Fast solvers for discretized Navier-Stokes problems using vector extrapolation

Duminil, Sebastien and Sadok, Hassane and Silvester, David

2014

MIMS EPrint: **2014.19**

Manchester Institute for Mathematical Sciences
School of Mathematics

The University of Manchester

Reports available from: http://eprints.maths.manchester.ac.uk/
And by contacting: The MIMS Secretary
School of Mathematics
The University of Manchester
Manchester, M13 9PL, UK

ISSN 1749-9097

Numer Algor (2014) 66:89–104
DOI 10.1007/s11075-013-9726-7

ORIGINAL PAPER

Fast solvers for discretized Navier-Stokes problems using vector extrapolation

Sebastien Duminil · Hassane Sadok · David Silvester

Received: 31 January 2013 / Accepted: 26 May 2013 / Published online: 30 June 2013

Abstract We discuss the design and implementation of a vector extrapolation method for computing numerical solutions of the steady-state Navier-Stokes equation system. We describe a "proof of concept" implementation of vector extrapolation, and we illustrate its effectiveness when integrated into the Incompressible Flow Iterative Solution Software (IFISS) package (http://www.manchester.ac.uk/ifiss).

Keywords Navier-Stokes problem · Vector extrapolation · Nonlinear system · Reduced Rank extrapolation · Picard method · Newton method

1 Introduction

An important problem that arises in many different areas of science and engineering is that of computing the limits of sequences of vectors (s_k), where $s_k \in \mathbb{R}^n$ with n large. Such sequences arise, for example, when solving systems of linear or nonlinear equations by fixed-point iterative methods, in which case $\lim_{k\to\infty} s_k$ is simply the desired solution. In many cases of interest, however, vector sequences converge to a limit too slowly. One practical way to accelerate the convergence is to use *vector extrapolation*. This transforms the original sequence into another sequence that converges to the same limit faster then the original one without having

S. Dumili · H. Sadok (✉)
Laboratoire de Mathématiques Pures et Appliquées, Université du Littoral, Calais, France
e-mail: sadok@lmpa.univ-littoral.fr

S. Dumili
e-mail: duminil@lmpa.univ-littoral.fr

D. Silvester
School of Mathematics, University of Manchester, Manchester, UK
e-mail: d.silvester@manchester.ac.uk

explicit knowledge of the sequence generator. In this paper we explore the potential of vector extrapolation in the context of computing steady state solutions of incompressible flow problems modelled by the Navier-Stokes equations. The fact that the Navier-Stokes system is nonlinear is what makes fluid dynamics so interesting. Indeed, boundary value problems associated with the Navier-Stokes equations can have more than one stable solution. Spatial discretization using mixed approximation of the velocity and pressure variables naturally lead us to consider high-dimensional nonlinear algebraic systems—so effective iteration methods are crucial for efficient computation. Existing vector extrapolation methods can be broadly classified into two categories: polynomial methods and ϵ-algorithms. The first family includes three approaches: minimal polynomial extrapolation (MPE); reduced rank extrapolation (RRE) and the modified minimal polynomial extrapolation (MMPE). The second family includes the topological ϵ-algorithm (TEA) and the scalar and vector ϵ-algorithms (SEA and VEA). We will restrict our attention to the RRE methodology in this work.

The paper is organised as follows. We review the Navier-Stokes problem and discuss conventional (Picard and Newton) nonlinear solution methods in Section 2. Then, in Section 3, we outline the idea of RRE extrapolation methods and we present a specific implementation which is efficient in terms of work and memory overhead. Finally, in Section 4, we give a performance comparison for two benchmark flow problems solved using the IFISS MATLAB toolbox [6].

2 The Navier-Stokes equations

Our notation is identical to that in Elman, Silvester & Wathen [7, ch. 7]. Our objective is to solve the following system of PDEs:

$$\begin{aligned} -\nu\nabla^2\vec{u} + \vec{u}\cdot\nabla\vec{u} + \nabla p &= \vec{f}, \\ \nabla\cdot\vec{u} &= 0, \end{aligned} \tag{1}$$

where $\nu > 0$ is a given constant associated with the kinematic viscosity. The variable $\vec{u}$ represents the velocity of the fluid and p represents the pressure. The system is nonlinear: the quadratic term $\vec{u}\cdot\nabla\vec{u}$ is the vector obtained by taking the convective derivative of each velocity component in turn.

The generic boundary value problem is the system (1) posed on a two- or a three-dimensional domain Ω, together with boundary conditions on $\partial\Omega = \partial\Omega_D \cup \partial\Omega_N$ given by

$$\vec{u} = \vec{w} \text{ on } \partial\Omega_D, \quad \nu\frac{\partial\vec{u}}{\partial n} - \vec{n}\,p = \vec{0} \text{ on } \partial\Omega_N, \tag{2}$$

where $\vec{n}$ denotes the outward-pointing normal to the boundary. If the velocity is specified everywhere on the boundary, that is, if $\partial\Omega_D \equiv \partial\Omega$, then the pressure solution to the Navier-Stokes problem (1)–(2) is only unique up to a hydrostatic constant.

2.1 Weak formulation and linearization

Our numerical experiments are restricted to model flow problems defined in two dimensions, see later. The extension of our methodology to three dimensions is completely natural, however. We thus denote by $(\cdot,\cdot)$ the Euclidean inner product in $\mathbb{R}^2$ and by $\|\cdot\|$ the corresponding norm. We denote by $L_2(\Omega)$ the space functions that are square-integrable in the sense of Lebesgue:

$$L_2(\Omega) := \left\{ u : \Omega \to \mathbb{R} \,\Big|\, \int_\Omega u^2 < \infty \right\}$$

and by $\mathcal{H}^1(\Omega)$ the Sobolev space given by

$$\mathcal{H}^1(\Omega) := \left\{ u : \Omega \to \mathbb{R} \,\big|\, u, \frac{\partial u}{\partial x}, \frac{\partial u}{\partial y} \in L_2(\Omega) \right\}.$$

We also define the standard solution and test spaces

$$H^1_E := \left\{ \vec{u} \in \mathcal{H}^1(\Omega)^2 \,|\, \vec{u} = \vec{w} \text{ on } \partial\Omega_D \right\},$$
$$H^1_{E_0} := \left\{ \vec{v} \in \mathcal{H}^1(\Omega)^2 \,|\, \vec{v} = \vec{0} \text{ on } \partial\Omega_D \right\}.$$

A weak formulation of (1)–(2) can then be stated as: Find $\vec{u} \in H^1_E$ and $p \in L_2(\Omega)$ such that

$$\nu \int_\Omega \nabla\vec{u} : \nabla\vec{v} + \int_\Omega (\vec{u}\cdot\nabla\vec{u})\cdot\vec{v} - \int_\Omega p\,(\nabla\cdot\vec{v}) = \int_\Omega \vec{f}\cdot\vec{v} \quad \forall\vec{v} \in H^1_{E_0} \tag{3}$$

$$\int_\Omega q\,(\nabla\cdot\vec{u}) = 0 \quad \forall q \in L_2(\Omega). \tag{4}$$

Solving (3)–(4) requires nonlinear iteration with a linearized problem being solve at every step. Thus, given an "initial guess" $(\vec{u_0}, p_0) \in H^1_E \times L_2(\Omega)$ a sequence of iterates $(\vec{u_0}, p_0), (\vec{u_1}, p_1), (\vec{u_2}, p_2), \ldots \in H^1_E \times L_2(\Omega)$ is computed, which converges to the solution of the weak formulation (3)–(4). In the next section, we recall the two classical linearization procedures: fixed point iteration and Newton's method.

Newton's iteration turns out to be a very natural approach. Given the iterate $(\vec{u_k}, p_k)$, we start by computing the nonlinear residual associated with the weak formulation. This is the pair $R_k(\vec{v}), r_k(q)$ satisfying

$$R_k(\vec{v}) = \int_\Omega \vec{f}\cdot\vec{v} - c(\vec{u_k};\vec{u_k},\vec{v}) - \nu\int_\Omega \nabla\vec{u_k} : \nabla\vec{v} + \int_\Omega p_k\,(\nabla\cdot\vec{v}),$$
$$r_k(q) = -\int_\Omega q\,(\nabla\cdot\vec{u_k})$$

for any $\vec{v} \in H^1_{E_0}$ and $q \in L_2(\Omega)$. The term c is the convection term. This can be identified with a trilinear form $c : H^1_{E_0} \times H^1_{E_0} \times H^1_{E_0} \to \mathbb{R}$ defined as follows:

$$c(\vec{z};\vec{u},\vec{v}) := \int_\Omega (\vec{z}\cdot\nabla\vec{u})\cdot\vec{v}.$$

With $\vec{u} = \vec{u_k} + \delta\vec{u_k}$ and $p = p_k + \delta p_k$, it is easy to see that the corrections $\delta\vec{u_k} \in H^1_{E_0}$, $\delta p_k \in L_2(\Omega)$ satisfy

$$\begin{aligned} D(\vec{u_k}, \delta\vec{u_k}, \vec{v}) + \nu \int_\Omega \nabla\delta\vec{u_k} : \nabla\vec{v} - \int_\Omega \delta p_k(\nabla\cdot\vec{v}) &= R_k(\vec{v}) \\ \int_\Omega q(\nabla\cdot\delta\vec{u_k}) &= r_k(q) \end{aligned} \tag{5}$$

for all $\vec{v} \in H^1_{E_0}$, $q \in L_2(\Omega)$, where $D(\vec{u_k}, \delta\vec{u_k}, \vec{v})$ is the difference in the nonlinear terms:

$$D(\vec{u_k}, \delta\vec{u_k}, \vec{v}) = c(\delta\vec{u_k}; \delta\vec{u_k}, \vec{v}) + c(\delta\vec{u_k}; \vec{u_k}, \vec{v}) + c(\vec{u_k}; \delta\vec{u_k}, \vec{v}).$$

Expanding $D(\vec{u_k}, \delta\vec{u_k}, \vec{v})$ and dropping the quadratic term $c(\delta\vec{u_k}; \delta\vec{u_k}, \vec{v})$ leads to the *linear* problem: for all $\vec{v} \in H^1_{E_0}$ and $q \in L_2(\Omega)$, find $\delta\vec{u_k} \in H^1_{E_0}$ and $\delta p_k \in L_2(\Omega)$ satisfying

$$\begin{aligned} c(\vec{u_k}; \delta\vec{u_k}, \vec{v}) + c(\delta\vec{u_k}; \vec{u_k}, \vec{v}) + \nu \int_\Omega \nabla\delta\vec{u_k} : \nabla\vec{v} - \int_\Omega \delta p_k(\nabla\cdot\vec{v}) &= R_k(\vec{v}) \\ \int_\Omega q(\nabla\cdot\delta\vec{u_k}) &= r_k(q). \end{aligned} \tag{6}$$

The solution of this system is the so-called Newton correction.

An even simpler linearization approach is Picard's method. Here the term $c(\delta\vec{u_k}; \delta\vec{u_k}, \vec{v})$ is dropped from (5) along with the linear term $c(\delta\vec{u_k}; \vec{u_k}, \vec{v})$. Thus, instead of (6), we have the following linear problem: for all $\vec{v} \in H^1_{E_0}$ and $q \in L_2(\Omega)$, find $\delta\vec{u_k} \in H^1_{E_0}$ and $\delta p_k \in L_2(\Omega)$ satisfying

$$\begin{aligned} c(\vec{u_k}; \delta\vec{u_k}, \vec{v}) + \nu \int_\Omega \nabla\delta\vec{u_k} : \nabla\vec{v} - \int_\Omega \delta p_k(\nabla\cdot\vec{v}) &= R_k(\vec{v}) \\ \int_\Omega q(\nabla\cdot\delta\vec{u_k}) &= r_k(q). \end{aligned} \tag{7}$$

The solution of this system is the Picard (or Oseen) correction. In both cases above, updating the previous iterate via $\vec{u}_{k+1} = \vec{u}_k + \delta\vec{u}_k$, $p_{k+1} = p_k + \delta p_k$ defines the next iterate in the sequence.

If we substitute $\delta\vec{u}_k = \vec{u}_{k+1} - \vec{u}_k$ and $\delta p_k = p_{k+1} - p_k$ into (7), we obtain an explicit definition for the new iterate: for all $\vec{v} \in H^1_{E_0}$ and $q \in L_2(\Omega)$, find $\vec{u}_{k+1} \in H^1_E$ and $p_{k+1} \in L_2(\Omega)$ such that

$$c(\vec{u}_k; \vec{u}_{k+1}, \vec{v}) + \nu \int_\Omega \nabla\vec{u}_{k+1} : \nabla\vec{v} - \int_\Omega p_{k+1}(\nabla\cdot\vec{v}) = \int_\Omega \vec{f}\cdot\vec{v} \tag{8}$$

$$\int_\Omega q(\nabla\cdot\vec{u}_{k+1}) = 0. \tag{9}$$

Comparing (8)–(9) with the weak formulation, we see that Picard's iteration corresponds to a simple fixed point iteration strategy for solving (3)–(4).

The main drawback of Newton's method is that the radius of the ball of convergence is typically proportional to the viscosity parameter ν. Thus, as the viscosity parameter is decreased (or equivalently, as the Reynolds number is increased) better and better initial guesses are needed in order for the Newton iteration to converge. The advantage of Picard's iteration is that, relative to the Newton iteration, it has a huge ball of convergence.

2.2 Mixed finite element approximation

A discrete weak formulation is defined using finite dimensional spaces $X_0^h \subset H_{E_0}^1$ and $M^h \subset L_2(\Omega)$. Specifically, given a velocity solution space X_E^h, the discrete version of (3)–(4) is: find $\vec{u}_{h,k+1} \in X_E^h$ and $p_{h,k+1} \in M^h$ such that

$$\nu \int_\Omega \nabla \vec{u}_{h,k+1} : \nabla \vec{v}_h + \int_\Omega (\vec{u}_{h,k} \cdot \nabla \vec{u}_{h,k+1}).\vec{v}_h - \int_\Omega p_{h,k+1}(\nabla \cdot \vec{v}_h) = \int_\Omega \vec{f} \cdot \vec{v}_h \quad \forall \vec{v}_h \in X_0^h,$$

$$\int_\Omega q_h(\nabla \cdot \vec{u}_{h,k+1}) = 0 \quad \forall q_h \in M^h.$$

Implementation entails defining appropriate bases for the finite element spaces, leading to a nonlinear system of algebraic equations. Linearization of this system using Newton's method gives the finite-dimensional analogue of (6): find corrections $\delta\vec{u}_{h,k} \in X_0^h$ and $\delta p_{h,k} \in M^h$ satisfying

$$\begin{aligned} c(\vec{u}_{h,k}; \delta\vec{u}_{h,k}, \vec{v}_h) + c(\delta\vec{u}_{h,k}; \vec{u}_{h,k}, \vec{v}_h) + \nu \int_\Omega \nabla \delta\vec{u}_{h,k} : \nabla \vec{v}_h - \int_\Omega \delta p_{h,k}(\nabla \cdot \vec{v}_h) &= R_k(\vec{v}_h), \\ \int_\Omega q_h(\nabla \cdot \delta\vec{u}_{h,k}) &= r_k(q_h), \end{aligned} \tag{10}$$

for all $\vec{v}_h \in X_0^h$ and $q_h \in M^h$. Dropping the term $c(\delta\vec{u}_{h,k}; \vec{u}_{h,k}, \vec{v}_h)$ from (10) gives the discrete analogue of Picard's method (7).

To define the associated linear algebra problem, we introduce a set of vector-valued basis functions $\{\vec{\phi}_j\}$, and write

$$\vec{u}_{h,k} = \sum_{j=1}^{n_u} u_{j,k}\vec{\phi}_j + \sum_{j=n_u+1}^{n_u+n_\delta} u_{j,k}\vec{\phi}_j, \qquad \delta\vec{u}_{h,k} = \sum_{j=1}^{n_u} \Delta u_{j,k}\vec{\phi}_j. \tag{11}$$

We also fix the coefficients $u_{j,k} : j = n_u + 1, \ldots, n_u + n_\delta$, so that the second term interpolates the boundary data on $\delta\Omega_D$. We then introduce a set of pressure basis functions $\{\psi_k\}$ and set

$$p_{h,k} = \sum_{l=1}^{n_p} p_{l,k}\psi_l, \qquad \delta p_{h,k} = \sum_{l=1}^{n_p} \Delta p_{l,k}\psi_l. \tag{12}$$

Substituting the expressions (11)–(12) into (10) gives a system of linear equations,

$$\begin{bmatrix} \nu A + N_k + W_k & B^T \\ B & O \end{bmatrix} \begin{bmatrix} \Delta u_k \\ \Delta p_k \end{bmatrix} = \begin{bmatrix} R_k \\ r_k \end{bmatrix} \tag{13}$$

where A is the vector-Laplacian matrix,

$$A = \left[a_{ij}\right], \quad a_{ij} = \int_\Omega \nabla\vec{\phi}_i : \nabla\vec{\phi}_j$$

B is the divergence matrix,

$$B = \left[b_{lj}\right], \quad b_{lj} = -\int_\Omega \psi_l \cdot \nabla\vec{\phi}_j$$

N_k is the vector-convection matrix,

$$N_k = \left[n_{ij}\right], \quad n_{ij} = \int_\Omega \left(\vec{u}_{h,k} \cdot \nabla\vec{\phi}_j\right) \cdot \vec{\phi}_i$$

and W_k is the Newton derivative matrix,

$$W_k = \left[w_{ij}\right], \quad w_{ij} = \int_\Omega \left(\vec{\phi}_j \cdot \nabla\vec{u}_{h,k}\right) \cdot \vec{\phi}_i$$

where $i = 1, \ldots, n_u$, $j = 1, \ldots, n_u$ and $l = 1, \ldots, n_p$. The right-hand side vectors in (13) are the nonlinear residuals associated with the current discrete solution estimates $\vec{u}_{h,k}$ and $p_{h,k}$, expanded via (11) and (12):

$$R_k = [R_i], \quad R_i = \int_\Omega \vec{f} \cdot \vec{\phi}_i - \int_\Omega \left(\vec{u}_{h,k} \cdot \nabla\vec{u}_{h,k}\right) \cdot \vec{\phi}_i - \nu \int_\Omega \nabla\vec{u}_{h,k} : \nabla\vec{\phi}_i + \int_\Omega p_{h,k} \left(\nabla \cdot \vec{\phi}_i\right),$$

$$r_k = [r_l], \quad r_l = \int_\Omega \psi_l \left(\nabla \cdot \vec{u}_{h,k}\right).$$

The system (13) is referred to as the discrete Newton problem. For Picard iteration, we omit the Newton derivative matrix to give the discrete Oseen problem:

$$\begin{bmatrix} \nu A + N_k & B^T \\ B & O \end{bmatrix} \begin{bmatrix} \Delta u_k \\ \Delta p_k \end{bmatrix} = \begin{bmatrix} R_k \\ r_k \end{bmatrix}. \tag{14}$$

Using stabilized mixed approximation, the zero block matrix in (13) and (14) is replaced by $-C$ a negative semi-definite matrix operator. Full details can be found in [7, Section 5.3].

An algorithm for solving a discretized Navier-Stokes problem using Picard's iteration is presented in Table 1. We will use this algorithm to compute numerical results in Section 4.

3 Vector extrapolation methods

Extrapolation methods are of interest whenever an iteration process converges slowly. For a survey of these methods see for example the review papers [14, 23] and the

Table 1 Picard iteration Algorithm

Step 1.	Input : the vector $x_0 = (\vec{u}_0,\ p_0)^T$, compute the vector $(R_0, r_0)^T$ and set $k = 0$.
Step 2.	Retrieve A, B, and C (if needed). Compute N_k.
	Solve the system (14) to find the vector $\Delta x_k = (\Delta\vec{u}_k, \Delta p_k)^T$.
	Set $x_{k+1} = x_k + \Delta x_k$.
	Compute the vector $(R_{k+1}, r_{k+1})^T$.
Step 3.	If $\left\|(R_{k+1}, r_{k+1})^T\right\| < tol$, stop. Otherwise set $k = k + 1$ and go to Step 2.

book [2]. The most popular vector extrapolation methods are the minimal polynomial extrapolation (MPE) of Cabay & Jackson [3], the reduced rank extrapolation (RRE) of Eddy [5] and Mesina [15], and the modified minimal polynomial extrapolation (MMPE) of Sidi, Ford & Smith [22], Brezinski [1] and Pugachev [16]. Convergence analysis of these methods can be found in [22] and also in the work of Jbilou & Sadok [12].

Some different recursive algorithms for implementing these methods have also been proposed in the literature, see [1, 2, 13, 20]. We note that, when applied to linearly generated vector sequences, the MPE, the RRE and the TEA methods are related to Krylov subspace methods. For example, Sidi's work [19] shows that the MPE and the RRE approaches are mathematically equivalent to Arnoldi's method [17] and to the generalized minimal residual method (GMRES) [18], respectively. Vector extrapolation methods are considered to be most effective when applied to nonlinear systems of equations, see [4, 10, 11, 14]. To keep things simple, we will focus on a representative approach—the Reduced Rank Extrapolation method—in the sequel.

To define the RRE method, we will consider the (linear or nonlinear) system of equations:

$$F(x) = 0; \qquad F : \mathbb{R}^N \to \mathbb{R}^N \tag{15}$$

whose solution is denote by s. Then, starting with a suitable vector s_0, an initial approximation to s, the sequence (s_n) can be generated by fixed-point iteration, so that

$$s_{n+1} = G(s_n), \quad n = 0, 1, \ldots \qquad G : \mathbb{R}^N \to \mathbb{R}^N \tag{16}$$

where $x - G(x) = 0$ represents a preconditioned form of (15), so that in case of convergence, $\lim_{n\to\infty} s_n = s$.

Let (s_n) be a sequence of vectors of $\mathbb{R}^N$ formed by (16), and let

$$u_n = \Delta s_n = s_{n+1} - s_n, \qquad n = 0, 1, \cdots$$

$$w_n = \Delta^2 s_n = \Delta s_{n+1} - \Delta s_n, \qquad n = 0, 1, \cdots$$

represent the first and the second forward differences of s_n, respectively. The RRE method, when applied to the sequence (s_n), can be shown to generate an

approximation $t_{n,k}^{RRE}$ of the limit or the antilimit of (s_n). This approximation is defined by

$$t_{n,k}^{RRE} = \sum_{j=0}^{k} \gamma_j^{(k)} s_{n+j}, \tag{17}$$

where

$$\sum_{j=0}^{k} \gamma_j^{(k)} = 1 \quad \text{and} \quad \sum_{j=0}^{k} \eta_{ij} \gamma_j^{(k)} = 0, \quad i = 0, \cdots, k-1, \tag{18}$$

with the scalars η_{ij} defined by the ℓ_2 inner product:

$$\eta_{ij} = \left(\Delta^2 s_{n+i}, \Delta s_{n+j} \right).$$

Alternatively, using (17) and (18), $t_{n,k}^{RRE}$ can be expressed as a ratio of two determinants

$$t_{n,k}^{RRE} = \frac{\begin{vmatrix} s_n & s_{n+1} & \cdots & s_{n+k} \\ \eta_{0,0} & \eta_{0,1} & \cdots & \eta_{0,k} \\ \vdots & \vdots & \vdots & \vdots \\ \eta_{k-1,0} & \eta_{k-1,1} & \cdots & \eta_{k-1,k} \end{vmatrix}}{\begin{vmatrix} 1 & 1 & \cdots & 1 \\ \eta_{0,0} & \eta_{0,1} & \cdots & \eta_{0,k} \\ \vdots & \vdots & \vdots & \vdots \\ \eta_{k-1,0} & \eta_{k-1,1} & \cdots & \eta_{k-1,k} \end{vmatrix}}. \tag{19}$$

We now let $\Delta^i S_{n,k-1}$ $(i = 1, 2)$ denote the matrices whose columns are $\Delta^i s_n, \ldots, \Delta^i s_{n+k-1}$. Using Schur's formula, $t_{n,k}^{RRE}$ can be written in matrix form as

$$t_{n,k}^{RRE} = s_n - \Delta S_{n,k-1} \Delta^2 S_{n,k-1}^{+} \Delta s_n, \tag{20}$$

where $\Delta^2 S_{n,k-1}^{+}$ is the Moore-Penrose generalized inverse of $\Delta^2 S_{n,k-1}$ defined by

$$\Delta^2 S_{n,k-1}^{+} = \left(\Delta^2 S_{n,k-1}^{T} \Delta^2 S_{n,k-1} \right)^{-1} \Delta^2 S_{n,k-1}^{T}.$$

$t_{n,k}^{RRE}$ exists and is unique if and only if $\det\left(\Delta^2 S_{n,k-1}^{T} \Delta^2 S_{n,k-1} \right) \neq 0$. For different values of n and k, $t_{n,k}^{RRE}$ may be efficiently computed using the algorithms proposed in [20]. Herein, we define a new approximation $\tilde{t}_{n,k}^{RRE}$ as follows

$$\tilde{t}_{n,k}^{RRE} = \sum_{j=0}^{k} \gamma_j^{(k)} s_{n+j+1}.$$

Then, following [14], we define the generalized residual of $t_{n,k}^{RRE}$ so that $\tilde{r}\left(t_{n,k}^{RRE}\right) = \tilde{t}_{n,k}^{RRE} - t_{n,k}^{RRE}$. This immediately leads to the characterization

$$\tilde{r}\left(t_{n,k}^{RRE}\right) = \Delta s_n - \Delta^2 S_{n,k-1} \Delta^2 S_{n,k-1}^{+} \Delta s_n,$$

and we find that $\tilde{r}\left(t_{n,k}^{RRE}\right)$ may be computed by projecting Δs_n orthogonally onto the subspace generated by the vectors $\Delta^2 s_n, \ldots, \Delta^2 s_{n+k-1}$.

From an implementation perspective, the case of primary interest is when n is kept fixed. Accordingly, we set $n = 0$ and we denote the matrices $\Delta^i S_{0,k-1}$, $(i = 1, 2)$ by $\Delta^i S_{k-1}$ and the vector $t_{0,k}^{RRE}$ by t_k^{RRE}. If we replace, in the numerator and in the denominator of (19), each column j, $j = 1, \cdots, k$ by its difference with the column $j + 1$, and then use Schur's formula, we get the new expression:

$$t_k^{RRE} = s_k - \Delta S_{k-1} \Delta^2 S_{k-1}^{+} \Delta s_k.$$

Let P represent the $k \times k$ lower triangular matrix with all elements equal to 1. The inverse P^{-1} is then a lower triangular matrix with diagonal entries equal to 1 and all entries below the diagonal equal to -1. Moreover,

$$t_k^{RRE} = s_k - \Delta S_{k-1} P P^{-1} \Delta^2 S_{k-1}^{+} \Delta s_k. \tag{21}$$

Next, by construction,

$$s_k = s_0 + \Delta S_{k-1} \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}.$$

Thus, setting $\gamma^{(k)} = P^{-1} \Delta^2 S_{k-1}^{+} \Delta s_k$, in (21), we obtain the simple update formula

$$t_k^{RRE} = s_0 + \Delta S_{k-1} \alpha^{(k)}, \quad k = 0, 1, \ldots \quad \text{with } \alpha^{(k)} = \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} - P\gamma^{(k)}.$$

Two numerically stable and efficient algorithms for computing $t_{n,k}^{RRE}$ with $n \geq 0$ can be found in the literature, see [9, 20]. A key feature of both of these algorithms is the solution of a least–squares problem using QR factorization. A generic construction is given below. We follow the description given by Sidi in [21].

First, it is convenient notationally to set

$$U_j = \left[u_n \,|\, u_{n+1} \,|\, \cdots \,|\, u_{n+j} \right], \qquad j = 0, 1, \cdots$$

where $u_k = \Delta s_k$. Let us assume that U_j has full rank, namely $\text{rank}(U_j) = j + 1$. Then, it has a QR factorization $U_j = Q_j R_j$, where $Q_j \in \mathbb{R}^{N \times (j+1)}$ is unitary and $R_j \in \mathbb{R}^{(j+1) \times (j+1)}$ is upper triangular with positive diagonal entries,

$$Q_j = \left[q_0 \,|\, q_1 \,|\, \cdots \,|\, q_j \right] \in \mathbb{R}^{N \times (j+1)}, \quad Q_j^T Q_j = I_{(j+1) \times (j+1)}$$

Table 2 RRE Algorithm

Step 0.	Input: the vectors $s_n, s_{n+1}, \cdots, s_{n+k+1}$
Step 1.	Compute $u_i = \Delta s_i = s_{i+1} - s_i, \quad i = n, n+1, \cdots, n+k.$
	Set $U_j = \left[u_n \,|\, u_{n+1} \,|\, \cdots \,|\, u_{n+j} \right], \quad j = 0, 1, \cdots$
	Compute the QR factorization of U_k, namely $U_k = Q_k R_k$.
	($U_{k-1} = Q_{k-1} R_{k-1}$ is contained in $U_k = Q_k R_k$).
Step 2.	Solve the linear system
	$R_k^T R_k d = e; \;\; d = [d_0, d_1, \cdots, d_k]^T, e = [1, 1, \cdots, 1]^T.$
	(This amounts to solving two triangular systems.)
	Set $\lambda = \sum_{i=0}^{k} d_i$.
	Set $\gamma_i = (1/\lambda)\, d_i$ for $i = 0, \cdots, k$.
Step 3.	Compute $\alpha = \left[\alpha_0, \alpha_1, \cdots, \alpha_{k-1}\right]^T$ via
	$\alpha_0 = 1 - \gamma_0; \;\; \alpha_j = \alpha_{j-1} - \gamma_j, \;\; j = 1, \cdots, k-1.$
	Compute $t_{n,k}^{RRE}$ via
	$t_{n,k}^{RRE} = s_n + Q_{k-1}(R_{k-1}\alpha).$

$$R_j = \begin{bmatrix} r_{00} & r_{01} & r_{02} & \cdots & r_{0j} \\ & r_{11} & r_{12} & \cdots & r_{1j} \\ & & r_{22} & \cdots & r_{2j} \\ & & & \ddots & \vdots \\ & & & & r_{jj} \end{bmatrix}, \qquad r_{ii} > 0.$$

Also, Q_j is obtained from Q_{j-1} by appending one column (the vector q_j) to the end of the latter. Similarly, R_j is obtained from R_{j-1} by appending one row of zeros and one column $\left(\left[r_{0j}, r_{1j}, \cdots, r_{jj}\right]\right)$ to the end of the latter. The details of the resulting algorithm are summarized in Table 2. Note that, in this algorithm, we need to store only the vector s_n and the matrix Q_k. The rest can be overwritten. (Actually, q_k does not need to be computed as it is not needed for determining $t_{n,k}^{RRE}$.) The QR factorization can be carried out inexpensively by applying the modified Gram–Schmidt process to the vectors $s_n, s_{n+1}, \cdots, s_{n+j}$ (see [20]).

The basic RRE algorithm in Table 2 becomes increasingly expensive as k is increased. A much more cost-effective strategy in practice is to periodically restart the RRE process. When applying the RRE method in their complete form to solve linear and nonlinear systems of equations, the work and storage requirements grow linearly with the number of iteration steps. A good way to control this is to use this method in the cycling mode. This means that we have to restart the algorithm after a fixed number of iterations. This leads to the algorithm that is presented in Table 3.

In practice, it is useful to apply the RRE method to an equivalent preconditionned system. The new system will be chosen such that the new basic iteration is convergent. In this case, the application of the extrapolation scheme after a number of basic iterations is recommended.

Table 3 RRE with restarts every m steps

Step 0.	Input: set $k = 0$, choose an integer m and the vectors s_0.
Step 1.	Generate $s_{j+1} = G(s_j), \quad j = 0, \cdots, m$ (see (16)).
Step 2.	Compute the approximation t_m^{RRE} using the RRE Algorithm in Table 2.
Step 3.	If t_m^{RRE} satisfies accuracy test, stop.
	Otherwise, set $s_0 = s_m^{RRE}$, $k = k + 1$ and go to Step 1.

4 Numerical experiments

All the computational experiments presented here were performed using MATLAB 7.1 with the most recent version of the IFISS toolbox [24]. For all tests, the iteration is stopped as soon as the nonlinear residual computed in Algorithm 1 is less than 10^{-8}. The standard nonlinear strategy that is built in to IFISS is to run a fixed number of Picard iterations followed by (a small number) of Newton iterations. This strategy is referred to here as *Picard–Newton*. Our aim here is to explore the potential for accelerating the fixed-point (Picard) iteration using the restarted RRE method discussed in Section 3. Results are presented for two different flow problem configurations and have been computed using two different approximation strategies. Further details of the mixed approximation issues can be found in [7, ch. 5]. In the sequel, we will give only the results obtained by RRE method, since we conducted tests with three polynomial vector extrapolation methods (RRE, MPE and MMPE) and RRE method is the one that gave the best results.

4.1 Flow over a backward facing step

This example represents flow in a rectangular duct with a sudden expansion. A Poiseuille flow profile is imposed on the inflow boundary ($x = -1; 0 \leq y \leq 1$), and no-flow (zero velocity) condition is imposed on the walls. The Neumann condition is applied at the outflow boundary ($x = 40; -1 \leq y \leq 1$) and automatically sets the mean outflow pressure to zero. Figure 1 shows streamlines and a three-dimensional rendering of the pressure solution when the kinematic viscosity ν is set to $1/600$. (Both plots are stretched in the vertical direction for added clarity.) Note that for this specific value of ν the flow is close to the stability limit which marks the transition from steady to unsteady flow.[1] A full discussion of this flow problem can be found in Gresho et al. [8]. The spatial resolution is such that the solution does not change in the "eyeball" norm when using a finer grid.

Figures 2, 3 and 4 show the evolution of the nonlinear residual norm, using a logarithmic scale, for the simple Picard method, the Picard–Newton method and non-restarted RRE method. Results are presented for two types of finite element discretization: a high order $Q_2 - P_{-1}$ solution and a low order $Q_1 - P_0$ solution—both computed on the same (uniform–) grid. The sizes of system are $N = 14946$ and

[1] The case $\nu = 1/600$ gives a Reynolds number of 800 with the non-dimensionalisation in [8].

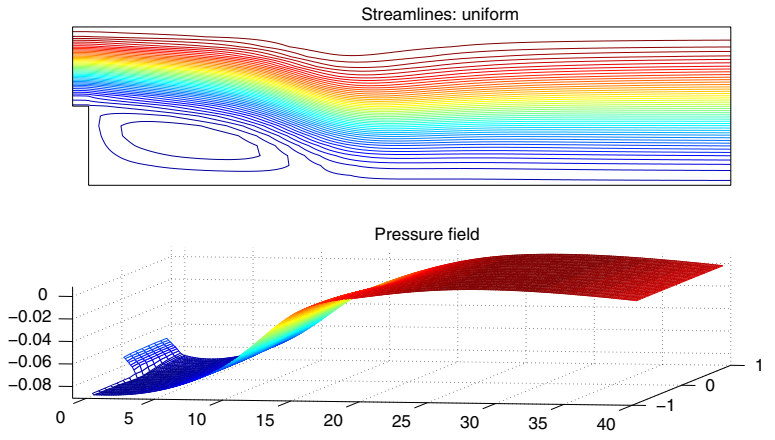

Fig. 1 Solution for $\nu = 1/600$ using $Q_2 - P_{-1}$ approximation

$N = 16242$ respectively. The three figures show results for three different values of ν which correspond to flow Reynolds numbers of $Re = 320$ (Fig. 2), $Re = 400$ (Fig. 3) and $Re = 800$ (Fig. 4), respectively.

The first observation is that the RRE method can be seen to accelerate the Picard iteration in every case. The bad news is that there is a Reynolds number dependent transition point before which RRE does not speed up convergence. For example, for $\nu = 3/800$, RRE is only helpful after $\sim$ 20 iterations, whereas for $\nu = 1/600$ the transition point is $\sim$ 40 iterations. A key issue in practice is that of guessing the time to switch to Newton iteration (for this problem, if the switch is made too early then Newton diverges!). The graphs suggest that the RRE method can help with this decision. If the switch is made after the transition point then Newton seems to converge. More experiments are needed to determine if this is a generic feature or not.

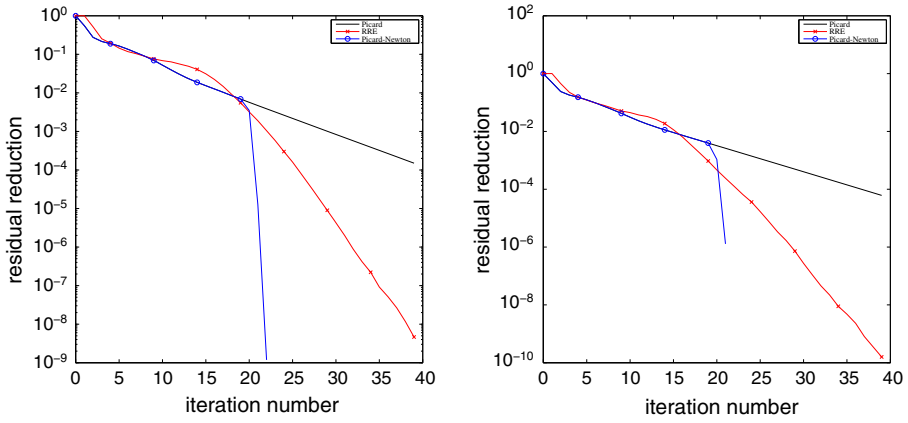

Fig. 2 Nonlinear convergence for $\nu = 3/800$ with $Q_2 - P_{-1}$ (*up*) and stabilized $Q_1 - P_0$ (*down*)

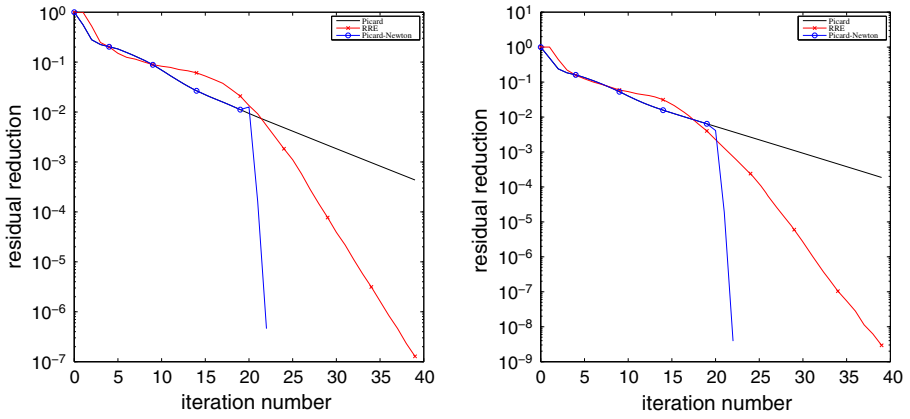

Fig. 3 Nonlinear convergence for $\nu = 1/300$ with $Q_2 - P_{-1}$ (*up*) and stabilized $Q_1 - P_0$ (*down*)

4.2 Flow over an obstacle

This example represents flow in a rectangular duct with a square obstacle. A Poiseuille flow profile is imposed on the inflow boundary ($x = 0; -1 \leq y \leq 1$), and no-flow (zero velocity) condition is imposed on the walls. The Neumann condition is again applied at the outflow boundary ($x = 8; -1 \leq y \leq 1$) and automatically sets the mean outflow pressure to zero. Figure 5 shows an equi-spaced streamline solution (top) and a pressure surface plot (bottom) in the case $\nu = 1/600$ computed using using stabilized $Q_1 - P_0$ approximation. Using $Q_2 - P_{-1}$ approximation on the same grid gives a visually identical picture. For this problem, the size of the system is $N = 3200$.

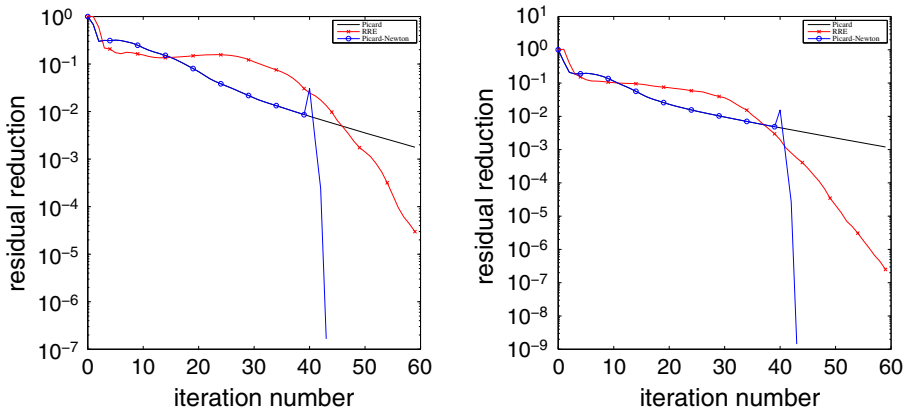

Fig. 4 Nonlinear convergence for $\nu = 1/600$ with $Q_2 - P_{-1}$ (*up*) and stabilized $Q_1 - P_0$ (*down*)

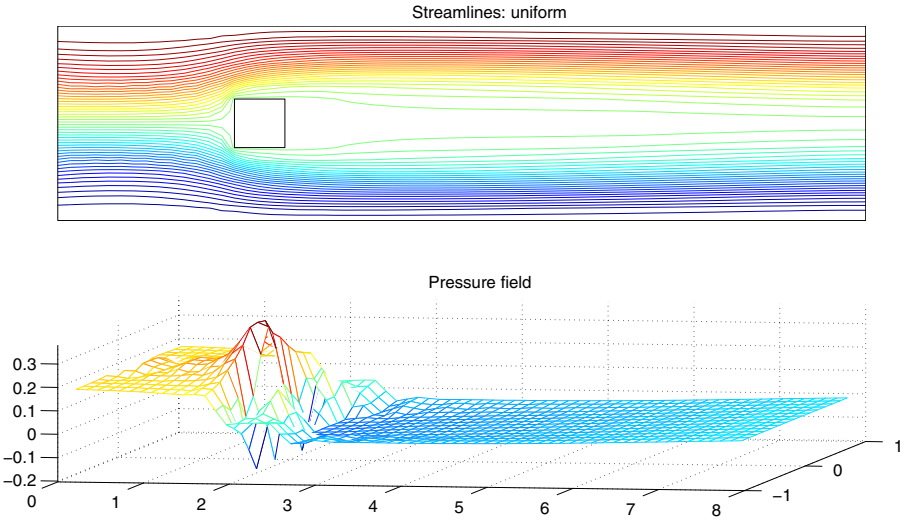

Fig. 5 Solution for $\nu = 1/600$ using stabilized $Q_1 - P_0$

Figures 6 and 7 show the evolution of the nonlinear residual norm, for the simple Picard method, the Picard–Newton method, the non-restarted RRE method and the restarted RRE–m method. The three figures show results for two different values of ν: namely of $\nu = 1/600$ (Fig. 6), and $\nu = 1/800$ (Fig. 7).

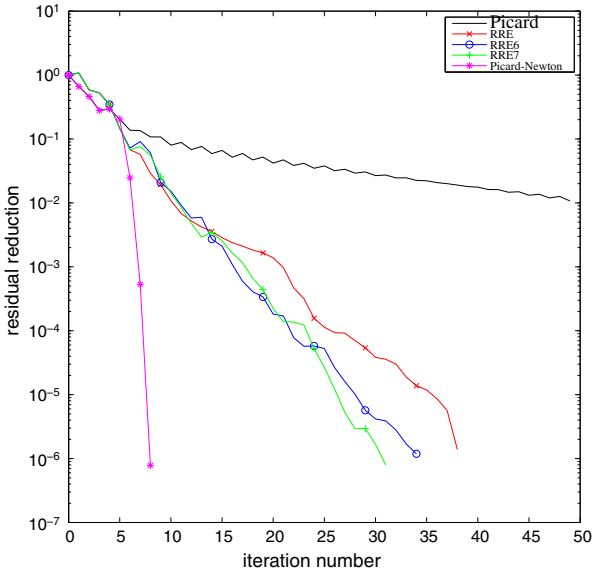

Fig. 6 Nonlinear convergence for $\nu = 1/600$ with stabilized $Q_1 - P_0$

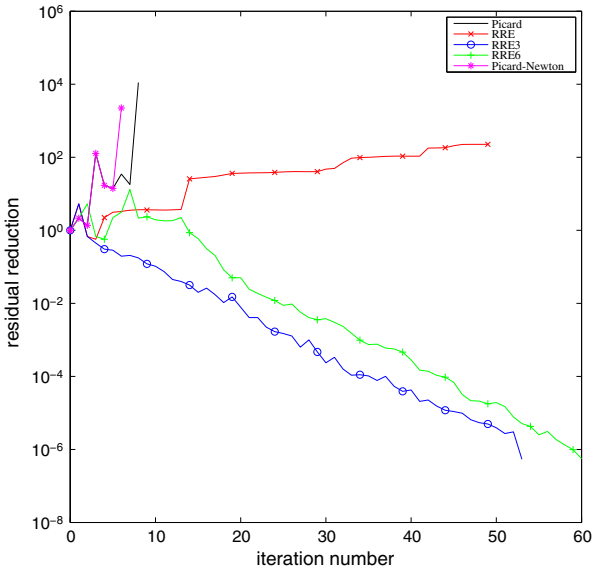

Fig. 7 Nonlinear convergence for $\nu = 1/800$ with stabilized $Q_1 - P_0$

Looking first at Fig. 6 we see that the transition point where it pays to use RRE is much earlier than in the first example, only $\sim$ 5 iterations. We note that switching to Newton at the transition point again leads to a convergent iteration. Whilst the restarted methods are slightly faster that the basic RRE method, the need to choose m a priori is an additional headache. Turning to Fig. 7 we see that the only methods that give convergence to a steady state when the Reynolds number is increased ($\nu = 1/800$) are the two restarted methods RRE–3 and RRE–6. This suggests that these methods could be useful when trying to compute steady flow solutions in cases which are close to (perhaps beyond) a bifurcation point.

5 Conclusion

Vector extrapolation methods may have a useful role to play in computing steady-state solutions to incompressible flow problems. Further research is needed to determine if extrapolation can provide a automatic way of switching between fixed point iteration and Newton iteration when solving flow problems that have steady solutions, but which are close to a (Hopf) bifurcation point to a periodic unsteady solution. In this paper we studied the steady-state Navier Stokes equation system which is time independent. Moreover the computational cost of RRE used in each iteration is negligible with respect to the cost needed for the solution of the PDE.

References

1. Brezinski, C.: Généralisation de la transformation de Shanks, de la table de Padé et de l'epsilon-algorithm. Calcolo **12**, 317–360 (1975)
2. Brezinski, C., Redivo Zaglia, M.: Extrapolation Methods Theory and Practice. North-Holland, Amsterdam (1991)
3. Cabay, S., Jackson, L.W.: A polynomial extrapolation method for finding limits and antilimits for vector sequences. SIAM J. Numer. Anal. **13**, 734–752 (1976)
4. Duminil, S., Sadok, H.: Reduced rank extrapolation applied to electronic structure computations. Electron. Trans. Numer. Anal. **38**, 347–362 (2011)
5. Eddy, R.P.: Extrapolation to the limit of a vector sequence. In: Wang, P.C.C. (ed.) Information Linkage Between Applied Mathematics and Industry, pp. 387–396. Academic Press, New-York (1979)
6. Elman, H., Ramage, A., Silvester, D.: Algorithm 866: IFISS, a Matlab toolbox for modelling incompressible flow. ACM Trans. Math. Softw. **33**, 2–14 (2007)
7. Elman, H., Silvester, D., Wathen, A.: Finite Elements and Fast Iterative Solvers: Eith Applications in Incompressible Fluid Dynamics. Oxford University Press, Oxford (2005). ISBN: 978-0-19-852868-5; 0-19-852868-X
8. Gresho, P.M., Gartling, D.K., Torczynski, J.R., Cliffe, K.A., Winters, K.H., Garratt, T.J., Spence, A., Goodrich, J.W.: Is the steady viscous incompressible 2D flow over a backward facing step at Re = 800 stable. Int. J. Numer. Methods Fluids **17**, 501–541 (1993)
9. Jbilou, K.: A general projection algorithm for solving linear systems of equations. Numer. Algoritm. **4**, 361–377 (1993)
10. Jbilou, K., Reichel, L., Sadok, H.: Vector extrapolation enhanced TSVD for linear discrete ill-posed problems. Numer. Algoritm. **51**, 195–208 (2009)
11. Jbilou, K., Sadok, H.: Some results about vector extrapolation methods and related fixed point iterations. J. Comput. Appl. Math. **36**, 385–398 (1991)
12. Jbilou, K., Sadok, H.: Analysis of some vector extrapolation methods for linear systems. Numer. Math. **70**, 73–89 (1995)
13. Jbilou, K., Sadok, H.: LU-implementation of the modified minimal polynomial extrapolation method. IMA J. Numer. Anal. **19**, 549–561 (1999)
14. Jbilou, K., Sadok, H.: Vector extrapolation methods. Applications and numerical comparison. J. Comp. Appl. Math. **122**, 149–165 (2000)
15. Mešina, M.: Convergence acceleration for the iterative solution of $x = Ax + f$. Comput. Methods Appl. Mech. Eng. **10**(2), 165–173 (1977)
16. Pugatchev, B.P.: Acceleration of the convergence of iterative processes and a method for solving systems of nonlinear equations. U.S.S.R. Comput. Math. Math. Phys. **17**, 199–207 (1978)
17. Saad, Y.: Krylov subspace methods for solving large unsymmetric linear systems. Math. Comput. **37**, 105–126 (1981)
18. Saad, Y., Schultz, M.H.: GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. SIAM J. Sci. Stat. Comput. **7**, 856–869 (1986)
19. Sidi, A.: Extrapolation vs. projection methods for solving linear systems of equations. J. Comput. Appl. Math. **22**, 71–88 (1988)
20. Sidi, A.: Efficient implementation of minimal polynomial and reduced rank extrapolation methods. J. Comput. Appl. Math. **36**, 305–337 (1991)
21. Sidi, A.: Vector extrapolation methods with applications to solution of large systems of equations and to Page rank computations. Comput. Math. Appl. **56**, 1–24 (2008)
22. Sidi, A., Ford, W.F., Smith, D.A.: Acceleration of convergence of vector sequences. SIAM J. Numer. Anal. **23**, 178–196 (1986)
23. Smith, D.A., Ford, W.F., Sidi, A.: Extrapolation methods for vector sequences. SIAM Rev. **29**, 199–233 (1987)
24. Silvester, D., Elman, H., Ramage, A.: Incompressible Flow and Iterative Solver Software (IFISS) version 3.2. Available online at http://www.manchester.ac.uk/ifiss (2012)