GDA-AM: ON THE EFFECTIVENESS OF SOLVING MINIMAX OPTIMIZATION VIA ANDERSON MIXING

Anonymous authors
Paper under double-blind review

ABSTRACT

Many modern machine learning algorithms such as generative adversarial networks (GANs) and adversarial training can be formulated as minimax optimization. Gradient descent ascent (GDA) is the most commonly used algorithm due to its simplicity. However, GDA can converge to non-optimal minimax points. We propose a new minimax optimization framework, GDA-AM, that views the GDA dynamics as a fixed-point iteration and solves it using Anderson Mixing to converge to the local minimax. It addresses the diverging issue of simultaneous GDA and accelerates the convergence of alternating GDA. We show theoretically that the algorithm can achieve global convergence for bilinear problems under mild conditions. We also empirically show that GDA-AM solves a variety of minimax problems and improves GAN training on several datasets.

1 INTRODUCTION

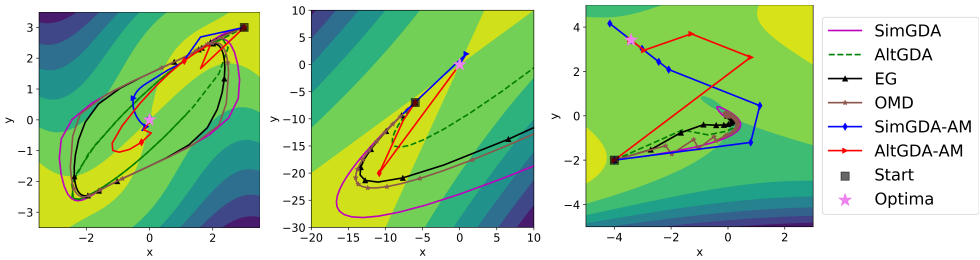
Minimax optimization has received a surge of interest due to its wide range of applications in modern machine learning, such as generative adversarial networks (GAN), adversarial training and multi-agent reinforcement learning (Goodfellow et al., 2014; Madry et al., 2018; Li et al., 2019). Formally, given a bivariate function $f(\boldsymbol{x}, \boldsymbol{y})$, the objective is to find a stable solution where the players cannot improve their objective, i.e., to find the Nash equilibrium of the underlying game (von Neumann & Morgenstern, 1944):

$$\arg\min_{x \in \mathcal{X}} \arg\max_{y \in \mathcal{Y}} f(\boldsymbol{x}, \boldsymbol{y}). \tag{1}$$

It is commonplace to use simple algorithms such as gradient descent ascent (GDA) to solve such problems, where both players take a gradient update simultaneously or alternatively. Despite its simplicity, GDA is known to suffer from a generic issue for minimax optimization: it may cycle around a stable point, exhibit divergent behavior, or converge very slowly since it requires very small learning rates (Gidel et al., 2019; Mertikopoulos et al., 2019). Given the widespread usage of gradient-based methods for solving machine learning problems, first-order optimization algorithms to solve minimax problems have gained considerable popularity in the last few years. Algorithms such as optimistic Gradient Descent Ascent (OG) (Daskalakis et al., 2018; Mertikopoulos et al., 2019) and extra-gradient (EG) (Gidel et al., 2019) can alleviate the issue of GDA for some problems. Yet, it has been shown that these methods can still diverge or cycle around a stable point (Adolphs et al.; Mazumdar et al., 2019; Parker-Holder et al., 2020). For example, these algorithms can fail to find the optimal minimax as shown in Figure 1. This leads to the following question: *Can we design better algorithms for minimax problems?* We answer this in the affirmative, by introducing GDA-AM. We cast the GDA dynamics as a fixed-point iteration problem and compute the iterates effectively using an advanced nonlinear extrapolation method. We show that indeed our algorithm has theoretical and empirical guarantees across a broad range of minimax problems, including GANs.

Our contributions: In this paper, we propose a different approach to solve minimax optimization. Our starting point is to cast the GDA dynamics as a fixed-point iteration. We then highlight that the fixed-point iteration can be solved effectively by using advanced non-linear extrapolation methods such as Anderson Mixing (Anderson, 1965), which we name as GDA-AM. To our best knowledge, this is the first work to improve the GDA dynamics by tapping into advanced fixed-point algorithms.

We demonstrate that GDA dynamics can benefit from Anderson Mixing. In particular, we study bilinear games and give a systematic analysis of GDA-AM for both simultaneous and alternating



(a) Cycling Behavoir (b) Diverging Behavoir (c) Converging to a non-optima

Figure 1: **Left**: $f(x,y) = (4x^2-(y-3x+0.05x^3)^2-0.1y^4)e^{-0.01(x^2+y^2)}$. **Middle**: $-3x^2-y^2+4xy$. **Right**: $f(x,y) = 2x^2 + y^2 + 4xy + \frac{4}{3}y^3 - \frac{1}{4}y^4$. Baseline methods fail to converge to a desirable fixed point, whereas `GDA-AM` with table size $p = 3$ always exhibits desirable behaviors.

versions of GDA. We theoretically show that `GDA-AM` can achieve global convergence guarantees under mild conditions.

We complement our theoretical results with numerical simulations across a variety of minimax problems. We show that for some convex-concave and non-convex-concave functions, `GDA-AM` can converge to the optimal point with little hyper-parameter tuning whereas existing first-order methods are prone to divergence and cycling behaviors.

We also provide empirical results for GAN training across two different datasets, CIFAR10 and CelebA. Given the limited computational overhead of our method, the results suggest that an extrapolation add-on to GDA can lead to significant performance gains. Moreover, the convergence behavior across a variety of problems and the ease-of-use demonstrate the potential of `GDA-AM` to become the minimax optimization workhorse.

2 Preliminaries and Background

2.1 Minimax Optimization

Definition 1. *Point* $(\mathbf{x}^*, \mathbf{y}^*)$ *is a* ***local Nash equilibrium*** *of* f *if there exists* $\delta > 0$ *such that for any* $(\mathbf{x}, \mathbf{y})$ *satisfying* $\|\mathbf{x} - \mathbf{x}^*\| \leq \delta$ *and* $\|\mathbf{y} - \mathbf{y}^*\| \leq \delta$ *we have:* $f(\mathbf{x}^*, \mathbf{y}) \leq f(\mathbf{x}^*, \mathbf{y}^*) \leq f(\mathbf{x}, \mathbf{y}^*)$.

To find the Nash equilibria, common algorithms including GDA, EG and OG, can be formulated as follows. For the two variants of GDA, simultaneous GDA (SimGDA) and alternating GDA (AltGDA), the updates have the following forms:

$$\begin{aligned} \text{Simultaneous}: \quad & \mathbf{x}_{t+1} = \mathbf{x}_t - \eta \nabla_{\mathbf{x}} f(\mathbf{x}_t, \mathbf{y}_t), \quad \mathbf{y}_{t+1} = \mathbf{y}_t + \eta \nabla_{\mathbf{y}} f(\mathbf{x}_t, \mathbf{y}_t) \\ \text{Alternating}: \quad & \mathbf{x}_{t+1} = \mathbf{x}_t - \eta \nabla_{\mathbf{x}} f(\mathbf{x}_t, \mathbf{y}_t), \quad \mathbf{y}_{t+1} = \mathbf{y}_t + \eta \nabla_{\mathbf{y}} f(\mathbf{x}_{t+1}, \mathbf{y}_t). \end{aligned} \tag{2}$$

The EG update has the following form:

$$\begin{aligned} \mathbf{x}_{t+\frac{1}{2}} &= \mathbf{x}_t - \eta \nabla_{\mathbf{x}} f(\mathbf{x}_t, \mathbf{y}_t), & \mathbf{y}_{t+\frac{1}{2}} &= \mathbf{y}_t + \eta \nabla_{\mathbf{y}} f(\mathbf{x}_t, \mathbf{y}_t) \\ \mathbf{x}_{t+1} &= \mathbf{x}_t - \eta \nabla_{\mathbf{x}} f(\mathbf{x}_{t+\frac{1}{2}}, \mathbf{y}_{t+\frac{1}{2}}), & \mathbf{y}_{t+1} &= \mathbf{y}_t + \eta \nabla_{\mathbf{y}} f(\mathbf{x}_{t+\frac{1}{2}}, \mathbf{y}_{t+\frac{1}{2}}). \end{aligned} \tag{3}$$

The OG update has the following form:

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \eta \nabla_{\mathbf{x}} f(\mathbf{x}_t, \mathbf{y}_t) + \frac{\eta}{2} \nabla_{\mathbf{x}} f(\mathbf{x}_{t-1}, \mathbf{y}_{t-1}), \mathbf{y}_{t+1} = \mathbf{y}_t + \eta \nabla_{\mathbf{y}} f(\mathbf{x}_t, \mathbf{y}_t) - \frac{\eta}{2} \nabla_{\mathbf{y}} f(\mathbf{x}_{t-1}, \mathbf{y}_{t-1}). \tag{4}$$

2.2 Fixed-Point Iteration and Anderson Mixing (AM)

Definition 2. $\mathbf{w}^\star$ *is a fixed point of the mapping* g *if* $\mathbf{w}^\star = g(\mathbf{w}^\star)$.

Consider the simple fixed-point iteration $w_{t+1} = g(w_t)$ which produces a sequence of iterates $\{w_0, w_1, \cdots, w_N\}$. In most cases, this converges to the fixed-point, $w^* = g(w^*)$. Take gradient

descent as an example, it can be viewed as iteratively applying the operation: $w_{t+1} = g(w_t) \triangleq w_t - \alpha_t \nabla f(w_t)$, where the limit is the fixed-point $\mathbf{w}^\star = g(\mathbf{w}^\star)$ (i.e.$\nabla f(w_t = 0)$. SimGDA updates can be defined as the repeated application of a nonlinear operator:

$$\mathbf{w}_{t+1} = G_\eta^{(\mathrm{sim})}(\mathbf{w}_t) \triangleq \mathbf{w}_t - \eta V(\mathbf{w}_t) \text{ with } \mathbf{w} = \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix}, V(\mathbf{w}) = \begin{bmatrix} \nabla_{\mathbf{x}} f(\mathbf{x}, \mathbf{y}) \\ -\nabla_{\mathbf{y}} f(\mathbf{x}, \mathbf{y}) \end{bmatrix}$$

Similarly, we can write AltGDA updates as $\mathbf{w}_{t+1} = G_\eta^{(\mathrm{alt})}(\mathbf{w}_t)$. An issue with fixed-point iteration is that it does not always converge, and even in the cases where it does converge, it might do so very slowly. GDA is one example that it could result in the possibility of the operator converging to a limit cycle instead of a single point for the GDA dynamic. A way of dealing with these problems is to use acceleration methods, which can potentially speed up the convergence process and in some cases even decrease the likelihood for divergence.

There are many different acceleration methods, but we will put our focus on an algorithm which we refer to as Anderson Mixing (or Anderson Acceleration). In short, Anderson Mixing (AM) shares the same idea as Nesterov's acceleration. Given a fixed-point iteration $w_t = g(w_{t-1})$, Anderson Mixing argues that a good approximation to the final solution w^* can be obtained as a linear combination of the previous p iterates $w_{t+1} = \sum_{i=0}^{p} \beta_i g(w_{t-p_t+i})$. Since obtaining the proper coefficients β_i is a nonlinear procedure, Anderson Mixing is also known as a nonlinear extrapolation method. The general form of Anderson Mixing is shown in Algorithm 1. For efficiency, we prefer a 'restarted' version with a small table size p that cleans up the table F every p iterations because it avoids solving a linear system of increasing size.

Algorithm 1: Anderson Mixing Prototype (truncated version)

Input: Initial point w_0, Anderson restart dimension p, fixed-point mapping $g : \mathbf{R}^n \to \mathbf{R}^n$.
Output: w_{t+1}
for *t = 0, 1, ...* **do**
  Set $p_t = \min\{t, p\}$.
  Set $F_t = [f_{t-p_t}, \ldots, f_t]$, where $f_i = g(w_i) - w_i$ for each $i \in [t - p_t \,..\, t]$.
  Determine weights $\beta = (\beta_0, \ldots, \beta_{p_t})^T$ that solves $\min_\beta \|F_t \beta\|_2$, s. t. $\sum_{i=0}^{p_t} \beta_i = 1$.
  Set $w_{t+1} = \sum_{i=0}^{p_t} \beta_i g(w_{t-p_t+i})$.
end

2.3 AM AND GENERALIZED MINIMAL RESIDUAL (GMRES)

Developed by Saad & Schultz (1986), Generalized Minimal Residual method (GMRES) is a Krylov subspace method for solving linear system equations. The method approximates the solution by the vector in a Krylov subspace with minimal residual, which is described below.

Definition 3. *Assume we have the linear system of equations* $\mathbf{A}\mathbf{x} = \mathbf{b}$ *with* $\mathbf{A} \in \mathbb{R}^{n \times n}, \mathbf{b} \in \mathbb{R}^n$ *and an initial guess* $\mathbf{x}_0$*. Then we denote the initial residual by* $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$ *and define the* t*th Krylov subspace as* $\mathcal{K}_t = span\{\mathbf{r}_0, \mathbf{A}\mathbf{r}_0, \cdots, \mathbf{A}^{t-1}\mathbf{r}_0\}$.

The tth iterate $\mathbf{x}_t$ of GMRES minimizes the norm of the residual $\mathbf{r}_t = \mathbf{b} - \mathbf{A}\mathbf{x}_t$ in $\mathcal{K}_t$, that is, $\mathbf{x}_t$ solves

$$\min_{\mathbf{x}_t \in \mathbf{x}_0 + \mathcal{K}_t} \|\mathbf{b} - \mathbf{A}\mathbf{x}_t\|_2 .$$

The following formulation is equivalent to GMRES minimization problem and more convenient for implementation. It computes $\widehat{\mathbf{x}}_t$ such that

$$\widehat{\mathbf{x}}_t = \underset{\widehat{\mathbf{x}}_t \in \mathcal{K}_t}{\arg\min} \|\mathbf{b} - \mathbf{A}(\mathbf{x}_0 + \widehat{\mathbf{x}}_t)\|_2 = \underset{\widehat{\mathbf{x}}_t \in \mathcal{K}_t}{\arg\min} \|\mathbf{r}_0 - \mathbf{A}\widehat{\mathbf{x}}_t\|_2 .$$

Using a larger Krylov dimension will improve the convergence of the method, but will require more memory. For this reason, a smaller Krylov subspace dimension t and 'restarted' versions of the method are used in practice Saad (2003).

The convergence of GMRES can be studied through the magnitude of the residual polynomial.

Theorem 2.1 (Lemma 6.31 of Saad (2003))**.** *Let* $\widehat{\mathbf{x}}_t$ *be the approximate solution obtained at the t-th iteration of GMRES being applied to solve* $\mathbf{A}\mathbf{x} = \mathbf{b}$*, and denote the residual as* $\mathbf{r}_t = \mathbf{b} - \mathbf{A}\widehat{\mathbf{x}}_t$*. Then,*

$\mathbf{r}_t$ *is of the form*

$$\mathbf{r}_t = f_t(\mathbf{A})\mathbf{r}_0, \tag{5}$$

where

$$\|\mathbf{r}_t\|_2 = \|f_t(\mathbf{A})\mathbf{r}_0\|_2 = \min_{f_t \in \mathcal{P}_t} \|f_t(\mathbf{A})\mathbf{r}_0\|_2, \tag{6}$$

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

Although GMRES is applied to a system of linear equations not a fixed-point problem, there is a strong connection between Anderson Mixing and GMRES. In AM we are looking for a fixed-point $\mathbf{x}$ such that $\mathbf{G}\mathbf{x} - \mathbf{b} - \mathbf{x} = 0$ and by rearranging this equation we get

$$\mathbf{b} + (\mathbf{G} - \mathbf{I})\mathbf{x} = 0 \Leftrightarrow (\mathbf{I} - \mathbf{G})\mathbf{x} = \mathbf{b}.$$

Theorem 2.2 shows that if GMRES is applied to the system $(\mathbf{I} - \mathbf{G})\mathbf{x} = \mathbf{b}$ and AM is applied to $g(\mathbf{x}) = \mathbf{G}\mathbf{x} + \mathbf{b}$ with the same initial guess and $\mathbf{I} - \mathbf{G}$ is non-singular, then these are equivalent in the sense that the iterates of each algorithm can be obtained directly from the iterates of the other algorithm.

Theorem 2.2 (Equivalence between AM with restart and GMRES (Walker & Ni, 2011a))**.** *Consider the fixed point iteration $\mathbf{x} = g(\mathbf{x})$ where $g(\mathbf{x}) = \mathbf{G}\mathbf{x} + \mathbf{b}$ for $\mathbf{G} \in \mathbb{R}^{n\times n}$ and $\mathbf{b} \in \mathbb{R}^n$. If $\mathbf{I} - \mathbf{G}$ is non-singular, Algorithm 1 produces exactly the same iterates as GMRES being applied to solve $(\mathbf{I} - \mathbf{G})\mathbf{x} = \mathbf{b}$ when both algorithms start with the same initial guess.*

Theorem 2.2 can also be generalized to the restart version of AM an GMRES as well.

3 GDA−AM : GDA WITH ANDERSON MIXING

We propose a novel minimax optimizer, called GDA−AM, that is inspired by recent advances in parameter (or weight) averaging (Wu et al., 2020; Yazici et al., 2019). We argue that a nonlinear adaptive average (combination) is a more appropriate choice for minimax optimization.

3.1 GDA WITH NAÏVE ANDERSON MIXING

We propose to exploit the dynamic information present in the GDA iterates to "smartly" combine the past iterates. This is in contrast to the classical averaging methods (moving averaging and exponential moving averaging) (Yang et al., 2019) that "blindly" combine past iterates. A naïve adoption of Anderson Mixing using the past p GDA iterates for both simGDA and altGDA has the following form:

$$\text{Anderson mixing :} \quad \mathbf{x}_{t+1} = \sum_{i=0}^{p} \beta_i \mathbf{x}_{t-p+i}, \mathbf{y}_{t+1} = \sum_{i=0}^{p} \beta_i \mathbf{y}_{t-p+i}.$$

A direct comparison of GDA with Anderson Mixing, GDA (equation 2), EG (equation 3) and OG (equation 4) illustrates that GDA with Anderson Mixing solves the minimax problem through non-linear parameter combination of the past updates. The Simultaneous and Alternating GDA−AM are briefly summarized in Algorithms 2 and 3 with the truncated Anderson Mixing Algorithm 1 using a table size p.

Algorithm 2: Simultaneous GDA−AM

Input: $\mathbf{x}_0, \mathbf{y}_0$, stepsize η, Anderson table size p
Output: $\mathbf{x}_t, \mathbf{y}_t$
Set $\mathbf{w}_0 = [\mathbf{x}_0, \mathbf{y}_0], sx = length(x_0)$
for *t = 0, 1, ...* **do**
  $\mathbf{x}_t, \mathbf{y}_t = \mathbf{w}_t[0 : sx - 1], \mathbf{w}_t[sx : end]$
  $\mathbf{x}_{t+1} = \mathbf{x}_t - \eta \nabla_{\mathbf{x}} f(\mathbf{x}_t, \mathbf{y}_t)$
  $\mathbf{y}_{t+1} = \mathbf{y}_t - \eta \nabla_{\mathbf{y}} f(\mathbf{x}_t, \mathbf{y}_t)$
  $\mathbf{w}_{t+1} = \begin{bmatrix} \mathbf{x}_{t+1} \\ \mathbf{y}_{t+1} \end{bmatrix}$
  Use Anderson Mixing with table size p to extrapolate $\mathbf{w}_{t+1}$
end
$\mathbf{x}_t, \mathbf{y}_t = \mathbf{w}_{t+1}[0 : sx - 1], \mathbf{w}_{t+1}[sx : end]$
return $\mathbf{x}_t, \mathbf{y}_t$

Algorithm 3: Alternating GDA−AM

Input: $\mathbf{x}_0, \mathbf{y}_0$, stepsize η, Anderson table size p
Output: $\mathbf{x}_t, \mathbf{y}_t$
Set $\mathbf{w}_0 = [\mathbf{x}_0, \mathbf{y}_0], sx = length(x_0)$
for *t = 0, 1, ...* **do**
  $\mathbf{x}_t, \mathbf{y}_t = \mathbf{w}_t[0 : sx - 1], \mathbf{w}_t[sx : end]$
  $\mathbf{x}_{t+1} = \mathbf{x}_t - \eta \nabla_{\mathbf{x}} f(\mathbf{x}_t, \mathbf{y}_t)$
  $\mathbf{y}_{t+1} = \mathbf{y}_t - \eta \nabla_{\mathbf{y}} f(\mathbf{x}_{t+1}, \mathbf{y}_t)$
  $\mathbf{w}_{t+1} = \begin{bmatrix} \mathbf{x}_{t+1} \\ \mathbf{y}_{t+1} \end{bmatrix}$
  Use Anderson Mixing with table size p to extrapolate $\mathbf{w}_{t+1}$
end
$\mathbf{x}_t, \mathbf{y}_t = \mathbf{w}_{t+1}[0 : sx - 1], \mathbf{w}_{t+1}[sx : end]$
return $\mathbf{x}_t, \mathbf{y}_t$

3.2 Implementation details

It is important to note that the Anderson Mixing form shown in Algorithm 1 is for illustrative purpose and not computationally efficient. For example, only one column of F_t needs to be updated at each iteration. In addition, the solution of the least-square problem in Algorithm 1 can also be solved by a quick QR update scheme which costs $(2n + 1)p^2$ (Walker & Ni, 2011a). Thus, from Algorithms 2 and 3, we can see that the major cost of GDA−AM arises from solving the additional linear least squares problem compared to regular GDA at each iteration. Additional implementation details are provided in the Appendix.

4 Convergence results for GDA−AM

In this section, we show that both simultaneous and alternating version GDA−AM converge to the equilibrium for bilinear problems. First, we do not require the learning rate to be sufficiently small. Second, we explicitly provide a linear convergence rate that is faster than EG and OG.

4.1 Bilinear Games

Bilinear games are often regarded as an important simple example for theoretically analyzing and understanding new algorithms and techniques for solving general minimax problems (Gidel et al., 2019; Mertikopoulos et al., 2019; Schaefer & Anandkumar, 2019). In this section, we analyze the convergence property of simultaneous GDA−AM and alternating GDA−AM schemes on the following zero-sum bilinear games:

$$\min_{\mathbf{x} \in \mathbb{R}^n} \max_{\mathbf{y} \in \mathbb{R}^n} f(\mathbf{x}, \mathbf{y}) = \mathbf{x}^T \mathbf{A} \mathbf{y} + \mathbf{b}^T \mathbf{x} + \mathbf{c}^T \mathbf{y}, \quad \mathbf{A} \text{ is full rank.} \tag{7}$$

The Nash equilibrium to the above problem is given by $(\mathbf{x}^*, \mathbf{y}^*) = (-\mathbf{A}^{-T}\mathbf{c}, -\mathbf{A}^{-1}\mathbf{b})$.

4.2 Simultaneous GDA−AM

Suppose $\mathbf{x}_0$ and $\mathbf{y}_0$ are the initial guesses for $\mathbf{x}^*$ and $\mathbf{y}^*$, respectively. Then each iteration of simultaneous GDA can be written in the following matrix form:

$$\begin{bmatrix} \mathbf{x}_{t+1} \\ \mathbf{y}_{t+1} \end{bmatrix} = \underbrace{\begin{bmatrix} \mathbf{I} & -\eta \mathbf{A} \\ \eta \mathbf{A}^T & \mathbf{I} \end{bmatrix}}_{\mathbf{G}^{(Sim)}} \underbrace{\begin{bmatrix} \mathbf{x}_t \\ \mathbf{y}_t \end{bmatrix}}_{\mathbf{w}_t^{(Sim)}} - \eta \underbrace{\begin{bmatrix} \mathbf{b} \\ \mathbf{c} \end{bmatrix}}_{\mathbf{b}^{(Sim)}}. \tag{8}$$

It has been shown that the iteration in equation 8 often cycles and fails to converge for the bilinear problem due to the poor spectrum/numerical range of the fixed point operator $\mathbf{G}^{(Sim)}$ (Gidel et al.,

2019; Azizian et al., 2020; Mokhtari et al., 2020a). Next we show that the convergence can be improved with Algorithm 2.

Theorem 4.1. *[Global convergence for simultaneous* GDA−AM *on bilinear problem] Denote the distance between the stationary point* $\mathbf{w}^*$ *and current iterate* $\mathbf{w}_{(k+1)p}$ *of Algorithm 2 with table size* p *as* $N_{(k+1)p} = \|\mathbf{w}^* - \mathbf{w}_{(k+1)p}\|$. *Then we have the following bound for* N_t

$$N_{(k+1)p}^2 \leq \rho(A) N_{kp}^2 \tag{9}$$

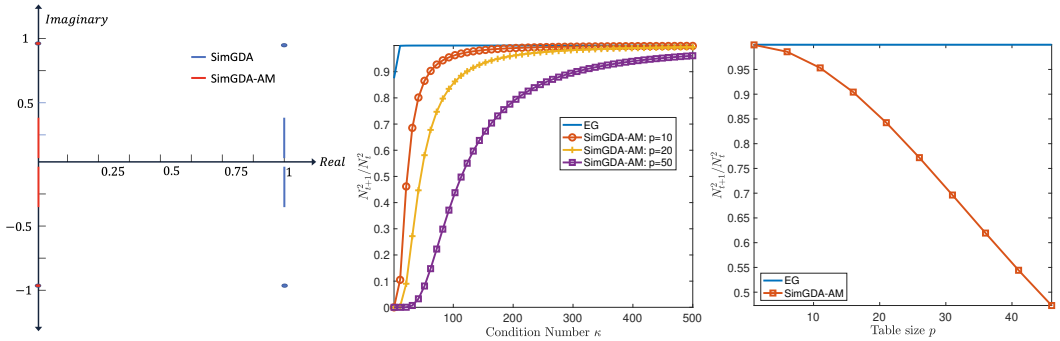
where $\rho(A) = (\frac{1}{T_p(1+\frac{2}{\kappa(\mathbf{A}^T\mathbf{A})-1})})^2$. *Here,* T_p *is the Chebyshev polynomial of first kind of degree* p *and* $\frac{1}{T_p(1+\frac{2}{\kappa(\mathbf{A}^T\mathbf{A})-1})} < 1$ *since* $1 + \frac{2}{\kappa(\mathbf{A}^T\mathbf{A})-1} > 1$.

It is worthy emphasizing that the convergence rate of Algorithm 2 is independent of learning rate η while the convergence results of other methods like EG and OG depend on the learning rate.

Remark 4.1.1. *Both EG and OG have the following form of convergence rate (Mokhtari et al., 2020a) for bilinear problem*

$$N_{t+1}^2 \leq (1 - \frac{c}{\kappa(\mathbf{A}^T\mathbf{A})})N_t^2,$$

where c is a positive constant independent of the problem parameters.



(a) Eigenvalues of iteration matrix of SimGDA and GDA−AM

(b) Different condition number

(c) Different table size, condition number $\kappa = 100$

Figure 2: **Figure 2a**: The blue line is the spectrum of matrix $\mathbf{G}^{(Sim)}$ while the red line is spectrum of matrix $\mathbf{I} - \mathbf{G}^{(Sim)}$. Our method transforms the divergent problem to a convergent problem due to the transformed spectrum. **Figure 2b**: Convergence rate comparison between SimGDA-AM and EG for different condition numbers of $\mathbf{A}$ and fixed table size $p = 10, 20, 50$. **Figure 2c**: Convergence rate comparison between SimGDA-AM and EG for increasing table size on a matrix $\mathbf{A}$ with condition number 100.

4.3 ALTERNATING GDA−AM

The underlying fixed point iteration in Algorithm 3 can be written in the following matrix form:

$$\begin{bmatrix}\mathbf{x}_{t+1}\\ \mathbf{y}_{t+1}\end{bmatrix} = \underbrace{\begin{bmatrix}\mathbf{I} & -\eta\mathbf{A}\\ \eta\mathbf{A}^T & \mathbf{I}-\eta^2\mathbf{A}^T\mathbf{A}\end{bmatrix}}_{\mathbf{G}^{(Alt)}} \underbrace{\begin{bmatrix}\mathbf{x}_t\\ \mathbf{y}_t\end{bmatrix}}_{\mathbf{w}_t^{(Alt)}} - \eta \underbrace{\begin{bmatrix}\mathbf{b}\\ \mathbf{c}\end{bmatrix}}_{\mathbf{b}^{(Alt)}}.$$

According to the equivalence between truncated Anderson acceleration and GMRES with restart, we can analyze the convergence of Algorithm 3 through the convergence analysis of applying GMRES to solve linear systems associated with $\mathbf{G} = \mathbf{I} - \mathbf{G}^{(Alt)}$:

$$\mathbf{G} = \begin{bmatrix}\mathbf{0} & \eta\mathbf{A}\\ -\eta\mathbf{A}^T & \eta^2\mathbf{A}^T\mathbf{A}\end{bmatrix}.$$

Theorem 4.2. *[Global convergence for alternating* GDA−AM *on bilinear problem] Denote the distance between the stationary point* $\mathbf{w}^*$ *and current iterate* $\mathbf{w}_{(k+1)p}$ *of Algorithm 3 with table size* p *as* $N_{(k+1)p} = \|\mathbf{w}^* - \mathbf{w}_{(k+1)p}\|$. *Assume* $\mathbf{A}$ *is normalized such that its largest singular value is equal to* 1. *Then when the learning rate* η *is less than* 2, *we have the following bound for* N_t

$$N_{(k+1)p}^2 \leq \sqrt{1 + \frac{2\eta}{2-\eta}}(\frac{r}{c})^p N_{kp}^2$$

where c *and* r *are the center and radius of a disk* $D(c, r)$ *which includes all the eigenvalues of* $\mathbf{G}$. *Especially,* $\frac{r}{c} < 1$.

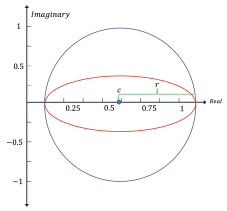


Figure 3: An illustration of the spectrum of $\mathbf{G}$ (red) and the closing circle (blue) in Theorem 4.2.

Theorem 4.2 shows that when $p > \frac{\log \sqrt{\frac{2-\eta}{2+\eta}}}{\log \frac{r}{c}}$, alternating GDA−AM will converge globally.

5 Related work

There is a rich literature on different strategies to alleviate the issue of minimax optimization. Several second-order methods (Adolphs et al.; Mescheder et al., 2017; Mazumdar et al., 2019; Parker-Holder et al., 2020) show that their stable fixed points are exactly either Nash equilibria or local minimax by incorporating second-order information. However, such methods are computationally expensive and thus unsuitable for large applications such as image generation. Focusing on variants of GDA, EG and OG are two widely studied algorithms on improving the GDA dynamics. EG proposed to apply extra-gradient to overcome the cycling behaviour of GDA. OG, originally proposed in Popov (1980) and rediscovered in Daskalakis et al. (2018); Mertikopoulos et al. (2019), is more efficient by storing and re-using the extrapolated gradient for the extrapolation step. Without projection, OG is equivalent to extrapolation from past. Mokhtari et al. (2020b) shows that both of these algorithms can be interpreted as approximations of the classical proximal point method and did a unified analysis for bilinear games. These approaches mentioned the GDA dynamics can be viewed as a fixed-point iteration, but none of them further provides a solution to improve it. In this work, we fill this gap by proposing the application of the extrapolation method directly on the entire GDA dynamics. Unlike OG, EG and their variants (Hsieh et al., 2019; Lei et al., 2021; Thekumparampil et al., 2019; Yang et al., 2019), which regard minimax problems as variational inequality problems (Bruck, 1977; Nemirovski, 2004), our work is from a new perspective and thus orthogonal to these previous approaches.

6 Experiments

In this section, we conduct experiments to see whether GDA−AM improves GDA for minimax optimization from simple to practical problems. We first investigate performance of GDA−AM on bilinear games. In addition, we evaluate the efficacy of our approach on GANs.

6.1 Bilinear Problems

In this section, we answer following questions: **Q1:** How is GDA−AM perform in terms of iteration number and running time? **Q2:** How is the scalability of GDA−AM ? **Q3:** How is the performance of GDA−AM using different table size p? **Q4:** Does GDA−AM converge for large step size η?

We compare the performance with SimGDA, AltGDA, EG, and OG, and EG with Negative Momentum(Azizian et al. (2020)) on bilinear minimax games shown in equation 7 without any constraint.

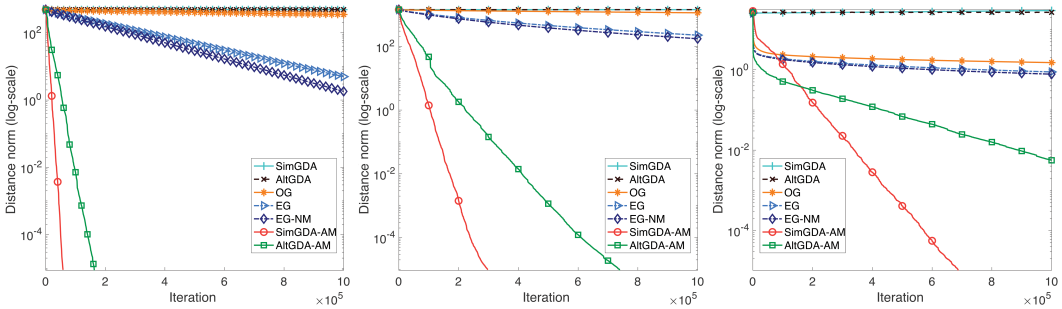
$\mathbf{A}, \mathbf{b}, \mathbf{c}$, and initial points are generated using normally distributed random number. We set the maximum iteration number as 1×10^6, stopping criteria 1×10^{-5} and depict convergence by use of the norm of distance to optima, which is defined as $\|\mathbf{w}^* - \mathbf{w}_t\|$. Similar to Azizian et al. (2020); Wei et al. (2021), the step size is set as 1 after rescaling $\mathbf{A}$ to have 2-norm 1. We present results of different settings in Figures 4, 5, and 6.

We first generate different problem size ($n = 100, 1000, 5000$) and present results of convergence in terms of iteration number in Figure 4. It can be observed that GDA−AM converges in much fewer

iterations for different problem sizes. Note that EG, EG-NM, and OG converge in the end but requires many iterations, thus we plot only a portion for illustrative purposes. Figure 5 depicts the convergence for all methods in terms of time. It can be observed that the running time of `GDA-AM` is faster than EG. Although slower than OG, we can observe `GDA-AM` converges in much less time for all problems. Figure 4 and Figure 5 answer Q1 and Q2; although there is additional computation for `GDA-AM`, it does not hinder the benefits of adopting Anderson Mixing. Even for a large problem size, `GDA-AM` still converges in much less time than the baselines.

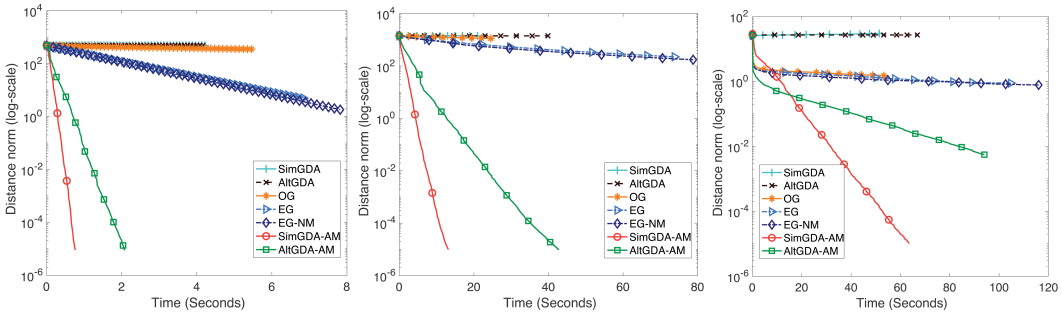
Next, we run `GDA-AM` using different table size p and show the results in Figure 6a and Figure 6b. Figure 6a indicates an increasing of table size results in faster convergence in terms of iteration number, which also verifies our claim in Theorem 4.1. However, we also observe an increased running time when using a larger table size in Figure 6b. Further, we can see that $p = 50$ converges in a comparable time and iterations to $p = 100$. Similar results are found in repeated experiments as well. As a result, our answer to Q3 is that although a larger p means less iterations, a medium p is sufficient and a small p still outperforms the baselines. The optimal choice of p is related to the condition number and step size, which is another interesting topic in the Anderson Mixing community.

Next, we answer Q4 on convergence under different step sizes. Although `GDA-AM` usually converges with suitable step size, our theorem suggests it requires a larger table size when combined with a extremely aggressive step size. Figure 6c shows the convergence under such circumstance. We can observe that although a very large step size goes the wrong way in the beginning, Anderson Mixing can still make it back on track except when $\eta > 1$. It answers the question and confirms our claim that `GDA-AM` can achieve global convergence for bilinear problems for a large step size $\eta > 0$.



(a) $n = 100$ (b) $n = 500$ (c) $n = 1000$

Figure 4: Comparison in terms of iteration: $\min_{\mathbf{x}} \max_{\mathbf{y}} f(\mathbf{x}, \mathbf{y}) = \mathbf{x}^T\mathbf{A}\mathbf{y} + \mathbf{b}^T\mathbf{x} + \mathbf{c}^T\mathbf{y}$. We use different problem size and fix $p = 10, \eta = 1$ for all experiments.

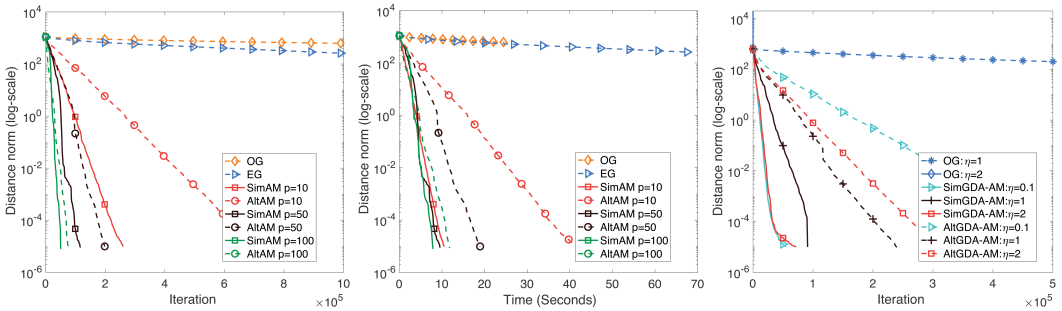


(a) Time comparison for Figure 4a (b) Time comparison for Figure 4b (c) Time comparison for Figure 4c

Figure 5: Comparison between methods in terms of time.

6.2 GAN Experiments: Image Generation

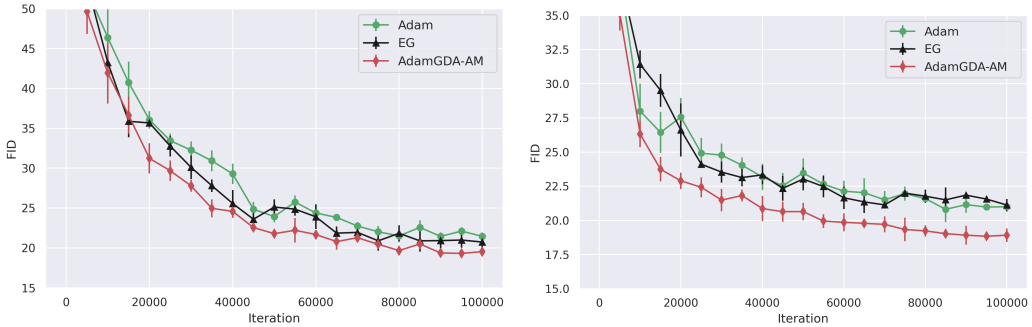
We apply our method to the CIFAR10 dataset (Krizhevsky, 2009) and use the ResNet architecture with WGAN-GP (Gulrajani et al., 2017) and SNGAN (Miyato et al., 2018) objective. We also compared



(a) Effects of p in terms of iteration (b) Time compasion for Figure 6a (c) Effect of step size η, $p = 10$

Figure 6: Effects of table size p and step size η, $n = 500$

the performance of GDA−AM using cropped CelebA (64×64) (Liu et al., 2015) on WGAN-GP. We compare with Adam and extra-gradient with Adam (EG) as it offers significant improvement over OG. Models are evaluated using the inception score (IS) (Salimans et al., 2016) and FID (Heusel et al., 2017) computed on 50,000 samples. For fair comparison, we fixed the same hyperparamters of Adam for all methods after an extensive search. Experiments were run with 5 random seeds. We show results in Figure 7 and Table 1. Table 1 reports the best IS and FID (averaged over 5 runs) achieved on these datasets by each method. We see that GDA−AM yields improvements over the baselines in terms of generation quality.



(a) WGAN-GP (ResNet) (b) SNGAN (ResNet)

Figure 7: FID (lower or ↓ is better)

Table 1: Best inception scores and FID for Cifar10 and FID for CelebA (IS is a less informative metric for celebA).

	WGAN-GP(ResNet)			SNGAN(ResNet)	
	CIFAR10		CelebA	CIFAR10	
Method	IS ↑	FID ↓	FID	IS	FID
Adam	7.76 ±.11	22.45 ±.65	8.43 ±.05	8.21 ±.05	20.81 ±.16
EG	7.83 ±.08	20.73 ±.22	8.15 ±.06	8.15 ±.07	21.12 ±.19
Ours (GDA-RAM)	**8.05** ±.06	**19.32** ±.16	**7.82** ±.06	**8.38** ±.04	**18.84** ±.13

7 Conclusion

The key takeaway from our work is that the GDA dynamic can be improved by advanced extrapolation method (Anderson Mixing). We prove the convergence property of GDA−AM and obtain a faster convergence rate than EG and OG on the bilinear problem. Empirically, we verify our claim for such a problem and show the efficacy of GDA−AM in a deep learning setting as well.

REFERENCES

Leonard Adolphs, Hadi Daneshmand, Aurelien Lucchi, and Thomas Hofmann. Local saddle point optimization: A curvature exploitation approach. Proceedings of Machine Learning Research. PMLR.

Donald G. Anderson. Iterative procedures for nonlinear integral equations. 1965.

Waïss Azizian, Damien Scieur, Ioannis Mitliagkas, Simon Lacoste-Julien, and Gauthier Gidel. Accelerating smooth games by manipulating spectral shapes. In *The 23rd International Conference on Artificial Intelligence and Statistics, AISTATS 2020, 26-28 August 2020, Online [Palermo, Sicily, Italy]*, volume 108 of *Proceedings of Machine Learning Research*, pp. 1705–1715. PMLR, 2020.

Ronald E. Bruck. On the weak convergence of an ergodic iteration for the solution of variational inequalities for monotone operators in hilbert space. *Journal of Mathematical Analysis and Applications*, 1977.

C. Daskalakis, Andrew Ilyas, Vasilis Syrgkanis, and Haoyang Zeng. Training gans with optimism. *ArXiv*, abs/1711.00141, 2018.

Haw-ren Fang and Yousef Saad. Two classes of multisecant methods for nonlinear acceleration. *Numerical Linear Algebra with Applications*, 16(3):197–221, 2009.

Gauthier Gidel, Hugo Berard, Gaëtan Vignoud, Pascal Vincent, and Simon Lacoste-Julien. A variational inequality perspective on generative adversarial networks. In *7th International Conference on Learning Representations, ICLR*, 2019.

I. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, S. Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, 2014.

Anne Greenbaum. *Iterative methods for solving linear systems*. SIAM, 1997.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, 2017.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems*, 2017.

Yu-Guan Hsieh, F. Iutzeler, J. Malick, and P. Mertikopoulos. On the convergence of single-call stochastic extra-gradient methods. In *NeurIPS*, 2019.

A. Krizhevsky. Learning multiple layers of features from tiny images. 2009.

Qi Lei, Sai Ganesh Nagarajan, Ioannis Panageas, and Xiao Wang. Last iterate convergence in no-regret learning: constrained min-max optimization for convex-concave landscapes. In *AISTATS*, 2021.

S. Li, Yi Wu, Xinyue Cui, Honghua Dong, Fei Fang, and Stuart J. Russell. Robust multi-agent reinforcement learning via minimax deep deterministic policy gradient. In *AAAI*, 2019.

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *6th International Conference on Learning Representations, ICLR 2018,*, 2018.

Eric V. Mazumdar, Michael I. Jordan, and S. Sastry. On finding local nash equilibria (and only local nash equilibria) in zero-sum games. *ArXiv*, abs/1901.00838, 2019.

Panayotis Mertikopoulos, Bruno Lecouat, Houssam Zenati, Chuan-Sheng Foo, Vijay Chandrasekhar, and Georgios Piliouras. Optimistic mirror descent in saddle-point problems: Going the extra gradient mile. In *7th International Conference on Learning Representations, ICLR*, 2019.

Lars Mescheder, Sebastian Nowozin, and Andreas Geiger. The numerics of gans. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, 2017.

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Y. Yoshida. Spectral normalization for generative adversarial networks. *ArXiv*, abs/1802.05957, 2018.

Aryan Mokhtari, Asuman Ozdaglar, and Sarath Pattathil. A unified analysis of extra-gradient and optimistic gradient methods for saddle point problems: Proximal point approach. In *International Conference on Artificial Intelligence and Statistics*, pp. 1497–1507. PMLR, 2020a.

Aryan Mokhtari, Asuman Ozdaglar, and Sarath Pattathil. A unified analysis of extra-gradient and optimistic gradient methods for saddle point problems: Proximal point approach. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, Proceedings of Machine Learning Research. PMLR, 2020b.

A. Nemirovski. Prox-method with rate of convergence o(1/t) for variational inequalities with lipschitz continuous monotone operators and smooth convex-concave saddle point problems. *SIAM J. Optim.*, 2004.

Jack Parker-Holder, Luke Metz, Cinjon Resnick, Hengyuan Hu, Adam Lerer, Alistair Letcher, Alexander Peysakhovich, Aldo Pacchiano, and Jakob Foerster. Ridge rider: Finding diverse solutions by following eigenvectors of the hessian. In *Advances in Neural Information Processing Systems*, 2020.

L. Popov. A modification of the arrow-hurwicz method for search of saddle points. *Mathematical notes of the Academy of Sciences of the USSR*, 1980.

Youcef Saad and Martin H. Schultz. Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 1986.

Yousef Saad. *Iterative methods for sparse linear systems*. SIAM, 2003.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, Xi Chen, and Xi Chen. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, 2016.

Florian Schaefer and Anima Anandkumar. Competitive gradient descent. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, 2019.

Kiran K Thekumparampil, Prateek Jain, Praneeth Netrapalli, and Sewoong Oh. Efficient algorithms for smooth minimax optimization. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, 2019.

John von Neumann and Oskar Morgenstern. *Theory of Games and Economic Behavior*. Princeton University Press, 1944.

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

Homer F. Walker and Peng Ni. Anderson acceleration for fixed-point iterations. 2011b.

Chen-Yu Wei, Chung-Wei Lee, Mengxiao Zhang, and Haipeng Luo. Linear last-iterate convergence in constrained saddle-point optimization. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=dx11_7vm5_r`.

Yue Wu, Pan Zhou, A. Wilson, E. Xing, and Zhiting Hu. Improving gan training with probability ratio clipping and sample reweighting. *ArXiv*, abs/2006.06900, 2020.

Minghan Yang, A. Milzarek, Z. Wen, and T. Zhang. A stochastic extra-step quasi-newton method for nonsmooth nonconvex optimization. *arXiv: Optimization and Control*, 2019.

Yasin Yazici, Chuan-Sheng Foo, Stefan Winkler, Kim-Hui Yap, Georgios Piliouras, and Vijay Chandrasekhar. The unusual effectiveness of averaging in GAN training. In *7th International Conference on Learning Representations, ICLR , 2019*, 2019.

A ANDERSON MIXING IMPLEMENTATION DETAILS

In this section, we discuss the efficient implementation of Anderson Mixing. We start with generic Anderson Mixing prototype (Algorithm 4) and then present the idea of Quick QR-update Anderson Mixing implementation as described in Walker & Ni (2011b), which is commonly used in practice. For each iteration $t \geq 0$, AM prototype solves a least squares problem with a normalization constraint. The intuition is to minimize the norm of the weighted residuals of the previous m iterates.

Algorithm 4: Anderson Mixing Prototype (truncated version)

Input: Initial point w_0, Anderson restart dimension p, fixed-point mapping $g : \mathbf{R}^n \to \mathbf{R}^n$.
Output: w_{t+1}
for *t = 0, 1, ...* **do**
  Set $p_t = \min\{t, p\}$.
  Set $F_t = [f_{t-p_t}, \ldots, f_t]$, where $f_i = g(w_i) - w_i$ for each $i \in [t - p_t \,..\, t]$.
  Determine weights $\beta = (\beta_0, \ldots, \beta_{p_t})^T$ that solves $\min_\beta \|F_t\beta\|_2$, s. t. $\sum_{i=0}^{p_t} \beta_i = 1$.
  Set $w_{t+1} = \sum_{i=0}^{p_t} \beta_i g\left(w_{t-p_t+i}\right)$.
end

The constrained linear least-squares problem in Algorithm AA can be solved in a number of ways. Our preference is to recast it in an unconstrained form suggested in Fang & Saad (2009); Walker & Ni (2011b) that is straightforward to solve and convenient for implementing efficient updating of QR.

Define $f_i = g(w_i) - w_i$, $\triangle f_i = f_{i+1} - f_i$ for each i and set $F_t = [f_{t-p_t}, \ldots, f_t]$, $\mathcal{F}_t = [\triangle f_{t-p_t}, \ldots, \triangle f_t]$. Then solving the least-squares problem ($\min_\beta \|F_t\beta\|_2$, s. t. $\sum_{i=0}^{p_t} \beta_i = 1$) is equivalent to

$$\min_{\gamma=\left(\gamma_0,\ldots,\gamma_{p_t-1}\right)^T} \|f_t - \mathcal{F}_t\gamma\|_2 \tag{10}$$

where α and γ are related by $\alpha_0 = \gamma_0, \alpha_i = \gamma_i - \gamma_{i-1}$ for $1 \leq i \leq p_t - 1$, and $\alpha_{p_t} = 1 - \gamma_{p_t-1}$.

Now the inner minimization subproblem can be efficiently solved as an unconstrained least squares problem by a simple variable elimination. This unconstrained least-squares problem leads to a modified form of Anderson Mixing

$$w_{t+1} = g\left(w_t\right) - \sum_{i=0}^{p_t-1} \gamma_i^{(t)} \left[g\left(w_{t-p_t+i+1}\right) - g\left(w_{t-p_t+i}\right)\right] = g\left(w_t\right) - \mathcal{G}_t\gamma^{(t)}$$

where $\mathcal{G}_t = [\triangle g_{t-p_t}, \ldots, \triangle g_{t-1}]$ with $\triangle g_i = g(w_{i+1}) - g(w_i)$ for each i.

To obtain $\gamma^{(t)} = \left(\gamma_0^{(t)}, \ldots, \gamma_{p_t-1}^{(t)}\right)^T$ by solving equation 10 efficiently, we show how the successive least-squares problems can be solved efficiently by updating the factors in the QR decomposition $\mathcal{F}_t = Q_t R_t$ as the algorithm proceeds. We assume a think QR decomposition, for which the solution of the least-squares problem is obtained by solving the $p_t \times p_t$ linear system $R\gamma = Q' * f_t$. Each $\mathcal{F}_t$ is $n \times p_t$ and is obtained from $\mathcal{F}_{t-1}$ by adding a column on the right and, if the resulting number of columns is greater than p, also cleaning up (re-initialize) the table. That is,we never need to delete the left column because cleaning up the table stands for a restarted version of AM. As a result, we only need to handle two cases; 1 the table is empty(cleaned). 2 the table is not full. When the table is empty, we initialize $\mathcal{F}_1 = Q_1 R_1$ with $Q_1 = \triangle f_0 / \|\triangle f_0\|_2$ and $R = \|\triangle f_0\|_2$. If the table size is smaller than p, we add a column on the right of $\mathcal{F}_{t-1}$. Have $\mathcal{F}_{t-1} = QR$, we update Q and R so that $\mathcal{F}_t = [\mathcal{F}_{t-1}, \Delta f_{t-1}] = QR$. It is a single modified Gram–Schmidt sweep that is described as follows:

Algorithm 5: QR-updating procedures

for $i = 1, \ldots, p_{t-1}$ **do**
  Set $R(i, p_t) = Q(:, i)' * \triangle f_{t-1}$.
  Update $\triangle f_{t-1} \leftarrow \Delta f_{t-1} - R\left(i, p_t\right) * Q(:, i)$
end
Set $Q\left(:, p_t\right) = \triangle f_{t-1} / \|\triangle f_{t-1}\|_2$ and $R\left(p_t, p_t\right) = \|\Delta f_{t-1}\|_2$

Note that we do not explicitly conduct QR decomposition in each iteration, instead we update the factors ($O(p^2 n)$) and then solve a linear system using back substitution which has a complexity of

$O(p^2)$. Based on this complexity analysis, we can find Anderson Mixing with QR-updating scheme has limited computational overhead than GDA (or OG). This explains why `GDA-AM` is faster than EG but slower than OG in terms of running time of each iteration.

B PROOF OF THEOREMS

We first provide proof of Theorem 4.1.

Theorem B.1 (Global convergence for simultaneous `GDA-AM` on bilinear problem)**.** *Denote the distance between the stationary point $\mathbf{w}^*$ and current iterate $\mathbf{w}_{(k+1)p}$ of Algorithm 2 with Anderson restart dimension p as $N_{(k+1)p} = dist(\mathbf{w}^*, \mathbf{w}_{(k+1)p})$. Then we have the following bound for N_t Algorithm 2 is unconditionally convergent*

$$N_{(k+1)p} \leq \frac{1}{T_p(1+\frac{2}{\kappa(\mathbf{A}^T\mathbf{A})-1})} N_{kp} \tag{11}$$

where T_p is the Chebyshev polynomial of first kind of degree p and $\frac{1}{T_p(1+\frac{2}{\kappa(\mathbf{A}^T\mathbf{A})-1})} < 1$ since $1+\frac{2}{\kappa(\mathbf{A}^T\mathbf{A})-1} > 1$.

Proof of Theorem 4.1. Note that $\mathbf{I} - \mathbf{G}^{(Sim)}$ is a normal matrix which will be denoted as $\mathbf{G}$ for notational simplicity. Thus it admits the following eigendecomposition:

$$\mathbf{G} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T, \quad \mathbf{U}\mathbf{U}^T = \mathbf{I}, \quad \Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_{2n}). \tag{12}$$

Based on the equivalence between GMRES and Anderson Mixing, we know that the convergence rate of simultaneous `GDA-AM` can be estimated by the spectrum of $\mathbf{G}$. Especially, it holds that

$$\mathbf{r}_{(k+1)p} = \mathbf{U} f_p(\mathbf{\Lambda}) \mathbf{U}^T \mathbf{r}_{kp}. \quad f_p \in \mathcal{P}_p \tag{13}$$

where $\mathcal{P}_p$ is the family of residual polynomials with degree p such that $f_p(0) = 1, \forall f_p \in \mathcal{P}_p$. According to Lemma 2.1, we have the following estimation

$$\|\mathbf{r}_{(k+1)p}\|_2 = \min_{f_p \in \mathcal{P}_p} \|f_p(\mathbf{G})\mathbf{r}_{kp}\|_2 \leq \min_{f_p \in \mathcal{P}_p} \max_i |f_p(\lambda_i)| \|\mathbf{r}_{kp}\|_2. \tag{14}$$

Due to the block structure of $\mathbf{G}$, the eigenvalues of $\mathbf{G}$ can be computed explicitly as

$$\pm\, \eta\sigma_i\sqrt{-1},\ i = 1, \ldots, n, \tag{15}$$

where σ_i is the ith largest singular value of matrix $\mathbf{A}$. This shows that the eigenvalues of $\mathbf{G}$ are n pairs of purely imaginary numbers excluding 0 since $\mathbf{A}$ has full rank.

Since the eigenvalues of $\mathbf{G}$ are distributed in two intervals excluding the origin

$$I = [-\eta\sigma_{max}(\mathbf{A})\sqrt{-1}, -\eta\sigma_{min}(\mathbf{A})\sqrt{-1}] \cup [\eta\sigma_{min}(\mathbf{A})\sqrt{-1}, \eta\sigma_{max}(\mathbf{A})\sqrt{-1}],$$

it can be shown that the following p-th degree polynomial with value 1 at the origin that has the minimal maximum deviation from 0 on I is given by:

$$f_p(z) = \frac{T_l(q(\sqrt{-1}z))}{T_l(q(0))}, \quad q(\sqrt{-1}z) = 1 - \frac{2(\sqrt{-1}z - \eta\sigma_{min})(\sqrt{-1}z + \eta\sigma_{min})}{(\eta\sigma_{max}(\mathbf{A}))^2 - (\eta\sigma_{min}(\mathbf{A}))^2} \tag{16}$$

where $l = [\frac{p}{2}]$ and T_l is the Chebyshev polynomial of first kind of degree l. The function $q(\sqrt{-1}z)$ maps I to $[-1, 1]$. Thus the numerator of the polynomial f_p is bounded by 1 on I. The size of denominator can be determined by the method discussed in Chapter 3 of Greenbaum (1997). Assume $q(0) = \frac{1}{2}(y + y^{-1})$, then $T_l(q(0)) = \frac{1}{2}(y^l + y^{-l})$. Then y can be determined by solving

$$q(0) = \frac{(\eta\sigma_{max}(\mathbf{A}))^2 + (\eta\sigma_{min}(\mathbf{A}))^2}{(\eta\sigma_{max}(\mathbf{A}))^2 - (\eta\sigma_{min}(\mathbf{A}))^2}. \tag{17}$$

The solutions to this equation are

$$y_1 = \frac{\eta\sigma_{max}(\mathbf{A}) + \eta\sigma_{min}(\mathbf{A})}{\eta\sigma_{max}(\mathbf{A}) - \eta\sigma_{min}(\mathbf{A})} \quad \text{or} \quad y_2 = \frac{\eta\sigma_{max}(\mathbf{A}) - \eta\sigma_{min}(\mathbf{A})}{\eta\sigma_{max}(\mathbf{A}) + \eta\sigma_{min}(\mathbf{A})}. \tag{18}$$

Then plugging the value of $q(0)$ into the polynomial f_p yields

$$\begin{aligned}\frac{\|\mathbf{r}_{(k+1)p}\|}{\|\mathbf{r}_{kp}\|} &\leq 2\Big(\frac{\sqrt{\eta^2\sigma_{max}^2(\mathbf{A})}-\sqrt{\eta^2\sigma_{min}^2(\mathbf{A})}}{\sqrt{\eta^2\sigma_{max}^2(\mathbf{A})}+\sqrt{\eta^2\sigma_{min}^2(\mathbf{A})}}\Big)^l \\ &= 2\Big(\frac{\sigma_{max}(\mathbf{A})-\sigma_{min}(\mathbf{A})}{\sigma_{max}(\mathbf{A})+\sigma_{min}(\mathbf{A})}\Big)^l = 2\Big(\frac{\kappa(\mathbf{A})-1}{\kappa(\mathbf{A})+1}\Big)^l\end{aligned} \tag{19}$$

Note that N_t and $\mathbf{r}_t$ is related through $\mathbf{G}(\mathbf{w}_t-\mathbf{w}^*)=\mathbf{r}_t$. Therefore,

$$\begin{aligned}N_{(k+1)p} &= \|\mathbf{w}_{(k+1)p}-\mathbf{w}^*\|_2 = \|\mathbf{G}^{-1}\mathbf{r}_{(k+1)p}\|_2 = \min_{f_p\in\mathcal{P}_p}\|\mathbf{G}^{-1}f_p(\mathbf{G})\mathbf{G}(\mathbf{w}_{kp}-\mathbf{w}^*)\|_2 \\ &\leq \min_{f_p\in\mathcal{P}_p}\max_i|f_p(\lambda_i)|\|\mathbf{w}_{kp}-\mathbf{w}^*\|_2 \leq 2\Big(1-\frac{2}{\kappa(\mathbf{A})+1}\Big)^{\frac{p}{2}}N_{kp}.\end{aligned} \tag{20}$$

Actually a tighter bound can be proved after noting that the problem is essentially equivalent to polynomial minmax problem on the interval:

$$\tilde{I} = [\eta^2\sigma_{min}^2(\mathbf{A}),\ \ \eta^2\sigma_{max}^2(\mathbf{A})],$$

Then it is well known that,

$$\begin{aligned}N_{(k+1)p} &\leq \min_{f_p\in\mathcal{P}_p}\max_{\lambda_i\in[\eta^2\sigma_{min}^2(\mathbf{A}),\ \eta^2\sigma_{max}^2(\mathbf{A})]}|f_p(\lambda_i)|\|\mathbf{w}_{kp}-\mathbf{w}^*\|_2 \leq \frac{1}{T_p(1+2\frac{\sigma_{min}^2}{\sigma_{max}^2-\sigma_{min}^2})}N_{kp} \\ &\leq \frac{1}{T_p(1+\frac{2}{\kappa(\mathbf{A}^T\mathbf{A})-1})}N_{kp}\end{aligned} \tag{21}$$

where T_p Chebyshev polynomial of degree p of the first kind and $\frac{1}{T_p(1+\frac{2}{\kappa(\mathbf{A}^T\mathbf{A})-1})}<1$. Explicitly,

$$\begin{aligned}T_p(1+\frac{2}{\kappa(\mathbf{A}^T\mathbf{A})-1}) = \frac{1}{2}\Big[&\Big(1+\frac{2}{\kappa(\mathbf{A}^T\mathbf{A})-1}+\sqrt{(1+\frac{2}{\kappa(\mathbf{A}^T\mathbf{A})-1})^2-1}\Big)^p \\ &+\Big(1+\frac{2}{\kappa(\mathbf{A}^T\mathbf{A})-1}+\sqrt{(1+\frac{2}{\kappa(\mathbf{A}^T\mathbf{A})-1})^2-1}\Big)^{-p}\Big]\end{aligned}$$

□

Next, we give the proof of Theorem 4.2.

Theorem B.2 (Global convergence for alternating GDA-RAM on bilinear problem)**.** *Denote the distance between the stationary point $\mathbf{w}^*$ and current iterate $\mathbf{w}_{(k+1)p}$ of Algorithm 3 with Anderson restart dimension p as $N_{(k+1)p}=dist(\mathbf{w}^*,\mathbf{w}_{(k+1)p})$. Assume $\mathbf{A}$ is normalized such that its largest singular value is equal to 1. Then when the learning rate η is less than 2, we have the following bound for N_t*

$$N_{(k+1)p}^2 \leq \sqrt{1+\frac{2\eta}{2-\eta}}(\frac{r}{c})^p N_{kp}^2$$

where c and r are the center and radius of a disk $D(c,r)$ which includes all the eigenvalues of $\mathbf{G}$ in equation 4.3. Especially, $\frac{r}{c}<1$.

Proof. Since the residual $\mathbf{r}_p$ of AA at p-th iteration has the form of

$$\mathbf{r}_p = (\mathbf{I}-\sum_{i=1}^p\mathbf{G}^i)\mathbf{r}_0,$$

and AA minimizes the residual, we have

$$\|\mathbf{r}_{(k+1)p}\|_2^2 \leq \min_\beta\|\mathbf{r}_{kp}-\beta\mathbf{G}^i\mathbf{r}_{kp}\|_2^2 \leq \min_{f_p\in\mathcal{P}_p}\|f_p(\mathbf{G})\mathbf{r}_{kp}\|_2^2,$$

where $\mathcal{P}_p$ is the family of polynomials with degree p such that $f_p(0) = 1, \forall f_p \in \mathcal{P}_p$. It's easy to see that $\mathbf{G}$ is unitarily similar to a block diagonal matrix Λ with 2×2 blocks as follows:

$$\begin{bmatrix} 0 & \eta\sigma_i \\ -\eta\sigma_i & (\eta\sigma_i)^2 \end{bmatrix} \quad \forall\, i \in [n].$$

Thus the eigenvalues of G can be easily identified as

$$\lambda_{\pm i} = \frac{(\eta\sigma_i(\eta\sigma_i \pm \sqrt{(\eta\sigma_i)^2 - 4}))}{2}, \quad i \in [n].$$

where $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n$ are the singular values of $\mathbf{A}$. Furthermore, the eigenvector and eigenvalue associated with each 2×2 diagonal block are

$$\begin{bmatrix} 0 & \eta\sigma_i \\ -\eta\sigma_i & (\eta\sigma_i)^2 \end{bmatrix} \begin{bmatrix} 1 \\ \frac{\lambda_{\pm i}}{\eta\sigma_i} \end{bmatrix} = \lambda_{\pm i} \begin{bmatrix} 1 \\ \frac{\lambda_{\pm i}}{\eta\sigma_i} \end{bmatrix}$$

Thus $\mathbf{G}$ is diagonalizable and denote the matrix with the columns of eigenvectors of $\mathbf{G}$ by X. The real part of the eigenvalues of $\mathbf{G}$ are at least

$$\mathcal{R}(\lambda_{\pm i}) \geq \frac{(\eta\sigma_i)^2}{2}, \quad i \in [n]. \tag{22}$$

And since $|\eta\sigma_i| \geq |\sqrt{(\eta\sigma_i)^2 - 4})|$, all the eigenvalues will be included in a disk $D(c, r)$ which is included in the right half plane. Moreover, both c and r being greater than zero indicates that $\frac{r}{c} < 1$. Finally, setting $f_p = (\frac{c-t}{c})^p$ will lead to the desired bound

$$\begin{aligned} N_{(k+1)p} &= \|\mathbf{w}_{(k+1)p} - \mathbf{w}^*\|_2 = \|\mathbf{G}^{-1}\mathbf{r}_{(k+1)p}\|_2 \leq \min_{f_p \in \mathcal{P}_p} \||\mathbf{G}^{-1} f_p(\mathbf{G})\mathbf{r}_{kp}\|_2 \\ &= \min_{f_p \in \mathcal{P}_p} \||\mathbf{G}^{-1} f_p(\mathbf{G})\mathbf{G}(\mathbf{w}_{kp} - \mathbf{w}^*)\|_2 \leq \kappa_{\mathbf{G}}(\frac{r}{c})^p \|\mathbf{w}_{kp} - \mathbf{w}^*\|_2 = \kappa_{\mathbf{G}}(\frac{r}{c})^p N_{kp} \end{aligned}$$

where $\kappa_{\mathbf{G}}$ is the condition number of X. Since $\mathbf{G}$ and Λ are unitarily similar, $\kappa_{\mathbf{G}}$ is equal to the condition number of the eigenvector matrix of Λ. The eigenvector matrix of Λ is a block diagonal matrix with the ith block as $\begin{bmatrix} 1 & 1 \\ \frac{\lambda_{+i}}{\eta\sigma_i} & \frac{\lambda_{-i}}{\eta\sigma_i} \end{bmatrix}$. Thus the singluar values of the eigenvector matrix of Λ is equal to the union of the singular values of these 2-by-2 blocks. Under the assumption that the largest singular value of $\mathbf{A}$ are equal to 1 and the learning rate is less than 2, it is easy to find the singular values of the eigenvector matrix of Λ are $\sqrt{2 \pm \eta\sigma_i}$. Thus, $\kappa_{\mathbf{G}} = \frac{\sqrt{2+\eta\sigma_{\max}}}{\sqrt{2-\eta\sigma_{\max}}} = \frac{\sqrt{2+\eta}}{\sqrt{2-\eta}} = \sqrt{1 + \frac{2\eta}{2-\eta}}$. □

C Additional Experiments

C.1 1d Minimax functions

We begin with investigating the empirical performance of `GDA-AM` for 6 non-trivial 1d bivariate functions. We set initial points as $(3, 3)$ and m as 20 or 5 for all functions. We use optimal learning rates for all methods on each problem. Results are shown in Figure 8, 9, 10, 11, 12 and 13. We observe `GDA-AM` consistently outperforms all baselines and improves convergence. It is worthwhile to mention that the difference between `GDA-AM` and traditional averaging is twofold. First, traditional averaging does not involve an adaptive averaging scheme and thus blindly converge to $(0, 0)$ for all 1d bivariate functions. In contrast, `GDA-AM` obtains optimal weights by solving a small linear system on past iterates. Using different weights for each iteration, `GDA-AM` is able to minimize the residual of past iterates and thus find the solution of a fixed-point iteration. More importantly, averaging does not change the GDA dynamic because averaging generates a new sequence of parameters based on GDA iterates. This means averaging is independent with base training algorithm (GDA here). However, `GDA-AM` changes the dynamic directly by overwriting the latest iterate. It means Anderson Mixing interacts with GDA, which is another major difference from averaging.

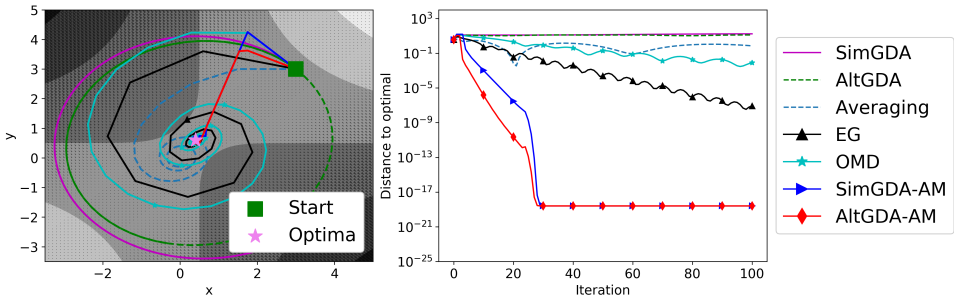


Figure 8: $f(x, y) = (x - \frac{1}{2})(y - \frac{1}{2}) + \frac{1}{3}e^{-(x-0.25)^2-(y-0.75)^2)}$. The optima for this function is not $(0, 0)$. Because averaging blindly converges to $(0, 0)$, it can never find the correct solution.

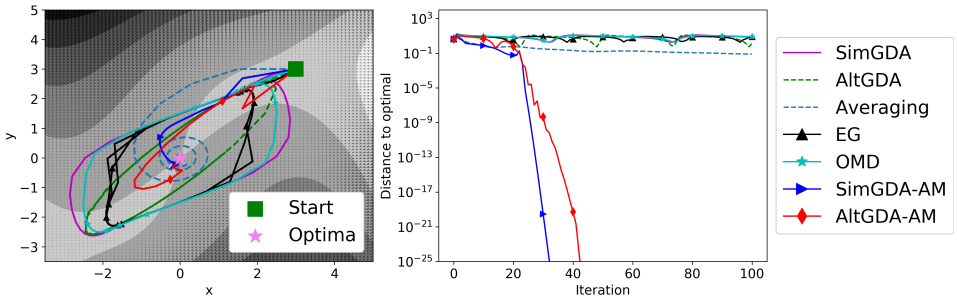


Figure 9: $f(x, y) = (4x^2-(y-3x+0.05x^3)^2-0.1y^4)e^{-0.01(x^2+y^2)}$. All baselines except averaging are cyclying around the optima. Averaging is converging slowly.

C.2 Density estimation

To test our proposed method, we evaluate our method on two low-dimension density estimation problems, mixture of 25 Gaussians and Swiss roll. For both generator and discriminator, we use fully connected neural networks with 3 hidden layers and 128 hidden units in each layer. Except for the output layer of discriminator that uses a sigmoid activation, we use tanh-activation for all other layers. We run Adam and `GDA-AM` for 50000 steps. The learning rate is set as 2×10^{-4} and $\beta_1 = 0, \beta_2 = 0.9$ after an extensive grid search, which is close to the maximal possible stepsize under which the methods rarely diverge. Figure 14 and 15 show the output after $\{1K, 10K, 30K, 50K\}$ iterations. It can be seen that our method converges faster to the target distribution offers a improvement over Adam. In addition, we can observe that the generated samples using our method gather around the circle and are less connected with other circles.

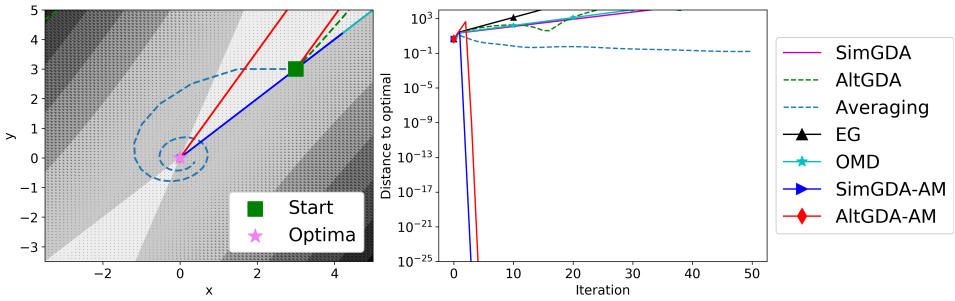


Figure 10: $f(x, y) = -3x^2 - y^2 + 4xy$. Baselines tend to diverge. Averaging is converging slowly again because averaging can only blindly converge to $(0, 0)$ and the optima for this function is $(0, 0)$.

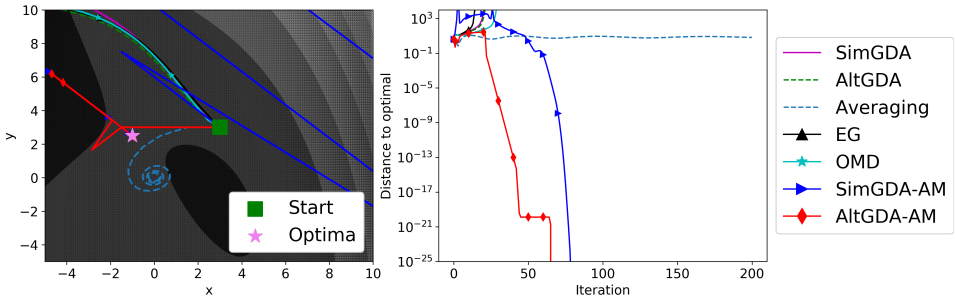


Figure 11: $f(x, y) = \frac{1}{3}x^3 + y^2 + 2xy - 6x - 3y + 4$.

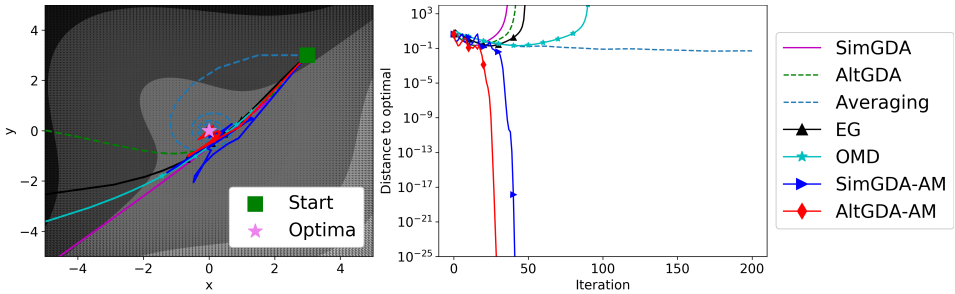


Figure 12: $f(x, y) = x^3 - y^3 - 2xy + 6$.

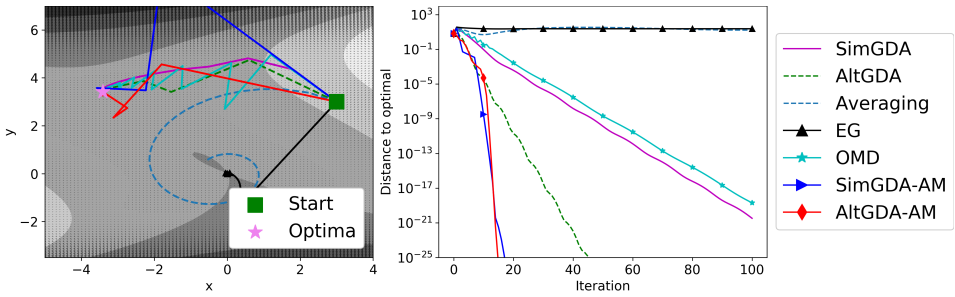
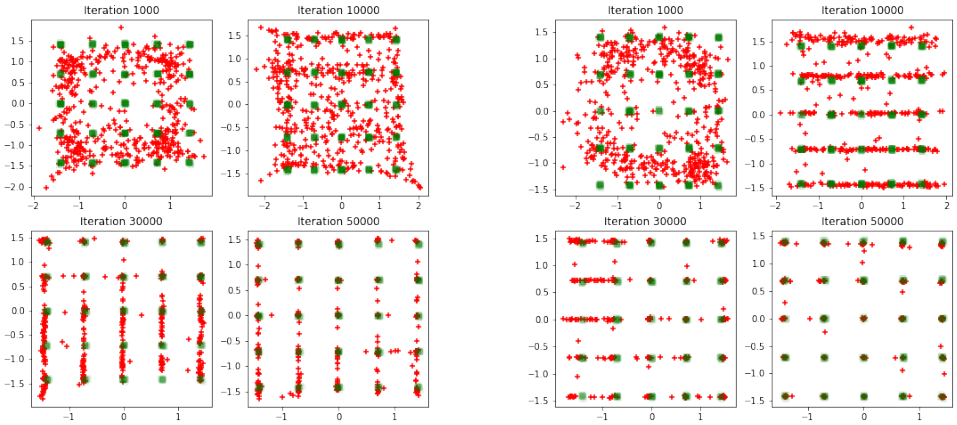
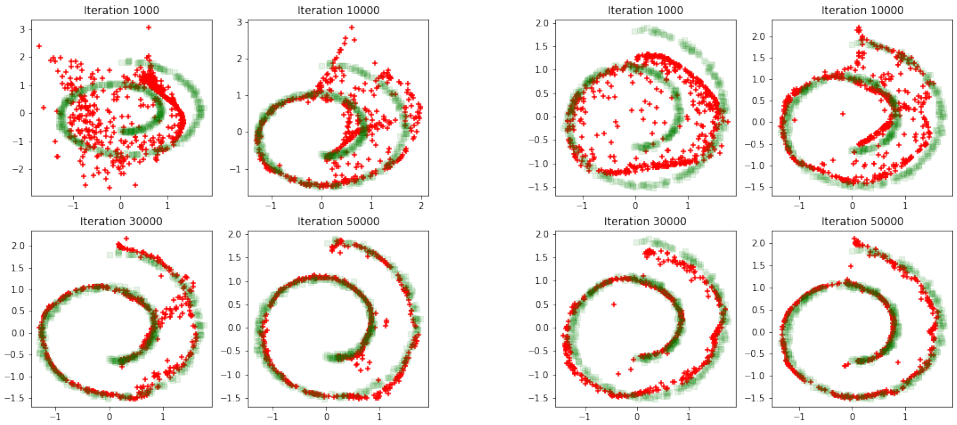


Figure 13: $f(x, y) = 2x^2 + y^2 + 4xy + \frac{4}{3}y^3 - \frac{1}{4}y^4$.



(a) Adam (b) GDA−AM

Figure 14: **25 Gaussians**: Evolution plot of Adam and GDA−AM . Green dots are observed points and red dots are generated points.

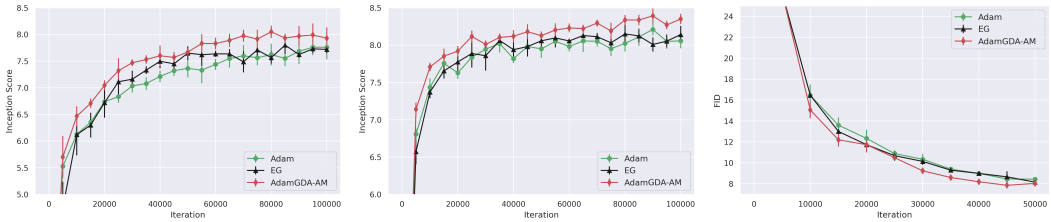


(a) Adam (b) GDA−AM

Figure 15: **Swiss roll**: Evolution plot of Adam and GDA−AM . Green dots are observed points and red dots are generated points.

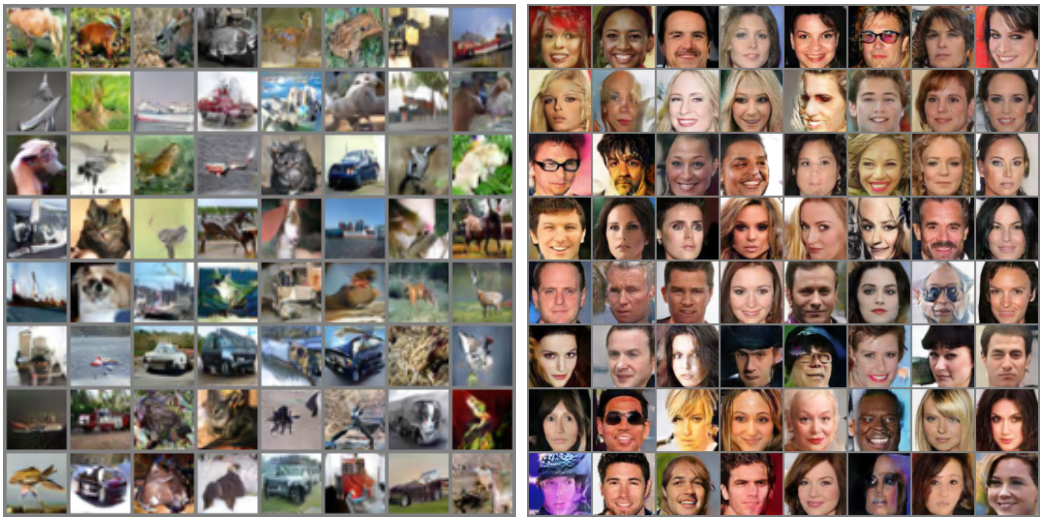
C.3 IMAGE GENERATION

In this section, we provide additional experimental results that are not given in Section 6. Figure 16a and 16b show the Inception Score for CIFAR10 using WGAN-GP and SNGAN. It can be observed that our method consistently performs better than Adam and EG during training. Further, on CIFAR-10 using WGAN-GP and SNGAN, `GDA-AM` is slightly slower than Adam (about 110-115% computational time), but significantly faster than EG (about 65-75% computational time).



(a) IS for CIFAR10 (b) IS for CIFAR10 (c) FID CelebA

Figure 16: **Left:** IS for CIFAR10 using WGANGP. **Middle:** IS for CIFAR10 using SNGAN. **Right:** FID for CelebA using WGANGP.



(a) Generated images for CIFAR10 (b) Generated images for CelebA

Figure 17: Generated Images for CIFAR10 and CelebA using WGAN-GP(ResNet)

C.4 DETAILS ON THE EXPERIMENTS

For our experiments, we used the PyTorch [1] deep learning framework. Experiments were run one NVIDIA V100 GPU. The residual network architecture for generator and discriminator are summarized in Table 2 and 3. We use a WGAN-GP loss, with gradient penalty $\lambda = 10$. When using the gradient penalty (WGAN-GP), we remove the batch normalization layers in the discriminator. When using SNGAN, we replace the batch normalization layers with spectral normalization. Hyperparamters of Adam are selected after grid search. We use a learning rate of 2×10^{-4} and batch size of 64. For table size of `GDA-AM` , we set it as 120 for CIFAR10 and 150 for CelebA. We set $\beta_1 = 0.0$ and $\beta_2 = 0.9$ as we find it gives us better models than default settings.

[1] https://pytorch.org/

Table 2: ResNet architecture used for our CIFAR-10 experiments.

Generator
Input: $z \in \mathbb{R}^{128} \sim \mathcal{N}(0, I)$
Linear $128 \rightarrow 256 \times 4 \times 4$
ResBlock $128 \rightarrow 128$
ResBlock $256 \rightarrow 256$
ResBlock $256 \rightarrow 256$
Batch Normalization
ReLu
transposed conv. (256, kernel:3×3, stride:1, pad: 1
$\tanh(\cdot)$

Discriminator
Input: $x \in \mathbb{R}^{3 \times 32 \times 32}$
Linear $128 \rightarrow 128 \times 4 \times 4$
ResBlock $128 \rightarrow 128$
ResBlock $128 \rightarrow 128$
ResBlock $128 \rightarrow 128$
Linear $128 \rightarrow 1$

Table 3: ResNet architecture used for our CelebA (64×64) experiments.

Generator
Input: $z \in \mathbb{R}^{128} \sim \mathcal{N}(0, I)$
Linear $128 \rightarrow 512 \times 8 \times 8$
ResBlock $512 \rightarrow 256$
ResBlock $256 \rightarrow 128$
ResBlock $128 \rightarrow 64$
Batch Normalization
ReLu
transposed conv. (64, kernel:3×3, stride:1, pad: 1
$\tanh(\cdot)$

Discriminator
Input: $x \in \mathbb{R}^{3 \times 64 \times 64}$
Linear $128 \rightarrow 128 \times 4 \times 4$
ResBlock $128 \rightarrow 128$
ResBlock $128 \rightarrow 256$
ResBlock $256 \rightarrow 512$
Linear $512 \rightarrow 1$