IMA Journal of Numerical Analysis (2021) **00**, 1–36
https://doi.org/10.1093/imanum/drab061

Shanks and Anderson-type acceleration techniques for systems of nonlinear equations

Claude Brezinski*
Université de Lille, CNRS, UMR 8524 - Laboratoire Paul Painlevé, F-59000 Lille, France
*Corresponding author: claude.brezinski@univ-lille.fr

Stefano Cipolla and Michela Redivo-Zaglia
Università degli Studi di Padova, Dipartimento di Matematica "Tullio Levi-Civita", Via Trieste 63, 35121–Padova, Italy

and

Yousef Saad
Dept. of Computer Science and Engineering, University of Minnesota, Mississippi National River and Recreation Area, Minneapolis, MN 55455, USA

[Received on 3 July 2020; revised on 1 July 2021; accepted on 7 July 2021]

This paper examines a number of extrapolation and acceleration methods and introduces a few modifications of the standard Shanks transformation that deal with general sequences. One of the goals of the paper is to lay out a general framework that encompasses most of the known acceleration strategies. The paper also considers the Anderson Acceleration (AA) method under a new light and exploits a connection with quasi-Newton methods in order to establish local linear convergence results of a stabilized version of the AA method. The methods are tested on a number of problems, including a few that arise from nonlinear partial differential equations.

Keywords: extrapolation methods; Anderson acceleration; quasi-Newton methods; Krylov subspace methods; regularization; nonlinear Poisson problems; Navier–Stokes equation.

1. Introduction

In numerical analysis and in applied mathematics, many applications lead to sequences of numbers, vectors, matrices or even tensors. When the sequence is slowly converging, or even diverging, and when one has only access to the sequence and nothing else (i.e., when it is produced by a 'black box'), it is possible to transform it, by a *sequence transformation*, into a new sequence, which, under some assumptions, converges faster than the original one to the same limit. It was necessary to develop a variety of such sequence transformations since, in fact, it was proved by Delahaye & Germain-Bonne (1980) that a universal sequence transformation able to accelerate all sequences, or even all monotonically converging scalar ones, cannot exist. For a review, see, for example, Brezinski & Redivo-Zaglia (1991, 2019, 2020), Delahaye (1988), Sidi (2016), Weniger (1989), Wimp (1981).

One way to transform a sequence into a faster converging one is to resort to *extrapolation*. Here, the transformation is built so that it yields the exact limit of all sequences satisfying a certain algebraic relation. The set of these sequences is called the *kernel* of the transformation. Among these, this paper focuses on the *Shanks transformation* (Shanks, 1955) and a number of its generalizations. As we will

see, this well-established method transforms a sequence $(\mathbf{s}_n)$ into a set of sequences $\{(\mathbf{t}_n^{(k)})\}$. Introduced by Shanks for scalar sequences (Shanks, 1955), it has been extensively studied and extended, in various ways, to sequences of vectors, matrices and tensors. Here, we only consider the vector case.

All these extensions to vectors of the scalar Shanks transformation share the property that, for a fixed value of k, $\mathbf{t}_n^{(k)} = \mathbf{s}$ for all n if the sequence $(\mathbf{s}_n)$ of elements of $\mathbb{R}^p$ or $\mathbb{C}^p$ satisfies, for all n, the following linear difference equation of order k:

$$\alpha_0(\mathbf{s}_n - \mathbf{s}) + \cdots + \alpha_k(\mathbf{s}_{n+k} - \mathbf{s}) = 0, \tag{1.1}$$

where $\mathbf{s}$ is the limit of $(\mathbf{s}_n)$ if it converges and is called its *antilimit* otherwise. The numbers α_i are independent of n, and it is assumed that $\alpha_0\alpha_k \neq 0$, so that the difference equation has order k exactly, and $\alpha_0 + \cdots + \alpha_k \neq 0$, so that $\mathbf{s}$ can be uniquely defined. Thus, these conditions imply that k cannot be replaced by a smaller value. It does not restrict the generality to assume that $\alpha_0 + \cdots + \alpha_k = 1$. The set of sequences satisfying (1.1) is called the *Shanks kernel*. Among sequences in this kernel are those produced by the iterations $\mathbf{s}_{n+1} = M\mathbf{s}_n + \mathbf{b}$, thus providing a link with Krylov subspace and Lanczos methods; see, in particular, Brezinski (1974, 1980), Sidi (1988), Sidi & Bridger (1988).

Besides their use in a number of different applications, extrapolation techniques have recently been promoted as an effective tool also for problems related to the emerging field of data science (Zhang *et al.*, 2020; Cipolla *et al.*, 2020a,b; Scieur *et al.*, 2020). But since there is often some confusion in the literature about the terminology used, we would like clarify it—using a high level of generality. Specifically, we would like to draw a distinction between *extrapolation methods*, *sequence transformations* and *convergence acceleration methods*. This distinction will help the reader to better understand the approaches described in Section 2 for building our sequence transformations.

Let $(\mathbf{s}_n)$ be a sequence of elements of a vector space E on $\mathbb{C}$. A common problem encountered in numerical analysis is to estimate the limit of this sequence from a certain number of its terms. The problem can be solved by an *extrapolation method* as follows (Brezinski, 1971; Brezinski & Redivo-Zaglia, 2021). Let

$$\boldsymbol{\varphi} : \mathbb{N} \times D \longmapsto E, \qquad D \subseteq \mathbb{C}^k$$

be such that

$$\text{for all } \mathbf{b} \in D, \quad \lim_{n\to\infty} \boldsymbol{\varphi}(n, \mathbf{b}) = 0.$$

Let $V_{\boldsymbol{\varphi}}$ be the linear variety of sequences of elements of E such that

$$\text{for all } n, \quad \mathbf{s}_n = \mathbf{s} + \boldsymbol{\varphi}(n, \mathbf{b}),$$

with $\mathbf{s} \in E$. Obviously, $\lim_{n\to\infty} \mathbf{s}_n = \mathbf{s}$.

By definition, if $(\mathbf{s}_n) \in V_{\boldsymbol{\varphi}}$, then, for all n, $\mathbf{s} = \mathbf{s}_n - \boldsymbol{\varphi}(n, \mathbf{b})$. Now, if $(\mathbf{s}_n) \notin V_{\boldsymbol{\varphi}}$, let us consider a sequence $(\mathbf{t}_n = \mathbf{t}+\boldsymbol{\varphi}(n, \boldsymbol{\beta})) \in V_{\boldsymbol{\varphi}}$, and impose that it satisfies the *interpolation conditions* $\mathbf{t}_{n+i} = \mathbf{s}_{n+i}$ for $i = 0, \ldots, k$. The vector $\boldsymbol{\beta} \in D$ can be computed, assuming that it exists and is unique, in different ways as the solution of a system of k scalar equations that can be obtained as follows. Let E^* be the algebraic dual vector space of E, which is the vector space of linear functionals on E. Let $\mathbf{y}, \mathbf{y}_1, \ldots, \mathbf{y}_k \in E^*$, and

let $\langle \cdot, \cdot \rangle$ denote the duality product between E^* and E. The first strategy consists in computing the vector $\boldsymbol{\beta}$ as the solution of the system

$$\langle \mathbf{y}_i, \mathbf{s}_{n+1} - \mathbf{s}_n \rangle = \langle \mathbf{y}_i, \boldsymbol{\varphi}(n+1, \boldsymbol{\beta}) - \boldsymbol{\varphi}(n, \boldsymbol{\beta}) \rangle, \quad i = 1, \ldots, k.$$

In the particular case of Shanks transformation, writing this system in matrix form leads to a relation having the same structure as *Approach 3* in the *minimal residual approach* of Section 2.1.3, but with different indexes.

In the second strategy, the vector $\boldsymbol{\beta}$ is the solution of the system

$$\langle \mathbf{y}, \mathbf{s}_{n+i+1} - \mathbf{s}_{n+i} \rangle = \langle \mathbf{y}, \boldsymbol{\varphi}(n+i+1, \boldsymbol{\beta}) - \boldsymbol{\varphi}(n+i, \boldsymbol{\beta}) \rangle, \quad i = 0, \ldots, k-1.$$

For Shanks transformation, this approach corresponds, in matrix form, to something similar to *Approach 6* in the *topological approach* of Section 2.2.3.

Then, in both cases, we set $\mathbf{t} = \mathbf{s}_n - \boldsymbol{\varphi}(n, \boldsymbol{\beta})$. Since $\mathbf{t} = \lim_{n\to\infty} \mathbf{t}_n$, it is an approximation of $\mathbf{s}$ and it has been obtained by *extrapolation*. Obviously, $\mathbf{t}$ depends on n and k, and we will now denote it by $\mathbf{t}_n^{(p_k)}$ where $p_k + 1$ denotes the number of elements of the initial sequence used in the process. Thus, when n and p_k vary, the sequence $(\mathbf{s}_n)$ has been transformed into the set of sequences $\{(\mathbf{t}_n^{(p_k)})\}$. This procedure is named an *extrapolation method*. An important remark to be made is that it is a purely algebraic procedure. Richardson's and Romberg's methods and Aitken's Δ^2 process are such well-known scalar extrapolation methods. Thus, an extrapolation method results in a *sequence transformation* $T : (\mathbf{s}_n) \longmapsto (\mathbf{t}_n^{(p_k)})$ when either p_k or n is fixed, and the other index tends to infinity. Conversely, most sequence transformations can be interpreted as extrapolation methods. The variety $V_{\boldsymbol{\varphi}}$ is usually named the *kernel* of the transformation T, and it is denoted $\mathcal{K}_T$. If, when n or p_k tends to infinity, the sequence $(\mathbf{t}_n^{(p_k)})$ converges to $\mathbf{s}$ faster than the sequence $(\mathbf{s}_n)$, the denomination *convergence acceleration method* is also used. Let us mention that extrapolation methods can also be applied to diverging sequences. They are often used for accelerating fixed-point iterations, sometimes coupled with a restarting strategy.

In this paper, instead of building Shanks transformation by computing the coefficients in (1.1) as the solution of a linear system in the usual way, we propose a new optimization approach, based on minimization. This allows us to easily introduce, for sequences not belonging to the Shanks kernel, a unified framework that also includes regularized and preconditioned techniques.

Anderson acceleration (AA) (Anderson, 1965, 2019), also called Anderson mixing, Pulay mixing or direct inversion in the iterative subspace (DIIS) (Pulay, 1980), in the computational physics and chemistry communities, has been widely used and applied to the solution of various fixed-point problems over the last decades. The literature on this method is too broad to allow for an exhaustive discussion but it suffices to search recent citations of this work to understand the truly exceptional renewed interest in AA across many disciplines. A few of the classical citations include the papers by Walker & Ni (2011), Higham & Strabić (2016), Toth & Kelley (2015) and Fang & Saad (2009), and a few papers that describe applications are Banerjee *et al.* (2016), Fu *et al.* (2020), Kelley (2018), Lupo Pasini (2019), Ouyang *et al.* (2020), Pollock *et al.* (2019), Zhang *et al.* (2020).

However, it is important here to stress that AA is not an extrapolation method in the exact sense defined above since it does not start from an arbitrary given sequence and transform it into a new sequence. Instead, it builds its own sequence step by step. AA is in fact more akin to quasi-Newton techniques than to extrapolation. It was viewed as a form of secant method in the classic book by

Ortega & Rheinboldt (1970, pp. 204–205). Its relations to secant-type methods, specifically 'multi-secant methods' was unraveled by Eyert (1996), and later exploited in Fang & Saad (2009) and also in Fu *et al.* (2020). In short, Anderson–Pulay mixing is a second-order method whose goal is to accelerate a fixed-point iteration. If we were to allow the number of preview iterates used in the process to increase indefinitely, we would get something rather similar to a standard quasi-Newton method whose convergence would be superlinear at the limit. This is not done in practice because of cost and numerical stability considerations. However, a certain relation with the Reduced Rank Extrapolation (RRE) method, which is an extrapolation method, exists, and AA can be recovered by using the coupled Shanks transformations, as explained in Brezinski *et al.* (2018). Due to this connection, we give, in Section 5, new procedures in the style of AA, that are called Anderson-Type Mixing (ATM in short). Stabilized and regularized versions of AA will be also proposed.

The outline of the paper is as follows:

- In Section 2 we present an overview of transformation techniques for sequences belonging to the Shanks kernel and show how their limit or antilimit can be obtained exactly from these transformations. Four out of six of these techniques are presented in a new way that comes out from an optimization problem. Coupled sequences used in Section 5 are also described.
- In Section 3 we present transformations based on the Shanks kernel. We show how to adapt and extend the idea proposed in Scieur *et al.* (2020) to our approaches. These modifications are specifically designed to accelerate general/nonlinear sequences, which do not belong to the Shanks kernel.
- In Section 4 we present the *Restarted* and the *Continuous-Updating* (CU) methods for exploiting the Shanks-based transformations presented in the previous section. In this way we are able to introduce a unified framework able to encompass simultaneously the newly introduced transformations and many of transformations already present in the literature.
- In Section 5 we present new ATM methods. We show how the classical AA fits into them. Then we introduce preconditioning and regularization strategies. Moreover, exploiting the connection with quasi-Newton methods, we prove the local linear convergence of a *stabilized* version of the classical AA, which allows us to substantiate theoretically the regularization strategy encompassed in the Anderson-type techniques previously presented in this section.
- In Section 6 we perform a comparative experimental study of some of the techniques proposed using, among other tests, a set of nonlinear problems arising from partial differential equations (PDEs).

Let us explain our notation. Given a sequence $(\mathbf{s}_n)$, we set $S_i^{(j)} = [\mathbf{s}_i, \ldots, \mathbf{s}_{i+j-1}] \in \mathbb{R}^{p\times j}$. Thus, the superscript j corresponds to the number of columns formed by the p-dimensional vectors of the sequence $(\mathbf{s}_n)$ and the lower index i is the index of the first of these vectors in the sequence. Whenever it is used, the forward difference operator Δ is applied to the lower index, that is, $\Delta S_i^{(j)} = S_{i+1}^{(j)} - S_i^{(j)} = [\Delta\mathbf{s}_i, \ldots, \Delta\mathbf{s}_{i+j-1}]$, and similarly for Δ^2. For a fixed value of k, we denote by $\overline{S}_i^{(j)}$ the $kp \times j$ matrix formed by stacking the k matrices $S_i^{(j)}, \ldots, S_{i+k-1}^{(j)}$ of dimension $p \times j$. When not explicitly indicated, the norm used is the Euclidean norm. Throughout the paper, if not explicitly indicated, all matrices whose inverses are needed are assumed to be nonsingular. If it is not the case, the pseudo-inverse may be used.

2. Transformations for sequences in the Shanks kernel

Let $(\mathbf{s}_n)$ be a sequence of vectors in $\mathbb{R}^p$ or $\mathbb{C}^p$ such that (1.1) holds for a fixed value of k and for all n. Assuming, without loss of generality, that $\sum_{i=0}^{k} \alpha_i = 1$, then we get from (1.1),

$$\alpha_0 \mathbf{s}_n + \cdots + \alpha_k \mathbf{s}_{n+k} = \mathbf{s} \qquad \text{for all } n \geq 0. \tag{2.1}$$

Alternatively, we can write

$$\mathbf{s}_{n+k} - \sum_{j=0}^{k-1} \beta_j \Delta \mathbf{s}_{n+j} = \mathbf{s}, \tag{2.2}$$

with $\beta_j = \sum_{i=0}^{j} \alpha_i$ for $j = 0, \ldots, k-1$ (note that the β_i are defined in a slightly different way than in Brezinski *et al.*, 2018, Sect. 3.1.3).

In Sections 2.1 and 2.2, we show that when $(\mathbf{s}_n)$ belongs to the Shanks kernel for a fixed value of k, it is possible to compute exactly the limit or the antilimit of the sequence from a certain number ℓ_k (which depends on k and on the transformation used) of consecutive vectors of the sequence, where $\ell_k = k+2$ (for the minimal residual approaches) or $\ell_k = 2k+1$ (for the topological approaches). For this purpose, we present six different strategies for computing the coefficients $\boldsymbol{\alpha} = (\alpha_0, \ldots, \alpha_k)^{\mathrm{T}}$ or $\boldsymbol{\beta} = (\beta_0, \ldots, \beta_{k-1})^{\mathrm{T}}$. It should be recalled that $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are not dependent on n if $(\mathbf{s}_n)$ satisfies (1.1) or (2.1) or (2.2). Four of these strategies (Approaches 1, 2, 4 and 5 below) are presented as the solution of a minimization problem. Approaches 1 and 4 proceed in what appears to be a new way, not considered before in the literature devoted to Shanks sequence transformations. Approaches 2 and 5 can be considered as particular cases of the least-squares strategy evoked in Brezinski *et al.* (2018, Sect. 3.1.3). These four strategies will be useful for the generalization presented in Section 3. Two of these strategies (Approaches 3 and 6 below) are already known since they enter into the framework of extrapolation methods as explained in Section 1, and are derived in Sections 2.1 and 2.2 by a purely algebraic process as the solution of a linear system and they can be easily obtained by a modification of Approaches 2 and 5. Moreover, as will be explained in Section 3, these two strategies could also be included in the framework of the minimization by changing the metric of the norm. Approaches 3 and 6 will be used in Section 2.3, where the notion of *coupled sequence*, defined in Brezinski *et al.* (2018), is invoked.

Let us explain the idea behind the minimization used for finding the vector $\boldsymbol{\alpha}$ (since $\boldsymbol{\beta}$ is related to $\boldsymbol{\alpha}$, the idea is similar). This idea was introduced in Scieur *et al.* (2020), but it was not related to Shanks transformations. In Section 4 and the following ones, our transformations are used to solve the fixed-point problem $\mathbf{s} = G(\mathbf{s})$ from iterates of the form $\mathbf{s}_{n+1} = G(\mathbf{s}_n)$. Under some assumptions, it holds that $\mathbf{s}_n - \mathbf{s} = (G'(\mathbf{s}))^n(\mathbf{s}_0 - \mathbf{s}) + \mathcal{O}(\|\mathbf{s}_0 - \mathbf{s}\|^2)$. Thus, neglecting the terms of second order,

$$\sum_{i=0}^{k} \alpha_i \mathbf{s}_{n+i} - \mathbf{s} \approx \big(G'(\mathbf{s})\big)^n \sum_{i=0}^{k} \alpha_i (G'(\mathbf{s}))^i (\mathbf{s}_0 - \mathbf{s}).$$

The idea is to minimize this error term. But $\Delta\mathbf{s}_n \approx (G'(\mathbf{s}) - I)(\mathbf{s}_n - \mathbf{s})$, and thus

$$\sum_{i=0}^{k} \alpha_i \Delta\mathbf{s}_{n+i} \approx (G'(\mathbf{s}) - I)(G'(\mathbf{s}))^n \sum_{i=0}^{k} \alpha_i (G'(\mathbf{s}))^i (\mathbf{s}_0 - \mathbf{s}),$$

which is similar to the expressions minimized for obtaining the vector $\boldsymbol{\alpha}$ in Approaches 1 and 4 below.

When $\boldsymbol{\alpha}$ or $\boldsymbol{\beta}$ has been computed, in any one of the ways described below, the vector $\mathbf{s}$ is directly obtained by (2.1) or (2.2) as

$$\mathbf{s} = [\mathbf{s}_{n+i}, \ldots, \mathbf{s}_{n+i+k}]\boldsymbol{\alpha} = S_{n+i}^{(k+1)}\boldsymbol{\alpha} \qquad \text{for all } i, \tag{2.3}$$

or

$$\mathbf{s} = \mathbf{s}_{n+i+k} - [\Delta\mathbf{s}_{n+i}, \ldots, \Delta\mathbf{s}_{n+i+k-1}]\boldsymbol{\beta} = \mathbf{s}_{n+i+k} - \Delta S_{n+i}^{(k)}\boldsymbol{\beta} \qquad \text{for all } i. \tag{2.4}$$

REMARK 2.1 As can be seen, (2.4) has the form of a Schur complement

$$\mathbf{u} = \mathbf{u}_0 - [\mathbf{u}_1, \ldots, \mathbf{u}_k]A^{-1}\mathbf{v},$$

where $\mathbf{u}, \mathbf{u}_0, \mathbf{u}_1, \ldots, \mathbf{u}_k \in \mathbb{R}^p$, $\mathbf{v} \in \mathbb{R}^k$ and $A \in \mathbb{R}^{k\times k}$. Several other expressions in the sequel have the same form.

From the extended Schur determinantal formula (Brezinski, 1988), $\mathbf{u}$ can be expressed as the ratio of two determinants

$$\mathbf{u} = \frac{\begin{vmatrix} \mathbf{u}_0 & \mathbf{u}_1 \cdots \mathbf{u}_k \\ \mathbf{v} & A \end{vmatrix}}{|A|}.$$

The determinant in the numerator is to be understood as the linear combination of the elements of its first row by applying the classical rules for expanding a determinant with respect its first row. It is exactly through this connection that all the transformations given in Brezinski *et al.* (2018) (least-squares strategy apart) have been defined.

2.1 *Minimal residual approaches*

All the minimal residual approaches described in this section for computing $\boldsymbol{\alpha}$ or $\boldsymbol{\beta}$ require the knowledge of the $k + 2$ vectors $\mathbf{s}_n, \ldots, \mathbf{s}_{n+k+1}$.

2.1.1 *Approach* 1.

Writing (2.1) for the indices n and $n + 1$ and subtracting, we obtain

$$\alpha_0\Delta\mathbf{s}_n + \cdots + \alpha_k\Delta\mathbf{s}_{n+k} = 0.$$

Then one way to compute $\boldsymbol{\alpha} = (\alpha_0, \ldots, \alpha_k)^{\mathrm{T}}$ is to solve the problem

$$\boldsymbol{\alpha} = \underset{\boldsymbol{\gamma}\in\mathbb{R}^{k+1}, \mathbf{e}^{\mathrm{T}}\boldsymbol{\gamma}=1}{\arg\min} \|\Delta S_n^{(k+1)}\boldsymbol{\gamma}\|^2, \tag{2.5}$$

where $\mathbf{e}$ is the vector of all 1s. This is exactly the same relation introduced in Scieur *et al.* (2020), but obtained from a different starting point and without regularization. The original paper by Pulay (1980) also solves the least squares problem with the same constraint that the sum of the α_i equals 1 by using Lagrange multipliers.

Observe that equation (2.1) and the minimality of k ensure that $\dim\ker(\Delta S_n^{(k+1)}) = 1$. Hence, the solution of problem (2.5) can be obtained by normalizing the unique vector in the kernel; alternatively, it can also be obtained as follows (which leads to the SVD-MPE approach; see Sidi, 2003):

$$\boldsymbol{\alpha} = \frac{\overline{\boldsymbol{\alpha}}}{\mathbf{e}^{\mathrm{T}}\overline{\boldsymbol{\alpha}}} \quad \text{where} \quad \overline{\boldsymbol{\alpha}} = \underset{\boldsymbol{\gamma}\in\mathbb{R}^{k+1},\|\boldsymbol{\gamma}\|^2=1}{\arg\min} \|\Delta S_n^{(k+1)}\boldsymbol{\gamma}\|^2. \tag{2.6}$$

2.1.2 *Approach* 2. Writing (2.2) for the indices n and $n+1$ and subtracting, we have

$$\Delta\mathbf{s}_{n+k} - \sum_{j=0}^{k-1} \beta_j[\Delta\mathbf{s}_{n+1+j} - \Delta\mathbf{s}_{n+j}] = \mathbf{0},$$

i.e., in compact form,

$$\Delta\mathbf{s}_{n+k} - \Delta^2 S_n^{(k)}\boldsymbol{\beta} = \mathbf{0}, \tag{2.7}$$

where $\Delta^2 S_n^{(k)} = [\Delta^2\mathbf{s}_n, \ldots, \Delta^2\mathbf{s}_{n+k-1}]$.

The vector $\boldsymbol{\beta}$ is solution of the problem

$$\boldsymbol{\beta} = \underset{\boldsymbol{\eta}\in\mathbb{R}^k}{\arg\min}\|\Delta\mathbf{s}_{n+k} - \Delta^2 S_n^{(k)}\boldsymbol{\eta}\|^2, \tag{2.8}$$

and therefore it can be obtained by solving the normal equations

$$(\Delta^2 S_n^{(k)})^{\mathrm{T}}\Delta^2 S_n^{(k)}\boldsymbol{\beta} = (\Delta^2 S_n^{(k)})^{\mathrm{T}}\Delta\mathbf{s}_{n+k}, \tag{2.9}$$

which leads to the strategy of the RRE due to Eddy (1979) and Meśina (1977).

2.1.3 *Approach* 3. This approach generalizes the one seen in the preceding section. We consider a matrix $Y \in \mathbb{R}^{p\times k}$, where p is the dimension of the vectors of the sequence. If we multiply (2.7) by Y^{T}, it is possible to obtain the β_i by solving the following system that generalizes (2.9), which is obtained when $Y = \Delta^2 S_n^{(k)}$:

$$Y^{\mathrm{T}}\Delta^2 S_n^{(k)}\boldsymbol{\beta} = Y^{\mathrm{T}}\Delta\mathbf{s}_{n+k}, \tag{2.10}$$

assuming that $\mathrm{rank}(Y^{\mathrm{T}}\Delta^2 S_n^{(k)}) = k$.

The best choice of the matrix Y is a difficult problem, which has not been studied yet. However, some experimental results show that an appropriate choice of it can improve the convergence. As shown, for example, in Brezinski *et al.* (2018), particular choices of Y yield several existing extrapolation methods. Thus, the choice $Y = [\mathbf{y}_1, \ldots, \mathbf{y}_k]$, where the $\mathbf{y}_i$ are

k linear independent vectors, corresponds to the Modified Minimal Polynomial Extrapolation (MMPE) of Brezinski (1975) and Pugachev (1978), which can be recursively implemented by the $S\beta$-algorithm of Jbilou & Sadok (1991). The choice $\mathbf{y}_i = \Delta\mathbf{s}_{n+i-1}$ leads to the Minimal Polynomial Extrapolation (MPE) of Cabay & Jackson (1976), and the RRE of Meśina (1977) and Eddy (1979) is recovered with $\mathbf{y}_i = \Delta^2\mathbf{s}_{n+i-1}$.

2.2 *Topological approaches*

These approaches differ from those presented in Section 2.1 in that the algebraic equations for computing the coefficients α_i or β_i require more vectors of the sequence $(\mathbf{s}_n)$, namely they now need to utilize the $2k+1$ vectors $\mathbf{s}_n, \ldots, \mathbf{s}_{n+2k}$.

2.2.1 *Approach* 4.

Writing (2.1) for the indices $n, \ldots, n+k$, and subtracting, we have

$$\alpha_0\Delta\mathbf{s}_{n+i} + \cdots + \alpha_k\Delta\mathbf{s}_{n+k+i} = 0, \qquad \text{for } i = 0, \ldots, k-1,$$

and the coefficients α_i are obtained by solving

$$\boldsymbol{\alpha} = \underset{\boldsymbol{\gamma}\in\mathbb{R}^{k+1}, \mathbf{e}^{\mathrm{T}}\boldsymbol{\gamma}=1}{\arg\min} \|\Delta\overline{S}_n^{(k+1)}\boldsymbol{\gamma}\|^2, \tag{2.11}$$

where

$$\Delta\overline{S}_n^{(k+1)} = \begin{pmatrix} \Delta\mathbf{s}_n & \Delta\mathbf{s}_{n+1} & \cdots & \Delta\mathbf{s}_{n+k} \\ \Delta\mathbf{s}_{n+1} & \Delta\mathbf{s}_{n+2} & \cdots & \Delta\mathbf{s}_{n+k+1} \\ \vdots & \vdots & & \vdots \\ \Delta\mathbf{s}_{n+k-1} & \Delta\mathbf{s}_{n+k} & \cdots & \Delta\mathbf{s}_{n+2k-1} \end{pmatrix} = \begin{pmatrix} \Delta S_n^{(k+1)} \\ \Delta S_{n+1}^{(k+1)} \\ \vdots \\ \Delta S_{n+k-1}^{(k+1)} \end{pmatrix} \in \mathbb{R}^{kp\times(k+1)}.$$

2.2.2 *Approach* 5.

The β_i can be computed by writing (2.2) for the indices $n+k, \ldots, n+2k$, and subtracting. We have

$$\Delta\mathbf{s}_{n+k+i} - \sum_{j=0}^{k-1}\beta_j\Delta^2\mathbf{s}_{n+i+j} = \mathbf{0} \qquad \text{for } i = 0, \ldots, k-1.$$

The coefficients β_i are solutions of the problem

$$\boldsymbol{\beta} = \underset{\boldsymbol{\eta}\in\mathbb{R}^k}{\arg\min}\|\Delta\overline{S}_{n+k}^{(1)} - \Delta^2\overline{S}_n^{(k)}\boldsymbol{\eta}\|^2, \tag{2.12}$$

where

$$\Delta\overline{S}_{n+k}^{(1)} = \begin{pmatrix} \Delta\mathbf{s}_{n+k} \\ \vdots \\ \Delta\mathbf{s}_{n+2k-1} \end{pmatrix} \in \mathbb{R}^{kp}, \quad \Delta^2\overline{S}_n^{(k)} = \begin{pmatrix} \Delta^2\mathbf{s}_n & \Delta^2\mathbf{s}_{n+1} & \cdots & \Delta^2\mathbf{s}_{n+k-1} \\ \Delta^2\mathbf{s}_{n+1} & \Delta^2\mathbf{s}_{n+2} & \cdots & \Delta^2\mathbf{s}_{n+k} \\ \vdots & \vdots & & \vdots \\ \Delta^2\mathbf{s}_{n+k-1} & \Delta^2\mathbf{s}_{n+k} & \cdots & \Delta^2\mathbf{s}_{n+2k-2} \end{pmatrix} \in \mathbb{R}^{kp\times k},$$

that is,

$$\boldsymbol{\beta} = ((\Delta^2\overline{S}_n^{(k)})^{\mathrm{T}}\Delta^2\overline{S}_n^{(k)})^{-1}(\Delta^2\overline{S}_n^{(k)})^{\mathrm{T}}\Delta\overline{S}_{n+k}^{(1)}.$$

2.2.3 *Approach* 6. As in Approach 3, choosing $Y \in \mathbb{R}^{kp\times k}$, we can alternatively solve

$$Y^{\mathrm{T}}\Delta^2\overline{S}_n^{(k)}\boldsymbol{\beta} = Y^{\mathrm{T}}\Delta\overline{S}_{n+k}^{(1)} \tag{2.13}$$

if $\operatorname{rank}(Y^{\mathrm{T}}\Delta^2\overline{S}_n^{(2k-2)}) = k$.

When $Y = I_k \otimes \mathbf{y}$, for some $\mathbf{y} \in \mathbb{R}^p$, we recover the so-called *Topological Shanks Transformation* that can be implemented recursively by the Topological ε-Algorithms of Brezinski (1975) (in short TEA) or, more economically, by the Simplified Topological ε-Algorithms (in short STEA) (Brezinski & Redivo-Zaglia, 2014, 2017).

2.3 *Coupled transformations*

We now recall the concept of *coupled sequences* introduced in Brezinski *et al.* (2018) since, by using this extension, it is possible to link AA to the transformations based on the Shanks kernel.

Given a sequence $(\mathbf{s}_n)$ belonging to the Shanks kernel, a coupled sequence $(\mathbf{c}_n)$ is a sequence that satisfies, for all n,

$$\alpha_0\mathbf{c}_n + \cdots + \alpha_k\mathbf{c}_{n+k} = 0,$$

where the coefficients α_i are the same as in (2.1), or equivalently a sequence satisfying

$$\mathbf{c}_{n+k} - \sum_{j=0}^{k-1}\beta_j\Delta\mathbf{c}_{n+j} = \mathbf{0} \quad \text{for all } n$$

with the same coefficients β_j as in (2.2). For example, the sequence $(\mathbf{c}_n = \Delta^m\mathbf{s}_n)$ is a sequence coupled to $(\mathbf{s}_n)$ for any $m \geq 1$.

By using a known coupled sequence, we can build additional generalizations of Approaches 3 and 6, which are recovered if we take $(\mathbf{c}_n = \Delta\mathbf{s}_n)$, and compute $\boldsymbol{\beta}$ as follows. Let $C_n^{(k)} = [\mathbf{c}_n, \ldots, \mathbf{c}_{n+k-1}] \in \mathbb{R}^{p\times k}$. Instead of (2.10), we solve the system

$$Y^{\mathrm{T}}\Delta C_n^{(k)}\boldsymbol{\beta} = Y^{\mathrm{T}}\mathbf{c}_{n+k}, \tag{2.14}$$

where $Y \in \mathbb{R}^{p\times k}$.

Similarly, by defining the matrix $\overline{C}_i^{(j)}$ as made for $\overline{S}_i^{(j)}$, we can, instead of (2.13), solve

$$Y^{\mathrm{T}}\Delta\overline{C}_n^{(k)}\boldsymbol{\beta} = Y^{\mathrm{T}}\overline{C}_{n+k}^{(1)}, \tag{2.15}$$

where now $Y \in \mathbb{R}^{kp\times k}$,

Particular choices of Y and of the coupled sequence $(\mathbf{c}_n)$ give expressions similar to those of well-known methods (see Brezinski *et al.*, 2018 for more details).

3. Shanks-based transformations for general sequences

The approaches described in the previous section are all equivalent for a sequence belonging to the Shanks kernel and they yield the exact limit or antilimit. It is clear however, that this is an idealistic situation. For extrapolating sequences that do not belong to the Shanks kernel (1.1), we still write down the systems of linear equations or the optimization problems giving the coefficients α_i or β_i (which now depend on k and n), and define a sequence transformation as the same linear combination of terms as above.

In the sequel, for the extrapolated vector, we use the double indexing $\mathbf{t}_i^{(j)}$ that highlights the fact that the transformations require the $j+1$ elements $\mathbf{s}_n, \ldots, \mathbf{s}_{n+j}$ of the sequence, in order to compute $\boldsymbol{\alpha}$ or $\boldsymbol{\beta}$.

Minimal residual: In the case of the minimal residual approaches there are $k+1$ vectors involved in the linear combination. Thus, since to compute $\boldsymbol{\alpha}$ or $\boldsymbol{\beta}$, we need the $k+2$ vectors $\mathbf{s}_n, \ldots, \mathbf{s}_{n+k+1}$, we have only the following two different transformations, with the same $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ (we denote the second transformation with a tilde symbol over the $\mathbf{t}$):

- $\mathbf{t}_n^{(k+1)} = [\mathbf{s}_{n+1}, \ldots, \mathbf{s}_{n+k+1}]\boldsymbol{\alpha} = S_{n+1}^{(k+1)}\boldsymbol{\alpha}$, or equivalently
  $\mathbf{t}_n^{(k+1)} = \mathbf{s}_{n+k+1} - [\Delta\mathbf{s}_{n+1}, \ldots, \Delta\mathbf{s}_{n+k}]\boldsymbol{\beta} = \mathbf{s}_{n+k+1} - \Delta S_{n+1}^{(k)}\boldsymbol{\beta}$,
- $\widetilde{\mathbf{t}}_n^{(k+1)} = [\mathbf{s}_n, \ldots, \mathbf{s}_{n+k}]\boldsymbol{\alpha} = S_n^{(k+1)}\boldsymbol{\alpha}$, or equivalently
  $\widetilde{\mathbf{t}}_n^{(k+1)} = \mathbf{s}_{n+k} - [\Delta\mathbf{s}_n, \ldots, \Delta\mathbf{s}_{n+k-1}]\boldsymbol{\beta} = \mathbf{s}_{n+k} - \Delta S_n^{(k)}\boldsymbol{\beta}$,

where $\boldsymbol{\alpha} \in \mathbb{R}^{k+1}$ solves (2.5) or (2.6) (Approach 1) and $\boldsymbol{\beta} \in \mathbb{R}^k$ solves (2.8) or (2.10) (Approach 2 or 3) or (2.14) (coupled approach).

Topological: In the topological case, there are again $k+1$ vectors involved in the linear combination, but since we need the $2k+1$ vectors $\mathbf{s}_n, \ldots, \mathbf{s}_{n+2k}$ to compute $\boldsymbol{\alpha}$ or $\boldsymbol{\beta}$, we have $k+1$ different transformations (depending on the choice of the vectors used in the linear combination), but with the same $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$, and we have

- $\mathbf{t}_{n,i}^{(2k)} = [\mathbf{s}_{n+i}, \ldots, \mathbf{s}_{n+i+k}]\boldsymbol{\alpha} = S_{n+i}^{(k+1)}\boldsymbol{\alpha}$, for $i = 0, \ldots, k$
  or equivalently
  $\mathbf{t}_{n,i}^{(2k)} = \mathbf{s}_{n+i+k} - [\Delta\mathbf{s}_{n+i}, \ldots, \Delta\mathbf{s}_{n+i+k-1}]\boldsymbol{\beta} = \mathbf{s}_{n+i+k} - \Delta S_{n+i}^{(k)}\boldsymbol{\beta}$, for $i = 0, \ldots, k$,

where $\boldsymbol{\alpha} \in \mathbb{R}^{k+1}$ solves (2.11) (Approach 4) and $\boldsymbol{\beta} \in \mathbb{R}^k$ given by (2.12) or (2.13) (Approach 5 or 6) or (2.15) (coupled approach). Among all the possible linear combinations, it seems more appropriate to use those involving the last available vector of the sequence, which is the transformation with $i = k$ that uses $\mathbf{s}_{n+2k}$. In the sequel, to simplify the notation we will set $\mathbf{t}_n^{(2k)} = \mathbf{t}_{n,k}^{(2k)}$.

Of course, if the sequence belongs to the Shanks kernel, all the preceding transformations are equivalent and give the same result, which is $\mathbf{s}$.

Now let us show how to adapt and extend to our approaches the idea proposed in Scieur *et al.* (2020). All the transformations summarized at the beginning of this section can be used, and the only change deals with the computation of the coefficients α_i or the β_i. In Scieur *et al.* (2020), in order to overcome the problems due to the ill conditioning of problem (2.5) (our Approach 1) the authors consider the following regularized problem for the computation of the α_i in the minimal residual approach:

$$\boldsymbol{\alpha}_\lambda = \underset{\boldsymbol{\gamma}\in\mathbb{R}^{k+1}, \mathbf{e}^{\mathrm{T}}\boldsymbol{\gamma}=1}{\arg\min} \left(\|\Delta S_n^{(k+1)}\boldsymbol{\gamma}\|^2 + \lambda\|\boldsymbol{\gamma}\|^2 \right),$$

with $\lambda \in \mathbb{R}$, and whose solution is (assuming that $\Delta S_n^{(k+1)}$ is of full rank)

$$\boldsymbol{\alpha}_\lambda = \frac{((\Delta S_n^{(k+1)})^{\mathrm{T}} \Delta S_n^{(k+1)} + \lambda I)^{-1}\mathbf{e}}{\mathbf{e}^{\mathrm{T}}((\Delta S_n^{(k+1)})^{\mathrm{T}} \Delta S_n^{(k+1)} + \lambda I)^{-1}\mathbf{e}}.$$

Observe that an alternative approach would be to change the metric in the evaluation of the norm, i.e., instead of using the Euclidean norm, solve the problem

$$\boldsymbol{\alpha}_{M,\lambda} = \underset{\boldsymbol{\gamma}\in\mathbb{R}^{k+1}, \mathbf{e}^{\mathrm{T}}\boldsymbol{\gamma}=1}{\arg\min} \left(\|\Delta S_n^{(k+1)}\boldsymbol{\gamma}\|_M^2 + \lambda \|\boldsymbol{\gamma}\|^2 \right), \tag{3.1}$$

where $\|\mathbf{x}\|_M^2 = (\mathbf{x}, M\mathbf{x})$ and $M \in \mathbb{R}^{p\times p}$ is a positive definite matrix.

In what follows we will need M to be positive semidefinite only instead of positive definite. In this case $\|\cdot\|_M$ is a seminorm but we abuse the terminology by calling it a 'norm'.

With this, we have the following lemma.

LEMMA 3.1 The solution of problem (3.1) is

$$\boldsymbol{\alpha}_{M,\lambda} = \frac{((\Delta S_n^{(k+1)})^{\mathrm{T}} M \Delta S_n^{(k+1)} + \lambda I)^{-1}\mathbf{e}}{\mathbf{e}^{\mathrm{T}}((\Delta S_n^{(k+1)})^{\mathrm{T}} M \Delta S_n^{(k+1)} + \lambda I)^{-1}\mathbf{e}}, \tag{3.2}$$

and the corresponding extrapolated vector is

$$\mathbf{t}_n^{(k+1)} = S_{n+1}^{(k+1)}\boldsymbol{\alpha}_{M,\lambda} \quad \text{or} \quad \widetilde{\mathbf{t}}_n^{(k+1)} = S_n^{(k+1)}\boldsymbol{\alpha}_{M,\lambda}. \tag{3.3}$$

Proof. The result follows by writing problem (3.1) as

$$\boldsymbol{\alpha}_{M,\lambda} = \underset{\boldsymbol{\gamma}\in\mathbb{R}^{k+1}, \mathbf{e}^{\mathrm{T}}\boldsymbol{\gamma}=1}{\arg\min} \left(\boldsymbol{\gamma}^{\mathrm{T}}(\Delta S_n^{(k+1)})^{\mathrm{T}} M \Delta S_n^{(k+1)}\boldsymbol{\gamma} + \lambda \boldsymbol{\gamma}^{\mathrm{T}}\boldsymbol{\gamma} \right),$$

and by applying a technique analogous to that used in Scieur *et al.* (2020). From (2.3) we obtain (3.3). □

Motivated by the equivalence of all the approaches described in Section 2.1 for sequences in the Shanks kernel, we can thus introduce the following problem:

$$\boldsymbol{\beta}_{M,\lambda} = \underset{\boldsymbol{\eta}\in\mathbb{R}^{k}}{\arg\min} \left(\|\Delta \mathbf{s}_{n+k} - \Delta^2 S_n^{(k)}\boldsymbol{\eta}\|_M^2 + \lambda \|\boldsymbol{\eta}\|^2 \right), \tag{3.4}$$

where M is a semipositive definite matrix. Referring to the gradient of the function $g(\boldsymbol{\eta}) = \|\Delta \mathbf{s}_{n+k} - \Delta^2 S_n^{(k)}\boldsymbol{\eta}\|_M^2 + \lambda\|\boldsymbol{\eta}\|^2$, the solution of (3.4) is given by

$$\boldsymbol{\beta}_{M,\lambda} = ((\Delta^2 S_n^{(k)})^{\mathrm{T}} M \Delta^2 S_n^{(k)} + \lambda I)^{-1}(\Delta^2 S_n^{(k)})^{\mathrm{T}} M \Delta \mathbf{s}_{n+k}, \tag{3.5}$$

and hence, the corresponding extrapolated vector is

$$\mathbf{t}_n^{(k+1)} = \mathbf{s}_{n+k+1} - [\Delta \mathbf{s}_{n+1}, \ldots, \Delta \mathbf{s}_{n+k}]\boldsymbol{\beta}_{M,\lambda} \tag{3.6}$$

or

$$\widetilde{\mathbf{t}}_n^{(k+1)} = \mathbf{s}_{n+k} - [\Delta\mathbf{s}_n, \ldots, \Delta\mathbf{s}_{n+k-1}]\boldsymbol{\beta}_{M,\lambda}. \tag{3.7}$$

In particular, if $M = YY^{\mathrm{T}}$ where $Y \in \mathbb{R}^{p\times k}$ is a given matrix and $\lambda = 0$, we have

$$(\Delta^2 S_n^{(k)})^{\mathrm{T}} Y \left(Y^{\mathrm{T}}\Delta^2 S_n^{(k)}\boldsymbol{\beta}_{YY^{\mathrm{T}},0} - Y^{\mathrm{T}}\Delta\mathbf{s}_{n+k}\right) = \mathbf{0}.$$

When $\operatorname{rank}(Y^{\mathrm{T}}\Delta^2 S_n^{(k)}) = k$, we see that Approach 3 (2.10) is a particular case of problem (3.4). As we already observed, different choices of $Y \in \mathbb{R}^{p\times k}$ give rise to different *acceleration* performances for different types of sequences.

Similarly, following the idea of the topological approaches of Section 2.2, we consider the problems

$$\boldsymbol{\alpha}_{M,\lambda} = \underset{\boldsymbol{\gamma}\in\mathbb{R}^{k+1},\mathbf{e}^{\mathrm{T}}\boldsymbol{\gamma}=1}{\arg\min} \left(\|\Delta\overline{S}_n^{(k+1)}\boldsymbol{\gamma}\|_M^2 + \lambda\|\boldsymbol{\gamma}\|^2\right) \tag{3.8}$$

or

$$\boldsymbol{\beta}_{M,\lambda} = \underset{\boldsymbol{\eta}\in\mathbb{R}^{k}}{\arg\min} \left(\|\Delta\overline{S}_{n+k}^{(1)} - \Delta^2\overline{S}_n^{(k)}\boldsymbol{\eta}\|_M^2 + \lambda\|\boldsymbol{\eta}\|^2\right), \tag{3.9}$$

where, in both cases, $M \in \mathbb{R}^{kp\times kp}$ is a semipositive definite matrix. The solution of (3.8) is

$$\boldsymbol{\alpha}_{M,\lambda} = \frac{((\Delta\overline{S}_n^{(k+1)})^{\mathrm{T}}M\Delta\overline{S}_n^{(k+1)} + \lambda I)^{-1}\mathbf{e}}{\mathbf{e}^{\mathrm{T}}((\Delta\overline{S}_n^{(k+1)})^{\mathrm{T}}M\Delta\overline{S}_n^{(k+1)} + \lambda I)^{-1}\mathbf{e}}, \tag{3.10}$$

and the corresponding extrapolated vector is

$$\mathbf{t}_n^{(2k)} = S_{n+k}^{(k+1)}\boldsymbol{\alpha}_{M,\lambda}. \tag{3.11}$$

The solution of problem (3.9) is

$$\boldsymbol{\beta}_{M,\lambda} = ((\Delta^2\overline{S}_n^{(k)})^{\mathrm{T}}M\Delta^2\overline{S}_n^{(k)} + \lambda I)^{-1}(\Delta^2\overline{S}_n^{(k)})^{\mathrm{T}}M\Delta\overline{S}_{n+k}^{(1)}$$

and the corresponding extrapolated vector is

$$\mathbf{t}_n^{(2k)} = \mathbf{s}_{n+2k} - [\Delta\mathbf{s}_{n+k}, \ldots, \Delta\mathbf{s}_{n+2k-1}]\boldsymbol{\beta}_{M,\lambda}. \tag{3.12}$$

We set $M = YY^{\mathrm{T}}$ with $Y \in \mathbb{R}^{kp\times k}$ and $\operatorname{rank}(Y^{\mathrm{T}}\Delta^2\overline{S}_n^{(k)}) = k$. If $\lambda = 0$, we see that Approach 6 is a particular case of problem (3.9). If $Y = I_k \otimes \mathbf{y}$, for some $\mathbf{y} \in \mathbb{R}^p$, we obtain a method similar to the topological Shanks transformation (Brezinski, 1975).

We refer the reader to Section 6 for a discussion of different possible strategies for the selection of the regularization parameter λ.

4. Possible uses of acceleration strategies

In this section and in the following one, we consider the solution of the fixed-point problem $G(\mathbf{s}) = \mathbf{s}$. There are three ways to proceed.

The simplest way is to use an extrapolation method. The vectors $\mathbf{s}_n$ are generated one by one by Picard's iteration as $\mathbf{s}_{n+1} = G(\mathbf{s}_n), n = 0, 1, \ldots$, from a given $\mathbf{s}_0$. The extrapolation method is applied after each computation of a new vector $\mathbf{s}_n$ by using a certain number of the preceding Picard iterates to produce a completely new extrapolated sequence. This procedure is called the *acceleration method* but it is not used in this paper (see Brezinski & Redivo-Zaglia, 2017 for details).

The second way consists in computing a certain number of Picard iterates, then using these in one of the extrapolation techniques introduced in Section 3, and finally to restarting the Picard iterates from the extrapolated vector that has been obtained. This is the *Restarted Method* (RM) treated below.

In the third way, the process builds its own sequence step by step. Each term of the sequence is obtained by combining, in a certain manner, Picard iterates, preceding terms of the sequence and extrapolated ones. We will focus on three possible algorithms of this type that are termed *Continuous-Updating* (CU), presented in this section, the ATM and the *periodic* ATM methods, both discussed in Section 5. The difference between these procedures lies in the way in which previous iterates are combined in the process to obtain a new vector.

4.1 *Restarted method*

In this methodology, already described, for example, in Brezinski (1970), Brezinski & Redivo-Zaglia (2017), Gekeler (1972), a certain number of Picard iterates are produced, an extrapolation strategy is then applied to them and the Picard iterates are *restarted* from the extrapolated vector; see Algorithm 1. The sequence of successive extrapolated terms will be denoted by $(\mathbf{x}_j)$.

Algorithm 1: The Restarted Method (RM).

Input: Choose M, λ, k, and $\mathbf{x}_0 \in \mathbb{R}^p$.

for $j = 0, 1, \ldots$ **do**
  Set $\mathbf{s}_0 = \mathbf{x}_j$
  for $i = 1, \ldots, \ell_k - 1$ *(basic or inner iterations)* **do**
    Compute $\mathbf{s}_i = G(\mathbf{s}_{i-1})$
  end
  Compute $\mathbf{t}_0^{(\ell_k - 1)}$ using (3.3) or (3.6) or (3.10) or (3.11)
  Set $\mathbf{x}_{j+1} = \mathbf{t}_0^{(\ell_k - 1)}$
end

Observe that $\ell_k = k + 2$ if we use (3.3) or (3.6), and $\ell_k = 2k + 1$ if we use (3.11) or (3.12). In the particular case of (3.6), we have

$$\mathbf{t}_0^{(k+1)} = \mathbf{s}_{k+1} - G_{M,\lambda} \Delta \mathbf{s}_k$$

where

$$G_{M,\lambda} = [\Delta \mathbf{s}_1, \ldots, \Delta \mathbf{s}_k]((\Delta^2 S_0^{(k)})^{\mathrm{T}} M \Delta^2 S_0^{(k)} + \lambda I)^{-1} (\Delta^2 S_0^{(k)})^{\mathrm{T}} M.$$

Setting $\mathbf{f}_k = G(\mathbf{s}_k) - \mathbf{s}_k = \Delta\mathbf{s}_k$ we have

$$\mathbf{t}_0^{(k+1)} = \mathbf{s}_{k+1} - G_{M,\lambda}\mathbf{f}_k.$$

Therefore, we can interpret the RM as a *cyclic projection method* (see Galántai, 2003 and Brezinski, 1997 for the linear case) for the solution of the problem $F(\mathbf{s}) = 0$ where $F(\mathbf{s}) = G(\mathbf{s}) - \mathbf{s}$.

The idea of the RM, which is to interleave a certain number of Picard iterates with one extrapolation step, can also be used in the CU and in the ATM methods (see Section 5 where a general 'periodic' algorithm of this type is presented).

A particular case of the RM is the *generalized Steffensen method* (GSM), which corresponds to the case where the dimension of the projection space coincides with the dimension of the system, which is for $k = p$. Under some assumptions, when $\lambda = 0$ and $M = I$, the sequence $(\mathbf{x}_j)$ obtained by the GSM asymptotically converges quadratically to the fixed point $\mathbf{s}^*$ of G even if G is not a contraction. The GSM is a generalization of the well-known Steffensen method (Steffensen, 1933) when $p = 1$. It was first proposed by Brezinski (1970) and Gekeler (1972) for the case of the vector ε-algorithm, but there was a gap in their proofs as in that of Skelboe for the MPE (Smith *et al.*, 1988) as noticed in Skelboe (1980). The first complete proof of the quadratic convergence of the GSM was given by Ortega & Rheinboldt (1970, p. 373) for Henrici's method (Henrici, 1964, pp. 115 ff.) (a particular case of the MMPE), Le Ferrand (1992) for the first topological Shanks transformation of Brezinski (1975) and Jbilou and Sadok for the MPE and the RRE (Jbilou & Sadok, 1991).

4.2 *CU method*

In this approach the sequence is *continuously accelerated* by computing a new basic iterate at each step, using it in the extrapolation process and, after the computation of the extrapolated vector, replacing the new basic iterate computed before by it. Thus, when compared with the original fixed-point sequence, the CU scheme builds a completely new sequence whose iterates replace those of the original sequence.

We start with the minimal residual approach for computing the α_i. We have the *CU method* given in Algorithm 2.

Algorithm 2: Continuous-Updating method (CU) with $\boldsymbol{\alpha}_{M,\lambda}$.

Input: Choose M, λ, $m \in \mathbb{N}, m \geq 1$, $\mathbf{s}_0 \in \mathbb{R}^p$.

for $j = 0, 1, \ldots$ **do**
- Set $m_j = \min(m, j)$
- Compute $\mathbf{s}_{j+1} = G(\mathbf{s}_j)$ *(Picard iteration)*
- Set $S_{j-m_j}^{(m_j+1)} = [\mathbf{s}_{j-m_j}, \ldots, \mathbf{s}_j]$
- Compute $\boldsymbol{\alpha}_{M,\lambda}$ using (3.2) and $\Delta S_{j-m_j}^{(m_j+1)}$
- Compute $\widetilde{\mathbf{t}}_{j-m_j}^{(m_j+1)} = S_{j-m_j}^{(m_j+1)}\boldsymbol{\alpha}_{M,\lambda}$
- Set $\mathbf{s}_{j+1} = \widetilde{\mathbf{t}}_{j-m_j}^{(m_j+1)}$

end

Algorithm 3 uses formulas (3.5) and (3.7) (i.e. the β_i, which solve problem (3.4), are computed by (3.5))

Algorithm 3: Continuous-Updating method (CU) with $\boldsymbol{\beta}_{M,\lambda}$.

Input: Choose M, λ, $m \in \mathbb{N}, m \geq 1$, $\mathbf{s}_0 \in \mathbb{R}^p$.

Compute $\mathbf{s}_1 = G(\mathbf{s}_0)$
for $j = 1, 2, \ldots$ **do**
  Set $m_j = \min(m, j)$
  Compute $\mathbf{s}_{j+1} = G(\mathbf{s}_j)$ *(Picard iteration)*
  Set $\Delta S_{j-m_j}^{(m_j)} = [\Delta \mathbf{s}_{j-m_j}, \ldots, \Delta \mathbf{s}_{j-1}]$
  Compute $\boldsymbol{\beta}_{M,\lambda}$ using (3.5) and $\Delta^2 S_{j-m_j}^{(m_j)}$
  Compute $\widetilde{\mathbf{t}}_{j-m_j}^{(m_j+1)} = \mathbf{s}_j - \Delta S_{j-m_j}^{(m_j)} \boldsymbol{\beta}_{M,\lambda}$
  Set $\mathbf{s}_{j+1} = \widetilde{\mathbf{t}}_{j-m_j}^{(m_j+1)}$
end

As in Algorithm 2, the new fixed-point iterate $\mathbf{s}_{j+1}$ is used only for computing $\boldsymbol{\beta}_{M,\lambda}$. Thereafter, this iterate is not used in the linear combination for computing the extrapolated vector as it is replaced by the extrapolated one that is computed.

It is possible to highlight the connection between acceleration techniques and the projection framework. We define

$$G_{M,\lambda}^{(j)} = [\Delta \mathbf{s}_{j-m_j}, \ldots, \Delta \mathbf{s}_{j-1}]((\Delta^2 S_{j-m_j}^{(m_j)})^{\mathrm{T}} M \Delta^2 S_{j-m_j}^{(m_j)} + \lambda I)^{-1} (\Delta^2 S_{j-m_j}^{(m_j)})^{\mathrm{T}} M.$$

If we set $\mathbf{f}_j = G(\mathbf{s}_j) - \mathbf{s}_j = \mathbf{s}_{j+1} - \mathbf{s}_j$ (where here $\mathbf{s}_{j+1}$ denotes the Picard iteration) we can compute a new vector $\mathbf{s}_{j+1}$ as

$$\mathbf{s}_{j+1} = \mathbf{s}_j - G_{M,\lambda}^{(j)} \mathbf{f}_j.$$

Observe that $G_{M,\lambda}^{(j)}$ satisfies the following *multisecant* condition, see, e.g., Fang & Saad (2009) (when $\lambda = 0$):

$$G_{M,\lambda}^{(j)} \Delta^2 S_{j-m_j}^{(m_j)} = [\Delta \mathbf{s}_{j-m_j}, \ldots, \Delta \mathbf{s}_{j-1}].$$

It is interesting to notice that, when $\lambda \neq 0$, we obtain a class of *regularized projection methods*, which do not yet seem to have been fully investigated in the literature.

For the sake of simplicity, we did not present here the topological approaches of Section 2.2, but the preceding algorithms can be easily modified for these transformations.

5. ATM methods

AA (also known as *Anderson mixing*) is a technique originally presented in Anderson (1965) for solving systems of nonlinear equations written as $F(\mathbf{s}) = G(\mathbf{s}) - \mathbf{s} = 0$. In this section we generalize the basic version of AA as given by Walker & Ni (2011) or by Higham & Strabić (2016). The

main idea of this generalization is that a procedure similar to AA can be built up with any of the Shanks transformations. We will name such methods ATM to emphasize the fact that, as will be explained, these methods use a CU scheme, which *mixes* information coming out from two different sequences.

Indeed, in the framework of the CU scheme presented in Section 4.2, two different sequences are generated, i.e., the continuously updated sequence $(\mathbf{s}_j)$ on the one hand, and the sequence $(G(\mathbf{s}_j))$ on the other. The main feature of the Anderson-mixing strategy is that it combines the information coming from these two sequences in order to obtain a better acceleration procedure. We will prove that it coincides with a quasi-Newton strategy. Since, in this case, the sequence $(\mathbf{s}_j)$ is not generated by a *fixed-point iteration*, we also consider the sequence $(\mathbf{f}_j)$, where $\mathbf{f}_j = G(\mathbf{s}_j) - \mathbf{s}_j = \mathbf{g}_j - \mathbf{s}_j$, which does not coincide with the sequence $(\Delta\mathbf{s}_j)$.

Algorithm 4 shown below is a prototype version of the ATM method where we define

$$F_{j-m_j}^{(m_j)} \equiv [\mathbf{f}_{j-m_j}, \ldots, \mathbf{f}_{j-1}]$$

and use the previous notation $S_{j-m_j}^{(m_j)} \equiv [\mathbf{s}_{j-m_j}, \ldots, \mathbf{s}_{j-1}]$.

Algorithm 4: The Anderson-Type Mixing (ATM) method.

Input: Choose $m \in \mathbb{N}, m \geq 1$, $\beta \in \mathbb{R}$, $\mathbf{s}_0 \in \mathbb{R}^p$.

Compute $\mathbf{f}_0 = G(\mathbf{s}_0) - \mathbf{s}_0$ and $\mathbf{s}_1 = \mathbf{s}_0 + \beta\mathbf{f}_0$
for $j = 1, 2, \ldots$ **do**
  Compute $\mathbf{f}_j = G(\mathbf{s}_j) - \mathbf{s}_j$
  Set $m_j = \min(m, j)$
  Set $\Delta S_{j-m_j}^{(m_j)} = [\Delta\mathbf{s}_{j-m_j}, \ldots, \Delta\mathbf{s}_{j-1}]$ and $\Delta F_{j-m_j}^{(m_j)} = [\Delta\mathbf{f}_{j-m_j}, \ldots, \Delta\mathbf{f}_{j-1}]$
  Compute $\boldsymbol{\theta}^{(j)} \in \mathbb{R}^{m_j}$
  Compute $\overline{\mathbf{s}}_j = \mathbf{s}_j - \Delta S_{j-m_j}^{(m_j)}\boldsymbol{\theta}^{(j)}$ and $\overline{\mathbf{f}}_j = \mathbf{f}_j - \Delta F_{j-m_j}^{(m_j)}\boldsymbol{\theta}^{(j)}$
  Set $\mathbf{s}_{j+1} = \overline{\mathbf{s}}_j + \beta\overline{\mathbf{f}}_j$
end

The scalar β, usually a fixed positive value with $0 < \beta \leq 1$, is called a *mixing* or *damping* parameter. It is also possible to change it at each cycle, and it can be used to improve convergence. A common choice is to take $\beta = 1$. In this case, since $\mathbf{g}_j = G(\mathbf{s}_j) = \mathbf{s}_j + \mathbf{f}_j$ we can define

$$G_{j-m_j}^{(m_j)} = [\mathbf{g}_{j-m_j}, \ldots, \mathbf{g}_{j-1}] = S_{j-m_j}^{(m_j)} + F_{j-m_j}^{(m_j)}.$$

By denoting $\overline{\mathbf{g}}_j = \overline{\mathbf{s}}_j + \overline{\mathbf{f}}_j = \mathbf{g}_j - \Delta G_{j-m_j}^{(m_j)}\boldsymbol{\theta}^{(j)}$, the new iterate can be simply computed as $\mathbf{s}_{j+1} = \overline{\mathbf{g}}_j$. This is the so-called *undamped iterate*.

Let us point out that line 8 in Algorithm 4 can be alternatively written as

$$\mathbf{s}_{j+1} = \mathbf{s}_j - (-\beta \mathbf{f}_j + (\Delta S_{j-m_j}^{(m_j)} + \beta \Delta F_{j-m_j}^{(m_j)})\boldsymbol{\theta}^{(j)}) \tag{5.1}$$

and that different choices of $\boldsymbol{\theta}^{(j)}$ give rise to different ATMs. Some particular cases are described in the sequel.

The original AA is obtained when

$$\boldsymbol{\theta}^{(j)} = \arg\min_{\boldsymbol{\eta} \in \mathbb{R}^{m_j}} \|\mathbf{f}_j - \Delta F_{j-m_j}^{(m_j)} \boldsymbol{\eta}\|^2, \tag{5.2}$$

that is, assuming that the columns of $\Delta F_{j-m_j}^{(m_j)}$ are linearly independent,

$$\boldsymbol{\theta}^{(j)} = ((\Delta F_{j-m_j}^{(m_j)})^{\mathrm{T}} \Delta F_{j-m_j}^{(m_j)})^{-1} (\Delta F_{j-m_j}^{(m_j)})^{\mathrm{T}} \mathbf{f}_j.$$

REMARK 5.1 It is interesting to observe that defining $\boldsymbol{\theta}^{(j)}$ as

$$\boldsymbol{\theta}^{(j)} = \arg\min_{\boldsymbol{\eta} \in \mathbb{R}^{m_j}} \|\Delta \mathbf{s}_j - \Delta^2 S_{j-m_j}^{(m_j)} \boldsymbol{\eta}\|^2,$$

i.e., using (2.7), would be a good choice if the sequence $(\mathbf{s}_j)$ is close to the Shanks kernel. Instead, in the original AA the derivation of the $\boldsymbol{\theta}^{(j)}$ using (5.2) could be interpreted as an implicit assumption that the sequence $(\mathbf{f}_j)$ is *closer* to the Shanks kernel than the sequence $(\mathbf{s}_j)$.

From (5.1) we have

$$\mathbf{s}_{j+1} = \mathbf{s}_j - (-\beta I + (\Delta S_{j-m_j}^{(m_j)} + \beta \Delta F_{j-m_j}^{(m_j)})((\Delta F_{j-m_j}^{(m_j)})^{\mathrm{T}} \Delta F_{j-m_j}^{(m_j)})^{-1} (\Delta F_{j-m_j}^{(m_j)})^{\mathrm{T}}) \mathbf{f}_j, \tag{5.3}$$

as also observed in Brezinski *et al.* (2018), Fang & Saad (2009).

Formula (5.1) highlights the connections between Anderson mixing and quasi-Newton methods. Indeed, in this case, defining

$$H_j^{(\beta)} = -\beta I + (\Delta S_{j-m_j}^{(m_j)} + \beta \Delta F_{j-m_j}^{(m_j)})((\Delta F_{j-m_j}^{(m_j)})^{\mathrm{T}} \Delta F_{j-m_j}^{(m_j)})^{-1} (\Delta F_{j-m_j}^{(m_j)})^{\mathrm{T}},$$

we can write

$$\mathbf{s}_{j+1} = \mathbf{s}_j - H_j^{(\beta)} \mathbf{f}_j,$$

with $H_j^{(\beta)}$ satisfying the multisecant condition $H_j^{(\beta)} \Delta F_{j-m_j}^{(m_j)} = \Delta S_{j-m_j}^{(m_j)}$. In the next section we will fully make use of this idea: by introducing a *stabilization procedure* to overcome problems connected to the ill conditioning of the matrix $(\Delta F_{j-m_j}^{(m_j)})^{\mathrm{T}} \Delta F_{j-m_j}^{(m_j)}$, it is possible to prove the local linear convergence of the AA method.

As indicated in the previous section, it is also possible to define a *periodic ATM* method whereby acceleration steps are interspersed with linear updates at regular intervals. Fixing the period $\mu \in \mathbb{N}$, with $\mu \geq 1$, an Anderson-type update is made each μ iterations. In between these updates, when $\mu > 1$, the iterates are computed simply as a linear mixing $\mathbf{s}_{j+1} = \mathbf{s}_j + \beta \mathbf{f}_j$, where β is the mixing parameter ($\beta = 1$ corresponds to Picard's iteration). Clearly, when $\mu = 1$ Algorithm 5 coincides with Algorithm 4.

Algorithm 5: The periodic Anderson-Type Mixing method.

Input: Choose $m, \mu \in \mathbb{N}$, $m, \mu \geq 1$, $\beta \in \mathbb{R}$, $\mathbf{s}_0 \in \mathbb{R}^p$.

Compute $\mathbf{f}_0 = G(\mathbf{s}_0) - \mathbf{s}_0$ and $\mathbf{s}_1 = \mathbf{s}_0 + \beta \mathbf{f}_0$
for $j = 1, 2, \ldots$ **do**
  Compute $\mathbf{f}_j = G(\mathbf{s}_j) - \mathbf{s}_j$
  if $(j+1) \bmod \mu = 0$ **then**
    Compute $\mathbf{s}_{j+1}$ using steps 4 to 8 of Algorithm 4 *(Anderson-type update)*
  else
    Compute $\mathbf{s}_{j+1} = \mathbf{s}_j + \beta \mathbf{f}_j$ *(linear mixing update)*
  end
end

It is important to underline that the values of μ and m can be chosen independently. However, when $\mu \geq 3$ and we choose $m = \mu - 2$, then the terms used for computing the Anderson-type update are only those terms obtained by the linear mixing update, and therefore in this situation Algorithm 5 proceeds as an RM method of Algorithm 1, with a different restarting formula. It must also be noticed that Algorithm 5 with $\boldsymbol{\theta}^{(j)}$ computed as in (5.2) is exactly the periodic Pulay method (Banerjee *et al.*, 2016; compare also with (5.3)). Interleaving AA with fixed-point iterations for improving the global convergence properties, but not necessarily the speed, has been recognized before in the physics literature as can be seen from the related discussion and the references in Banerjee *et al.* (2016). This idea is also somewhat similar to the A2DR (Anderson accelerated Douglas–Rachford) algorithm proposed in Fu *et al.* (2020).

To start the derivation of the new ATMs, we observe that a possible generalization for the derivation of the $\boldsymbol{\theta}^{(j)}$ can be obtained by using the coupled sequences defined in Section 2.3, that is, by taking

$$\boldsymbol{\theta}^{(j)} = (Y^{\mathrm{T}} \Delta C_{j-m_j}^{(m_j)})^{-1} Y^{\mathrm{T}} \mathbf{c}_j. \tag{5.4}$$

If we take $\mathbf{c}_j = \mathbf{f}_j$ for all j, and $Y = \Delta C_{j-m_j}^{(m_j)} = \Delta F_{j-m_j}^{(m_j)}$ we recover the AA choice for $\boldsymbol{\theta}^{(j)}$. It is easy to see that taking into account the transformations defined at the beginning of Section 3, if we consider the extrapolated vector $\widetilde{\boldsymbol{t}}_{j-m_j}^{(m_j+1)} = \mathbf{s}_j - \Delta S_{j-m_j}^{(m_j)} \boldsymbol{\theta}^{(j)}$ we recover exactly the $\bar{\mathbf{s}}_j$ computed in Algorithms 4 and 5. If, in this relation, we extrapolate the coupled sequence $(\mathbf{f}_j)$ with the same $\boldsymbol{\theta}^{(j)}$, we obtain $\bar{\mathbf{f}}_j$.

An additional generalization can be made by considering, as in problem (3.4), a different metric in the evaluation of the norm, and also a regularization parameter λ. We consider the

problem

$$\boldsymbol{\theta}_{M,\lambda}^{(j)} = \underset{\boldsymbol{\eta}\in\mathbb{R}^{m_j}}{\arg\min} \left(\|\mathbf{c}_j - \Delta C_{j-m_j}^{(m_j)} \boldsymbol{\eta}\|_M^2 + \lambda \|\boldsymbol{\eta}\|^2 \right). \tag{5.5}$$

The solution is

$$\boldsymbol{\theta}_{M,\lambda}^{(j)} = ((\Delta C_{j-m_j}^{(m_j)})^{\mathrm{T}} M \Delta C_{j-m_j}^{(m_j)} + \lambda I)^{-1} (\Delta C_{j-m_j}^{(m_j)})^{\mathrm{T}} M \mathbf{c}_j. \tag{5.6}$$

By taking in (5.6) $\mathbf{c}_j = \mathbf{f}_j$ and $M = I$, that is, by introducing only an ℓ_2-regularization term in the original AA problem, we obtain a method that we call *Regularized Anderson Acceleration* (in short RAA).

If we take $M = YY^T$ and $\lambda = 0$, it is possible to see that $\boldsymbol{\theta}_{M,\lambda}^{(j)}$ in (5.6) can be obtained, alternatively, as the solution of the linear system

$$(\Delta C_{j-m_j}^{(m_j)})^{\mathrm{T}} Y (Y^{\mathrm{T}} \Delta C_{j-m_j}^{(m_j)} \boldsymbol{\theta}_{YY^T,0}^{(j)} - Y^{\mathrm{T}} \mathbf{c}_j) = 0,$$

corresponding exactly to (5.4), assuming that $\mathrm{rank}((\Delta C_{j-m_j}^{(m_j)})^{\mathrm{T}} Y) = m_j$.

The ATM methods can thus be obtained by considering the coupled sequence $(\mathbf{c}_j) = (\mathbf{f}_j)$ fixed and changing the matrix Y. The following particular cases are of interest:

1. ATM-RRE: $Y = [\Delta^2 \mathbf{s}_{j-m_j}, \ldots, \Delta^2 \mathbf{s}_{j-1}] = \Delta^2 S_{j-m_j}^{(m_j)} \in \mathbb{R}^{p\times m_j}$ corresponds to a method in the style of the RRE. For this choice, since we also need knowledge of the vector $\mathbf{s}_{j+1}$, we have to edit Algorithm 4 slightly by beginning the loop (line 2) with $j = 2$ and by adding before it the computation of $\mathbf{s}_2 = \mathbf{s}_1 + \beta \mathbf{f}_1$. Modifications that take this into account must also be made in Algorithm 5. The choice $Y = [\Delta^2 \mathbf{f}_{j-m_j}, \ldots, \Delta^2 \mathbf{f}_{j-1}] = \Delta^2 F_{j-m_j}^{(m_j)} \in \mathbb{R}^{p\times m_j}$ is also possible.

2. ATM-MPE: $Y = [\Delta \mathbf{s}_{j-m_j}, \ldots, \Delta \mathbf{s}_{j-1}] = \Delta S_{j-m_j}^{(m_j)} \in \mathbb{R}^{p\times m_j}$ or $Y = [\mathbf{f}_{j-m_j}, \ldots, \mathbf{f}_{j-1}] = F_{j-m_j}^{(m_j)} \in \mathbb{R}^{p\times m_j}$ leads to two methods in the style of the MPE.

3. ATM-MMPE: $Y = [\mathbf{y}_1, \ldots, \mathbf{y}_{m_j}] \in \mathbb{R}^{p\times m_j}$ leads to an ATM in the style of the MMPE.

4. ATM-TEA: suitably modifying the structure of Algorithms 4 and 5, it is possible to use a topological approach (see Section 2.2) to obtain the coefficients $\boldsymbol{\theta}_{M,\lambda}^{(j)}$. As in Section 4.2 we omit all the details for the sake of brevity.

Before concluding this section, we point out that the introduction of an ℓ_2-regularization term for AA has already been studied in recent papers (Anderson, 2019; Fu *et al.*, 2020; Ouyang *et al.*, 2020), and that (5.5) represents a generalization to the ATM methods of the ℓ_2-regularization approach for AA. In Section 6, for the particular AA case, we will propose and experimentally analyze the choice of the regularization parameter λ using the *Generalized Cross Validation* (Golub *et al.*, 1979). This choice represents a major difference from the above-mentioned works, where the choice of the regularization parameter is made adaptively based on quantities related to the most recent iterates (see, for example, Fu *et al.*, 2020, eq. (3.4) and Ouyang *et al.*, 2020, eq. (3)). Sections 5.1 and 5.2 below further justify/clarify the introduction of an ℓ_2-regularization strategy.

5.1 *Stabilized AA*

The aim of this section is to present an algorithm that can be viewed as a *stabilized* version of the AA method. In particular, in this new version of AA, a check on the linear independence of the vectors $\Delta\mathbf{f}_d$ is performed (lines 7–16): the residual difference $\Delta\mathbf{f}_d$ is discarded if its projection $\widehat{\mathbf{f}}_d$ onto the orthogonal of the previously computed residual differences is *close* to the null vector, i.e., if it results in a vector of sufficiently small norm when compared to the original one (see Section 5.2 for further details). It is interesting to note that when, in Algorithm 6, we choose $m = 1$ (and likely for small values of m) the introduced stabilization procedure is not required and Algorithm 6 coincides with the classic AA scheme (compare, in this case, (5.3) and the update at line 20 in Algorithm 6).

Algorithm 6: Stabilized Anderson Acceleration.

Input: Choose $m \in \mathbb{N}, m \geq 1$, $\beta \in \mathbb{R}$, $\mathbf{s}_0 \in \mathbb{R}^p$ and $\tau > 1$.

1 Compute $\mathbf{f}_0 = G(\mathbf{s}_0) - \mathbf{s}_0$ and $\mathbf{s}_1 = \mathbf{s}_0 + \beta\mathbf{f}_0$
2 **for** $j = 1, \ldots$ **do**
3 Set $m_j = \min(m, j)$.
4 Compute $\mathbf{f}_j = G(\mathbf{s}_j) - \mathbf{s}_j$
5 Compute $\widehat{\mathbf{f}}_{j-m_j} = \Delta\mathbf{f}_{j-m_j}$
6 Set $P_{j-m_j} = (\widehat{\mathbf{f}}_{j-m_j}\widehat{\mathbf{f}}_{j-m_j}^{\mathrm{T}})/(\widehat{\mathbf{f}}_{j-m_j}^{\mathrm{T}}\widehat{\mathbf{f}}_{j-m_j})$
7 **for** $d = j - m_j + 1, \ldots, j - 1$ **do**
8 Set $Q_{j-m_j}^{d-1} = \sum_{i=j-m_j}^{d-1} P_i$
9 Compute $\widehat{\mathbf{f}}_d = (I - Q_{j-m_j}^{d-1})\Delta\mathbf{f}_d$
10 **if** $\|\widehat{\mathbf{f}}_d\|\tau \geq \|\Delta\mathbf{f}_d\|$ **then**
11 Set $P_d = (\widehat{\mathbf{f}}_d\widehat{\mathbf{f}}_d^{\mathrm{T}})/(\widehat{\mathbf{f}}_d^{\mathrm{T}}\widehat{\mathbf{f}}_d)$
12 **else**
13 Set $\widehat{\mathbf{f}}_d = \mathbf{0}$
14 Set $P_d = \mathbf{0}$
15 **end**
16 **end**
17 Set $\mathcal{I}_j = \{k_1, \ldots, k_{\widehat{m}_j}\} \subseteq \{j - m_j, \ldots, j - 1\}$ the set of indices such that $\widehat{\mathbf{f}}_{k_1}, \ldots, \widehat{\mathbf{f}}_{k_{\widehat{m}_j}}$ are non null vectors
18 Set $\Delta F_{\mathcal{I}_j} = [\Delta\mathbf{f}_{k_1}, \ldots, \Delta\mathbf{f}_{k_{\widehat{m}_j}}]$, $\Delta S_{\mathcal{I}_j} = [\Delta\mathbf{s}_{k_1}, \ldots, \Delta\mathbf{s}_{k_{\widehat{m}_j}}]$
19 Set $H_j^{(\beta)} = \big[-\beta I + (\Delta S_{\mathcal{I}_j} + \beta\Delta F_{\mathcal{I}_j})((\Delta F_{\mathcal{I}_j})^{\mathrm{T}}\Delta F_{\mathcal{I}_j})^{-1}(\Delta F_{\mathcal{I}_j})^{\mathrm{T}}\big]$
20 Compute $\mathbf{s}_{j+1} = \mathbf{s}_j - H_j^{(\beta)}\mathbf{f}_j$
21 **end**

5.1.1 *Local convergence.* There already exist in the literature different proofs of the local convergence for the stabilized versions of AA; see, for example, Fu *et al.* (2020), Gay & Schnabel (1978), Ouyang *et al.* (2020), Rohwedder (2010), Rohwedder & Schneider (2011). In principle, our convergence analysis can be obtained using ideas and techniques from Rohwedder (2010, Sec. 4.2), but we prefer to present here a full detailed proof. The reasons to present such a detailed proof can be mainly summarized as follows: (a) our derivation is not completely analogous to that in Rohwedder (2010): simplifying some arguments, we are able to obtain slightly more general results than those presented in Rohwedder

(2010, Sec. 4.2) (the interested reader can compare our Theorem 5.8 with Rohwedder, 2010, Th. 4.10); (b) our analysis does not require the contractivity or nonexpansivity of the fixed-point map G, a major difference if compared to what has been proved in Fu *et al.* (2020), Ouyang *et al.* (2020); (c) our proof of convergence holds for every mixing parameter $\beta \in \mathbb{R}$, shedding further light on the significance and the relevance of the parameter β in the AA procedure: it can be interpreted as a scaling factor of the initial Jacobian approximation (see Theorem 5.8); (d) when $m = 1$, since Algorithm 6 coincides with the classic AA scheme (see the beginning of Section 5.1), we obtain, as a by-product of our analysis, an alternative proof of that given in Toth & Kelley (2015, Sec. 2.3) for the convergence of the classic AA with $m = 1$ without assuming, once more, any contractivity of the fixed-point map G. We consider the function $F(\mathbf{s}) = G(\mathbf{s}) - \mathbf{s}$, and we make the following assumption:

ASSUMPTION 5.2 $F : \mathbf{R}^n \to \mathbf{R}^n$ is differentiable in a open convex set $E \subseteq \mathbb{R}^n$ and there exists $\mathbf{s}^* \in E$ such that $\mathbf{f}^* = F(\mathbf{s}^*) = \mathbf{0}$. Moreover, $J = F'(\mathbf{s}^*)$ is invertible and for all $\mathbf{s} \in E$ we have

$$\|F'(\mathbf{s}) - F'(\mathbf{s}^*)\| \leq L\|\mathbf{s} - \mathbf{s}^*\|.$$

The above assumption implies that

$$\|F(\mathbf{u}) - F(\mathbf{v}) - J(\mathbf{u} - \mathbf{v})\| \leq L\|\mathbf{u} - \mathbf{v}\| \max\{\|\mathbf{u} - \mathbf{s}^*\|, \|\mathbf{v} - \mathbf{s}^*\|\}$$

for all $\mathbf{u}, \mathbf{v} \in E$ and that there exists $U_\kappa(\mathbf{s}^*) := \{\mathbf{u} \in \mathbb{R}^n \ : \ \|\mathbf{u} - \mathbf{s}^*\| \leq \kappa\}$ s.t., for some $\rho > 0$,

$$\frac{1}{\rho}\|\mathbf{u} - \mathbf{v}\| \leq \|F(\mathbf{u}) - F(\mathbf{v})\| \leq \rho\|\mathbf{u} - \mathbf{v}\|.$$

In the remainder of this section we use the notation introduced in Algorithm 6.

LEMMA 5.3 The matrices $H_j^{(\beta)}$ (defined at line 19 of Algorithm 6) satisfy the multisecant condition

$$H_j^{(\beta)} \Delta F_{\mathcal{I}_j} = \Delta S_{\mathcal{I}_j}.$$

Proof. The proof is by direct verification. □

LEMMA 5.4 $H_j^{(\beta)}$ can be computed recursively from $H_j^0 = -\beta I$ using

$$H_j^d = H_j^{d-1} + \frac{(\Delta \mathbf{s}_{k_d} - H_j^{d-1}\Delta \mathbf{f}_{k_d})\widehat{\mathbf{f}}_{k_d}^{\mathrm{T}}}{\widehat{\mathbf{f}}_{k_d}^{T}\Delta \mathbf{f}_{k_d}} \quad \text{for } d = 1, \ldots, \widehat{m}_j$$

with $H_j^{\widehat{m}_j} = H_j^{(\beta)}$ (see line 17 in Algorithm 6 for the definitions of $\widehat{\mathbf{f}}_{k_d}$). In particular, for all $d = 1, \ldots, \widehat{m}_j$, we have $H_j^d \Delta \mathbf{f}_{k_p} = \Delta \mathbf{s}_{k_p}$ for $p = 1, \ldots, d$.

Proof. Define $Z \in \mathbb{R}^{n \times n - \widehat{m}_j}$ as a basis for $\mathrm{span}(\Delta F_{\mathcal{I}_j})^\perp$. From the definition of $H_j^{(\beta)}$ we have $H_j^{(\beta)} Z = -\beta Z$ and $H_j^{(\beta)} \Delta F_{\mathcal{I}_j} = \Delta S_{\mathcal{I}_j}$. To prove the theorem, we will prove (by induction) that $H_j^{\widehat{m}_j}$ satisfies the same relations. For $d = 1$ we have $H_j^1 = H_j^0 + \frac{(\Delta \mathbf{s}_{k_1} - H_j^0 \Delta \mathbf{f}_{k_1})\widehat{\mathbf{f}}_{k_1}^{\mathrm{T}}}{\widehat{\mathbf{f}}_{k_1}^{\mathrm{T}} \Delta \mathbf{f}_{k_1}}$ and hence $H_j^1 \Delta \mathbf{f}_{k_1} = \Delta \mathbf{s}_{k_1}$. Suppose now

the assumption is true for $d = \ell$. By definition we have $H_j^{\ell+1}\Delta\mathbf{f}_{k_{\ell+1}} = \Delta\mathbf{s}_{k_{\ell+1}}$ and $H_j^{\ell+1}\Delta\mathbf{f}_{k_p} = \Delta\mathbf{s}_{k_p}$ for all $p = 1, \ldots, \ell$ since $\widehat{\mathbf{f}}_{k_{\ell+1}} \perp \Delta\mathbf{f}_{k_p}$. Finally, since

$$\operatorname{span}(\widehat{\mathbf{f}}_{k_1}, \ldots, \widehat{\mathbf{f}}_{k_{\widehat{m}_j}}) = \operatorname{span}(\Delta\mathbf{f}_{k_1}, \ldots, \Delta\mathbf{f}_{k_{\widehat{m}_j}}),$$

this implies that Z is also a basis for $\operatorname{span}(\widehat{\mathbf{f}}_{k_1}, \ldots, \widehat{\mathbf{f}}_{k_{\widehat{m}_j}})^{\perp}$, and hence we have $H_j^{\widehat{m}_j} Z = -\beta Z$. The result follows observing that, since $[\Delta F_{\mathcal{I}_j}, Z]$ is invertible, the equation $B[\Delta F_{\mathcal{I}_j}, Z] = [\Delta S_{\mathcal{I}_j}, -\beta Z]$ has a unique solution. □

Observe that, as already pointed out in Zhang *et al.* (2020), Lemma 5.4 highlights the connections between the Jacobian approximations produced by the *bad (or type-II) Broyden* update (Broyden, 1965) and the matrices produced by AA.

LEMMA 5.5 Let us define $\widehat{\mathbf{s}}_{k_1} = \Delta\mathbf{s}_{k_1}$ and for $d = 2, \ldots, \widehat{m}_j$ define $\widehat{\mathbf{s}}_{k_d} = \Delta\mathbf{s}_{k_d} - H_j^{d-1} Q_{k_1}^{k_{d-1}} \Delta\mathbf{f}_{k_d}$ where $Q_{k_1}^{k_{d-1}} = \sum_{p=1}^{d-1} (\widehat{\mathbf{f}}_{k_p} \widehat{\mathbf{f}}_{k_p}^{\mathrm{T}} / \widehat{\mathbf{f}}_{k_p}^{\mathrm{T}} \widehat{\mathbf{f}}_{k_p})$. Then $H_j^{(\beta)}$ can be computed recursively from $H_j^0 = -\beta I$ using

$$H_j^d = H_j^{d-1} + \frac{(\widehat{\mathbf{s}}_{k_d} - H_j^{d-1}\widehat{\mathbf{f}}_{k_d})\widehat{\mathbf{f}}_{k_d}^{\mathrm{T}}}{\widehat{\mathbf{f}}_{k_d}^{\mathrm{T}} \widehat{\mathbf{f}}_{k_d}} \quad \text{for } d = 1, \ldots, \widehat{m}_j$$

with $H_j^{\widehat{m}_j} = H_j^{(\beta)}$. In particular, for all $d = 1, \ldots, \widehat{m}_j$, we have $H_j^d \widehat{\mathbf{f}}_{k_p} = \widehat{\mathbf{s}}_{k_p}$ for $p = 1, \ldots, d$.

Proof. The proof follows from the definition of H_j^d, and observing that

$$\widehat{\mathbf{f}}_{k_d} = (I - Q_{k_1}^{k_{d-1}})\Delta\mathbf{f}_{k_d} \quad \Rightarrow \quad \widehat{\mathbf{f}}_{k_d}^{\mathrm{T}} \widehat{\mathbf{f}}_{k_d} = \widehat{\mathbf{f}}_{k_d}^{\mathrm{T}} \Delta\mathbf{f}_{k_d}$$

(since $(I - Q_{k_1}^{k_{d-1}})$ is a projector) and that $\widehat{\mathbf{s}}_{k_d} - H_j^{d-1}\widehat{\mathbf{f}}_{k_d} = \Delta\mathbf{s}_{k_d} - H_j^{d-1}\Delta\mathbf{f}_{k_d}$. □

LEMMA 5.6 Suppose that $\mathbf{s}_{k_d}, \mathbf{s}_{k_d+1} \in U_\kappa(\mathbf{s}^*)$ for all $d = 1, \ldots, \widehat{m}_j$. Then the following inequality is satisfied:

$$\|\widehat{\mathbf{s}}_{k_d} - J^{-1}\widehat{\mathbf{f}}_{k_d}\| \le C\|\Delta\mathbf{f}_{k_d}\| \sum_{p=1}^{d} n_{k_p}^{k_p+1} (2\tau)^{p-d},$$

where $C = \|J^{-1}\| L\rho$ and $n_{k_p}^{k_p+1} = \max\{\|\mathbf{s}_{k_p+1} - \mathbf{s}^*\|, \|\mathbf{s}_{k_p} - \mathbf{s}^*\|\}$.

Proof. For $d = 1$ we have

$$\|\widehat{\mathbf{s}}_{k_1} - J^{-1}\widehat{\mathbf{f}}_{k_1}\| = \|\Delta\mathbf{s}_{k_1} - J^{-1}\Delta\mathbf{f}_{k_1}\| \le C\|\Delta\mathbf{f}_{k_1}\| n_{k_1}^{k_1+1},$$

where the last inequality follows from Assumption 5.2. Suppose now the assumption is true for $d = \ell$. To prove the statement for $d = \ell + 1$ we have

$$\begin{aligned}
\|\widehat{\mathbf{s}}_{k_{\ell+1}} - J^{-1}\widehat{\mathbf{f}}_{k_{\ell+1}}\| &\le \|\Delta\mathbf{s}_{k_{\ell+1}} - J^{-1}\Delta\mathbf{f}_{k_{\ell+1}}\| + \|H_j^{\ell}Q_{k_1}^{k_\ell}\Delta\mathbf{f}_{k_{\ell+1}} - J^{-1}Q_{k_1}^{k_\ell}\Delta\mathbf{f}_{k_{\ell+1}}\| \\
&\le C\|\Delta\mathbf{f}_{k_{\ell+1}}\|n_{k_{\ell+1}}^{k_{\ell+1}+1} + \sum_{p=1}^{\ell}\frac{\|H_j^{\ell}\widehat{\mathbf{f}}_{k_p} - J^{-1}\widehat{\mathbf{f}}_{k_p}\|}{\|\widehat{\mathbf{f}}_{k_p}\|}\|\Delta\mathbf{f}_{k_{\ell+1}}\| \\
&= C\|\Delta\mathbf{f}_{k_{\ell+1}}\|n_{k_{\ell+1}}^{k_{\ell+1}+1} + \sum_{p=1}^{\ell}\frac{\|\widehat{\mathbf{s}}_{k_p} - J^{-1}\widehat{\mathbf{f}}_{k_p}\|}{\|\widehat{\mathbf{f}}_{k_p}\|}\|\Delta\mathbf{f}_{k_{\ell+1}}\| \\
&\le C\|\Delta\mathbf{f}_{k_{\ell+1}}\|(n_{k_{\ell+1}}^{k_{\ell+1}+1} + \tau\sum_{p=1}^{\ell}\sum_{h=1}^{p}n_{k_h}^{k_h+1}(2\tau)^{p-h}) \\
&= C\|\Delta\mathbf{f}_{k_{\ell+1}}\|(n_{k_{\ell+1}}^{k_{\ell+1}+1} + \tau\sum_{p=1}^{\ell}n_{k_p}^{k_p+1}\sum_{h=0}^{\ell-p}(2\tau)^{h}),
\end{aligned}$$

where, in the first inequality we use the definition of $\widehat{\mathbf{s}}_{k_{\ell+1}}$, in the second inequality we use the definition of $Q_{k_1}^{k_\ell}$, in the first equality we use the fact that $H_j^{\ell}\widehat{\mathbf{f}}_{k_p} = \widehat{\mathbf{s}}_{k_p}$ for $p = 1, \ldots, \ell$ (see Lemma 5.5) and, in the third inequality we use our induction hypothesis. Finally, since

$$\sum_{h=0}^{\ell-p}(2\tau)^h \le \tau^{\ell-p}\sum_{h=0}^{\ell-p}2^h = \tau^{\ell-p}(2^{\ell-p+1} - 1) \le \tau^{\ell-p}2^{\ell-p+1},$$

we have

$$C\|\Delta\mathbf{f}_{k_{\ell+1}}\|\left(n_{k_{\ell+1}}^{k_{\ell+1}+1} + \tau\sum_{p=1}^{\ell}n_{k_p}^{k_p+1}\sum_{h=0}^{\ell-p}(2\tau)^h\right) \le C\|\Delta\mathbf{f}_{k_{\ell+1}}\|\sum_{p=1}^{\ell+1}n_{k_p}^{k_p+1}(2\tau)^{\ell+1-p},$$

which concludes the proof. □

LEMMA 5.7 The following equality is satisfied:

$$H_j^{(\beta)} - J^{-1} = (-\beta I - J^{-1})(I - Q_{k_1}^{k_{\widehat{m}_j}}) + \sum_{d=1}^{\widehat{m}_j}\frac{(\widehat{\mathbf{s}}_{k_d} - J^{-1}\widehat{\mathbf{f}}_{k_d})\widehat{\mathbf{f}}_{k_d}^{\mathrm{T}}}{\widehat{\mathbf{f}}_{k_d}^{\mathrm{T}}\widehat{\mathbf{f}}_{k_d}}.$$

Moreover, if $\mathbf{s}_{k_d}, \mathbf{s}_{k_d+1} \in U_\kappa(\mathbf{s}^*)$ and $n_{k_d}^{k_d+1} \le \varepsilon$ for all $d = 1, \ldots, \overline{m}_j$, there exists a constant $\alpha = \alpha(\tau, m, C)$ such that

$$\sum_{d=1}^{\widehat{m}_j}\frac{\|(\widehat{\mathbf{s}}_{k_d} - J^{-1}\widehat{\mathbf{f}}_{k_d})\widehat{\mathbf{f}}_{k_d}^{\mathrm{T}}\|}{\widehat{\mathbf{f}}_{k_d}^{\mathrm{T}}\widehat{\mathbf{f}}_{k_d}} \le \alpha\varepsilon.$$

Proof. The first part of the statement follows from direct computation using the fact that the vectors $\widehat{\mathbf{f}}_{k_d}$ are orthogonal (see also Rohwedder, 2010, Lemma 4.17). For the second part, observe that

$$\begin{aligned}\sum_{d=1}^{\widehat{m}_j} \frac{\|(\widehat{\mathbf{s}}_{k_d} - J^{-1}\widehat{\mathbf{f}}_{k_d})\widehat{\mathbf{f}}_{k_d}^{\mathrm{T}}\|}{\widehat{\mathbf{f}}_{k_d}^T\widehat{\mathbf{f}}_{k_d}} &\leq \sum_{d=1}^{\widehat{m}_j} \frac{\|(\widehat{\mathbf{s}}_{k_d} - J^{-1}\widehat{\mathbf{f}}_{k_d})\|}{\|\widehat{\mathbf{f}}_{k_d}\|} \\ &\leq C\sum_{d=1}^{\widehat{m}_j} \frac{\|\Delta\mathbf{f}_{k_d}\|}{\|\widehat{\mathbf{f}}_{k_d}\|} \sum_{p=1}^{d} n_{k_d}^{n_d+1}(2\tau)^{d-p} \leq \varepsilon C\tau \sum_{d=1}^{\widehat{m}_j}\sum_{p=1}^{d}(2\tau)^{d-p} \\ &\leq \varepsilon C\tau \sum_{d=1}^{m}\sum_{h=0}^{d-1}(2\tau)^h \leq \varepsilon C m(2\tau)^m,\end{aligned}$$

where in the second inequality we use Lemma 5.6 and in the fourth one the fact that $\widehat{m}_j \leq m_j \leq m$ for all j. □

THEOREM 5.8 Let $\mathbf{s}_0, \mathbf{s}_1, \ldots$ be the iterates produced by Algorithm 6 (stabilized AA). Then, for all $q \in (0,1)$, there exists $\delta = \delta(q,\alpha)$, $\varepsilon(q,\alpha)$ such that if

$$\|-\beta I - J^{-1}\| \leq \delta \quad \text{and} \quad \|\mathbf{s}_0 - \mathbf{s}^*\| \leq \varepsilon,$$

we have

$$\mathbf{s}_{j+1} \in E \quad \text{and} \quad \|\mathbf{s}_{j+1} - \mathbf{s}^*\| \leq q\|\mathbf{s}_j - \mathbf{s}^*\|$$

for all $j \in \mathbf{N}$.

Proof. For a fixed q, choose δ and ε such that

$$\|J^{-1}\|L\varepsilon + \rho(\delta + \alpha\varepsilon) < q$$

in a way that $U_\varepsilon(\mathbf{s}^*) \subseteq U_\kappa(\mathbf{s}^*) \subseteq E$ (where κ and α are the same as in Lemma 5.7, and ρ is the same as in Assumption 5.2). For $j = 0$ we have

$$\begin{aligned}\|\mathbf{s}_1 - \mathbf{s}^*\| &\leq \|\mathbf{s}_0 + \beta\mathbf{f}_0 - \mathbf{s}^*\| \leq \|\mathbf{s}_0 - \mathbf{s}^* - J^{-1}(\mathbf{f}_0 - \mathbf{f}^*)\| + \|(-\beta I - J^{-1})(\mathbf{f}_0 - \mathbf{f}^*)\| \\ &\leq \|J^{-1}\|\|J(\mathbf{s}_0 - \mathbf{s}^*) - (\mathbf{f}_0 - \mathbf{f}^*)\| + \delta\|\mathbf{f}_0 - \mathbf{f}^*\| \leq (\|J^{-1}\|L\varepsilon + \delta\rho)\|\mathbf{s}_0 - \mathbf{s}^*\| \\ &\leq q\|\mathbf{s}_0 - \mathbf{s}^*\| \leq \varepsilon,\end{aligned}$$

which proves that $\mathbf{s}_1 \in U_\varepsilon(\mathbf{s}^*)$. Assume now that, for all $j \geq 0$, $\|\mathbf{s}_j - \mathbf{s}^*\| \leq q^j \|\mathbf{s}_0 - \mathbf{s}^*\|$ and hence that $\mathbf{s}_j \in U_\varepsilon(\mathbf{s}^*)$. We have

$$\begin{aligned}
\|\mathbf{s}_{j+1} - \mathbf{s}^*\| &= \|\mathbf{s}_j - H_j^{(\beta)}\mathbf{f}_j - \mathbf{s}^*\| \\
&\leq \|J^{-1}\|\|J(\mathbf{s}_j - \mathbf{s}^*) - (\mathbf{f}_j - \mathbf{f}^*)\| + \|H_j^{(\beta)} - J^{-1}\|\|\mathbf{f}_j - \mathbf{f}^*\| \\
&\leq \|J\|^{-1} L\|\mathbf{s}_j - \mathbf{s}^*\|^2 + \rho\|H_j^{(\beta)} - J^{-1}\|\|\mathbf{s}_j - \mathbf{s}^*\| \\
&\leq (\|J\|^{-1} L q^j \varepsilon + \rho(\delta + \alpha\varepsilon))\|\mathbf{s}_j - \mathbf{s}^*\| \leq q\|\mathbf{s}_j - \mathbf{s}^*\|,
\end{aligned}$$

where in the last inequality we use our induction hypothesis and Lemma 5.7. □

It is interesting to note that, in the particular case that G is contractive, Theorem 5.8 proves that, at least locally, the stabilized version of AA (Algorithm 6) could improve the rate of convergence of the fixed-point map $\mathbf{s}_{j+1} = G(\mathbf{s}_j)$ since the linear convergence parameter q in Theorem 5.8 can be chosen smaller than the contraction factor of G (see also Evans *et al.*, 2020; Ouyang *et al.*, 2020). Observe, moreover, that if the inequality $\| - \beta I - J^{-1}\| \leq \delta$ is not fulfilled, we can consider the *preconditioned* nonlinear function $\tilde{F} = P^{-1}F$ where P is some approximation of $J = F(\mathbf{s}^*)$, and we obtain in this way $\| - \beta I - J^{-1}P\| \leq \delta$.

Finally, let us observe that, as customary in the quasi-Newton literature, we can improve the global convergence properties of the AA procedure by introducing a step-length parameter α_j and transforming the sequence generated by Algorithm 6 into the sequence

$$\mathbf{s}_{j+1} = \mathbf{s}_j - \alpha_j H_j^{(\beta)}\mathbf{f}_j.$$

5.2 *Connections between stabilized AA and regularized ATM*

As already pointed out in the previous section, from a theoretical point of view, the stabilization procedure introduced in Algorithm 6, in order to ensure the convergence, aims to detect a subset of the vectors in $\Delta F_{j-m_j}^{(m_j)}$ that are *sufficiently linearly independent*: the proposed stabilization procedure in Algorithm 6 (lines 7–16) can be interpreted simply as a Gram–Schmidt procedure with threshold, i.e., the residual difference $\Delta\mathbf{f}_d$ is discarded if it is *close* to a vector linearly dependent from the previously computed residual differences. The above observation naturally links the stabilization procedure with rank-revealing QR factorizations (Chan, 1987; Gu & Eisenstat, 1996). We find this issue particularly interesting and deserving of further investigation. Here we prefer to adopt a regularization point of view, as in Anderson (2019), Fu *et al.* (2020), Ouyang *et al.* (2020), Scieur *et al.* (2020), to motivate the introduction of the regularization parameter λ in the ATM methods as we did at the beginning of Section 5. To this end, let us consider the ATM obtained by (5.6) with $\mathbf{c}_j = \mathbf{f}_j$ and $M = I$. As already pointed out, when $\lambda = 0$ it coincides with the classical AA but, when $\lambda \neq 0$, the method obtained can be viewed as an RAA.

In this setting, we interpret the magnitude of the singular values of the matrix $\Delta F_{j-m_j}^{(m_j)}$ as a measure of the linear independence of its columns: the presence of linearly dependent vectors in $\Delta F_{j-m_j}^{(m_j)}$ is highlighted by the presence of very small singular values. Let us consider now the SVD decomposition $\Delta F_{j-m_j}^{(m_j)} = U\Sigma V^{\mathrm{T}}$. We add a regularization parameter λ to the matrix Σ and we set

$\Delta\widetilde{F}^{(m_j)}_{j-m_j} = U\sqrt{\Sigma^2+\lambda}V^{\mathrm{T}}$. By direct computation, it is possible to show that (5.6) can be written as

$$\boldsymbol{\theta}^{(j)}_{I,\lambda} = ((\Delta\widetilde{F}^{(m_j)}_{j-m_j})^{\mathrm{T}}\Delta\widetilde{F}^{(m_j)}_{j-m_j})^{-1}(\Delta F^{(m_j)}_{j-m_j})^{\mathrm{T}}\mathbf{f}_j.$$

The statement regarding the linear independence of the columns of the matrix $\Delta\widetilde{F}^{(m_j)}_{j-m_j}$ can be obtained by observing that all its singular values are bounded from below by $\sqrt{\lambda}$. We consider the above argument as an explanation of the fact that the introduction of a regularization parameter in the AA method (and, in general, in all the ATMs) could achieve numerically the same task as the stabilization procedure of Algorithm 6. Adopting a quasi-Newton point of view, it is important to observe that using (5.1) with $\boldsymbol{\theta}^{(j)} = \boldsymbol{\theta}^{(j)}_{I,\lambda}$, the ATM update (see line 8 in Algorithm 4) can be written as

$$\mathbf{s}_{j+1} = \mathbf{s}_j - \widetilde{H}^{(\beta)}_j\mathbf{f}_j,$$

with

$$\widetilde{H}^{(\beta)}_j = -\beta I + (\Delta S^{(m_j)}_{j-m_j} + \beta\Delta F^{(m_j)}_{j-m_j})((\Delta\widetilde{F}^{(m_j)}_{j-m_j})^{\mathrm{T}}\Delta\widetilde{F}^{(m_j)}_{j-m_j})^{-1}(\Delta F^{(m_j)}_{j-m_j})^{\mathrm{T}}. \tag{5.7}$$

The quasi-Newton matrices defined in (5.7) satisfy only an *approximated multisecant condition*, namely

$$\widetilde{H}^{(\beta)}_j\Delta F^{(m_j)}_{j-m_j} = \Delta S^{(m_j)}_{j-m_j} + \beta(\Delta F^{(m_j)}_{j-m_j}((\Delta\widetilde{F}^{(m_j)}_{j-m_j})^{\mathrm{T}}\Delta\widetilde{F}^{(m_j)}_{j-m_j})^{-1}(\Delta F^{(m_j)}_{j-m_j})^{\mathrm{T}}\Delta F^{(m_j)}_{j-m_j} - \Delta F^{(m_j)}_{j-m_j}),$$

which represents a noteworthy difference from the multisecant conditions satisfied by the quasi-Newton matrices used in the classical AA and in its stabilized version (see Lemma 5.3).

6. Numerical results

In this section we investigate the numerical behavior of some of the methods studied in the previous sections for different test problems.

6.1 *Details on the methods and their implementations*

We select a subset of the methods presented in the previous sections with the main aims to compare their numerical performance (with a focus on the rate of convergence), and to prove that the acceleration performance they deliver behaves consistently. Our choices are, among other things, driven by the fact that all the acceleration methods considered share the same order of complexity (linear in the dimension of the problem) per acceleration step. A comprehensive detailed numerical study and the relative implementations of all the methods described in the previous sections are out of the scope of this work and are postponed to future works. Table 1 summarizes the methods we consider in our numerical experiments. In the first column we report the name and the relative abbreviation for the particular acceleration scheme we consider. In the second column we report the reference equations of the acceleration scheme and, for the sake of completeness, in the third column we report the strategy type of the considered acceleration: RM or CU. Finally, in the last column, we report the details concerning the choice of the regularization parameter: in the *Grid Search* (GS) approach the regularization parameter $\overline{\lambda}$ is chosen, as proposed in Scieur *et al.* (2020), as the parameter that achieves the smallest fixed-point

TABLE 1 *Methods tested*

Name	Ref. eq.	Type	Choice of λ
Singular Value Decomposition Acceleration (SVDA)	(2.6)	RM	$\lambda = 0$
Regularized Nonlinear Acceleration (RNA)	(3.2)	RM	GS (Alg. 7)
Regularized Reduced Rank Extrapolation (RRRE)	(3.5)	RM	GCV, Golub *et al.* (1979)
Regularized Topological Shanks Acceleration (RTSA)	(3.10)	RM	GS (Alg. 7)
Anderson Acceleration (AA) with $0 < \beta \leq 1$	(5.3)	CU	$\lambda = 0$
Regularized Anderson Acceleration (RAA)	(5.7)	CU	GCV, Golub *et al.* (1979)

residual; the interval $[10^{-12}, 1]$ is discretized logarithmically into 7 values of λ (for more details see Algorithm 7, which is a modification of Algorithm 1) among which, one of them, $\overline{\lambda}$, is selected. For the sake of completeness, let us recall that, also in this new algorithm, $\ell_k = k + 2$ if we use (3.3) or (3.6), and $\ell_k = 2k + 1$ if we use (3.11) or (3.12) For the *Generalized Cross Validation* (GCV) approach, which is a natural approach for regularizing ill-posed regression-like problems, we refer the interested reader to Golub *et al.* (1979).

Let us point out that, to the best of our knowledge, among the methods presented in Table 1, RTSA and RRRE/RAA with the regularization parameter chosen using the GCV are new approaches introduced in this work. Instead, for the other methods, we refer in particular to Sidi (2003) for the SVDA (which is called SVD-MPE in the original paper) and to Scieur *et al.* (2020) for the RNA.

Finally, we mention that in all the numerical experiments we used $M = I$ and that, in the SVDA approach, we use as extrapolated term $\mathbf{t}_n^{(k+1)} = S_{n+1}^{(k+1)}\boldsymbol{\alpha}$ where $\boldsymbol{\alpha}$ is the normalized singular vector corresponding to the smallest singular value of $\Delta S_n^{(k+1)}$ (see equation (2.6)).

Algorithm 7: The Restarted Method (RM) with Grid Search (GS).

Input: Choose M, k, $\lambda_{\min}$, $\lambda_{\max}$, n, and $\mathbf{x}_0 \in \mathbb{R}^p$.

for $j = 0, 1, \ldots$ **do**
  Set $\mathbf{s}_0 = \mathbf{x}_j$
  for $i = 1, \ldots, \ell_k - 1$ *(basic or inner iterations)* **do**
    Compute $\mathbf{s}_i = G(\mathbf{s}_{i-1})$
  end
  Choose $\lambda_0, \ldots, \lambda_{n-1} \in [\lambda_{\min}, \lambda_{\max}]$
  for $i = 0, \ldots, n - 1$ **do**
    Compute $\mathbf{t}_{0,\lambda_i}^{(\ell_k-1)}$ using (3.3) or (3.6) or (3.10) or (3.11)
  end
  $\overline{\lambda} = \underset{\lambda_i \in \{\lambda_0, \ldots, \lambda_{n-1}\}}{\arg\min} \|G(\mathbf{t}_{0,\lambda_i}^{(\ell_k-1)}) - \mathbf{t}_{0,\lambda_i}^{(\ell_k-1)}\|$
  Set $\mathbf{x}_{j+1} = \mathbf{t}_{0,\overline{\lambda}}^{(\ell_k-1)}$
end

All the numerical experiments are performed on a laptop running Linux with 16 Gb memory and CPU Intel® Core™ i7-4510U with a clock speed of 2.00 GHz. The code is written and executed in Python. For the discretization of the PDEs we used Fenics (Alnæs *et al.*, 2015) and, for the GCV

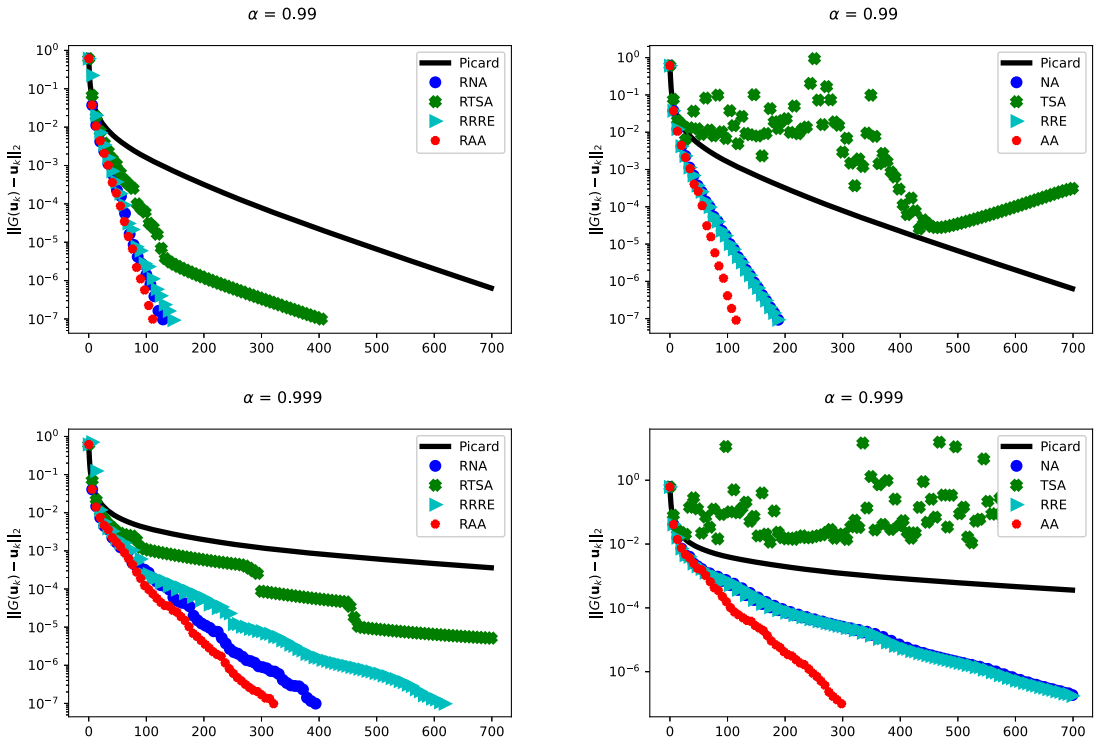

FIG. 1. PageRank problem.

choice of the regularization parameter, we used the Scikit-learn package (Pedregosa *et al.*, 2011). Throughout the experiments, to show and test the robustness of the different extrapolation approaches, we base all our extrapolation schemes on seven previous iterates, i.e., $\ell_k = 7$ in Algorithm 7 or $m = 7$ in Algorithm 4.

6.2 *PageRank*

The aim of this first numerical example is to highlight the benefits of introducing regularization strategies in Shanks-based extrapolation methods. In particular, in this section we consider the PageRank problem (see Eldén, 2006), i.e., the problem of computing the Perron eigenvector of the matrix

$$G = \alpha S + \frac{(1-\alpha)}{n}\mathbf{e}\mathbf{e}^{\mathrm{T}}, \quad \alpha \in (0,1),$$

where S is a non-negative column stochastic matrix. For the solution of this problem, we consider the power method, i.e., $\mathbf{u}_{k+1} = G(\mathbf{u}_k)$ where $\mathbf{u}_0$ is a non-negative stochastic vector, which is known to be a linear fixed-point iteration globally convergent with a rate of convergence of $\mathcal{O}(\alpha^k)$ (Eldén, 2006). As the previous convergence bound confirms, the rate of convergence of the power method for the PageRank computation becomes slower as α approaches 1, but this is usually the case of interest in applications (Eldén, 2006). In this experiment we use as stopping criterion $\|G(\mathbf{u}_k) - \mathbf{u}_\mathbf{k}\| < 10^{-7}$.

In the left panel of Fig. 1, we report the acceleration performance of the regularized versions of the methods considered when compared to the nonregularized ones (in the right panel), for the computation of the PageRank vector of the matrix `amazon-0202` from Davis & Hu (2011) (which has been suitably modified in order to be stochastic and dangling-nodes-free; Eldén, 2006). Recall that the sequence generated by the power method belongs to the Shanks kernel and hence, at least theoretically, all the extrapolation strategies should be equivalent and should work consistently without any requirement of regularization. Nevertheless, as Fig. 1 clearly shows, the introduction of a regularization strategy improves the robustness of the extrapolation procedures permitting us, for the restarted extrapolation methods (namely RNA, RTSA, RRRE), to obtain a more effective acceleration performance across different choices of the parameter α. Observe also that, in this case, the introduction of a regularization procedure in the AA scheme (RAA) does not sensibly improve the acceleration performance.

6.3 *Nonlinear Poisson problems*

In this section we consider the solution of the nonlinear PDE (see (6.1))

$$\begin{aligned} -\nabla(q(u)\nabla u) + g(u) + u_x &= f \quad \text{in } \mathcal{D} = [0,1] \times [0,1], \\ u &= v \quad \text{on } \partial\mathcal{D}. \end{aligned} \tag{6.1}$$

We use a 1/64 uniform triangular mesh of $\Omega = [0,1]^2$ with a (P_2) discretization (Alnæs *et al.*, 2015) that provides a total of 16,641 degrees of freedom. In particular, we consider the following choices of the functions:

- $q(u) = 1 + u^2$ or $q(u) = 1 + u^4$, $g(u) = 0$ and f such that the exact solution of (6.1) is given by $\overline{u} = \exp(-2x)\sin(3\pi y)$ and $v = \overline{u}$ on $\partial\Omega$. We refer to these choices as the *nonlinear Poisson problem*;
- $q(u) = 1$, $g(u) = \lambda e^u$ with $\lambda = 1$ or $\lambda = -1$, $f = 0$ and $v = 0$ on $\partial\Omega$. We refer to these choices as the *Bratu problem* (Hajipour *et al.*, 2018).

After the discretization of (6.1), the corresponding problems can be written as the solution of $F(\mathbf{s}) = 0$, i.e., as the solution of a linear system of equations. We assume that the derivatives of F are not readily available or that a sufficiently accurate initial guess is not at our disposal in order to apply Newton's method. In this experiment we use as stopping criterion $\|F(\mathbf{u}_k)\| < 10^{-7}$. Figures 2 and 3 show the acceleration performance of AA when compared to its regularized version RAA (these problems are not well scaled and a good choice for the mixing parameter was $\beta = 0.1$) for the problems previously discussed. The figures clearly show that the introduction of the regularization strategy, in these cases characterized by a *higher nonlinearity* than for the PageRank example, leads to a better robustness of the schemes with respect to the choice of the memory parameter m. In particular, the introduction of the regularization procedure permits us to have a satisfactory rate of convergence independently from the value m. We point out that, interestingly enough, the need for a stabilization procedure, needed from the theoretical point of view to prove the convergence of the AA scheme (see Algorithm 6), is echoed by the experimental observation that increasing m could result in a loss of

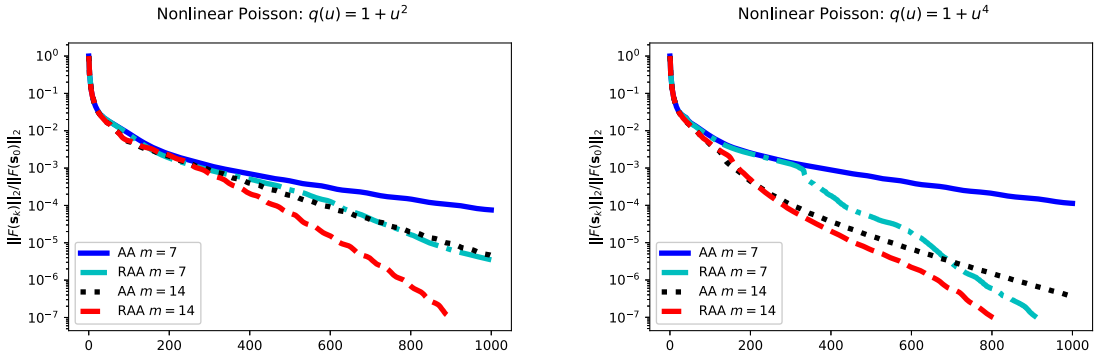

FIG. 2. Nonlinear Poisson problem.

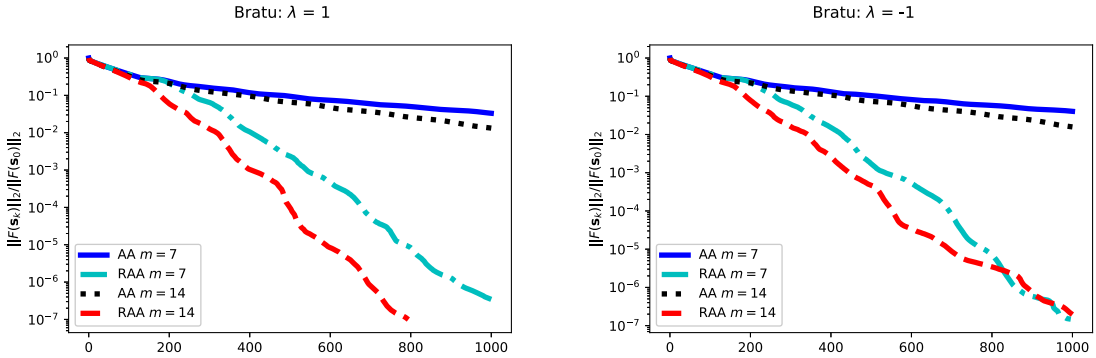

FIG. 3. Bratu problem.

efficiency for the AA scheme (see Fig. 3). The introduction of a regularization procedure mitigates such a drawback.

6.3.1 *Navier–Stokes equation.* In this section we compare the numerical performance of the different restarted extrapolation approaches on the incompressible Navier–Stokes equation (NSE)

$$\begin{aligned} u \cdot \nabla u + \nabla p - \nu \Delta u &= f, \\ \nabla \cdot u &= 0, \\ u|_{\partial\Omega} &= g, \end{aligned}$$

where ν is the kinematic viscosity, f is the forcing, u and p represent velocity and pressure and Ω is a given domain in $\mathbb{R}^2$. Following Pollock *et al.* (2019), we consider a Picard iteration (6.2) to solve the problem. The iteration, which is commonly used for its stability and global convergence properties,

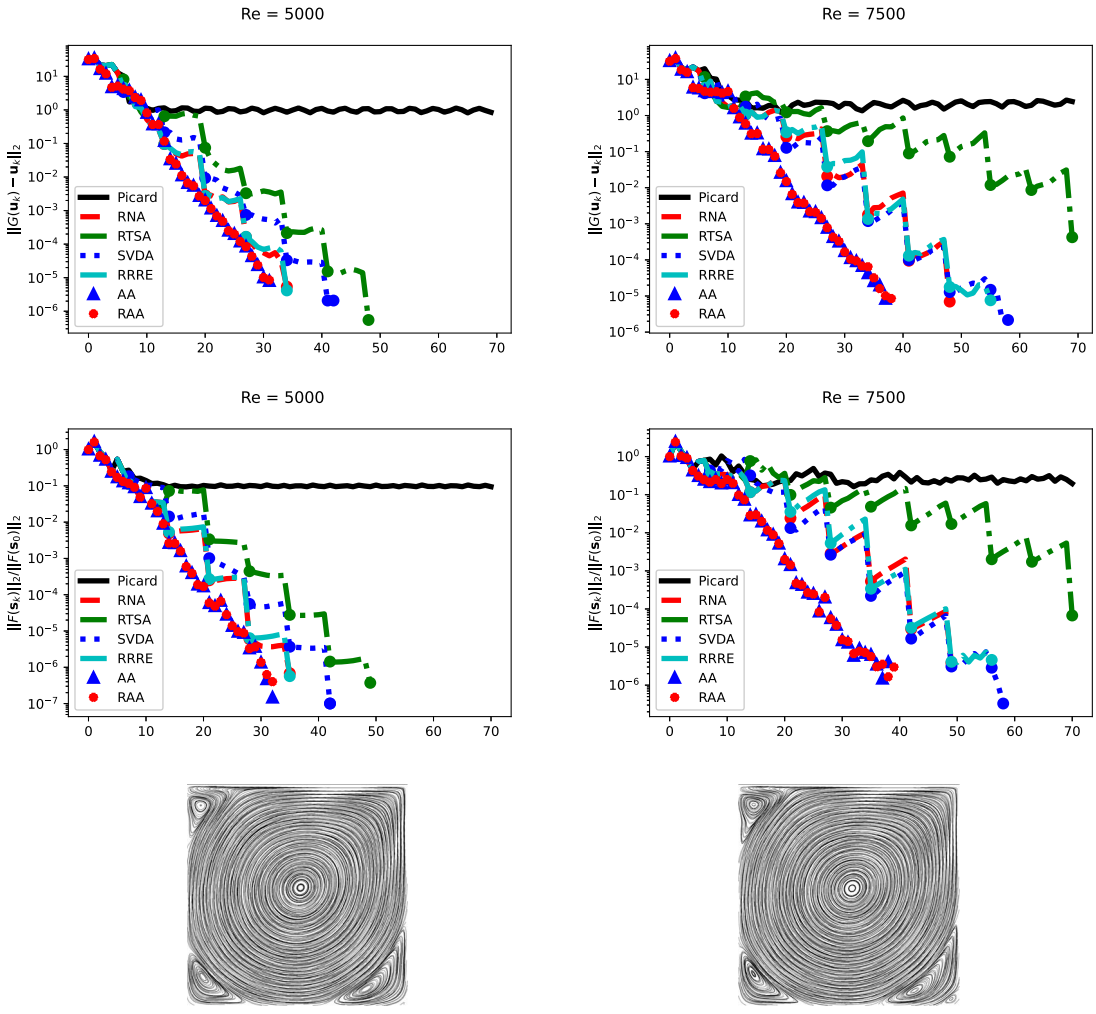

FIG. 4. Lid-driven problem: acceleration performance (moving lid on the top).

takes the form

$$\begin{aligned} u_k \cdot \nabla u_{k+1} + \nabla p_{k+1} - \nu \Delta u_{k+1} &= f, \\ \nabla \cdot u_{k+1} &= 0, \\ u_{k+1}|_{\partial\Omega} &= g. \end{aligned} \tag{6.2}$$

The above scheme is written in the fixed-point form $u_{k+1} = G(u_k)$, where G denotes the solution operator for the linearization (6.2). To be specific, we consider the two-dimensional lid-driven cavity ($\Omega = (0,1)^2$) and a 'deep' lid-driven cavity with $\Omega = (0,1) \times (0,3)$. No-slip ($u = 0$) boundary conditions are imposed on the sides and the bottom, and the Dirichlet boundary condition

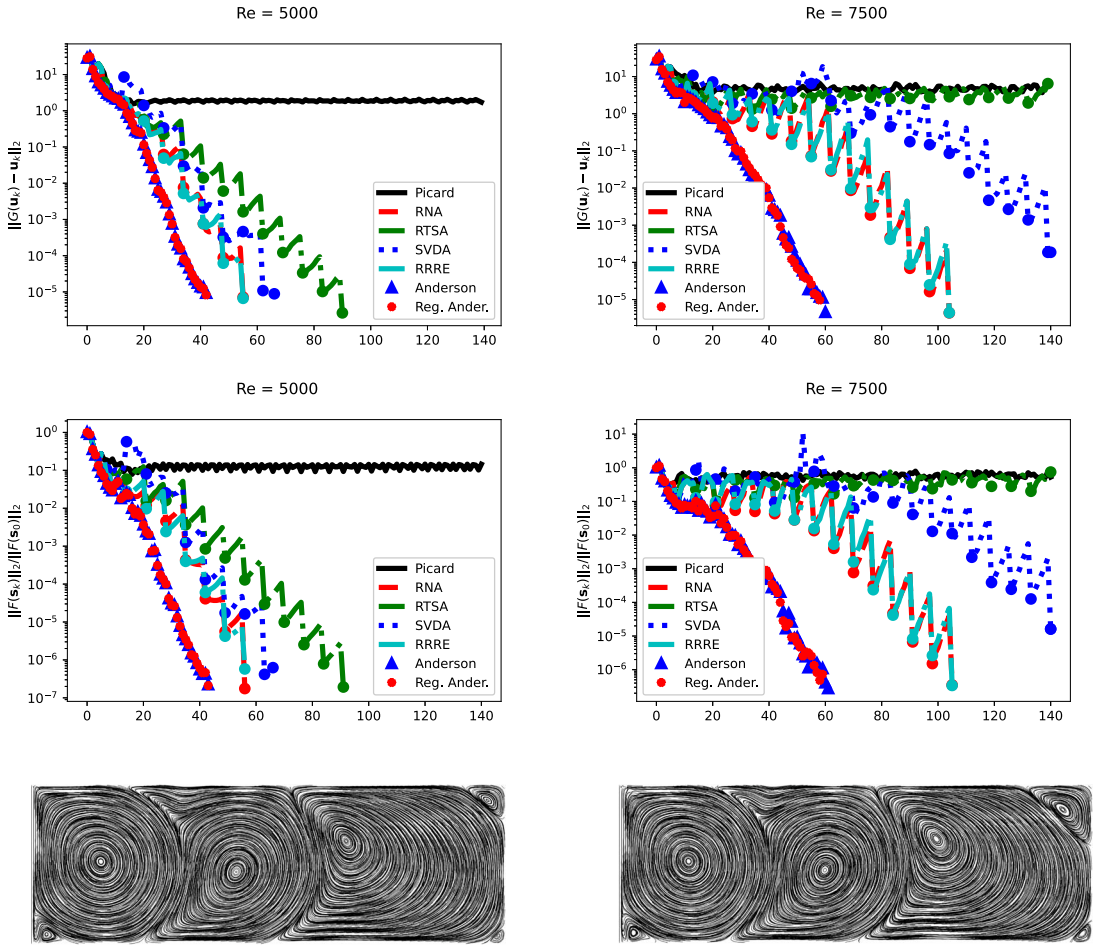

FIG. 5. Lid-driven problem: acceleration performance (moving lid on the left).

$u(x,1) = (1,0)^T$ is imposed on the top to enforce the 'moving lid' condition. There is no forcing ($f = 0$) and the kinematic viscosity ($\nu = \mathrm{Re}^{-1}$) is considered at benchmark values $\mathrm{Re} = 5000,\ 7500$. We discretize with (P_2, P_1) Taylor Hood elements. In the case $\Omega = (0,1)^2$ we use a $\frac{1}{64}$ uniform triangular mesh that provides 37,507 total degrees of freedom and in the case $\Omega = (0,1) \times (0,3)$ we use a $\frac{1}{40} \times \frac{1}{120}$ mesh that provides 87,203 total degrees of freedom. Similarly to the results presented in Pollock *et al.* (2019), our experiments confirm that Newton's method starting with a zero initial guess never converges. In this experiment we use as stopping criterion $\|G(\mathbf{u}_k) - \mathbf{u_k}\| < 10^{-5}$. Figures 4 and 5 show the acceleration performance of the methods described in Table 1 for the solution of the steady NSE. The best performer in terms of achieved acceleration is AA and the introduction of a regularization procedure in this scheme (RAA) seems not to have a relevant impact on the rate of convergence. This is probably due to the fact that the fixed-point iteration we are considering generates a sequence that is *close* to being a linear sequence and, as in the PageRank case, regularization of the AA scheme does not seem to have a great influence. Concerning the restarted regularized methods, we should notice that the RTSA is not

able to achieve an acceleration performance in the *deep* case for Re $= 7500$. Finally, let us highlight the particularly interesting performance of the SVDA approach: this approach does not require the computation of any regularization parameter and only one SVD decomposition every $\ell_k - 1$ fixed-point iterations is needed, whereas AA requires the solution of a least square solution per step and all the regularized methods that use the RM approach require the selection of a regularization parameter. The nonregularized versions of the methods using the RM strategy, as in the PageRank case, exhibited worse performance and are not reported for this reason.

7. Conclusions

In this work we presented a unified framework for Shanks-based transformations. If, on one hand, the introduction of this framework allowed us to link apparently different extrapolation/acceleration techniques with Shanks-based transformations, on the other hand it allowed us to introduce suitable generalizations able to numerically outperform the existing ones, as highlighted in the preliminary numerical results presented, especially on problems characterized by a high degree of nonlinearity. To conclude, we note that the highlighted connection between the Shanks-based transformations and the quasi-Newton methods and AA shed light on some of its theoretical and numerical behaviors, furthering our knowledge of the powerful, but poorly understood, AA (Kelley, 2018).

Acknowledgements

We would like to thank the reviewers for their very careful reading of our paper, and for their constructive comments. S.C. and M.R.-Z. are members of the INdAM Research group GNCS.

Funding

Labex CEMPI (ANR-11-LABX-0007-01 to C.B.); GNCS–INdAM (project 'Efficient methods for large scale problems with applications to data analysis and preconditioning' to S.C.); Department of Computer Science & Engineering, University of Minnesota (project no. UMF0002384 to S.C.); University of Padua ('Numerical linear algebra and extrapolation methods with applications', project no. DOR 1903575/19 to M.R.-Z.); National Science Foundation grant (DMS-1912048 to Y.S.).

References

Alnæs, M., Blechta, J., Hake, J., Johansson, A., Kehlet, B., Logg, A., Richardson, C., Ring, J., Rognes, M. E., & Wells, G. N. (2015) The FEniCS project version 1.5. *Arch. Numer. Softw.*, **3**, 9–23.

Anderson, D. G. (1965) Iterative procedures for nonlinear integral equations. *J. Assoc. Comput. Mach.*, **12**, 547–560.

Anderson, D. G. (2019) Comments on Anderson acceleration, mixing and extrapolation. *Numer. Algorithms*, **80**, 135–234.

Banerjee, A. S., Suryanarayana, P. & Pask, J. E. (2016) Periodic Pulay method for robust and efficient convergence acceleration of self-consistent field iterations. *Chem. Phys. Lett.*, **647**, 31–35.

Brezinski, C. (1970) Application de l'ε-algorithme à la résolution des systèmes non linéaires. *C. R. Acad. Sci. Paris*, **271A**, 1174–1177.

Brezinski, C. (1971) Méthodes d'accélération de la convergence en analyse numérique. *Thèse de Doctorat d'État*, Université Scientifique et Médicale de Grenoble. https://tel.archives-ouvertes.fr/tel-00282774.

Brezinski, C. (1974) Some results in the theory of the vector ε-algorithm. *Linear Algebra Appl.*, **8**, 77–86.

Brezinski, C. (1975) Généralisation de la transformation de Shanks, de la table de Padé et de l'ε-algorithme. *Calcolo*, **12**, 317–360.

Brezinski, C. (1980) *Padé-Type Approximation and General Orthogonal Polynomials,* ISNM, vol. 50. Basel: Birkhäuser.

Brezinski, C. (1988) Other manifestations of the Schur complement. *Linear Algebra Appl.*, **111**, 231–247.

Brezinski, C. (1997) *Projection Methods for Systems of Equations*. Amsterdam: Elsevier.

Brezinski, C. & Redivo-Zaglia, M. (1991) *Extrapolation Methods: Theory and Practice*. Amsterdam: North-Holland.

Brezinski, C. & Redivo-Zaglia, M. (2014) The simplified topological ε-algorithms for accelerating sequences in a vector space. *SIAM J. Sci. Comput.*, **36**, A2227–A2247.

Brezinski, C. & Redivo-Zaglia, M. (2017) The simplified topological ε-algorithms: software and applications. *Numer. Algorithms*, **74**, 1237–1260.

Brezinski, C. & Redivo-Zaglia, M. (2019) The genesis and early developments of Aitken's process, Shanks' transformation, the ε-algorithm, and related fixed point methods. *Numer. Algorithms*, **80**, 11–33.

Brezinski, C. & Redivo-Zaglia, M. (2020) *Extrapolation and Rational Approximation: The Works of the Main Contributors*. Cham, Switzerland: Springer Nature.

Brezinski, C. & Redivo-Zaglia, M. (2021) Extrapolation and prediction of sequences in a vector space. Submitted.

Brezinski, C., Redivo-Zaglia, M. & Saad, Y. (2018) Shanks sequence transformations and Anderson acceleration. *SIAM Rev.*, **60**, 646–669.

Broyden, C. G. (1965) A class of methods for solving nonlinear simultaneous equations. *Math. Comp.*, **19**, 577–593.

Cabay, S. & Jackson, L. W. (1976) A polynomial extrapolation method for finding limits and antilimits of vector sequences. *SIAM J. Numer. Anal.*, **13**, 734–752.

Chan, T. F. (1987) Rank revealing QR factorizations. *Linear Algebra Appl.*, **88**, 67–82.

Cipolla, S., Redivo-Zaglia, M. & Tudisco, F. (2020a) Extrapolation methods for fixed-point multilinear PageRank computations. *Numer. Linear Algebra Appl.*, **27**, e2280.

Cipolla, S., Redivo-Zaglia, M. & Tudisco, F. (2020b) Shifted and extrapolated power methods for tensor ℓ^p-eigenpairs. *Electron. Trans. Numer. Anal*, **53**, 1–27.

Davis, T. A. & Hu, Y. (2011) The University of Florida sparse matrix collection. *ACM Trans. Math. Softw.*, **38**, 1–25.

Delahaye, J. P. (1988) *Sequence Transformations*. Berlin: Springer.

Delahaye, J. P. & Germain-Bonne, B. (1980) Résultats négatifs en accélération de la convergence. *Numer. Math.*, **35**, 443–457.

Eddy, R. P. (1979) Extrapolation to the limit of a vector sequence. *Information Linkage between Applied Mathematics and Industry* (P. C. C. Wang ed). New York: Academic Press, pp. 387–396.

Eldén, L. (2006) Numerical linear algebra in data mining. *Acta Numerica*, **15**, 327–384.

Evans, C., Pollock, S., Rebholz, L. G. & Xiao, M. (2020) A proof that Anderson acceleration improves the convergence rate in linearly converging fixed-point methods (but not in those converging quadratically). *SIAM J. Numer. Anal.*, **58**, 788–810.

Eyert, V. (1996) A comparative study on methods for convergence acceleration of iterative vector sequences. *J. Comput. Phys.*, **124**, 271–285.

Fang, H. R. & Saad, Y. (2009) Two classes of multisecant methods for nonlinear acceleration. *Numer. Linear Algebra Appl.*, **16**, 197–221.

Fu, A., Zhang, J. & Boyd, S. (2020) Anderson accelerated Douglas-Rachford splitting. *SIAM J. Sci. Comput.*, **42**, A3560–A3583.

Galántai, A. (2003) *Projectors and Projection Methods*, New York: Springer Science & Business Media.

Gay, D. M. & Schnabel, R. B. (1978) Solving systems of nonlinear equations by Broyden's method with projected updates. *Nonlinear Programming* (O. Mangasarian, R. Meyer & S. Robinson eds), vol. 3. New York: Academic Press, pp. 245–281.

GEKELER, E. (1972) On the solution of systems of equations by the epsilon algorithm of Wynn. *Math. Comp.*, **26**, 427–436.

GOLUB, G. H., HEATH, M. & WAHBA, G. (1979) Generalized cross-validation as a method for choosing a good ridge parameter. *Technometrics*, **21**, 215–223.

GU, M. & EISENSTAT, S. C. (1996) Efficient algorithms for computing a strong rank-revealing QR factorization. *SIAM J. Sci. Comput.*, **17**, 848–869.

HAJIPOUR, M., JAJARMI, A. & BALEANU, D. (2018) On the accurate discretization of a highly nonlinear boundary value problem. *Numer. Algorithms*, **79**, 679–695.

HENRICI, P. (1964) *Elements of Numerical Analysis*. New York: Wiley.

HIGHAM, N. J. & STRABIĆ, N. (2016) Anderson acceleration of the alternating projections method for computing the nearest correlation matrix. *Numer. Algorithms*, **72**, 1021–1042.

JBILOU, K. & SADOK, H. (1991) Some results about vector extrapolation methods and related fixed point iteration. *J. Comp. Appl. Math.*, **36**, 385–398.

KELLEY, C. T. (2018) Numerical methods for nonlinear equations. *Acta Numer.*, **27**, 207–287.

LE FERRAND, H. (1992) The quadratic convergence of the topological epsilon algorithm for systems of nonlinear equations. *Numer. Algorithms*, **3**, 273–284.

LUPO PASINI, M. (2019) Convergence analysis of Anderson-type acceleration of Richardson's iteration. *Numer. Linear Algebra Appl.*, **26**, e2241.

MEŠINA, M. (1977) Convergence acceleration for the iterative solution of $x = Ax + f$. *Comput. Methods Appl. Mech. Eng.*, **10**, 165–173.

ORTEGA, J. M. & RHEINBOLDT, W. C. (1970) *Iterative Solution of Nonlinear Equations in Several Variables*. New York: Academic Press.

OUYANG, W., TAO, J., MILZAREK, A. & DENG, B. (2020) Nonmonotone globalization for Anderson acceleration using adaptive regularization. arXiv:2006.02559.

PEDREGOSA, F., VAROQUAUX, G., GRAMFORT, A., MICHEL, V., THIRION, B., GRISEL, O., BLONDEL, M., PRETTENHOFER, P., WEISS, R., DUBOURG, V. & VANDERPLAS, J. (2011) Scikit-learn: machine learning in Python. *J. Mach. Learn. Res.*, **12**, 2825–2830.

POLLOCK, S., REBHOLZ, L. G. & XIAO, M. (2019) Anderson-accelerated convergence of Picard iterations for incompressible Navier-Stokes equations. *SIAM J. Numer. Anal.*, **57**, 615–637.

PUGACHEV, B. P. (1978) Acceleration of convergence of iterative processes and a method of solving systems of non-linear equations. *USSR Comput. Maths. Maths. Phys.*, **17**, 199–207.

PULAY, P. (1980) Convergence acceleration in iterative sequences: the case of SCF iteration. *Chem. Phys. Lett.*, **73**, 393–398.

ROHWEDDER, T. (2010) An analysis for some methods and algorithms of quantum chemistry. *Ph.D. thesis*, TU Berlin .

ROHWEDDER, T. & SCHNEIDER, T. R. (2011) An analysis for the DIIS acceleration method used in quantum chemistry calculations. *J. Math. Chem.*, **49**, 1889–1914.

SCIEUR, D., D'ASPREMONT, A. & BACH, F. (2020) Regularized nonlinear acceleration. *Math. Programming*, **179**, 47–83.

SHANKS, D. (1955) Non linear transformations of divergent and slowly convergent sequences. *J. Math. Phys.*, **34**, 1–42.

SIDI, A. (1988) Extrapolation vs. projection methods for linear systems of equations. *J. Comput. Appl. Math.*, **22**, 71–88.

SIDI, A. (2003) *Practical Extrapolation Methods. Theory and Applications*, Cambridge: Cambridge University Press.

SIDI, A. (2016) SVD-MPE: an SVD-based vector extrapolation method of polynomial type. *Appl. Math. J. Chinese Univ. Ser. B*, **7**, 1260–1278.

SIDI, A. & BRIDGER, J. (1988) Convergence and stability analyses for some vector extrapolation methods in the presence of defective iteration matrices. *J. Comp. Appl. Math.*, **22**, 35–61.

Skelboe, S. (1980) Computation of the periodic steady-state response to non linear networks by extrapolation methods. *IEEE Trans. Circuits Syst.*, **27**, 161–175.

Smith, D. A., Ford, W. F. & Sidi, A. (1988) Extrapolation methods for vector sequences. *SIAM Rev.*, 29, 199-233. Correction: *SIAM Rev.*, **30**, 623–624.

Steffensen, J. F. (1933) Remarks on iteration. *Skand. Aktuarietidskr.*, **16**, 64–72.

Toth, A. & Kelley, C. T. (2015) Convergence analysis for Anderson acceleration. *SIAM J. Numer. Anal.*, **53**, 805–819.

Walker, H. F. & Ni, P. (2011) Anderson acceleration for fixed-point iterations. *SIAM J. Numer. Anal.*, **49**, 1715–1735.

Weniger, E. J. (1989) Nonlinear sequence transformations for the acceleration of convergence and the summation of divergent series. *Comput. Phys. Rep.*, **10**, 189–371.

Wimp, J. (1981) *Sequence Transformations and Their Applications*. New York: Academic Press.

Zhang, J., O'Donoghue, B. & Boyd, S. (2020) Globally convergent type-I Anderson acceleration for non-smooth fixed-point iterations. *SIAM J. Optim.*, **30**, 3170–3197.