NONLINEAR ACCELERATION OF MOMENTUM AND PRIMAL-DUAL ALGORITHMS

RAGHU BOLLAPRAGADA, DAMIEN SCIEUR, AND ALEXANDRE D'ASPREMONT

ABSTRACT. We describe a convergence acceleration scheme for multistep optimization algorithms. The extrapolated solution is written as a nonlinear average of the iterates produced by the original optimization algorithm. Our scheme does not need the underlying fixed-point operator to be symmetric, hence handles e.g. algorithms with momentum terms such as Nesterov's accelerated method, or primal-dual methods. The weights are computed via a simple linear system and we analyze performance in both online and offline modes. We use Crouzeix's conjecture to show that acceleration performance is controlled by the solution of a Chebyshev problem on the numerical range of a non-symmetric operator modelling the behavior of iterates near the optimum. Numerical experiments are detailed on image processing problems, logistic regression and neural network training for CIFAR10 and ImageNet.

1. INTRODUCTION

Extrapolation techniques, such as Aitken's Δ^2 or Wynn's ε-algorithm, provide an improved estimate of the limit of a sequence using its last few iterates, and we refer the reader to [Brezinski and Zaglia, 2013] for a complete survey. These methods have been extended to vector sequences, where they are known as e.g. Anderson acceleration [Walker and Ni, 2011], minimal polynomial extrapolation [Cabay and Jackson, 1976] or reduced rank extrapolation [Eddy, 1979].

Classical optimization algorithms typically retain only the last iterate or the average [Polyak and Juditsky, 1992] of iterates as their best estimate of the optimum, throwing away all the information contained in the converging sequence. This is highly wasteful from a statistical perspective and extrapolation schemes estimate instead the optimum using a weighted average of the last iterates produced by the underlying algorithm, where the weights depend on the iterates (i.e. a *nonlinear* average). Overall, computing those weights means solving a small linear system so nonlinear acceleration has marginal computational complexity.

Recent results by [Scieur et al., 2016] adapted classical extrapolation techniques related to Aitken's Δ^2 and minimal polynomial extrapolation to design extrapolation schemes for accelerating the convergence of basic optimization methods such as gradient descent. They showed that by using only iterates from fixed-step gradient descent, these extrapolation algorithms achieve the optimal convergence rate of [Nesterov, 2013] *without any modification to the original algorithm*. However, these results are only applicable to iterates produced by single-step algorithms such as gradient descent, where the underlying operator is symmetric, thus excluding much faster momentum-based methods such as SGD with momentum or Nesterov's algorithm. Our results here seek to extend those of [Scieur et al., 2016] to multistep methods, i.e. to accelerate accelerated methods.

Our contribution here is twofold. First, we show that the accelerated convergence bounds in [Scieur et al., 2016] can be directly extended to multistep methods when the operator describing convergence near the optimum has a particular block structure, by adjusting the extrapolating sequence. This result applies in particular to Nesterov's method and the stochastic gradient algorithms with a momentum term. Second, we use Crouzeix's recent results [Crouzeix, 2007, Crouzeix and Palencia, 2017, Greenbaum et al., 2017] to show that, in the general non-symmetric case, acceleration performance is controlled by the solution of a Chebyshev problem on the numerical range of the linear, non-symmetric operator modelling the behavior of iterates near the optimum. We characterize the shape of this numerical range for various classical multistep algorithms such as Nesterov's method [Nesterov, 1983], and Chambolle-Pock's algorithm [Chambolle and Pock, 2011].

Date: October 11, 2018.

We then study the performance of our techniques on several classical applications: image processing problems using extrapolation on Chambolle-Pock's algorithm and ℓ_2-regularized logistic regression using acceleration on Nesterov's accelerated method. We also study acceleration on stochastic gradient algorithms with momentum terms used for training neural networks.

On convex problems, the online version (which modifies iterations) is competitive with L-BFGS in our experiments and significantly faster than classical accelerated algorithms. Furthermore, it is robust to misspecified strong convexity parameters.

Stochastic gradient descent is a popular and effective method to train neural networks [Moulines and Bach, 2011, Deng et al., 2013]. A lot of effort has been invested in accelerating stochastic algorithms, in particular, by exploiting the problem structure. Algorithms such as RMSProp [Tieleman and Hinton, 2012] or Adam [Kingma and Ba, 2014] are examples of direct modifications of gradient descent, and estimate some statistical momentum during optimization to speed up the convergence. Unfortunately, these methods can fail to converge on some simple problems [Reddi et al., 2018], and may fail to achieve state-of-the-art test accuracy on image classification problems. Other techniques have been developed to improve convergence speed or accuracy, such as adaptive batch-size for distributed SGD [Goyal et al., 2017], or quasi-second-order methods which improve the rate of convergence of stochastic algorithms [Bollapragada et al., 2018]. However, only a limited number of settings are covered by such techniques, which are not compatible with state-of-the-art architectures.

On these neural network training problems, the offline version of our acceleration scheme improves both test accuracy of early iterations, as well as the final accuracy (see Figures 8 and 10), with only very minor modifications to existing learning pipelines. Our scheme never hurts performance as it runs on top of the algorithm and does not affect iterations. Finally, our scheme produces smoother learning curves. When training neural networks, we observe in our experiments that the convergence speedup produced by acceleration is much more significant in early iterations, which means our method could be used for *rapid prototyping* of network architectures, with significant computational savings.

2. Nonlinear Acceleration of Momentum Methods

Consider the following optimization problem

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

in the variable $x \in \mathbb{R}^n$, where $f(x)$ is strongly convex with parameter μ with respect to the Euclidean norm, and has a Lipschitz continuous gradient with parameter L with respect to the same norm. Assume we solve this problem using an iterative algorithm of the form

$$\begin{cases} x_i &= g(y_{i-1}) \\ y_i &= \sum_{j=1}^{i} \left(\alpha_j^{(i)} x_j + \beta_j^{(i)} y_{j-1}\right) \end{cases} \quad \text{for } i = 1, ..., k, \tag{2}$$

where $x_i \in \mathbb{R}^n$, k is the number of iterations, and $\alpha_j^{(i)}$, $\beta_j^{(i)}$ are arbitrary real scalar coefficients. This iterate update form is more general and includes many classical algorithms such as accelerated gradient method in [Nesterov, 2013],

$$\begin{cases} x_i &= g(y_{i-1}) = y_{i-1} - \frac{1}{L}\nabla f(y_{i-1}) \\ y_i &= \left(1 + \frac{i-1}{i+2}\right) x_i - \frac{i-1}{i+2}\, x_{i-1} \end{cases}.$$

As in [Scieur et al., 2016] we will focus on improving our estimates of the solution to problem (1) by tracking only the sequence of iterates (x_i, y_i) produced by an optimization algorithm, without any further oracle calls to $g(x)$. The main difference with the work of [Scieur et al., 2016] is the presence of a linear combination of previous iterates in the definition of y in (2), so the mapping from x_i to x_{i+1} is usually

non-symmetric. For instance, for Nesterov's algorithm, the Jacobian of x_{i+1} with respect to x_i, y_i reads

$$J_{x_{i+1}} = \begin{bmatrix} 0 & J_g \\ \left(1 + \frac{i-2}{i+1}\right)\mathbf{I} & -\frac{i-2}{i+1}\mathbf{I} \end{bmatrix} \neq J_{x_{i+1}}^T$$

where J_g is the Jacobian of the function g. In what follows, we show that looking at the residue

$$r_i \triangleq x_i - y_{i-1} \tag{3}$$

allows us to recover the convergence results from [Scieur et al., 2016] when the Jacobian of the function g, written J_g, is symmetric. This allows us to accelerate accelerated methods. We now briefly recall the key ideas driving nonlinear acceleration schemes.

2.1. **Regularized Nonlinear Acceleration Scheme.** Nonlinear acceleration aims to find an approximation of the fixed point x^* (assumed to be unique) of g, i.e.

$$x^* = g(x^*).$$

using a linear combination of previous (x_i, y_i) with coefficients c_i. The optimal coefficients c^* to approximate the fixed point using y are found by minimizing the residual of the linear combination,

$$c^* = \arg\min_c \left\| g\left(\sum_{i=1}^k c_i y_i\right) - \sum_{i=1}^k c_i y_i \right\|.$$

in the variable $c \in \mathbb{R}^k$. Of course, this subproblem can be hard to solve for nonlinear functions g, so we will use the residues defined in (3) and solve instead

$$c^* = \arg\min_{c:c^T\mathbf{1}=1} \left\| \sum_{i=1}^k c_i g\left(y_i\right) - \sum_{i=1}^k c_i y_i \right\| = \arg\min_{c:c^T\mathbf{1}=1} \left\| \sum_{i=1}^k c_i r_i \right\| \tag{4}$$

because $x_i = g(y_{i-1})$ and $r_i = x_i - y_{i-1}$. Both formulations are of course equivalent when g is a linear mapping

$$g(x) = A(x - x^*) + x^*. \tag{5}$$

Minimizing on the residues may be unstable, so we add a Tychonov regularization term, which leads to the Regularized Nonlinear Acceleration (RNA) Algorithm [Scieur et al., 2016].

Algorithm 1 Regularized Nonlinear Acceleration (Complexity: $\mathcal{O}(k^2 d)$ if $k \ll d$)

Input: Sequences of k pairs (x_i, y_{i-1}) generated by (2), regularization parameter λ.
Compute matrix of residues $R = [x_1 - y_0, \ldots, x_k - y_{k-1}]$.
Solve the linear system $(R^T R + \lambda I)z = \mathbf{1}$.
Normalize $c = z/(\mathbf{1}^T z)$.
Output: The extrapolated point $\sum_{i=1}^k c_i y_{i-1}$, or $\sum_{i=1}^k c_i x_i$.

2.2. **Generic Convergence Results for Nonlinear Acceleration.** We now link the accuracy of the extrapolation $\sum_{i=1}^k c_i y_{i-1}$ with the norm of a matrix polynomial $p(A)$, where A is the linear fixed point operator in (5), under mild assumptions on the coefficients $\alpha_j^{(i)}$ and $\beta_j^{(i)}$ in (2). For simplicity, we analyze the combination of y_{i-1}, but, in practice, it is better to use the combination of x_i since, for linear g, we have

$$\sum_{i=1}^k c_i x_i = \sum_{i=1}^k c_i g(y_{i-1}) = g\left(\sum_{i=1}^k c_i y_{i-1}\right).$$

so the points x_i are one iteration ahead of y_i. We will prove

$$\min_{c:c^T\mathbf{1}=1} \left\| \sum_{i=1}^{k} c_i r_i \right\| = \min_{p\in\mathcal{P}_{k-1}:p(1)=1} \|p(A)r_1\| \tag{6}$$

$$\leq \|r_1\| \min_{p\in\mathcal{P}_{k-1}:p(1)=1} \|p(A)\|, \tag{7}$$

where $\mathcal{P}_k$ is the linear space of polynomials of degree at most k. This then directly yields a convergence bound on $\| \sum_{i=0}^{k} c_i y_i - x^* \|$, since

$$\left\| \sum_{i=1}^{k} c_i y_{i-1} - x^* \right\| = \left\| (A-I)^{-1} \sum_{i=1}^{k} c_i (x_i - y_{i-1}) \right\| \leq \left\| (A-I)^{-1} \right\| \left\| \sum_{i=1}^{k} c_i r_i \right\|.$$

Thus, if the coefficients c^* in (4) are used, and if we assume $A - I$ invertible,

$$\left\| \sum_{i=1}^{k} c_i^* y_{i-1} - x^* \right\| \leq \left\| (A-I)^{-1} \right\| \|r_1\| \min_{p\in\mathcal{P}_{k-1}:p(1)=1} \|p(A)\|. \tag{8}$$

Studying nonlinear acceleration accuracy thus reduces to studying Chebyshev polynomials under constraint, and we now prove (7).

Theorem 2.1. *Let $\{x_1, \ldots, x_k\}$ and $\{y_0, \ldots, y_{k-1}\}$ be the two sequences produced by running k iterations of the optimization algorithm (2), where $g(x)$ is the linear mapping (5) and $\alpha_i^{(k)}$, $\beta_i^{(k)}$ are arbitrary coefficients such that*

$$\alpha_N^{(k)} \neq 0 \quad \textit{and} \quad \textstyle\sum_{i=1}^{k} \alpha_i^{(k)} + \sum_{i=1}^{k} \beta_i^{(k)} = 1.$$

Then,

$$\min_{c:c^T\boldsymbol{I}=1} \left\| \sum_{i=1}^{k} c_i r_i \right\| = \min_{p\in\mathcal{P}_k:p(1)=1} \|p(A)r_1\|. \tag{9}$$

By (8), extrapolation accuracy is then bounded by

$$\left\| \sum_{i=1}^{k} c_i^* y_{i-1} - x^* \right\| \leq \left\| (A-I)^{-1} \right\| \|r_1\| \min_{p\in\mathcal{P}_{k-1}:p(1)=1} \|p(A)\|.$$

Proof. For clarity, we remove the superscript (k) on the parameters α and β, in the scope of this proof. The key part of the proof is showing the following relation,

$$r_i = q_i(A) r_1, \tag{10}$$

where q_i is a polynomial of degree *exactly* $i-1$, with $q_i(1) = 1$. The rest of the proof is easy, since $\{r_i\}_{i=1\ldots k}$ are formed using independent polynomials (they all have different degrees). Since we have an independent family with k elements, and the maximum degree of those polynomials is $k-1$ (which is q_k, associated to r_k), then we have a basis for $\mathcal{P}_{k-1}$. This means

$$\forall\, p \in \mathcal{P}_{k-1}, \quad \exists\, c \in \mathbb{R}^k : \sum_{i=1}^{k} c_i r_i = p(A) r_1.$$

In addition, if we force $\sum_{i=1}^{k} c_i = 1$, then the coefficients of the resulting polynomial also sum to one since

$$p(1) = \sum_{i=1}^{k} c_i q_i(1) = \sum_{i=1}^{k} c_i = 1.$$

This finally proves (9). It thus remains to show (10). We will prove the claim by recursion. We start with the simple case $i = 1$, i.e.,

$$r_1 = I r_1$$

and the identity matrix I indeed corresponds to the "polynomial" $p(A) = 1 \cdot A^0$ of degree 0, whose "coefficients sum to one". Now, assume the claim is true for r_{i-1}, and we will show it is also true for r_i. Indeed,

$$\begin{aligned} r_i &= x_i - y_{i-1}, \\ &= A(y_{i-1} - x^*) + x^* - y_{i-1}, \\ &= (A - I)\left(\sum_{j=1}^{i-1} \alpha_j x_j + \sum_{j=1}^{i-1} \beta_j y_{j-1} - x^*\right). \end{aligned} \tag{11}$$

Since $\sum_{j=1}^{i-1}(\alpha_j + \beta_j) = 1$,

$$(A - I)\left(\sum_{j=1}^{i-1} \alpha_j x_j + \sum_{j=1}^{i-1} \beta_j y_{j-1} - x^*\right) = (A - I)\left(\sum_{j=1}^{i-1} \alpha_j (x_j - x^*) + \sum_{j=1}^{i-1} \beta_j (y_{j-1} - x^*)\right). \tag{12}$$

We can link $x_j - x^*$ with $y_{j-1} - x^*$ as follow,

$$x_j - x^* = A(y_{j-1} - x^*) + x^* - x^* = A(y_{j-1} - x^*).$$

With this relation, equation (12) simplifies into

$$(A - I)\left(\sum_{j=1}^{i-1} \alpha_j (x_j - x^*) + \sum_{j=1}^{i-1} \beta_j (y_{j-1} - x^*)\right) \quad = \quad (A - I)\sum_{j=1}^{i-1}\Big((A\alpha_j + \beta_j)\,(y_{j-1} - x^*)\Big).$$

In fact, this is a combination of residuals since

$$(A - I)(y_j - x^*) = A(y_j - x^*) - y_j = x_{j+1} - y_j = r_{j+1}.$$

We finally have

$$r_i = \sum_{j=1}^{i-1} (A\alpha_j + \beta_j)\, r_{j-1}.$$

We assumed by recursion that $r_j = q_j(A) r_1$, where p_j is a polynomial of degree *exactly* $j-1$, and $q_j(1) = 1$. The previous equation is thus equivalent to

$$r_i = \underbrace{\sum_{j=1}^{i-1} \Big(\alpha_j A q_j(A) + \beta_j q_j(A)\Big)}_{=q_i(A)} r_1.$$

Let us check if $q_i(1) = 1$. Indeed,

$$q_i(1) = \sum_{j=1}^{i-1} \Big(\alpha_j q_j(1) + \beta_j q_j(1)\Big) = \sum_{j=1}^{i-1} \Big(\alpha_j + \beta_j\Big) = 1.$$

Now, we check if the degree of the polynomial is indeed equal to $i - 1$. We have

$$\deg(q_i(x)) \quad = \quad \deg\left(\sum_{j=0}^{i-1} (\alpha_j x + \beta_j) q_j(x)\right).$$

Since by assumption $\deg(q_j(x)) < \deg(q_{i-1}(x))$ for $j < i - 1$,

$$\deg(q_i(x)) = \deg\left((\alpha_i x + \beta_{i-1}) q_{i-1}(x)\right).$$

Because $\alpha_i \neq 0$ by assumption,

$$\deg(q_i(x)) = \deg\left(\alpha_i x q_{i-1}(x)\right) = 1 + \deg(q_{i-1}(x)).$$

Finally, because the degree of q_{i-1} is exactly equal to $i-2$,

$$\deg(q_i) = i - 1.$$

We proved by recursion that $\deg(q_i) = i-1$ and $q_i(1) = 1$, which concludes the proof. ■

In the next section, we will see how to control the convergence bound in Theorem 2.1.

3. Crouzeix's Conjecture & Chebyshev Polynomials on the Numerical Range

In the previous section, we have seen that the convergence rate of nonlinear acceleration is bounded by the norm of a matrix polynomial,

$$\min_{\substack{p\in\mathcal{P}_k\\ p(1)=1}} \|p(A)\|.$$

Here and in the rest of this paper, $\|\cdot\|$ is the ℓ_2 norm. We first recall the setting studied in [Scieur et al., 2016] where A is symmetric. We then analyze the case where A is non-symmetric using Crouzeix's conjecture.

3.1. Symmetric Case.

Here, we assume A is a symmetric matrix, with $\|A\| < 1-\kappa$, for $\kappa \in]0,1[$. In many applications, such as optimization, κ often refers to the *inverse condition number*. [Scieur et al., 2016] show that extrapolating fixed-step graident descent using nonlinear acceleration algorithm achieves an optimal rate of convergence in the sense of Nesterov [Nesterov, 2013].

Proposition 3.1. *Assume we have a sequence of pairs (x_i, y_{i-1}) generated by (2) where*

$$0 \preceq A \preceq 1-\kappa\,; \qquad \sum_{j=1}^{i}(\alpha_j^{(i)} + \beta_j^{(i)}) = 1 \quad \textit{and} \quad \alpha_i^{(i)} \neq 0 \qquad \forall\, i = 1\dots k.$$

Then the extrapolation $\sum_{i=1}^k c_i^* y_{i-1}$, where c^* is computed using (4), follows

$$\left\|\sum_{i=1}^{k} c_i^* y_{i-1} - x^*\right\|_2 \leq \frac{1}{\kappa}\frac{2\xi^{k-1}}{1+\xi^{2k-1}}\|r_1\|, \quad \xi = \frac{1-\sqrt{\kappa}}{1+\sqrt{\kappa}}.$$

Proof. We quickly recall the main arguments of the proof present in [Scieur et al., 2016]. It mainly consists in upper-bounding the term

$$\min_{p\in\mathcal{P}_k,\, p(1)=1} \|p(A)\|. \tag{13}$$

Because A is *symmetric*, it suffices to look at the eigenvalue of A which maximizes the norm,

$$\min_{p\in\mathcal{P}_k,\, p(1)=1} \max_{\lambda_i(A)} |p(\lambda_i)|. \tag{14}$$

This problem is complex due to the discrete constraints on λ. However, the maximum can be relaxed on the continuous segment $[0, 1-\kappa]$,

$$\min_{p\in\mathcal{P}_k,\, p(1)=1} \max_{\lambda\in[0,1-\kappa]} |p(\lambda)|.$$

This has been already studied for example by Golub and Varga [1961], and the solution is given by a "Chebyshev-like" polynomial, written

$$T_{\kappa,k-1}(x) = \frac{C_{k-1}(t_\kappa(x))}{C_{k-1}(t_\kappa(1))}, \quad t_\kappa(x) = \frac{2x}{1-\kappa} - 1,$$

where $C_{k-1}(x)$ is the Chebyshev polynomial of degree $k-1$. The exact rate of convergence is obtained by evaluating $T_{\kappa,k-1}(x)$ at its maximum value, $x = 1-\kappa$. Finally, the term $\|(I-A)^{-1}\|_2$ is bounded by κ^{-1} given the assumption on A. ■

The crucial part of the proof is moving from (13) to (14), i.e. moving from matrix polynomials to scalar polynomials on a line segment. This step only works here because A has an orthonormal eigenvalue

decomposition with real eigenvalues, i.e. when A is symmetric. We will now recall Crouzeix's conjecture, which will allow us to study the norm of matrix polynomials for non-symmetric matrices.

3.1.1. *Global Asymptotic Bounds for Nonlinear & Stochastic Problems.* We now derive a more general bound for acceleration on generic nonlinear problems. As in [Scieur et al., 2016], convergence of RNA in the general case is essentially obtained via a perturbation analysis of the quadratic case. Indeed, at a step k consider the perturbed linear iteration

$$g(y_k) = A(y_k - x^*) + x^* + \epsilon_k. \tag{15}$$

where ϵ_k corresponds to any kind of perturbation (non-linear or stochastic residual noise, for instance), and A corresponds to the Hessian at the optimum x^*. This corresponds to a perturbation of (5), and includes for example, SGD steps on a non-linear function $f(x)$, where ϵ_k is the sum of the second order term in Taylor series expansion of the function with the stochastic noise induced by the SGD step at iteration k.

Given our analysis of the linear case above, all perturbation results of [Scieur et al., 2016] and [Scieur et al., 2017] still apply. Assuming bounded perturbations in expectation, i.e.,

$$\|\mathbb{E}[\epsilon_i]\| \leq \nu, \qquad \mathbb{E}[\|\epsilon_i\|] \leq \varepsilon \;\; \forall i$$

Then it is possible to derive the global bound of Proposition 5.2 in [Scieur et al., 2017]. This bound is difficult to analyze since it depends on the value of a so-called *regularized Chebyshev polynomial* whose explicit value is still unknown. However, it is possible to give the relation between λ and $(\nu + \sigma)$ to ensure the recovery of an optimal rate of convergence when $\varepsilon \to 0$. In particular, if

$$\lambda \in \left]\left(\frac{\varepsilon}{\|x_0 - x^*\|}\right)^{2/3}, \left(\frac{\varepsilon}{\|x_0 - x^*\|}\right)^{0}\right[,$$

then we can recover the asymptotic rate in Theorem 3.1. The estimation of ε may be hard, but in practice, to ensure good numerical convergence, it suffices to set $\lambda = O\left(\|R^T R\|\right)$, where the constant used is usually small (e.g. 10^{-6}).

3.2. **Non-Symmetric case & Crouzeix's Conjecture.** The results in [Scieur et al., 2016] recalled above handle the case where the operator A is *symmetric*. Bounding $\|p(A)\|_2$ when A is non-symmetric is not as direct. Fortunately, Crouzeix's conjecture [Crouzeix, 2004] allows us to bound $\|p(A)\|_2$ by solving a Chebyshev problem on the numerical range of A, in the complex plane.

Theorem 3.2 ([Crouzeix, 2004])**.** *Let $A \in \mathbb{C}^{n\times n}$, and $p(x) \in \mathbb{C}[x]$, we have*

$$\|p(A)\|_2 \leq c \max_{z \in W(A)} |p(z)|$$

for some absolute constant $c \geq 2$.

Here $W(A) \subset \mathbb{C}$ is the numerical range of the matrix $A \in \mathbb{R}^{n\times n}$, i.e. the range of the Rayleigh quotient

$$W(A) \triangleq \{x^* A x : \|x\|_2 = 1, x \in \mathbb{C}^n\}. \tag{16}$$

[Crouzeix, 2007] shows $c \leq 11.08$ and Crouzeix's conjecture states that this can be further improved to $c = 2$, which is tight. A more recent bound in [Crouzeix and Palencia, 2017] yields $c = 1 + \sqrt{2}$ and there is significant numerical evidence in support of the $c = 2$ conjecture [Greenbaum et al., 2017]. This conjecture has played a vital role in providing convergence results for e.g. the GMRES method [Saad and Schultz, 1986] (see [Choi and Greenbaum, 2015]).

Crouzeix's result allows us to turn the problem of finding uniform bounds for the norm of the matrix polynomial $\|p(A)\|_2$ to that of bounding $p(z)$ over the numerical range of A in the complex plane, an arguably much simpler two-dimensional Chebyshev problem.

3.3. **Numerical Range Approximations.** There are no tractable methods for computing the exact numerical range of a general operator A. However, efficient numerical methods approximate the numerical range based on its key properties. The Toeplitz-Hausdorff theorem [Hausdorff, 1919, Toeplitz, 1918] in particular states that the numerical range $W(A)$ is a closed convex bounded set. Therefore, it suffices to characterize points on the boundary, the convex hull then yields the numerical range.

Johnson [1978] made the following observations using the properties of the numerical range,

$$\max_{z \in W(A)} Re(z) = \max_{r \in W(H(A))} r = \lambda_{max}(H(A)) \tag{17}$$

$$W(e^{i\theta} A) = e^{i\theta} W(A), \quad \forall \theta \in [0, 2\pi), \tag{18}$$

where $Re(z)$ is the real part of complex number z, $H(A)$ is the Hermitian part of A, i.e. $H(A) = (A + A^*)/2$ and $\lambda_{max}(H(A))$ is the maximum eigenvalue of $H(A)$. The first property implies that the line parallel to the imaginary axis is tangent to $W(A)$ at $\lambda_{max}(H(A))$. The second property can be used to determine other tangents via rotations. Using these observations Johnson [1978] showed that the points on the boundary of the numerical range can be characterized as

$$p_\theta = \{v_\theta^* A v_\theta : \theta \in [0, 2\pi)\} \tag{19}$$

where v_θ is the normalized eigenvector corresponding to the largest eigenvalue of the Hermitian matrix

$$H_\theta = \frac{1}{2}(e^{i\theta} A + e^{-i\theta} A^*) \tag{20}$$

The numerical range can thus be characterized as follows.

Theorem 3.3. *[Johnson, 1978] For any $A \in \mathbb{C}^{n \times n}$, we have*

$$W(A) = Co\{p_\theta : 0 \leq \theta < 2\pi\}$$

where $Co\{Z\}$ is the convex hull of the set Z.

Note that p_θ cannot be uniquely determined as the eigenvectors v_θ may not be unique but the convex hull above is uniquely determined.

3.4. **Chebyshev Bounds & Convergence Rate.** Crouzeix's result means that getting a priori bounds on the convergence rate of accelerated algorithms by bounding (9) as in Theorem 2.1 can be achieved by bounding the optimum of the Chebyshev problem

$$\min_{\substack{p \in \mathbb{C}[z] \\ p(1)=1}} \max_{z \in W(A)} |p(z)| \tag{21}$$

where $A \in \mathbb{C}^{n \times n}$. This problem has a trivial answer when the numerical range $W(A)$ is spherical, but the convergence rate can be significantly improved when $W(A)$ is less isotropic.

3.4.1. *Exact Bounds on Ellipsoids.* We can use an outer ellipsoidal approximation of $W(A)$, bounding the optimum value of the Chebyshev problem (21) by

$$\min_{\substack{p(z) \in \mathbb{C}[x] \\ p(1)=1}} \max_{z \in \mathcal{E}_r} |p(z)| \tag{22}$$

where

$$\mathcal{E}_r \triangleq \{z \in \mathbb{C} : |z - 1| + |z + 1| \leq r + 1/r\}. \tag{23}$$

This Chebyshev problem has an explicit solution in certain regimes. As in the real case, we will use $C_n(z)$, the Chebyshev polynomial of degree k. Fischer and Freund [1991] show the following result on the optimal solution to problem (22).

Theorem 3.4. *[Fischer and Freund, 1991, Th. 2] Let $k \geq 5$, $r > 1$ and $c \in \mathbb{R}$. The polynomial*

$$T_{k,\kappa}(z) = T_k(z)/T_k(1-\kappa)$$

is the unique solution of problem (22) if either

$$|1-\kappa| \geq \frac{1}{2}\left(r^{\sqrt{2}} + r^{-\sqrt{2}}\right) \quad or \quad |1-\kappa| \geq \frac{1}{2a_r}\left(2a_r^2 - 1 + \sqrt{2a_r^4 - a_r^2 + 1}\right)$$

where $a_r = (r + 1/r)/2$.

The optimal polynomial for a general ellipse $\mathcal{E}$ can be obtained by a simple change of variables. That is, the polynomial

$$\frac{C_k(\frac{c-z}{d})}{C_k(\frac{c-1}{d})} \tag{24}$$

is optimal for the problem (22) over any ellipse $\mathcal{E}$ with center c, focal distance d and semi-major axis a. It can be easily seen that the maximum value is achieved at the point a on the real axis. That is the solution to the min max problem is given by $\bar{T}_k(a)$. Figure 1 shows the surface of the optimal polynomial with degree 5 for $a = 0.8, d = 0.76$ and $c = 0$.

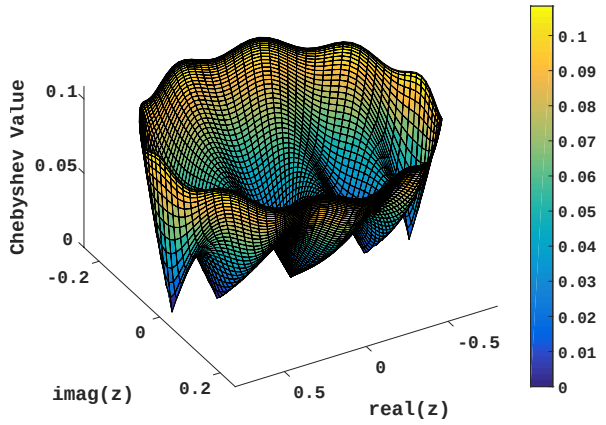

FIGURE 1. Surface of the optimal polynomial $\bar{T}_n(z)$ with degree 5 for $a = 0.8, d = 0.76$ and $c = 0$.

Figure 2 shows the solutions to the problem (22) with degree 5 for various ellipses with center at origin, various eccentricity values $e = d/a$ and semi-major axis a. Here, zero eccentricity corresponds to a sphere, while an eccentricity of one corresponds to a line.

3.4.2. *Sum of Squares Bounds on Semi Algebraic Sets.* The Chebyshev problem (21) minimizing $p(z)$ on the numerical range can also be approximated by a Sum of Squares (SOS) program. This allows us to study, numerically, problems where the numerical range is approximated by compact semialgebraic sets. Suppose that $W(A) \subset K$ where

$$K \triangleq \{g_i(x) \geq 0,\ i = 1, \ldots, r\}$$

for multivariate polynomials $g_i \in \mathbb{R}[x] : \mathbb{R}^2 \to \mathbb{R}$. Then [Lasserre, 2001, Th. 4.2] shows that the optimum value of

$$\min_{\substack{p(z)\in\mathbb{C}[x] \\ p(1)=1}} \max_{z\in K} |p(z)|$$

is upper bounded by the solution to the following SOS program.

$$\begin{array}{ll} \text{min.} & v \\ \text{subject to} & v - p(x) = q^-(x) + \sum_{i=1}^r t_i^-(x) g_i(x) \\ & v + p(x) = q^+(x) + \sum_{i=1}^r t_i^+(x) g_i(x) \end{array} \tag{25}$$

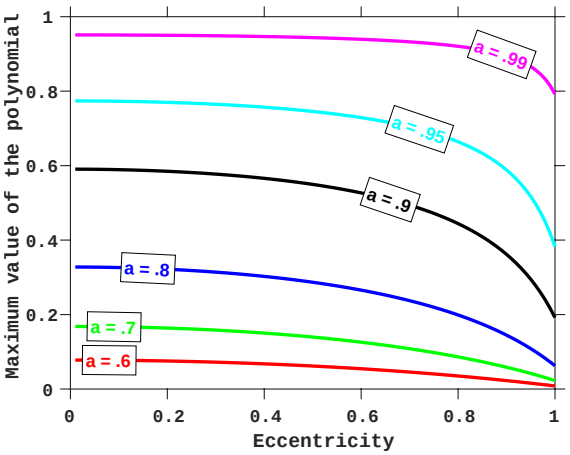

FIGURE 2. Optimal value of the Chebyshev problem (22) for ellipses with centers at origin. Lower values of the maximum mean faster convergence. The higher the eccentricity, the faster the convergence.

in the mutlivariate polynomial variables $p(x), q^{\pm}(x), t_i^{\pm}(x) \in \mathbb{R}[x]$ where $q^{\pm}(x), t_i^{\pm}(x)$ are SOS polynomials. This last problem is equivalent to an SDP and can be solved efficiently when the degree of the polynomials is reasonably small.

4. ACCELERATING NON-SYMMETRIC ALGORITHMS

Using the results from Section 3, we will now apply Theorem 2.1 to some standard non-symmetric algorithms and detail the performance of regularized nonlinear acceleration in these particular cases. In this section, we study generic algorithms of the form

$$x_{i+1} = g(x_i).$$

In particular, this a particular case of our fixed point iterations (2) with $\alpha_i^{(i)} = 1$ and all other coefficients are equal to zero (hence $x_i = y_i$). We have seen in the previous section that controlling the convergence rate of the nonlinear acceleration scheme in Algorithm 1 means bounding the optimal value of the Chebyshev optimization problem in (21) over the numerical range of the operator driving iterations. In what follows, we explicitly detail this operator and approximate its numerical range for two classical algorithms, Nesterov's accelerated method [Nesterov, 1983] and Chambolle-Pock's Primal-Dual Algorithm [Chambolle and Pock, 2011].

4.1. Nesterov's Accelerated Gradient Method.

The iterates formed by Nesterov's accelerated gradient descent method for minimizing smooth strongly convex functions with constant stepsize follow

$$\begin{cases} x_k = y_{k-1} - \alpha \nabla f(y_{k-1}) \\ y_k = x_k + \beta(x_k - x_{k-1}) \end{cases} \tag{26}$$

with

$$\beta = \frac{\sqrt{L} - \sqrt{\mu}}{\sqrt{L} + \sqrt{\mu}},$$

where L is the gradient's Lipschitz continuity constant and μ is the strong convexity parameter. This algorithm is better handled using the results of Section 2, and we only use it here to better illustrate our results on non-symmetric operators.

4.1.1. *Nesterov's Operator in the quadratic case.* When minimizing quadratic functions $f(x) = \frac{1}{2}\|Bx - b\|^2$, using constant stepsize $1/L$, these iterations become,

$$\begin{cases} x_k - x^* &= y_{k-1} - x^* - \frac{1}{L}B^T(By_{k-1} - b) \\ y_k - x^* &= x_k - x^* + \beta(x_k - x^* - x_{k-1} + x^*). \end{cases}$$

or again,

$$\begin{bmatrix} x_k - x^* \\ y_k - x^* \end{bmatrix} = \begin{bmatrix} 0 & A \\ -\beta I & (1+\beta)A \end{bmatrix} \begin{bmatrix} x_{k-1} - x^* \\ y_{k-1} - x^* \end{bmatrix}$$

where $A = I - \frac{1}{L}B^TB$. We write O the *non-symmetric* linear operator in these iterations, i.e.

$$O = \begin{bmatrix} 0 & A \\ -\beta I & (1+\beta)A \end{bmatrix} \tag{27}$$

The results in Section 2 show that we can accelerate the sequence $z_k = (x_k, y_k)$ if the solution to the minmax problem (21) defined over the numerical range of the operator O is bounded.

4.1.2. *Numerical Range.* We can compute the numerical range of the operator O using the techniques described in Section (2). In the particular case of Nesterov's accelerated gradient method, the numerical range is a convex hull of ellipsoids. We show this by considering the 2×2 operators obtained by replacing the symmetric positive matrix A with its eigenvalues, to form

$$O_j = \begin{bmatrix} 0 & \lambda_j \\ -\beta I & (1+\beta)\lambda_j \end{bmatrix} \qquad \text{for all} \qquad j \in \{1, 2, \cdots, n\} \tag{28}$$

where $0 < \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n < 1$ are the eigenvalues of the matrix A. We have the following result.

Theorem 4.1. *The numerical range of operator O is given as the convex hull of the numerical ranges of the operators O_j, i.e.*

$$W(O) = \mathbf{Co}\{W(O_1), W(O_2), \cdots, W(O_n)\} \tag{29}$$

Proof. Let $v_1, v_2, \cdots, v_n$ be eigen vectors associated with eigen values $\lambda_1, \lambda_2, \cdots, \lambda_n$ of the matrix A. We can write

$$A = \sum_{j=0}^{n} \lambda_j v_j v_j^T \qquad I = \sum_{j=0}^{n} v_j v_j^T$$

Let $t \in W(O) \subset \mathbb{C}$. By definition of the numerical range, there exists $z \in \mathbb{C}^{2n}$ with $z^*z = 1$ and

$$\begin{aligned} t &= z^* \begin{bmatrix} 0 & A \\ -\beta I & (1+\beta)A \end{bmatrix} z \\ &= z^* \begin{bmatrix} 0 & \sum_{j=1}^n \lambda_j v_j v_j^T \\ -\beta \sum_{j=1}^n v_j v_j^T & (1+\beta)\sum_{j=1}^n \lambda_j v_j v_j^T \end{bmatrix} z \\ &= \sum_{j=0}^{n} z^* \left(\begin{bmatrix} 0 & \lambda_j \\ -\beta & (1+\beta)\lambda_j \end{bmatrix} \otimes v_j v_j^T \right) \mathbf{vec}([z_1, z_2]) \\ &= \sum_{j=0}^{n} z^* \, \mathbf{vec}\left(v_j v_j^T [z_1, z_2] \begin{bmatrix} 0 & \lambda_j \\ -\beta & (1+\beta)\lambda_j \end{bmatrix}^T \right) \end{aligned}$$

and since $v_j v_j^T v_j v_j^T = v_j v_j^T$, this last term can be written

$$t = \sum_{j=0}^{n} \mathbf{Tr}\left(v_j v_j^T [z_1, z_2] \begin{bmatrix} 0 & \lambda_j \\ -\beta & (1+\beta)\lambda_j \end{bmatrix}^T [z_1, z_2]^* v_j v_j^T \right)$$
$$= \sum_{j=0}^{n} \mathbf{Tr}(v_j v_j^T) \left([v_j^T z_1, v_j^T z_2] \begin{bmatrix} 0 & \lambda_j \\ -\beta & (1+\beta)\lambda_j \end{bmatrix}^T [z_1^* v_j, z_2^* v_j]^T \right)$$

Now, let $w_j = [z_1^* v_j, z_2^* v_j]^T$, and

$$y_j = \frac{w_j^T O_j w_j}{\|w_j\|_2^2}$$

and by the definition of the numerical range, we have $y_j \in W(O_j)$. Therefore,

$$t = \sum_{j=0}^{n} \left(\frac{w_j^T O_j w_j}{\|w_j\|_2^2} \right) \|w_j\|_2^2$$

hence

$$t \in \mathbf{Co}(W(O_1), W(O_2), \cdots, W(O_n)).$$

We have shown that if $t \in W(O)$ then $t \in \mathbf{Co}(W(O_1), W(O_2), \cdots, W(O_n))$. We can show the converse by following the above steps backwards. That is, if $t \in \mathbf{Co}(W(O_1), W(O_2), \cdots, W(O_n))$ then we have,

$$t = \sum_{j=0}^{n} \theta_j \left(\frac{w_j^T O_j w_j}{\|w_j\|_2^2} \right)$$

where $\theta_j > 0$, $\sum_{j=0}^{n} \theta_j = 1$ and $w_j \in \mathbb{C}^2$. Now, let

$$z = \sum_{j=0}^{n} \frac{\mathbf{vec}(v_j w_j^T) \theta_j^{1/2}}{\|w_j\|}$$

and we have,

$$t = \sum_{j=0}^{n} [z_1^* v_j z_2^* v_j] O_j \begin{bmatrix} v_j^T z_1 \\ v_j^T z_2 \end{bmatrix}$$

wherein we used the fact that $v_j^T v_k = 0$ for any $j \neq k$ and $v_j^T v_j = 1$ in computing $w_j^T = [z_1^* v_j z_2^* v_j]$. We also note that $z^* z = 1$ by the definition of z and rewriting the sum in the matrix form we can show that $t \in W(O)$ which completes the proof. ■

To minimize the solution of the Chebyshev problem in (21) and control convergence given the normalization constraint $p(1) = 1$, the point $(1, 0)$ should be outside the numerical range. Because the numerical range is convex and symmetric w.r.t. the real axis (the operator O is real), this means checking if the maximum real value of the numerical range is less than 1.

For 2×2 matrices, the boundary of the numerical range is given by an ellipse [Donoghue, 1957], so the numerical range of Nesterov's accelerated gradient method is the convex hull of ellipsoids. The ellipse in [Donoghue, 1957] can be determined directly from the entries of the matrix as in Johnson [1974], as follows.

Theorem 4.2. *[Johnson, 1974] For any real 2 by 2 matrix*

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

the boundary of the numerical range is an ellipse whose axes are the line segments joining the points x to y and w to z respectively where,

$$x = \frac{1}{2}(a+d-((a-d)^2+(b+c)^2)^{1/2}) \qquad w = \frac{a+d}{2} - i\left|\frac{b-c}{2}\right|$$
$$y = \frac{1}{2}(a+d+((a-d)^2+(b+c)^2)^{1/2}) \qquad z = \frac{a+d}{2} + i\left|\frac{b-c}{2}\right|$$

are the points in the complex plane.

This allows us to compute the maximum real value of $W(O)$, as the point of intersection of $W(O)$ with the real line which can be computed explicitly as,

$$re(O) = \max Re(W(O)) = \max_j Re(W(O_j)) = \frac{1}{2}\left((1+\beta)\lambda_n + (\lambda_n^2(1+\beta)^2 + (\lambda_n-\beta)^2)^{1/2}\right) \quad (30)$$

where $\lambda_n = 1 - \frac{\mu}{L}$.

We observe that $re(O)$ is a function of the condition number of the problem and takes the values in the interval $[0, 2]$. Therefore, RNA will only work on Nesterov's accelerated gradient method when $re(O) < 1$ holds, which implies that the condition number of the problem $\kappa = \frac{L}{\mu}$ should be less than around 2.5 which is highly restrictive.

An alternative approach is to use RNA on a sequence of iterates sampled every few iterations, which is equivalent to using powers of the operator O. We expect the numerical radius of some power of operator O to be less than 1 for any conditioning of the problem. This is because the iterates are converging at an $R-$linear rate and so the norm of the power of the operator is decreasing at an $R-$linear rate with the powers. Therefore, using the property that the numerical radius is bounded by the norm of the operator we have,

$$re(O^p) = \max Re(W(O^p)) \leq r_{O^p} \leq \|O^p\| \leq C_p\rho^p$$

where r_{O^p} is the numerical radius of O^p. Figure 3 shows the numerical range of the powers of the operator O for a random matrix B^TB with dimension $d = 50$. We observe that after some threshold value for the power p, $(1, 0)$ lies outside the field values corresponding to O^p thus guaranteeing that the acceleration scheme will work. We also observe that the boundaries of the field values are almost circular for higher powers p, which is consistent with results on optimal matrices in [Lewis and Overton, 2018]. When the numerical range is circular, the solution of the Chebyshev problem is trivially equal to z^p so RNA simply picks the last iterate and does not accelerate convergence.

The difficulty in performing RNA on Nesterov's accelerated gradient method arises due to the fact that the iterates can be non-monotonic. The restriction that 1 should be outside the numerical range is necessary for both non-symmetric and symmetric operators. In symmetric operators, the numerical range is a line segment on the real axis and the numerical radius and spectral radius are equal, so this restriction is equivalent to having spectral radius less than 1, i.e. having monotonically converging iterates.

4.2. **Chambolle-Pock's Primal-Dual Algorithm.** Chambolle-Pock is a first-order primal-dual algorithm used for minimizing composite functions of the form

$$\min_x h_p(x) := f(Ax) + g(x) \quad (31)$$

where f and g are convex functions and A is a continuous linear map. Optimization problems of this form arise in e.g. imaging applications like total variation minimization (see Chambolle and Pock [2016]). The Fenchel dual of this problem is given by

$$\max_y h_d(y) := -f^*(-y) - g^*(A^*y) \quad (32)$$

where f^*, g^* are the convex conjugate functions of f, g respectively. These problems are primal-dual formulations of the general saddle point problem,

$$\min_x \max_y < Ax, y > +g(x) - f^*(y), \quad (33)$$

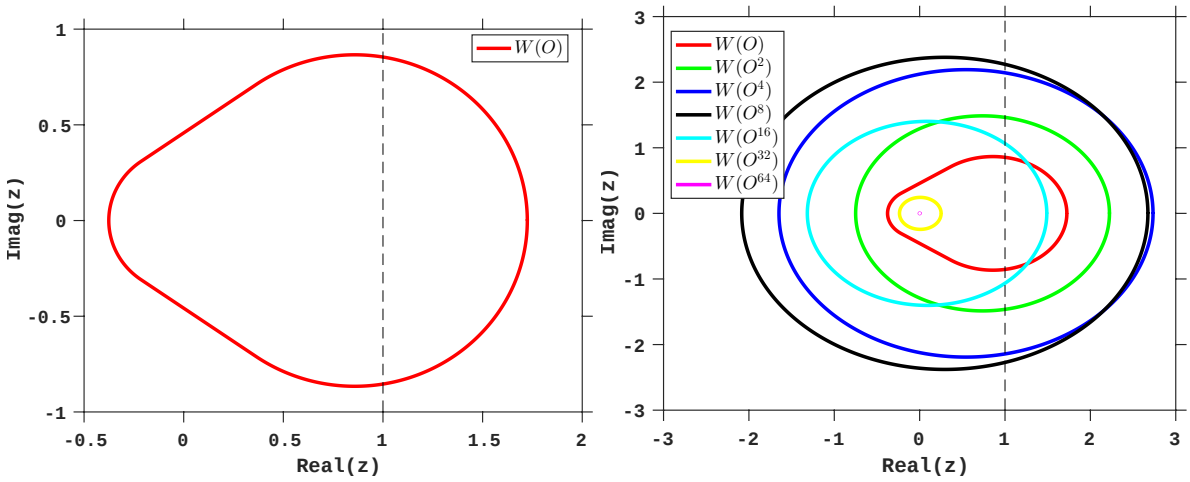

FIGURE 3. Numerical range for the linear operator in Nesterov's method, on a random quadratic problem with dimension 50. Left: Operator O. Right: Various operator powers O^p. The RNA scheme will improve convergence whenever the point $(1, 0)$ lies outside of the numerical range of the operator.

where f^*, g are closed proper functions. Chambolle and Pock [2011] designed a first-order primal-dual algorithm for solving these problems, where primal-dual iterates are given by

$$\begin{cases} y_{k+1} = \mathbf{Prox}_{f^*}^{\sigma}(y_k + \sigma A \bar{x}_k) \\ x_{k+1} = \mathbf{Prox}_{g}^{\tau}(x_k - \tau A^* y_{k+1}) \\ \bar{x}_{k+1} = x_{k+1} + \theta(x_{k+1} - x_k) \end{cases} \tag{34}$$

where σ, τ are the step length parameters, $\theta \in [0, 1]$ is the momentum parameter and the proximal mapping of a function f is defined as

$$\mathbf{Prox}_f^{\tau}(y) = \arg\min_x \left\{ \frac{1}{2\tau}\|y - x\|^2 + f(x) \right\}$$

Note that if the proximal mapping of a function is available then the proximal mapping of the conjugate of the function can be easily computed using Moreau's identity, with

$$\mathbf{Prox}_f^{\tau}(y) + \mathbf{Prox}_{f^*}^{1/\tau}(y/\tau) = y$$

The optimal strategy for choosing the step length parameters σ, τ and the momentum parameter θ depend on the smoothness and strong convexity parameters of the problem. When f^* and g are strongly convex with strong convexity parameters δ and γ respectively then these parameters are chosen to be constant values given as

$$\sigma = \frac{1}{\|A\|}\sqrt{\frac{\gamma}{\delta}} \qquad \tau = \frac{1}{\|A\|}\sqrt{\frac{\delta}{\gamma}} \qquad \theta = \frac{1}{1 + \frac{2\sqrt{\gamma\delta}}{\|A\|}} \tag{35}$$

to yield the optimal linear rate of convergence. When only one of f^* or g is strongly convex with strong convexity parameter γ, then these parameters are chosen adaptively at each iteration as

$$\theta_k = \frac{1}{\sqrt{1 + 2\gamma\tau_k}} \qquad \sigma_{k+1} = \frac{\sigma_k}{\theta_k} \qquad \tau_{k+1} = \tau_k \theta_k \tag{36}$$

to yield the optimal sublinear rate of convergence.

A special case of the primal-dual algorithm with no momentum term, i.e., $\theta = 0$ in (34) is also known as the Arrow-Hurwicz method (Mizoguchi [1960]). Although theoretical complexity bounds for this algorithm are worse compared to methods including a momentum term, it is observed experimentally that the performance is either on par or sometimes better, when step length parameters are chosen as above.

We first consider algorithms with no momentum term and apply RNA to the primal-dual sequence $z_k = (y_k, x_k)$. We note that, as observed in the Nesterov's case, RNA can only be be applied on non-symmetric operators for which the normalization constant 1 is outside their numerical range. Therefore, the step length parameters τ, σ should be suitably chosen such that this condition is satisfied.

4.2.1. *Chambolle-Pock's Operator in the Quadratic Case.* When minimizing smooth strongly convex quadratic functions where $f(Ax) = \frac{1}{2}\|Ax - b\|^2$ and $g(x) = \frac{\mu}{2}\|x\|^2$, the proximal operators have closed form solutions. That is

$$\mathbf{Prox}^{\sigma}_{f^*}(y) = \frac{y - \sigma b}{1 + \sigma} \quad \text{and} \quad \mathbf{Prox}^{\tau}_{g}(x) = \frac{1}{1 + \tau\mu}.$$

Iterates of the primal-dual algorithm with no momentum term can be written as,

$$y_{k+1} = \frac{y_k + \sigma A x_k - \sigma b}{1 + \sigma} \quad \text{and} \quad x_{k+1} = \frac{x_k - \tau A^T y_{k+1}}{1 + \tau\mu}$$

Note that the optimal primal and dual solutions satisfy $y^* = Ax^* - b$ and $x^* = \frac{-1}{\mu} A^T y$. This yields the following operator form for iterations

$$\begin{bmatrix} y_k - y^* \\ x_k - x^* \end{bmatrix} = \begin{bmatrix} \frac{I}{1+\sigma} & \frac{\sigma A}{1+\sigma} \\ \frac{\tau A^T}{(1+\sigma)(1+\tau\mu)} & \frac{I}{1+\tau\mu} - \frac{\tau\sigma A^T A}{(1+\sigma)(1+\tau\mu)} \end{bmatrix} \begin{bmatrix} y_{k-1} - y^* \\ x_{k-1} - x^* \end{bmatrix} \tag{37}$$

with operator,

$$O = \begin{bmatrix} \frac{I}{1+\sigma} & \frac{\sigma A}{1+\sigma} \\ \frac{\tau A^T}{(1+\sigma)(1+\tau\mu)} & \frac{I}{1+\tau\mu} - \frac{\tau\sigma A^T A}{(1+\sigma)(1+\tau\mu)}. \end{bmatrix} \tag{38}$$

Note also that O is a non-symmetric operator except when $\sigma = \frac{\tau}{1+\tau\mu}$, in which case the numerical range is a line segment on the real axis and the spectral radius is equal to the numerical radius.

4.2.2. *Numerical Range.* The numerical range of the operator can be computed using the techniques described in Section 2. As mentioned earlier, the point 1 should be outside the numerical range for the Chebyshev polynomial to be bounded. Therefore, using (17), we have,

$$re(O) = \max Re(W(O)) = \lambda_{max}(H(O)) = \lambda_{max}\left(\frac{O + O^*}{2}\right)$$

The step length parameters σ, τ should be chosen such that the above condition is satisfied. We observe empirically that there exists a range of values for the step length parameters such that $re(O) < 1$. Figure 4 shows the numerical range of operator O for $\sigma = 4, \tau = 1/\|A^T A\|$ with two different regularization constants and Figure 5 shows the contours of $re(O)$ for different values of σ and τ.

We also consider non-smooth problems in addition to the smooth strongly convex problems in the numerical experiments section. While our scheme does not explicitly handle nonsmoothness but we report some preliminary empirical results which show the benefits of RNA.

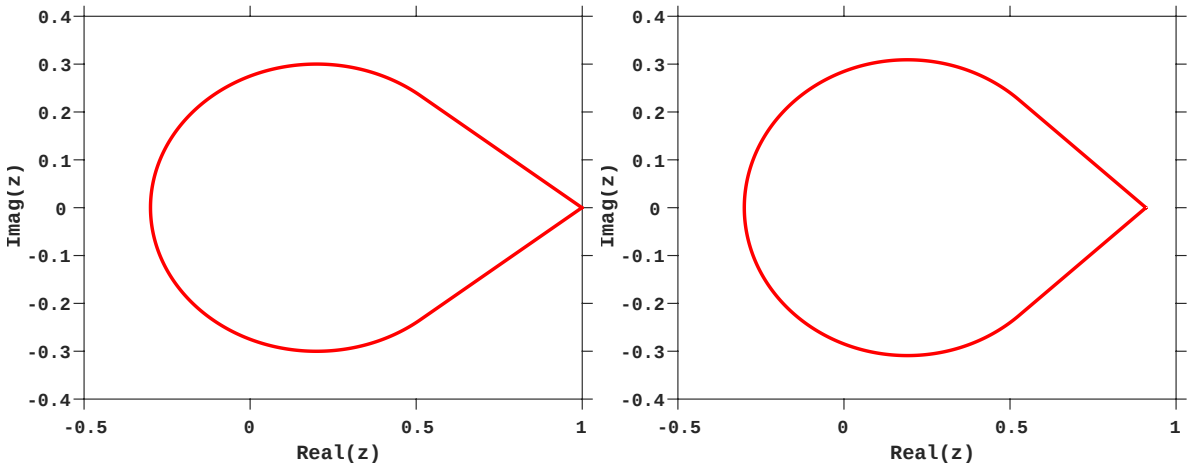

FIGURE 4. Field values for the Sonar dataset [Gorman and Sejnowski, 1988] with $\sigma = 4, \tau = 1/\|A^TA\|$. The dataset has been scaled such that $\|A^TA\| = 1$. Left: $\mu = 10^{-3}$. Right: $\mu = 10^{-1}$. The smaller numerical range on the right means faster convergence.

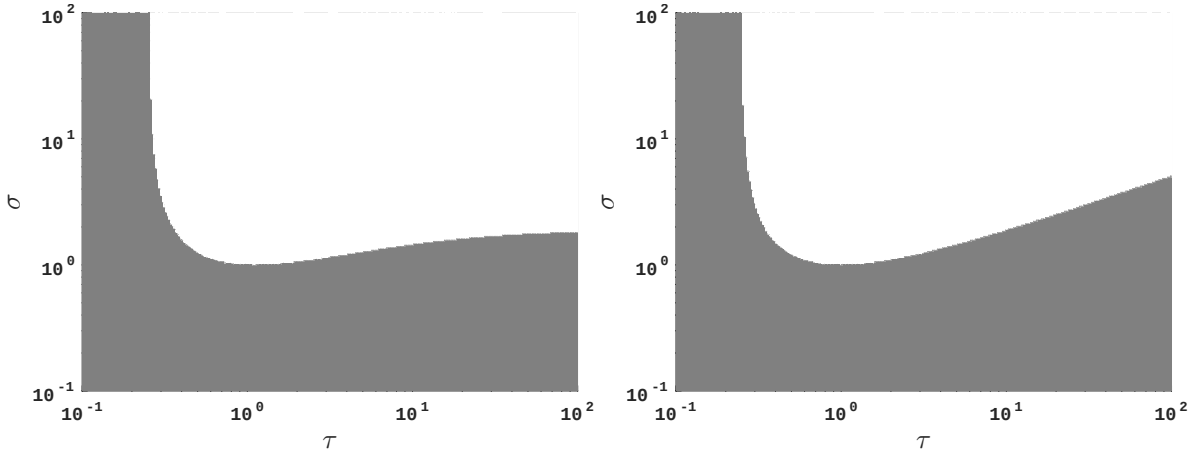

FIGURE 5. Plot of $re(O^p)$ with degree $p = 5$ for the Sonar dataset [Gorman and Sejnowski, 1988] for different values of τ and σ. White color represents values for which $re(O^p) \leq 1$ (converging) and grey color represents values $re(O^p) > 1$ (not converging). Left: $\mu = 10^{-3}$. Right: $\mu = 10^{-1}$.

5. ONLINE NONLINEAR ACCELERATION

In this section, we develop the online-RNA algorithm, which injects the extrapolated point built by RNA directly inside the algorithm. This means accelerating and restarting the algorithm at each step, thus improving its rate of convergence.

5.1. **Online-RNA Algorithm.** The design of the online-RNA algorithm is straightforward. Since the extrapolation is a linear combination of previous iterates, it fits exactly in the template for y_i in method (2) of Section 2, so any algorithm $g(x)$ can be accelerated online as follows,

$$\begin{cases} x_i &= g(y_{i-1}) \\ y_i &= \textbf{RNA}(\{(x_j, y_{j-1})\}_{j=1\ldots i}, \lambda) \end{cases} \tag{39}$$

where the second step corresponds to the application of Algorithm (1) using the sequence of x_j and y_{j-1} generated so far. The only restriction, regarding Theorem 2.1, is that the coefficient associated to x_i which

should not be zero. This is very unlikely and can be easily fixed by using another value for λ. Assuming this condition holds, the algorithm has an optimal rate of convergence on quadratic problems.

However, the complexity grows quadratically with k, the iteration counter, because this is also the length of sequences $\{x_i\}$ and $\{y_{i-1}\}$. In practice, it is better to use a fixed window where k is bounded. This strategy is similar to the limited version of BFGS, where a fixed number of gradients is stored in memory [Boyd and Vandenberghe, 2004].

5.2. **Accelerating Algorithms with Momentum Terms.** The main limitation with the online version of RNA is the acceleration of already-accelerated algorithm. Usually, such methods already present an inner linear combination of previous iterates, for example Nesterov's algorithm [Nesterov, 2013]. Combining the online-RNA and already existing linear combination is not that straightforward. A naive, but innefficient way to combine both approaches consists in computing sequentially the extrapolation with RNA then apply the linear combination (or vice-versa), i.e.

$$\begin{cases} z_i &= g(y_{i-1}) \\ x_i &= \mathbf{RNA}(\{(z_i, y_i)\}, \lambda) \\ y_i &= \sum_{j=1}^{i} \left(\alpha_j^{(i)} x_j + \beta_j^{(i)} y_{j-1}\right) \end{cases}. \tag{40}$$

In fact, the combination of the second and third step is meaningless. At the end, we still end with a linear combination of previous iterates. On one hand, it deteriorates the RNA solution, so it does not correspond to the optimal solution (4). On the other hand, it does not correspond to the original linear combination, thus potentially voiding theoretical convergence guarantees.

However, some of these methods are designed in a way that the point x_i has to satisfy a "sufficient descent condition", satisfied when $x_i = g(y_{i-1})$. For example, Nesterov's accelerated gradient can be written

$$\begin{cases} x_i & \text{such that } f(x_i) \leq f(x_{i-1}) - \frac{1}{2L}\|\nabla f(x_{i-1})\|_2^2 \\ y_i & = (1+\beta)x_i - \beta x_{i-1} \end{cases}$$

In this case, the sufficient descent condition is satisfied when $g(x)$ is a gradient step with step length $\frac{1}{L}$. The linear combination in this algorithm ensures a better convergence rate than gradient method, which is not true anymore when using the (bad) strategy described in (40).

In this section, we accelerate methods with momentum that can be written as

$$\begin{cases} x_i & \text{such that } h(x_i, \{(x_j, y_{j-1})\}_{j=1,\dots,(i-1)}) \leq 0 \\ y_i & = \sum_{j=1}^{i} \left(\alpha_j^{(i)} x_j + \beta_j^{(i)} y_{j-1}\right) \end{cases} \tag{41}$$

where h can be assimilated, for example, to a sufficient descent condition. Usually, the step in y_i has been designed to ensure a theoretical convergence rate, usually better than applying the simple fixed-point iteration $x_i = g(x_{i-1})$. However, the application of online RNA will break this theoretical guarantee. The following algorithm is a slight modification to online RNA, which checks if we can use the extrapolation step while ensuring the theoretical rate of convergence.

This method preserves the rate of convergence of the original algorithm, as shown in the following proposition.

Proposition 5.1. *Assume we have a proven rate of convergence for y_i when it follows (41). In addition, assume the condition $h(x_i, \{(x_j, y_{j-1})\}_{j=1,\dots,(i-1)}) \leq 0$ is satisfied when $x_i = g(y_{i-1})$. Then the sequence generated by Algorithm 2 also has the same rate of convergence.*

Proof. Indeed, when z_i does not satisfy $h(z_i, \{(x_j, y_{j-1})\}_{j=1,\dots,(i-1)}) \leq 0$, then one step of Algorithm 2 corresponds exactly to one step of (41). When z_i satisfy the condition, then y_i in (41) becomes

$$y_i = \alpha_i^{(i)} z_i + \sum_{j=1}^{i-1} \alpha_j^{(i)} x_j + \sum_{j=1}^{i} \beta_j^{(i)} y_{j-1}.$$

Algorithm 2 Regularized Nonlinear Acceleration for Momentum-based Algorithms

Input: Sequences of k pairs (x_j, y_{j-1}) generated by (41), regularization parameter λ.
Compute extrapolation $x_{\text{extr}} = \textbf{RNA}(\{(x_j, y_{j-1}\}, \lambda)$.
Compute the conditional point z_k, written

$$z_k = \frac{1}{\alpha_k^{(k)}} \left(x_{\text{extr}} - \sum_{j=1}^{k-1} \alpha_j^{(k)} x_j - \sum_{j=1}^{k} \beta_j^{(k)} y_j \right).$$

if the condition $h(z_k, \{(x_j, y_{j-1})\}_{j=1,\dots,(k-1)}) \leq 0$ is satisfied **then**
Use **RNA** algorithm, $y_k = x_{\text{extr}}$.
else
Use the original combination, $y_k = \sum_{j=1}^{k} \left(\alpha_j^{(k)} x_j + \beta_j^{(k)} y_{j-1} \right)$.
end if

Replacing z_i by its expression gives

$$y_i = \sum_{j=1}^{i} c_j^* x_j,$$

which preserves the rate of convergence since z_i satisfy the condition. ■

In fact, we can see Algorithm 2 to be a 'smart' line-search technique, since it has the same complexity than a backtracking line search (the function is evaluated two times per iteration). The algorithm is particularly simple when using Nesterov's method because it has only two parameters in the linear combination. The acceleration scheme makes accelerated gradient adaptive to both smoothness and strong convexity constant, while keeping the optimal rate of convergence. We illustrate the explicit acceleration of Nesterov's method in Algorithm 3. It achieves the optimal $1 - \sqrt{\frac{\mu}{L}}$ rate of convergence on smooth and strongly convex functions, and usually performs much better than Nesterov's method. However, we will see in the numerical experiment below that its numerical performance is often worse than simply accelerating gradient descent with the online-RNA scheme, so preserving the optimal theoretical convergence bound has a nontrivial numerical cost here.

Algorithm 3 Adaptive Optimal First Order Method for Smooth and Strongly Convex Functions

Input: Smoothness parameter L, strong convexity constant μ, starting point $x_0 = y_0$.
Compute $\beta = \frac{\sqrt{L} - \sqrt{\mu}}{\sqrt{L} + \sqrt{\mu}}$.
for $i = 1$ to k **do**
Perform a gradient step $x_i = y_{i-1} - \frac{1}{L} \nabla f(y_{i-1})$.
Compute the extrapolation $x_{\text{extr}} = \textbf{RNA}(\{x_j, y_{j-1}\}, \lambda)$
Compute the conditional point

$$z_i = \frac{1}{1+\beta} \left(x_{\text{extr}} + \beta x_{i-1} \right)$$

if the descent condition $f(z_i) \leq f(y_i) - \frac{1}{2L} \|\nabla f(y_i)\|_2^2$ is satisfied **then**
Use the extrapolation $y_i = x_{\text{extr}}$
else
Use the original combination $y_i = (1+\beta) x_i - \beta x_{i-1}$
end if
end for

6. Numerical Results

We now study the performance of our techniques on several classical applications: image processing problems using extrapolation on Chambolle-Pock's algorithm and ℓ_2-regularized logistic regression using acceleration on Nesterov's accelerated method[1]. We also study acceleration on stochastic gradient algorithms with momentum terms used for training neural networks.

6.1. **Accelerating Algorithms with Momentum Terms.** The following numerical experiments seek to highlight the benefits of RNA in its offline and online versions when applied to the gradient method (with or without momentum term). Since the complexity grows quadratically with the number N of points in the sequences $\{x_i\}$ and $\{y_i\}$, we will use RNA with a fixed window size ($N = 5$ for stochastic and $N = 10$ for convex problems) and regularization parameter $\lambda = 10^{-8}\|R^T R\|_2$ in all these experiments. These values are sufficiently large to show a significant improvement in the rate of convergence, but can of course be fine-tuned.

6.1.1. *Logistic Regression.* We solve a classical regression problem on the Madelon-UCI dataset [Guyon, 2003] using the logistic loss with ℓ_2 regularization. The regularization has been set such that the condition number of the function is equal to 10^6. We compare to standard algorithms such as the simple gradient scheme, Nesterov's method for smooth and strongly convex objectives [Nesterov, 2013] and L-BFGS. For the step length parameter, we used a backtracking line-search strategy. We compare these methods with their offline RNA accelerated counterparts, as well as with the online version of RNA described in (39).

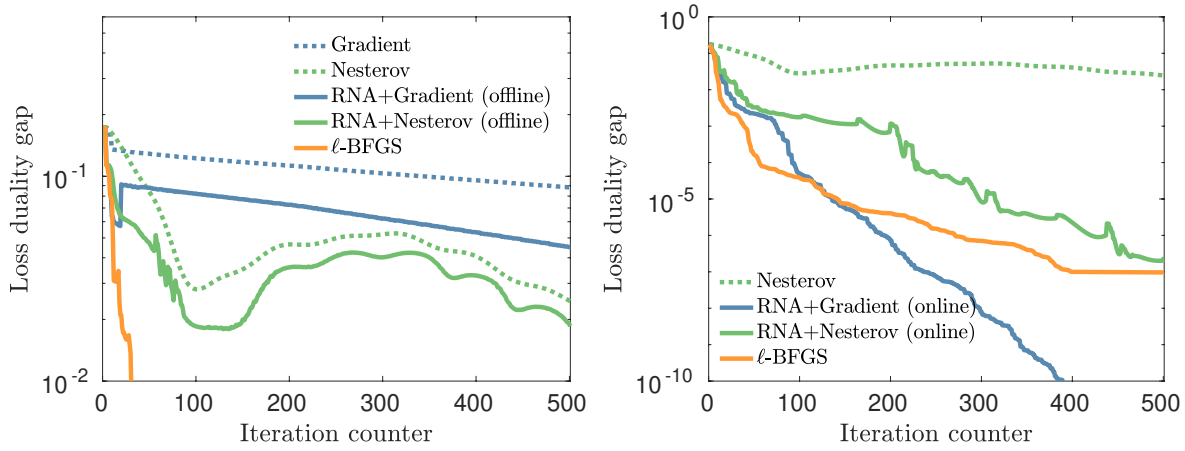

FIGURE 6. Logistic loss on the Madelon [Guyon, 2003]. Comparison between offline (*left*) and online (*right*) strategies for RNA on gradient and Nesterov's method. We use ℓ-BFGS (with $\ell = 100$ gradients stored in memory) as baseline. Clearly, one step of acceleration improves the accuracy. The performance of online RNA, which applies the extrapolation at *each* step, is similar to that of L-BFGS methods, though RNA does not use line-search and requires 10 times less memory.

We observe in Figure 6 that offline RNA improves the convergence speed of gradient descent and Nesterov's method. However, the improvement is only a constant factor: the curves are shifted but have the same slope. Meanwhile, the online version greatly improves the rate of convergence, transforming the basic gradient method into an optimal algorithm competitive with line-search L-BFGS.

In contrast to most quasi-Newton methods (such as L-BFGS), RNA does *not* require a Wolfe line-search to be convergent. This is because the algorithm is stabilized with a Tikhonov regularization. In addition, the regularization in a way controls the impact of the noise in the iterates, making the RNA algorithm suitable for stochastic iterations [Scieur et al., 2017].

[1]The source code for the numerical experiments can be found on GitHub at https://github.com/windows7lover/RegularizedNonlinearAcceleration

6.1.2. *Training CNNs for image classification.* Because one stochastic iteration is not informative due to the noise, we refer to x_k as the model parameters (including batch normalization statistics) corresponding to the final iteration of the epoch k. In this case, we do not have an explicit access to "$(x_k - y_{k-1})$", so we will estimate it during the stochastic steps. Let $y_k^{(t)}$ be the parameters of the network at epoch k after t stochastic iterations, and $x_k^{(t+1)}$ be the parameters after one stochastic gradient step. Then, for a data set of size D,

$$x_k - y_{k-1} \approx \frac{1}{D}\sum_{t=1}^{D}(x_k^{(t+1)} - y_k^{(t)}) = -h\frac{1}{D}\sum_{t=1}^{D}\nabla f(y_k^{(t)}).$$

This means the matrix R in Algorithm 1 will be the matrix of (estimated) gradients. Because the learning curve is highly dependent on the learning rate schedule, we decided to use a linearly decaying learning rate to better illustrate the benefits of acceleration, even if acceleration also works with a constant learning rate schedule (see [Scieur et al., 2018] and Figure 7). In all our experiments, until epoch T, the learning rate decreases linearly from an initial value h_0 to a final value h_T, with

$$h_k = h_0 + (k/T)(h_T - h_0). \tag{42}$$

We then continue the optimization during 10 additional epochs using h_T to stabilize the curve. We summarize the parameters used for the optimization in Table 1.

	h_0	h_T	momentum
SGD and Online RNA (39)	1.0	0.01	0
SGD + momentum	0.1	0.001	0.9

TABLE 1. Parameters used in (42) to generate the learning rate for optimizers. We used the same setting for their accelerated version with RNA.

6.1.3. *CIFAR10.* CIFAR-10 is a standard 10-class image dataset comprising $5 \cdot 10^4$ training samples and 10^4 samples for testing. Except for the linear learning rate schedule above, we follow the standard practice for CIFAR-10. We applied the standard augmentation via padding of 4 pixels. We trained the networks VGG19, ResNet-18 and DenseNet121 during 100 epochs ($T = 90$) with a weight decay of $5 \cdot 10^{-4}$.

We observe in Figure 9 that the online version does not perform as well as in the convex case. More surprisingly, it is outperformed by its offline version (Figure 8) which computes the iterates on the side. In fact, the offline experiments detailed in Figure 8 exhibit much more significant gains. It produces a similar test accuracy, and the offline version converges faster than SGD, especially for early iterations. We reported speedup factors to reach a certain tolerance in Table 2. This suggests that the offline version of RNA is a good candidate for training neural networks, as it converges faster while guaranteeing performance *at least* as good as the reference algorithm.

6.1.4. *ImageNet.* Here, we apply the RNA algorithm to the standard ImageNet dataset. We trained the networks during 90 epochs ($T = 80$) with a weight decay of 10^{-4}. We reported the test accuracy on Figure 10 for the networks ResNet-50 and ResNet-152. We only tested the offline version of RNA here, because in previous experiments it gives better result than its online counterpart.

We again observe that the offline version of Algorithm 1 improves the convergence speed of SGD with and without momentum. In addition, we show a substantial improvement of the accuracy over the non-accelerated baseline. The improvement in the accuracy is reported in Table 3. Interestingly, the resulting training loss is smoother than its non accelerated counterpart, which indicates a noise reduction.

6.2. **Algorithms with Non-symmetric Operators.** We conducted numerical experiments to illustrate the performance of RNA on non-symmetric algorithms. We consider two different classes of problems: smooth strongly convex problems and non-smooth convex problems.

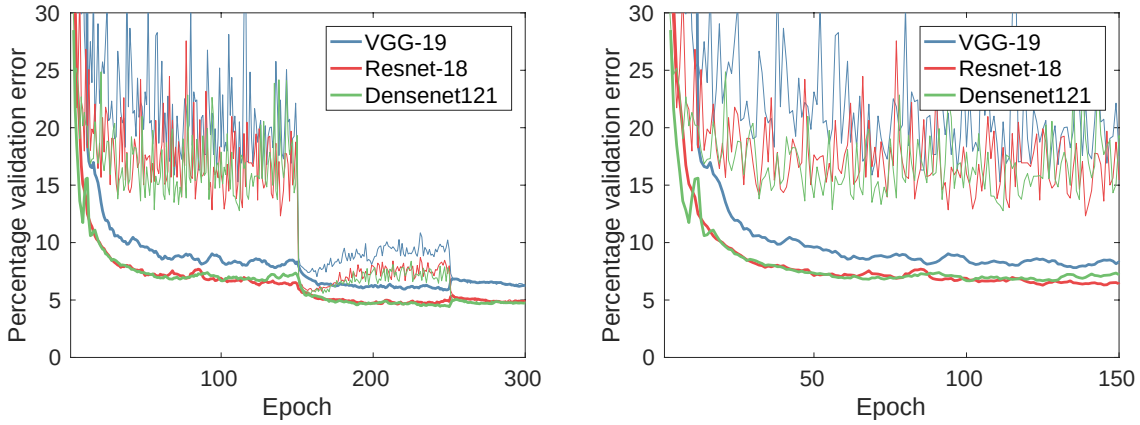

FIGURE 7. Prototyping networks: acceleration (bottom curves) gives a smoother convergence, producing a clearer ranking of architectures, much earlier (we use a flat learning rate). The right plot zooms on left one.

Tolerance	SGD	SGD+momentum	SGD+RNA	SGD+momentum+RNA
5.0%	68 (0.87×)	59	21 (2.81×)	**16** (3.69×)
2.0%	78 (0.99×)	77	47 (1.64×)	**40** (1.93×)
1.0%	82 (1.00×)	82	67 (1.22×)	**59** (1.39×)
0.5%	84 (1.02×)	86	75 (1.15×)	**63** (1.37×)
0.2%	86 (1.13×)	97	**84** (1.15×)	85 (1.14×)
Tolerance	SGD	SGD+momentum	SGD+RNA	SGD+momentum+RNA
5.0%	69 (0.87×)	60	26 (2.31×)	**24** (2.50×)
2.0%	83 (0.99×)	82	52 (1.58×)	**45** (1.82×)
1.0%	84 (1.02×)	86	71 (1.21×)	**60** (1.43×)
0.5%	89 (0.98×)	87	73 (1.19×)	**62** (1.40×)
0.2%	**N/A**	90	99 (0.90×)	**63** (1.43×)
Tolerance	SGD	SGD+momentum	SGD+RNA	SGD+momentum+RNA
5.0%	65 (0.86×)	56	22 (2.55×)	**13** (4.31×)
2.0%	80 (0.98×)	78	45 (1.73×)	**38** (2.05×)
1.0%	83 (1.00×)	83	60 (1.38×)	**56** (1.48×)
0.5%	87 (0.99×)	86	80 (1.08×)	**66** (1.30×)
0.2%	92 (1.01×)	93	86 (1.08×)	**75** (1.24×)

TABLE 2. Number of epochs required to reach the best test accuracy + *Tolerance*% on CIFAR10 with a (*top* to *bottom*) VGG, Resnet18 and Densenet, using several algorithms. The best accuracies are 6.54% (VGG), 5.0% (Resnet-18) and 4.62%(Densenet). The speed-up compared to the SGD+momentum baseline is in parenthesis.

6.2.1. *Smooth Problems.* We consider ridge regression and l_2 regularized logistic regression problems which are of the form,

$$h(x) := f(Ax) + g(x)$$

where $f(Ax) = \frac{1}{2}\|Ax - b\|^2$ for ridge regression and $f(Ax) = \sum \log(1 + \exp(-a_i^T x b_i))$ for logistic regression, and $g(x) = \frac{\mu}{2}\|x\|^2$. The following methods are tested in this experiment.

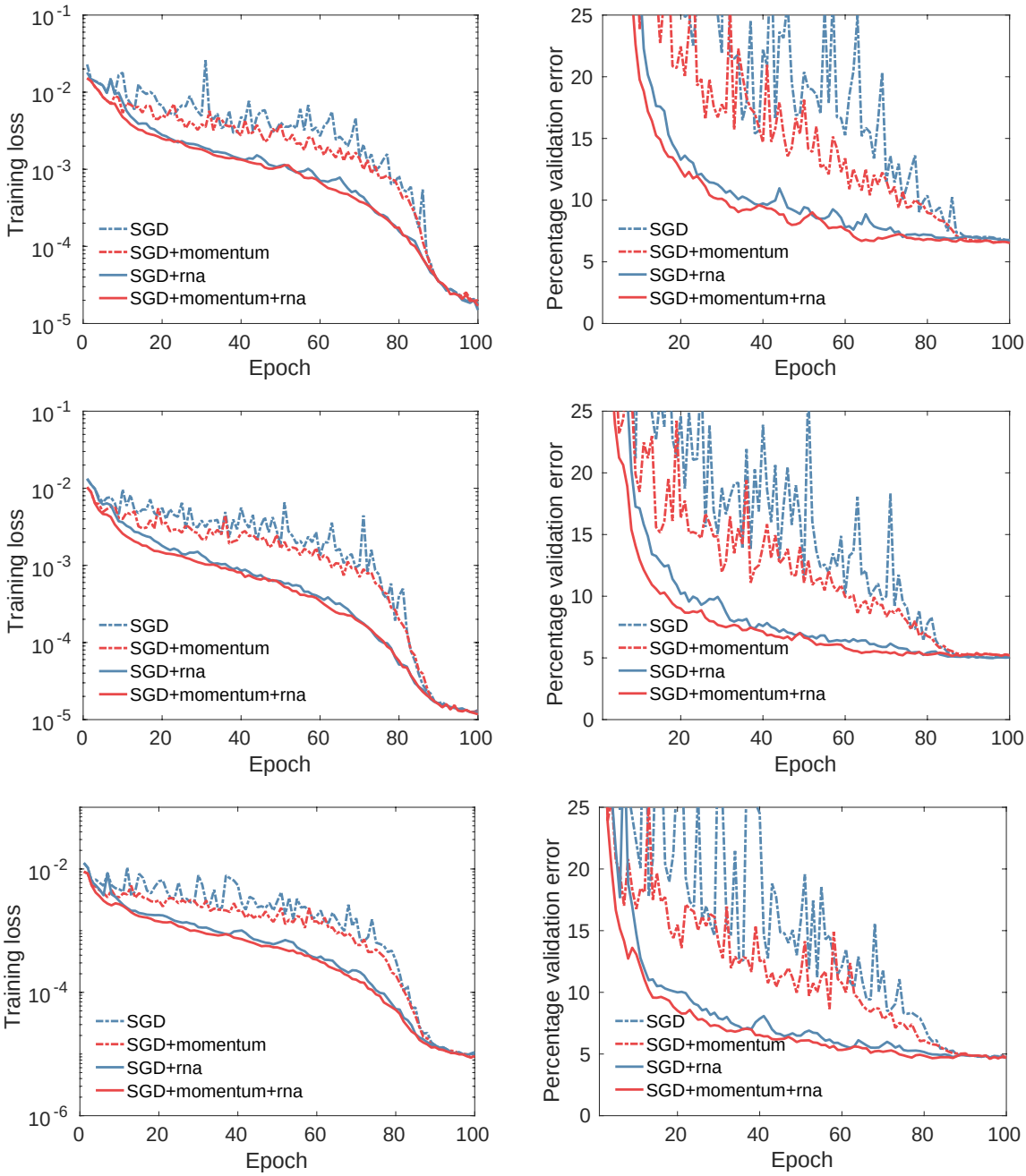

FIGURE 8. (Top to bottom) VGG, Resnet-18 and Densenet networks on Cifar10, 100 epochs. SGD with and without momentum, and their off-line accelerated versions with a window size 5. Left: training loss. Right: top-1 validation error.

- **GD.** The gradient descent method $x_{k+1} = x_k - \frac{1}{L}\nabla h(x_k)$, where L is the Lipschitz constant of the gradient.
- **Nesterov.** The Nesterov's accelerated gradient method [Nesterov, 2013]

$$x_{k+1} = y_k - \frac{1}{L}\nabla h(y_k) \qquad y_{k+1} = y_k + \beta(y_k - y_{k-1})$$

where $\beta = \frac{\sqrt{L}-\sqrt{\mu}}{\sqrt{L}+\sqrt{\mu}}$, L is the Lipschitz constant of the gradient.

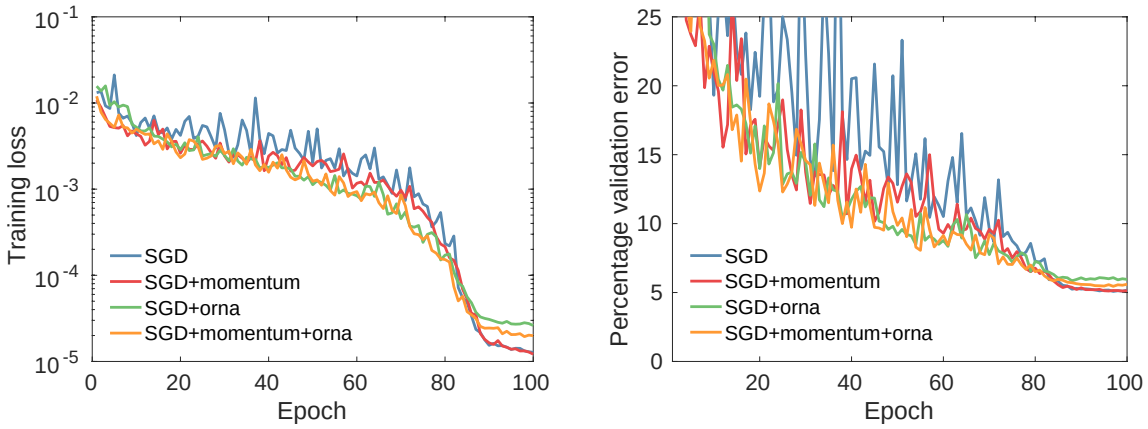

FIGURE 9. Online RNA for training a Resnet-18 on CIFAR-10.

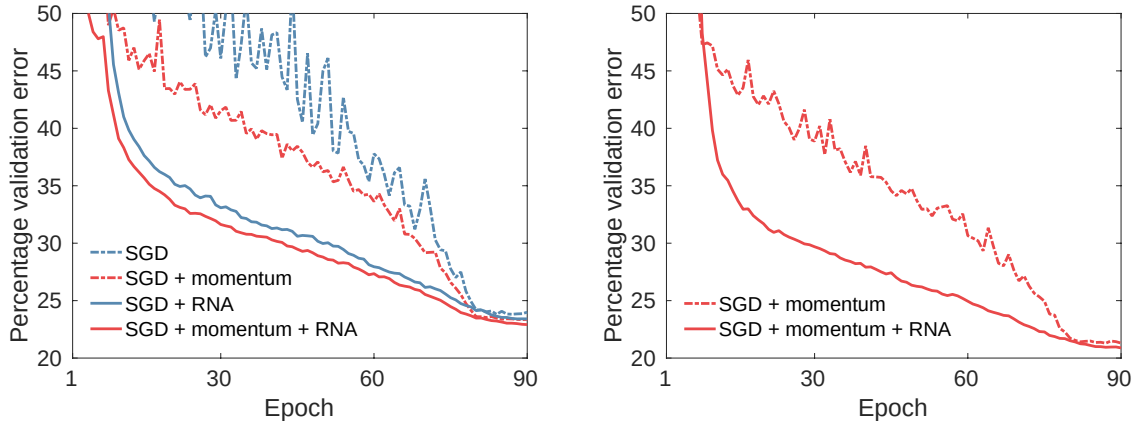

FIGURE 10. Training a Resnet-52 (*left*) and ResNet-152 (*right*) on validation ImageNet for 90 epochs using SGD with and without momentum, and their off-line accelerated versions.

	Pytorch	SGD	SGD+mom.	SGD+RNA	SGD+mom.+RNA
Resnet-50	23.85	23.808	23.346	23.412 (-0.396%)	**22.914** (-0.432%)
Resnet-152	21.69	N/A	21.294	N/A	**20.884** (-0.410%)

TABLE 3. Best validation top-1 error percentage on ImageNet. In parenthesis the improvement due to RNA. The first column corresponds to the performance of Pytorch pre-trained models.

- **L-BFGS.** The L-BFGS method [Liu and Nocedal, 1989] $x_{k+1} = x_k - \alpha_k H_k \nabla h(x_k)$ where the steplength parameter α_k is chosen via Armijo backtracking line search and the memory parameter is chosen to be 10.
- **PDGM.** The primal-dual gradient method [Chambolle and Pock, 2011, Mizoguchi, 1960]

$$y_{k+1} = Prox^{\sigma}_{f^*}(y_k + \sigma A x_k) \qquad x_{k+1} = Prox^{\tau}_{g}(x_k - \tau A^* y_{k+1})$$

where $\sigma = \frac{1}{\|A\|}\sqrt{\frac{\mu}{\delta}}$, $\tau = \frac{1}{\|A\|}\sqrt{\frac{\delta}{\mu}}$, δ is the strong convexity parameters of f^*.

- **PDGM + Momentum.** The primal-dual gradient method with momentum [Chambolle and Pock, 2011]

$$y_{k+1} = Prox^{\sigma}_{f^*}(y_k + \sigma A\bar{x}_k) \quad x_{k+1} = Prox^{\tau}_{g}(x_k - \tau A^* y_{k+1})$$
$$\bar{x}_{k+1} = x_{k+1} + \theta(x_{k+1} - x_k)$$

where $\sigma = \frac{1}{\|A\|}\sqrt{\frac{\mu}{\delta}}$, $\tau = \frac{1}{\|A\|}\sqrt{\frac{\delta}{\mu}}$, $\theta = \frac{1}{1+\frac{2\sqrt{\mu\delta}}{\|A\|}}$, δ is the strong convexity parameters of f^*.

The Lipschitz constant L is $\|A\|^2 + \mu$ for ridge regression and is $\frac{\|A\|^2}{4} + \mu$ for logistic regression. The strong convexity parameter δ of the dual function f^* is 1 for ridge regression and is 4 for logistic regression. The proximal operators used in the primal - dual algorithms have closed form solutions for ridge regression. That is, $Prox^{\tau}_{g}(x) = \frac{1}{1+\tau\mu}$ and $Prox^{\sigma}_{f^*}(y) = \frac{y-\sigma b}{1+\sigma}$. In logistic regression, the approximate proximal operator of f^* is obtained by running Newton's method till some tolerance on the accuracy is achieved or a maximum of 100 iterations is reached. Note that the dominant cost in computing the gradients or proximal operators is the cost of computing the matrix vector products Ax and A^*y which are of the order $O(Nd)$ and the cost of performing Newton's method to obtain the proximal operator is of order N times the maximum number of iterations t. Therefore, when $t < d$ one can ignore the additional cost of performing Newton's method.

We use online RNA on GD, Nesterov and PDGM with a fixed window size $m = 10$ and set $\lambda = 10^{-8}\|R^T R\|_2$. As discussed in Section 4, RNA can be applied only with specific choices of the step-length parameters in the case of primal-dual methods. In the case of smooth problems, we observe that the choice $\tau = \frac{1}{\|A\|}$ and $\sigma = \frac{1}{\|A\|}$ yields stability for applying RNA on PDGM. We note this choice is not an optimal choice and one can improve the results by suitably tuning these parameters.

Figure 11 shows the performance of different variants of the primal-dual algorithms on ridge regression problems for two different regularization constants. We observe that there is no significant difference in the performance of the method with the momentum term (θ) as compared to the one with no momentum term. We also observe that although the choice of the steplength parameters mentioned above have consistent performance across different problems, the improvements obtained with RNA are not very significant. However, choosing $\sigma = \tau = 1/\|A\|$ and applying RNA to the PDGM has consistently outperformed all other variants. This is in consistent with theoretical observations made in Section 4 that one can find optimal steplength parameters for which RNA is stable and obtains the optimal performance.

Figure 12 compares the performance of primal-dual algorithms with other well know algorithms on ridge regression problems. We observe that Nesterov's accelerated gradient method and primal-dual gradient method consistently outperformed gradient descent as suggested by the theory as these methods achieve the optimal rates. The RNA variants of gradient descent and primal-dual methods are competitive and outperform their base algorithms.

Figure 13 shows the performance of the methods on logistic regression problems. We observe that the RNA variants have substantially improved the performance of the base algorithms. The L-BFGS method with Armijo backtracking line-search has the optimal performance across different problems and the RNA variants are competitive to this method.

We now illustrate the effects of RNA on Nesterov's accelerated gradient method. As discussed in Section 4, RNA is applied to sequence of iterates that are obtained after regular intervals, and the length of the interval needs to be chosen based on the problem characteristics. Figures 14 and 15 compare the performance of RNA on Nesterov's sequence of iterates for various interval lengths p. We observe that the length of the interval has significant effect on the performance of the algorithm and this choice depends on the trade off between stability and speed of convergence. That is, the larger the interval length, higher the chance of getting an accelerated sequence but lower the speed of convergence. Higher powers are generally needed for highly ill-conditioned problems. Due to these difficulties, it is clear that for simple momentum terms, one should consider the symmetric part of these iterations and apply RNA on these sequences as discussed in Section 5. We report the results with this approach in the next section.

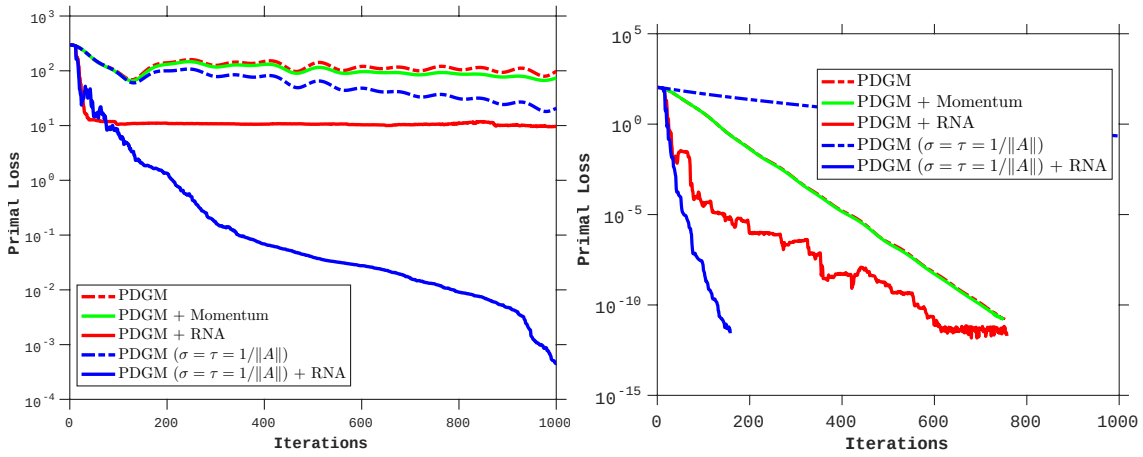

FIGURE 11. Quadratic loss on the Madelon [Guyon, 2003]. Left : $\mu = 10^{-2}$. Right : $\mu = 10^{2}$. Comparison of online RNA with other variants of primal-dual gradient methods.

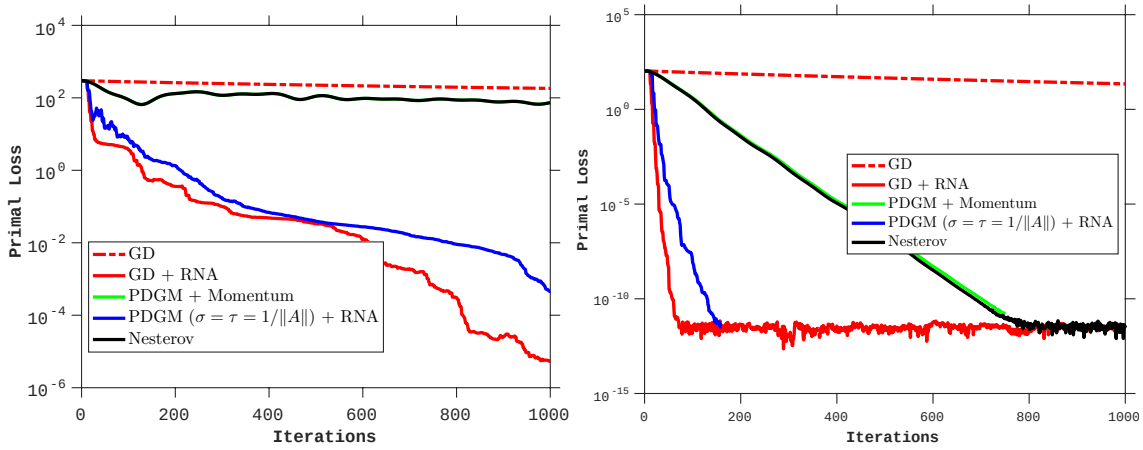

FIGURE 12. Quadratic loss on the Madelon [Guyon, 2003]. Left : $\mu = 10^{-2}$. Right : $\mu = 10^{2}$. Comparison of online RNA on primal-dual gradient methods with other first-order algorithms.

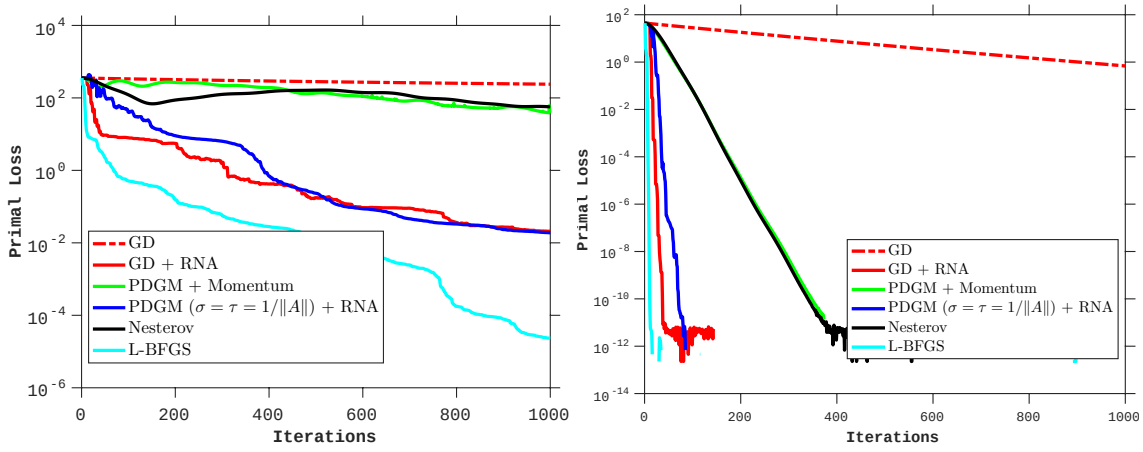

FIGURE 13. Logistic loss on the Madelon [Guyon, 2003]. Left : $\mu = 10^{-2}$. Right : $\mu = 10^{2}$. Comparison of online RNA on primal-dual gradient methods with other first-order algorithms.

Lastly, we compare the performance of offline, restart and online versions of RNA on primal-dual gradient methods in Figure 16. We observe that the improvement in the performance is more pronounced in the online version of RNA as compared to the offline version.

6.2.2. *Non-Smooth Problems.* We consider denoising an image that is degraded by Gaussian noise using total variation. We refer the reader to Chambolle and Pock [2016] for details about the total variation models. The optimization problem is given as,

$$\min_x \|\nabla x\|_1 + \frac{\mu}{2}\|x-b\|^2$$

where,

$$\|\nabla x\|_1 = \sum_{i,j} |(\nabla x)_{i,j}| \qquad |(\nabla x)_{i,j}| = \sqrt{((\nabla x)^1_{i,j})^2 + ((\nabla x)^2_{i,j})^2}$$

and b is a 256 by 256 noisy input image. This optimization problem is in the form (31) with $f(\nabla x) = \|\nabla x\|_1$ and $g(x) = \frac{\mu}{2}\|x-b\|^2$. The gradient operator ∇x is discretized by forward differencing (see Chambolle and Pock [2011]). The convex conjugate of f is an indicator function of the convex set P where,

$$P = \{p : \|p\|_\infty \leq 1\}, \qquad \|p\|_\infty = \max_{i,j} |p_{i,j}|, \qquad |p_{i,j}| = \sqrt{(p^1_{i,j})^2 + (p^2_{i,j})^2}$$

and so the proximal operator is a point wise projection on to this set. That is $Prox^\sigma_{f^*}(p)_{i,j} = \frac{p_{i,j}}{\max(1,|p_{i,j}|)}$.

We compare the performance of the two variants of primal-dual methods with RNA for two different noise levels ζ with two different regularization constants μ. The step-length parameters are chosen adaptively at each iteration as follows:

- **PDGM**

$$\hat{\theta}_k = \frac{1}{\sqrt{1+2\gamma\tau_k}} \qquad \sigma_{k+1} = \frac{\sigma_k}{\hat{\theta}_k} \qquad \tau_{k+1} = \tau_k\hat{\theta}_k$$

  with $\gamma = 0.2\mu$, $\theta = 0$, $\tau_0 = 0.02$, $\sigma_0 = \frac{4}{\tau_0\|\nabla\|^2}$
- **PDGM + Momentum**

$$\theta_k = \frac{1}{\sqrt{1+2\gamma\tau_k}} \qquad \sigma_{k+1} = \frac{\sigma_k}{\theta_k} \qquad \tau_{k+1} = \tau_k\theta_k$$

  with $\gamma = 0.7\mu$ and $\sigma_0 = \tau_0 = \frac{1}{\|\nabla\|}$

with $\|\nabla\|^2 = 8$. These adaptive choices are the standard choices used in the literature and yield the optimal theoretical convergence rates for the momentum variants. We note that these parameters are not carefully fine-tuned to give the best performance for each variant but are chosen based on some simple observations. We used the offline RNA instead of online RNA as we consistently observed that the offline RNA is more robust in the high accuracy regime and the online variants needed some stability inducing techniques like linesearches. Moreover, for the online RNA, the improvement in the performance on these non-smooth problems is small and so the additional cost of solving the linear system is not well justified.

Table 4 reports the number of iterations required for the distance between the primal function value and the optimal primal function value to be below certain accuracy. We observe that the PDGM + RNA has consistently outperformed the PDGM and its' momentum variant for all the accuracies.

	$\zeta = 0.1, \mu = 8$			$\zeta = 0.05, \mu = 16$		
	$\epsilon = 10^{-2}$	$\epsilon = 10^{-4}$	$\epsilon = 10^{-6}$	$\epsilon = 10^{-2}$	$\epsilon = 10^{-4}$	$\epsilon = 10^{-6}$
PDGM	488	1842	7146	257	943	3706
PDGM + Momentum	377	1744	6813	226	921	3879
PDGM + offlineRNA	**221**	**1151**	**5801**	**141**	**671**	**3241**

TABLE 4. Number of iterations required for the primal accuracy to be below ϵ on the images shown in Figure 17 using primal-dual gradient methods.

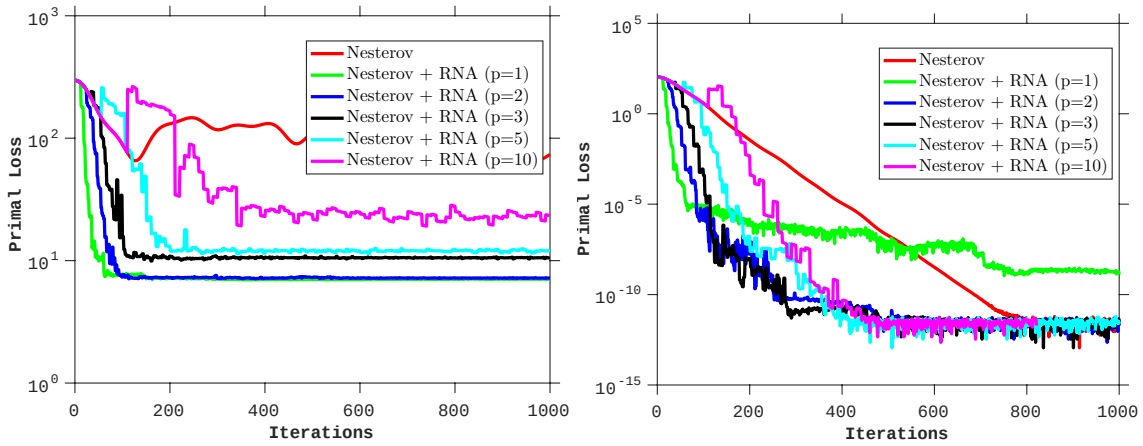

FIGURE 14. Quadratic loss on the Madelon [Guyon, 2003]. Left : $\mu = 10^{-2}$. Right : $\mu = 10^{2}$. Comparison of online RNA on different iterate sampling rates, i.e. different powers of the operator for Nesterov's strongly convex acceleration algorithm.

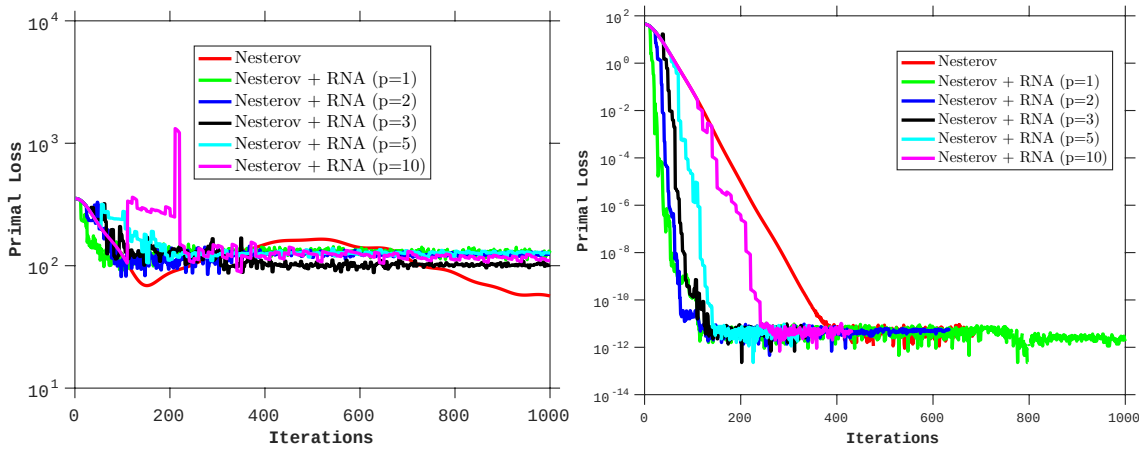

FIGURE 15. Logistic loss on Madelon [Guyon, 2003]. Left : $\mu = 10^{-2}$. Right : $\mu = 10^{2}$. Comparison of online RNA on different iterate sampling rates, i.e. different powers of the operator for Nesterov's strongly convex acceleration algorithm.

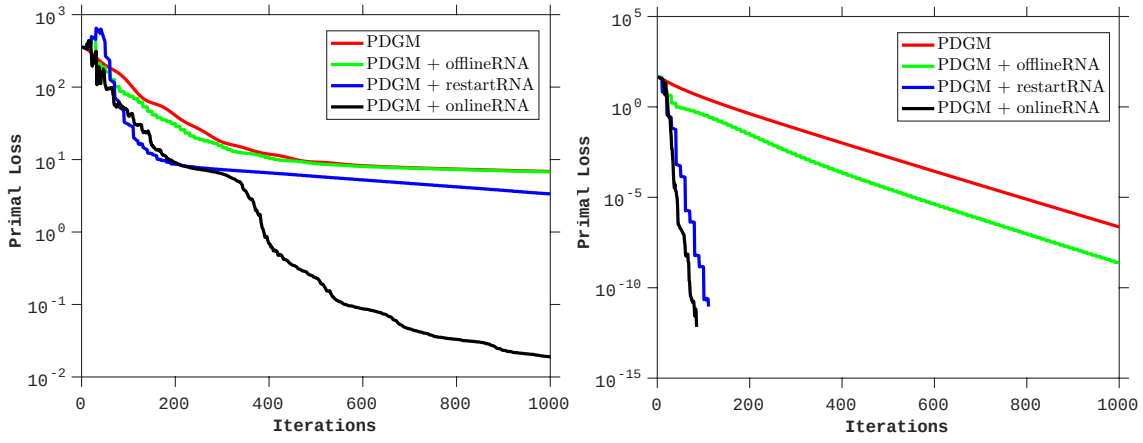

FIGURE 16. Logistic loss on the Madelon [Guyon, 2003]. Left : $\mu = 10^{-2}$. Right : $\mu = 10^{2}$. Comparison of offline, restart and online variants of RNA on primal-dual gradient methods.

FIGURE 17. Images used in the experiments. Left: True data. Middle: Noisy data with Gaussian noise $\zeta = 0.1$. Right: Noisy data with Gaussian noise $\zeta = 0.05$

ACKNOWLEDGEMENTS

AA is at CNRS & département d'informatique, École normale supérieure, UMR CNRS 8548, 45 rue d'Ulm 75005 Paris, France, INRIA and PSL Research University. The authors would like to acknowledge support from the *data science* joint research initiative with the *fonds AXA pour la recherche* and Kamet Ventures, as well as a Google focused award. DS was supported by a European Union Seventh Framework Programme (FP7- PEOPLE-2013-ITN) under grant agreement n.607290 SpaRTaN. RB was supported by Department of Energy grant DE-FG02-87ER25047 and DARPA grant 650-4736000-60049398.

REFERENCES

Raghu Bollapragada, Dheevatsa Mudigere, Jorge Nocedal, Hao-Jun Michael Shi, and Ping Tak Peter Tang. A progressive batching L-BFGS method for machine learning. *arXiv preprint arXiv:1802.05374*, 2018.

Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

Claude Brezinski and M Redivo Zaglia. *Extrapolation methods: theory and practice*, volume 2. Elsevier, 2013.

Stan Cabay and LW Jackson. A polynomial extrapolation method for finding limits and antilimits of vector sequences. *SIAM Journal on Numerical Analysis*, 13(5):734–752, 1976.

Antonin Chambolle and Thomas Pock. A first-order primal-dual algorithm for convex problems with applications to imaging. *Journal of mathematical imaging and vision*, 40(1):120–145, 2011.

Antonin Chambolle and Thomas Pock. An introduction to continuous optimization for imaging. *Acta Numerica*, 25: 161–319, 2016.

Daeshik Choi and Anne Greenbaum. Roots of matrices in the study of gmres convergence and crouzeix's conjecture. *SIAM Journal on Matrix Analysis and Applications*, 36(1):289–301, 2015.

Michel Crouzeix. Bounds for analytical functions of matrices. *Integral Equations and Operator Theory*, 48(4):461–477, 2004.

Michel Crouzeix. Numerical range and functional calculus in hilbert space. *Journal of Functional Analysis*, 244(2): 668–690, 2007.

Michel Crouzeix and César Palencia. The numerical range as a spectral set. *arXiv preprint arXiv:1702.00668*, 2017.

Li Deng, Jinyu Li, Jui-Ting Huang, Kaisheng Yao, Dong Yu, Frank Seide, Michael Seltzer, Geoff Zweig, Xiaodong He, Jason Williams, et al. Recent advances in deep learning for speech research at microsoft. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 8604–8608. IEEE, 2013.

William F. Donoghue. On the numerical range of a bounded operator. *Michigan Math. J.*, 4(3):261–263, 1957. doi: 10.1307/mmj/1028997958. URL https://doi.org/10.1307/mmj/1028997958.

RP Eddy. Extrapolating to the limit of a vector sequence. In *Information linkage between applied mathematics and industry*, pages 387–396. Elsevier, 1979.

Bernd Fischer and Roland Freund. Chebyshev polynomials are not always optimal. *Journal of Approximation Theory*, 65(3):261–272, 1991.

Gene H Golub and Richard S Varga. Chebyshev semi-iterative methods, successive overrelaxation iterative methods, and second order richardson iterative methods. *Numerische Mathematik*, 3(1):157–168, 1961.

R Paul Gorman and Terrence J Sejnowski. Analysis of hidden units in a layered network trained to classify sonar targets. *Neural Networks*, 1:75, 1988.

Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

Anne Greenbaum, Adrian S Lewis, and Michael L Overton. Variational analysis of the crouzeix ratio. *Mathematical Programming*, 164(1-2):229–243, 2017.

Isabelle Guyon. Design of experiments of the nips 2003 variable selection benchmark, 2003.

Felix Hausdorff. Der wertvorrat einer bilinearform. *Mathematische Zeitschrift*, 3(1):314–316, 1919.

Charles R Johnson. Computation of the field of values of a $2\times$ 2 matrix. *J. Res. Nat. Bur. Standards Sect. B*, 78:105, 1974.

Charles R Johnson. Numerical determination of the field of values of a general complex matrix. *SIAM Journal on Numerical Analysis*, 15(3):595–602, 1978.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

J. B. Lasserre. Global optimization with polynomials and the problem of moments. *SIAM Journal on Optimization*, 11(3):796–817, 2001.

A. Lewis and M. Overton. Partial smoothness of the numerical radius at matrices whose fields of values are disks. *Working paper (mimeo)*, 2018.

Dong C Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1-3):503–528, 1989.

Toshiyuki Mizoguchi. K.j. arrow, l. hurwicz and h. uzawa, studies in linear and non-linear programming. *Economic Review*, 11(3):349–351, 1960. URL https://EconPapers.repec.org/RePEc:hit:ecorev:v:11:y:1960:i:3:p:349-351.

Eric Moulines and Francis Bach. Non-asymptotic analysis of stochastic approximation algorithms for machine learning. In *Advances in Neural Information Processing Systems*, 2011.

Y. Nesterov. A method of solving a convex programming problem with convergence rate $O(1/k^2)$. *Soviet Mathematics Doklady*, 27(2):372–376, 1983.

Yurii Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2013.

Boris T Polyak and Anatoli B Juditsky. Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization*, 30(4):838–855, 1992.

Sashank J Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond. In *International Conference on Learning Representations*, 2018.

Youcef Saad and Martin H Schultz. Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on scientific and statistical computing*, 7(3):856–869, 1986.

Damien Scieur, Alexandre d'Aspremont, and Francis Bach. Regularized nonlinear acceleration. In *Advances In Neural Information Processing Systems*, pages 712–720, 2016.

Damien Scieur, Francis Bach, and Alexandre d'Aspremont. Nonlinear acceleration of stochastic algorithms. In *Advances in Neural Information Processing Systems*, pages 3985–3994, 2017.

Damien Scieur, Edouard Oyallon, Alexandre dAspremont, and Francis Bach. Nonlinear acceleration of cnns. In *Workshop track of International Conference on Learning Representations (ICLR)*, 2018.

Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

Otto Toeplitz. Das algebraische analogon zu einem satze von fejér. *Mathematische Zeitschrift*, 2(1-2):187–197, 1918.

Homer F Walker and Peng Ni. Anderson acceleration for fixed-point iterations. *SIAM Journal on Numerical Analysis*, 49(4):1715–1735, 2011.

DEPARTMENT OF INDUSTRIAL ENGINEERING AND MANAGEMENT SCIENCES
NORTHWESTERN UNIVERSITY.
E-mail address: raghu.bollapragada@u.northwestern.edu

INRIA & D.I., UMR 8548,
ÉCOLE NORMALE SUPÉRIEURE, PARIS, FRANCE.
E-mail address: damien.scieur@inria.fr

CNRS & D.I., UMR 8548,
ÉCOLE NORMALE SUPÉRIEURE, PARIS, FRANCE.
E-mail address: aspremon@ens.fr