
An Efficient Nonlinear Acceleration method that Exploits Symmetry of the Hessian

Huan He

Harvard University
huan_he@hms.harvard.edu

Shifan Zhao

Emory University
szhao89@emory.edu

Ziyuan Tang

University of Minnesota
tang0389@umn.edu

Joyce C Ho

Emory University
joyce.c.ho@emory.edu

Yousef Saad

University of Minnesota
saad@umn.edu

Yuanzhe Xi

Emory University
yuanzhe.xi@emory.edu

Abstract

Nonlinear acceleration methods are powerful techniques to speed up fixed-point iterations. However, many acceleration methods require storing a large number of previous iterates and this can become impractical if computational resources are limited. In this paper, we propose a nonlinear Truncated Generalized Conjugate Residual method (nTGCR) whose goal is to exploit the symmetry of the Hessian to reduce memory usage. The proposed method can be interpreted as either an inexact Newton or a quasi-Newton method. We show that, with the help of global strategies like residual check techniques, nTGCR can converge globally for general nonlinear problems and that under mild conditions, nTGCR is able to achieve superlinear convergence. We further analyze the convergence of nTGCR in a stochastic setting. Numerical results demonstrate the superiority of nTGCR when compared with several other competitive baseline approaches on a few problems. Our code will be available in the future.

1 Introduction

In this paper, we consider solving the fixed-point problem:

$$\text{Find } x \in \mathbb{R}^n \text{ such that } x = H(x). \quad (1)$$

This problem has received a surge of interest due to its wide range of applications in mathematics, computational science and engineering. Most optimization algorithms are iterative, and their goal is to find a related fixed-point of the form (1), where $H : \mathbb{R}^n \rightarrow \mathbb{R}^n$ is the iteration mapping which can potentially be nonsmooth or noncontractive. When the optimization problem is convex, H is typically nonexpansive, and the solution set of the fixed-point problem is the same as that of the original optimization problem, or closely related to it. Consider the simple fixed-point iteration $x_{k+1} = H(x_k)$ which produces a sequence of iterates $\{x_0, x_1, \dots, x_K\}$. When the iteration converges, its limit is a fixed-point, i.e., $x^* = H(x^*)$. However, an issue with fixed-point iteration is that it does not always converge, and when it does, it might reach the limit very slowly.

To address this issue, a number of acceleration methods have been proposed and studied over the years, such as the reduced-rank extrapolation (RRE) [58], minimal-polynomial extrapolation (MPE) [12], modified MPE (MMPE) [33], and the vector ϵ -algorithms [8]. Besides these algorithms, Anderson Acceleration (AA) [1] has received enormous recent attention due to its nice properties and its success in machine learning applications [56, 22, 57, 14, 60, 44, 25, 63]. In practice, since computing the Hessian of the objective function is commonly difficult or even unavailable, AA can be seen as a practical alternative to Newton's method [34]. Also, compared with the classical iterative methods

such as the nonlinear conjugate gradient (CG) method [24], no line-search or trust-region technique is performed in AA, and this is a big advantage in large-scale unconstrained optimization. Empirically, it is observed that AA is quite successful in accelerating convergence. We refer readers to [9] for a recent survey of acceleration methods.

However, classical AA has one undesirable disadvantage in that it is expensive in terms of memory as well as computational cost, especially in a nonconvex stochastic setting, where only sublinear convergence can be expected when only stochastic gradients can be accessed [3]. In light of this, a number of variants of AA have been proposed which aim at improving its performance and robustness (e.g., [39, 63, 62, 56, 67]). The above-cited works focus on improving the convergence behavior of AA, but they do not consider reducing the memory cost. In machine learning, we often encounter practical situations where the number of parameters is quite large and for this reason, it is not practical to use a large number of vectors in the acceleration methods. It is not clear whether or not the symmetric structure of the Hessian can be exploited in a scheme like AA to reduce the memory cost while still maintaining the convergence guarantees. In this paper, we will demonstrate how this can be accomplished with a new algorithm that is superior to AA in practice.

Our contributions. This paper develops a nonlinear acceleration method, nonlinear Truncated Generalized Conjugate Residual method (nTGCR), that takes advantage of symmetry. This work is motivated by the observation that the Hessian of a nonlinear function, or the Jacobian of a gradient of a mapping, f is symmetric and therefore more effective, conjugate gradient-like schemes can be exploited.

We demonstrate that nonlinear acceleration methods can benefit from the symmetry property of the Hessian. In particular, we study both linear and nonlinear problems and give a systematic analysis of TGCR and nTGCR. We show that TGCR is efficient and optimal for linear problems. By viewing the method from the angle of an inexact Newton approach, we also show that adding a few global convergence strategies ensures that nTGCR can achieve global convergence guarantees.

We complement our theoretical results with numerical simulations on several different problems. The experimental results demonstrate advantages of our methods. To the best of our knowledge, this is still the first work to investigate and improve the AA dynamics by exploiting symmetry of the Hessian.

Related work. Designing efficient optimization methods has received much attention. Several recent works [65, 4, 40, 48, 18] consider second order optimization methods that employ sketching or approximation techniques. Different from these approaches, our method is a first-order method that utilizes symmetry of the Hessian instead of constructing it. A variant of inexact Newton method was proposed in [47] where the least-squares sub-problems are solved approximately using Minimum Residual method. Similarly, a new type of quasi Newton symmetric update [54] uses several secant equations in a least-squares sense. These approaches have the same goal as ours. However, they are more closely related to a secant or a multi-secant technique, and as will be argued it does a better job of capturing the nonlinearity of the problem. [63] proposed a short-term AA algorithm that is different from ours because it is still based on the parameter sequence instead of the gradient sequence and does not exploit symmetry of the Hessian.

2 Background

2.1 Extrapolation, acceleration, and the Anderson Acceleration procedure

Consider a general fixed-point problem and the associated fixed-point iteration as shown in (1). Denote by $r_j = H(x_j) - x_j$ the *residual* vector at the j th iteration. Classical extrapolation methods including RRE, MPE and the vector ϵ -Algorithm, have been designed to accelerate the convergence of the original sequence by generating a new and independent sequence of the form: $t_j^{(k)} = \sum_{i=0}^k \alpha_i x_{j+i}$. An important characteristic of these classical extrapolation methods is that the two sequences are not mixed in the sense that no accelerated item $t_j^{(k)}$, is used to produce the iterate x_j . These *extrapolation* methods must be distinguished from *acceleration* methods such as the AA procedure which aim at generating their own sequences to find a fixed point of a certain mapping H .

AA was originally designed to solve a system of nonlinear equations written in the form $F(x) = H(x) - x = 0$ [1, 61, 41, 30]. Denote $F_i = F(x_i)$. AA starts with an initial x_0 and sets $x_1 =$

$H(x_0) = x_0 + \beta F_0$, where $\beta > 0$ is a parameter. At step $j > 1$ we define $X_j = [x_{j-m}, \dots, x_{j-1}]$, and $\bar{F}_j = [F_{j-m}, \dots, F_{j-1}]$ along with the differences:

$$\begin{aligned}\mathcal{X}_j &= [\Delta x_{j-m} \ \cdots \ \Delta x_{j-1}] \in \mathbb{R}^{n \times m}, \\ \mathcal{F}_j &= [\Delta F_{j-m} \ \cdots \ \Delta F_{j-1}] \in \mathbb{R}^{n \times m}.\end{aligned}\tag{2}$$

We then define the next AA iterate as follows:

$$x_{j+1} = x_j + \beta F_j - (\mathcal{X}_j + \beta \mathcal{F}_j) \theta^{(j)} \quad \text{where:}\tag{3}$$

$$\theta^{(j)} = \operatorname{argmin}_{\theta \in \mathbb{R}^m} \|F_j - \mathcal{F}_j \theta\|_2.\tag{4}$$

To define the next iterate in (3) the algorithm uses the term $F_{j+1} = F(x_{j+1})$ where x_{j+1} is the current accelerated iterate. AA belongs to the class of *multi-secant methods*. Indeed, the approximation (3) can be written as:

$$\begin{aligned}x_{j+1} &= x_j - [-\beta I + (\mathcal{X}_j + \beta \mathcal{F}_j)(\mathcal{F}_j^T \mathcal{F}_j)^{-1} \mathcal{F}_j^T] F_j \\ &\equiv x_j - G_j F_j.\end{aligned}\tag{5}$$

Thus, G_j can be seen as an update to the (approximate) inverse Jacobian $G_{j-m} = -\beta I$

$$G_j = G_{j-m} + (\mathcal{X}_j - G_{j-m} \mathcal{F}_j)(\mathcal{F}_j^T \mathcal{F}_j)^{-1} \mathcal{F}_j^T,\tag{6}$$

and is the minimizer of $\|G_j + \beta I\|_F$ under the *multi-secant condition* of type II ¹

$$G_j \mathcal{F}_j = \mathcal{X}_j.\tag{7}$$

This link between AA and Broyden multi-secant type updates was first unraveled by Eyert [21] and expanded upon in [46].

2.2 Inexact and quasi-Newton methods

Given a nonlinear system of equations $F(x) = 0$. Inexact Newton methods [15, 10], start with an initial guess x_0 and compute a sequence of iterates as follows

$$\text{Solve} \quad J(x_j) \delta_j \approx -F(x_j)\tag{8}$$

$$\text{Set} \quad x_{j+1} = x_j + \delta_j\tag{9}$$

Here, $J(x_j)$ is the Jacobian of F at the current iterate x_j . In (8) the system is solved inexactly, typically by some iterative method. In quasi-Newton methods [17, 17, 54], the inverse of the Jacobian is approximated progressively. Because it is the inverse Jacobian that is approximated, the method is akin to Broyden's second (or type-II) update method. This method replaces Newton's iteration: $x_{j+1} = x_j - DF(x_j)^{-1} F(x_j)$ with $x_{j+1} = x_j - G_j F(x_j)$ where G_j approximates the inverse of the Jacobian $DF(x_j)$ at x_j by the update formula $G_{j+1} = G_j + (\Delta x_j - G_j \Delta F(x_j)) v_j^T$ in which v_j is defined in different ways see [46] for details.

3 Exploiting symmetry

In the following, we specifically consider the case where the nonlinear mapping F is the gradient of some objective function $\phi : \mathbb{R}^n \rightarrow \mathbb{R}$ to be minimized, i.e.,

$$F(x) = \nabla \phi(x).$$

In this situation, the Jacobian of F becomes $\nabla^2 \phi$ the Hessian of ϕ . An obvious observation here is that the symmetry of the Hessian is not taken into account in the approximate inverse Hessian update formula (6). This has only been considered in the literature (very) recently (e.g., [6, 55, 7]). In a 1983 report, [53] showed that the matrix G_j obtained by a multi-secant method that satisfies the secant condition (7) is symmetric iff the matrix $\mathcal{X}_j^T \mathcal{F}_j$ is symmetric. It is possible to explicitly force symmetry by employing generalizations of the symmetric versions of Broyden-type methods. Thus, the authors of [6, 7] developed a multiseccant version of the Powell Symmetric Broyden (PSB) update

¹Type I Broyden conditions involve approximations to the Jacobian, while type II conditions deal with the inverse Jacobian.

due to Powell [45] while the article [55] proposed a symmetric multiseant method based on the popular Broyden-Fletcher-Goldfarb-Shanno (BFGS) approach as well as the Davidon-Fletcher-Powell (DFP) update. However, there are a number of issues with the symmetric versions of multiseant updates, some of which are discussed in [55].

We observe that when we are close to the limit, the condition $\mathcal{X}_j^T \mathcal{F}_j = \mathcal{F}_j^T \mathcal{X}_j$ is nearly satisfied. This is because if x^* is the limit with $F(x^*) = 0$ we can write

$$\begin{aligned} F(x_k) - F(x_{k-1}) &= [F(x_k) - F(x^*)] \\ &\quad - [F(x_{k-1}) - F(x^*)] \\ &\approx \nabla^2 \phi(x^*)(x_k - x_{k-1}). \end{aligned} \tag{10}$$

This translates to $\mathcal{F}_j \approx \nabla^2 \phi(x^*) \mathcal{X}_j$ from which it follows that $\mathcal{X}_j^T \mathcal{F}_j \approx \mathcal{X}_j^T \nabla^2 \phi(x^*) \mathcal{X}_j$ which is a symmetric matrix under mild smoothness conditions on ϕ . Therefore, the issue of symmetry can be mitigated if we are able to develop nonlinear acceleration methods that take advantage of near-symmetry.

3.1 The linear case: Truncated GCR (TGCR)

We first consider solving the linear system $Ax = b$ with a general matrix A . The Generalized Conjugate Residual (GCR) algorithm, see, e.g., [19, 51], solves this linear system by building a sequence of search directions p_i , for $i = 0, \dots, j$ at step j so that the vectors Ap_i are orthogonal to each other. With this property it is easy to generate iterates that minimize the residual at each step, and this leads to GCR, see [51, pp 195-196] for details.

Next we will make two changes to GCR. First, we will develop a truncated version in which any given Ap_j is orthogonal to the previous m Ap_i 's only. This is dictated by practical considerations, because keeping all Ap_i vectors may otherwise require too much memory. Second, we will keep a set of vectors for the p_i 's and another set for the vectors $v_i \equiv Ap_i$, for $i = 1, \dots, j$ at step j in order to avoid unnecessary additional products of the matrix A with vectors. The Truncated GCR (TGCR) is summarized in Algorithm 1.

Algorithm 1 TGCR (m)

```

1: Input: Matrix  $A$ , RHS  $b$ , initial  $x_0$ .
2: Set  $r_0 \equiv b - Ax_0$ ;  $v = Ar_0$ ;
3:  $v_0 = v/\|v\|$ ;  $p_0 = r_0/\|v\|$ ;
4: for  $j = 0, 1, 2, \dots$ , Until convergence do
5:    $\alpha_j = (r_j, v_j)$ 
6:    $x_{j+1} = x_j + \alpha_j p_j$ 
7:    $r_{j+1} = r_j - \alpha_j v_j$ 
8:    $p = r_{j+1}$ ;  $v = Ap$ ;
9:    $i_0 = \max(1, j - m + 1)$ 
10:  for  $i = i_0 : j$  do
11:     $\beta_{ij} := (v, Ap_i)$ 
12:     $p := p - \beta_{ij} p_i$ ;
13:     $v := v - \beta_{ij} v_i$ ;
14:  end for
15:   $p_{j+1} := p/\|v\|$ ;    $v_{j+1} := v/\|v\|$ ;
16: end for

```

With $m = \infty$ we obtain the non-restarted GCR method, which is equivalent to the non-restarted (i.e., full) GMRES. However, when A is symmetric, but not necessarily symmetric positive definite, then TGCR (1) is identical with TGCR (m) in exact arithmetic. This leads to big savings both in terms of memory and in computational costs.

Theorem 3.1. *When the coefficient matrix A is symmetric, TGCR (m) generates the same iterates as TGCR (1) for any $m > 0$. In addition, when A is positive definite, the k -th residual vector $r_k = b - Ax_k$ satisfies the following inequality where κ is the spectral condition number of A :*

$$\|r_k\| \leq 2 \left[\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1} \right]^k \|r_0\|. \tag{11}$$

3.2 The nonlinear case: nlTGCR

Assume now that we want to solve the nonlinear problem $F(x) = 0$. We need to make three major changes to Algorithm 1. First, any residual is now the negative of $F(x)$ so Line 2 and Line 7 must be replaced by $r_0 = -F(x_0)$ and $r_{j+1} = -F(x_{j+1})$, respectively. In addition, we originally need to calculate the products Ar_0 and Ap in Line 2 and Line 8 respectively. Here A needs to be replaced by the Jacobian $J(x_j)$ of F at the current iterate. We also use the notation $P_j \equiv [p_{i_0}, \dots, p_j]$, and $V_j \equiv [J(x_{i_0})p_{i_0}, \dots, J(x_j)p_j]$. The most important change is in lines 5-6 where α_j of Algorithm 1 needs to be replaced by a vector y_j . This is because when we write the linear model used in the form of an inexact Newton method:

$$\begin{aligned} F(x_j + P_j y) &\approx F(x_j) + [J]P_j y \quad \text{where} \\ [J]P_j &\equiv [J(x_{i_0})p_{i_0}, \dots, J(x_j)p_j] = V_j. \end{aligned} \quad (12)$$

The projection method that minimizes the norm $\|F(x_j) + [J]P_j y\| = \|F(x_j) + V_j y\|$ of the right-hand side determines y in such a way that

$$\begin{aligned} F(x_j) + V_j y &\perp \text{Span}\{V_j\} \rightarrow (V_j)^T [F(x_j) + V_j y] = 0 \\ &\rightarrow y = V_j^T r_j \end{aligned} \quad (13)$$

where it is assumed the v_i 's are fully orthogonal. Note that in the linear case, it can be shown that $V_j^T r_j$ has only one nonzero component when one assumes that the vectors $J(x_i)p_i$ are fully orthogonal, i.e., that $i_0 = 1$ always. The nonlinear version of TGCR (m) is summarized in Algorithm 2 where the indication 'Use Frechet' means that the vector $v = J(x)u$ is to be computed as $v = (F(x + \epsilon u) - F(x))/\epsilon$ for some small ϵ .

Algorithm 2 nlTGCR (m)

- 1: **Input:** $F(x)$, initial x_0 .
 - 2: Set $r_0 = -F(x_0)$.
 - 3: Compute $v = J(x_0)r_0$; (Use Frechet)
 - 4: $v_0 = v/\|v\|$, $p_0 = r_0/\|v\|$;
 - 5: **for** $j = 0, 1, 2, \dots$, Until convergence **do**
 - 6: $y_j = V_j^T r_j$
 - 7: $x_{j+1} = x_j + P_j y_j$
 - 8: $r_{j+1} = -F(x_{j+1})$
 - 9: Set: $p := r_{j+1}$;
 - 10: $i_0 = \max(1, j - m + 1)$
 - 11: Compute $v = J(x_{j+1})p$ (Use Frechet)
 - 12: **for** $i = i_0 : j$ **do**
 - 13: $\beta_{ij} := (v, v_i)$
 - 14: $p := p - \beta_{ij}p_i$
 - 15: $v := v - \beta_{ij}v_i$
 - 16: **end for**
 - 17: $p_{j+1} := p/\|v\|$; $v_{j+1} := v/\|v\|$;
 - 18: **end for**
-

Remark. *nlTGCR (m) requires 2 function evaluations per step: one in Line 8 and the other in Line 11. In the situation when computing the Jacobian is inexpensive, then one can compute Jp in Line 11 as a matrix-vector product and this will reduce the number of function evaluations per step from 2 to 1. The inner loop in Line 12-16 corresponds to exploiting symmetry of Hessian. At a given step, nlTGCR (m) attempts to approximate a Newton step: $x_{j+1} = x_j + \delta$ where δ is an approximate solution to $J(x_j)\delta + F(x_j) = 0$.*

High-Level Clarification. At this point, one might ask the question: why not just use an inexact Newton method whereby the Jacobian system is solved with the *linear* GCR or TGCR method? This is where AA provides an interesting insight on some weaknesses of Newton-Krylov method. A Newton-Krylov method generates a Krylov subspace $\text{Span}\{r_0, Jr_0, \dots, J^k r_0\}$ at a current iterate

– say $K = x_0$ – (so $J \equiv J(x_0) \equiv DF(x_0)$) and tries to minimize $F(x_0 + \delta)$ where $\delta \in K$, by exploiting the linear model: $F(x_0 + \delta) \approx F(x_0) + J\delta$. If we generate a basis $V = [v_1, \dots, v_k]$ of K and express δ as $\delta = Vy$ then we would need to minimize $\|F(x_0) + JVy\|$ which is a small least-squares problem. One usually adds to this a global convergence strategy, e.g., a linesearch or a trust-region technique to produce the next iterate x_1 . The problem with this approach is this: *the approximate solution obtained after k steps of a Krylov subspace approach is based on the Jacobian at the initial point x_0* . The intermediate calculation is entirely linear and based on $J(x_0)$. It is not exploited in any way to produce intermediate (nonlinear) iterates which in turn could be used to produce more accurate information on some local Jacobian. In contrast, a method like AA (or in fact any of the secant or multiseant methods) will do just this, i.e., it will use information on the nonlinear mapping near the most recent approximation to produce the new iterate. This distinction is rather important although if the problem is nearly linear, then it could make little difference.

In nITGCR, we try to improve on the Newton-Krylov approach, since our starting point is TGCR which is generalized to nonlinear problems. We also take the viewpoint of improving on AA or multiseant methods by not relying on the approximation $F(x_{j+1}) - F(x_j) \approx J(x_{j+1} - x_j)$ mentioned above. This is achieved by adopting the projection viewpoint. Instead of minimizing $\|F(x_0) + JPy\|$ as in the inexact Newton mode, we would like to now minimize $\|F(x_k) + JPy\|$ where x_k is the most recent iterate. This initial idea leads to difficulty since there is not one but several J at previous points and a single one of them will not be satisfactory. Thus, we have a few directions p_i just like the differences Δx_i in Anderson, *but now each p_i will lead to a $J(x_i)p_i$ which – unlike in AA – is accurately computed and then saved*. This feature is what we believe makes a difference in the performance of the algorithm, although this is something that is rather difficult to prove theoretically. We leave it as future work. Overall, the method described in this paper mixes a number of ideas coming from different horizons. A further high-level discussion and detailed complexity analysis are provided in Appendix A.

Next, we analyze two possible versions of nITGCR in the next two sections. In what follows we assume that all the Jp_i 's are computed exactly.

3.2.1 Linearized update version

First, we consider a variant of Algorithm 2 which we call the “linearized update version” – whereby in Line 8 we update the residual by using the linear model, namely, we replace Line 8 by its linear analogue: 8a: $r_{j+1} = r_j - V_j y_j$. In addition, the matrix-vector product in Line 11 is performed with $J(x_0)$ instead of $J(x_{j+1})$. When F is linear, it turns out that y_j has only one nonzero component, namely the last one and this will yield the standard truncated GCR algorithm. Assume that we perform k steps of the algorithm to produce x_k , i.e., that Line 5 is replaced by “for $j = 0, 1, 2, \dots, k$ do”. Then the algorithm is exactly equivalent to an *inexact Newton method* in which GMRES (or GCR) is invoked to solve the Jacobian linear system [10]. Indeed, in this situation Lines 4-15 of Algorithm 1 and Lines 5-17 of Algorithm 2 are identical. In other words, in Lines 5-17, Algorithm 2 performs k steps of the GCR algorithm for approximately solving the linear systems $J(x_0)\delta = -F(x_0)$. Note that while the update is written in progressive form as $x_{j+1} = x_j + \alpha_j p_j$, *the right-hand side does not change during the algorithm and it is equal to $r_0 = -F(x_0)$* . In effect x_k is updated from x_0 by adding a vector from the span of P_k . See the related global convergence result shown in Theorem B.7 in the Appendix, for a version of this algorithm that includes a line-search. A weakness of this linear update version is that the Jacobian is not evaluated at the most recent update but at x_0 , which in practice is the iterate at each restart.

3.2.2 Non-linear update version with residual check

Next we consider the ‘nonlinear-update version’ as described in Algorithm 2. This version explicitly enforces the linear optimality condition of GCR, as represented by the Equation (13). In this section, we will analyze the convergence of nITGCR through the function $\phi(x) = \frac{1}{2}\|F(x)\|^2$.

In order to prove the global convergence of nITGCR, we need to make a small modification to Algorithm 2 because as implemented in Algorithm 2 P_j is fixed and the solution obtained at this step may not necessarily satisfy the following *residual check* condition which is often used in inexact Newton methods [15, 11, 20] to guarantee the global convergence:

$$\|F(x_j) + [J]P_j y\| \leq \eta \|F(x_j)\|, \quad (14)$$

where $\eta < 1$ is a parameter.

The residual norm on the left-hand side of (14) is readily available at no additional cost and this can help devise globally converging strategies, by monitoring to what extent (14) is satisfied. If (14) is not satisfied, we can either use a line-search technique 4 or restart the process and take the next iterate as the output of the fixed point iteration mapping H . When the residual check condition is implemented after Line 8 in Algorithm 2, we can prove the global convergence of nITGCR in the next theorem. Similar global strategies have also been proposed in [68, 49, 23, 64, 59, 43].

Theorem 3.2 (Global convergence of nITGCR with residual check). *Assume ϕ is twice differentiable and $F(x)$ is L -lipschitz. If the residual check is satisfied $\|J(x_n)P_n y_n + F(x_n)\| \leq \eta_n \|F(x_n)\|$ where $0 \leq \eta_n \leq \eta < 1$ and $J(x_n)$ is non-singular and its norm is bounded from above for all n , then $P_n y_n$ produced in line 7 of Algorithm 2 is a descent direction and the iterates x_n produced by Algorithm 2 will converge to the minimizer x^* :*

$$\lim_{n \rightarrow \infty} \phi(x_n) = \phi(x^*) = 0.$$

In the next theorem, we prove that nITGCR can achieve superlinear and quadratic convergence under mild conditions.

Theorem 3.3 (Superlinear and quadratic convergence of nITGCR). *With the same setting as Theorem 3.2. Assume $\nabla^2 \phi$ is L -lipschitz. Consider a sequence generated by Algorithm 2 such that residual check is satisfied $\|J(x_n)P_n y_n + F(x_n)\| \leq \eta_n \|F(x_n)\|$ where $0 \leq \eta_n \leq \eta < 1$. Moreover, if the following conditions hold*

$$\begin{aligned} \phi(x_n + P_n y_n) &\leq \phi(x_n) + \alpha \nabla \phi(x_n)^T P_n y_n \\ \phi(x_n + P_n y_n) &\geq \phi(x_n) + \beta \nabla \phi(x_n)^T P_n y_n \end{aligned}$$

for $\alpha < \frac{1}{2}$ and $\beta > \frac{1}{2}$. Then there exists a N_s such that $x_n \rightarrow x^*$ superlinearly for $n \geq N_s$ if $\eta_n \rightarrow 0$, as $n \rightarrow \infty$. Moreover, if $\eta_n = O(\|F(x_n)\|^2)$, the convergence is quadratic.

If the property of the function is bad (non-expansive/non-convex), it will be more difficult to satisfy the assumptions of Theorem 3.2 and 3.3. For example, in Theorem 3.2, the non-singularity and boundedness is required for $J(x)$. If the function does not satisfy the assumption, say, degenerate at a point, then the algorithm may not converge to a stationary point.

Remark. *This superlinear/quadratic convergence of nITGCR does not contradict with the linear convergence of TGCR shown in 3.1. 3.1 is obtained from the equivalence between TGCR and CG in that short-term recurrence holds for symmetric matrix. Like CG, TGCR can still have a superlinear convergence rate. In practice, the second stage of convergence of Krylov Space methods is typically well defined by the theoretical convergence bound with $\sqrt{\kappa(A)}$ but may be super-linear, depending on a distribution of the spectrum of the matrix A and the spectral distribution of the error.*

Finally, we analyze the convergence of nITGCR when the gradient F is subsampled. In the analysis, we make the following five assumptions.

Assumptions for stochastic setting **A₁** : The variance of subsampled gradients is uniformly bounded by a constant C , $\text{tr}(\text{Cov}(F(x))) \leq C^2, \forall x$.

A₂ : The eigenvalues of the Hessian matrix for any sample $|\mathcal{H}| = \beta$ is bounded from below and above in Loewner order $\mu_\beta I \preceq J(x, \mathcal{H}) \preceq L_\beta I$. Further more, we require there is uniform lower and upper bound for all submaps. That is, there exists $\hat{\mu}$ and \hat{L} such that $0 \leq \hat{\mu} \leq \mu_\beta$ and $L_\beta \leq \hat{L} < \infty, \forall \beta \in \mathbb{N}$. And the full Hessian is bounded below and above $\mu I \preceq J(x) \preceq L I, \forall x$.

A₃ : Hessian is M -Lipschitz, that is $\|J(x) - J(y)\| \leq M \|x - y\|, \forall x, y$

A₄ : The variance of subsampled Hessian is bounded by a constant σ .

$$\|\mathbb{E}_{\mathcal{H}}[(J(x; \mathcal{H}) - J(x))]\| \leq \sigma, \quad \forall x \quad (15)$$

A₅ : There exists a constant γ such that $\mathbb{E}[\|x_n - x^*\|^2] \leq \gamma (\mathbb{E}[\|x_n - x^*\|])^2$.

Theorem 3.4 (Convergence of stochastic version of nITGCR). *Assume $|\mathcal{H}_n| = \beta \geq \frac{16\sigma^2}{\mu}, \forall n$, residual check is satisfied for $\eta_n \leq \eta \leq \frac{1}{4L}$ and assumptions A1 – A5 hold. The iterates generated by the stochastic version Algorithm 2 converge to x^* if $\|x_k - x^*\| \leq \frac{\mu}{2M\gamma}$ and*

$$\mathbb{E}\|x_{n+1} - x^*\| \leq \frac{3}{4} \mathbb{E}\|x_n - x^*\|. \quad (16)$$

3.3 Connections with other methods

This section explores the connection between nITGCR with inexact Newton and AA. We provide the connection between nITGCR and quasi-Newton in A.1.

1) The inexact Newton viewpoint. Inexact Newton methods minimize $\|F(x_0) + J(x_0)P_j y\|$ over y by using some iterative method and enforcing a condition like

$$\|F(x_0) + J(x_0)P_j y\| \leq \eta \|F(x_0)\|$$

where $\eta < 1$ is a parameter, see, e.g., [15, 10, 11, 20]. In nITGCR, we are trying to solve a similar equation

$$F(x_j) + J(x_j)\delta = 0$$

by minimizing $\|F(x_j) + [J]P_j y\|$. We can prove the following properties of nITGCR.

Proposition 1. *As defined in Algorithm 2, $\delta_j = x_{j+1} - x_j = P_j y_j$ minimizes $\|F(x_j) + \delta\|$ over vectors of the form $\delta = V_j y$, where $y \in \mathbb{R}^{n_j}$ and $n_j = j - i_0 + 1$.*

As noted earlier, in the linear case, the vector y_j has only one nonzero component, namely the top one. In the general case, it is often observed that the other components are not zero but small. Let us then suppose that we replace the update in Lines 6-7 by the simpler form: $\mu_j = v_j^T r_j$, and $x_{j+1} = x_j + \mu_j p_j$. Then the direction $\delta_j = x_{j+1} - x_j$ is a descent direction for $\frac{1}{2}\|F(x)\|^2$.

Proposition 2. *Assume that $J(x_j)$ is nonsingular and that $\mu_j \equiv v_j^T r_j \neq 0$. Then $\delta_j = \mu_j p_j$ is a descent direction for the function $\frac{1}{2}\|F(x)\|^2$ at x_j .*

2) The quasi-Newton viewpoint. It is also possible to view the algorithm from the alternative angle of a quasi-Newton approach instead of inexact Newton. In nITGCR, the approximate inverse Jacobian G_j at step j is equal to

$$G_j = P_j V_j^T. \quad (17)$$

If we apply this to the vector v_j we get $G_j v_j = P_j V_j^T v_j = p_j = J(x_j)^{-1} v_j$. So G_j inverts $J(x_j)$ exactly when applied to v_j . It therefore satisfies the *secant* equation ([46, sec. 2.3])

$$G_j v_j = p_j. \quad (18)$$

This is equivalent to the secant condition $G_j \Delta f_j = \Delta x_j$ used in Broyden's second update method.

In addition, the update G_j satisfies the 'no-change' condition:

$$G_j q = 0 \quad \forall q \perp v_j. \quad (19)$$

The usual no-change condition for secant methods is of the form $(G_j - G_{j-m})q = 0$ for $q \perp \Delta f_j$ which in our case would be $(G_j - G_{j-m})q = 0$ for $q \perp v_j$. One can therefore consider that we are updating $G_{j-m} \equiv 0$. In this sense, we can prove the optimality of nITGCR(m).

Theorem 3.5 (Optimality of nITGCR). *The matrix G_j in (24) is the best approximation to the inverse Jacobian $J(x_j)^{-1}$ of $F(x)$ at x_j among all the matrices G whose range $\text{Range}(G) = \text{Span}\{V_j\}$ and satisfies the multisecant equation Equation $GV_j = P_j$. That is,*

$$G_j = \arg \min_{\{G \in \mathbb{R}^{d \times d} \mid \text{Range}(G) = \text{Span}\{V_j\}, GV_j = P_j\}} \|GJ(x_j) - I\|. \quad (20)$$

3) Comparison with Anderson Acceleration. Let us set $\beta = 0$ in Anderson Acceleration. Without loss of generality and in an effort to simplify notation we also assume that $i_0 = 1$ each time. According to (3-4), the j -th iterate becomes simply $x_{j+1} = x_j - \mathcal{F}_j \theta_j$ where θ_j is a vector that minimizes $\|F_j - \mathcal{F}_j \theta\|$. For nITGCR, we have $x_{j+1} = x_j + P_j y_j$ where y_j minimizes $\|F_j + V_j y\|$. So this is identical with Equation (3) when $\beta = 0$ in which $P_j \equiv \mathcal{X}_j$, and \mathcal{F}_j is replaced by V_j .

The most important relation for both cases is the multi-secant relation. For Anderson, with $G_{j-m} = 0$, the multi-secant matrix in (6) becomes

$$G_j = \mathcal{X}_j (\mathcal{F}_j^T \mathcal{F}_j)^{-1} \mathcal{F}_j^T \quad (21)$$

which can be easily seen to minimize $\|G\|_F$ for matrices G that satisfy the multisecant condition $G\mathcal{F}_j = \mathcal{X}_j$ and the no-change condition $G_j^T (G_j - G) = 0$. Therefore the two methods differ mainly

in the way in which the sets \mathcal{F}_j/V_j , and \mathcal{X}_j/P_j are defined. Let us use the more general notation V_j, P_j for the pair of subspaces.

In both cases, a vector v_j is related to the corresponding p_j by the fact that $v_j \approx J(x_j)p_j$. In the case of nITGCR this relation is explicitly enforced by a Frechet differentiation (Line 10)– before we perform an orthogonalization - which combines this vector with others – without changing the span of the new P_j (and also V_j).

In the case of AA, we have $v_j = \Delta F_{j-1} = F_j - F_{j-1}$ and the relation exploited is that

$$\begin{aligned} f_j &\approx F_{j-1} + J(x_{j-1})(x_j - x_{j-1}) \rightarrow \Delta f_{j-1} \\ &\approx J(x_{j-1})\Delta x_{j-1} \end{aligned} \quad (22)$$

However, the approximation $v_j \approx J(x_j)p_j$ in nITGCR is *more accurate*- because we use an additional function evaluation to explicitly obtain a more accurate approximation (ideally exact value) for $J(x_j)p_j$. In contrast when x_j and x_{j-1} are not close, then (22) can be a very rough approximation. This is a key difference between the two methods.

4 Experimental Results

This section compares our proposed algorithms TGCR and nITGCR to existing methods in the literature with the help of a few experiments. We first compare the convergence for linear problems and then for a softmax classification problem in the case where the gradients are either deterministic or stochastic. More experiments and experimental details are available in the Appendix C.

4.1 Linear Problems

We first compare the performance on linear equations $\mathbf{Ax} = \mathbf{b}$ with Conjugate Gradient [29], generalized minimal residual method (GMRES) [52] and Anderson Acceleration under different settings.

Linear Systems. The advantages of TGCR for linear systems are two-folds. **1:)** Theorem 3.1 shows that TGCR (1) is already optimal (equivalent to Conjugate Residual) when \mathbf{A} is symmetric positive definite. A larger table size is unnecessary while AA and GMRES require more past iterates to converge fast. It can be observed from Figure 1a and 1b that TGCR (1) requires much less memory and computation overhead to converge compared to GMRES and AA. It also has the same convergence behavior and similar running time as CG. **2:)** It is easy to show that TGCR can converge for indefinite systems while CG fails. Figure 1c verifies our point. This can be helpful when it is not known in advance if the system is indefinite. The numerical results shown in Figure 1 demonstrate the power of TGCR as a variant of Krylov subspace methods. Figure 1 clearly verifies the correctness of Theorem 3.1 that TGCR (1) is identical with TGCR (m) in exact arithmetic, which leads to big savings both in terms of memory and in computational costs. We include more experimental results in the Appendix C.2.

4.2 Nonlinear Problems: Softmax Classification

Next, we consider a softmax multi-class classification problem shown in (79) without regularization.

$$f = -\frac{1}{s} \sum_{i=1}^s \log \left(\frac{e^{w_{y_j}^T x^{(i)}}}{\sum_{j=1}^k e^{w_j^T x^{(i)}}} \right), \quad (23)$$

where s is the total number of sample, k is the total number of classes, $x^{(i)}$ is vector of all features of sample i , w_j is the weights for the j^{th} class, and y_j is the correct class for the i^{th} sample. We compare nITGCR with Gradient Descent (GD), Nonlinear Conjugate Gradient (NCG) [13], L-BFGS [38] and Anderson Acceleration using the MNIST dataset [16] and report results in Figure 2. Figure 2a and 2b plot the objective value vs. iteration number and wall-clock time respectively. It can be seen that nITGCR converges significantly faster than baselines even without a line-search strategy. In addition, for this convex and symmetric problem, it is not surprising to observe that nITGCR(1) exhibits a similar convergence rate with nITGCR(m), which saves even more memory and computation time.

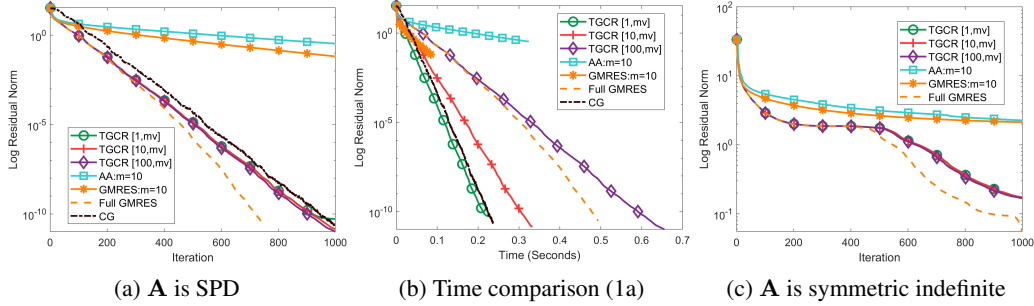


Figure 1: **Linear Systems $Ax = b, A \in \mathbb{R}^{1000 \times 1000}$** : **1a**: Comparison in terms of iteration, TGCR $[m, mv]$ means table size = m and moving window (no restart). **1b**: Comparison in terms of time for problem in 1a. **1c**: Indefinite System. It is well known that CG fails for indefinite systems. The gap between full GMRES and TGCR is due to the numerical issue. It can be concluded that TGCR is ideal for solving linear systems because of its nice convergence property (compared to CG and AA) as well as the memory-efficient design (compared to GMRES).

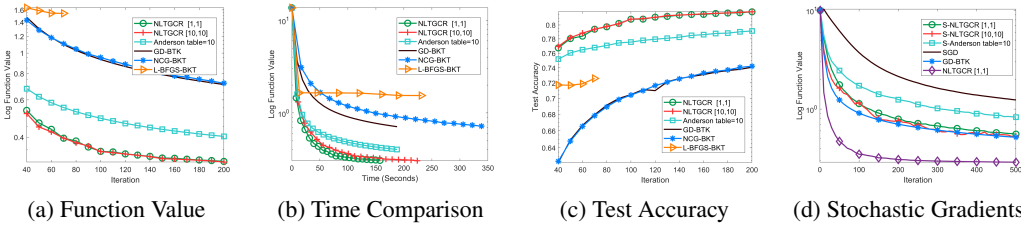


Figure 2: **Softmax Classification (MNIST Dataset)**: 2a: Function Value vs. Iterations; 2b: Function Value vs. Time; 2c: Test Accuracy vs. Iterations. 2d: Function Value vs. Iterations Stochastic gradients are calculated using a batch size of 500.

Figure 2c shows that nLTGCR greatly outperforms baselines by achieving high test accuracy in the very early stage of training. Figure 2d shows the effectiveness of our method in the stochastic setting. ‘S-’ stands for a stochastic version. We use a step size of 0.2 for SGD and a batch size (B) of 500 for all stochastic algorithms. It can be observed that nLTGCR(1) with a small batch size is comparable with the full batch GD with line-search, which confirms that TGCR takes advantage of symmetry in a very effective way even in the case of stochastic gradients.

4.3 Deep learning applications

We then evaluate nLTGCR on several widely-used deep learning applications using different frameworks. We run experiments on image classification using CNN [42] and ResNet [27], time series forecasting using LSTM [31], and node classification using GCN [35]. Due to space limitation, we provide full results in Appendix C.5. It shows that nLTGCR(1) outperforms baselines (SGD, Nesterov, and Adam) for the above DL experiments, highlighting its effectiveness in large-scale and stochastic non-convex optimization.

5 Conclusion

This paper describes an efficient nonlinear acceleration method that takes advantage of the symmetry of the Hessian. We studied the convergence properties of the proposed method and established a few connections with existing methods. The numerical results suggest that nLTGCR can be a competitive iterative algorithm from both theoretical and practical perspectives. We plan to conduct a more detailed theoretical and experimental investigation of the method for a nonconvex stochastic setting.

Social Impact. This work does not present any foreseeable societal consequence.

References

- [1] D. G. Anderson. Iterative procedures for non-linear integral equations. *Assoc. Comput. Mach.*, 12(547):547–560, 1965.
- [2] W. Azizian, I. Mitliagkas, S. Lacoste-Julien, and G. Gidel. A tight and unified analysis of gradient-based methods for a whole spectrum of games, 2019.
- [3] C. Blair. Problem complexity and method efficiency in optimization (a. s. nemirovsky and d. b. yudin). *SIAM Review*, 27(2):264–265, 1985.
- [4] R. Bollapragada, R. Byrd, and J. Nocedal. Exact and inexact subsampled newton methods for optimization, 2016.
- [5] R. Bollapragada, R. H. Byrd, and J. Nocedal. Exact and inexact subsampled newton methods for optimization. *IMA Journal of Numerical Analysis*, 39(2):545–578, 2019.
- [6] N. Boutet, R. Haelterman, and J. Degroote. Secant update version of quasi-newton psb with weighted multiseant equations. *Computational Optimization and Applications*, 75(2):441–466, 2020.
- [7] N. Boutet, R. Haelterman, and J. Degroote. Secant update generalized version of psb: a new approach. *Computational Optimization and Applications*, 78(3):953–982, 2021.
- [8] C. Brezinski and M. Redivo-Zaglia. The simplified topological ϵ -algorithms for accelerating sequences in a vector space. *SIAM Journal on Scientific Computing*, 36(5):A2227–A2247, 2014.
- [9] C. Brezinski, M. Redivo-Zaglia, and Y. Saad. Shanks sequence transformations and anderson acceleration. *SIAM Review*, 60(3):646–669, 2018.
- [10] P. N. Brown and Y. Saad. Hybrid Krylov methods for nonlinear systems of equations. *SIAM J. Sci. Stat. Comp.*, 11:450–481, 1990.
- [11] P. N. Brown and Y. Saad. Convergence theory of nonlinear Newton-Krylov algorithms. *SIAM Journal on Optimization*, 4:297–330, 1994.
- [12] S. Cabay and L. W. Jackson. A polynomial extrapolation method for finding limits and antilimits of vector sequences. *SIAM Journal on Numerical Analysis*, 13(5):734–752, 1976.
- [13] Y. H. Dai and Y. Yuan. A nonlinear conjugate gradient method with a strong global convergence property. *SIAM Journal on Optimization*, 10(1):177–182, 1999.
- [14] A. d’Aspremont, D. Scieur, and A. Taylor. Acceleration methods. *Foundations and Trends® in Optimization*, 5(1-2):1–245, 2021.
- [15] R. S. Dembo, S. C. Eisenstat, and T. Steihaug. Inexact Newton methods. *SIAM J. Numer. Anal.*, 18(2):400–408, 1982.
- [16] L. Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.
- [17] J. E. Dennis and J. J. Moré. Quasi-newton methods, motivation and theory. *SIAM Rev.*, 19:46–89, 1977.
- [18] M. Dereziński, J. Lacotte, M. Pilanci, and M. W. Mahoney. Newton-LESS: Sparsification without trade-offs for the sketched newton update. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021.
- [19] S. C. Eisenstat, H. C. Elman, and M. H. Schultz. Variational iterative methods for nonsymmetric systems of linear equations. *SIAM Journal on Numerical Analysis*, 20:345–357, 1983.
- [20] S. C. Eisenstat and H. F. Walker. Globally convergent inexact newton methods. *SIAM Journal on Optimization*, 4:393–422, 1994.
- [21] V. Eyert. A comparative study on methods for convergence acceleration of iterative vector sequences. *J. Computational Phys.*, 124:271–285, 1996.
- [22] M. Geist and B. Scherrer. Anderson acceleration for reinforcement learning, 2018.
- [23] D. Goldfarb, Y. Ren, and A. Bahamou. Practical quasi-newton methods for training deep neural networks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems, NIPS’20*, Red Hook, NY, USA, 2020. Curran Associates Inc.

- [24] W. W. Hager and H. Zhang. A survey of nonlinear conjugate gradient methods. 2005.
- [25] H. He, S. Zhao, Y. Xi, J. Ho, and Y. Saad. GDA-AM: ON THE EFFECTIVENESS OF SOLVING MIN-IMAX OPTIMIZATION VIA ANDERSON MIXING. In *International Conference on Learning Representations*, 2022.
- [26] H. He, S. Zhao, Y. Xi, J. C. Ho, and Y. Saad. Solve minimax optimization by anderson acceleration, 2021.
- [27] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.
- [28] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition, 2015.
- [29] M. R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of research of the National Bureau of Standards*, 49:409–435, 1952.
- [30] N. J. Higham and N. Strabi. Anderson acceleration of the alternating projections method for computing the nearest correlation matrix. to appear.
- [31] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 1997.
- [32] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997.
- [33] K. Jbilou and H. Sadok. Lu implementation of the modified minimal polynomial extrapolation method for solving linear and nonlinear systems. *IMA Journal of Numerical Analysis*, 19(4):549–561, 1999.
- [34] C. T. Kelley. Newton’s method in mixed precision. *SIAM Review*, 64(1):191–211, 2022.
- [35] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.
- [36] T. N. Kipf and M. Welling. Semi-Supervised Classification with Graph Convolutional Networks. *arXiv:1609.02907 [cs, stat]*, Feb. 2017. arXiv: 1609.02907.
- [37] A. Krizhevsky. Learning multiple layers of features from tiny images. 2009.
- [38] D. C. Liu and J. Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical Programming*, 45:503–528, 1989.
- [39] V. V. Mai and M. Johansson. Nonlinear acceleration of constrained optimization algorithms. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4903–4907, 2019.
- [40] J. Martens. Deep learning via hessian-free optimization. In *Proceedings of the 27th International Conference on International Conference on Machine Learning, ICML’10*, page 735–742, Madison, WI, USA, 2010. Omnipress.
- [41] P. Ni. *Anderson Acceleration of Fixed-point Iteration with Applications to Electronic Structure Computations*. PhD thesis, Worcester Polytechnic Institute, Worcester, Massachusetts, USA, 2009.
- [42] K. O’Shea and R. Nash. An introduction to convolutional neural networks, 2015.
- [43] W. Ouyang, Y. Liu, and A. Milzarek. Descent properties of an anderson accelerated gradient method with restarting. 06 2022.
- [44] M. L. Pasini, J. Yin, V. Reshniak, and M. K. Stoyanov. Anderson acceleration for distributed training of deep learning models. In *SoutheastCon 2022*, pages 289–295, 2022.
- [45] M. POWELL. A new algorithm for unconstrained optimization. In J. Rosen, O. Mangasarian, and K. Ritter, editors, *Nonlinear Programming*, pages 31–65. Academic Press, 1970.
- [46] H. ren Fang and Y. Saad. Two classes of multiseccant methods for nonlinear acceleration. *Numerical Linear Algebra with Applications*, 16(3):197–221, 2009.
- [47] F. Roosta, Y. Liu, P. Xu, and M. W. Mahoney. Newton-mr: Inexact newton method with minimum residual sub-problem solver, 2018.
- [48] F. Roosta-Khorasani and M. W. Mahoney. Sub-sampled newton methods i: Globally convergent algorithms, 2016.
- [49] C. W. Royer, M. O’Neill, and S. J. Wright. A newton-cg algorithm with complexity guarantees for smooth unconstrained optimization. *Math. Program.*, 180(1–2):451–488, mar 2020.

- [50] Y. Saad. *Numerical Methods for Large Eigenvalue Problems*. Halstead Press, New York, 1992.
- [51] Y. Saad. *Iterative Methods for Sparse Linear Systems, 2nd edition*. SIAM, Philadelphia, PA, 2003.
- [52] Y. Saad and M. H. Schultz. GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 7:856–869, 1986.
- [53] R. B. Schnabel. Quasi-newton methods using multiple secant equations. Technical Report CU-CS-247-83, Department of Computer Science, University of Colorado at Boulder, Boulder, CO, 1983.
- [54] D. Scieur, L. Liu, T. Pumar, and N. Boumal. Generalization of quasi-newton methods: Application to robust symmetric multisecond updates, 2020.
- [55] D. Scieur, L. Liu, T. Pumar, and N. Boumal. Generalization of quasi-newton methods: Application to robust symmetric multisecond updates. In A. Banerjee and K. Fukumizu, editors, *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pages 550–558. PMLR, 13–15 Apr 2021.
- [56] D. Scieur, E. Oyallon, A. d’Aspremont, and F. Bach. Online regularized nonlinear acceleration, 2018.
- [57] W. Shi, S. Song, H. Wu, Y. Hsu, C. Wu, and G. Huang. Regularized anderson acceleration for off-policy deep reinforcement learning. In *NeurIPS*, 2019.
- [58] D. A. Smith, W. F. Ford, and A. Sidi. Extrapolation methods for vector sequences. *SIAM Review*, 29(2):199–233, 1987.
- [59] H. D. Sterck and Y. He. On the asymptotic linear convergence speed of anderson acceleration, nesterov acceleration, and nonlinear gmres. *SIAM Journal on Scientific Computing*, 43(5):S21–S46, 2021.
- [60] 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 *NeurIPS*, 2021.
- [61] H. F. Walker and P. Ni. Anderson acceleration for fixed-point iterations. *SIAM J. Numer. Anal.*, 49(4):1715–1735, 2011.
- [62] F. Wei, C. Bao, and Y. Liu. Stochastic anderson mixing for nonconvex stochastic optimization, 2021.
- [63] F. Wei, C. Bao, and Y. Liu. A class of short-term recurrence anderson mixing methods and their applications. In *International Conference on Learning Representations*, 2022.
- [64] Y. Xie, R. H. Byrd, and J. Nocedal. Analysis of the bfgs method with errors. *SIAM Journal on Optimization*, 30(1):182–209, 2020.
- [65] Z. Yao, A. Gholami, S. Shen, M. Mustafa, K. Keutzer, and M. W. Mahoney. Adahessian: An adaptive second order optimizer for machine learning. 2020.
- [66] G. Zhang and Y. Yu. Convergence of gradient methods on bilinear zero-sum games. In *ICLR*, 2020.
- [67] J. Zhang, B. O’Donoghue, and S. Boyd. Globally convergent type-i anderson acceleration for non-smooth fixed-point iterations, 2018.
- [68] J. Zhang, B. O’Donoghue, and S. Boyd. Globally convergent type-i anderson acceleration for nonsmooth fixed-point iterations. *SIAM Journal on Optimization*, 30(4):3170–3197, 2020.

Appendix A Additional Discussion

A.1 High-Level Clarification

The method described in this paper mixes a number of ideas coming from different horizons. Some of the high-level discussion provided below is expanding further in later sections.

Linear case: TGCR. Our initial idea was motivated by considering the linear case, in an attempt to exploit Conjugate-gradient like methods for solving a linear system $Ax = b$. When A is symmetric, it is known that it is possible to minimize the objective function $f(x) = \|b - Ax\|_2^2$ on the k -th Krylov subspace $\text{Span}\{r_0, Ar_0, \dots, A^{k-1}r_0\}$ by a nice algorithm that uses a short-term recurrence. This algorithm, called the Conjugate Residual algorithm, is quite similar to the Conjugate Gradient but its residual vectors are conjugate (instead of being orthogonal) and its search directions are $A^T A$ -conjugate (instead of being A -conjugate). Its generalization to the nonsymmetric case, called the Generalized Conjugate Residual method, is easy to obtain by enforcing these two properties. Enforcing the $A^T A$ conjugacy of the p_i 's is the same as enforcing the orthogonality of the vectors Ap_i and this is expensive when we have many vectors. For this reason, practical versions of the algorithm are *truncated*, i.e., the orthogonalization is enforced against only a few previous directions. The result is the TGCR(m) algorithm (Algorithm 1) – which has been known since the 1980s. It is clear that *we expect that when the matrix A is nearly symmetric* TGCR(m) will perform nearly as well as the full version GCR - because when A is symmetric, taking $m = 1$ will yield full orthogonality of the Ap_i s (Theorem B.2).

Nonlinear case: Newton Krylov. Suppose now that we have to solve the nonlinear system $F(x) = 0$ (in optimization F is just the gradient of the objective function). At this point, we may ask the question: why not just use an inexact Newton method whereby the Jacobian system is solved with the *linear* GCR or TGCR method? This is where Anderson acceleration provides an interesting insight on some weaknesses of Newton-Krylov method. A Newton Krylov method generates a Krylov subspace $\text{Span}\{r_0, Jr_0, \dots, J^k r_0\}$ at a current iterate – say $K = x_0$ – (so $J \equiv J(x_0) \equiv DF(x_0)$) and tries to minimize $F(x_0 + \delta)$ where $\delta \in K$, by exploiting the linear model: $F(x_0 + \delta) \approx F(x_0) + J\delta$. If we generate a basis $V = [v_1, \dots, v_k]$ of K and express δ as $\delta = Vy$ then we would need to minimize $\|F(x_0) + JVy\|$ which is a small least-squares problem. One usually adds to this a global convergence strategies, e.g., a linesearch or a trust-region technique to produce the next iterate x_1 . The problem with this approach is this: *the approximate solution obtained after k steps of a Krylov subspace approach is based on the Jacobian at the initial point x_0* . The intermediate calculation is entirely linear and based on $J(x_0)$. It is not exploited in any way to produce intermediate (nonlinear) iterates which in turn could be used to produce more accurate information on some local Jacobian. In contrast, a method like Anderson acceleration (or in fact any of the secant or multiseccant methods) will do just this, i.e., it will tend to use information on the nonlinear mapping near the most recent approximation to produce the new iterate. This distinction is rather important although if the problem is nearly linear, then it could make little difference.

Nonlinear case: Anderson and nITGCR. Anderson acceleration can be viewed as a form of Quasi-Newton method whereby the approximate inverse Jacobian is updated at each step by using the collection of the previous iterates $x_k, x_{k-1}, \dots, x_{k-m+1}$ and the corresponding function values $F_k, F_{k-1}, \dots, F_{k-m+1}$. To be more accurate it uses the differences $\Delta x_j = x_{j+1} - x_j$ and the corresponding ΔF_j defined in the same way. Similarly to Newton-Krylov, it generates an approximation of the form $x_k + Py$ where P is a basis of the subspace spanned by the Δx_j 's. Notice how the update now is on x_k the latest point generated. The previous iterates are used to essentially provide information on the nonlinear mapping and its differential. This information is constantly updated using the most recent iterate. Note that this is informally stated: Anderson does not formally get an approximation to the Jacobian. It is based implicitly on exploiting the relation $F(x_{j+1}) - F(x_j) \approx J(x_{j+1} - x_j)$. Herein lies a problem that nITGCR aims at correcting: this relation is only vaguely verified. For example, *if we take J to be $J(x_j)$, the Jacobian at x_j , the resulting linear model is bound to be extremely inaccurate at the beginning of the iteration*.

In nITGCR, we try to improve on the Newton-Krylov approach, since our starting point is TGCR which is generalized to nonlinear problems. We also take the viewpoint of improving on Anderson Acceleration or multiseccant methods by not relying on the approximation $F(x_{j+1}) - F(x_j) \approx J(x_{j+1} - x_j)$ mentioned above. This is achieved by adopting the projection viewpoint. Instead of minimizing $\|F(x_0) + JP_y\|$ as in the inexact Newton mode, we would like to now minimize

$\|F(x_k) + JPy\|$ where x_k is the most recent iterate. This initial idea leads to a difficulty since there is not one J but several ones at previous points and a single one of them will not be satisfactory. Thus, we have a few directions p_i just like the differences Δx_i in Anderson, *but now each p_i will lead to a $J(x_i)p_i$ which - unlike in AA - is accurately computed and then saved.* This feature is what we believe makes a difference in the performance of the algorithm – although this is something that would be rather difficult to prove theoretically.

The Quasi-Newton viewpoint.. It is also possible to view the algorithm from the alternative angle of a Quasi-Newton approach instead of Inexact Newton. In this viewpoint, the inverse of the Jacobian is approximated progressively. Because it is the inverse Jacobian that is approximated, the method is akin to Broyden’s second update method.

In our case, the approximate inverse Jacobian G_j at step j is equal to

$$G_j = P_j V_j^T. \quad (24)$$

If we apply this to the vector v_j we get $G_j v_j = P_j V_j^T v_j = p_j = J(x_j)^{-1} v_j$. So G_j inverts $J(x_j)$ exactly when applied to v_j . It therefore satisfies the *secant* equation ([46, sec. 2.3])

$$G_j v_j = p_j. \quad (25)$$

This is the equivalent to the secant condition $G_j \Delta f_j = \Delta x_j$ used in Broyden’s second update method. Broyden type-II methods replace Newton’s iteration: $x_{j+1} = x_j - Df(x_j)^{-1} f_j$ with $x_{j+1} = x_j - G_j f_j$ where G_j approximates the inverse of the Jacobian $Df(x_j)$ at x_j by the update formula $G_{j+1} = G_j + (\Delta x_j - G_j \Delta f_j) v_j^T$ in which v_j is defined in different ways see [46] for details.

In addition, the update G_j satisfies the ‘no-change’ condition:

$$G_j q = 0 \quad \forall q \perp v_j. \quad (26)$$

The usual no-change condition for secant methods is of the form $(G_j - G_{j-m})q = 0$ for $q \perp \Delta f_j$ which in our case would be $(G_j - G_{j-m})q = 0$ for $q \perp v_j$. One can therefore consider that we are updating $G_{j-m} \equiv 0$.

It is also possible to find a link between the method proposed herein and the Anderson acceleration, by unraveling a relation with multi-secant methods. Note that equation (25) is satisfied for (at most) m previous instances of j , i.e., at step j we have (i_0 defined in the algorithm) $G_j v_i = p_i$ for $i = i_0, \dots, j$. In other words we can also write

$$G_j V_j = P_j. \quad (27)$$

This is similar to the multi-secant condition $G_j \mathcal{F}_j = \mathcal{X}_j$ of Equation (7) – see also equation (13) of [46] where \mathcal{F}_j and \mathcal{X}_j are defined in (2). In addition, we clearly also have a multi secant version of the no-change condition (26) seen above, which becomes:

$$G_j q = 0 \quad \forall \quad q \perp \text{Span}\{V_j\}. \quad (28)$$

This is similar to the no-change condition represented by eq. (15) of [46], which stipulates that $(G_j - G_{j-m})q = 0$ for all q orthogonal to the span of the subspace $\text{Span}\{\mathcal{F}_j\}$ mentioned above, provided we define $G_{j-m} = 0$.

A.2 Complexity Analysis

Assume that the iteration number is k and the model parameter size is d . The full memory AA stores all previous iterations, thus the additional memory is $2kd$. To reduce the memory overhead, the limited-memory (Truncated) AA(m) maintains the most recent m iterations while discarding the older historical information. In comparison, TGCR and NLTGCR only requires **the most recent iterate** to achieve optimal performance, thus the additional memory is $2d$. The reduced number of past iterates also saves the orthogonalization costs from TGCR and NLTGCR compared to AA(m). In TGCR and NLTGCR, only one orthogonalization is needed to be performed which costs $O(kd)$ while AA(m) requires $O(k^2d)$.

For TGCR(m), $(2d - 1)$ flops are performed in Line 5, $4d$ flops are performed in Lines 6-7 and $m(6d - 1)$ flops are performed in the for loop and $2d$ flops are performed in Line 15. If TGCR(m)

performs k iterations, the computational complexity is $((6m + 8)d - 1 - m)k$. Thus, TGCR costs $O(mdk)$. For symmetric problems, $m = 1$ is guaranteed to generate the same iterates as $m > 1$ and TGCR costs $O(dk)$.

Then we analyze the complexity of nITGCR(m). $m(2d - 1)$ flops are performed in Line 6, $2md$ flops are performed in Line 7, two evaluations of F are performed in Lines 8 and 11. The for loop costs $m(6d - 1)$ flops and $2d$ flops are performed in Line 15. When k iterations are performed, nITGCR costs $O(mdk)$ plus the costs of $2k$ function evaluations of F . When $m = 1$ is used in nonlinear problems, nITGCR costs $O(dk)$ plus the costs of $2k$ function evaluations of F .

A.3 The Frechet derivative

In vector analysis, derivatives provide local linear approximations. Frechet differentiation can be used to calculate directional derivatives of gradients. We use Frechet Differentiation to compute the directional derivative of a gradient mapping f at x in direction h , which is $v = J(x_{j+1})p$ in algorithm 2. We define Frechet derivative as follows,

Definition 1. Let $(S, \|\cdot\|)$ and $(T, \|\cdot\|)$ be two normed spaces and let X be an open set in $(S, \|\cdot\|)$.

A function $f : X \rightarrow T$ is Fréchet differentiable at x_0 , where $x_0 \in X$, if there exists a linear operator $(D_x f)(x_0) : X \rightarrow T$ such that

$$\lim_{h \rightarrow 0} \frac{\|f(x_0 + h) - f(x_0) - (D_x f)(x_0)(h)\|}{\|h\|} = 0$$

The operator $(D_x f)(x_0) : X \rightarrow T$ is referred to the Fréchet derivative at x_0 .

Appendix B Proofs

B.1 Optimality for Linear Problem

We can write the Generalized Conjugate residual formally as follows

Algorithm 3 GCR

- 1: **Input:** Matrix A , RHS b , initial x_0 .
 - 2: **Set** $p_0 = r_0 \equiv b - Ax_0$.
 - 3: **for** $j = 0, 1, 2, \dots$, **Until convergence do**
 - 4: $\alpha_j = (r_j, Ap_j) / (Ap_j, Ap_j)$
 - 5: $x_{j+1} = x_j + \alpha_j p_j$
 - 6: $r_{j+1} = r_j - \alpha_j Ap_j$
 - 7: $p_{j+1} = r_{j+1} - \sum_{i=1}^j \beta_{ij} p_i$ where $\beta_{ij} := (Ar_{j+1}, Ap_i) / (Ap_i, Ap_i)$
 - 8: **end for**
-

Theorem B.1 (Lemma 6.21 in [51]). *If $\{p_0, \dots, p_{n-1}\}$ is the basis of the Krylov space $\mathcal{K}_n(A, r_0)$ which are also $A^T A$ orthogonal. Then*

$$x_n = x_0 + \sum_{i=0}^{n-1} \frac{\langle r_0, Ap_i \rangle}{\langle Ap_i, Ap_i \rangle} p_i$$

minimizes the residual among all the iterates with form $x_0 + \mathcal{K}_n(A, r_0)$. Further more, we have

$$x_n = x_{n-1} + \frac{\langle r_{n-1}, Ap_{n-1} \rangle}{\langle Ap_{n-1}, Ap_{n-1} \rangle} p_{n-1}$$

Proof. We can write $x_n = x_0 + \sum_{i=0}^{n-1} \beta_i p_i$ and $r_n = r_0 - \sum_{i=0}^{n-1} \beta_i Ap_i$. Since x_n minimizes the residual, we know the following Petrov–Galerkin condition must hold

$$(r_n, Ap_j) = 0, \quad j = 0, \dots, n - 1$$

The $A^T A$ orthogonality gives us

$$\beta_i = \frac{\langle r_0, Ap_i \rangle}{\langle Ap_i, Ap_i \rangle}.$$

Similarly, we can write $x_n = x_{n-1} + \beta_{n-1}p_{n-1}$ and $r_n = r_{n-1} - \beta_{n-1}Ap_{n-1}$. Again, the optimality condition reads

$$\langle r_n, p_{n-1} \rangle = 0$$

which gives us

$$\frac{\langle r_{n-1}, Ap_{n-1} \rangle}{\langle Ap_{n-1}, Ap_{n-1} \rangle}$$

□

Theorem B.2. *When the coefficient matrix A is symmetric, TGCR(m) generates exactly the same iterates as TGCR(1) for any $m > 0$.*

Proof. Lines 8 to 14 in Algorithm 1 computes the new direction p_{j+1} – by ortho-normalizing the vector Ar_{j+1} against all previous Ap_i 's. In fact the loop of lines 9–13, implements a modified Gram-Schmidt procedure, which in exact arithmetic amounts simply to setting p_{j+1} to

$$\beta_{j+1,j}p_{j+1} := r_{j+1} - \sum_{i=i_0}^j \beta_{ij}p_i \quad \text{where} \quad \beta_{ij} = (Ar_{j+1}, Ap_i) \text{ for } i_0 \leq i \leq j. \quad (29)$$

In the above relation, $\beta_{j+1,j}$ is the scaling factor $\|v\|$ used to normalize p and v in Line 14. Then, $v_{j+1} \equiv Ap_{j+1}$ is computed accordingly as is reflected in lines 12 and 14. The update relation $Ap_{j+1} = Ar_{j+1} - \sum_{i=i_0, j} \beta_{ij}Ap_i$ (from Line 12) shows that $Ap_{j+1} \perp Ap_i$ for $i = i_0, \dots, j$. In addition, it can easily be shown that in this case ($m = \infty$) the residual vectors produced by the algorithm are A -conjugate in that $(r_{j+1}, Ar_i) = 0$ for $i \leq j$. Indeed, this requires a simple induction argument exploiting the equality:

$$(r_{j+1}, Ar_i) = (r_j - \alpha_j Ap_j, Ar_i) = (r_j, Ar_i) - \alpha_j (Ap_j, Ar_i)$$

and relation (29) which shows that $Ar_i = \sum \beta_{k,i-1} Ap_k$.

When A is symmetric, exploiting the relation $r_{i+1} = r_i - \alpha_i Ap_i$, we can see that the scalar β_{ij} in Line 11 of Algorithm 1 is

$$\beta_{ij} = (Ar_{j+1}, Ap_i) = \frac{1}{\alpha_i} (Ar_{j+1}, r_i - r_{i+1}) = \frac{1}{\alpha_i} (r_{j+1}, Ar_i - Ar_{i+1})$$

which is equal to zero for $i < j$. Therefore we need to orthogonalize Ar_{j+1} against vector Ap_j only in the loop of lines 9 to 13. This completes the proof. □

Theorem B.3. *Let $\hat{\mathbf{x}}_t$ be the approximate solution obtained at the t -th iteration of TGCR being applied to solve $\mathbf{A}\mathbf{x} = \mathbf{b}$, and denote the residual as $\mathbf{r}_t = \mathbf{b} - \mathbf{A}\hat{\mathbf{x}}_t$. Then, \mathbf{r}_t is of the form*

$$\mathbf{r}_t = f_t(\mathbf{A})\mathbf{r}_0, \quad (30)$$

where

$$\|\mathbf{r}_t\|_2 = \|f_t(\mathbf{A})\mathbf{r}_0\|_2 = \min_{f_t \in \mathcal{P}_t} \|f_t(\mathbf{A})\mathbf{r}_0\|_2, \quad (31)$$

where \mathcal{P}_p is the family of polynomials with degree p such that $f_p(0) = 1, \forall f_p \in \mathcal{P}_p$, which are usually called residual polynomials.

Theorem B.4 (Convergence of TGCR (Indefinite Case)). *Suppose \mathbf{A} is hermitian, invertible, and indefinite. Divide its eigenvalues into positive and negative sets Λ_+ and Λ_- , and define*

$$\kappa_+ = \frac{\max_{\lambda \in \Lambda_+} |\lambda|}{\min_{\lambda \in \Lambda_+} |\lambda|}, \quad \kappa_- = \frac{\max_{\lambda \in \Lambda_-} |\lambda|}{\min_{\lambda \in \Lambda_-} |\lambda|}$$

Then \mathbf{x}_m , the m th solution estimate of TGCR, satisfies

$$\frac{\|\mathbf{r}_m\|_2}{\|\mathbf{b}\|_2} \leq 2 \left(\frac{\sqrt{\kappa_+ \kappa_-} - 1}{\sqrt{\kappa_+ \kappa_-} + 1} \right)^{\lfloor m/2 \rfloor}$$

where $\lfloor m/2 \rfloor$ means to round $m/2$ down to the nearest integer.

Proof. When A is hermitian indefinite, an estimate on the min-max approximation

$$\frac{\|\mathbf{r}_m\|_2}{\|\mathbf{b}\|_2} \leq \min_{p \in \mathcal{P}_m} \max_k |p(\lambda_k)| \quad (32)$$

that represents the worst-case TGCR convergence behavior, can be obtained by replacing the discrete set of the eigenvalues by the union of two intervals containing all of them and excluding the origin, say Λ_+ and Λ_- . Then the classical bound for the min-max value can be used to obtain an estimate for the convergence of the residual [50]

$$\begin{aligned} \min_{p \in \mathcal{P}_m} \max_k |p(\lambda_k)| &\leq \min_{p \in \mathcal{P}_m} \max_{z \in \Lambda_+ \cup \Lambda_-} |p(z)| \\ &\leq 2 \left(\frac{\sqrt{\kappa_+ \kappa_-} - 1}{\sqrt{\kappa_+ \kappa_-} + 1} \right)^{\lfloor m/2 \rfloor}, \end{aligned}$$

where $\lfloor m/2 \rfloor$ denotes the integer part of $m/2$. \square

The optimality of TGCR(m) is proved in Theorem B.5.

Theorem B.5. *Let r be the residual generated by the basic TGCR (m), the following relations hold:*

1. $\text{Span}\{p_0, \dots, p_{m-1}\} = \text{Span}\{r_0, \dots, A^{m-1}r_0\} \equiv K_m(r_0, A)$
2. If $m \geq j$, the set of vectors $\{Ap_i\}_{i=1:j}$ is orthonormal.
3. More generally: $(Ap_i, Ap_j) = \delta_{ij}$ for $|i - j| \leq m - 1$
4. If $m \geq j$, then $\|b - Ax_j\| = \min\{\|b - Ax\| \mid x \in x_0 + K_m(r_0, A)\}$

Proposition 3. *Assume that $J(x_j)$ is nonsingular and that $\mu_j \equiv v_j^T r_j \neq 0$. Then $\delta_j = \mu_j p_j$ is a descent direction for the function $\phi(x) = \frac{1}{2} \|F(x)\|^2$ at x_j .*

Proof. It is known [11] that the gradient of $\phi(x)$ at x is $\nabla\phi(x) = J(x)^T F(x)$. In order for p_j to be a descent direction at x_j it is sufficient that the inner product of p_j and $\nabla\phi(x_j)$ is negative. Consider this inner product

$$(\nabla\phi(x_j), \mu_j p_j) = \mu_j (J(x_j)^T F(x_j), p_j) = \mu_j (F(x_j), J(x_j) p_j) = \mu_j (-r_j, v_j) = -\mu_j^2 < 0. \quad (33)$$

which proves the result. \square

B.2 Optimality from Quasi-Newton Viewpoint

Theorem B.6 (Optimality of nltgcr(m) from Quasi-Newton Viewpoint). *The matrix G_j is the best approximation to the inverse Jacobi $J(x_i)^{-1}$ of $F(x)$ at x_i among all the matrices G whose range $\text{Range}(G) = \text{Span}\{V_j\}$. That is,*

$$G_j = \arg \min_{\{G \in \mathbb{R}^{d \times d} \mid GV_j = P_j\}} \|GJ(x_i) - I\|. \quad (34)$$

Proof. Assume G is an arbitrary matrix satisfying the multisection condition $GV_j = P_j$. We have $(G_j - G)G_j^T = 0$ and $\text{Range}(G) = V_j$. This can be derived as follows

$$0 = P_j(P_j^T - P_j^T) = P_j V_j^T (G^T - G_j^T) = G_j (G^T - G_j^T).$$

We also have $(G_j - G)V_j = 0$. Then set $\Delta = G - G_j$, we have

$$\begin{aligned} \|GJ(x_i) - I\| &= \|(G_j + \Delta)J(x_i) - I\| \\ &= \|G_j J(x_i) - I\| + \|\Delta J(x_i)\| + 2 \text{Trace}((G_j J(x_i) - I)^T \Delta J(x_i)) \\ &\geq \|G_j J(x_i) - I\| + 2 \text{Trace}((G_j J(x_i) - I)^T \Delta J(x_i)). \end{aligned}$$

Then we prove $\text{Trace}((G_j J(x_i) - I)^T \Delta J(x_i)) = 0$. In order to prove this, we compute the trace explicitly. We will denote the natural basis in \mathbb{R}^d by $\{e_l\}_{l=1}^d$ and l -th column of $J(x_i)$ by J_l .

$$\begin{aligned} \text{Trace}((G_j J(x_i) - I)^T \Delta J(x_i)) &= \sum_{l=1}^d e_l^T ((G_j J(x_i) - I)^T \Delta J(x_i)) e_l \\ &= \sum_{l=1}^d (J_l^T G_j^T - e_l^T) \Delta J_l \end{aligned}$$

Recall we have $\text{Range}(G) = \text{Range}(G_j) = \text{Span}\{V_j\}$ and $GV_j = G_j V_j = P_j$, so

$$\left\{ \begin{array}{ll} \Delta J_l = 0 & J_l \in V_j \\ J_l^T G_j^T = 0, G J_l = 0 & J_l \in V_j^\perp \end{array} \right\}$$

□

B.3 Convergence Analysis

Firstly, we will show the global convergence of the Algorithm 2 from inexact Newton perspective. Usually, some global strategies like line search or residue check are required for inexact Newton method to converge for $\phi(x)$. In [11], authors showed with the general line search algorithm 4,

Algorithm 4 Linesearch Algorithm

- 1: $\beta = \max\{1, \epsilon^* \frac{|\nabla \phi(x_n)^T p_n|}{\|p_n\|^2}\}$.
 - 2: If $\phi(x_n + \beta p_n) \leq \phi(x_n) + \alpha \beta \nabla \phi(x_n)$, then set $\beta_n = \beta$ and exit. Else:
 - 3: Shrink β to be $\beta \in [\theta_{\min} \beta, \theta_{\max} \beta]$ where $0 < \theta_{\min} \leq \theta_{\max} < 1$. Go back to Step. 2.
-

inexact Newton-Krylov method can converge globally under some mild conditions.

Theorem B.7 (Global Convergence from Algorithm 2 with linearized update and line search). *Assume ϕ is continuously differentiable and $F(x)$ is L -lipschitz. Furthre more, the residual check is satisfied $\|J(x_n)P_n y_n + F(x_n)\| \leq \eta \|\nabla f(x_n)\|$ where $0 \leq \eta_n \leq \eta < 1$. If $J(x_n)$ is nonsingular and its norm is bounded from above for all n , then $P_n y_n$ produced in line 7 of Algorithm 2 is a descent direction and the iterates x_n produced by Algorithm 2 with linearized update and line search in Algorithm 4 will converge to the minimizer:*

$$\lim_{n \rightarrow \infty} \phi(x_n) = 0$$

Proof. Since Algorithm 2 with linearized update is equivalent to inexact Newton Krylov method with TGCR as the solver for the Jacobian system $J(x_n)p_n = -F(x_n)$, the theorem is just a result of Theorem B.8. □

Theorem B.8 ([11]). *Assume ϕ is continuously differentiable and $F(x)$ is L -Lipschitz and let p_n be such that $\|F(x_n) + J(x_n)p_n\|_2 \leq \eta_n \|F(x_n)\|_2$ for each $\eta_n \leq \eta < 1$. Further more, let the next iterate be decided by Algorithm 4 and $J(x_n)$ is nonsingular and bounded from above for all n . Then*

$$\lim_{n \rightarrow \infty} \phi(x_n) = 0.$$

The proof of this theorem depends on the following lemma in [11],

Lemma B.9 (Lemma 3.8 of [11]). *Assume ϕ is differentiable and $\nabla \phi$ is L -lipschitz. Let $0 < \alpha < 1$ and p_n denote a descent direction. Then the iterates $x_{n+1} = x_n + \beta p_n$ in Algorithm 4 will generated in finite backtracking steps and β_n satisfies*

$$\beta_n \|p_n\|_2 \geq -\frac{\nabla \phi^T p_n}{\|p_n\|} \min\left(\epsilon^*, \frac{1 - \alpha}{L} \theta_{\min}\right).$$

Theorem B.10 (Global convergence of nITGCR with residual check). *Assume ϕ is twice differentiable and $F(x)$ is L -lipschitz. If the residual check is satisfied $\|J(x_n)P_n y_n + F(x_n)\| \leq \eta_n \|F(x_n)\|$ where $0 \leq \eta_n \leq \eta < 1$ and $J(x_n)$ is non-singular and the norm of its inverse is bounded from above*

for all n , then $P_n y_n$ produced in line 7 of Algorithm 2 is a descent direction and the iterates x_n produced by Algorithm 4 will converge to the minimizer x^* :

$$\lim_{n \rightarrow \infty} \phi(x_n) = \phi(x^*) = 0.$$

Proof. Denote $P_n y_n$ by p_n , $J(x_n)p_n + F(x_n) = r_n$. Since $\nabla\phi(x_n) = J(x_n)^T F(x_n)$ and $\|r_n\| \leq \eta_n \|F(x_n)\|$, we have $\nabla\phi(x_n)^T p_n = F(x_n)^T r_n - F(x_n)^T F(x_n) \leq (\eta - 1)\nabla\|F\|^2 = -2(1 - \eta)\phi$ which implies p_n is a descent direction. To see the second part of the theorem, we have

$$\phi(x_n + \beta_n \alpha p_n) \leq \phi(x_n) + \beta_n \alpha \nabla\phi(x_n)^T p_n \quad (35)$$

$$\leq \phi(x_n) - 2\beta_n \alpha (1 - \eta)\phi(x_n) = [1 - 2\beta_n \alpha (1 - \eta)]\phi(x_n). \quad (36)$$

Denote $\min\left(\epsilon^*, \frac{1-\alpha}{L}\theta_{\min}\right)$ by C then,

$$-\beta_n \|p_n\| \leq C \frac{\nabla\phi(x_n)^T p_n}{\|p_n\|_2}.$$

Inserting it back to Inequality (27), we have

$$\phi(x_{n+1}) \leq \left(1 + 2\alpha(1 - \eta)C \frac{\nabla\phi(x_n)^T p_n}{\|p_n\|_2^2}\right)\phi(x_n) \quad (37)$$

Denote $2\alpha(1 - \eta)C$ by λ and $\frac{\nabla\phi(x_n)^T p_n}{\|p_n\|_2^2}$ by t_n , then $\phi(x_{n+1}) \leq (1 + \lambda t_n)\phi(x_n)$. Since $\phi(x_n)$ is bounded from below and non-increasing by the inequality. It must converge to a finite limit ϕ^* . If $\phi^* = 0$, we're done. Otherwise, dividing the Inequality 37 by $\phi(x_n)$ on both sides, we have

$$\frac{\phi(x_{n+1})}{\phi(x_n)} \leq (1 + \lambda t_n) \rightarrow 1, \quad \text{as } n \rightarrow \infty. \quad (38)$$

We also know $1 + \lambda t_n \leq 1$. Therefore, $t_n \rightarrow 0$, as $n \rightarrow \infty$. In the above discussion, we showed $2(1 - \eta)\phi(x_n) \leq |t_n| \|p_n\|_2^2$ which implies $\|p_n\| \rightarrow \infty$. Recall $p_n = J(x_n)^{-1}(r_n - F(x_n))$, we must have $\|p_n\|$ bounded. This contradicts with the fact $\|p_n\| \rightarrow \infty$. Therefore, $\phi^* = 0$ \square

To proceed to the superlinear and quadratic convergence results, we need the following lemma from [20]

Lemma B.11. Assume F is continuously differentiable, $\{x_k\}$ is a sequence such that $F(x_k) \rightarrow 0$, and for each k ,

$$\|F(x_{k+1})\| \leq \|F(x_k)\| \quad \text{and} \quad \|F(x_k) + J(x_k)p_k\| \leq \eta \|F(x_k)\| \quad (39)$$

where $p_k = x_{k+1} - x_k$ and $\eta > 0$ is independent of k . If x_* is a limit point of $\{x_k\}$ such that $J(x_*)$ is nonsingular, then $F(x_*) = 0$ and $x_k \rightarrow x_*$. In this lemma, we don't require $\eta < 1$.

Theorem B.12 (Superlinear and quadratic convergence of nITGCR). With the same setting as Theorem B.10. Assume both $\nabla\phi$ and $\nabla^2\phi$ are L -Lipschitz. Consider a sequence generated by Algorithm 2 such that residual check is satisfied $\|J(x_n)P_n y_n + F(x_n)\| \leq \eta_n \|F(x_n)\|$ where $0 \leq \eta_n \leq \eta < 1$. Moreover, if the following conditions hold

$$\phi(x_n + P_n y_n) \leq \phi(x_n) + \alpha \nabla\phi(x_n)^T P_n y_n \quad (40)$$

$$\phi(x_n + P_n y_n) \geq \phi(x_n) + \beta \nabla\phi(x_n)^T P_n y_n \quad (41)$$

for $\alpha < \frac{1}{2}$ and $\beta > \frac{1}{2}$. If $x_n \rightarrow x_*$ with $J(x_*)$ nonsingular, then $F(x_*) = 0$. Moreover, there exists N_s such that $x_n \rightarrow x_*$ superlinearly for $n \geq N_s$ if $\eta_n \rightarrow 0$, as $n \rightarrow \infty$. Furthermore, if $\eta_n = O(\|F(x_n)\|^2)$, the convergence is quadratic.

Proof. In the proof, we denote $P_n y_n$ by p_n for convenience and utilize the proof of Theorem 3.15 in [11]. According to assumptions, $x_n \rightarrow x_*$ with $J(x_*)$ nonsingular, then $J(x_n)$ is nonsingular for $n > n_J$ for some large enough n_J . Next, if $F(x_n) = 0$ for some $n \geq n_J$, then residual check condition will imply $p_n = 0$ which means $x_m = x_n$ for all $m \geq n$. Then the results hold

automatically because the sequence converges in finite steps. Therefore, we can assume $J(x_n)$ is nonsingular and $F(x_n)$ is nonzero for all n .

The residual check condition implies p_n is a descent direction according to Lemma B.9. That is, $\nabla\phi^\top p_n < 0$. Then we can show

$$\lim_{n \rightarrow \infty} \frac{\nabla\phi_n^\top p_n}{\|p_n\|} = 0.$$

To show this notice that according to 40, the following inequality holds

$$\phi_n - \phi_{n+1} \geq -\alpha \nabla\phi(x_n + p_n)^\top (x_{n+1} - x_n) = \|p_n\| \frac{\nabla\phi_n^\top p_n}{\|p_n\|}$$

Since ϕ_n is monotone decreasing, thus $\|p_n\| \frac{\nabla\phi_n^\top p_n}{\|p_n\|} \rightarrow 0$ as $n \rightarrow \infty$. To show $\lim_{n \rightarrow \infty} \frac{\nabla\phi_n^\top p_n}{\|p_n\|} \rightarrow 0$. We also need to apply 41. Firstly, according to mean value theorem, there exists a $\lambda \in (0, 1)$ such that

$$\phi_{n+1} - \phi_n = \nabla\phi(x_n + \lambda p_n)^\top p_n. \quad (42)$$

According to 41,

$$\phi_{n+1} - \phi_n = \nabla\phi(x_n + \lambda p_n)^\top p_n \geq \beta \nabla\phi_n^\top p_n. \quad (43)$$

This yields

$$[\nabla\phi(x_n + \lambda p_n) - \nabla\phi(x_n)]^\top p_n \geq (\beta - 1) \nabla\phi_n^\top p_n > 0.$$

According to Cauchy-Schwartz inequality,

$$(\beta - 1) \frac{\nabla\phi_n^\top p_n}{\|p_n\|} \|p_n\| \leq \|p_n\| \|\nabla\phi(x_n + \lambda p_n) - \nabla\phi(x_n)\| \leq L\lambda \|p_n\|^2 \quad (44)$$

Therefore,

$$\|p_n\| \geq \frac{(\beta - 1) \nabla\phi_n^\top p_n}{L\lambda \|p_n\|} > 0. \quad (45)$$

which means we can draw the conclusion that $\|p_n\| \frac{\nabla\phi_n^\top p_n}{\|p_n\|} \rightarrow 0$ implies $\frac{\nabla\phi_n^\top p_n}{\|p_n\|} \rightarrow 0$. If $x_n \rightarrow x_*$ with $J(x_*)$ nonsingular, then by Lemma B.11, we know $F(x_*) = 0$. According to the definition,

$$p_n = -J_n^{-1} F_n + J_n^{-1} (F_n + J_n p_n). \quad (46)$$

This implies

$$\|p_n\| \leq \|J_n^{-1}\| \|F_n\| + \|J_n^{-1}\| \|F_n + J_n p_n\| \leq (1 + \eta) \|J_n^{-1}\| \|F_n\|. \quad (47)$$

We know $\|F_n\| \rightarrow 0$ as $x_n \rightarrow x_*$. The above inequality implies that $\|p_n\| \rightarrow 0$ as $x_n \rightarrow x_*$ since J_n is nonsingular. Denote the residual $F_n + J_n p_n$ by r_n . Then $\|r_n\| \leq \eta \|F_n\|$ and $p_n = J_n^{-1} (r_n - F_n)$. Therefore,

$$\nabla\phi_n^\top = (J_n^\top F_n)^\top J_n^{-1} (r_n - F_n) = F^\top r - F^\top F. \quad (48)$$

This implies

$$\frac{|\nabla\phi_n^\top p_n|}{\|p_n\|} = \frac{|F^\top r - F^\top F|}{\|J_n^{-1} (r_n - F_n)\|} \geq \frac{|F^\top F| - |F^\top r|}{\|J_n^{-1} (r_n - F_n)\|}. \quad (49)$$

Since $\|r_n\| \leq \|F_n\|$ implies $|F^\top r| \leq \eta \|F_n\|^2$, we have

$$|F^\top F| - |F^\top r| \geq (1 - \eta) \|F_n\|^2. \quad (50)$$

Moreover,

$$\|J_n^{-1} (r_n - F_n)\| \leq \|J_n^{-1}\| \|F_n\| + \|J_n^{-1} r\|_2 \leq (1 + \eta) \|J_n^{-1}\| \|F_n\|. \quad (51)$$

Finally, we have

$$\frac{|\nabla\phi_n^\top p_n|}{\|p_n\|} \geq \frac{(1 - \eta) \|F_n\|^2}{(1 + \eta) \|J_n^{-1}\| \|F_n\|} = \frac{(1 - \eta) \|F_n\|}{(1 + \eta) \|J_n^{-1}\|}. \quad (52)$$

Using $\|\nabla\phi_n\| = \|J_n^\top F_n\| \leq \|J_n\|\|F_n\|$, we have

$$\frac{|\nabla\phi_n^\top p_n|}{\|\nabla\phi_n\|\|p_n\|} \geq \frac{(1-\eta)}{(1+\eta)M_n}, \quad (53)$$

where $M_n = \text{cond}_2(J_n)$. Therefore, we have

$$\frac{-\nabla\phi_n^\top p_n}{\|p_n\|} \geq \frac{(1-\eta)}{(1+\eta)M_n} \|\nabla\phi_n\| \geq \frac{(1-\eta)}{(1+\eta)M_n} \|J_n\|^{-1} \|F_n\|, \quad (54)$$

This yields,

$$\|F_n\| \leq \frac{(1+\eta)M_n}{(1-\eta)} \|J_n\| \frac{-\nabla\phi_n^\top p_n}{\|p_n\|}. \quad (55)$$

Hence,

$$\|F_n\|\|p_n\| \leq \frac{(1+\eta)M_n}{(1-\eta)} \|J_n\| (-\nabla\phi_n^\top p_n) = -a_n \nabla\phi_n^\top p_n, \quad (56)$$

where $a_n = \frac{(1+\eta)M_n}{(1-\eta)} \|J_n\|$. Combining 47, we have

$$\|p_n\|^2 \leq \frac{(1+\eta)^2 M_n^2}{(1-\eta)^2} (-\nabla\phi_n^\top p_n) = -b_n \nabla\phi_n^\top p_n, \quad (57)$$

where $b_n = \frac{(1+\eta)^2 M_n^2}{(1-\eta)^2}$. Next we show the convergence of the algorithm with the aid of the second order Taylor expansion. Notice

$$\nabla\phi(x) = J^\top F(x) = [\nabla F_1(x) \dots \nabla F_n(x)] \begin{bmatrix} F_1(x) \\ \vdots \\ F_n(x) \end{bmatrix} \quad (58)$$

The Hessian can be computed as follows

$$\nabla^2\phi(x) = J^\top J + \sum_{i=1}^n \nabla^2 F_i(x) F_i(x) = J^\top J + G(x), \quad (59)$$

where $\|G(x)\| = \|\sum_{i=1}^n \nabla^2 F_i(x) F_i(x)\| \rightarrow 0$ as $x_n \rightarrow x_*$ since $F(x_*) = 0$. using second order Taylor expansion, we have

$$\phi_{n+1} - \phi_n - \frac{1}{2} \nabla\phi_n^\top p_n = \frac{1}{2} (\nabla\phi_n + \nabla^2\phi(\bar{x})p_n)^\top p_n. \quad (60)$$

where $\bar{x} = \gamma x_n + (1-\gamma)x_{n+1}$ for some $\gamma \in (0, 1)$. Then we can have

$$\begin{aligned} |\phi_{n+1} - \phi_n - \frac{1}{2} \nabla\phi_n^\top p_n| &= \frac{1}{2} |(\nabla\phi_n + \nabla^2\phi(\bar{x})p_n)^\top p_n| \\ &= \frac{1}{2} |(\nabla\phi_n + \nabla^2\phi_n p_n)^\top p_n + p_n^\top (\nabla^2\phi(\bar{x}) - \nabla^2\phi_n) p_n| \\ &\leq \frac{1}{2} (\|J_n^\top (F_n + J_n p_n)\| \|p_n\| + \|G_n\| \|p_n\|^2 + L \|p_n\| \|p_n\|^2) \\ &\leq (\eta \|J_n\| \|F_n\| \|p_n\| + (\|G_n\| + L \|p_n\|) \|p_n\|^2) \\ &\leq -\frac{1}{2} (a_n \eta_n \|J_n\| + b_n (\|G_n\| + L \|p_n\|)) \nabla\phi_n^\top p_n \\ &= -\frac{1}{2} \epsilon_n \nabla\phi_n^\top p_n, \end{aligned} \quad (61)$$

where $\epsilon_n = a_n \eta_n \|J_n\| + b_n (\|G_n\| + L \|p_n\|)$. Therefore,

$$\frac{1}{2} (1 + \epsilon_n) \nabla\phi_n^\top p_n \leq \phi_{n+1} - \phi_n \leq \frac{1}{2} (1 - \epsilon_n) \nabla\phi_n^\top p_n \quad (62)$$

Notice that $\|J_n\|$, a_n and b_n are all bounded from above and η_n , $\|S_n\|$ and $\|p_n\|$ all converges to 0 as $x_n \rightarrow x_*$. Therefore $\epsilon_n \rightarrow 0$ as $x_n \rightarrow x_*$. And choose lager enough N such that for all $n \geq N$ the following holds

$$\epsilon_n \leq \min\{1 - 2\alpha, 2\beta - 1\}. \quad (63)$$

Then for all $n \geq N$, the Goldsetin-Armijo condition is satisfied,

$$\beta\phi_n^\top p_n \leq \phi_{n+1} - \phi_n \leq \alpha\phi_n^\top p_n. \quad (64)$$

We then finish the proof following Theorem 3.3 in [15]. It's easy to see

$$J(x_*)(x_{k+1} - x_*) = [I + J_*(J_k^{-1} - J_*^{-1})](r_k + [J_k - J_*)(x_k - x_*) - [F_k - F_* - J_*(x_k - x_*)]) \quad (65)$$

Taking norm yields

$$\begin{aligned} \|x_{k+1} - x_*\| &\leq [\|J_*^{-1}\| + \|J_*\|\|J_k^{-1} - J_*^{-1}\|][\|r_k\| + \|J_k - J_*\|\|x_k - x_*\| + \\ &\quad \|F_k - F_* - J_*(x_k - x_*)\|] \\ &= [\|J_*^{-1}\| + \|J_*\|\|J_k^{-1} - J_*^{-1}\|][\|r_k\| + \|J_k - J_*\|\|x_k - x_*\| \\ &\quad + \|F_k - F_* - J_*(x_k - x_*)\|] \\ &= [\|J_*^{-1}\| + o(1)][o(F_k) + o(1)\|x_k - x_*\| + o(\|x_k - x_*\|)] \end{aligned} \quad (66)$$

Therefore,

$$\|x_{k+1} - x_*\| = o(F_k) + o(1)\|x_k - x_*\| + o(\|x_k - x_*\|), \quad k \rightarrow \infty. \quad (67)$$

where we used the fact that for sufficient small $\|y - x_*\|$

$$\frac{1}{\alpha}\|y - x_*\| \leq \|F(y)\| \leq \alpha\|y - x_*\|. \quad (68)$$

which is Lemma 3.1 in [15]. Similarly, to show quadratic convergence, juts notice

$$\|F(y) - F(x_*) - F'(x_*)(y - x_*)\| \leq L'\|y - x_*\|^2 \quad (69)$$

for some constant L' and sufficient small $\|y - x_*\|$. For more details, check Lemma 3.2 in [15]. \square

B.4 Stochastic nITGCR

Denote the noisy gradient by $F(x; \xi_{\mathcal{G}})$ and the noisy evaluation of Hessian along a vector p by $J(x; \xi_{\mathcal{H}})p$. The subsample exact Newton algorithm is defined in Algorithm 5. At k -th iteration, we uniformly subsample $\mathcal{G}_k, \mathcal{H}_k$ from full sample set to estimate the noisy gradient and Hessian, so both of them are unbiased. Before we start the theoretical analysis, we need to make some assumptions

Algorithm 5 subsample Exact Newton

- 1: **for** $i = 1, \dots, k$ **do**
 - 2: Estimate $F(x_i; \mathcal{G}_i)$ and $J(x_i; \mathcal{H}_i)$
 - 3: $x_{i+1} \leftarrow x_i - s_i J^{-1}(x_i; \mathcal{H}_i) F(x_i; \mathcal{G}_i)$
 - 4: **end for**
-

which are usual in stochastic setting.

Assumptions for stochastic setting

E_1 The eigenvalues of Hessian matrix for any sample $|\mathcal{H}| = \beta$ is bounded form below and above in Loewner order

$$\mu_\beta I \preceq J(x, \mathcal{H}) \preceq L_\beta I. \quad (70)$$

Further more, we require there is uniform lower and upper bound for all subsmamples. That is, there exists $\hat{\mu}$ and \hat{L} such that

$$0 \leq \hat{\mu} \leq \mu_\beta \quad \text{and} \quad L_\beta \leq \hat{L} < \infty, \quad \forall \beta \in \mathbb{N}. \quad (71)$$

And the full Hessian is bounded below and above

$$\mu I \preceq J(x) \preceq L I, \quad \forall x. \quad (72)$$

E_2 The variance of subsampled gradients is uniformly bounded by a constant C .

$$\text{tr}(\text{Cov}(F(x))) \leq C^2, \quad \forall x \quad (73)$$

E_3 Hessian is M-Lipschitz, that is

$$\|J(x) - J(y)\| \leq M\|x - y\|, \quad \forall x, y \quad (74)$$

E_4 The variance of subsampled Hessians is bounded by a constant σ .

$$\|\mathbb{E}_{\mathcal{H}}[(J(x; \mathcal{H}) - J(x))]\| \leq \sigma, \quad \forall x \quad (75)$$

E_5 There exists a constant γ such that

$$\mathbb{E}[\|x_n - x^*\|^2] \leq \gamma(\mathbb{E}[\|x_n - x^*\|])^2.$$

Firstly, we recall the few results on subsample Newton method from [5].

Theorem B.13 (Theorem 2.2 in [5]). *Assume x_n is generated by Algorithm 5 with $|\mathcal{G}_i| = \eta^i$ for some $\eta > 1$, $|\mathcal{H}| = \beta \geq 1$ and $s_i = s = \frac{\mu\beta}{L}$ and Assumptions E1-E2 hold, then*

$$\mathbb{E}_k[\phi(x_k) - \phi(x^*)] \leq \alpha\tau^k, \quad (76)$$

where

$$\alpha = \max \left\{ \phi(x_0) - \phi(x^*), \frac{C^2 L \beta}{\mu \mu \beta} \right\} \quad \text{and} \quad \tau = \max \left\{ 1 - \frac{\mu \mu \beta}{2 L L \beta}, \frac{1}{\eta} \right\}.$$

Theorem B.14 (Lemma 2.3 from [5]). *Assume x_n is generated by Algorithm 5 with $s_i \equiv 1$ and Assumptions E1-E3 hold. Then*

$$\mathbb{E}_k[\|x_{n+1} - x^*\|] \leq \frac{1}{\mu|\mathcal{H}_n|} \left[\frac{M}{2} \|x_n - x^*\|^2 + \mathbb{E}_k[\|(J(x_n; \xi_{\mathcal{H}_n}) - J(x_n))(x_n - x^*)\|] + \frac{C}{\sqrt{|\mathcal{G}_n|}} \right]$$

Lemma B.15 (Lemma 2.4 from [5]). *Assume the assumption E1 and E4 hold. Then*

$$\mathbb{E}_k[\|(J(x_n; \xi_{\mathcal{H}_n}) - J(x_n))(x_n - x^*)\|] \leq \frac{\sigma}{\sqrt{|\mathcal{H}_n|}} \|x_n - x^*\|. \quad (77)$$

Theorem B.16 (Convergence of stochastic version of nITGCR). *Assume $|\mathcal{H}_n| = \beta \geq \frac{16\sigma^2}{\mu}$, $\forall n$, residue check is satisfied for $\eta_n \leq \eta \leq \frac{1}{4L}$ and assumptions E1-E5 hold. The iterates generated by the stochastic version Algorithm 2 converges to x^* if $\|x_k - x^*\| \leq \frac{\mu}{2M\gamma}$.*

$$\mathbb{E}\|x_{n+1} - x^*\| \leq \frac{3}{4} \mathbb{E}\|x_n - x^*\| \quad (78)$$

Proof.

$$\mathbb{E}_n[\|x_{n+1} - x^*\|] = \mathbb{E}_n[\|x_n - x^* - J(x_n)^{-1}F(x_n)\|] + \mathbb{E}_n[\|J(x_n)^{-1}F(x_n) + P_n V_n^T y_n\|]$$

The first term can be bounded using the Theorem B.14 and Lemma B.15,

$$\begin{aligned} \mathbb{E}_n[\|x_n - x^* - J(x_n)^{-1}F(x_n)\|] &\leq \frac{1}{\mu|\mathcal{H}_n|} \left[\frac{M}{2} \|x_n - x^*\|^2 + \right. \\ &\quad \left. \mathbb{E}_k[\|(J(x_n; \xi_{\mathcal{H}_n}) - J(x_n))(x_n - x^*)\|] + \frac{C}{\sqrt{|\mathcal{G}_n|}} \right] \\ &\leq \frac{1}{\mu} \left[\frac{M}{2} \|x_n - x^*\|^2 + \frac{\sigma}{\sqrt{|\mathcal{H}_n|}} \|x_n - x^*\| \right] \\ &= \frac{M}{2\mu} \|x_n - x^*\|^2 + \frac{\sigma}{\mu\sqrt{\beta}} \|x_n - x^*\| \end{aligned}$$

We can bound the second term through the line search, recall at each iteration we have

$$\begin{aligned}\mathbb{E}_n[\|J(x_n)^{-1}F(x_n) + P_n V_n^T y_n\|] &= \mathbb{E}_n[\|F(x_n) + J(x_n)P_n V_n^T y_n\|] \\ &\leq \eta_n \mathbb{E}_k[\|F(x_n)\|] \leq \eta_n L \|x_n - x^*\|.\end{aligned}$$

The last inequality comes from the assumption that eigenvalues of $J(x)$ is uniformly upper bounded by L . Finally, combining the above inequalities gives us

$$\mathbb{E}_n[\|x_{n+1} - x^*\|] \leq \frac{M}{2\mu} \|x_n - x^*\|^2 + \frac{\sigma}{\mu\sqrt{\beta}} \|x_n - x^*\| + \eta_n L \|x_n - x^*\|$$

Taking the total expectation on both sides leads to

$$\begin{aligned}\mathbb{E}\mathbb{E}_n[\|x_{n+1} - x^*\|] &= \mathbb{E}\|x_{n+1} - x^*\| \leq \frac{M}{2\mu} \mathbb{E}[\|x_n - x^*\|^2] + \left(\frac{\sigma}{\mu\sqrt{\beta}} + \eta_n L\right) \mathbb{E}[\|x_n - x^*\|] \\ &\leq \frac{M\gamma}{2\mu} \mathbb{E}\|x_n - x^*\| \mathbb{E}\|x_n - x^*\| + \left(\frac{\sigma}{\mu\sqrt{\beta}} + \eta_n L\right) \mathbb{E}[\|x_n - x^*\|]\end{aligned}$$

We prove the convergence by induction, notice that

$$\begin{aligned}\mathbb{E}\|x_1 - x^*\| &\leq \frac{M\gamma}{2\mu} \mathbb{E}\|x_0 - x^*\| \mathbb{E}\|x_0 - x^*\| + \left(\frac{\sigma}{\mu\sqrt{\beta}} + \eta_n L\right) \mathbb{E}[\|x_0 - x^*\|] \\ &\leq \left(\frac{M\gamma}{2\mu} \mathbb{E}\|x_0 - x^*\| + \frac{\sigma}{\mu\sqrt{\beta}} + \eta_n L\right) \mathbb{E}[\|x_0 - x^*\|] \\ &\leq \left(\frac{M\gamma}{2\mu} * \frac{\mu}{2M\gamma} + \frac{\sigma}{\mu\sqrt{\frac{16\sigma^2}{\mu}}} + \frac{1}{4L} * L\right) \mathbb{E}\|x_0 - x^*\| = \frac{3}{4} \mathbb{E}\|x_0 - x^*\|\end{aligned}$$

Now assume inequality 78 holds for $n - th$ iteration, we prove it for $n + 1$ -th iteration

$$\mathbb{E}\mathbb{E}_n[\|x_{n+1} - x^*\|] \leq \left(\frac{M\gamma}{2\mu} \mathbb{E}\|x_n - x^*\| + \frac{\sigma}{\mu\sqrt{\beta}} + \eta_n L\right) \mathbb{E}[\|x_n - x^*\|] \leq \frac{3}{4} \mathbb{E}[\|x_n - x^*\|]$$

□

Appendix C Experimental Details and More Experiments

In this section, we first include more experimental details that could not be placed in the main paper due to the space limitation. We then present more experimental results of NLGCR for different settings and difficult problems.

C.1 Experimental Details

We provide codes implemented in both Matlab and Python. All experiments were run on a Dual Socket Intel E5-2683v3 2.00GHz CPU with 64 GB memory and NVIDIA GeForce RTX 3090.

For linear problems considered in Section 4.1, **A**, **b**, **c**, and initial points are generated using normally distributed random number. We use $\mathbf{A}^T \mathbf{A} + \alpha \mathbf{I}$ to generate symmetric matrices. The step size is set as 1 after rescaling **A** to have the unit 2-norm. For solving linear equations, we depict convergence by use of the norm of residual, which is defined as $\|\mathbf{b} - \mathbf{A}\mathbf{x}\|$. For solving bilinear games, we depict convergence by use of the norm of distance to optima, which is defined as $\|\mathbf{w}^* - \mathbf{w}_t\|$. For most baselines, we use the Matlab official implementation.

The softmax regression problem considered in Section 4.2 is defined as follows,

$$f = -\frac{1}{s} \sum_{i=1}^s \log \left(\frac{e^{w_{y_j}^T x^{(i)}}}{\sum_{j=1}^k e^{w_j^T x^{(i)}}} \right), \quad (79)$$

where s is the total number of sample, k is the total number of classes, $x^{(i)}$ is vector of all features of sample i , w_j is the weights for the j^{th} class, and y_j is the correct class for the i^{th} sample.

C.2 TGCR(1) for linear system

We first test the robustness of TGCR(1) for solving linear systems by running with 50 different initials. Figure 3 indicates TGCR converge well regardless of initialization. We then compare the performance on bilinear games with Anderson Acceleration as [26] shows AA outperforms existing methods on such problems.

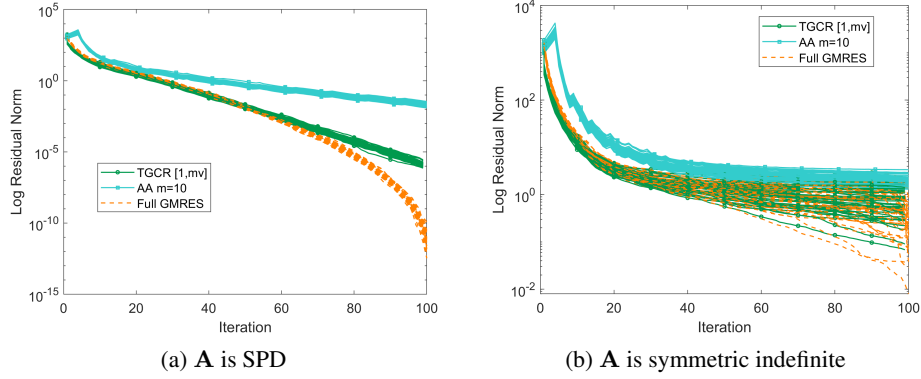


Figure 3: **Linear Systems $\mathbf{Ax} = \mathbf{b}$, $\mathbf{A} \in \mathbb{R}^{100 \times 100}$** : Comparison in terms of iteration over 50 random runs. We can observe that TGCR(1) match full memory GMRES in the first stage and consistently converge faster than AA(10).

Minimax Optimization. Next, we test TGCR on the following zero-sum bilinear games:

$$\min_{\mathbf{x} \in \mathbb{R}^n} \max_{\mathbf{y} \in \mathbb{R}^n} f(\mathbf{x}, \mathbf{y}) = \mathbf{x}^T \mathbf{A} \mathbf{y} + \mathbf{b}^T \mathbf{x} + \mathbf{c}^T \mathbf{y}, \quad \mathbf{A} \text{ is full rank.} \quad (80)$$

Bilinear games are often regarded as an important but simple class of problems for theoretically analyzing and understanding algorithms for solving general minimax problems [66, 2]. Here we consider simultaneous GDA mapping for minimax bilinear games $\begin{pmatrix} \mathbf{I} & -\eta \mathbf{A} \\ \eta \mathbf{A}^T & \mathbf{I} \end{pmatrix}$ [26]. Although this mapping is skew-symmetric, TGCR can still exploit the short-term recurrence. It can be observed from Figure 4 that Krylov subspace methods such as TGCR and AA converge fast for bilinear problem when A is either SPD or random generated. More importantly, Figure 4 demonstrates that TGCR (1) exhibits a *superlinear* convergence rate and converges to optimal significantly faster than AA(m) in terms of both iteration number and computation time.

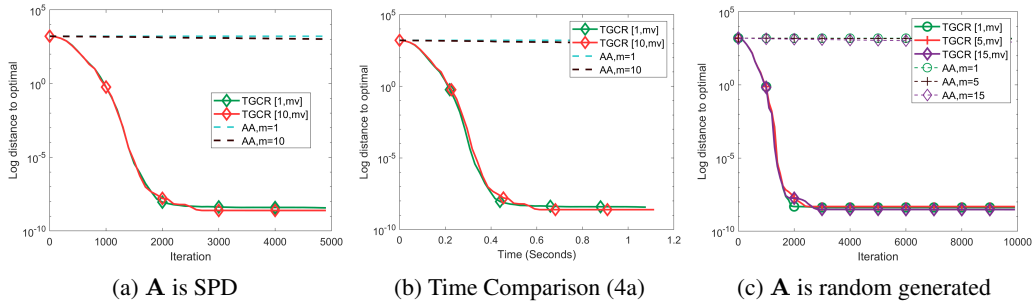


Figure 4: **Bilinear Problems**: 4a: Distance to optimal vs. Iterations; 4b: Distance to optimal vs. Time; 4c: Distance to optimal vs. Iterations. It can be observed that the short-term property (Theorem 3.1) holds as long as the mapping is symmetric or skew-symmetric.

C.3 TGCR(1) for nonsymmetric quadratic minimization and linear system

A quadratic form is simply a scalar, quadratic function of a vector with the form

$$f(x) = \frac{1}{2} \mathbf{x}^T \mathbf{A} \mathbf{x} - \mathbf{b}^T \mathbf{x} + c, \quad (81)$$

where \mathbf{A} is a matrix, \mathbf{x} and \mathbf{b} are vectors, and c is a scalar constant. When \mathbf{A} is symmetric and positive-definite, $f(x)$ is minimized by the solution to $\mathbf{A}\mathbf{x} = \mathbf{b}$.

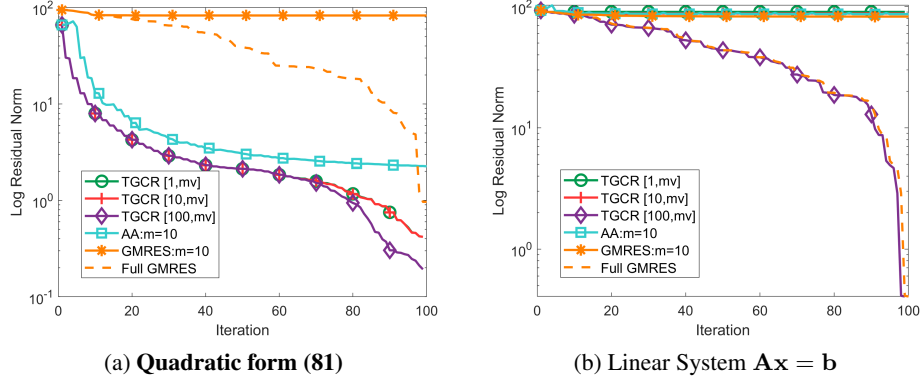


Figure 5: \mathbf{A} is nonsymmetric. In Figure 5a, we can find TGCR (1) still converges fast because the Hessian of Equation 81 is $\frac{1}{2}(\mathbf{A}^T + \mathbf{A})$, which is still symmetric. In Figure 5b, we can find TGCR(100) has the same convergence rate with GMRES. This shows that TGCR(m) can still converge for solving nonsymmetric linear systems with $m > 1$ and is mathematically equivalent to non-restart GMRES when m is equal to the matrix size.

C.4 Investigation of Stochastic NLTGCR

In the main paper, we report the effectiveness of NLTGCR for softmax regression on MNIST. We give the result about using deterministic and stochastic gradients in Figure 2 and find that NLTGCR only requires a small batch size and table size (1). In this section, we further investigate the effectiveness of NLTGCR in a stochastic setting, provide additional experimental results in Figure 6 and Figure 7. From Figure 6, we can observe that using the same batch size and table size $m = 1$, stochastic NLTGCR consistently outperforms stochastic AA. Also, it can be observed from Figure 7 that stochastic NLTGCR outperforms stochastic AA for different table size using a fixed batch size of 1500. Although the short-term property does not strictly hold in a stochastic setting, we can still observe that NLTGCR outperforms AA with a smaller variance. In addition, it is worth noting that a smaller table size works better for both NLTGCR and AA. We suspect this is due to the accumulation of inaccurate gradient estimates.

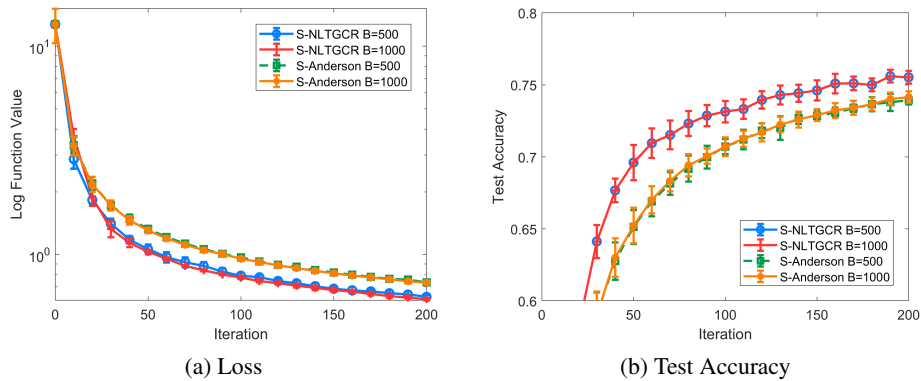


Figure 6: Softmax Regression: Effects of batch size, $m = 1$

Compatible with Momentum Another important technique in optimization is momentum, which speeds up convergence significantly both in theory and in practice. We experimentally show that it is possible to further accelerate the convergence of NLTGCR by using Momentum. We run stochastic NLTGCR with different momentum term and present results in Figure 8. It suggests that by incorporating momentum into NLTGCR momentum further accelerates the convergence, although the

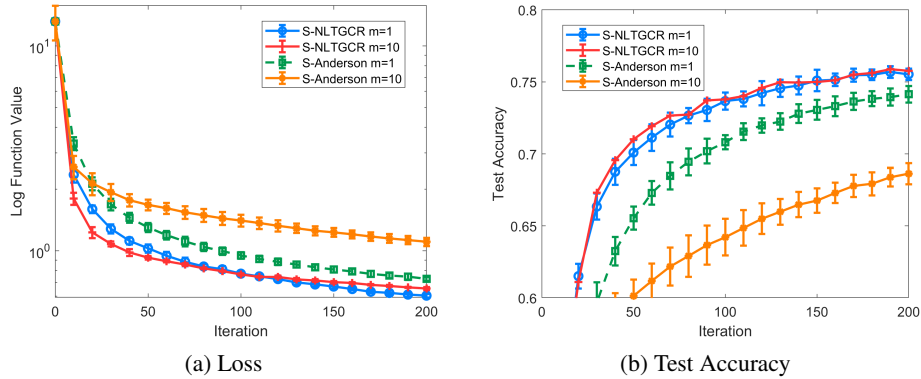


Figure 7: **Softmax Regression: Effects of table size in a stochastic setting, $B = 1000$**

variance gets larger for a large momentum term v . We leave the theoretical analysis of of NLTGCR with momentum for future work.

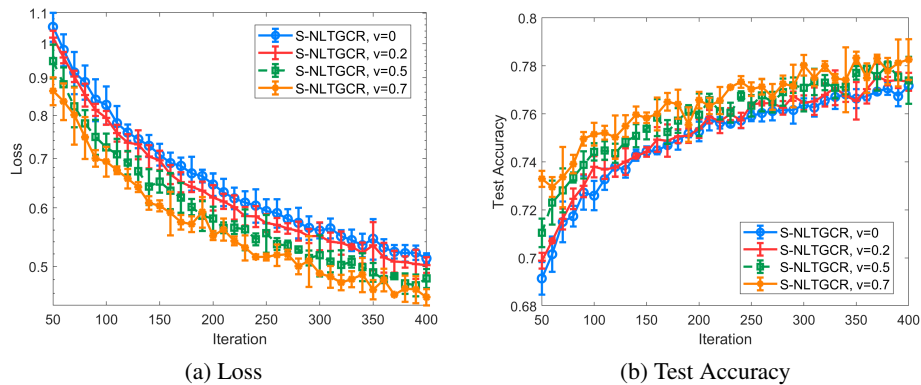


Figure 8: **Softmax Regression: Compatible with momentum.** We further test the acceleration effect of momentum on stochastic NLTGCR. It shows that stochastic NLTGCR with momentum converges faster than the one without momentum.

C.5 Results for Deep learning applications

C.5.1 Image classification using CNN

In a more realistic setting, we test our algorithm for neural networks on an image classification task. Particularly, we use the standard MNIST dataset². The architecture of the network is based on the official PyTorch implementation³. We tried our best to ensure that the baselines had the best performance in the tests. Hyperparameters are selected after grid search. We use a batch size of 64. For SGD, Nestrov ($v = 0.9$), Adam (default β_1 and β_2), and NLTGCR ($m = 1$), we use a learning rate of 1×10^{-2} , 1×10^{-1} , 1×10^{-3} , and 1×10^{-3} , respectively. Figure 9a shows the curves of loss for training the neural network on MNIST. Figure 9b shows the curves of test accuracy on MNIST. It can be found that NLTGCR outperforms SGD and Nestrov and is comparable to Adam. In addition, we conduct experiments on the effects of table size for NLTGCR and present results in Figure 10. Although Figure 10b, shows $m = 10$ does slightly better, we found that $m = 1$ generally yields robust results. In addition, $m = 1$ significantly reduces the memory and computation overhead. As a result, we would like to suggest $m = 1$ for general experiments. These preliminary results provide insights of the effectiveness of our algorithm for training neural networks. Our algorithm is comparable with the widely used optimizer, Adam. In addition, our algorithm is more memory and computation efficient than other nonlinear acceleration methods including AA and RNA. As a result,

²<http://yann.lecun.com/exdb/mnist/>

³implementation <https://github.com/pytorch/examples/blob/master/mnist>.

it is worth investigating the performance of NLTGCR for different tasks and more complex networks. We leave it for our future work.

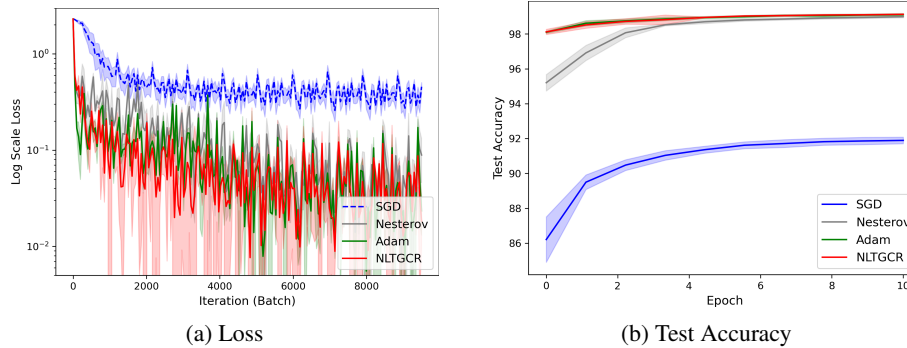


Figure 9: **Training on MNIST.** Averaged on 5 runs, $m = 1$ for our algorithm.

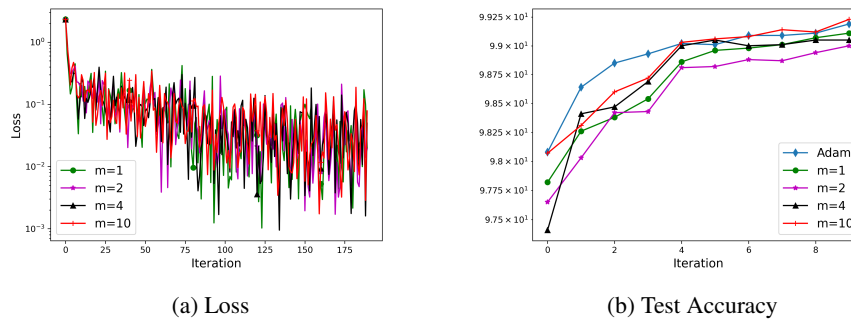


Figure 10: **Effects of table size m** We run experiments with a fixed seed. It shows S-NLTGCR(10) does slightly better than Adam and S-NLTGCR($m < 10$). We would suggest to use $m = 1$ for saving memory and computation.

C.5.2 Image classification using ResNet

We now perform our tests on ResNet32 [28]⁴ using CIFAR10 [37]. We randomly split the training set of all the datasets into two subsets, train and validation. The former is used to train the neural network, whereas the latter is used for measuring the performance of the learned model. Hyperparameters are selected after a grid search. We use a batch size of 128. For Nesterov ($v = 0.9$), Adam (default β_1 and β_2), and NLTGCR ($m = 1$), we use a learning rate of 3×10^{-4} , 1×10^{-3} , and 1×10^{-1} , respectively. For better visualization, figure 11 shows the curves of training loss and validation accuracy using 50 epochs. It can be observed that nLTGCR(1) consistently outperforms baselines and has a smaller variance.

C.5.3 Time series forecasting using LSTM

Next, we test our algorithm for Long Short-Term Memory [32] on time series forecasting task using Airplane Passengers and Shampoo Sales Dataset⁵. We use a learning rate of 0.04 and the mean squared error (MSE) as our loss function and evaluation metric. Figure 12 depicts the MSE on validation set during training. It shows TGCR converges better than baselines. It also suggests TGCR is capable of optimizing complex deep learning architectures.

⁴https://github.com/akamaster/pytorch_resnet_cifar10

⁵<https://github.com/spdin/time-series-prediction-lstm-pytorch>

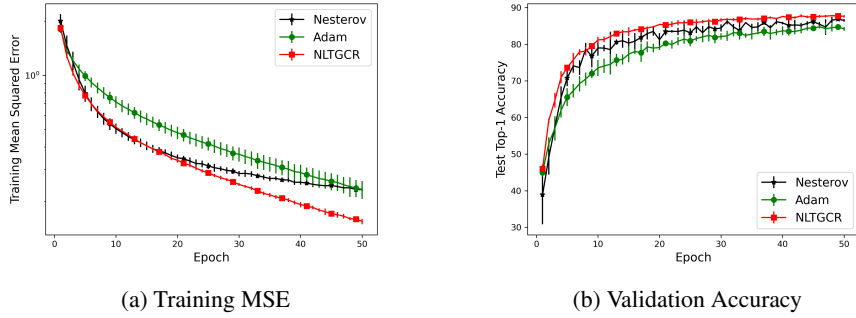


Figure 11: **Validation MSE of training ResNet32 on CIFAR10.** Averaged on 5 runs (manual random seed 0 to 5 for all methods), $m = 1$ for our algorithm. nLTGCR(1) consistently outperforms baselines and has a smaller variance.

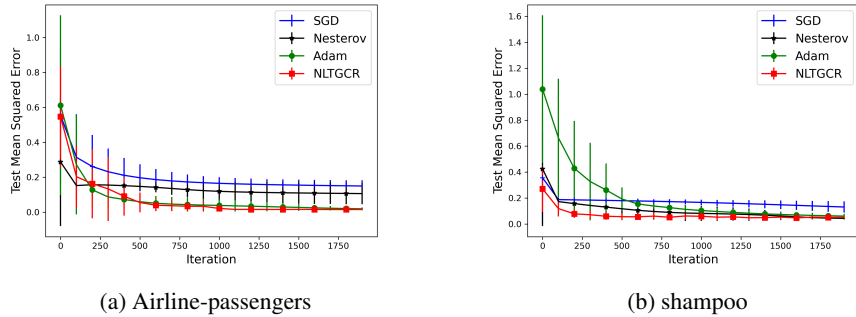
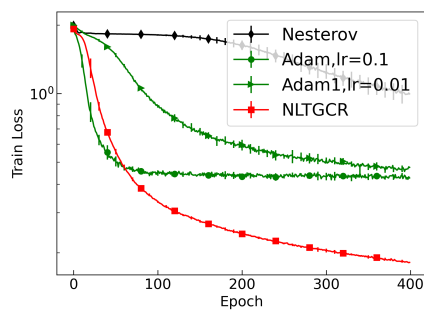


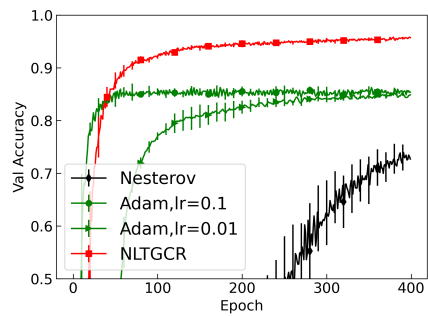
Figure 12: **Validation MSE of LSTM on two datasets.** Averaged on 5 runs, $m = 1$ for our algorithm. It can be observed that nLTGCR(1) outperforms baselines.

C.5.4 Semi-supervised classification of graph data using GCN

We also test the effectiveness of nLTGCR on GCN [36] using Cora Dataset. It consists of 2708 scientific publications classified into one of seven different classes. The citation network consists of 5429 links. Each publication in the dataset is described by a 0/1-valued word vector indicating the absence/presence of the corresponding word from the dictionary. The objective is to accurately predict the subject of a paper given its words and citation network, as known as node classification. Hyperparameters are selected after a grid search. For Nesterov ($v = 0.9$), Adam (default β_1 and β_2), and nLTGCR ($m = 1$), we use a learning rate of 1×10^{-1} , 1×10^{-2} , and 1×10^{-1} , respectively. Figure 13a and 13b depict the training loss and validation accuracy averaged on 5 runs. It shows TGCR converges faster and achieves higher accuracy than baselines, which demonstrates the effectiveness of TGCR optimizing complex deep learning architectures.



(a) Train Loss



(b) Validation Accuracy

Figure 13: **Validation accuracy on CORA using GCN.** Averaged on 5 runs, $m = 1$ for our algorithm. It can be observed that nLTGCR(1) significantly outperforms baselines.