ANDERSON ACCELERATION WITH TRUNCATED GRAM-SCHMIDT

ZIYUAN TANG*, TIANSHI XU†, HUAN HE‡, YOUSEF SAAD*, AND YUANZHE XI†

Abstract. Anderson Acceleration (AA) is a popular algorithm designed to enhance the convergence of fixed-point iterations. In this paper, we introduce a variant of AA based on a Truncated Gram-Schmidt process (AATGS) which has a few advantages over the classical AA. In particular, an attractive feature of AATGS is that its iterates obey a three-term recurrence in the situation when it is applied to solving symmetric linear problems and this can lead to a considerable reduction of memory and computational costs. We analyze the convergence of AATGS in both full-depth and limited-depth scenarios and establish its equivalence to the classical AA in the linear case. We also report on the effectiveness of AATGS through a set of numerical experiments, ranging from solving nonlinear partial differential equations to tackling nonlinear optimization problems. In particular, the performance of the method is compared with that of the classical AA algorithms.

Key words. Anderson Acceleration, Gram-Schmidt process, short-term recurrence, Krylov subspace, nonlinear equations

AMS subject classifications. 65F10, 68W25, 65F08, 90C53

1. Introduction and Motivation.

This paper considers numerical schemes for solving the nonlinear system of equations

$$f(x) = 0, \tag{1.1}$$

where f is a continuously differentiable mapping from $\mathbb{R}^n$ to $\mathbb{R}^n$. Problem (1.1) can be reformulated as an equivalent fixed point problem

$$x = g(x), \tag{1.2}$$

for a suitable mapping g from $\mathbb{R}^n$ to $\mathbb{R}^n$. For example, we can set $g(x) = x + \beta f(x)$ for some nonzero scalar β. When the fixed point iteration, i.e., the sequence generated by $x_{j+1} = g(x_j)$, converges to the fixed point of (1.2) then this limit is a solution to the problem (1.1). However, the fixed-point iteration can be slow or it can diverge and therefore acceleration methods are often invoked to improve or establish convergence. Anderson Acceleration (AA) [1], which is equivalent to the DIIS method - or Pulay Mixing [20, 21] in quantum chemistry, is a popular acceleration technique that has been developed for this purpose. AA has found extensive applications in scientific computing and, more recently, in machine learning [2, 12, 15, 17, 19, 27, 28, 29].

If the j-th iterate is denoted by x_j and if we set $f_j \equiv f(x_j)$, then AA starts with an initial x_0 and defines $x_1 = g(x_0) = x_0 + \beta_0 f_0$, where $\beta_0 > 0$ is a parameter. Let $m_j = \min\{m, j\}$ and $j_m = \max\{0, j - m\}$ and assume that the most recent m_j iterates are saved at each step. At step j, we define the matrices of differences:

$$\mathcal{X}_j = [\Delta x_{j_m} \ \ldots \ \Delta x_{j-1}] \in \mathbb{R}^{n\times m_j}, \qquad \mathcal{F}_j = [\Delta f_{j_m} \ \ldots \ \Delta f_{j-1}] \in \mathbb{R}^{n\times m_j}, \tag{1.3}$$

where $\Delta x_i := x_{i+1} - x_i$ and $\Delta f_i := f_{i+1} - f_i$. Then AA defines the next iterate as follows:

$$x_{j+1} = x_j + \beta_j f_j - (\mathcal{X}_j + \beta_j \mathcal{F}_j)\theta_j \quad \text{where:} \tag{1.4}$$

$$\theta_j = \text{argmin}_{\theta\in\mathbb{R}^{m_j}} \|f_j - \mathcal{F}_j\theta\|_2. \tag{1.5}$$

Note that x_{j+1} can be expressed with the help of intermediate vectors:

$$\bar{x}_j = x_j - \mathcal{X}_j\theta_j, \quad \bar{f}_j = f_j - \mathcal{F}_j\theta_j, \quad x_{j+1} = \bar{x}_j + \beta_j \bar{f}_j. \tag{1.6}$$

AA is closely related to Broyden's multi-secant type methods. This connection was initially revealed in [7] and further discussed in [23]. Essentially, AA acts as a 'block version' of Broyden's second update method where an update of rank m_j is applied at each step, instead of the traditional rank 1 update.

*Department of Computer Science and Engineering, University of Minnesota, Minneapolis (tang0389@umn.edu, saad@umn.edu). The research of Tang and Saad is supported by the NSF award DMS 2208456.

†Department of Mathematics, Emory University, Atlanta, GA 30322 (tianshi.xu@emory.edu, yxi26@emory.edu). The research of Xi is supported by NSF award DMS 2208412.

‡Work done in Department of Computer Science, Emory University, Atlanta, GA 30322 (hehuannb@gmail.com)

Note that the AA scheme just discussed retains m past iterates where m is often called the *window size*, or sometimes *depth*, of the AA procedure in the literature. In subsequent sections, we will refer to this scheme as AA(m). Retaining and using all past iterates is equivalent to setting $m = \infty$ in the procedure and so it will be often denoted by AA(∞). This is often referred to as the *full-depth* Anderson Acceleration, while when $m < \infty$, AA(m) is known as a *limited-depth* or *windowed* variant of AA.

The study of the convergence of AA has been an active research area in recent years. It was shown in [27] that the full-depth AA(∞) applied to $g(x) = Gx + b$ is "essentially equivalent" to the GMRES method [25] applied to $(I - G)x = b$ when $I - G$ is nonsingular and the linear residuals are strictly decreasing in the norm. Under these assumptions, the iterate x_j returned by AA(∞) at step j is equal to $Gx_{j-1}^{GMRES} + b$ where x_{j-1}^{GMRES} is the iterate returned by GMRES(j-1) with the same initial guess x_0. The first rigorous convergence analysis of AA(m) for contractive fixed point mappings was conducted in [26] where the authors prove the q-linear convergence of the residuals for linear problems and the local r-linear convergence for nonlinear problems when the coefficients in the linear combination remain bounded. In addition, they also prove the q-linear convergence of the residuals for AA(1) separately. These convergence results show that the convergence rate of AA(m) is not worse than that of the underlying fixed point iteration. The explicit improvement of AA(m) over the underlying fixed point iteration at each step is studied in [6] where the authors show that AA(m) can improve the convergence rate to first order by a factor $\tau_j \leq 1$ that is equal to the ratio of $\|f_j - \mathcal{F}_j\theta_j\|_2$ to $\|f_j\|_2$. They also point out that although AA(m) can increase the radius of convergence, AA(m) typically fails to improve the convergence in quadratically converging fixed point iterations. The asymptotic convergence analysis of AA(m) is conducted in [4], where the authors show that the r-linear convergence factor strongly depends on the initial condition for the r-linearly convergent AA(m) sequence and the coefficients θ_j do not converge but oscillate as the sequence converges. The one-step convergence analysis of inexact AA(m) with a potentially non-contractive mapping is conducted in [32]. The convergence rate of AA(m) on superlinearly and sublinearly converging fixed point iterations has recently been studied in [22].

While recent studies have concentrated on studying the convergence of AA as well as on improving its convergence properties, relatively little attention has been devoted to reducing its memory usage. This paper addresses this gap by developing a variant of AA that can exploit the symmetry (or near symmetry) of the Jacobian of the function f. In doing so, the iterates will obey short-term update expressions akin to those of the Conjugate Gradient or Conjugate Residual methods. The end result is a substantial reduction in memory and computational costs when solving large-scale nonlinear equations or optimization problems. Short-term recurrences often lead to numerical instabilities and so the proposed algorithm may encounter numerical issues in some situations. To circumvent this problem we introduce a restarting strategy that aims at monitoring the growth of floating point errors.

The remaining sections are organized as follows. AATGS is introduced in Section 2 which also presents a convergence analysis. The restarting strategy is discussed in Section 3 and numerical experiments are provided in Section 4. Finally, a few concluding remarks are drawn in Section 5.

2. Anderson Acceleration with Truncated Gram-Schmidt (AATGS). The variant of Anderson Acceleration to be introduced in this section relies on building an orthonormal basis. The idea of using an orthonormal basis in AA is not completely new. For example, it is common to use the QR decomposition to determine the minimizer θ_j in (1.5) by orthonormalizing the columns of $\mathcal{F}_j$. This will lead to a process that is less prone to numerical errors than an approach based on normal equations. However, in the limited-depth case, this approach requires the successive QR factorization of an evolving set of vectors in which the oldest vector is removed at each step once the buffer that stores $\mathcal{F}_j$ is full - which occurs when $j + 1 \geq m$. The proper way to implement this effectively in order to obtain the QR factorization of each new set of vectors, is through a simple QR-downdating scheme, see, e.g., [27]. In this paper, we will adopt a different viewpoint that proceeds similarly to the truncated GCR algorithm [5] to produce a 'locally' orthonormal basis, i.e., a basis in which the last vector is orthogonal to the most recent $m_j - 1$ vectors instead of all previous vectors. We will show that this variant has some advantages over classical AA.

2.1. AATGS(m). The basic idea of AATGS(m) is to exploit an orthonormal basis for the Δf_i's to simplify and improve the solution of the least-squares problem (1.5). Specifically, at each step j we orthonormalize Δf_{j-1} against previous $m_j - 1$ Δf_i's to get an orthonormal basis $Q_j = [q_{j_m+1}, q_{j_m+2}, \ldots, q_j]$ of $\mathcal{F}_j$. The same transformation is applied to the Δx_i's to get a new basis

$U_j = [u_{j_m+1}, u_{j_m+2}, \ldots, u_j]$ of $\mathcal{X}_j$ so that the two sets of vectors Q_j and U_j are paired by the same relation. The full algorithm is sketched as Algorithm 2.1.

Algorithm 2.1 AATGS(m)

1: **Input**: Function $f(x)$, initial guess x_0, window size m
2: Set $f_0 \equiv f(x_0), \quad x_1 = x_0 + \beta_0 f_0, \quad f_1 \equiv f(x_1)$
3: **for** $j = 1, 2, \cdots$, until convergence **do**
4: $\quad u := \Delta x = x_j - x_{j-1}$
5: $\quad q := \Delta f = f_j - f_{j-1}$
6: $\quad$ **for** $i = j_m + 1, \ldots, j-1$ **do**
7: $\quad\quad s_{ij} := (q, q_i)$
8: $\quad\quad u := u - s_{ij} u_i$
9: $\quad\quad q := q - s_{ij} q_i$
10: $\quad$ **end for**
11: $\quad s_{jj} = \|q\|_2$
12: $\quad q_j := q/s_{jj}, \quad u_j := u/s_{jj}$
13: $\quad$ Set $Q_j = [q_{j_m+1}, \ldots, q_j], \quad U_j = [u_{j_m+1}, \ldots, u_j]$
14: $\quad$ Compute $\theta_j = Q_j^\top f_j$
15: $\quad x_{j+1} = (x_j - U_j \theta_j) + \beta_j (f_j - Q_j \theta_j)$
16: $\quad f_{j+1} = f(x_{j+1})$
17: **end for**

Define the upper triangular matrix $S_j = \{s_{ik}\}_{i=j_m+1:j, k=j_m+1:j}$ resulting from the orthogonalization process, where the nonzero entries s_{ij} are defined in Lines 7 and 11 of the algorithm. The block in Lines 4–12 essentially performs a Gram-Schmidt QR factorization of the matrix $\mathcal{F}_j$, and enacts identical operations on the set $\mathcal{X}_j = [\Delta x_{j_m}, \Delta x_{j_m+1}, \cdots, \Delta x_{j-1}]$. A result of the algorithm is that

$$\mathcal{F}_j = Q_j S_j; \qquad \mathcal{X}_j = U_j S_j. \tag{2.1}$$

After step j of Algorithm 2.1 is applied, we would have built the orthonormal basis $Q_j = [q_{j_m+1}, \cdots, q_j]$ along with a paired system $U_j = [u_{j_m+1}, \cdots, u_j]$. Note that Q_j has orthonormal columns but not U_j. The vector θ_j computed in Line 14 by a simple matrix-vector product, is the least-squares solution of $\min_\theta \|f_j - Q_j \theta\|_2$ which is not necessarily the same as the solution of the least-squares problem (1.5) - since the span of $\mathcal{F}_j$ differs from the span of Q_j when $j > m$. Finally, Line 15 computes the next iterate x_{j+1} using the two paired bases Q_j and U_j and θ_j. Note that as for classical Anderson, we can also define $\bar{x}_j$ and $\bar{f}_j$ and rewrite x_{j+1} in the following form:

$$\bar{x}_j = x_j - U_j \theta_j, \quad \bar{f}_j = f_j - Q_j \theta_j, \quad x_{j+1} = \bar{x}_j + \beta_j \bar{f}_j. \tag{2.2}$$

We mentioned the case $j > m$ in the above discussion. It is easy to see that in the case where $j \le m$, the subspaces spanned by $\mathcal{F}_j$ and Q_j are identical and in this situation the iterates x_{j+1} resulting from AA and AATGS will be the same. In particular, when $m = \infty$ this will always be the case, i.e., the full-depth AATGS(∞) and AA(∞) return the same iterate x_{j+1} in exact arithmetic at each iteration and thus are mathematically equivalent.

In the next section, we will study the properties of AATGS(∞) and exhibit a particularly interesting short-term recurrence of the algorithm when it is applied to symmetric linear systems.

2.2. Theoretical analysis of AATGS(∞). Consider a linear problem where $f(x) = b - Ax$ and A is invertible. Note that in this case we have

$$\mathcal{F}_j = -A\mathcal{X}_j. \tag{2.3}$$

In the next lemma, we show that the matrix U_j returned by Algorithm 2.1 forms a basis of the Krylov subspace $\mathcal{K}_j(A, f_0)$ and that under mild conditions, Q_j, U_j satisfy the same relation as $\mathcal{F}_j, \mathcal{X}_j$ in (2.3) for AATGS(∞).

LEMMA 2.1. *Assume A is invertible and $f(x) = b - Ax$. If Algorithm 2.1 applied for solving $f(x) = 0$ with $m = \infty$ does not break at step j, then the system U_j forms a basis of the Krylov subspace $\mathcal{K}_j(A, f_0)$. In addition, the orthonormal system Q_j built by Algorithm 2.1 satisfies $Q_j = -AU_j$.*

Proof. We first prove $Q_j = -AU_j$ by induction. When $j = 1$, we have $q_1 = (f_1 - f_0)/s_{11} = -Au_1$. Assume $Q_{j-1} = -AU_{j-1}$. Then we have

$$\begin{aligned} s_{jj}q_j &= (f_j - f_{j-1}) - \sum_{i=1}^{j-1} s_{ij}q_i = -A(x_j - x_{j-1}) - \sum_{i=1}^{j-1} s_{ij}(-Au_i) \\ &= -A[(x_j - x_{j-1}) - \sum_{i=1}^{j-1} s_{ij}u_i] \\ &= s_{jj}(-Au_j). \end{aligned}$$

Thus, since $s_{jj} \neq 0$ we get $q_j = -Au_j$ and therefore $Q_j = -AU_j$, completing the induction proof.

Next, we prove by induction that U_j forms a basis of $\mathcal{K}_j(A, f_0)$. It is more convenient to prove by induction the property that for each $i \leq j$, U_i forms a basis of $\mathcal{K}_i(A, f_0)$. The result is true for $j = 1$ since we have $u_1 = (x_1 - x_0)/s_{11} = \beta_0 f_0/s_{11}$. Now let us assume the property is true for $j - 1$, i.e., that for each $i = 1, 2, \cdots, j-1$, U_i is a basis of the Krylov subspace $\mathcal{K}_i(A, f_0)$. Then we have

$$\begin{aligned} s_{jj}u_j &= (x_j - x_{j-1}) - \sum_{i=1}^{j-1} s_{ij}u_i \\ &= -U_{j-1}\theta_{j-1} + \beta_{j-1}(f_{j-1} - Q_{j-1}\theta_{j-1}) - \sum_{i=1}^{j-1} s_{ij}u_i \\ &= -U_{j-1}\theta_{j-1} + \beta_{j-1}f_{j-1} - \beta_{j-1}Q_{j-1}\theta_{j-1} - \sum_{i=1}^{j-1} s_{ij}u_i \\ &= \beta_{j-1}f_{j-1} - U_{j-1}\theta_{j-1} + \beta_{j-1}AU_{j-1}\theta_{j-1} - \sum_{i=1}^{j-1} s_{ij}u_i. \end{aligned} \tag{2.4}$$

The induction hypothesis shows that $-U_{j-1}\theta_{j-1}+\beta_{j-1}AU_{j-1}\theta_{j-1}-\sum_{i=1}^{j-1} s_{ij}u_i \in \mathcal{K}_j(A, f_0)$. It remains to show that $f_{j-1} = b - Ax_{j-1} \in \mathcal{K}_j(A, f_0)$. For this, we expand $b - Ax_{j-1}$ as

$$b - Ax_{j-1} = b - Ax_{j-1} + Ax_{j-2} - Ax_{j-2} + \ldots - Ax_1 + Ax_0 - Ax_0 = \sum_{i=1}^{j-1} -A(x_i - x_{i-1}) + f_0.$$

From the relation (2.4) applied with j replaced by i, we see that $x_i - x_{i-1}$ is a linear combination of $u_1, u_2, \cdots, u_i$, i.e., it is a member $\mathcal{K}_i$ by the induction hypothesis. Therefore $-A(x_i - x_{i-1}) \in \mathcal{K}_{i+1}$ - but since $i \leq j - 1$ then $-A(x_i - x_{i-1}) \in \mathcal{K}_j$. The remaining term f_0 is clearly in $\mathcal{K}_j$. Because $U_j = -A^{-1}Q_j$ has full column rank and $u_i \in \mathcal{K}_j(A, f_0)$ for $i = 1, \ldots, j$, U_j forms a basis of $\mathcal{K}_j(A, f_0)$. This completes the induction proof. ∎

From (2.2), we see that in the linear case under consideration the vector $\bar{f}_j$ is the residual for $\bar{x}_j$:

$$\bar{f}_j = f_j - Q_j\theta_j = (b - Ax_j) - Q_j\theta_j = (b - Ax_j) + AU_j\theta_j = b - A(x_j - U_j\theta_j) = b - A\bar{x}_j. \tag{2.5}$$

The next theorem shows that $\bar{x}_j$ minimizes $\|b - Ax\|_2$ over the affine space $x_0 + \mathcal{K}_j(A, f_0)$.

THEOREM 2.2. *The vector $\bar{x}_j$ generated at the j-th step of AATGS(∞) minimizes the residual norm $\|b - Ax\|_2$ over all vectors x in the affine space $x_0 + \mathcal{K}_j(A, f_0)$. It also minimizes the same residual norm over the subspace $x_k + \mathcal{K}_j(A, f_0)$ for any k such that $0 \leq k \leq j$.*

Proof. Consider a vector of the form $x = x_j - \delta$ where $\delta = U_j y$ is an arbitrary member of $\mathcal{K}_j(A, f_0)$. We have

$$b - Ax = b - A(x_j - U_j y) = f_j + AU_j y = f_j - Q_j y. \tag{2.6}$$

The minimal norm $\|b - Ax\|$ is reached when $y = Q_j^\top f_j$ and the corresponding optimal x is $\bar{x}_j$. Therefore, $\bar{x}_j$ is the vector x of the affine space $x_j + \mathcal{K}_j(A, f_0)$ with the smallest residual norm. We now write x as:

$$\begin{aligned} x &= x_j - U_j y \\ &= x_0 + (x_1 - x_0) + (x_2 - x_1) + (x_3 - x_2) + \cdots (x_{i+1} - x_i) + \cdots (x_j - x_{j-1}) - U_j y \qquad (2.7) \\ &= x_0 + \Delta x_0 + \Delta x_1 + \cdots + \Delta x_{j-1} - U_j y. \qquad (2.8) \end{aligned}$$

We will exploit the relation obtained from the QR factorization of Algorithm 2.1, namely $\mathcal{X}_j = U_j S_j$ in (2.1): If e is the vector of all ones, then $\Delta x_0 + \Delta x_1 + \cdots + \Delta x_{j-1} = \mathcal{X}_j e = U_j S_j e$. Define $t_j \equiv S_j e$. Then, from (2.8) we obtain

$$x = x_j - \delta = x_0 - U_j[y - t_j]. \tag{2.9}$$

This means that the set of all vectors of the form $x_j - \delta$ is the same as the set of all vectors of the form $x_0 - \delta'$ where $\delta' \in \mathcal{K}_j(A, f_0)$. As a result, $\bar{x}_j$ also minimizes $b - Ax$ over all vectors in the affine space $x_0 + \mathcal{K}_j(A, f_0)$. The proof can be easily repeated for any k between 0 and j. The expansion (2.7 –2.8) becomes

$$x_j - U_j y = x_k + (x_{k+1} - x_k) + (x_{k+2} - x_{k+1}) + \cdots (x_{i+1} - x_i) + \cdots (x_j - x_{j-1}) - U_j y \tag{2.10}$$

$$= x_k + \Delta x_k + \Delta x_{k+1} + \cdots + \Delta x_{j-1} - U_j y. \tag{2.11}$$

The rest of the proof is similar and straightforward. ∎

Theorem 2.2 shows that $\bar{x}_j$ is the j-th iterate of the GMRES algorithm for solving $Ax = b$ with the initial guess x_0 and that $\bar{f}_j$ is the corresponding residual. The value of $\bar{x}_j$ is independent of the choice of β_i for $i \le j$. Now consider the residual f_{j+1} of AATGS(∞) at step $j+1$. From the relations $x_{j+1} = \bar{x}_j + \beta_j \bar{f}_j$ and (2.5) we get:

$$f_{j+1} = b - A[\bar{x}_j + \beta_j \bar{f}_j] = b - A\bar{x}_j - \beta_j A\bar{f}_j = \bar{f}_j - \beta_j A\bar{f}_j = (I - \beta_j A)\bar{f}_j. \tag{2.12}$$

This implies that the vector f_{j+1} is the residual for x_{j+1} obtained from $x_{j+1} = \bar{x}_j + \beta_j \bar{f}_j$ - which is a simple Richardson iteration starting from the iterate $\bar{x}_j$. Therefore, x_{j+1} in Line 15 of Algorithm 2.1 is nothing but a Richardson iteration step from this GMRES iterate. This is stated in the following proposition.

PROPOSITION 2.3. *The residual f_{j+1} of the iterate x_{j+1} generated at the j-th step of AATGS(∞) is equal to $(I - \beta_j A)\bar{f}_j$ where $\bar{f}_j = b - A\bar{x}_j$ minimizes the residual norm $\|b - Ax\|_2$ over all vectors x in the affine space $x_0 + \mathcal{K}_j(A, f_0)$. In other words, the $(j+1)$-st iterate of AATGS(∞) can be obtained by performing one step of a Richardson iteration applied to the j-th GMRES iterate.*

A similar result has also been proved for the standard AA by Walker and Ni [27] under slightly different assumptions.

2.3. Short-term recurrence in AATGS for linear symmetric problems. We now show that the orthogonalization process (Lines 6-10 of Algorithm 2.1) simplifies in the linear symmetric case under consideration. Indeed, we will see that S_j consists of only 3 non-zero diagonals in the upper triangular part when A is symmetric. This implies that we only need to save q_{j-2}, q_{j-1} and u_{j-2}, u_{j-1} in order to generate q_j and u_j in the full-depth AATGS(∞). Before we prove this result, we first examine the components of the vector $Q_j^\top f_j$ in Line 14 of Algorithm 2.1.

LEMMA 2.4. *When $f(x) = b - Ax$ where A is a real non-singular symmetric matrix then the entries of the vector $\theta_j = Q_j^T f_j$ in Algorithm 2.1 are all zeros except the last two.*

Proof. Let $i \le j - 1$. From (2.12), we have

$$(f_j, q_i) = (\bar{f}_{j-1} - \beta_{j-1} A\bar{f}_{j-1}, q_i) = (\bar{f}_{j-1}, q_i) - \beta_{j-1}(A\bar{f}_{j-1}, q_i).$$

The first term equals zero because $(f_{j-1} - Q_{j-1}\theta_{j-1}, q_i) = ((I - Q_{j-1}Q_{j-1}^T)f_{j-1}, q_i) = 0$. Consider the second term:

$$(A\bar{f}_{j-1}, q_i) = (\bar{f}_{j-1}, Aq_i).$$

Observe that since $u_i \in \mathcal{K}_i(A, f_0)$, then $q_i = -Au_i$ belongs to the Krylov subspace $\mathcal{K}_{i+1}(A, f_0)$ which is the same as Span$\{U_{i+1}\}$ according to Lemma 2.1. Thus, it can be written as $Au_i = U_{i+1}y$ for some y and hence, $Aq_i = -AU_{i+1}y = Q_{i+1}y$, i.e., Aq_i is in the span of $q_1, \cdots, q_{i+1}$. Therefore, recalling that $\bar{f}_{j-1} \perp$ Span$\{Q_{j-1}\}$, we have:

$$(\bar{f}_{j-1}, Aq_i) = 0 \quad \text{for} \quad i \le j - 2.$$

In the end, we obtain $(f_j, q_i) = 0$ for $i \le j - 2$. ∎

Lemma 2.4 indicates that the computation of x_{j+1} in Line 15 of Algorithm 2.1 only depends on the two most recent q_i's and u_i's. The next theorem will further show that q_j and u_j in Line 12 can be computed based on q_{j-2}, q_{j-1} and u_{j-2}, u_{j-1} instead of all previous q_i's and u_i's.

THEOREM 2.5. *When $f(x) = b - Ax$ where A is a real non-singular symmetric matrix, then the upper triangular matrix S_k is banded with bandwidth 3, i.e., we have $s_{ij} = 0$ for $i < j - 2$.*

Proof. It is notationally more convenient to consider column $j+1$ of S_k where $k > j$. Denote $\Delta f_j = f_{j+1} - f_j$, and $\Delta x_j = x_{j+1} - x_j$. Consider $s_{i,j+1} = (\Delta f_j, q_i)$ for $i \le j$ and note that $s_{i,j+1} = -(A\Delta x_j, q_i)$. We note that

$$\Delta x_j = x_{j+1} - x_j = \bar{x}_j + \beta_j \bar{f}_j - x_j = x_j - U_j\theta_j + \beta_j \bar{f}_j - x_j = -U_j\theta_j + \beta_j \bar{f}_j.$$

We write

$$\begin{aligned} A\Delta x_j &= -AU_j\theta_j + \beta_j A\bar{f}_j = Q_j\theta_j + \beta_j A\bar{f}_j \\ &= -(f_j - Q_j\theta_j) + f_j + \beta_j A\bar{f}_j \\ &= -\bar{f}_j + f_j + \beta_j A\bar{f}_j, \end{aligned}$$

and hence,

$$(A\Delta x_j, q_i) = -(\bar{f}_j, q_i) + (f_j, q_i) + \beta_j (A\bar{f}_j, q_i). \tag{2.13}$$

The first term on the right-hand side, $(\bar{f}_j, q_i)$ vanishes since $i \le j$. According to Lemma 2.4 the inner product (f_j, q_i) is zero for $i \le j - 2$. The last term $(A\bar{f}_j, q_i)$ is equal to zero when $i \le j - 1$ as shown in the proof of Lemma 2.4. This completes the proof as it shows that $s_{i,j+1} = 0$ for $i < j - 1$. ∎

Lemma 2.4 and Theorem 2.5 show that when AATGS(∞) is applied to solving linear symmetric problems, only the two most recent q_{j-2}, q_{j-1} and u_{j-2}, u_{j-1} are needed to compute the next iterate x_{j+1}, which significantly reduces both memory and orthogonalization costs. In other words, AATGS(3) is equivalent to AATGS(∞) in the linear symmetric case.

Staying with the linear case, the next theorem examines the convergence rate of AATGS(∞) when A is symmetric positive definite.

THEOREM 2.6. *Assume that A is symmetric positive definite and that a constant β is used in AATGS. Then we have the following error bound for the iterate x_{j+1}^{AATGS} obtained at the $(j+1)$-st step of AATGS(∞):*

$$\|x_{j+1}^{AATGS} - x_*\|_2 \le \frac{\kappa(A)\|I - \beta A\|_2}{T_j(1 + \frac{2}{\kappa(A)-1})} \|x_0^{AATGS} - x_*\|_2, \tag{2.14}$$

where T_j is the Chebyshev polynomial of first kind of degree j, x_* is the exact solution of the system, and $\kappa(A)$ is the 2-norm condition number of A.

Proof. Assume x_* is the fixed point of $g(x) = x + \beta(b - Ax)$. Based on Proposition 2.3, we have

$$\begin{aligned} \|x_{j+1}^{AATGS} - x_*\|_2 &= \|g(x_j^{GMRES}) - g(x_*)\| = \|(I - \beta A)(x_j^{GMRES} - x_*)\|_2 \\ &= \|(I - \beta A)A^{-1}A(x_j^{GMRES} - x_*)\|_2 \\ &= \|(I - \beta A)A^{-1}r_j^{GMRES}\|_2, \end{aligned}$$

where x_j^{GMRES} and r_j^{GMRES} denote the j-th iterate from GMRES and its associated residual, respectively.

Since $A \in \mathbb{R}^{n\times n}$ is symmetric, it admits the following eigendecomposition:

$$A = U\Lambda U^\top, \quad U^\top U = I, \quad \Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_n), \tag{2.15}$$

where $0 < \lambda_1 \le \lambda_2 \le \cdots \le \lambda_n$. It is known that the GMRES residual vector can be expressed as

$$r_j^{GMRES} = \rho(A)r_0^{GMRES} = U\rho(\Lambda)U^\top r_0^{GMRES}, \quad \rho \in \mathcal{P}_j, \tag{2.16}$$

where $\mathcal{P}_j$ is the affine space of polynomials p of degree j such that $p(0) = 1$ and

$$\|\rho(A)r_0^{GMRES}\|_2 = \min_{p\in\mathcal{P}_j} \|p(A)r_0^{GMRES}\|_2 \leq \min_{p\in\mathcal{P}_j} \max_i |p(\lambda_i)|\|r_0^{GMRES}\|_2 \tag{2.17}$$

$$\leq \min_{p\in\mathcal{P}_j} \max_{\lambda\in[\lambda_1,\lambda_n]} |p(\lambda)|\|r_0^{GMRES}\|_2 \tag{2.18}$$

$$\leq \frac{\|r_0^{GMRES}\|_2}{T_j(1+2\frac{\lambda_1}{\lambda_n-\lambda_1})}. \tag{2.19}$$

The last inequality follows from well-known results on the optimality properties of Chebyshev polynomials, see, e.g., [24]. Note that $1 + 2\lambda_1/(\lambda_n - \lambda_1) = 1 + 2/(\kappa(A) - 1)$. Next we write

$$\|r_0^{GMRES}\|_2 = \|A(x_0^{GMRES} - x_*)\|_2 \leq \|A\|_2 \, \|x_0^{GMRES} - x_*\|_2. \tag{2.20}$$

Thus, we obtain

$$\begin{aligned}
\|x_{j+1}^{AATGS} - x_*\|_2 = \|(I-\beta A)A^{-1}r_j^{GMRES}\|_2 &= \|(I-\beta A)A^{-1}\rho(A)r_0^{GMRES}\|_2 \\
&\leq \|(I-\beta A)A^{-1}\|_2\|\rho(A)r_0^{GMRES}\|_2 \\
&\leq \|(I-\beta A)\|_2\|A^{-1}\|_2\frac{\|r_0^{GMRES}\|_2}{T_j(1+2\frac{\lambda_1}{\lambda_n-\lambda_1})} \\
&\leq \|(I-\beta A)\|_2\kappa(A)\frac{\|x_0^{GMRES}-x_*\|_2}{T_j(1+\frac{2}{\kappa(A)-1})}.
\end{aligned}$$

This completes the proof. ∎

The convergence results can be generalized to the case where the eigenvalues of A are distributed in two intervals excluding the origin. This result is omitted.

Another case of interest is when A is skew-symmetric. In this situation, when the β_j's are constant, it can be seen that the AATGS algorithm yields $x_2 = x_1$ after the first iteration, and consequently, the process breaks at Line 12 due to s_{22} being equal to zero. To circumvent this problem, one could adjust β_j at each iteration. Alternatively, reformulating the problem $f(x)$ itself presents another viable strategy. An example demonstrating this approach is provided in Section 4.6 for solving minimax optimization problems. Note that there is no issue in the interesting case when A is of the form $A = I + S$ where S is skew-symmetric, for which it can be shown that we do have a simplification similar to that of the symmetric case.

2.4. Limited-depth AATGS. We now explore the limited-depth version of AATGS(m) for a fixed m. Recall the notation $j_m = \max\{0, j - m\}$. At step j, Algorithm 2.1 orthogonalizes the latest Δf vector against $q_{j_m+1}, ..., q_{j-1}$ to produce q_j. We set $U_j \equiv [u_{j_m+1}, u_{j_m+2}, \cdots, u_{j-1}, u_j]$ and $Q_j \equiv [q_{j_m+1}, q_{j_m+2}, \cdots, q_{j-1}, q_j]$ in Line 13. Note that U_j and Q_j have $\min\{j, m\}$ columns, which is the same number of columns as the block $\mathcal{F}_j$ in Anderson Acceleration. As it turns out, $\bar{x}_j = x_j - U_j\theta_j$ satisfies a similar result to that of Theorem 2.2.

PROPOSITION 2.7. *The intermediate iterate $\bar{x}_j = x_j - U_j\theta_j$ obtained at the j-th step of AATGS(m) minimizes $\|b - Ax\|_2$ over all vectors x of the form $x = x_j - \delta$ where $\delta \in$ Span$\{U_j\}$.*

Proof. We consider a generic vector $x = x_j - \delta$ where $\delta \in \text{span}\{U_j\}$ which we write as $\delta = U_j y$. Then Equation (2.6) in the proof of Theorem 2.2 still holds, i.e., we can write $r \equiv b - Ax = f_j - Q_j y$. It is known that the residual norm is minimal iff $r \perp \text{Span}\{Q_j\}$, i.e., iff: $f_j - Q_j y \perp \text{Span}\{Q_j\}$ which is precisely the condition imposed to get θ_j. This means that $\bar{x}_j$ minimizes $\|b - Ax\|_2$ over all vectors x of the form $x = x_j - \delta$ where $\delta \in \text{Span}\{U_j\}$. ∎

Note that this result is a little weaker than that of Theorem 2.2 which allowed the affine spaces on which the residual norm is minimized to be of the form $x_k + \text{Span}\{U_j\}$ for any k between 0 and j. Similar to the full-depth case, we may now ask whether the vector $\bar{x}_j$ corresponds to the result of some other classical algorithms for linear systems. One may think that there should exist an equivalence with a similar method such as Truncated GCR (TGCR also known as ORTHOMIN, see e.g., [24]) or one of the other Krylov methods that rely on truncation in the orthogonalization, e.g., ORTHODIR, or DQGMRES [24]. While this is possible, we did not find an obvious result that showed such an equivalence.

3. Restarting AATGS. Lines 6-12 of Algorithm 2.1 carry-out an orthonormalization of the vector q_j versus $q_{j_m+1}, \cdots, q_{j-1}$ and imposes the same operations undergone by the sequence $\{q_i\}$ to the sequence $\{u_i\}$. While the columns of Q_j are orthonormal, those of U_j are not, and they are prone to numerical instability. Therefore, it is essential to check for the onset of instability, especially when the problem is neither linear nor positive definite. To take advantage of the short-term recurrence while also preserving accuracy, we introduce a lightweight strategy to determine when a restart is deemed necessary.

Using the same notation as in Algorithm 2.1, the propagation of U_j can be expressed in the following matrix form:

$$\begin{bmatrix} u_{j_m+2}^T \\ \vdots \\ u_{j-1}^T \\ u_j^T \end{bmatrix} = \begin{bmatrix} 0 & 1 & & \\ \vdots & & \ddots & \\ 0 & & & 1 \\ -\frac{s_{j_m+1,j}}{s_{jj}} & -\frac{s_{j_m+2,j}}{s_{jj}} & \cdots & -\frac{s_{j-1,j}}{s_{jj}} \end{bmatrix} \begin{bmatrix} u_{j_m+1}^T \\ \vdots \\ u_{j-2}^T \\ u_{j-1}^T \end{bmatrix} + \frac{1}{s_{jj}} \begin{bmatrix} 0 \\ \vdots \\ 0 \\ \Delta x^T \end{bmatrix} \tag{3.1}$$

where $\Delta x := x_j - x_{j-1}$. Note that the above system need not be formed explicitly. We point out that (3.1) is applied element-wise to the columns of U_j. This means that the k-th component of u_j can be derived by applying the operations in (3.1) to the k-th element of Δx and the k-th row of U_j, i.e., if $v^{(k)}$ refers to the k-th component of a vector v, we have

$$u_j^{(k)} = \frac{1}{s_{jj}} \Delta x^{(k)} - \sum_{i=j_m+1}^{j-1} \frac{s_{ij}}{s_{jj}} u_i^{(k)}. \tag{3.2}$$

We now analyze how the accumulation of the errors from the computation of previous u_i's affect the accuracy of the most recent u_j. For this we denote the computed version of u_i as $\tilde{u}_i = u_i + \varepsilon_i$, where $\varepsilon_i \in \mathbb{R}^n$ represents the error introduced during the computation of u_i. We also denote the rounding errors introduced during the computation of u_j at step $j-1$ by δ_j and we assume that

$$\|\delta_j\|_\infty \leq C \cdot \|\Delta x\|_\infty / s_{jj} \tag{3.3}$$

where C is a constant.

Then, the perturbed version of (3.2) becomes:

$$\tilde{u}_j^{(k)} = \frac{1}{s_{jj}} \Delta x^{(k)} - \sum_{i=j_m+1}^{j-1} \frac{s_{ij}}{s_{jj}} \tilde{u}_i^{(k)} + \delta_j^{(k)}. \tag{3.4}$$

We then substitute $\tilde{u}_j = u_j + \varepsilon_j$ and $\tilde{u}_i = u_i + \varepsilon_i$ into Equation (3.4) and subtract Equation (3.2). This leads to:

$$\varepsilon_j^{(k)} = - \sum_{i=j_m+1}^{j-1} \frac{s_{ij}}{s_{jj}} \varepsilon_i^{(k)} + \delta_j^{(k)}. \tag{3.5}$$

Therefore,

$$\begin{aligned} |\varepsilon_j^{(k)}| &\leq \sum_{i=j_m+1}^{j-1} \frac{|s_{ij}|}{s_{jj}} |\varepsilon_i^{(k)}| + |\delta_j^{(k)}| \\ &\leq \sum_{i=j_m+1}^{j-1} \frac{|s_{ij}|}{s_{jj}} \|\varepsilon_i\|_\infty + \frac{C}{s_{jj}} \|\Delta x\|_\infty. \end{aligned} \tag{3.6}$$

This leads us to define the following scalar sequence w_j to monitor the behavior of the bound (3.6):

$$w_j := \sum_{i=j_m+1}^{j-1} \frac{|s_{ij}|}{s_{jj}} w_i + \frac{C}{s_{jj}} \|\Delta x\|_\infty. \tag{3.7}$$

The sequence w_j is just an upper bound for the infinity norm of the error vector ε_j and it can be used to monitor the growth of the rounding errors. When w_j exceeds a threshold $\eta > 0$, we

should discard all vectors in U_j and Q_j and set $j_m \equiv j$. The next iteration then computes Δx and Δf in Lines 4-5 of Algorithm 2.1 using the latest pairs x_{j+1}, x_j along with related f_{j+1}, f_j and set $w_{j+1} = C \cdot \|\Delta x\|_\infty / s_{j+1,j+1}$ to restart monitoring the growth of the rounding errors. The auto-restart version of Algorithm 2.1 is briefly summarized in Algorithm 3.1. In the experiments section, we set $C = 1$, unless otherwise specified.

Algorithm 3.1 AATGS(m) with Restarting

1: **Input**: Function $f(x)$, initial guess x_0, window size m, threshold η, constant C.
2: Set $f_0 \equiv f(x_0), \quad x_1 = x_0 + \beta_0 f_0, \quad f_1 \equiv f(x_1), \quad w_0 \equiv 0, \quad j_m = 0$
3: **for** $j = 1, 2, \cdots$, until convergence **do**
4: Update $j_m := \max\{j_m, j - m\}$
5: Run lines 4-16 of Algorithm 2.1
6: $w_j := C \cdot \|\Delta x\|_\infty / s_{jj} + \sum_{i=j_m+1}^{j-1} (|s_{ij}|/s_{jj}) w_i$
7: **if** $w_j > \eta$ **then**
8: Set $j_m \equiv j$ and $Q_j \equiv [\,]$, $U_j \equiv [\,]$
9: **end if**
10: **end for**

Here are some details and comments on Algorithm 3.1:

- **Line 2:** Same as the initialization step in Algorithm 2.1. In the implementation, we can allocate a vector of length m to store w_j's.
- **Line 6:** Note that when $j - 1 < j_m$, i.e., in the first step after a restart, the sum in the expression is empty and therefore equal to zero. In this case both Q_j and U_j are empty. In this situation $w_j := C\|\Delta x\|_\infty / s_{jj}$ reflecting the fact that there are no errors propagating from earlier steps.
- **Lines 7-9:** When w_j surpasses the given threshold η, a restart is necessary because the stability is compromised. For a restart, we set $j_m \equiv j$ and discard all stored vectors in Q_j and U_j. We only retain the last two iterates, x_j and x_{j+1} as well as f_j and f_{j+1} to continue the process when we compute Δx and Δf in the next iteration. Algorithm 3.1 will generate mathematically the same iterates as Algorithm 2.1 if this condition is not met.

4. Experiments. This section presents a few experiments on nonlinear problems to compare AATGS with the standard AA. We also include the results of the fixed point iteration in our experiments. Since it is common practice to add a fixed restart for AA (i.e., clearing $\mathcal{X}_j$ and $\mathcal{F}_j$ every fixed number of iterations), we incorporate a fixed restart for both AATGS (in addition to the auto-restart strategy discussed in Section 3) and AA. For AA, to restart after obtaining x_{j+1}, we discard all vectors in $\mathcal{X}_j$ and $\mathcal{F}_j$. The next iteration then begins by computing Δx_j and Δf_j using x_{j+1} and x_j along with related f_{j+1} and f_j. In the figures in this section, we use the notation AATGS$[m, d]$ and AA$[m, d]$ to represent AATGS and AA with window size m and fixed restart dimension d. When d is replaced with a '$-$', fixed restart is disabled. Unless otherwise noted, we set the threshold parameter η in Algorithm 3.1 to 10^3. Note that standard AA performance is highly dependent on window size m and fixed restart dimension d. While we present results for only a few AA parameters, we employ a grid search to select the best-performing AA configurations. In our tests, the max number of iterations and stopping tolerance for the relative norm of $f(x)$ varies based on problem size, convergence rate, and the initial norm of $f(x)$.

Our results demonstrate that AATGS achieves performance comparable to the standard AA method with equivalent window sizes when applied to highly non-symmetric and nonlinear problems. Furthermore, because of the short-term recurrence incorporated in AATGS, it outperforms AA on problems that are close to symmetric linear, even with a much smaller window size. These experiments illustrate the properties of AATGS shown in the previous sections. We also demonstrate the effectiveness of the restarting strategy. Although it is possible to carefully tune the parameters and generate competitive results using standard AA, the proposed auto-restart AATGS has the advantage of not requiring the selection of the restart dimension.

All of the methods were implemented in MATLAB 2023a. We implemented AA by solving the least-square problem shown in Equation 1.5 using the pseudo-inverse with MATLAB's `pinv` function. All experiments were conducted on the `Agate` cluster at the Minnesota Supercomputing Institute. The

computing node features 64 GB of memory and is equipped with two sockets, each having a 2.45GHz AMD EPYC 7763 64-Core Processor.

4.1. Bratu Problem. In our first experiment, we solve a problem with a low degree of non-linearity to demonstrate the benefits of the short-term recurrence in AATGS. We consider a finite difference discretization of the following *modified Bratu problem* [10] with Dirichlet boundary condition:

$$\begin{aligned} \Delta u + \alpha u_x + \lambda e^u = 0 \quad &\text{in } \Omega, \\ u = 0 \quad &\text{on } \partial\Omega, \end{aligned} \tag{4.1}$$

where $\Omega = [0,1]^2$. We use *centered finite differences* [3, 8, 31] to discretize the equation on a 202×202 grid (including boundary points). For our boundary value problem, this discretization results in a system of nonlinear equations with $n = 200 \times 200 = 40,000$ unknowns of the form:

$$f(v) = Av + h \cdot \alpha Bv + h^2 \cdot \lambda \exp(v) = 0, \tag{4.2}$$

where $v \in \mathbb{R}^n$ is the numerical solution at n interior grid points, $h = 1/201$ is the mesh size, $A \in \mathbb{R}^{n\times n}$ is a symmetric matrix, and $B \in \mathbb{R}^{n\times n}$ is a skew-symmetric matrix. The fixed point iteration takes the form:

$$g(v) = v + \beta f(v) = v + \beta(Av + h \cdot \alpha Bv + h^2 \cdot \lambda \exp(v)). \tag{4.3}$$

The parameter λ in the equation influences the change rate in the Jacobian of the problem. Denoting by $J(v)$ the Jacobian at v, we have

$$\|J(v_{j+1}) - J(v_j)\|_{\max} \le h^2 \cdot \lambda \| \exp(v_{j+1}) - \exp(v_j)\|_\infty, \tag{4.4}$$

where $\|\cdot\|_{\max}$ is the matrix max norm. This indicates that a larger λ can potentially increase the non-linearity of the problem. We take $\lambda = 1$ in all of our experiments so that the equation is physically meaningful. In this case, the Jacobian's variation is limited, resulting in an almost linear problem. The parameter α controls the degree of symmetry of the problem. We test both symmetric ($\alpha = 0$) and non-symmetric ($\alpha \neq 0$) cases.

In all our experiments on Bratu problem, we use the zero vector as the initial solution and set the parameter $\beta = 1.0$ for both AATGS and AA. For comparison, we also include the results of fixed point iteration with $\beta = 0.1$.

We begin our experiments with the symmetric case where $\alpha = 0$. To highlight the benefits of AATGS's short-term recurrence, we compare AATGS(3) with AA(20) and AA(100), and disable the restart for both AATGS and AA. The left panel of Figure 4.1 plots the iteration number versus the residual norm $\|f(v)\|_2$. We can observe from the figure that AATGS performs better than AA in this experiment, even with a much smaller window size. This is because when $\lambda = 1$, the problem is close to a symmetric linear problem. In this case, AATGS(3) behaves similarly to AATGS(∞), which is equivalent to AA(∞). This explains its superior performance compared to AA(20) and AA(100), demonstrating the potential advantage of AATGS over AA in handling nearly linear and symmetric problems.

Next, we set α to 20 and solve a non-symmetric problem, noting that it remains nearly linear. We first study the performance of AATGS with different restart strategies: no restart, fixed restart with dimension 50, and auto-restart with $\eta = 10^3$. Since the problem is no longer symmetric, we slightly increase the window size to $m = 5$. We can observe from the middle panel of Figure 4.1 that the AATGS(5) without restart underperforms the other two options. The two restart versions have similar performance and the auto-restart is slightly better in this experiment. This shows the importance of restart strategies. As restart strategies can be very useful, in the following tests, we enable restart strategies for both AATGS and AA.

Finally, we compare the performance of AATGS and AA for the same non-symmetric problem with $\alpha = 20$. We compare AATGS(5) with auto-restart ($\eta = 10^3$) against AA(5) and AA(20), both with a fixed restart dimension of 50. The results, shown in the right panel of Figure 4.1, demonstrate that AATGS(5) outperforms AA(5) and shows results comparable to those of AA(20). This indicates that AATGS constructs a more effective subspace than standard AA even when the Jacobian is not symmetric.

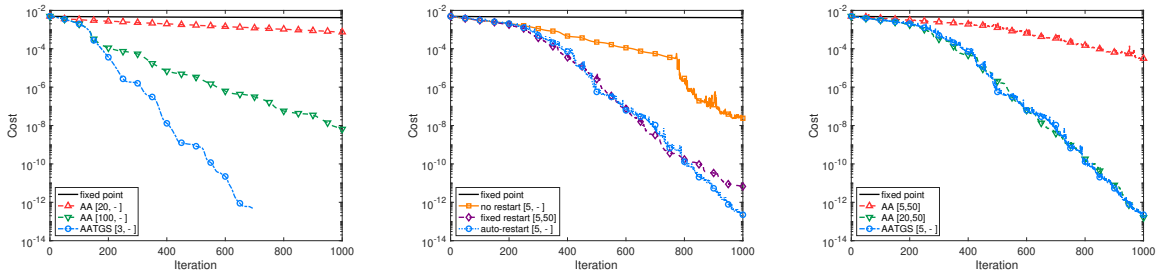

Figure 4.1: Bratu problem with initial solution $v_0 = 0$ and $\lambda = 1$. (left) AATGS and AA with no restart for symmetric Jacobian with $\alpha = 0$; (middle) AATGS with no restart, a fixed restart, and auto-restart for the non-symmetric Jacobian case. (right) AATGS with auto-restart and AA with a fixed restart for non-symmetric Jacobian with $\alpha = 20$. x-axis is the iteration number and y-axis is the residual norm $\|f(v)\|_2$. Here, $[\cdot, \cdot]$ indicates the window size and the restart dimension of each method.

4.2. Chandrasekhar's H-equation. Next, we evaluate our method for Chandrasekhar's H-equation [13]. A form of the equation can be written as:

$$H(\mu) - \left(1 - \frac{\omega}{2}\int_0^1 \frac{\mu H(\nu)}{\mu + \nu} d\nu\right)^{-1} = 0, \tag{4.5}$$

where $\omega \in [0, 1]$ is a parameter, and we seek a solution $H \in C[0, 1]$. We discretize (4.5) on a uniform grid and obtain the following discretized problem [13]:

$$[f(h)]_i := h_i - \left(1 - \frac{\omega}{2n}\sum_{j=1}^{n} \frac{\mu_i h_j}{\mu_i + \mu_j}\right)^{-1} \tag{4.6}$$

where $h \in \mathbb{R}^n$ is the numerical solution at n grid points, $\mu_i = \frac{i-0.5}{n}$ for $1 \le i \le n$, and the component-wise expression of the corresponding fixed point iteration $h = g(h)$ is given by

$$[g(h)]_i = h_i + \beta[f(h)]_i = h_i + \beta\left[h_i - \left(1 - \frac{\omega}{2n}\sum_{j=1}^{n} \frac{\mu_i h_j}{\mu_i + \mu_j}\right)^{-1}\right]. \tag{4.7}$$

It is known that the Jacobian in this problem is non-symmetric [14], as indicated by its expression:

$$[J(h)]_{ik} = \delta_{ik} - \frac{\omega}{2n} \cdot \frac{\mu_i}{\mu_i + \mu_k} \cdot \left(1 - \frac{\omega}{2n}\sum_{j=1}^{n} \frac{\mu_i h_j}{\mu_i + \mu_j}\right)^{-2}, \tag{4.8}$$

where $\delta_{ik} = 1$ if $i = k$ and 0 otherwise. The choice of ω can have an impact on the convergence of solution algorithms [30]. In our experiments, we set $n = 1,000$ and consider cases with $\omega = 0.99$ and $\omega = 1.0$, both of which require careful timing for restarts in AA and AATGS.

In this group of experiments, we use the vector of all ones as the initial solution and again set the parameter $\beta = 1.0$ for both AATGS and AA. Since the problem size is much smaller, we apply a smaller fixed restart dimension of 20 for AA. We compare AATGS and AA with window sizes $m = 5$ and $m = 20$ and again include results for fixed point iteration with $\beta = 0.1$. In this problem, a larger m does not necessarily yield faster convergence, as observed from Figure 4.2 that AA(5) consistently outperforms AA(20). Furthermore, we can see that AA(20) stagnates before a restart is triggered at step 20, which demonstrates the usefulness of the restarting procedure in this problem. With auto-restart, AATGS makes a stable selection of the window size, as shown by the identical performance of AATGS(5) and AATGS(20) in both figures.

It is worth noting that a larger ω leads to a more challenging problem. When $\omega = 0.5$, the trajectories of AA(5), AA(20), AATGS(5), and AATGS(20) all overlap. However, when ω increases

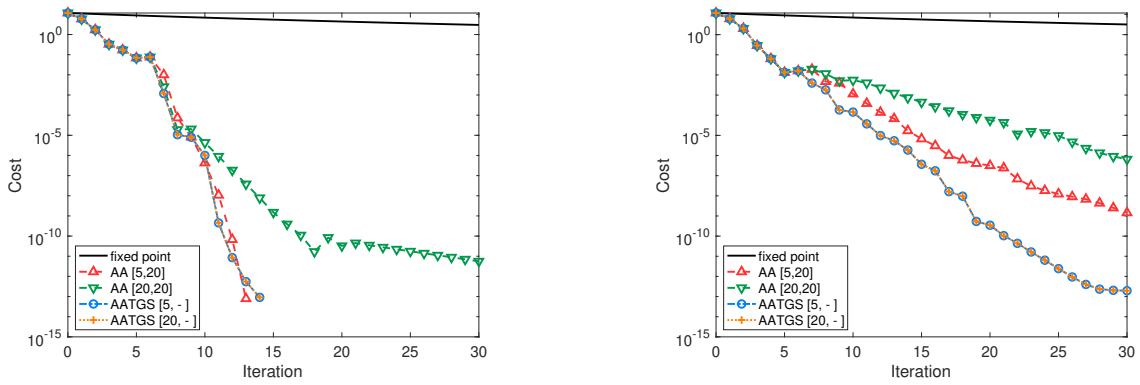

Figure 4.2: Chandrasekhar's H-equation with dimension $n = 1,000$. (left) The simpler case with $\omega = 0.99$; (right) The harder case with $\omega = 1.0$. x-axis is the iteration number and y-axis is the residual norm $\|f(h)\|_2$. Here, $[\cdot, \cdot]$ indicates the window size and the restart dimension of each method.

to 0.99, AA(20) fails to catch up with the other methods. When $\omega = 1.0$, AATGS outperforms AA. This enhanced robustness of AATGS in dealing with numerical stability issues in the sequence of x_j's can also be attributed to the auto-restart strategy.

4.3. Lennard-Jones problem. Next, we evaluate the performance of AATGS when solving the unconstrained minimization problem of the form

$$\min_x \phi(x). \tag{4.9}$$

We define $f(x) = -\nabla\phi(x)$, and write the fixed point iteration in the gradient descent form

$$g(x) = x + \beta f(x) = x + \beta(-\nabla\phi(x)). \tag{4.10}$$

Specifically, we optimize the geometry of molecules to achieve a minimum total Lennard-Jones (LJ) potential energy. The LJ potential is defined as follows[1]:

$$E(Y) = \sum_{i=1}^{N}\sum_{j=1}^{i-1} 4\epsilon\left[\left(\frac{\delta}{\|Y_{i,:} - Y_{j,:}\|}\right)^{12} - \left(\frac{\delta}{\|Y_{i,:} - Y_{j,:}\|}\right)^{6}\right]. \tag{4.11}$$

In this formulation, N is the number of atoms, ϵ represents the well depth, δ is the distance between two non-bonding particles, and $Y \in \mathbb{R}^{N\times 3}$ with its i-th row $Y_{i,:}$ representing the coordinates of atom i. We reformulate the problem by reshaping Y into $x \in \mathbb{R}^{3N}$ where $[x_{3i-2}, x_{3i-1}, x_{3i}] = Y_{i,:}$ and defining the loss function $\phi(x) = E(Y)$. In our experiments, we set both ϵ and δ to 1 and simulate an Argon cluster starting from a perturbed initial Face-Centered-Cubic (FCC) structure [16]. We took 3 cells per direction, resulting in 27 unit cells. Given that each FCC cell includes 4 atoms, there are $N = 108$ atoms in total. The challenge in this problem arises from the high exponents in the potential.

Figure 4.3 (left) shows an illustration of the geometry optimization in this problem, where the initial positions of the atoms are shown as blue dots, and the red triangles indicate the optimized final positions, which represent a local minimum around the initial positions rather than a global optimum. We take $\beta = 1.5 \times 10^{-4}$ in our experiments for both AATGS and AA. Given that this is an unconstrained optimization problem, the Jacobian of $\nabla\phi(x)$ is also the Hessian of $\phi(x)$, which is always symmetric. Therefore, we set the window size of AATGS to $m = 3$. In Figure 4.3 (right), we compare AATGS against standard AA in three configurations. We can see that AATGS with a window size of $m = 3$ and auto-restart strategy outperforms others. AA with $m = 20$ and a restart dimension of 100 performs similarly to AA with $m = 3$ and restart 10, and both surpass AA with $m = 3$ and a restart dimension of 100. It again demonstrates the usefulness of the auto-restart strategy in AATGS for a non-trivial optimization problem.

[1]Thanks: We benefited from Stefan Goedecker's course site at Basel University.

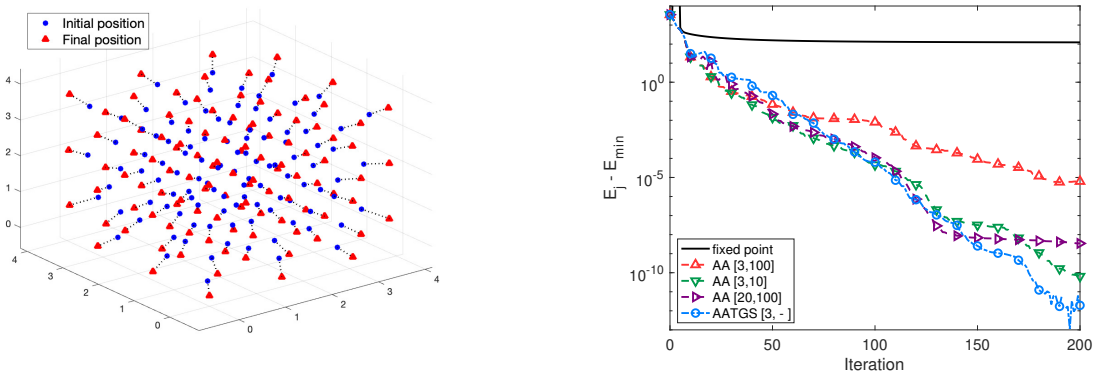

Figure 4.3: The Lennard-Jones problem. (left) The geometry of particles at the initial state and the final state; (right) The results of various methods in this experiment. x-axis is the iteration number and y-axis is the shifted energy $E_j - E_{\min}$. Note that, $E_{\min}$ is the minimum energy achieved by all considered methods so that the shifted energy is always positive. $[\cdot, \cdot]$ indicates the window size and the restart dimension of each method.

4.4. Steady Navier–Stokes equations. In our next experiment, we aim to solve a 2D lid-driven cavity problem described by the steady Navier-Stokes equations (NSEs):

$$\begin{aligned} u \cdot \nabla u + \nabla p - Re^{-1} \Delta u &= f, \\ \nabla \cdot u &= 0, \end{aligned} \tag{4.12}$$

with the domain $\Omega = (0,1)^2$ and the Dirichlet boundary condition $(u,p) = (0,0)$ on the sides and bottom and $(1,0)$ on the lid. Following the settings in [18], we set the Reynolds number $Re = 10,000$ and use an initial guess of all zeros [2]. The discretization results in a problem of size 190,643. Readers can refer to [18] for details of the mesh. The fixed point iteration used by both AATGS and AA takes the form:

$$g(v) = v + \beta f(v) = v + \beta(h(v) - v), \tag{4.13}$$

where v is the discretization of (u,p) on grid points, and $h(v)$ performs one step of Picard iteration which maps v to some specific approximate solution. Details on $h(v)$ can be found in [19].

The results in Figure 4.4 compare Picard iterations, AA with window sizes $m = 5$ and $m = 10$, and AATGS with a window size of $m = 5$. A restart is not necessary in this experiment since we can observe that both AA and AATGS converge without stagnation. We also observe that Picard iteration fails to converge, which is likely due to the extremely large Reynolds number. Both AATGS and AA manage to converge at a similar rate. Given the non-symmetric and nonlinear nature of this problem, we cannot expect significant gains from AATGS over AA in this case. Indeed, the methods behave similarly.

4.5. Regularized Logistic Regression. Regularized logistic regression is a powerful tool for binary classification tasks, particularly when dealing with datasets that have a large number of features. In this experiment, we investigate the application of regularized logistic regression to the Madelon dataset [3]. The training set consists of $N = 2,000$ samples and $n = 500$ features. The objective can be formulated as follows:

$$\min_{\theta} \ \frac{1}{N} \sum_{i=1}^{N} \log(1 + \exp(-y_i \cdot x_i^\top \theta)) + \frac{\lambda}{2} \|\theta\|_2^2, \tag{4.14}$$

[2]Thanks: We would like to thank Sara Pollack and Leo G. Rebholz for sharing their 2D Steady Navier-Stokes equation codes with us.

[3]https://archive.ics.uci.edu/dataset/171/madelon

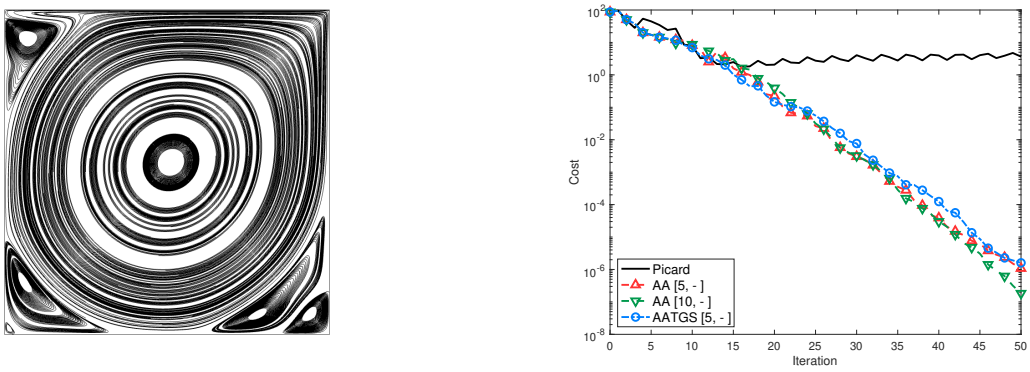

Figure 4.4: 2D Steady Navier-Stokes equations with the Reynolds number $Re = 10,000$. (left) The streamlines of the solution given by AATGS at step 50; (right) The results of various methods in this experiment. x-axis is the iteration number and y-axis is the residual norm of $\|\text{Picard}(v) - v\|_2$. $[\cdot, \cdot]$ indicates the window size and the restart dimension of each method.

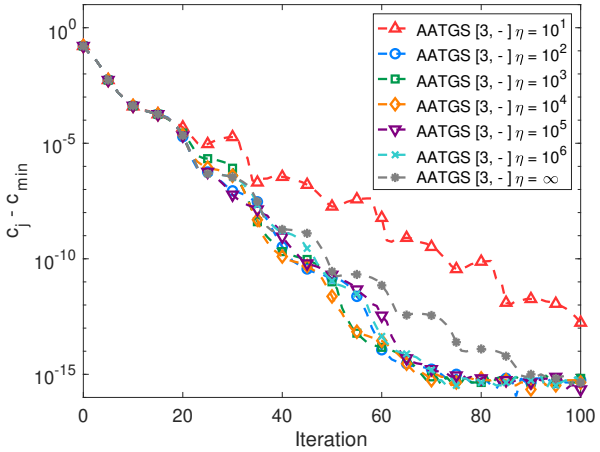

Figure 4.5: The results of various η's for the regularized logistic regression on the Madelon dataset. x-axis is the iteration number and y-axis is the shifted training loss $c_j - c_{\min}$. Note that, $c_{\min}$ is the minimum training loss achieved by all considered methods so that the shifted loss is always positive. $[\cdot, \cdot]$ indicates the window size and the restart dimension of each method.

where x_i represents the feature vector of the i-th sample (each feature is normalized to have a mean of 0 and a standard deviation of 1 across all samples), y_i represents the label of the i-th sample (either -1 or 1 for binary classification), $\theta \in \mathbb{R}^n$ is the parameter vector to be optimized, λ is the regularization parameter that controls the balance between fitting the training data well and preventing overfitting by penalizing large parameter values.

Figure 4.5 illustrates the shifted training loss as a function of the iteration number. We set the fixed point iteration parameter $\beta = 1.0$, the regularization parameter $\lambda = 0.01$, and the window size $m = 3$. We use the zero vector as the initial solution. In this comparison, we focus on AATGS with varying auto-restart threshold η ranging from 10^1 to ∞. The results demonstrate the efficacy and simplicity of parameter tuning for our auto-restart strategy, as the loss curves for $\eta = 10^2$ to 10^6 show small variance. The performance deteriorates only when $\eta = 10^1$ – resulting in excessive, redundant restarts – and when $\eta = \infty$ – leads to the absence of restarts. Through our testing across many experiments, the default setting of $\eta = 10^3$ often delivers a sufficiently accurate solution.

In Table 4.1, we present the number of iterations (up to 1000) required for AATGS to achieve a relative loss smaller than 10^{-12}. The regularization parameter λ varies from 10^0 to 10^{-5}, changing the

optimization problems from relatively simple to significantly difficult to solve. It is important to note that our goal in these comparisons is not to achieve the highest accuracy but rather to elucidate the characteristics of AATGS. With a window size of $m = 3$, it is observed that as the problem becomes more challenging (with smaller λ), the number of required iterations generally increases. However, AATGS with $\eta = 10^3$ to 10^5 always exhibits similar performance. Only extremely high or low η's tend to be significantly slower than other values and fail to converge within 1000 iterations. This further confirms that η offers a broad selection range.

λ	Number of Iterations					
	$\eta = 10^1$	$\eta = 10^2$	$\eta = 10^3$	$\eta = 10^4$	$\eta = 10^5$	$\eta = \infty$
10^0	21	20	22	22	22	22
10^{-1}	52	50	48	51	51	56
10^{-2}	200	113	105	117	113	167
10^{-3}	F	F	188	173	201	418
10^{-4}	F	473	251	209	228	F
10^{-5}	F	F	254	228	251	F

Table 4.1: A comparison of AATGS with a fixed window size $m = 3$ across various auto-restart thresholds η (columns) and regularization parameters λ (rows) is presented. This table displays the number of iterations required for AATGS to achieve a relative loss smaller than 10^{-12}. The notation 'F' indicates cases where the method fails to converge within 1000 iterations.

4.6. Minimax Optimization. Bilinear games are often regarded as an important example of understanding new algorithms and techniques for solving general minimax problems [9, 12]. In this experiment, we study the following zero-sum bilinear games:

$$\min_{x\in\mathbb{R}^n}\max_{y\in\mathbb{R}^n} \phi(x,y) = x^T A y + b^T x + c^T y, \tag{4.15}$$

where A is a full-rank matrix. The Nash equilibrium to the above problem is given by $(x^*, y^*) = (-A^{-T}c, -A^{-1}b)$. We use the alternating Gradient Descent Ascent (GDA) algorithm to solve the problem in the following form:

$$\begin{aligned}\begin{bmatrix} x_{j+1} \\ y_{j+1} \end{bmatrix} &= \begin{bmatrix} x_j \\ y_j \end{bmatrix} + \beta \cdot \begin{bmatrix} -\nabla_x \phi(x_j, y_j) \\ \nabla_y \phi(x_{j+1}, y_j) \end{bmatrix} \\ &= \begin{bmatrix} I & -\beta A \\ \beta A^T & I - \beta^2 A^T A \end{bmatrix} \begin{bmatrix} x_j \\ y_j \end{bmatrix} - \beta \begin{bmatrix} b \\ \beta A^T b - c \end{bmatrix}\end{aligned} \tag{4.16}$$

where the solution of the above fixed point iteration is the root of the following nonlinear equation f:

$$f\left(\begin{bmatrix} x \\ y \end{bmatrix}\right) := \begin{bmatrix} 0 & -A \\ A^T & -\beta A^T A \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} - \begin{bmatrix} b \\ \beta A^T b - c \end{bmatrix}. \tag{4.17}$$

The coefficients of the initial problem, $A \in \mathbb{R}^{100\times 100}$, $b \in \mathbb{R}^{100}$, and $c \in \mathbb{R}^{100}$, are generated using random numbers following the distribution $\mathcal{N}(0,1)$. Subsequently, A undergoes normalization to ensure its 2-norm equals 1. The initial guess is also generated using random numbers following the distribution $\mathcal{N}(0,1)$. The cost of this problem is defined as the relative distance to the optimal solution, i.e., $c_j := \|(x_j, y_j) - (x^*, y^*)\|_2 / \|(x^*, y^*)\|_2$, where (x_j, y_j) is the iteration at step j.

Note that Equation (4.16) is referred to as the *alternating* GDA since we update x_{j+1} and y_{j+1} in an alternating manner. For the *simultaneous* GDA, we update x_{j+1} and y_{j+1} simultaneously. However, this leads to a skew-symmetric linear system to solve (e.g., consider $\beta = 0$ in Equation (4.17)) which has numerical issues as mentioned at the end of Section 2.3. Furthermore, the difference in spectra is shown in Fig. 4.6 (left), where the eigenvalues of the coefficient matrix in simultaneous GDA are purely imaginary, while those of the alternating GDA have small real parts. Therefore, we consider the alternating GDA in this experiment.

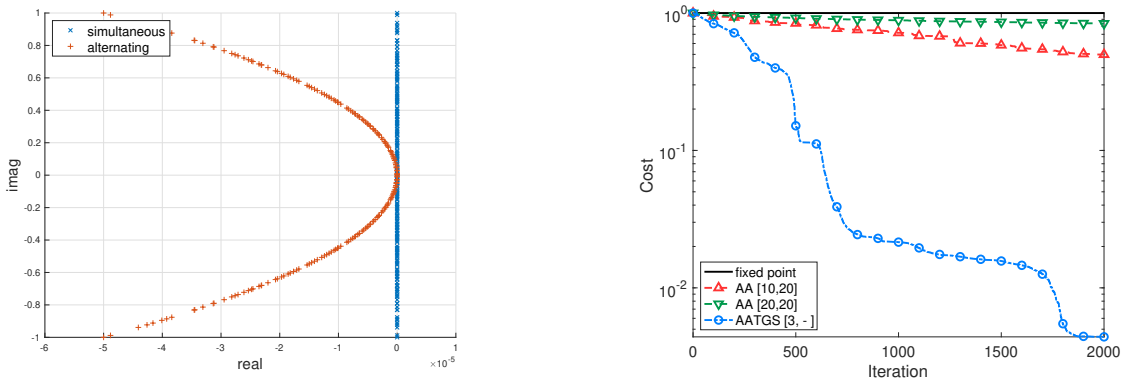

Figure 4.6: Minimax optimization on a bilinear game. (left) Spectrum of the linear systems corresponding to the simultaneous GDA and alternating GDA. x-axis is the real part and y-axis is the imaginary part. Blue crosses represent the eigenvalues of the simultaneous GDA. Red plus signs represent the eigenvalues of the alternating GDA; (right) The results of various methods in this experiment. x-axis is the iteration number and y-axis is the relative Euclidean distance to the optimal solution. $[\cdot, \cdot]$ indicates the window size and the restart dimension of each method.

In Figure 4.6 (right), we compare AATGS with standard AA under different settings. Since the coefficient matrix defined in function f in Equation (4.17) is skew-symmetric plus a symmetric perturbation, we expect a similar short-term recurrence in AATGS and therefore we set the window size to $m = 3$. In addition, we employ the auto-restart strategy instead of a fixed restart. For the baseline methods, we consider AA with window sizes $m = 10$ and $m = 20$, along with a fixed restart dimension of 20. Note that we use a smaller restart dimension because both AA options fail to converge if we use a restart dimension of 50. Moreover, we set $\beta = 10^{-4}$ to ensure that all methods do not diverge in most cases. We observe that after 2000 iterations, AATGS manages to converge with a relative distance of around 0.0044, while the AAs still have relative distances of 0.69 and 0.84 from the optimal solution. This experiment illustrates the appealing behavior of AATGS in solving linear problems that are nearly skew-symmetric.

5. Conclusion. This paper introduced what may be termed a 'symmetric version' of Anderson Acceleration. When the fixed point iteration handled by Anderson Acceleration is a linear iteration, then AA does not take advantage of symmetry in the case when the iteration matrix is also symmetric. The Truncated Gram-Schmidt variant of AA (AATGS) introduced in this paper, addresses this issue. AATGS is mathematically equivalent to AA when the depth of both algorithms is $m = \infty$. However, when the problem is linear and symmetric, AATGS(∞) simplifies in that only a few vectors must be saved instead of all of the previous directions generated, in order to produce the same iterates as $AA(\infty)$. This can lead to substantial savings in memory and computational requirements for large problems. From a practical point of view, the original AATGS algorithm without any modification can suffer from numerical stability issues. A careful restarting strategy was developed to restart when deemed necessary by a simple short-term scalar recurrence designed to mimic the behavior of the numerical errors. Equipped with this artifice, the algorithm showed good robustness, often outperforming the original AA at a lower cost. This was confirmed by a few numerical experiments, with applications ranging from nonlinear partial differential equations to challenging optimization problems. The numerical experiments showed that for problems whose Jacobien is nearly symmetric and for optimization problems (Hessian is symmetric), AATGS can be vastly superior to AA and this is expected from theory.

In the future, we plan to explore the applicability and efficacy of AATGS when applied to stochastic optimization problems. We will also study the exploitation of information on the Jacobian during the iteration in order to improve both robustness and efficiency, as done in [11].

6. Acknowledgements. The authors acknowledge the Minnesota Supercomputing Institute (MSI) at the University of Minnesota for providing resources that contributed to the research results

reported within this paper (http://www.msi.umn.edu).

REFERENCES

[1] D. G. Anderson, *Iterative procedures for non-linear integral equations*, Assoc. Comput. Mach., 12 (1965), pp. 547–560.

[2] W. Bian, X. Chen, and C. T. Kelley, *Anderson acceleration for a class of nonsmooth fixed-point problems*, SIAM Journal on Scientific Computing, 43 (2021), pp. S1–S20.

[3] M. H. Chaudhry et al., *Open-channel flow*, vol. 523, Springer, 2008.

[4] H. De Sterck and Y. He, *Linear asymptotic convergence of anderson acceleration: Fixed-point analysis*, SIAM Journal on Matrix Analysis and Applications, 43 (2022), pp. 1755–1783.

[5] S. C. Eisenstat, H. C. Elman, and M. H. Schultz, *Variational iterative methods for nonsymmetric systems of linear equations*, SIAM Journal on Numerical Analysis, 20 (1983), pp. 345–357.

[6] C. Evans, S. Pollock, L. G. Rebholz, and M. Xiao, *A proof that anderson acceleration improves the convergence rate in linearly converging fixed-point methods (but not in those converging quadratically)*, SIAM Journal on Numerical Analysis, 58 (2020), pp. 788–810.

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

[8] G. B. Folland, *Introduction to partial differential equations*, vol. 102, Princeton university press, 1995.

[9] G. Gidel, H. Berard, G. Vignoud, P. Vincent, and S. Lacoste-Julien, *A variational inequality perspective on generative adversarial networks*, in 7th International Conference on Learning Representations, ICLR, 2019.

[10] M. Hajipour, A. Jajarmi, and D. Baleanu, *On the accurate discretization of a highly nonlinear boundary value problem*, Numerical Algorithms, 79 (2018), pp. 679–695.

[11] H. He, Z. Tang, S. Zhao, Y. Saad, and Y. Xi, *nltgcr: A class of nonlinear acceleration procedures based on conjugate residuals*, SIAM Journal on Matrix Analysis and Applications, 45 (2024), pp. 712–743.

[12] H. He, S. Zhao, Y. Xi, J. C. Ho, and Y. Saad, *GDA-AM: on the effectiveness of solving min-imax optimization via anderson mixing*, in The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022, OpenReview.net, 2022.

[13] C. T. Kelley, *Iterative Methods for Linear and Nonlinear Equations*, Society for Industrial and Applied Mathematics, 1995.

[14] D. Lin, H. Ye, and Z. Zhang, *Explicit superlinear convergence rates of broyden's methods in nonlinear equations*, arXiv preprint arXiv:2109.01974, (2021).

[15] V. V. Mai and M. Johansson, *Anderson acceleration of proximal gradient methods*, in Proceedings of the 37th International Conference on Machine Learning, ICML'20, JMLR.org, 2020.

[16] L. Meyer, C. Barrett, and P. Haasen, *New crystalline phase in solid argon and its solid solutions*, The Journal of Chemical Physics, 40 (1964), pp. 2744–2745.

[17] Y. Peng, B. Deng, J. Zhang, F. Geng, W. Qin, and L. Liu, *Anderson acceleration for geometry optimization and physics simulation*, ACM Trans. Graph., 37 (2018).

[18] S. Pollock and L. G. Rebholz, *Filtering for anderson acceleration*, SIAM Journal on Scientific Computing, 45 (2023), pp. A1571–A1590.

[19] S. Pollock, L. G. Rebholz, and M. Xiao, *Anderson-accelerated convergence of picard iterations for incompressible navier–stokes equations*, SIAM Journal on Numerical Analysis, 57 (2019), pp. 615–637.

[20] P. Pulay, *Convergence acceleration of iterative sequences. the case of SCF iteration*, Chem. Phys. Lett., 73 (1980), pp. 393–398.

[21] ———, *Improved SCF convergence acceleration*, J. Comput. Chem., 3 (1982), pp. 556–560.

[22] L. G. Rebholz and M. Xiao, *The effect of anderson acceleration on superlinear and sublinear convergence*, J. Sci. Comput., 96 (2023).

[23] H. ren Fang and Y. Saad, *Two classes of multisecant methods for nonlinear acceleration*, Numer Linear Algebra Appl., 16 (2009), pp. 197–221.

[24] Y. Saad, *Iterative Methods for Sparse Linear Systems, 2nd edition*, SIAM, Philadelpha, PA, 2003.

[25] Y. Saad and M. H. Schultz, *GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Stat. Comput., 7 (1986), pp. 856–869.

[26] A. Toth and C. T. Kelley, *Convergence analysis for anderson acceleration*, SIAM Journal on Numerical Analysis, 53 (2015), pp. 805–819.

[27] H. F. Walker and P. Ni, *Anderson acceleration for fixed-point iterations*, SIAM Journal on Numerical Analysis, 49 (2011), pp. 1715–1735.

[28] D. Wang, Y. He, and H. De Sterck, *On the asymptotic linear convergence speed of anderson acceleration applied to admm*, J. Sci. Comput., 88 (2021).

[29] F. Wei, C. Bao, and Y. Liu, *Stochastic anderson mixing for nonconvex stochastic optimization*, in Advances in Neural Information Processing Systems, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, eds., 2021.

[30] F. Wei, C. Bao, Y. Liu, and G. Yang, *Convergence analysis for restarted anderson mixing and beyond*, arXiv preprint arXiv:2307.02062, (2023).

[31] P. Wilmott, S. Howson, S. Howison, J. Dewynne, et al., *The mathematics of financial derivatives: a student introduction*, Cambridge university press, 1995.

[32] F. Xue, *One-step convergence of inexact anderson acceleration for contractive and non-contractive mappings*, Electronic Transactions on Numerical Analysis, 55 (2022), pp. 285–309.