Journal of Computational Physics 347 (2017) 1–19

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

# Anderson acceleration and application to the three-temperature energy equations ☆

Hengbin An [a,*], Xiaowei Jia [b], Homer F. Walker [c]

[a] Laboratory of Computational Physics, Institute of Applied Physics and Computational Mathematics, Beijing 100094, China
[b] Graduate School of China Academy Engineering Physics, Beijing 100083, China
[c] Department of Mathematical Sciences, Worcester Polytechnic Institute, Worcester, MA 01609-2280, USA

ARTICLE INFO

*Article history:*
Received 21 June 2016
Received in revised form 16 June 2017
Accepted 18 June 2017
Available online 23 June 2017

*Keywords:*
Anderson acceleration
Picard method
Fixed-point iteration
Jacobian-free Newton–Krylov
Iteration acceleration
Three-temperature energy equations

ABSTRACT

The Anderson acceleration method is an algorithm for accelerating the convergence of fixed-point iterations, including the Picard method. Anderson acceleration was first proposed in 1965 and, for some years, has been used successfully to accelerate the convergence of self-consistent field iterations in electronic-structure computations. Recently, the method has attracted growing attention in other application areas and among numerical analysts.

Compared with a Newton-like method, an advantage of Anderson acceleration is that there is no need to form the Jacobian matrix. Thus the method is easy to implement. In this paper, an Anderson-accelerated Picard method is employed to solve the three-temperature energy equations, which are a type of strong nonlinear radiation-diffusion equations. Two strategies are used to improve the robustness of the Anderson acceleration method. One strategy is to adjust the iterates when necessary to satisfy the physical constraint. Another strategy is to monitor and, if necessary, reduce the matrix condition number of the least-squares problem in the Anderson-acceleration implementation so that numerical stability can be guaranteed. Numerical results show that the Anderson-accelerated Picard method can solve the three-temperature energy equations efficiently. Compared with the Picard method without acceleration, Anderson acceleration can reduce the number of iterations by at least half. A comparison between a Jacobian-free Newton–Krylov method, the Picard method, and the Anderson-accelerated Picard method is conducted in this paper.



## 1. Introduction

In many scientific and engineering computing areas, large-scale nonlinear equations need to be solved. Newton-based methods and Picard methods are two main classes of nonlinear iterative methods for solving these nonlinear equations. Newton–Krylov methods, in which Krylov-subspace methods are used to solve the Newton linearized equations, constitute a class of popular algorithms. These have been developed primarily since the early 1990s [11] and have been used successfully

☆ This project was supported by National Natural Science Foundation of China (No. 11671051, No. 61370066), Science Challenge Program (No. JCKY2016212A502) and National Key R&D Program of China (No. 2017YFA0603903). Some of the work was implemented on the *Turing* machine at WPI, supported by US National Science Foundation Major Research Instrumentation grant DMS-1337943 to Worcester Polytechnic Institute, while the first author was visiting WPI.

* Corresponding author.
*E-mail addresses:* an_hengbin@iapcm.ac.cn (H. An), dutjxw@163.com (X. Jia), walker@wpi.edu (H.F. Walker).

http://dx.doi.org/10.1016/j.jcp.2017.06.031
0021-9991/

in many areas. At the same time, Picard methods have also been widely used in many complex numerical simulation applications.

Newton–Krylov methods have many advantages. Perhaps the most notable advantage is that there is no need to form the Jacobian matrix, which may be very difficult or expensive to obtain in some complex applications [11]. Additionally, appropriately implemented Newton–Krylov methods, like many other Newton-based methods, exhibit rapid local convergence that, in particular, is typically independent of the mesh size on discretized nonlinear partial differential equation (PDE) problems [10,17].

On the other hand, Newton–Krylov methods may have some disadvantages. In a Jacobian-free Newton–Krylov method, in which Jacobian-vector products are approximated by finite differences, the errors in the finite-difference scheme may adversely affect the robustness of the method, especially in complex multi-scale applications. Additionally, compared to Picard methods, the linear systems in Newton-based and Newton–Krylov methods may be more difficult to solve. For example, the Jacobian matrix of the three-temperature energy equations is nonsymmetric, while the Picard linearized matrix is symmetric. Although a Jacobian-free Newton–Krylov method does not require computing and saving the Jacobian matrix, a preconditioning matrix may still be needed in many cases.

Picard methods are often preferred over Newton-based methods because they are relatively easier to implement. However, as fixed-point iterations, their convergence is usually linear and may be undesirably slow. Recently, a kind of nonlinear acceleration method, the Anderson acceleration method, has attracted considerable attention as a means of mitigating slow convergence of fixed-point iterations [1,8,14–16,5,18,19]. This method was first proposed in 1965 by Anderson to solve a class of nonlinear integral equations [1]. It has since been used successfully in electronic-structure computations to accelerate the convergence of self-consistent field iterations. (In electronic-structure computations and some other applications, Anderson acceleration is called Anderson mixing because of its physical association with charge mixing.) Anderson acceleration has also been used in simulating fluid–structure interactions. In fact, the quasi-Newton inverse least squares (QN-ILS) method [9] for solving fluid–structure problems is essentially the same as Anderson acceleration.

In 2009, Fang and Saad discussed the relationship between Anderson acceleration and certain quasi-Newton methods [5]. They proposed two classes of general quasi-Newton methods, the Broyden-like class and the nonlinear Eirola–Nevanlinna-type methods. In particular, the Anderson acceleration method is a special case of the Broyden-like class methods.

In 2011, Walker and Ni further discussed Anderson acceleration [16]. They showed that, on linear fixed-point problems, Anderson acceleration is “essentially equivalent” in a certain sense to the generalized minimal residual (GMRES) method for linear equations. Subsequently, Potra and Engler also considered Anderson acceleration and gave a more extensive characterization of the method when it is used to solve linear problems [14]. The first convergence result for Anderson acceleration on general nonlinear problems was given by Toth and Kelley in 2015 [19]. They proved that the Anderson acceleration method is locally r-linear convergent under reasonable assumptions, which they showed to be satisfied in their experiments.

In recent years, there have been a number of new applications of Anderson acceleration. For example, Lott, Walker, Woodward and Yang considered the use of Anderson acceleration with Picard iterations in variably saturated flow applications [13]. Willert, Taitano and Knoll applied Anderson acceleration to improve the convergence of Picard iterations for solving two classes of transport equations [18]. (They also considered combinations of Anderson acceleration and moment-based acceleration methods.) In both of these studies, the results show that Anderson acceleration can improve not only the convergence rate but also the robustness of the Picard method. Lipnikov, Svyatskiy and Vassilevski used Anderson acceleration to accelerate the Picard method for steady-state advection–diffusion equations discretized by a positivity-preserving finite-volume scheme [12]. In particular, they introduced an inexact modification of Anderson acceleration with an adaptive strategy for choosing linear-solver tolerances that significantly improved efficiency in their experiments.

The three-temperature energy equations are a kind of strong nonlinear radiation-diffusion model [2,6]. In [2], a Jacobian-free Newton–Krylov method was employed to solve the discretized equations. Based on the character of the energy-front propagation, an efficient method for choosing an initial iterate for the two-dimensional case was given.

In this paper, an Anderson-accelerated Picard method is used to solve the three-temperature energy equations. To improve the robustness of Anderson acceleration, some modifications were made. First, modifications were made so that Anderson acceleration preserves certain physical properties (for example, the non-negativity of temperature) during the iterations. Second, a strategy like that in [16] and [15] was used to monitor the matrix condition number in the least-squares problem in Anderson acceleration and, if necessary, to take steps to improve it in order to retain the effectiveness of the method. Our numerical results show that this modified Anderson acceleration method can improve the convergence rate of the Picard method when it is used to solve the three-temperature energy equations.

This paper is organized as follows: in Section 2, Anderson acceleration and its implementation are discussed; in Sections 3 and 4, the modifications of Anderson acceleration are described; the three-temperature energy equations are introduced in Section 5; some numerical results are presented in Section 6; and finally the conclusion is given in Section 7.

## 2. The Anderson acceleration method

In many scientific and engineering computing applications, it is necessary to solve a system of nonlinear equations, which we write generically as

$$F(x) = 0, \quad F : \mathbb{R}^n \to \mathbb{R}^n. \tag{1}$$

Often the system (1) can be expressed equivalently as a fixed-point problem

$$x = G(x), \quad G : \mathbb{R}^n \to \mathbb{R}^n, \tag{2}$$

such that

$$x^* = G(x^*) \Longleftrightarrow F(x^*) = 0.$$

Based on the fixed-point problem (2), Algorithm 1 can be given.

**Algorithm 1** Fixed-point iteration (FPI).

1: Given $x_0$.
2: **for** $k = 0, 1, \ldots$ **do**
3: Set $x_{k+1} = G(x_k)$.
4: **end for**

In many applications, one of the main advantages of Algorithm 1 is that it is easy to implement. In addition, it is often possible to construct $G$ so that desirable physical characteristics of the solution are preserved during the iteration process. However, the FPI iterates can only be guaranteed to converge if $G$ is a contraction mapping, at least locally. Additionally, FPI is usually only linearly convergent in practice and undesirably slow in many cases.

To improve the convergence rate of FPI, we consider Anderson acceleration [1], formulated as in [16] as follows:

**Algorithm 2** Anderson acceleration (AA).

1: Given $x_0$ and $m \geq 1$.
2: Set $x_1 = G(x_0)$.
3: **for** $k = 1, 2, \ldots$ **do**
4: Set $m_k = \min\{m, k\}$.
5: Compute $G(x_k)$ and let $f_k = G(x_k) - x_k$.
6: Set $F_k = (f_{k-m_k}, \ldots, f_k)$.
7: Determine $\alpha^{(k)} = (\alpha_0^{(k)}, \ldots, \alpha_{m_k}^{(k)})^T$ that solves

$$\begin{cases} \min_{\alpha=(\alpha_0,\ldots,\alpha_{m_k})^T} \|F_k\alpha\|_2 \\ \text{s.t.} \sum_{i=0}^{m_k} \alpha_i = 1 \end{cases} \tag{3}$$

8: Set $x_{k+1} = \sum_{i=0}^{m_k} \alpha_i^{(k)} G(x_{k-m_k+i})$.
9: **end for**

We note that the original Anderson acceleration method in [1] allows a more general form

$$x_{k+1} = (1 - \beta_k) \sum_{i=0}^{m_k} \alpha_i^{(k)} x_{k-m_k+i} + \beta_k \sum_{i=0}^{m_k} \alpha_i^{(k)} G(x_{k-m_k+i}),$$

where $\beta_k > 0$ is a relaxation parameter. In some application areas, such as electronic-structure computations, $\beta_k$ is called the Anderson mixing coefficient, and the Anderson acceleration method is called the Anderson mixing method. In this paper, as in [16], we consider only the case $\beta_k = 1$ in Algorithm 2.

The idea of Algorithm 2 is to define the $(k+1)$-th iterate as a combination of the $G$ values, in which the coefficients are determined by minimizing the norm of an affine combination of residual vectors.

In Algorithm 2, at most $m+1$ residual vectors are saved at each iteration. At the $k$th iteration, if $k < m$, then the latest residual vector $f_k$ will be appended to the matrix $F_k$ on the right; if $k \geq m$, then the oldest residual vector $f_{k-m_k}$ is deleted from $F_k$ on the left as well.

At the $k$-th iteration, $m_k + 1 \leq m$ vectors are saved in $F_k$. For purpose of discussion, $m_k$ in Algorithm 2 is called the *Anderson depth*. The maximal depth is limited by the parameter $m$, and we often denote the method by AA($m$) in the following. Note that if $m = 0$, then Anderson acceleration becomes Algorithm 1.

### 2.1. Form of the least-squares problem

In each Anderson acceleration iteration, a constrained least-squares problem (3) must be solved. In most references, this least-squares problem is transformed into an unconstrained least-squares problem [5,16]. In the following, we use the notation in [15] to introduce the transformation process.

Let

$$\Delta f_i = f_{i+1} - f_i, \quad i = k - m_k, \ldots, k-1$$

and

$$\mathcal{F}_k = (\Delta f_{k-m_k}, \ldots, \Delta f_{k-1}), \tag{4}$$

then least-squares problem (3) is equivalent to

$$\min_{\gamma=(\gamma_0,\ldots,\gamma_{m_k-1})^T} \| f_k - \mathcal{F}_k \gamma \|_2, \tag{5}$$

where $\alpha$ and $\gamma$ are related by

$$\alpha_i = \begin{cases} \gamma_0, & i = 0 \\ \gamma_i - \gamma_{i-1}, & 1 \le i \le m_k - 1 \\ 1 - \gamma_{m_k-1}, & i = m_k \end{cases}$$

and

$$\gamma_i = \sum_{j=0}^{i} \alpha_j, \quad i = 0, 1, \ldots, m_k - 1.$$

Based on the unconstrained least-squares problem (5), a different form of the Anderson acceleration algorithm can be given. In fact, if the solution of (5) is given by $\gamma^{(k)} = (\gamma_0^{(k)}, \ldots, \gamma_{m_k-1}^{(k)})^T$, then

$$x_{k+1} = G(x_k) - \sum_{i=0}^{m_k-1} \gamma_i^{(k)} [G(x_{k-m_k+i+1}) - G(x_{k-m_k+i})] = G(x_k) - \mathcal{G}_k \gamma^{(k)},$$

where

$$\mathcal{G}_k = (\Delta G_{k-m_k}, \ldots, \Delta G_{k-1}) \tag{6}$$

with

$$\Delta G_i = G(x_{i+1}) - G(x_i), \quad i = k - m_k, \ldots, k - 1.$$

We now give a more specific version of the Anderson acceleration algorithm in Algorithm 3.

**Algorithm 3** Anderson acceleration (AA).

1: Given $x_0$ and $m \ge 1$.
2: Set $x_1 = G(x_0)$.
3: **for** $k = 1, 2, \ldots$ **do**
4: Set $m_k = \min\{m, k\}$.
5: Compute $G(x_k)$ and let $f_k = G(x_k) - x_k$.
6: Update $\mathcal{F}_k$ and $\mathcal{G}_k$ by (4) and (6).
7: Determine $\gamma^{(k)} = (\gamma_0^{(k)}, \ldots, \gamma_{m_k-1}^{(k)})^T$ that solves

$$\min_{\gamma=(\gamma_0,\ldots,\gamma_{m_k-1})^T} \| f_k - \mathcal{F}_k \gamma \|_2.$$

8: Set $x_{k+1} = G(x_k) - \mathcal{G}_k \gamma^{(k)}$.
9: **end for**

### 2.2. Solution of the least-squares problem

In this paper, the solution method for the least-squares problem is based on QR decomposition as suggested in [16]. Assume that the QR decomposition of $\mathcal{F}_k$ is given by

$$\begin{aligned} \mathcal{F}_k &= \hat{Q}_k \times \hat{R}_k \\ &= \begin{bmatrix} Q_k & \bar{Q}_k \end{bmatrix} \times \begin{bmatrix} R_k \\ 0 \end{bmatrix} \\ &= Q_k \times R_k, \end{aligned} \tag{7}$$

where

$$\hat{Q}_k \in R^{n\times n},\ \hat{R}_k \in R^{n\times m_k},\ Q_k \in R^{n\times m_k},\ \bar{Q}_k \in R^{n\times(n-m_k)},\ \text{and } R_k \in R^{m_k\times m_k}.$$

Then the solution of the least-squares problem (5) is given by

$$\begin{aligned}\gamma^{(k)} &= \arg\min_{\gamma} \|f_k - \mathcal{F}_k\gamma\|_2 \\ &= \arg\min_{\gamma} \|f_k - \hat{Q}_k\hat{R}_k\gamma\|_2 \\ &= \arg\min_{\gamma} \|\hat{Q}_k^T f_k - \hat{R}_k\gamma\|_2 \\ &= \arg\min_{\gamma} \left\| \begin{pmatrix} Q_k^T \\ \bar{Q}_k^T \end{pmatrix} f_k - \begin{pmatrix} R_k \\ 0 \end{pmatrix} \gamma \right\|_2 \\ &= \arg\min_{\gamma} \left\| \begin{pmatrix} Q_k^T f_k - R_k\gamma \\ \bar{Q}_k^T f_k \end{pmatrix} \right\|_2 \\ &= \arg\min_{\gamma} \|Q_k^T f_k - R_k\gamma\|_2.\end{aligned}$$

Specifically, the solution is obtained by solving an $m_k \times m_k$ triangular system $R_k\gamma = Q_k^T f_k$. This shows that only $Q_k$ and $R_k$ need to be computed. In other words, only the "thin" QR decomposition (7) is necessary.

Since $\mathcal{F}_k$ is obtained from $\mathcal{F}_{k-1}$ by appending a new column on the right and possibly dropping one column from the left, the QR decomposition of $\mathcal{F}_k$ can be efficiently obtained by updating that of $\mathcal{F}_{k-1}$. For details about this aspect, see [15].

## 3. Condition number monitoring and Anderson depth modification

In Anderson acceleration, if the allowed maximal Anderson depth $m$ is too small, then the retained iteration history may be not enough to sufficiently accelerate the convergence. However, if $m$ is too large, then the matrix $\mathcal{F}_k$ may become so ill-conditioned that the solution of the least-squares problem is inaccurate and the numerical stability of the overall algorithm is adversely affected. In our numerical experiments, we indeed found cases in which the condition number of $\mathcal{F}_k$ became too large if remedial steps were not taken.

In this paper, we use the strategy given in [16] to monitor the condition number of the matrix $\mathcal{F}_k$ and, if necessary, to modify the matrix to reduce the condition number, as follows: When the condition number of $\mathcal{F}_k$ is larger than a given tolerance, then the left-most columns of $\mathcal{F}_k$ are dropped one by one until the condition number is less than the given tolerance.

This is a reasonable filtering strategy since the left-most columns contain the oldest information about the iteration. In some exceptional cases, however, there is a risk that some useful columns may be deleted by using this strategy. For example, as one of the reviewers pointed out, assume that $\mathcal{F}_k$ contains five columns, that is, $\mathcal{F}_k = (\Delta f_1, \ldots, \Delta f_5)$. If the newest two columns $\Delta f_4$ and $\Delta f_5$ are almost linearly dependent, while $\Delta f_1$, $\Delta f_2$, $\Delta f_3$, and $\Delta f_4$ are mutually perpendicular, then removing the oldest column $\Delta f_1$ would not improve the condition number. Repeatedly removing columns from the left would finally leave $\Delta f_5$, while the discarded columns $\Delta f_1$, $\Delta f_2$, and $\Delta f_3$ probably contain useful information. This may lower the convergence speed of the Anderson-acceleration method. However, in our numerical experiments, this phenomenon never happened.

It should be pointed out that some other filtering strategies can also be used. For example, the authors in [9] considered three filtering techniques in the QN-ILS method: old QR filtering, new QR filtering and proper orthogonal decomposition (POD) filtering. For these three filtering techniques, the filtered vector may not be the left-most one. The key idea of old QR and new QR filtering methods is to filter vectors by the magnitude of the diagonal of the matrix $R$, with $R$ being the factor in QR factorization. POD filtering is based on the eigenvalues and eigenvectors of the autocorrelation matrix of the retained vectors. Among all of these filtering methods, it is not clear which strategy is the best. Further research on this would be helpful.

Note that the $l^2$-norm condition number of $\mathcal{F}_k$ is just that of $R_k$ in the QR decomposition of $\mathcal{F}_k$. Therefore, for the filtering strategy used in this paper, it is only necessary to monitor the condition number of $R_k$ and keep it less than the given tolerance. When the condition number of $R_k$ is larger than the given tolerance, then deleting the left-most column of $\mathcal{F}_k$ entails updating the factors $Q_k$ and $R_k$; see [15] for details. The overall process is outlined in Algorithm 4.

**Algorithm 4** Condition number management.

1: Given $tol > 0$ and the QR decomposition $\mathcal{F}_k = Q_k R_k$.
2: **while** $\text{cond}(R_k) > tol$ **do**
3: Delete the left-most column of $\mathcal{F}_k$ and update the QR decomposition $\mathcal{F}_k = Q_k R_k$.
4: **end while**

## 4. Physical constraints in the Anderson acceleration algorithm

In some applications, the iteration process needs to satisfy physical constraints. For example, the temperature in the three-temperature energy equations should be non-negative. However, from the Anderson-acceleration iteration $x_{k+1} =$

$\sum_{i=0}^{m_k} \alpha_i^{(k)} G(x_{k-m_k+i})$, it is evident that even though each $G(x_{k-m_k+i})$ satisfies this physical constraint, it can not be guaranteed that $x_{k+1}$ satisfies this constraint because some $\alpha_i^{(k)}$ obtained from the least-squares problem may be negative.

In fact, we observed that some components of temperature sometimes became negative when Anderson acceleration was used to solve the three-temperature energy equations. This halted the simulation because the nonlinear residual can not be evaluated and the preconditioning matrix can not be constructed with negative temperature; see the expressions for the diffusion coefficients and energy-exchanging coefficients in Section 5.

By employing the idea of modification for the Jacobian-free Newton–Krylov method in [3], a similar modification can be made for Anderson acceleration. Assume that the physical constraint set is $\mathcal{D}$, a convex subset of $\mathbb{R}^n$. Assume further that $x_k$ and $x_{k+1}$ are two consecutive AA iterates, and that $x_k \in \mathcal{D}$. If $x_{k+1}$ violates the physical constraint, then a new next iterate can be defined as

$$\hat{x}_{k+1} = \beta_k x_{k+1} + (1-\beta_k)x_k, \quad \beta_k \in [0,1), \tag{8}$$

where $\beta_k$ is given by

$$\beta_k = 0.9 \times \sup\{\beta : \beta \in [0,1) \text{ and } \beta x_{k+1} + (1-\beta)x_k \in \mathcal{D}\}.$$

Since the constraint set $\mathcal{D}$ is convex, it is easy to see that there exists a $\beta_k$ satisfying the above condition and that $\hat{x}_{k+1}$ satisfies the physical constraint. In an AA iteration, if $x_{k+1}$ violates the physical constraint, then $x_{k+1}$ will be replaced by $\hat{x}_{k+1}$.

After incorporating the modifications for Anderson acceleration in Sections 3–4, we arrive at Algorithm 5 below. Note that lines 14–16 in the algorithm describe the check for the physical constraint. This algorithm will be used to solve the three-temperature energy equations at each time step.

**Algorithm 5** Improved Anderson acceleration.

1: Given $x_0$, $m \geq 0$ and $tol > 0$.
2: Set $x_1 = G(x_0)$.
3: **for** $k = 1, 2, \ldots$ **do**
4: Set $m_k = \min\{m, k\}$.
5: Compute $G(x_k)$ and let $f_k = G(x_k) - x_k$.
6: Update the QR decomposition $\mathcal{F}_k = Q_k R_k$ with $\mathcal{F}_k$ defined by (4).
7: Update $\mathcal{G}_k$ defined by (6).
8: **while** $\text{cond}(R_k) > tol$ **do**
9: Delete the left-most column of $\mathcal{F}_k$ and update the QR decomposition $\mathcal{F}_k = Q_k R_k$.
10: Set $m_k \leftarrow m_k - 1$.
11: **end while**
12: Solve the triangular system $R_k \gamma = Q_k^T f_k$.
13: Set $x_{k+1} = G(x_k) - \mathcal{G}_k \gamma^{(k)}$.
14: **if** $x_{k+1}$ violates the physical constraint **then**
15: Replace $x_{k+1}$ with $\hat{x}_{k+1}$ defined by (8).
16: **end if**
17: **end for**

## 5. The three-temperature energy equations and Picard iteration

The three-temperature energy equations [2] are defined as

$$\begin{cases} C_{ve}\frac{\partial T_e}{\partial t} - \frac{1}{\rho}\nabla\cdot(K_e\nabla T_e) &= \omega_{ei}(T_i - T_e) + \omega_{er}(T_r - T_e) \\ C_{vi}\frac{\partial T_i}{\partial t} - \frac{1}{\rho}\nabla\cdot(K_i\nabla T_i) &= \omega_{ei}(T_e - T_i) \\ C_{vr}\frac{\partial T_r}{\partial t} - \frac{1}{\rho}\nabla\cdot(K_r\nabla T_r) &= \omega_{er}(T_e - T_r) \end{cases} \tag{9}$$

where $T_e$, $T_i$ and $T_r$ are the electron, ion and radiation temperatures, respectively. The related physical coefficients in the equations are:

- $C_{v\alpha}$ $(\alpha = e, i, r)$ are isochore specific-heat coefficients;
- $\rho$ is the density of the materials;
- $K_\alpha = K_\alpha(\rho, T_\alpha)(\alpha = e, i, r)$ are heat-conduction (diffusion) coefficients;
- $\omega_{ei}$ and $\omega_{er}$ are the energy-exchanging coefficients between electron and ion, and between electron and photon, respectively.

Specifically, the isochore specific-heat coefficients, heat-conduction coefficients and energy-exchanging coefficients are defined as

$$C_{v\alpha}=\begin{cases} c_e, & \alpha=e \\ c_i, & \alpha=i \\ c_r T_r^3, & \alpha=r \end{cases} \quad K_\alpha=\begin{cases} A_e T_e^{5/2}, & \alpha=e \\ A_i T_i^{5/2}, & \alpha=i \\ A_r T_r^{3+\beta}, & \alpha=r \end{cases} \quad \omega_{e\alpha}=\begin{cases} A_{ei}\rho T_e^{-2/3}, & \alpha=i \\ A_{er}\rho T_e^{-1/2}, & \alpha=r \end{cases} \tag{10}$$

where $c_\alpha$ are constants, $A_\alpha$, $A_{e\alpha}$, $\beta$, and $\rho$ are material dependent. These parameters are continuous in the interior of the material. For specific expressions for the parameters, see [2]. Usually, the boundary and initial conditions are as follows.

- Boundary conditions:
  (a) on rigid walls, $K_\alpha \nabla T_\alpha \cdot \mathrm{n}=0$, $\alpha=e,i,r$, where n is the outer normal vector of the boundary;
  (b) on free surfaces, $K_\alpha \nabla T_\alpha \cdot \mathrm{n}=0$, $\alpha=e,i$; $T_r=T_r(t,x,y)|_{(x,y)\in\partial\Omega_{xy}}$.
- Concatenation conditions: $T_\alpha$ and $K_\alpha \nabla T_\alpha \cdot \mathrm{n}$ are continuous over the material interfaces, $\alpha=e,i,r$, n is the outer normal vector of the interface.
- Initial conditions:

$$T_\alpha(0,x,y)=T_\alpha^0(x,y), \quad \alpha=e,i,r.$$

The electron, ion and radiation energies are defined, respectively, as

$$E_e=c_e T_e, \quad E_i=c_i T_i, \quad E_r=\frac{1}{4}c_r T_r^4.$$

Note that

$$\frac{\partial E_e}{\partial T_e}=C_{ve}, \quad \frac{\partial E_i}{\partial T_i}=C_{vi}, \quad \frac{\partial E_r}{\partial T_r}=C_{vr}.$$

Therefore,

$$C_{v\alpha}\frac{\partial T_\alpha}{\partial t}=\frac{\partial E_\alpha}{\partial T_\alpha}\frac{\partial T_\alpha}{\partial t}=\frac{\partial E_\alpha}{\partial t}, \quad \alpha=e,i,r.$$

Consequently, the three-temperature energy equations (9) can also be written as

$$\begin{cases} \frac{\partial E_e}{\partial t}-\frac{1}{\rho}\nabla\cdot(K_e\nabla T_e) &= \omega_{ei}(T_i-T_e)+\omega_{er}(T_r-T_e) \\ \frac{\partial E_i}{\partial t}-\frac{1}{\rho}\nabla\cdot(K_i\nabla T_i) &= \omega_{ei}(T_e-T_i) \\ \frac{\partial E_r}{\partial t}-\frac{1}{\rho}\nabla\cdot(K_r\nabla T_r) &= \omega_{er}(T_e-T_r) \end{cases} \tag{11}$$

By applying the fully implicit backward Euler method to equations (11), one obtains

$$\begin{cases} \frac{E_e^{n+1}-E_e^n}{\Delta t_n}-\frac{1}{\rho}\nabla\cdot(K_e\nabla T_e)^{n+1} &= \omega_{ei}^{n+1}(T_i-T_e)^{n+1}+\omega_{er}^{n+1}(T_r-T_e)^{n+1} \\ \frac{E_i^{n+1}-E_i^n}{\Delta t_n}-\frac{1}{\rho}\nabla\cdot(K_i\nabla T_i)^{n+1} &= \omega_{ei}^{n+1}(T_e-T_i)^{n+1} \\ \frac{E_r^{n+1}-E_r^n}{\Delta t_n}-\frac{1}{\rho}\nabla\cdot(K_r\nabla T_r)^{n+1} &= \omega_{er}^{n+1}(T_e-T_r)^{n+1} \end{cases}$$

where the superscripts $n$ and $n+1$ represent two consecutive time levels. For simplicity, omitting the superscript $n+1$, the equations are given by

$$\begin{cases} \frac{E_e-E_e^n}{\Delta t_n}-\frac{1}{\rho}\nabla\cdot(K_e\nabla T_e) &= \omega_{ei}(T_i-T_e)+\omega_{er}(T_r-T_e) \\ \frac{E_i-E_i^n}{\Delta t_n}-\frac{1}{\rho}\nabla\cdot(K_i\nabla T_i) &= \omega_{ei}(T_e-T_i) \\ \frac{E_r-E_r^n}{\Delta t_n}-\frac{1}{\rho}\nabla\cdot(K_r\nabla T_r) &= \omega_{er}(T_e-T_r) \end{cases}$$

or

$$\begin{cases} F_e(\mathbf{T})\equiv\frac{E_e-E_e^n}{\Delta t_n}-\frac{1}{\rho}\nabla\cdot(K_e\nabla T_e)-\omega_{ei}(T_i-T_e)-\omega_{er}(T_r-T_e)=0 \\ F_i(\mathbf{T})\equiv\frac{E_i-E_i^n}{\Delta t_n}-\frac{1}{\rho}\nabla\cdot(K_i\nabla T_i)-\omega_{ei}(T_e-T_i)=0 \\ F_r(\mathbf{T})\equiv\frac{E_r-E_r^n}{\Delta t_n}-\frac{1}{\rho}\nabla\cdot(K_r\nabla T_r)-\omega_{er}(T_e-T_r)=0 \end{cases} \tag{12}$$

where

$$\mathbf{T} = (T_e, T_i, T_r)^T.$$

Let

$$\mathbf{F} = (F_e, F_i, F_r)^T.$$

The nonlinear equations that should be solved are then

$$\mathbf{F}(\mathbf{T}) = 0.$$

The Picard method can be used to solve the three-temperature energy equations. The specific method is given by Algorithm 6. Note that in this algorithm, the temperature at time step $n$ is used as the initial iterate for solving the equations at time step $n+1$.

**Algorithm 6** Picard method for the three-temperature energy equations.

1: Given the initial iterate $\mathbf{T}^{(0)} = (T_e^{(0)}, T_i^{(0)}, T_r^{(0)})^T = (T_e^n, T_i^n, T_r^n)^T$.
2: **for** $k = 0, 1, \ldots$ **do**
3: Solve the linear equations

$$\begin{cases} \frac{C_{ve}T_e - C_{ve}T_e^n}{\Delta t_n} - \frac{1}{\rho}\nabla\cdot(K_e^{(k)}\nabla T_e) - \omega_{ei}^{(k)}(T_i - T_e) - \omega_{er}^{(k)}(T_r - T_e) = 0 \\ \frac{C_{vi}T_i - C_{vi}T_i^n}{\Delta t_n} - \frac{1}{\rho}\nabla\cdot(K_i^{(k)}\nabla T_i) - \omega_{ei}^{(k)}(T_e - T_i) = 0 \\ \frac{C_{vr}^{(k)}T_r - C_{vr}^n T_r^n}{4\Delta t_n} - \frac{1}{\rho}\nabla\cdot(K_r^{(k)}\nabla T_r) - \omega_{er}^{(k)}(T_e - T_r) = 0 \end{cases}$$

and let the solution be the next iterate $\mathbf{T}^{(k+1)} = (T_e^{(k+1)}, T_i^{(k+1)}, T_r^{(k+1)})^T$.
4: **end for**

Note that the linear operator for the Picard method (Picard linearization) is given by

$$A_{\text{Picard}}^{(k)} = \mathcal{C}^{(k)} + \mathcal{D}^{(k)} + \mathcal{W}^{(k)},$$

where

$$\mathcal{C}^{(k)} = \frac{1}{\Delta t_n}\begin{pmatrix} C_{ve}^{(k)} & & \\ & C_{vi}^{(k)} & \\ & & \frac{1}{4}C_{vr}^{(k)} \end{pmatrix},$$

$$\mathcal{D}^{(k)} = -\begin{pmatrix} \frac{1}{\rho}\nabla\cdot K_e^{(k)}\nabla & & \\ & \frac{1}{\rho}\nabla\cdot K_i^{(k)}\nabla & \\ & & \frac{1}{\rho}\nabla\cdot K_r^{(k)}\nabla \end{pmatrix},$$

and

$$\mathcal{W}^{(k)} = \begin{pmatrix} \omega_{ei}^{(k)} + \omega_{er}^{(k)} & -\omega_{ei}^{(k)} & -\omega_{er}^{(k)} \\ -\omega_{ei}^{(k)} & \omega_{ei}^{(k)} & \\ -\omega_{er}^{(k)} & & \omega_{er}^{(k)} \end{pmatrix}.$$

Note that (12) can be expressed as

$$\mathbf{F}(\mathbf{T}) = A_{\text{Picard}}(\mathbf{T})\mathbf{T} - b^n = 0,$$

where

$$b^n = \left(\frac{C_{ve}T_e^n}{\Delta t_n}, \frac{C_{vi}T_i^n}{\Delta t_n}, \frac{C_{vr}^n}{4\Delta t_n}\right)^T.$$

Picard iteration can be expressed compactly as

$$\begin{aligned} \mathbf{T}^{(k+1)} &= \left(A_{\text{Picard}}^{(k)}\right)^{-1} b^n \\ &= \left(A_{\text{Picard}}^{(k)}\right)^{-1}\left(b^n - A_{\text{Picard}}^{(k)}\mathbf{T}^{(k)} + A_{\text{Picard}}^{(k)}\mathbf{T}^{(k)}\right) \\ &= \mathbf{T}^{(k)} - \left(A_{\text{Picard}}^{(k)}\right)^{-1}\mathbf{F}(\mathbf{T}^{(k)}). \end{aligned}$$

It is easy to see that the difference between Picard iteration and Newton iteration is only the matrix (operator) expression used to determine the step $\mathbf{T}^{(k+1)} - \mathbf{T}^{(k)}$.

In Section 6, $A^{(k)}_{\text{Picard}}$ will be used as the preconditioner for both the Newton–Krylov and Picard methods.

Note that square-root computations for temperature are used when a nonlinear residual is evaluated or a preconditioning matrix is constructed (see the expressions in (10) for diffusion coefficients and energy-exchanging coefficients). Therefore, in the iteration process, if some components of the temperature in an intermediate iteration are negative, then the iteration will be halted. In particular, if the procedure for maintaining the physical constraint is disabled in Anderson acceleration, then the method will halt and the simulation can not be finished. Physical-constraint checking plays a critical role when Anderson acceleration is used to solve the three-temperature energy equations, where the physical-constraint domain is defined as

$$\mathcal{D} = R^N_+ \equiv \left\{ x = (x_1, \ldots, x_N)^T \in R^N : x_i > 0, i = 1, \ldots, N \right\},$$

and $N$ is the number of unknowns of the discretization system.

## 6. Numerical results

In this section, numerical results for the Picard-GMRES method (PG) and the Anderson-accelerated Picard-GMRES method are given. For the purpose of comparison, results for a Jacobian-free Newton-GMRES method (JFNG) are also given.

Both two-dimensional (2D) and three-dimensional (3D) cases are considered. For the 2D case, the computing domain is $[0, 10] \times [0, 10]$ with the upper half domain filled with CH material and the lower half domain filled with $SO_2$ material. For the 3D case, the computing domain is $[0, 10] \times [0, 10] \times [0, 10]$. Similar to the 2D case, the upper half domain is filled with CH and the lower half domain is filled with $SO_2$. For both 2D and 3D cases, the upper boundary is the free surface, and the rest are rigid walls. The initial temperature is set to $3 \times 10^{-4}$, and the initial time-step size is $10^{-3}$ for all cases. For more information about the three-temperature energy equations, including specific parameters in the equations, see [2].

For spatial discretizations, the standard five-point finite-difference scheme for the 2D case and the seven-point finite-difference scheme for the 3D case are used. In the numerical results, the grid scales for the 2D case include $32 \times 32$, $64 \times 64$, $128 \times 128$, $256 \times 256$, $512 \times 512$ and $1024 \times 1024$; and the grid scales for the 3D case include $16 \times 16 \times 16$, $32 \times 32 \times 32$ and $64 \times 64 \times 64$.

### 6.1. Control of the time-step size

In the simulations, the time-step size is determined adaptively by the variation of energy. The concrete algorithm is given in Algorithm 7.

**Algorithm 7** Control of the time-step size.

1: Let $\Delta t_{n-1}$ be known. Given energy-variation thresholds $\beta$ and $\gamma$, such that $0 < \beta < \gamma < 1$; time-step shortening and enlarging factors $\theta_0$ and $\theta_1$, such that $0 < \theta_0 < 1 < \theta_1$; and minimal and maximal time-step sizes $\Delta t^{\text{low}}$ and $\Delta t^{\text{up}}$, such that $\Delta t^{\text{low}} < \Delta t^{\text{up}}$.
2: If $\Delta E^{n-1}_{\text{relative}} < \beta$, then $\Delta t_n = \theta_1 \Delta t_{n-1}$.
3: If $\Delta E^{n-1}_{\text{relative}} > \gamma$, then $\Delta t_n = \theta_0 \Delta t_{n-1}$.
4: If $\beta \le \Delta E^{n-1}_{\text{relative}} \le \gamma$, then $\Delta t_n = \Delta t_{n-1}$.
5: Let $\Delta t_n = \max\{\min\{\Delta t_n, \Delta t^{\text{up}}\}, \Delta t^{\text{low}}\}$.

In Algorithm 7, $\Delta E^{n-1}_{\text{relative}}$ is defined by

$$\Delta E^{n-1}_{\text{relative}} = \max_i \{|E^n_i - E^{n-1}_i| / E^{n-1}_i\},$$

which is the maximal relative energy variation on all cells. Here $i$ is the index for cells. In the numerical results, $\beta = 0.01$, $\gamma = 0.2$, $\theta_0 = 0.8$, $\theta_1 = 1.2$, $\Delta t^{\text{low}} = 10^{-9}$, and $\Delta t^{\text{up}} = 10^{-1}$.

For each of the considered grid scales, the same time-step size histories are determined by Algorithm 7 for all compared methods. It should be pointed out that, by using three different methods, the same solution is obtained at each time step. Therefore, exactly the same time-step sizes are obtained at each time step for the different methods. To show specific time-step sizes in the simulation, the time-step size curves for $32 \times 32 \times 32$ and $64 \times 64 \times 64$ are given in Figs. 1 and 2, respectively.

From Fig. 1, one can see that at the first several time steps, the time-step size decreases quickly from $1.0 \times 10^{-3}$ to $1.25 \times 10^{-4}$, and then it is stable with this value until time step 550. Later, the time-step size increases gradually to $3.92 \times 10^{-4}$, and this value is retained from time step 940 to the end of simulation.

Similar observations can be made for $64 \times 64 \times 64$ in Fig. 2. At early time steps, the time-step size decreases quickly from $1.0 \times 10^{-3}$ to $1.25 \times 10^{-4}$, and then it remains stable with this value until time step 337. Later, the time-step size increases gradually to $2.21 \times 10^{-4}$, and this value is retained from time step 445 until the end of simulation.

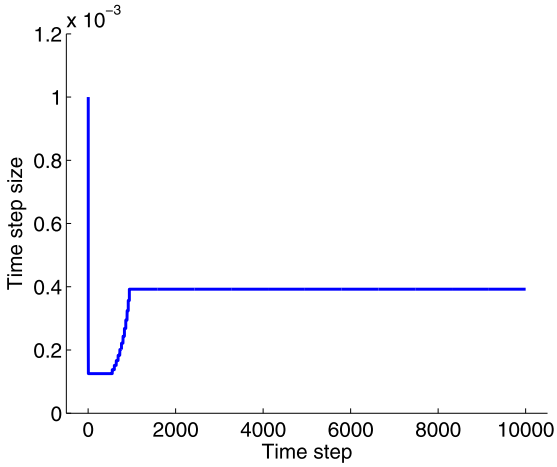


**Fig. 1.** Time-step size curve for $32 \times 32 \times 32$.

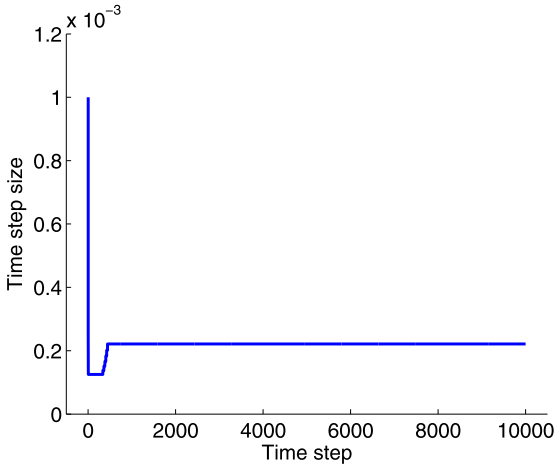


**Fig. 2.** Time-step size curve for $64 \times 64 \times 64$.

### 6.2. Compared methods

For comparison, the performance of the following three methods will be evaluated.

- JFNG;
- PG;
- Anderson-accelerated Picard-GMRES method with maximal Anderson depth $m$ (PG-AA($m$)).

All tests for these methods were carried out using KINSOL 2.8.2 [4], with some modifications for PG-AA($m$) to include condition-number monitoring and physical-constraint checking in Anderson acceleration.

For all methods, the convergence criterion for the nonlinear iteration was

$$\|F(x_k)\|_\infty \leq 10^{-12},$$

and the allowed maximal nonlinear iteration number was 100.

The maximal Krylov dimension for GMRES was 60, and the allowed number of restarts was 2. The convergence criterion for GMRES was $10^{-4}$. The matrix $A^{(k)}_{\text{Picard}}$ was used as the preconditioning matrix for all methods. The BoomerAMG algebraic-multigrid solver in HYPRE [7] was used as the solver for the preconditioning system.

For iterative methods, the cost can be represented by the numbers of iterations, function evaluations, preconditioning setups, and preconditioning solves. For brevity, the following notation will be used.

- `NNI`: number of nonlinear iterations;
- `NLI`: number of linear iterations;

**Table 1**
Cost comparison for three methods.

| | JFNG | PG | PG-AA($m$) |
|---|---|---|---|
| NFE | NNI+NLI+1 | NNI+1 | NNI+1 |
| NPreSet | NNI | NNI | NNI |
| NPreSol | NNI+NLI | NNI+NLI | NNI+NLI |

**Table 2**
Average iteration numbers and CPU times for the 2D case (10000 time steps).

| | Method | 32×32 | 64×64 | 128×128 | 256×256 | 512×512 | 1024×1024 |
|---|---|---|---|---|---|---|---|
| NNI | JFNG | 2.91 | 2.90 | 2.91 | 2.93 | 2.90 | 2.88 |
| | PG | 22.27 | 21.90 | 21.22 | 21.05 | 21.43 | 24.32 |
| | PG-AA(10) | 8.52 | 8.35 | 8.20 | 8.09 | 8.43 | 9.11 |
| NLI | JFNG | 13.74 | 13.62 | 13.14 | 12.65 | 12.46 | 12.90 |
| | PG | 22.27 | 21.90 | 21.23 | 21.05 | 21.43 | 24.32 |
| | PG-AA(10) | 8.52 | 8.35 | 8.20 | 8.09 | 8.43 | 9.11 |
| NFE | JFNG | 17.65 | 17.52 | 17.05 | 16.58 | 16.36 | 16.78 |
| | PG | 23.27 | 22.9 | 22.22 | 22.05 | 22.43 | 25.32 |
| | PG-AA(10) | 9.52 | 9.35 | 9.2 | 9.09 | 9.43 | 10.11 |
| NPreSol | JFNG | 16.65 | 16.52 | 16.05 | 15.58 | 15.36 | 15.78 |
| | PG | 44.54 | 43.8 | 42.45 | 42.1 | 42.86 | 48.64 |
| | PG-AA(10) | 17.04 | 16.7 | 16.4 | 16.18 | 16.86 | 18.22 |
| CPU | JFNG | 0.34 | 1.36 | 3.51 | 6.85 | 13.27 | 28.77 |
| | PG | 0.89 | 3.59 | 9.32 | 18.97 | 38.95 | 92.28 |
| | PG-AA(10) | 0.34 | 1.37 | 3.61 | 7.31 | 15.38 | 34.78 |

- NFE: number of function evaluations;
- NPreSet: number of preconditioning setups;
- NPreSol: number of preconditioning solves;
- CPU: CPU time for solving the three-temperature energy equations.

For the three methods compared here, the numbers of function evaluations, preconditioning setups, and preconditioning solves were determined by the numbers of nonlinear iterations and linear iterations, as shown in Table 1. Here, it is assumed that no globalization strategy (such as line search or trust-region techniques) is used in the JFNG method.

In the numerical results, condition-number monitoring was used in Anderson acceleration with $tol = 10^{10}$. The maximal depth in Anderson acceleration was $m = 10$. This value is relatively large; however, since a condition-number monitoring strategy was used in Anderson acceleration, it was not necessary to worry about the condition number becoming problematically large. The physical constraint used with Anderson acceleration was that the temperature was required to be positive.

### 6.3. Performance of the methods

Table 2 shows the average numbers of nonlinear iterations, linear iterations, function evaluations, preconditioning solves and CPU times for the 2D case. From Table 2, one can see that, in these tests, the accelerated Picard method was always better than the Picard method without acceleration, and that acceleration improved the performance of the Picard method by at least a factor of two with regard to the numbers of nonlinear iterations, linear iterations, function evaluations and preconditioning solves. One can also see that, on all grids, JFNG required the smallest numbers of nonlinear iterations of the three methods; however, PG-AA(10) required the smallest numbers of linear iterations and functions evaluations. In terms of preconditioning solves, JFNG performed slightly better than PG-AA(10). The CPU times for JFNG were slightly less than those of PG-AA(10), while the CPU times for PG were much greater than those of PG-AA(10).

Compared with the PG method, the CPU-time speedups of JFNG and PG-AA(10) are given in Table 3 for the 2D case. From this table, one can see that the least speedup of PG-AA(10) is 2.53, and in most cases, the speedup is greater than 2.60. The speedups of JFNG are greater than those of PG-AA(10) because the numbers of nonlinear iterations and preconditioning solves for JFNG are less than those of PG-AA(10).

Table 4 shows the average numbers of nonlinear iterations, linear iterations, function evaluations, preconditioning solves and CPU times for the 3D case. From this table, similar conclusions can be drawn to those in the 2D case. In all cases, Anderson acceleration improved the efficiency of the Picard method by at least a factor of two. JFNG was the best in terms of numbers of nonlinear iterations and preconditioning solves, and PG-AA(10) was the best in terms of numbers of linear iterations and function evaluations. The CPU times of PG-AA(10) were slightly greater than those of JFNG. In all cases, the performance of the Picard method without acceleration was the worst.

**Table 3**
Speedups of JFNG and PG-AA(10) compared to PG for the 2D case.

| | 32×32 | 64×64 | 128×128 | 256×256 | 512×512 | 1024×1024 |
|---|---|---|---|---|---|---|
| JFNG | 2.62 | 2.64 | 2.66 | 2.77 | 2.94 | 3.21 |
| PG-AA(10) | 2.61 | 2.62 | 2.58 | 2.60 | 2.53 | 2.65 |

**Table 4**
Average iteration numbers and CPU times for the 3D case (10000 time steps).

| | Method | 16×16×16 | 32×32×32 | 64×64×64 |
|---|---|---|---|---|
| NNI | JFNG | 2.94 | 2.91 | 2.96 |
| | PG | 20.75 | 22.27 | 25.24 |
| | PG-AA(10) | 9.40 | 9.82 | 10.53 |
| NLI | JFNG | 13.61 | 13.98 | 15.45 |
| | PG | 20.75 | 22.27 | 25.24 |
| | PG-AA(10) | 9.40 | 9.82 | 10.53 |
| NFE | JFNG | 17.56 | 17.89 | 19.41 |
| | PG | 21.74 | 23.27 | 26.24 |
| | PG-AA(10) | 10.40 | 10.82 | 11.53 |
| NPreSol | JFNG | 16.56 | 16.89 | 18.41 |
| | PG | 41.50 | 44.54 | 50.48 |
| | PG-AA(10) | 18.80 | 19.64 | 21.07 |
| CPU | JFNG | 0.25 | 2.21 | 13.06 |
| | PG | 0.66 | 6.23 | 38.20 |
| | PG-AA(10) | 0.30 | 2.75 | 15.96 |

**Table 5**
Speed up of JFNG and PG-AA(10) for 3D case.

| | 16×16×16 | 32×32×32 | 64×64×64 |
|---|---|---|---|
| JFNG | 2.64 | 2.82 | 2.92 |
| PG-AA(10) | 2.20 | 2.27 | 2.39 |

**Table 6**
Percentage of Anderson-acceleration cost in PG-AA(10) for 2D case.

| | 32×32 | 64×64 | 128×128 | 256×256 | 512×512 | 1024×1024 |
|---|---|---|---|---|---|---|
| $100 \times \frac{\mathrm{CPU_{AA}}}{\mathrm{CPU_{total}}}$ | 0.98 | 0.91 | 0.82 | 0.52 | 0.44 | 0.38 |

**Table 7**
Percentage of Anderson-acceleration cost in PG-AA(10) for 3D case.

| | 16×16×16 | 32×32×32 | 64×64×64 |
|---|---|---|---|
| $100 \times \frac{\mathrm{CPU_{AA}}}{\mathrm{CPU_{total}}}$ | 0.97 | 0.87 | 0.79 |

Similar to the 2D case, the speedups of JFNK and PG-AA(10) compared to PG are given in Table 5 for the 3D case. From this table, one can see that the speedups of PG-AA(10) are greater than 2.2 in all cases. Because the numbers of nonlinear iterations and preconditioning solves for JFNG are less than those of PG-AA(10), the speedups of JFNG are slightly greater than those of PG-AA(10).

Compared with the PG method, the extra cost of PG-AA($m$) is the solution of the least-squares problem (5) plus a modest number of arithmetic operations at each iteration. In our tests, this extra cost was very low. Tables 6 and 7 show the Anderson-acceleration module costs as a percentage of the whole solution cost for the 2D and 3D cases. In these two tables, $\mathrm{CPU_{AA}}$ represents the cost for the Anderson acceleration module, and $\mathrm{CPU_{total}}$ represents the whole solution cost. One can see that for all cases, the cost of Anderson acceleration is less than 1%.

To show the influence of the parameter $m$ for PG-AA($m$) method, Table 8 and Table 9 are given for the 2D and 3D cases, respectively. Note that when $m = 0$, PG-AA(0) is just the PG method. From Table 8 one can see that the numbers of nonlinear iterations, linear iterations, function evaluations and preconditioning solves decrease dramatically when $m$ increases from 0 to 2. For example, the average number of nonlinear iterations is 21.05 for the PG method, while the average number of nonlinear iterations is 9.58 for PG-AA(2) method. As the parameter $m$ increases, the performance of PG-AA($m$) continues to improve until $m = 8$. When $m > 8$, there is no further improvement. This suggests that $m = 8$ is a near-optimal value.

**Table 8**
Average iterations of PG-AA($m$) with different $m$ (256×256, 10000 time steps).

| | $m$ | | | | | | |
|---|---|---|---|---|---|---|---|
| | 0 | 2 | 4 | 6 | 8 | 10 | 12 |
| NNI | 21.05 | 9.58 | 8.43 | 8.13 | 8.08 | 8.09 | 8.08 |
| NLI | 21.05 | 9.58 | 8.43 | 8.13 | 8.08 | 8.09 | 8.08 |
| NFE | 22.05 | 10.58 | 9.43 | 9.13 | 9.08 | 9.09 | 9.08 |
| NPreSol | 42.10 | 19.16 | 16.86 | 16.26 | 16.16 | 16.18 | 16.16 |

**Table 9**
Average iterations of PG-AA($m$) with different $m$ (64×64×64, 100 time steps (2000∼2100)).

| | $m$ | | | | | | |
|---|---|---|---|---|---|---|---|
| | 0 | 2 | 4 | 6 | 8 | 10 | 12 |
| NNI | 34.10 | 10.64 | 9.15 | 8.92 | 8.92 | 8.92 | 8.92 |
| NLI | 34.10 | 10.64 | 9.15 | 8.92 | 8.92 | 8.92 | 8.92 |
| NFE | 35.10 | 11.64 | 10.15 | 9.92 | 9.92 | 9.92 | 9.92 |
| NPreSol | 68.20 | 21.28 | 18.30 | 17.84 | 17.84 | 17.84 | 17.84 |

**Table 10**
Average and maximal Anderson depths for the 2D case.

| | 32×32 | 64×64 | 128×128 | 256×256 | 512×512 | 1024×1024 |
|---|---|---|---|---|---|---|
| $\bar{m}_k$ | 4.2 | 4.1 | 4.1 | 4.0 | 4.1 | 4.2 |
| $\hat{m}_k$ | 10 | 10 | 10 | 10 | 10 | 10 |

**Table 11**
Average and maximal Anderson depths for the 3D case.

| | 16×16×16 | 32×32×32 | 64×64×64 |
|---|---|---|---|
| $\bar{m}_k$ | 4.8 | 4.9 | 5.2 |
| $\hat{m}_k$ | 10 | 10 | 10 |

Table 9 shows the performance of PG-AA($m$) on 100 time steps for the 3D case. One can draw similar conclusions as in the 2D case. Notable differences in this case are that the improvement is even greater when $m$ increases from 0 to 2 and that the optimal value of $m$ is about 6.

Recall that the number of vectors retained in Anderson acceleration is the Anderson depth, represented by $m_k$ in Algorithm 5. This $m_k$ is determined by the allowed maximal depth $m$, the iteration index, and the condition-number monitoring strategy. Tables 10 and 11 show the average and maximal Anderson depth in the 2D and 3D simulations, respectively. In the tables, $\bar{m}_k$ is the average Anderson depth over all iterations and all time steps, and $\hat{m}_k$ is the maximal Anderson depth for all iterations and all time steps.

From Table 10, one can see that the average Anderson depth is about 4.0 to 4.2, which is less than the allowed maximal number of vectors 10. This shows that the condition-number monitoring had a strong influence on Anderson depth during the simulations. At the same time, one can see that the maximal Anderson depth is 10. This shows that at some time steps, the condition number of the matrix $\mathcal{F}_k$ was relatively good, and no vectors were filtered by the condition-number monitoring strategy in some iterations. Similar conclusions can be drawn from Table 11 for the 3D case; the main difference is that the average Anderson depth varies over slightly larger values (4.8 to 5.2) for the different grid scales.

Figs. 3–6 plot the numbers of nonlinear iterations, linear iterations, function evaluations, and preconditioning solves, respectively, versus time steps for the $32 \times 32 \times 32$ grid. Fig. 3 shows that the nonlinear iteration numbers for JFNG are very stable, with 3 nonlinear iterations at most time steps. The numbers of nonlinear iterations of PG-AA(10) are higher than those of JFNG, but much lower than those of PG. From Figs. 4–5, one can see that at most time steps, the linear iterations and function evaluations of PG-AA(10) are the lowest. Fig. 6 shows that the numbers of preconditioning solves for JFNG and PG-AA(10) are very similar, with those for JFNG a little less than those for PG-AA(10). One can also see that at early time steps (about from time step 1 to 1300), the performance of PG-AA(10) is almost the same as that of PG. This is because the time-step size is relatively small and the temperature variation is low in this period. About from time step 1300, PG-AA(10) performs much better than the PG method. One can obtain similar conclusions from Figs. 7–10 for the $64 \times 64 \times 64$ grid.

Fig. 11 shows the specific numbers of nonlinear iterations of PG-AA($m$) at each time step with different $m$. From this figure one can see that there is a dramatic drop from the curve for $m = 0$ (the PG method) to the curves for $m \geq 2$. This shows that the Anderson acceleration method is very effective, even for small $m$. When $m$ increases from 2 to 10, there is some further decrease, but not as great. Our numerical experiments showed that $m = 10$ is large enough in the 2D case; the number of iterations can not be decreased further by additionally increasing $m$.

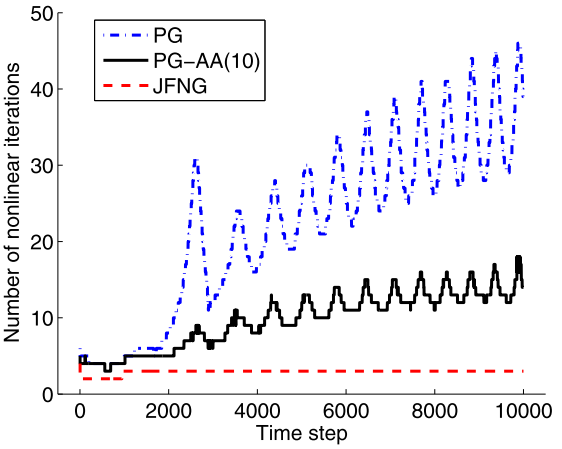


**Fig. 3.** Number of nonlinear iterations, $32 \times 32 \times 32$ grid.

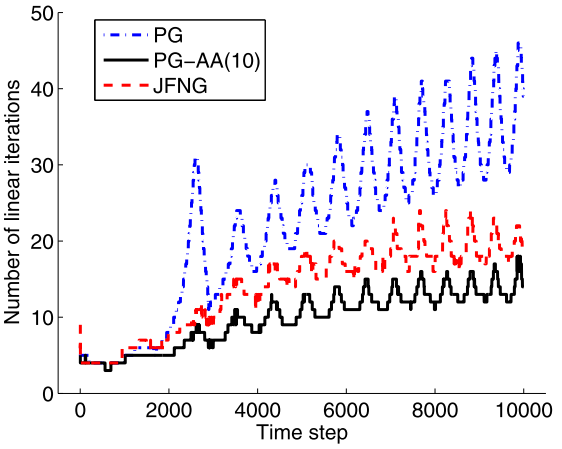


**Fig. 4.** Number of linear iterations, $32 \times 32 \times 32$ grid.

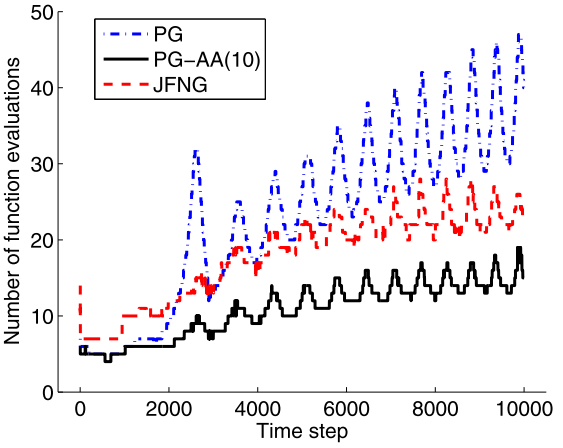


**Fig. 5.** Number of function evaluations, $32 \times 32 \times 32$ grid.

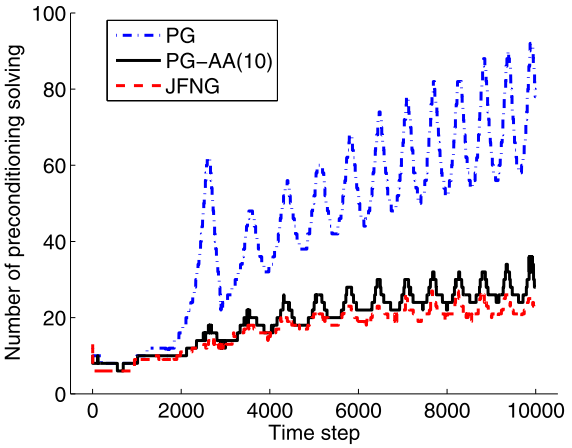


**Fig. 6.** Number of preconditioning solves, $32 \times 32 \times 32$ grid.

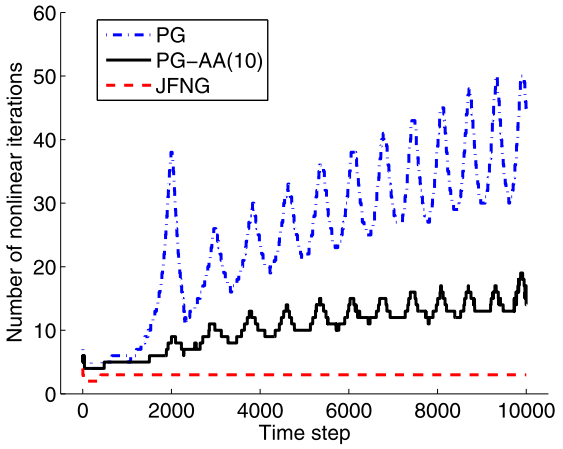


**Fig. 7.** Number of nonlinear iterations, $64 \times 64 \times 64$ grid.

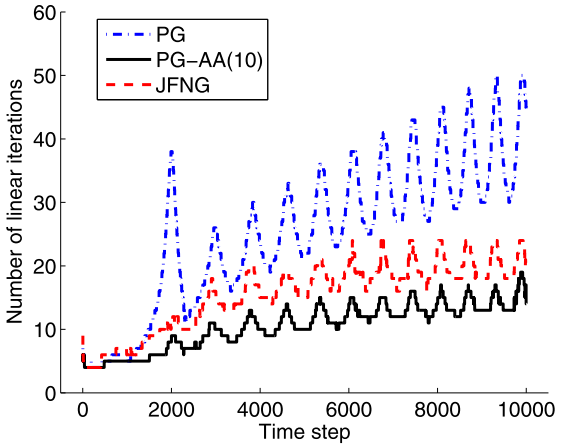


**Fig. 8.** Number of linear iterations, $64 \times 64 \times 64$ grid.

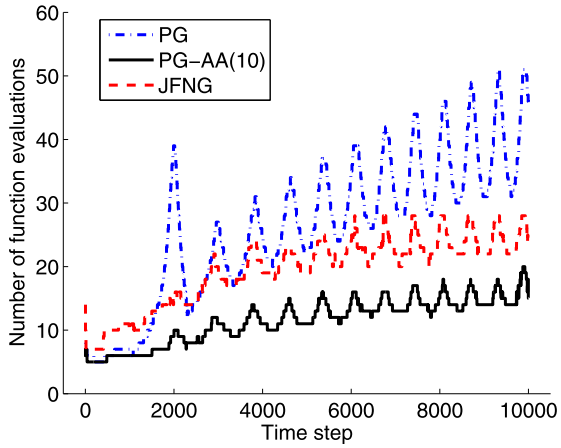


**Fig. 9.** Number of function evaluations, $64 \times 64 \times 64$ grid.

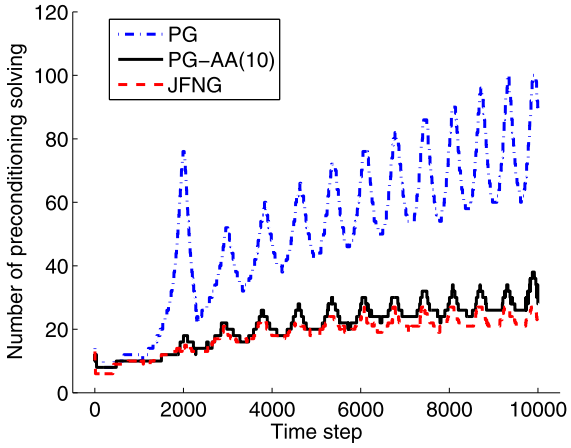


**Fig. 10.** Number of preconditioning solves, $64 \times 64 \times 64$ grid.

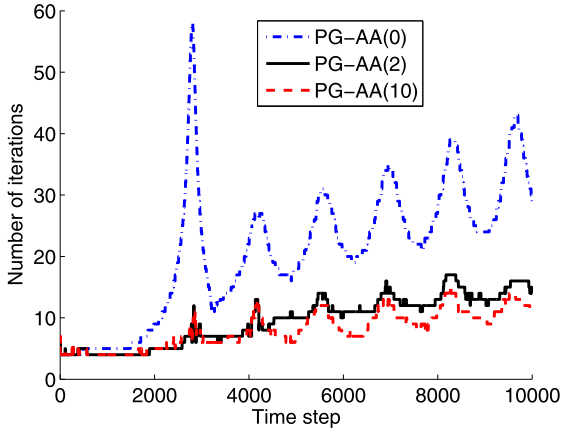


**Fig. 11.** Number of nonlinear iterations with different $m$, $256 \times 256$ grid.

Fig. 12 shows the specific numbers of nonlinear iterations at 100 time steps for the $64 \times 64 \times 64$ grid. Similar conclusions can be drawn as in the 2D case. This figure shows that the number of nonlinear iterations is decreased by at least two-thirds when Anderson acceleration is used compared to the unaccelerated Picard method. When $m$ increases from 2 to 10, the numbers of iterations can be further decreased. As in the 2D case, $m = 10$ is large enough; the number of iterations can not be further decreased by additionally increasing $m$.

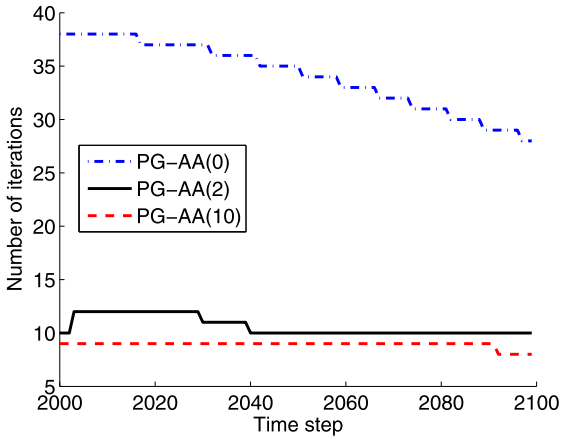


**Fig. 12.** Number of nonlinear iterations with different $m$, $64 \times 64 \times 64$ grid.

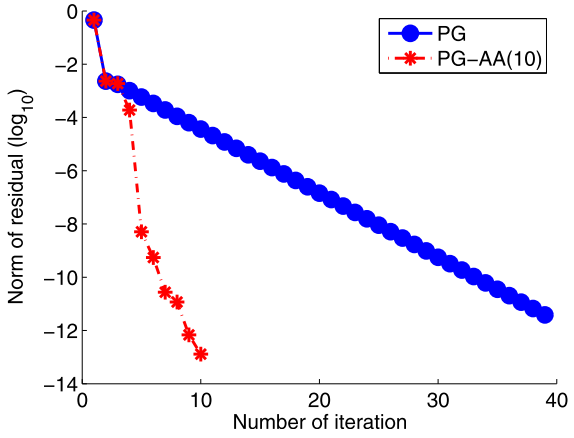


**Fig. 13.** Nonlinear-residual norm curve at time step 2000, $64 \times 64 \times 64$ grid.

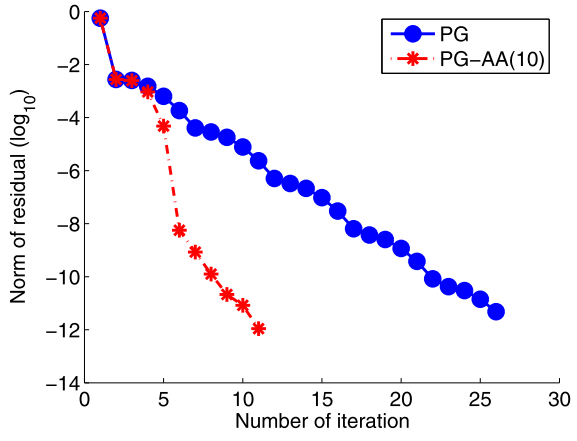


**Fig. 14.** Nonlinear-residual norm curve at time step 3000, $64 \times 64 \times 64$ grid.

Figs. 13 and 14 show the nonlinear residual norm curves for the PG and PG-AA(10) methods on two time steps. From these two figures one can see that after four or five iterations, Anderson acceleration begins to accelerate the convergence of the PG-AA(10) iterates, while the PG iterates converge at a significantly slower linear rate.

The simulations can be carried out to completion when the physical-constraint preserving strategy is used in Anderson acceleration. However, without this strategy, a simulation may halt because negative temperatures occur at some iteration. This did not happen very often in our experiments; for example, it occurred a total of 75 times for the $64 \times 64 \times 64$-grid

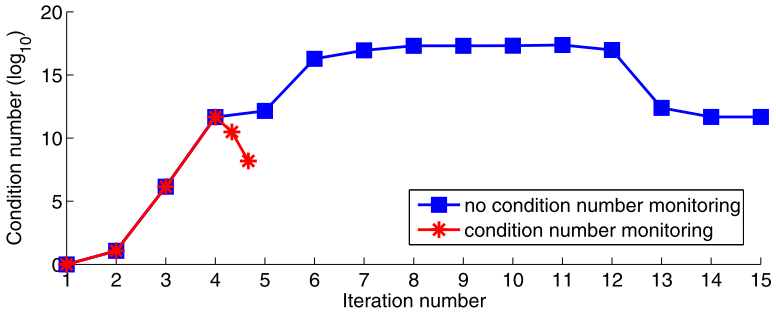


**Fig. 15.** Condition-number curves at the first time step, $64 \times 64 \times 64$ grid.

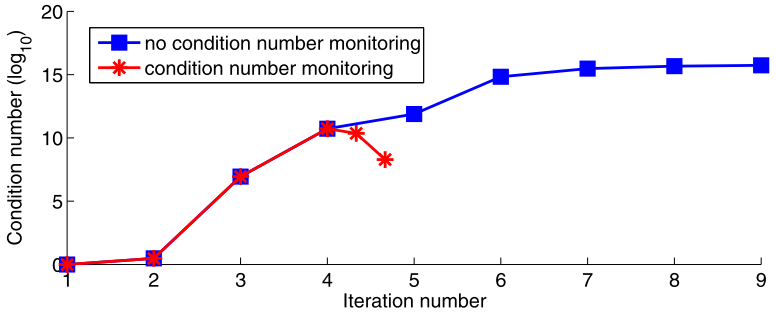


**Fig. 16.** Condition-number curves at the second time step, $64 \times 64 \times 64$ grid.

simulation in 10000 time steps. Most occurrences were in early time steps. The reason is that the temperatures are low at earlier simulation times, which increases the likelihood that unconstrained steps will result in negative temperatures. The higher temperatures at later simulation times make this outcome less likely.

In our Anderson-acceleration implementation, the condition number is monitored to ensure stability and robustness. In all numerical results, the tolerance for the condition-number monitoring is $10^{10}$. Figs. 15 and 16 show the specific effects of condition-number monitoring. Fig. 15 shows that the condition number becomes more than $10^{15}$ if there is no condition-number monitoring, but fortunately the iteration still converges after 15 iterations. With condition-number monitoring, when a new vector is added to $\mathcal{F}_4$ at iteration step 4, the condition number is initially greater than $10^{10}$ ($\mathcal{F}_k$ is defined by (4) in Section 2). However, after deleting two vectors from $\mathcal{F}_4$ on the left, the condition number is less than $10^{10}$, and the iteration succeeds after 4 iterations. Similar observations can be made from Fig. 16.

## 7. Conclusion

Anderson acceleration is a method for improving the convergence of fixed-point iterations. Picard methods constitute a class of nonlinear fixed-point iterations that are widely used in complex numerical simulations. In this paper, an Anderson-accelerated Picard method is used to solve a kind of radiation-diffusion equation — the three-temperature energy equations.

To improve the robustness of the Anderson-accelerated Picard method, two strategies have been used in Anderson acceleration. One is the modification of the iteration so that the positive-temperature physical constraint remains satisfied. In our experiments, there were some cases in which this was necessary for the simulations to be carried out to completion. Another strategy is a procedure for monitoring and, if necessary, reducing the condition numbers of the least-squares problems in Anderson acceleration. Condition-number monitoring is very important in numerical computation. Our results demonstrate that, by using this strategy, both the numerical stability and the convergence rate can be improved.

In our experiments, Anderson acceleration improved the time to convergence of Picard iteration by at least a factor of two. For the maximum Anderson depth (the maximum number $m$ of residuals saved in the Anderson-acceleration algorithm), our numerical results show that the optimal value is less than 10.

## Acknowledgements

The authors gratefully acknowledge the referees and Luis Chacon for many valuable suggestions and comments, which helped to improve the original paper.

## References


[1] D.G. Anderson, Iterative procedures for non-linear integral equations, J. ACM 12 (547) (1965) 547–560.
[2] H.B. An, Z.Y. Mo, X.W. Xu, X. Liu, On choosing a nonlinear initial iterate for solving the 2-D 3-T heat conduction equations, J. Comput. Phys. 228 (5) (2009) 3268–3287.
[3] H.B. An, Z.Y. Mo, Iteration process of JFNK method and physical constraint, Chin. J. Comput. Phys. 29 (5) (2012) 130–136.
[4] A.M. Collier, A.C. Hindmarsh, R. Serban, C.S. Woodward, User Documentation for KINSOL v2.8.2, Technical Report UCRL-SM-208116, Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, Livermore, 2015.
[5] H. Fang, Y. Saad, Two classes of multisecant methods for nonlinear acceleration, Numer. Linear Algebra Appl. 16 (3) (2009) 197–221.
[6] T. Feng, H.B. An, X.J. Yu, Q. Li, R.P. Zhang, On linearization and preconditioning for radiation diffusion coupled to material thermal conduction equations, J. Comput. Phys. 236 (2013) 28–40.
[7] HYPRE home page: http://computation.llnl.gov/projects/hypre-scalable-linear-solvers-multigrid-methods.
[8] V. Ganine, N.J. Hills, B.L. Lapworth, Nonlinear acceleration of coupled fluid–structure transient thermal problems by Anderson mixing, Int. J. Numer. Methods Fluids 71 (2013) 939–959.
[9] R. Haelterman, A.E.J. Bogaers, K. Scheufele, B. Uekermann, M. Mehl, Improving the performance of the partitioned QN-ILS procedure for fluid–structure interaction problems: filtering, Comput. Struct. 171 (2016) 9–17.
[10] J. Karátson, Characterizing mesh independent quadratic convergence of Newton's method for a class of elliptic problems, SIAM J. Math. Anal. 44 (2012) 1279–1303.
[11] D.A. Knoll, D.A. Keyes, Jacobian-free Newton–Krylov method: a survey of approaches and applications, J. Comput. Phys. 193 (2004) 357–397.
[12] K. Lipnikov, D. Svyatskiy, Y. Vassilevski, Anderson acceleration for nonlinear finite volume scheme for advection–diffusion problems, SIAM J. Sci. Comput. 35 (2) (2013) 1120–1136.
[13] P. Lott, H.F. Walker, C.S. Woodward, U. Yang, An accelerated Picard method for nonlinear systems related to variably saturated flow, Adv. Water Resour. 38 (2012) 92–101.
[14] F.A. Potra, H. Engler, A characterization of the behavior of the Anderson acceleration on linear problems, Linear Algebra Appl. 428 (2013) 1002–1011.
[15] H.F. Walker, Anderson Acceleration: Algorithms and Implementations, Research Report, MS-6-15-50, Worcester Polytechnic Institute Mathematical Sciences Department, 2011.
[16] H.F. Walker, P. Ni, Anderson acceleration for fixed-point iterations, SIAM J. Numer. Anal. 49 (4) (2011) 1715–1735.
[17] M. Weiser, A. Schiela, P. Deuflhard, Asymptotic mesh independence of Newton's method revisited, SIAM J. Numer. Anal. 42 (5) (2005) 1830–1845.
[18] J. Willert, W.T. Taitano, D.A. Knoll, Leveraging Anderson acceleration for improved convergence of iterative solutions to transport systems, J. Comput. Phys. 273 (2014) 278–286.
[19] A. Toth, C.T. Kelley, Convergence analysis for Anderson acceleration, SIAM J. Numer. Anal. 53 (2) (2015) 805–819.