Analysis of Acceleration Techniques and Fast Nonlinear Solvers

Ning Wan

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Mathematics

Agnieszka Miedlar, Chair
Paul Cazeaux
Eric de Sturler
Mark Embree
Serkan Gugercin

December 11, 2025
Blacksburg, Virginia

Keywords: Anderson Acceleration, SCF Iterations, Quasi-Newton Methods, Krylov Methods

Analysis of Acceleration Techniques and Fast Nonlinear Solvers

Ning Wan

(ABSTRACT)

This dissertation focuses on the analysis and development of acceleration techniques and fast solvers for nonlinear systems of equations. Building upon the fixed-point and extrapolation frameworks introduced in the early chapters, we explore structural connections between residual-based acceleration methods and Krylov subspace techniques.

The first main contribution is a unified algebraic framework establishing the equivalence between the Anderson Acceleration method and the CROP (Conjugate Residual with Optimal Trial Vector) algorithm. By formulating both methods within a common affine subspace representation, we show that their full, untruncated forms produce identical iterates, motivating new hybrid variants such as CROP-Anderson and real-residual CROP (rCROP) methods.

The second contribution is a perturbation analysis of Anderson-type variants, examining the effects of deterministic and stochastic errors on convergence. Numerical experiments confirm that acceleration efficiency depends critically on both the choice of update strategy and the nature of perturbations.

The third contribution extends this unified perspective to nonlinear Krylov subspace methods. Nonlinear extensions of GMRESR, GCRO, and LGMRES are derived, forming the *nlKrylov* family of algorithms, and analyzed in the context of inexact Newton solvers, with convergence results established under relaxed conditions on residual and Jacobian approximations.

Analysis of Acceleration Techniques and Fast Nonlinear Solvers

Ning Wan

(GENERAL AUDIENCE ABSTRACT)

This dissertation studies mathematical methods that accelerate the solution of nonlinear equations, which arise in many areas of science and engineering. Traditional iterative solvers can be slow or may fail to converge on challenging problems. Acceleration techniques improve convergence by intelligently using information from previous iterations. This work focuses on developing and analyzing techniques that accelerate these computations, making them faster and more reliable.

The first major contribution shows a deep connection between two popular acceleration methods, demonstrating that, under ideal conditions, they produce identical results. This insight leads to new hybrid approaches that combine the strengths of both methods.

The second contribution studies how errors-either from approximations or random perturbations-affect the performance of these acceleration techniques. Numerical experiments show that careful design of the update strategies is essential for maintaining efficiency.

Finally, the dissertation extends these ideas to a broader family of advanced iterative methods, providing new algorithms and theoretical guarantees for solving large-scale nonlinear problems more effectively.

Overall, this dissertation provides a unified perspective on acceleration strategies and demonstrates how they can be applied to modern large-scale nonlinear computations.

Dedication

To my family, for their endless love, patience, and encouragement throughout this journey. To my teachers and my advisor, who have guided me with wisdom and kindness. And to my friends and colleagues, for sharing both challenges and laughter along the way.

Acknowledgments

I would like to express my deepest gratitude to my advisor, Prof. Agnieszka Międlar, for her invaluable guidance, continuous encouragement, and insightful discussions throughout the course of this research. Her expertise, patience, and high standards of scholarship have been a constant source of inspiration and have greatly shaped the quality of this work.

I am also sincerely thankful to the members of my thesis committee: Prof. Paul Cazeaux, Prof. Eric de Sturler, Prof. Mark Embree, and Prof. Serkan Gugercin, for their constructive feedback and valuable suggestions, which helped refine both the theoretical and experimental aspects of this dissertation.

Special thanks go to my colleague Tom Werner for his stimulating discussions, technical assistance, and friendship, which made the research environment both productive and enjoyable. His visits during Spring 2024 and Fall 2025 were particularly inspiring and contributed significantly to the progress of this work.

I am deeply grateful to Prof. Yousef Saad, Prof. Antoine Levitt, Prof. Mi-Song Dupuy, Prof. Massimiliano Lupo Pasini, and all the researchers who shared their insights and took the time to discuss ideas related to this project. Their valuable input and generous exchange of knowledge have greatly enriched this study.

I also wish to express my appreciation to the Department of Mathematics at Virginia Tech for providing an excellent academic environment and for the continuous technical and administrative support. I am especially thankful to the departmental staff for their assistance and dedication, which have made my graduate experience smooth and rewarding.

This work was partially supported by the National Science Foundation (NSF) under grants

DMS #2144181 and #2324958. Their support is gratefully acknowledged.

Lastly, I owe my deepest thanks to my family and friends for their endless encouragement, patience, and unconditional support throughout this journey. Their love and understanding have been my greatest source of strength.

Contents

List of Abbreviations

AA **A**nderson **A**cceleration

CROP **C**onjugate **R**esidual algorithm with **OP**timal trial vectors

SCF **S**elf-**C**onsistent **F**ield

Chapter 1

Introduction

This chapter outlines the motivation, context, and scope of the thesis. Section 1.1 presents the motivation for this study and frames the central problem under investigation. Section 1.2 specifies the main research objectives. Section 1.3 introduces representative sample problems arising from real-world applications, illustrating how basic iterative methods perform in practice and motivating the development of acceleration techniques. Section 1.4 summarizes the notation used throughout the thesis to ensure clarity and consistency. Finally, Section 1.5 provides an overview of the thesis structure and guides the reader through the progression of subsequent chapters.

1.1 Motivation

Nonlinear equations and fixed-point problems form the computational core of many scientific and engineering applications, such as fluid dynamics, quantum chemistry, optimization, machine learning, or data analysis. These problems are written as

$$f(x) = 0, \qquad \text{or} \quad \text{equivalently} \qquad x = g(x),$$

where the second form is the fixed-point formulation, which is conceptually simple and general. It is widely used in modeling considerations, splitting schemes, or operator compo-

sitions. Classical fixed-point iteration

$$x^{(k+1)} = g(x^{(k)}),$$

is one of the simplest nonlinear solvers. However, it typically converges only linearly under strong contractivity assumptions and may diverge in more challenging settings. As a result, many applications favor Newton-type methods, which reformulate the problem as a nonlinear equation to benefit from superlinear or quadratic local convergence and well-developed globalization strategies. Yet, Newton's method and its variants rely on Jacobian information, either explicitly computed or approximated, and therefore can be prohibitively expensive or impractical in large-scale or structure-rich settings.

This trade-off between simplicity and performance motivates the development of acceleration techniques that improve the convergence of fixed-point iterations while retaining their favorable structural properties. Among these methods, *Anderson Acceleration (AA)* [1, 2] has emerged as one of the most influential. Introduced by D. G. Anderson in 1965, AA leverages a history of past residuals and iterates to construct multisecant-like corrections, often yielding substantial gains in both speed and robustness. Its versatility has led to widespread adoption in a diverse range of fields, including self-consistent field (SCF) iterations in quantum chemistry and electronic structure calculations [3].

A particularly instructive example of these challenges appears in the *Self-Consistent Field (SCF)* iteration, a classical fixed-point method used in quantum chemistry and condensed-matter physics. In the SCF framework, one seeks a self-consistent solution to a nonlinear eigenvector problem

$$H(V)V = V\Lambda,$$

where $H(V)$ is a Hamiltonian matrix depending on the eigenvectors V, and Λ denotes a di-

agonal matrix of the eigenvalues. The SCF method updates the current iterate by repeatedly solving linearized eigenproblems, i.e.,

Algorithm 1 SCF iteration for (1.8)

Input: Initial guess $V_0 \in \mathbb{C}^{n\times r}$, with $V_0^H V_0 = I_r$

Output: $V \in \mathbb{C}^{n\times r}$ and $\Lambda \in \mathbb{C}^{r\times r}$, where Λ contains the r smallest eigenvalues of $H(V)$

1: **for** $k = 1, 2, \ldots$ until convergence **do**

2: Construct $H_k = H(V_{k-1})$

3: Compute V_k such that $H_k V_k = V_k \Lambda_k$, where Λ_k contains the r smallest eigenvalues of H_k.

4: **end for**

where Φ represents the process of solving the corresponding linearized eigenproblem and updating the electronic density or potential. Although conceptually simple, the SCF iteration often converges very slowly or even diverges, especially for systems with near-degenerate states or poor initial guesses. Let us follow an example given in [4]. We consider the nonlinear eigenvector problem with

$$H(V) = L + \alpha \cdot diag(L^{-1}\rho(V)),$$

where $\rho(V) = diag(VV^T)$ is the charge density of electrons in an electronic structure calculation, $V = [v_1, v_2]^T$, $\alpha = 12$ and

$$L = \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix}.$$

For this simple example, the SCF Algorithm 1 with initial guess $V_0 = [0.1389, 0.2028]^T$ gives a divergent sequence of the charge densities ρ, as we see in Figures 1.1 and 1.2. Applying Anderson Acceleration (AA) [1, 2] to the same problem, however, restores convergence, see

Figures 1.3 and 1.4. This striking improvement highlights both the practical power of AA and the open theoretical question of *why* it succeeds.

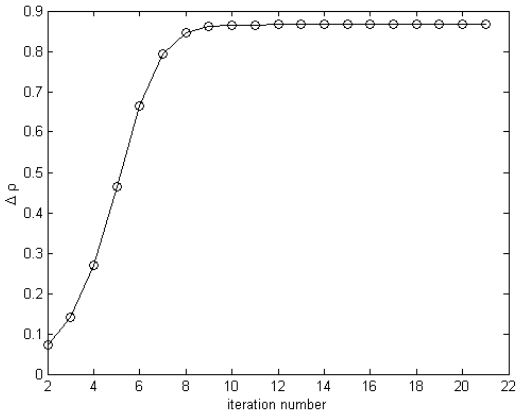


Figure 1.1: The change in charge density $\Delta\rho$ fails to converge to zero

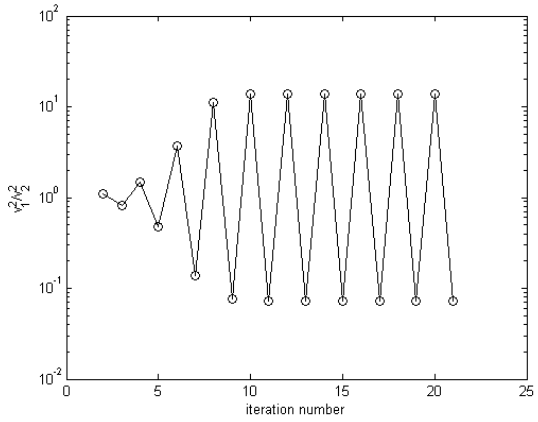


Figure 1.2: The ratio between v_1^2 and v_2^2 fails to converge

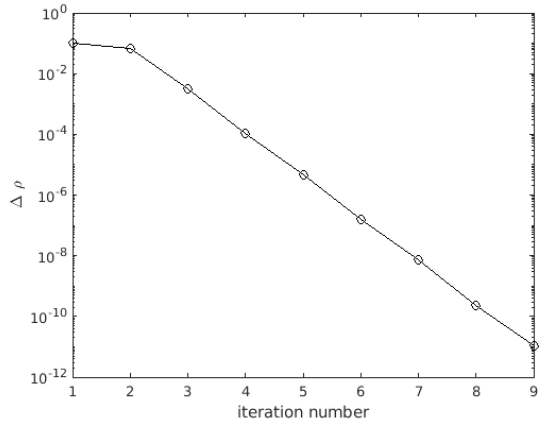


Figure 1.3: The change in $\Delta\rho$ in Anderson Acceleration.

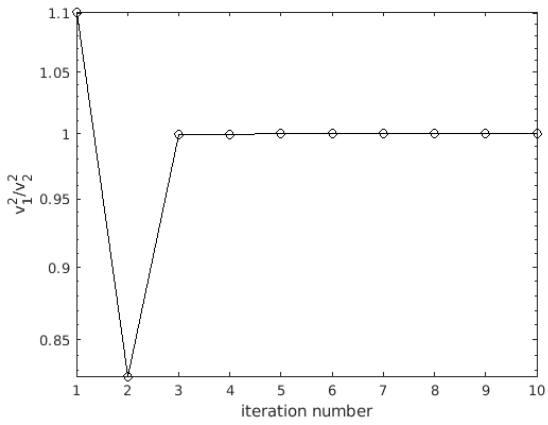


Figure 1.4: The ratio between v_1^2 and v_2^2 in Anderson Acceleration.

The success of Anderson Acceleration and related mixing schemes in improving the convergence of SCF iterations illustrates the remarkable effectiveness of such extrapolation-based

strategies. The relationship between Anderson Acceleration and Krylov subspace methods has been well documented. In exact arithmetic, AA can be interpreted as a variant of the Generalized Minimal Residual (GMRES) method [5, Section 6.5], and it belongs to the broader class of multisecant quasi-Newton schemes [6, 7, 8, 9]. Convergence analysis of Anderson Acceleration under contraction assumptions has been developed in [10], and subsequent studies have examined its depth-1 variant AA(1) [11, 12, 13], damping strategies [14, 15, 16], and convergence acceleration properties [11, 14, 17, 18, 19]. Despite these efforts, a complete understanding of when and why Anderson Acceleration improves convergence of general nonlinear problems remains an active topic of research.

In parallel, the Conjugate Residual method with Optimal trial vectors (CROP) [20, 21] has gained attention as an alternative acceleration strategy, particularly in electronic structure calculations and quantum chemistry. CROP generalizes the classical Conjugate Residual (CR) method [22] to nonlinear settings and often achieves performance comparable to or better than Anderson Acceleration, frequently with reduced memory requirements. These connections motivate a deeper investigation of the relationship between CROP and AA, aiming to determine whether they can be understood within a common analytical framework and whether CROP variants can inherit or extend the robustness of Anderson-type strategies.

A complementary research direction arises from *nonlinear Krylov subspace methods*, which generalize powerful linear iterative solvers to nonlinear systems. In particular, the nonlinear truncated GCR (nlTGCR) method [23] extends the classical GCR framework [24] to nonlinear equations and offers strong links to both Anderson Acceleration and multisecant methods. These developments raise a natural question: Can other well-known Krylov-type algorithms be systematically generalized to the nonlinear setting? This question forms one of the key motivations of the present work, which seeks to construct a unified and rigorous framework for nonlinear Krylov methods and to explore their theoretical and computational

relationships with Anderson and CROP-type acceleration schemes.

Collectively, these observations reveal several key open questions:

- What mechanisms allow CROP to achieve substantial speedups, and under what conditions does it reliably accelerate a fixed-point iteration?
- What is the precise relationship among Anderson Acceleration, CROP, and multisecant methods?
- Can a unified framework encompass fixed-point acceleration, quasi-Newton updates, and nonlinear Krylov subspace techniques?
- Can other well-known Krylov-type algorithms be systematically generalized to the nonlinear setting?
- How can these ideas be extended to modern large-scale and matrix-valued problems?

These questions form the central motivation of this thesis. By developing unified perspectives, rigorous convergence analysis, and practical algorithmic extensions, this work aims to advance both the theoretical foundations and the computational performance of acceleration methods and fast nonlinear solvers across a wide spectrum of scientific and engineering applications.

1.2 Objectives of the Thesis

Addressing the questions raised in Section 1.1 requires both theoretical investigation and practical algorithmic development. The overarching goal of this thesis is to bridge the gap between empirical success and analytical understanding of modern acceleration techniques

and nonlinear solvers. Building upon the classical foundations of fixed-point theory, Newton-type methods, and nonlinear Krylov subspace techniques, this work develops new theoretical results, algorithmic formulations, and computational tools that clarify the mechanisms underlying Anderson Acceleration and related approaches, contributing an important step toward a comprehensive understanding.

The emphasis lies on establishing rigorous convergence guarantees, revealing structural connections among different acceleration strategies, and improving robustness and scalability in large-scale scientific computing.

To realize these goals, the thesis introduces several original developments that address gaps in current theory and practice. The main objectives and contributions are outlined as follows:

1. **Unified framework connecting Anderson Acceleration and CROP.**
   A general framework is developed to reveal the structural relationship between Anderson Acceleration and the Conjugate Residual with Optimal trial vectors (CROP) method.

   - Establishes the theoretical connection between CROP and Anderson Acceleration in exact arithmetic.
   - Describes the connection of CROP with existing methods, such as Krylov methods and multisecant methods.
   - Provides convergence analysis for the CROP algorithm, theoretically and numerically.
   - Interprets both algorithms within a shared residual minimization perspective.
   - Introduces variants and extensions of the CROP algorithm.

2. **Perturbation analysis of Anderson-type acceleration methods.**

The sensitivity of Anderson Acceleration and its variants to deterministic and stochastic perturbations is systematically studied.

- Derives theoretical bounds for error propagation under function perturbations about variants of Anderson Acceleration, such as restarted and adaptive-depth Anderson Acceleration.
- Demonstrates, both analytically and numerically, that errors remain linearly bounded under suitable assumptions.
- Explores stochastic cases motivated by Monte Carlo–type noise, highlighting stability limits in practice.

3. **Nonlinear Krylov subspace acceleration and inexact Newton frameworks.**
   A general class of nonlinear GMRESR- and GCR-type algorithms is developed and analyzed as a bridge between Anderson-type and Newton-type methods.

   - Develops a unified framework for nonlinear Krylov methods derived from nested GCR-type linear solvers.
   - Provides convergence results without requiring an exact line search condition.
   - Extends the formulation to matrix-valued root-finding problems

   $$F(X) = 0, \qquad X \in \mathbb{R}^{n\times p},$$

     including orthogonality constraints.
   - Validates the proposed algorithms on large-scale nonlinear problems, demonstrating efficiency and robustness.

Together, these objectives aim to establish a unified theoretical and computational framework linking Anderson-type, multisecant, and Krylov-based acceleration methods. The expected

outcomes include both new theoretical insights and practical guidance for designing efficient and reliable solvers for large-scale nonlinear problems.

1.3 Specific Problems of Interest

The theoretical and algorithmic developments of this thesis are motivated by a range of nonlinear problems that can be expressed in fixed-point or root-finding form. While problems arising in physics, chemistry, and engineering are often defined in complex-valued spaces, this thesis focuses on the real-valued setting for clarity of presentation and analysis. Before introducing acceleration and Krylov-based techniques in Chapter 2, we summarize here several representative classes of problems to which the proposed methods apply. These examples also serve to illustrate how the abstract formulations introduced earlier arise naturally in scientific and engineering computations.

1.3.1 Principal Problem Formulation

At the foundation of numerous nonlinear solution methods lies the fixed-point formulation.

Problem 1.1. *Given a function $g : \mathbb{R}^n \to \mathbb{R}^n$, find $x \in \mathbb{R}^n$ such that*

$$x = g(x). \tag{1.1}$$

Equivalently, one can consider the associated *root-finding problem.*

Problem 1.2. *Given a function $f : \mathbb{R}^n \to \mathbb{R}^n$, find $x \in \mathbb{R}^n$ such that*

$$f(x) = 0. \tag{1.2}$$

Problem 1.1 and Problem 1.2 are directly related by setting $g(x) = x + \beta f(x)$ for any nonzero scalar β. Throughout this thesis, both formulations are used depending on the algorithmic setting. Moreover, root-finding problems arise naturally in the context of unconstrained optimization:

Problem 1.3. *Given a differentiable function $\phi : \mathbb{R}^n \to \mathbb{R}$, find $x \in \mathbb{R}^n$ such that*

$$\min_x \phi(x). \tag{1.3}$$

Indeed, the critical points of ϕ correspond to the roots of its gradient, i.e., $f(x) = \nabla\phi(x) = 0$. The most basic iterative method for solving (1.1) is the *fixed-point iteration*

$$x^{(k+1)} = g(x^{(k)}), \qquad k = 0, 1, 2, \ldots \tag{1.4}$$

While simple to implement, this approach often converges very slowly, motivating the development of multiple acceleration techniques such as Anderson Acceleration, CROP, and nonlinear Krylov subspace methods, the central focus of this thesis.

1.3.2 Linear Systems of Equations

A foundational example is the system of linear equations

$$Ax = b, \tag{1.5}$$

where $A \in \mathbb{R}^{n\times n}$ is nonsingular and $b \in \mathbb{R}^n$. It can be written equivalently as the root-finding problem $f(x) = b - Ax = 0$ or as the fixed-point problem $x = g(x)$ with

$$g(x) = (I - \beta A)x + \beta b, \qquad \beta \neq 0.$$

Linear systems serve as the prototype for understanding convergence behavior of iterative and acceleration schemes. They also provide the natural setting for Krylov subspace methods such as GMRES and Conjugate Residual, which later inspire their nonlinear counterparts analyzed in this thesis.

1.3.3 Nonlinear Eigenvalue Problems

A more challenging class of problems considered here is the *nonlinear eigenvalue problem*, i.e.,

$$T(\lambda)v = 0, \quad v \neq 0, \tag{1.6}$$

where $v \in \mathbb{R}^n$ is a vector and $T : \mathbb{R} \to \mathbb{R}^{n\times n}$ is a matrix-valued function that depends nonlinearly on the eigenvalue parameter λ. This problem can also be reformulated as the root-finding problem

$$F\left(\begin{bmatrix} v \\ \lambda \end{bmatrix}\right) = \begin{bmatrix} T(\lambda)v \\ c^T v - 1 \end{bmatrix} = 0,$$

where $c^T v = 1$ enforces normalization. Nonlinear eigenvalue problems arise in vibration analysis, delay-differential systems, and quantum mechanical models. They naturally fit within the root-finding framework (1.2), allowing the use of the nonlinear acceleration techniques developed in later chapters.

1.3.4 Nonlinear Eigenvector Problems

An even broader class is the *nonlinear eigenvector problem*: find $V \in \mathbb{R}^{n\times r}$ with orthonormal columns, i.e., $V^T V = I_r$, and a diagonal matrix $\Lambda \in \mathbb{R}^{r\times r}$ such that

$$H(V)V = G(V)V\Lambda, \tag{1.7}$$

where $H(V)$ and $G(V)$ are symmetric matrix-valued functions of V. Such problems occur in many scientific and engineering contexts, including computational chemistry [25], linear discriminant analysis (LDA) [26, 27], and balanced graph cuts [28].

Two important special cases are:

- $G(V) = I$, where (1.7) reduces to

$$H(V)V = V\Lambda, \tag{1.8}$$

  which corresponds to the nonlinear eigenvector problem frequently encountered in electronic structure calculations and density functional theory (DFT) [29, 30].

- $r = 1$ and $V \equiv v$, where (1.7) yields

$$H(v)v = \lambda G(v)v, \tag{1.9}$$

  a nonlinear eigenpair problem widely studied in numerical linear algebra [31, 32].

The discretized Kohn-Sham equations, a cornerstone of DFT, are a prominent instance of (1.8), where the Hamiltonian [33] has the form

$$H(V) = -\tfrac{1}{2}\Delta + V_{\text{ion}} + V_{\text{elec}}(v_1, \ldots, v_r) + V_{\text{xc}},$$

where V_{ion}, V_{elec} , V_{xc} denote the ionic potential, the electron-electron interaction, and the exchange correlation potential, respectively. In such settings, convergence of the underlying SCF iteration can be extremely slow, making these problems an ideal benchmark for acceleration techniques such as Anderson Acceleration and its variants.

In summary, the nonlinear problems considered in this thesis, ranging from fixed-point equa-

tions and linear systems to nonlinear eigenvalue and eigenvector formulations, share a common structure that makes them amenable to iterative and accelerated solution methods. The subsequent chapters build upon these formulations to develop, analyze, and test new acceleration frameworks capable of unifying and extending the existing methods.

1.4 Notation

For simplicity, all scalars, vectors, and matrices are assumed to belong to real vector spaces. Matrices are denoted by uppercase Roman or Greek letters, while vectors and scalars are denoted by lowercase Roman or Greek letters. The Euclidean inner product is denoted by $\langle \cdot, \cdot \rangle$, and the associated 2-norm by $\| \cdot \|_p$. More generally, $\| \cdot \|_p$ denotes the p-norm when needed. Unless specified otherwise, $\| \cdot \|$ denotes the Euclidean ($p = 2$) norm. By $\mathcal{R}(V)$ and $\mathcal{N}(V)$, we denote the range and Null space of a rectangular matrix $V \in \mathbb{R}^{n,j}$, $j \leq n$, respectively. For a (nonlinear) function $f : \mathbb{R}^n \to \mathbb{R}^n$, its Jacobian at a point $x \in \mathbb{R}^n$ is denoted by $J_f(x) \in \mathbb{R}^{n \times n}$, and its Hessian by $H_f(x) \in \mathbb{R}^{n \times n \times n}$ when needed. The identity function is denoted by id, i.e., $\mathrm{id}(x) = x$ for all $x \in \mathbb{R}^n$.

In algorithms, iterative quantities are indexed using superscripts: $x^{(j)}$ denotes the value of x at iteration j, and $x_i^{(j)}$ denotes the ith component of that vector. Residuals and function values follow the same convention, e.g., $r^{(j)}$, $f^{(j)}$. Throughout this thesis, the notation $\texttt{GMRES}(A, b, m)$ refers to the application of the Generalized Minimal Residual (GMRES) method to the linear system $Ax = b$ for m steps, returning the approximate solution $x^{(m)}$. Unless a better initial guess is available, GMRES is assumed to be initialized with the zero vector.

1.5 Content of the Thesis

This thesis is organized into six chapters that build progressively from foundational ideas to theoretical analysis and algorithmic developments. The structure reflects the logical flow of the research, from identifying the key challenges in nonlinear solvers to proposing, analyzing, and testing new acceleration frameworks.

Chapter 2 establishes the theoretical and algorithmic background necessary for the subsequent developments. It introduces classical and modern iterative techniques, including multisecant methods, extrapolation-based acceleration schemes, and Anderson-type methods. This chapter also provides the mathematical preliminaries used throughout the thesis.

Chapter 3 investigates the connection between the Conjugate Residual algorithm with Optimal trial vectors (CROP) and other well-known iterative methods. It establishes the equivalence between CROP and Anderson Acceleration in exact arithmetic, explores their relationships with Krylov subspace and multisecant methods, and presents convergence analysis for both linear and nonlinear problems. Several new variants, including CROP-Anderson and rCROP, are proposed and evaluated.

Chapter 4 examines the sensitivity of Anderson-type acceleration methods to deterministic and stochastic perturbations. In particular, it studies the restarted and adaptive-depth Anderson variants discussed in Section 2.6 and analyzes how perturbations in function evaluations and residual computations affect convergence. Theoretical insights and numerical experiments reveal the impact of noise, rounding errors, and model inaccuracies on acceleration performance.

Chapter 5 develops a unified framework for nonlinear Krylov methods derived from nested GCR-type linear solvers. This framework systematically generalizes algorithms such as GMRESR, GCRO, and LGMRES to nonlinear problems, resulting in methods including nl-

GMRESR, nlGCRO, and nlLGMRES. Convergence results are established under relaxed assumptions, and the framework is further extended to nonlinear matrix equations and operator-based formulations.

Chapter 6 concludes the thesis by summarizing the main findings, highlighting the theoretical and computational contributions, and discussing promising directions for future research. In particular, it reflects on the connections among Anderson Acceleration, CROP, and nonlinear Krylov methods, and outlines how these insights can inform the design of next-generation solvers for large-scale nonlinear problems.

In summary, the thesis progresses from fundamental iterative schemes to advanced acceleration frameworks, contributing to the development of a coherent theory that unifies fixed-point acceleration, multisecant updates, and nonlinear Krylov methods. Together, these chapters provide both theoretical understanding and practical strategies for improving the efficiency and robustness of modern nonlinear solvers.

Chapter 2

Background

The efficient solution of nonlinear equations and fixed-point problems is one of the central topics in numerical analysis and scientific computing. Nonlinear systems of the form

$$f(x) = 0, \quad f : \mathbb{R}^n \to \mathbb{R}^n,$$

arise in a wide range of applications, including optimization, fluid dynamics, quantum chemistry, and machine learning. Although classical Newton-type methods remain among the most popular and powerful iterative schemes, their efficiency strongly depends on the availability and conditioning of the Jacobian matrix, as well as on the quality of the initial guess. When explicit Jacobians are difficult or expensive to compute, alternative Jacobian-free or secant-type approaches have proven invaluable.

In practice, nonlinear solvers must often deal with large-scale problems where full Jacobian evaluation is prohibitive. In such settings, researchers have developed a wide range of quasi-Newton and acceleration strategies that approximate or avoid the Jacobian entirely, while improving convergence beyond that of simple fixed-point iterations. These include classical secant and Broyden-type methods, vector extrapolation schemes such as MPE and RRE, and acceleration methods such as Anderson Acceleration and CROP.

This chapter reviews the theoretical and algorithmic background necessary for understanding the developments in later chapters. This chapter is organized as follows: Section 2.1

introduces the fixed-point iteration and discusses its convergence behavior. Section 2.2 presents Newton's method and its theoretical properties. Section 2.3 reviews quasi-Newton and multisecant methods. Section 2.4 discusses inexact Newton and Newton–Krylov methods. Section 2.5 provides an overview of Krylov subspace methods and their extensions. Section 2.6 concludes with extrapolation and acceleration techniques that provide the conceptual foundation for the new algorithms and theoretical results presented in the following chapters.

2.1 Fixed-Point Method

The most elementary approach to solving nonlinear equations is the fixed-point iteration. Although conceptually simple, it serves as a unifying framework for many more elaborate methods. By expressing a nonlinear problem $f(x) = 0$ as a fixed-point relation $x = g(x)$, the fixed-point method can be written as

$$x^{(k+1)} = g(x^{(k)}), \quad \text{for all} \quad k \in \mathbb{N}, \tag{2.1}$$

with an initial guess $x^{(0)}$. The convergence of the fixed-point iteration is governed by the contraction mapping principle known as the Banach Fixed-Point Theorem, i.e.,

Theorem 2.1. *Let C be a (nonempty) closed subset of a Banach space X, and let $g : C \to C$. If there exists a contraction constant $\theta \in [0, 1)$ such that*

$$\|g(x) - g(y)\| \leq \theta \|x - y\|, \qquad \textit{for all} \quad x, y \in C,$$

then g is a contraction mapping and has a unique fixed point $x^* \in C$ such that

$$\|g^n(x) - x^*\| \leq \frac{\theta^n}{1-\theta}\|g(x) - x_0\|.$$

A prominent example of a fixed-point method in applied mathematics and computational science is the *Self-Consistent Field (SCF)* iteration [3]. Due to its simplicity, the SCF method is widely used by domain specialist (chemists, material scientists and physicists) [29, 30, 34] to solve large-scale nonlinear eigenvector problems, in particular those of the form (1.8). In general, the SCF method is a simple fixed-point iteration method for the nonlinear problem $X = g(X)$, where $X \in \mathbb{R}^{n\times r}$, i.e., under certain assumptions on function g the solution X^* of the problem $X = g(X)$ is a result of the iterative procedure $X^{(k+1)} = g(X^{(k)})$. An SCF iteration method for solving problem (1.8) is presented in Algorithm 1. The SCF method is thus an application of the general fixed-point principle

$$V^{(k+1)} = g(V^{(k)}),$$

with g defined by the mapping between the current and next approximations of the potential or density. Note that in this context, the mapping g cannot generally be expressed in an explicit analytical form, which makes it difficult to apply classical fixed-point theorems directly to analyze convergence. Indeed, in many practical applications, including SCF iterations, g is defined implicitly through the solution of an auxiliary problem that depends on the current iterate, rather than as a closed-form function. Although the SCF iteration method is very popular, extensively used in practice and its convergence properties have already been investigated [4, 35, 36, 37, 38], there are still many open questions to be answered. In particular, it is well-known that even the simplest version of the SCF iteration method may fail to converge. Figures 1.1 and 1.2 in Section 1.1 illustrate such a case, where the charge

density oscillates without reaching self-consistency. For this reason, various modifications such as the level shifting [39] and the trust region SCF [33, 40] have been proposed to help with the convergence of the original SCF method. A number of heuristics such as charge mixing [41], modified Broyden's method [42], Anderson's method [1], the Direct Inversion of Iterative Subspace (DIIS) method [43, 44, 45] and the Periodic Pulay method [46] have also been used to solve these nonlinear eigenvector problems. From a mathematical standpoint, the SCF iteration provides the foundational framework upon which acceleration strategies are built. By viewing the SCF process as a nonlinear fixed-point problem, one can apply general-purpose accelerators to significantly enhance its convergence. In later sections (see Sections 2.3 to 2.6), we will explore how multisecant techniques and acceleration methods can be interpreted as extensions of the SCF approach. In summary, fixed-point iteration and its key application, the SCF method, establish the basis for many iterative schemes in nonlinear analysis. While simple to implement, their inherent slow convergence motivates the development of more advanced algorithms such as Newton's method and multisecant updates, which we discuss next.

2.2 Newton's Method

Although fixed-point iterations provide a general conceptual framework, their convergence is often slow unless the mapping $g(x)$ is well-conditioned, which means it behaves as a contraction with a small θ. Newton's method addresses this limitation by incorporating derivative information through the Jacobian matrix, achieving quadratic convergence near a solution. In the following section, we review the derivation of Newton's method and its computational challenges, particularly the cost of forming and inverting the Jacobian at each iteration.

The classical Newton's method [47] is an iterative root-finding algorithm used for continuously differentiable functions. Let $f : \mathbb{R}^n \to \mathbb{R}^n$ be continuously differentiable. Then near a root (or zero) of the function f, i.e., $x \in \mathbb{R}^n$ such that $f(x) = 0$, the function f can be approximated via the following iterative process. Since

$$f(x + \Delta x) \approx f(x) + J_f(x)\Delta x, \tag{2.2}$$

at the iterate $x^{(k+1)}$

$$f(x^{(k+1)}) \approx f(x^{(k)}) + J_f(x^{(k)})(x^{(k+1)} - x^{(k)}) = f(x^{(k)}) + J_f(x^{(k)})\Delta x^{(k)}. \tag{2.3}$$

Hence, near convergence, $f(x^{(k+1)}) \approx 0$, and the Newton step $\Delta x^{(k)}$ satisfies

$$J_f(x^{(k)})\Delta x^{(k)} = -f(x^{(k)}). \tag{2.4}$$

Therefore, new approximations are obtained by repeating iterations

$$x^{(k+1)} = x^{(k)} + \Delta x^{(k)} = x^{(k)} - J_f(x^{(k)})^{-1} f(x^{(k)}). \tag{2.5}$$

Note that (2.5) is also a fixed-point iteration. While the formulas above are mathematically precise, their numerical realizations need several adjustments. In general, computing the Jacobian in each iteration is not practical, and computing its inverse is also expensive. Therefore, there are many different ways to approximate it [48].

2.3 Quasi-Newton and Multisecant Methods

Because of the disadvantages of Newton's method, several alternative schemes, commonly called *quasi-Newton* methods [49], have been proposed over the years. In most of these mitigation schemes

(1) the exact Jacobian $J_f(x^{(k)})$ is replaced with an approximation $J^{(k)}$, and

(2) an approximation $J^{(k+1)}$ to $J_f(x^{(k+1)})$ can be obtained from $J^{(k)}$ by adding a low-rank matrix at each iteration step.

Therefore, a quasi-Newton iterate is updated via

$$x^{(k+1)} = x^{(k)} + \Delta x^{(k)}, \quad \text{where} \quad J^{(k)}\Delta x^{(k)} = -f(x^{(k)}). \tag{2.6}$$

In the paragraphs that follow, we will discuss Broyden's method [50] and Broyden's family [9] as an extension of this method.

Broyden's method is a widely used quasi-Newton scheme, in which the Jacobian approximation $J^{(k)}$ is updated using

$$J^{(k+1)} = J^{(k)} + (\Delta f^{(k)} - J^{(k)}\Delta x^{(k)})\frac{(\Delta x^{(k)})^T}{(\Delta x^{(k)})^T\Delta x^{(k)}}, \tag{2.7}$$

which corresponds to the solution of $\min \|J - J^{(k)}\|_F$ satisfying $J\Delta x^{(k)} = \Delta f^{(k)}$ [49].

Alternatively, if we take $G^{(k)}$ as the inverse of the approximate Jacobian $J^{(k)}$, $G^{(k)}$ can be updated using a formula similar to (2.7), that is,

$$G^{(k+1)} = G^{(k)} + (\Delta x^{(k)} - G^{(k)}\Delta f^{(k)})\frac{(\Delta f^{(k)})^T}{(\Delta f^{(k)})^T\Delta f^{(k)}}, \tag{2.8}$$

which corresponds to the solution of $\min \|G - G^{(k)}\|_F$ satisfying $G\Delta f^{(k)} = \Delta x^{(k)}$ [49]. The update formula (2.8) results in the so-called *"bad" Broyden's method*.

A different update formula for $G^{(k+1)}$ emerges by applying the Sherman-Morrison formula to (2.7), i.e.,

$$G^{(k+1)} = G^{(k)} + (\Delta x^{(k)} - G^{(k)}\Delta f^{(k)})\frac{(\Delta x^{(k)})^T G^{(k)}}{(\Delta x^{(k)})^T G^{(k)} \Delta f^{(k)}}, \tag{2.9}$$

which is called the *"good" Broyden's method*.

Other quasi-Newton methods, such as Powell Symmetric Broyden method (PSB) [51], Broyden-Fletcher-Goldfarb-Shanno (BFGS) method [52, 53, 54, 55], and Davidon-Fletcher-Powell (DFP) method [56], use updates similar to Broyden's, minimizing a Frobenius-norm change under specific conditions [49], and are locally q-superlinearly convergent under standard assumptions [57]. Multisecant methods further generalize quasi-Newton updates by using several recent iterates and function evaluations rather than just the last one. Multisecant extensions of Broyden, BFGS, and DFP are discussed in [58, 59], and the Broyden-like class was introduced in [9]. These approaches avoid explicit Jacobian computation and support limited-memory implementations. However, since these methods do not ensure that the approximate Newton step is a descent direction, they are often replaced by Newton-Krylov methods [47], which will be introduced in Section 2.4.

We introduce the generalized Broyden's method as an example of multisecant methods. Let

$$\mathscr{F}^{(k)} = [\Delta f^{(k-m)}, \ldots, \Delta f^{(k-1)}] \in \mathbb{R}^{n\times n} \quad \text{and} \quad \mathscr{X}^{(k)} = [\Delta x^{(k-m)}, \ldots, \Delta x^{(k-1)}] \in \mathbb{R}^{n\times m}. \tag{2.10}$$

The *generalized "bad" Broyden's method* is written as

$$G^{(k)} = G^{(k-m)} + (\mathscr{X}^{(k)} - G^{(k-m)}\mathscr{F}^{(k)})((\mathscr{F}^{(k)})^T\mathscr{F}^{(k)})^{-1}(\mathscr{F}^{(k)})^T, \tag{2.11}$$

which minimizes $\|G^{(k)} - G^{(k-m)}\|_F$ subject to $G^{(k)}\mathscr{F}^{(k)} = \mathscr{X}^{(k)}$. The update formula for the iterates $x^{(k)}$ in the *generalized "bad" Broyden's method* is given as

$$x^{(k+1)} = x^{(k)} - G^{(k)} f^{(k)} = x^{(k)} - G^{(k-m)} f^{(k)} - (\mathscr{X}^{(k)} - G^{(k-m)}\mathscr{F}^{(k)})((\mathscr{F}^{(k)})^T \mathscr{F}^{(k)})^{-1}(\mathscr{F}^{(k)})^T f^{(k)}. \tag{2.12}$$

Analogously, we can also formulate *generalized "good" Broyden's method* with approximate inverse Jacobian updates

$$G^{(k)} = G^{(k-m)} + (\mathscr{X}^{(k)} - G^{(k-m)}\mathscr{F}^{(k)})((\mathscr{X}^{(k)})^T G^{(k-m)} \mathscr{F}^{(k)})^{-1}(\mathscr{X}^{(k)})^T G^{(k-m)}. \tag{2.13}$$

Notice that (2.13) is the inverse of

$$J^{(k)} = J^{(k-m)} + (\mathscr{F}^{(k)} - J^{(k-m)}\mathscr{X}^{(k)})((\mathscr{X}^{(k)})^T \mathscr{X}^{(k)})^{-1}(\mathscr{X}^{(k)})^T, \tag{2.14}$$

which minimizes $\|G^{(k)} - G^{(k-m)}\|_F$ subject to $J^{(k)}\mathscr{X}^{(k)} = \mathscr{F}^{(k)}$. The generalized Broyden's method closely relates to Anderson Acceleration and other acceleration methods; see [9] and Section 3.2.2.

2.4 Newton-Krylov and Inexact Newton's Methods

In the previous sections, we discussed classical and quasi-Newton methods, which approximate or update the Jacobian matrix to reduce the computational cost of Newton's method. An alternative strategy is to avoid the explicit formation or storage of the Jacobian altogether. Instead, the Newton correction equation

$$J_f(x^{(k)})\,\Delta x^{(k)} = -f(x^{(k)}), \tag{2.15}$$

can be solved iteratively using *Krylov subspace methods*. This approach gives rise to the class of *Newton-Krylov methods* [60], which are among the most efficient and scalable solvers for large-scale nonlinear problems.

Unlike quasi-Newton approaches, which construct approximate Jacobians $J^{(k)}$ from previous iterates, Newton-Krylov methods treat (2.15) as a linear subproblem to be solved approximately. The key idea is that the Krylov subspace can be built using only matrix-vector products of the form $J_f(x^{(k)})v$ without ever forming $J_f(x^{(k)})$ explicitly. When the Jacobian-vector product is unavailable, it can be approximated by finite differences:

$$J_f(x^{(k)})v \approx \frac{f(x^{(k)} + \epsilon v) - f(x^{(k)})}{\epsilon}, \tag{2.16}$$

where ϵ is a small parameter chosen to balance truncation and round-off errors. The Krylov basis vectors can then be constructed recursively from the residual $r^{(0)}$ as

$$J_f(x^{(k)})^{n+1} r^{(0)} \approx \frac{f(x^{(k)} + \epsilon J_f(x^{(k)})^n r^{(0)}) - f(x^{(k)})}{\epsilon}. \tag{2.17}$$

Through this process, the update $\Delta x^{(k)}$ is computed as a linear combination of these Krylov vectors, yielding an approximate solution to (2.15).

Formally, Newton-Krylov methods can be viewed as members of the broader family of *inexact Newton methods* [61], in which the linear system in each Newton step is solved only approximately:

$$x^{(k+1)} = x^{(k)} + \Delta x^{(k)}, \quad \text{where} \quad J_f(x^{(k)})\,\Delta x^{(k)} = -f(x^{(k)}) + t^{(k)}, \tag{2.18}$$

with the residual error $t^{(k)}$ satisfying

$$\frac{\|t^{(k)}\|_2}{\|f(x^{(k)})\|_2} \leq \eta_k. \tag{2.19}$$

Here, $\eta_k \in [0, 1)$ controls the tolerance of the inner solve and can be adapted dynamically to balance computational cost and convergence speed, and $t^{(k)} \in \mathbb{R}^n$ represents the resulting inexactness. If the tolerance η_k is small, the resulting step is close to that of the exact Newton method; larger values of η_k yield cheaper but less accurate steps. This flexibility enables Newton-Krylov methods to handle very large nonlinear systems efficiently, even when forming or factorizing the Jacobian is impractical.

In practice, the inner linear system (2.15) is often solved using Krylov methods such as the Generalized Minimal Residual Method (GMRES), the Generalized Conjugate Residual Method (GCR), or their flexible and restarted variants. Since these methods only require Jacobian-vector products, they enable *Jacobian-free* implementations, often referred to as *Jacobian-Free Newton-Krylov (JFNK)* schemes [60]. These algorithms combine the global convergence properties of Newton's method with the efficiency of Krylov subspace solvers, making them widely used in large-scale simulations and nonlinear partial differential equation solvers.

The next section provides a concise overview of Krylov subspace methods for linear systems, highlighting their residual-minimization properties and algorithmic structure. This background will be essential for understanding both the Newton-Krylov framework discussed here and the nonlinear Krylov extensions developed in Chapter 5.

2.5 Krylov Subspace Methods

Krylov subspace methods are among the most powerful and widely used classes of iterative techniques for solving large-scale linear systems and eigenvalue problems. They play a central role in scientific computing, numerical linear algebra, and are foundational building blocks for many modern nonlinear solvers.

Consider the linear system (1.5) with $A \in \mathbb{R}^{n\times n}$ and $b \in \mathbb{R}^n$. In many practical applications, the matrix A is large and sparse, making direct solvers computationally infeasible. Krylov subspace methods construct a sequence of approximations $x^{(j)}$, i.e.,

$$x^{(j+1)} \in x^{(0)} + K_{j+1}(A, r^{(0)}), \tag{2.20}$$

where $x^{(0)}$ is an initial guess and $r^{(0)} = b - Ax^{(0)}$ is the corresponding initial residual, and

$$K_j(A, r^{(0)}) := \text{span}\{r^{(0)}, Ar^{(0)}, A^2r^{(0)}, \ldots, A^{j-1}r^{(0)}\} \tag{2.21}$$

is the Krylov subspace of order j. It is well-known [5, §6.2] that for a nonsingular matrix A, the Krylov subspace $K_n(A, r^{(0)})$ contains the exact solution of (1.5). In practice, however, one seeks an approximate solution $x^{(j)}$ with $j \ll n$ that achieves the desired accuracy. Two main principles guide the design of Krylov subspace methods: orthogonality and minimization. In orthogonality-based approaches, one seeks an approximation $x^{(j)}$ whose residual $r^{(j)} = b - Ax^{(j)}$ is orthogonal to a chosen subspace (test space), usually the Krylov subspace $K_j(A, r^{(0)})$. Prominent examples include the Conjugate Gradient (CG) method [62], Conjugate Residual (CR) [22], Generalized Conjugate Residual (GCR) [24], Orthomin [63], and the Full Orthogonalization Method (FOM), and Lanczos method [64]. In minimization-based approaches, one instead seeks an approximation that minimizes the residual norm over

$K_j(A, r^{(0)})$, i.e.,

$$x^{(j+1)} = x^{(0)} + p^{(j)}, \qquad p^{(j)} = \arg \min_{p \in K_{j+1}(A, r^{(0)})} \|b - A(x^{(0)} + p)\|_2.$$

This formulation leads to the Minimum Residual (MINRES) method [65] for symmetric matrices and the Generalized Minimum Residual (GMRES) method [66] for general non-symmetric systems. GMRES is particularly notable due to its strong theoretical properties and its conceptual connection to several acceleration schemes discussed in this thesis. The methods mentioned above fall under the general framework of (Truncated) Petrov-Galerkin Krylov ((T)PGK) methods [67], where well-known equivalences hold in exact arithmetic, e.g., GMRES and GCR produce identical iterates, and Orthomin corresponds to a truncated version of GCR. Moreover, GMRES has been shown to be closely related to the rank-one acceleration method of Eirola and Nevanlinna [68] and can be interpreted as a Broyden-type method when applied to linear problems [69]. These connections are essential for understanding how Krylov methods naturally generalize many of the acceleration concepts introduced in later chapters. Preconditioning plays a vital role in the efficiency of Krylov subspace methods. Traditional preconditioners, such as incomplete factorizations [5, §10], are constant throughout the iteration. More flexible strategies employ variable preconditioners that adapt dynamically, for example by using a few inner iterations of another solver [5, §10.2]. This idea led to the development of Flexible GMRES [70] and recursive variants such as GMRESR [71], which use a Krylov solver as a variable preconditioner. Subsequent developments introduced inner orthogonalization (GCRO) [72], optimal truncation (GCROT) [73], and the Loose GMRES (LGMRES) method [74], as well as their flexible extensions [75]. These methods form the foundation for the nonlinear Krylov generalizations discussed in Chapter 5. Krylov subspace techniques have also been extended to block and

global variants for systems with multiple right-hand sides,

$$Ax_i = b_i, \quad i = 1, \ldots, p \quad \Leftrightarrow \quad AX = B, \tag{2.22}$$

where $X = [x_1, \ldots, x_p]$ and $B = [b_1, \ldots, b_p]$. In this case, one constructs block Krylov subspaces

$$\mathcal{K}_j^{\square}(A, R^{(0)}) = \text{blockspan}\{R^{(0)}, AR^{(0)}, A^2R^{(0)}, \ldots, A^{j-1}R^{(0)}\},$$

using block-Arnoldi or block-Lanczos algorithms [5, 76, 77]. For more general linear operator equations

$$\mathcal{A}(X) = B, \qquad X, B \in \mathbb{R}^{n\times p}, \quad \mathcal{A} : \mathbb{R}^{n\times p} \to \mathbb{R}^{n\times p} \text{ linear},$$

global Krylov subspace methods were introduced in [78], initially for large sparse Sylvester equations. Global variants of GCRO, Flexible GMRES, the Quasi-Minimal Residual (QMR), and Least Squares (LSQR) methods have since been developed [79, 80, 81, 82]. Such approaches have proven highly effective in modern matrix-free contexts, including Newton-Krylov frameworks [83]. Krylov subspace concepts also underpin many iterative algorithms for linear eigenvalue problems of the form

$$Ax = \lambda x, \tag{2.23}$$

and their generalized version

$$Ax = \lambda Bx, \tag{2.24}$$

where $A, B \in \mathbb{R}^{n\times n}$. A wide range of iterative methods has been developed for solving eigenvalue problems. Classical Krylov methods, such as the Lanczos and Arnoldi methods, have long been standard for computing extremal eigenpairs but often require factorizations or suffer from storage limitations in large problems. Preconditioned Krylov subspace

methods, including conjugate gradient variants, provide effective alternatives by improving convergence without direct factorization. Among these, the Locally Optimal Block Preconditioned Conjugate Gradient (LOBPCG) method, proposed by Knyazev in the 1990s, has become a popular choice for symmetric definite eigenvalue problems. LOBPCG is matrix-free and factorization-free, while still benefiting from preconditioning to accelerate convergence. Its block formulation enables the simultaneous computation of multiple eigenpairs through Rayleigh-Ritz projections, making it an efficient alternative to Lanczos-type solvers for large-scale applications.

2.6 Extrapolation and Acceleration Methods

The methods introduced so far primarily focus on constructing better local models of the nonlinear system, either through Jacobian evaluation or secant approximations. However, an alternative path to faster convergence lies in *using information from past iterates more effectively*. Extrapolation and acceleration methods pursue this direction by combining previous iterations and residuals to predict improved approximations, often without explicit Jacobian information.

As we have seen in Chapter 1, simple iterative methods often converge slowly or fail to converge. The problem of slow (or no) convergence of a sequence of iterates has been extensively studied by researchers since the early 20th century. Aitken's delta-squared process was introduced in 1926 [84] for nonlinear sequences, and since then, people have been investigating various extrapolation and convergence acceleration methods, with Shanks transformation [85, 86] providing one of the most important and fundamental ideas. Introduced as a generalization of Aitken's delta-squared process, it laid the foundations for many acceleration schemes, including the ε-algorithm. Some notable acceleration methods include

the ε-algorithm, which contains the scalar ε-algorithm (SEA) [87], the vector ε-algorithm (VEA) [88], the topological ε-algorithm (TEA) [89], simplified TEA (STEA) [90]. Polynomial methods, such as minimal polynomial extrapolation (MPE) [91], reduced rank extrapolation (RRE) [92], modified minimal polynomial extrapolation (MMPE) [93], are also widely used. For further reading about extrapolation and acceleration methods, see [86, 94, 95, 96, 97].

In this section, we collect some background information on the Anderson Acceleration [1, 2] (also referred to as Pulay mixing [43, 44] in computational chemistry) and the Conjugate Residual algorithm with OPtimal trial vectors (CROP) [20, 21] methods. Beyond classical extrapolation, *acceleration methods* such as Anderson Acceleration and its variants have become central tools in solving large-scale nonlinear problems. These techniques reinterpret extrapolation in an optimization framework, constructing new iterates as affine combinations of previous residuals. The following section provides an overview of these nonlinear acceleration methods, their theoretical underpinnings, and their relationships with the multisecant and quasi-Newton schemes introduced earlier.

2.6.1 Anderson Acceleration

The Anderson Acceleration (AA) method has a long history in the mathematics literature, which goes back to Anderson's 1965 seminal paper [1]. Over the years, the method has been successfully applied to many challenging problems [98, 99, 100, 101]. An independent line of research on accelerating convergence of nonlinear solvers established by physicists and chemists has led to the development of techniques such as Pulay mixing [43, 44], also known as the Direct Inversion of the Iterative Subspace (DIIS) algorithm, which is instrumental in accelerating the self-consistent field iteration method in electronic structure calculations [102]. It is well known that, in exact arithmetic, the Anderson Acceleration method

has connections with the Generalized Minimal Residual Method (GMRES) algorithm [5, Section 6.5] and can be categorized as a multisecant method [6, 7, 8, 9]. The first convergence theory for Anderson Acceleration, under the assumption of a contraction mapping, appears in [10]. Various different proofs of convergence for Anderson Acceleration under different assumptions have been published in the past ten years [11, 17, 18]. The convergence of Anderson(1), a method of particular interest for many researchers, is discussed separately in [11, 12, 13]. Several other variants of Anderson Acceleration have been explored in [16, 17, 103, 104, 105]. The acceleration properties of Anderson Acceleration have been theoretically established in [11, 14, 17, 18, 19]. Additionally, discussions on the depth parameter can be found in [11, 106], while the damping parameter is examined in [14, 15, 16]. We further refer readers to [86, 94, 95, 96, 97] and references therein for detailed and more comprehensive presentations of the history, theoretical and practical results on acceleration methods, and their applications.

Consider the nonlinear fixed-point problem (1.1), whose simple iteration $x^{(k+1)} = g(x^{(k)})$ may converge slowly or fail to converge. Anderson Acceleration generates a sequence of Anderson iterates designed to accelerate this basic fixed-point iteration by forming optimized linear combinations of previous iterates and residuals. Given Anderson iterates $x_A^{(k)}, k = 0, 1, \ldots$ and corresponding residual (error) vectors, e.g., $f_A^{(k)} := g(x_A^{(k)}) - x_A^{(k)}$, consider weighted averages of the prior iterates, i.e.,

$$\bar{x}_A^{(k)} := \sum_{i=0}^{m_A^{(k)}} \alpha_{A,i}^{(k)} x_A^{(k-m_A^{(k)}+i)} \quad \text{and} \quad \bar{f}_A^{(k)} := \sum_{i=0}^{m_A^{(k)}} \alpha_{A,i}^{(k)} f_A^{(k-m_A^{(k)}+i)}, \tag{2.25}$$

with weights $\alpha_{A,0}^{(k)}, \ldots, \alpha_{A,m_A^{(k)}}^{(k)} \in \mathbb{R}$ satisfying $\sum_{i=0}^{m_A^{(k)}} \alpha_{A,i}^{(k)} = 1$, a fixed depth (history or window size) parameter m and a truncation parameter $m_A^{(k)} := \min\{m, k\}$. Note that (2.25) can be

written in the equivalent matrix form,

$$\bar{x}_A^{(k)} := X_A^{(k)} \alpha_A^{(k)} \quad \text{and} \quad \bar{f}_A^{(k)} := F_A^{(k)} \alpha_A^{(k)}, \tag{2.26}$$

with $\mathbb{R}^{n \times (m_A^{(k)}+1)}$ matrices $X_A^{(k)} = \left[x_A^{(k-m_A^{(k)})}, \ldots, x_A^{(k)}\right]$, $F_A^{(k)} = \left[f_A^{(k-m_A^{(k)})}, \ldots, f_A^{(k)}\right]$, and coefficient vector $\alpha_A^{(k)} = [\alpha_{A,0}^{(k)}, \ldots, \alpha_{A,m_A^{(k)}}^{(k)}]^T \in \mathbb{R}^{m_A^{(k)}+1}, \sum_{i=0}^{m_A^{(k)}} \alpha_{A,i}^{(k)} = 1$. Anderson Acceleration achieves a faster convergence than a simple fixed-point iteration by using the past information to generate new iterates as linear combinations of previous $m_A^{(k)}$ iterates [6, 43, 44], i.e.,

$$\begin{aligned} x_A^{(k+1)} &= \bar{x}_A^{(k)} + \beta^{(k)} \bar{f}_A^{(k)} \\ &= (1-\beta^{(k)}) \sum_{i=0}^{m_A^{(k)}} \alpha_{A,i}^{(k)} x_A^{(k-m_A^{(k)}+i)} + \beta^{(k)} \sum_{i=0}^{m_A^{(k)}} \alpha_{A,i}^{(k)} g(x^{(k-m_A^{(k)}+i)}), \end{aligned} \tag{2.27}$$

with given relaxation (or damping) parameters $\beta^{(k)} \in \mathbb{R}^+$ and mixing coefficients $\alpha_{A,i}^{(k)} \in \mathbb{R}$, $i = 0, \ldots, m_A^{(k)}$ selected to minimize the linearized residual (error) of a new iterate within an affine space $\text{Aff}\{f_A^{(k-m_A^{(k)})}, \ldots, f_A^{(k)}\}$, i.e., obtained as a solution of the least-squares problem

$$\min_{\alpha \in \mathbb{R}^{m_A^{(k)}+1}} \|F_A^{(k)} \alpha\|_2^2 = \min_{\alpha_0, \ldots, \alpha_{m_A^{(k)}}} \left\| \sum_{i=0}^{m_A^{(k)}} \alpha_i f_A^{(k-m_A^{(k)}+i)} \right\|_2^2 \quad \text{such that} \quad \sum_{i=0}^{m_A^{(k)}} \alpha_i = 1. \tag{2.28}$$

Note that in the case of $\beta^{(k)} = 1$ a general formulation (2.27) introduced in the original work of Anderson [1, 2] reduces to the Pulay mixing [43, 44], i.e.,

$$x_A^{(k+1)} = \sum_{i=0}^{m_A^{(k)}} \alpha_{A,i}^{(k)} g(x_A^{(k-m_A^{(k)}+i)}). \tag{2.29}$$

The fixed depth Anderson Acceleration method can be summarized in Algorithm 2 and is often denoted as the Anderson(m) method.

Algorithm 2 Anderson Acceleration Method (of fixed depth m)

Input: Initial Anderson iterate $x_A^{(0)}$, a fixed depth $m \geq 1$ and a fixed damping parameter β

1: Compute $x_A^{(1)} = g(x_A^{(0)})$

2: **for** $k = 1, 2, \ldots$ until convergence **do**

3: Set truncation and relaxation parameters $m_A^{(k)}$, $\beta^{(k)}$

`/* e.g.` $m_A^{(k)} = \min\{k, m\}$, $\beta^{(k)} = \beta$ `*/`

4: Set Anderson residuals $F_A^{(k)} = [f_A^{(k-m_A^{(k)})}, \ldots, f_A^{(k)}]$, with

$f_A^{(i)} := f(x_A^{(i)}) = g(x_A^{(i)}) - x_A^{(i)}$

5: Determine mixing coefficients, i.e., $\alpha_A^{(k)} := [\alpha_{A,0}^{(k)}, \ldots, \alpha_{A,m_A^{(k)}}^{(k)}]^T$ that solves Problem (2.28)

6: Set $x_A^{(k+1)} = (1 - \beta^{(k)}) \sum_{i=0}^{m_A^{(k)}} \alpha_{A,i}^{(k)} x_A^{(k-m_A^{(k)}+i)} + \beta^{(k)} \sum_{i=0}^{m_A^{(k)}} \alpha_{A,i}^{(k)} g(x_A^{(k-m_A^{(k)}+i)})$

7: **end for**

Output: $x_A^{(k)}$ that solves $f(x) = 0$.

Note that in what follows we consider the case of $\beta = 1$, since $\beta \neq 1$ can be reduced to the latter by setting $f_\beta(x) = \beta f(x)$ and $g_\beta(x) = x + \beta f(x)$.

For the classical Anderson Acceleration method, obtaining mixing coefficients $\alpha_0^{(k)}, \ldots, \alpha_{m^{(k)}}^{(k)}$ requires solving constrained least-squares problems, i.e., (2.28), of the general form

$$\min_{\alpha_0^{(k)}, \ldots, \alpha_{m^{(k)}}^{(k)}} \left\| \sum_{i=0}^{m^{(k)}} \alpha_i^{(k)} f^{(k-m^{(k)}+i)} \right\|_2^2 \quad \text{such that} \quad \sum_{i=0}^{m^{(k)}} \alpha_i^{(k)} = 1, \tag{2.30}$$

defined within the affine subspace $\mathrm{Aff}\{f^{(k-m^{(k)})}, \ldots, f^{(k)}\}$. In general, (2.30) can be solved by the method of Lagrange multipliers [43]. By transforming the barycentric coordinates $\alpha^{(k)}$ into the affine coordinates $\gamma^{(k)}$ relative to $f^{(k)}$, (2.30) becomes

$$\min_{\gamma_1^{(k)},\dots,\gamma_{m^{(k)}}^{(k)}} \left\| f^{(k)} - \sum_{i=1}^{m^{(k)}} \gamma_i^{(k)} \Delta f^{(k-m^{(k)}+i)} \right\|_2^2, \tag{2.31}$$

where $\Delta f^{(i)} = f^{(i+1)} - f^{(i)}$. Also, $\alpha^{(k)}$ and $\gamma^{(k)}$ can be transformed, i.e., $\alpha_0^{(k)} = \gamma_1^{(k)}$, $\alpha_i^{(k)} = \gamma_{i+1}^{(k)} - \gamma_i^{(k)}, i = 1, \dots, m^{(k)} - 1$, $\alpha_{m^{(k)}}^{(k)} = 1 - \gamma_{m^{(k)}}^{(k)}$. Now, (2.31) can be solved using the normal equations. Let $\mathscr{F}^{(k)} := \left[\Delta f^{(k-m^{(k)})}, \dots, \Delta f^{(k-1)}\right] \in \mathbb{R}^{n \times m^{(k)}}$. Then, $\bar{f}^{(k)} = f^{(k)} - \mathscr{F}^{(k)}\gamma^{(k)}$ and the solution of (2.31) is given as

$$\gamma^{(k)} = \left[(\mathscr{F}^{(k)})^T \mathscr{F}^{(k)} \right]^{-1} (\mathscr{F}^{(k)})^T f_k. \tag{2.32}$$

Using (2.31) and (2.32) we can write the iterates of Anderson Acceleration as

$$x_A^{(k+1)} = x_A^{(k)} + \beta f_A^{(k)} - (\mathscr{X}_A^{(k)} + \beta \mathscr{F}_A^{(k)})[(\mathscr{F}_A^{(k)})^T \mathscr{F}_A^{(k)}]^{-1} (\mathscr{F}_A^{(k)})^T f_A^{(k)}, \tag{2.33}$$

where

$$\mathscr{X}_A^{(k)} = \left[\Delta x_A^{(k-m_A^{(k)})}, \dots, \Delta x_A^{(k-1)}\right], \quad \mathscr{F}_A^{(k)} = \left[\Delta f_A^{(k-m_A^{(k)})}, \dots, \Delta f_A^{(k-1)}\right].$$

In a practical implementation, the basis $\Delta f^{(k)}$ is often orthogonalized using a QR factorization, which also enables the least-squares problem to use the information from the previous iteration steps [6, 107]. In Algorithm 2, the least-squares problem (line 5) is solved using QR factorization; however, if the least-squares problem is small, an explicit pseudoinverse formulation of (2.33) is used instead. It is well-known that a good choice of the damping parameter $\beta^{(k)}$ significantly influences the convergence of Anderson Acceleration. In the case of fixed damping, i.e., $\beta^{(k)} = \beta$, if $\beta \neq 1$, the update formula (2.29) has the form

$$x_A^{(k+1)} = (1-\beta) \sum_{i=0}^{m_A^{(k)}} \alpha_i^{(k)} x_A^{(k-m_A^{(k)}+i)} + \beta \sum_{i=0}^{m_A^{(k)}} \alpha_i^{(k)} g(x_A^{(k-m_A^{(k)}+i)}).$$

Defining $g_\beta(x) := (1-\beta)x + \beta g(x) = x + \beta(g(x) - x) = x + \beta f(x)$ yields an equivalent

formulation

$$x_A^{(k+1)} = \sum_{i=0}^{m_A^{(k)}} \alpha_i^{(k)} g_\beta(x_{k-m_A^{(k)}+i}),$$

which has the same form as the iterates in Algorithm 2. Hence, Anderson Acceleration with a fixed depth parameter β can be regarded as running Algorithm 2 with a new fixed-point iteration function g_β. It is worth mentioning that the corresponding residual (error) $f_\beta(x) = g_\beta(x) - x = \beta(g(x) - x) = \beta f(x)$, although different than $f(x)$, has the same zeros as $f(x)$. More details on damping parameters can be found in [15, 18, 108, 109].

The subspace truncation parameter $m^{(k)} \geq 1$ ($m_A^{(k)}$ of Anderson Acceleration) determines the dimension of the search space for the next trial vector $x_A^{(k+1)}$, e.g., the size of the least-squares problem (2.30). Hence, the affine subspace in iteration step k involves $m^{(k)} + 1$ vectors. Obviously, $m^{(k)} = 0$ corresponds to the fixed-point iteration method. The proper choice of $m^{(k)}$ is very important, as it affects the convergence speed, time and overall complexity of these methods. For the fixed depth methods, a fixed parameter m is set before the iteration starts and the truncation parameter is chosen at each iteration step k to be $m^{(k)} = \min\{k, m\}$. For further discussions regarding various choices of $m^{(k)}$, see [11, 12]. Restarting and adaptivity are used to maintain the dimension of the subspace by choosing parameter $m^{(k)}$ in each step [17]. Also, some techniques like filtering [15] do not use the most recent $m^{(k)}$ trial vectors and residuals. However, they usually need to store a list of trial vectors and residuals of size $m^{(k)}$.

In addition to the classical Anderson Acceleration scheme, several variant forms have been developed to enhance robustness and adaptivity. Two notable examples are Restarted Anderson Acceleration and Adaptive Anderson Acceleration, both introduced in [17]. Assuming $m^{(k)} \geq 1$ at iteration k, the Restarted Anderson Acceleration method, Algorithm 3, forms

the residual differences

$$s^{(k-m^{(k)}+i)} := r^{(k-m^{(k)}+i)} - r^{(k-m^{(k)})}, \quad i = 1, \ldots, m^{(k)}.$$

A restart is triggered, i.e., the method sets $m^{(k+1)} = 0$, whenever

$$\tau \|s^{(k+1)}\|_2 > \|(\mathrm{id} - \Pi_k) s^{(k+1)}\|_2,$$

where Π_k denotes the orthogonal projection onto $span\{s^{(k-m^{(k)}+1)}, \ldots, s^{(k)}\}$, and τ is a small positive parameter. This mechanism detects when newly generated information is nearly linearly dependent on the existing subspace and resets the memory to maintain numerical stability. Another important variant is Adaptive Anderson Acceleration [17], see Algorithm 4. In each iteration, $m^{(k+1)}$ is chosen as the largest integer $m \leq m^{(k)} + 1$ such that

$$\delta \|r^{(i)}\|_2 < \|r^{(k+1)}\|_2,$$

holds for all indices i satisfying $k + 1 - m \leq i \leq k$, where δ is a small positive parameter. This adaptive rule adjusts the memory length dynamically based on the behavior of recent residuals, allowing the algorithm to respond automatically to changes in local nonlinear behavior. Together, these variants illustrate how memory management strategies can significantly influence the performance and stability of Anderson-type acceleration methods.

Algorithm 3 Restarted Anderson Acceleration

Input: $x^{(0)}$, tol, f and g, τ (restarting parameter)

Output: x that solves $f(x) = 0$.

Set $r^{(0)} = f(x^{(0)})$

Set $x^{(1)} = g(x^{(0)})$

Set $r^{(1)} = f(x^{(1)})$

Set $s^{(1)} = r^{(1)} - r^{(0)}$

Set $k = 1$, $m^{(k)} = 1$

while $\|r^{(k)}\|_2 > tol$ **do**

    Determine $\{\gamma_i^{(k)}\}_{i=1,\dots,m^{(k)}}$ that solves

$$\min_{\gamma_1,\dots,\gamma_{m^{(k)}}} \|r^{(k-m^{(k)})} + \sum_{i=1}^{m^{(k)}} \gamma_i r^{(k-m^{(k)}+i)}\|_2$$

    Set $x^{(k+1)} = g(x^{(k-m^{(k)})}) + \sum_{i=1}^{m^{(k)}} \gamma_i^{(k)} (g(x^{(k-m^{(k)}+i)}) - g(x^{(k-m^{(k)})})$

    Set $r^{(k+1)} = f(x^{(k+1)})$

    Set $s^{(k+1)} = r^{(k+1)} - r^{(k-m^{(k)})}$

    if $\tau\|s^{(k+1)}\|_2 > \|(\mathrm{id} - \Pi_k)s^{(k+1)}\|_2$ **then**

        $m^{(k+1)} = 0$

    else

        $m^{(k+1)} = m^{(k)} + 1$

    end if

    Set $k = k + 1$

end while

Algorithm 4 Adaptive-depth Anderson Acceleration

Input: $x^{(0)}$, tol, f and g, δ (parameter controlling adaptive process, how long of a history to keep)

Output: x that solves $f(x) = 0$.

1: Set $r^{(0)} = f(x^{(0)})$

2: Set $x^{(1)} = g(x^{(0)})$

3: Set $r^{(1)} = f(x^{(1)})$

4: Set $s^{(1)} = r^{(1)} - r^{(0)}$

5: Set $k = 1$, $m^{(k)} = 1$

6: **while** $\|r^{(k)}\|_2 > tol$ **do**

7: Determine $\{\gamma_i^{(k)}\}_{i=1,\dots,m^{(k)}}$ that solves

$$\min_{\gamma_1,\dots,\gamma_{m^{(k)}}} \|r^{(k-m^{(k)})} + \sum_{i=1}^{m^{(k)}} \gamma_i r^{(k-m^{(k)}+i)}\|_2$$

8: Set $x^{(k+1)} = g(x^{(k-m^{(k)})}) + \sum_{i=1}^{m^{(k)}} \gamma_i^{(k)}(g(x^{(k-m^{(k)}+i)}) - g(x^{(k-m^{(k)})})$

9: Set $r^{(k+1)} = f(x^{(k+1)})$

10: Set $s^{(k+1)} = r^{(k+1)} - r^{(k)}$

11: Set $m^{(k+1)}$ to be the largest integer $m \leq m^{(k)} + 1$ such that for $k + 1 - m \leq i \leq k$, $\delta\|r^{(i)}\|_2 < \|r^{(k+1)}\|_2$

12: Set $k = k + 1$

13: **end while**

2.6.2 CROP Method

Anderson Acceleration is a representative and widely adopted mixing-based acceleration technique. Algorithms that, like Anderson Acceleration, construct updates by solving a

least-squares problem involving linear combinations of past iterates may therefore be viewed collectively as Anderson-type acceleration methods. Within this class, the Conjugate Residual with OPtimal trial vectors (CROP) [20, 21] algorithm occupies an interesting position due to its structural and conceptual relationship to Anderson Acceleration. In analogy with Anderson Acceleration, let $x_C^{(k)}$ denote the CROP iterates, and define the sequence of recorded search directions

$$\Delta x_C^{(i)} := x_C^{(i+1)} - x_C^{(i)}, \quad i = k - m_C^{(k)}, \ldots, k-1,$$

together with the corresponding CROP residual (error) vectors $f_C^{(k)}$. Then the new search direction $\Delta x_C^{(k)} = x_C^{(k+1)} - x_C^{(k)}$ is chosen in the space spanned by the prior $m_C^{(k)}$ search directions $\Delta x_C^{(i)}, i = k - m_C^{(k)}, \ldots, k-1$ and the most recent residual (error) vector $f_C^{(k)}$, i.e.,

$$x_C^{(k+1)} \quad = \quad x_C^{(k)} + \sum_{i=k-m_C^{(k)}}^{k-1} \eta_i \Delta x_C^{(i)} + \eta_k f_C^{(k)},$$

with some coefficients $\eta_{k-m_C^{(k)}}, \ldots, \eta_k \in \mathbb{R}$.

Let us assume we have carried k steps of the CROP algorithm, i.e., we have the subspace of optimal vectors $\text{span}\{x_C^{(1)}, \ldots, x_C^{(k)}\}$ at hand. From the residual vector $f_C^{(k)}$, we can introduce a preliminary improvement of the current iterate $x_C^{(k)}$, i.e.,

$$\widetilde{x}_C^{(k+1)} := x_C^{(k)} + f_C^{(k)}. \tag{2.34}$$

Now, since (2.34) is equivalent to $f_C^{(k)} = \widetilde{x}_C^{(k+1)} - x_C^{(k)}$, we can find the optimal vector $x_C^{(k+1)}$ within the affine subspace $\text{span}\{x_C^{(1)}, \ldots, x_C^{(k)}, \widetilde{x}_C^{(k+1)}\}$, i.e.,

$$x_C^{(k+1)} = \sum_{i=0}^{m_C^{(k+1)}-1} \alpha_{C,i}^{(k+1)} x_C^{(k+1-m_C^{(k+1)}+i)} + \alpha_{C,m_C^{(k+1)}}^{(k+1)} \widetilde{x}_C^{(k+1)}, \tag{2.35}$$

with $\sum_{i=0}^{m_C^{(k+1)}} \alpha_{C,i}^{(k+1)} = 1$. The (linear) estimated residual (error) $f_C^{(k+1)}$ corresponding to the iterate $x_C^{(k+1)}$ is constructed as the linear combination of the estimated residuals (errors) of each component in (2.35) with the same coefficients, i.e.,

$$f_C^{(k+1)} = \sum_{i=0}^{m_C^{(k+1)}-1} \alpha_{C,i}^{(k+1)} f_C^{(k+1-m_C^{(k+1)}+i)} + \alpha_{C,m_C^{(k+1)}}^{(k+1)} \widetilde{f}_C^{(k+1)}. \tag{2.36}$$

Note that in general, unlike for the Anderson Acceleration method, $f_A^{(k+1)} = f(x_A^{(k+1)})$, as in Algorithm 2. In CROP we have $f_C^{(k+1)} \neq f(x_C^{(k+1)})$.

Minimizing the norm of the residual (error) defined in (2.36) results in a constrained least-squares problem

$$\min_{\alpha_0,\ldots,\alpha_{m_C^{(k+1)}}} \left\| \sum_{i=0}^{m_C^{(k+1)}-1} \alpha_i f_C^{(k+1-m_C^{(k+1)}+i)} + \alpha_{m_C^{(k+1)}} \widetilde{f}_C^{(k+1)} \right\|_2^2, \tag{2.37}$$

such that $\sum_{i=0}^{m_C^{(k+1)}} \alpha_i = 1$, with a vector of mixing coefficients $\alpha_C^{(k+1)}$ as a solution. The updates $x_C^{(k+1)}$ and $f_C^{(k+1)}$ can be written in the matrix form,

$$x_C^{(k+1)} = X_C^{(k+1)} \alpha_C^{(k+1)} \quad \text{and} \quad f_C^{(k+1)} = F_C^{(k+1)} \alpha_C^{(k+1)}, \tag{2.38}$$

with $X_C^{(k)} = \left[x_C^{(k-m_A^{(k)})}, \ldots, x_C^{(k)}, \widetilde{x}_C^{(k+1)}\right]$ and $F_C^{(k)} = \left[f_C^{(k-m_C^{(k)})}, \ldots, f_C^{(k)}, \widetilde{f}_C^{(k+1)}\right] \in \mathbb{R}^{n \times (m_C^{(k+1)}+1)}$, and coefficient vector $\alpha_C^{(k+1)} = \left[\alpha_{C,0}^{(k+1)}, \ldots, \alpha_{C,m_C^{(k+1)}}^{(k+1)}\right]^T \in \mathbb{R}^{m_C^{(k+1)}+1}$.

Algorithm 5 CROP Algorithm (of fixed depth m)

Input: Initial CROP iterate $x_C^{(0)}$, initial fixed depth $m \geq 1$

1: Compute $f_C^{(0)} = f(x_C^{(0)})$

2: **for** $k = 0, 1, 2, \ldots$ until convergence **do**

3: Set $\widetilde{x}_C^{(k+1)} = x_C^{(k)} + f_C^{(k)}$ and truncation parameter $m_C^{(k+1)}$

/* e.g. $m_C^{(k+1)} = \min\{k+1, m\}$ */

4: Set $F_C^{(k+1)} = \left[f_C^{(k+1-m_C^{(k)})}, \ldots, f_C^{(k)}, \widetilde{f}_C^{(k+1)}\right]$ with $\widetilde{f}_C^{(k+1)} = f(\widetilde{x}_C^{(k+1)})$

5: Determine mixing coefficients, i.e., $\alpha_C^{(k+1)} = \left[\alpha_{C,0}^{(k+1)}, \ldots, \alpha_{C,m_C^{(k+1)}}^{(k+1)}\right]^T$ that solves Problem (2.37)

6: Set $x_C^{(k+1)} = \sum_{i=0}^{m_C^{(k+1)}-1} \alpha_{C,i}^{(k+1)} x_C^{(k+1-m_C^{(k+1)}+i)} + \alpha_{C,m_C^{(k+1)}}^{(k+1)} \widetilde{x}_C^{(k+1)}$

7: Set CROP residuals as $f_C^{(k+1)} = \sum_{i=0}^{m_C^{(k+1)}-1} \alpha_{C,i}^{(k+1)} f_C^{(k+1-m_C^{(k+1)}+i)} + \alpha_{C,m_C^{(k+1)}}^{(k+1)} \widetilde{f}_C^{(k+1)}$

8: **end for**

Output: $x_C^{(k+1)}$ that solves $f(x) = 0$.

A variant of this method using directly evaluated residuals $f_C^{(k+1)} = f(x_C^{(k+1)})$ instead of the linear combination will be introduced later in Chapter 3 (rCROP algorithm).

In Algorithm 5 the superscript of the iterates in step k is chosen as $k+1$ instead of k for a reason. In this way, the case of $m_C^{(k)} = k$ indicates no truncation. Also, as we will see in Section 3.1, this choice enables us to understand the correspondence between classical Anderson Acceleration and the CROP algorithm.

Note that the least squares problem (2.37) is also of the form of (2.30). It means that we can use the solution (2.32) to write the update of the CROP method as

$$x_C^{(k+1)} = \widetilde{x}_C^{(k)} - \mathscr{X}_C^{(k+1)}[(\mathscr{F}_C^{(k+1)})^T \mathscr{F}_C^{(k+1)}]^{-1} (\mathscr{F}_C^{(k+1)})^T \widetilde{f}_C^{(k+1)}, \tag{2.39}$$

where

$$\mathscr{X}_C^{(k+1)} = \left[\Delta x_C^{(k+1-m_C^{(k+1)})}, \ldots, \Delta x_C^{(k-1)}, \widetilde{x}_C^{(k+1)} - x_C^{(k)}\right],$$

$$\mathscr{F}_C^{(k+1)} = [\Delta f_C^{(k+1-m_C^{(k+1)})}, \ldots, \Delta f_C^{(k-1)}, \widetilde{f}_C^{(k+1)} - f_C^{(k)}].$$

As for the truncation parameter for the CROP algorithm, $m_C^{(k)}$, it is slightly different from that for Anderson Acceleration. In Anderson Acceleration, at step k iterate $x_A^{(k+1)}$ is computed from iterates $x_A^{(k-m^{(k)})}, \ldots, x_A^{(k)}$, whereas in CROP, $x_C^{(k+1)}$ is computed using vectors $x_C^{(k+1-m^{(k+1)})}, \ldots, x_C^{(k)}, \widetilde{x}_C^{(k+1)}$. Therefore, we can immediately see that Anderson(1) is a mixing method that needs the historical information in every step, while CROP(1) is a fixed-point iteration, i.e., in the Anderson(1) step to compute $x_A^{(k+1)}$ we need access to $x_A^{(k-1)}$ and $x_A^{(k)}$, whereas in the case of CROP(1) we only need $x_C^{(k)}$ to determine next iterate $x_C^{(k+1)}$.

More information about the CROP algorithm will be introduced in Chapter 3. We will discuss the connection between CROP and other methods, and investigate the convergence theory of CROP.

2.6.3 Other Nonlinear Acceleration Methods

It is worth noting that the extrapolation methods in Section 2.6 can all be regarded as acceleration methods. Also, many of them can be viewed as variants of Anderson Acceleration. For example, the Regulized Nonlinear Acceleration (RNA) categorizes Anderson Acceleration, MPE, and RRE as the same nonlinear acceleration algorithm and applies Tikhonov regularization.

Many nonlinear acceleration methods have been developed to accelerate the self-consistent field (SCF) method, the standard algorithm for finding electronic structure configurations within Hartree-Fock and density functional theory. The Direct Inversion in the Iterative

Subspace (DIIS) [43] is a method that is equivalent to applying Anderson Acceleration to the SCF method.

Commutator-DIIS (CDIIS) [44] is a variant of DIIS for solving the Hartree-Fock equations. The density matrix D is updated until it commutes with the associated Fock matrix $F(D)$, i.e., $F(D)D - DF(D) = 0$. The new iterates $D^{(k+1)} = g(D^{(k)})$ are computed by the Roothaan SCF process. Hence, the residual (error) function $f(D)$ is defined as the commutator $f(D) = F(D)D - DF(D)$ instead of the difference between two consecutive iterates. This method still fits in the Anderson Acceleration framework.

The Optimal Damping Algorithm (ODA) [98] and its variant Energy-DIIS (EDIIS) are variants of DIIS that minimize the SCF energy. EDIIS has the same style of minimizing the weighted average as in Anderson Acceleration, but the weights are in $[0, 1]$.

Recently, more variants like Argumented-DIIS (ADIIS), Residual-DIIS (RDIIS) have been developed in similar ways. They all seek to accelerate SCF convergence in cases where DIIS performs poorly in the initial iterations.

Chapter 3

The CROP Algorithm

This chapter investigates the relationship between two mixing-based acceleration techniques: Anderson Acceleration [1, 2]—also known as Pulay mixing in computational chemistry [43, 44]—and the Conjugate Residual algorithm with OPtimal trial vectors (CROP) [20, 21]. The latter, originally proposed in [20], generalizes the classical Conjugate Residual (CR) method [22], a widely used iterative solver for linear systems. Motivated by the growing interest in nonlinear acceleration techniques, this chapter examines the conceptual and structural connections between these two methods with the goal of clarifying the mechanisms underlying CROP, analyzing its convergence behavior, and identifying circumstances under which it provides a viable alternative to Anderson Acceleration. Rather than proposing new algorithms, our aim is to develop a unified perspective that situates CROP within the broader landscape of Krylov subspace and multisecant methods. In exact arithmetic, we establish the equivalence between CROP and Anderson Acceleration and highlight the influence of various algorithmic parameters on their behavior. We further investigate the convergence properties of CROP and its truncated variants for both linear and nonlinear problems, and assess their practical performance through numerical experiments. The chapter is organized as follows. In Section 3.1, we introduce a unified framework that makes explicit the structural connections between Anderson Acceleration and the CROP algorithm, including the role of key parameters, and present their theoretical analysis. Section 3.3 is devoted to convergence results for CROP and its truncated forms. Finally, in Section 3.4, we provide numerical

experiments that illustrate and support the main theoretical findings.

This chapter is a part of the paper "On the Convergence of CROP-Anderson Acceleration Method" [110] with CRediT author statement:

Ning Wan: Conceptualization, Investigation, Methodology, Software, Writing - Original Draft.

Agnieszka Międlar: Conceptualization, Methodology, Supervision, Writing - Original Draft.

3.1 Anderson Acceleration vs. CROP Method

The aim of this section is to establish a unified framework which will enable us to show that Anderson Acceleration (see Section 2.6.1) and the CROP algorithm (see Section 2.6.2) are equivalent. By showing the equivalence of these two methods with no truncation, some new variants, i.e., CROP-Anderson and rCROP, can be developed and adapted to use the real residuals.

Theorem 3.1. *Let us consider applying Anderson Acceleration ($\beta^{(k)} = 1$) and the CROP algorithm to the nonlinear problem $f(x) = 0$ with initial values $x_A^{(0)} = x_C^{(0)}$ and no truncation ($m^{(k)} = k$). Then, for $k = 0, 1, \ldots$ until convergence ($f_C^{(0)} \neq 0$)*

$$x_C^{(k)} = \bar{x}_A^{(k)}, \quad f_C^{(k)} = \bar{f}_A^{(k)} \quad \text{and} \quad x_A^{(k+1)} = \widetilde{x}_C^{(k+1)}, \quad f_A^{(k+1)} = \widetilde{f}_C^{(k+1)},$$

with $\bar{x}_A^{(k)}$, $x_A^{(k+1)}$ defined as in (2.25) and (2.27), and $\widetilde{x}_C^{(k+1)}$, $x_C^{(k)}$ as in (2.34) and (2.35).

Proof. The proof follows by induction. Since $x_A^{(0)} = x_C^{(0)}$, then for $k = 0$, $x_C^{(0)} = \bar{x}_A^{(0)}$, $f_C^{(0)} = f(x_C^{(0)}) = f(x_A^{(0)}) = f_A^{(0)} = \bar{f}_A^{(0)}$. Hence, $\widetilde{x}_C^{(1)} = x_C^{(0)} + f_C^{(0)} = x_A^{(0)} + f_A^{(0)} = x_A^{(1)}$

and $\widetilde{f}_C^{(1)} = f(\widetilde{x}_C^{(1)}) = f(x_A^{(1)}) = f_A^{(1)}$. Assume that for all $\ell \leq k$, $x_C^{(\ell)} = \bar{x}_A^{(\ell)}$, $f_C^{(\ell)} = \bar{f}_A^{(\ell)}$, $\widetilde{x}_C^{(\ell+1)} = x_A^{(\ell+1)}$ and $\widetilde{f}_C^{(\ell+1)} = f_A^{(\ell+1)}$. Then, for $k+1$

$$\begin{aligned} f_C^{(k+1)} &= \sum_{i=0}^{m^{(k+1)}-1} \alpha_{C,i}^{(k+1)} f_C^{(k+1-m_C^{(k+1)}+i)} + \alpha_{C,m^{(k+1)}}^{(k+1)} \widetilde{f}_C^{(k+1)} \\ &= \sum_{i=0}^{k} \alpha_{C,i}^{(k+1)} \bar{f}_A^{(i)} + \alpha_{C,k+1}^{(k+1)} f_A^{(k+1)} = \sum_{i=0}^{k} \alpha_{C,i}^{(k+1)} \sum_{j=0}^{i} \alpha_{A,j}^{(i)} f_A^{(j)} + \alpha_{C,k+1}^{(k+1)} f_A^{(k+1)} \\ &= \sum_{j=0}^{k} \sum_{i=j}^{k} \alpha_{C,i}^{(k+1)} \alpha_{A,j}^{(i)} f_A^{(j)} + \alpha_{C,k+1}^{(k+1)} f_A^{(k+1)}. \end{aligned}$$

Let $\widehat{\alpha}_j^{(k+1)} = \sum_{i=j}^{k} \alpha_{C,i}^{(k+1)} \alpha_{A,j}^{(i)}$, $j = 1, \ldots, k$, and $\widehat{\alpha}_{k+1}^{(k+1)} = \alpha_{C,k+1}^{(k+1)}$. Then

$$f_C^{(k+1)} = \sum_{j=0}^{k+1} \widehat{\alpha}_j^{(k+1)} f_A^{(j)} \quad \text{and} \quad \sum_{j=0}^{k+1} \widehat{\alpha}_j^{(k+1)} = \sum_{i=0}^{k+1} \alpha_{C,i}^{(k+1)} = 1.$$

Since $f_C^{(k+1)} = \sum_{i=0}^{k+1} \widehat{\alpha}_i^{(k+1)} f_A^{(i)}$ and $\bar{f}_A^{(k+1)} = \sum_{i=0}^{k+1} \alpha_{A,i}^{(k+1)} f_A^{(i)}$ are the solutions of the least-squares problem in the same affine space $\mathrm{Aff}\{f_A^{(0)}, \ldots, f_A^{(k+1)}\}$, $\widehat{\alpha}_i^{(k+1)} = \alpha_{A,i}^{(k+1)}$, and $f_C^{(k+1)} = \bar{f}_A^{(k+1)}$. Also,

$$\begin{aligned} x_C^{(k+1)} &= \sum_{i=0}^{k+1} \widehat{\alpha}_i^{(k+1)} x_A^{(i)} = \sum_{i=0}^{k+1} \alpha_{A,i}^{(k+1)} f_A^{(i)} = \bar{x}_A^{(k+1)}, \\ \widetilde{x}_C^{(k+2)} &= x_C^{(k+1)} + f_C^{(k+1)} = \bar{x}_A^{(k+1)} + \bar{f}_A^{(k+1)} = x_A^{(k+2)}, \end{aligned}$$

and $\widetilde{f}_C^{(k+2)} = f(\widetilde{x}_C^{(k+2)}) = f(x_A^{(k+2)}) = f_A^{(k+2)}$. Therefore by induction $x_C^{(k)} = \bar{x}_A^{(k)}$, $f_C^{(k)} = \bar{f}_A^{(k)}$, $\widetilde{x}_C^{(k+1)} = x_A^{(k+1)}$ and $\widetilde{f}_C^{(k+1)} = f_A^{(k+1)}$ for all $k \in \mathbb{N}$. □

Since the CROP residual $f_C^{(k)}$ may become exactly zero, Algorithm 5 can potentially break down. However, Theorem 3.1 remains valid up to the point of breakdown. If the CROP

algorithm terminates at step k with $f_C^{(k)} = 0$, the iteration stops. In contrast, in Anderson Acceleration $x_A^{(k+1)} = x_A^{(k)}$, leading to stagnation. To illustrate a connection between Anderson Acceleration and the CROP algorithm, in Figure 3.1, we consider one and a half step of Anderson Acceleration from iterate $x_A^{(k)}$ to $\bar{x}_A^{(k+1)}$. By the equivalence of the least-squares problems in (2.28) and (2.37), block (II) in Figure 3.1 is equivalent to a single step of CROP. Changing the weighted averages of Anderson Acceleration to CROP type averages enables us to obtain a different scheme, illustrated in Figure 3.2.

$$
(I)\ \left\{\begin{array}{c}
x_A^{(k)} = \bar{x}_A^{(k-1)} + \bar{f}_A^{(k-1)} \\
\downarrow \\
f_A^{(k)} := f(x_A^{(k)}) \\
\downarrow \\
\text{Find } \alpha_A^{(k)} \text{ that minimizes } \|\bar{f}_A^{(k)}\|_2 \\
\text{with } \bar{f}_A^{(k)} = \sum_{i=0}^{m_A^{(k)}} \alpha_{A,i}^{(k)} f_A^{(k-m_A^{(k)}+i)} \\
\downarrow \\
\bar{x}_A^{(k)} = \sum_{i=0}^{m_A^{(k)}} \alpha_{A,i}^{(k)} x_A^{(k-m_A^{(k)}+i)} \\
\downarrow \\
x_A^{(k+1)} = \bar{x}_A^{(k)} + \bar{f}_A^{(k)} \\
\downarrow \\
f_A^{(k+1)} := f(x_A^{(k+1)}) \\
\downarrow \\
\text{Find } \alpha_A^{(k+1)} \text{ that minimizes } \|\bar{f}_A^{(k+1)}\|_2 \\
\text{with } \bar{f}_A^{(k+1)} = \sum_{i=0}^{m_A^{(k+1)}} \alpha_{A,i}^{(k+1)} f_A^{(k+1-m_A^{(k+1)}+i)} \\
\downarrow \\
\bar{x}_A^{(k+1)} = \sum_{i=0}^{m_A^{(k+1)}} \alpha_{A,i}^{(k+1)} x_A^{(k+1-m_A^{(k+1)}+i)}
\end{array}\right\} (II)
$$

Figure 3.1: Anderson Acceleration in Anderson notation.

Hence, blocks (I - IV) in Figures 3.1 and 3.2 can be associated with four different algorithms: block (I) illustrates Anderson Acceleration steps (2.25)–(2.27); block (II) is equivalent to

$$
(III)\left\{\begin{array}{c}
\widetilde{x}_C^{(k)} = x_C^{(k-1)} + f_C^{(k-1)} \\
\downarrow \\
\widetilde{f}_C^{(k)} := f(\widetilde{x}_C^{(k)}) \\
\downarrow \\
\text{Find } \alpha_C^{(k)} \text{ that minimizes } \|f_C^{(k)}\|_2 \\
\text{with } f_C^{(k)} = \sum_{i=0}^{m_C^{(k)}-1} \alpha_{C,i}^{(k)} f_C^{(k-m_C^{(k)}+i)} + \alpha_{C,m_C^{(k)}}^{(k)} \widetilde{f}_C^{(k)} \\
\downarrow \\
x_C^{(k)} = \sum_{i=0}^{m_C^{(k)}-1} \alpha_{C,i}^{(k)} x_C^{(k-m_C^{(k)}+i)} + \alpha_{C,m_C^{(k)}}^{(k)} \widetilde{x}_C^{(k)}
\end{array}\right.
$$

$$
(IV)\left\{\begin{array}{c}
\downarrow \\
\widetilde{x}_C^{(k+1)} = x_C^{(k)} + f_C^{(k)} \\
\downarrow \\
\widetilde{f}_C^{(k+1)} := f(\widetilde{x}_C^{(k+1)}) \\
\downarrow \\
\text{Find } \alpha_C^{(k+1)} \text{ that minimizes } \|f_C^{(k+1)}\|_2, \\
\text{with } f_C^{(k+1)} = \sum_{i=0}^{m_C^{(k+1)}-1} \alpha_{C,i}^{(k+1)} f_C^{(k+1-m_C^{(k+1)}+i)} + \alpha_{C,m_C^{(k+1)}}^{(k+1)} \widetilde{f}_C^{(k+1)} \\
\downarrow \\
x_C^{(k+1)} = \sum_{i=0}^{m_C^{(k+1)}-1} \alpha_{C,i}^{(k+1)} x_C^{(k+1-m_C^{(k+1)}+i)} + \alpha_{C,m_C^{(k+1)}}^{(k+1)} \widetilde{x}_C^{(k+1)}
\end{array}\right.
$$

Figure 3.2: the CROP method in CROP notation.

steps (2.34)–(2.35) of CROP; block (III) is equivalent to Anderson Acceleration steps (2.25)–(2.27) and block (IV) illustrates CROP steps (2.34)–(2.35). Note that following the notation of Figures 3.1 and 3.2 allows us to use the same variables $x^{(k)}$ and $\bar{x}^{(k)}$ in all four different algorithms (blocks I – IV) and write them all in the unified framework in terms of previous iterates of Anderson Acceleration. However, we can also express all quantities of interest in terms of CROP past information. Since the CROP algorithm has better behavior while keeping less historical information, we can run Anderson Acceleration executing block (III), which is called the CROP generalization of Anderson Acceleration [21] or, for the simplicity, the CROP-Anderson method. The CROP-Anderson method, denoted as CA, follows the steps of Algorithm 5, with the only difference of checking the size of residuals (errors) at

each iteration step and the final output result being $\widetilde{x}_C^{(k+1)}$.

Algorithm 6 CROP-Anderson method (of fixed depth m)

Input: Initial CROP-Anderson iterate $x^{(0)}$, initial fixed depth $m \geq 1$

Compute $f_C^{(0)} = f(x_C^{(0)})$

for $k = 0, 1, 2, \ldots$ **do**

  Set $\widetilde{x}_C^{(k+1)} = x_C^{(k)} + f_C^{(k)}$ and $\widetilde{f}_C^{(k+1)} = f(\widetilde{x}_C^{(k+1)})$

  if $\widetilde{f}_C^{(k+1)} < tol$ **then**

    break

  end if

  Set truncation parameter $m_C^{(k+1)}$

  `/* e.g.` $m_C^{(k+1)} = \min\{k+1, m\}$ `*/`

  Set $F_C^{(k+1)} = [f_C^{(k-m_C^{(k)})}, \ldots, f_C^{(k)}, \widetilde{f}_C^{(k+1)}]$.

  Determine mixing coefficients, i.e., $\alpha_C^{(k+1)} = [\alpha_{C,0}^{(k+1)}, \ldots, \alpha_{C,m_C^{(k+1)}}^{(k+1)}]^T$ that solves Problem (2.37)

  Set $x_C^{(k+1)} = \sum_{i=0}^{m_C^{(k+1)}-1} \alpha_{C,i}^{(k+1)} x_C^{(k+1-m_C^{(k+1)}+i)} + \alpha_{C,m_C^{(k+1)}}^{(k+1)} \widetilde{x}_C^{(k+1)}$

  Set $f_C^{(k+1)} = \sum_{i=0}^{m_C^{(k+1)}-1} \alpha_{C,i}^{(k+1)} f_C^{(k+1-m_C^{(k+1)}+i)} + \alpha_{C,m_C^{(k+1)}}^{(k+1)} \widetilde{f}_C^{(k+1)}$

end for

Output: $\widetilde{x}_C^{(k+1)}$ that solves $f(x) = 0$.

Although we can express the steps of the CROP-Anderson method using CROP algorithm (C) notation, we can easily use the alternative CROP-Anderson (CA) formulation, i.e., we first set $x_{CA}^{(k)} = \widetilde{x}_C^{(k)}$, $f_{CA}^{(k)} = \widetilde{f}_C^{(k)}$, $\bar{x}_{CA}^{(k)} = x_C^{(k)}$ and $\bar{f}_{CA}^{(k)} = f_C^{(k)}$. Then by Theorem 3.1, we get the following corollary.

Corollary 3.2. *Let us consider applying Anderson Acceleration ($\beta_k = 1$) and the CROP-Anderson algorithm to the nonlinear problem $f(x) = 0$ with initial values $x_A^{(0)} = x_C^{(0)} = x_{CA}^{(0)}$*

and no truncation ($m^{(k)} = k$). Then, for $k = 0, 1, \ldots$

$$x_{CA}^{(k)} = x_A^{(k)}, \; f_{CA}^{(k)} = f_A^{(k)} \quad \text{and} \quad \bar{x}_{CA}^{(k)} = \bar{x}_A^{(k)}, \; \bar{f}_{CA}^{(k)} = \bar{f}_A^{(k)}.$$

In the CROP algorithm, updates $x_C^{(k)}$ have the corresponding residuals $f_C^{(k)}$ associated with the least-squares Problem (2.37). Although $f_C^{(k)}$ are used to terminate the iterations, they are still approximated residuals and are affected by the initial guess $x_C^{(0)}$. Thus, we call them *control residuals*. Relatively, $r_C^{(k)} = f(x_C^{(k)})$ are the *real residuals* corresponding to iterates $x_C^{(k)}$, which obviously are never explicitly computed. Consequently, when solving nonlinear problems, the control residuals may not estimate the real residuals very well, and may cause a lot of problems, e.g. breakdowns. One remedy is to use the CROP-Anderson method. Alternatively, the real residuals can be used instead of the control residuals. By changing line 7 of Algorithm 5 and the corresponding line in CROP-Anderson (line 11 in Algorithm 6) into $f_C^{(k+1)} = f(x_C^{k+1})$, CROP and CROP-Anderson become completely different algorithms. In what follows, we will refer to them as rCROP and rCROP-Anderson. Note that in the case of Anderson Acceleration, the control residuals and the real residuals are the same, i.e., $f_A^{(k)} = f(x_A^{(k)}) = r_A^{(k)}$.

The diagrams above are summarized in Table 3.1 which illustrates two steps of Anderson Acceleration and CROP. The k^{th} step of each method is highlighted in **bold**. Note that the "optimization" steps have different results for the two methods, but the "average x" and the "iteration k" steps should have the same value. Analogously, the "average f" and the "residual k" steps should be the same. If we consider $\widetilde{x}_C$ and $\widetilde{f}_C$ from CROP as iterates and residuals, respectively, then we get steps of CROP-Anderson (shaded part of CROP), which is equivalent to Anderson Acceleration. If in CROP and CROP-Anderson we use $f_C^{(k)} = f(x_C^{(k)})$ instead of $f_C^{(k)} = F_C^{(k)} \alpha_C^{(k)}$ as the k^{th} residual, then we get real residual CROP

(rCROP) and the corresponding rCROP-Anderson (shaded part of CROP method). The rCROP algorithm is presented along with Anderson Acceleration in Table 3.2.

	Anderson Acceleration	**CROP**	
iteration k	$x_A^{(k)} = \bar{x}_A^{(k-1)} + \bar{f}_A^{(k-1)}$	$\widetilde{x}_C^{(k)} = x_C^{(k-1)} + f_C^{(k-1)}$	new direction
residual k	$f_A^{(k)} = f(x_A^{(k)})$	$\widetilde{f}_C^{(k)} = f(\widetilde{x}_C^{(k)})$	new direction
optimization	$\alpha_{\mathbf{A}}^{(\mathbf{k})} = \arg\min \|\mathbf{F}_{\mathbf{A}}^{(\mathbf{k})}\alpha\|_{\mathbf{2}}$	$\alpha_C^{(k)} = \arg\min \|F_C^{(k)}\alpha\|_2$	optimization
average x	$\mathbf{x}_{\mathbf{A}}^{(\mathbf{k})} = \mathbf{X}_{\mathbf{A}}^{(\mathbf{k})}\alpha_{\mathbf{A}}^{(\mathbf{k})}$	$x_C^{(k)} = X_C^{(k)}\alpha_C^{(k)}$	iteration k
average f	$\mathbf{f}_{\mathbf{A}}^{(\mathbf{k})} = \mathbf{F}_{\mathbf{A}}^{(\mathbf{k})}\alpha_{\mathbf{A}}^{(\mathbf{k})}$	$f_C^{(k)} = F_C^{(k)}\alpha_C^{(k)}$	residual k
iteration $k+1$	$\mathbf{x}_{\mathbf{A}}^{(\mathbf{k+1})} = \mathbf{x}_{\mathbf{A}}^{(\mathbf{k})} + \mathbf{f}_{\mathbf{A}}^{(\mathbf{k})}$	$\widetilde{\mathbf{x}}_{\mathbf{C}}^{(\mathbf{k+1})} = \mathbf{x}_{\mathbf{C}}^{(\mathbf{k})} + \mathbf{f}_{\mathbf{C}}^{(\mathbf{k})}$	**new direction**
residual $k+1$	$\mathbf{f}_{\mathbf{A}}^{(\mathbf{k+1})} = \mathbf{f}(\mathbf{x}_{\mathbf{A}}^{(\mathbf{k+1})})$	$\widetilde{\mathbf{f}}_{\mathbf{C}}^{(\mathbf{k+1})} = \mathbf{f}(\widetilde{\mathbf{x}}_{\mathbf{C}}^{(\mathbf{k+1})})$	**new direction**
optimization	$\alpha_A^{(k+1)} = \arg\min \|F_A^{(k+1)}\alpha\|_2$	$\alpha_{\mathbf{C}}^{(\mathbf{k+1})} = \arg\min \|\mathbf{F}_{\mathbf{C}}^{(\mathbf{k+1})}\alpha\|_{\mathbf{2}}$	**optimization**
average x	$\bar{x}_A^{(k+1)} = X_A^{(k+1)}\alpha_A^{(k+1)}$	$\mathbf{x}_{\mathbf{C}}^{(\mathbf{k+1})} = \mathbf{X}_{\mathbf{C}}^{(\mathbf{k+1})}\alpha_{\mathbf{C}}^{(\mathbf{k+1})}$	**iteration** $k+1$
average f	$\bar{f}_A^{(k+1)} = F_A^{(k+1)}\alpha_A^{(k+1)}$	$\mathbf{f}_{\mathbf{C}}^{(\mathbf{k+1})} = \mathbf{F}_{\mathbf{C}}^{(\mathbf{k+1})}\alpha_{\mathbf{C}}^{(\mathbf{k+1})}$	**residual** $k+1$

Table 3.1: Two iteration steps of Anderson Acceleration and CROP.

	Anderson Acceleration	**rCROP**	
iteration k	$x_A^{(k)} = \bar{x}_A^{(k-1)} + \bar{f}_A^{(k-1)}$	$\widetilde{x}_C^{(k)} = x_C^{(k-1)} + f_C^{(k-1)}$	new direction
residual k	$f_A^{(k)} = f(x_A^{(k)})$	$\widetilde{f}_C^{(k)} = f(\widetilde{x}_C^{(k)})$	new direction
optimization	$\alpha_{\mathbf{A}}^{(\mathbf{k})} = \arg\min \|\mathbf{F}_{\mathbf{A}}^{(\mathbf{k})}\alpha\|_{\mathbf{2}}$	$\alpha_C^{(k)} = \arg\min \|F_C^{(k)}\alpha\|_2$	optimization
average x	$\mathbf{x}_{\mathbf{A}}^{(\mathbf{k})} = \mathbf{X}_{\mathbf{A}}^{(\mathbf{k})}\alpha_{\mathbf{A}}^{(\mathbf{k})}$	$x_C^{(k)} = X_C^{(k)}\alpha_C^{(k)}$	iteration k
average f	$\mathbf{f}_{\mathbf{A}}^{(\mathbf{k})} = \mathbf{F}_{\mathbf{A}}^{(\mathbf{k})}\alpha_{\mathbf{A}}^{(\mathbf{k})}$	$\mathbf{f}_{\mathbf{C}}^{(\mathbf{k})} = \mathbf{f}(\mathbf{x}_{\mathbf{C}}^{(\mathbf{k})})$	residual k
iteration $k+1$	$\mathbf{x}_{\mathbf{A}}^{(\mathbf{k+1})} = \mathbf{x}_{\mathbf{A}}^{(\mathbf{k})} + \mathbf{f}_{\mathbf{A}}^{(\mathbf{k})}$	$\widetilde{\mathbf{x}}_{\mathbf{C}}^{(\mathbf{k+1})} = \mathbf{x}_{\mathbf{C}}^{(\mathbf{k})} + \mathbf{f}_{\mathbf{C}}^{(\mathbf{k})}$	**new direction**
residual $k+1$	$\mathbf{f}_{\mathbf{A}}^{(\mathbf{k+1})} = \mathbf{f}(\mathbf{x}_{\mathbf{A}}^{(\mathbf{k+1})})$	$\widetilde{\mathbf{f}}_{\mathbf{C}}^{(\mathbf{k+1})} = \mathbf{f}(\widetilde{\mathbf{x}}_{\mathbf{C}}^{(\mathbf{k+1})})$	**new direction**
optimization	$\alpha_A^{(k+1)} = \arg\min \|F_A^{(k+1)}\alpha\|_2$	$\alpha_{\mathbf{C}}^{(\mathbf{k+1})} = \arg\min \|\mathbf{F}_{\mathbf{C}}^{(\mathbf{k+1})}\alpha\|_{\mathbf{2}}$	**optimization**
average x	$\bar{x}_A^{(k+1)} = X_A^{(k+1)}\alpha_A^{(k+1)}$	$\mathbf{x}_{\mathbf{C}}^{(\mathbf{k+1})} = \mathbf{X}_{\mathbf{C}}^{(\mathbf{k+1})}\alpha_{\mathbf{C}}^{(\mathbf{k+1})}$	**iteration** $k+1$
average f	$\bar{f}_A^{(k+1)} = F_A^{(k+1)}\alpha_A^{(k+1)}$	$\mathbf{f}_{\mathbf{C}}^{(\mathbf{k+1})} = \mathbf{f}(\mathbf{x}_{\mathbf{C}}^{(\mathbf{k+1})})$	**residual** $k+1$

Table 3.2: Two iteration steps of Anderson Acceleration and **rCROP**.

Note that the standard Anderson Acceleration does not correspond to rCROP. To establish correspondence, we consider a variant of Anderson Acceleration in which the "average f" step is defined as $\bar{f}_A^{(k)} = f(\bar{x}_A^{(k)})$, as proposed in [17], and referred to as Anderson Acceleration (Version P). It is important to emphasize that the correspondence between rCROP and Anderson Acceleration (Version P) does not imply residual equivalence, since the residuals in these two methods cannot be obtained from one another through a linear transformation and therefore span different subspaces. Instead, this correspondence highlights the structural similarity between the algorithms and the equivalence in the number of floating-point operations (FLOPs) per iteration.

3.2 Broader View of Anderson Acceleration Method and CROP Algorithm

This section discusses the connections between the CROP algorithm and some other state-of-the-art iterative methods. Following our findings in Section 3.1, we explore links between CROP and Krylov subspace methods in Section 3.2.1, and multisecant methods in Section 3.2.2.

In [111, 112], a Krylov acceleration method equivalent to flexible GMRES was introduced, and its Jacobian-free version utilizes least-squares similar to (2.28). The work [7] pointed out the connection between Anderson Acceleration and the GMRES method, and details on the equivalence between Anderson Acceleration without truncation and the GMRES method were provided [6, 102]. The paper [13] showed that truncated Anderson Acceleration is a multi-Krylov method.

3.2.1 Connection with Krylov Methods

Let us consider applying the CROP algorithm to the simple linear problem: Find $x \in \mathbb{R}^n$ such that $Ax = b$, with a nonsingular $A \in \mathbb{R}^{n\times n}$ and $b \in \mathbb{R}^n$. Then, the associated residual (error) vectors can be chosen as $f(x) = b - Ax$, with the corresponding $g(x) = b + (I - A)x$. First, we will present some facts about the CROP algorithm's application to the linear problem $Ax = b$.

Lemma 3.3. *Consider using the CROP algorithm to solve the linear problem $Ax = b$. Then, the real and the control residuals are equal, i.e., $r_C^{(k)} = f_C^{(k)}$ for any $k \in \mathbb{N}$.*

Proof. The proof follows by induction. For the initial residuals we have $r_C^{(0)} = f_C^{(0)}$. Assume that $r_C^{(\ell)} = f_C^{(\ell)}$ for all $\ell \leq k$. Then, for $k+1$

$$\begin{aligned}
f_C^{(k+1)} &= \sum_{i=0}^{m_C^{(k+1)}-1} \alpha_{C,i}^{(k+1)} f_C^{(k+1-m_C^{(k+1)}+i)} + \alpha_{C,m_C^{(k+1)}}^{(k+1)} \widetilde{f}_C^{(k+1)} \\
&= \sum_{i=0}^{m_C^{(k+1)}-1} \alpha_{C,i}^{(k+1)} r_C^{(k+1-m_C^{(k+1)}+i)} + \alpha_{C,m_C^{(k+1)}}^{(k+1)} \widetilde{f}_C^{(k+1)} \\
&= \sum_{i=0}^{m_C^{(k+1)}-1} \alpha_{C,i}^{(k+1)} \big(b - Ax_C^{(k+1-m_C^{(k+1)}+i)}\big) + \alpha_{C,m_C^{(k+1)}}^{(k+1)} \big(b - A\widetilde{x}_C^{(k+1)}\big) \\
&= b - A\Bigg(\sum_{i=0}^{m_C^{(k+1)}-1} \alpha_{C,i}^{(k+1)} x_C^{(k+1-m_C^{(k+1)}+i)} + \alpha_{C,m_C^{(k+1)}}^{(k+1)} \widetilde{x}_C^{(k+1)} \Bigg) \\
&= b - Ax_C^{(k+1)} = r_C^{(k+1)}.
\end{aligned}$$

Hence, by induction $r_C^{(k)} = f_C^{(k)}$ for all $k \in \mathbb{N}$. □

Lemma 3.4. *If the CROP algorithm is used to solve the linear problem $Ax = b$, then $\widetilde{f}_C^{(k+1)} = (I - A)f_C^{(k)}$ for any $k \in \mathbb{N}$.*

Proof.

$$\begin{aligned}\widetilde{f}_C^{(k+1)} &= b - A\widetilde{x}_C^{(k+1)} = b - A\Big(b + (I-A)x_C^{(k)}\Big) = (I-A)(b - Ax_C^{(k)}) \\ &= (I-A)r_C^{(k)} = (I-A)f_C^{(k)}.\end{aligned}$$

□

The equivalence of Anderson Acceleration and the GMRES method is presented in [6, 102]. A similar result for the CROP algorithm can be proved.

Theorem 3.5. *Consider using the CROP algorithm and the GMRES method to solve the linear problem $Ax = b$ under the following assumptions:*

1. *A is nonsingular.*
2. *Run CROP with $f(x) = b - Ax$, $g(x) = f(x) + x = b + (I - A)x$ and no truncation.*
3. *The initial values of the GMRES and CROP coincide, i.e., $x_G^{(0)} = x_C^{(0)}$.*
4. *GMRES does not break down in the k_0-th step and no stagnation occurs for GMRES before the k_0-th step, i.e., $r_G^{(k_0-1)} > 0$ and $r_G^{(k-1)} > r_G^{(k)}$ for each k such that $0 < k < k_0$, where $r_G^{(k)}$ denotes the GMRES residual defined as $r_G^{(k)} := b - Ax_G^{(k)}$.*

Then, for $0 < k < k_0$, $x_C^{(k)} = x_G^{(k)}$, $f_C^{(k)} = r_G^{(k)}$.

Remark 3.6. This result can also be established by combining Theorem 3.1 and the known equivalence between Anderson Acceleration and the GMRES method [6, 102]. Here, however, we provide a direct proof for completeness.

Before proving Theorem 3.5, we first need the following lemma.

Lemma 3.7. *Consider using the CROP algorithm and the GMRES method to solve the linear problem $Ax = b$ under the assumptions from Theorem 3.5, and let the Krylov subspace associated with the matrix A and vector $r_G^{(0)}$ be given as*

$$\mathcal{K}_n = \mathcal{K}_n(A, r_G^{(0)}) = span\{r_G^{(0)}, Ar_G^{(0)}, \dots, A^{n-1}r_G^{(0)}\}.$$

Then, $\mathcal{K}_n = span\{f_C^{(0)}, \dots, f_C^{(n-1)}\}$ for all $n \in \mathbb{N}^+$.

Proof. The proof proceeds by induction. For $n = 1$, the Krylov subspace $\mathcal{K}_1 = span\{r_G^{(0)}\} = span\{f_C^{(0)}\}$. Assume that for $n = k + 1$, the Krylov subspace $\mathcal{K}_{k+1} = span\{f_C^{(0)}, \dots, f_C^{(k)}\}$. Then, for $n = k + 2$,

$$\widetilde{f}_C^{(k+1)} = (I - A)f_C^{(k)} = f_C^{(k)} - Af_C^{(k)} \in \mathcal{K}_{k+2}.$$

Since $f_C^{(k+1)}$ is a linear combination of vectors $f_C^{(0)}, \dots, f_C^{(k)}, \widetilde{f}_C^{(k+1)}$, thus $f_C^{(k+1)} \in \mathcal{K}_{k+2}$. □

Finally, we are ready to prove Theorem 3.5.

Proof of Theorem 3.5. We show by induction that at step k of CROP, the vectors $f_C^{(0)}, \dots, f_C^{(k)}$ form the Krylov subspace $\mathcal{K}_{k+1}$, i.e., $\mathcal{K}_{k+1} := span\{f_C^{(0)}, \dots, f_C^{(k)}\}$. For $k = 0$, the Krylov subspace $\mathcal{K}_1 = span\{r_G^{(0)}\} = span\{f_C^{(0)}\}$, which proves the base case.
Assume that at step k we have Krylov subspace $\mathcal{K}_{k+1} = span\{f_C^{(0)}, \dots, f_C^{(k)}\}$ as an induction hypothesis. Then at step $k + 1$

$$\widetilde{f}_C^{(k+1)} = (I - A)f_C^{(k)} = f_C^{(k)} - Af_C^{(k)} \in \mathcal{K}_{k+2}.$$

Since $f_C^{(k+1)}$ is a linear combination of vectors $f_C^{(0)}, \dots, f_C^{(k)}, \widetilde{f}_C^{(k+1)}$, $f_C^{(k+1)} \in \mathcal{K}_{k+2}$. Moreover, we need to show that $f_C^{(k+1)} \notin \mathcal{K}_{k+1}$ which requires $\alpha_{C,m_C^{(k+1)}}^{(k+1)} \neq 0$ and $\widetilde{f}_C^{(k+1)} \notin \mathcal{K}_{k+1}$. These,

however, are satisfied unless the algorithm stagnates. The equivalence of CROP algorithm and GMRES follows directly from the fact that $f_C^{(k)} = r_G^{(k)} = r_0 - Av$ where $v = \arg\min_{v \in \mathcal{K}_k} \|r_0 - Av\|$. □

Stagnation of GMRES in the k-th step implies stagnation in the k-th step of CROP. As we know, if GMRES does not break down upon stagnation, it can continue making progress and is guaranteed to converge, but this is not the case for CROP.

If CROP stagnates in the k-th step, then $f_C^{(k)} = f_C^{(k-1)}$, and thus $\widetilde{f}_C^{(k+1)} = \widetilde{f}_C^{(k)}$. So for the $(k+1)$-th step, we have

$$\begin{aligned} F_C^{(k+1)}\alpha^{(k+1)} &= \alpha_0^{(k+1)} f_C^{(0)} + \cdots + \alpha_{k-1}^{(k+1)} f_C^{(k-1)} + \alpha_k^{(k+1)} f_C^{(k)} + \alpha_{k+1}^{(k+1)} \widetilde{f}_C^{(k+1)} \\ &= \alpha_0^{(k+1)} f_C^{(0)} + \cdots + \alpha_{k-1}^{(k+1)} f_C^{(k-1)} + \alpha_k^{(k+1)} f_C^{(k-1)} + \alpha_{k+1}^{(k+1)} \widetilde{f}_C^{(k)} \\ &= \alpha_0^{(k+1)} f_C^{(0)} + \cdots + (\alpha_{k-1}^{(k+1)} + \alpha_k^{(k+1)}) f_C^{(k-1)} + \alpha_{k+1}^{(k+1)} \widetilde{f}_C^{(k)}. \end{aligned}$$

Therefore, the least-squares problem $\min_\alpha \|F_C^{(k+1)}\alpha\|$ and $\min_\alpha \|F_C^{(k)}\alpha\|$ are on the same affine subspace, and should have linear dependent solutions. If $\alpha_C^{(k)} = [\alpha_{C,0}^{(k)}, \ldots, \alpha_{C,k-1}^{(k)}, \alpha_{C,k}^{(k)}]^T$ is a solution of $\min_\alpha \|F_C^{(k)}\alpha\|$, then $\alpha_C^{(k+1)} = [\alpha_{C,0}^{(k)}, \ldots, \lambda\alpha_{C,k-1}^{(k)}, (1-\lambda)\alpha_{C,k-1}^{(k)}, \alpha_{C,k}^{(k)}]^T$ with any $\lambda \in \mathbb{R}$ is a solution of $\min_\alpha \|F_C^{(k+1)}\alpha\|$. And thus we have $f_C^{(k+1)} = f_C^{(k)} = f_C^{(k-1)}$. Repeating the process above, we will see that $f_C^{(k-1)} = f_C^{(k)} = f_C^{(k+1)} = \ldots$, which means that stagnation will occur in every step after step k, and the CROP algorithm will not make any further progress. Next, we will show that, similarly to the full CROP algorithm, there exists a truncated linear method equivalent to the truncated CROP(m) algorithm. In Theorem 3.5, we have shown the equivalence between CROP and GMRES. Since GMRES is equivalent to the Generalized Conjugate Residual (GCR) algorithm, let us consider the truncated GCR, namely, the ORTHOMIN method [5, Section 6.9]). For the general CROP(m) method, the following theorem holds.

Theorem 3.8. *Consider solving the linear system $Ax = b$ by the CROP(m) algorithm and the ORTHOMIN(m − 1) method, under the following assumptions:*

1. *A is nonsingular.*
2. *Run CROP(m) with $f(x) = b - Ax$ and $g(x) = f(x) + x = b + (I - A)x$.*
3. *CROP algorithm is truncated with parameter m, and ORTHOMIN with parameter $m - 1$.*
4. *The initial values of CROP(m) and ORTHOMIN(m − 1) coincide, i.e., $x^{(0)}_{Omin(m-1)} = x^{(0)}_{CROP(m)}$.*

Then for $k > 0$, $x^{(k)}_{CROP(m)} = x^{(k)}_{Omin(m-1)}$ and $f^{(k)}_{CROP(m)} = r^{(k)}_{Omin(m-1)}$, with the ORTHOMIN residuals $r^{(k)}_{Omin(m-1)} = b - Ax^{(k)}$.

Proof. We prove Theorem 3.8 by induction on m. Let us start with $m = 1$. At step $k + 1$ of CROP(1), iterate $x_C^{(k+1)}$ is determined directly from $x_C^{(k)}$ and $f_C^{(k)}$, hence making it a fixed-point iteration. According to (2.36), the control residuals in CROP(1) are given as

$$f_C^{(k+1)} = \alpha_{C,0}^{(k+1)} f_C^{(k)} + \alpha_{C,1}^{(k+1)} \widetilde{f}_C^{(k+1)}. \tag{3.1}$$

Since $\widetilde{f}_C^{(k+1)} = (I - A)f_C^{(k)}$ and $\alpha_{C,0}^{(k+1)} = 1 - \alpha_{C,1}^{(k+1)}$, (3.1) yields

$$f_C^{(k+1)} = \big(1 - \alpha_{C,1}^{(k+1)}\big) f_C^{(k)} + \alpha_{C,1}^{k+1} \big(I - A\big) f_C^{(k)} = f_C^{(k)} - \alpha_{C,1}^{(k+1)} A f_C^{(k)}. \tag{3.2}$$

To obtain a solution of the least-squares problem (2.37) which minimizes $\|f_C^{(k+1)}\|_2$, we need $f_C^{(k+1)} \perp A f_C^{(k)}$ and thus

$$\alpha_{C,1}^{(k+1)} = \frac{(f_C^{(k)})^T A f_C^{(k)}}{(A f_C^{(k)})^T A f_C^{(k)}}. \tag{3.3}$$

Therefore,

$$\begin{aligned} x_C^{(k+1)} &= \alpha_{C,0}^{k+1} x_C^{(k)} + \alpha_{C,1}^{(k+1)} \widetilde{x}_C^{(k+1)} = \big(1 - \alpha_{C,1}^{(k+1)}\big) x_C^{(k)} + \alpha_{C,1}^{(k+1)} \big(x_C^{(k)} + f_C^{(k)}\big) \\ &= x_C^{(k)} + \alpha_{C,1}^{(k+1)} f_C^{(k)}. \end{aligned} \tag{3.4}$$

Hence, iterates $x_C^{(k+1)}$ and residuals $f_C^{(k+1)}$ are updated according to (3.4) and (3.2), respectively, with $\alpha_{C,1}^{(k+1)}$ computed as in (3.3). Moreover, they are exactly the same as those calculated by the ORTHOMIN(0), i.e., CROP(1) = ORTHOMIN(0). Assume that CROP(ℓ) = ORTHOMIN($\ell - 1$) for $\ell = 1, \ldots, m-1$. Then, we can prove CROP(m) = ORTHOMIN($m-1$) by induction on k. Since $m_C^{(k+1)} = \min\{k+1, m\}$, for $k = 0$, $m_C^{(k+1)} = 1$ and CROP(m) = CROP(1) = ORTHOMIN(0), which is the same as the first step of ORTHOMIN(m) for any $m > 1$. For $k+1 < m$, step $k+1$ of CROP(m) is the same as step $k+1$ of CROP($k+1$), which is equivalent to step $k+1$ of ORTHOMIN(k) by the induction assumption, and thus is the same as step $k+1$ of ORTHOMIN($m-1$). Hence, we only need to consider the case of $k+1 \geq m$. Assume that for some $k > 0$, the vectors $\left\{A\Delta x_C^{(k+1-m_C^{(k)})}, \ldots, A\Delta x_C^{(k-1)}, f_C^{(k)}\right\}$ are pairwise orthogonal and $x_{CROP(m)}^{(k)} = x_{Omin(m-1)}^{(k)}$. Then, for general CROP(m), the $k+1$ residual is

$$f_C^{(k+1)} = \sum_{i=0}^{m-1} \alpha_{C,i}^{(k+1)} f_C^{(k+1-m)} + \alpha_{C,m}^{(k+1)} \widetilde{f}_C^{(k+1)}. \tag{3.5}$$

By Lemma 3.4, $\widetilde{f}_C^{(k+1)} = (I - A) f_C^{(k)}$. Then, with $\alpha_{C,0}^{(k+1)} = \gamma_{C,1}^{(k+1)}$, $\alpha_{C,i}^{(k+1)} = \gamma_{C,i+1}^{(k+1)} - \gamma_{C,i}^{(k+1)}$ and $\gamma_{C,m}^{(k+1)} = 1 - \alpha_{C,m}^{(k+1)}$, and noting that $\Delta f_C^{(k-m+i)} = -A\Delta x_C^{(k-m+i)}$, we get

$$\begin{aligned} f_C^{(k+1)} &= \sum_{i=0}^{m} \alpha_{C,i}^{(k+1)} f_C^{(k+1-m)} + \alpha_{C,m}^{(k+1)} (I - A) f_C^{(k)} \\ &= f_C^{(k)} - \alpha_{C,m}^{(k+1)} A f_C^{(k)} - \sum_{i=1}^{m-1} \gamma_{C,i}^{(k+1)} A\Delta x_C^{(k-m+i)}. \end{aligned} \tag{3.6}$$

Since by the induction assumption $\{A\Delta x_C^{(k-m+1)}, \ldots, A\Delta x_C^{(k-1)}, f_C^{(k)}\}$ are pairwise orthogonal, if we orthogonalize $Af_C^{(k)}$ against all $A\Delta x_C^{(k-m+i)}$ for $i = 1, \ldots, m-1$ and let

$$Ap^{(k)} = Af_C^{(k)} - \sum_{i=1}^{m-1} \frac{(Af_C^{(k)})^T A\Delta x_C^{(k-m+i)}}{(A\Delta x_C^{(k-m+i)})^T A\Delta x_C^{(k-m+i)}} A\Delta x_C^{(k-m+i)}, \tag{3.7}$$

then (3.6) yields $f_C^{(k+1)} = f_C^{(k)} - \alpha_{C,m}^{k+1} Ap^{(k)} + \sum_{i=1}^{m-1} c_i A\Delta x_C^{(k-m+i)}$. To obtain a solution of the least-squares problem (2.37) which minimizes $\|f_C^{(k+1)}\|_2$, we need $f_C^{(k+1)} \perp Ap^{(k)}$ and $f_C^{(k+1)} \perp A\Delta x_C^{(k-m+i)}$ for all $i = 1, \ldots, m-1$. Thus

$$\alpha_{C,m}^{(k+1)} = \frac{(f_C^{(k)})^T Ap^{(k)}}{(Ap^{(k)})^T Ap^{(k)}}, \tag{3.8}$$

and $c_i = 0$ for all $i = 1, \ldots, m-1$. The actual updates of $x_C^{(k+1)}$ and $f_C^{(k+1)}$ are

$$x_C^{(k+1)} = x_C^{(k)} + \alpha_{C,m}^{(k+1)} p^{(k)} \qquad \text{and} \qquad f_C^{(k+1)} = f_C^{(k)} - \alpha_{C,m}^{(k+1)} Ap^{(k)}. \tag{3.9}$$

Since $\Delta x_C^{(k)} = \alpha_{C,m}^{(k+1)} p^{(k)}$ is parallel to $p^{(k)}$, we can rewrite (3.7) by replacing $\Delta x_C^{(k-m+i)}$ with $p^{(k-m+i)}$. Hence, the orthogonalization step (3.7) and the update formulas (3.9) are the same as those in ORTHOMIN$(m-1)$. Therefore, in the case of solving a linear system, the CROP(m) algorithm is equivalent to ORTHOMIN$(m-1)$. □

Remark 3.9. Since for a symmetric matrix A, ORTHOMIN(0) is the Minimal Residual (MINRES) method [5, Section 5.3.2], and ORTHOMIN(1) is the CR method, for symmetric linear systems CROP(1) is equivalent to MINRES and CROP(2) to the CR method.

3.2.2 Connection with Multisecant Methods

The generalized Broyden's second method is a multisecant method [9] equivalent to Anderson Acceleration [8, 9] with iterates generated according to the update formula

$$x^{(k+1)} = x^{(k)} - G^{(k)} f^{(k)}, \tag{3.10}$$

where $G^{(k)}$ is an approximated inverse of the Jacobian updated by

$$G^{(k)} = G^{(k-m)} + (\mathscr{X}^{(k)} - G^{(k-m)}\mathscr{F}^{(k)})\left[(\mathscr{F}^{(k)})^T\mathscr{F}^{(k)}\right]^{-1}(\mathscr{F}^{(k)})^T, \tag{3.11}$$

which minimizes $\|G^{(k)} - G^{(k-m)}\|_F$ subject to $G^{(k)}\mathscr{F}^{(k)} = \mathscr{X}^{(k)}$. Here, $\mathscr{X}^{(k)}$ and $\mathscr{F}^{(k)}$ represent the differences of iterates and residuals, respectively, i.e.,

$$\mathscr{X}^{(k)} = [\Delta x^{(k-m)}, \ldots, \Delta x^{(k-1)}] \quad \text{and} \quad \mathscr{F}^{(k)} = [\Delta f^{(k-m)}, \ldots, \Delta f^{(k-1)}].$$

Hence, the update formula (3.10) can be written as (2.12)

$$x^{(k+1)} = x^{(k)} - G^{(k-m)} f^{(k)} - (\mathscr{X}^{(k)} - G^{(k-m)}\mathscr{F}^{(k)})\left[(\mathscr{F}^{(k)})^T\mathscr{F}^{(k)}\right]^{-1}(\mathscr{F}^{(k)})^T f^{(k)}.$$

When $G^{(k-m)} = -\beta I$, $\mathscr{X}^{(k)} = \mathscr{X}_A^{(k)}$ and $\mathscr{F}^{(k)} = \mathscr{F}_A^{(k)}$, (2.12) reduces to (2.33). Anderson Acceleration forms an approximate inverse of the Jacobian implicitly

$$G_A^{(k)} = -\beta I + (\mathscr{X}^{(k)} + \beta\mathscr{F}_A^{(k)})\left[(\mathscr{F}_A^{(k)})^T\mathscr{F}_A^{(k)}\right]^{-1}(\mathscr{F}_A^{(k)})^T,$$

that minimizes $\|G_A^{(k)} + \beta I\|_F$ subject to $G_A^{(k)} \mathscr{F}_A^{(k)} = \mathscr{X}_A^{(k)}$. Following the same notation, the updates of the CROP algorithm can be written as

$$x_C^{(k+1)} = \widetilde{x}_C^{(k+1)} - \mathscr{X}_C^{(k+1)} \left[(\mathscr{F}_C^{(k+1)})^T \mathscr{F}_C^{(k+1)} \right]^{-1} (\mathscr{F}_C^{(k+1)})^T \widetilde{f}_C^{(k+1)}. \tag{3.12}$$

With a common framework in place, we can now describe the connection between the CROP algorithm and the multisecant methods. First, if we consider $\widetilde{x}_C^{(k)}$ as an iterate, then we can view (3.12) as an update of a generalized Broyden's second method. When $G^{(k-m)} = 0$, $\mathscr{X}^{(k)} = \mathscr{X}_C^{(k)}$ and $\mathscr{F}^{(k)} = \mathscr{F}_C^{(k)}$, so (2.12) reduces to (3.12). The CROP algorithm forms implicitly an approximate inverse of the Jacobian

$$G_C^{(k+1)} = \mathscr{X}_C^{(k+1)} \left[(\mathscr{F}_C^{(k+1)})^T \mathscr{F}_C^{(k+1)} \right]^{-1} (\mathscr{F}_C^{(k+1)})^T,$$

that minimizes $\|G_C^{(k+1)}\|_F$ subject to $G_C^{(k+1)} \mathscr{F}_C^{(k+1)} = \mathscr{X}_C^{(k+1)}$.

3.3 Convergence Theory for CROP Algorithm

In this section, we provide some initial results on the convergence of the CROP algorithm. The first convergence result for Anderson Acceleration was given in [10], under the assumption of a contraction mapping. Following this work, several other convergence results utilizing various different assumptions were established; see for example [11, 14, 17, 18, 19]. Most of the existing assumptions are needed to determine the existence and uniqueness of the exact solution x^* of $f(x) = 0$ in an open set. Mappings f and g are usually chosen to be Lipschitz continuous. In this section, we use the same assumptions and follow the same process as the one in [10]. We first prove in Section 3.3.1 that for the linear case the CROP algorithm is q-linearly convergent. Then, in Section 3.3.2, we show that for the general nonlinear case,

the convergence is r-linear.

3.3.1 Convergence of CROP Algorithm for Linear Problems

We now examine the behavior of the CROP algorithm in the context of the simple linear system $Ax = b$, where $A \in \mathbb{R}^{n\times n}$ is nonsingular and $b \in \mathbb{R}^n$. Then, the associated residual (error) vectors take the form $f(x) = b - Ax$, and the corresponding fixed-point operator may be written as $g(x) = b + (I - A)x$. This linear framework provides a convenient starting point for analyzing the fundamental convergence properties of CROP. Let us present our first convergence result.

Theorem 3.10. *Let us consider solving the linear system $Ax = b$. If $\|I - A\|_2 = c < 1$, then the CROP algorithm converges to the exact solution $x^* = A^{-1}b$, and the control and real residuals converge q-linearly to zero with the q-factor c.*

Proof. Since

$$f_C^{(k+1)} = \sum_{i=0}^{m_C^{(k+1)}-1} \alpha_{C,i}^{(k+1)} f_C^{(k+1-m_C^{(k+1)}+i)} + \alpha_{C,m_C^{(k+1)}}^{(k+1)} \widetilde{f}_C^{(k+1)}$$

is the least-squares residual corresponding to (2.37), we have

$$\|f_C^{(k+1)}\|_2 \leq \|\widetilde{f}_C^{(k+1)}\|_2.$$

Moreover, by Lemma 3.4, we have $\widetilde{f}_C^{(k+1)} = (I - A)f_C^{(k)}$. Finally, Lemma 3.3 yields

$$\|f_C^{(k+1)}\|_2 \leq \|\widetilde{f}_C^{(k+1)}\|_2 \leq c\|f_C^{(k)}\|_2 \quad \text{and} \quad \|r_C^{(k+1)}\|_2 \leq c\|r_C^{(k)}\|_2, \tag{3.13}$$

which shows the q-linear convergence. □

Now, if we set $e^{(k)} = x^{(k)} - x^*$, then $f(x^{(k)}) = b - Ax^{(k)} = A\big(x^* - x^{(k)}\big) = -Ae^{(k)}$. Consequently, following Theorem 3.10 gives

$$(1-c)\|e^{(k)}\|_2 \leq \|f(x^{(k)})\|_2 \leq c^k\|f(x^{(0)})\|_2 \leq c^k(1+c)\|e^{(0)}\|_2.$$

Hence,

$$\|e^{(k)}\|_2 \leq \left(\frac{1+c}{1-c}\right) c^k\|e^{(0)}\|_2.$$

To fully understand the convergence of the CROP method for linear problems, we refer to the equivalence of CROP and GMRES, and CROP(m) and ORTHOMIN($m-1$).

Remark 3.11. The convergence of ORTHOMIN is studied in [24]. Note that CROP algorithm with no truncation, ORTHOMIN and GMRES are all mathematically equivalent methods.

3.3.2 Convergence of the CROP Algorithm for Nonlinear Problems

In the case of nonlinear problems, investigating the convergence of the fixed-depth CROP Algorithm 5 or CROP(m) algorithm, requires an assumption of the functions f and g to be "good enough". Let us consider the following assumption.

Assumption 3.12. *Consider a nonlinear fixed-point problem*

$$x = g(x),$$

or equivalently, the root-finding problem

$$f(x) = 0,$$

such that:

1. *there exists a point* x^* *satisfying* $f(x^*) = 0$ *and* $g(x^*) = x^*$*;*

2. *the function* g *is Lipschitz continuously differentiable in a ball* $\mathcal{B}_{\widehat{\rho}}(x^*)$ *for some* $\widehat{\rho} > 0$*, with Lipschitz constant* $c \in (0, 1)$*; and*

3. *the derivative* f' *is Lipschitz continuous with constant* $L > 0$*.*

In the case of no truncation, the equivalence between the CROP algorithm and Anderson Acceleration was established in Theorem 3.1. For the truncated CROP(m) algorithm, we can still try to explore the relation between the two methods. Let us investigate the connection between $\widetilde{x}_C$ and x_C, $\widetilde{f}_C$ and f_C. Note that when $m \neq k$, $\widetilde{x}_C$ and $\widetilde{f}_C$ are not the iterates and residuals of Anderson Acceleration. For CROP(m), we have

$$X_C^{(k)} = \widetilde{X}_C^{(k)} A_0 A_1 \cdot \ldots \cdot A_k \quad \text{and} \quad F_C^{(k)} = \widetilde{F}_C^{(k)} A_0 A_1 \cdot \ldots \cdot A_k,$$

with

$$X_C^{(k)} = \begin{bmatrix} x_C^{(0)} & \cdots & x_C^{(k)} \end{bmatrix} \quad \text{and} \quad F_C^{(k)} = \begin{bmatrix} f_C^{(0)} & \cdots & f_C^{(k)} \end{bmatrix},$$

and A_i is the identity matrix with the $(i+1)$-th column changed to the coefficient vector $\alpha_C^{(i)} = \left[\alpha_{C,0}^{(i)}, \ldots, \alpha_{C,m_C^{(i)}}^{(i)}\right]^T$ inserted in the range of rows starting at index $(i - m_C^{(i)} + 1)$ and

ending at index $(i+1)$, i.e.,

$$A_0 = I, \quad A_1 = \begin{bmatrix} 1 & \alpha_{C,0}^{(1)} & 0 & \cdots \\ 0 & \alpha_{C,1}^{(1)} & 0 & \cdots \\ 0 & 0 & 1 & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}, \quad A_k = \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 \\ \vdots & \ddots & \cdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & \alpha_{C,0}^{(k)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & \alpha_{C,m_C^{(k)}-1}^{(k)} \\ 0 & 0 & \cdots & 0 & \alpha_{C,m_C^{(k)}}^{(k)} \end{bmatrix}, k = 0, 1, \ldots$$

Since matrices A_i have column sum 1, $x_C^{(k)}$ and $f_C^{(k)}$, as the last columns of $X_C^{(k)}$ and $F_C^{(k)}$, respectively, can be written as

$$x_C^{(k)} = \sum_{j=0}^{k} s_j^{(k)} \widetilde{x}_C^{(k)} \quad \text{and} \quad f_C^{(k)} = \sum_{j=0}^{k} s_j^{(k)} \widetilde{f}_C^{(k)}, \quad \text{with} \quad \sum_{j=0}^{k} s_j^{(k)} = 1.$$

Now, we impose the following assumption that will allow us to establish the convergence result for CROP(m) along the lines of [10, Theorem 2.3].

Assumption 3.13. *For all $k > 0$, there exists $M > 0$ such that $\sum_{j=0}^{k} |s_j^{(k)}| < M$.*

Let us consider Assumption 3.12 again. From the Lipschitz condition for the function g, we have $\|g(x) - g(x^*)\|_2 \leq c\|x - x^*\|_2$ or equivalently $\|f(x) - (x - x^*)\|_2 \leq c\|x - x^*\|_2$. By the triangle inequality we get

$$(1-c)\|x - x^*\|_2 \leq \|f(x)\|_2 \leq (1+c)\|x - x^*\|_2. \tag{3.14}$$

Moreover, by the Lipschitz condition on f', there exists $\rho > 0$ sufficiently small such that in

the ball $\mathcal{B}_\rho(x^*)$ we have

$$\|f(x) - f'(x^*)(x - x*)\|_2 \leq \frac{L}{2}\|x - x^*\|_2^2, \tag{3.15}$$

which can be written in terms of g, as

$$\|g(x) - g'(x^*)(x - x*) - x^*\|_2 \leq \frac{L}{2}\|x - x^*\|_2^2. \tag{3.16}$$

The assumptions above provide the regularity conditions necessary for analyzing the local convergence behavior of the truncated CROP(m) algorithm. Assumption 3.12 ensures that the nonlinear mappings f and g possess sufficient smoothness and that a locally unique fixed point x^* exists, while Assumption 3.13 guarantees boundedness of the combination of coefficients that arise in the recursive construction of the CROP iterates. Together, these assumptions allow us to control the propagation of linearization errors and maintain stability of the residual updates within a neighborhood of the solution.

Building on these properties, we can now establish a local convergence result for the truncated CROP(m) algorithm and its CROP–Anderson(m) counterpart.

Theorem 3.14. *Let Assumption 3.12 and Assumption 3.13 hold and let $c < \widehat{c} < 1$. Then, if $x_{C,m}^{(0)}$ is sufficiently close to x^*, the control residual $f_C^{(k)}$ of CROP(m) and CROP-Anderson(m) and the residual $f_{CA}^{(k)} = \widetilde{f}_C^{(k)}$ satisfy*

$$\|f_C^{(k)}\|_2 \leq \widehat{c}^k \|f_C^{(0)}\|_2 \quad \text{and} \quad \|f_{CA}^{(k)}\|_2 \leq \widehat{c}^k \|f_{CA}^{(0)}\|_2.$$

Moreover, this implies r-linear convergence of CROP(m) and CROP-Anderson(m) with r-factor no greater than $\widehat{c}$.

Proof. Since

$$\lim_{\rho\to 0}\frac{2(1-c)c\widehat{c}^k+ML\rho}{2(1-c)-L\rho}=c\widehat{c}^k<\widehat{c}^{k+1},$$

we can choose ρ small enough such that

$$\frac{2(1-c)c\widehat{c}^k+ML\rho}{2(1-c)-L\rho}\leq\widehat{c}^{k+1}.$$

Suppose $x_C^{(0)}\in\mathcal{B}_\rho(x^*)$, and $x_C^{(0)}$ is sufficiently close to x^* such that

$$\frac{M(c+L\rho/2)}{1-c}\|f(x_C^{(0)})\|_2\leq\frac{M(1+c)(c+L\rho/2)}{1-c}\|x_C^{(0)}-x^*\|_2\leq\rho.$$

Then, we can prove by induction on k that $\|f(\widetilde{x}_C^{(k)})\|_2\leq\widehat{c}^k\|f(x_C^{(0)})\|_2$ and $\widetilde{x}_C^{(k)}\in\mathcal{B}_\rho(x^*)$ for all $k\geq 0$. For $k=0$, the conditions are satisfied because $\widetilde{x}_C^{(0)}=x_C^{(0)}$. Now, assume that for all $n\leq k$, we have $\|f(\widetilde{x}_C^{(n)})\|_2\leq\widehat{c}^k\|f(x_C^{(0)})\|_2$ and $\widetilde{x}_C^{(n)}\in\mathcal{B}_\rho(x^*)$. By (3.16)

$$g(\widetilde{x}_C^{(n)})=x^*+g'(x^*)(\widetilde{x}_C^{(n)}-x^*)+\Delta^{(n)},$$

with $\|\Delta^{(n)}\|_2\leq\frac{L}{2}\|\widetilde{x}_C^{(n)}-x^*\|_2$. Then, for $n=k+1$,

$$\begin{aligned}\widetilde{x}_C^{(k+1)}&=x^*+\sum_{j=0}^{k}s_j^{(k)}\big(g'(x^*)(\widetilde{x}_C^{(j)}-x^*)+\Delta^{(j)}\big)\\&=x^*+\sum_{j=0}^{k}s_j^{(k)}g'(x^*)(\widetilde{x}_C^{(j)}-x^*)+\sum_{j=0}^{k}s_j^{(k)}\Delta^{(j)}.\end{aligned}$$

Since $\big\|\sum_{j=0}^{k}s_j^{(k)}\Delta^{(j)}\big\|_2=\sum_{j=0}^{m_C^{(k)}}\big|s_j^{(k)}\big|\frac{L}{2}\big\|\widetilde{x}_C^{(j)}-x^*\big\|_2^2$, following (3.14) we get

$$\|\widetilde{x}_C^{(j)}-x^*\|_2<\frac{1}{1-c}\|f(\widetilde{x}_C^{(j)})\|_2\leq\frac{1}{1-c}\|f(x_C^{(0)})\|_2.$$

Then $\|\widetilde{x}_C^{(j)} - x^*\|_2 < \rho$ yields $\|\widetilde{x}_C^{(j)} - x^*\|_2^2 \leq \frac{\rho}{1-c}\|f(x_C^{(0)})\|_2$. Under Assumption 3.13, we have

$$\begin{aligned}\|\widetilde{x}_C^{(k+1)} - x^*\|_2 &= \Big\| \sum_{j=0}^{k} s_j^{(k)} g'(x^*)(\widetilde{x}_C^{(j)} - x^*) + \sum_{j=0}^{k} s_j^{(k)} \Delta^{(j)} \Big\|_2 \\ &\leq \frac{M(c + L\rho/2)}{1-c}\|f(x_C^{(0)})\|_2 \leq \rho.\end{aligned}$$

Thus, $\widetilde{x}_C^{(k+1)} \in \mathcal{B}_\rho(x^*)$ and by (3.15) we obtain $f(\widetilde{x}_C^{(k+1)}) = f'(x^*)(\widetilde{x}_C^{(k+1)} - x^*) + \Delta^{(k+1)}$, with $\|\Delta^{(k+1)}\|_2 \leq \frac{L}{2}\|\widetilde{x}_C^{(k+1)} - x^*\|_2^2 \leq \frac{L\rho}{2(1-c)}\|f(\widetilde{x}_C^{(k+1)})\|_2$. Since $f'(x^*) = g'(x^*) - I$ and $g'(x^*)$ commute,

$$\begin{aligned}f(\widetilde{x}_C^{(k+1)}) &= f'(x^*) \sum_{j=0}^{k} s_j^{(k)} g'(x^*)(\widetilde{x}_C^{(j)} - x^*) + f'(x^*) \sum_{j=0}^{k} s_j^{(k)} \Delta^{(j)} + \Delta^{(k+1)} \\ &= g'(x^*) \sum_{j=0}^{k} s_j^{(k)} f'(x^*)(\widetilde{x}_C^{(j)} - x^*) + f'(x^*) \sum_{j=0}^{k} s_j^{(k)} \Delta^{(j)} + \Delta^{(k+1)} \\ &= g'(x^*) \sum_{j=0}^{k} s_j^{(k)} (f(\widetilde{x}_C^{(k)}) - \Delta^{(j)}) + f'(x^*) \sum_{j=0}^{k} s_j^{(k)} \Delta^{(j)} + \Delta^{(k+1)} \\ &= g'(x^*) \sum_{j=0}^{k} s_j^{(k)} f(\widetilde{x}_C^{(j)}) - \sum_{j=0}^{k} s_j^{(k)} \Delta^{(j)} + \Delta^{(k+1)} \\ &= g'(x^*) f_C^{(k+1)} - \sum_{j=0}^{k} s_j^{(k)} \Delta^{(j)} + \Delta^{(k+1)}.\end{aligned}$$

Now, from $\|f_C^{(k)}\|_2 \leq \|\widetilde{f}_C^{(k)}\|_2 \leq \widehat{c}^k \|f(x_C^{(0)})\|_2$, we have

$$\|f(\widetilde{x}_C^{(k+1)})\|_2 \leq \|g'(x^*) f_C^{(k)}\|_2 + \|\sum_{j=0}^{k} s_j^{(k)} \Delta^{(j)}\|_2 + \|\Delta^{(k+1)}\|_2,$$

which finally yields $\|f(\widetilde{x}_C^{(k+1)})\|_2 \leq \dfrac{2(1-c)c\widehat{c}^k + ML\rho}{2(1-c) - L\rho}\|f(x_C^{(0)})\|_2 \leq \widehat{c}^{k+1}\|f(x_C^{(0)})\|_2$. □

Although the control residuals $f_C^{(k)}$ are not the real residuals $r_C^{(k)}$, which may cause a break-

down ($f_C^{(k)} = 0$) without finding a reasonable approximation, they do have some good properties.

Theorem 3.15. *The control residuals $f_C^{(k)}$ of CROP(m) for $m \geq 1$ are nonincreasing in norm.*

Proof. Since the coefficients $\alpha_{C,i}^{(k+1)}$ in

$$f_C^{(k+1)} = \sum_{i=0}^{m_C^{(k+1)}-1} \alpha_{C,i}^{(k+1)} f_C^{(k+1-m_C^{(k+1)}+i)} + \alpha_{C,m_C^{(k+1)}}^{(k+1)} \widetilde{f}_C^{(k+1)} \tag{3.17}$$

are chosen to minimize $\|f_C^{(k+1)}\|_2$, and $f_C^{(k)}$ is itself an element in the sum on the right of (3.17), we must have $\|f_C^{(k+1)}\|_2 \leq \|f_C^{(k)}\|_2$. □

In the case of smaller values of m, the control residuals are approximating better the real residuals. Also, using CROP-Anderson or rCROP is a good way to avoid breakdown. We will see some examples in Section 3.4.

3.4 Numerical Experiments

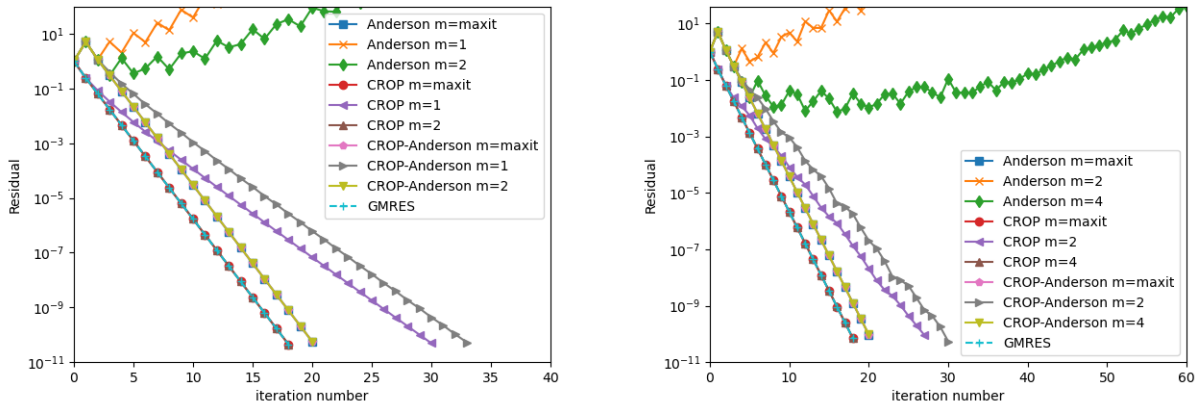
In this section, we present a small selection of numerical results illustrating some observations discussed in the previous sections. We consider both linear and nonlinear problems of various sizes.

3.4.1 Linear Problems

First, we present numerical experiments to demonstrate the behavior of the discussed methods when applied to a linear system $Ax = b$ with a nonsingular matrix A.

Example 3.16 (Synthetic Problem)**.**

Let us consider a nonsingular matrix $A \in \mathbb{R}^{100\times 100}$ given as a tridiagonal matrix with entries $(1, -4, 1)$ and a seven-diagonal matrix with entries $(0, 0, 1, -4, 1, 1, 1)$, and vector $b \in \mathbb{R}^{100}$ with its first entry 1 and others 0. We choose $f(x) = b - Ax$, $g(x) = x + f(x)$, $maxit = 100$, $tol = 10^{-10}$, and run Anderson Acceleration, CROP and CROP-Anderson with initial vector $x_0 = 0$ and different values of parameter m. The corresponding results are shown in Figure 3.3. In Figure 3.3a we can see that the convergence curve for the CROP-Anderson



(a) symmetric tridiagonal A (b) nonsymmetric seven-diagonal A

Figure 3.3: A linear problem in Example 3.16 with (a) a tridiagonal and (b) a seven-diagonal matrix A.

method is parallel to that for the CROP algorithm. As we discussed in Section 3.3.1, CROP, CROP(2) and GMRES admit the same convergence similarly as Anderson Acceleration, CROP-Anderson and CROP-Anderson(2). Analogous behavior can be observed for CROP, CROP(4) and GMRES, as well as for Anderson Acceleration, CROP-Anderson algorithm and CROP-Anderson(4), see Figure 3.3b.

Example 3.17 (Stagnation Problem)**.**

Let us consider a nonsingular matrix $A \in \mathbb{R}^{10\times 10}$ given by $\texttt{A} = \texttt{diag}(\texttt{1} : \texttt{10}) - \texttt{ones}(\texttt{10})$, and

vector $b \in \mathbb{R}^{10}$ with its first entry 1 and others 0. We choose $f(x) = b - Ax$, $g(x) = x + f(x)$, $maxit = 10$, $tol = 10^{-15}$, and run Anderson Acceleration, CROP and CROP-Anderson with initial vector $x_0 = 0$ and no truncation ($m_k = \infty$). The GMRES algorithm for this problem, along with the same initial vector, is also included for comparison. The corresponding results are shown in Figure 3.4. In Figure 3.4, we can see that GMRES and CROP stagnate

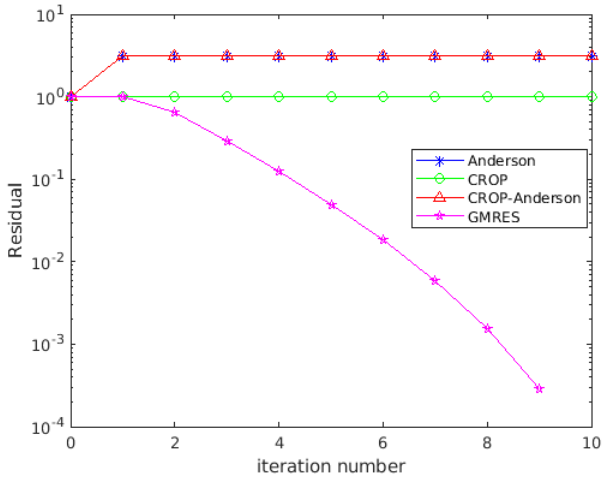


Figure 3.4: Convergence for the problem in Example 3.17.

in the first step, while Anderson and CROP-Anderson stagnate in the second step. As we discussed in Section 3.3.1, GMRES makes progress after stagnation, while Anderson, CROP, and CROP-Anderson make no further progress.

Example 3.18 (A Real World Problem)**.**

Let us now consider the same linear problem $Ax = b$ with a symmetric positive definite matrix `cfd1` from the SuiteSparse Matrix Collection [113] and vector $b \in \mathbb{R}^{70656}$ with its first entry 1 and others 0. The corresponding convergence results for different methods are shown in Figure 3.5. Notice that choice of $m = 2$ is the optimal for both CROP and the CROP-Anderson algorithm.

Example 3.19 (A Small Linear Problem)**.**

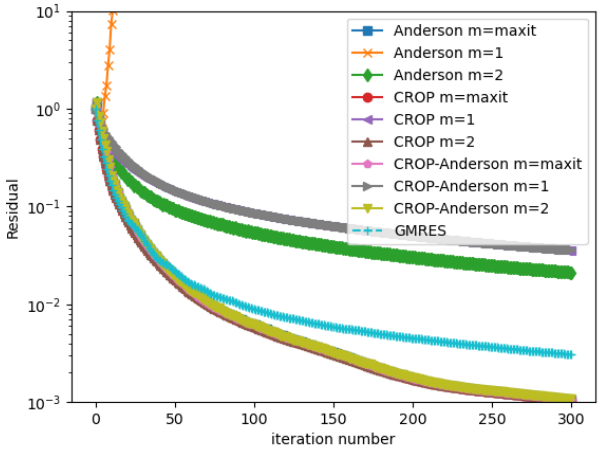


Figure 3.5: Convergence for a linear problem in Example 3.18.

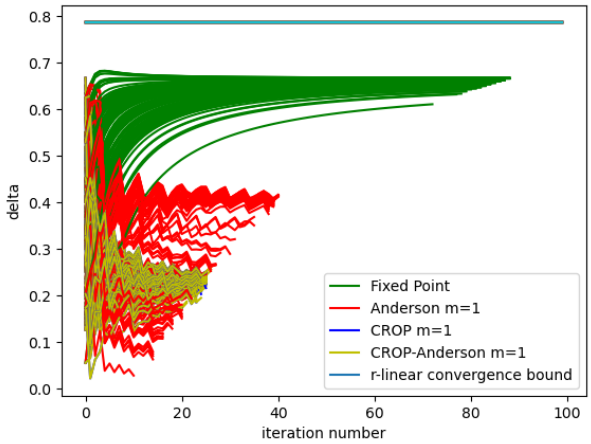


Figure 3.6: r-linear convergence factor for a small linear system in Example 3.19.

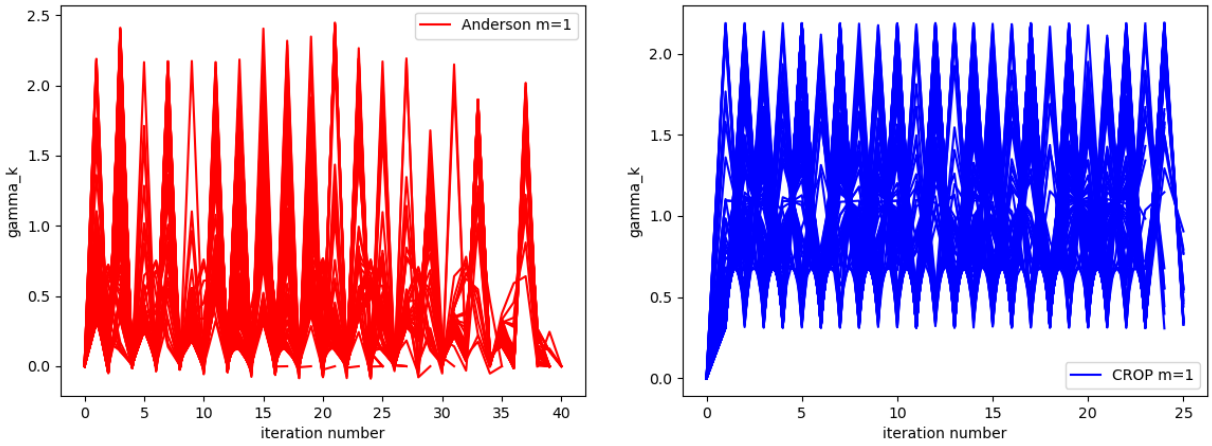
In this example, we consider a linear system $Ax = b$ [12, 13, Problem 1] with

$$A = \begin{bmatrix} 1/3 & -1/4 \\ 0 & 2/3 \end{bmatrix} \in \mathbb{R}^{2\times 2} \quad \text{and} \quad b = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \in \mathbb{R}^2. \tag{3.18}$$

We choose $f := b - Ax$ and $g := x + f = Mx$ with

$$M = \begin{bmatrix} 2/3 & 1/4 \\ 0 & 1/3 \end{bmatrix} \in \mathbb{R}^{2\times 2}.$$

We set $maxit = 100$, $tol = 10^{-16}$ and run fixed-point iteration, Anderson Acceleration, CROP, and CROP-Anderson algorithms with $m = 1$. The components $x_1^{(0)}$ and $x_2^{(0)}$ of the initial vector $x^{(0)}$ are chosen randomly between $[-0.5, 0.5]$. The r-linear convergence factors are shown in Figure 3.6. The damping parameters $\gamma^{(k)}$ for Anderson and CROP/CROP-Anderson are presented in Figures 3.7a and 3.7b. Note that they have different oscillation behaviors. For Anderson Acceleration, $\gamma^{(k)}$ displays the same oscillation pattern independent on the choice of initial vector, whereas two different kinds of oscillations are occurring for



(a) Anderson Acceleration (b) CROP Algorithm

Figure 3.7: γ_k for (a) Anderson Acceleration and (b) CROP Algorithm in Example 3.19.

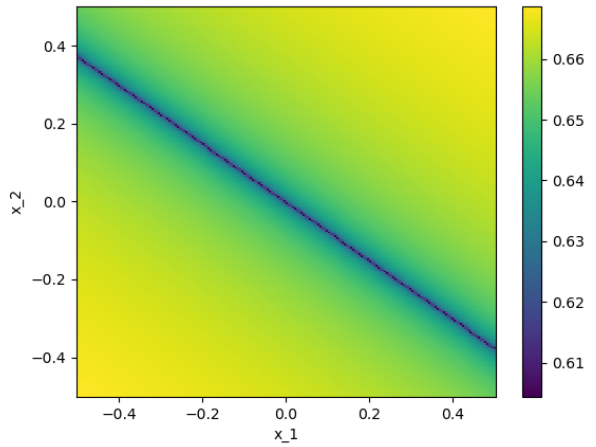
the CROP algorithm. We show the r-linear convergence factors of fixed-point iteration, Anderson Acceleration, CROP, and CROP-Anderson in Figures 3.8a, 3.8b, 3.8c, and 3.8d. We can see that the r-linear convergence factor is a function of the angle $\arctan \frac{x_2^{(0)}}{x_1^{(0)}}$. The factors for CROP and CROP-Anderson oscillate much faster than the factor for Anderson Acceleration.

3.4.2 Nonlinear Problems

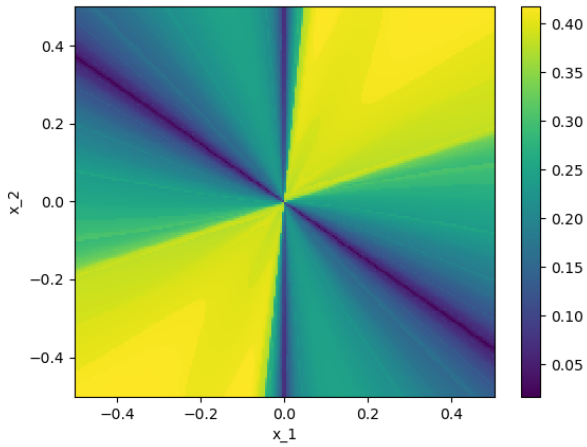
As the presented methods are primarily used to accelerate convergence in the case of nonlinear problems, in this section we introduce a variety of nonlinear examples. Starting with small and weakly nonlinear problems, we move towards more complex examples.

Example 3.20 (Dominant Linear Part Problem)**.**

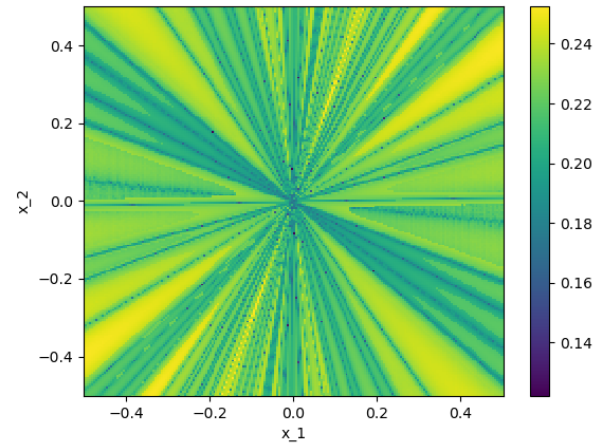
Consider a nonlinear problem $Ax + \frac{\mu\|x\|_2^2}{n}x = b$ with $A = tridiag(1, -4, 1) \in \mathbb{R}^{n\times n}$, a right-hand side vector $b \in \mathbb{R}^n$ with first entry 1 and others 0, $n = 100$ and parameter $\mu = 1/100$.



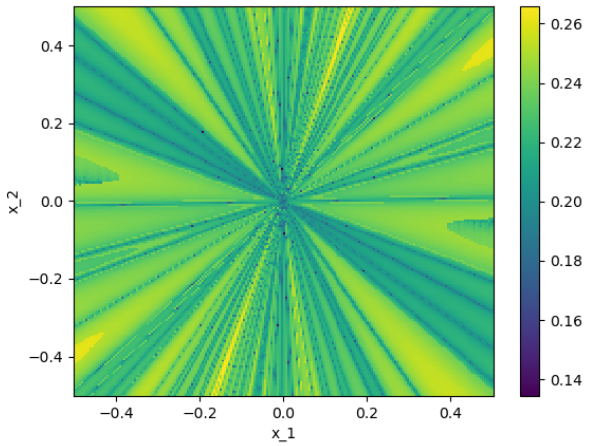
(a) Fixed-point Iteration



(b) Anderson Acceleration



(c) CROP Algorithm



(d) CROP-Anderson Algorithm

Figure 3.8: r-linear convergence factors in Example 3.19.

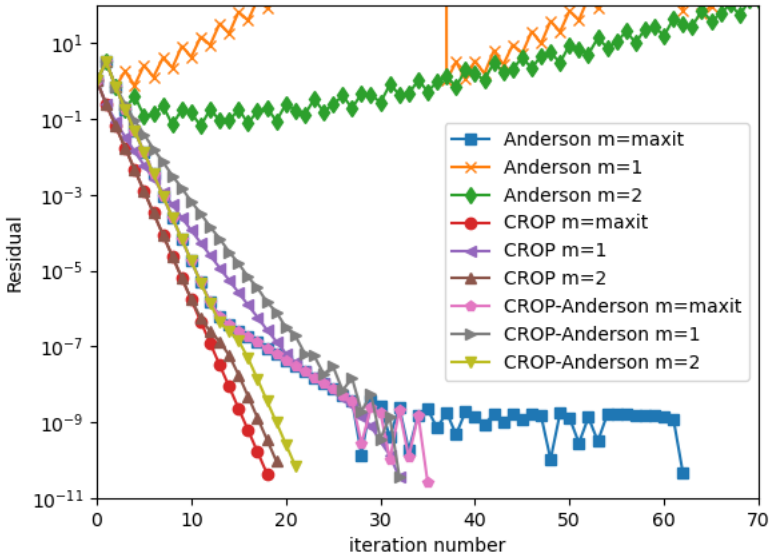


Figure 3.9: Convergence for a nonlinear problem in Example 3.20.

We choose $f(x) = Ax + \frac{\mu\|x\|_2^2}{n}x - b$ and $g(x) = x + f(x)$, parameters $maxit = 100$ and $tol = 10^{-10}$, and run Anderson Acceleration, CROP and CROP-Anderson with initial vector $x^{(0)} = 0$ and different values of parameter m. Figure 3.9 presents our findings. We can see that the first several steps of all methods are controlled by the linear part of the problem and can be compared with the results displayed in Figure 3.3a. The difference occurs when the residual reaches $\approx 10^{-6}$. We can see that CROP, CROP(2), and CROP-Anderson(2) converge in $18, 19$ and 21 steps, respectively, which is better than the other methods. It is worth mentioning that for the CROP algorithm, although the control residuals get below the tolerance, the real residuals may not. Actually, in this example, $\|r^{(18)}_{CROP}\|_2 = 6.28 \cdot 10^{-8}$, $\|r^{(19)}_{CROP(2)}\|_2 = 9.56 \cdot 10^{-11}$ and $\|r^{(32)}_{CROP(1)}\|_2 = 5.19 \cdot 10^{-11}$. In the case of CROP with small values of m, the control residuals are good approximations of the real residuals. However, this does not have to be the case for the larger values of m. We will see some further disadvantages of the control residuals in Example 3.21.

Example 3.21 (A Small Nonlinear Problem)**.**

Consider problem (1.1) [12, 13, Problem 2] with

$$g\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = \frac{1}{2}\begin{bmatrix} x_1 + x_1^2 + x_2^2 \\ x_2 + x_1^2 \end{bmatrix} \quad \text{and exact solution} \quad \begin{bmatrix} x_1^* \\ x_2^* \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

The fixed-point function g is Lipschitz continuous and the spectral radius of J_g near x^* is smaller than 1, thus convergent. If we run Anderson Acceleration, CROP and CROP-Anderson with $m = 1, 2$ on this problem, they should all have good behavior. We set $maxit = 100$, $tol = 10^{-10}$ and run fixed-point iteration, Anderson Acceleration, CROP and CROP-Anderson with $x^{(0)} = [0.1, 0.1]^T$ and different values of the parameter m.

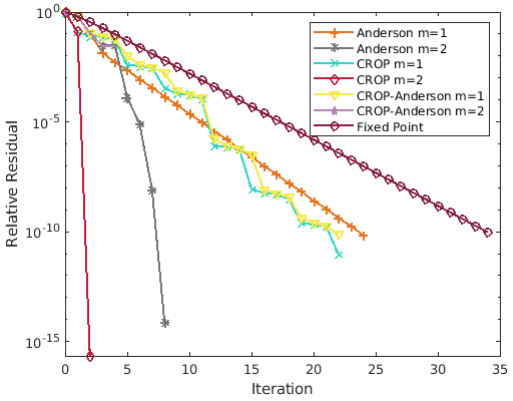
The corresponding results are shown in Figure 3.10. Note that CROP(2) and CROP-Anderson(2) break down in iteration 2, but rCROP and rCROP-Anderson converge well. rCROP(1) and rCROP(2) converge in 4 iterations, while Anderson(2) converges in 8 and Anderson(1) in 24 iterations. Even though using real residuals requires additional function evaluation in each iteration, Example 3.21 illustrates that this extra cost is worthy, especially for the case of $m = 1$. In Figure 3.10 we can also check the convergence result for CROP and CROP-Anderson by comparing them with fixed-point iteration. Since CROP and CROP-Anderson are always bounded by the fixed-point iteration, they satisfy Theorem 3.14. Also, we can see that the Anderson Acceleration methods satisfy [10, Theorem 2.3].

Example 3.22 (Bratu Problem)**.**

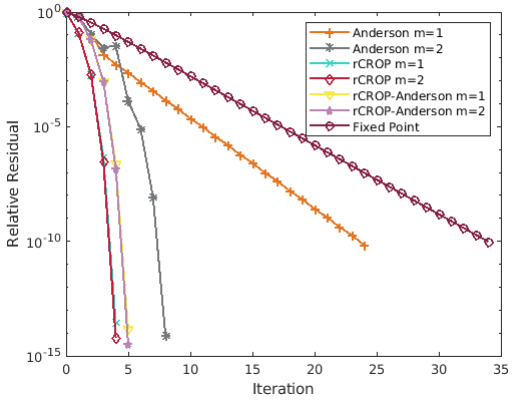
In this example, we consider the Bratu Problem [23, Section 5.1]

$$\begin{aligned} \Delta u + \lambda e^u = 0 \text{ in } \Omega = (0,1) \times (0,1) \\ u(y,z) = 0 \quad \text{for} \quad (y,z) \in \partial\Omega \end{aligned}. \tag{3.19}$$

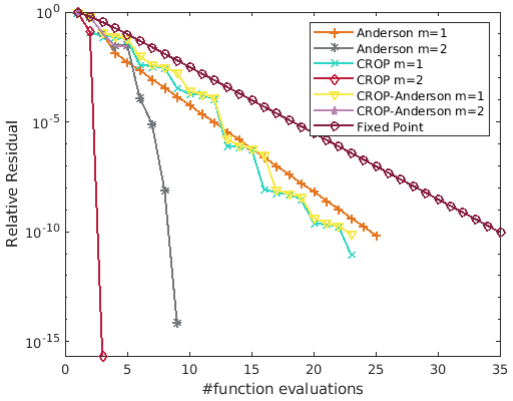
Using a finite difference method with grid size 100×100, the problem becomes $Lx +$



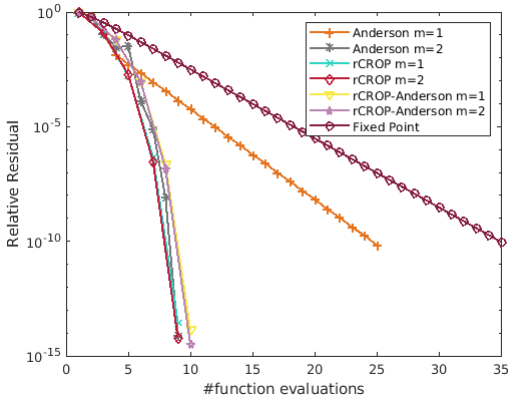
(a) iterations



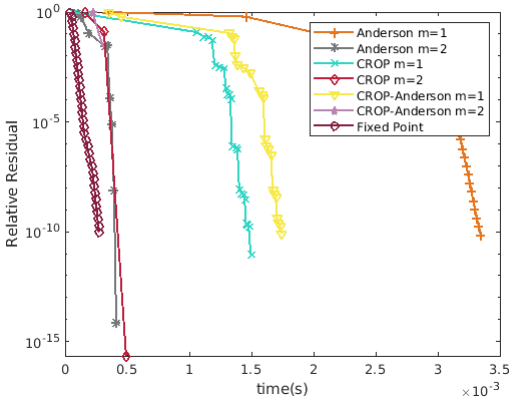
(b) iterations



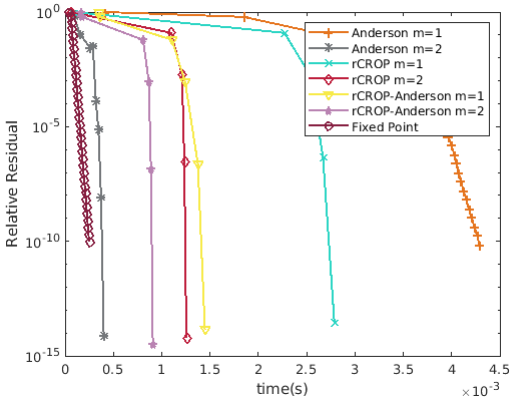
(c) function evaluations



(d) function evaluations



(e) time



(f) time

Figure 3.10: A small nonlinear problem in Example (3.21) with linear (left) and real (right) residuals for iterations (top) and function (middle) evaluations and time (bottom).

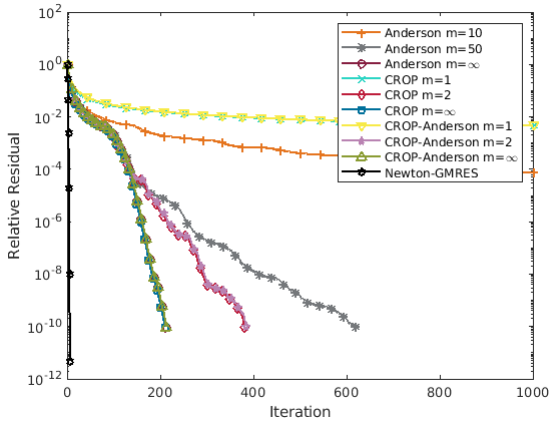
$h^2\lambda\exp(x) = 0$, where L is the 10000×10000 2D Laplace matrix and $h = 1/101$. We choose $\lambda = 0.5$, $f(x) = Lx + h^2\lambda\exp(x)$, $g(x) = x + f(x)$ as well as parameters $maxit = 400$, $tol = 10^{-10}$, $x^{(0)} = 0$. Since Anderson Acceleration with $m = 1, 2$ does not converge for this problem, we run Anderson Acceleration with $m = 10, 50, \infty$, and CROP and CROP-Anderson with $m = 1, 2, \infty$. For comparison, we also run the Newton-GMRES method, with the inner GMRES maximum iteration set to 120. Since the Bratu problem (3.19) is symmetric and has small nonlinearities, the rCROP results are almost the same as those of the CROP algorithm, and confirm that $m = 2$ is a good choice of the truncation parameter. Comparing CROP and rCROP results in Figure 3.11, we can see that function evaluation is not the time-consuming part of this example. As shown in the time plot, most of the computational cost for this problem is dominated by solving the least-squares subproblems, making methods with smaller m more efficient overall. For instance, consider the performance of the rCROP(m) method with $m = 1, 2, 3, 5, 10, 20$. Figure 3.12 shows that increasing m generally reduces the number of iterations required for convergence. However, except for $m = 1$, which converges too slowly, larger values of m lead to longer total runtimes due to the higher cost per iteration. The optimal balance between convergence rate and computational cost is achieved with $m = 2$.

Example 3.23 (The Lennard-Jones Problem)**.**

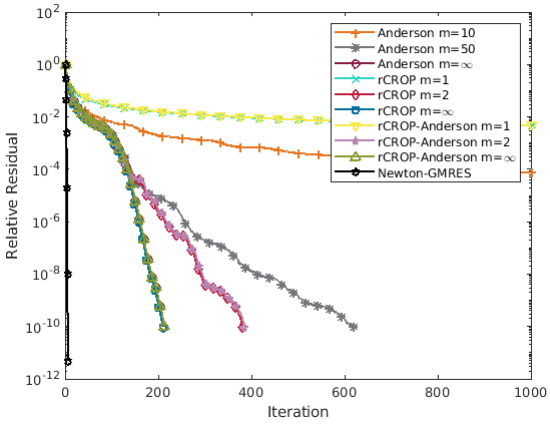
This problem originates from molecular optimization, where the goal is to determine atom position in a given molecule in a way that minimizes the total potential energy

$$E(Y) = 4\sum_{i=1}^{N}\sum_{j=1}^{i-1}\left(\frac{1}{\|y_i - y_j\|_2^{12}} - \frac{1}{\|y_i - y_j\|_2^{6}}\right), \tag{3.20}$$

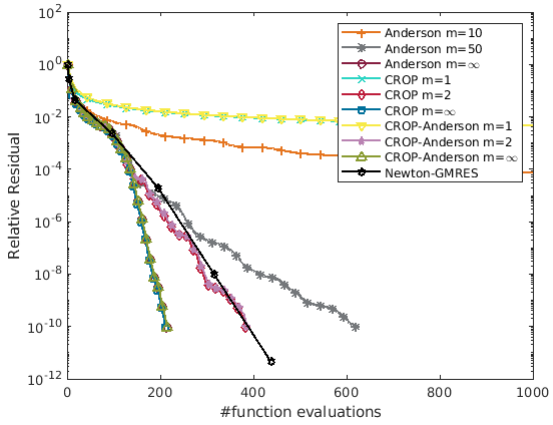
described as the Lennard-Jones Potential, where $Y = [y_1, \ldots, y_N] \in \mathbb{R}^{3\times N}$ is the matrix containing the 3-dimensional coordinate vectors $y_i \in \mathbb{R}^3$ of atom $i = 1, \ldots, N$. A local



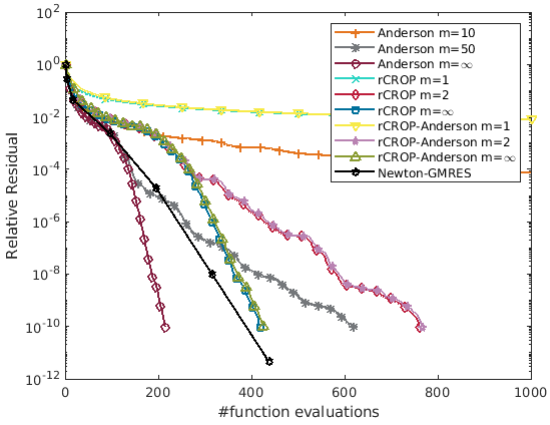
(a) iterations



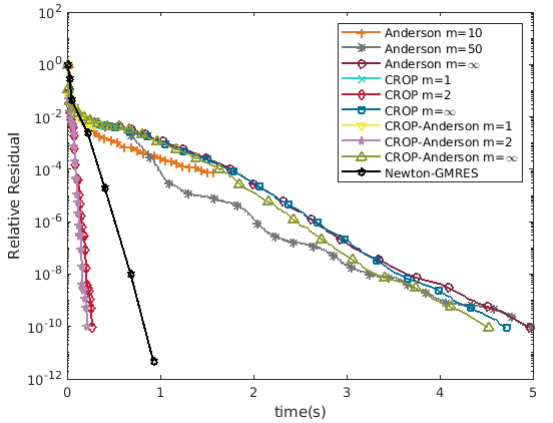
(b) iterations



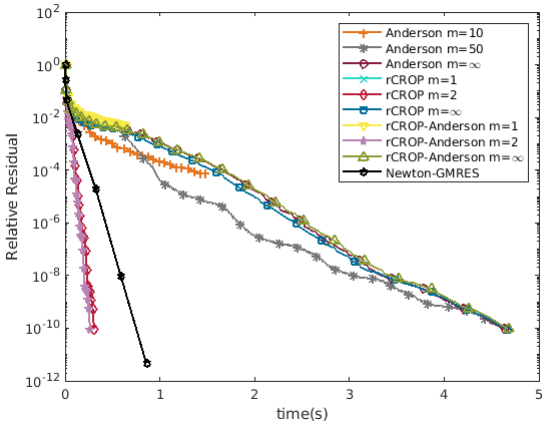
(c) function evaluations



(d) function evaluations

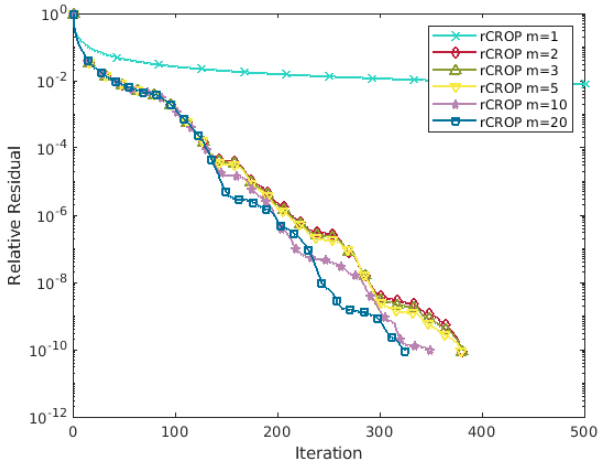


(e) time

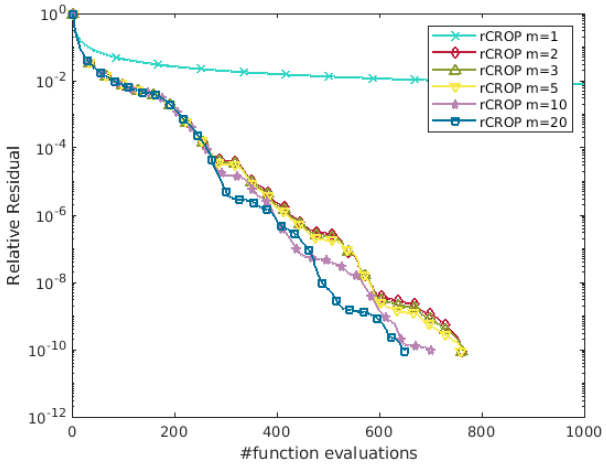


(f) time

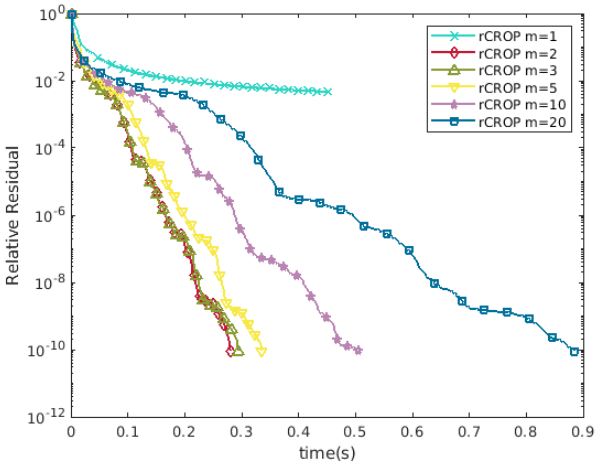
Figure 3.11: Bratu problem in Example (3.22) with linear (left) and real (right) residuals for iterations (top) and function (middle) evaluations and time (bottom).



(a) iterations



(b) function evaluations



(c) time

Figure 3.12: Bratu problem in Example (3.22) with real residuals (a) iterations and (b) function evaluations and (c) time.

minimizer of (3.20) can be computed by finding roots of

$$f(x) = -\nabla E(\mathrm{vec}(Y)),$$

where $x \in \mathbb{R}^{3\cdot N}$ is the vectorization of Y. In this example, we took 27 unit cells and 4 particles per unit cell, so $N = 108$. We choose $maxit = 500$, $tol = 10^{-10}$, $g(x) = x + \beta f(x)$ with the damping parameter $\beta = 5 \times 10^{-4}$. Since the Lennard-Jones problem is nonlinear, we run Anderson Acceleration with $m = 10, 20, \infty$, and rCROP and rCROP-Anderson with $m = 1, 2, \infty$. We can see in Figure 3.13 that the CROP and CROP-Anderson methods with $m = 1, 2$ stagnate after just a few steps. The rCROP and rCROP-Anderson methods with $m = 2, \infty$ converge very well. rCROP(1) and rCROP-Anderson(1) are not very good methods for this problem. We can see $m = 2$ is a good choice and $m = 1$ is not. An interesting observation is that the time plot in Figure 3.13f looks very similar to the plot of numbers of function evaluations in Figure 3.13d. This shows that the time spent in this example is mainly on the function evaluations, which shows in the plot that the methods with $m = \infty$ spent less time than the truncated methods.

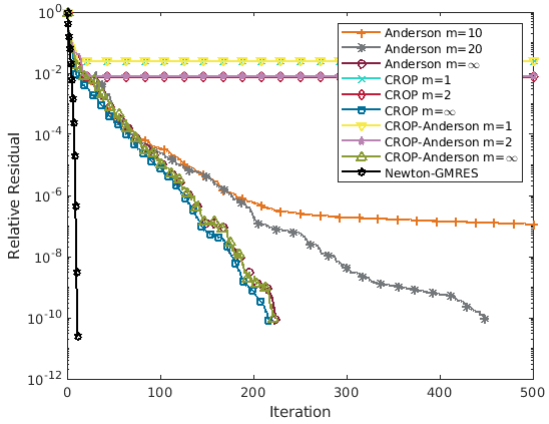
Example 3.24 (The Chandrasekhar H-equation)**.**

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

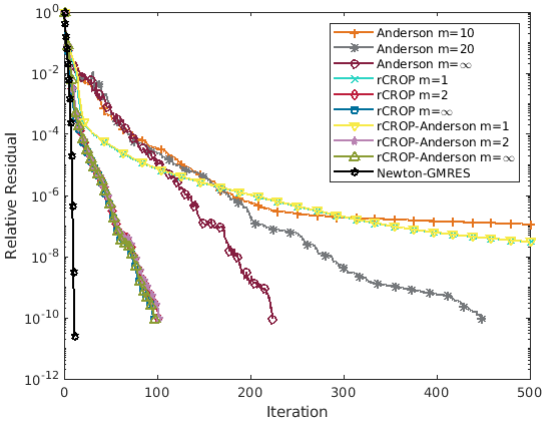
This problem is singular when $\omega = 1$. As stated in [114], using $\mu_i = (i - 1/2)/N$, this equation can be discretized with the midpoint rule as

$$F^N(h)_i = h_i - \left(1 - \frac{\omega}{2N}\sum_{i=1}^{N} \frac{(i-1/2)h_j}{i+j-1}\right)^{-1},$$

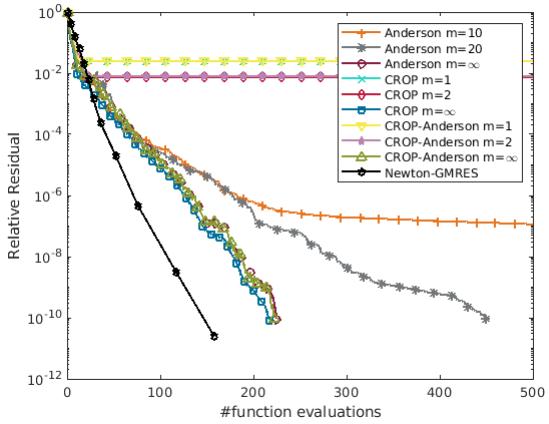
where the summation part can be calculated with the fast Fourier transform.



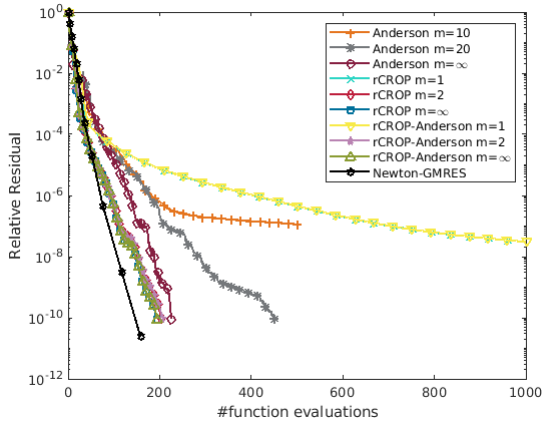
(a) iterations



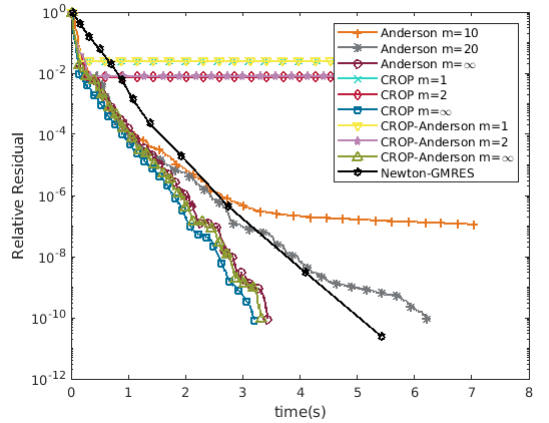
(b) iterations



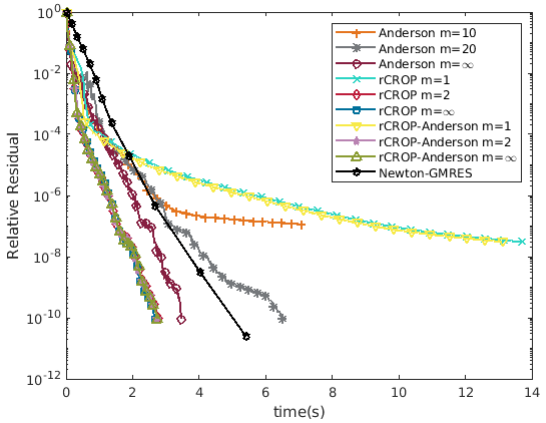
(c) function evaluations



(d) function evaluations



(e) time



(f) time

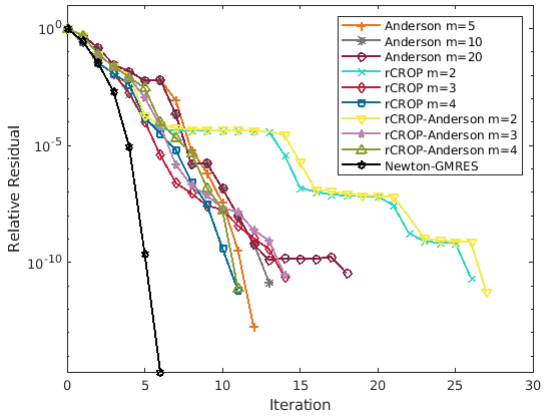
Figure 3.13: Lennard-Jones problem in Example (3.22) with linear (left) and real (right) residuals for iterations (top) and function (middle) evaluations and time (bottom).

In the experiment, we consider both the nonsingular case ($\omega = 0.99$) and the singular case ($\omega = 1$). We choose $maxit = 100$, $tol = 10^{-10}$, $g(x) = x + f(x)$, where x is a discretized function with $N = 1000$ points. Since the CROP method with small m's stagnate, we run Anderson Acceleration with $m = 5, 10, 20$, and rCROP and rCROP-Anderson with $m = 2, 3, 4$. We can see in Figure 3.14 that the rCROP and rCROP-Anderson methods with $m = 2$ stagnate after just a few steps. For the nonsingular case ($\omega = 0.99$), they can recover and finally converge, while for the singular case ($\omega = 1$) they remain divergent. rCROP and rCROP-Anderson methods with $m = 3, 4$ converge very well. For the nonsingular case ($\omega = 0.99$) $m = 4$ is better, while for the singular case ($\omega = 1$) $m = 3$ is better. Similar convergence results are observed about Anderson Acceleration: smaller m leads to better convergence for the singular case of the problem. The result of Newton-GMRES shows the difficulties of the singular cases, for it converges quadratically in the nonsingular case and linearly in the singular case.

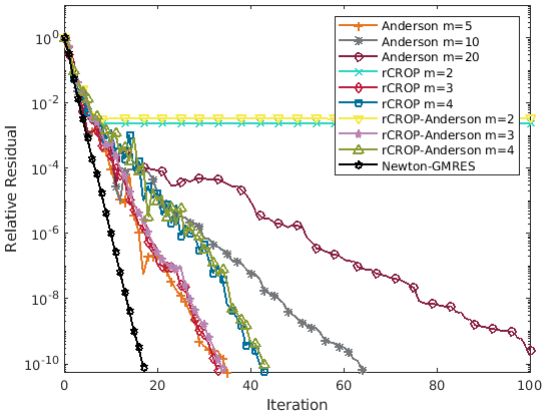
3.5 Conclusions

In this chapter, we conducted a comprehensive investigation of the CROP algorithm and its relationship to Anderson Acceleration and other well-known iterative methods. Our analysis revealed that CROP provides a compelling alternative for both linear problems and nonlinear problems with relatively weak nonlinearities. While the original CROP method may struggle on highly nonlinear problems, our experiments demonstrate that its real-residual variant, rCROP, can match or even surpass the performance of Anderson Acceleration in such settings. These findings underscore the potential of CROP-based strategies as practical acceleration tools across a broad range of applications.

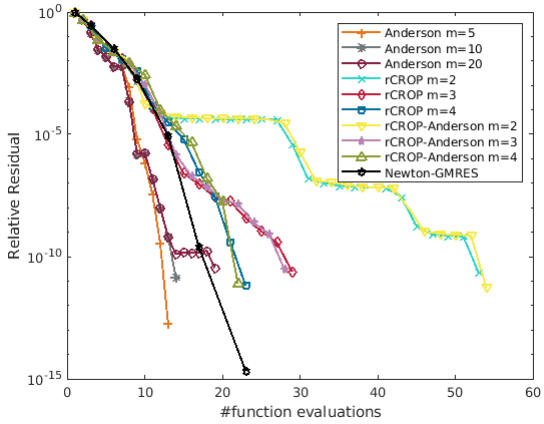
Our main contributions of this chapter can be summarized as follows:



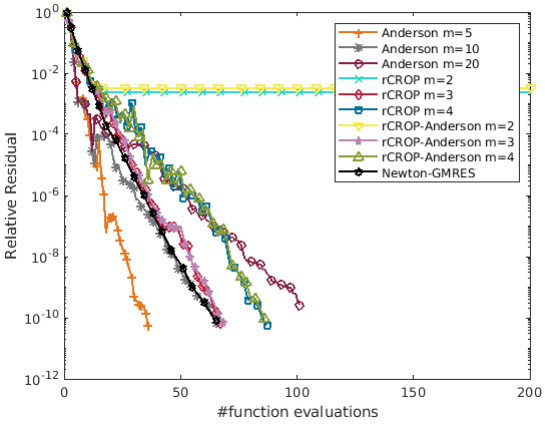
(a) iterations



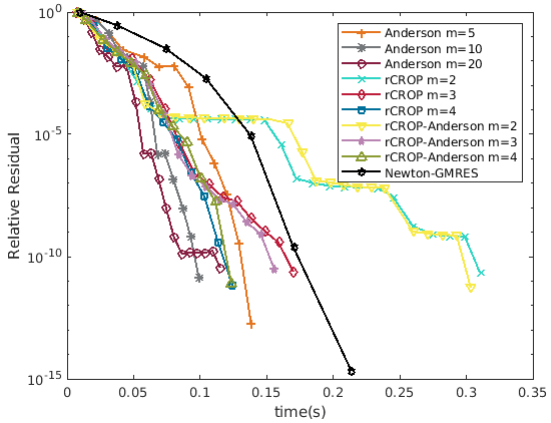
(b) iterations



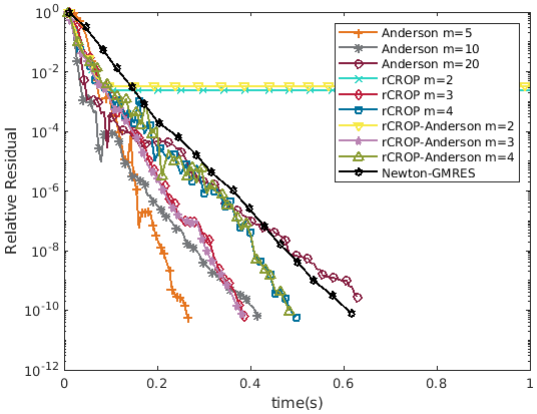
(c) function evaluations



(d) function evaluations



(e) time



(f) time

Figure 3.14: The Chandrasekhar H-equation in Example (3.24) with $\omega = 0.99$ (left) and $\omega = 1$ (right) for iterations (top) and function (middle) evaluations and time (bottom).

- We developed a unified Anderson-type framework for analyzing CROP and proved its equivalence to Anderson Acceleration in exact arithmetic.
- We situated CROP within the context of established methods, comparing it to Krylov subspace approaches for linear problems and to multisecant techniques in the nonlinear setting.
- We identified and clarified the structural parallels between CROP and Anderson Acceleration, leading to the introduction of the CROP–Anderson variant.
- We characterized problem classes in which CROP and CROP–Anderson offer advantages over classical Anderson Acceleration.
- We established convergence results for both linear and nonlinear versions of CROP and CROP–Anderson.
- We extended the methodology by incorporating real residuals, resulting in the rCROP and rCROP–Anderson algorithms, which exhibit improved robustness for nonlinear problems.
- We provided numerical experiments that validate the theoretical findings and highlight practical scenarios where CROP-based methods are particularly effective.

Overall, the results presented in this chapter lay a foundation for further theoretical and computational exploration of CROP and its variants. In particular, their application to challenging large-scale nonlinear problems—such as those arising in computational chemistry—remains a promising direction for future research.

Chapter 4

Robustness of Restarted and Adaptive Anderson Acceleration under Perturbations

In this chapter, we examine how Anderson-type acceleration methods, in particular the restarted and adaptive versions introduced in [17] and discussed in Section 2.6, behave when the underlying computations are performed inexactly. Our focus is on understanding the impact of such inexactness arising from approximate evaluations of the residual function f or the update mapping g on the convergence and robustness of the accelerated iteration. Through this analysis, we aim to characterize the sensitivity of Anderson Acceleration to perturbations and to provide insight into its performance in practical, error-prone computational settings.

The convergence results for Anderson Acceleration in the presence of deterministic and stochastic perturbations in the fixed-point mapping g, have been introduced in [115]. The presented analysis was motivated by the study of deterministic and stochastic errors arising in Jacobian-free Newton-Krylov (JFNK) methods [116, 117]. Following a similar line of reasoning and building on the convergence framework developed in [17], we extend this analysis to investigate the perturbation behavior of variants of Anderson Acceleration, such as restarted and adaptive Anderson Acceleration.

4.1 Deterministic and Stochastic Perturbations

We first consider perturbations that arise deterministically, for example, from discretization errors, truncation of series expansions, or approximate evaluations of nonlinear operators. Deterministic perturbations can often be modeled as bounded deviations $\epsilon(x)$, i.e.,

$$\|\epsilon(x)\|_2 \leq \epsilon_0,$$

for some constant $\epsilon_0 > 0$. Here and throughout, $\varepsilon(x)$ denotes a generic perturbation function (or vector), and ε_0 is its uniform upper bound; the same notation will be used consistently across all subsequent error models (absolute, relative, and stochastic).

In this case, the perturbed residual and iteration functions take the form

$$\widehat{f}(x) := f(x) + \epsilon(x), \quad \widehat{g}(x) := g(x) + \epsilon(x), \tag{4.1}$$

with $\|\epsilon(x)\|_2 \leq \epsilon_0$. The above expression defines an absolute error model.

Alternatively, one may consider perturbations that scale with the magnitude of the function being evaluated. We define the relative perturbations of the residual and the fixed-point map as $\varepsilon_f(x),\ \varepsilon_g(x) \in \mathbb{R}^n$, with $\|\varepsilon_f(x)\|_2 \leq \varepsilon_0\|f(x)\|_2$, $\|\varepsilon_g(x)\|_2 \leq \varepsilon_0\|g(x)\|_2$, for some $\varepsilon_0 > 0$. The corresponding perturbed functions are written as

$$\widehat{f}(x) := f(x) + \epsilon_f(x), \quad \widehat{g}(x) := g(x) + \epsilon_g(x),$$

Two common realizations of these perturbations are norm-based scaling

$$\varepsilon_f(x) = \|f(x)\|_2\varepsilon(x), \quad \varepsilon_g(x) = \|g(x)\|_2\varepsilon(x),$$

with $\|\varepsilon(x)\|_2 \leq \varepsilon_0$ and component-wise (Hadamard) scaling

$$\varepsilon_f(x) = f(x) \odot \varepsilon(x), \quad \varepsilon_g(x) = g(x) \odot \varepsilon(x),$$

where $\odot$ denotes elementwise multiplication and $\|\varepsilon(x)\|_\infty \leq \varepsilon_0$.

As $f(x) \to 0$, the relative error $\varepsilon_f(x)$ naturally decreases, whereas the perturbation $\varepsilon_g(x)$ behaves like an absolute error near the fixed point since $g(x) \to x^*$.

Stochastic perturbations, unlike deterministic ones, are inherently random and are often modeled as zero-mean noise. Such noise commonly arises in Monte Carlo simulations, stochastic optimization, or molecular dynamics sampling. Although the expected iterate may still converge to the true fixed point, the induced variance can produce random oscillations that slow convergence or potentially destabilize the iteration. In this case, the perturbed residual and iteration functions take the form

$$\begin{aligned} \widehat{f}(x, N_{MC}) &:= f(x) + \epsilon(x, N_{MC}), \\ \widehat{g}(x, N_{MC}) &:= g(x) + \epsilon(x, N_{MC}), \end{aligned} \tag{4.2}$$

where the noise $\epsilon(x, N_{MC})$ depends on the number of Monte Carlo samples N_{MC}, and can be absolute or relative form.

4.2 Perturbation Effects in Anderson-Type Acceleration

Acceleration techniques, including Anderson Acceleration and its variants, enhance convergence by leveraging information from previous iterates and residuals. However, this same

mechanism can increase the sensitivity to perturbations, as errors in residual evaluations may be amplified during the least-squares update. In this section, we analyze the convergence behavior of Anderson-type methods in the presence of both deterministic and stochastic perturbations. We conduct our convergence analysis of Anderson Acceleration under the general assumptions outlined below, following the framework established in [17].

Assumption 4.1. *Let V be an open subset of $\mathbb{R}^n$, Σ be a smooth submanifold in $\mathbb{R}^n$, $f \in C^2(V, \mathbb{R}^n)$, $g \in C^2(V, \Sigma)$ and $x^* \in V \cap \Sigma$. We assume the following:*

1. *There exists a constant K in $(0, 1)$ such that*

$$\|(f \circ g)(x)\|_2 \leq K\|f(x)\|_2, \quad \textit{for all} \quad x \in V \cap g^{-1}(V) \cap \Sigma.$$

2. *There exists a constant $\sigma > 0$ such that*

$$\sigma \|x - x_*\|_2 \leq \|f(x)\|_2, \quad \textit{for all} \quad x \in V \cap \Sigma.$$

3. *Let $U \subset V \cap g^{-1}(V)$ be a sufficiently small neighborhood of x^* such that, for all $x, y \in U$, the following hold:*

   (a) ***Lipschitz continuity of** f **and** $f \circ g$, i.e.,*

$$\|f(x) - f(y)\|_2 \leq 2\|\mathrm{D}f(x^*)\|_2 \, \|x - y\|_2,$$
$$\|(f \circ g)(x) - (f \circ g)(y)\|_2 \leq 2\|\mathrm{D}f(x^*) \circ \mathrm{D}g(x^*)\|_2 \, \|x - y\|_2,$$

   where $\mathrm{D}f(x^*)$ and $\mathrm{D}g(x^*)$ denote the differentials of f and g at x^*.

   (b) ***Quadratic approximation bounds**, i.e., there exist positive constants L, L'*

such that, for all $x \in U$,

$$\|f(x) - \mathrm{D}f(x^*)(x - x^*)\|_2 \leq \frac{L}{2}\|x - x^*\|_2^2,$$
$$\|g(x) - x^* - \mathrm{D}g(x^*)(x - x^*)\|_2 \leq \frac{L'}{2}\|x - x^*\|_2^2.$$

To provide intuition for Assumption 4.1, consider a linear system $Ax = b$ with a matrix $A \in \mathbb{R}^{n\times n}$. Let $f(x) = Ax - b$ be the residual, and define the fixed-point function

$$g(x) = x - B^{-1}f(x) = x - B^{-1}(Ax - b),$$

where $B \in \mathbb{R}^{n\times n}$ is a nonsingular approximation of A. Then $x^* = A^{-1}b$ satisfies $f(x^*) = 0$ and $g(x^*) = x^*$. In this setting, $\Sigma = \mathbb{R}^n$ and V is a neighborhood of x^*. If $\|I - B^{-1}A\|_2 < 1$, the iteration is a contraction, implying

$$\|f \circ g(x)\|_2 = \|A\big(x - B^{-1}(Ax - b)\big) - b\|_2 \leq \|I - B^{-1}A\|_2 \, \|f(x)\|_2.$$

Hence, Assumption 4.1(1) holds with $K = \|I - B^{-1}A\|_2 < 1$, consistent with the standard contraction condition. Assumption 4.1(2) holds with $0 < \sigma \leq \sigma_{\min}(A)$. Note that if A is singular, such a σ does not exist and the solution cannot be locally unique. In this case, the abstract assumptions reduce to the usual contractivity and uniqueness conditions of standard fixed-point iterations.

However, if A is singular, and we take $\Sigma = \mathcal{R}(A^\top)$, the row space of A, then the residuals $f(x) = Ax - b$ lie in Σ for all x. The fixed-point mapping $g(x) = x - B^{-1}f(x)$ is no longer a contraction on $\mathbb{R}^n$, but Assumption 4.1(1) and Assumption 4.1(2) still describe a residual-reduction property and local uniqueness within Σ.

For nonlinear problems, especially those involving eigenvalue or eigenvector equations such

as (5.44) or (1.7), the submanifold Σ is not the entire ambient space. In these cases, Σ can often be chosen as the unit sphere $\mathbb{S}^{n-1}$ by normalization (e.g., the power method for (2.23)), or as the Grassmann manifold $\mathrm{Gr}(r, n)$ by projection (e.g., the CDIIS method [44] for the Hartree–Fock or Kohn–Sham equations of the form (1.8)).

These examples clarify how Assumption 4.1 can be verified in practical settings. We now analyze the propagation of perturbations in Anderson Acceleration.

Considering the absolute error (4.1), we obtain the following iterations for the Anderson-Pulay Acceleration method (2.29)

$$x^{(k+1)} = \sum_{i=0}^{m_k} \alpha_i^{(k)} \widehat{g}(x^{(k-m_k+i)}) = \sum_{i=0}^{m_k} \alpha_i^{(k)} g(x^{(k-m_k+i)}) + \sum_{i=0}^{m_k} \alpha_i^{(k)} \epsilon(x^{(k-m_k+i)}),$$

where $\left\| \sum_{i=0}^{m_k} c_i^{(k)} \epsilon(x^{(k-m_k+i)}) \right\|_2 \leq (1+m_k)\|c\|_\infty \epsilon_0$. We study the accumulation of errors over successive iterations and derive sufficient conditions under which convergence is maintained, potentially to a neighborhood of the true fixed point. This analysis serves as a foundation for the subsequent treatment of the more complex stochastic setting. The following lemma, adapted from [17, Lemma 3.7], provides a bound on the residuals and will be instrumental in our subsequent analysis to prove the convergence of the restarted and adaptive Anderson Acceleration in the presence of perturbations. Let $\mathcal{A}_\ell^{(k)}$ be the affine of $\mathrm{span}\{r^{(k-m_k)}, \dots, r^{(k-m_k+\ell)}\}$ and $d_\ell^{(k)}$ be the distance between $r^{(k-m_k+\ell)}$ and $\mathcal{A}_{\ell-1}^{(k)}$, i.e.,

$$\mathcal{A}_\ell^{(k)} := \left\{ r = \sum_{i=0}^{\ell} c_i r^{(k-m_k+i)} \;\middle|\; \sum_{i=0}^{\ell} c_i = 1 \right\}, \quad d_\ell^{(k)} = \min_{r \in \mathcal{A}_{\ell-1}^{(k)}} \|r^{(k-m_k+\ell)} - r\|_2.$$

Lemma 4.2. *Let (4.1) and Assumption 4.1 hold, and let k and m_k be two integers such that $1 \leq m_k \leq k$, $\left(x^{(k-m_k)}, \dots, x^{(k)}\right)$ be a family of vectors in $U \cap \Sigma \setminus \{x_*\}$ and t be a positive real*

number. For any integer $i \in \{0, \dots, m_k\}$, *set* $r^{(k-m_k+i)} = f\left(x^{(k-m_k+i)}\right)$ *and assume that*

$$d_i^{(k)} \geq t \max_{j \in \{i,\dots,m_k\}} \left\| r^{(k-m_k+j)} \right\|_2, \quad \text{for all } i \in \{1, \dots, m_k\}.$$

Then, vectors $r^{(k-m_k)}, \dots, r^{(k)}$ *are affinely independent, so that* $m_k \leq p$ *and there is a unique* $\widetilde{c} \in \mathbb{R}^{m_k+1}$ *such that* $\sum_{i=0}^{m_k} \widetilde{c}_i = 1$ *and* $\left\| \sum_{i=0}^{m_k} \widetilde{c}_i r^{(k-m_k+i)} \right\|_2 = \operatorname{dist}_2\left(0, \mathcal{A}_{m_k}^{(k)}\right)$. *Moreover, there exists a positive constant* C_{m_k} *such that*

$$\|\widetilde{c}\|_\infty \leq C_{m_k}\left(1 + t^{-m_k}\right).$$

Under the additional assumption that $\widetilde{x}^{(k+1)} = \sum_{i=0}^{m_k} \widetilde{c}_i x^{(k-m_k+i)} \in U$, $g\left(\widetilde{x}^{(k+1)}\right) \in U$ *and* $\sum_{i=0}^{m_k} \widetilde{c}_i g\left(x^{(k-m_k+i)}\right) \in U$, *we consider the Anderson update* $x^{(k+1)} = \sum_{i=0}^{m_k} \widetilde{c}_i g\left(x^{(k-m_k+i)}\right)$, *and set* $r^{(k+1)} = f\left(x^{(k+1)}\right)$ *and* $d_{m_k+1}^{(k)} = \operatorname{dist}_2\left(r^{(k+1)}, \mathcal{A}_{m_k}^{(k)}\right)$. *Then, there exists a constant* $\kappa > 0$ *such that*

$$\left\| r^{(k+1)} \right\|_2 \leq K \operatorname{dist}_2\left(0, \mathcal{A}_{m_k}^{(k)}\right) + \kappa\left(1 + t^{-2m_k}\right) \max_{i \in \{0,\dots,m_k\}} \left\| r^{(k-m_k+i)} \right\|_2^2,$$

and

$$(1-K)\left\| r^{(k+1)} \right\|_2 \leq K d_{m_k+1}^{(k)} + \kappa\left(1 + t^{-2m_k}\right) \max_{i \in \{0,\dots,m_k\}} \left\| r^{(k-m_k+i)} \right\|_2^2.$$

The result of [17, Lemma 3.7] addresses both Version A and Version P of Anderson Acceleration. Version A represents the Anderson Acceleration studied in this thesis, whereas Version P, introduced in Chapter 3 for comparison with real residual CROP, is not considered further; the thesis focuses exclusively on Version A. Using the absolute error measure (4.1), Lemma 4.2 can be modified to a perturbed version, which is stated below.

Lemma 4.3. *Let* (4.1) *and Assumption 4.1 hold, and let* k *and* m_k *be two integers such*

that $k \geq m_k \geq 1$. *Also, let* $\left(x^{(k-m_k)}, \ldots, x^{(k)}\right)$ *be a family of vectors in* $U \cap \Sigma \backslash \{x_*\}$, *and* t *be a positive real number. For any integer* i *in* $\{0, \ldots, m_k\}$, *set* $r^{(k-m_k+i)} = \widehat{f}\left(x^{(k-m_k+i)}\right)$ *and assume that*

$$\text{for all } \; i \in \{1, \ldots, m_k\}, d_i^{(k)} \geq t \max_{j \in \{i, \ldots, m_k\}} \left\| r^{(k-m_k+j)} \right\|_2 .$$

Then, the vectors $r^{(k-m_k)}, \ldots, r^{(k)}$ *are affinely independent, so that* $m_k \leq p$, *and there is a unique* $\widetilde{c}$ *in* $\mathbb{R}^{m_k+1}$ *such that* $\sum_{i=0}^{m_k} \widetilde{c}_i = 1$ *and* $\left\| \sum_{i=0}^{m_k} \widetilde{c}_i r^{(k-m_k+i)} \right\|_2 = \operatorname{dist}_2\left(0, \mathcal{A}_{m_k}^{(k)}\right)$. *Moreover, there exists a positive constant* C_{m_k} *such that*

$$\|\widetilde{c}\|_\infty \leq C_{m_k}\left(1 + t^{-m_k}\right).$$

Under the additional assumption that $\widetilde{x}^{(k+1)} = \sum_{i=0}^{m_k} \widetilde{c}_i x^{(k-m_k+i)} \in U$, $g\left(\widetilde{x}^{(k+1)}\right) \in U$ *and* $\sum_{i=0}^{m_k} \widetilde{c}_i g\left(x^{(k-m_k+i)}\right) \in U$, *define*

$$x^{(k+1)} = \sum_{i=0}^{m_k} \widetilde{c}_i \widehat{g}\left(x^{(k-m_k+i)}\right),$$

and set $r^{(k+1)} = \widehat{f}\left(x^{(k+1)}\right)$ *and* $d_{m_k+1}^{(k)} = \operatorname{dist}_2\left(r^{(k+1)}, \mathcal{A}_{m_k}^{(k)}\right)$. *Then, there exist constants* $\kappa_1 > 0$ *and* $\kappa_2 > 0$ *such that*

$$\left\| r^{(k+1)} \right\|_2 \leq K \operatorname{dist}_2\left(0, \mathcal{A}_{m_k}^{(k)}\right) + \kappa_1\left(1 + t^{-2m_k}\right) \max_{i \in \{0, \ldots, m_k\}} \left\| r^{(k-m_k+i)} \right\|_2^2 + \kappa_2 \epsilon_0,$$

and

$$(1-K)\left\| r^{(k+1)} \right\|_2 \leq K d_{m_k+1}^{(k)} + \kappa_1\left(1 + t^{-2m_k}\right) \max_{i \in \{0, \ldots, m_k\}} \left\| r^{(k-m_k+i)} \right\|_2^2 + \kappa_2 \epsilon_0.$$

Proof. Set $s^{(i)} = r^{(k-m_k+i)} - r^{(k-m_k+i-1)}$ for any integer $i \in \{1, \ldots, m_k\}$, $q^{(1)} = s^{(1)}$ and $q^{(i)} = \left(\mathrm{id} - \Pi_{\mathcal{A}_{i-1}^{(k)}}\right) s^{(i)}$ for any integer $i \in \{2, \ldots, m_k\}$, where $\Pi_{\mathcal{A}_{i-1}^{(k)}}$ denotes the orthogonal projector onto $\mathcal{A}_{i-1}^{(k)}$, and then $q^{(i)}$ is the component of $s^{(i)}$ orthogonal to $\mathcal{A}_{i-1}^{(k)}$. Thus, the

vectors $q^{(1)}, \ldots, q^{(m_k)}$ are mutually orthogonal and, for any integer $i \in \{1, \ldots, m_k\}$, $\left\|q^{(i)}\right\|_2 = d_i^{(k)}$. Hence,

$$\sum_{i=0}^{m_k} \widetilde{c}_i r^{(k-m_k+i)} = r^{(k)} + \sum_{i=1}^{m_k} \widetilde{\zeta}_i s^{(i)} = r^{(k)} + \sum_{i=1}^{m_k} \widetilde{\lambda}_i q^{(i)},$$

with $\widetilde{\lambda}_i = -\frac{\left(q^{(i)}\right)^\top r^{(k)}}{\left(d_i^{(k)}\right)^2}$, so that $\left|\widetilde{\lambda}_i\right| \leq \frac{\left\|r^{(k)}\right\|_2}{d_i^{(k)}} \leq \frac{1}{t}$. On the other hand, $\widetilde{\zeta} = P\widetilde{\lambda}$, where P is the change-of-basis matrix from $\left\{s^{(i)}\right\}_{i=1,\ldots,m_k}$ to $\left\{q^{(i)}\right\}_{i=1,\ldots,m_k}$. We need a bound on $\|P\|_2$. For that purpose, we introduce the matrix factorization $P = P^{(m_k)} P^{(m_k-1)} \ldots P^{(2)}$ where, for any integer $j \in \{2, \ldots, m_k\}$, $P^{(j)}$ is the change-of-basis matrix from $\left\{q^{(1)}, \ldots, q^{(j-1)}, s^{(j)}, \ldots, s^{(m_k)}\right\}$ to $\left\{q^{(1)}, \ldots, q^{(j)}, s^{(j+1)}, \ldots, s^{(m_k)}\right\}$. Since $q^{(1)} = s^{(1)}$, and $q^{(j)} = s^{(j)} - \sum_{i=1}^{j-1} \frac{\left(s^{(j)}\right)^\top q^{(i)}}{\left(q^{(i)}\right)^\top q^{(i)}} q^{(i)}$, we have

$$\left(P^{(j)}\right)_{ii} = 1 \quad \text{for all} \quad i \in \{1, \ldots, m_k\},$$

$$\left(P^{(j)}\right)_{ij} = -\frac{\left(s^{(j)}\right)^\top q^{(i)}}{\left(q^{(i)}\right)^\top v^{(i)}}, \quad \text{for all} \quad i \in \{1, \ldots, j-1\},$$

and all entries of this matrix being zero. It follows that

$$\left|\left(P^{(j)}\right)_{ij}\right| \leq \frac{\left\|s^{(j)}\right\|_2}{d_i^{(k)}} \leq \frac{\left\|r^{(k-m_k+j)}\right\|_2 + \left\|r^{(k-m_k+j-1)}\right\|_2}{d_i^{(k)}} \leq \frac{2}{t}, \quad \text{for all } i \in \{1, \ldots, j-1\}.$$

Consequently,

$$\left\|P^{(j)}\right\|_2 \leq C\left(1 + t^{-1}\right),$$

for some positive constant C depending only on m_k, which thus yields

$$\|P\|_2 \leq \left\|P^{(m_k)}\right\|_2 \left\|P^{(m_k-1)}\right\|_2 \cdot \ldots \cdot \left\|P^{(2)}\right\|_2 \leq C^{m_k-1}\left(1 + t^{-1}\right)^{m_k-1}.$$

Hence, it follows that

$$\|\widetilde{\zeta}\|_\infty \leq C^{m_k-1}\left(1 + t^{-1}\right)^{m_k-1} \|\widetilde{\lambda}\|_\infty \leq C^{m_k-1}\left(1 + t^{-1}\right)^{m_k-1} t^{-1}.$$

Finally, $\widetilde{c}_0 = -\widetilde{\zeta}_1, \widetilde{c}_i = \widetilde{\zeta}_i - \widetilde{\zeta}_{i+1}, 1 \leq i \leq m_k - 1$, and $\widetilde{c}_{m_k} = 1 + \widetilde{\zeta}_{m_k}$, so that $\|\widetilde{c}\|_\infty \leq C_{m_k}(1 + t^{-m_k})$ for some positive constant C_{m_k} depending only on m_k. Since Anderson Acceleration iterates are given as

$$x^{(k+1)} = \sum_{i=0}^{m_k} c_i^{(k)} \widehat{g}(x^{(k-m_k+i)}) = \sum_{i=0}^{m_k} c_i^{(k)} g(x^{(k-m_k+i)}) + \sum_{i=0}^{m_k} c_i^{(k)} \epsilon(x^{(k-m_k+i)}),$$

with

$$\| \sum_{i=0}^{m_k} c_i^{(k)} \epsilon(x^{(k-m_k+i)}) \|_2 \leq (1 + m_k) \|c\|_\infty \epsilon_0,$$

setting $f^{(k+1)} = f(x^{(k+1)})$ yields the residuals $r^{(k+1)} = \widehat{f}(x^{(k+1)}) = f(x^{(k+1)}) + \epsilon(x^{(k+1)}) = f^{(k+1)} + \epsilon^{(k+1)}$. To estimate $\widetilde{r}^{(k+1)} = \widehat{f}(\widetilde{x}^{(k+1)}) = f(\widetilde{x}^{(k+1)}) + \epsilon(\widetilde{x}^{(k+1)})$, we have

$$\begin{aligned}
&\left\| \widetilde{r}^{(k+1)} - \sum_{i=0}^{m_k} \widetilde{c}_i r^{(k-m_k+i)} \right\|_2 \\
\leq &\left\| \widetilde{f}^{(k+1)} + \epsilon(\widetilde{x}^{(k+1)}) - \sum_{i=0}^{m_k} \widetilde{c}_i (f^{(k-m_k+i)} + \epsilon_{k-m_k+i}) \right\|_2 \\
\leq &\left\| f(\widetilde{x}^{(k+1)}) - \mathrm{D}f(x_*)(\widetilde{x}^{(k+1)} - x_*) \right\|_2 \\
&+ \left\| \sum_{i=0}^{m_k} \widetilde{c}_i f(x^{(k-m_k+i)}) - \mathrm{D}f(x_*)(\widetilde{x}^{(k+1)} - x_*) \right\|_2 + \left\| \epsilon(\widetilde{x}) - \sum_{i=0}^{m_k} \widetilde{c}_i \epsilon_{k-m_k+i} \right\|_2 \\
\leq &\frac{L}{2} \left\| \sum_{i=0}^{m_k} \widetilde{c}_i \left(x^{(k-m_k+i)} - x_* \right) \right\|_2^2 \\
&+ \left\| \sum_{i=0}^{m_k} \widetilde{c}_i \left(f\left(x^{(k-m_k+i)} \right) - \mathrm{D}f\left(x_* \right) \left(x^{(k-m_k+i)} - x_* \right) \right) \right\|_2 \\
&+ \left\| \epsilon(\widetilde{x}) - \sum_{i=0}^{m_k} \widetilde{c}_i \epsilon_{k-m_k+i} \right\|_2 \\
\leq &\frac{L}{2} (m_k + 1) \|\widetilde{c}\|_\infty \Big(\Big(m_k + 1 \Big) \|\tilde{c}\|_\infty + 1 \Big) \max_{i \in \{0, \ldots, m_k\}} \left\| x^{(k-m_k+i)} - x_* \right\|_2^2 \\
&+ \Big(1 + (1 + m_k) \|\widetilde{c}\|_\infty \Big) \epsilon_0.
\end{aligned}$$

It thus follows from the definition of the coefficients $\widetilde{c}_i$ that

$$\begin{aligned}\left\|\widetilde{r}^{(k+1)}\right\|_2 \leq& \left\|\widetilde{r}^{(k+1)} - \sum_{i=0}^{m_k} \widetilde{c}_i r^{(k-m_k+i)}\right\|_2 + \operatorname{dist}_2\left(0, \mathcal{A}_{m_k}^{(k)}\right) \\ \leq& \frac{L}{2}\left(m_k+1\right)\|\widetilde{c}\|_\infty\left(\left(m_k+1\right)\|\widetilde{c}\|_\infty+1\right) \max_{i\in\{0,\dots,m_k\}}\left\|x^{(k-m_k+i)}-x_*\right\|_2^2 \\ &+ \operatorname{dist}_2\left(0, \mathcal{A}_{m_k}^{(k)}\right) + (1+(1+m_k)\|\widetilde{c}\|_\infty)\epsilon_0.\end{aligned}$$

Then, we find a constant $C'_{m_k} > 0$ independent of t, such that

$$\left(m_k+1\right)\|\widetilde{c}\|_\infty\left(\left(m_k+1\right)\|\widetilde{c}\|_\infty+1\right) \leq C'_{m_k}\left(1+t^{-2m_k}\right).$$

Now, using part 2 of Assumption 4.1, we finally obtain

$$\left\|\widetilde{r}^{(k+1)}\right\|_2 \leq \frac{L}{2\sigma^2} C'_{m_k}\left(1+t^{-2m_k}\right) \max_{i\in\{0,\dots,m_k\}}\left\|f^{(k-m_k+i)}\right\|_2^2 + \operatorname{dist}_2\left(0, \mathcal{A}_{m_k}^{(k)}\right) + (1+(1+m_k)\|\widetilde{c}\|_\infty)\epsilon_0,$$

and consequently

$$\begin{aligned}\left\|\widetilde{r}^{(k+1)}\right\|_2 \leq& \frac{L}{2\sigma^2} C'_{m_k}\left(1+t^{-2m_k}\right) \max_{i\in\{0,\dots,m_k\}}\left\|r^{(k-m_k+i)}\right\|_2^2 \\ &+ \operatorname{dist}_2\left(0, \mathcal{A}_{m_k}^{(k)}\right) + \left(1+(1+m_k)\|\widetilde{c}\|_\infty + \frac{L}{2\sigma^2} C'_{m_k}\left(1+t^{-2m_k}\right)\right)\epsilon_0.\end{aligned}$$

Following Anderson Acceleration, we have

$$x^{(k+1)} = \sum_{i=0}^{m_k} \widetilde{c}_i^{(k)} \widehat{g}(x^{(k-m_k+i)}) = \sum_{i=0}^{m_k} \widetilde{c}_i^{(k)} g(x^{(k-m_k+i)}) + \sum_{i=0}^{m_k} \widetilde{c}_i^{(k)} \epsilon(x^{(k-m_k+i)}).$$

Hence, we can bound $\|x^{k+1} - g(\widetilde{x}^{(k+1)})\|_2$ by

$$
\begin{aligned}
&\|g(\widetilde{x}^{(k+1)}) - \sum_{i=0}^{m_k} \widetilde{c}_i \hat{g}(x^{(k-m_k+i)})\|_2 \\
\leq &\left\| g(\widetilde{x}^{(k+1)}) - \sum_{i=0}^{m_k} \widetilde{c}_i (g^{(k-m_k+i)} + \epsilon_{k-m_k+i}) \right\|_2 \\
\leq &\left\| g(\tilde{x}^{(k+1)}) - x^* - \mathrm{D}g(x_*)(\widetilde{x}^{(k+1)} - x_*) \right\|_2 \\
&+ \| \sum_{i=0}^{m_k} \widetilde{c}_i g(x^{(k-m_k+i)}) - x^* - \mathrm{D}g(x_*)(\tilde{x}^{(k+1)} - x_*) \|_2 \\
&+ \left\| \sum_{i=0}^{m_k} \widetilde{c}_i \epsilon_{k-m_k+i} \right\|_2 \\
\leq &\frac{L'}{2} \left\| \sum_{i=0}^{m_k} \widetilde{c}_i \left(x^{(k-m_k+i)} - x_* \right) \right\|_2^2 \\
&+ \left\| \sum_{i=0}^{m_k} \widetilde{c}_i \left(g\left(x^{(k-m_k+i)} \right) - x^* - \mathrm{D}g\left(x_* \right) \left(x^{(k-m_k+i)} - x_* \right) \right) \right\|_2 \\
&+ \left\| \sum_{i=0}^{m_k} \widetilde{c}_i \epsilon_{k-m_k+i} \right\|_2 \\
\leq &\frac{L'}{2} \left(m_k + 1 \right) \|\widetilde{c}\|_\infty \left(\left(m_k + 1 \right) \|\widetilde{c}\|_\infty + 1 \right) \max_{i \in \{0,\ldots,m_k\}} \left\| x^{(k-m_k+i)} - x_* \right\|_2^2 \\
&+ (1 + m_k) \|\widetilde{c}\|_\infty \epsilon_0 \\
\leq &\frac{L'}{2\sigma^2} C'_{m_k} \left(1 + t^{-2m_k} \right) \max_{i \in \{0,\ldots,m_k\}} \left\| r^{(k-m_k+i)} \right\|_2^2 \\
&+ \left((1 + m_k) \|\widetilde{c}\|_\infty + \frac{L}{2\sigma^2} C'_{m_k} \left(1 + t^{-2m_k} \right) \right) \epsilon_0.
\end{aligned}
$$

Then, using part 3(a) of Assumption 4.1 and the above estimate, we obtain

$$
\begin{aligned}
\left\| r^{(k+1)} \right\|_2 &\leq \left\| f\left(g\left(\widetilde{x}^{(k+1)} \right) \right) \right\|_2 + 2 \left\| \mathrm{D}f\left(x_* \right) \right\|_2 \left\| x^{(k+1)} - g\left(\widetilde{x}^{(k+1)} \right) \right\|_2 + \epsilon_0 \\
&\leq K \left\| \widetilde{r}^{(k+1)} \right\|_2 + 2 \left\| \mathrm{D}f\left(x_* \right) \right\|_2 \left\| x^{(k+1)} - g\left(\widetilde{x}^{(k+1)} \right) \right\|_2 + \epsilon_0 \\
&\leq K \operatorname{dist}_2\left(0, \mathcal{A}_{m_k}^{(k)} \right) + \kappa_1 \left(1 + t^{-2m_k} \right) \max_{i \in \{0,\ldots,m_k\}} \left\| r^{(k-m_k+i)} \right\|_2^2 + \kappa_2 \epsilon_0,
\end{aligned}
$$

where

$$\kappa_1 = \frac{1}{\sigma^2}\left(\frac{KL}{2} + L' \left\|\mathrm{D}f\left(x_*\right)\right\|_2\right) \max_{m\in\{1,\dots,p\}} C'_m,$$

and

$$\begin{aligned}\kappa_2 = 1 &+ \left((1+(1+m_k)\|\widetilde{c}\|_\infty)K + 2(1+m_k)\|\widetilde{c}\|_\infty \left\|\mathrm{D}f\left(x_*\right)\right\|_2\right) \\ &+ \frac{1}{\sigma^2}\left(\frac{KL}{2} + L' \left\|\mathrm{D}f\left(x_*\right)\right\|_2\right) \max_{m\in\{1,\dots,p\}} C'_m.\end{aligned}$$

□

Lemma 4.3 provides a foundation for analyzing the effect of deterministic perturbations in various variants of Anderson Acceleration, including the restarted and adaptive-depth schemes. Lemma 4.3 shows that a linear error term is introduced in the residual as a consequence of perturbations in the function evaluations. This error is bounded by a constant multiple of the magnitude of the perturbation. The perturbation properties established here can also be extended to other Anderson-type acceleration methods that share a similar update structure.

Let us consider a case where f and g are evaluated by randomized simulations using a variable number of trials N_{MC}, e.g., the outputs of experiments, with outputs $\widehat{f}(x, N_{MC})$ and $\widehat{g}(x, N_{MC})$. Hence, the associated error function $\epsilon(x, N_{MC})$ will not be bounded like in the deterministic case.

Assumption 4.4. *There is a function c_f and an open set $\mathcal{B}$ which contains U (the small neighborhood of x^*) such that, for all $x \in \mathcal{B}$ and $\delta > 0$,*

$$P\left(\|\epsilon(x, N_{MC})\|_2 > \frac{c_f(\delta)}{\sqrt{N_{MC}}}\right) < \delta.$$

Analogously to [115, Theorem 3] and [117, Theorem 3.1], we can connect the stochastic Assumption 4.4 with Lemma 4.3.

Lemma 4.5. *Let Assumptions 4.1 and 4.4 hold. Then, for any $\epsilon_0 > 0$, $\omega \in (0,1)$, and $K > 0$, there exists N_{MC}^* such that for all $N_{MC} \geq N_{MC}^*$, Lemma 4.3 holds for $0 \leq m_k \leq k \leq K$ with probability at least $1-\omega$.*

Proof. Let N_{MC}^* be large enough such that $\frac{c_f(\delta)}{\sqrt{N_{MC}}} \leq \epsilon_0$, and let $\delta < 1-(1-\omega)^{1/(K+2)}$. Then by Assumption 4.4

$$P\Big(\|\epsilon(x^{(i)}, N_{MC})\|_2 \leq \epsilon_0\Big) \geq 1-\delta = (1-\omega)^{1/(K+2)},$$

for any $0 \leq i \leq k+1$, $\|\epsilon(x^{(i)}, N_{MC})\|_2 \leq \epsilon_0$ are independent events for $i \in \{0,\dots,K+1\}$. Hence, $\|\epsilon(x^{(i)}, N_{MC})\|_2 \leq \epsilon_0$ for all $i \in \{0,\dots,K+1\}$ holds with probability at least $1-\omega$. And thus Lemma 4.3 holds for $0 \leq m_k \leq k \leq K$ with probability at least $1-\omega$. □

Therefore, presented variants of the Anderson Acceleration method can also be applied in the stochastic setting, for example, when the evaluation of the functions f or g involves a Monte Carlo simulation with N_{MC} trials. In such cases, the algorithm remains applicable, but its convergence behavior and attainable accuracy strongly depend on the statistical properties of the stochastic noise introduced by the sampling process. We will see the influence of the stochastic noise on different variants of Anderson Acceleration in the forthcoming numerical examples.

4.3 Numerical Experiments

To complement our theoretical findings in Section 4.2, we carry out several numerical experiments on representative linear and nonlinear problems with controlled perturbations. Plots of residual norms and convergence histories highlight the effect of perturbation magnitude on

convergence across different Anderson variants. Specifically, we will compare the behavior of three different Anderson Acceleration algorithms under absolute and relative perturbations, i.e., the fixed-depth Anderson Algorithm 2, with history depth m; the restarted Anderson Algorithm 3, with restart parameter τ; and the adaptive-depth Anderson Algorithm 4, where parameter δ controls the adaptive history depth.

4.3.1 Experiment: Linear Problem 1 (Deterministic Perturbation)

Consider a linear system $Ax = b$, with a Poisson system matrix $A = \texttt{tridiag}(\texttt{1}, -\texttt{4}, \texttt{1}) \in \mathbb{R}^{n\times n}$ and right-hand side vector $b = [\texttt{1}; \texttt{zeros}(\texttt{n} - \texttt{1}, \texttt{1})] \in \mathbb{R}^n$.

(1a) We choose the exact $f = Ax - b$ and $g = x + f$ for comparison, and run the fixed-depth (Algorithm 2), restarted (Algorithm 3) and adaptive-depth (Algorithm 4) variants of Anderson Acceleration on f and g, with $maxit = 100$, $tol = 10^{-10}$ and different choices of the parameter m, τ and δ.

(1b) We choose $f = Ax - b + \epsilon N$ with absolute error, $g = x + f$, where $\epsilon = 10^{-2}$, N is a random vector indicating a Gaussian noise with mean 0 and variance 10^{-2}. We run the fixed-depth, restarted, and adaptive-depth variants of Anderson Acceleration (Version A) on f and g, with $maxit = 100$, $tol = 10^{-10}$ and different choices of m, τ and δ.

(1c) We choose $f = Ax - b + r\epsilon N$ with relative error, $g = x + f$, where $\epsilon = 10^{-2}$, N is a random vector indicating a Gaussian noise with mean 0 and variance 10^{-2}, and r is the residual norm in the current iteration step. We run the fixed-depth, restarted and adaptive-depth variants of Anderson Acceleration (Version A) on f and g, with $maxit = 100$, $tol = 10^{-10}$ and different choices of m, τ and δ.

Figures 4.1 to 4.3 present the convergence behavior of three Anderson Acceleration algorithms applied to the problem in Section 4.3.1 under three different settings: no perturbation (Figure 4.1), absolute perturbation (Figure 4.2), and relative perturbation (Figure 4.3).

In the unperturbed case (Figure 4.1), the fixed-depth Anderson Acceleration with full history ($m = \text{maxit}$) achieves a convergence rate identical to that of the GMRES algorithm. This outcome directly reflects the well-known equivalence between Anderson Acceleration and GMRES for linear systems in exact arithmetic [6]. For the restarted Anderson method with thresholds $\tau = 0.1$ and $\tau = 0.001$, restarts are observed, with the higher threshold $\tau = 0.1$ triggering restarts more frequently, consistent with theoretical expectations. The adaptive-depth Anderson method with adaptation parameters $\delta = 0.1$ and $\delta = 0.001$ exhibits temporary slowdowns at certain iterations, corresponding to steps where the history length is adjusted.

When an absolute perturbation is introduced (Figure 4.2), all Anderson variants converge

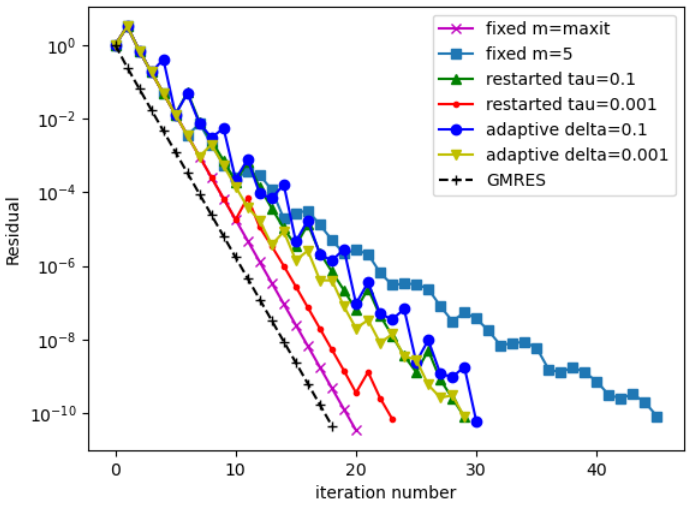


Figure 4.1: Convergence history for Algorithms 2 to 4 applied to the problem in Section 4.3.1 with $f = Ax - b$, $g = x + f$, $maxit = 100$ and $tol = 10^{-10}$.

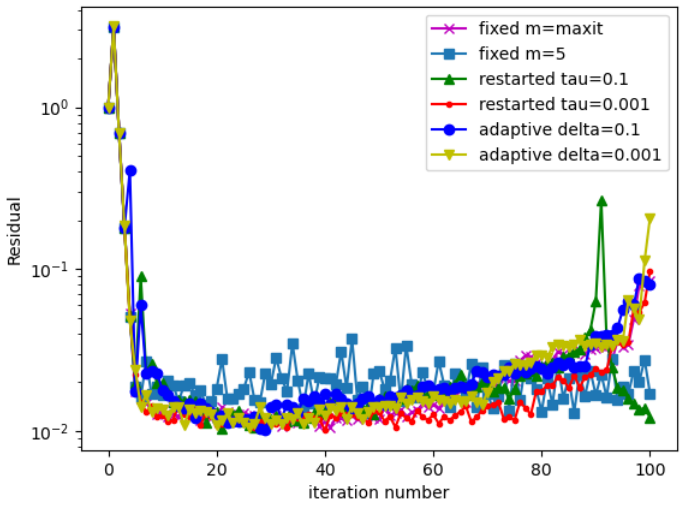


Figure 4.2: Convergence history for Algorithms 2 to 4 applied to the problem in Section 4.3.1 with $f = Ax - b + \epsilon N$, $g = x + f$, $maxit = 100$ and $tol = 10^{-10}$.

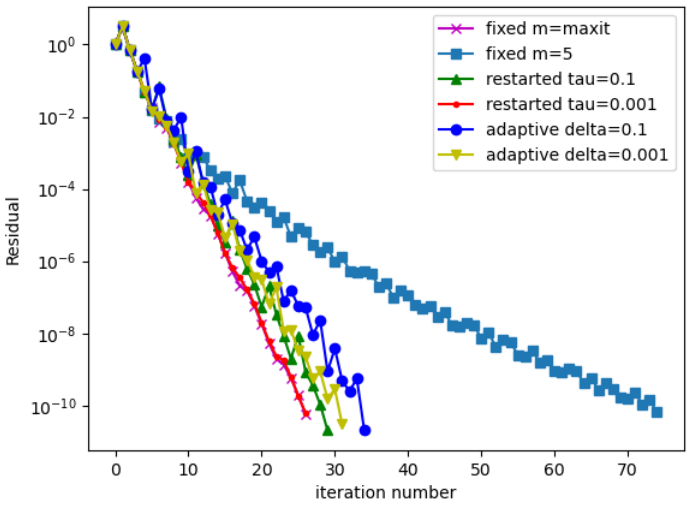


Figure 4.3: Convergence history for Algorithms 2 to 4 applied to the problem in Section 4.3.1 with $f = Ax - b + r\epsilon N$, $g = x + f$, $maxit = 100$ and $tol = 10^{-10}$.

only to an accuracy of approximately 10^{-2}, matching the magnitude of the imposed perturbation ϵ. This plateau demonstrates that Anderson Acceleration cannot eliminate random additive noise through iteration; thus, the attainable precision is inherently limited by the perturbation level.

In the relative perturbation case (Figure 4.3), noise is applied proportionally to $f(x)$, with magnitude on the order of 10^{-2}. As a result, the evaluations of $f(x)$ and $g(x)$ vary by roughly 1%, causing only minor deviations in the convergence profiles. The fixed-depth, restarted, and adaptive-depth methods all retain similar convergence characteristics to the unperturbed case, though with slightly reduced rates. Notably, the restarts observed for $\tau = 0.001$ in Figure 4.1 do not occur here, as the relative perturbation slightly alters the linear dependence among residual vectors, thereby modifying the restart condition.

4.3.2 Experiment: Linear Problem 2 (Deterministic Perturbation)

Let us now consider a linear system $Ax = b$ with a Poisson system matrix $A = \texttt{tridiag}(1, -2, 1) \in \mathbb{R}^{n\times n}$ and right-hand side vector $b = [1; \texttt{zeros}(n-1, 1)] \in \mathbb{R}^n$. Compared to the problem in Section 4.3.1, this linear system is ill-conditioned when n is large.

(2a) We choose the exact $f = Ax - b$ and $g = x + f$ for comparison, and run the fixed-depth (Algorithm 2), restarted (Algorithm 3) and adaptive-depth (Algorithm 4) variants of Anderson Acceleration on f and g, with $maxit = 100$, $tol = 10^{-10}$ and different choices of the parameters m, τ and δ.

(2b) We choose $f = Ax - b + \epsilon N$ with absolute error, $g = x + f$, where $\epsilon = 10^{-2}$, N is a random vector indicating a Gaussian noise with mean 0 and variance 10^{-2}. We run the fixed-depth, restarted and adaptive-depth variants of Anderson Acceleration Version A on f and g, with $maxit = 100$, $tol = 10^{-10}$ and different choices of m, τ and δ.

(2c) We choose $f = Ax - b + r\epsilon N$ with relative error, $g = x + f$, where $\epsilon = 10^{-2}$ and N is a random vector indicating a Gaussian noise with mean 0 and variance 10^{-2}, and r is the residual norm in the current iteration step. We run the fixed-depth, restarted and adaptive-depth variants of Anderson Acceleration Version A on f and g, with $maxit = 100$, $tol = 10^{-10}$ and different choices of m, τ and δ.

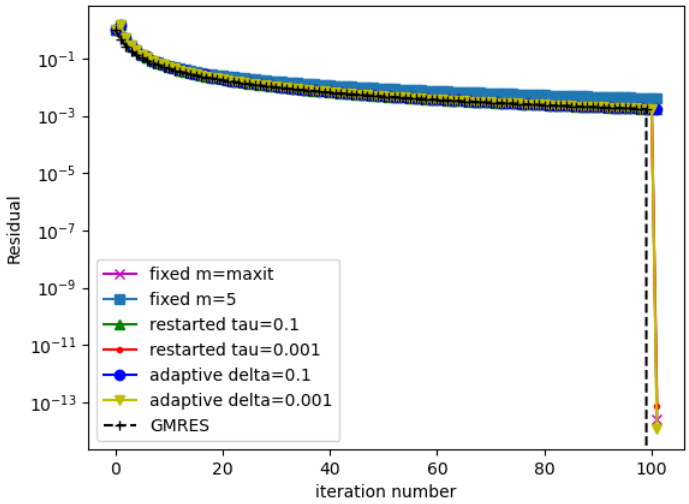


Figure 4.4: Convergence history for Algorithms 2 to 4 to the problem in Section 4.3.2 with $f = Ax - b$, $g = x + f$, $maxit = 100$ and $tol = 10^{-10}$.

Figures 4.4 to 4.6 illustrate the convergence behavior of three Anderson Acceleration algorithms applied to the problem introduced in Section 4.3.2, under three conditions: no perturbation (Figure 4.4), absolute perturbation (Figure 4.5), and relative perturbation (Figure 4.6).

In the unperturbed case (Figure 4.4), all three Anderson variants exhibit nearly identical convergence profiles. This behavior is primarily governed by the conditioning of the underlying problem, which dominates the performance and masks differences among algorithmic parameters. No restarts are observed in this case, confirming the numerical stability of the

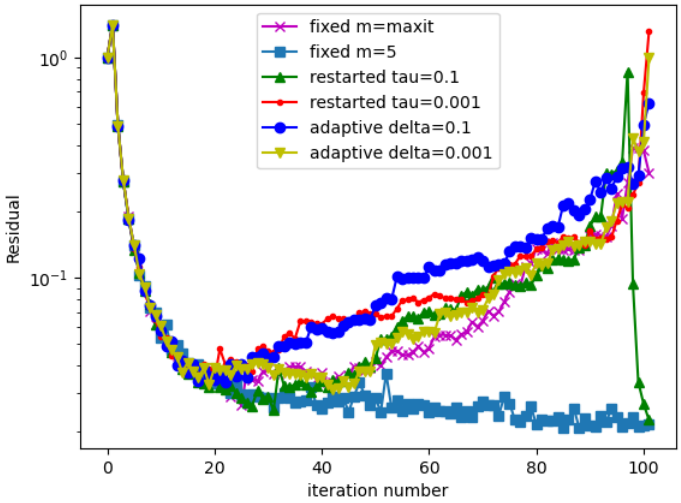


Figure 4.5: Convergence history for Algorithms 2 to 4 applied to the problem in Section 4.3.2 with $f = Ax - b + \epsilon N$, $g = x + f$, $maxit = 100$ and $tol = 10^{-10}$.

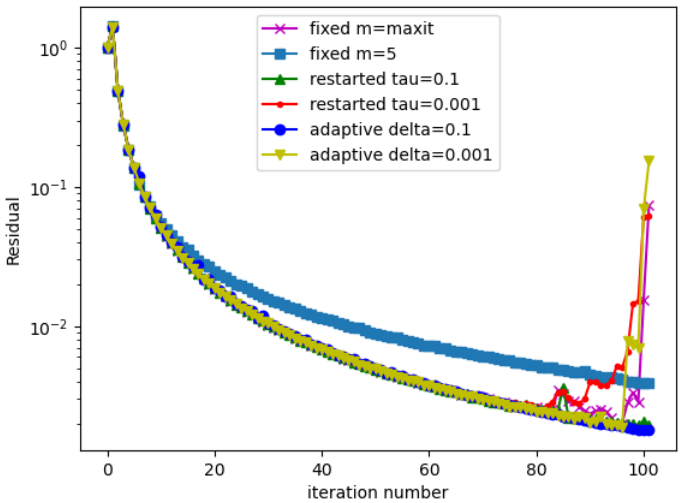


Figure 4.6: Convergence history for Algorithms 2 to 4 applied to the problem in Section 4.3.2 with $f = Ax - b + r\epsilon N$, $g = x + f$, $maxit = 100$ and $tol = 10^{-10}$.

iteration when function evaluations are exact.

When an absolute perturbation is introduced in Figure 4.5, its effect on convergence is significantly stronger than that observed for the problem in Section 4.3.1. The presence of additive noise alters the residual structure and slows the decay rate of the error norm, reflecting the problem's higher sensitivity to perturbations. For this test, a single restart is triggered for the restarted Anderson method with threshold $\tau = 0.1$ at iteration $k = 96$.

Figure 4.6 presents results under a relative perturbation of approximately 1%, which becomes noticeable after about 90 iterations. The rank of the system matrix is $\text{rank}(A) = 100$, implying that the noise affects all components of the solution space. A restart event is again observed for $\tau = 0.1$ at $k = 95$. Interestingly, both the restarted Anderson method with $\tau = 0.1$ and the adaptive-depth variant with $\delta = 0.1$ exhibit slightly better performance than their counterparts with smaller parameters ($\tau = 0.001$ and $\delta = 0.001$). Although this trend appears counterintuitive, it can be attributed to the dominant influence of perturbations at later iterations, which outweighs the nominal advantage of tighter parameter thresholds.

To verify this last claim of the above discussion, we propose the following set of additional experiments.

(2A) We choose $f = Ax - b$ and $g = x + f$, and run the fixed-depth (Algorithm 2), restarted (Algorithm 3) and adaptive-depth (Algorithm 4) variants of Anderson Acceleration on f and g, with $maxit = 300$, $tol = 10^{-10}$ and different choices of the parameters m, τ and δ.

(2C) We choose $f = Ax - b + r\epsilon N$, $g = x + f$, where $\epsilon = 10^{-2}$, N is a random vector indicating a Gaussian noise with mean 0 and variance 10^{-2}, and r is the residual norm in the current iteration step. We run the fixed-depth (Algorithm 2), restarted (Algorithm 3) and adaptive-depth (Algorithm 4) variants of Anderson Acceleration Version A on f

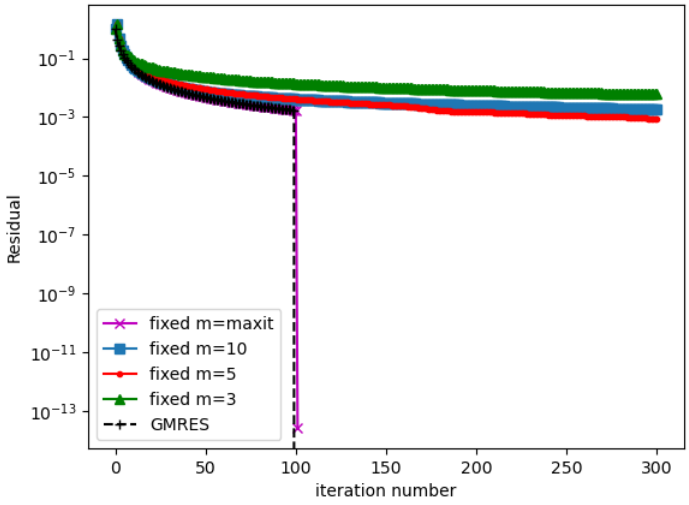


Figure 4.7: Convergence history for the fixed-depth Algorithm 2 applied to the problem in Section 4.3.2 with $f = Ax - b$, $g = x + f$, $maxit = 300$ and $tol = 10^{-10}$ for varying values of history depth m.

and g, with $maxit = 300$, $tol = 10^{-10}$ and different choices of m, τ and δ.

Figures 4.7 to 4.9 present the convergence of Algorithms 2 to 4 applied to the problem in Section 4.3.2 in the absence of perturbations (Experiment 2A). Given that $\text{rank}(A) = 100$, all variants of Anderson Acceleration with the full iteration history reach the exact solution with high accuracy after 100 iterations. For the restarted Anderson method with $\tau = 0.1$, restarts occur periodically at iterations $k = 101 + 6j$, $j = 0, \ldots, 33$. In contrast, no restarts are observed for smaller thresholds, so they all converge after 100 iterations. Similarly, for the adaptive-depth algorithm, the adaptation mechanism is inactive for the smallest parameters $\delta = 0.001$ and $\delta = 0.0001$, whereas the first adaptation occurs at iterations $k = 6$ and $k = 26$ for $\delta = 0.1$ and $\delta = 0.01$, respectively.

The convergence curves in Figure 4.10 confirm the advantage of maintaining a longer history size m in the fixed-depth Anderson method, even if it temporarily slows convergence at

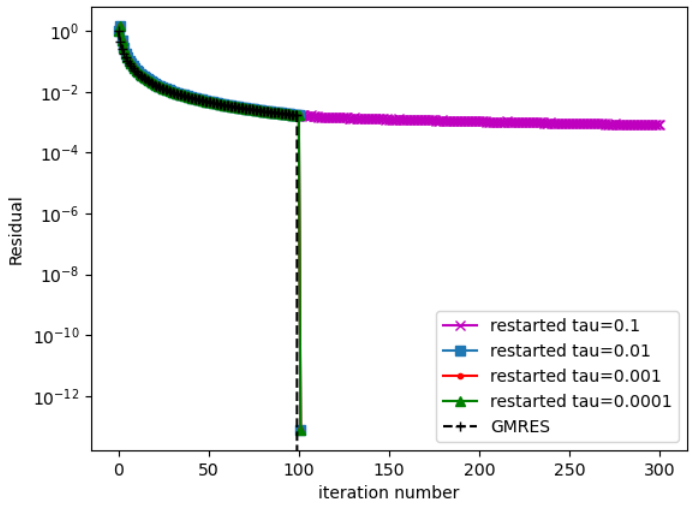


Figure 4.8: Convergence history for the restarted Algorithm 3 applied to the problem in Section 4.3.2 with $f = Ax - b$, $g = x + f$, $maxit = 300$ and $tol = 10^{-10}$ for varying values of parameter τ.

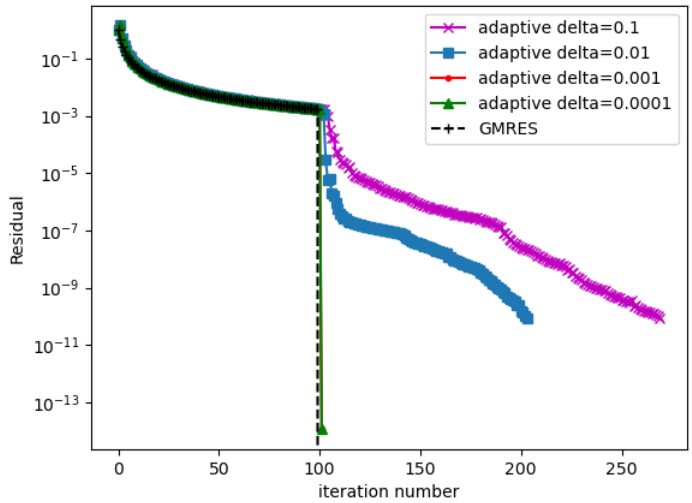


Figure 4.9: Convergence history for the adaptive Algorithm 4 applied to the problem in Section 4.3.2 with $f = Ax - b$, $g = x + f$, $maxit = 300$ and $tol = 10^{-10}$ for varying values of parameter δ.

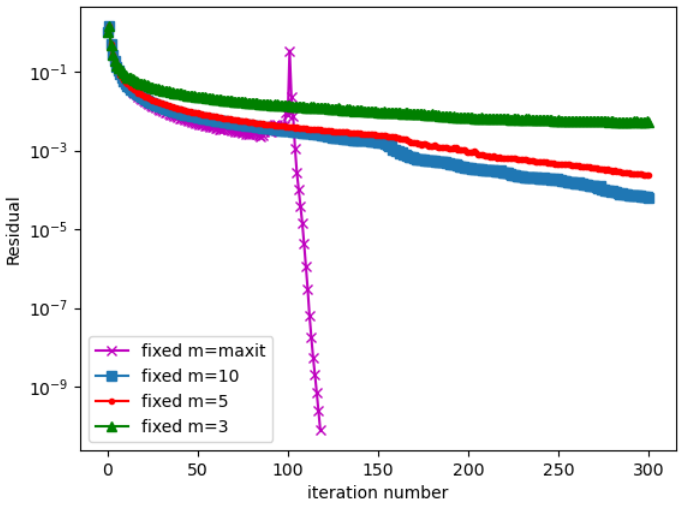


Figure 4.10: Convergence history for the fixed-depth Algorithm 2 applied to the problem in Section 4.3.2 with $f = Ax - b + r\epsilon N$, $g = x + f$, $maxit = 300$ and $tol = 10^{-10}$ for varying values of history depth m.

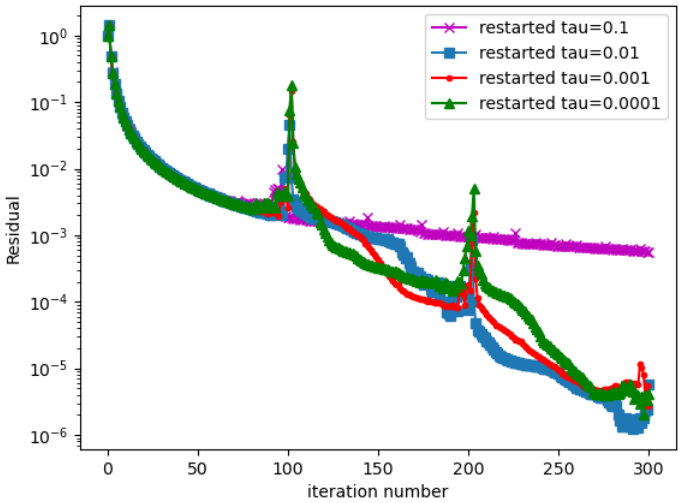


Figure 4.11: Convergence history for the restarted Algorithm 3 applied to the problem in Section 4.3.2 with $f = Ax - b + r\epsilon N$, $g = x + f$, $maxit = 300$ and $tol = 10^{-10}$ for varying values of parameter τ.

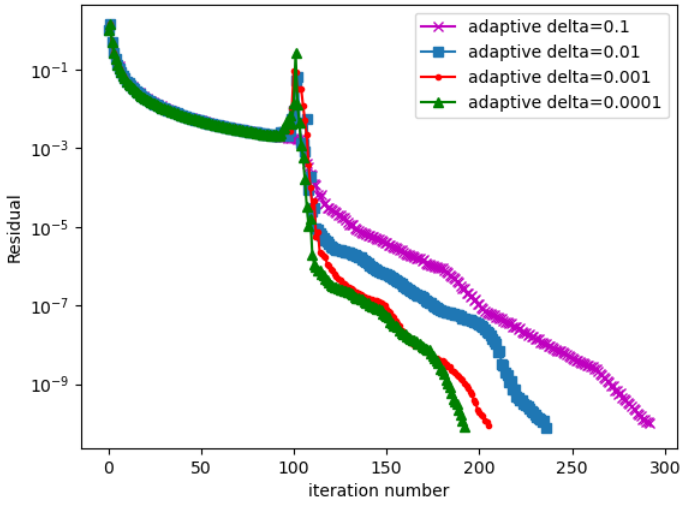


Figure 4.12: Convergence history for the adaptive Algorithm 4 applied to the problem in Section 4.3.2 with $f = Ax - b + r\epsilon N$, $g = x + f$, $maxit = 300$ and $tol = 10^{-10}$ for varying values of parameter δ.

certain steps.

A detailed view of the restarted Anderson method, shown in Figure 4.11, reveals that for $\tau = 0.1$, frequent restarts occur, while for smaller thresholds only two restarts are triggered: at $(k = 101, 202)$ for $\tau = 0.01$ and at $(k = 102, 203)$ for both $\tau = 0.001$ and $\tau = 0.0001$. When random relative perturbations are introduced, the observed convergence behavior varies between parameter choices. The random nature of the noise is the primary factor influencing these differences, and repeating the experiment produces a different ranking of the methods each time.

For the adaptive-depth Anderson algorithm, the first adaptation steps are observed at $k = 6$, 26, 108, and 105 for $\delta = 0.1$, 0.01, 0.001, and 0.0001, respectively. If the slow convergence behavior during the initial 100 iterations is disregarded (see Figure 4.12), smaller values of δ consistently lead to improved convergence performance due to the retention of more historical information. This trend remains consistent across multiple experimental repetitions.

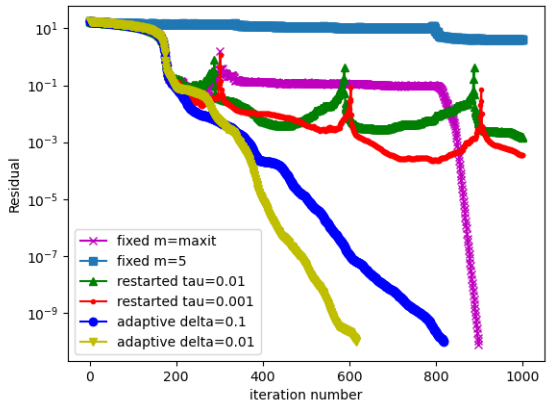
4.3.3 Experiment: Linear Problem 3 (Stochastic Perturbation)

Consider a linear system $Ax = b$ with a Poisson system matrix $A = \texttt{tridiag}(1, -2, 1) \in \mathbb{R}^{n\times n}$ and a vector $b = \texttt{ones}(\texttt{n}, 1) \in \mathbb{R}^n$ of size $n = 300$.

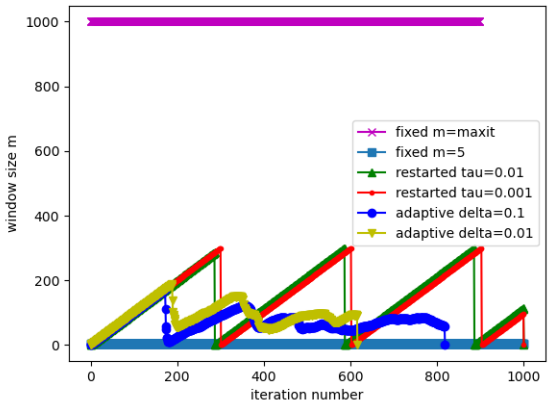
We choose $f(x) = Ax - b$ and $g(x) = x + f(x)$. And we introduce the errors by applying a relative perturbation to the output of the residual function f that is normally distributed with mean zero and variance $1/N_{MC}$, i.e., $\hat{f}(x, N_{MC}) = f(x) + f(x) \odot \epsilon(N_{MC})$. In this experiment, we run the fixed-depth, restarted, and adaptive-depth variants of Anderson Acceleration on f and g, with $maxit = 1000$, $tol = 10^{-10}$ and different choices of parameters $m = \infty$, $m = 5$, $\tau = 0.01$, $\tau = 0.001$, $\delta = 0.1$, $\delta = 0.01$. We plot the convergence history for each method with some chosen N_{MC}.

Figure 4.13 illustrates the convergence behavior of variants of Anderson Acceleration algorithms applied to the problem in Section 4.3.3 when the function evaluations are affected by stochastic perturbations relative to the residuals.

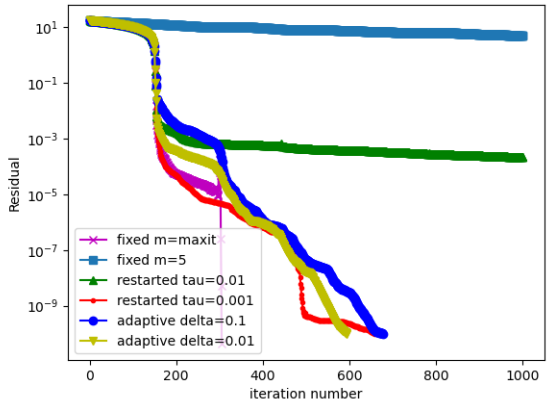
Figure 4.13a and Figure 4.13b show the residuals and the window sizes of an experiment with $N_{MC} = 10^6$. Figure 4.13c and Figure 4.13d show the residuals and the window sizes of an experiment with $N_{MC} = 10^{10}$. Comparing Figure 4.13a and Figure 4.13c, we can see that the number of convergent methods increases with N_{MC} as expected, reflecting that weaker noise allows a better approximation of the true fixed point. N_{MC} needs to be large enough, otherwise the errors are too large for the methods to converge. Figure 4.13e shows when $N_{MC} = 10^5$, all methods fail to converge. The additional realization with $N_{MC} = 10^6$ (Figure 4.13f) demonstrates that convergence is not guaranteed for a particular N_{MC} due to randomness in the perturbation; for instance, the fixed-depth Anderson method with



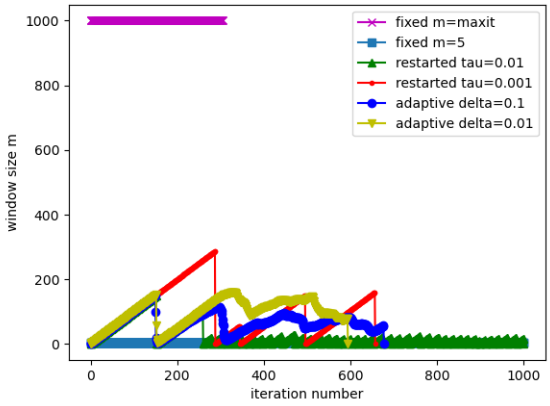
(a) Residual: $N_{MC} = 10^6$.



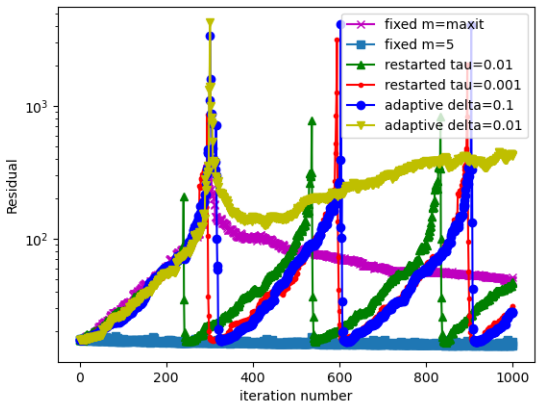
(b) Window size: $N_{MC} = 10^6$.



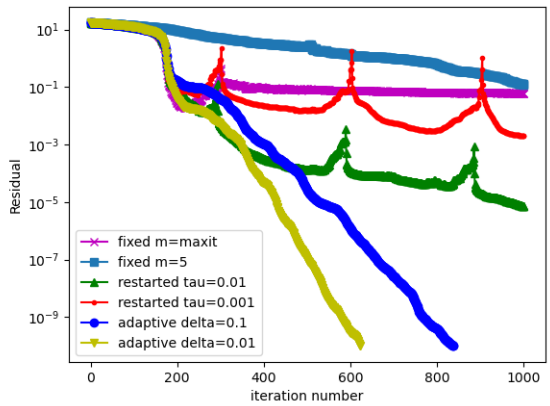
(c) Residual: $N_{MC} = 10^{10}$.



(d) Window size: $N_{MC} = 10^{10}$.



(e) Residual: $N_{MC} = 10^5$.



(f) Another example: $N_{MC} = 10^6$.

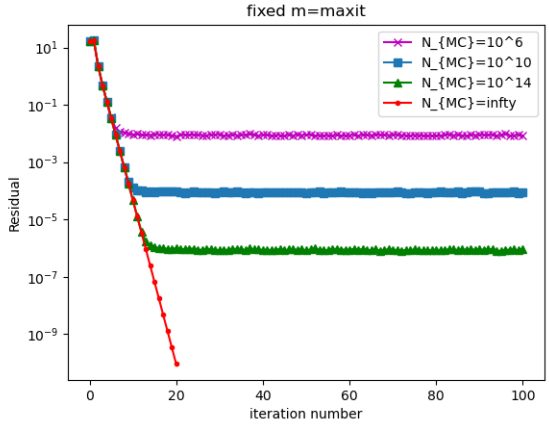
Figure 4.13: Results of the experiment in Section 4.3.3.

$m = \infty$ converges in Figure 4.13a but fails in Figure 4.13f. Overall, these results highlight the sensitivity of Anderson-type methods to stochastic noise and the critical dependence of convergence on the perturbation level.

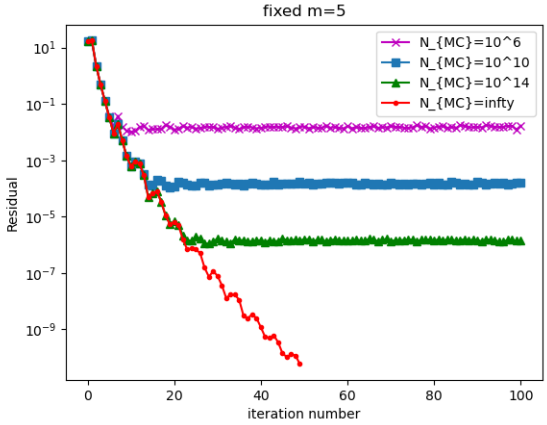
4.3.4 Experiment: Linear Problem 4 (Stochastic Perturbation)

Consider a linear system $Ax = b$ with a Poisson system matrix $A = \texttt{tridiag}(1, -4, 1) \in \mathbb{R}^{n\times n}$ and a vector $b = \texttt{ones}(\texttt{n}, 1) \in \mathbb{R}^n$ of size $n = 300$. We choose $f(x) = Ax - b$ and $g(x) = x + f(x)$. And we introduce the errors by applying a relative perturbation to the output of the fixed-point map g that is normally distributed with mean zero and variance $1/N_{MC}$, i.e., $\hat{g}(x, N_{MC}) = g(x) + g(x) \odot \epsilon(N_{MC})$. In this experiment, we run the fixed-depth, restarted, and adaptive-depth variants of Anderson Acceleration on f and g, with $maxit = 1000$, $tol = 10^{-10}$ and different choices of parameters $m = \infty$, $m = 5$, $\tau = 0.01$, $\tau = 0.001$, $\delta = 0.1$, $\delta = 0.01$. We plot the convergence history for each method with four different values of N_{MC}. Note that $N_{MC} = \infty$ is the error-free case.

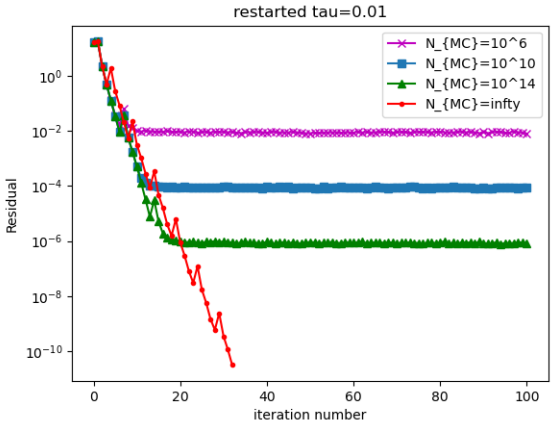
Figure 4.14 illustrates the convergence behavior of variants of Anderson Acceleration algorithms applied to the problem in Section 4.3.4 when the function evaluations are subject to stochastic noise originating from Monte Carlo sampling. In all cases, the convergence curves exhibit stagnation once the residual norm reaches the order of the sampling noise, approximately $\mathcal{O}(1/\sqrt{N_{\mathrm{MC}}})$. Beyond this level, further iterations do not yield noticeable improvement, regardless of the Anderson variant employed. This behavior confirms that the attainable accuracy is fundamentally limited by the statistical noise inherent to the Monte Carlo estimator, and that acceleration cannot overcome this variance-induced convergence barrier.



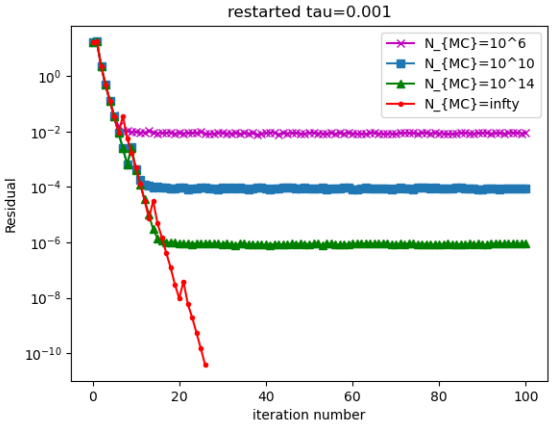
(a) Fixed Anderson: $m = \infty$.



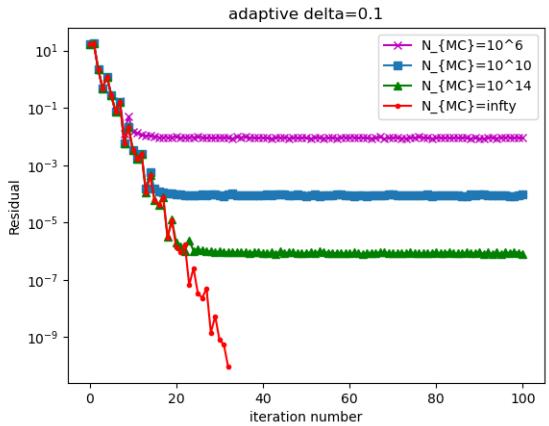
(b) Fixed Anderson: $m = 5$.



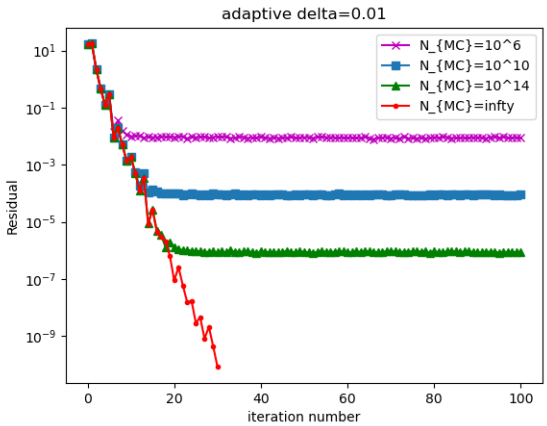
(c) Restarted Anderson: $\tau = 0.01$.



(d) Restarted Anderson: $\tau = 0.001$.



(e) Adaptive Anderson: $\delta = 0.1$.



(f) Adaptive Anderson: $\delta = 0.01$.

Figure 4.14: Results of the experiment in Section 4.3.4.

4.3.5 Experiment: Nonlinear Eigenvalue Problem (Stochastic Perturbation)

Consider the following NEP corresponding to a distributed time-delay system discussed in [118, Example 2] where the distributed term is a Gaussian distribution,

$$\begin{aligned}\dot{x}(t) =&\frac{1}{10}\begin{pmatrix} 25 & 28 & -5 \\ 18 & 3 & 3 \\ -23 & -14 & 35 \end{pmatrix} x(t) + \frac{1}{10}\begin{pmatrix} 17 & 7 & -3 \\ -24 & -21 & -2 \\ 20 & 7 & 4 \end{pmatrix} x(t-\tau) \\ &+ \int_{-\tau}^{0} \begin{pmatrix} 14 & -13 & 4 \\ 14 & 7 & 10 \\ 6 & 16 & 17 \end{pmatrix} \frac{e^{(s+\frac{1}{2})^2} - e^{\frac{1}{4}}}{10} x(t+s)ds.\end{aligned} \tag{4.3}$$

The eigenvalues of (4.3) are given as the solutions of the nonlinear eigenvalue problem (NEP) $M(\lambda)v = 0$ associated with a matrix-valued function

$$M(\lambda) = -\lambda I + A_0 + A_1 e^{-\lambda\tau} + \int_{-\tau}^{0} F(s)e^{\lambda s}ds,$$

where

$$A_0 = \frac{1}{10}\begin{pmatrix} 25 & 28 & -5 \\ 18 & 3 & 3 \\ -23 & -14 & 35 \end{pmatrix}, \quad A_1 = \frac{1}{10}\begin{pmatrix} 17 & 7 & -3 \\ -24 & -21 & -2 \\ 20 & 7 & 4 \end{pmatrix},$$

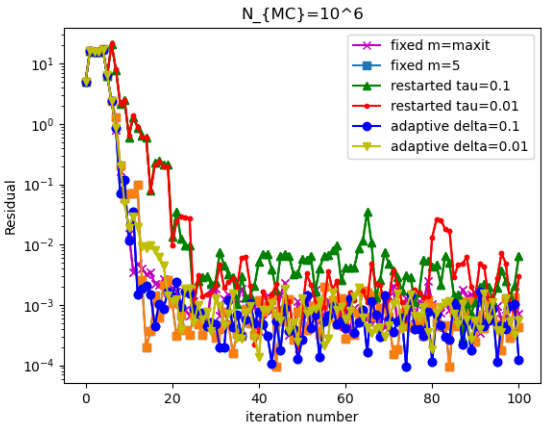
$$F(s) = \begin{pmatrix} 14 & -13 & 4 \\ 14 & 7 & 10 \\ 6 & 16 & 17 \end{pmatrix} \frac{e^{(s+\frac{1}{2})^2} - e^{\frac{1}{4}}}{10}.$$

Consider $x = \begin{bmatrix} v \\ \lambda \end{bmatrix}$. Then, the nonlinear eigenvalue problem associated with (4.3) can be written as a nonlinear equation of the form

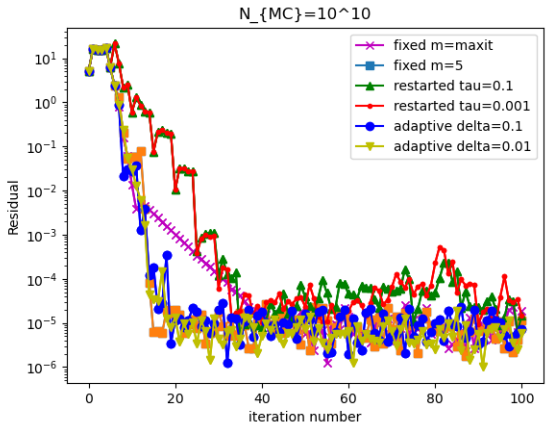
$$f(x) = f\left(\begin{bmatrix} v \\ \lambda \end{bmatrix}\right) := \begin{bmatrix} M(\lambda)v \\ c^H v - 1 \end{bmatrix} = 0,$$

where c is used to normalize the eigenvector v and we choose c the vector of ones here. Let $g := x + f$. To model inexact evaluations, we apply a relative perturbation to the output of the fixed-point map g; the perturbation is normally distributed with mean zero and variance $1/N_{MC}$, i.e., $\hat{g}(x, N_{MC}) = g(x) + g(x) \odot \epsilon(N_{MC})$. In this experiment, we apply the fixed-depth, restarted, and adaptive-depth variants of Anderson Acceleration to both f and g, using $maxit = 1000$, $tol = 10^{-10}$, and various parameter values: $m = \infty$, $m = 5$ or the fixed-depth method, $\tau = 0.1$, $\tau = 0.01$ or the restarted method, and $\delta = 0.1$, $\delta = 0.01$ for the adaptive-depth method.

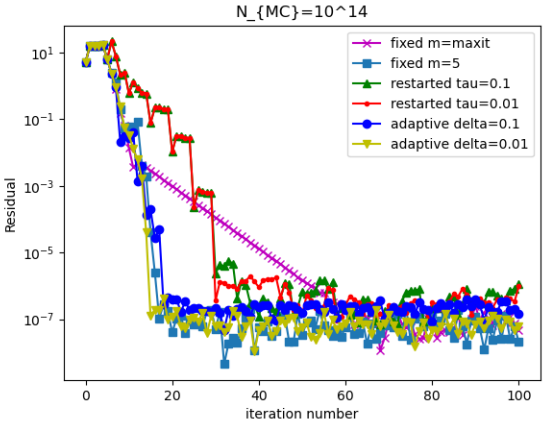
Figure 4.15 shows the convergence behavior of the Anderson Acceleration variants applied to the problem in Equation (4.3) for different values of N_{MC}. Across all variants, the residual norms level off at approximately the sampling noise magnitude $1/\sqrt{N_{\text{MC}}}$. In contrast to the linear experiment in Section 4.3.4, the residuals exhibit pronounced oscillations around this noise level, with fluctuations ranging between $0.1/\sqrt{N_{\text{MC}}}$ and $10/\sqrt{N_{\text{MC}}}$. Overall, none of the Anderson Acceleration variants achieve further improvement once the residual reaches the scale of the statistical standard deviation, confirming that convergence is ultimately limited by the inherent variance of the Monte Carlo sampling process.



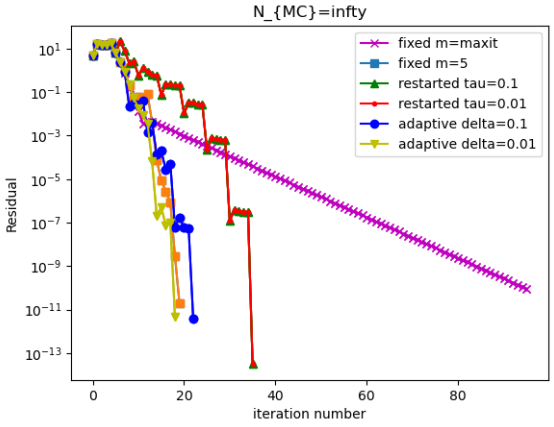
(a) Residual: $N_{MC} = 10^6$.



(b) Residual: $N_{MC} = 10^{10}$.



(c) Residual: $N_{MC} = 10^{14}$.



(d) Residual: $N_{MC} = \infty$.

Figure 4.15: Results of the experiment in Section 4.3.5.

4.4 Conclusions

This chapter examined the behavior of Anderson Acceleration and its variants under both deterministic and stochastic perturbations. Through systematic numerical experiments and theoretical interpretation, we demonstrated that perturbations in function evaluations or residual computations can significantly influence the convergence of different variants of Anderson Acceleration.

The analysis confirmed several key trends. In the presence of deterministic noise, the attainable accuracy of Anderson-type methods is bounded by the perturbation magnitude, beyond which further iterations do not improve convergence. Stochastic perturbations, modeled through random noise or Monte Carlo sampling, introduce oscillations in the residual norms, whose amplitude is proportional to the standard deviation of the perturbation. Among the tested variants, restarted and adaptive-depth Anderson methods showed improved robustness by dynamically adjusting their effective memory and mitigating the accumulation of noise-induced linear dependence.

These findings reinforce the understanding that the efficiency of acceleration schemes is based not only on the choice of update strategy but also on their sensitivity to numerical and modeling perturbations. The insights obtained here provide valuable guidance for designing stable acceleration techniques in inexact or stochastic computational environments.

Chapter 5

nlKrylov: A Unified Framework for Nonlinear GCR-type Krylov Subspace Methods

Building on the analysis of fixed-point and acceleration methods established earlier in the thesis, this chapter extends the study of convergence acceleration to the broader context of nonlinear Krylov subspace algorithms. Chapters 2 to 4 developed the theoretical foundations of iterative acceleration beginning with fixed-point formulations, then exploring extrapolation and Anderson-type schemes, and finally assessing their stability under deterministic and stochastic perturbations. While these methods are effective in many nonlinear settings, they remain inherently local: they exploit information from a limited number of past iterates without constructing a global subspace representation of the problem. Krylov subspace techniques, by contrast, achieve rapid convergence for linear problems through systematic subspace expansion and orthogonalization. Extending these ideas to nonlinear systems offers a natural pathway toward more powerful and globally convergent acceleration methods.

The purpose of this chapter is to establish a unified framework that connects Anderson Acceleration, multisecant updates, and nonlinear extensions of Krylov methods within a single mathematical structure. In particular, we focus on nonlinear generalizations of the generalized Conjugate Residual (GCR) method and related schemes, which lead to the family

of nonlinear Krylov methods collectively referred to as *nlKrylov*. These include nonlinear counterparts of GMRESR, GCRO, and LGMRES, each derived as special instances of a common abstract formulation. The framework highlights the relationships between subspace-projection approaches and multisecant or quasi-Newton updates, revealing how seemingly distinct acceleration mechanisms can be interpreted through a shared algebraic viewpoint. From a practical standpoint, this generalization enables the design of efficient solvers for large-scale nonlinear systems, combining the robustness of Anderson-type residual minimization with the global convergence characteristics of Krylov subspace methods. Moreover, it provides a systematic way to incorporate preconditioning, restarts, and adaptive depth control concepts already explored in earlier chapters within a nonlinear subspace iteration context.

The remainder of this chapter follows the structure of the accompanying research paper. Section 5.1 introduces the nonlinear GCR formulation and establishes the foundational notation used throughout. Subsequent sections present specific algorithms, theoretical properties, and numerical results, demonstrating how the proposed framework unifies and extends existing acceleration methods for nonlinear problems.

This chapter is a part of the paper "nlKrylov: A Unified Framework for Nonlinear GCR-type Krylov Subspace Methods" [119] with CRediT author statement:

Tom Werner: Conceptualization, Investigation, Methodology, Software, Writing - Original Draft.

Ning Wan: Conceptualization, Investigation, Methodology, Software, Writing - Original Draft.

Agnieszka Międlar: Conceptualization, Methodology, Supervision, Writing - Original Draft.

5.1 The Nonlinear Truncated Generalized Conjugate Residual (nlTGCR)

In [23], the Nonlinear Truncated GCR (nlTGCR) method was introduced, extending the linear GCR method [24] to systems of nonlinear equations. This approach closely relates to Anderson Acceleration and inexact Newton methods, offering a unified view of nonlinear iterative acceleration techniques. Recall that GCR solves the linear system (1.5) using the Krylov subspace (2.21), where a basis $P^{(j)} = [p^{(0)},\ p^{(1)}, \ldots,\ p^{(j)}]$ of $K_{j+1}(A, r^{(0)})$ is constructed to satisfy $(P^{(j)})^T A^T A P^{(j)} = I_{j+1}$, i.e., $P^{(j)}$ is $A^T A$-orthogonal, or equivalently, $V^{(j)} = AP^{(j)}$ is orthogonal. Orthogonality of $V^{(j)}$ is maintained using a modified Gram-Schmidt step after constructing a new direction $v^{(j+1)}$. A truncated variant, GCR(k) with truncation window $k \geq 1$, was proposed in [24] (see Algorithm 7) to limit orthogonalization costs, where $k = \infty$ recovers the full GCR. Here, only the k most recent directions $v^{(j-k+1)}, v^{(j-k+2)}, \ldots, v^{(j)}$ and $p^{(j-k+1)}, p^{(j-k+2)}, \ldots, p^{(j)}$ are stored and used for orthogonalization. Throughout this paper, we adopt the notation from [23], defining $j_k := \max\{j - k + 1, 0\}$ and

$$P^{(j)} := \left[p^{(j_k)}, p^{(j_k+1)}, \ldots, p^{(j)}\right], \quad V^{(j)} := \left[v^{(j_k)}, v^{(j_k+1)}, \ldots, v^{(j)}\right],$$

so that $P^{(j)}, V^{(j)} \in \mathbb{R}^{n\times k}$ when $j \geq k$ (truncated), and $P^{(j)}, V^{(j)} \in \mathbb{R}^{n\times(j+1)}$ otherwise. Hence, we will drop the "T" in nlTGCR(k) since all methods use truncation, and define $n_j := j - j_k + 1$ as the number of columns in $P^{(j)}$ (resp. $V^{(j)}$).

While extending the GCR method to nonlinear systems, the authors of [23] specified four key features that the generalized algorithm should exhibit.

F1 When applied to a linear problem, the nonlinear generalization (in exact arithmetic) should be mathematically equivalent to the linear algorithm.

F2 The algorithm can be adapted to fit into the inexact Newton or multi-secant framework if desired.

F3 The algorithm should exploit a more accurate linear model than Newton or Quasi-Newton methods at the cost of potential extra function evaluations.

F4 The algorithm can be adapted to handle "fuzzy" functions appearing in stochastic or machine learning applications.

To achieve **F1**–**F4**, the linear algorithm is generalized to the nonlinear setting as follows. At step j of nlGCR(k), it is assumed that there exist two matrices $P^{(j)}, V^{(j)} \in \mathbb{R}^{n \times n_j}$ with $J_f(x^{(j)})P^{(j)} \approx V^{(j)}$ that allow the local representation of the negative nonlinear residual $-r^{(j+1)} := f(x^{(j+1)})$ as

$$f(x^{(j+1)}) = f(x^{(j)} + P^{(j)}y^{(j)}) \approx f(x^{(j)}) + J_f(x^{(j)})P^{(j)}y^{(j)} \approx f(x^{(j)}) + V^{(j)}y^{(j)}, \tag{5.1}$$

with $y^{(j)} \in \mathbb{R}^{n_j}$ selected such that the residual norm $\|r^{(j+1)}\|_2$ is minimized and

$$f(x^{(j)}) + V^{(j)}y^{(j)} \ \perp \ \mathcal{R}(V^{(j)}).$$

Note that, in the case of equality in (5.1), $y^{(j)}$ is directly determined in terms of the normal equations as

$$y^{(j)} = \underset{y \in \mathbb{R}^{n_j}}{\operatorname{argmin}} \|f(x^{(j)}) + V^{(j)}y\|_2 = -(V^{(j)})^+ f(x^{(j)}) \overset{(\star)}{=} (V^{(j)})^T r^{(j)}, \tag{5.2}$$

where we assume the columns of $V^{(j)}$ to be orthonormal for $(\star)$. Once $y^{(j)}$ is computed, the update $x^{(j+1)} = x^{(j)} + P^{(j)}y^{(j)}$ is performed and $r^{(j+1)} = -f(x^{(j+1)})$ is set. The matrices $P^{(j)}$ and $V^{(j)}$ are then extended by setting $p^{(j+1)} = r^{(j+1)}$ and $v^{(j+1)} = J_f(x^{(j+1)})p^{(j+1)}$ as

well as orthogonalizing $v^{(j+1)}$ against all columns of $V^{(j)}$ and modifying $p^{(j+1)}$ accordingly. Notice that, in the linear case $f(x) = Ax - b$, $J_f(x) = A$ and, by the formulation of the GCR-algorithm,

$$V^{(j)} = AP^{(j)} = J_f(x^{(j)})P^{(j)}, \tag{5.3}$$

and

$$P^{(j)}y^{(j)} = P^{(j)}(V^{(j)})^T r^{(j)} = P^{(j)}((v^{(j)})^T r^{(j)})e_{n_j} = ((v^{(j)})^T r^{(j)})p^{(j)} \equiv \alpha^{(j)}p^{(j)}, \tag{5.4}$$

due to the orthogonality condition $r^{(j)} \perp \mathcal{R}(V^{(j-1)})$ in GCR, where $e_{n_j} \in \mathbb{R}^{n_j}$ is the n_j-th unit vector. However, equations (5.2), (5.3) and (5.4) are not true in the nonlinear version since, generally, $J_f(x^{(j+1)}) \neq J_f(x^{(j)})$ and, as such,

$$\begin{aligned} V^{(j)} &= \left[J_f(x^{(j_k)})p^{(j_k)}, J_f(x^{(j_k+1)})p^{(j_k+1)}, \ldots, J_f(x^{(j)})p^{(j)}\right] \\ &\neq J_f(x^{(j)})\left[p^{(j_k)}, p^{(j_k+1)}, \ldots, p^{(j)}\right] = J_f(x^{(j)})P^{(j)}. \end{aligned}$$

A direct comparison between GCR for linear equations and its nonlinear counterpart is presented in Algorithm 7 and Algorithm 8.

Algorithm 7 GCR(k) for $Ax = b$ [71, 72]

Input: $A \in \mathbb{R}^{n\times n}$, $x^{(0)}, b \in \mathbb{R}^n$
Output: x^* exact solution to $Ax = b$

$\widehat{p} = r^{(0)} = b - Ax^{(0)}, \quad \widehat{v} = A\widehat{p}$
$p^{(0)} = \frac{\widehat{p}}{\|\widehat{v}\|_2}, \quad v^{(0)} = \frac{\widehat{v}}{\|\widehat{v}\|_2}$
$j = 0$
while not converged **do**
  $\alpha^{(j)} = \langle v^{(j)}, r^{(j)}\rangle$
  $x^{(j+1)} = x^{(j)} + \alpha^{(j)} p^{(j)}$
  $r^{(j+1)} = r^{(j)} - \alpha^{(j)} v^{(j)}$
  $\widehat{p} = r^{(j+1)}, \widehat{v} = A\widehat{p}$
  for $i = j_k : j$ **do**
    $\beta_i = \langle \widehat{v}, v^{(i)}\rangle$
    $\widehat{p} = \widehat{p} - \beta_i p^{(i)}, \quad \widehat{v} = \widehat{v} - \beta_i v^{(i)}$
  end for
  $p^{(j+1)} = \frac{\widehat{p}}{\|\widehat{v}\|_2}, \quad v^{(j+1)} = \frac{\widehat{v}}{\|\widehat{v}\|_2}$
  $j = j + 1$
end while
return $x^* = x^{(j)}$

Algorithm 8 nlGCR(k) for $f(x) = 0$ [23]

Input: $x^{(0)} \in \mathbb{R}^n$, f, J_f
Output: x^* exact solution to $f(x) = 0$

$\widehat{p} = r^{(0)} = -f(x^{(0)}), \quad \widehat{v} = J_f(x^{(0)})\widehat{p}$
$p^{(0)} = \frac{\widehat{p}}{\|\widehat{v}\|_2}, \quad v^{(0)} = \frac{\widehat{v}}{\|\widehat{v}\|_2}$
$j = 0$
while not converged **do**
  $y^{(j)} = (V^{(j)})^T r^{(j)}$
  $x^{(j+1)} = x^{(j)} + P^{(j)} y^{(j)}$
  $r^{(j+1)} = -f(x^{(j+1)})$
  $\widehat{p} = r^{(j+1)}, \widehat{v} = J_f(x^{(j+1)})\widehat{p}$
  for $i = j_k : j$ **do**
    $\beta_i = \langle \widehat{v}, v^{(i)}\rangle$
    $\widehat{p} = \widehat{p} - \beta_i p^{(i)}, \quad \widehat{v} = \widehat{v} - \beta_i v^{(i)}$
  end for
  $p^{(j+1)} = \frac{\widehat{p}}{\|\widehat{v}\|_2}, \quad v^{(j+1)} = \frac{\widehat{v}}{\|\widehat{v}\|_2}$
  $j = j + 1$
end while
return $x^* = x^{(j)}$

5.2 From Linear to Nonlinear Krylov methods

In this section, we will derive a unified framework for generalizing linear nested Krylov methods based on GCR to nonlinear equations and present three notable members of the resulting class of methods. Following the terminology of [23], which introduces nlGCR as the nonlinear analogue of GCR, we refer to methods that fit within our framework as *nonlinear Krylov* (*nlKrylov*) methods.

We first revisit the ideas from the original nlTGCR-paper presented in Section 5.1 to develop a more general framework for this family of methods in Section 5.2.1. Afterwards, we will specify how to apply the introduced framework to three popular nested GCR-based algorithms, namely GMRESR, GCRO and LGMRES in Section 5.2.2, Section 5.2.3 and Section 5.2.4, respectively, obtaining nonlinear extensions of these methods.

5.2.1 The general nlKrylov framework

In establishing the framework for nlKrylov methods, we recall that at step j of the nlGCR algorithm, the matrices $P^{(j)}$ and $V^{(j)}$ are employed as sets of directions for progressing the iteration, as shown in lines 5 and 6 of Algorithm 8. Since in the derivation of the nlGCR, the basis matrices $P^{(j)}, V^{(j)}$ seem rather arbitrary, one might also think of improving the local linear model (5.1) at the cost of some additional function evaluations, as indicated by **F3**. Inspired by the idea of nested GCR-type methods [72], our aim is to compute $y^{(j)}$, $x^{(j)}$, and $r^{(j)}$ as in nlGCR, but with $p^{(j)}$ defined via a specified subroutine, $p^{(j)} = \mathcal{SR}_j(r^{(j)}, J_f(x^{(j)}))$, which will involve the current residual $r^{(j)}$ as well as applications of the Jacobian $J_f(x^{(j)})$. In the simplest case, where $\mathcal{SR}_j(r^{(j)}, J_f(x^{(j)})) = r^{(j)}$, the standard nlGCR algorithm is recovered. Building on the concept of nested Krylov methods, $\mathcal{SR}_j$ represents a specific algorithm for solving the linear equation

$$J_f(x^{(j)})\widehat{p} = r^{(j)}, \tag{5.5}$$

which requires only $r^{(j)}$ as well as applications of $J_f(x^{(j)})$ in the process. We will refer to every algorithm that uses nlGCR as an outer iteration and is equipped with a subroutine $\mathcal{SR}_j$ for (5.5) as a *nonlinear Krylov method*. Observe that performing a few iterations of an iterative inner solver for (5.5) effectively incorporates the inexact Newton update direction $\Delta x^{(j)}$ from (2.18) into the search space $P^{(j)}$, thereby improving the local linear model. However, as a large number of applications of $J_f(x^{(j)})$ within $\mathcal{SR}_j$ may introduce substantial computational overhead, their use should therefore be carefully balanced to enhance the local linear model (5.1) as effectively as possible without dominating the overall complexity. In [72], classical inner methods include Krylov subspace methods such as GMRES or the Bi-Conjugate Gradient Stabilized method (BiCGStab, [120]) if $J_f(x^{(j)})$ is non-symmetric and

CG for symmetric Jacobians. However, our framework is not restricted to Krylov subspace methods to obtain $p^{(j)}$. Different techniques such as classical iterative methods, randomized projection approaches or whatever algorithm is viable for subproblem (5.5) can be used. Nevertheless, in this paper, we will focus on three particular Krylov subspace approaches that will lead to nonlinear extensions of GMRESR(m,k) Section 5.2.2, GCRO(m,k) Section 5.2.3 and LGMRES(m,k) Section 5.2.4.

5.2.2 nlGMRESR

In [71], GMRESR(m) was originally introduced as a nested GCR algorithm, where m steps of GMRES for the linear system $A\widehat{p} = r^{(j)}$ are used as an inner method. By choosing $\mathcal{SR}_j(r^{(j)}, J_f(x^{(j)}))$ as taking m steps of GMRES for the subproblem (5.5), we immediately get a nonlinear extension of GMRESR(m), which we will call nlGMRESR(m). In Algorithm 9 and Algorithm 10, we provide a comparison of their truncated versions, i.e., GMRESR(m,k) for linear systems and nlGMRESR(m,k) for nonlinear equations. We expect nlGMRESR(m,k) to require fewer iterations than nlGCR(k) due to its enhanced local linear model, while the additional function evaluations introduced by the inner solve can be controlled through an appropriate choice of m.

Algorithm 9 GMRESR(m, k) [71, 72]

Input: $A \in \mathbb{R}^{n\times n}$, $m \in \mathbb{N}$, $x^{(0)}, b \in \mathbb{R}^n$
Output: x^* exact solution to $Ax = b$

1: $r^{(0)} = b - Ax^{(0)}$
2: $\widehat{p} = \texttt{GMRES}(A, r^{(0)}, m)$
3: $\widehat{v} = A\widehat{p}$
4: $p^{(0)} = \frac{\widehat{p}}{\|\widehat{v}\|_2}, \quad v^{(0)} = \frac{\widehat{v}}{\|\widehat{v}\|_2}$
5: $j = 0$
6: **while** not converged **do**
7: $\quad \alpha^{(j)} = \langle v^{(j)}, r^{(j)} \rangle$
8: $\quad x^{(j+1)} = x^{(j)} + \alpha^{(j)} p^{(j)}$
9: $\quad r^{(j+1)} = r^{(j)} - \alpha^{(j)} v^{(j)}$
10: $\quad \widehat{p} = \texttt{GMRES}(A, r^{(j+1)}, m)$
11: $\quad \widehat{v} = A\widehat{p}$
12: $\quad$ **for** $i = j_k : j$ **do**
13: $\quad\quad \beta_i = \langle \widehat{v}, v^{(i)} \rangle$
14: $\quad\quad \widehat{p} = \widehat{p} - \beta_i p^{(i)}, \quad \widehat{v} = \widehat{v} - \beta_i v^{(i)}$
15: $\quad$ **end for**
16: $\quad p^{(j+1)} = \frac{\widehat{p}}{\|\widehat{v}\|_2}, \quad v^{(j+1)} = \frac{\widehat{v}}{\|\widehat{v}\|_2}$
17: $\quad j = j + 1$
18: **end while**
19: **return** $x^* = x^{(j)}$

Algorithm 10 nlGMRESR(m, k)

Input: $x^{(0)} \in \mathbb{R}^n$, $m \in \mathbb{N}$, f, J_f
Output: x^* exact solution to $f(x) = 0$

1: $r^{(0)} = -f(x^{(0)})$
2: $\widehat{p} = \texttt{GMRES}(J_f(x^{(0)}), r^{(0)}, m)$
3: $\widehat{v} = J_f(x^{(0)})\widehat{p}$
4: $p^{(0)} = \frac{\widehat{p}}{\|\widehat{v}\|_2}, \quad v^{(0)} = \frac{\widehat{v}}{\|\widehat{v}\|_2}$
5: $j = 0$
6: **while** not converged **do**
7: $\quad y^{(j)} = (V^{(j)})^T r^{(j)}$
8: $\quad x^{(j+1)} = x^{(j)} + P^{(j)} y^{(j)}$
9: $\quad r^{(j+1)} = -f(x^{(j+1)})$
10: $\quad \widehat{p} = \texttt{GMRES}(J_f(x^{(j+1)}), r^{(j+1)}, m)$
11: $\quad \widehat{v} = J_f(x^{(j+1)})\widehat{p}$
12: $\quad$ **for** $i = j_k : j$ **do**
13: $\quad\quad \beta_i = \langle \widehat{v}, v^{(i)} \rangle$
14: $\quad\quad \widehat{p} = \widehat{p} - \beta_i p^{(i)}, \quad \widehat{v} = \widehat{v} - \beta_i v^{(i)}$
15: $\quad$ **end for**
16: $\quad p^{(j+1)} = \frac{\widehat{p}}{\|\widehat{v}\|_2}, \quad v^{(j+1)} = \frac{\widehat{v}}{\|\widehat{v}\|_2}$
17: $\quad j = j + 1$
18: **end while**
19: **return** $x^* = x^{(j)}$

Note that, after executing m steps of GMRES for $Az = r$ with $z_0 = 0$ and $c = \|r\|e_1$, we have

$$AQ_m = Q_{m+1}\underline{H_m}, \qquad \gamma_m = \underset{\gamma\in\mathbb{R}^m}{\operatorname{argmin}} \|c - \underline{H}_m\gamma\|_2 \tag{5.6}$$

$$z_m = Q_m\gamma_m, \qquad Az_m = A(Q_m\gamma_m) = Q_{m+1}(\underline{H}_m\gamma_m),$$

meaning that $\widehat{v} = J_f(x^{(j+1)})\widehat{p}$ in line 11 of Algorithm 10 can be computed from a low-rank update inside GMRES without the need of an additional function evaluation. Thus, one step of nlGMRESR(m) requires $(m-1)$ additional function evaluations compared to nlGCR, assuming that the matrix vector products $J_f(x^{(j)})q_\ell$ inside GMRES are computed

in a matrix-free way, e.g. by the finite difference formula

$$J_f(x^{(j)})q_\ell \approx \frac{f(x^{(j)}+\varepsilon q_\ell)-f(x^{(j)})}{\varepsilon} = \frac{f(x^{(j)}+\varepsilon q_\ell)+r^{(j)}}{\varepsilon}, \tag{5.7}$$

where ε is a small nonnegative scalar, and $J_f(x^{(j)})$ is never formed explicitly.

Obviously, the particular choice of m is key to the performance of the algorithm. For linear systems, a detailed discussion on the choice of m can be found in [71, §5.2]. A comprehensive analysis of nlGMRESR(m) is still an open question. In the numerical experiments presented in Section 5.7, we select m heuristically, typically between $m = 4$ and $m = 30$ depending on the problem.

5.2.3 nlGCRO

As noted in [72], a limitation of GMRESR(m) is that it treats the subproblem (5.5) in isolation, without leveraging information from the outer GCR loop to improve the GMRES-solve step. To address this, GCRO(m) has been developed as a refinement of GMRESR(m) that incorporates the GCR-basis $V^{(j)}$ in the inner solve by introducing a projection onto its orthogonal complement, i.e., instead of solving $A\widehat{p} = r^{(j+1)}$ in line 10 of Algorithm 9, GMRES is used to solve

$$\underbrace{(I - V^{(j)}(V^{(j)})^T)A}_{=:A^{\perp V^{(j)}}}\ \widehat{p} = \underbrace{(I - V^{(j)}(V^{(j)})^T)r^{(j+1)}}_{=:\widetilde{r}^{(j+1)}}, \tag{5.8}$$

where the second equation is true since, in the linear case, $r^{(j+1)} \perp \mathcal{R}(V^{(j)})$. Given that, in the linear case, $AP^{(j)} = V^{(j)}$ and

$$A^{\perp V^{(j)}} Q_m = Q_{m+1}\underline{H}_m \tag{5.9}$$

holds in the GMRES solve, (5.9) yields

$$\widehat{p} = Q_m \gamma_m, \quad \text{with} \quad \gamma_m = \underset{\gamma \in \mathbb{R}^m}{\operatorname{argmin}} \left\| \|r^{(j+1)}\|_2 e_1 - \underline{H}_m \gamma \right\|_2,$$

and after orthogonalization we get

$$\|v^{(j+1)}\|_2 p^{(j+1)} = A^{-1} A^{\perp V^{(j)}} Q_m \gamma_m = (I - A^{-1} V^{(j)} (V^{(j)})^T A) Q_m \gamma_m = (Q_m - P^{(j)} B_m) \gamma_m, \tag{5.10}$$

with $B_m = (V^{(j)})^T A Q_m \in \mathbb{R}^{n_j \times m}$ [72, Thm. 2.2]. In the nonlinear case, we generally do not have $J_f(x^{(j)}) P^{(j)} = V^{(j)}$, which means that the information in $V^{(j)}$ is not solely associated with $J_f(x^{(j)})$ but rather reflects contributions from all previously used Jacobians. As such, including a projection onto $(V^{(j)})^\perp$ in the linear solve is only beneficial when the Jacobian is changing slowly, i.e., when $J_f(x^{(j+1)}) \approx J_f(x^{(j)})$. If the subspace $\mathcal{R}(V^{(j)})$ becomes outdated relative to the subproblem (5.5), projecting $J_f(x^{(j+1)})$ onto $\mathcal{R}(V^{(j)})^\perp$ may fail to improve the local linear model and can even hinder convergence. Additionally, the second equation in (5.8) generally does not hold for nonlinear problems, since $r^{(j+1)} \not\perp \mathcal{R}(V^{(j)})$. However, (5.10) remains valid with some minor modifications, namely

$$\|v^{(j+1)}\|_2 p^{(j+1)} = (Q_m - P^{(j)} B_m) \gamma_m,$$

but now, $q_1 = \dfrac{\widetilde{r}^{(j+1)}}{\|\widetilde{r}^{(j+1)}\|_2}$ and

$$(I - V^{(j)} (V^{(j)})^T) J_f(x^{(j+1)}) Q_m = Q_{m+1} \underline{H}_m, \quad \gamma_m = \underset{\gamma \in \mathbb{R}^m}{\operatorname{argmin}} \left\| \|\widetilde{r}^{(j+1)}\|_2 e_1 - \underline{H}_m \gamma \right\|_2.$$

Following the derivations in [72], we see that after the GCR-orthogonalization process, we have

$$\|v^{(j+1)}\|_2 p^{(j+1)} = \widehat{p} - P^{(j)}(V^{(j)})^T \widehat{c} = Q_m \gamma_m - P^{(j)}(V^{(j)})^T J_f(x^{(j+1)}) Q_m \gamma_m = (Q_m - P^{(j)} B_m)\gamma_m,$$

where $B_m = (V^{(j)})^T J_f(x^{(j+1)}) Q_m$, and similarly

$$\|v^{(j+1)}\|_2 v^{(j+1)} = \widehat{c} - V^{(j)}(V^{(j)})^T \widehat{c} = (I - V^{(j)}(V^{(j)})^T) J_f(x^{(j+1)}) Q_m \gamma_m = Q_{m+1}(\underline{H}_m \gamma_m).$$

Consequently, $p^{(j+1)}$ and $v^{(j+1)}$ can be recovered directly from the inner solve (up to normalization), making orthogonalization in the outer GCR loop unnecessary, although it may still enhance numerical stability. The GCRO(m, k)-algorithm and its nonlinear counterpart are presented in Algorithm 11 and Algorithm 12, respectively.

Algorithm 11 GCRO(m, k)

Input: $A \in \mathbb{R}^{n\times n}$, $m \in \mathbb{N}$, $x^{(0)}, b \in \mathbb{R}^n$
Output: x^* exact solution to $Ax = b$

1: $r^{(0)} = b - Ax^{(0)}$
2: $\widehat{p} = \texttt{GMRES}(A, r^{(0)}, m)$
3: $\widehat{v} = A\widehat{p}$
4: $p^{(0)} = \frac{\widehat{p}}{\|\widehat{v}\|_2}, \quad v^{(0)} = \frac{\widehat{v}}{\|\widehat{v}\|_2}$
5: $j = 0$
6: **while** not converged **do**
7: $\quad \alpha^{(j)} = \langle v^{(j)}, r^{(j)} \rangle$
8: $\quad x^{(j+1)} = x^{(j)} + \alpha^{(j)} p^{(j)}$
9: $\quad r^{(j+1)} = r^{(j)} - \alpha^{(j)} v^{(j)}$
10: $\quad \widehat{p} = \texttt{GMRES}(A^{\perp V^{(j)}}, r^{(j+1)}, m)$
11: $\quad \widehat{v} = A\widehat{p}$
12: $\quad$ **for** $i = j_k : j$ **do** //Optional
13: $\quad\quad \beta_i = \langle \widehat{v}, v^{(i)} \rangle$
14: $\quad\quad \widehat{p} = \widehat{p} - \beta_i p^{(i)}, \quad \widehat{v} = \widehat{v} - \beta_i v^{(i)}$
15: $\quad$ **end for**
16: $\quad p^{(j+1)} = \frac{\widehat{p}}{\|\widehat{v}\|_2}, \quad v^{(j+1)} = \frac{\widehat{v}}{\|\widehat{v}\|_2}$
17: $\quad j = j + 1$
18: **end while**
19: **return** $x^* = x^{(j)}$

Algorithm 12 nlGCRO(m, k)

Input: $x^{(0)} \in \mathbb{R}^n$, $m \in \mathbb{N}$, f, J_f
Output: x^* exact solution to $f(x) = 0$

1: $r^{(0)} = -f(x^{(0)})$
2: $\widehat{p} = \texttt{GMRES}(J_f(x^{(0)}), r^{(0)}, m)$
3: $\widehat{v} = J_f(x^{(0)})\widehat{p}$
4: $p^{(0)} = \frac{\widehat{p}}{\|\widehat{v}\|_2}, \quad v^{(0)} = \frac{\widehat{v}}{\|\widehat{v}\|_2}$
5: $j = 0$
6: **while** not converged **do**
7: $\quad y^{(j)} = (V^{(j)})^T r^{(j)}$
8: $\quad x^{(j+1)} = x^{(j)} + P^{(j)} y^{(j)}$
9: $\quad r^{(j+1)} = -f(x^{(j+1)})$
10: $\quad \widehat{p} = \texttt{GMRES}(J_f(x^{(j+1)})^{\perp V^{(j)}}, \widetilde{r}^{(j+1)}, m)$
11: $\quad \widehat{v} = J_f(x^{(j+1)})\widehat{p}$
12: $\quad$ **for** $i = j_k : j$ **do** //Optional
13: $\quad\quad \beta_i = \langle \widehat{v}, v^{(i)} \rangle$
14: $\quad\quad \widehat{p} = \widehat{p} - \beta_i p^{(i)}, \quad \widehat{v} = \widehat{v} - \beta_i v^{(i)}$
15: $\quad$ **end for**
16: $\quad p^{(j+1)} = \frac{\widehat{p}}{\|\widehat{v}\|_2}, \quad v^{(j+1)} = \frac{\widehat{v}}{\|\widehat{v}\|_2}$
17: $\quad j = j + 1$
18: **end while**
19: **return** $x^* = x^{(j)}$

We also want to emphasize that truncation is particularly important for (nl)GCRO, as orthogonalization against $V^{(j)}$ must be performed every time the operator $J_f(x^{(j+1)})^{\perp V^{(j)}}$ is applied in the inner solve in line 10, which can become the dominant computational cost over time if no truncation is employed. Various truncation strategies for the linear solvers are discussed in [73], and automatic schemes can be adopted from adaptive acceleration techniques [17]. For simplicity, we consider here only truncation by dropping the first column of the basis corresponding to the oldest iterate.

5.2.4 nlLGMRES

Another approach for solving the linear system (1.5) that incorporates the outer space during the inner GMRES solve is the Loose-GMRES (LGMRES) algorithm [74, 75]. Here, instead of projecting onto $(V^{(j)})^{\perp}$, the outer basis is included by augmenting the inner Krylov subspace with $P^{(j)}$. Specifically, at step j, given truncated basis matrices $P^{(j)}, V^{(j)} \in \mathbb{R}^{n \times n_j}$, the inner solve iteratively builds the augmented Krylov subspace

$$A[Q_m, P^{(j)}] = Z_{m+k}\underline{H}_{m+k}$$

using Algorithm 13. In case of solving linear systems, we can replace line 7 of Algorithm 13

Algorithm 13 AGMRES(m, k) for $Ax = b$

Input: $A \in \mathbb{R}^{n\times n}$, $m, k \in \mathbb{N}$, $x^{(0)}, b \in \mathbb{R}^n$, $P = [p^{(1)}, \ldots, p^{(s)}] \in \mathbb{R}^{n\times s}$, $s \leq k$
Output: x^* exact solution to $Ax = b$

1: $r^{(0)} = b - Ax^{(0)}, \quad \beta = \|r^{(0)}\|_2$
2: $q_1 = \frac{r^{(0)}}{\beta}, \quad m_s = m + (k - s)$
3: **for** $i = 1 : m + k$ **do**
4: **if** $i \leq m_s$ **then**
5: $w = Aq_i$ //Standard GMRES
6: **else**
7: $w = Ap^{(i-m_s)}$ //Augment by P
8: **end if**
9: **for** $\ell = 1 : i$ **do**
10: $h_{\ell,i} = \langle w, q_\ell \rangle, \quad w = w - h_{\ell,i}q_\ell$ //Orthogonalize against GMRES-basis
11: **end for**
12: $h_{i+1,i} = \|w\|_2, \quad q_{i+1} = \frac{w}{h_{i+1,i}}$
13: **end for**
14: Define $Z_{m+k} = [q_1, \ldots, q_{m_s}, P] \in \mathbb{R}^{n\times(m+k)}$
15: $\gamma_{m+k} = \underset{\gamma\in\mathbb{R}^{m+k}}{\operatorname{argmin}} \|\beta e_1 - \underline{H}_{m+k}\gamma\|_2$
16: $x^{(m+k)} = x^{(0)} + Z_{m+k}t_{m+k}$
17: **return** $x^* = x^{(m+k)}$

by $w = v^{(i-m_s)}$ to avoid the extra matrix-vector multiplication, leading to the relation

$$A[Q_m, P^{(j)}] = [Q_{m+1}, V^{(j)}]\underline{H}_{m+k}. \tag{5.11}$$

We note that, for linear problems, the connection between augmentation, deflation, flexible preconditioning, and subspace recycling has been established and thoroughly analyzed in the survey [121]. For nonlinear problems, however, we generally have $v^{(j-i)} \neq J_f(x^{(j)})p^{(j-i)}$, if $i > 0$, so we need to apply the algorithm exactly as stated above. If the problem is moderately nonlinear and the Jacobian varies slowly, $v^{(i-m_s)}$ can be used instead to reduce the number of additional function evaluations. Using Algorithm 13, we can easily write the LGMRES(m, k) and nlLGMRES(m, k) algorithms in terms of GCR; see Algorithm 14 and Algorithm 15.

Algorithm 14 LGMRES(m, k)

Input: $A \in \mathbb{R}^{n\times n}$, $m, k \in \mathbb{N}$, $x^{(0)}, b \in \mathbb{R}^n$
Output: x^* exact solution to $Ax = b$

$r^{(0)} = b - Ax^{(0)}$
$p^{(0)} = \texttt{GMRES}(A, r^{(0)}, m + k)$
$v^{(0)} = Ap^{(0)}$
$j = 0$
while not converged **do**

    $x^{(j+1)} = x^{(j)} + p^{(j)}$
    $r^{(j+1)} = r^{(j)} - v^{(j)}$
    $p^{(j+1)} = \texttt{AGMRES}(A, r^{(j+1)}, m, k, P^{(j)})$
    $v^{(j+1)} = Ap^{(j+1)}$
    $j = j + 1$
end while
return $x^* = x^{(j)}$

Algorithm 15 nlLGMRES(m, k)

Input: $x^{(0)} \in \mathbb{R}^n$, $m, k \in \mathbb{N}$, f, J_f
Output: x^* solution to $f(x) = 0$

$r^{(0)} = -f(x^{(0)})$
$p^{(0)} = \texttt{GMRES}(J_f(x^{(0)}), r^{(0)}, m + k)$
$v^{(0)} = J_f(x^{(0)})p^{(0)}$
$j = 0$
while not converged **do**
    $y^{(j)} = (V^{(j)})^+ r^{(j)}$
    $x^{(j+1)} = x^{(j)} + P^{(j)}y^{(j)}$
    $r^{(j+1)} = -f(x^{(j+1)})$
    $p^{(j+1)} = \texttt{AGMRES}(J_f(x^{(j+1)}), r^{(j+1)}, m, k, P^{(j)})$
    $v^{(j+1)} = J_f(x^{(j+1)})p^{(j+1)}$
    $j = j + 1$
end while
return $x^* = x^{(j)}$

Investigating Algorithm 15 a bit closer, we have two remarks to make: First of all, the linear version of LGMRES does not incorporate a scaling term $\alpha^{(j)} = (v^{(j)})^T r^{(j)}$ in the update of $x^{(j+1)}$. However, nlGCR requires solving the inexact line-search problem (5.2) prior to com-

puting the update direction. This step can be incorporated by adding line 6 to Algorithm 15. To balance accuracy and efficiency, we approximate the solution by setting $y^{(j)} = (V^{(j)})^+ r^{(j)}$ instead of $y^{(j)} = (J_f(x^{(j)})(P^{(j)})^+ r^{(j)}$, thus favoring computational efficiency at the expense of additional inexactness. Secondly, orthogonalizing $V^{(j)}$ is not mandatory in the linear version. To improve stability and efficiently solve the least-squares problem (5.2) in line 6, we modify the base nonlinear algorithm to orthogonalize $V^{(j)}$ after line 10 of Algorithm 15. This results in a new method, which we will call nlLGMRES with Orthogonalization (nlLGMRESO) and present in Algorithm 16. Although these modifications violate **F1**, as Algorithm 16 no longer

Algorithm 16 nlLGMRESO(m, k) for $f(x) = 0$

Input: $x^{(0)} \in \mathbb{R}^n$, $m, k \in \mathbb{N}$, f, J_f
Output: x^* exact solution to $f(x) = 0$

1: $r^{(0)} = -f(x^{(0)})$
2: $\widehat{p} = \texttt{GMRES}(J_f(x^{(0)}), r^{(0)}, m + k), \quad \widehat{v} = J_f(x^{(0)})\widehat{p}$
3: $p^{(0)} = \frac{\widehat{p}}{\|\widehat{v}\|_2}, \quad v^{(0)} = \frac{\widehat{v}}{\|\widehat{v}\|_2}$
4: $j = 0$
5: **while** not converged **do**
6: $\quad y^{(j)} = (V^{(j)})^T r^{(j)}$
7: $\quad x^{(j+1)} = x^{(j)} + P^{(j)} y^{(j)}, \quad r^{(j+1)} = -f(x^{(j+1)})$
8: $\quad \widehat{p} = \texttt{AGMRES}(J_f(x^{(j+1)}), r^{(j+1)}, m, k, P^{(j)}), \quad \widehat{v} = J_f(x^{(j+1)})\widehat{p}$
9: $\quad$ **for** $i = j_k : j$ **do**
10: $\quad\quad \beta_i = \langle \widehat{v}, v^{(i)} \rangle, \quad \widehat{p} = \widehat{p} - \beta_i p^{(i)}, \quad \widehat{v} = \widehat{v} - \beta_i v^{(i)}$
11: $\quad$ **end for**
12: $\quad p^{(j+1)} = \frac{\widehat{p}}{\|\widehat{v}\|_2}, \quad v^{(j+1)} = \frac{\widehat{v}}{\|\widehat{v}\|_2}$
13: $\quad j = j + 1$
14: **end while**
15: **return** $x^* = x^{(j)}$

reduces to Algorithm 14 for linear systems due to the outer orthogonalization, it integrates naturally into the nested GCR framework and outperforms nlGMRESR in some cases, as shown in Section 5.7. Whenever we reference nlLGMRES in this paper, we are referring to Algorithm 16.

5.3 Connection of nlKrylov methods to existing nonlinear methods

5.3.1 Local updating and connection to quasi-Newton methods

Recall that in the quasi-Newton framework, the classical Newton-update (2.5) is replaced by its quasi-Newton counterpart (2.6), i.e.,

$$x^{(j+1)} = x^{(j)} - G^{(j)} f(x^{(j)}),$$

where $G^{(j)} \in \mathbb{R}^{n\times n}$ is an approximation of the inverse Jacobian $G^{(j)} \approx J_f(x^{(j)})^{-1}$. Among such methods, the notable Broyden's scheme [50] updates $G^{(j)}$ iteratively via a low-rank correction similar to (2.8) or (2.9). In what follows, we show that the proposed class of *nlKrylov methods* falls within the quasi-Newton framework discussed in Section 2.3.

In the iteration step $j \geq 1$, the nlGCR, nlGMRESR, nlGCRO and nlLGMRES all perform the update

$$\begin{aligned} x^{(j+1)} &= x^{(j)} + P^{(j)}(V^{(j)})^T r^{(j)} = x^{(j)} + P^{(j)}(V^{(j)})^T(-f(x^{(j)})) = x^{(j)} - P^{(j)}(V^{(j)})^T f_j, \\ r^{(j+1)} &= -f(x^{(j+1)}), \end{aligned} \tag{5.12}$$

with the sole difference between the methods arising from how the $P^{(j)}$ and $V^{(j)}$ are selected via the corresponding local linear model (5.1). From (5.12), we can immediately see that any *nlKrylov* method can be viewed as a quasi-Newton method with $G^{(j)} \approx P^{(j)}(V^{(j)})^T$ being used as an approximation to the inverse Jacobian $J_f(x^{(j)})^{-1}$. Although this has already been observed for nlGCR in [23], this result in fact extends to all nlKrylov methods. The difference between the methods comes from the basis matrices $P^{(j)}$ and $V^{(j)}$ constructed by

each of the algorithms through the subproblem (5.5). For each of the methods introduced in Section 5.2, the underlying space used to update vector p at iteration j, prior to the Gram-Schmidt orthogonalization, is given in Table 5.1. We also display the number of function evaluations, i.e., matrix-free products of $J_f(x^{(j+1)})$ computed via (5.7), and inner products required to perform m steps of the subroutines, assuming k columns in $P^{(j)}$ and $V^{(j)}$.

Method	Space for $\widehat{p}$	# fevals	# inner products
nlGCR(k)	span$\{r^{(j+1)}\}$	1	0
nlGMRESR(m,k)	$\mathcal{K}_m((J_f(x^{(j+1)}), r^{(j+1)})$	$m-1$	$\frac{m(m+1)}{2}+1$
nlGCRO(m,k)	$\mathcal{R}(P^{(j)}) \oplus \mathcal{K}_m(J_f(x^{(j+1)})^{\perp V^{(j)}}, \widetilde{r}_{j+1})$	$m-1$	$\frac{(m+1)(m+2k)}{2}+1$
nlLGMRES(m,k)	$\mathcal{R}(P^{(j)}) \oplus \mathcal{K}_m(J_f(x^{(j+1)}), r^{(j+1)})$	$m+k-1$	$\frac{(m+k)(m+k+1)}{2}+1$

Table 5.1: Underlying subspaces, # of necessary function evaluations and inner products for various *nlKrylov* methods.

Note that in [23], a connection between nlTGCR(k) and Anderson Acceleration with truncation (AA(k), [1]) has been established through the framework of multi-secant methods. Thus, the *nlKrylov* family can be regarded as nonlinear acceleration methods for fixed-point schemes.

5.3.2 Connection to nonlinear Orthomin

It is well known that, for linear problems, the truncated version of the GCR method, denoted GCR(k), is equivalent to Orthomin(k) [24]. Extensions of the classical Orthomin [63] to nonlinear problems have been first proposed in the form of nonlinear Orthomin(1) [122] and later generalized to nonlinear Orthomin(k) in [123] for $k \geq 2$. In Algorithm 17, we recall the nonlinear Orthomin(k) algorithm, with notation consistent with the conventions of this chapter.

Algorithm 17 nlOrthomin(k) for $f(x) = 0$ [123, Alg. 3]

Input: $x^{(0)} \in \mathbb{R}^n$, $k \in \mathbb{N}$, f, J_f

Output: x^* solution to $f(x) = 0$

1: $p^{(0)} = r^{(0)} = -f(x^{(0)})$

2: $j = 0$

3: **while** not converged **do**

4: Solve $y^{(j)} = \underset{y \in \mathbb{R}^j}{\operatorname{argmin}} \|f(x^{(j)} + P^{(j)}y)\|_2$ (stagnation for $y_j = 0$)

5: $x^{(j+1)} = x^{(j)} + P^{(j)}y^{(j)}$

6: $r^{(j+1)} = -f(x^{(j+1)})$

7: $p^{(j+1)} = r^{(j+1)} - \sum_{i=j_k}^{j} \beta_i^{(j)} p^{(i)}$, where $\beta_i^{(j)} = \frac{\left(J_f(x^{(j+1)})r^{(j+1)}\right)^T \left(J_f(x^{(j+1)})p^{(i)}\right)}{\|J_f(x^{(j+1)})p^{(i)}\|_2^2}$

8: $j = j + 1$

9: **end while**

10: **return** $x^* = x^{(j)}$

Two significant differences emerge when comparing Algorithm 17 and Algorithm 8. Firstly, in line 4 of nlOrthomin(k), the vector $y^{(j)}$ is the exact minimizer of $\|f(x^{(j)} + d^{(j)})\|_2$ along $d^{(j)} \in \mathcal{R}(P^{(j)})$. In nlGCR($k$), $y^{(j)} = V_j^T r^{(j)}$ is the minimizer of the linearized problem

$$\begin{aligned}\min_y \|f(x^{(j)} + P^{(j)}y)\|_2 &\approx \min_y \|f(x^{(j)}) + J_f(x^{(j)})P^{(j)}y\|_2 \\ &\approx \min_y \|f(x^{(j)}) + V^{(j)}y\|_2 = \min_y \|r^{(j)} - V^{(j)}y\|_2,\end{aligned}$$

where we use a first order Taylor expansion of f to get the first approximation and relation $J_f(x^{(j)})P^{(j)} \approx V^{(j)}$ for the second approximation (see (5.1)). If these approximations are *sufficiently* accurate, the nlGCR update is essentially equivalent to a Gauss-Newton step

[124] for the exact minimization problem

$$\min_y \|s_j(y)\|, \quad \text{where} \quad s_j : \mathbb{R}^{n_j} \to \mathbb{R}^n,\ y \mapsto f(x^{(j)} + P^{(j)}y), \tag{5.13}$$

i.e., (5.13) is solved by updating

$$y^{(i+1)} = y^{(i)} + \Delta y^{(i)}, \quad \text{where} \quad \left(J_{s_j}(y^{(i)})^T J_{s_j}(y^{(i)})\right) \Delta y^{(i)} = -J_{s_j}(y^{(i)})^T s_j(y^{(i)}). \tag{5.14}$$

Since $J_{s_j}(y) = J_f(x^{(j)})P^{(j)}$, if we assume $J_f(x^{(j)})P^{(j)} = V^{(j)}$, with $(V^{(j)})^T V^{(j)} = I_{n_j}$, and choose $y^{(0)} = 0$, the update (5.14) yields

$$\begin{aligned} y^{(1)} &= y^{(0)} - \left(J_{s_j}(y^{(0)})^T J_{s_j}(y^{(0)})\right)^{-1} J_{s_j}(y^{(0)})^T s_j(y^{(0)}) \\ &= -((V^{(j)})^T V^{(j)})^{-1}(V^{(j)})^T s_j(0) = -(V^{(j)})^T f(x^{(j)}) = (V^{(j)})^T r^{(j)}, \end{aligned}$$

which is equivalent to the inexact line search step (5.2) in line 5 of Algorithm 8. Secondly, in nlGCR, the matrix $V^{(j)}$ is stored as an additional set of vectors and is successively orthogonalized as each new pair $(p^{(j)}, v^{(j)})$ is computed, ensuring $(V^{(j)})^T V^{(j)} = I_{n_j}$ and $V^{(j)} \approx J_f(x^{(j)})P^{(j)}$. In contrast, in nlOrthomin(k), $V^{(j)} = J_f(x^{(j)})P^{(j)}$ holds exactly and is recomputed at each iteration using the current Jacobian applied to $P^{(j)}$. The orthogonalization of $V^{(j)}$ is performed implicitly using the coefficients $\beta_i^{(j)}$ before $V^{(j)}$ is discarded, which corresponds to the classical Gram-Schmidt process in line 7 of Algorithm 17. The similarity between nlGCR(k) and nlOrthomin(k) primarly depends on the quality of the local linearization of $f(x^{(j)} + P^{(j)}y^{(j)})$, i.e., on how slowly the Jacobian varies during the iteration, which– following [23]–can be measured by the error matrix

$$W^{(j)} = J_f(x^{(j)})P^{(j)} - V^{(j)}. \tag{5.15}$$

Hence, if $\|W^{(j)}\|_2$ is small, nlGCR(k) and nlOrthomin(k) should exhibit comparable behavior. In [123], it is also shown that preconditioning nlOrthomin(k) with a constant preconditioner $\mathbf{M}$, i.e.,

$$0 = g(z) = f(\mathbf{M}z), \quad z = \mathbf{M}^{-1}x,$$

can significantly improve convergence. In this context,

$$J_g(z)p = J_f(x)\mathbf{M}p,$$

and the coefficients $\beta_i^{(j)}$ are given by

$$\beta_i^{(j)} = \frac{(J_f(x^{(j+1)})\mathbf{M}r^{(j+1)})^T(J_f(x^{(j+1)})p^{(i)})}{\|J_f(x^{(j+1)})p^{(i)}\|_2^2}.$$

In nlGMRESR(m,k), nlGCRO(m,k) and nlLGMRES(m,k), the vector $p^{(j+1)}$ is computed by performing m (resp. $(m+k)$) steps of a GMRES-type method on $r^{(j+1)}$ using $J_f(x^{(j+1)})$ as the coefficient matrix. As discussed in [123], this approach can be viewed as a flexibly preconditioned version of nlOrthomin(k), which can be generalized to choosing $\mathcal{SR}_j$ as the preconditioner. Upon comparing *nlKrylov* methods to nlOrthomin, one main difference in behavior was observed. The convergence, measured by the number of iterations, was comparable between nlOrthomin and nlGCR, despite the fact that nlOrthomin employed an exact line-search, whereas nlGCR relied on the linear model. However, because the exact line-search involves an inner optimization, the number of function evaluations of f was significantly higher for nlOrthomin than for most other *nlKrylov* methods.

5.3.3 Connection to subspace projection methods

In [125], a *Preconditioned Subspace Projection* approach is proposed to accelerate the convergence of Newton's method, which can be summarized as the simple three step Algorithm 18.

Algorithm 18 Preconditioned Projection Newton method

Input: $x^{(0)} \in \mathbb{R}^n$, $k \in \mathbb{N}$, f, J_f

Output: x^* solution to $f(x) = 0$

1: $j = 0$
2: **while** not converged **do**
3: Choose a *Preconditioner* $Y^{(j)} \in \mathbb{R}^{n \times k}$ and form the matrix $J^{(j)} = J_f(x^{(j)})Y^{(j)}$
4: Solve $J^{(j)}y^{(j)} = -f(x^{(j)})$ for $y^{(j)} \in \mathbb{R}^k$, e.g. using $y^{(j)} = -(J^{(j)})^+ f(x^{(j)})$
5: Update $x^{(j+1)} = x^{(j)} + Y^{(j)}y^{(j)}$
6: $j = j + 1$
7: **end while**
8: **return** $x^* = x^{(j)}$

Note that the term *Preconditioner* in the context of Algorithm 18 should not be confused with the preconditioner $\mathbf{M}$ used in Section 5.3.2 for Krylov subspace methods. In Algorithm 18, $Y^{(j)}$ can be understood as a set of carefully selected directions that "sketch" the action of the current Jacobian $J_f(x^{(j)})$, enabling the Newton update equation to be solved within this subspace. If we assume that locally, the Jacobian $J_f(x^{(j)})$ does not change too much (see Assumption 5.1 for details) and we choose $Y^{(j)} = P^{(j)}$ in iteration j (assuming $n_j = k$), then

$$J^{(j)} = J_f(x^{(j)})P^{(j)} \approx V^{(j)}.$$

Hence, assuming $V^{(j)}$ is orthogonal and $r^{(j)} = -f(x^{(j)})$ is the nonlinear residual, the optimal

mixing parameter $y^{(j)} \in \mathbb{R}^k$ in line 4 of Algorithm 18 can be computed as

$$y^{(j)} = (V^{(j)})^{+}(-f(x^{(j)})) = (V^{(j)})^{T} r^{(j)}.$$

Note that this is exactly the update (5.2) within the *nlKrylov framework*. In [125], Algorithm 18 is called a *Preconditioned Angle Algorithm* (PAA), if at each step, some column $z \in \mathbb{R}^n$ of $J^{(j)}$ satisfies the angle criterion

$$\cos^2(\theta) = \frac{((r^{(j)})^T z)^2}{\|r^{(j)}\|_2^2 \cdot \|z\|_2^2} > \tau, \quad \tau \in (0,1), \tag{5.16}$$

with angle $\theta = \angle(z, r^{(j)})$ and tolerance τ. Local convergence of PAA is ensured by [125, Theorem 5], under the assumption that $J_f(x^*)$ is invertible at a solution $x^* \in \mathbb{R}^n$ to $f(x) = 0$. The proof frames PAA as an inexact Newton method, measuring the inexactness by means of $Y^{(j)}$ and $J^{(j)}$ as well as (5.16). In the following section, we present a comprehensive convergence analysis of *nlKrylov* methods, combining both theoretical insights and practical implications, and employing a similar approach based on $P^{(j)}$ and $V^{(j)}$.

5.4 Convergence analysis of nlKrylov methods

The goal of this section is to examine the convergence behavior of nlKrylov methods, beginning with problems that have nonsingular Jacobians at the solution and then addressing those with singular Jacobians.

5.4.1 Problems with nonsingular Jacobian

Suppose that at iteration j, the spaces $P^{(j)}, V^{(j)} \in \mathbb{R}^{n\times n_j}$ have been constructed, with $(V^{(j)})^T V^{(j)} = I_{n_j}$. Let $W^{(j)} \in \mathbb{R}^{n\times n_j}$ denote the error matrix defined in (5.15), such that

$$W^{(j)} = J_f(x^{(j)})P^{(j)} - V^{(j)} \;\Leftrightarrow\; V^{(j)} = J_f(x^{(j)})P^{(j)} - W^{(j)}. \tag{5.17}$$

Then, assuming that the Jacobian is locally invertible in the neighborhood of x^*–and in particular invertible at $x^{(j)}$–we obtain from Section 5.3.3, upon defining

$$Y^{(j)} = P^{(j)} - J_f(x^{(j)})^{-1}W^{(j)} \in \mathbb{R}^{n\times n_j},$$

directly the relation

$$J^{(j)} = J_f(x^{(j)})Y^{(j)} = J_f(x^{(j)})P^{(j)} - W^{(j)} = V^{(j)} \in \mathbb{R}^{n\times n_j}$$

from (5.17). Thus, we can compute $y^{(j)}$ by solving $y^{(j)} = -(J^{(j)})^+ f(x^{(j)}) = -(V^{(j)})^T f(x^{(j)})$, which is precisely how it is computed in the *nlKrylov* framework. This relation provides a natural interpretation of nlKrylov methods through the lens of preconditioned projection methods, where the dependence on the underlying subspaces is explicit and these subspaces are dictated by the choice of the subroutine $\mathcal{SR}_j$. We use this perspective to further analyze the convergence behavior of nlKrylov methods. In [23, Thm. 4.5], the convergence of nlGCR was established under the assumption of an exact line search, smoothness of $f(x)$, the existence of uniform bounds $\eta, \mu \in [0, 1)$ for the local linear model

$$\|W^{(j)}y^{(j)}\|_2 \leq \mu\|f(x^{(j)})\|_2, \tag{5.18}$$

and a sufficient decrease of the linearized residual

$$\|f(x^{(j)}) + V^{(j)}y^{(j)}\|_2 \leq \eta \|f(x^{(j)})\|_2. \tag{5.19}$$

As we will see, in our analysis we do not require the first two assumptions and can relax the uniform bounds (5.18) and (5.19) as follows.

Assumption 5.1. *Let* $\mu_j := \dfrac{\|W^{(j)}y^{(j)}\|_2}{\|f(x^{(j)})\|_2}$ *and* $\eta_j := \dfrac{\|f(x^{(j)}) + V^{(j)}y^{(j)}\|_2}{\|f(x^{(j)})\|_2}$. *Then, for all* j *there exists a constant* $c > 0$ *such that*

$$\mu_j + \eta_j =: c_j \leq c < 1, \tag{5.20}$$

Note that, within the PAA framework, η_j can be interpreted as the angle between $f(x^{(j)})$ and $\mathcal{R}(V^{(j)})$; see (5.16). The convergence of the nlKrylov methods can now be established by leveraging [125, Theorem 5].

Theorem 5.2. *Let* $f(x)$ *be continuously differentiable in a neighborhood* $B_1(x^*)$ *of a solution* x^* *to* $f(x) = 0$, *with* $J_f(x^*)$ *nonsingular. Suppose that Assumption 5.1 holds. Then there exists a neighborhood* $B_0(x^*)$ *of* x^* *such that, for any* $x^{(0)} \in B_0(x^*)$, *the nlKrylov iterates* $\{x^{(j)}\}_j$ *converge to* x^*.

Proof. Following the proof of [125, Theorem 5], it is sufficient to show that nlKrylov methods are in fact inexact Newton methods, with the ratio $\dfrac{\|t^{(j)}\|_2}{\|f(x^{(j)})\|_2}$ uniformly bounded by a constant less than 1, where $t^{(j)}$ denotes the deviation from the exact update (see (2.18)). Local convergence then follows from [61, Theorem 2.3]. Within our nlKrylov formulation, we have

$$\Delta x^{(j)} = x^{(j+1)} - x^{(j)} = -P^{(j)}(V^{(j)})^T f(x^{(j)})$$

and

$$t^{(j)} = J_f(x^{(j)})\Delta x^{(j)} + f(x^{(j)}) = (I - J_f(x^{(j)})P^{(j)}(V^{(j)})^T)f(x^{(j)}).$$

Since (5.15) yields $J_f(x^{(j)})P^{(j)} = W^{(j)} + V^{(j)}$, we obtain

$$t^{(j)} = J_f(x^{(j)})\Delta x^{(j)} + f(x^{(j)}) = (I - V^{(j)}(V^{(j)})^T - W^{(j)}(V^{(j)})^T)f(x^{(j)}). \tag{5.21}$$

Then, by (5.2) and Assumption 5.1, we get

$$\|W^{(j)}(V^{(j)})^T f(x^{(j)})\|_2 = \|W^{(j)}y^{(j)}\|_2 = \mu_j\|f(x^{(j)})\|_2.$$

Thus, using (5.21) and denoting by $\theta_j \in [0, \pi/2]$ the angle between $f(x^{(j)})$ and $\mathcal{R}(V^{(j)})$, gives

$$\begin{aligned}
\|t^{(j)}\|_2^2 &= \|f(x^{(j)})^T(I - V^{(j)}(V^{(j)})^T)^T(I - V^{(j)}(V^{(j)})^T)f(x^{(j)}) \\
&\quad - f(x^{(j)})^T(I - V^{(j)}(V^{(j)})^T)^T W^{(j)}(V^{(j)})^T f(x^{(j)}) \\
&\quad - f(x^{(j)})^T V^{(j)}(W^{(j)})^T(I - V^{(j)}(V^{(j)})^T)f(x^{(j)}) \\
&\quad + f(x^{(j)})^T V^{(j)}(W^{(j)})^T W^{(j)}(V^{(j)})^T f(x^{(j)})\|_2 \\
&\leq \|f(x^{(j)})\|_2^2 \sin^2\theta_j + 2\|f(x^{(j)})\|_2^2\mu_j \sin\theta_j + \|f(x^{(j)})\|_2^2\mu_j^2 \\
&= \|f(x^{(j)})\|_2^2(\sin\theta_j + \mu_j)^2.
\end{aligned}$$

By Assumption 5.1,

$$\begin{aligned}
\eta_j\|f(x^{(j)})\|_2 &= \|f(x^{(j)}) + V^{(j)}y^{(j)}\|_2 = \|f(x^{(j)}) - V^{(j)}(V^{(j)})^T f(x^{(j)})\|_2 \\
&= \|(I - V^{(j)}(V^{(j)})^T)f(x^{(j)})\|_2 = \|f(x^{(j)})\|_2 \sin(\theta_j),
\end{aligned}$$

and hence $\sin(\theta_j) = \eta_j$. Consequently, local convergence is guaranteed by (5.20) and [61,

Theorem 2.3] as

$$\frac{\|t^{(j)}\|_2}{\|f(x^{(j)})\|_2} \leq \eta_j + \mu_j \leq c_j \leq c < 1.$$

□

Remark 5.3. Up to this point, we have relied on the implicit assumption that the Jacobian does not vary rapidly during the iteration process. However, this assumption has not been explicitly quantified. In our analysis, it is captured by (5.15) and Assumption 5.1: the alignment between $\mathcal{R}(V^{(j)})$ and $\mathcal{R}(J_f(x^{(j)})P^{(j)})$ and consequently the small norm of the error matrix $W^{(j)}$ implicitly assumes a slowly varying linear model. Imposing stronger conditions, such as Lipschitz continuity of $J_f(x)$, would lead to assumptions similar to [23, Ass. 2], which require a uniform bound on the quadratic term of the Taylor expansion of $f(x)$ – a condition that is too restrictive for our framework. Our result, Theorem 5.2, is formulated in terms of the subspaces $\mathcal{R}(P^{(j)})$ and $\mathcal{R}(V^{(j)})$, allowing more flexibility in selecting the subroutine $\mathcal{SR}_j$.

5.4.2 Problems with singular Jacobian

Extensive results exist on convergence of Newton's method for singular problems, where the derivative at the root is singular. Reddien [126, 127] considered C^3 functions with a one-dimensional null space of the Jacobian. These results were extended by Decker, Keller, Kelley, and Suresh [128, 129, 130, 131], who provided detailed convergence rate analyses under various scenarios. Griewank [132, 133] showed a starlike domain for which Newton's method converges linearly to the root.

For inexact Newton methods applied to singular problems, Kelley and Xue [134] provided convergence results analogous to [61, Theorem 2.3], while Shores and Tiahrt [125] showed

that subspace projected inexact Newton methods converge whenever the corresponding exact Newton method does. Argyros [135] extended the results of Decker, Keller, and Kelley [130] to inexact Newton schemes. Convergence of quasi-Newton methods, such as Broyden's method, for singular problems has also been studied in [136, 137]. For singular systems, including underdetermined and overdetermined cases, several results [138, 139, 140, 141, 142, 143] establish convergence of inexact Newton methods under weak Lipschitz or majorant conditions. Although nlKrylov methods can be interpreted within the inexact Newton framework, the forcing sequence $c_j = \mu_j + \eta_j$ is not explicitly user-controlled; rather, it is implicitly determined by the Krylov subspace construction and the associated projections at each iteration. Consequently, the aforementioned convergence results do not apply directly. Nevertheless, under additional assumptions on $t^{(j)}$, generalized convergence results for singular systems can be derived by leveraging the frameworks of [130] and [134].

Assumption 5.4. *Let $J_f(x^*)$ have a one dimensional null space $\mathcal{N}$ spanned by $\phi \in \mathbb{R}^n$ and closed range $\mathcal{X}$ such that $\mathbb{R}^n = \mathcal{N} \oplus \mathcal{X}$. In addition, let f be twice Lipschitz continuously differentiable in a neighborhood of x^*, and for any projection $P_{\mathcal{N}}$ onto $\mathcal{N}$ parallel to $\mathcal{X}$, we have*

$$P_{\mathcal{N}} H_f(x^*)(\phi, \phi) \neq 0,$$

where $H_f(x)$ denotes the Hessian of a (nonlinear) function $f : \mathbb{R}^n \to \mathbb{R}^n$.

Let $\widetilde{x} = x - x^*$ for $x \in \mathbb{R}^n$ and $P_{\mathcal{X}} = I - P_{\mathcal{N}}$. For singularities satisfying Assumption 5.4, a neighborhood of x^* for singular problems can be defined as

$$W(\rho, \gamma) = \{x \in \mathbb{R}^n \mid 0 < \|\widetilde{x}\|_2 < \rho, \|P_{\mathcal{X}}\widetilde{x}\|_2 \leq \gamma \|P_{\mathcal{N}}\widetilde{x}\|_2\},$$

with $\rho > 0$ and $\gamma > 0$ sufficiently small. Let

$$A(x) = P_{\mathcal{X}} J_f(x) P_{\mathcal{X}}, \qquad A_1(x)(\cdot) = P_{\mathcal{X}} H_f(x^*)(\widetilde{x}, P_{\mathcal{X}}(\cdot)), \tag{5.22}$$

$$B(x) = P_{\mathcal{X}} J_f(x) P_{\mathcal{N}}, \qquad B_1(x)(\cdot) = P_{\mathcal{X}} H_f(x^*)(\widetilde{x}, P_{\mathcal{N}}(\cdot)), \tag{5.23}$$

$$C(x) = P_{\mathcal{N}} J_f(x) P_{\mathcal{X}}, \qquad C_1(x)(\cdot) = P_{\mathcal{N}} H_f(x^*)(\widetilde{x}, P_{\mathcal{X}}(\cdot)), \tag{5.24}$$

$$D(x) = P_{\mathcal{N}} J_f(x) P_{\mathcal{N}}, \qquad D_1(x)(\cdot) = P_{\mathcal{N}} H_f(x^*)(\widetilde{x}, P_{\mathcal{N}}(\cdot)),$$

and

$$\bar{D}_1(x)(\cdot) := P_{\mathcal{N}} H_f(x^*)(P_{\mathcal{N}}\widetilde{x}, P_{\mathcal{N}}(\cdot)),$$

be a linear map from $\mathcal{N}$ to $\mathcal{N}$.

Lemma 5.5. *If $\bar{D}_1(x)(\cdot)$ is invertible whenever $P_{\mathcal{N}}\widetilde{x} \neq 0$, there exist constants $\rho > 0$ and $\gamma > 0$ such that $J_f(x)$ is invertible in the region $W(\rho, \gamma)$, and*

$$J_f(x)^{-1} = P_{\mathcal{N}} D_1(x)^{-1} P_{\mathcal{N}} + \mathcal{O}(1).$$

Remark 5.6. The invertibility of $J_f(x)$ is discussed in [126, 128, 130, 131].

Based on [130, Theorem 5.1], and assumption $\|t^{(j)}\|_2 \leq c\|f(x^{(j)})\|_2^2$, we have the following convergence result for the cases covered under Assumption 5.4.

Theorem 5.7. *Let Assumption 5.4 hold, and assume for all $P_{\mathcal{N}}\widetilde{x} \neq 0$, that $\bar{D}_1$ is nonsingular as a map on $\mathcal{N}$. If $\|t^{(j)}\|_2 \leq c\|f(x^{(j)})\|_2^2$ for some c, then for $\rho > 0$ and $\gamma > 0$ sufficiently small, $f(x^{(0)})^{-1}$ exists for all $x^{(0)} \in W(\rho, \gamma)$, and the sequence $\{x^{(j)}\}_j$ of iterates generated by a nlKrylov method converges to x^* with*

$$\|P_{\mathcal{X}}(x^{(j+1)} - x^*)\|_2 \leq K_1 \|x^{(j)} - x^*\|_2^2, \tag{5.25}$$

for some $K_1 > 0$*, and*

$$\lim_{j\to\infty} \frac{\|P_{\mathcal{N}}(x^{(j+1)} - x^*)\|_2}{\|P_{\mathcal{N}}(x^{(j)} - x^*)\|_2} = \frac{1}{2}. \tag{5.26}$$

Proof. Expansion of f and J_f around x^* gives

$$f(x) = J_f(x^*)\widetilde{x} + \frac{1}{2}[A_1(x) + B_1(x) + C_1(x) + D_1(x)]\widetilde{x} + \mathcal{O}(\|\widetilde{x}\|_2^3), \tag{5.27}$$

$$J_f(x) = J_f(x^*) + [A_1(x) + B_1(x) + C_1(x) + D_1(x)] + \mathcal{O}(\|\widetilde{x}\|_2^2), \tag{5.28}$$

and (5.28) yields

$$J_f(x)\widetilde{x} = J_f(x^*)\widetilde{x} + [A_1(x) + B_1(x) + C_1(x) + D_1(x)]\widetilde{x} + \mathcal{O}(\|\widetilde{x}\|_2^3).$$

Therefore,

$$f(x) = J_f(x)\widetilde{x} - \frac{1}{2}[A_1(x) + B_1(x) + C_1(x) + D_1(x)]\widetilde{x} + \mathcal{O}(\|\widetilde{x}\|_2^3).$$

Also, from (5.27) we have $f(x) = J_f(x^*)\widetilde{x} + \mathcal{O}(\|\widetilde{x}\|_2^2) = J_f(x^*)P_{\mathcal{X}}\widetilde{x} + \mathcal{O}(\|\widetilde{x}\|_2^2)$, which shows that there is a constant $K > 0$ such that

$$\|f(x)\|_2 \;\le\; K\|P_{\mathcal{X}}\widetilde{x}\|_2 \;\le\; \gamma K\|P_{\mathcal{N}}\widetilde{x}\|_2 \;\le\; \gamma K\|\widetilde{x}\|_2.$$

Thus, $\|t^{(j)}\|_2 \;\le\; c\|f(x^{(j)})\|_2^2 \;\le\; c\gamma^2K^2\|\widetilde{x}^{(j)}\|_2^2$, i.e., $\|t^{(j)}\|_2 = \gamma^2\mathcal{O}(\|\widetilde{x}_j\|_2^2)$. It follows from

(5.5) that

$$\begin{aligned}\widetilde{x}^{(j+1)} =&\widetilde{x}^{(j)} - J_f(x^{(j)})^{-1}(f(x^{(j)}) - t^{(j)})\\ =&J_f(x^{(j)})^{-1}\left(\frac{1}{2}[A_1(x^{(j)}) + B_1(x^{(j)}) + C_1(x^{(j)}) + D_1(x^{(j)})]\widetilde{x}^{(j)} + t^{(j)} + \mathcal{O}(\|\widetilde{x}^{(j)}\|_2^3)\right)\\ =&\frac{1}{2}\big(P_{\mathcal{N}}D_1(x^{(j)})^{-1}P_{\mathcal{N}} + \mathcal{O}(1)\big)\big([A_1(x^{(j)}) + B_1(x^{(j)}) + C_1(x^{(j)}) + D_1(x^{(j)})]\widetilde{x}^{(j)}\\ &+ t^{(j)} + \mathcal{O}(\|\widetilde{x}^{(j)}\|_2^3)\big).\end{aligned}$$

We have by equations (5.22)-(5.24) that

$$P_{\mathcal{N}}A_1(x^{(j)}) = P_{\mathcal{N}}B_1(x^{(j)}) = 0 \quad \text{and} \quad C_1(x^{(j)})\widetilde{x}^{(j)} = \gamma\mathcal{O}(\|\widetilde{x}^{(j)}\|_2^2).$$

Thus,

$$\widetilde{x}^{(j+1)} = \frac{1}{2}P_{\mathcal{N}}\widetilde{x}^{(j)}+\gamma P_{\mathcal{N}}\mathcal{O}(\|\widetilde{x}^{(j)}\|_2)+\gamma\mathcal{O}(\|\widetilde{x}^{(j)}\|_2^2)+\gamma^2 P_{\mathcal{N}}\mathcal{O}(\|\widetilde{x}^{(j)}\|_2)+\gamma^2\mathcal{O}(\|\widetilde{x}^{(j)}\|_2^2)+\mathcal{O}(\|\widetilde{x}^{(j)}\|_2^3). \tag{5.29}$$

Apply $P_{\mathcal{X}}$ to (5.29). There exists a constant $K_1 > 0$ such that

$$\|P_{\mathcal{X}}\widetilde{x}^{(j+1)}\|_2 \le K_1\|\widetilde{x}^{(j)}\|_2^2,$$

proving (5.25). Apply $P_{\mathcal{N}}$ to (5.29). For small enough $\gamma > 0$, there exists a constant $K_0 > 0$ with $K_0\gamma < \frac{1}{2}$ such that

$$\big(\frac{1}{2} - K_0\gamma\big)\|P_{\mathcal{N}}\widetilde{x}^{(j)}\|_2 \le \|P_{\mathcal{N}}\widetilde{x}^{(j+1)}\|_2 \le \big(\frac{1}{2} + K_0\gamma\big)\|P_{\mathcal{N}}\widetilde{x}^{(j)}\|_2.$$

Now we prove the convergence by induction starting with $j = 0$. To show $x^{(1)} \in W(\rho, \gamma)$, we set $\rho_j = \|x^{(j)} - x^*\|_2$, $\gamma_0 = \gamma$, and note that $\rho_0 \le \|P_{\mathcal{N}}\widetilde{x}^{(0)}\|_2 + \|P_{\mathcal{X}}\widetilde{x}^{(0)}\|_2 \le (1+\gamma_0)\|P_{\mathcal{N}}\widetilde{x}^{(0)}\|_2$.

Then

$$\|P_{\mathcal{X}}\widetilde{x}^{(1)}\|_2 \leq K_1\rho_0^2 \leq K_1\rho_0(1+\gamma_0)\|P_{\mathcal{N}}\widetilde{x}^{(0)}\|_2 \leq K_1\rho_0(1+\gamma_0)(\frac{1}{2}-K_0\gamma_0)^{-1}\|P_{\mathcal{N}}\widetilde{x}^{(1)}\|_2.$$

By letting $\gamma_{j+1} = K_1\rho_j(1+\gamma_j)(\frac{1}{2}-K_0\gamma_j)^{-1}$, we have $\|P_{\mathcal{X}}\widetilde{x}^{(1)}\|_2 \leq \gamma_1\|P_{\mathcal{N}}\widetilde{x}^{(1)}\|_2$ and $x^{(1)} \in W(\rho_1,\gamma_1)$. Also,

$$\rho_1 \leq \|P_{\mathcal{N}}\widetilde{x}^{(1)}\|_2+\|P_{\mathcal{X}}\widetilde{x}^{(1)}\|_2 \leq (\frac{1}{2}+K_0\gamma)\|P_{\mathcal{N}}\widetilde{x}^{(0)}\|_2+K_1\rho_0^2 \leq \left((\frac{1}{2}+K_0\gamma)(1+\gamma_0)^{-1}+K_1\rho_0\right)\rho_0.$$

For ρ_0,γ_0 small enough, there exists $\tau \in (\frac{1}{2},1)$ such that $\rho_1 < \tau\rho_0$, $\gamma_1 < \tau\gamma_0$.
By induction with the same process, we can get $\rho_{j+1} < \tau\rho_j$, $\gamma_{j+1} < \tau\gamma_j$, $x^{(j)} \in W(\rho_j,\gamma_j)$ for all j. Thus $x^{(j)} \to x^*$ with a convergence rate less than τ, and

$$\lim_{j\to\infty}\frac{\|P_{\mathcal{N}}(x^{(j+1)}-x^*)\|_2}{\|P_{\mathcal{N}}(x^{(j)}-x^*)\|_2} = \frac{1}{2}.$$

□

Since Theorem 5.7 and its proof are rather technical, we illustrate the underlying result using a simple example. Consider the problem

$$f(x) = \begin{bmatrix} x_1+x_2^2 \\ \frac{3}{2}x_1x_2+x_2^2+x_2^3 \end{bmatrix}, \tag{5.30}$$

discussed in [130]. It is straightforward to verify that $x^* = [0,0]^T$ is a root of f, with the null space of $J_f(x^*)$ given by $\mathcal{N} = \text{span}\{\phi\}$, where $\phi = [0,1]^T$, so that $\dim(\mathcal{N}) = 1$. Moreover, $P_{\mathcal{N}}H_f(x^*)(\phi,\phi) \neq 0$, showing that (5.30) satisfies Assumption 5.4. We solve (5.30) using nl-GCR(2) and nlGMRESR(2,2) under two different configurations. In Figure 5.1, the domain $\Omega = [-0.1,0.1]^2$ is discretized with $N = 200$ grid points per dimension, and the convergence

rates of both methods are evaluated for various initial values. In Figure 5.2, we select two initial points at $x_l^{(0)} = [0.1, 1]^T$ and $x_r^{(0)} = [0.001, 0.05]^T$, and show the trajectories of the iterates generated by the two methods, overlaid with contour lines of $\|f(x)\|_2$.

From Figure 5.1, we observe that for both methods, convergence along the null space (the x_2-axis) is considerably slower than in most other regions of the plane. In contrast, convergence orthogonal to the null space, along the x_1-axis, is the fastest. For nlGMRESR, an additional diagonal region connecting $p_1 = [-0.06, 0.1]^T$ and $p_2 = [0.06, -0.1]^T$ also exhibits rapid convergence. Overall, nlGMRESR demonstrates superior convergence behavior compared to nlGCR. In both plots, white points refer to initial values that did not lead to convergence within 100 steps, either due to very slow convergence rates or stagnation.

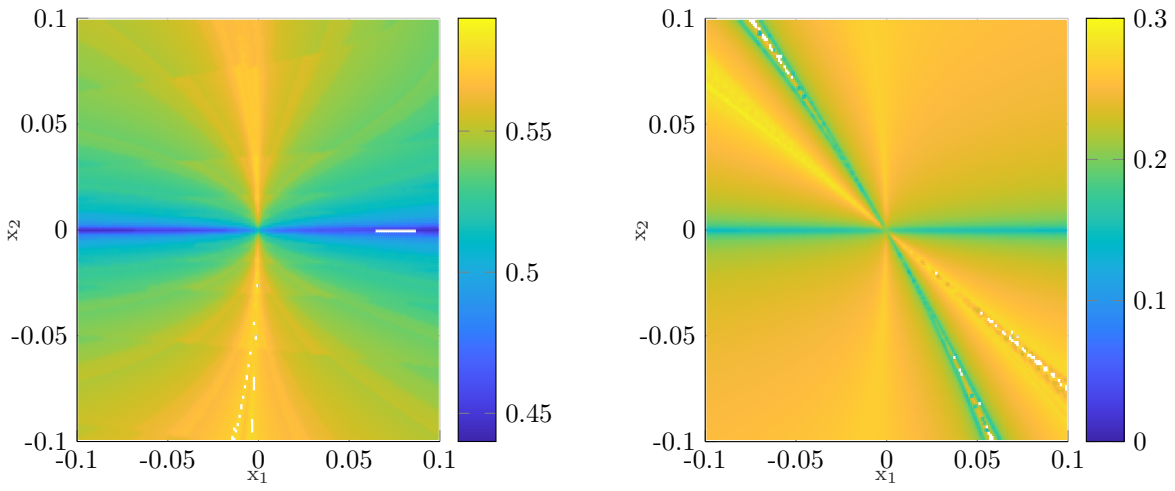


Figure 5.1: Convergence map for nlGCR(2) (left) and nlGMRESR(2,2) (right) on $\Omega = [-0.1, 0.1]^2$

To further examine the convergence behavior along the null space, we consider the two initial guesses introduced above and visualize the corresponding convergence trajectories in Figure 5.2, where the blue dashed line represents the null space $\mathcal{N}(J_f(x^*))$.

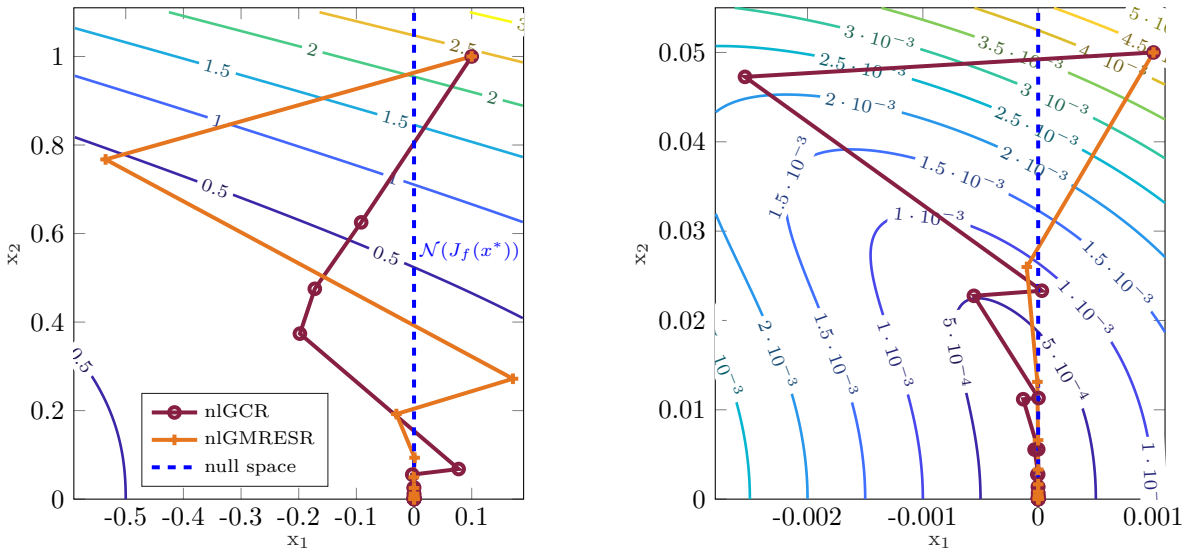


Figure 5.2: Convergence trajectories of nlGCR(2) and nlGMRESR(2,2) for starting vectors $x_l^{(0)} = [0.1, 1]^T$ (left) and $x_r^{(0)} = [0.001, 0.05]^T$ (right).

Note that both methods follow this direction as they approach the root, with nlGMRESR reaching the null space in fewer steps and remaining closer to it over successive iterations. Upon convergence, the algorithm achieved an approximate twofold reduction in x_2 per step, indicating that (5.26) holds.

5.5 Practical implementation

5.5.1 Linear and nonlinear updates

In [23], an adaptive version of nlTGCR(k) was developed to reduce the number of function evaluations by adapting the residual update from the original (linear) GCR algorithm. That is, if an angle condition is satisfied, we replace the nonlinear residual

$$r_{nl}^{(j+1)} = -f(x^{(j+1)}) \tag{5.31}$$

in line 7 of Algorithm 8 by the linearized residual

$$r_{lin}^{(j+1)} = r^{(j)} - V^{(j)} y^{(j)}. \tag{5.32}$$

Assume we have used (5.31) so far, i.e., $r^{(j)} \equiv r_{nl}^{(j)}$ in (5.32). Then, if the cosine angle between the two residuals is sufficiently close to 1, i.e.,

$$\theta_{j+1} := 1 - \frac{(r_{nl}^{(j+1)})^T r_{lin}^{(j+1)}}{\|r_{nl}^{(j+1)}\|_2 \|r_{lin}^{(j+1)}\|_2} < \theta \tag{5.33}$$

for some predefined threshold θ (see (5.16)), we adaptively switch to the linear version and check periodically, if the local linear model is still valid by computing the nonlinear residual (5.31) and checking (5.33) for a violation. If the angle grows too large, we switch back to the nonlinear update version. Since in that case, the subspaces $\mathcal{R}(P^{(j)})$, $\mathcal{R}(V^{(j)})$ created by the linear update version are likely to contain inaccurate directions that can slow down or even impede convergence, we discard $P^{(j)}$ and $V^{(j)}$ and effectively restart the algorithm using $x^{(0)} = x^{(j)}$. This strategy can be directly tranferred to the nlKrylov framework as it depends only on the quality of the local linear model (5.1).

Remark 5.8. In the case where the linearized residual (5.32) is considered, we also use the linearized augmentation (5.11) by $w = v^{(i-m_s)}$ instead of $w = J_f(x)p^{(i-m_s)}$ in the inner solve by Algorithm 13 of nlLGMRES to further reduce function evaluations.

5.5.2 Automatic restarts

In the standard GCR-algorithm, we have $(V^{(j)})^T V^{(j)} = I_{n_j}$. Moreover, since $V^{(j)} = AP^{(j)}$, $P^{(j)}$ is orthogonal with respect to the inner product induced by $A^T A$, i.e.,

$$\delta_{i,j} = \langle v^{(i)}, v^{(j)} \rangle = \langle Ap^{(i)}, Ap^{(j)} \rangle = (p^{(i)})^T (A^T A) p^{(j)} := \langle p^{(i)}, p^{(j)} \rangle_{A^T A}.$$

If A is *well-conditioned*, the orthogonality of $V^{(j)}$ is well-preserved in finite precision, inducing $A^T A$-orthogonality of $P^{(j)}$ and ensuring a numerically stable procedure. However, in the nonlinear case, we generally do not have $V^{(j)} = AP^{(j)}$ since $A_j = J_f(x^{(j)})$ changes in every step. To avoid ill-conditioning of $P^{(j)}$ caused by the loss of $A_j^T A_j$-orthogonality, we employ the automatic restart strategy presented in [144]. Consider the Gram-Schmidt process in the j-th step of the nlGCR, i.e., lines 8-13 of Algorithm 8. With $\beta_i = \widehat{v}^T v^{(i)}$, we expand $P^{(j)}$ by

$$p^{(j+1)} = \frac{1}{\|\widehat{v}\|_2} \left(r^{(j+1)} - \sum_{i=j_k}^{j} \beta_i p^{(i)} \right) = \frac{1}{\|\widehat{v}\|_2} \left(\widehat{p} - \sum_{i=j_k}^{j} \beta_i p^{(i)} \right).$$

If we assume rounding errors on $p^{(i)}$, i.e., the numerically stored basis matrices are $\widetilde{p}^{(i)} = p^{(i)} + \varepsilon_i$, $i = j_k, \ldots, j$, and an error on $p^{(j+1)}$ denoted by δ_{j+1}, which is assumed to be bounded by $\|\delta_{j+1}\|_\infty \leq \frac{C}{\|\widehat{v}\|} \|\widehat{p}\|_\infty$, we get an element-wise estimate of the error ε_{j+1} by

$$|\varepsilon_{j+1}| \leq \frac{1}{\|\widehat{v}\|_2} (C \cdot \|\widehat{p}\|_\infty + \sum_{i=j_k}^{j} |\beta_i| \|\varepsilon_i\|_\infty),$$

meaning we can keep track of the level of ill-conditioning by introducing the scalar sequence

$$w_{j+1} = \frac{1}{\|\widehat{v}\|_2} (C \cdot \|\widehat{p}\|_\infty - \sum_{i=j_k}^{j} |\beta_i| w_i), \quad w_0 = 0, \tag{5.34}$$

and restarting whenever $w_{j+1} > \tau$, where $\tau > 0$ is a user-defined tolerance. In the case of general nlKrylov methods, we have to replace $\widehat{p} = r^{(j+1)}$ in (5.34) by the corresponding preconditioned residual $\widehat{p} = \mathcal{SR}_{j+1}(r^{(j+1)}, J_f(x^{(j+1)}))$ before the Gram-Schmidt orthogonalization. If ill-conditioning is detected and the algorithm is restarted, we set $P^{(j)} = V^{(j)} = \emptyset$ and restart using

$$P^{(j+1)} = \frac{\widehat{p}}{\|J_f(x^{(j+1)})\widehat{p}\|_2}, \quad V^{(j+1)} = \frac{J_f(x^{(j+1)})\widehat{p}}{\|J_f(x^{(j+1)})\widehat{p}\|_2}, \quad w_{j+1} = C\frac{\|\widehat{p}\|_\infty}{\|J_f(x^{(j+1)})\widehat{p}\|_2}.$$

In our implementation, we use $C = 1$ and $\tau = 10^3$ as default parameters and refer the interested reader to [144] for a discussion of different parameter choices.

5.5.3 Damping and choosing descent directions

Within the quasi-Newton framework, $d^{(j)} = P^{(j)}y^{(j)} = P^{(j)}(V^{(j)})^T r^{(j)}$ can be interpreted as an inexact update direction obtained using the approximation $J_f(x^{(j)})^{-1} \approx P^{(j)}(V^{(j)})^T$. Hence, we cannot guarantee that $d^{(j)}$ is a descent direction for $f(x)$ [145]. Therefore, if $d^{(j)}$ is not a descent direction, its negation $-d^{(j)}$ should be used to ensure a decrease of $f(x^{(j)} + \alpha^{(j)}d^{(j)})$ along the update direction. This condition can be verified by computing

$$\zeta^{(j)} := \langle r^{(j)}, J_f(x^{(j)})d^{(j)}\rangle.$$

If $\zeta^{(j)} > 0$, then $d^{(j)}$ is a descent direction, otherwise,

$$-\zeta^{(j)} = -\langle r^{(j)}, J_f(x^{(j)})d^{(j)}\rangle = \langle r^{(j)}, J_f(x^{(j)})(-d^{(j)})\rangle$$

is likely to be positive and we use $-d^{(j)}$ as an update direction instead. Additionally, we want to perform a simple line search to ensure that $\alpha^{(j)} \in (0, 1]$ satisfies the Armijo-Goldstein

condition [146], i.e.,

$$\|f(x^{(j)}+\alpha^{(j)}d^{(j)})\|_2^2 \le \|f(x^{(j)})\|_2^2 - c_1\alpha^{(j)}\langle r^{(j)}, J_f(x^{(j)})d^{(j)}\rangle = \|r^{(j)}\|_2^2 - c_1\alpha^{(j)}\zeta^{(j)}. \quad (5.35)$$

Note that, within the backtracking procedure, $\zeta^{(j)}$ can be inexpensively approximated using the finite difference approximation, i.e.,

$$\zeta^{(j)} = \langle r^{(j)}, J_f(x^{(j)})d^{(j)}\rangle \approx \frac{1}{\varepsilon}\langle r^{(j)}, f(x^{(j)}+\varepsilon d^{(j)}) - f(x^{(j)})\rangle = \frac{1}{\varepsilon}\langle r^{(j)}, f(x^{(j)}+\varepsilon d^{(j)}) + r^{(j)}\rangle,$$

where $\varepsilon = \alpha_0^{(j)}$ is used before the backtracking loop to check if $d^{(j)}$ is a descent direction and the estimate is refined inside the loop using $\varepsilon = \alpha_\ell^{(j)} = \frac{\alpha_0^{(j)}}{2^\ell}$. Also, we can split $\zeta_\ell^{(j)}$ into

$$\zeta_\ell^{(j)} = \frac{1}{\alpha_\ell^{(j)}}\langle r^{(j)}, f(x^{(j)}+\alpha_\ell^{(j)}d^{(j)}) + r^{(j)}\rangle = \frac{1}{\alpha_\ell^{(j)}}\left(\|r^{(j)}\|_2^2 + \langle r^{(j)}, f(x^{(j)}+\alpha_\ell^{(j)}d^{(j)})\rangle\right)$$

to save one vector addition. If a more accurate estimate of $\zeta^{(j)}$ is needed, numerical extrapolation can be used, e.g., by finding an interpolating polynomial $p(\alpha)$ for the data points $(\alpha_i^{(j)}, \zeta_i^{(j)})$, $i = 0, \dots, \ell$, and evaluating it at $\alpha = 0$, using the Neville-Aitken algorithm. In our implementation, we use $c_1 = 10^{-3}$ and adapt the initial step-size heuristic from [23], where it is suggested to use $\alpha_0^{(0)} = 1$ and

$$\alpha_{j+1}^{(0)} = \begin{cases} \min\{1, 2\alpha_0^{(j)}\}, & \text{if } \alpha_0^{(j)} \text{ was accepted in iteration } j \\ \frac{\alpha_0^{(j)}}{2}, & \text{else} \end{cases}$$

to avoid small steps while simultaneously reducing the number of line-search steps. The detailed statement of line-search is given in Algorithm 19.

Note that, if the linear update is performed, the damped linear residual

$$r_{lin}^{(j+1)} = r^{(j)} - \alpha^{(j)} V^{(j)} y^{(j)}$$

is used on the left-hand side of (5.35) instead of $f(x^{(j)} + \alpha^{(j)} d^{(j)})$.

Algorithm 19 Armijo-Goldstein-Linesearch

Input: $f, x^{(j)}, d^{(j)}, r^{(j)}, c_1, \alpha_0^{(j)}$

Output: $\alpha^{(j)}$

Compute $f_{j+1}^{(0)} = f(x^{(j)} + \alpha_0^{(j)} d^{(j)})$ and estimate $\zeta_0^{(j)} = \frac{1}{\alpha_0^{(j)}} \langle r^{(j)}, f_{j+1}^{(0)} + r^{(j)} \rangle$
if $\zeta_0^{(j)} < 0$ **then**
    Set $d^{(j)} = -d^{(j)}$ and $\zeta_0^{(j)} = -\zeta_0^{(j)}$, recompute $f_{j+1}^{(0)} = f(x^{(j)} + \alpha_0^{(j)} d^{(j)})$
end if
Set $\ell = 0$
while $\|f_{j+1}^{(\ell)}\|_2^2 > \|r^{(j)}\|_2^2 - c_1 \alpha_\ell^{(j)} \zeta_\ell^{(j)}$ **do**
    Set $\alpha_{\ell+1}^{(j)} = \frac{\alpha_\ell^{(j)}}{2}$ and compute damped step $f_{j+1}^{(\ell+1)} = f(x^{(j)} + \alpha_{\ell+1}^{(j)} d^{(j)})$
    If desired, refine $\zeta_{\ell+1}^{(j)} = \frac{1}{\alpha_{\ell+1}^{(j)}} \langle r^{(j)}, f_{j+1}^{(\ell+1)} + r^{(j)} \rangle$, else, just use $\zeta_{\ell+1}^{(j)} = \zeta_\ell^{(j)}$
    Set $\ell = \ell + 1$
end while
return $\alpha^{(j)} = \alpha_\ell^{(j)}$

5.6 Global nlKrylov methods for nonlinear matrix equations

All strategies introduced in Section 5.2 can be easily modified to solve nonlinear matrix equations of the form

$$F(X) = 0, \quad F : \mathbb{R}^{n\times p} \to \mathbb{R}^{n\times p},\ p \leq n. \tag{5.36}$$

For problems of this form, the Newton iteration (2.5) can be generalized by replacing the system involving the Jacobian $J_f(x^{(j)})$ with a linear matrix equation

$$L_F(X^{(j)}, \Delta X^{(j)}) = -F(X^{(j)}), \tag{5.37}$$

where $L_F(X^{(j)}) : \mathbb{R}^{n\times p} \to \mathbb{R}^{n\times p}$ is the Fréchet derivative of F at $X^{(j)}$. In Section 5.6.1, we briefly recall the idea of global Krylov solvers for linear operator equations and afterwards, in Section 5.6.2, we explain how to combine these ideas and the results on nlKrylov methods introduced in this paper to acquire global *nlKrylov* methods for nonlinear matrix equations. For the remainder of this section, we use capital letters to denote matrices and calligraphic letters the denote the columnwise concatenation of multiple matrices into a block matrix. Specifically, if $U_1, \ldots, U_j$ are matrices of size $n \times p$, we define the corresponding block matrix $\mathcal{U}_j = [U_1, U_2, \ldots, U_j] \in \mathbb{R}^{n\times p\cdot j}$. On such block matrices, we define the matrix-vector product

$$\mathcal{U}_j \diamondsuit \gamma = \mathcal{U}_j(\gamma \otimes I_p) = \sum_{i=1}^{j} \gamma_i U_i, \quad \gamma = (\gamma_i)_{i=1,\ldots,j} \in \mathbb{R}^j. \tag{5.38}$$

Finally, we denote by $\langle \cdot, \cdot \rangle$ and $\|\cdot\|$ the Frobenius inner product and norm, respectively. For a matrix $W \in \mathbb{R}^{n \times p}$ and a block matrix $\mathcal{U}_j \in \mathbb{R}^{n \times p \cdot j}$, we define

$$\langle \mathcal{U}_j, W \rangle = \begin{bmatrix} \langle U_1, W \rangle \\ \vdots \\ \langle U_j, W \rangle \end{bmatrix} \in \mathbb{R}^j \quad \text{and} \quad \langle \mathcal{U}_j, \mathcal{U}_j \rangle = \begin{bmatrix} \langle U_1, U_1 \rangle & \cdots & \langle U_1, U_j \rangle \\ \vdots & \ddots & \vdots \\ \langle U_j, U_1 \rangle & \cdots & \langle U_j, U_j \rangle \end{bmatrix} \in \mathbb{R}^{j \times j}.$$

We call the block matrix $\mathcal{U}_j$ *F-orthogonal*, if $\langle \mathcal{U}_j, \mathcal{U}_j \rangle = I_j$. Additionally, for a linear operator $\mathcal{A} : \mathbb{R}^{n \times p} \to \mathbb{R}^{n \times p}$, and an F-orthogonal block matrix $\mathcal{U}_j$, we denote by $\mathcal{A}^{\perp \mathcal{U}_j} := (I - \mathcal{U}_j \diamondsuit \langle \mathcal{U}_j, \cdot \rangle)\mathcal{A}$ the operator such that

$$\mathcal{A}^{\perp \mathcal{U}_j}(X) := ((I - \mathcal{U}_j \diamondsuit \langle \mathcal{U}_j, \cdot \rangle)\mathcal{A})(X) = \mathcal{A}(X) - \mathcal{U}_j \diamondsuit \langle \mathcal{U}_j, \mathcal{A}(X) \rangle,$$

i.e., the operator that orthogonalizes $\mathcal{A}(X)$ against $\mathcal{U}_j$ with respect to the Frobenius inner product. It is easy to see that this is the matrix-valued equivalent to the GCRO-operator introduced in (5.9), hence, we use the same notation.

5.6.1 Global linear methods

Global Krylov subspace methods [78] have been introduced in recent years as an efficient iterative tool for solving linear matrix equations

$$\mathcal{A}(X) = B, \quad \mathcal{A} : \mathbb{R}^{n \times p} \to \mathbb{R}^{n \times p}, \quad B \in \mathbb{R}^{n \times p}, \tag{5.39}$$

where usually $p \ll n$ and $\mathcal{A}$ is a linear operator that is inexpensive to evaluate. Classical examples include Sylvester equations

$$A_1 X + X A_2 = B,$$

where $A_1 \in \mathbb{R}^{n\times n}$ and $A_2 \in \mathbb{R}^{p\times p}$ [81, 82] are large and sparse, or linear systems with multiple right hand sides [80]. Let us recall that, given an initial residual $R^{(0)} = B - \mathcal{A}(X^{(0)})$, $X^{(0)} \in \mathbb{R}^{n\times p}$, global GMRES [78] seeks to find a solution of the linear equation (5.39) in the block Krylov-subspace

$$\mathcal{K}_j^{\square}(\mathcal{A}, R^{(0)}) = \text{blockspan}(R^{(0)}, \mathcal{A}(R^{(0)}), \ldots, \mathcal{A}^{j-1}(R^{(0)})).$$

After m steps of global GMRES, we obtain the global Arnoldi-relation

$$\mathcal{A}\mathcal{Q}_m = \mathcal{Q}_{m+1} \Diamond \underline{H}_m,$$

where $\mathcal{Q}_{m+1} \in \mathbb{R}^{n\times(m+1)\cdot p}$ is F-orthogonal and $\underline{H}_m \in \mathbb{R}^{(m+1)\times m}$ is upper Hessenberg with an extra row (see (5.6)). Note that the size of $\underline{H}_m$ depends only on the size of the Krylov subspace and is not affected by the block size p of the problem. The global Krylov methods rely on the fact that every linear operator $\mathcal{A} : \mathbb{R}^{n\times p} \to \mathbb{R}^{n\times p}$ has a matrix representation $\mathcal{M}_{\mathcal{A}} \in \mathbb{R}^{n\cdot p\times n\cdot p}$ such that

$$\text{vec}(\mathcal{A}(X)) = \mathcal{M}_{\mathcal{A}} \cdot \text{vec}(X).$$

Hence, solving (5.39) by m steps of global GMRES is mathematically equivalent to solving the vectorized equation

$$\mathcal{M}_{\mathcal{A}} x = b, \quad \text{where} \quad x = \text{vec}(X),\ b = \text{vec}(B) \in \mathbb{R}^{n\cdot p},$$

by standard GMRES. Although, to the best of our knowledge it has not been done, GCR-based nested Krylov subspace methods can also be extended to linear matrix equations by using the Frobenius inner product in step 5 and 10 of Algorithm 7 and the Frobenius norm for normalization. Analogously, GMRESR, GCRO and LGMRES can be adapted by adding global GMRES as the inner solver. As shown in [83], global Krylov methods can serve as efficient inner solvers for (5.37) in the context of matrix-valued Newton methods. In the remainder of this section, we extend the nonlinear Krylov methods discussed in Section 5.2 to matrix-valued problems.

5.6.2 Global nonlinear methods

Using the notation introduced in Section 5.2 and the ideas of global Krylov subspace methods, we can easily extend Algorithm 8, Algorithm 10, Algorithm 12 and Algorithm 16 to the case of nonlinear matrix equations. For this sake, we will denote by $\mathcal{P}^{(j)}, \mathcal{V}^{(j)} \in \mathbb{R}^{n \times n_j \cdot p}$ the truncated block basis matrices of global GCR, i.e.,

$$\mathcal{P}^{(j)} = [P^{(j_k)}, \dots, P^{(j)}], \quad \mathcal{V}^{(j)} = [V^{(j_k)}, \dots, V^{(j)}] = \mathcal{A}\mathcal{P}^{(j)},$$

and require three minor modifications to the vector-valued nlKrylov framework: First, we have to replace all inner products and norms in the base algorithms by their Frobenius counterparts. Second, matrix-vector multiplications have to be performed using the $\lozenge$-operator (5.38) instead of the standard matrix-vector-product, e.g., to update $X^{(j+1)} = X^{(j)} + \mathcal{P}^{(j)} \lozenge y^{(j)}$. Note that computing $y^{(j)} = (V^{(j)})^T r^{(j)}$ requires column-wise inner products instead of standard matrix-vector multiplication, i.e., $y^{(j)} = \langle \mathcal{V}^{(j)}, R^{(j)} \rangle \in \mathbb{R}^{n_j}$. Finally, all matrix-vector multiplications involving the Jacobian, e.g., $J_f(x^{(j)})z$, $z \in \mathbb{R}^n$, have to be replaced by evaluations of the Fréchet derivative $L_F(X^{(j)}, Z)$ of F at $X^{(j)} \in \mathbb{R}^{n \times p}$ in the

direction of $Z \in \mathbb{R}^{n\times p}$. Hence, global Krylov methods can also be used to efficiently solve the inner equation (5.5). In the next section, we will comment on the efficient implementation of these modifications using **MATLAB**'s high level linear algebra routines. The details on the resulting algorithms are given in Algorithm 20 – Algorithm 23. Using the three modifications, the general framework from Section 5.2.1 can in principle be extended to any scheme $\mathcal{SR}_j(R^{(j)}, L_F(X^{(j)}))$ that employs $R^{(j)}$ together with $L_F(X^{(j)}, \cdot) : \mathbb{R}^{n\times p} \to \mathbb{R}^{n\times p}$ for the refinement of the local linear model (5.1) in order to solve nonlinear matrix equations (5.36), thereby encompassing the broader class of *global nlKrylov methods*.

Algorithm 20 GL-nlGCR(k)

Input: $X^{(0)} \in \mathbb{R}^{n\times p}$, F, L_F
Output: X^* solution to $F(X) = 0$

$R^{(0)} = -F(X^{(0)})$
$\widehat{P} = R^{(0)}$
$\widehat{V} = L_F(X^{(0)}, \widehat{P})$
$P_0 = \frac{\widehat{P}}{\|\widehat{V}\|_2}, \quad V_0 = \frac{\widehat{V}}{\|\widehat{V}\|_2}$
$j = 0$
while not converged **do**
  $y^{(j)} = \langle \mathcal{V}^{(j)}, R^{(j)} \rangle$
  $X^{(j+1)} = X^{(j)} + \mathcal{P}^{(j)} \diamondsuit y^{(j)}$
  $R^{(j+1)} = -F(X^{(j+1)})$
  $\widehat{P} = R^{(j+1)}$
  $\widehat{V} = L_F(X^{(j+1)}, \widehat{P})$
  for $i = j_k : j$ **do**
    $\beta_i = \langle \widehat{V}, V^{(i)} \rangle$
    $\widehat{P} = \widehat{P} - \beta_i P^{(i)}, \quad \widehat{V} = \widehat{V} - \beta_i V^{(i)}$
  end for
  $P^{(j+1)} = \frac{\widehat{P}}{\|\widehat{V}\|_2}, \quad V^{(j+1)} = \frac{\widehat{V}}{\|\widehat{V}\|_2}$
  $j = j + 1$
end while
return $X^* = X^{(j)}$

Algorithm 21 GL-nlGMRESR(m, k)

Input: $X^{(0)} \in \mathbb{R}^{n\times p}$, $m \in \mathbb{N}$, F, L_F
Output: X^* solution to $F(X) = 0$

$R^{(0)} = -F(X^{(0)})$
$\widehat{P} = \texttt{GLGMRES}(L_F(X^{(0)}), R^{(0)}, m)$
$\widehat{V} = L_F(X^{(0)}, \widehat{P})$
$P_0 = \frac{\widehat{P}}{\|\widehat{V}\|_2}, \quad V_0 = \frac{\widehat{V}}{\|\widehat{V}\|_2}$
$j = 0$
while not converged **do**
  $y^{(j)} = \langle \mathcal{V}^{(j)}, R^{(j)} \rangle$
  $X^{(j+1)} = X^{(j)} + \mathcal{P}^{(j)} \diamondsuit y^{(j)}$
  $R^{(j+1)} = -F(X^{(j+1)})$
  $\widehat{P} = \texttt{GLGMRES}(L_F(X^{(j+1)}), R^{(j+1)}, m)$
  $\widehat{V} = L_F(X^{(j+1)}, \widehat{P})$
  for $i = j_k : j$ **do**
    $\beta_i = \langle \widehat{V}, V^{(i)} \rangle$
    $\widehat{P} = \widehat{P} - \beta_i P^{(i)}, \quad \widehat{V} = \widehat{V} - \beta_i V^{(i)}$
  end for
  $P^{(j+1)} = \frac{\widehat{P}}{\|\widehat{V}\|_2}, \quad V^{(j+1)} = \frac{\widehat{V}}{\|\widehat{V}\|_2}$
  $j = j + 1$
end while
return $X^* = X^{(j)}$

Algorithm 22 GL-nlGCRO(m,k)

Input: $X^{(0)} \in \mathbb{R}^{n\times p}$, $m \in \mathbb{N}$, F, L_F
Output: X^* solution to $F(X) = 0$

$R^{(0)} = -F(X^{(0)})$
$\widehat{P} = \texttt{GLGMRES}(L_F(X^{(0)}), R^{(0)}, m)$
$\widehat{V} = L_F(X^{(0)}, \widehat{P})$
$P_0 = \frac{\widehat{P}}{\|\widehat{V}\|_2}, \quad V_0 = \frac{\widehat{V}}{\|\widehat{V}\|_2}$
$j = 0$
while not converged **do**
  $y^{(j)} = \langle \mathcal{V}^{(j)}, R^{(j)} \rangle$
  $X^{(j+1)} = X^{(j)} + \mathcal{P}^{(j)} \diamondsuit y^{(j)}$
  $R^{(j+1)} = -F(X^{(j+1)})$
  $\widehat{P} = \texttt{GLGMRES}(L_F(X^{(j+1)})^{\perp \mathcal{V}^{(j)}}, \widetilde{R}_{j+1}, m)$
  $\widehat{V} = L_F(X^{(j+1)}, \widehat{P})$
  for $i = j_k : j$ **do**
    $\beta_i = \langle \widehat{V}, V^{(i)} \rangle$
    $\widehat{P} = \widehat{P} - \beta_i P^{(i)}, \quad \widehat{V} = \widehat{V} - \beta_i V^{(i)}$
  end for
  $P^{(j+1)} = \frac{\widehat{P}}{\|\widehat{V}\|_2}, \quad V^{(j+1)} = \frac{\widehat{V}}{\|\widehat{V}\|_2}$
  $j = j + 1$
end while
return $X^* = X^{(j)}$

Algorithm 23 GL-nlLGMRESO(m,k)

Input: $X^{(0)} \in \mathbb{R}^{n\times p}$, $m, k \in \mathbb{N}$, F, L_F
Output: X^* solution to $F(X) = 0$

$R^{(0)} = -F(X^{(0)})$
$\widehat{P} = \texttt{GLFGMRES}(L_F(X^{(0)}), R^{(0)}, m + k)$
$\widehat{V} = L_F(X^{(0)}, \widehat{P})$
$P_0 = \frac{\widehat{P}}{\|\widehat{V}\|_2}, \quad V_0 = \frac{\widehat{V}}{\|\widehat{V}\|_2}$
$j = 0$
while not converged **do**
  $y^{(j)} = \langle \mathcal{V}^{(j)}, R^{(j)} \rangle$
  $X^{(j+1)} = X^{(j)} + \mathcal{P}^{(j)} \diamondsuit y^{(j)}$
  $R^{(j+1)} = -F(X^{(j+1)})$
  $\widehat{P} = \texttt{GLAGMRES}(L_F(X^{(j+1)}), R^{(j+1)}, m, k, \mathcal{P}^{(j)})$
  $\widehat{V} = L_F(X^{(j+1)}, \widehat{P})$
  for $i = j_k : j$ **do**
    $\beta_i = \langle \widehat{V}, V^{(i)} \rangle$
    $\widehat{P} = \widehat{P} - \beta_i P^{(i)}, \quad \widehat{V} = \widehat{V} - \beta_i V^{(i)}$
  end for
  $P^{(j+1)} = \frac{\widehat{P}}{\|\widehat{V}\|_2}, \quad V^{(j+1)} = \frac{\widehat{V}}{\|\widehat{V}\|_2}$
  $j = j + 1$
end while
return $X^* = X^{(j)}$

5.6.3 Implementation details

In this section, we briefly comment on the necessary changes to the algorithms introduced in Section 5.2 necessary to solve matrix-valued problems. First, as mentioned above, we have to replace the inner products and norms in every method by their Frobenius equivalents. Note that in **MATLAB**, the inner product $\langle X, Y \rangle$ can be computed efficiently in $\mathcal{O}(np)$ flops using

$$\langle X, Y \rangle = \sum_{i,j} x_{ij} y_{ij} = \texttt{sum(X.*Y,'all')}.$$

Secondly, every matrix-vector multiplication involving the Jacobian $J(x)$ has to be replaced by the evaluation of the Fréchet derivative $L(X, \Delta X)$, which can be performed efficiently using a finite difference approximation

$$L_F(X, \Delta X) \approx \frac{F(X + \varepsilon \Delta X) - F(X)}{\varepsilon}$$

if no explicit formula is available. If there is a need for a high-accuracy approximation, the complex step approximation

$$L_F(X, \Delta X) \approx \frac{\Im(F(X + \imath \varepsilon \Delta X))}{\varepsilon}, \quad \imath = \sqrt{-1},\ \Im(a + \imath b) = b,\ a, b \in \mathbb{R},$$

can be used if appropriate [147]. Finally, one may apply the automatic restart strategy from Section 5.5.2 when the maximum matrix norm is used in (5.34).

5.7 Numerical Experiments

In this section, we evaluate the performance of the *nlKrylov* methods developed in this work on a variety of nonlinear problems. We compare them against related approaches discussed in Section 5.3, including inexact Newton methods-particularly the Jacobian-free Newton-Krylov approach, nlOrthomin(k), and Anderson Acceleration (AA). Unless otherwise specified, we use GMRES as inner solver for all methods. In nlOrthomin, we use the Gauss-Newton-procedure as described in (5.14) to compute the optimal $y^{(j)} = \text{argmin}_y \|f(x^{(j)} + P^{(j)}y)\|$ at each iteration, where we set $V^{(j)} \equiv J_{s_j}(y)$ and restrict the Gauss-Newton iterations to a maximum of 20 steps. For Newton-Krylov methods, we follow the Eisenstat-Walker heuristic

[148] to adaptively control the linear solver tolerance at each nonlinear iteration, i.e.,

$$\eta_j = \left(\frac{\|r^{(j)}\|}{\|r^{(j-1)}\|} \right)^{\alpha}, \quad \eta_0 = \frac{1}{3}, \quad \alpha = \frac{1+\sqrt{5}}{2}.$$

In Anderson Acceleration, updates are damped using a fixed parameter $\beta \in \mathbb{R}$, such that $x^{(j+1)} = x^{(j)} + \beta f_j$, which corresponds to a root-finding problem for the scaled function $f_\beta(x) := \beta f(x)$.

Solver results are color-coded in all experiments as follows: nlGCR results are displayed with dark red circles ($\circ$), nlGMRESR with orange pluses ($+$), nlGCRO with grey asterisks ($*$) and nlLGMRES with blue squares ($\square$). Solid lines represent the full nonlinear methods, while dashed lines indicate their linear-nonlinear variants, e.g., nlGCR-A. For comparison, nlOrthomin is shown with turquoise crosses ($\times$), Newton-Krylov with red diamonds ($\diamond$) and Anderson Acceleration with green triangles ($\triangle$).

We compare solvers in terms of (outer) iterations as well as total function evaluations, where each evaluation of $f(x)$ or matrix-free evaluation of $J_f(x)y$ is counted as a single function evaluation. This measure reflects the extra evaluations needed in inner iterations of nested methods to enhance the local linear model, illustrating the potential gains of linear-update approaches. We note, however, that the total computational cost also depends on other operations such as inner products, which are not included in this count and may constitute a significant portion of runtime depending on the relative cost of function evaluations. All experiments are conducted using **MATLAB** 2024b on a notebook with AMD Ryzen 7 PRO 4750U CPU and integrated Radeon RX Vega 7 GPU.

5.7.1 The Lennard-Jones problem

We first consider the Lennard-Jones problem (also covered in [23]), a molecular optimization problem that seeks to determine atomic positions that minimize the total potential energy of a molecule

$$E(Y) = 4\sum_{i=1}^{N}\sum_{j=1}^{i-1}\left(\frac{1}{\|y_i - y_j\|_2^{12}} - \frac{1}{\|y_i - y_j\|_2^{6}}\right), \tag{5.40}$$

described by the Lennard-Jones-Potential, where the matrix $Y = [y_1, \ldots, y_N] \in \mathbb{R}^{3\times N}$ contains the 3-dimensional coordinate vectors $y_i \in \mathbb{R}^3$ for atoms $i = 1, \ldots, N$. A local minimizer of (5.40) can be computed by finding roots of

$$f(x) = \nabla E(\mathrm{vec}(Y)),$$

where $x \in \mathbb{R}^{3\cdot N}$ is the vectorization of Y. The derivative $J_f(x)\delta x$ is computed to high accuracy by the complex-step approximation

$$J_f(x)\Delta x = \frac{\Im(f(x + \imath h\Delta x))}{h} + \mathcal{O}(h^2), \qquad \text{with} \qquad h = 10^{-10}.$$

We follow the setup from [23], where an Argon cluster is simulated using a perturbed Face-Centered-Cubic (FCC) structure as an initial guess. Taking 3 cells in every direction, yields $3^3 = 27$ units cells in total. As FCC cells have 4 atoms, the calculation yields a problem of size $N = 4 \cdot 27 = 108$ and a root-finding problem of size of $n = 3 \cdot N = 324$. The necessary setup files and initial state are freely available from GitHub [149].

Experiment 1: In our first experiment, we compare the convergence properties of various methods using the Lennard-Jones problems as a test case. We use a truncation window of $k = 2$ for *nlKrylov* methods and $m = 5$ for all nested methods. Adaptive methods use $\theta = 10^{-3}$ as the switching tolerance. For Anderson acceleration, we use a truncation window

of $k_{AA} = 10$ and damping parameter $\beta = -3 \times 10^{-4}$. Newton-Krylov uses a maximum of 40 steps of GMRES for the inner solve. The iteration is stopped after 250 steps or when a relative tolerance of 10^{-14} is reached. The convergence results are displayed in Figure 5.3, where we restrict the view to at most 200 iterations and 600 function evaluations.

The left panel of Figure 5.3 illustrates that the nested nlKrylov methods-nlGMRES, nlGCRO, and nlLGMRES-converge more rapidly than nlGCR, owing to the improved local linear model provided by the inner solve. Among these, nlGMRESR and nlLGMRES exhibit comparable iteration counts, while nlGCRO requires few more iterations. The nlOrthomin method behaves similarly to nlGCR in terms of iteration count. However, when considering the number of function evaluations, the exact line search and full construction of $V^{(j)} = J_f(x^{(j)})P^{(j)}$ in nlOrthomin result in more than three times as many evaluations as nlGCR. Comparing the full nonlinear methods, both nlGMRESR and nlLGMRES require fewer function evaluations than nlGCR. Among the adaptive variants, all nlKrylov methods except nlGCRO benefit from switching to the linear-update version, with the effect being most pronounced for nlGCR-A, which surpasses nlGMRESR in efficiency upon convergence. Because the inner solve incurs additional function evaluations, the relative gain is smaller for the nested variants. In this example, nlGCR-A, nlGMRESR, and nlGMRESR-A perform comparably to Newton–GMRES in terms of total function evaluations. The relatively strong nonlinearity of the problem limits the benefit of reusing the outer basis in the inner iteration, as done in nlGCRO and nlLGMRES.

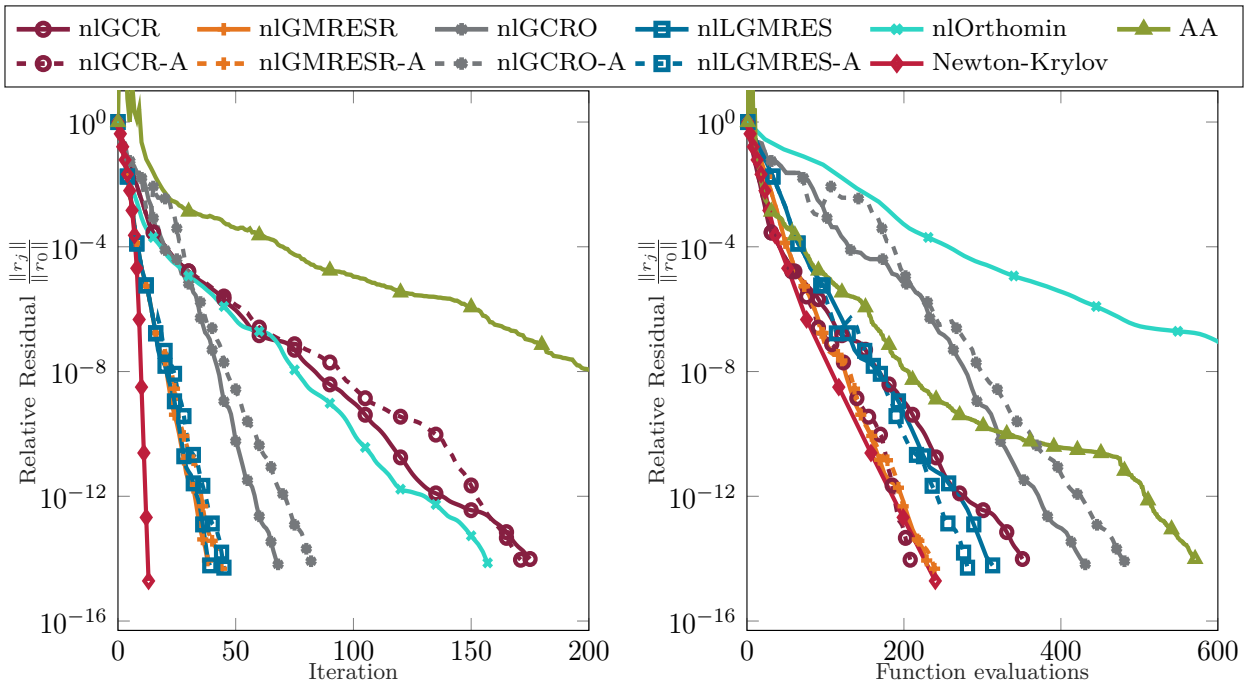


Figure 5.3: Convergence results for Lennard-Jones problem.

Experiment 2: In the second experiment, we use the Lennard-Jones problem to compare the theoretical and observed convergence bounds for nlGCR and nlGMRESR (Theorem 5.2) with those from [23]. Hence, we use the same setup as before but now we choose $m = 2$ and $k = 5$. In Figure 5.4, we display the uniform bound $c = \mu + \eta$ required in [23, Eq. (4.33) and (4.34)] as well as $c_j = \mu_j + \eta_j$ from (5.20) and the observed relative inexactness $\vartheta_j := \frac{\|t^{(j)}\|_2}{\|r^{(j)}\|_2}$. Recall that Theorem 5.2 guarantees the convergence of *nlKrylov* methods provided that $\vartheta_j \leq c_j < 1$. We can see in Figure 5.4, that the uniform bound $c > 1$ does not satisfy the assumption and, as such, cannot provide a theoretical foundation for convergence. However, the same figure shows that we always have $c_j < 1$ and $\vartheta_j \leq c_j$ in the experiment and for some $j_0 \in \mathbb{N}$, $\vartheta_j = c_j$ holds for all $j \geq j_0$. For nlGCR, $j_0 \approx 40$ while for nlGMRESR, $j_0 \approx 25$. Thus, we can see that the bound presented in (5.20) is sufficient for convergence of the methods as it is always an upper bound for ϑ_j and, at convergence, this upper bound is sharp.

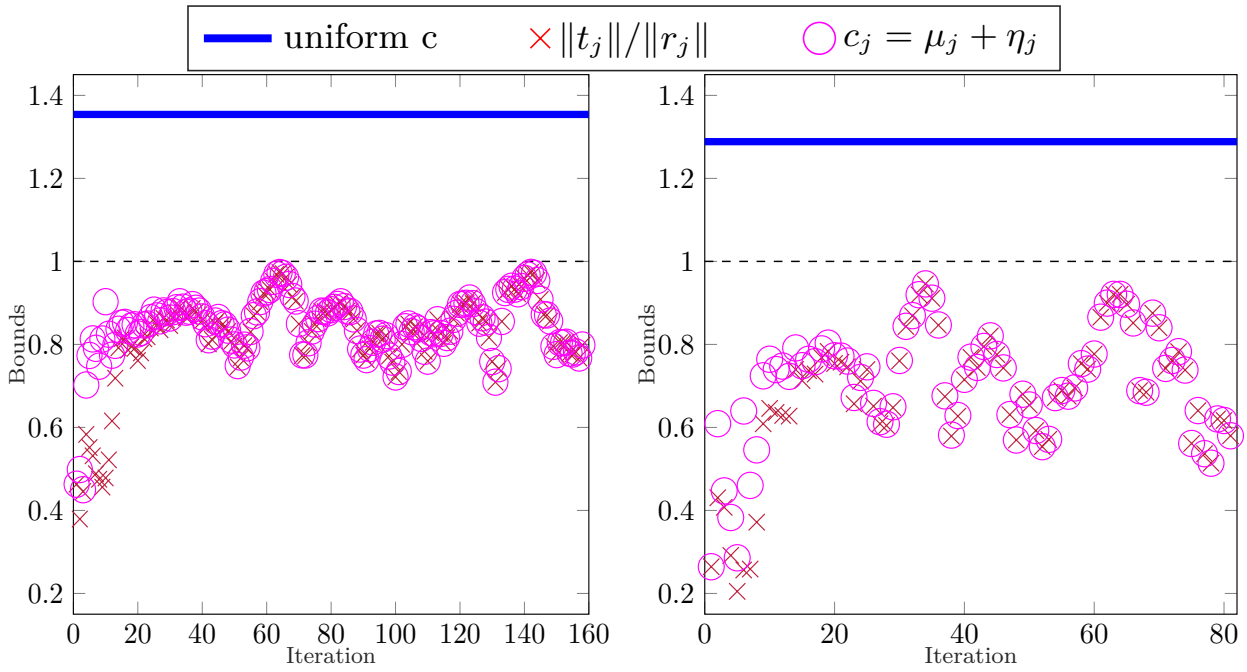


Figure 5.4: Theoretical and observed convergence bounds for nlGCR (left) and nlGMRESR (right) applied to the Lennard-Jones problem.

5.7.2 The Chandrasekhar H-equation

For our second example, we consider the integral equation

$$F(H)(\mu) = H(\mu) - \left(1 - \frac{\omega}{2}\int_0^1 \frac{\mu}{\mu+\nu} H(\nu)d\nu\right)^{-1} = 0, \tag{5.41}$$

where a continuously differentiable function $H : [0,1] \to [0,1]$ satisfying (5.41) is to be determined. Equation (5.41), known as the Chandrasekhar H-equation [150], originates from radiative transfer problems. This equation has been examined in several studies focusing on the convergence properties of Newton's method for singular problems [114, 129]. In particular, [128] shows that (5.41) has a singular Jacobian when $\omega = 1$, with a one dimensional null space, i.e., $\dim(\mathcal{N}) = 1$. As described in [114], using n equally spaced grid points $\mu_i = (i - 1/2)/n$, $i = 1, \ldots, n$, and setting $h_i = H(\mu_i)$, equation (5.41) can be discretized

via the midpoint rule as

$$[f(h)]_i = h_i - \left(1 - \frac{\omega}{2n}\sum_{j=1}^{n}\frac{ih_j}{i+j-1}\right)^{-1}, \quad i = 1, \ldots, n,$$

where the summation term can be computed efficiently using the Fast Fourier Transform (FFT). The resulting formulation is an n-dimensional root-finding problem $f(h) = 0$. In our experiments, we use $n = 10^3$ grid points and consider both the non-singular case ($\omega = 0.99$) and the singular case ($\omega = 1$).

Experiment 1: In the first experiment, we consider the non-singular case $\omega = 0.99$. We use a truncation window of $k = 10$ for the *nlKrylov* methods, nlOrthomin and Anderson Acceleration. For nested variants, the inner linear systems are solved using $m = 4$ steps of GMRES. The adaptive methods use $\theta = 10^{-2}$ and Anderson Acceleration is executed with $\beta = -0.1$. The Newton-Krylov method is limited to a maximum of 100 GMRES iterations per Newton step. All iterations are terminated after 30 steps or once the relative residual falls below 10^{-12}. As shown in Section 5.7.2, the adaptive variants of nlGMRESR and nlLGMRES did not switch to their linear counterparts during the iteration, resulting in overlapping convergence curves. In contrast, both nlGCR-A and nlGCRO-A activated the linear updates and demonstrated improved convergence compared to their fully nonlinear versions. Notably, nlGCRO-A was able to overcome the stagnation observed in its nonlinear counterpart. In terms of iteration counts, nlGMRESR and nlGMRES achieved convergence behavior comparable to that of Newton-GMRES. When considering the total number of function evaluations, nlGMRESR and nlGCR-A remained close to Newton-GMRES in efficiency, whereas nlLGMRES required more than twice as many evaluations. Although nlOrthomin reached the desired tolerance in one fewer iteration than Newton-GMRES, it did so at the cost of significantly more function evaluations.

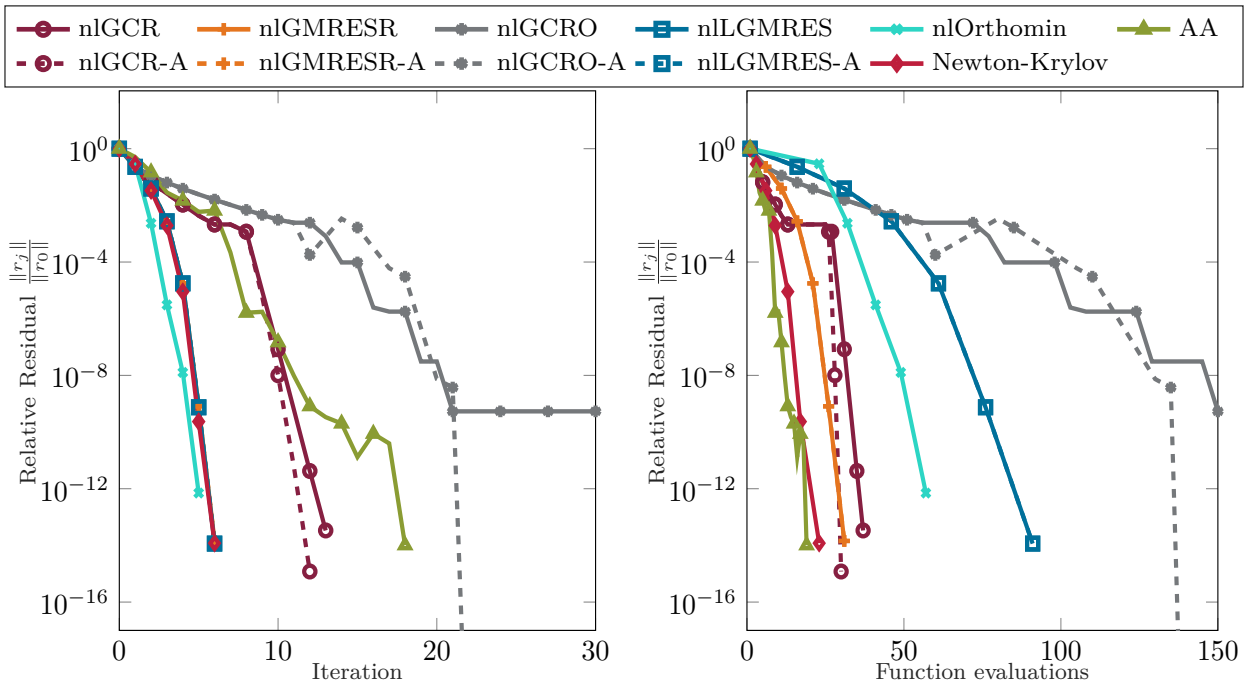


Figure 5.5: Convergence result for H-equation with non-singular Jacobian ($\omega = 0.99$).

Experiment 2: Our second experiment considers the singular case $\omega = 1$. We use the same setup as in Experiment 1, except that the maximum number of (outer) iterations is increased to 100. As shown in Figure 5.6 all methods require substantially more iterations compared to the non-singular case. Once again, the adaptive variants of nlGMRESR and nlLGMRES did not switch to their linear counterparts. We observe that nlGCR and nlGCRO failed to converge for this problem, whereas their adaptive variants significantly improved accuracy-nlGCR-A increased from approximately 10^{-8} and nlGCRO-A from 10^{-2} to around 5×10^{-12} before stagnating. All other methods reached the desired accuracy. Among them, nlGMRESR, nlLGMRES and Newton-GMRES required the same number of iterations, with nlGMRESR and Newton-Krylov showing comparable numbers of function evaluations. In contrast, nlOrthomin achieved convergence in fewer interations but incurred more than twice the number of function evaluations. These results indicate that the adaptive variants can enhance convergence relative to the fully nonlinear methods and, in some cases, overcome the

stagnation observed in the original nonlinear iterations. Moreover, most methods successfully solved the singular problem with the desired accuracy.

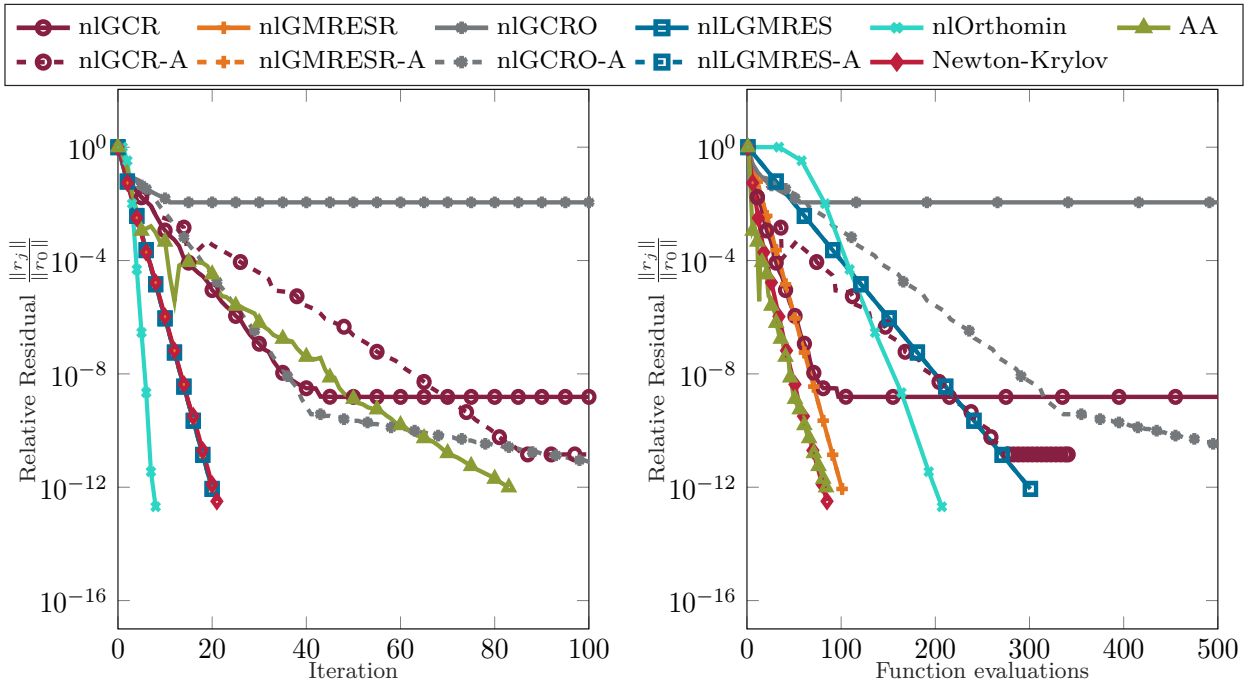


Figure 5.6: Convergence result for H-equation with singular Jacobian ($\omega = 1$).

5.7.3 The symmetric Bratu problem

As a third benchmark problem, we study the well-known symmetric Bratu problem [151], arising from the discretization of the nonlinear PDE

$$\begin{aligned} \Delta u + \lambda e^{u} &= 0, \quad (y,z) \in \Omega \\ u &= 0, \quad (y,z) \in \partial\Omega, \end{aligned}$$

with $\Omega = (0,1) \times (0,1)$ and $\lambda = 0.5$. Using an equally spaced grid $y_i = z_i = i \cdot h$, $h = \frac{1}{N+2}$, $i = 1, \ldots, N$ in both spatial directions, the problem reduces to the following nonlinear

problem

$$f(x) = Lx - h^2\lambda \exp(x) = 0, \tag{5.42}$$

where $L \in \mathbb{R}^{n\times n}$ is a 2D-Laplacian ($L = L_N \otimes I_N + I_N \otimes L_N$, $L_N = \text{tridiag}(-1, 2, -1) \in \mathbb{R}^{N\times N}$), $n = N^2$ and $\exp(\cdot)$ is applied element-wise. From (5.42), it follows immediately that

$$J_f(x)\Delta x = L\Delta x - h^2\lambda \exp(x) \odot \Delta x,$$

where $x \odot \Delta x$ is the Hadamard product. In our experiments, we follow the setup of [23] with $N = 100$, giving a problem size $n = 10,000$, and initialize with the all-ones vector $x^{(0)} = \mathbf{1}_n$. *Experiment 1:* As before, we analyze the convergence of various methods on the Bratu problem. All truncated methods use a truncation window of $k = 10$, nested *nlKrylov* methods employ 20 GMRES steps, and adaptive methods switch with a threshold $\theta = 10^{-3}$. Anderson Acceleration is damped with $\beta = 0.1$, and MINRES is used as the inner solver in Newton's method due to the symmetry of the Jacobian. Iterations are stopped once a relative tolerance of 10^{-15} is reached. The results are presented in Figure 5.7. Here, we observe the impact of refined local linear models for moderately nonlinear problems. With a relatively large truncation window of $k = 10$ and 20 inner GMRES steps ($m = 20$), all nested methods converged within 30 iterations, whereas nlGCR required approximately 500 iterations. In particular, nlGCRO benefits from the slowly varying Jacobian, needing only about one-third of the function evaluations of nlGCR. Both the nested methods and their adaptive variants, as well as nlGCR-A, outperform Newton-MINRES in terms of function evaluations. Remarkably, nlOrthomin achieves a similar number of iterations as nlGCR but requires more than ten times as many function evaluations.

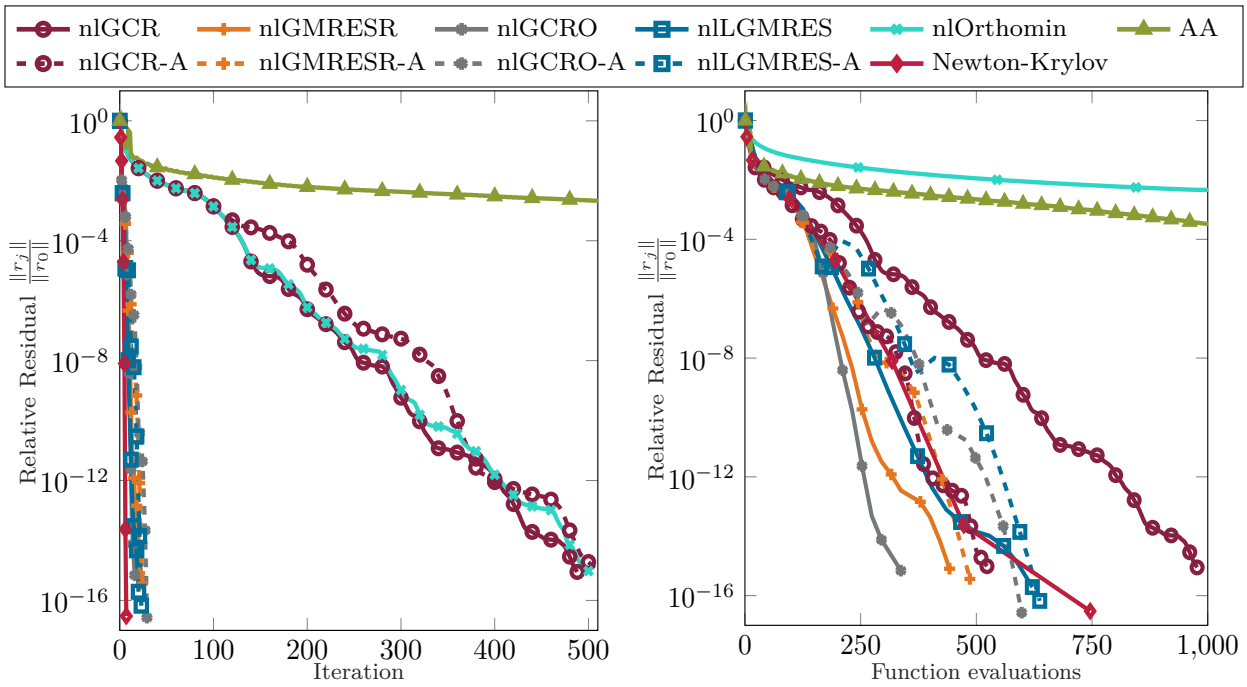


Figure 5.7: Convergence results for Bratu problem

Experiment 2: In the second experiment, we analyze the theoretical bounds from Theorem 5.2. Using the same setup as before, we compare the uniform bound c from [23], the sequence $c_j = \mu_j + \eta_j$ from (5.20), and the actual ratio $\vartheta_j = \frac{\|t^{(j)}\|}{\|r^{(j)}\|}$. As shown in Section 5.7.3, both the uniform bound c as well as c_j are smaller than one, thereby guaranteeing convergence of all algorithms. We also observe that for nlGCR and nlGCRO, c provides a sharp upper bound for c_j and ϑ_j, whereas for nlGMRESR and nlLGMRES, c_j is consistently slightly smaller than c. On the other hand, for most iterations j, we observe that $c_j \ll c$, indicating that the uniform bound c significantly overestimates the actual error, whereas the c_j's remain highly accurate. Furthermore, the smaller values of c_j and ϑ_j for the nested methods compared to nlGCR demonstrate that convergence, measured by the number of outer GCR steps, is substantially faster for these methods, consistent with the results shown in Figure 5.7. Once again, c_j provides an asymptotically sharp bound for ϑ_j in all cases.

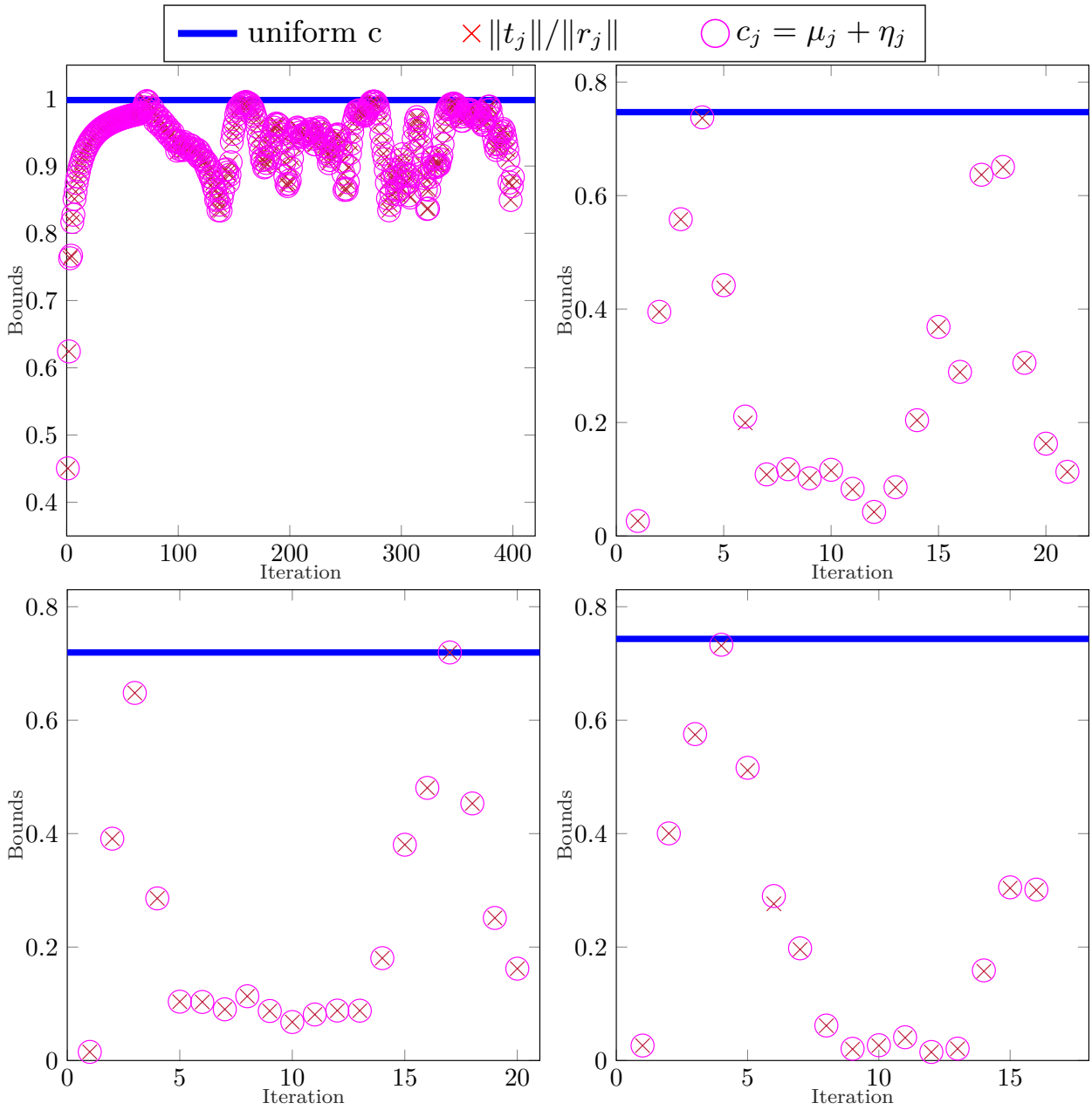


Figure 5.8: Theoretical and observed convergence bounds for the Bratu problem, shown for GCR, GMRESR, GCRO, and LGMRES (clockwise from top left).

Remark 5.9. Note that the 2D-Bratu problem (5.42) can be reformulated as a matrix-valued problem using

$$F(X) = L_N X + X L_N - h^2 \lambda \exp(X),$$

where $X \in \mathbb{R}^{N \times N}$ and the exponential is applied element-wise. The Fréchet derivative is given by

$$L_F(X, \Delta X) = L_N \Delta X + \Delta X L_N - h^2 \lambda \left(\exp(X) \odot \Delta X \right).$$

Using this formulation, we can apply the techniques from Section 5.6.2 with the same parameters as before. The Sylvester/Lyapunov part $S(X) = L_N X + X L_N$ allows for a specialized subroutine that efficiently enhances the local linear model, especially for small values of λ.

With this note, we now turn our attention to a matrix-valued example to conclude our numerical experiments.

5.7.4 A nonlinear algebraic Riccati equation

In our fourth example, we consider a non-symmetric algebraic Riccati equation (NARE) of the form

$$\mathcal{R}(X) = FG^T + AX + XB - XPQ^T X = 0, \tag{5.43}$$

where $A \in \mathbb{R}^{n \times n}$, $B \in \mathbb{R}^{p \times p}$, $F \in \mathbb{R}^{n \times r}$, $G \in \mathbb{R}^{p \times r}$, $P \in \mathbb{R}^{p \times s}$, $Q \in \mathbb{R}^{n \times s}$ and $X \in \mathbb{R}^{n \times p}$ is the solution of interest. In our application, $p \ll n$ and all matrices are sparse. In particular, we

adopt a setup similar to [152, Example 1], where

$$A = \begin{bmatrix} 3 & -1 & & \\ & \ddots & \ddots & \\ & & 3 & -1 \\ -1 & & & 1.9 \end{bmatrix}, \qquad B = \begin{bmatrix} 2 & -1 & & \\ & 3 & \ddots & \\ & & \ddots & -1 \\ -1 & & & 3 \end{bmatrix},$$

$$G = I_{p,r}, \quad F = \begin{bmatrix} F_1 \\ 0 \end{bmatrix}, \qquad F_1 = \begin{bmatrix} -1 & -1 & & \\ & \ddots & \ddots & \\ & & -1 & -1 \\ & & & -0.9 \end{bmatrix} \in \mathbb{R}^{r \times r},$$

$$Q = I_{n,s}, \quad P = \begin{bmatrix} P_1 \\ 0 \end{bmatrix}, \qquad P_1 = \begin{bmatrix} 1 & & & \\ 1 & 1 & & \\ & \ddots & \ddots & \\ & & 1 & 1 \end{bmatrix} \in \mathbb{R}^{s \times s},$$

with $n = 30,000$, $p = 200$, $r = 3$ and $s = 5$. Equation (5.43) yields the Fréchet derivative

$$L_{\mathcal{R}}(X, \Delta X) = A\Delta X + \Delta X B - \left(\Delta X P Q^T X + X P Q^T \Delta X\right).$$

A highly efficient low-rank Newton-ADI algorithm for NAREs is proposed in [152]. For this illustrative example, we instead compare our methods to standard JFNK applied to (5.43), which requires reducing p by two orders of magnitude relative to [152] for memory reasons. *Experiment 1:* In this experiment, we use a truncation window of $k = 6$ for the nlKrylov methods and nlOrthomin. Nested methods use $m = 8$, adaptive methods switch with $\theta = 10^{-4}$, and Anderson Acceleration uses a truncation window $k_{AA} = 12$ with damping parameter $\beta = 0.1$. Newton's method performs up to 50 steps of global GMRES for the

inner solve. Iterations are stopped after 100 steps or when the relative residual falls below $\tau = 10^{-15}$. Results with the all-zeros initial guess $X^{(0)} = 0_{n,p}$ are shown in Figure 5.9. The results demonstrate that *nlKrylov* methods can successfully solve matrix-valued root-finding problems. As in previous experiments, the adaptive versions of nlGMRESR, nlGCRO, and nlLGMRES did not switch to the linear update, whereas nlGCR-A switched after about 10 iterations. Both nlGMRESR and nlLGMRES converged in the same number of iterations as Newton-GMRES; nlGMRESR required a comparable number of function evaluations, while nlLGMRES needed significantly more. Consistent with other experiments, nlOrthomin shows similar iteration counts to nlGCR, but its function evaluations grow too quickly to remain competitive.

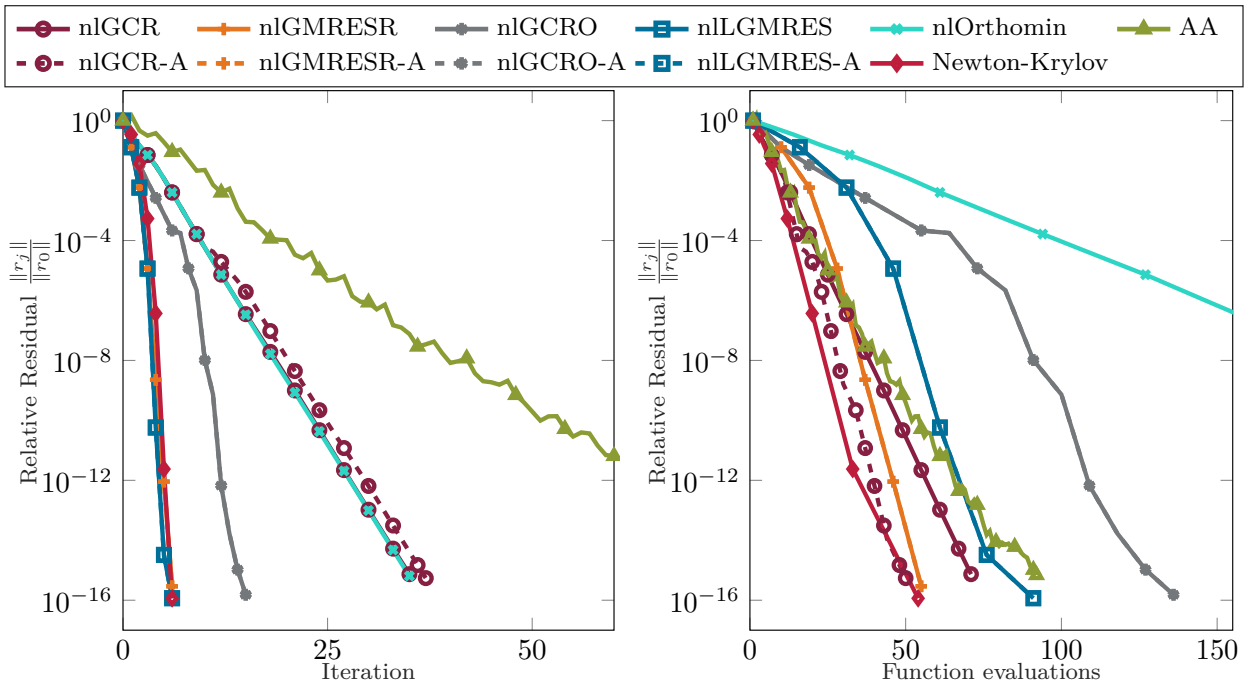


Figure 5.9: Convergence results for NARE problem

5.7.5 A nonlinear eigenvalue problem

In this vector valued example, we consider a nonlinear eigenvalue problem (NEP) originating from the discretization of the partial delay-differential equation

$$\begin{aligned} \Delta u(x,t) + a(x)u(x,t) + b(x)u(x,t-2) - u_t(x,t) = 0, \qquad & x \in \Omega, \\ u(x,t) = 0, \qquad & x \in \partial\Omega, \end{aligned}$$

where $\Omega = [0,\pi]^2$, $t \geq 0$ and

$$a(x) = \sin^2(x_1)\sin^2(x_2), \quad b(x) = \sin(x_1 + x_2) + 1.31.$$

The example is called the `pdde_symmetric` problem in the NLEVP database [153, 154]. Upon discretization, we obtain the NEP

$$T(\lambda)v = \lambda v, \tag{5.44}$$

with a nonlinear function $T(\lambda) : \mathbb{R} \to \mathbb{R}^{n\times n},\ \lambda \mapsto (M + A) + e^{-2\lambda}B$, where $n = (N-1)^2$, N is the number of discretization points, and M, A and B are sparse symmetric $n \times n$ matrices that discretize $\Delta(\cdot)$, $a(\cdot)$ and $b(\cdot)$, respectively. The eigenvalue λ^* nearest to zero is of interest in this example. Using an arbitrary normalization vector $c \in \mathbb{R}^n$, the eigenvalue problem (5.44) can be rewritten in terms of a root finding problem [155], i.e.,

$$0 = f(x) = \begin{bmatrix} T(\lambda)v - \lambda v \\ c^T v - 1 \end{bmatrix}, \quad x = \begin{bmatrix} v \\ \lambda \end{bmatrix} \in \mathbb{R}^{n+1}. \tag{5.45}$$

Again, it is straightforward to see that

$$J_f(x)\Delta x = \begin{bmatrix} T(\lambda) - \lambda I & (T'(\lambda) - I)v \\ c^T & 0 \end{bmatrix} \begin{bmatrix} \Delta v \\ \Delta\lambda \end{bmatrix} = \begin{bmatrix} (T(\lambda) - \lambda I)\Delta v + \Delta\lambda(T'(\lambda) - I)v \\ c^T \Delta v \end{bmatrix},$$

where $T'(\lambda) = -2e^{-2\lambda}B$.

Experiment 1: In this experiment, we use $N = 64$ grid points in both directions, which results in a problem of size $n + 1 = (N - 1)^2 + 1 = 3,970$. We set $c = \frac{1}{\sqrt{n}}1_n$ and initial guess $x^{(0)} = [(v^{(0)})^T, \lambda^{(0)}]^T$, where $(v^{(0)}, \lambda^{(0)}$ is the smallest magnitude eigenpair of $T(0) = M + A + B$. We use a truncation window of $k = 20$ for nlKrylov methods and $k_{AA} = 30$ for Anderson Acceleration. Nested nlKrylov methods use $m = 30$, their adaptive versions use $\theta = 10^{-3}$. Anderson Acceleration used damping parameter $\beta = 10^{-3}$ and the Newton-Krylov method allows for a maximum of 200 GMRES steps. The iterations are stopped once the tolerance of 10^{-12} is reached. In Figure 5.10, all nested Krylov methods converge within 30 iterations. Among them, nlLGMRES and nlGCRO require the same number of iterations as Newton–Krylov, while nlGMRESR requires roughly ten additional iterations. The methods nlGCR, nlGCR-A, and nlOrthomin did not converge in this experiment. The adaptive variants of nlGMRESR, nlGCRO, and nlLGMRES did not switch to linear mode. Considering function evaluations, all nested methods outperform Newton–Krylov: nlGCRO converges after approximately 500 function evaluations, nlGMRESR and nlLGMRES after about 800, and Newton–Krylov requires roughly 1200. The eigenvalue $\lambda^* \approx -0.2467$ of interest is found by all converged methods.

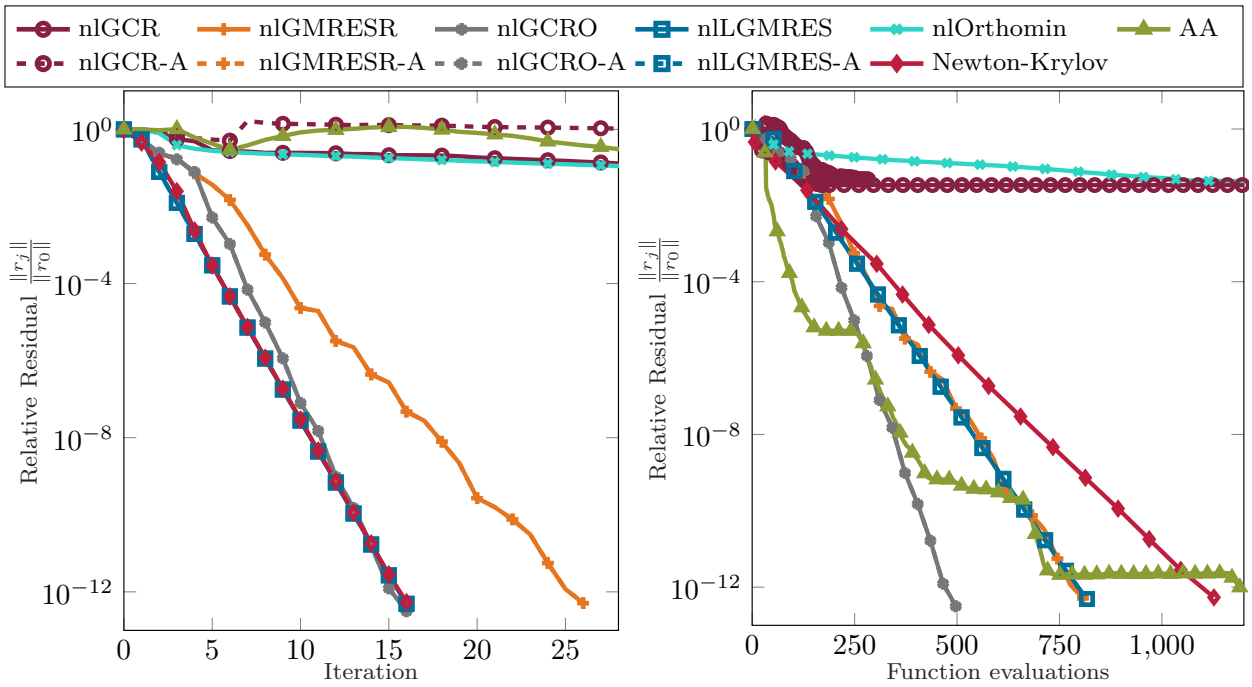


Figure 5.10: Convergence results for PDDE-NEP

5.7.6 A nonlinear eigenvector problem (NEPv)

For our last experiment, we consider a nonlinear eigenvector problem of the form

$$H(V)V = V\Lambda, \quad V^TV = I_p, \quad \Lambda = \Lambda^T, \tag{5.46}$$

where $H : \mathbb{R}^{n\times p} \to \mathbb{R}^{n\times n}$ is a symmetric matrix function, i.e., $H(V) = H(V)^T$, and $p \ll n$. Problems of this type arise frequently in quantum physics, most notably discretized Kohn-Sham and Gross-Pitaevskii equations [156, 157], in data science applications such as Robust-Rayleigh-Quotient or Trace-Ratio optimization [156, 158] as well as the modeling of dissipative Hamiltonian DAEs [159]. Recently, the use of matrix-valued Newton methods to

solve (5.46) via the root finding problem

$$F(X) = \begin{bmatrix} H(V)V - V\Lambda \\ I_p - V^T V \end{bmatrix}, \quad X = \begin{bmatrix} V \\ \Lambda \end{bmatrix} \in \mathbb{R}^{(n+p)\times p}, \tag{5.47}$$

has been studied in [83]. We consider the discrete 3D-Kohn-Sham model

$$H(V) = L + \mathrm{Diag}(L^{-1}\rho(V) - \gamma\rho(V)^{1/3}), \quad \gamma \geq 0,$$

where $L \in \mathbb{R}^{n\times n}$ is a 3D-Laplacian on a cube discretized using N equally spaced grid points in every direction, leading to $n = N^3$, and $\rho(V) = \mathrm{diag}(VV^T)$ is the charge density of electrons. Note that the Fréchet derivative of (5.47) is singular close to the solution due to the orthogonal invariance of $H(V)$, as was pointed out in [83]. However, if one chooses a different normalization criterion similar to (5.45), i.e., using a constant matrix $C \in \mathbb{R}^{n\times p}$, we can rewrite (5.47) to

$$\widetilde{F}(X) \begin{bmatrix} H(V)V - V\Lambda \\ I_p - C^T V \end{bmatrix}, \quad X = \begin{bmatrix} V \\ \Lambda \end{bmatrix} \in \mathbb{R}^{(n+p)\times p}, \tag{5.48}$$

to obtain a non-singular problem.

Experiment 1: We first want to examine the singular case, i.e., the root finding problem (5.47). In our experiment, we use $N = 16$ and $p = 2$, leading to $X \in \mathbb{R}^{4098\times 2}$. The iteration is started with an initial value generated by two steps of SCF initialized using the two smallest eigenpairs of L. We use a truncation window of $k = 10$ for the nlKrylov methods, the inner solve in nested nlKrylov methods is carried out by $m = 20$ steps of global GMRES, Newton's method uses at most 400 steps of global GMRES. The iteration is terminated when the relative residual is below $\tau = (N+p)10^{-12} \approx 1.8 \cdot 10^{-11}$ or after 50

iterations. The adaptive methods use an angle of $\theta = 10^{-3}$. For NEPv, acceleration schemes such as Anderson, Pulay or secant acceleration have proven to improve and stabilize SCF convergence when used as a mixing scheme [17, 160, 161]. As a consequence, here, we are using Anderson Acceleration as an accelerator rather than a solver, with SCF as the underlying fixed-point scheme. The window size is $k = 30$ and the update is damped using $\beta = -0.1$. The convergence results are displayed in Figure 5.11.

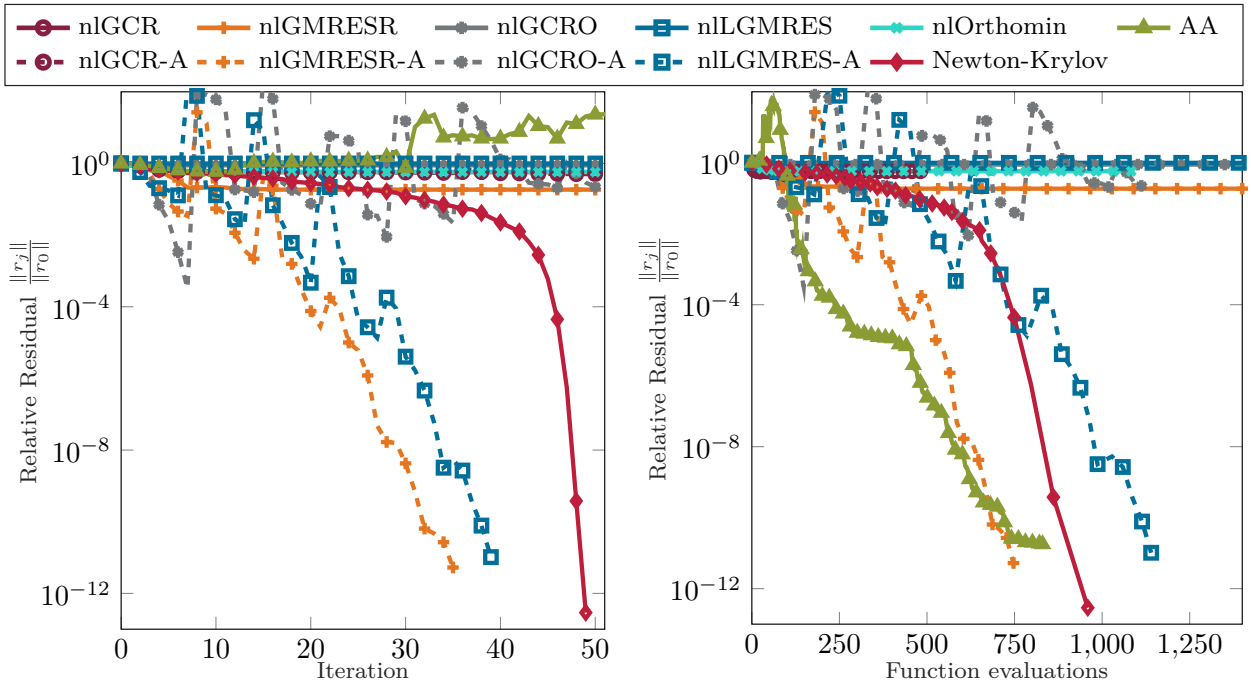


Figure 5.11: Convergence results for singular Kohn-Sham NEPv

Here we can see that only four methods managed to converge to the desired accuracy while all the others stagnated around $10^{0} - 10^{-1}$. The four convergent methods are nlGMRESR-A, nlLGMRES-A, Newton-Krylov and Anderson Acceleration. Among these methods, the adaptive nlKrylov methods required the fewest iterations to convergence: 36 for nlGMRESR-A and 39 for nlLGMRES-A, while Newton's method required 49 steps. Although the convergence of Anderson Acceleration is not visible in the iteration plot due to the cutoff at 50 iterations, it can be seen in the function evaluation plot. Looking at the function evalu-

ations, nlGMRESR-A performed the best, with Anderson Acceleration being close, while Newton-Krylov and nlLGMRES required over 250 function evaluations more.

These results underline the observation from the H-equation example that adaptive versions of nlKrylov methods exhibit improved convergence compared to their full nonlinear counterparts and can achieve convergence in cases where the standard nonlinear methods stagnate.

Experiment 2: In the second experiment, we use the same setup as before to solve the nonsingular problem (5.48), with the only change being an increased angle parameter $\theta = 5 \times 10^{-3}$. Here, we adopt the somewhat academic choice of normalization matrix $C = V^*$, where (V^*, Λ^*) is an approximate solution to (5.46) obtained via SCF. The convergence results are displayed in Figure 5.12. Compared to the singular case, more methods achieve the desired accuracy, including the full nonlinear versions of nlGMRESR and nlLGMRES, as well as nlGCRO-A.

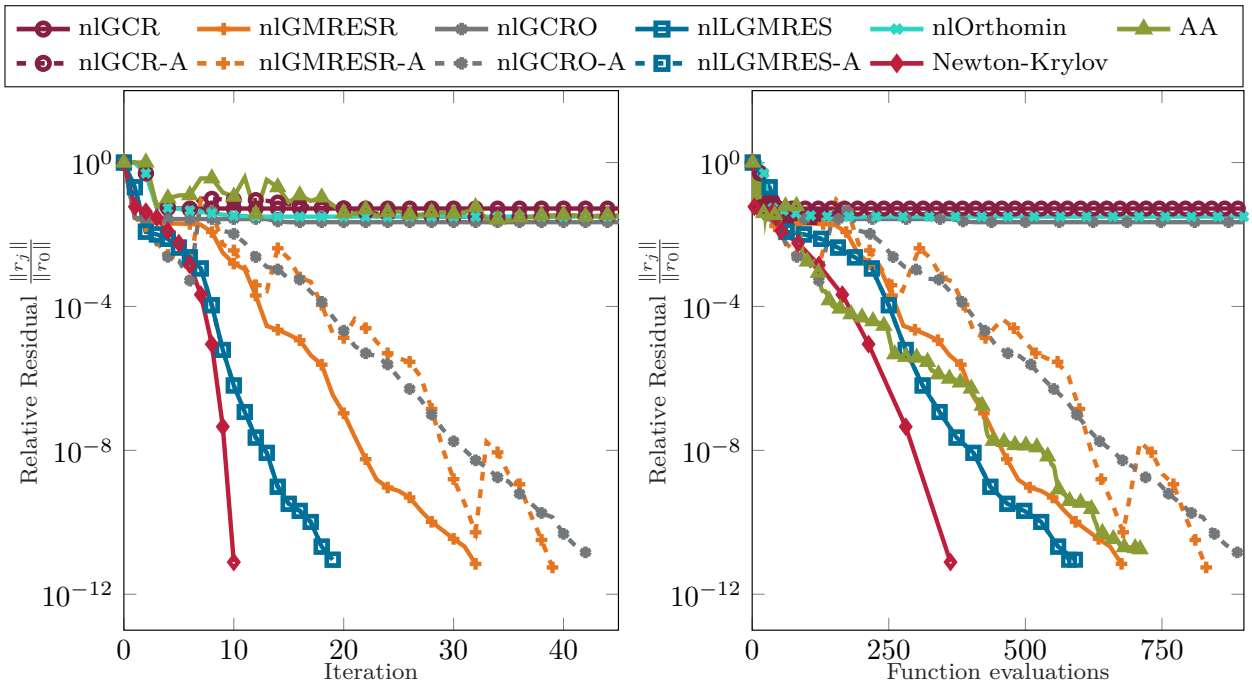


Figure 5.12: Convergence results for non-singular Kohn-Sham NEPv

In this setup, Newton-Krylov converges fastest, reaching the solution in just 10 steps with nearly quadratic residual reduction. nlLGMRES and nlLGMRES-A follow, converging after approximately 20 iterations, with the linear update used only in the final step. The remaining methods require over 30 iterations, with nlGCRO-A achieving the target accuracy after 42 steps. A similar trend is observed in the function evaluations, with Anderson Acceleration requiring roughly the same number of function evaluations as nlGMRESR.

5.8 Conclusions

In this chapter, we have developed a unified framework for nonlinear Krylov methods derived from nested GCR-type linear solvers. Our approach systematically generalizes methods like GMRESR, GCRO, and LGMRES to the nonlinear problems, yielding algorithms such as nlGMRESR, nlGCRO, and nlLGMRES. These methods preserve the essential structural properties of their linear counterparts while admitting an interpretation as inexact Newton schemes. We have established convergence results under relaxed assumptions and clarified their relationships with other nonlinear solvers, including nlOrthomin. Moreover, the framework extends naturally to nonlinear matrix equations, which are increasingly important in data-driven and high-dimensional applications.

The main contributions of this chapter can be summarized as follows:

- We introduced a flexible and systematic framework that allows the inclusion of "arbitrary" subroutines, providing a unified perspective across a broad family of nonlinear Krylov methods.
- We developed nonlinear extensions of several well-known linear Krylov methods, including GMRESR, GCRO, and LGMRES.

- We demonstrated the applicability of the proposed framework to matrix-valued problems, enabling extensions to operator and manifold-constrained settings.
- We established explicit connections between the proposed nonlinear Krylov methods, nlOrthomin(k), and the Projected Anderson Acceleration (PAA) methods.
- We provided convergence analysis under relaxed assumptions, offering a different perspective based on subspace pairs $(P^{(j)}, V^{(j)})$ and the inexact Newton interpretation, rather than relying on strong smoothness assumptions on f.
- We analyzed both nonsingular and singular cases, providing theoretical insights and experimental validation even in the presence of degeneracy (although under strong assumptions that may not always hold in practice).

Overall, the results presented in this chapter establish a general and flexible foundation for further theoretical and computational exploration of nonlinear Krylov methods. The developed framework unifies multiple algorithmic families under a single theoretical viewpoint, offering both conceptual clarity and practical effectiveness. Future directions include adapting the framework to stochastic or derivative-free settings, incorporating adaptive memory truncation for enhanced scalability, and exploiting the connection to Newton–Krylov methods by selecting m adaptively to further improve performance.

Chapter 6

Conclusions and Future Work

In this dissertation we have presented a comprehensive study of acceleration techniques for nonlinear solvers, with a particular emphasis on Anderson-type and Krylov-based methods. Building upon the fixed-point and extrapolation frameworks developed in the early chapters, we have proposed several unified perspectives that reveal deep structural connections among existing acceleration strategies, thereby setting the stage for new algorithmic developments.

6.1 Summary of Contributions

This thesis advances the theoretical and practical understanding of acceleration techniques for nonlinear fixed-point and nonlinear equation solvers. Through the development of unified frameworks, rigorous convergence results, and extensive computational validation, the work addresses fundamental questions regarding the structure, stability, and performance of modern nonlinear accelerators. The main contributions are summarized below.

1. **Unified framework connecting Anderson Acceleration and CROP.**

    - We developed a unified Anderson-type formulation for the CROP algorithm and proved its equivalence to Anderson Acceleration in exact arithmetic.
    - The analysis situates CROP within the landscape of Krylov and multisecant methods, clarifying structural parallels with GMRES, Orthomin, and quasi–

Newton updates.

- New hybrid variants CROP-Anderson and the real residual extensions rCROP and rCROP–Anderson were introduced to improve robustness for nonlinear problems.
- Convergence results for both linear and nonlinear cases were established, and extensive experiments demonstrated that CROP-based schemes can match or surpass Anderson Acceleration in several problem classes.

2. **Analysis of Anderson Acceleration variants under deterministic and stochastic perturbations.**

   - We systematically examined the sensitivity of Anderson Acceleration and its variants to deterministic and stochastic perturbations.
   - Theoretical results showed that deterministic errors introduce bounded linear perturbations in the residuals, limiting attainable accuracy to the magnitude of the perturbation.
   - Stochastic perturbations, modeled via Monte Carlo noise, were shown to cause oscillations in convergence proportional to the sampling variance.
   - Numerical experiments confirmed that restarted and adaptive-depth Anderson methods mitigate instability by controlling memory and decorrelating noise effects.
   - These findings provide practical guidelines for designing stable acceleration methods in inexact or stochastic computational environments.

3. **Generalized framework for nonlinear Krylov subspace methods.**

   - We formulated a unified framework for nonlinear Krylov solvers derived from nested GCR-type algorithms, generalizing GMRESR, GCRO, and LGMRES to nonlinear settings.

- The resulting methods-nlGMRESR, nlGCRO, and nlLGMRES-preserve the structural properties of their linear counterparts while admitting an inexact Newton interpretation.
- Convergence was established under relaxed conditions, and algorithmic relationships with nlOrthomin and other nonlinear solvers were clarified.
- The framework naturally extends to matrix-valued equations $F(X) = 0$, enabling applications in data-driven and high-dimensional problems.
- Directions for further development include stochastic, derivative-free, and adaptive-memory extensions.

Together, these contributions provide a coherent theoretical foundation and practical toolkit for the design, analysis, and implementation of accelerated nonlinear solvers, with potential impact across applied mathematics, computational physics, and scientific computing.

6.2 Future Work

Although this thesis has explored several theoretical and algorithmic aspects of accelerated fixed-point and nonlinear Krylov methods, further investigations are still needed. Future research directions include developing a more refined understanding of how truncation depth, damping, and scaling influence the stability and convergence of acceleration methods in highly nonlinear systems. It will be important to study the behavior of CROP-based methods in non-smooth or non-differentiable settings, where Jacobians may be unavailable or discontinuous, and to develop adaptive parameter selection strategies that automatically balance acceleration and stability in the presence of noise or stochastic error. Further work will focus on designing adaptive or truncated nonlinear Krylov methods that maintain effi-

ciency for large-scale problems with limited memory, and on integrating the nonlinear Krylov framework with CROP- or Anderson-type updates to form hybrid multisecant-Krylov accelerators. Extending the current framework to handle stochastic, matrix-valued, or manifold-constrained problems such as orthogonality-constrained optimization, tensor decompositions, and low-rank matrix equations, also represents a promising direction. Additional avenues include exploring preconditioned and derivative-free nonlinear Krylov variants for data-driven or machine learning applications, analyzing the behavior of the discussed methods in floating-point and mixed=precision arithmetic to ensure numerical stability, and developing parallel implementations to investigate their performance and scalability in high-performance computing environments.

Future research directions include extending the proposed frameworks to:

- Developing a more refined understanding of how truncation depth, damping, and scaling influence stability and convergence of acceleration methods in highly nonlinear systems.

- Investigating the behavior of CROP-based methods in non-smooth or non-differentiable settings, where Jacobians may be unavailable or discontinuous.

- Developing adaptive parameter selection strategies that automatically balance acceleration and stability in the presence of noise or stochastic error.

- Developing adaptive or truncated nonlinear Krylov methods that maintain efficiency for large-scale problems with limited memory.

- Integrating the nonlinear Krylov framework with CROP- or AA-type updates to form hybrid multisecant-Krylov accelerators.

- Extending the current framework to handle stochastic, matrix-valued, or manifold-constrained problems, such as orthogonality-constrained optimization, tensor decompositions, and low-rank matrix equations.
- Exploring preconditioned and derivative-free nonlinear Krylov variants for data-driven or machine learning applications.
- Analyzing the behavior of discussed methods in floating-point arithmetic to ensure numerical stability.
- Exploring mixed-precision implementations to balance computational efficiency and numerical stability, particularly for large-scale or hardware-accelerated computations.
- Developing parallel versions of these methods and investigating their performance to ensure scalability in high-performance computing environments.

Bibliography

[1] D. G. Anderson, "Iterative procedures for nonlinear integral equations," *J. ACM*, vol. 12, no. 4, pp. 547–560, 1965.

[2] D. G. Anderson, "Comments on "Anderson acceleration, mixing and extrapolation"," *Numer. Algorithms*, vol. 80, no. 1, pp. 135–234, 2019.

[3] R. M. Martin, *Electronic Structure: Basic Theory and Practical Methods*. Cambridge University Press, 2004.

[4] C. Yang, W. Gao, and J. C. Meza, "On the convergence of the self-consistent field iteration for a class of nonlinear eigenvalue problems," *SIAM J. Matrix Anal. Appl.*, vol. 30, no. 4, pp. 1773–1788, 2008/09.

[5] Y. Saad, *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, Philadelphia, PA, second ed., 2003.

[6] H. F. Walker and P. Ni, "Anderson acceleration for fixed-point iterations," *SIAM J. Numer. Anal.*, vol. 49, no. 4, pp. 1715–1735, 2011.

[7] T. Rohwedder and R. Schneider, "An analysis for the DIIS acceleration method used in quantum chemistry calculations," *J. Math. Chem.*, vol. 49, no. 9, pp. 1889–1914, 2011.

[8] V. Eyert, "A comparative study on methods for convergence acceleration of iterative vector sequences," *J. Comput. Phys.*, vol. 124, no. 2, pp. 271–285, 1996.

[9] H. Fang and Y. Saad, "Two classes of multisecant methods for nonlinear acceleration," *Numer. Linear Algebra Appl.*, vol. 16, no. 3, pp. 197–221, 2009.

[10] A. Toth and C. T. Kelley, "Convergence analysis for Anderson acceleration," *SIAM J. Numer. Anal.*, vol. 53, no. 2, pp. 805–819, 2015.

[11] L. G. Rebholz and M. Xiao, "The effect of Anderson acceleration on superlinear and sublinear convergence," *J. Sci. Comput.*, vol. 96, no. 2, Paper No. 34, 23, 2023.

[12] H. De Sterck and Y. He, "Linear asymptotic convergence of Anderson acceleration: Fixed-point analysis," *SIAM J. Matrix Anal. Appl.*, vol. 43, no. 4, pp. 1755–1783, 2022.

[13] H. De Sterck, Y. He, and O. A. Krzysik, "Anderson acceleration as a Krylov method with application to convergence analysis," *J. Sci. Comput.*, vol. 99, no. 1, Paper No. 12, 30, 2024.

[14] S. Pollock and L. G. Rebholz, "Anderson acceleration for contractive and noncontractive operators," *IMA J. Numer. Anal.*, vol. 41, pp. 2841–2872, 01 2021.

[15] S. Pollock and L. G. Rebholz, "Filtering for Anderson acceleration," *SIAM J. Sci. Comput.*, vol. 45, no. 4, pp. A1571–A1590, 2023.

[16] F. Wei, C. Bao, Y. Liu, and G. Yang, "Convergence analysis for restarted Anderson mixing and beyond," *arXiv:2307.02062*, 2023. https://arxiv.org/abs/2307.02062.

[17] M. Chupin, M.-S. Dupuy, G. Legendre, and É. Séré, "Convergence analysis of adaptive DIIS algorithms with application to electronic ground state calculations," *ESAIM Math. Model. Numer. Anal.*, vol. 55, pp. 2785–2825, 2021.

[18] 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 J. Numer. Anal.*, vol. 58, no. 1, pp. 788–810, 2020.

[19] S. Pollock, L. G. Rebholz, and M. Xiao, "Anderson-accelerated convergence of Picard iterations for incompressible Navier–Stokes equations," *SIAM J. Numer. Anal.*, vol. 57, no. 2, pp. 615–637, 2019.

[20] M. Ziółkowski, V. Weijo, P. Jørgensen, and J. Olsen, "An efficient algorithm for solving nonlinear equations with a minimal number of trial vectors: Applications to atomic-orbital based coupled-cluster theory," *J. Chem. Phys*, vol. 128, no. 20, p. 204105, 2008.

[21] P. Ettenhuber and P. Jørgensen, "Discarding information from previous iterations in an optimal way to solve the coupled cluster amplitude equations," *J. Chem. Theory. Comput.*, vol. 11, no. 4, pp. 1518–1524, 2015.

[22] E. Stiefel, "Relaxationsmethoden bester Strategie zur Lösung linearer Gleichungssysteme," *Comment. Math. Helv.*, vol. 29, pp. 157–179, 1955.

[23] H. He, Z. Tang, S. Zhao, Y. Saad, and Y. Xi, "nlTGCR: a class of nonlinear acceleration procedures based on conjugate residuals," *SIAM J. Matrix Anal. Appl.*, vol. 45, no. 1, pp. 712–743, 2024.

[24] S. C. Eisenstat, H. C. Elman, and M. H. Schultz, "Variational iterative methods for nonsymmetric systems of linear equations," *SIAM J. Numer. Anal.*, vol. 20, no. 2, pp. 345–357, 1983.

[25] C. L. Bris, "Computational chemistry from the perspective of numerical analysis," *Acta Numer.*, vol. 14, pp. 363–444, 2005.

[26] L.-H. Zhang, L.-Z. Liao, and M. K. Ng, "Fast algorithms for the generalized Foley-

Sammon discriminant analysis," *SIAM J. Matrix Anal. Appl.*, vol. 31, no. 4, pp. 1584–1605, 2009/10.

[27] T. T. Ngo, M. Bellalij, and Y. Saad, "The trace ratio optimization problem for dimensionality reduction," *SIAM J. Matrix Anal. Appl.*, vol. 31, no. 5, pp. 2950–2971, 2010.

[28] L. Jost, S. Setzer, and M. Hein, "Nonlinear eigenproblems in data analysis: balanced graph cuts and the RatioDCA-prox," in *Extraction of Quantifiable Information from Complex Systems*, vol. 102 of *Lect. Notes Comput. Sci. Eng.*, pp. 263–279, Springer, Cham, 2014.

[29] R. Martin, *Electronic Structure: Basic Theory and Practical Methods*. Cambridge University Press, 2004.

[30] L. Lin and J. Lu, *A Mathematical Introduction to Electronic Structure Theory*, vol. 4 of *SIAM Spotlights*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2019.

[31] Z. Bai, D. Lu, and B. Vandereycken, "Robust Rayleigh quotient minimization and nonlinear eigenvalue problems," *SIAM J. Sci. Comput.*, vol. 40, no. 5, pp. A3495–A3522, 2018.

[32] D. Lu, "Nonlinear eigenvector methods for convex minimization over the numerical range." http://www.unige.ch/~dlu/publications/subsprq_preprint.pdf, 2018.

[33] C. Yang, J. C. Meza, and L.-W. Wang, "A trust region direct constrained minimization algorithm for the Kohn-Sham equation," *SIAM J. Sci. Comput.*, vol. 29, no. 5, pp. 1854–1875, 2007.

[34] T. Helgaker, P. Jørgensen, and J. Olsen, *Molecular Electronic-Structure Theory*. John Wiley & Sons, Ltd, 2000.

[35] Y. Cai, L.-H. Zhang, Z. Bai, and R.-C. Li, "On an eigenvector-dependent nonlinear eigenvalue problem," *SIAM J. Matrix Anal. Appl.*, vol. 39, no. 3, pp. 1360–1382, 2018.

[36] X. Liu, X. Wang, Z. Wen, and Y. Yuan, "On the convergence of the self-consistent field iteration in Kohn-Sham density functional theory," *SIAM J. Matrix Anal. Appl.*, vol. 35, no. 2, pp. 546–558, 2014.

[37] P. Upadhyaya, E. Jarlebring, and E. H. Rubensson, "A density matrix approach to the convergence of the self-consistent field iteration," *Numer. Algebra Control Optim.*, vol. 11, no. 1, pp. 99–115, 2021.

[38] J. Koutecký and V. Bonačić, "On convergence difficulties in the iterative Hartree-Fock procedure," *J. Chem. Phys*, vol. 55, no. 5, pp. 2408–2413, 1971.

[39] V. Saunders and I. Hillier, "A "Level-Shifting" method for converging closed shell Hartree-Fock wave functions," *Int. J. Quantum Chem.*, vol. 7, no. 4, pp. 699–705, 1973.

[40] L. Thøgersen, J. Olsen, D. L. Yeager, P. Jørgensen, P. Sałek, and T. Helgaker, "The trust-region self-consistent field method: Towards a black-box optimization in Hartree-Fock and Kohn-Sham theories," *J. Chem. Phys*, vol. 121, no. 1, pp. 16–27, 2004.

[41] G. P. Kerker, "Efficient iteration scheme for self-consistent pseudopotential calculations," *Phys. Rev. B*, vol. 23, pp. 3082–3084, 1981.

[42] D. D. Johnson, "Modified Broyden's method for accelerating convergence in self-consistent calculations," *Phys. Rev. B*, vol. 38, pp. 12807–12813, 1988.

[43] P. Pulay, "Convergence acceleration of iterative sequences. The case of SCF iteration," *Chem. Phys. Lett.*, vol. 73, no. 2, pp. 393–398, 1980.

[44] P. Pulay, "Improved SCF convergence acceleration," *J. Comput. Chem.*, vol. 3, no. 4, pp. 556–560, 1982.

[45] T. Rohwedder and R. Schneider, "An analysis for the DIIS acceleration method used in quantum chemistry calculations," *J. Math. Chem.*, vol. 49, no. 9, pp. 1889–1914, 2011.

[46] A. S. Banerjee, P. Suryanarayana, and J. E. Pask, "Periodic Pulay method for robust and efficient convergence acceleration of self-consistent field iterations," *Chem. Phys. Lett*, vol. 647, pp. 31 – 35, 2016.

[47] C. T. Kelley, *Solving Nonlinear Equations with Newton's Method*. Philadelphia: SIAM, 2003.

[48] E. Jarlebring, A. Koskela, and G. Mele, "Disguised and new quasi-Newton methods for nonlinear eigenvalue problems," *Numer. Algorithms*, vol. 79, no. 1, pp. 311–335, 2018.

[49] J. E. Dennis and J. J. Moré, "Quasi-Newton methods, motivation and theory," *SIAM Rev.*, vol. 19, no. 1, pp. 46–89, 1977.

[50] C. Broyden, "A class of methods for solving nonlinear simultaneous equations," *Math. Comp.*, vol. 19, no. 92, pp. 577 – 593, 1965.

[51] M. J. D. Powell, "A new algorithm for unconstrained optimization," in *Nonlinear Programming (Proc. Sympos., Univ. of Wisconsin, Madison, Wis., 1970)*, pp. 31–65, Academic Press, New York-London, 1970.

[52] C. G. Broyden, "The convergence of a class of double-rank minimization algorithms. II. The new algorithm," *J. Inst. Math. Appl.*, vol. 6, pp. 222–231, 1970.

[53] R. Fletcher, "A new approach to variable metric algorithms," *Comput. J.*, vol. 13, pp. 317–322, 01 1970.

[54] D. Goldfarb, "A family of variable-metric methods derived by variational means," *Math. Comp.*, vol. 24, pp. 23–26, 1970.

[55] D. F. Shanno, "Conditioning of quasi-Newton methods for function minimization," *Math. Comp.*, vol. 24, pp. 647–656, 1970.

[56] W. C. Davidon, "Variable metric method for minimization," *SIAM J. Optim.*, vol. 1, no. 1, pp. 1–17, 1991.

[57] J. E. Dennis, Jr. and J. J. Moré, "A characterization of superlinear convergence and its application to quasi-Newton methods," *Math. Comp.*, vol. 28, pp. 549–560, 1974.

[58] D. M. Gay and R. B. Schnabel, "Solving systems of nonlinear equations by Broyden's method with projected updates," in *Nonlinear Programming, 3 (Proc. Sympos., Special Interest Group Math. Programming, Univ. Wisconsin, Madison, Wis., 1977)*, pp. 245–281, Academic Press, New York-London, 1978.

[59] R. B. Schnabel, "Quasi-Newton Methods Using Multiple Secant Equations," Tech. Rep. CU-CS-247-83, University of Colorado at Boulder, USA, Department of Computer Science, 1983.

[60] D. Knoll and D. Keyes, "Jacobian-free Newton-Krylov methods: a survey of approaches and applications," *J. Comput. Phys.*, vol. 193, no. 2, pp. 357–397, 2004.

[61] R. S. Dembo, S. C. Eisenstat, and T. Steihaug, "Inexact Newton methods," *SIAM J. Numer. Anal.*, vol. 19, no. 2, pp. 400–408, 1982.

[62] M. R. Hestenes and E. Stiefel, "Methods of conjugate gradients for solving linear systems," *J. Res. Natl. Bur. Stand.*, vol. 49, no. 6, pp. 409–436, 1952.

[63] P. K. Vinsome, "Orthomin, an iterative method for solving sparse sets of simultaneous linear equations," *SPE Symposium on Numerical Simulation of Reservoir Performance*, 1976.

[64] C. Lanczos, "An iteration method for the solution of the eigenvalue problem of linear differential and integral operators," *J. Research Nat. Bur. Standards*, vol. 45, pp. 255–282, 1950.

[65] C. C. Paige and M. A. Saunders, "Solution of sparse indefinite systems of linear equations," *SIAM J. Numer. Anal.*, vol. 12, no. 4, pp. 617–629, 1975.

[66] Y. Saad and M. H. Schultz, "GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems," *SIAM J. Sci. Stat. Comput.*, vol. 7, no. 3, pp. 856–869, 1986.

[67] Y. Saad and M. H. Schultz, "Conjugate gradient-like algorithms for solving nonsymmetric linear systems," *Math. Comp.*, vol. 44, no. 170, pp. 417–424, 1985.

[68] T. Eirola and O. Nevanlinna, "Accelerating with rank-one updates," *Linear Algebra Appl.*, vol. 121, pp. 511–520, 1989.

[69] U. Meier Yang and K. A. Gallivan, "A new family of preconditioned iterative solvers for nonsymmetric linear systems," *Appl. Numer. Math.*, vol. 19, no. 3, pp. 287–317, 1995. Special issue on iterative methods for linear equations (Atlanta, GA, 1994).

[70] Y. Saad, "A flexible inner-outer preconditioned GMRES algorithm," *SIAM J. Sci. Comput.*, vol. 14, no. 2, pp. 461–469, 1993.

[71] H. A. van der Vorst and C. Vuik, "GMRESR: a family of nested GMRES methods," *Numer. Linear Algebra Appl.*, vol. 1, no. 4, pp. 369–386, 1994.

[72] E. de Sturler, "Nested Krylov methods based on GCR," *J. Comput. Appl. Math.*, vol. 67, no. 1, pp. 15–41, 1996.

[73] E. de Sturler, "Truncation strategies for optimal Krylov subspace methods," *SIAM J. Numer. Anal.*, vol. 36, no. 3, pp. 864–889, 1999.

[74] A. H. Baker, E. R. Jessup, and T. Manteuffel, "A technique for accelerating the convergence of restarted GMRES," *SIAM J. Matrix Anal. Appl.*, vol. 26, no. 4, pp. 962–984, 2005.

[75] J. E. Hicken and D. W. Zingg, "A simplified and flexible variant of GCROT for solving nonsymmetric linear systems," *SIAM J. Sci. Comput.*, vol. 32, no. 3, pp. 1672–1694, 2010.

[76] A. Ruhe, "Implementation aspects of band Lanczos algorithms for computation of eigenvalues of large sparse symmetric matrices," *Math. Comp.*, vol. 33, no. 146, pp. 680–687, 1979.

[77] M. Sadkane, "Block-Arnoldi and Davidson methods for unsymmetric large eigenvalue problems," *Numer. Math.*, vol. 64, no. 1, pp. 195–211, 1993.

[78] K. Jbilou, A. Messaoudi, and H. Sadok, "Global FOM and GMRES algorithms for matrix equations," *Appl. Numer. Math.*, vol. 31, no. 1, pp. 49–63, 1999.

[79] J. Meng, X.-M. Gu, W.-H. Luo, and L. Fang, "A flexible global GCRO-DR method for shifted linear systems and general coupled matrix equations," *J. Math.*, pp. 1–17, 2021.

[80] F. Toutounian and S. Karimi, "Global least squares method (Gl-LSQR) for solving general linear systems with several right-hand sides," *Appl. Math. Comput.*, vol. 178, no. 2, pp. 452–460, 2006.

[81] Y. Wang and G.-d. Gu, "Global quasi-minimal residual method for the Sylvester equations," *J. Shanghai Univ.*, vol. 11, no. 1, pp. 52–57, 2007.

[82] N. A. Zadeh, A. Tajaddini, and G. Wu, "Weighted and deflated global GMRES algorithms for solving large Sylvester matrix equations," *Numer. Algorithms*, vol. 82, no. 1, pp. 155–181, 2019.

[83] T. Werner, "An inexact Matrix-Newton method for solving NEPv," *Linear Algebra Appl.*, 2024.

[84] A. C. Aitken, "On Bernoulli's numerical solution of algebraic equations," *Proc. R. Soc. Edinb.*, vol. 46, pp. 289–305, 1926.

[85] D. Shanks, "Non-linear transformations of divergent and slowly convergent sequences," *J. Math. Phys*, vol. 34, no. 1-4, pp. 1–42, 1955.

[86] C. Brezinski, S. Cipolla, M. Redivo-Zaglia, and Y. Saad, "Shanks and Anderson-type acceleration techniques for systems of nonlinear equations," *IMA J. Numer. Anal.*, vol. 42, no. 4, pp. 3058–3093, 2022.

[87] P. Wynn, "On a device for computing the $e_m(S_n)$ transformation," *MTAC*, vol. 10, no. 54, pp. 91–96, 1956.

[88] P. Wynn, "Acceleration techniques for iterated vector and matrix problems," *Math. Comp.*, vol. 16, no. 79, pp. 301–322, 1962.

[89] C. Brezinski, "Application de l'ε-algorithme à la résolution des systèmes non linéaires," *C. R. Acad. Sci. Paris Sér. A-B*, vol. 271, pp. A1174–A1177, 1970.

[90] C. Brezinski and M. Redivo-Zaglia, "The simplified topological ε-algorithms for accelerating sequences in a vector space," *SIAM J. Sci. Comput.*, vol. 36, no. 5, pp. A2227–A2247, 2014.

[91] S. Cabay and L. W. Jackson, "A polynomial extrapolation method for finding limits and antilimits of vector sequences," *SIAM J. Numer. Anal.*, vol. 13, no. 5, pp. 734–752, 1976.

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

[93] A. Sidi, W. F. Ford, and D. A. Smith, "Acceleration of convergence of vector sequences," *SIAM J. Numer. Anal.*, vol. 23, no. 1, pp. 178–196, 1986.

[94] K. Jbilou and H. Sadok, "Vector extrapolation methods. Applications and numerical comparison," in *Numerical Analysis 2000, Vol. II: Interpolation and Extrapolation*, vol. 122 of *J. Comput. Appl. Math.*, pp. 149–165, 2000.

[95] C. Brezinski, "Convergence acceleration during the 20th century," in *Numerical Analysis 2000, Vol. II: Interpolation and Extrapolation*, vol. 122 of *J. Comput. Appl. Math.*, pp. 1–21, 2000.

[96] C. Brezinski, M. Redivo-Zaglia, and Y. Saad, "Shanks sequence transformations and Anderson acceleration," *SIAM Rev.*, vol. 60, no. 3, pp. 646–669, 2018.

[97] C. Brezinski and M. Redivo-Zaglia, *Extrapolation and rational approximation. The works of the main contributors.* Springer Nature Switzerland AG, 2020.

[98] E. Cancès and C. L. Bris, "Can we outperform the DIIS approach for electronic structure calculations?," *Int. J. Quantum Chem.*, vol. 79, no. 2, pp. 82–90, 2000.

[99] E. Cancès and C. L. Bris, "On the convergence of SCF algorithms for the Hartree-Fock equations," *M2AN Math. Model. Numer. Anal.*, vol. 34, no. 4, pp. 749–774, 2000.

[100] L. Lin, J. Lu, and L. Ying, "Numerical methods for Kohn-Sham density functional theory," *Acta Numer.*, vol. 28, pp. 405–539, 2019.

[101] E. Cancès, G. Kemlin, and A. Levitt, "Convergence analysis of direct minimization and self-consistent iterations," *SIAM J. Matrix Anal. Appl.*, vol. 42, no. 1, pp. 243–274, 2021.

[102] P. Ni, *Anderson Acceleration of Fixed-Point Iteration with Applications to Electronic Structure Computations*. PhD thesis, Worcester Polytechnic Institute, 2009.

[103] A. S. Banerjee, P. Suryanarayana, and J. E. Pask, "Periodic Pulay method for robust and efficient convergence acceleration of self-consistent field iterations," *Chem. Phys. Lett.*, vol. 647, pp. 31–35, 2016.

[104] P. Suryanarayana, P. P. Pratapa, and J. E. Pask, "Alternating Anderson–Richardson method: An efficient alternative to preconditioned Krylov methods for large, sparse linear systems," *Comput. Phys. Commun.*, vol. 234, pp. 278–285, 2019.

[105] X. Chen and C. T. Kelley, "Convergence of the EDIIS algorithm for nonlinear equations," *SIAM J. Sci. Comput.*, vol. 41, no. 1, pp. A365–A379, 2019.

[106] K. Chen and C. Vuik, "Composite Anderson acceleration method with two window sizes and optimized damping," *Int. J. Numer. Methods Eng.*, vol. 123, no. 23, pp. 5964–5985, 2022.

[107] H. W. Peng Ni, "A linearly constrained least-squares problem in electronic structure computations," *ICCES*, vol. 7, no. 1, pp. 43–50, 2008.

[108] D. Tomatis, R. Gross, and E. Gilad, "On the Ronen method in simple 1-D geometries for neutron transport theory solutions," *J. Comput. Theor. Transp.*, vol. 50, no. 2, pp. 134–157, 2021.

[109] K. Sun, Y. Wang, Y. Liu, Y. Zhao, B. Pan, S. Jui, B. Jiang, and L. Kong, "Damped Anderson mixing for deep reinforcement learning: Acceleration, convergence, and stabilization," in *Advances in Neural Information Processing Systems* (M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, eds.), vol. 34, pp. 3732–3743, Curran Associates, Inc., 2021.

[110] N. Wan and A. Międlar, "On the convergence of CROP-Anderson acceleration method," *arXiv:2410.03970*, 2024. https://arxiv.org/abs/2410.03970.

[111] T. Washio and C. W. Oosterlee, "Krylov subspace acceleration for nonlinear multigrid schemes," *Electron. Trans. Numer. Anal.*, vol. 6, pp. 271–290, 1997. Special issue on multilevel methods (Copper Mountain, CO, 1997).

[112] C. W. Oosterlee and T. Washio, "Krylov subspace acceleration of nonlinear multigrid with application to recirculating flows," *SIAM J. Sci. Comput.*, vol. 21, no. 5, pp. 1670–1690, 2000. Iterative methods for solving systems of algebraic equations (Copper Mountain, CO, 1998).

[113] T. A. Davis and Y. Hu, "The University of Florida sparse matrix collection," *ACM Transactions on Mathematical Software (TOMS)*, vol. 38, no. 1, pp. 1–25, 2011.

[114] C. T. Kelley, "Numerical methods for nonlinear equations," *Acta Numer.*, vol. 27, p. 207–287, 2018.

[115] A. Toth, J. A. Ellis, T. Evans, S. Hamilton, C. T. Kelley, R. Pawlowski, and S. Slat-

tery, "Local improvement results for Anderson acceleration with inaccurate function evaluations," *SIAM J. Sci. Comput.*, vol. 39, no. 5, pp. S47–S65, 2017.

[116] J. Willert, C. T. Kelley, D. A. Knoll, and H. Park, "Hybrid deterministic/Monte Carlo neutronics," *SIAM J. Sci. Comput.*, vol. 35, no. 5, pp. S62–S83, 2013.

[117] J. Willert, X. Chen, and C. T. Kelley, "Newton's method for Monte Carlo–based residuals," *SIAM J. Numer. Anal.*, vol. 53, no. 4, pp. 1738–1757, 2015.

[118] E. Jarlebring, W. Michiels, and K. Meerbergen, "The infinite arnoldi method and an application to time-delay systems with distributed delays," in *Time Delay Systems: Methods, Applications and New Trends* (R. Sipahi, T. Vyhlídal, S.-I. Niculescu, and P. Pepe, eds.), pp. 229–239, Berlin, Heidelberg: Springer Berlin Heidelberg, 2012.

[119] T. Werner, N. Wan, and A. Mięldlar, "nlKrylov: A unified framework for nonlinear GCR-type Krylov subspace methods," *arXiv:2511.14713*, 2025. https://arxiv.org/abs/2511.14713.

[120] H. A. van der Vorst, "Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems," *SIAM J. Sci. Stat. Comput.*, vol. 13, no. 2, pp. 631–644, 1992.

[121] K. M. Soodhalter, E. De Sturler, and M. E. Kilmer, "A survey of subspace recycling iterative methods," *GAMM-Mitteilungen*, vol. 43, no. 4, 2020.

[122] A. Chronopoulos, "Nonlinear CG-like iterative methods," *J. Comput. Appl. Math.*, vol. 40, no. 1, pp. 73 – 89, 1992.

[123] Y. Chen and D. Cai, "Nonlinear orthomin(k) methods," *Appl. Math. Comput.*, vol. 124, no. 3, pp. 351–363, 2001.

[124] P. Deuflhard, "Least Squares Problems: Gauss-Newton Methods," in *Newton Methods for Nonlinear Problems: Affine Invariance and Adaptive Algorithms*, pp. 173–231, Berlin, Heidelberg: Springer Berlin Heidelberg, 2011.

[125] T. Shores and C. Tiahrt, "Subspace Projection Variants on Newton's Method," *Comput. Math. Appl.*, vol. 27, pp. 7–17, 1994.

[126] G. W. Reddien, "On Newton's method for singular problems," *SIAM J. Numer. Anal.*, vol. 15, no. 5, pp. 993–996, 1978.

[127] G. W. Reddien, "Newton's method and high order singularities," *Comput. Math. Appl.*, vol. 5, no. 2, pp. 79–86, 1979.

[128] D. W. Decker and C. T. Kelley, "Newton's method at singular points. I," *SIAM J. Numer. Anal.*, vol. 17, no. 1, pp. 66–70, 1980.

[129] D. W. Decker and C. T. Kelley, "Convergence acceleration for Newton's method at singular points," *SIAM J. Numer. Anal.*, vol. 19, no. 1, pp. 219–229, 1982.

[130] D. W. Decker, H. B. Keller, and C. T. Kelley, "Convergence rates for Newton's method at singular points," *SIAM J. Numer. Anal.*, vol. 20, no. 2, pp. 296–314, 1983.

[131] C. T. Kelley and R. Suresh, "A new acceleration method for Newton's method at singular points," *SIAM J. Numer. Anal.*, vol. 20, no. 5, pp. 1001–1009, 1983.

[132] A. O. Griewank, "Starlike domains of convergence for Newton's method at singularities," *Numer. Math.*, vol. 35, no. 1, pp. 95–111, 1980.

[133] A. Griewank and M. R. Osborne, "Newton's method for singular problems when the dimension of the null space is > 1," *SIAM J. Numer. Anal.*, vol. 18, no. 1, pp. 145–149, 1981.

[134] C. T. Kelley and Z. Q. Xue, "Inexact Newton methods for singular problems," *Optim. Methods Softw.*, vol. 2, no. 3-4, pp. 249–267, 1993.

[135] I. K. Argyros, "Convergence rates for inexact Newton-like methods at singular points and applications," *Appl. Math. Comput.*, vol. 102, p. 185–201, July 1999.

[136] D. W. Decker and C. T. Kelley, "Broyden's method for a class of problems having singular Jacobian at the root," *SIAM J. Numer. Anal.*, vol. 22, no. 3, pp. 566–574, 1985.

[137] F. Mannel, "On the convergence of Broyden's method and some accelerated schemes for singular problems," *IMA J. Numer. Anal.*, vol. 43, no. 1, pp. 414–442, 2023.

[138] J. Chen, "The convergence analysis of inexact Gauss-Newton methods for nonlinear problems," *Comput. Optim. Appl.*, vol. 40, no. 1, pp. 97–118, 2008.

[139] O. P. Ferreira and M. L. N. Gonçalves, "Local convergence analysis of inexact Newton-like methods under majorant condition," *Comput. Optim. Appl.*, vol. 48, no. 1, pp. 1–21, 2011.

[140] F. Zhou, "An analysis on local convergence of inexact Newton-Gauss method for solving singular systems of equations," *Sci. World J.*, vol. 2014, no. 1, p. 752673, 2014.

[141] F. Zhou, "On local convergence analysis of inexact Newton method for singular systems of equations under majorant condition," *Sci. World J.*, vol. 2014, no. 1, p. 498016, 2014.

[142] I. K. Argyros and D. González, "Local convergence analysis of inexact Gauss-Newton method for singular systems of equations under majorant and center-majorant condition," *SeMA J.*, vol. 69, pp. 37–51, 2015.

[143] G. A. Anastassiou and I. K. Argyros, "Inexact Gauss-Newton method for singular equations," in *Intelligent Numerical Methods: Applications to Fractional Calculus*, pp. 263–281, Cham: Springer International Publishing, 2016.

[144] Z. Tang, T. Xu, H. He, Y. Saad, and Y. Xi, "Anderson acceleration with truncated Gram-Schmidt," *SIAM J. Matrix Anal. Appl.*, vol. 45, no. 4, pp. 1850–1872, 2024.

[145] C. Arias and C. Gómez, "Inexact free derivative quasi-Newton method for large-scale nonlinear system of equations," *Numer. Algorithms*, vol. 94, pp. 1103–1123, 2023.

[146] L. Armijo, "Minimization of functions having Lipschitz continuous first partial derivatives.," *Pacific J. Math.*, vol. 16, no. 1, pp. 1 – 3, 1966.

[147] A. H. Al-Mohy and N. J. Higham, "The complex step approximation to the Fréchet derivative of a matrix function," *Numer. Algorithms*, vol. 53, no. 1, pp. 133–148, 2010.

[148] S. C. Eisenstat and H. F. Walker, "Choosing the Forcing Terms in an Inexact Newton Method," *SIAM J. Sci. Comput.*, vol. 17, no. 1, pp. 16–32, 1996.

[149] H. He, Z. Tang, S. Zhao, Y. Saad, and Y. Xi, "GitHub-Project on "Nonlinear-Truncated-Conjugate-Residual"," 2024.

[150] S. Chandrasekhar, *Radiative Transfer*. Dover Publications, New York, 1960.

[151] M. Hajipour, A. Jajarmi, and D. Baleanu, "On the accurate discretization of a highly nonlinear boundary value problem," *Numer. Algorithms*, vol. 79, no. 3, pp. 679–695, 2018.

[152] P. Benner, P. Kürschner, and J. Saak, "Low-rank Newton-ADI methods for large nonsymmetric algebraic Riccati equations," *J. Franklin Inst.*, vol. 353, no. 5, pp. 1147–1167, 2016.

[153] T. Betcke, N. J. Higham, V. Mehrmann, C. Schröder, and F. Tisseur, "NLEVP: A collection of nonlinear eigenvalue problems," *ACM Trans. Math. Softw.*, vol. 39, no. 2, pp. 1–28, 2013.

[154] N. J. Higham, G. M. N. Porzio, and F. Tisseur, "An updated set of nonlinear eigenvalue problems," tech. rep., Manchester Institute for Mathematical Sciences, The University of Manchester, UK, 2019.

[155] S. Güttel and F. Tisseur, "The nonlinear eigenvalue problem," *Acta Numer.*, vol. 26, pp. 1–94, 2017.

[156] Y. Cai, L.-H. Zhang, Z. Bai, and R.-C. Li, "On an eigenvector-dependent nonlinear eigenvalue problem," *SIAM J. Matrix Anal. Appl.*, vol. 39, no. 3, pp. 1360–1382, 2018.

[157] E. Jarlebring, S. Kvaal, and W. Michiels, "An inverse iteration method for eigenvalue problems with eigenvector nonlinearities," *SIAM J. Sci. Comput.*, vol. 36, no. 4, pp. A1978–A2001, 2014.

[158] Z. Bai, D. Lu, and B. Vandereycken, "Robust Rayleigh quotient minimization and nonlinear eigenvalue problems," *SIAM J. Sci. Comput.*, vol. 40, no. 5, pp. A3495–A3522, 2018.

[159] Z. Bai and D. Lu, "Variational characterization of monotone nonlinear eigenvector problems and geometry of self-consistent field iteration," *SIAM J. Matrix Anal. Appl.*, vol. 45, no. 1, pp. 84–111, 2024.

[160] R. Claes and K. Meerbergen, "Interpolating self consistent field for eigenvector nonlinearities," *Appl. Math. Lett.*, vol. 135, p. 108412, 2023.

[161] A. J. Garza and G. E. Scuseria, "Comparison of self-consistent field convergence acceleration techniques," *J. Chem. Phys.*, vol. 137, p. 054110, 08 2012.