EURO Journal on Computational Optimization 10 (2022) 100047

Contents lists available at ScienceDirect

EURO Journal on Computational Optimization

journal homepage: www.elsevier.com/locate/ejco

Direct nonlinear acceleration

Aritra Dutta [c,b,1], El Houcine Bergou [a,b,*,1], Yunming Xiao [b,d], Marco Canini [b], Peter Richtárik [b]

[a] *Mohammed VI Polytechnic University (UM6P), Ben Guerir, Morocco*
[b] *King Abdullah University of Science and Technology (KAUST), Saudi Arabia*
[c] *University of Southern Denmark (SDU), Denmark*
[d] *Northwestern University, USA*

ARTICLE INFO

ABSTRACT

Optimization acceleration techniques such as momentum play a key role in state-of-the-art machine learning algorithms. Recently, generic vector sequence extrapolation techniques, such as regularized nonlinear acceleration (RNA) of Scieur et al. [22], were proposed and shown to accelerate fixed point iterations. In contrast to RNA which computes extrapolation coefficients by (approximately) setting the gradient of the objective function to zero at the extrapolated point, we propose a more direct approach, which we call *direct nonlinear acceleration (DNA)*. In DNA, we aim to minimize (an approximation of) the function value at the extrapolated point instead. We adopt a regularized approach with regularizers designed to prevent the model from entering a region in which the functional approximation is less precise. While the computational cost of DNA is comparable to that of RNA, our direct approach significantly outperforms RNA on both synthetic and real-world datasets. While the focus of this

* Corresponding author.
E-mail address: elhoucine.bergou@um6p.ma (E.H. Bergou).
[1] Equal contribution.

https://doi.org/10.1016/j.ejco.2022.100047

paper is on convex problems, we obtain very encouraging results in accelerating the training of neural networks.

1. Introduction

In this paper we consider the generic unconstrained minimization problem

$$\min_{x\in\mathbb{R}^n} f(x), \tag{1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is a smooth objective function and bounded from below. One of the most fundamental methods for solving (1) is *gradient descent (GD)*, on which many state-of-the-art methods are based. Given current iterate $x_k \in \mathbb{R}^n$, the update rule of GD is

$$x_{k+1} = x_k - \alpha_k \nabla f(x_k), \tag{2}$$

where $\alpha_k > 0$ is a stepsize. The efficiency of GD depends on further properties of f. Assuming f is L–smooth and μ–strongly convex, for instance, the iteration complexity of GD is $\mathcal{O}(\kappa \log(1/\epsilon))$, where $\kappa = L/\mu$ and ϵ is the target error tolerance. However, it is known that GD is not the "optimal" gradient type method: it can be *accelerated*.

The idea of accelerating converging optimization algorithms can track its history back to 1964 when Polyak proposed his "heavy ball" method [20]. In 1983, Nesterov proposed his accelerated version for general convex optimization problems. Comparing with Polyak's method, Nesterov's method gives acceleration for general convex and smooth problems and the iteration complexity improves to $\mathcal{O}(1/\sqrt{\epsilon})$ [17]. In 2009, Beck and Teboulle proposed fast iterative shrinkage thresholding algorithm (FISTA) [3] that uses Nesterov's momentum coefficient and accelerates *proximal* type algorithms to solve a more complex class of objective functions that combine a smooth, convex loss function (not necessarily differentiable) and a strongly convex, smooth penalty function (also see [18,19]). To develop further insights into Nesterov's method, Su et al. [26] examined a continuous time 2nd-order ODE which at its limit reduces to Nesterov's accelerated gradient method. In addition, Lin et al. [16] introduced a generic approach known as *catalyst* that minimizes a convex objective function via an accelerated proximal point algorithm and gains acceleration in Nesterov's sense. [6] proposed a geometric alternative to gradient descent that is inspired by ellipsoid method and produces acceleration with complexity $\mathcal{O}(1/\sqrt{\epsilon})$. Recently, [32] used a linear *coupling* of gradient descent and mirror descent and claimed to attain acceleration in Nesterov's sense as well. In contrast, the *sequence acceleration* techniques accelerate a sequence independently from the iterative

method that produces this sequence. In other words, these techniques take a sequence $\{x_k\}$ and produce an accelerated sequence based on the linear combination of x_ks such that the new accelerated sequence converges faster than the original. In the same spirit, recently, Scieur et al. [22,23,4] proposed an acceleration technique called regularized nonlinear acceleration (RNA). Scieur et al.'s idea is based on Aitken's Δ^2-algorithm [1] and Wynn's ϵ-algorithm [30] (or recursive formulation of generalized Shanks transform [24,30,5]). To achieve acceleration, Scieur et al. considered a technique known as minimum polynomial approximation and they assumed a *linear* model for the iterates near the optimum. They also proposed a *regularized* variant of their method to stabilize it numerically. The intuition behind the regularized nonlinear acceleration of Scieur et al. is very natural. To minimize f as in (1), they considered the sequence of iterates $\{x_k\}_{k\geq 0}$ is generated by a fixed-point map. If $x^\star$ is a minimizer of f, $\nabla f(x^\star) = 0$, and hence through extrapolation one can find:

$$c^\star \approx \arg\min_c \left\{ \left\| \nabla f\left(\sum_{k=0}^{K} c_k x_k \right) \right\| \ : \ c \in \mathbb{R}^{K+1}, \ \sum_{k=0}^{K} c_k = 1 \right\}, \tag{3}$$

such that the next (accelerated) point can be generated as a linear combination of $K+1$ previous iterates: $x = \sum_{i=0}^{K} c_i^\star x_i$. We review RNA in detail in Section 1.2.2.

Notation. We denote the ℓ_2-norm of a vector x and the spectral norm of a matrix A by $\|x\|$ and $\|A\|$, respectively. Further define $\|x\|_M$ by $\|x\|_M := \sqrt{x^\top M x}$.

1.1. Contributions

We highlight our main contributions in this paper as follows:

Direct nonlinear acceleration (DNA). Inspired by Anderson's acceleration technique [2] (see Appendix for a brief description of Anderson's acceleration) and the work of Scieur et al. [22], we propose an extrapolation technique that accelerates a converging iterative algorithm. However, in contrast to [22], we find the extrapolation coefficients $c^\star$ by directly minimizing the function at the linear combination of $K+1$ iterates $\{x_k\}_{k\geq 0}^K$ with respect to $c \in \mathbb{R}^{K+1}$. In particular, for a given sequence of iterates $\{x_k\}_{k\geq 0}^K$ we propose to approximately solve:

$$\boxed{\min_{c\in\mathbb{R}^{K+1}} f\left(\sum_{k=0}^{K} c_k x_k \right) + \lambda g(c),} \tag{4}$$

where $\lambda > 0$ is a balancing parameter and g is a penalty function. As our approach tries to minimize the functional value directly, we call it as *direct nonlinear acceleration* (DNA). We note that our formulation shares some similarities with that of Zhang et al. [31]. Additionally, Riseth [21] proposed an objective acceleration that also minimizes an approximation to the *objective function* on subspaces of $\mathbb{R}^n$. However, unlike these

works, we do not require line search and check a decrease condition at each step of our algorithm.

Regularization. We propose several versions of DNA by varying the penalty function $g(c)$. This helps us to deal with the numerical instability in solving a linear system as well as to control errors in gradient approximation. In our first version, we let $g(c) = 1_S(c)$, where $S := \{c \ : \ \sum_i c_i = 1\}$ and $1_S(c) = 0$ if $c \in S$, while $1_S(c) = +\infty$, otherwise. Later, we propose two regularized *constraint-free* versions to find a better minimum of the function f by expanding the search space of extrapolating coefficients to $\mathbb{R}^{K+1}$ rather than restricting them over the space S. To this end, the first constraint-free version adds a quadratic regularization $g(c) = \left\|\sum_{i=0}^{K} c_i x_i - y\right\|^2$ to the objective function, where y is a reference point and $g(c)$ controls how far we want the linear combination $\sum_i c_i x_i$ to deviate from y. In the second constraint-free version, we add the regularization directly on c. We add a quadratic term of the form $g(c) = \|c - e\|^2$ to the objective function, where e is a reference point to c and $g(c)$ controls how far we want c to deviate from e. In contrast, the *regularized* version of RNA only considers a ridge regularization $\|c\|^2$ for numerical stability. Trivially, we note that by setting $e = 0$, we recover the regularization proposed in RNA. We argue that by using a different penalty function $g(c)$ as regularizer our DNA is more robust than RNA.

General convergence result and quantification between RNA and DNA in minimizing quadratic functions by using GD iterates. We provide a general convergence result of DNA in Theorem 2. If $g(c) = 0$ or $g(c) = 1_{\sum_i c_i = 1}$, in terms of the functional value, we always obtain a better accelerated point than RNA. Moreover, the acceleration obtained by DNA can be *theoretically* directly implied from the existing results of Scieur et al. [22]. If $g(c) = 0$, we show by a simple example on quadratic functions that DNA outperforms RNA by an arbitrary large margin. If $g(c) = 1_{\sum_i c_i = 1}$, we also quantify the functional values obtained from both RNA and DNA for quadratic functions and provide a bound on how DNA outperforms RNA in this setup.

Numerical results. Our empirical results show that for smooth and strongly convex functions, minimizing the functional value converges faster than RNA. In practice, our acceleration techniques are robust and outperform that of Scieur et al. [22] by large margins in almost all experiments on both synthetic and real datasets. To further push the robustness of our methods, we test them on nonconvex problems as well. As a proof of concept, we trained a simple neural network classifier on MNIST dataset [15] via GD and accelerate the GD iterates via the online scheme in [22] for both RNA and DNA. Next, we train ResNet18 network [12] on CIFAR10 dataset [14] by SGD and accelerate the SGD iterates via the online scheme in [22] for both RNA and DNA. In both cases, DNA outperform RNA in lowering the generalization errors of the networks.

1.2. Related work

The sequence acceleration or non-linear acceleration has rich history in optimization literature. We mention about a few works that are similar to ours. In [25], nonlinear

generalized minimal residual (N-GMRES) is used as an accelerator for solving the nonlinear system, $\nabla f(x) = 0$ that arises from the first order optimality condition. This is similar to the classical work of Washio et al. [29] and can be connected to Anderson's acceleration [2]. In contrast, objective acceleration by [21], proposed to minimize a more direct objective—an approximation to the objective unction f on subspace of $\mathbb{R}^n$. In that regard it is closely related to our work. However, [21] requires line search and check a decrease condition at each step. In our algorithm we do not require such kind of conditions. Additionally, we propose several regularized versions to bring numerical stability in solving the linear systems. In another line of work, Zhang et al. [31] do not consider a direct acceleration scheme as they use type-I Anderson acceleration to solve general non-smooth fixed-point problems.

1.2.1. Anderson's acceleration [2]

There are several acceleration techniques that have been proposed in the literature and they pose a lot of similarities. We quote the authors from [5] – "Methods for accelerating the convergence of various processes have been developed by researchers in a wide range of disciplines, often without being aware of similar efforts undertaken elsewhere." In 1965 Anderson's acceleration was designed to accelerate Picard iteration for electronic structure computations. Because it is relevant in our current work, we give a brief description of it for completeness.

For a given sequence of iterate, $\{x_k\}$ with $x_k \in \mathbb{R}^n$ and a mapping, $\Phi(\cdot) : \mathbb{R}^n \to \mathbb{R}^n$, the fixed-point algorithm generates a recursive update of the iterates as:

$$x_{k+1} = \Phi(x_k). \tag{5}$$

Let there be $m_k + 1$ evaluations of the fixed point map ϕ. Anderson's acceleration technique computes a new iteration as a linear combination of the previous m_k+1 evaluations. We explain it formally in Algorithm 1. In Algorithm 1, m is considered as a hyperparameter that sets the quantity m_k as $\min\{m, k\}$, where k is the iteration counter and m is known as the depth. This is used to determine the window size to compute $\hat{c}$–the coefficients for linear combination of the fixed point evaluations. In other words, in each iteration, by solving the optimization problem:

$$\hat{c}^{(k)} = \arg\min_{c} \|F^k c\| \quad \text{subject to} \quad \sum_i c_i = 1,$$

where $F^k = (f_{k-m_k}, f_{k-m_k+1}, \cdots, f_k) \in \mathbb{R}^{n\times(m_k+1)}$, and $f_i = \Phi(x_i) - x_i$, one can obtain the extrapolation coefficients $\hat{c}^{(k)}$ that help to determine the accelerated point x_{k+1}. Toth and Kelley pointed out that, in principle, any norm can be used in the minimization step [27]. The summability of the coefficients c_i or the *normalization condition* was not explicitly mentioned in the original work of Anderson. Because c_i's can be determined

Algorithm 1: Anderson Acceleration.

Input : $x_0 \in \mathbb{R}^n$ and $m \geq 1$;
Initialize: Set $x_1 = \Phi(x_0), m_k = \min\{m, k\}, F^k = (f_{k-m_k}, f_{k-m_k+1}, \cdots, f_k) \in \mathbb{R}^{n\times(m_k+1)}$, where $f_i = \Phi(x_i) - x_i$;
for $k = 1, 2, \cdots$ **do**
  Find $\hat{c}^{(k)} \in \mathbb{R}^{(m_k+1)}$ such that: $\hat{c}^{(k)} = \arg\min_\alpha \|F^k c\|$ subject to $\sum_i c_i = 1$;
  Set $x_{k+1} = \sum_{i=0}^{m_k} \hat{c}_i^{(k)} \Phi(x_{k-m_k+i})$.
end

up to a multiplicative scalar, one can impose the *normalization condition*. However, it does not restrict generality. We refer the readers to [13,27,28,2,5] for a comprehensive idea of Anderson's acceleration technique. See more discussions involving Anderson's acceleration, Krylov subspace methods, and our work in Remarks 1 and 2.

1.2.2. Regularized nonlinear acceleration

In RNA, one solves (3) by assuming that the gradient can be approximated by linearizing it in the neighborhood of $\{x_k\}_{k=0}^K$. Thus, by assuming $\sum_{k=0}^K c_k = 1$, the relation $\left\|\nabla f\left(\sum_{k=0}^K c_k x_k\right)\right\| \approx \left\|\sum_{k=0}^K c_k \nabla f(x_k)\right\|$ holds. Hence, one can approximately solve (3) via:

$$c^\star = \arg\min_c \left\|\sum_{k=0}^K c_k \nabla f(x_k)\right\| = \left\|\sum_{k=0}^K c_k \tilde{R}_k\right\| \quad \text{subject to } c \in \mathbb{R}^{K+1},\ \sum_{k=0}^K c_k = 1, \tag{6}$$

where $\tilde{R}_k$ is the k^{th} column of the matrix $\tilde{R}$, which holds information about $\nabla f(x_k)$. Moreover (6) does not need an explicit access to the gradient and it can be seen as an approximated minimal polynomial extrapolation (AMPE) as in [7,22,23]. If the sequence $\{x_k\}$ is generated via GD (as in (2)), then $\tilde{R} = [{}^{(x_0-x_1)}/_{\alpha_0}, \ldots, {}^{(x_K-x_{K+1})}/_{\alpha_K}]$. Also, if $\tilde{R}^\top\tilde{R}$ is nonsingular, then the minimizer of (6) is explicitly given as: $c^\star = \frac{(\tilde{R}^\top\tilde{R})^{-1}\mathbb{1}}{\mathbb{1}^\top(\tilde{R}^\top\tilde{R})^{-1}\mathbb{1}}$. If $\tilde{R}^\top\tilde{R}$ is singular then c is not necessarily unique. Any c of the form $\frac{z}{z^\top\mathbb{1}}$, where z is a solution of $\tilde{R}^\top\tilde{R}z = \mathbb{1}$, is a solution of (6). To deal with the numerical instabilities and the case when the matrix $\tilde{R}^\top\tilde{R}$ is singular, Scieur et al. proposed to add a regularizer of the form $\lambda\|c\|^2$ to their problem, where $\lambda > 0$. As a result, $c^\star$ is unique and given as $c^\star = \frac{(\tilde{R}^\top\tilde{R}+\lambda I)^{-1}\mathbb{1}}{\mathbb{1}^\top(\tilde{R}^\top\tilde{R}+\lambda I)^{-1}\mathbb{1}}$. The numerical procedure of RNA is given in Algorithm 2. For further details about RNA we refer the readers to [22,23]. Scieur et al. also explained several acceleration schemes to use with Algorithm 2.

2. Direct nonlinear acceleration

Instead of minimizing the norm of the gradient, we propose to minimize the objective function f directly to obtain the coefficients $\{c_k\}$. We set $g(c) = 0$ in (4) and we propose to solve the unconstrained minimization problem

Algorithm 2: RNA.

Input : Sequence of iterates $x_0, \ldots, x_{K+1}$; sequence of step sizes $\alpha_0, \ldots, \alpha_K$; $\mathbb{1} \in \mathbb{R}^{K+1}$: a vector of all 1s; and $\lambda > 0$.

Set $\tilde{R} = \left[\frac{x_0 - x_1}{\alpha_0}, \ldots, \frac{x_K - x_{K+1}}{\alpha_K}\right]$;

Solve the linear system: $\left(\tilde{R}^\top \tilde{R} + \lambda I\right) z = \mathbb{1}$;

Set $c = \frac{z}{z^\top \mathbb{1}} \in \mathbb{R}^{K+1}$;

Output : $x = \sum_{k=0}^{K} c_k x_k$.

$$\boxed{\min_{c \in \mathbb{R}^{K+1}} f\left(Xc\right),} \tag{7}$$

where $X = [x_0, \ldots, x_K]$. We call problem (7) as direct nonlinear acceleration (DNA) without any constraint. If f is quadratic, then we have the following lemma that calculates the extrapolation coefficients, c as the solution of a linear system.

Lemma 1. *Let the objective function f be quadratic and let $\{x_k\}$ be the iterates produced by (2) to minimize f. Then c is a solution of the linear system $X^\top R z = -X^\top \nabla f(0)$, where $R \in \mathbb{R}^{n \times (K+1)}$ is a matrix such that its i^{th} column is $R_i = \frac{x_i - x_{i+1}}{\alpha_i} - \nabla f(0)$ and $X = [x_0, \ldots, x_K]$.*

Proof. Let $h(c) = f(Xc)$, from the first order optimality condition we have

$$\nabla h(c) = X^\top \nabla f\left(Xc\right) = 0$$

For quadratic objective function the gradient is affine, i.e.

$$\nabla f(Xc) = AXc + \nabla f(0) = \sum_{k=0}^{K} c_k A x_k + \nabla f(0) = \sum_{k=0}^{K} c_k \left(\nabla f(x_k) - \nabla f(0)\right) + \nabla f(0).$$

By using the relation between the iterates of GD method we find $\nabla f\left(x_k\right) = \frac{x_k - x_{k+1}}{\alpha_k}$. Hence $\nabla f(Xc) = Rc + \nabla f(0)$. By injecting this in the first order optimality condition we get the result. □

If f is non-quadratic then we can *approximately* solve problem (7) by approximating its gradient by a linear model. In fact, we use the following approximation $\nabla f(x) \approx A(x - y_x) + \nabla f(y_x)$, where we assume that x is close to y_x and A is an approximation of the Hessian. Therefore, by setting $x = Xc$ and $y_x = y$ in the above, we have $\nabla f(Xc) \approx A(Xc - y) + \nabla f(y) = \sum_i c_i A x_i - Ay + \nabla f(y)$, where y is a reference point that is assumed to be in the neighborhood of Xc. For instance, one may choose y to be x_K. Let x_{i-1} be a reference point for x_i, that is, assume that $\nabla f(x_i) \approx A(x_i - x_{i-1}) + \nabla f(x_{i-1})$. Then one can show that $Ax_i = \nabla f(x_i) - \nabla f(0)$. As a result, we have

$$\nabla f(Xc) \approx \sum_i c_i (\nabla f(x_i) - \nabla f(0)) - Ay + \nabla f(y)$$

Algorithm 3: DNA.

Input : Sequence of iterates $x_0, \ldots, x_{K+1}$ and sequence of step sizes $\alpha_0, \ldots, \alpha_K$;

1 Set $R = \left[\frac{x_0 - x_1}{\alpha_0} - \nabla f(0), \ldots, \frac{x_K - x_{K+1}}{\alpha_K} - \nabla f(0)\right]$ and $X = [x_0, \ldots, x_K]$;

2 Set c as a solution of the linear system $X^\top R z = -X^\top \nabla f(0)$;

Output : $x = \sum_{k=0}^{K} c_k x_k$.

$$
\begin{aligned}
&= \sum_i c_i \left(\tfrac{x_i - x_{i+1}}{\alpha_i} - \nabla f(0)\right) - Ay + \nabla f(y) \\
&= \sum_i c_i R_i - Ay + \nabla f(y) \\
&= Rc - Ay + \nabla f(y) \\
&\approx \quad Rc + \nabla f(0). \qquad (8)
\end{aligned}
$$

Therefore, from the first optimality condition and by using (8), we conclude that the solutions of (7) can be approximated by the solutions of the linear system $X^\top R z = -X^\top \nabla f(0)$.

Remark 1. One may consider the following more general approximation

$$\nabla f(Xc) \approx \nabla f(y) + H_y(Xc - y),$$

where H_y is an approximation of the Hessian at y. Therefore, from the first optimality condition and by using the latter approximation, we conclude that the solutions of (7) can be approximated by the solutions of the linear system, $X^T \nabla f(y) + X^T H_y(Xc - y) = 0$ which is equivalent to

$$X^T H_y X c = X^T H_y y - X^T \nabla f(y). \qquad (9)$$

We note that the equation (9) involves the Hessian, H_y, and we wanted to avoid its computation. That is why, in our analysis, we assume $y = 0$. We note that is not valid for the general non-quadratic functions. However, our approximation is feasible for strongly convex functions with smaller condition numbers, which is the setting we consider in this work. For more general setting, one may explore when $y \neq 0$, and we leave it for future work.

2.1. Convergence of DNA on strongly convex functions

Assume that f is μ-strongly convex and we will show the iterates generated by DNA converge with an accelerated rate to the optimal point for the online scheme. To show the convergence of DNA for μ-strongly convex functions, we follow the setup in [22]. For completeness, we will briefly mention it in the following. In [22], Scieur et al. considered

the updates of the iterative algorithm (say, GD in (2)) linear. That is, we can rewrite (2) as:

$$x_{k+1} - x^\star = \mathcal{A}(x_k - x^\star),$$

where $\mathcal{A} \in \mathbb{R}^{n\times n}$ with $\|\mathcal{A}\| \leq \sigma < 1$. The minimal polynomial, p of $\mathcal{A}$ with respect to the vector v is the lowest degree polynomial such that $p(\mathcal{A})v = 0$ and $p(1) = 1$. By setting each column, $\tilde{R}_k$ of $\tilde{R}$ as $\tilde{R}_k = (\mathcal{A} - I_n)(x_k - x^\star)$, we have

$$\|Xc_R - x^\star\| \leq \|(I - \mathcal{A})^{-1}\|\|\tilde{R}c^\star\|.$$

In [22], Scieur et al. bounded $\|\tilde{R}c^\star\|$ as follows: let $\mathcal{A} = Q\Lambda Q^\top$ be the eigen decomposition of $\mathcal{A}$ and let the degree of the minimal polynomial be k. Hence,

$$\|\tilde{R}c^\star\| = \|p_\star(\mathcal{A})(x_1 - x_0)\| \leq \|x_1 - x_0\| \min_{p:p(1)=1} \max_{\mathcal{A}:0\preccurlyeq\mathcal{A}\preccurlyeq\sigma I} \|p(\mathcal{A})\|_2,$$

where $p_\star$ is a polynomial with coefficient $c^\star$. The quantity, $\max_{\mathcal{A}:0\preccurlyeq\mathcal{A}\preccurlyeq\sigma I} \|p(\mathcal{A})\|_2$ can be further bounded by $\max_{0\leq\lambda\leq\sigma} |p(\lambda)|$, resulting an upper bound on $\|\tilde{R}c^\star\|$ as $\|x_1 - x_0\| \min_{p:p(1)=1} \max_{0\leq\lambda\leq\sigma} |p(\lambda)|$. Denote $C_k := \operatorname{argmin}_{p:p \text{ is monic}} \max_{x:x\in[-1,1]} |p(x)|$ be the Chebyshev polynomial of degree k. Following [11], Scieur et al. in [22] derived:

$$\min_{p:p(1)=1} \max_{0\leq\lambda\leq\sigma} |p(\lambda)| = \frac{2\xi^k}{1 + \xi^{2k}},$$

where $\xi = \frac{1-\sqrt{1-\sigma}}{1+\sqrt{1-\sigma}}$. This concludes bounding the quantity $\|\tilde{R}c^\star\|$. Based on this, the result can be further generalized in the following theorem.

Theorem 1. *Denote* $\hat{\xi} = (\sqrt{L}-\sqrt{\mu})/(\sqrt{L}+\sqrt{\mu})$. *Then the coefficients,* c_R *produced by RNA in Algorithm 2 follow:*

$$\|Xc_R - x^\star\| \leq \frac{2\hat{\xi}^k}{1+\hat{\xi}^{2k}}\|x_0 - x^\star\|.$$

Theorem 2. *For Algorithm 3, we have* $\|Xc_D - x^\star\| \leq \sqrt{\kappa}\frac{2\hat{\xi}^k}{1+\hat{\xi}^{2k}}\|x_0 - x^\star\|$. *where* $\kappa = \frac{L}{\mu}$.

Proof. On one hand, by using strong convexity of f, we have

$$\frac{\mu}{2}\|Xc_D - x^\star\|^2 \leq f(Xc_D) - f^*.$$

On the other hand, by definition of DNA we have

$$f(Xc_D) - f^* \leq f(Xc_R) - f^*.$$

In addition, we have

$$f(Xc_R) - f^* \leq \frac{L}{2}\|Xc_R - x^\star\|^2.$$

Therefore, by putting together these inequalities we get

$$\frac{\mu}{2}\|Xc_D - x^\star\|^2 \leq \frac{L}{2}\|Xc_R - x^\star\|^2.$$

By rearranging the terms we obtain the desired result. □

Remark 2. Anderson's acceleration is used to find solutions (via acceleration) to fixed point equations of the form: $x = f(x)$. DNA, on the other hand, finds solutions to linear fixed-point methods of the form: $x_{k+1} = Mx_k + b$, via acceleration. Therefore, DNA is a special case of Anderson's acceleration with a fixed window-size, $K+1$, in solving linear fixed-point problems. We note that, the convergence rate for DNA in Theorem 2 is the same as that for Krylov subspace methods up to a multiplicative scalar.

Remark 3. Interestingly, Walker and Ni in [28] showed that, for linear problems Anderson acceleration is equivalent to generalized minimal residual (GMRES); see Theorem 2.2. in [28]. Additionally, we note that, in an earlier work, Fang and Saad [10] showed the quasi-Newton methods fall in a broader family of Anderson acceleration method. But Walker and Ni were the first give this precise details. That is, the iterates that result from Anderson's acceleration, x_{k+1}^{AA} can be written as: $x_{k+1}^{\mathrm{AA}} = x_k^{\mathrm{GMRES}} + r_k^{\mathrm{GMRES}}$, where $r_k^{\mathrm{GMRES}} = b - (I - A)x_k^{GMRES}$ and $\{x_k^{\mathrm{GMRES}}\}$ be the sequence of iterates of GMRES. Additionally, we mention that, the objective acceleration technique proposed in [21] is a modification of nonlinear GMRES (N-GMRES). Toth and Kelley [27] provided the first convergence result for Anderson acceleration on general nonlinear problems.

Remark 4. Recently, Sterck and He [9] showed that, the standard Anderson's acceleration with window size, m (AA(m)), and with initial iterates, x_k, $k = 0, 1, \cdots, m$, defined recursively using AA(k), is a Krylov space methods (for example, conjugate gradient algorithm, GMRES, etc.); see Proposition 2.8 and 2.12 in [9]. Moreover, [9] showed that k iterations of AA(m) cannot produce a smaller residual than k iterations of GMRES without restart; see Proposition 2.10 in [9].

The above remarks establish that DNA, Anderson's acceleration, and Krylov space methods are closely related. More detailed investigation into these relations is left for future work.

However, numerically DNA is unstable like RNA without regularization. In fact, the matrix $X^\top R$ can be highly ill-conditioned and can lead to large errors in computing $c^\star$. Moreover, we accumulate errors in approximating the gradient via linearization as our approximation of the gradient is valid only in the neighborhood of the iterates $x_0, \ldots, x_K$.

Algorithm 4: DNA-1.

Input : Sequence of iterates $x_0, \ldots, x_{K+1}$; sequence of step sizes $\alpha_0, \ldots, \alpha_K$; and $\mathbb{1} \in \mathbb{R}^{K+1}$, a vector of all 1s;

1 Set $\tilde{R} = \left[\frac{x_0 - x_1}{\alpha_0}, \ldots, \frac{x_K - x_{K+1}}{\alpha_K}\right]$ and $X = [x_0, \ldots, x_K]$;

2 Solve the linear system for $z \in \mathbb{R}^{K+1}$: $X^\top \tilde{R} z = \mathbb{1}$;

3 Set $c = \frac{z}{z^\top \mathbb{1}} \in \mathbb{R}^{K+1}$;

Output : $x = \sum_{k=0}^{K} c_k x_k$.

2.2. Regularized DNA

To remedy the numerical instability of unconstrained DNA, we propose three regularized versions of DNA by using three different regularizers in the form of $g(c)$ and show that they work well in practice. But one can explore different forms of $g(c)$ as regularizer; their performance largely depends on the optimization problem. In the following, we explain them in details.

2.2.1. DNA-1

This regularized version of DNA is directly influenced by Scieur et al. [22]. Here, we generate the extrapolated point x as a linear combination of the set of $K+1$ iterates such that, $x = \sum_k c_k x_k$. Additionally, as in [22,23], we assume the sum of the coefficients c_k to be equal to 1. Therefore, for $c \in \mathbb{R}^{K+1}$ with sum of its elements equal to 1, we set $g(c) = 1_{\sum_i c_i = 1}$ in (4) and consider the following constrained problem:

$$\min_{c \in \mathbb{R}^{K+1}} f(Xc) + \lambda 1_S(c) = \min_{\substack{c \in \mathbb{R}^{K+1} \\ \sum_{k=0}^{K} c_k = 1}} f(Xc), \tag{10}$$

where $X = [x_0, \ldots, x_K]$. We call this version of DNA as DNA-1.

Lemma 2. *If the objective function f is quadratic and $X^\top \tilde{R}$ is nonsingular then $c = (X^\top \tilde{R})^{-1}\mathbb{1}/\delta$, where $\delta = \mathbb{1}^\top (X^\top \tilde{R})^{-1}\mathbb{1}$, and $\mathbb{1}$ is the vector of dimension $K+1$ with all the components equal to 1 and $\tilde{R} = [{}^{(x_0 - x_1)}/_{\alpha_0}, \ldots, {}^{(x_K - x_{K+1})}/_{\alpha_K}]$.*

Proof. The Lagrangian of the problem (6) is

$$L(c, \lambda) = h(c) + \lambda \left(\sum_{k=0}^{K} c_k - 1 \right),$$

where $\lambda > 0$ is the Lagrange multiplier. The first order optimality conditions are

$$\nabla L_x(x, \lambda) = X^\top \nabla f(Xc) + \lambda \mathbb{1} = 0 \tag{11}$$

$$\nabla L_\lambda(x, \lambda) = c^\top \mathbb{1} - 1 = 0. \tag{12}$$

For quadratic objective functions, the gradient is affine and because $\sum_{k=0}^{K} c_k = 1$ we have

$$\nabla f\left(Xc\right) = \sum_{k=0}^{K} c_k \nabla f\left(x_k\right). \tag{13}$$

By using the relation between the iterates of GD method we find $\nabla f\left(x_k\right) = \frac{x_k - x_{k+1}}{\alpha_k}$. By using the above expression in equation (13) we further get

$$\nabla f\left(Xc\right) = \sum_{k=0}^{K} c_k \frac{x_k - x_{k+1}}{\alpha_k} = \tilde{R}c. \tag{14}$$

Substituting (14) in the first optimality condition and solving for c we get $c = -\lambda \left(X^\top \tilde{R}\right)^{-1} \mathbb{1}$. Next, we use it in the second optimality condition and solve it for λ to find $\lambda = \frac{-1}{\mathbb{1}^\top (X^\top \tilde{R})^{-1} \mathbb{1}}$, and, therefore, the final expression for c is $c = \frac{\left(X^\top \tilde{R}\right)^{-1} \mathbb{1}}{\mathbb{1}^\top (X^\top \tilde{R})^{-1} \mathbb{1}}$. □

Similar to RNA, if $X^\top \tilde{R}$ is singular then c is not necessarily unique. Any c of the form $\frac{z}{z^\top \mathbb{1}}$, where z is a solution of $X^\top \tilde{R} z = \mathbb{1}$, is a solution of (10). DNA-1 is described in Algorithm 4.

Quantifying the difference between DNA and RNA Theorem 2 gives a general accelerated convergence result without any comparison between RNA and DNA. To get more insight about how DNA performs compare to RNA, we quantify the difference between them by using simple quadratic function. Denote the functional value obtained by DNA, DNA-1 (Algorithm 4), and RNA (Algorithm 2) at an extrapolated point as f_D, f_{D1} and f_R, respectively.

Let $f(x) = \frac{1}{2} x^\top A x$, where A is symmetric and positive definite. We know $\nabla f(x) = Ax$. By using the extrapolation we find the coefficients c_is such that $x = \sum_i c_i x_i = Xc$, where $X = [x_0 \;\; x_1 \;\; \cdots x_k]$ is a matrix generated by stacking k iterates as its column and $c \in \mathbb{R}^k$ is a vector of coefficients. We know $f(Xc) = \frac{1}{2} c^\top X^\top A X c$, for DNA $c_{\mathrm{D}} = 0$, and for DNA-1 $c_{\mathrm{D1}} = \frac{z}{\mathbb{1}^\top z}$ where $z = (X^\top \tilde{R})^{-1} \mathbb{1}$. Therefore, we find

$$f_D = 0, \;\; \text{and } f_{\mathrm{D1}} = \frac{\mathbb{1}^\top (\tilde{R}^\top X)^{-1} X^\top A X (X^\top \tilde{R})^{-1} \mathbb{1}}{2(\mathbb{1}^\top z)^2},$$

which for $\tilde{R} = [\nabla f(x_0) \;\; \nabla f(x_1) \;\; \cdots \nabla f(x_k)] = [Ax_0 \;\; Ax_1 \;\; \cdots Ax_k] = AX$ further reduces to

$$f_{\mathrm{D1}} = \frac{1}{2\mathbb{1}^\top (X^\top A X)^{-1} \mathbb{1}}.$$

Similarly, we find for RNA

Algorithm 5: DNA-2.

Input : Sequence of iterates $x_0, \ldots, x_{K+1}$; sequence of step sizes $\alpha_0, \ldots, \alpha_K$; regularizer $\lambda > 0$; and reference vector $y \in \mathbb{R}^{k+1}$;

1 Set $R = \left[\frac{x_0 - x_1}{\alpha_0} - \nabla f(0), \ldots, \frac{x_K - x_{K+1}}{\alpha_K} - \nabla f(0)\right]$ and $X = [x_0, \ldots, x_K]$;

2 Set c as a solution of the linear system $(X^\top R + \lambda X^\top X)z = \lambda X^\top y - X^\top \nabla f(0)$;

Output : $x = \sum_{k=0}^{K} c_k x_k$.

$$c_{\mathrm{R}} = \frac{(\tilde{R}^\top \tilde{R})^{-1}\mathbb{1}}{\mathbb{1}^\top (\tilde{R}^\top \tilde{R})^{-1}\mathbb{1}}.$$

Therefore,

$$f_{\mathrm{R}} = \frac{\mathbb{1}^\top (\tilde{R}^\top \tilde{R})^{-1} X^\top A X (\tilde{R}^\top \tilde{R})^{-1}\mathbb{1}}{2(\mathbb{1}^\top (X^\top A^2 X)^{-1}\mathbb{1})^2},$$

which further reduces to

$$f_{\mathrm{RNA}} = \frac{\mathbb{1}^\top (X^\top A^2 X)^{-1} X^\top A X (X^\top A^2 X)^{-1}\mathbb{1}}{2(\mathbb{1}^\top (X^\top A^2 X)^{-1}\mathbb{1})^2}.$$

We formalize the above in the following proposition.

Proposition 1. *Let $A \in \mathbb{R}^{n\times n}$ be symmetric and positive definite and $f(x) = \frac{1}{2}x^\top Ax$ be a quadratic objective function. Let $X = [x_0 \quad x_1 \quad \cdots x_k]$ be a matrix generated by stacking k iterates of GD to minimize f. Then the functional values of DNA, DNA-1, and RNA at the accelerated point are: $f_D = 0$, $f_{\mathrm{D1}} = \frac{1}{2\mathbb{1}^\top (X^\top A X)^{-1}\mathbb{1}}$, and $f_{\mathrm{R}} = \frac{\mathbb{1}^\top (X^\top A^2 X)^{-1} X^\top A X (X^\top A^2 X)^{-1}\mathbb{1}}{2(\mathbb{1}^\top (X^\top A^2 X)^{-1}\mathbb{1})^2}$, respectively.*

We conclude that for this simple objective function, DNA reaches the optimal solution after the first acceleration. Moreover, one can choose the matrix A such that f_R is arbitrary large, and this example shows that DNA may outperform RNA by a large margin. The comparison between DNA-1 and RNA on the previous example is given in the following lemma and theorem.

Lemma 3. *If the sequence of iterates $\{x_k\}$ are linearly independent then we have:*
(i) the matrices AX and $A^{\frac{1}{2}}X$ have full column ranks.
(ii) $(A^{1/2}X)^\dagger A^{-1/2}((AX)^\dagger)^\top = (X^\top A^2 X)^{-1}$.

Proof. (i) Since A is symmetric and positive definite, $\mathrm{rank}(A) = \mathrm{rank}(A^{1/2}) = n$. As the iterates $\{x_k\}$ are linearly independent, $X = [x_0, \ldots, x_K] \in \mathbb{R}^{n\times(K+1)}$ has full column rank. Therefore, the matrices AX and $A^{\frac{1}{2}}X$ have full column ranks.

(ii) We know if a matrix B is of full column rank then $B^\dagger = (B^\top B)^{-1}B^\top$. By using the above and (i) and we find

$$
\begin{aligned}
(A^{1/2}X)^{\dagger}A^{-1/2}((AX)^{\dagger})^{\top} &= (X^{\top}A^{1/2}A^{1/2}X)^{-1}X^{\top}A^{1/2}A^{-1/2} \\
&\quad ((X^{\top}AAX)^{-1}X^{\top}A)^{\top} \\
&= (X^{\top}AX)^{-1}X^{\top}((X^{\top}A^{2}X)^{-1}X^{\top}A)^{\top} \\
&\overset{(X^{\top}A^{2}X)^{\top}=X^{\top}A^{2}X}{=} (X^{\top}AX)^{-1}(X^{\top}AX)(X^{\top}A^{2}X)^{-1} \\
&= (X^{\top}A^{2}X)^{-1}.
\end{aligned}
$$

Hence the result. □

Lemma 3 is needed to prove the following Lemma.

Lemma 4. *We assume that the matrix $\tilde{R}$ has full column rank. With the notations used in Proposition 1, we have $f_{\mathrm{R}}/f_{\mathrm{D1}} = \|z\|_{A^{-1}}^{2}\|y\|^{2}\|z\|^{-4}$, where $z := (\tilde{R}^{\dagger})^{\top}\mathbb{1} = ((AX)^{\dagger})^{\top}\mathbb{1}$ and $y := ((A^{1/2}X)^{\dagger})^{\top}\mathbb{1}$. We have $y^{\top}A^{-1/2}z = z^{\top}z$; then, by using Cauchy-Schwarz inequality, we conclude that $\|z\|_{A^{-1}}\|y\|_{2} \geq y^{\top}A^{-1/2}z = \|z\|_{2}^{2}$ whence $f_{\mathrm{R}}/f_{\mathrm{D1}} \geq 1$.*

Proof. We have $\tilde{R} = AX$. Since A is symmetric and positive definite, it is invertible and $X = A^{-1}R$. Set $y = ((A^{1/2}X)^{\dagger})^{\top}\mathbb{1}$, and let $\tilde{R}^{\dagger}$ be the pseudo-inverse of R. Therefore, $\tilde{R}^{\dagger}$ can be computed as

$$\tilde{R}^{\dagger} = (\tilde{R}^{\top}\tilde{R})^{-1}\tilde{R}^{\top},$$

and $(\tilde{R}^{\dagger})^{\top}$ is

$$(\tilde{R}^{\dagger})^{\top} = \tilde{R}(\tilde{R}^{\top}\tilde{R})^{-1}.$$

We also note that $\tilde{R}^{\dagger}\tilde{R} = I_k$ and $\tilde{R}^{\top}(\tilde{R}^{\dagger})^{\top} = I_k$, where I_k is an identity matrix of size k. Therefore, we have

$$
\begin{aligned}
2f_{\mathrm{D1}} &= \frac{1}{\mathbb{1}^{\top}(X^{\top}AX)^{-1}\mathbb{1}} \\
&= \frac{1}{\mathbb{1}^{\top}(X^{\top}A^{1/2}A^{1/2}X)^{\dagger}\mathbb{1}}.
\end{aligned}
$$

Since, $(X^{\top}A^{1/2})^{\top} = A^{1/2}X$, by using the property of pseudo-inverse, we can write

$$(X^{\top}AX)^{-1} = (X^{\top}A^{1/2}A^{1/2}X)^{\dagger} = (A^{1/2}X)^{\dagger}((A^{1/2}X)^{\dagger})^{\top}$$

and the above expression becomes

$$2f_{\mathrm{D1}} = \frac{1}{\mathbb{1}^{\top}(A^{1/2}X)^{\dagger}((A^{1/2}X)^{\dagger})^{\top}\mathbb{1}} \overset{y=((A^{1/2}X)^{\dagger})^{\top}\mathbb{1}}{=} \frac{1}{y^{\top}y} = \frac{1}{\|y\|_{2}^{2}}. \tag{15}$$

Similarly we find, $(X^\top A)^\top = AX$, and again by using the property of pseudo-inverse, we can write

$$(X^\top A^2 X)^{-1} = (X^\top AAX)^\dagger = (AX)^\dagger((AX)^\dagger)^\top$$

and

$$\begin{aligned} 2f_{\mathrm{R}} &= \frac{\mathbb{1}^\top (AX)^\dagger((AX)^\dagger)^\top X^\top AX(AX)^\dagger((AX)^\dagger)^\top \mathbb{1}}{(\mathbb{1}^\top (AX)^\dagger((AX)^\dagger)^\top \mathbb{1})^2} \\ &\overset{AX=\tilde{R}}{=} \frac{\mathbb{1}^\top \tilde{R}^\dagger(\tilde{R}^\dagger)^\top(\tilde{R}^\dagger)^\top A^{-1}\tilde{R}\tilde{R}^\dagger(\tilde{R}^\dagger)^\top \mathbb{1}}{(\mathbb{1}^\top \tilde{R}^\dagger(\tilde{R}^\dagger)^\top \mathbb{1})^2} \\ &= \frac{\mathbb{1}^\top \tilde{R}^\dagger A^{-1}(\tilde{R}^\dagger)^\top \mathbb{1}}{(\mathbb{1}^\top (\tilde{R}^\top \tilde{R})^{-1}\mathbb{1})^2} \\ &\overset{z:=\tilde{R}^\dagger \mathbb{1}}{=} \frac{z^\top A^{-1} z}{(z^\top z)^2}. \end{aligned}$$

Therefore,

$$2f_{\mathrm{R}} = \frac{z^\top A^{-1} z}{(z^\top z)^2} = \frac{\|z\|^2_{A^{-1}}}{\|z\|_2^4}. \tag{16}$$

Combining (15) and (16) we obtain the ratio between f_R and f_D.

From Lemma 3 we have, $y^\top A^{-1/2} z = z^\top z$. Then by using Cauchy Swartz inequality we conclude that $\|z\|_{A^{-1}}\|y\|_2 \geq y^\top A^{-1/2} z = \|z\|_2^2$ whence $\frac{f_{\mathrm{R}}}{f_{\mathrm{D1}}} \geq 1$. □

Note that, the ratio $\frac{f_{\mathrm{R}}}{f_{\mathrm{D1}}} \geq 1$ can be directly concluded from the definition of f_{R} and f_{D1}. The main goal of the previous lemma is to exactly quantify the ratio between these two quantities. The following theorem gives more insight.

Theorem 3. *We have* $\frac{f_{\mathrm{R}}}{f_{\mathrm{D1}}} \leq U_R := \|z\|^2_{A^{-1}}\|z\|^2_A\|z\|^{-4}$, *and* $U_R \in [1/2 + \kappa(A)/2, \kappa(A)]$, *where* $\kappa(A)$ *is the condition number of* A.

Proof. Recall that $\frac{f_{\mathrm{R}}}{f_{\mathrm{D1}}} = \frac{\|z\|^2_{A^{-1}}\|y\|_2^2}{\|z\|_2^4}$. Also recall that $z := (\tilde{R}^\dagger)^\top \mathbb{1} = ((AX)^\dagger)^\top \mathbb{1}$ and $y := ((A^{1/2}X)^\dagger)^\top \mathbb{1}$. Therefore, $A^{1/2}z$ is the minimum norm solution to the linear system: $X^\top A^{1/2}A^{1/2}z = \mathbb{1}$ and similarly, y is the minimum norm solution to the linear system: $X^\top A^{1/2} y = \mathbb{1}$. By using the above fact, we find $\|y\|_2 \leq \|A^{1/2}z\|_2 = \|z\|_A$ and we can rewrite the ratio as:

$$\frac{f_{\mathrm{R}}}{f_{\mathrm{D1}}} = \frac{\|z\|^2_{A^{-1}}\|y\|_2^2}{\|z\|_2^4} \overset{y=A^{1/2}z}{=} \frac{\|z\|^2_{A^{-1}}\|z\|^2_A}{\|z\|_2^4}. \tag{17}$$

From (17) the quantity $\max\limits_{z\neq 0} \frac{\|z\|^2_{A^{-1}}\|z\|^2_A}{\|z\|_2^4}$ is equivalent to $\max\limits_{\|z\|_2=1} \|z\|^2_{A^{-1}}\|z\|^2_A \leq \max\limits_{\|z\|_2=1} \|z\|^2_{A^{-1}} \max\limits_{\|z\|_2=1} \|z\|^2_A := U_R$.

Note that, $U_R \leq \lambda_{\max}(A^{-1})\lambda_{\max}(A) = \frac{\lambda_{\max}(A)}{\lambda_{\min}(A)} = \kappa(A)$. Let $U\Sigma U^\top$ be an eigen decomposition of A then $U_R = \max_{\|z\|_2=1} \|z\|^2_{\Sigma^{-1}} \max_{\|z\|_2=1} \|z\|^2_{\Sigma}$. By considering a vector with $1/\sqrt{2}$ at the first and last position and zero everywhere else we conclude that

$$\begin{aligned} U_R &\geq \left(\frac{1}{2\lambda_{\max}(A)} + \frac{1}{2\lambda_{\min}(A)}\right)\left(\frac{1}{2\lambda_{\min}(A)} + \frac{1}{2\lambda_{\max}(A)}\right) \\ &\geq \frac{2+\kappa(A)}{4}. \quad \square \end{aligned}$$

The above theorem tells us, for a simple quadratic function, the ratio of the objective function values of DNA-1 and RNA may attain an order of $\kappa(A)$, but it never exceeds it. The theoretical quantification of the acceleration obtained by DNA and its different versions compared to RNA in more general problems is left for future work. Although DNA-1 can be seen as a regularized version of DNA, we still need to remedy the fact that the linearization of the gradient is not a *good* approximation in the entire space, and that the matrix $X^\top \tilde{R}$ may be singular. To this end, we impose some regularization such that the new extrapolated point *stays* near to some reference point. We propose two different ways in the following two sections.

2.2.2. DNA-2

We set $g(c) = \|Xc - y\|^2$ in (4) and consider a regularized version of problem (7):

$$\min_{c\in\mathbb{R}^{K+1}} f(Xc) + \frac{\lambda}{2}\|Xc - y\|^2, \tag{18}$$

where $\lambda > 0$ is a balancing parameter and y is a reference point (a point supposed to be in the neighborhood of Xc). By taking the derivative of the objective in (18) with respect to c and setting it to 0, we find $X^\top \nabla f(Xc) + \lambda X^\top(Xc - y) = 0$, which after using the approximation (8) becomes $X^\top(Rc + \nabla f(0)) + \lambda X^\top(Xc - y) = 0$. Finally, $c^\star$ is given as a solution to the linear system

$$(X^\top R + \lambda X^\top X)c = \lambda X^\top y - X^\top \nabla f(0). \tag{19}$$

In general, $X^\top R$ is not necessarily symmetric. To justify the regularization further, one might symmetrize $X^\top R$ by its transpose. In our experiments, we obtained good performance without this.

We call this method DNA-2 (see Algorithm 5). Note that $X^\top R + \lambda X^\top X$ can be singular, especially near the optimal solution. To remedy this, we propose either to add another regularization to the problem (18), or to consider a direct regularization on c instead of Xc. We explain this next.

Algorithm 6: DNA-3.

Input : Sequence of iterates $x_0, \ldots, x_{K+1}$; sequence of step sizes $\alpha_0, \ldots, \alpha_K$; regularizer $\lambda > 0$; and $e \in \mathbb{R}^{k+1}$;

1 Set $R = \left[\frac{x_0 - x_1}{\alpha_0} - \nabla f(0), \ldots, \frac{x_K - x_{K+1}}{\alpha_K} - \nabla f(0)\right]$ and $X = [x_0, \ldots, x_K]$;

2 Set c as a solution of the linear system $(X^\top R + \lambda I)z = \lambda e - X^\top \nabla f(0)$;

Output : $x = \sum_{k=0}^{K} c_k x_k$.

2.2.3. DNA-3

We set $g(c) = \|c - e\|^2$ in (4) and consider a regularized version of (7) as

$$\min_{c \in \mathbb{R}^{K+1}} f(Xc) + \frac{\lambda}{2}\|c - e\|^2, \tag{20}$$

where $\lambda > 0$ and e is a reference point for c. By taking the derivative with respect to c and setting it to 0, we find $X^\top \nabla f(Xc) + \lambda(c - e) = 0$, which after using the approximation (8) becomes $X^\top(Rc + \nabla f(0)) + \lambda X^\top(c - e) = 0$. Therefore, $c^\star$ is given as a solution to the linear system: $(X^\top R + \lambda I)c = \lambda e - X^\top \nabla f(0)$. We call this method DNA-3, and describe it in Algorithm 6.

A formal algorithm to summarize the results In the previous sections, we present three different versions of DNA. For better clarity, we present an unified framework in Algorithm 7 that shows how the acceleration via DNA (both constrained and regularized versions) is interleaved with the GD updates. Additionally, we note that multiplying the function, f by a factor, β, makes the linear system of standard AA, β^2 larger, while the linear system of DNA is multiplied by a factor β.

3. Numerical illustration

We evaluate our techniques and compare against RNA and GD by using both synthetic data as well as real-world datasets. Overall, we find that DNA outperforms RNA in most settings by large margins.

Experimental setup. Our experimental setup comprises of 3 typical problems, least squares, ridge regression, and logistic regression, for which the optimal solution $x^\star$ is either known or can be evaluated using a numerical solver. We apply the online acceleration scheme in [22] and compare 3 versions of DNA against RNA and GD. Our results show the difference between the functional values at the extrapolated point and at the optimal solution on a logarithmic scale (the lower the better), as the iterations progress. The primary objective of our simulations is to show the effectiveness of DNA and its different versions to accelerate a converging, deterministic optimization algorithm. Therefore, we do not report any computation time of the algorithms and we do not claim these implementations are optimized. Note that the computation bottleneck of all algorithms (including RNA) is solving the linear system to calculate c, and because the dimensionality of the linear systems is the same in RNA and DNA, the extra cost is

Algorithm 7: DNA—A unified framework.

Input : Initial point, $x_0 \in \mathbb{R}^n$ and stepsize sequence $\{\alpha_i\}$, regularizer $\lambda > 0$, reference vector, $y \in \mathbb{R}^{K+1}$ or $e \in \mathbb{R}^{K+1}$ (for DNA-2 and 3, respectively);
for $i = 1, 2 \cdots, K+1$ **do**
  $x_i = x_{i-1} - \alpha_i \nabla f(x_{i-1})$;
end
while *not convergent* **do**
  Initialize: Sequence of iterates, $\{x_i\}_{i=1}^{K+1}$; sequence of step sizes, $\{\alpha_i\}_{i=1}^{K}$; regularizer $\lambda > 0$ (if using unconstrained variants–DNA-2 and 3); and a reference vector, $y \in \mathbb{R}^{K+1}$ or $e \in \mathbb{R}^{K+1}$ (for DNA-2 and 3, respectively);
  Set $X = [x_0, \ldots, x_K]$;
  Calculate $\tilde{R}$ via Step 1 of Algorithm 4; or calculate R via Step 1 of Algorithm 3, 5, or 6;
  Set c as a solution of the linear system via Step 2 of Algorithm 3, 4, 5, or 6;
  Calculate: $\hat{x} = \sum_{k=0}^{K} c_k x_k$;
  Set: $\hat{x} = x_0$;
  for $i = 1, 2 \cdots, K+1$ **do**
    $x_i = x_{i-1} - \alpha_i \nabla f(x_{i-1})$;
  end
end

the same in both approaches. In our experiments, we consider a fixed stepsize $\alpha_k = 1/L$ for GD, where L is the Lipschitz constant of ∇f. We note that for DNA-1 and 2 we need to use the stepsize explicitly to construct R as defined in Lemma 1.

Least squares. We consider a least squares regression problem of the form

$$\min_x f(x) := \frac{1}{2}\|Ax - y\|^2, \tag{21}$$

where $A \in \mathbb{R}^{m\times n}$ with $m > n$ is the data matrix, $y \in \mathbb{R}^m$ is the response vector. For $m > n$ and $\operatorname{rank}(A) = n$, the objective function f in (21) is strongly convex. The optimal solution $x^\star$ to (21) is given by $x^\star = \operatorname{argmin}_x f(x) = (A^\top A)^{-1} A^\top y$. For least squares we only consider the overdetermined systems, that is, $m > n$.

Ridge regression. The classic ridge regression problem is of the form:

$$\min_x f(x) := \frac{1}{2}\|Ax - y\|^2 + \frac{1}{2n}\|x\|^2, \tag{22}$$

where $A \in \mathbb{R}^{m\times n}$ is the data matrix, $y \in \mathbb{R}^n$ is the response vector. The optimal solution $x^\star$ to (22) is given by $x^\star = \operatorname{argmin}_x f(x) = (A^\top A + \frac{1}{2n} I)^{-1} A^\top y$.

Logistic regression. In logistic regression with ℓ_2 regularization, the objective function $f(x)$ is the summation of n loss function of the form:

$$f_i(x) = \log(1 + \exp(-y_i \langle A(:, i), x\rangle) + \frac{1}{2m}\|x\|^2. \tag{23}$$

We use the MATLAB function `fminunc` to numerically obtain the minimizer of f in this case.

Synthetic data. To compare the performance of different methods under different acceleration schemes, we are interested in the case where matrix A has a known singular

value distribution and we consider the cases where A has varying condition numbers. We note that the condition number of A is defined as $\kappa(A) := {}^{\lambda_{\max}(A)}/_{\lambda_{\min}(A)}$, where λ is the eigenvalue of A. We first generate a random matrix and let $U\Sigma V^\top$ be its SVD. Next we create a vector $S \in \mathbb{R}^{\min\{m,n\}}$ with entries $s_i \in \mathbb{R}^+$ arranged in an nonincreasing order such that s_1 is maximum and $s_{\min\{m,n\}}$ is minimum. Finally, we form the test matrix A as $A = U\text{diag}(s_1 \;\; s_2 \;\; \cdots s_{\min\{m,n\}})V^\top$ such that A will have a higher condition number if ${}^{s_1}/_{s_{\min\{m,n\}}}$ is large and smaller condition number if ${}^{s_1}/_{s_{\min\{m,n\}}}$ is small. We create the vector y as a random vector.

Real data. We use 15 different real-world datasets from the `LIBSVM` repository [8]. We set apart the datasets with y-labels as $\{-1, 1\}$ for Logistic regression and used the remaining 12 multi-label datasets for least squares and ridge regression problems. We use the matrix A in its crude form, that is, without any scaling/normalizing or centralizing its rows or columns.

Acceleration results. We use GD as our baseline algorithm and the online acceleration scheme as explained in [22] for both RNA and DNAs to accelerate the GD iterates. For synthetic data (see Fig. 1), we see that for smaller condition numbers and for quadratic objective functions, DNA-1 and RNA has almost similar performance and DNA-2 and 3 show faster decrease of $f(x_k) - f(x^\star)$, but all of them are very competitive. As the condition number of the problems becomes huge, DNA-2 and 3 outperform RNA by large margins. However, for logistic regression problems we see performance gains for all versions of DNA compared to RNA. Though for huge condition numbers, for logistic regression problems, the performance of DNA-2 depends on the hyperparameter λ.[2] We argue with experimental evidence as in Fig. 1 that for huge condition numbers the sensitivity of the performance of DNA-2 is problem specific. We use an additional regularizer $\epsilon\|c\|^2$, where $\epsilon \approx 10^{-14}$ to find a stable solution to (19) of DNA-2. Next, on real-world datasets in Figs. 2 and 3, we see that all versions of DNA outperform RNA, except in a few cases, where RNA and DNA-1 have almost similar performance. We indicate the oscillating nature of DNA-2 in some plots is due to its problem-specific sensitivity to the regularizer. In Fig. 4, we find for logistic regression problems on real datasets, DNA outperfoms RNA. We owe the success of DNA on non-quadratic problems to its adaptive gradient approximation. We note that the performance of all algorithms on the offline scheme of [22] is similar to online scheme of [22]. However, on the second online scheme used in [23], all the algorithms perform extremely poorly. Therefore, we do not report the results in this paper.

Solving the linear system. The linear system that arises within DNA is very small. The coefficient matrix is of dimension $(K+1) \times (K+1)$, where K is generally small. In our experiments, K is set to $3, 5$ or 6. Since one can view the (vector) extrapolation technique (RNA or DNA) as some type of a restarting scheme, K never grows larger as the iteration progresses and the computational cost in finding the extrapolation coeffi-

[2] We note that, multiplying the function, f by a factor, β, makes the linear system of standard AA, β^2 larger, while the linear system of DNA is multiplied by a factor β.

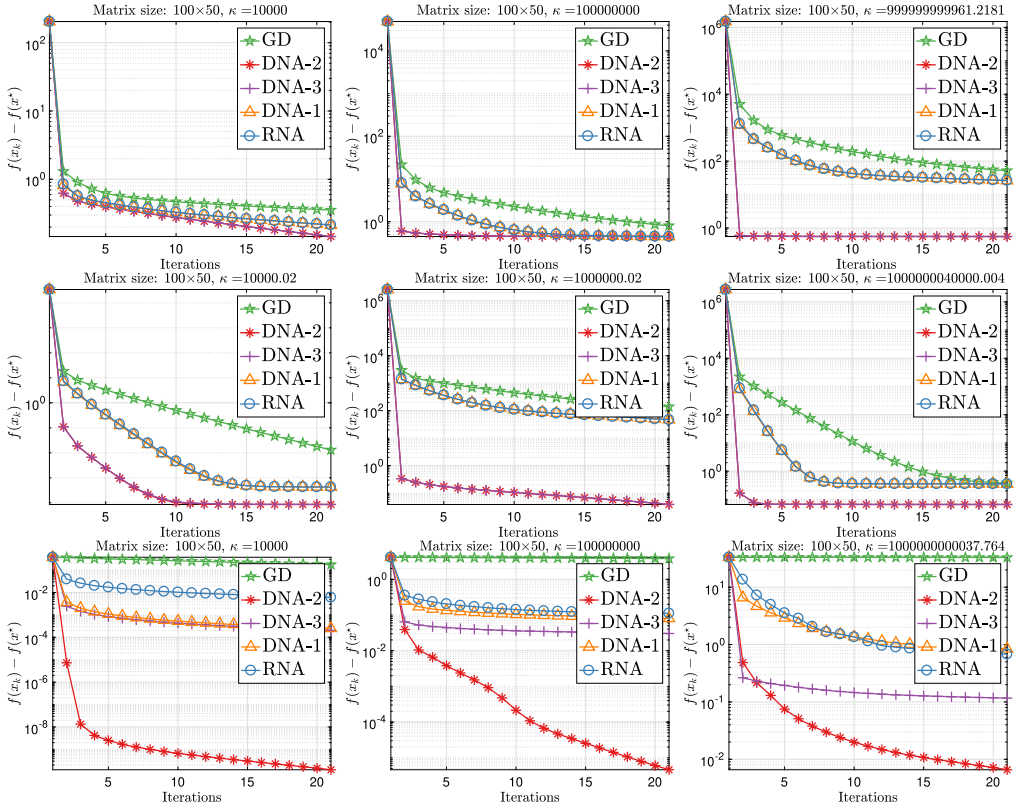

Fig. 1. Acceleration on synthetic data by using online acceleration scheme in [22]. First and second row represent quadratic, strong convex objective function as Least Squares and Ridge Regression, respectively. The last row represents non-quadratic but strong convex objective function as Logistic Regression. For all plots we use $k = 3$. For RNA, we have $\lambda = 10^{-8}$; for DNA, we set $\lambda = 10^{-8}$, except for the last LR plot where for DNA-2, we set $\lambda = 10$.

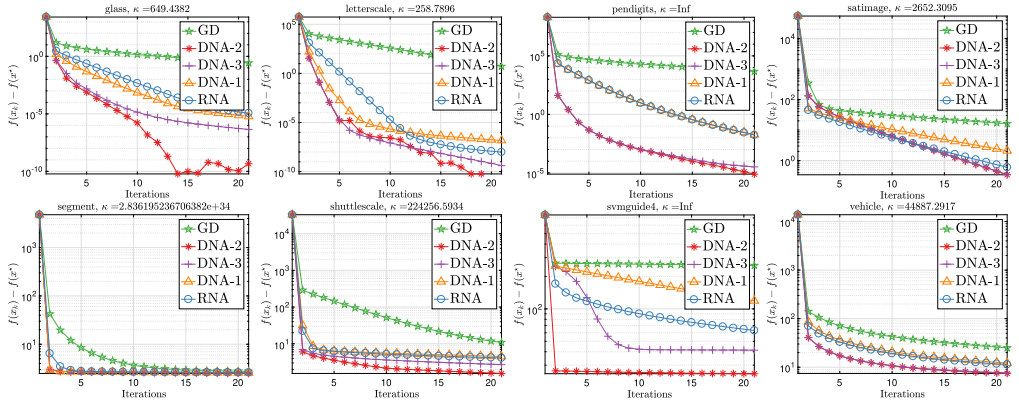

Fig. 2. Acceleration on `LIBSVM` dataset by using online acceleration scheme in [22] on Least Squares problems. For all datasets, we use $k = 3$. For RNA and DNA, we set $\lambda = 10^{-8}$.

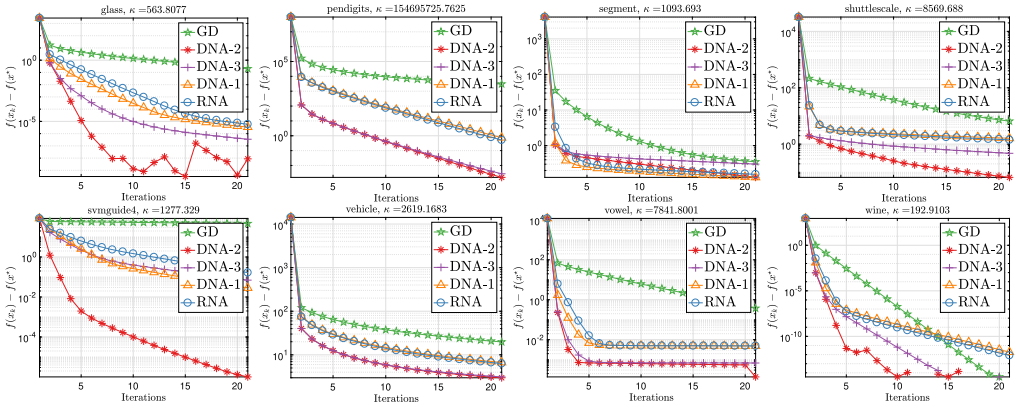

Fig. 3. Acceleration on LIBSVM dataset by using online acceleration scheme in [22] on Ridge Regression problems. For all datasets, we use $k = 3$. For RNA and DNA, we set $\lambda = 10^{-8}$.

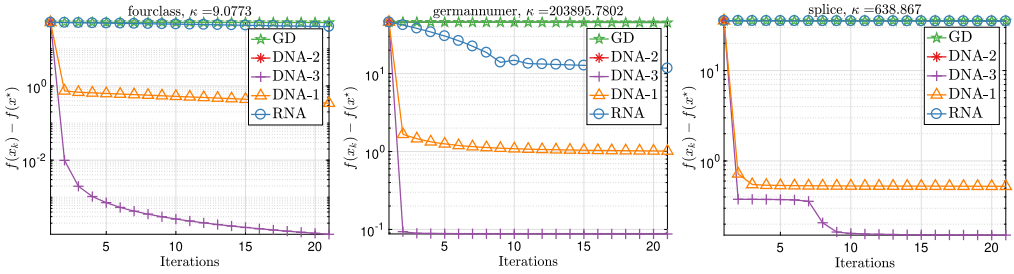

Fig. 4. Acceleration on LIBSVM dataset by using online acceleration scheme in [22] on Logistic Regression problems. For all datasets, we use $k = 3$. For RNA and DNA, we set $\lambda = 10^{-8}$.

cients remains very modest. For instance, the solution of the linear system can be done by using Gaussian elimination or by using an iterative method. It takes at most $O(K^3)$ flops but as K is very small, practically it only adds a negligible computation overhead compared to a full gradient calculation in higher dimensions.

Effect of different K. Note that, K is a hyper parameter of the algorithms. For larger K, we observe the better the performance of the algorithms. It is natural because larger K implies more information from the oracle algorithm. In Fig. 5, we see that for $K = 3$, RNA and DNA-1 have similar performances, but when K increases RNA is better than DNA-1. Fig. 6 shows similar performances for DNA-2 and 3 for small K and DNA-2 becomes better for larger values of K. DNA-1 and RNA have almost similar performances for all the values of K. An in-depth study of the impact of this parameter is left to future work.

Comparison with Nesterov's acceleration. We performed some preliminary experiments (see Fig. 7), where we compared DNA with Nesterov's acceleration (NA) on logistic regression problems. DNA outperforms NA in all cases. Our preliminary insight is that NA is based only on the last two iterations while DNA supports more than two iterates.

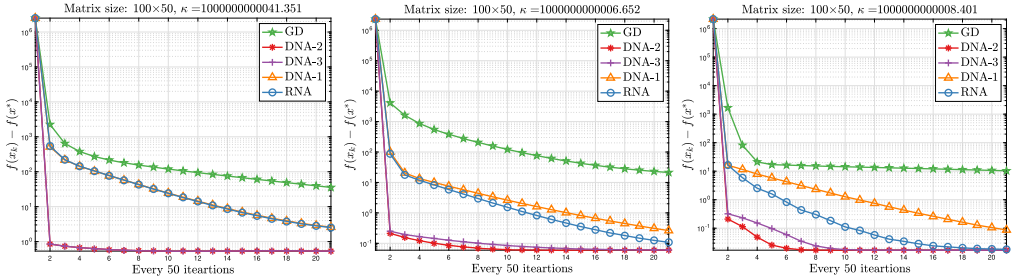

Fig. 5. Acceleration on synthetic dataset by using online acceleration scheme in [22] on Least Squares problems. For all datasets, we use $K = 3$, $K = 10$ and $K = 15$ respectively. We set $\lambda = 10^{-8}$.

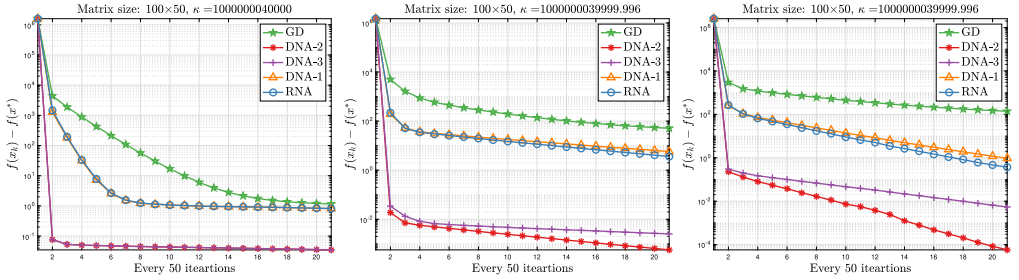

Fig. 6. Acceleration on synthetic dataset by using online acceleration scheme in [22] on Ridge Regression problems. For all datasets, we use $K = 3$, $K = 8$ and $K = 15$ respectively. We set $\lambda = 10^{-8}$.

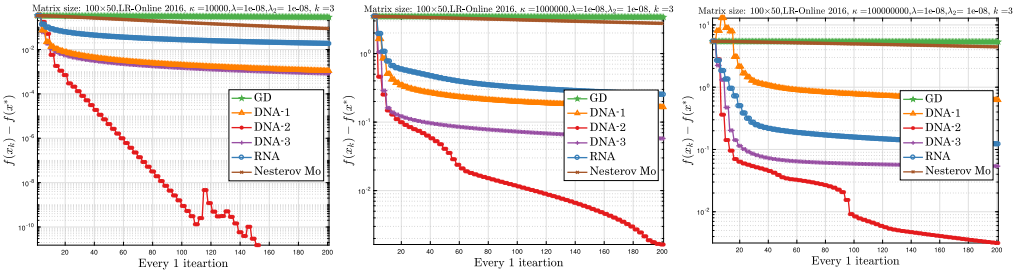

Fig. 7. Acceleration on synthetic data to solve logistic regression (LR) problem by using online acceleration scheme. For all plots we use $k = 3$. In all cases, RNA and DNA outperform Nesterov's acceleration. For RNA and DNA, we set $\lambda = 10^{-8}$.

However, this needs a thorough investigation (both in theory and in practice) and this is not the scope of this paper.

Application to the non-convex world: accelerating neural network training. Modern deep learning requires optimization algorithms to work in a nonconvex setup. Although this is not the main goal of this paper, nevertheless, we implement our acceleration techniques for training neural networks and obtain surprisingly promising results. We only use DNA-1 for experiments in this section. Tuning the hyperparameter λ for the other versions of DNAs requires more time, and we leave this for future research. The `Pytorch` implementation of RNA is based on [23].

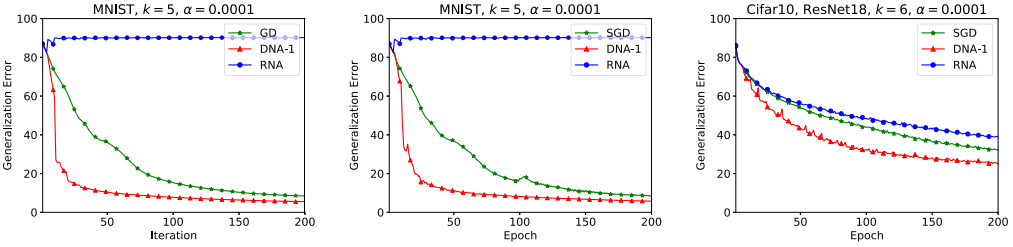

Fig. 8. Accelerating neural-network training. (a) A 2-layer neural network with GD optimizer and fixed stepsize 0.0001. (b) A 2-layer neural network with SGD optimizer and fixed stepsize 0.0001. For both, we use $k = 5$. (c) ResNet18 on CIFAR10 dataset with SGD optimizer and fixed stepsize 0.0001. We use $k = 6$. Note that these are not the best stepsize setting for the networks. Codebase `Pytorch`.

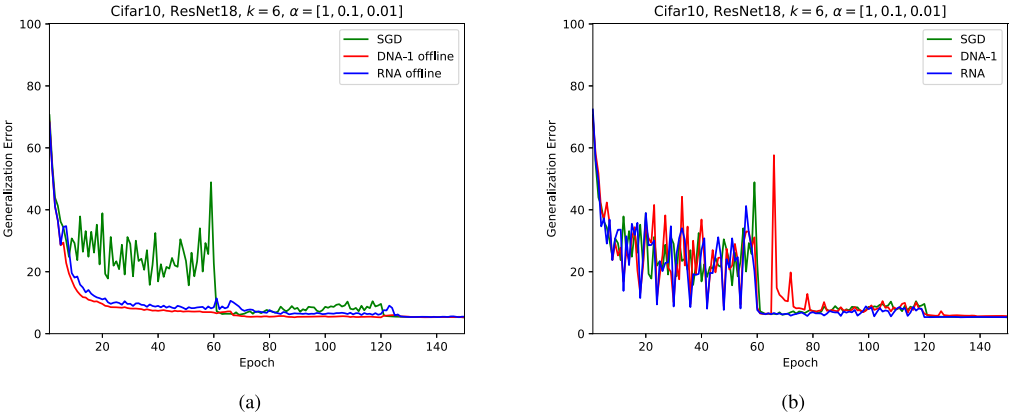

Fig. 9. Acceleration on Neural Network. (a) Experiment implementing ResNet18 on Cifar10 dataset with SGD as training algorithm with decaying stepsize across the epochs. For both DNA-1 and RNA, the window size is set to $k = 6$, and we use the offline scheme of [22]. (b) Experiment implementing ResNet18 on Cifar10 dataset with SGD as training algorithm with decaying stepsize across the epochs. For both DNA-1 and RNA, the window size is set to $k = 6$, and we use the online scheme of [22]. In both cases, RNA and DNA-1 fail to accelerate the SGD iterates.

MNIST classification. First, we trained a simple two-layer neural network classifier on MNIST dataset [15] via GD and accelerate the GD iterates via the online scheme in [22] for both RNA and DNA-1. The two-layer neural network is widely adopted in most tutorials that use MNIST dataset.[3]

In Fig. 8 (a), DNA-1 gains acceleration by using GD iterates with a window size $k = 5$. However, RNA fails to accelerate the GD iterates. This motivated us to train the same network on MNIST dataset classification [15] via SGD as baseline algorithm and accelerate the SGD iterates via the online scheme in [22] for both RNA and DNA-1 (as in Fig. 8 (b)). Again, with window size $k = 5$, DNA-1 achieves better acceleration than RNA.

[3] https://github.com/pytorch/examples/blob/master/mnist/main.py.

ResNet18 on CIFAR10. Finally, we train the ResNet18 network [12] on CIFAR10 dataset [14] by SGD. Each epoch of SGD consists of multiple iterations and each iteration applies to 128 training samples. The size of the training set is 5×10^4 and the size of validation set is 10^4. Each sample is a 32×32 resolution color image and they are categorized into 10 classes. We accelerate the SGD iterates via the online scheme in [22] for both RNA and DNA-1. Again DNA-1 outperforms RNA in lowering the generalization error of the network (see Fig. 8 (c)).

Effect of decaying step-size. Finally, in Fig. 9 we show training ResNet18 on Cifar10 dataset with SGD as training algorithm with decaying stepsize across the epochs. In both cases, RNA and DNA-1 fail to accelerate the SGD iterates. This indicates the fact that the stepsize is an important hyperparameter, and one needs to further explore it in case of accelerating a neural network training.

Declaration of competing interest

There is no possible conflicts of interest.

Acknowledgement

Aritra Dutta acknowledges being an affiliated researcher at the Pioneer Centre for AI, Denmark.

Appendix A. Acceleration schema

In this section, we explain different acceleration schema used by Scieur et al. in [22,23] for completeness.

Remark 5. For GD, the updates of the iterates are done via the simple update rule (2), which is explained in Fig. 10(a).

A.0.1. Online Scheme 1

We explain the acceleration scheme proposed in [22] herein. First, we run k iterations of GD to produce the sequence of iterates $\{x_i\}_{i=1}^k$ and then use extrapolation to generate a new point x'_k. We use x'_k as the initial point of GD and produce a set of next k iterates via GD. At this end, we further use the extrapolation scheme to produce a second offline update x'_{k+1} which is used as next the initial point of GD, and this process continues. See Fig. 10(b).

A.0.2. Online Scheme 2

The acceleration scheme proposed in [23], is more involved than the one proposed in [22]. First, we run k iterations of GD to produce a sequence of iterates $\{x_i\}_{i=1}^k$ and then use extrapolation to generate x'_k. Next, we use x'_k as the starting point of GD to

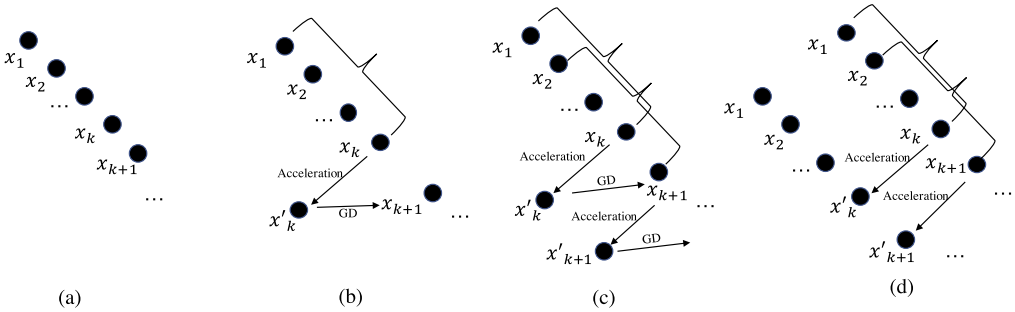

Fig. 10. Updates via: (a) gradient descent, (b) online extrapolation on gradient descent [22], (c) online extrapolation on gradient descent [23], and (d) offline scheme.

produce x_{k+1}. Now, we start from the second iterate x_2 and consider a set of k iterates $\{x_2, x_3, \cdots, x_k, x_{k+1}\}$ to produce the second offline update x'_{k+1} via extrapolation which is to be used as the next starting point of GD, and this process continues. See Fig. 10(c).

A.0.3. Offline scheme

Lastly, we describe an offline update scheme, as illustrated in Fig. 10(d). First, we run the GD to produce the sequence of iterates $\{x_k\}$ and then use the acceleration on the set of first k iterates to produce the first offline update x'_k and concatenate it with the previous $(k-1)$ GD updates. Next, we start from the second iterate x_2 and consider a set of k iterates to produce the second offline update x'_{k+1} via acceleration and this process continues. As a result, the offline accelerated updates are generated as $\{x_1, x_2 \cdots, x'_k, x'_{k+1}, \cdots\}$.

Appendix B. Reproducible research

See the `LIBSVM` dataset from the repository online: https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/. See the source code of RNA from: https://github.com/windows7lover/RegularizedNonlinearAcceleration. For `MATLAB` and `Pytorch` code that is used to produce all the results for our DNA, please email the authors.

References

[1] A.C. Aitken, On Bernoulli's numerical solution of algebraic equations, Proc. R. Soc. Edinb. 46 (1927) 289–305.
[2] D.G. Anderson, Iterative procedures for nonlinear integral equations, J. ACM 12 (4) (1965) 547–560.
[3] A. Beck, M. Teboulle, A fast iterative shrinkage thresholding algorithm for linear inverse problems, SIAM J. Imaging Sci. 2 (1) (2009) 183–202.
[4] R. Bollapragada, D. Scieur, A. d'Aspremont, Nonlinear acceleration of primal-dual algorithms, in: International Conference on Artificial Intelligence and Statistics (AISTATS), 2019, pp. 739–747.
[5] C. Brezinski, M. Redivo-Zaglia, Y. Saad, Shanks sequence transformations and Anderson acceleration, SIAM Rev. 60 (3) (2018) 646–669.
[6] Sébastien Bubeck, Yin Tat Lee, Mohit Singh, A geometric alternative to Nesterov's accelerated gradient descent, CoRR, arXiv:1506.08187 [abs], 2015.

[7] S. Cabay, L.W. Jackson, A polynomial extrapolation method for finding limits and antilimits of vector sequences, SIAM J. Numer. Anal. 13 (5) (1976) 734–752.
[8] C.C. Chang, C.J. Lin, LIBSVM: a library for support vector machines, ACM Trans. Intell. Syst. Technol. (2011).
[9] Hans De Sterck, Yunhui He, Anderson acceleration as a Krylov method with application to asymptotic convergence analysis, arXiv preprint, arXiv:2109.14181, 2021.
[10] Haw-ren Fang, Yousef Saad, Two classes of multisecant methods for nonlinear acceleration, Numer. Linear Algebra Appl. 16 (3) (2009) 197–221.
[11] Gene H. Golub, Richard S. Varga, Chebyshev semi-iterative methods, successive overrelaxation iterative methods, and second order Richardson iterative methods, Numer. Math. 3 (1) (1961) 157–168.
[12] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770–778.
[13] C.T. Kelley, Numerical methods for nonlinear equations, Acta Numer. 27 (2018) 207–287.
[14] A. Krizhevsky, G. Hinton, Learning multiple layers of features from tiny images, Technical report, University of Toronto, 2009, 1(4).
[15] Y. LeCun, C. Cortes, C.J.C. Burges, MNIST handwritten digit database, 2010, http://yann.lecun.com/exdb/mnist, 2010.
[16] H. Lin, J. Mairal, Z. Harchaoui, A universal catalyst for first-order optimization, in: Proceedings of Neural Information Processing Systems, 2015, pp. 3384–3392.
[17] Y. Nesterov, A method of solving a convex programming problem with convergence rate $o(1/k^2)$, Sov. Math. Dokl. 27 (2) (1983) 372–376.
[18] Y. Nesterov, Gradient methods for minimizing composite objective function, CORE Discussion Papers, 2007.
[19] Y. Nesterov, Gradient methods for minimizing composite functions, Math. Program. 140 (1) (2013) 125–161.
[20] B. Polyak, Some methods of speeding up the convergence of iteration methods, USSR Comput. Math. Math. Phys. 4 (5) (1964) 1–17.
[21] A.N. Riseth, Objective acceleration for unconstrained optimization, Numer. Linear Algebra Appl. 26 (1) (2019) e2216.
[22] D. Scieur, A. d'Aspremont, F. Bach, Regularized nonlinear acceleration, in: Proceedings of Neural Information Processing Systems, 2016, pp. 712–720.
[23] D. Scieur, E. Oyallon, A. d'Aspremont, F. Bach, Nonlinear acceleration of deep neural networks, arXiv:1805.09639, 2018.
[24] D. Shanks, Non-linear transformations of divergent and slowly convergent sequences, J. Math. Phys. 34 (1) (1955) 1–42.
[25] Hans De Sterck, Steepest descent preconditioning for nonlinear gmres optimization, Numer. Linear Algebra Appl. 20 (3) (2013) 453–471.
[26] W. Su, S. Boyd, E. Candés, A differential equation for modeling Nesterov's accelerated gradient method: theory and insights, in: Proceedings of Neural Information Processing Systems, 2014, pp. 2510–2518.
[27] A. Toth, C.T. Kelley, Convergence analysis for Anderson's acceleration, SIAM J. Numer. Anal. 53 (2) (2015) 805–819.
[28] H.F. Walker, P. Ni, Anderson acceleration for fixed point iteration, SIAM J. Numer. Anal. 49 (4) (2011) 1715–1735.
[29] Takumi Washio, Cornelis W. Oosterlee, Krylov subspace acceleration for nonlinear multigrid schemes, Electron. Trans. Numer. Anal. 6 (1997) 271–290, 3–1.
[30] P. Wynn, On a device for computing the $e_m(s_n)$ transformation, Math. Tables Other Aids Comput. 10 (54) (1956) 91–96.
[31] J. Zhang, B. O'Donoghue, S. Boyd, Globally convergent type-i Anderson acceleration for non-smooth fixed-point iterations, arXiv:1808.03971, 2018.
[32] Z. Allen Zhu, L. Orecchia, Linear coupling: an ultimate unification of gradient and mirror descent, in: ITCS, 2017.