_i\|^2 \le \left(1 - \frac{1}{N\kappa(AA^{\mathrm{T}})}\right) \|\mathbf{x} - \mathbf{p}_{i-1}\|^2.$$

In the general case, this coefficient could be taken as an approximation of the convergence factor of the method. This result is quite similar to a corresponding result proved in [16] for Gastinel's method [17] and other methods, or to an extension (Theorem 4.27 in [18]) of another result given in [16].

### 2.1 Matrix interpretation

Following [15], where it seems that it first appeared, let us give the matrix interpretation of Kaczmarz's method (2) (see also [8,10,14]).

We set

$$\boldsymbol{\alpha}_i = \frac{A^{\mathrm{T}}\mathbf{e}_i}{\|A^{\mathrm{T}}\mathbf{e}_i\|^2} = \frac{\mathbf{a}_i}{\|\mathbf{a}_i\|^2}, \quad \boldsymbol{\varrho}_i = \mathbf{b} - A\mathbf{p}_i,$$
$$P_i = I - A\boldsymbol{\alpha}_i\mathbf{e}_i^{\mathrm{T}}, \quad Q_i = A^{-1}P_iA.$$

We have

$$Q_i = I - \frac{A^{\mathrm{T}}\mathbf{e}_i\mathbf{e}_i^{\mathrm{T}}A}{\|A^{\mathrm{T}}\mathbf{e}_i\|^2} = I - \frac{\mathbf{a}_i\mathbf{a}_i^{\mathrm{T}}}{\|\mathbf{a}_i\|^2} = I - \boldsymbol{\alpha}_i\mathbf{a}_i^{\mathrm{T}}.$$

The matrix $P_i$ represents the oblique projection on $\mathbf{e}_i^{\perp}$ along $AA^{\mathrm{T}}\mathbf{e}_i$, while $Q_i$ is the rank $N-1$ orthogonal projection on $(A^{\mathrm{T}}\mathbf{e}_i)^{\perp}$ along $A^{\mathrm{T}}\mathbf{e}_i$. Thus, for any vector $\mathbf{y}$, $(P_i\mathbf{y}, \mathbf{e}_i) = 0$, and $(Q_i\mathbf{y}, A^{\mathrm{T}}\mathbf{e}_i) = 0$.

Inside one iteration, we have, for $i = 1, \ldots, N$,

$$\left.\begin{aligned} \mathbf{p}_i &= Q_i\mathbf{p}_{i-1} + (\mathbf{b}, \mathbf{e}_i)\boldsymbol{\alpha}_i = Q_i\mathbf{p}_{i-1} + (I - Q_i)\mathbf{x}, \\ \mathbf{x} - \mathbf{p}_i &= \mathbf{x} - \mathbf{p}_{i-1} - \boldsymbol{\alpha}_i\mathbf{e}_i^{\mathrm{T}}\boldsymbol{\varrho}_{i-1} = Q_i(\mathbf{x} - \mathbf{p}_{i-1}), \\ \boldsymbol{\varrho}_i &= \boldsymbol{\varrho}_{i-1} - (\boldsymbol{\varrho}_{i-1}, \mathbf{e}_i)A\boldsymbol{\alpha}_i = P_i\boldsymbol{\varrho}_{i-1} \end{aligned}\right\}. \tag{4}$$

Replacing $\boldsymbol{\alpha}_i$ by its expression in the second relation in (4), it follows that

$$\|\mathbf{x} - \mathbf{p}_i\|^2 = \|\mathbf{x} - \mathbf{p}_{i-1}\|^2 - \frac{(\boldsymbol{\varrho}_{i-1}, \mathbf{e}_i)^2}{\|A^{\mathrm{T}}\mathbf{e}_i\|^2}.$$

Summing up this identity for $i = 1, \ldots, N$, we obtain the gain achieved after each iteration:

$$\|\mathbf{x} - \mathbf{x}_{n+1}\|^2 = \|\mathbf{x} - \mathbf{x}_n\|^2 - \sum_{i=1}^{N} \frac{(\boldsymbol{\varrho}_{i-1}, \mathbf{e}_i)^2}{\|A^{\mathrm{T}}\mathbf{e}_i\|^2}.$$

Setting

$$P = P_N \cdots P_1,$$
$$Q = Q_N \cdots Q_1 = A^{-1}P_N \cdots P_1 A = A^{-1}PA,$$
$$\mathbf{r}_n = \mathbf{b} - A\mathbf{x}_n,$$

it holds that

$$\mathbf{x} - \mathbf{x}_{n+1} = Q(\mathbf{x} - \mathbf{x}_n), \quad \mathbf{r}_{n+1} = P\mathbf{r}_n. \tag{5}$$

It obviously follows that

$$\mathbf{x} - \mathbf{x}_n = Q^n(\mathbf{x} - \mathbf{x}_0), \quad \mathbf{r}_n = P^n\mathbf{r}_0, \tag{6}$$

and we have

$$\mathbf{x}_{n+1} = Q\mathbf{x}_n + A^{-1}(I - P)\mathbf{b}. \tag{7}$$

Since, after the first iteration of Kaczmarz's method (which ends with $\mathbf{p}_N$), we set $\mathbf{x}_1 = \mathbf{p}_N$, and we start the second iteration from $\mathbf{p}_0 = \mathbf{x}_1$, this means that we are continuing the original steps (1), that the new vector $\mathbf{p}_1$ is, in fact, the vector $\mathbf{p}_{N+1}$ of (1), that the new vector $\mathbf{p}_2$ is $\mathbf{p}_{N+2}$, and so on. Thus, Kaczmarz's method is only a renumbering of the single steps, keeping only those whose index is a multiple of $N$, that is, $\mathbf{x}_{n+1} = \mathbf{p}_{(n+1)N}$ and $\mathbf{r}_{n+1} = \boldsymbol{\varrho}_{(n+1)N}$. The main interest of this renumbering lies in the relations (5), which express the connection between two consecutive iterations by means of the fixed matrix $Q$, instead of relations using matrices changing at each step of an iteration. We similarly have $\boldsymbol{\varrho}_{(n+1)N+j-1} = Q^{(j)}\boldsymbol{\varrho}_{nN+j-1}$, $j = 1, \ldots, N$, with $Q^{(j)} = P_{j-1} \cdots P_1 P_n \cdots P_j$. Notice that $Q^{(1)}$ is identical to $Q$.

The matrices $P$ and $Q$ are similar, and it holds that $\rho(Q) < 1$, as proved in [15]. Thus, the sequence $(\mathbf{x}_n)$ generated by Kaczmarz's method converges to the solution $\mathbf{x}$ of the system. Moreover, it follows from standard results on iterations of the form (6) that

$$\|\mathbf{x} - \mathbf{x}_n\| = \mathcal{O}\left(\rho(Q)^n\right).$$

*Remark 3* All the matrices $Q_i$ are singular and, thus, have a zero eigenvalue. The matrix $Q_k \cdots Q_1$ has $k$ eigenvalues with a modulus strictly smaller than one (at least one of them being zero) and $N - k$ equal to one. The matrix $Q$ is singular, and all its eigenvalues have a modulus strictly smaller than one.

By slightly improving Theorem 4.4 of [18], which is based on a result by Meany [19] on the norm of a product of orthogonal projections of rank $N-1$, we have the following proposition.

**Proposition 2**

$$\|\mathbf{x}-\mathbf{x}_{n+1}\|^2 \leq \left(1-\frac{(\det A)^2}{\prod_{i=1}^{N}\|A^{\mathrm{T}}\mathbf{e}_i\|^2}\right)\|\mathbf{x}-\mathbf{x}_n\|^2.$$

A generalization of this result was recently given in [20] for the case where $A$ and $\mathbf{b}$ are partitioned into blocks of rows.

Let us give some details about the block version. Assume that matrix $A$ is partitioned into the blocks of rows $A_1^{\mathrm{T}}, A_2^{\mathrm{T}}, \ldots$, where the matrix $A_i^{\mathrm{T}} \in \mathbb{R}^{N_i \times N}$ contains the rows $N_1+\cdots+N_{i-1}+1$ up to $N_1+\cdots+N_{i-1}+N_i$ of $A$ (with $N_0=0$), and vector $\mathbf{b}$ is partitioned accordingly. Each single step of the method now consists in the treatment of a block as a whole, and one complete cycle of all the blocks in their natural ordering is called the (cyclic) *block Kaczmarz method*. The matrix interpretation of this block version is the same as above, now with $\boldsymbol{\alpha}_i = (A_i^{\mathrm{T}})^{\dagger} = A_i(A_i^{\mathrm{T}}A_i)^{-1}$ and $P_i = I - A\boldsymbol{\alpha}_i E_i^{\mathrm{T}}$, where $E_i \in \mathbb{R}^{N\times N_i}$ is the matrix whose columns are the vectors $\mathbf{e}_{N_1+\cdots+N_{i-1}+1}, \ldots, \mathbf{e}_{N_1+\cdots+N_{i-1}+N_i}$ of the canonical basis of $\mathbb{R}^N$, and it holds that $Q_i = I - A_i(A_i^{\mathrm{T}}A_i)^{-1}A_i^{\mathrm{T}}$. With these notations, one step of the block Kaczmarz method is written

$$\mathbf{p}_i = \mathbf{p}_{i-1} + \left(A_i^{\mathrm{T}}\right)^{\dagger} E_i^{\mathrm{T}}\left(\mathbf{b} - A\mathbf{p}_{i-1}\right).$$

This relation clearly generalizes (3).

## 3 Convergence acceleration

For accelerating the convergence of a sequence, it can be transformed into another one that, under some assumptions, converges faster to the same limit. The idea behind such a *sequence transformation* is to assume that the sequence to be accelerated behaves as a model sequence or satisfies some property depending on unknown parameters. The set of these sequences is called the *kernel* of the transformation. The unknown parameters are determined by imposing that the sequence interpolates the model sequence from a certain starting index $n$. Then the limit of the model sequence, which depends on $n$ since the parameters depend on it, is taken as an approximation of the limit of the sequence to be accelerated, which is thus transformed into a new sequence. Of course, by construction, if the initial sequence belongs to the kernel of the transformation, then for all $n$ its limit is obtained. Although this observation was never proved rigorously, if the sequence is *close* in some sense to the kernel of the transformation, there is a good chance that it will be accelerated. This is why the notion of kernel is so important.

In practice, when having to accelerate a given sequence, one can use a known sequence transformation (also called an *acceleration algorithm* or an *extrapolation method*) and verify that the sequence can be accelerated by that transformation. Another approach is, starting from some algebraic property of the sequence to be accelerated, to construct a special transformation adapted to it. In both cases, the behavior of the sequence has to be analyzed or be characterized by some property.

On convergence acceleration methods for vector sequences, see, for instance [1].

### 3.1 Sequence generated by Kaczmarz's method

Consider the sequence of vectors $(\mathbf{x}_n)$ obtained by Kaczmarz's method. Let $\Pi_\nu$ be the minimal polynomial of the matrix $Q$ for the vector $\mathbf{x}-\mathbf{x}_0$, that is, the polynomial of smallest degree $\nu \leq N$ such that $\Pi_\nu(Q)(\mathbf{x}-\mathbf{x}_0)=0$. Since $\Pi_\nu(Q)(\mathbf{x}-\mathbf{x}_0) = A^{-1}\Pi_\nu(P)A(\mathbf{x}-\mathbf{x}_0) = A^{-1}\Pi_\nu(P)\mathbf{r}_0$, $\Pi_\nu$ is also the minimal polynomial of $P$ for the vector $\mathbf{r}_0$. Setting $\Pi_\nu(\xi) = c_0 + c_1\xi + \cdots + c_\nu\xi^\nu$, it holds from (6) that

$$Q^n\Pi_\nu(Q)(\mathbf{x}-\mathbf{x}_0) = c_0(\mathbf{x}-\mathbf{x}_n) + c_1(\mathbf{x}-\mathbf{x}_{n+1}) + \cdots + c_\nu(\mathbf{x}-\mathbf{x}_{n+\nu}) = 0, \quad n=0,1,\ldots \tag{8}$$

Thus, it follows that the vectors $\mathbf{x}-\mathbf{x}_n$ produced by Kaczmarz's method satisfy such a homogeneous linear difference equation of order $\nu$ whose solution is well known.

If $Q$ is nondefective and if we denote by $\tau_1, \ldots, \tau_\nu$ the distinct zeros of $\Pi_\nu$, then its solution is given by

$$\mathbf{x} - \mathbf{x}_n = \sum_{i=1}^{\nu} d_i \tau_i^n \mathbf{v}_i, \quad n = 0, 1, \ldots,$$

where the $d_i$ are scalars and the $\mathbf{v}_i$ the eigenvectors corresponding to the $\tau_i$. If $Q$ is defective, then the expression for $\mathbf{x} - \mathbf{x}_n$ still involves powers of the eigenvalues, but it is more complicated since each eigenvalue is multiplied by a polynomial in $n$ whose degree is its algebraic multiplicity minus one (see [21] for more details).

### 3.2 Sequence transformations

The vector $\mathbf{x}$ can be exactly computed if we are able to build a sequence transformation whose kernel consists of sequences of the form (8). However, since, in practical applications, $\nu$ could be quite large, we will restrict ourselves to building a sequence transformation whose kernel contains all sequences of the form

$$a_0^{(n)}(\mathbf{x} - \mathbf{x}_n) + a_1^{(n)}(\mathbf{x} - \mathbf{x}_{n+1}) + \cdots + a_k^{(n)}(\mathbf{x} - \mathbf{x}_{n+k}) = 0, \quad n = 0, 1, \ldots, \tag{9}$$

for some $k \leq \nu$. Thus, solving this equation for the vector $\mathbf{x}$ gives us an approximation of it denoted $\mathbf{y}_k^{(n)}$ since it depends on $k$ and $n$.

For that purpose, one has to find a procedure for computing the unknown coefficients $a_0^{(n)}, \ldots, a_k^{(n)}$, taking into account that they are numbers while $\mathbf{x} - \mathbf{x}_{n+i}$ are vectors. All such transformations can be considered as vector generalizations of *Shanks' transformation* [22] for sequences of numbers. The common idea behind these transformations is to obtain a system of linear equations whose solution is $a_0^{(n)}, \ldots, a_k^{(n)}$, and then to compute $\mathbf{y}_k^{(n)} \simeq \mathbf{x}$.

It turns out that several vector sequence transformations based on (or including) such a kernel already exist and have been studied by various authors: the $\varepsilon$-algorithms [23–25], the Minimal Polynomial Extrapolation (MPE) [26], the Modified Minimal Polynomial Extrapolation (MMPE) [23,27,28], the Reduced Rank Extrapolation (RRE) [29,30], Germain-Bonne transformations [31], the $H$-algorithm [32], and the $E$-algorithm [33]. They are described and analyzed, for example, in [23,28,33–39].

Writing (9) for the indexes $n$ and $n+1$, subtracting, and multiplying scalarly by a vector $\mathbf{y}$ leads to the scalar equation

$$a_0^{(n)}(\mathbf{y}, \Delta\mathbf{x}_n) + a_1^{(n)}(\mathbf{y}, \Delta\mathbf{x}_{n+1}) + \cdots + a_k^{(n)}(\mathbf{y}, \Delta\mathbf{x}_{n+k}) = 0, \tag{10}$$

where $\Delta$ is the difference operator defined by $\Delta\mathbf{x}_n = \mathbf{x}_{n+1} - \mathbf{x}_n$.

As explained in [9, p. 39], there are several possible strategies for constructing our system that, apart from an additional normalization condition, must contain $k$ equations:

- Use only one vector $\mathbf{y}$, and write (10) for the indexes $n, \ldots, n+k-1$;
- Write (10) only for the index $n$, and choose $k$ linearly independent vectors $\mathbf{y}$;
- Write several relations (10), and choose several linearly independent vectors $\mathbf{y}$.

Adding to these $k$ equations the condition $a_0^{(n)} + \cdots + a_k^{(n)} = 1$, which does not restrict the generality (and is needed since the sum of the coefficients must be nonzero in order for $\mathbf{x}$ in (9) to be uniquely defined), we obtain a system of $k+1$ equations in the $k+1$ unknowns $a_0^{(n)}, \ldots, a_k^{(n)}$ (which depend on $n$ and $k$). Its coefficients are denoted $d_{i,j}^{(n)}$, and for any of the preceding strategies this system is

$$\begin{cases} a_0^{(n)} + \cdots + a_k^{(n)} = 1, \\ d_{i,0}^{(n)} a_0^{(n)} + \cdots + d_{i,k}^{(n)} a_k^{(n)} = 0, \qquad i = 1, \ldots, k, \end{cases} \tag{11}$$

where the coefficients $d_{i,j}^{(n)}$, for $i = 1, \ldots, k$ and $j = 0, \ldots, k$, will be given below according to the chosen strategy. Then, for a fixed value of $k$, our sequence transformation $(\mathbf{x}_n) \longmapsto (\mathbf{y}_k^{(n)})$ is defined by

$$\mathbf{y}_k^{(n)} = a_0^{(n)}\mathbf{x}_n + \cdots + a_k^{(n)}\mathbf{x}_{n+k}, \quad n = 0, 1, \ldots, \tag{12}$$

which can be written as

$$\mathbf{y}_k^{(n)} = \frac{\begin{vmatrix} \mathbf{x}_n & \cdots & \mathbf{x}_{n+k} \\ d_{1,0}^{(n)} & \cdots & d_{1,k}^{(n)} \\ \vdots & & \vdots \\ d_{k,0}^{(n)} & \cdots & d_{k,k}^{(n)} \end{vmatrix}}{\begin{vmatrix} 1 & \cdots & 1 \\ d_{1,0}^{(n)} & \cdots & d_{1,k}^{(n)} \\ \vdots & & \vdots \\ d_{k,0}^{(n)} & \cdots & d_{k,k}^{(n)} \end{vmatrix}} = \frac{\begin{vmatrix} \mathbf{x}_n & \Delta\mathbf{x}_n & \cdots & \Delta\mathbf{x}_{n+k-1} \\ d_{1,0}^{(n)} & \delta d_{1,0}^{(n)} & \cdots & \delta d_{1,k-1}^{(n)} \\ \vdots & & \vdots & \\ d_{k,0}^{(n)} & \delta d_{k,0}^{(n)} & \cdots & \delta d_{k,k-1}^{(n)} \end{vmatrix}}{\begin{vmatrix} \delta d_{1,0}^{(n)} & \cdots & \delta d_{1,k-1}^{(n)} \\ \vdots & & \vdots \\ \delta d_{k,0}^{(n)} & \cdots & \delta d_{k,k-1}^{(n)} \end{vmatrix}}, \tag{13}$$

where $\delta$ is the difference operator defined by $\delta d_{i,j}^{(n)} = d_{i,j+1}^{(n)} - d_{i,j}^{(n)}$. The second determinantal expression shows that the vector $\mathbf{y}_k^{(n)}$ is the Schur complement [9]:

$$\mathbf{y}_k^{(n)} = \mathbf{x}_n - \left[\Delta\mathbf{x}_n, \ldots, \Delta\mathbf{x}_{n+k-1}\right] \begin{pmatrix} \delta d_{1,0}^{(n)} & \cdots & \delta d_{1,k-1}^{(n)} \\ \vdots & & \vdots \\ \delta d_{k,0}^{(n)} & \cdots & \delta d_{k,k-1}^{(n)} \end{pmatrix}^{-1} \begin{pmatrix} d_{1,0}^{(n)} \\ \vdots \\ d_{k,0}^{(n)} \end{pmatrix}.$$

However, $\mathbf{y}_k^{(n)}$ is not a projection.

The second determinantal formula in (13) means that all our sequence transformations can also be written as

$$\mathbf{y}_k^{(n)} = \mathbf{x}_n - \alpha_1^{(n)}\Delta\mathbf{x}_n - \cdots - \alpha_k^{(n)}\Delta\mathbf{x}_{n+k-1}, \quad k = 0, 1, \ldots, \tag{14}$$

where the $\alpha_i^{(n)}$ are the solution of the linear system

$$\delta d_{i,0}^{(n)}\alpha_1^{(n)} + \cdots + \delta d_{i,k-1}^{(n)}\alpha_k^{(n)} = d_{i,0}^{(n)}, \quad i = 1, \ldots, k. \tag{15}$$

Let us now specify various choices of the coefficients $d_{i,j}^{(n)}$, for $i = 1, \ldots, k$ and $j = 0, \ldots, k$.

### *3.2.1 Vector Shanks' transformations*

We first consider sequence transformations that are directly inspired by the scalar sequence transformation of Shanks [22] and can be recursively implemented by Wynn's $\varepsilon$-algorithm [24] or its generalizations, or by the $E$-algorithm [33], or the $H$-algorithm [32].

- Choosing $d_{i,j}^{(n)} = (\mathbf{e}_p, \Delta\mathbf{x}_{n+i+j-1})$ and replacing $\mathbf{x}_j$ in the first row of the numerator of the first ratio in (13) by $(\mathbf{e}_p, \mathbf{x}_j)$ corresponds to Shanks' transformation applied componentwise to the vector sequence $(\mathbf{x}_n)$, and it gives the $p$th component of the vector $\mathbf{y}_k^{(n)}$. This transformation can be recursively implemented, separately for each component $p = 1, \ldots, N$, by the scalar $\varepsilon$-algorithm of Wynn [24].
- The $\varepsilon$-algorithm can also be applied directly to a sequence of vectors after defining the inverse $\mathbf{u}^{-1}$ of a vector $\mathbf{u}$ as $\mathbf{u}^{-1} = \mathbf{u}/(\mathbf{u}, \mathbf{u})$. This vector $\varepsilon$-algorithm was also proposed by Wynn [25]. However, for this algorithm, (13) is no longer valid, and the determinants have to be replaced either by bigger and more complicated

ones [40] or by designants that generalize them in a noncommutative algebra [41,42]. Thus, in this case, an underlying system of linear equations for $\mathbf{y}_k^{(n)}$ does not exist, and this transformation has to be recursively implemented by the vector $\varepsilon$-algorithm whose rules are, for a sequence $(\mathbf{u}_n)$ of real (to simplify) vectors in $\mathbb{R}^N$,

$$\boldsymbol{\varepsilon}_{-1}^{(n)} = 0 \in \mathbb{R}^N, \quad \boldsymbol{\varepsilon}_0^{(n)} = \mathbf{u}_n \in \mathbb{R}^N, \quad n = 0, 1, \ldots$$

$$\boldsymbol{\varepsilon}_{k+1}^{(n)} = \boldsymbol{\varepsilon}_{k-1}^{(n+1)} + \left(\boldsymbol{\varepsilon}_k^{(n+1)} - \boldsymbol{\varepsilon}_k^{(n)}\right)^{-1}, \quad k, n = 0, 1, \ldots$$

- The choice $d_{i,j}^{(n)} = (\mathbf{y}, \Delta\mathbf{x}_{n+i+j-1})$, where $\mathbf{y}$ is any nonzero vector so that the denominators of (13) differ from zero, leads to the topological Shanks transformation introduced in [23]. It can be implemented via the topological $\varepsilon$-algorithm. In this algorithm, the inverses are defined in a different way for an even or an odd lower index by

$$\left(\boldsymbol{\varepsilon}_{2k}^{(n+1)} - \boldsymbol{\varepsilon}_{2k}^{(n)}\right)^{-1} = \mathbf{y} \Big/ \left(\mathbf{y}, \boldsymbol{\varepsilon}_{2k}^{(n+1)} - \boldsymbol{\varepsilon}_{2k}^{(n)}\right),$$

$$\left(\boldsymbol{\varepsilon}_{2k+1}^{(n+1)} - \boldsymbol{\varepsilon}_{2k+1}^{(n)}\right)^{-1} = \left(\boldsymbol{\varepsilon}_{2k}^{(n+1)} - \boldsymbol{\varepsilon}_{2k}^{(n)}\right) \Big/ \left(\boldsymbol{\varepsilon}_{2k+1}^{(n+1)} - \boldsymbol{\varepsilon}_{2k+1}^{(n)}, \boldsymbol{\varepsilon}_{2k}^{(n+1)} - \boldsymbol{\varepsilon}_{2k}^{(n)}\right).$$

In the preceding $\varepsilon$-algorithms, only those vectors (or numbers) with an even lower index $\boldsymbol{\varepsilon}_{2k}^{(n)}$ are interesting for the purpose of convergence acceleration, those with an odd lower index $\boldsymbol{\varepsilon}_{2k+1}^{(n)}$ being intermediate computations. Applying any of them to the sequence $(\boldsymbol{\varepsilon}_0^{(n)} = \mathbf{x}_n)$ produced by Kaczmarz's method gives $\boldsymbol{\varepsilon}_{2k}^{(n)} = \mathbf{y}_k^{(n)}$. Notice that the computation of one vector $\boldsymbol{\varepsilon}_{2k}^{(n)}$ requires the $2k+1$ vectors $\mathbf{x}_n, \ldots, \mathbf{x}_{n+2k}$. Thus, when $k$ is increased by one, the computation of each new vector $\boldsymbol{\varepsilon}_{2k}^{(n)}$ needs two additional iterates of Kaczmarz's method, while it only requires one when $n$ increases.

### *3.2.2 MPE, MMPE, and RRE*

The following choices also lead to known transformations, but in this case, the computation of $\mathbf{y}_k^{(n)}$ only requires the $k+2$ vectors $\mathbf{x}_n, \ldots, \mathbf{x}_{n+k+1}$. Algorithms for their recursive implementation also exist [32,43,44].

- The choice $d_{i,j}^{(n)} = (\mathbf{y}_i, \Delta\mathbf{x}_{n+j})$, for $i = 1, \ldots, k$ and $j = 0, \ldots, k$, corresponds to the MMPE [23,27,28], where the $\mathbf{y}_i$ are linearly independent vectors. This transformation is another generalization of the topological Shanks transformation also given in [23].
- If we take $d_{i,j}^{(n)} = (\Delta\mathbf{x}_{n+i-1}, \Delta\mathbf{x}_{n+j})$ for $i = 1, \ldots, k$ and $j = 0, \ldots, k$, we obtain the MPE [26]. This method is mathematically equivalent to a transformation due to Germain-Bonne [31]. Other choices are proposed in the same reference.
- The choice $d_{i,j}^{(n)} = (\Delta^2\mathbf{x}_{n+i-1}, \Delta\mathbf{x}_{n+j})$ leads to the RRE [29,30].

It must be noted that the vector $\mathbf{y}_k^{(n)}$ always exists for the RRE but not for the MPE or the MMPE. Existence conditions are discussed in [45] and [46].

### *3.2.3 The simplest transformations*

For $k = 1$, the transformations can be compared with those given in [2]. They have the following very simple forms:

- The MMPE and the topological Shanks' transformation become

$$\mathbf{y}_1^{(n)} = \mathbf{x}_n - \frac{(\mathbf{y}_1, \Delta\mathbf{x}_n)}{(\mathbf{y}_1, \Delta^2\mathbf{x}_n)}\Delta\mathbf{x}_n.$$

- For the MPE and the transformation due to Germain-Bonne, we have

$$\mathbf{y}_1^{(n)} = \mathbf{x}_n - \frac{(\Delta\mathbf{x}_n, \Delta\mathbf{x}_n)}{(\Delta\mathbf{x}_n, \Delta^2\mathbf{x}_n)}\Delta\mathbf{x}_n.$$

- For the RRE, it holds that

$$\mathbf{y}_1^{(n)} = \mathbf{x}_n - \frac{(\Delta^2\mathbf{x}_n, \Delta\mathbf{x}_n)}{(\Delta^2\mathbf{x}_n, \Delta^2\mathbf{x}_n)}\Delta\mathbf{x}_n,$$

  which is the same as that proposed in [3].
- For the vector $\varepsilon$-algorithm we obtain

$$\mathbf{y}_1^{(n)} = \mathbf{x}_{n+1} + \frac{\boldsymbol{\varepsilon}_1^{(n+1)} - \boldsymbol{\varepsilon}_1^{(n)}}{\left(\boldsymbol{\varepsilon}_1^{(n+1)} - \boldsymbol{\varepsilon}_1^{(n)}, \boldsymbol{\varepsilon}_1^{(n+1)} - \boldsymbol{\varepsilon}_1^{(n)}\right)} \quad \text{with } \boldsymbol{\varepsilon}_1^{(n)} = \frac{\Delta\mathbf{x}_n}{(\Delta\mathbf{x}_n, \Delta\mathbf{x}_n)}.$$

*3.2.4 Orthogonality properties*

Using the determinantal formula (13), we have orthogonality properties as given in the following proposition.

**Proposition 3** *The vector* $Q(\mathbf{y}_k^{(n)} - \mathbf{x}) - (\mathbf{y}_k^{(n)} - \mathbf{x})$ *is orthogonal to*

- $\Delta\mathbf{x}_n, \ldots, \Delta\mathbf{x}_{n+k-1}$ *for the MPE (which is identical to the Germain-Bonne transformation);*
- $\mathbf{y}$ *for the topological Shanks transformation (topological* $\varepsilon$*-algorithm);*
- $\Delta^2\mathbf{x}_n, \ldots, \Delta^2\mathbf{x}_{n+k-1}$ *for the RRE;*
- $\mathbf{y}_1, \ldots, \mathbf{y}_k$ *for the MMPE.*

For a discussion of the properties of these algorithms and their application to the solution of systems of linear and nonlinear equations, see [45].

## 4 Acceleration of Kaczmarz's method

Although they are different, all the acceleration methods presented in Sect. 3 have the same kernel, namely, sequences satisfying (9). Thus, when applied to any sequence of vectors generated by iterations of the form $\mathbf{x}_{n+1} = B\mathbf{x}_n + \mathbf{c}$, $n = 0, 1, \ldots$, we obtain $\mathbf{y}_\nu^{(n)} = (I - B)^{-1}\mathbf{c}$ for all $n$, where $\nu$ is the degree of the minimal polynomial of $B$ for the vector $\mathbf{x} - \mathbf{x}_0$. This result means that these methods are direct methods for solving systems of linear equations [23]. Therefore, this property holds for the iterates of Kaczmarz's method since $\Pi_\nu(1) \neq 0$ (otherwise the matrix $I - Q$ would be singular). Of course, in practice, these algorithms cannot be used to obtain the exact solution when the dimension of the system is large since, for computing $\mathbf{y}_\nu^{(n)}$, they require the storage of too many vectors.

### 4.1 AK and RK algorithms

The preceding algorithms can be used in two different ways to accelerate the iterations (2) produced by Kaczmarz's method, as will now be explained. Since the computation of $\mathbf{y}_k^{(n)}$ does not require the same number of vectors issued from Kaczmarz's method, according to the transformation used, in order to have a unified presentation, we will denote by $\ell + 1$ the number of vectors $\mathbf{x}_i$ needed to compute $\mathbf{y}_k^{(n)}$ by any of the preceding procedures. Remember that $\ell$ depends on $k$ and that $\ell = 2k$ for the $\varepsilon$-algorithms and $\ell = k + 1$ for the other algorithms.

- The first strategy consists in applying one of the algorithms implementing a sequence transformation on the sequence $\mathbf{x}_0, \mathbf{x}_1, \ldots$ given by Kaczmarz's method and, after fixing the index $k$, in building simultaneously the sequence $\mathbf{z}_0 = \mathbf{y}_k^{(0)}, \mathbf{z}_1 = \mathbf{y}_k^{(1)}, \ldots$. The computation of $\mathbf{y}_k^{(0)}$ can only begin after the iterate $\mathbf{x}_\ell$ has been computed, but the computation of each new transformed vector needs only one new iterate of Kaczmarz's method. This procedure is called the *accelerated Kaczmarz* (AK) algorithm. Let us give the general structure for the implementation of the AK algorithm.

**Accelerated Kaczmarz (AK) algorithm**

**Require** $A \in \mathbb{R}^{N\times N}$, $\mathbf{b} \in \mathbb{R}^N$, $\mathbf{x}_0 \in \mathbb{R}^N$
**Choose** $k \in \mathbb{N}, k \geq 1$
**Set** $\ell = k+1$ or $\ell = 2k$
**for** $n = 0, 1, \ldots$ until convergence **do**
  $\mathbf{p}_0 \leftarrow \mathbf{x}_n$
  **Compute** $\mathbf{p}_i, \quad i = 1, \ldots, N$
  $\mathbf{x}_{n+1} \leftarrow \mathbf{p}_N$
  **If** $n \geq \ell - 1$ **then**
    **Compute** $\mathbf{y}_k^{(n-\ell+1)}$
    $\mathbf{z}_{n-\ell+1} \leftarrow \mathbf{y}_k^{(n-\ell+1)}$
  **end if**
**end for** $n$

- In the second strategy, we set $\mathbf{x}_0$, and we compute $\mathbf{x}_1, \ldots, \mathbf{x}_\ell$ by Kaczmarz's method; we apply one of the algorithms implementing a sequence transformation on them, and we obtain $\mathbf{z}_0 = \mathbf{y}_k^{(0)}$. Then we restart Kaczmarz's method from $\mathbf{z}_0$, that is, we set $\mathbf{x}_0 = \mathbf{z}_0 = \mathbf{y}_k^{(0)}$, we compute the new $\ell$ vectors $\mathbf{x}_1, \ldots, \mathbf{x}_\ell$ by Kaczmarz's method, we apply again the acceleration algorithm to these vectors $\mathbf{x}_0, \ldots, \mathbf{x}_\ell$, we obtain $\mathbf{z}_1 = \mathbf{y}_k^{(0)}$, we restart Kaczmarz's method from $\mathbf{x}_0 = \mathbf{z}_1$, and so on. This second procedure is called the *restarted Kaczmarz* (RK) algorithm, and it can be implemented as follows.

**Restarted Kaczmarz (RK) algorithm**

**Require** $A \in \mathbb{R}^{N\times N}$, $\mathbf{b} \in \mathbb{R}^N$, $\mathbf{x}_0 \in \mathbb{R}^N$
**Choose** $k \in \mathbb{N}, k \geq 1$
**Set** $\ell = k+1$ or $\ell = 2k$
**for** $n = 0, 1, \ldots$ until convergence **do**
  **for** $j = 0, \ldots, \ell - 1$ **do**
    $\mathbf{p}_0 \leftarrow \mathbf{x}_j$
    **Compute** $\mathbf{p}_i, \quad i = 1, \ldots, N$
    $\mathbf{x}_{j+1} \leftarrow \mathbf{p}_N$
  **end for** $j$
  **Compute** $\mathbf{y}_k^{(0)}$
  $\mathbf{z}_n \leftarrow \mathbf{y}_k^{(0)}$
  $\mathbf{x}_0 \leftarrow \mathbf{z}_n$
**end for** $n$

### 4.2 The case of the $\varepsilon$-algorithms

Let us now consider the particular case of the $\varepsilon$-algorithms. If we assume that the eigenvalues of the matrix $Q$, defined in Sect. 2.1, are numbered such that $|\tau_1| > |\tau_2| > \cdots > |\tau_N| > 0$, then the sequences $(\boldsymbol{\varepsilon}_{2k}^{(n)})$, for $k \leq \nu$ fixed, constructed by these algorithms are such that [37,39,47]

$$\left\|\mathbf{x}-\boldsymbol{\varepsilon}_{2k}^{(n)}\right\| = \mathcal{O}\left(\tau_{k+1}^{n}\right), \quad n \to \infty.$$

If the preceding assumptions on the $\tau_i$ are not satisfied (which corresponds to several eigenvalues of $Q$ having the same modulus or to a defective matrix $Q$), the expression for $\mathbf{x}-\mathbf{x}_n$ is more complicated than that given above [38] (see [21] for the complete expression in the scalar case), but the convergence is still accelerated by the $\varepsilon$-algorithms. These results also hold for the topological $\varepsilon$-algorithm, independently of the choice of the arbitrary vector $\mathbf{y}$ occurring in this algorithm. Thus, each sequence $(\boldsymbol{\varepsilon}_{2k}^{(n)})$ obtained by the AK algorithm converges to $\mathbf{x}$ faster than the preceding sequence $(\boldsymbol{\varepsilon}_{2k-2}^{(n)})$ when $n$ tends to infinity. Quite similar results hold for the RRE and the MPE [46].

If the topological $\varepsilon$-algorithm is applied to a sequence $(\mathbf{x}_n)$ obtained by any iterative method of the form $\mathbf{x}_{n+1} = B\mathbf{x}_n + \mathbf{c}$, with the choice $\mathbf{y} = \mathbf{r}_0 = \mathbf{b} - A\mathbf{x}_0$, then the vectors $\boldsymbol{\varepsilon}_{2k}^{(0)}, k = 0, 1, \ldots$, are identical to those obtained by Lanczos' method (that is, for example, by the biconjugate algorithm or the conjugate gradient algorithm in the symmetric positive-definite case) [48, Theorems 4.1 and 4.2, pp. 186–187]. This property derives from the determinantal expressions of the vectors produced by the topological $\varepsilon$-algorithm and by Lanczos' method. From (7) we immediately see that this is the case of the vectors given by Kaczmarz's method that correspond to $B = Q$ and $\mathbf{c} = A^{-1}(I - P)\mathbf{b}$. Thus, if the topological $\varepsilon$-algorithm is applied, with $\mathbf{y} = \mathbf{r}_0$, to the sequence $(\mathbf{x}_n)$ given by Kaczmarz's method, then the sequence $(\boldsymbol{\varepsilon}_{2k}^{(0)})$ is exactly the sequence produced by Lanczos' method. Using it as explained in the RK algorithm corresponds to restarting Kaczmarz's method with the vector obtained by the Lanczos' method after $k$ of its iterates. At each restart, the vector $\mathbf{y}$ has to be taken as equal to the corresponding residual.

## 5 Implementation

The implementation of our convergence acceleration procedures can be realized in three different ways. The first one consists, for a fixed value of $k$, in solving the linear system (11) and using (12). The second way is to employ (14) and system (15). The third possibility is to apply, when it exists, a recursive algorithm. Since, for the vector $\varepsilon$-algorithm, no underlying system of linear equations is known, its implementation can only be realized through its recursive rules as given in Sect. 3.2.1.

An important practical problem is the choice of $k$. On one hand, the effectiveness of our procedures seems to increase with $k$, but on the other hand, the number of vectors to be stored also increases with it. Thus, if the system is large, then the value of $k$ has to be kept quite small since too many vectors will have to be stored. The numerical stability of the procedures must also be taken into consideration when $k$ increases. Since Kaczmarz's method often converges slowly, the quantities $\Delta\mathbf{x}_{n+i}$ are small, and the elements of the linear systems to be solved for obtaining the coefficients of the linear combination giving $\mathbf{y}_k^{(n)}$ approach zero. Thus, whatever the method used, the linear systems (11) and (15) are ill-conditioned even for small values of $k$, and rounding errors can degrade the results. We tried several ways of computing their solution, but the influence on the vectors $\mathbf{y}_k^{(n)}$ was small.

Instead of solving the systems, it is possible to use a recursive algorithm for the computation of the vectors $\mathbf{y}_k^{(n)}$. Usually, sequence transformations are more efficiently implemented via a recursive algorithm like those mentioned earlier, and in our case, this kind of approach seems to give results that are less sensitive to rounding errors for the topological and the vector $\varepsilon$-algorithms.

The quantities computed by all the recursive algorithms for implementing our transformations can be displayed in a triangular array (or a lozenge array for the $\varepsilon$-algorithms), and $\mathbf{y}_k^{(n)}$ corresponds to the lowest rightmost element of the array. Its computation forces one to compute first all the preceding elements in the array. Thus, it could be much too expensive to store all these vectors if $k$ is not quite small and if $N$ is large. Hopefully it is possible to proceed in the array by ascending diagonals (that is, starting from $\mathbf{y}_0^{(n+k)} = \mathbf{x}_{n+k}$ to compute $\mathbf{y}_1^{(n+k-1)}, \ldots, \mathbf{y}_k^{(n)}$) and, thus, storing only one ascending diagonal. For more details on such a technique, see [1].

Another problem is the choice of the arbitrary vector $\mathbf{y}$ in the topological Shanks transformation and of the vectors $\mathbf{y}_1, \ldots, \mathbf{y}_k$ appearing in the MMPE (see [45] for a discussion of this choice). This is an unsolved problem, and further studies will be necessary.

In the literature, the system $A\mathbf{x} = \mathbf{b}$ to be solved by Kaczmarz's method is often replaced by the system $DA\mathbf{x} = D\mathbf{b}$, where $D = \text{diag}(1/\|\mathbf{a}_i\|)$. Thus, each row vector of the new matrix $DA$ will have a norm equal to one. This preconditioning, of course, simplifies the computation of $\mathbf{p}_i$ in (3), and we will use it.

As a last comment, notice that, although some transformations are equivalent from a theoretical point of view (for instance, the MPE and the Germain-Bonne transformation, or the MMPE and the $H$-algorithm), they do not produce exactly the same numerical results. The same is true when we compare the results of a transformation implemented in the different ways described in this paper.

## 6 Numerical results

Our numerical experiments were performed using MATLAB 7.11 and with matrices from the `gallery` set. The solution was set to $\mathbf{x} = (1, \ldots, 1)^{\mathrm{T}}$, and the right-hand side $\mathbf{b}$ was computed accordingly. We took random vectors with components uniformly distributed in $[-1, +1]$ for the arbitrary vector $\mathbf{y}$ in the topological $\varepsilon$-algorithm and for the vectors $\mathbf{y}_i$ in the MMPE.

In some figures, we show the Euclidean norms of the errors (in log scale). To compare the acceleration brought by each procedure, we give, in some other figures, the ratios of the norms of the errors (in log scale) between the iterate $\mathbf{z}_n$ obtained by the AK or the RK algorithm and the iterate of Kaczmarz's method with the highest index used in its construction (for AK) or the iterate with the highest index that would have been used had we let the method continue without restarting it (for RK), that is,

$$\frac{\|\mathbf{z}_n - \mathbf{x}\|}{\|\mathbf{x}_{n+\ell} - \mathbf{x}\|} \text{ (AK)} \quad \text{and} \quad \frac{\|\mathbf{z}_n - \mathbf{x}\|}{\|\mathbf{x}_{(n+1)(\ell+1)} - \mathbf{x}\|} \text{ (RK)}.$$

We consider the `parter` matrix $A$, $N = 1000$, $\kappa(A) \simeq 4.2306$, a Toeplitz matrix with singular values near $\pi$. In Figs. 1 and 2, we compare Kaczmarz's method with the MMPE, MPE, RRE and the topological $\varepsilon$-algorithm, implemented by solving system (11), and the vector $\varepsilon$-algorithm, respectively, for the AK and RK algorithms, with $k = 5$.

In these figures, we see that all methods achieve a good precision with an advantage for the vector $\varepsilon$-algorithm. Moreover, its convergence is smoother. The ratios increase because all methods almost stagnate when a good precision is attained while the error of Kaczmarz's method continues to decrease slowly. In particular, for the RK algorithm, the vector $\varepsilon$-algorithm attains its full precision after four iterations while the AK algorithm needs more iterations. For this example, the dominant eigenvalue of $A$ is 0.8732178, and the second one is 0.3170877. Thus, according to the theoretical results of Sect. 4.2, a good acceleration is observed even with $k = 1$ for all procedures.

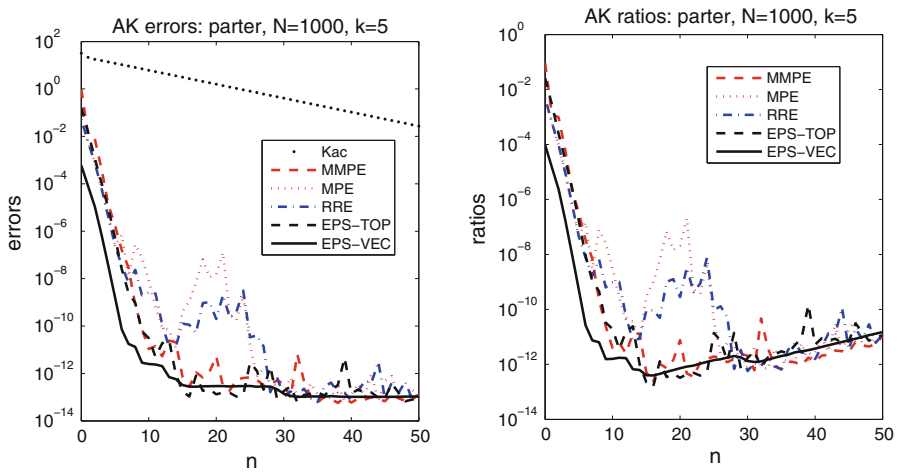


**Fig. 1** AK algorithm: errors and ratios for `parter` matrix, $N = 1000$, $k = 5$

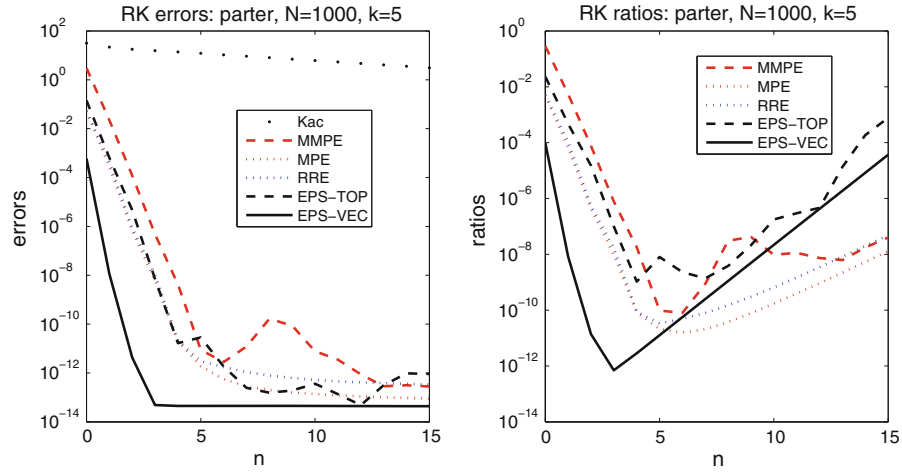


**Fig. 2** RK algorithm: errors and ratios for `parter` matrix, $N = 1000,\ k = 5$

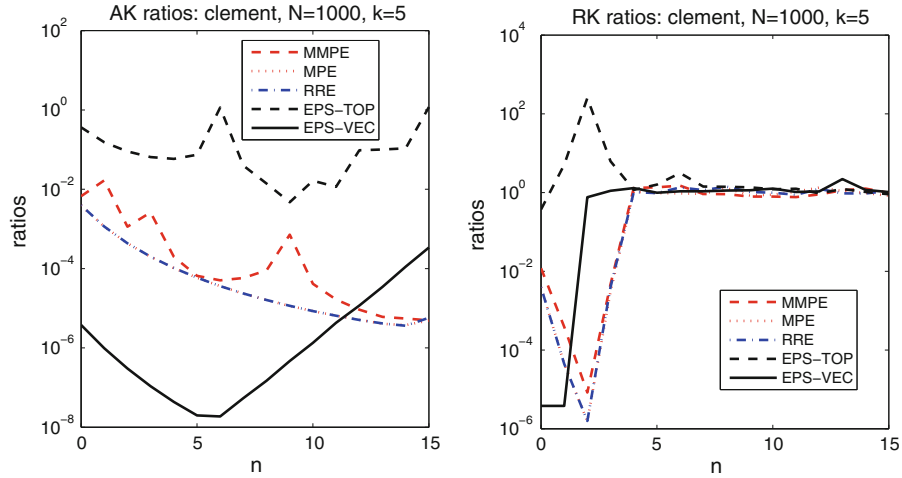


**Fig. 3** AK and RK algorithms: ratios for `clement` matrix, $N = 1000,\ k = 5$

Figure 3 shows the ratios for the `clement` matrix, $\kappa(A) \simeq 1.13145 \times 10^{83}$, a tridiagonal matrix with zero diagonal entries, again with $N = 1000$ and $k = 5$. For this example, the MPE and the RRE coincide. Again the vector $\varepsilon$-algorithm is the best.

For `toeppen`, a pentadiagonal Toeplitz matrix, with $N = 1000,\ k = 5$, we have Fig. 4. Notice that the MMPE and the topological $\varepsilon$-algorithm do not work well and that this behavior could be due to the choice of the vectors $\mathbf{y}$ and $\mathbf{y}_i$. In fact, these choices can affect the results in a quite serious way. For instance, these methods sometimes exhibit better convergence and acceleration with $\mathbf{y} = (1, \ldots, 1)^{\mathrm{T}}$ and $\mathbf{y}_i = \mathbf{e}_i$. With the RK algorithm, considering the first 50 iterations for $k = 8$, the vector $\varepsilon$-algorithm attains a ratio of $10^{-9}$ at iteration 20, and after 24 iterations a division by zero occurs. The ratios for the RRE and the MPE have a minimum of $10^{-10}$ at iteration 25. The topological $\varepsilon$-algorithm diverges from the beginning. The MMPE exhibits an erratic convergence, and the ratio goes down to $10^{-7}$ at iteration 42 and then increases.

We also tried our procedures on a larger matrix. The results for `lesp`, a tridiagonal matrix with real, sensitive eigenvalues, $N = 10000,\ k = 5,\ \kappa(A) \simeq 6.9553 \times 10^3$, with the AK algorithm are given in Fig. 5. We see that Kaczmarz's method and the acceleration procedures all attain full precision after 90 iterations. Thus all

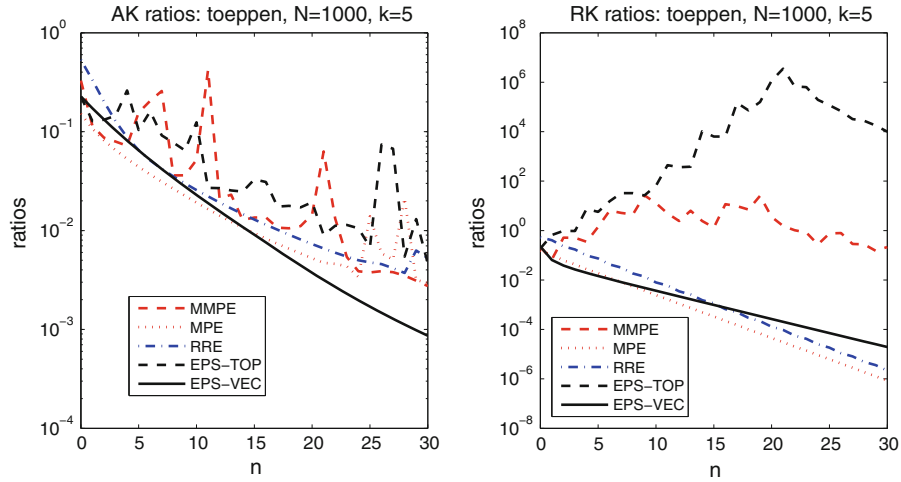


**Fig. 4** AK and RK algorithms: ratios for `toeppen` matrix, $N = 1000,\ k = 5$

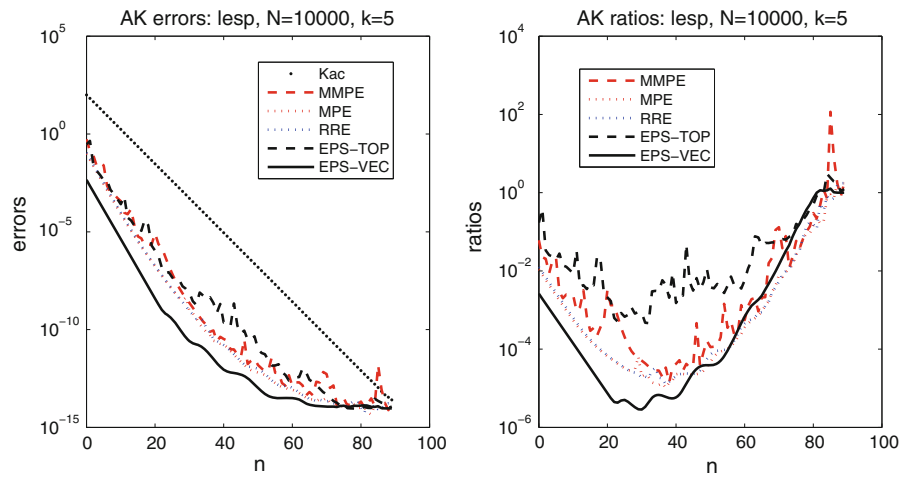


**Fig. 5** AK algorithm: errors and ratios for `lesp` matrix, $N = 10000,\ k = 5$

ratios increase. However, the vector $\varepsilon$-algorithm attains an error of less than $10^{-11}$ after about 20 iterations, while Kaczmarz's method has only an error of order $10^{-3}$ at the same iteration.

An important point in any iterative method is to have a quite reliable stopping criterion. Usually such methods are stopped by using the residual. However, its computation requires a matrix–vector product that, in our case, removes one of interests of the method. Thus, since the results given by our acceleration procedures often stagnate when some precision is attained while those of Kaczmarz continue to decrease, the iterations could be stopped as soon as the following ratios increase significantly:

$$\frac{\|\Delta \mathbf{z}_n\|}{\|\Delta \mathbf{x}_{n+\ell}\|}\ \text{(AK)} \quad \text{and} \quad \frac{\|\Delta \mathbf{z}_n\|}{\left\|\Delta \mathbf{x}_{(n+1)(\ell+1)}\right\|}\ \text{(RK)}.$$

An example with the vector $\varepsilon$-algorithm is given in Fig. 6 (left: AK algorithm, right: RK algorithm). The solid line corresponds to the norm of the error and the dashed line to the ratio for the stopping criterion.

However, it must be noted that, in the case of the RK algorithm, this stopping criterion involves iterates of Kaczmarz's method that have not yet been computed and used in the acceleration procedures. Thus, it is not usable in practice. Since the quantity $\|\Delta \mathbf{z}_n\|$ usually decreases rapidly, the iterates can be stopped when it is small enough and begins to stagnate or even to increase.

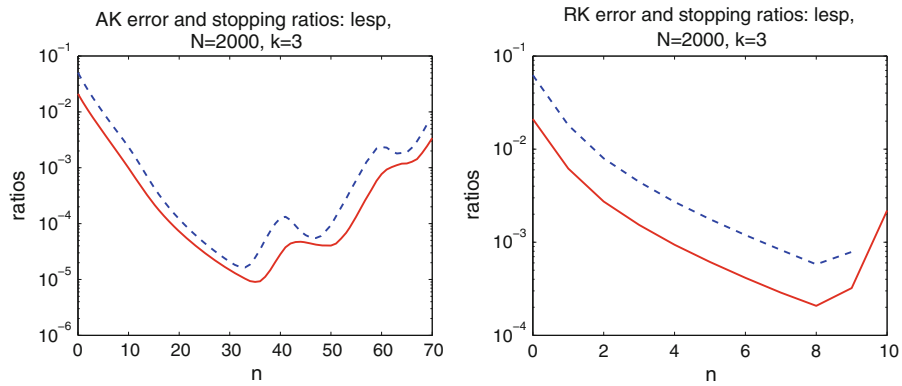


**Fig. 6** AK (*left*) and RK (*right*) algorithms: errors (*solid*) and stopping (*dashed*) ratios for `lesp` matrix, $N = 2{,}000,\ k = 3$

In our examples, we also add a white noise between $10^{-2}$ and $10^{-8}$ to the vector $\mathbf{b}$. The norm of the error of the results obtained by our acceleration procedures attains the level of the noise in most cases.

For the matrix `baart` of the MATLAB Regularization toolbox [49] of dimension 120 whose condition number is $2.28705 \times 10^{18}$, an error of the form $\mathbf{e} = \delta \|\mathbf{b}\| \mathbf{u}/\sqrt{N}$, where $\delta$ is between $10^{-2}$ and $10^{-8}$ and $\mathbf{u}$ is a vector whose components are random variables from a normal distribution with mean 0 and standard deviation 1, was added to $\mathbf{b}$. The norm of the error achieved with the vector $\varepsilon$-algorithm diminishes to $10^{-0.75}$ for $k = 1, 2$, and 3, which is slightly better than the results obtained in [50]. Let us mention that our stopping criterion based on the ratios only works correctly for small noises, perhaps because the vector $\mathbf{b}$ is not involved in it.

In all our examples, the norm of the residuals behaves quite similarly to the norm of the errors, even in the case where the matrix is ill-conditioned. However, in some examples, the curves are shifted by a factor almost equal to the condition number of the matrix $A$, but their shapes are roughly the same. This is due to the relations (5) and the fact that the matrices $P$ and $Q$ are similar. The additional computing time for the implementation of all our acceleration procedures is negligible compared to the time required by Kaczmarz's method itself.

## 7 Conclusion

From our numerical results, it seems that the vector $\varepsilon$-algorithm is the best procedure for accelerating Kaczmarz's method. However, the recursive implementation of the other procedures has to be tested numerically to see if it leads to better and more stable results. In our examples, we tried several values of $k$. Although, when the dominant eigenvalue of $A$ is well separated, $k = 1$ leads to quite a good acceleration, it seems that higher values produce better results in general. Anyway, the choice of $k$ and that of the vectors $\mathbf{y}$ and $\mathbf{y}_i$ are important points that need to be studied. Other recursive algorithms, such as those developed by Germain-Bonne [31], or the Vector $\theta$-type Transform (VTT) and the Biorthogonal Vector $\theta$-type Transform (BVTT) [43], not considered in this work, must also be tested. The case of inconsistent linear systems must also be treated.

The acceleration of the symmetric Kaczmarz and the randomized Kaczmarz methods (which are also sequential row-action methods that update a solution using one row of $A$ at each step), of other methods of the MAP class, and of the Simultaneous Iterative Reconstruction Techniques (SIRT) methods, has to be considered. Finally, applications to tomography and, in general, to image reconstruction must be considered.

**Acknowledgments** We would like to thank Zhong-Zhi Bai for a discussion of Meany's inequality, Andrezj Cegielski for interesting exchanges on the terminology for the various ways of using Kaczmarz's method, and Paolo Novati for consulting on the numerical examples. This work was partially supported by the University of Padova, Project 2010 No. CPDA104492.

## References


1. Brezinski C, Redivo-Zaglia M (1991) Extrapolation methods: theory and practice. North-Holland, Amsterdam
2. Escalante R, Raydan M (2011) Alternating projection methods. SIAM, Philadelphia
3. Gearhart WB, Koshy M (1989) Acceleration schemes for the method of alternating projections. J Comput Appl Math 26:235–249
4. Cegielski A (2012) Iterative methods for fixed point problems in Hilbert spaces. Lecture notes in mathematics, vol 2057. Springer, Heidelberg
5. Kaczmarz S (1937) Angenäherte Auflösung von Systemen linearer Gleichungen. Bull Acad Pol Sci A35:355–357 (English translation: Approximate solution of systems of linear equations. Int J Control 57:1269–1271, 1993)
6. Gordon R, Bender R, Herman GT (1970) Algebraic reconstruction techniques (ART) for three-dimensional electron microscopy and X-ray photography. J Theor Biol 29:471–481
7. Björck Å, Elfving T (1979) Accelerated projection methods for computing pseudoinverse solutions of systems of linear equations. BIT 19:145–163
8. Bramley R, Sameh A (1992) Row projection methods for large nonsymmetric linear systems. SIAM J Sci Stat Comput 13:168–193
9. Brezinski C (1997) Projection methods for systems of equations. North-Holland, Amsterdam
10. Tanabe K (1971) Projection method for solving a singular system of linear equations and its applications. Numer Math 17:203–214
11. Cegielski A (2010) A list of publications on the Kaczmarz method. http://www.wmie.uz.zgora.pl/~acegiels/Publikacje-Kaczmarz.pdf
12. Bai Z-Z, Jin C-H (2003) Column-decomposed relaxation methods for the overdetermined systems of linear equations. Intern J Appl Math 13:71–82
13. Altman M (1957) On the approximate solution of linear algebraic equations. Bull Pol Acad Sci Math Cl III 3:365–370
14. Durand É (1961) Solution numérique des Équations algébriques, vol 2. Masson, Paris
15. Gastinel N (1970) Numerical linear algebra. Academic Press, New York (English translation of Analyse Numérique Linéaire, Hermann, Paris, 1966)
16. Householder AS, Bauer FL (1960) On certain iterative methods for solving linear systems. Numer Math 2:55–59
17. Gastinel N (1958) Procédé itératif pour la résolution numérique d'un système d'équations linéaires. C R Acad Sci Paris 246:2571–2574
18. Galántai A (2004) Projectors and projection methods. Kluwer Academic Publishers, Dordrecht
19. Meany R (1969) A matrix inequality. SIAM J Numer Anal 6:104–107
20. Bai Z-Z, Liu X-G (2013) On the Meany inequality with applications to convergence analysis of several row-action iteration methods. Numer Math 124:215–236
21. Brezinski C, Crouzeix M (1970) Remarques sur le procédé $\Delta^2$ d'Aitken. C R Acad Sci Paris 270 A:896–898
22. Shanks D (1955) Non linear transformations of divergent and slowly convergent sequences. J Math Phys 34:1–42
23. Brezinski C (1975) Généralisation de la transformation de Shanks, de la table de Padé et de l'$\varepsilon$-algorithme. Calcolo 12:317–360
24. Wynn P (1956) On a device for computing the $e_m(S_n)$ transformation. MTAC 10:91–96
25. Wynn P (1962) Acceleration techniques for iterated vector and matrix problems. Math Comput 16:301–322
26. Cabay S, Jackson LW (1976) A polynomial extrapolation method for finding limits and antilimits of vector sequences. SIAM J Numer Anal 13:734–752
27. Pugachev BP (1978) Acceleration of convergence of iterative processes and a method of solving systems of non-linear equations. USSR Comput Maths Maths Phys 17(5):199–207
28. Sidi A, Ford WF, Smith DA (1986) Acceleration of convergence of vector sequences. SIAM J Numer Anal 23:178–196
29. Eddy RP (1979) Extrapolation to the limit of a vector sequence. In: Wang PCC (ed) Information linkage between applied mathematics and industry. Academic Press, New York, pp 387–396
30. Mešina M (1977) Convergence acceleration for the iterative solution of $x = Ax + f$. Comput Methods Appl Mech Eng 10:165–173
31. Germain-Bonne B (1978) Estimation de la Limite de Suites et Formalisation de Procédés d'Accélération de la Convergence. Université des Sciences et Techniques de Lille, Thèse d'Etat
32. Brezinski C, Sadok H (1987) Vector sequence transformations and fixed point methods. In: Taylor C et al (eds) Numerical methods in laminar and turbulent flows. Pineridge Press, Swansea, pp 3–11
33. Brezinski C (1980) A general extrapolation algorithm. Numer Math 35:175–187
34. Brezinski C (1974) Some results in the theory of the vector $\varepsilon$-algorithm. Linear Algbera Appl 8:77–86
35. Brezinski C (1979) Sur le calcul de certains rapports de déterminants. In: Wuytack L (ed) Padé approximation and its applications. Lecture notes in mathematics, vol 765. Springer, Heidelberg, pp 184–210
36. Sidi A (1986) Convergence and stability properties of minimal polynomial and reduced rank extrapolation. SIAM J Numer Anal 23:197–209
37. Sidi A (1988) Extrapolation vs. projection methods for linear systems of equations. J Comput Appl Math 22:71–88
38. Sidi A, Bridger J (1988) Convergence and stability analyses for some vector extrapolation methods in the presence of defective iteration matrices. J Comput Appl Math 22:35–61
39. Smith DA, Ford WF, Sidi A (1987) Extrapolation methods for vector sequences. SIAM Rev 29:199–233
40. Graves-Morris PR, Jenkins CD (1986) Vector-valued rational interpolants III. Constr Approx 2:263–289
41. Salam A (1994) Non-commutative extrapolation algorithms. Numer Algorithms 7:225–251

42. Salam A (1996) An algebraic approach to the vector $\varepsilon$-algorithm. Numer Algorithms 11:327–337
43. Brezinski C, Redivo-Zaglia M (1996) Vector and matrix sequence transformations based on biorthogonality. Appl Numer Math 21:353–373
44. Ford WF, Sidi A (1988) Recursive algorithms for vector extrapolation methods. Appl Numer Math 4:477–489
45. Jbilou K, Sadok H (1999) LU implementation of the minimal polynomial extrapolation method for solving linear and nonlinear systems. IMA J Numer Anal 19:549–561
46. Sidi A (1991) Efficient implementation of minimal polynomial and reduced rank extrapolation methods. J Comput Appl Math 36:305–337
47. Brezinski C (1975) Computation of the eigenelements of a matrix by the $\varepsilon$-algorithm. Linear Algbera Appl 11:7–20
48. Brezinski C (1980) Padé-type approximation and general orthogonal polynomials, ISNM, vol 50. Birkhäuser, Basel
49. Hansen PC (1994) Regularization tools: a Matlab package for analysis and solution of discrete ill-posed problems. Numer Algorithms 6:1–35
50. Brezinski C, Redivo-Zaglia M, Novati P (2012) A rational Arnoldi approach for ill-conditioned linear systems. J Comput Appl Math 236:2063–2077