AN EFFICIENT AND ROBUST SAV BASED ALGORITHM FOR DISCRETE GRADIENT SYSTEMS ARISING FROM OPTIMIZATIONS

XINYU LIU[1], JIE SHEN[1], AND XIAONGXIONG ZHANG[1]

Abstract. We propose in this paper a new minimization algorithm based on a slightly modified version of the scalar auxialiary variable (SAV) approach coupled with a relaxation step and an adaptive strategy. It enjoys several distinct advantages over popular gradient based methods: (i) it is unconditionally energy diminishing with a modified energy which is intrinsically related to the original energy, thus no parameter tuning is needed for stability; (ii) it allows the use of large step-sizes which can effectively accelerate the convergence rate. We also present a convergence analysis for some SAV based algorithms, which include our new algorithm without the relaxation step as a special case. We apply our new algorithm to several illustrative and benchmark problems, and compare its performance with several popular gradient based methods. The numerical results indicate that the new algorithm is very robust, and its adaptive version usually converges significantly faster than those popular gradient descent based methods.

1. Introduction

Minimization plays an important role in many branches of science and engineering. In particular, how to accelerate the convergence rate of the minimization process is a central issue in data science and machine learning problems. We consider in this paper an unconstrained minimization problem

$$\min_{\theta\in\mathbb{R}^N} f(\theta) \tag{1.1}$$

which arises in many applications, including particularly machine learning problems. For large scale minimization problems, the first order methods such as gradient descent, its variants such as stochastic gradient descent [21], Nesterov's accelerated gradient descent [17], adaptive momentum estimation method [15, 24, 10], are popular choices. We refer to [19, 18, 23], and the references therein, for more detail on the design and analysis of gradient descent method and its various variants.

The vanilla gradient descent method for (1.1) is

$$\theta_{k+1} = \theta_k - \eta_k \nabla f(\theta_k), \tag{1.2}$$

2020 *Mathematics Subject Classification.* 65K10,49M05,90C26.

Key words and phrases. discrete gradient system; optimization; scalar auxiliary variable; adaptive; stability; convergence.

[1]Department of Mathematics, Purdue University, West Lafayette, IN 47907, USA (liu1957@purdue.edu, shen7@purdue.edu, zhan1966@purdue.edu).

which can also be regarded as a numerical scheme for the gradient flow

$$\theta_t = -\nabla f(\theta), \tag{1.3}$$

with time step η_k. The gradient flow (1.3) is energy diminishing in the sense that

$$\frac{d}{dt} f(\theta) = (\nabla f(\theta), \theta_t) = -(\theta_t, \theta_t) = -\|\theta_t\|^2 \leq 0,$$

where $(\cdot,\cdot)$ (resp. $\|\cdot\|$) denotes the l^2 inner product (resp. norm). However, gradient decent type schemes are not necessarily energy diminishing, and may blow up if the time step is too large. Although the stability of gradient descent based methods is well understood, the main challenge in practice is how to choose the step-size, i.e. learning rate, to balance between stability and efficiency [22].

We propose in this paper a different class of minimization algorithms inspired from the recently developed scalar auxiliary variable (SAV) approach for gradient flows [25, 26]. The SAV approach enjoys a particular advantage of unconditional energy diminishing compared to popular gradient decent based methods. This advantage avoids tuning step sizes and allows the use of large step sizes, which may effectively accelerate the convergence rate.

Assume the cost function has a splitting

$$f(\theta) = \frac{1}{2}(\mathcal{L}\theta, \theta) + [f(\theta) - \frac{1}{2}(\mathcal{L}\theta, \theta)] := \frac{1}{2}(\mathcal{L}\theta, \theta) + g(\theta), \tag{1.4}$$

where $\mathcal{L}$ is a self-adjoint positive semi-definite linear operator. Note that $\mathcal{L} = 0$ is a trivial splitting. Then, the gradient flow (1.3) becomes

$$\theta_t + \mathcal{L}\theta + \nabla g(\theta) = 0. \tag{1.5}$$

Inspired by the IEQ and SAV approaches [28, 25], assuming there exists $C > 0$ such that $g(\theta) > -C$ for all θ, we introduce a scalar auxiliary variable $r(t) = \sqrt{g(\theta) + C}$, and expand (1.5) to:

$$\begin{cases} \theta_t + \mathcal{L}\theta + \frac{\nabla g(\theta)}{\sqrt{g(\theta)+C}} r = 0, \\ r_t = \frac{1}{2\sqrt{g(\theta)+C}} (\nabla g(\theta), \theta_t). \end{cases} \tag{1.6}$$

Obviously, with $r(0) = \sqrt{g(\theta|_{t=0}) + C}$, the above system admits a solution $r(t) = \sqrt{g(\theta) + C}$ with θ being the solution of (1.5). The main advantage of the expanded system, which includes an energy evolution equation, is that it allows us to construct simple numerical schemes with modified energy diminishing. For example, the following scheme

$$\begin{cases} \frac{\theta_{k+1}-\theta_k}{\delta t} + \mathcal{L}\theta_{k+1} + \frac{\nabla g(\theta_k)}{\sqrt{g(\theta_k)+C}} r_{k+1} = 0, \\ \frac{r_{k+1}-r_k}{\delta t} = \left(\frac{\nabla g(\theta_k)}{2\sqrt{g(\theta_k)+C}}, \frac{\theta_{k+1}-\theta_k}{\delta t} \right), \end{cases} \tag{1.7}$$

can be easily implemented by solving only two linear systems of the form $(I + \delta t\mathcal{L})x = b$, and is unconditionally energy stable with a modified energy [26].

While the scheme (1.7) has been shown to be very effective for gradient flows, it is not particularly suitable for the minimization problem (1.1). Indeed, for any fixed δt, assuming

$\theta_k \to \theta_*$ and $r_k \to r_*$, then $\frac{r_k}{\sqrt{g(\theta_k)+C}} \to \frac{r_*}{\sqrt{g(\theta_*)+C}}$ which is usually not equal to 1, and consequently, the first equation of (1.7) leads to

$$0 = \mathcal{L}\theta_* + \frac{r_*}{\sqrt{g(\theta_*)+C}}\nabla g(\theta_*) = \mathcal{L}\theta_* + \frac{r_*}{\sqrt{g(\theta_*)+C}}\left(-\mathcal{L}\theta_* + \nabla f(\theta_*)\right).$$

If $\mathcal{L} \neq 0$, we observe from the above that in general $\nabla f(\theta_*) \neq 0$ thus θ_* is not a solution for (1.1). Another complication of this approach is that it is not obvious how to choose $\mathcal{L}$ such that $g(\theta)$ is bounded from below for all θ. The main goal of this paper is to design suitable SAV based schemes for (1.1), develop their convergence theory, and validate them through extensive numerical experiments.

The rest of the paper is organized as follows. In Section 2, we first discuss a suitable SAV algorithm for minimization, introduce its relaxed version RSAV, and discuss the adaptive rule and choices for the non-negative operator $\mathcal{L}(\theta)$. Then, we present in Section 3 several numerical results to show the performance of the RSAV in different optimization problems. We provide a convergence study in Section 4, some concluding remarks in Section 5.

2. A new SAV approach and its relaxed version

We have observed in the last section that the standard SAV approach is not suitable for the minimization problem (1.1). In this section, we propose a different SAV approach and its related version which are well suited for (1.1).

2.1. **A modified SAV approach.**

We still assume the splitting (1.4), and rewrite (1.3) as

$$\theta_t + \mathcal{L}\theta + \nabla f(\theta) - \mathcal{L}\theta = 0. \tag{2.1}$$

Since $f(\theta)$ in a minimization problem should always be bounded from below, there exists $C > 0$ such that $f(\theta) > -C$ for all θ. We introduce $r(t) = \sqrt{f(\theta)+C}$, and expand (2.1) to:

$$\begin{cases} \theta_t + \mathcal{L}\theta + \frac{\nabla f(\theta)}{\sqrt{f(\theta)+C}} r - \mathcal{L}\theta = 0, \\ r_t = \frac{1}{2\sqrt{f(\theta)+C}}(\nabla f(\theta), \theta_t). \end{cases} \tag{2.2}$$

Then, a simple SAV scheme to approximate the above is

$$\begin{cases} \frac{\theta_{k+1}-\theta_k}{\delta t} + \mathcal{L}\theta_{k+1} + \frac{\nabla f(\theta_k)}{\sqrt{f(\theta_k)+C}} r_{k+1} - \mathcal{L}\theta_k = 0, \\ \frac{r_{k+1}-r_k}{\delta t} = \left(\frac{\nabla f(\theta_k)}{2\sqrt{f(\theta_k)+C}}, \frac{\theta_{k+1}-\theta_k}{\delta t}\right). \end{cases} \tag{2.3}$$

Note that if $\theta_k \to \theta_*$ and $r_k \to r_*$, then (2.3) leads to $\nabla f(\theta_*) = 0$, and consequently θ_* is a solution of (1.1).

The scheme (2.3) leads to a coupled linear system for (θ_{k+1}, r_{k+1}), but it can be implemented explicitly after solving a linear system as will be shown in the Section 4.1. Let

$A = I + \delta t\mathcal{L}$, then (2.3) can be equivalently implemented as

$$r_{k+1} = \frac{1}{1+\delta t\frac{(\nabla f(\theta_k), A^{-1}\nabla f(\theta_k))}{2[f(\theta_k)+C]}} r_k,$$
$$\theta_{k+1} = \theta_k - \frac{r_{k+1}}{\sqrt{f(\theta_k)+C}}\delta t A^{-1}\nabla f(\theta_k).$$

Moreover, taking the discrete inner product of the first (resp. second) equation in (2.3) with $\theta_{k+1}-\theta_k$ (resp. $2r_{k+1}$), summing up the results, we obtain the following:

Theorem 1. *If $\mathcal{L}$ is non-negative, then for any $\delta t > 0$, the modified energy r^2 in the scheme* (2.7) *is unconditionally diminishing in the sense that*

$$r_{k+1}^2 - r_k^2 = -\frac{1}{\delta t}\|\theta_{k+1}-\theta_k\|^2 - (\mathcal{L}(\theta_{k+1}-\theta_k), (\theta_{k+1}-\theta_k)) - (r_{k+1}-r_k)^2.$$

The above result shows the key advantage of (2.3): the energy dissipation holds for any $\delta t > 0$ and any splitting with non-negative $\mathcal{L}$.

As we shall demonstrate in numerical tests in Section 3, when the cost functional $f(\theta)$ has a suitable splitting, the above algorithm usually converge much faster the vanilla gradient decent method. When the cost function does not have any obvious quadratic part, we can either choose any suitable non-negative linear operator $\mathcal{L}$ in (1.4), or simply take $\mathcal{L} = 0$, which results in a fully explicit method:

$$\begin{cases} \frac{\theta_{k+1}-\theta_k}{\delta t} + \frac{\nabla f(\theta_k)}{\sqrt{f(\theta_k)+C}} r_{k+1} = 0, \\ \frac{r_{k+1}-r_k}{\delta t} = \left(\frac{\nabla f(\theta_k)}{2\sqrt{f(\theta_k)+C}}, \frac{\theta_{k+1}-\theta_k}{\delta t}\right), \end{cases} \tag{2.4}$$

which, we refer as *the SAV gradient descent method*. As will be shown in the Section 4.1, the scheme (2.4) can be decoupled and implemented as

$$\begin{aligned} r_{k+1} &= \frac{r_k}{1+\delta t\frac{(\nabla f(\theta_k), \nabla f(\theta_k))}{2(f(\theta_k)+C)}}, \\ \theta_{k+1} &= \theta_k - \delta t\frac{r_{k+1}}{\sqrt{f(\theta_k)+C}}\nabla f(\theta_k). \end{aligned} \tag{2.5}$$

Compared with the vanilla gradient descent method (1.2), there are extra computational costs of computing both $f(\theta_k)$ and $(\nabla f(\theta_k), \nabla f(\theta_k))$ in (2.5), but Theorem 1 ensures stability for any δt. In contrast, δt in (1.2) needs to be small enough to ensure stability.

2.2. **A relaxed version of the modified SAV approach.** While for fixed δt, the solution of the SAV scheme (2.3) converges to a solution of the minimization problem (1.1), the evolution of r_{k+1} is not directly linked to $\sqrt{f(\theta_{k+1})+C}$, and its value may decrease rapidly to ensure stability when $\|\nabla f(\theta_k) - \mathcal{L}\theta_k\|$ becomes large. In this case, the ratio $\frac{r_{k+1}}{\sqrt{f(\theta_{k+1})+C}}$ may deviate significantly from 1, which indicates that r_{k+1} is no longer a good approximation of $\sqrt{f(\theta(t_{k+1}))+C}$, thus θ_{k+1} will not be a good approximation of $\theta(t_{k+1})$. For dynamic simulation of gradient flows, a remedy is to monitor the ratio $\frac{r_{k+1}}{\sqrt{f(\theta_{k+1})+C}}$ and

adjust the time step so that it stays close to 1. For the minimization problem (1.1), since we are mainly interested in the steady steady state solutions of (1.3), it is found in [32] that setting $r_{k+1} = \sqrt{f(\theta_{k+1}) + C}$ at each time step is also very effective. More precisely, we can use the following modified SAV scheme:

$$\begin{cases} \frac{\theta_{k+1}-\theta_k}{\delta t} + \mathcal{L}\theta_{k+1} + \frac{\nabla f(\theta_k)}{\sqrt{f(\theta_k)+C}}\tilde{r}_{k+1} - \mathcal{L}\theta_k = 0, \\ \frac{\tilde{r}_{k+1}-r_k}{\delta t} = \left(\frac{\nabla f(\theta_k)}{2\sqrt{f(\theta_k)+C}}, \frac{\theta_{k+1}-\theta_k}{\delta t}\right), \\ r_{k+1} = \sqrt{f(\theta_{k+1}) + C}. \end{cases} \tag{2.6}$$

However, the above modified SAV scheme is no longer energy diminishing. Recently, another way to link r_{k+1} with $\sqrt{f(\theta_{k+1}) + C}$ while still being energy diminishing is proposed in [30] (see also [14, 29]). When applied to (2.1), the relaxed SAV method takes the following form:

$$\begin{cases} \frac{\theta_{k+1}-\theta_k}{\delta t} + \mathcal{L}\theta_{k+1} + \frac{\nabla f(\theta_k)}{\sqrt{f(\theta_k)+C}}\tilde{r}_{k+1} - \mathcal{L}\theta_k = 0, \\ \frac{\tilde{r}_{k+1}-r_k}{\delta t} = \left(\frac{\nabla f(\theta_k)}{2\sqrt{f(\theta_k)+C}}, \frac{\theta_{k+1}-\theta_k}{\delta t}\right), \\ r_{k+1} = \xi\tilde{r}_{k+1} + (1-\xi)\sqrt{f(\theta_{k+1}) + C}. \end{cases} \tag{2.7}$$

Here, the relaxation parameter ξ is a scalar chosen from the set

$$\mathcal{V} = \{\xi \in [0,1] : (r_{k+1})^2 - (\tilde{r}_{k+1})^2 - (\tilde{r}_{k+1} - r_k)^2 \leq \eta\mathcal{G}(\theta_{k+1}, \theta_k)\} \tag{2.8}$$

where $\mathcal{G}(\theta_{k+1}, \theta_k) = \frac{1}{\delta t}\left((\theta_{k+1} - \theta_k), A(\theta_{k+1} - \theta_k)\right)$ with $A = I + \delta t\mathcal{L}$, and $\eta \in [0, 1]$ is an artificial parameter with default value $\eta = 0.99$. In particular, it is shown in [14] that we can choose

$$\xi = \max\{0, \frac{-b - \sqrt{b^2 - 4ac}}{2a}\}, \tag{2.9}$$

with the coefficients that

$$a = (\tilde{r}_{k+1} - \sqrt{f(\theta_{k+1}) + C})^2 \tag{2.10}$$

$$b = 2\left(\tilde{r}_{k+1} - \sqrt{f(\theta_{k+1}) + C}\right)\sqrt{f(\theta_{k+1}) + C} \tag{2.11}$$

$$c = f(\theta_{k+1}) + C - (\tilde{r}_{k+1})^2 - (\tilde{r}_{k+1} - r_k)^2 - \eta\mathcal{G}(\theta_{k+1}, \theta_k). \tag{2.12}$$

Taking the discrete inner product of the first (resp. second) equation in (2.7) with $\theta_{k+1} - \theta_k$ (resp. $2\tilde{r}_{k+1}$), summing up the results, we get

$$\mathcal{G}(\theta_{k+1}, \theta_k) = \frac{1}{\delta t}\left((\theta_{k+1} - \theta_k), A(\theta_{k+1} - \theta_k)\right) = -2(\tilde{r}_{k+1} - r_k)\tilde{r}_{k+1}, \tag{2.13}$$

then the non-zero choice of ξ can be rewritten as

$$\begin{aligned} \xi &= \frac{-b - \sqrt{b^2 - 4ac}}{2a} = \frac{\sqrt{f(\theta_{k+1}) + C} - \sqrt{(\tilde{r}_{k+1})^2 + (\tilde{r}_{k+1} - r_k)^2 + \eta\mathcal{G}(\theta_{k+1}, \theta_k)}}{\sqrt{f(\theta_{k+1}) + C} - \tilde{r}_{k+1}} \\ &= \frac{\sqrt{f(\theta_{k+1}) + C} - \sqrt{(1-\eta)\tilde{r}_{k+1}^2 + \eta r_k^2 + (1-\eta)(\tilde{r}_{k+1} - r_k)^2}}{\sqrt{f(\theta_{k+1}) + C} - \tilde{r}_{k+1}}. \end{aligned}$$

The implementation of (2.7) is summarized in **Algorithm 1**.

Theorem 2. *If $\mathcal{L}$ is non-negative and linear, then for any $\delta t > 0$, the modified energy r^2 in the scheme (2.7) is unconditionally diminishing in the sense that*

$$r_{k+1}^2 - r_k^2 = -(1-\eta)\mathcal{G}(\theta_{k+1}, \theta_k) \leq 0.$$

Proof. By (2.13), we obtain

$$\tilde{r}_{k+1}^2 - r_k^2 = -\frac{1}{\delta t}\|\theta_{k+1} - \theta_k\|^2 - (\mathcal{L}(\theta_{k+1} - \theta_k), (\theta_{k+1} - \theta_k)) - (\tilde{r}_{k+1} - r_k)^2.$$

Adding $r_{k+1}^2 - \tilde{r}_{k+1}^2$ on both sides, noticing

$$\mathcal{G}(\theta_{k+1}, \theta_k) = \frac{1}{\delta t}\|\theta_{k+1} - \theta_k\|^2 + (\mathcal{L}(\theta_{k+1} - \theta_k), (\theta_{k+1} - \theta_k)),$$

we obtain

$$r_{k+1}^2 - r_k^2 = -\mathcal{G}(\theta_{k+1}, \theta_k) - (\tilde{r}_{k+1} - r_k)^2 + r_{k+1}^2 - \tilde{r}_{k+1}^2,$$

which implies the desired result thanks to (2.8). □

Algorithm 1 The basic RSAV scheme

Inputs:
δt: step-size,
C: constant to guarantee the positivity of $f(x) + C$,
$A = I + \delta t\mathcal{L}$: the linear operator,
θ_0: initial parameter vector
$r_0 \leftarrow \sqrt{f(\theta_0) + C}$
for $k = 0, 1, 2, ..., M-1$ **do**
  $g_k = \frac{\nabla f(\theta_k)}{\sqrt{f(\theta_k)+C}}$
  $\hat{g}_k = A^{-1} g_k$
  $\tilde{r}_{k+1} = \frac{r_k}{1 + \frac{\delta t}{2}(g_k, \hat{g}_k)}$
  $\theta_{k+1} = \theta_k - \delta t \tilde{r}_{k+1} \hat{g}_k$
  $\xi = \frac{\sqrt{f(\theta_{k+1})+C} - \sqrt{(1-\eta)\tilde{r}_{k+1}^2 + \eta r_k^2 + (1-\eta)(\tilde{r}_{k+1} - r_k)^2}}{\sqrt{f(\theta_{k+1})+C} - \tilde{r}_{k+1}}$
  $\xi = \max\{0, \xi\}$
  $r_{k+1} = \xi \tilde{r}_{k+1} + (1-\xi)\sqrt{f(\theta_{k+1}) + C}$
return θ_M

2.3. **Selection of the operator $\mathcal{L}$.** In the SAV approach for gradient flows [26], it is found that a proper choice of the splitting (1.4), i.e., the choice of the quadratic term $\frac{1}{2}(\mathcal{L}\theta, \theta)$, can significantly increase the robustness and efficiency of the SAV schemes. For gradient flows coming from materials science or fluid dynamics, there are usually obvious candidates in the free energy. However, for minimization problems, particularly those from machine learning problems, there are usually no obvious quadratic terms in the energy functions.

In these cases, we can artificially choose some simple operators. In this paper, we consider two simple operators below, for which the inverse operator $(I + \delta t\mathcal{L})^{-1}$ can be efficiently implemented.

2.3.1. *Diagonal Matrix.* In many optimization problems, an l^2 regularization term is often added into the loss function to avoid overfitting to the data in training sets, namely,

$$J(x) = f(x) + \frac{\lambda}{2}\|x\|^2. \tag{2.14}$$

In this case, a natural choice is to set $\mathcal{L} = \lambda I$. More generally, we can use $\mathcal{L} = D$ with D being a diagonal matrix with positive entries, e.g., D can be the diagonal entries of the Hessian of the cost function.

2.3.2. *Discrete Laplacian Matrix.* In some machine learning problems, the discrete Laplacian matrix is used as a smoothing operator which can reduce the variance during the mini-batch training process [20]. This corresponds to $\mathcal{L} = -\sigma\Delta$ where σ is a positive parameter and Δ is a discrete Laplacian matrix, and $(I + \delta t\mathcal{L})^{-1}$ can be efficiently inverted by FFT based methods. The acceleration by using discrete Laplacian in classical primal dual algorithms has been also justified in [13].

2.4. **An adaptive algorithm based on the RSAV scheme** (2.7)**.** Similar to the modified SAV scheme (2.3), the RSAV scheme (2.7) is also unconditionally energy diminishing. A main advantage of unconditionally stable schemes is that one can adaptively adjust the time step size to achieve faster convergence. In particular, we can use

$$I_k(r, \theta) = \frac{r_k}{\sqrt{f(\theta_k) + C}} \tag{2.15}$$

as an indicator to control the deviation between modified energy and true energy. For solving differential equations $\theta_t = -\nabla f(\theta)$, $I_k(r, \theta)$ should be as close as to 1 for the sake of the time accuracy. But for a minimization problem, there is no time accuracy issue thus we can allow $I_k(r, \theta)$ to deviate from 1 to achieve faster convergence. However, $I_k(r, \theta)$ needs to be away from zero to avoid slow convergence, as the SAV and RSAV algorithms may converge much slower than the vanilla gradient descent when the ratio $I_k(r, \theta)$ becomes too small.

We observe from (2.7) that the true step-size for the gradient $\nabla f(\theta_k)$ is $\frac{\tilde{r}_{k+1}}{\sqrt{f(\theta_k)+C}}\delta t$. Thus if the ratio is small i.e. $I_k(r, \theta) < \gamma$, the true step-size for the gradient would be too small resulting in slow convergence. To this end, we present a simple adaptive rule with an adaptive constant $\rho > 1$ with default value $\rho = 1.1$ described in **Algorithm 2**.

Remark 1. *Note that in many applications of neural networks and machine learning, the cost of computing the full batch is generally too high. In these cases, we can adopt the mini-batch approach commonly used in stochastic gradient decent, and restart the RSAV scheme at the beginning of each mini-batch.*

Algorithm 2 The adaptive RSAV scheme

Inputs:
δt_0: initial step-size, δt_{min}: the lower bound of step-size,
C: constant to guarantee the positivity of $f(x)+C$,
$A = I + \delta t \mathcal{L}$: the linear operator,
θ_0: initial parameter vector,
ρ: adaptive constant which is greater than 1,
γ: threshold for the indicator $I(r,\theta)$.
$r_0 \leftarrow \sqrt{f(\theta_0)+C}$: Initialize the SAV,
for $k = 0, 1, 2, ..., M-1$ **do**
  if $\frac{r_k}{\sqrt{f(\theta_k)+C}} < \gamma$ and $\delta t > \delta t_{min}$ **then**
    $\delta t_{k+1} = \max\{\frac{r_k}{\sqrt{f(\theta_k)+C}}\delta t_k, \delta t_{min}\}$
  else
    $\delta t_{k+1} = \rho \delta t_k$

  $g_k = \frac{\nabla f(\theta_k)}{\sqrt{f(\theta_k)+C}}$
  $\hat{g}_k = A^{-1} g_k$
  $\tilde{r}_{k+1} = \frac{r_k}{1+\frac{\delta t_{k+1}}{2}(g_k, \hat{g}_k)}$
  $\theta_{k+1} = \theta_k - \delta t_{k+1} \tilde{r}_{k+1} \hat{g}_k$
  $\xi = \frac{\sqrt{f(\theta_{k+1})+C} - \sqrt{(1-\eta)\tilde{r}_{k+1}^2 + \eta r_k^2 + (1-\eta)(\tilde{r}_{k+1}-r_k)^2}}{\sqrt{f(\theta_{k+1})+C} - \tilde{r}_{k+1}}$
  $\xi = \max\{0, \xi\}$
  $r_{k+1} = \xi \tilde{r}_{k+1} + (1-\xi)\sqrt{f(\theta_{k+1})+C}$
return θ_M

Remark 2. *As a further generalization, we may replace the operator $\mathcal{L}$ in* (2.3) *and* (2.7) *by a linear nonnegative operator $\mathcal{L}_k$ which may depend on θ_k at each step. Then, Theorems 1 and 2 still hold with $\mathcal{L}$ replaced by $\mathcal{L}_k$.*

3. Numerical Results

We present in this section several illustrative numerical experiments by using our RSAV approach, and compare it with popular gradient based approaches.

In order to present a fair comparison to gradient descent (GD), we consider a composite gradient method. By abuse of notation, we shall refer to *GD with $\mathcal{L}$* as the following method for the splitting (1.4):

$$\theta_{k+1} + \delta t \mathcal{L} \theta_{k+1} = \theta_k + \delta t(\mathcal{L}\theta_k - \nabla f(\theta_k)),$$

which is equivalent to

$$\theta_{k+1} = \theta_k - \delta t (I + \delta t \mathcal{L})^{-1} \nabla f(\theta_k). \tag{3.1}$$

The scheme (3.1) can also be regarded as the forward-backward splitting scheme

$$\theta_{k+1} = \theta_k - \delta t \nabla F(\theta_k) - \delta t \nabla G(\theta_{k+1})$$

for minimizing a composite function $F(\theta) + G(\theta)$ with $F(\theta) = f(\theta) - \frac{1}{2}\theta^T \mathcal{L}\theta$ and $G(\theta) = \frac{1}{2}\theta^T \mathcal{L}\theta$.

Note that the scheme (3.1) reduces to the vanilla gradient descent (1.2) if setting $\mathcal{L} = 0$, and (3.1) reduces to the vanilla gradient descent (1.2) with step size $\frac{\delta t}{1+\delta t}$ if setting $\mathcal{L} = I$. Therefore, we do not consider GD with $\mathcal{L} = I$, and we compare the adaptive RSAV in **Algorithm 2** to the following algorithms:

(1) *GD with* $\mathcal{L} = 0$, which is the vanilla gradient descent (1.2).
(2) *GD with* $\mathcal{L} = -\sigma\Delta$ *with discrete Laplacian* Δ, which is similar to the Laplacian Smoothing Gradient Descent [20].
(3) ADAM [15]
(4) Nesterov accelerated gradient decent (NAG) [16].

Unless specified otherwise, we use ADAM and NAG with the default parameter settings as in [22]: NAG with $\gamma = 0.9$, and ADAM with $\beta_1 = 0.9, \beta_2 = 0.999, \varepsilon = 10^{-8}$.

3.1. **A quadratic cost function.** We start with a quadratic loss function from [20]:

$$f(\theta_1, \theta_2, \ldots, \theta_{100}) = \sum_{i=1}^{50} \theta_{2i-1}^2 + \sum_{i=1}^{50} \frac{1}{100}\theta_{2i}^2. \tag{3.2}$$

For this simple function, we take either $\mathcal{L} = 0$ or $\mathcal{L} = D$ where the diagonal matrix D is chosen to be the Hessian $\nabla^2 f(\theta)$.

To demonstrate the unconditional stability of SAV-based approaches, in Table 1 and Figure 1 we show the results of different initial step sizes δt for the vanilla gradient descent, i.e., GD with $\mathcal{L} = 0$, as well as GD with $\mathcal{L} = \mathcal{D}$. We observe that the vanilla gradient descent blows up for the constant step size $\delta t = 1$, while the adaptive RSAV works quite well.

(initial) step-size δt	0.01	0.1	1
GD ($\mathcal{L} = 0$)	0.3351	0.009121	50
adaptive RSAV ($\mathcal{L} = 0$)	6.34e-12	5.749e-12	2.264e-18
GD ($\mathcal{L} = D$)	0.3352	0.009194	3.152e-18
adaptive RSAV ($\mathcal{L} = D$)	0	0	0

TABLE 1. Loss of quadratic function after 1000 iterations.

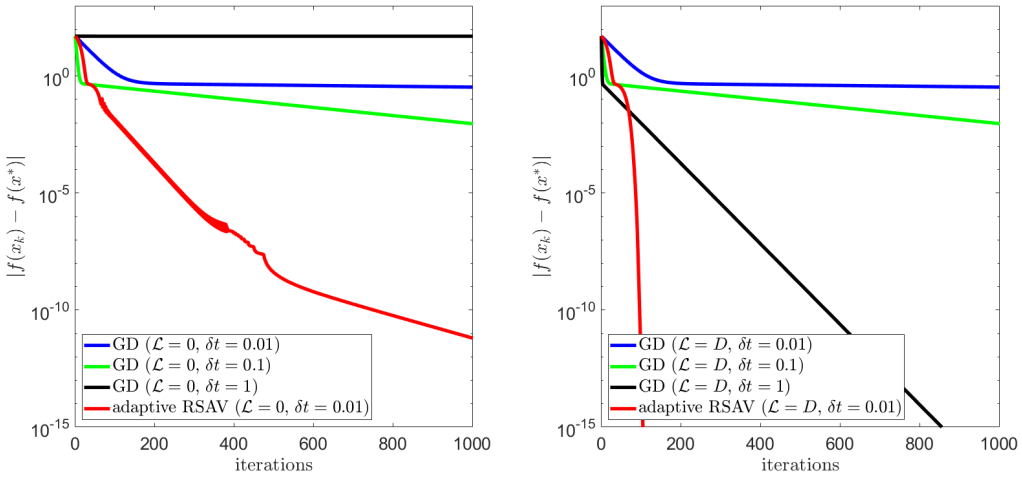


FIGURE 1. Loss curves for GD and adaptive RSAV with different splits and (initial) step-sizes δt.

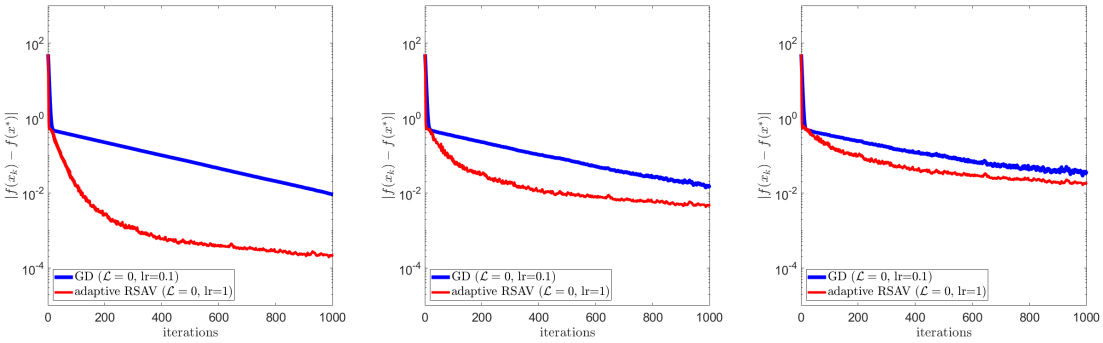
Next we consider the gradient perturbed by a Gaussian noise

$$\nabla_\epsilon f(x) := \nabla f(x) + \epsilon \mathcal{N}(0, I), \tag{3.3}$$

where ϵ controls the noise level, $\mathcal{N}(0, I)$ is the Gaussian noise vector with zero mean and unit variance in each coordinate. The comparison is given in Table 2 and Figure 2 where $\mathcal{L} = 0$ is used for both GD and adaptive RSAV. We observe that the adaptive RSAV converges much faster than GD. The fast convergence of adaptive RSAV is partly due to the indicator $I_k(r, \theta)$ which can give a proper step size. Especially in the noisy case, the true step size is given at a proper level to reach a better convergence than GD and reduce the oscillation in the loss curves.

(initial) step-size δt	0.01	0.1	1
GD ($\epsilon = 0.01$)	0.335	0.009223	58.58
adaptive RSAV ($\epsilon = 0.01$)	0.0002283	0.0002298	0.0002251
GD ($\epsilon = 0.05$)	0.3348	0.01584	diverge
adaptive RSAV ($\epsilon = 0.05$)	0.004934	0.005023	0.004889
GD ($\epsilon = 0.1$)	0.3354	0.03808	diverge
adaptive RSAV ($\epsilon = 0.1$)	0.01746	0.01924	0.0188

TABLE 2. Loss of quadratic function after 1000 iterations with different noise levels ϵ and (initial) step-sizes δt. $\mathcal{L} = 0$ is used for both GD and adaptive RSAV.



(A) $\epsilon = 0.01$ (B) $\epsilon = 0.05$ (C) $\epsilon = 0.1$

FIGURE 2. Loss curves for GD and adaptive RSAV with different noise levels. The learning rate (lr) refers to the step size δt for GD and the initial step size δt in the adaptive RSAV.

3.2. **Rastrigin function.** Consider

$$f(x) = f(\theta_1, \theta_2, \ldots, \theta_n) = \sum_{i=1}^{n} \theta_i^2 + 10n - 10 \sum_{i=1}^{n} \cos(2\pi\theta_i), \tag{3.4}$$

which has many local minima. The function can be defined on any input domain but it is usually evaluated on $x \in [-5.12, 5.12]$ for $i = 1, 2, \ldots, n$. The function has a global minimum at $f(x^*) = 0$ located at $x^* = (0, 0, \ldots, 0)$. In this example, we compare the adaptive RSAV with popular optimization methods ADAM and NAG with their default parameter settings as in [22]: NAG with $\gamma = 0.9$, and ADAM with $\beta_1 = 0.9, \beta_2 = 0.999, \varepsilon = 10^{-8}$. We shall keep using these default settings in all following experiments.

initial stepsize δt	0.001	0.01	0.1	1
GD ($\mathcal{L} = 0$)	12.93	37.86	109.2	diverge
adaptive RSAV ($\mathcal{L} = 0$)	13.05	13.03	12.95	2.608e-9
NAG	12.93	152	505.1	diverge
ADAM	32.28	12.94	12.93	8.958

TABLE 3. Loss of Rastrigin function after 100 iterations in 2D.

We plot in the left of Fig. 3 the convergence curves of different methods, and observe that the RSAV converges much faster than ADAM with the same initial step-size. We also plot in the right of Fig. 3 the paths towards the minimum by different methods. We observe that RSAV enjoys a fast convergence if using a large initial step size δt.

3.3. **Rosenbrock function.** This is a benchmark problem for optimization of **non-convex** functions. We first consider the 2D case with

$$f(x, y) = (a - x)^2 + b(y - x^2)^2, \tag{3.5}$$

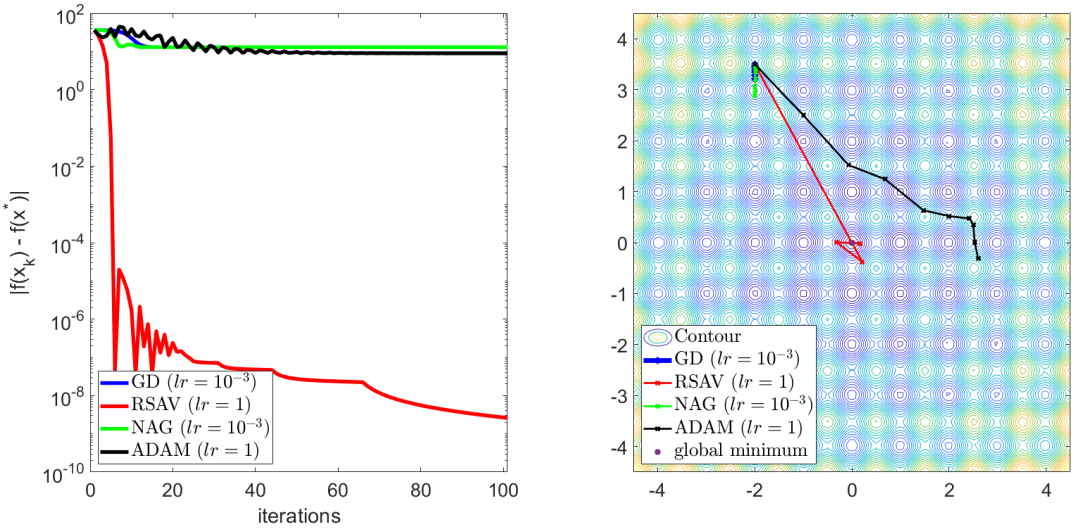


FIGURE 3. Rastrigin function: Left: Convergence curves with 100 iterations; Right: Paths with first 10 iterations. The learning rate (lr) refers to the initial step size δt in the adaptive RSAV.

it has a global minimum at $(x, y) = (a, a^2)$, which is inside a long narrow, parabolic shaped flag valley. To find the valley is trivial, but to converge to the global minimum is usually difficult.

We set $a = 1$ and $b = 100$ in the following experiments and other parameters the same as in [20], and start with the initial point with coordinate $(-3, -4)$. In Table 4, we observe that a large step size can lead to blow-up for other methods except for RSAV. Thus in Figure 4, we only show the results with the largest suitable step sizes for other algorithms. For adaptive RSAV, we just use the same initial step size as ADAM. This example reveals the benefits of modified energy decreasing property of the RSAV. Although ADAM can get close to the global minimum at first, it still goes to the wrong direction caused by the momentum and eventually goes back after wasting many iterations. Only RSAV converges to the global minimum directly with the guide of decreasing (modified) energy.

step-size δt	10^{-4}	10^{-2}	1
GD ($\mathcal{L} = 0$)	0.7142	diverge	diverge
adaptive RSAV ($\mathcal{L} = 0$)	0.01086	0.01122	0.0107
NAG	5.326	diverge	diverge
ADAM	15198	12.5	1.2

TABLE 4. Loss of Rosenbrock function after 1000 iterations in 2D

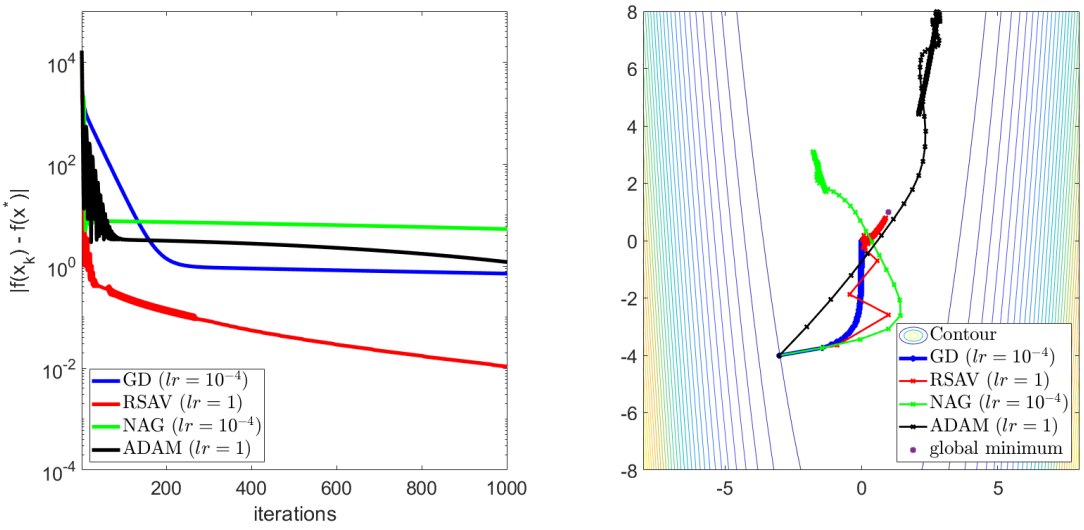
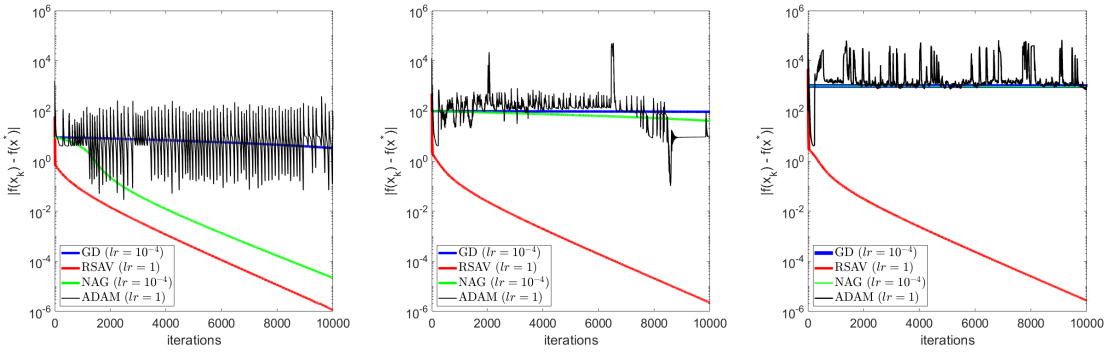


FIGURE 4. 2D Rosenbrock problem: Left: Convergence curves; Right: Paths with 1000 iterations in the (θ_1, θ_2) domain.

Next, we consider the high dimensional cases with

$$f(x) = \sum_{i=1}^{n} (a - \theta_i)^2 + b \sum_{i=1}^{n-1} (\theta_{i+1} - \theta_i^2)^2, \tag{3.6}$$

with the global minimum $f(x^*) = 0$ at $x^* = (a, a^2, a, a^2, \ldots, a, a^2)$. We take $a = 1$ and $b = 100$, and the initial point $(0, \ldots, 0)$. The results with the dimension equal to 10, 100 and 1000 are shown in Fig. 5. We observe similar convergence behavior for all cases as in the two dimensional case.



(A) $n = 10$ (B) $n = 100$ (C) $n = 1000$

FIGURE 5. Loss of Rosenbrock function with dimension n.

Remark 3. *If we compare the adaptive RSAV with Nesterov accelerated gradient decent, the results are still quite good especially when the dimension is* 1000*. Thanks to the adaptive scheme, the performance of RSAV in this problem independent of the dimension.*

3.4. **Phase Retrieval.** The phase retrieval problem [5] can be formulated as

$$\min_{z\in\mathbb{C}^N} f(z) := \frac{1}{2}\|\mathcal{A}(z^*z) - b\|^2$$

where $z^* \in \mathbb{C}^{1\times N}$ is the conjugate transpose of z, $b \in \mathbb{R}^M$ and $\mathcal{A} : \mathbb{C}^{N\times N} \longrightarrow \mathbb{R}^M$ is a linear operator. For the real-valued function $f(z)$ with complex variable $z := a + \mathbb{i}b$, where $\mathbb{i}$ is the imaginary unit and $a, b \in \mathbb{R}$ are real and imaginary parts of z, we can define the Fréchet derivative induced by the natural choice of real inner product for $\mathbb{C}^N$ as the following [31]:

$$\nabla f(z) := \frac{\partial f(z)}{a} + \mathbb{i}\frac{\partial f(z)}{b} = 2\mathcal{A}^*(\mathcal{A}(z^*z - b))z,$$

where $\mathcal{A}^*$ is the adjoint operator of $\mathcal{A}$. Then the vanilla gradient descent algorithm for $\min_{z\in\mathbb{C}^N} f(z)$ can be defined as in (1.2) using $\nabla f(z)$ above. The gradient descent method with a suitable step sizing rule is also also referred to as the Wirtinger flow [6].

In particular, $f(z)$ is a non-convex quartic polynomial function of z. For the theorectical convergence of minimizing such a non-convex function, with a spectral initialization, i.e., z_0 being the leading eigenvector of $\mathcal{A}^*(b)$, the convergence of Wirtinger flow with high probability can be proven for a very special class of phase retrieval problems [6, 4]. For solving phase retrieval with random initial guess, the convergence for minimizing a smoothed amplitude flow based model was proven in [3]. In terms of practical performance with only random initialization, state-of-the-art algorithms such as the Riemannian LBFGS method could be much more efficient than gradient descent algorithms [6].

We emphasize that we only use such phase retrieval problems as a testing example to validate the performance of the RSAV method. So we test the algorithms with a random initialization.

We compare vanilla gradient descent (GD), adaptive RSAV with $\mathcal{L} = 0$, and steepest descent (SD) [8, 2]. The steepest descent method is to use the optimial step size in (1.2), and it is possible to compute such an optimal step size for a polynomial cost function. We test the performance of RSAV algorithm on the following phase retrieval problem. Let $\mathcal{M}_i \in \mathbb{C}^N$ be i.i.d Gaussian and $\circ$ denote the entrywise product. Let $\mathcal{F}$ denote the Fourier transform. The linear operator $\mathcal{A}$ is defined by assigning $\|\mathcal{F}(\mathcal{M}_i \circ z)\|^2$ to b, e.g., $\frac{M}{N} = m$. We consider the test case for the true solution z_* being an image of size $n \times n$ with $n = 256$. So the size of unknown is $N = n^2 = 256^2$. We consider two test cases:

(1) The true minimizer z_* is a real image of camera man with size 256×256 as shown in Figure 6, $m = 6$ Gaussian random masks and a random initial guess.
(2) The true minimizer z_* is a complex image of golden ball with size 256×256, see [12] for details, $m = 10$ Gaussian random masks and a random initial condition.

See Figure 7 for the comparision of the performance of gradient based algorithms. For the vanilla gradient descent (GD), we use the nearly largest stable constant step size $\delta t = 0.5$. In Figure 8, we list a comparsion of the step size δ_k in the adaptive RSAV method with the optimital step size in the steepest descent method. We can see the performance of adaptive RSAV has the closest performance to the steepest descent. For the real image case, SD converged after 10000 iterations and RSAV converged after 20000 iterations. However, to compute the optimal step size in the steepest descent, at least two more evaluations of $\mathcal{A}$ are needed, thus quite expensive. More importantly, for a general cost function, it is difficult to find the optimal step size. See Section 4.4 for an analysis of the step size in the explicit SAV gradient descent with restarting r_k every iteration for quadratic functions.

(A) GD (B) adaptive RSAV (C) SD

FIGURE 6. Results after 20000 iterations for a phase retrieval problem with z_* being a 256×256 real image of camera man, $m = 6$ Gaussian random masks and a random initial guess. The vanilla gradient descent (GD) uses nearly largest stable constant step size $\delta t = 0.5$

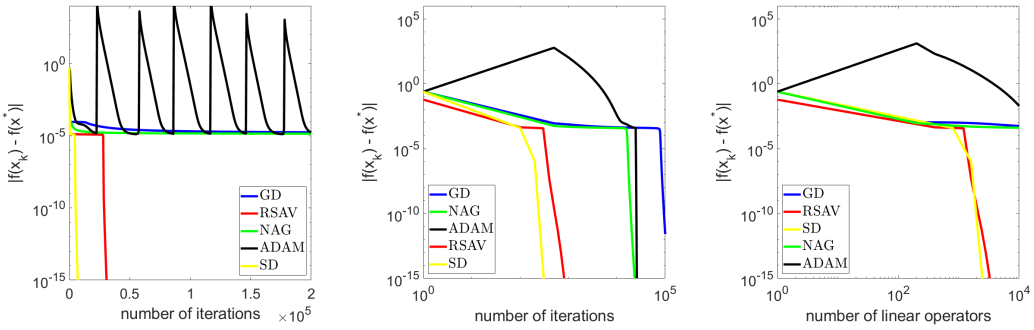


FIGURE 7. (Left) Loss of different optimization algorithms V.S. number of iteration for phase retrieval: the real image of camera man using 6 Gaussian masks; (Middle) Loss of different optimization algorithms V.S. number of iteration for phase retrieval: the complex image of golden balls using 10 Gaussian masks. (Right) Loss of different optimization algorithms V.S. number of iteration for phase retrieval: the complex image of golden balls using 10 Gaussian masks. The vanilla gradient descent (GD) uses nearly largest stable step size $\delta t = 0.5$.

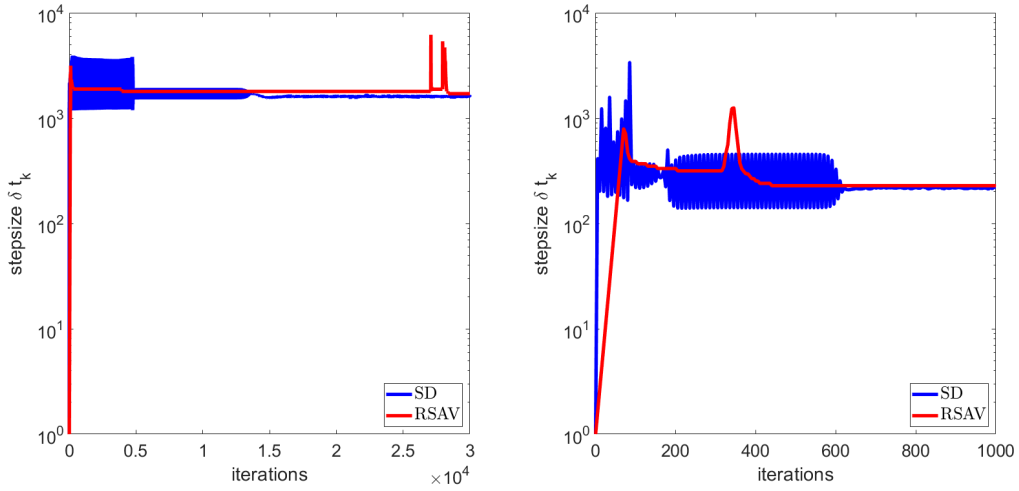


FIGURE 8. The value of δ_k in each iteration for phase retrieval: Left: the real image of camera man using 6 Gaussian masks; Right: the complex image of golden balls using 10 Gaussian masks.

3.5. **Recommendation System.** Consider applying the optimization scheme to train a recommendation system based on matrix factorization model. Given a rate matrix $R \in \mathbb{R}^{m\times n}$ where m is the number of users and n is the number of items, the model learns the user embedding matrix $X \in \mathbb{R}^{m\times d}$ and item embedding matrix $Y \in \mathbb{R}^{n\times d}$ such that the product XY^T is a good approximation for R. Here, d is the embedding dimension and usually much

smaller than m and n. Denote the user and item matrix by $X = [X_1, \ldots, X_u, \ldots, X_m]^T$ and $Y = [Y_1, \ldots, Y_i, \ldots, Y_n]^T$, we have the loss function as

$$f(X, Y) = \frac{1}{N_\kappa} \sum_{(u,i)\in\kappa} (R_{u,i} - X_u Y_i^T)^2 + \lambda_u \sum_u \|X_u\|_2^2 + \lambda_i \sum_i \|Y_i\|_2^2, \tag{3.7}$$

where κ the training set that the (u, i) pairs for which $R_{u,i}$ is known, N_κ is the number of training data, λ_u and λ_i are the penalty parameters for embedding matrix. We train the model with the MovieLens 100K dataset [11] which contains $100,000$ ratings $(1 - 5)$ from 943 users on 1682 movies. There is 80% data split as the training data and the rest date is used for the testing data, e.g., the training data set κ has size $80,000$. All algorithms use the mini-batch gradient with batch size 80. For l_2 regularization, we set $\lambda_u = \lambda_i = 10^{-4}$. For the linear operator $\mathcal{L} = \lambda I - \sigma\Delta$ in GD and RSAV, we let $\lambda = 10^{-4}$ and $\sigma = 0.1$. In Figure 9, for the training step, we run $10,000$ iterations for the mini-batch gradient based methods with batch size 80, which is equal to 10 epochs. The result on the test data is shown in Table 5. We can see that RSAV performs well in the training step, though its testing result is not the best, which suggests issues of overfitting during the training step. This is more or less a modelling issue, rather than the optimizaiton issue.

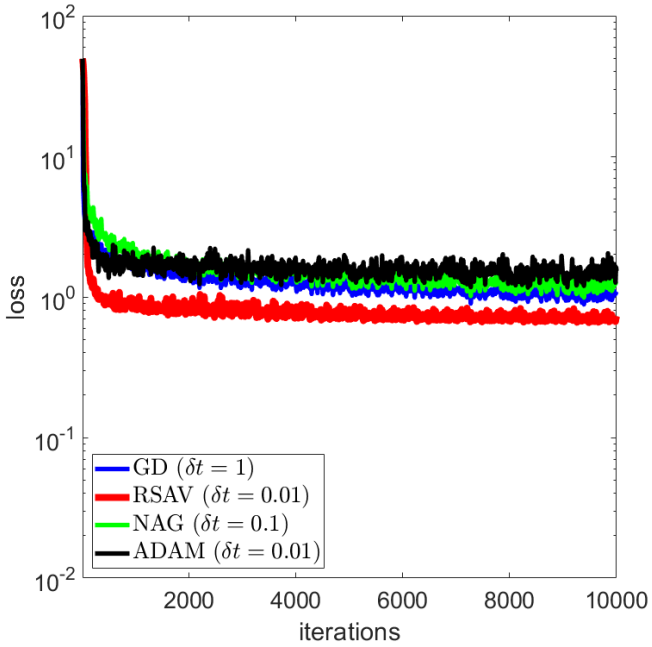


FIGURE 9. The training loss curve of different optimization algorithms for Recommendation System. Here GD refers to GD ($\mathcal{L} = \lambda I - \sigma\Delta$) and RSAV refers to the adaptive RSAV ($\mathcal{L} = \lambda I - \sigma\Delta$).

step-size δt	0.01	0.1	1	10
GD ($\mathcal{L} = \lambda I - \sigma\Delta$)	4.6504	2.1465	1.6438	diverge
NAG	2.1451	1.6439	diverge	diverge
ADAM	1.8820	4.7194	diverge	diverge
adaptive RSAV ($\mathcal{L} = \lambda I - \sigma\Delta$)	1.9090	1.9102	1.9156	1.9156

TABLE 5. The loss function on the test data after $10,000$ training iterations (10 epochs) with different step-sizes. Here *diverge* means that the training step already diverges.

4. Convergence study for some SAV based algorithms

In this section, we consider a more general version of the SAV scheme based on the following expanded system

$$\begin{cases} \theta_t = -\left(\frac{r}{[f(\theta)+C]^q}\nabla f(\theta) + \mathcal{L}\theta - \mathcal{L}\theta\right) \\ r_t = q[f(\theta)+C]^{q-1}(\nabla f(\theta), \theta_t), \end{cases} \tag{4.1}$$

where $r(t) = [f(\theta)+C]^q$ and $q \in (0,1)$. Note that (2.2) is a special case of the above formulation with $q = \frac{1}{2}$. Similar to (2.3), we can construct a SAV scheme for (4.1) as follows:

$$\begin{cases} \frac{\theta_{k+1}-\theta_k}{\delta t} = -\left(\frac{r_{k+1}}{[f(\theta_k)+C]^q}\nabla f(\theta_k) + \mathcal{L}(\theta_{k+1}-\theta_k)\right) \\ \frac{r_{k+1}-r_k}{\delta t} = q[f(\theta_k)+C]^{q-1}(\nabla f(\theta_k), \frac{\theta_{k+1}-\theta_k}{\delta t}). \end{cases} \tag{4.2}$$

4.1. **Interpretation of the SAV method as a line search method.** Let $A = (I + \delta t\mathcal{L})$. The system (4.2) can be rewrite as

$$\begin{bmatrix} A & \delta t\frac{\nabla f(\theta_k)}{[f(\theta_k)+C]^q} \\ -q[f(\theta_k)+C]^{q-1}\nabla f(\theta_k) & 1 \end{bmatrix} \begin{bmatrix} \theta_{k+1} \\ r_{k+1} \end{bmatrix} = \begin{bmatrix} A\theta_k \\ r_k - q[f(\theta_k)+C]^{q-1}(\nabla f(\theta_k), \theta_k) \end{bmatrix}$$

After a simple Gaussian elimination, we obtain an explicit update formula for (4.2):

$$\begin{cases} r_{k+1} &= \frac{1}{1+\delta tq\frac{(\nabla f(\theta_k), A^{-1}\nabla f(\theta_k))}{f(\theta_k)+C}} r_k \\ \theta_{k+1} &= \theta_k - \frac{r_{k+1}}{[f(\theta_k)+C]^q}\delta t A^{-1}\nabla f(\theta_k) \end{cases}.$$

Notice that the scheme above can be regarded as a line search method:

$$\begin{aligned} \theta_{k+1} &= \theta_k + \alpha_k P_k \\ P_k &= -A^{-1}\nabla f(\theta_k) \\ \alpha_k &= \frac{\delta t}{1+\delta tq\frac{(\nabla f(\theta_k), A^{-1}\nabla f(\theta_k))}{f(\theta_k)+C}} \frac{r_k}{[f(\theta_k)+C]^q} > 0, \end{aligned}$$

with a search direction P_k and step size α_k.

The step size α_k is guranteed to be positive. On the other hand, it is difficult to establish any *a priori* control of α_k, and in practice α_k could become very small if r_k becomes very

small. To avoid small r_k, we consider a special version of SAV method by redefining $r_k = [f(\theta_k) + C]^q$, then we have

$$\alpha_k = \frac{\delta t}{1 + \delta t q \frac{(\nabla f(\theta_k), A^{-1}\nabla f(\theta_k))}{f(\theta_k)+C}}.$$

In this case, we can view q as a parameter, and the SAV method with $r_k = [f(\theta_k) + C]^q$ at every iteration becomes the following line search method:

$$\theta_{k+1} = \theta_k + \alpha_k P_k \tag{4.4a}$$

$$P_k = -A^{-1}\nabla f(\theta_k) \tag{4.4b}$$

$$\alpha_k = \frac{\delta t}{1 + \delta t q \frac{(\nabla f(\theta_k), A^{-1}\nabla f(\theta_k))}{f(\theta_k)+C}}, \tag{4.4c}$$

which is equivalent to

$$\begin{cases} r_k = [f(\theta_k) + C]^q \\ \frac{\theta_{k+1}-\theta_k}{\delta t} = -\left(\frac{\tilde{r}_{k+1}}{[f(\theta_k)+C]^q}\nabla f(\theta_k) + \mathcal{L}(\theta_{k+1} - \theta_k)\right) \\ \frac{\tilde{r}_{k+1}-r_k}{\delta t} = q[f(\theta_k) + C]^{q-1}(\nabla f(\theta_k), \frac{\theta_{k+1}-\theta_k}{\delta t}). \end{cases} \tag{4.5}$$

In particular, for any $\mathcal{L} \geq 0$, A^{-1} is always positive definite, thus the search direction $P_k = -A^{-1}\nabla f(\theta_k)$ is always a descent direction, i.e., $-\nabla f^T(\theta_k)P_k = \nabla f(\theta_k)^T A^{-1}\nabla f(\theta_k) > 0$. The Wolfe condition [27] for the line search method (4.4) is: there exists $0 < c_1 < c_2 < 1$ such that

$$f(\theta_k + \alpha_k P_k) \leq f(\theta_k) + c_1\alpha_k \nabla f(\theta_k)^T P_k \tag{4.6a}$$

$$\nabla f(\theta_k + \alpha_k P_k)^T P_k \geq c_2 \nabla f(\theta_k)^T P_k. \tag{4.6b}$$

We recall first the following result [19]:

Theorem 3. *Assume $f(\theta) \in C^1$ and $f(\theta)$ is bounded from below. For any descent direction P_k, there exist intervals of step lengths satisfying the Wolfe condition.*

Notice that $\alpha(\delta t, q) = \frac{\delta t}{1+\delta t q \frac{(\nabla f(\theta_k), A^{-1}\nabla f(\theta_k))}{f(\theta_k)+C}}$ is an increasing function of δt and an decreasing function of q, thus there exists δt and q such that $\alpha_k = \frac{\delta t}{1+\delta t q \frac{(\nabla f(\theta_k), A^{-1}\nabla f(\theta_k))}{f(\theta_k)+C}}$ satisfies the Wolfe conditions (4.6).

We recall below another result [19]:

Theorem 4. *Assume $f(\theta) \in C^1$, $f(\theta)$ is bounded from below and $\nabla f(\theta)$ is Lipschitz continuous. Let $\cos\phi_k = \frac{-\nabla f(\theta_k)^T P_k}{\|\nabla f(\theta_k)\|\|P_k\|}$. If P_k is a descent direction and α_k satisfies the Wolfe Conditions, then the iteration $\theta_{k+1} = \theta_k + \alpha_k P_k$ satisfies*

$$\sum_{k\geq 0} \cos^2\phi_k \|\nabla f(\theta_k)\|^2 < \infty. \tag{4.7}$$

Let $\lambda_{\min}(A)$ and $\lambda_{\max}(A)$ be the smallest and largest eigenvalues of the real symmetric positive definite matrix A, then by the Courant-Fischer-Weyl min-max principle [7] and the spectral norm $\|A\| = \lambda_{\max}(A)$, we have

$$\frac{P_k^T A P_k}{\|P_k\|^2} \geq \lambda_{\min}(A), \quad \|AP_k\| \leq \|A\|\|P_k\| = \lambda_{\max}(A)\|P_k\|,$$

thus

$$\cos\phi_k = \frac{P_k^T A P_k}{\|AP_k\|\|P_k\|} = \frac{P_k^T A P_k}{\|P_k\|^2}\frac{\|P_k\|}{\|AP_k\|} \geq \frac{\lambda_{\min}(A)}{\lambda_{\max}(A)}.$$

Therefore, the uniform lower bound on $\cos\phi_k$ and (4.7) implies that $\|\nabla f(\theta_k)\| \to 0$.

Thus the convergence of the SAV method (4.4) is ensured if using a line search to find $\delta t, q$ such that α_k satisfies the Wolfe condition (4.6). We observe that the above algorithm involves computing $A^{-1}\nabla f(\theta_k)$, and evaluation of $f(\theta_k)$ and $f(\theta_{k+1})$.

Remark 4. *In practice, one can use backtracking line search on α_k to ensure that the Wolfe conditions are satisfied. This in general it does not seem advantageous over a simple backtracking line search on α. However, for the SAV gradient descent method* (2.4)*, i.e., $A = I$, our numerical observation is that the SAV scheme is often more efficient than the backtracking line search on α. With $A = I$, the scheme* (4.4) *reduces to the following SAV gradient descent method with two parameters $\delta t > 0$ and $q_k > 0$:*

$$\theta_{k+1} = \theta_k - \alpha_k \nabla f(\theta_k) \tag{4.8a}$$

$$\alpha_k = \frac{\delta t}{1 + \delta t q_k \frac{\|\nabla f(\theta_k)\|^2}{f(\theta_k)+C}} \tag{4.8b}$$

4.2. **Standard convergence results.** We recall that if α_k in the line search method (4.4) satisfies the Goldstein-Armijo rule [1, 9]: there exists $0 < c_1 < c_2 < 1$ such that

$$f(\theta_k) - c_2\alpha_k\|\nabla f(\theta_k)\|^2 \leq f(\theta_k - \alpha_k\nabla f(\theta_k)) \leq f(\theta_k) - c_1\alpha_k\|\nabla f(\theta_k)\|^2, \tag{4.9}$$

then it is shown (cf. Theorem 2.1.14 in [17]) that $\|\nabla f(\theta_k)\| \to 0$.

Theorem 2.1.14 in [17] can be easily adapted to prove the following result for the line search method (4.4):

Theorem 5. *Assume that $f(\theta)$ is convex and $\nabla f(\theta)$ is Lipshitz continuous with the Lipshitz constant L, i.e., $\|\nabla f(y) - \nabla f(x)\| \leq L\|x - y\|$. If $\nabla f(\theta_*) = 0$ and $\alpha_k \in (0, \frac{2}{L})$, then*

$$\|\theta_{k+1} - \theta_*\|^2 \leq \|\theta_k - \theta_*\|^2 - \alpha_k(\frac{2}{L} - \alpha_k)\|\nabla f(\theta_k)\|^2$$

and

$$f(\theta_k) - f(\theta_*) \leq \frac{1}{[f(\theta_0) - f(\theta_*)]^{-1} + \|\theta_0 - \theta_*\|^{-2}\sum_k \alpha_k(1 - \frac{L}{2}\alpha_k)}.$$

So for convergence, we need $\sum_{k=0}^{\infty} \alpha_k(1 - \frac{L}{2}\alpha_k) = +\infty$, which can be ensured if $\alpha_k \in [a, b] \in (0, \frac{2}{L})$ for constant bounds $a > 0$ and $b < \frac{2}{L}$. Also, $\alpha_k < \frac{2}{L}$ will ensure $f(\theta_{k+1}) < f(\theta_k)$.

4.3. **Decreasing step sizes for the SAV gradient descent method.** We derive from (4.8) that

$$\alpha_k = \frac{\delta t}{1+\delta t q_k \frac{\|\nabla f(\theta_k)\|^2}{f(\theta_k)}} = \frac{1}{1/\delta t + q_k \frac{\|\nabla f(\theta_k)\|^2}{f(\theta_k)}} \le \min\left\{\delta t, \frac{f(\theta_k)}{q_k \|\nabla f(\theta_k)\|^2}\right\}.$$

For fixed δt, the above is often sufficient to ensure $f(\theta_{k+1}) < f(\theta_k)$ when θ_k is far away from the minimizer. It can be understood as follows.

Theorem 6. *Assume that $f(\theta)$ is strongly convex, i.e., $(\nabla f(y) - \nabla f(x), y - x) \ge m\|x-y\|^2$ with $m > 0$, and $\nabla f(\theta)$ is Lipshitz continuous with the Lipshitz constant L. Let θ_* be the minimizer and assume $f(\theta_*) = 0$. Then the following are sufficient conditions to ensure $\alpha_k < \frac{2}{L}$ for the SAV gradient descent method with two parameters* (4.8)*:*

1. *For any $\delta t > 0$, $q_k > \frac{L^2}{4m^2}$.*
2. *Let $\delta t \equiv a\frac{2}{L}$ where $a > 0$, $q_k > \frac{a-1}{a}\frac{L^2}{4m^2}$.*

Remark 5. *The first sufficient condition implies that $\alpha_k = \frac{\delta t}{1+\delta t q_k \frac{\|\nabla f(\theta_k)\|^2}{f(\theta_k)}}$ will be a decreasing step size for any δt if $q_k \equiv q > \frac{L^2}{4m^2}$. Of course, finding $\frac{1}{q} < \frac{4m^2}{L^2}$ in general is not easier than finding $\delta t < \frac{2}{L}$. But if $2m^2 > L$, then $\frac{4m^2}{L^2} > \frac{2}{L}$ implies $\frac{1}{q} < \frac{4m^2}{L^2}$ is easier to achieve.*

Remark 6. *As an example of the second sufficient condition, if we pick $q_k \equiv \frac{1}{2}$, and $a = 2$, then $\frac{1}{2} > \frac{1}{2}\frac{L^2}{4m^2} \Leftrightarrow L < 2m$ is sufficient to ensure the SAV gradient descent method with $q_k \equiv \frac{1}{2}$ is decreasing with $\delta t = \frac{4}{L}$, instead of $\delta t < \frac{2}{L}$ in* (1.2).

Proof. First, by the strong convexity and Lipschitz continuity, we have

$$f(x) \ge f(y) + (\nabla f(y), x-y) + \frac{m}{2}\|x-y\|^2,$$

$$f(x) \le f(y) + (\nabla f(y), x-y) + \frac{L}{2}\|x-y\|^2.$$

Since θ_* is the minimizer, $\nabla f(\theta_*) = 0$. For any θ, we have

$$(\nabla f(\theta) - \nabla f(\theta_*), \theta - \theta_*) \ge m\|\theta-\theta_*\|^2 \Rightarrow (\nabla f(\theta), \theta - \theta_*) \ge m\|\theta-\theta_*\|^2$$

$$\Rightarrow m\frac{\|\theta-\theta_*\|}{\|\nabla f(\theta)\|} = \frac{(\nabla f(\theta), \theta-\theta_*)}{\|\theta-\theta_*\|\|\nabla f(\theta)\|} \le 1 \Rightarrow \|\nabla f(\theta)\| \ge m\|\theta-\theta_*\|.$$

Hence,

$$m\|\theta-\theta_*\| \le \|\nabla f(\theta)\| \le L\|\theta-\theta_*\|,$$

and

$$\frac{m}{2}\|\theta-\theta_*\|^2 \le f(\theta) - f(\theta_*) \le \frac{L}{2}\|\theta-\theta_*\|^2.$$

With strong convexity $m > 0$, we have

$$\frac{2m^2}{L} \le \frac{\|\nabla f(\theta)\|^2}{f(\theta) - f(\theta_*)} \le \frac{2L^2}{m}$$

thus

$$\frac{2m^2}{L} \leq \frac{\|\nabla f(\theta)\|^2}{f(\theta)} \leq \frac{2L^2}{m}.$$

Finally,

$$\frac{f(\theta_k)}{q_k\|\nabla f(\theta_k)\|^2} < \frac{2}{L} \Leftrightarrow \frac{1}{q_k} < \frac{\|\nabla f(\theta_k)\|^2}{f(\theta_k)}\frac{2}{L} \Leftarrow \frac{1}{q_k} < \frac{4m^2}{L^2}.$$

□

Remark 7. *In general, if $f(\theta)$ is only convex but not strong convex, i.e., $m = 0$, and $f(\theta) + C > 0$, then we only have*

$$\frac{\|\nabla f(\theta)\|^2}{f(\theta)+C} \leq \frac{L^2\|\theta-\theta_*\|^2}{f(\theta_*)+C}.$$

This gives a lower bound control on step size:

$$\alpha_k = \frac{\delta t}{1+\delta t q_k \frac{\|\nabla f(\theta_k)\|^2}{f(\theta_k)+C}} \geq \frac{\delta t}{1+q_k\delta t\frac{L^2\|\theta_k-\theta_*\|^2}{f(\theta_*)+C}}.$$

In this case, we can set $q_k \equiv q$ and do back tracking on δt for α_k to satisfy the convergence condition or Goldstein-Armijo rule.

4.4. **The step size for quadratic functions.** To see why the step size $\alpha_k = \frac{\delta t_k}{1+\delta t_k q_k \frac{\|\nabla f(\theta_k)\|^2}{f(\theta_k)+C}}$ could be a good step size to use, at least for a quadratic cost function, consider a cost function $f(\theta) = \frac{1}{2}\|A\theta - b\|^2$ with a square and positive definite matrix $A > 0$. The steepest descent algorithm can be written as

$$\theta_{k+1} = \theta_k - \beta_k\nabla f(\theta_k), \quad \beta_k = \frac{\|A^T(A\theta_k - b)\|^2}{[A^T(A\theta_k-b)]^TA^TA[A^T(A\theta_k-b)]}.$$

The method (4.8) with $C = 0$ is

$$\theta_{k+1} = \theta_k - \alpha_k\nabla f(\theta_k), \quad \alpha_k = \frac{1}{\frac{1}{\delta t_k} + q_k\frac{\|A^T(A\theta_k-b)\|^2}{\frac{1}{2}(A\theta_k-b)^T(A\theta_k-b)}}.$$

Then for a very large δt_k and $q_k \equiv \frac{1}{2}$, we have

$$\alpha_k \approx \frac{(A\theta_k-b)^T(A\theta_k-b)}{(A\theta_k-b)^TAA^T(A\theta_k-b)}, \quad \beta_k = \frac{(A\theta_k-b)^TAA^T(A\theta_k-b)}{(A\theta_k-b)^TAA^TAA^T(A\theta_k-b)}.$$

Let v_i be orthornormal eigenvectors of A with eigenvalues of λ_i. Since v_i form a basis for $\mathbb{R}^N$, let $A\theta_k - b = r = \sum_i r_iv_i$. Let $z = A^T(A\theta_k - b)$, then $z = A^T\sum_i r_iv_i = \sum_i r_i\lambda_iv_i$ and $Az = \sum_i r_i\lambda_i^2v_i$. We get

$$\alpha_k \approx \frac{r^Tr}{r^TAA^Tr} = \frac{r^Tr}{z^Tz} = \frac{[\sum_i r_iv_i]^T[\sum_i r_iv_i]}{[\sum_i r_i\lambda_iv_i]^T[\sum_i r_i\lambda_iv_i]} = \frac{\sum_i r_i^2}{\sum_i\lambda_i^2r_i^2}, \quad \beta_k = \frac{z^Tz}{(Az)^T(Az)} = \frac{\sum_i\lambda_i^2r_i^2}{\sum_i\lambda_i^4r_i^2}.$$

We can see that α_k is very similar to the optimal step size β_k but not the same. In practice, a random initial guess θ_0 usually makes α_k a descent step size in the first few or many iterations for $\delta t_k \equiv 1$ and $q_k \equiv q = \frac{1}{2}$.

5. Concluding remarks

We proposed in this paper a new minimization algorithm inspired by the scalar auxiliary variable (SAV) approach for gradient flows. Since the direct application of the SAV approach to minimization problems may converge to wrong solutions, we developed a modified version of the SAV approach coupled with a relaxation step and an adaptive stradegy. The new algorithm enjoys several distinct advantages, including unconditionally energy diminishing with a modified energy, and empirical better performance than many first order methods. In particular, it overcomes the difficulty in selecting proper step sizes associated with the usual gradient based algorithms. The energy diminishing property ensures the convergence, and the relaxation step, built on a connection between the decreasing modified energy and the original energy, helps to accelerate the convergence.

We also presented a converence analysis for some SAV based algorithms which include the new algorithm without the relaxation step as a special case. Numerical results for several illustrative and benchmark problems indicates that the new algorithm is very robust and usually converges significantly faster than those popular gradient decent based methods.

While we only considered a basic version of the SAV based approach which already showed its promise, it is clear that this approach can be combined with other techniques of acceleration and generalization to the gradient decent methods. How to further improve the robustness and accelerate the convergence rate of the SAV based approach will be the subject of a future study.

Acknowledgement

This work is partially supported by AFOSR FA9550-16-1-0102, NSF DMS-2012585 and DMS-2208518.

References

[1] Larry Armijo. Minimization of functions having Lipschitz continuous first partial derivatives. *Pacific Journal of mathematics*, 16(1):1–3, 1966.

[2] Arthur Earl Bryson and Walter F Denham. A steepest-ascent method for solving optimum programming problems. 1962.

[3] Jian-Feng Cai, Meng Huang, Dong Li, and Yang Wang. Solving phase retrieval with random initial guess is nearly as good as by spectral initialization. *Applied and Computational Harmonic Analysis*, 58:60–84, 2022.

[4] T Tony Cai, Xiaodong Li, and Zongming Ma. Optimal rates of convergence for noisy sparse phase retrieval via thresholded Wirtinger flow. *The Annals of Statistics*, 44(5):2221–2251, 2016.

[5] Emmanuel J Candes, Yonina C Eldar, Thomas Strohmer, and Vladislav Voroninski. Phase retrieval via matrix completion. *SIAM review*, 57(2):225–251, 2015.

[6] Emmanuel J Candes, Xiaodong Li, and Mahdi Soltanolkotabi. Phase retrieval via wirtinger flow: Theory and algorithms. *IEEE Transactions on Information Theory*, 61(4):1985–2007, 2015.

[7] Richard Courant and David Hilbert. *Methods of mathematical physics: partial differential equations*. John Wiley & Sons, 2008.

[8] Haskell B Curry. The method of steepest descent for non-linear minimization problems. *Quarterly of Applied Mathematics*, 2(3):258–261, 1944.

[9] Allen A Goldstein. On steepest descent. *Journal of the Society for Industrial and Applied Mathematics, Series A: Control*, 3(1):147–151, 1965.
[10] Zhishuai Guo, Yi Xu, Wotao Yin, Rong Jin, and Tianbao Yang. A novel convergence analysis for algorithms of the Adam family. *arXiv preprint arXiv:2112.03459*, 2021.
[11] F Maxwell Harper and Joseph A Konstan. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)*, 5(4):1–19, 2015.
[12] Wen Huang, Kyle A Gallivan, and Xiangxiong Zhang. Solving phaselift by low-rank riemannian optimization methods for complex semidefinite constraints. *SIAM Journal on Scientific Computing*, 39(5):B840–B859, 2017.
[13] Matt Jacobs, Flavien Léger, Wuchen Li, and Stanley Osher. Solving large-scale optimization problems with a convergence rate independent of grid size. *SIAM Journal on Numerical Analysis*, 57(3):1100–1123, 2019.
[14] Maosheng Jiang, Zengyan Zhang, and Jia Zhao. Improving the accuracy and consistency of the scalar auxiliary variable (SAV) method with relaxation. *Journal of Computational Physics*, 456:110954, 2022.
[15] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
[16] Yurii Nesterov. A method for unconstrained convex minimization problem with the rate of convergence o (1/kˆ 2). In *Doklady an ussr*, volume 269, pages 543–547, 1983.
[17] Yurii Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2013.
[18] Yurii Nesterov. *Lectures on convex optimization*, volume 137. Springer, 2018.
[19] Jorge Nocedal and Stephen Wright. *Numerical optimization*. Springer Science & Business Media, 2006.
[20] Stanley J. Osher, Bao Wang, Penghang Yin, Xiyang Luo, Minh Pham, and Alex Tong Lin. Laplacian smoothing gradient descent. *CoRR*, abs/1806.06317, 2018.
[21] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.
[22] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.
[23] Ernest K Ryu and Wotao Yin. *Large-Scale Convex Optimization: Algorithms & Analyses via Monotone Operators*. Cambridge University Press, 2022.
[24] J Reddi Sashank, Kale Satyen, and Kumar Sanjiv. On the convergence of adam and beyond. In *International Conference on Learning Representations*, volume 5, page 7, 2018.
[25] Jie Shen, Jie Xu, and Jiang Yang. The scalar auxiliary variable (sav) approach for gradient flows. *Journal of Computational Physics*, 353, 10 2017.
[26] Jie Shen, Jie Xu, and Jiang Yang. A new class of efficient and robust energy stable schemes for gradient flows. *SIAM Review*, 61(3):474–506, 2019.
[27] Philip Wolfe. Convergence conditions for ascent methods. *SIAM review*, 11(2):226–235, 1969.
[28] Xiaofeng Yang. Linear, first and second-order, unconditionally energy stable numerical schemes for the phase field model of homopolymer blends. *Journal of Computational Physics*, 327:294–316, 2016.
[29] Yanrong Zhang and Jie Shen. A generalized sav approach with relaxation for dissipative systems. *Journal of Computational Physics*, page 111311, 2022.
[30] Jia Zhao. A revisit of the energy quadratization method with a relaxation technique. *Appl. Math. Lett.*, 120:Paper No. 107331, 11, 2021.
[31] Shixin Zheng, Wen Huang, Bart Vandereycken, and Xiangxiong Zhang. Riemannian optimization using three different metrics for Hermitian PSD fixed-rank constraints: an extended version. *arXiv preprint arXiv:2204.07830*, 2022.
[32] Qingqu Zhuang and Jie Shen. Efficient SAV approach for imaginary time gradient flows with applications to one- and multi-component Bose-Einstein condensates. *J. Comput. Phys.*, 396:72–88, 2019.