LEAD: Min-Max Optimization from a Physical Perspective

Anonymous authors
Paper under double-blind review

Abstract

Adversarial formulations such as generative adversarial networks (GANs) have rekindled interest in two-player min-max games. A central obstacle in the optimization of such games is the rotational dynamics that hinder their convergence. In this paper, we show that game optimization shares dynamic properties with particle systems subject to multiple forces, and one can leverage tools from physics to improve optimization dynamics. Inspired by the physical framework, we propose LEAD, an optimizer for min-max games. Next, using Lyapunov stability theory and spectral analysis, we study LEAD's convergence properties in continuous and discrete time settings for a class of quadratic min-max games to demonstrate linear convergence to the Nash equilibrium. Finally, we empirically evaluate our method on synthetic setups and CIFAR-10 image generation to demonstrate improvements in GAN training.

1 Introduction

Much of the advances in traditional machine learning can be attributed to the success of gradient-based methods. Modern machine learning systems such as GANs (Goodfellow et al., 2014), multi-task learning, and multi-agent settings (Sener & Koltun, 2018) in reinforcement learning (Bu et al., 2008) require joint optimization of two or more objectives which can often be formulated as games. In these *game* settings, best practices and methods developed for single-objective optimization are observed to perform noticeably poorly (Mescheder et al., 2017; Balduzzi et al., 2018b; Gidel et al., 2019). Specifically, they exhibit rotational dynamics in parameter space about the *Nash Equilibria* (Mescheder et al., 2017), slowing down convergence. Recent work in game optimization (Wang et al., 2019; Mazumdar et al., 2019; Mescheder et al., 2017; Balduzzi et al., 2018b; Abernethy et al., 2019; Loizou et al., 2020) demonstrates that introducing additional second-order terms in the optimization algorithm helps to suppress these rotations, thereby improving convergence.

Taking inspiration from recent work in single-objective optimization that re-derives existing accelerated methods from a variational perspective (Wibisono et al., 2016; Wilson et al., 2016), in this work, we adopt a similar approach in the context of games. To do so, we borrow formalism from physics by likening the gradient-based optimization of two-player (zero-sum) games to the dynamics of a system where we introduce relevant forces that helps curb these rotations. We consequently utilize the dynamics of this resultant system to propose our novel second-order optimizer for games, *LEAD*.

Next, using Lyapunov and spectral analysis, we demonstrate linear convergence of our optimizer (LEAD) in both continuous and discrete-time settings for a class of quadratic min-max games. In terms of empirical performance, LEAD achieves an FID of 10.49 on CIFAR-10 image generation, outperforming existing baselines such as BigGAN (Brock et al., 2018), which is approximately 30-times larger than our baseline ResNet architecture.

What distinguishes LEAD from other second-order optimization methods for min-max games such as Mescheder et al. (2017); Wang et al. (2019); Mazumdar et al. (2019); Schäfer & Anandkumar (2019) is its computational complexity. All these different methods involve Jacobian (or Jacobian-inverse) vector-product computation commonly implemented using a form of approximation. Thus making a majority of them intractable in real-world large scale problems. On the other hand, LEAD involves computing only *one-block* of the full Jacobian of the gradient vector-field multiplied by a vector. This makes our method significantly cheaper and comparable to several first-order methods, as we show in section 6.

We summarize our contributions below:

- In section 3, we model gradient descent-ascent as a physical system. Armed with the physical model, we introduce counter-rotational forces to curb the existing rotations in the system. Next, we employ the principle of least action to determine the (continuous-time) dynamics. We then accordingly discretize these resultant dynamics to obtain our optimization scheme, Least Action Dynamics (LEAD).

- In section 4, we use Lyapunov stability theory and spectral analysis to prove a linear convergence of LEAD in continuous and discrete-time settings for quadratic min-max games.

- Finally, in section 7, we empirically demonstrate that LEAD is computationally efficient. Additionally, we demonstrate that LEAD improves the performance of GANs on different tasks such as 8-Gaussians and CIFAR-10 while comparing the performance of our method against other first and second-order methods.

- The source code for all the experiments is available as a supplementary. Furthermore, we provide a blog post that summarizes our work, which is also available as part of the supplementary material.

2 Problem Setting

Notation Continuous time scalar variables are in uppercase letters (X), discrete-time scalar variables are in lower case (x) and vectors are in boldface ($\mathbf{A}$). Matrices are in blackboard bold ($\mathbb{M}$) and derivatives w.r.t. time are denoted as an over-dot ($\dot{x}$).

Setting In this work, we study the optimization problem of two-player zero-sum games,

$$\min_X \max_Y f\left(X, Y\right), \tag{1}$$

where $f : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$, and is assumed to be a convex-concave function which is continuous and twice differentiable w.r.t. $X, Y \in \mathbb{R}$. It is to be noted that though in developing our framework below, X, Y are assumed to be scalars, it is nevertheless found to hold for the more general case of vectorial X and Y, as we demonstrate both analytically (Appendix C) and empirically, our theoretical analysis is found to hold.

3 Optimization Mechanics

In our effort to study min-max optimization from a physical perspective, we note from classical physics the following: under the influence of a net force F, the equation of motion of a physical object of mass m, is determined by Newton's 2^{nd} Law,

$$m\ddot{X} = F, \tag{2}$$

with the object's coordinate expressed as $X_t \equiv X$. According to the *principle of least action*[1] (Landau & Lifshitz, 1960), nature "selects" this particular trajectory over other possibilities, as a quantity called the *action* is extremized along it.

We start with a simple observation that showcases the connection between optimization algorithms and physics. Polyak's heavy-ball momentum (Polyak, 1964) is often perceived from a physical perspective as a ball moving in a "potential" well (cost function). In fact, it is straightforward to show that Polyak momentum is a discrete counterpart of a continuous-time equation of motion governed by Newton's 2^{nd} Law. For single-objective minimization of an objective function $f\left(x\right)$, Polyak momentum follows:

$$x_{k+1} = x_k + \beta\left(x_k - x_{k-1}\right) - \eta\nabla_x f\left(x_k\right), \tag{3}$$

---

[1] Also referred to as the Principle of Stationary Action.

where η is the learning rate and β is the momentum coefficient. For simplicity, setting β to one, and moving to continuous time, one can rewrite this equation as,

$$\frac{(x_{k+\delta}-x_k)-(x_k-x_{k-\delta})}{\delta^2}=-\frac{\eta}{\delta^2}\nabla_x f\left(x_k\right), \tag{4}$$

and in the limit $\delta,\eta\to 0$, Eq.(4) then becomes ($x_k\to X\left(t\right)\equiv X$),

$$m\ddot{X}=-\nabla_X f\left(X\right). \tag{5}$$

This is equivalent to Newton's 2^{nd} Law of motion (Eq.(2)) of a particle of mass $m=\delta^2/\eta$, and identifying $F=-\nabla_X f\left(X\right)$ (i.e. $f\left(X\right)$ acting as a *potential* function (Landau & Lifshitz, 1960)). Thus, Polyak's heavy-ball method Eq.(3) can be interpreted as an object (ball) of mass m rolling down under a potential $f\left(X\right)$ to reach the minimum.

Armed with this observation, we perform an extension of Eq.equation 5 to our min-max setup,

$$\begin{aligned} m\ddot{X}&=-\nabla_X f\left(X,Y\right),\\ m\ddot{Y}&=\nabla_Y f\left(X,Y\right), \end{aligned} \tag{6}$$

which represents the dynamics of an object moving under a *curl force* (Berry & Shukla, 2016): $\boldsymbol{F}_{\text{curl}}=\left(-\nabla_X f,\nabla_Y f\right)$ in the 2-dimensional $X-Y$ plane. It is to be noted that discretization of Eq.(6) corresponds to Gradient Descent-Ascent (GDA) with momentum 1. Authors in (Gidel et al., 2019) found that this optimizer is divergent in the prototypical min-max objective, $f\left(X,Y\right)=XY$, thus indicating the need for further improvement.

To this end, we note that the failure modes of the optimizer obtained from the discretization of Eq.(6), can be attributed to: *(a)* an outward rotatory motion by our particle of mass m, accompanied by *(b)* an increase in its velocity with time. Following these observations, we aim to introduce suitable *counter-rotational* and *dissipative* forces to our system above, in order to tackle *(a)* and *(b)* in an attempt to achieve converging dynamics.

Specifically, as an initial consideration, we choose to add to our system, two ubiquitous forces:

- magnetic force,

$$\boldsymbol{F}_{\text{mag}}=\Big(-q\nabla_{XY}f\ \dot{Y},q\nabla_{XY}f\ \dot{X}\Big) \tag{7}$$

  known to produce rotational motion (in charged particles), to counteract the rotations introduced by $\boldsymbol{F}_{\text{curl}}$. Here, q is the charge imparted to our particle.

- friction,

$$\boldsymbol{F}_{\text{fric}}=\left(\mu\dot{X},\mu\dot{Y}\right) \tag{8}$$

  to prevent the increase in velocity of our particle (μ: coefficient of friction).

Assimilating all the above forces $\boldsymbol{F}_{\text{curl}}$, $\boldsymbol{F}_{\text{mag}}$ and $\boldsymbol{F}_{\text{fric}}$, the equations of motion (EOMs) of our crafted system then becomes,

$$\begin{aligned} m\ddot{X}&=\boldsymbol{F}_{\text{curl}}+\boldsymbol{F}_{\text{mag}}+\boldsymbol{F}_{\text{fric}},\\ m\ddot{Y}&=\boldsymbol{F}_{\text{curl}}+\boldsymbol{F}_{\text{mag}}+\boldsymbol{F}_{\text{fric}}. \end{aligned} \tag{9}$$

Or equivalently,

$$\begin{aligned} m\ddot{X}&=-\mu\dot{X}-\nabla_X f-q\nabla_{XY}f\dot{Y},\\ m\ddot{Y}&=-\mu\dot{Y}+\nabla_Y f+q\nabla_{XY}f\dot{X}. \end{aligned} \tag{10}$$

Without loss of generality, from hereon we set the mass of our object to be unity. In the rest of this work, we study the above EOMs in continuous and discrete time for min-max games.

3.1 Discretization

With the continuous-time trajectory of Eq.(10) in hand, we now proceed to discretize it using a combination of Euler's implicit and explicit discretization schemes. To discretize $\dot{X} = V_X$ we have,

$$\begin{aligned} \text{Euler Implicit Discretization}: \ & x_{k+1} - x_k = \delta v^x_{k+1} \\ \text{Euler Explicit Discretization}: \ & x_{k+1} - x_k = \delta v^x_k. \end{aligned} \tag{11}$$

where δ is the discretization step-size and k is the iteration step.

Proposition 1. *The continuous-time EOMs (10) can be discretized in an implicit-explicit way, to yield,*

$$\begin{aligned} x_{k+1} &= x_k + \beta(x_k - x_{k-1}) - \eta \nabla_x f(x_k, y_k) \\ &\quad - \alpha \nabla_{xy} f(x_k, y_k)(y_k - y_{k-1}), \\ y_{k+1} &= y_k + \beta(y_k - y_{k-1}) + \eta \nabla_y f(x_k, y_k) \\ &\quad + \alpha \nabla_{yx} f(x_k, y_k)(x_k - x_{k-1}), \end{aligned} \tag{12}$$

where we have defined $\alpha = 2q\delta, \beta = 1 - \mu\delta$ and $\eta = \delta^2$ (Proof in Appendix B).
Taking inspiration from the fact that Eq. equation 10 corresponds to the trajectory of a charged particle under a curl, magnetic and frictional force, as governed by the principle of least action, we refer to the discrete update rules of Eq. equation 12 as *Least Action Dynamics (LEAD)*. (Algorithm 1 details the pseudo-code of LEAD).

Understanding the terms in LEAD: Analyzing our novel optimizer, we note that it consist of three types of terms, namely,

1. **Gradient Descent or Ascent**: $-\nabla_x f$ or $\nabla_y f$: Each player's immediate direction of improving their own objective.

2. **Momentum**: Standard Polyak momentum term; known to accelerate convergence in optimization and recently in smooth games. (Gidel et al., 2019; Azizian et al., 2020b; Lorraine & Duvenaud, 2022)

3. **Coupling term**:

$$-\nabla_{xy} f(x_k, y_k)(y_k - y_{k-1}), \nabla_{yx} f(x_k, y_k)(x_k - x_{k-1})$$

   Main new term in our method. It captures the first-order interaction between players. This cross-derivative corresponds to the counter-rotational force in our physical model; it allows our method to exert control on rotations.

Algorithm 1 Least Action Dynamics (LEAD)

```
Input: learning rate $\eta$, momentum $\beta$, coupling coefficient $\alpha$.
Initialize: $x_0 \leftarrow x_{init}$, $y_0 \leftarrow y_{init}$, $t \leftarrow 0$
while not converged do
    $t \leftarrow t + 1$
    $g_x \leftarrow \nabla_x f(x_t, y_t)$
    $g_{xy}\Delta y_t \leftarrow \nabla_y(g_x)(y_t - y_{t-1})$
    $x_{t+1} \leftarrow x_t + \beta(x_t - x_{t-1}) - \eta g_x - \alpha g_{xy}\Delta y_t$
    $g_y \leftarrow \nabla_y f(x_t, y_t)$
    $g_{xy}\Delta x_t \leftarrow \nabla_x(g_y)(x_t - x_{t-1})$
    $y_{t+1} \leftarrow y_t + \beta(y_t - y_{t-1}) + \eta g_y + \alpha g_{xy}\Delta x_t$
end while
return $(x_{k+1}, y_{k+1})$
```

4 Convergence Analysis

We now study the behavior of LEAD on the quadratic min-max game,

$$f\left(\mathbf{X}, \mathbf{Y}\right) = \frac{h}{2}||\mathbf{X}||^2 - \frac{h}{2}||\mathbf{Y}||^2 + \mathbf{X}^T \mathbb{A} \mathbf{Y} \tag{13}$$

where $\mathbf{X}, \mathbf{Y} \in \mathbb{R}^n$, $\mathbb{A} \in \mathbb{R}^n \times \mathbb{R}^n$ is a (constant) coupling matrix and h is a scalar constant. Additionally, the Nash equilibrium of the above game lies at $\mathbf{X}^* = 0, \mathbf{Y}^* = 0$. Let us further define the *vector field* $\mathbf{v}$ of the above game, f, as,

$$\boldsymbol{v} = \begin{bmatrix} \nabla_{\mathbf{X}} f\left(\mathbf{X}, \mathbf{Y}\right) \\ -\nabla_{\mathbf{Y}} f\left(\mathbf{X}, \mathbf{Y}\right) \end{bmatrix} = \begin{bmatrix} h\boldsymbol{X} + \mathbb{A}\mathbf{Y} \\ h\boldsymbol{Y} - \mathbb{A}^\top \mathbf{X} \end{bmatrix}. \tag{14}$$

4.1 Continuous Time Analysis

A general way to prove the stability of a dynamical system is to use a Lyapunov function (Hahn et al., 1963; Lyapunov, 1992). A scalar function $\mathcal{E}_t : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$, is a Lyapunov function of a continuous-time dynamics if $\forall\ t$,

$$\begin{aligned} &(i)\ \ \mathcal{E}_t(\mathbf{X}, \mathbf{Y}) \geq 0, \\ &(ii)\ \ \dot{\mathcal{E}}_t(\mathbf{X}, \mathbf{Y}) \leq 0 \end{aligned}$$

The Lyapunov function $\mathcal{E}_t$ can be perceived as a generalization of the total energy of the system and the requirement (ii) ensures that this generalized energy decreases along the trajectory of evolution, leading the system to convergence as we will show next.

For the quadratic min-max game defined in Eq.(13), Eq.(10) generalizes to,

$$\begin{aligned} \ddot{\boldsymbol{X}} &= -\mu\dot{\boldsymbol{X}} - (h + \mathbb{A})\boldsymbol{Y} - q\mathbb{A}\dot{\boldsymbol{Y}} \\ \ddot{\boldsymbol{Y}} &= -\mu\dot{\boldsymbol{Y}} - (h - \mathbb{A}^T)\boldsymbol{X} + q\mathbb{A}^T\dot{\boldsymbol{X}}, \end{aligned} \tag{15}$$

Theorem 1. *For the dynamics of Eq.(15),*

$$\begin{aligned} \mathcal{E}_t = &\frac{1}{2}\left(\dot{\boldsymbol{X}} + \mu\boldsymbol{X} + \mu\mathbb{A}\boldsymbol{Y}\right)^T\left(\dot{\boldsymbol{X}} + \mu\boldsymbol{X} + \mu\mathbb{A}\boldsymbol{Y}\right) \\ &+ \frac{1}{2}\left(\dot{\boldsymbol{Y}} + \mu\boldsymbol{Y} - \mu\mathbb{A}^T\boldsymbol{X}\right)^T\left(\dot{\boldsymbol{X}} + \mu\boldsymbol{Y} - \mu\mathbb{A}^T\boldsymbol{X}\right) \\ &+ \frac{1}{2}\left(\dot{\boldsymbol{X}}^T\dot{\boldsymbol{X}} + \dot{\boldsymbol{Y}}^T\dot{\boldsymbol{Y}}\right) + \boldsymbol{X}^T(h + \mathbb{A}\mathbb{A}^T)\boldsymbol{X} + \boldsymbol{y}^T(h + \mathbb{A}^T\mathbb{A})\boldsymbol{Y} \end{aligned} \tag{16}$$

is a Lyapunov function of the system.

Furthermore, setting $q = (2/\mu) + \mu$, we find $\dot{\mathcal{E}}_t \leq -\rho\mathcal{E}_t$ for

$$\rho \leq \min\left\{\frac{\mu}{1+\mu},\ \frac{2\mu(\sigma_{\min}^2 + h)}{(1 + \sigma_{\min}^2 + 2h)\left(\mu^2 + \mu\right) + 2\sigma_{\min}^2}\right\}$$

with $\sigma_{\min}$ being the smallest singular value of $\mathbb{A}$. This consequently ensures linear convergence of the dynamics of Eq. equation 15,

$$\boxed{||\boldsymbol{X}||^2 + ||\boldsymbol{Y}||^2 \leq \frac{\mathcal{E}_0}{h + \sigma_{min}^2}\exp\left(-\rho t\right)}. \tag{17}$$

(Proof in Appendix C).

4.2 Discrete-Time Analysis

In this section, we next analyze the convergence behavior of LEAD, Eq.(12) in the case of the quadratic min-max game of Eq.(13), using spectral analysis,

$$\begin{aligned}\boldsymbol{x}_{k+1} &= \boldsymbol{x}_k + \beta \Delta \boldsymbol{x}_k - \eta h \boldsymbol{x}_k - \eta \mathbb{A} \boldsymbol{y}_k - \alpha \mathbb{A} \Delta \boldsymbol{y}_k \\ \boldsymbol{y}_{k+1} &= \boldsymbol{y}_k + \beta \Delta \boldsymbol{y}_k - \eta h \boldsymbol{y}_k + \eta \mathbb{A}^T \boldsymbol{x}_k + \alpha \mathbb{A}^T \Delta \boldsymbol{x}_k,\end{aligned} \tag{18}$$

where $\Delta \boldsymbol{x}_k = \boldsymbol{x}_k - \boldsymbol{x}_{k-1}$.

For brevity, consider the joint parameters $\boldsymbol{\omega}_t := (\boldsymbol{x}_t, \boldsymbol{y}_t)$. We start by studying the update operator of simultaneous gradient descent-ascent,

$$F_\eta(\boldsymbol{\omega}_t) = \boldsymbol{\omega}_t - \eta \boldsymbol{v}(\boldsymbol{\omega}_{t-1}).$$

where, the vector-field is given by Eq. equation 14. Thus, the fixed point $\boldsymbol{\omega}^*$ of $F_\eta(\boldsymbol{\omega}_t)$ satisfies $F_\eta(\boldsymbol{\omega}^*) = \boldsymbol{\omega}^*$. Furthermore, at $\boldsymbol{\omega}^*$, we have,

$$\nabla F_\eta(\boldsymbol{\omega}^*) = \mathbb{I}_n - \eta \nabla \boldsymbol{v}(\boldsymbol{\omega}^*), \tag{19}$$

with $\mathbb{I}_n$ being the $n \times n$ identity matrix. Consequently the spectrum of $\nabla F_\eta(\boldsymbol{\omega}^*)$ in the quadratic game considered, is,

$$\mathrm{Sp}(\nabla F_\eta(\boldsymbol{\omega}^*)) = \{1 - \eta h - \eta \lambda \mid \lambda \in \mathrm{Sp}(\text{off-diag}[\nabla \boldsymbol{v}(\boldsymbol{\omega}^*)])\}\,. \tag{20}$$

The next proposition outlines the condition under which the fixed point operator is guaranteed to converge around the fixed point.

Proposition 2 (Prop. 4.4.1 (Bertsekas, 1999)). *For the spectral radius,*

$$\rho_{max} := \rho\{\nabla F_\eta(\boldsymbol{\omega}^*)\} < 1 \tag{21}$$

and for some $\boldsymbol{\omega}_0$ in a neighborhood of $\boldsymbol{\omega}^*$, the update operator F, ensures linear convergence to $\boldsymbol{\omega}^*$ at a rate,

$$\Delta_{t+1} \leq \mathcal{O}(\rho + \epsilon)\Delta_t \ \forall \ \epsilon > 0,$$

where $\Delta_{t+1} := ||\boldsymbol{\omega}_{t+1} - \boldsymbol{\omega}^*||_2^2 + ||\boldsymbol{\omega}_t - \boldsymbol{\omega}^*||_2^2$.

Next, we proceed to define the update operator of Eq.(12) as $F_{\text{LEAD}}(\boldsymbol{\omega}_t, \boldsymbol{\omega}_{t-1}) = (\boldsymbol{\omega}_{t+1}, \boldsymbol{\omega}_t)$. For the quadratic min-max game of Eq.(13), the Jacobian of F_{LEAD} takes the form,

$$\nabla F_{\text{LEAD}} = \begin{bmatrix} \mathbb{I}_{2n} + \beta \mathbb{I}_{2n} - (\eta + \alpha) \nabla \boldsymbol{v} & -\beta \mathbb{I}_{2n} + \alpha \nabla \boldsymbol{v} \\ \mathbb{I}_{2n} & 0 \end{bmatrix}. \tag{22}$$

In the next Theorem 2, we find the set of eigenvalues corresponding to the update operator ∇F_{LEAD} which are then used in Theorem 3, where we show for a selected values of η and α, LEAD attains a linear rate.

Theorem 2. *The eigenvalues of $\nabla F_{LEAD}(\boldsymbol{\omega}^*)$ are,*

$$\mu_\pm = \frac{1 - (\eta + \alpha)\lambda + \beta - \eta h \pm \sqrt{\Delta}}{2} \tag{23}$$

where,

$$\Delta = (1 - (\eta + \alpha) \lambda + \beta - \eta h)^2 - 4 (\beta - \alpha \lambda)$$

and $\lambda \in Sp(\text{off-diag}[\nabla \boldsymbol{v}(\boldsymbol{\omega}^*)])$.

Furthermore, for $h, \eta, |\alpha|, |\beta| << 1$, we have,

$$\begin{aligned}\mu_+ \approx & 1 - \eta h + \frac{(\eta + \alpha)^2 \lambda^2 + \eta^2 h^2 + \beta^2 - 2\eta h \beta}{4} \\ & + \lambda \left(\frac{\eta + \alpha}{2} (\eta h - \beta) - \eta \right)\end{aligned} \tag{24}$$

and

$$\begin{aligned}\mu_- \approx &\beta - \frac{(\eta+\alpha)^2\lambda^2 + \eta^2 h^2 + \beta^2 - 2\eta h\beta}{4} \\ &+ \lambda\left(\frac{\eta+\alpha}{2}(\beta - \eta h) - \alpha\right)\end{aligned} \tag{25}$$

See proof in Appendix D.

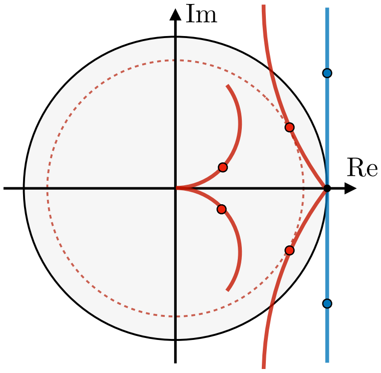


Figure 1: Diagram depicts positioning of the eigenvalues of GDA in blue (Eq. equation 19) and those of LEAD (eqs. (24) and (25) with $\beta = h = 0$) in red. Eigenvalues inside the black unit circle imply convergence such that the closer to the origin, the faster the convergence rate (Prop. 2). Every point on solid blue and red lines corresponds to a specific choice of learning rate. No choice of learning rate results in convergence for gradient ascent descent method as the blue line is tangent to the unit circle. At the same time, for a fixed value of α, LEAD shifts the eigenvalues (μ_+) into the unit circle which leads to a convergence rate proportional to the radius of the red dashed circle. Note that LEAD also introduces an extra set of eigenvalues (μ_-) which are close to zero and do not affect convergence .

In the following Proposition, we next show that locally, a choice of positive α decreases the spectral radius of $\nabla F_\eta(\boldsymbol{\omega}^*)$ defined as,

$$\rho := \max\{|\mu_+|^2, |\mu_-|^2\} \ \forall\ \lambda.$$

Proposition 3. *For any $\lambda \in Sp(\text{off-diag}[\nabla \boldsymbol{v}(\omega^*)])$,*

$$\nabla_\alpha \rho(\lambda)\big|_{\alpha=0} < 0 \Leftrightarrow \eta \in \left(0, \frac{2}{\mathrm{Im}(\lambda_{max})}\right), \tag{26}$$

where $\mathrm{Im}(\lambda_{max})$ is the imaginary component of the largest eigenvalue λ_{max}.

See proof in Appendix E.

Having established that a small positive value of α improves the rate of convergence, in the next theorem, we prove that for a specific choice of positive α and η in the quadratic game Eq.(13), a linear rate of convergence to its Nash equilibrium is attained.

Theorem 3. *Setting $\eta = \alpha = \frac{2}{\sigma_{max}(\mathbb{A})}$, then we have $\forall\ \epsilon > 0$,*

$$\Delta_{t+1} \in \mathcal{O}\left(\left(1 - 6\frac{2\sigma_{\min}^2 + h^2}{\sigma_{\max}^2} - 2h\frac{2+\beta}{\sigma_{\max}} + \frac{\beta^2}{2}\right)^t \Delta_0\right) \tag{27}$$

where $\sigma_{max}(\sigma_{min})$ is the largest (smallest) singular value of $\mathbb{A}$ and

$$\Delta_{t+1} := ||\boldsymbol{\omega}_{t+1} - \boldsymbol{\omega}^*||_2^2 + ||\boldsymbol{\omega}_t - \boldsymbol{\omega}^*||_2^2.$$

Theorem 3 ensures a linear convergence of LEAD in the quadratic min-max game. (Proof in Appendix F).

5 Comparison of Convergence Rate

In this section, we perform a Big-O comparison of rates of convergence of LEAD (Eq. equation 27), with Extragradient (Korpelevich, 1976) in the quadratic min-max game of Eq. equation 13, with $\beta = 0$. Specifically, from (Azizian et al., 2020a) we find,

$$\begin{aligned} r_{\mathrm{EG}} &= \mathcal{O}\left(\frac{h^2}{L^2}\right) + \mathcal{O}\left(\frac{\sigma_{\min}^2(\mathbb{A})}{L^2}\right) + \mathcal{O}\left(\frac{h}{L}\right) \\ r_{\mathrm{LEAD}} &= \mathcal{O}\left(\frac{h^2}{\sigma_{\max}^2(\mathbb{A})}\right) + \mathcal{O}\left(\frac{\sigma_{\min}^2(\mathbb{A})}{\sigma_{\max}^2(\mathbb{A})}\right) - \mathcal{O}\left(\frac{h}{\sigma_{\max}(\mathbb{A})}\right) \end{aligned} \tag{28}$$

where, $L := \max\{h, \sigma_{\max}(\mathbb{A})\}$.

Therefore, for $h < \sigma_{\max}(\mathbb{A})$, we observe that $r_{\mathrm{LEAD}} \lesssim r_{\mathrm{EG}}$. While for $h > \sigma_{\max}(\mathbb{A})$, we note that,

$$\begin{aligned} r_{\mathrm{EG}} &= \mathcal{O}(1) + \mathcal{O}\left(\frac{\sigma_{\min}^2(\mathbb{A})}{h^2}\right) \\ r_{\mathrm{LEAD}} &= \mathcal{O}\left(\frac{h^2}{\sigma_{\max}^2(\mathbb{A})}\right) + \mathcal{O}\left(\frac{\sigma_{\min}^2(\mathbb{A})}{\sigma_{\max}^2(\mathbb{A})}\right) - \mathcal{O}\left(\frac{h}{\sigma_{\max}(\mathbb{A})}\right) \end{aligned} \tag{29}$$

Hence, for $h \gtrsim 1.62\sigma_{\max}$, we find $r_{\mathrm{LEAD}} \gtrsim r_{\mathrm{EG}}$.

6 Comparison of Computational Cost

In this section we first study several second-order algorithms and perform computational comparisons on an 8-Gaussians generation task.
The Jacobian of the gradient vector field $\boldsymbol{v} = (\nabla_x f(\boldsymbol{x}, \boldsymbol{y}), -\nabla_y f(\boldsymbol{x}, \boldsymbol{y}))$ is given by,

$$\mathbb{J} = \begin{bmatrix} \nabla_x^2 f(\boldsymbol{x}, \boldsymbol{y}) & \nabla_{xy} f(\boldsymbol{x}, \boldsymbol{y}) \\ -\nabla_{yx} f(\boldsymbol{x}, \boldsymbol{y}) & -\nabla_y^2 f(\boldsymbol{x}, \boldsymbol{y}) \end{bmatrix}. \tag{30}$$

Considering player $\boldsymbol{x}$, a LEAD update requires the computation of the term $\nabla_{xy} f(\boldsymbol{x}_k, \boldsymbol{y}_k)(\boldsymbol{y}_k - \boldsymbol{y}_{k-1})$, thereby involving only one block of the full Jacobian $\mathbb{J}$. On the other hand Symplectic Gradient Adjustment (SGA) (Balduzzi et al., 2018a), requires the full computation of two Jacobian-vector products $\mathbb{J}\boldsymbol{v}, \mathbb{J}^\top \boldsymbol{v}$. Similarly, Competitive Gradient Descent (CGD) (Schäfer & Anandkumar, 2019) involves the computation of the following term,

$$\left(1 + \eta \nabla_{xy}^2 f(\boldsymbol{x}_k, \boldsymbol{y}_k) \nabla_{yx}^2 f(\boldsymbol{x}_k, \boldsymbol{y}_k)\right)^{-1}$$

along with the Jacobian-vector product,

$$\nabla_{xy}^2 f(\boldsymbol{x}_k, \boldsymbol{y}_k) \nabla_y f(\boldsymbol{x}_k, \boldsymbol{y}_k).$$

While the inverse term is approximated using conjugate gradient method, it still involves the computation of approximately ten Jacobian-vector products for each update. To explore these comparisons in greater detail and on models with many parameters, we experimentally compare the computational cost of our method with several other second as well as first-order methods on the 8-Gaussians problem in Figure 2 (architecture reported in Appendix I). We calculate the average wall-clock time (in milliseconds) per iteration. Results are reported on an average of 1000 iterations, computed on the same architecture and the same machine with forced synchronous execution. All the methods are implemented in PyTorch Paszke et al. (2017) and SGA is replicated based on the official implementation [2].

---

[2] SGA official DeepMind implementation (non-zero sum setting): `https://github.com/deepmind/symplectic-gradient-adjustment/blob/master/Symplectic_Gradient_Adjustment.ipynb`

Furthermore, we observe that the computational cost per iteration of LEAD while being much lower than SGA and CGD, is similar to WGAN-GP and Extra-Gradient. The similarity to Extra-Gradient is due to the fact that for each player, Extra-Gradient requires the computation of a half-step and a full-step, so in total each step requires the computation of two gradients. LEAD also requires the computation of a gradient (∇f_x) which is then used to compute (∇f_{xy}) multiplied by ($\boldsymbol{y}_k - \boldsymbol{y}_{k-1}$). Using PyTorch, we do not require to compute ∇f_{xy} and then perform the multiplication. Given ∇f_x the whole term $\nabla f_{xy}(\boldsymbol{y}_k - \boldsymbol{y}_{k-1})$, is computed using PyTorch's Autograd Vector-Jacobian product, with the computational cost of a single gradient. Thus, LEAD also requires the computation of two gradients for each step.

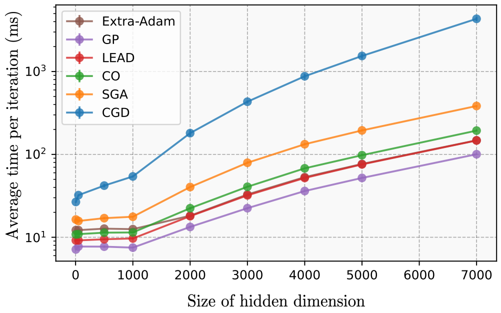


Figure 2: Average computational cost per iteration of several well-known methods for (non-saturating) GAN optimization. The numbers are reported on the 8-Gaussians generation task and averaged over 1000 iterations. Note that the y-axis is log-scale. We compare Competitive Gradient Descent (CGD) (52) (using official CGD optimizer code), Symplectic Gradient Adjustment (SGA) (6), Consensus Optimization (CO) (39), Extra-gradient with Adam (Extra-Adam) (17), WGAN with Gradient Penalty (WGAN GP) (21). We observe that per-iteration time complexity of our method is very similar to Extra-Adam and WGAN GP and is much cheaper than other second order methods such as CGD. Furthermore, by increasing the size of the hidden dimension of the generator and discriminator's networks we observe that the gap between different methods increases.

7 Experiments

In this section, we first empirically validate the performance of LEAD on several toy as well as large-scale experiments. Furthermore, we extend LEAD based on the Adam algorithm to be used in large-scale experiments. See 2 for the detailed Algorithm.

7.1 Adversarial vs Cooperative Games

In Section 6 we showed that using Auto-grad software tools such as TensorFlow and PyTorch, LEAD can be computed very efficiently and as fast as extra-gradient. In this section we compare the perforamce of LEAD with several first order methods in a toy setup inspired by (Lorraine & Duvenaud, 2022). Consider the following game,

$$\min_{\boldsymbol{x}} \max_{\boldsymbol{y}} \boldsymbol{x}^T(\boldsymbol{\gamma}\boldsymbol{A})\boldsymbol{y} + \boldsymbol{x}^T((\mathbf{I}-\boldsymbol{\gamma})\boldsymbol{B}_1)\boldsymbol{x} - \boldsymbol{y}^T((\mathbf{I}-\boldsymbol{\gamma})\boldsymbol{B}_2)\boldsymbol{y}. \tag{31}$$

Such formulation in Eq. 31 enables us to compare the performance of different methods in cooperative games, adversarial games, and any interpolation between the two. Namely, varying γ from $\mathbf{0}$ to $\mathbf{I}$ changes the dynamics of the game from purely cooperative to adversarial. Many real-world applications such as GANs exhibit an analogous range of adversarial to cooperative changes during training (Lorraine & Duvenaud, 2022).

In Figure 3, we compare LEAD against several methods including gradient descent-ascent (GDA), extragradient (EG) (Korpelevich, 1976), optimistic gradient (OG) (Mertikopoulos et al., 2018), complex momentum (CM) (Lorraine & Duvenaud, 2022), negative momentum (NM) (Gidel et al., 2019), and positive momentum (PM) (Polyak, 1964). Each method is tuned to optimality for each setup.

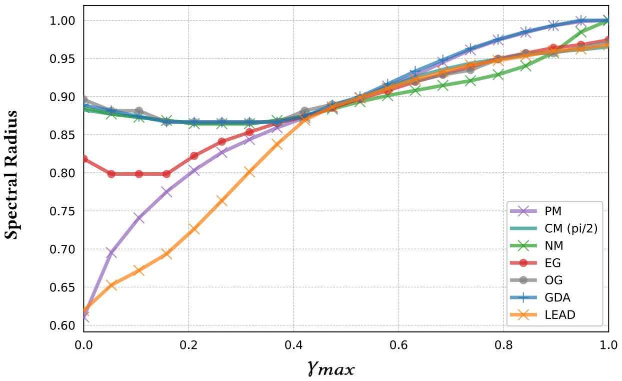


Figure 3: Comparison of several methods on the game in Eq. 31. The diagonal matrix γ determines the degree of adversarialness along each direction. Elements on the diagonal are sampled from a uniform distribution,$\gamma_{ii} \sim \text{Unif}[0, \gamma_{max}]$. By varying γ_{max} from 0 to 1, we move from a purely cooperative setup to a hybrid setup with a mixture of cooperative and adversarial games. The spectral radius (shown on the y-axis) determines the convergence rate in this game and is a function of γ_{max}. The smaller the spectral radius, the faster the convergence rate. A spectral radius of 1 corresponds to non-convergent dynamics. We compare several methods including gradient descent-ascent (GDA), extragradient (EG) (Korpelevich, 1976), optimistic gradient (OG) (Mertikopoulos et al., 2018), complex momentum (CM) (Lorraine & Duvenaud, 2022), negative momentum (NM) (Gidel et al., 2019), and positive momentum (PM) (Polyak, 1964). Each method has been tuned to optimality for each setup. For cooperative games (on the leftmost), LEAD and Positive Momentum achieve great performance. In more adversarial settings (on the rightmost), LEAD performs on par with other game-specific optimization methods (excluding Negative Momentum, GDA and Positive Momentum which diverge). This plot suggests that LEAD is a robust optimizer across different types of games. We conjecture that for the same reason, LEAD performs desirably in real-world setups such as GANs where the adversarialness changes dynamically throughout the training (Lorraine & Duvenaud, 2022).

7.2 Generative Adversarial Networks

We study the performance of LEAD in zero-sum as well as non-zero sum settings. See H for a comparison of LEAD-Adam against vanilla Adam on the generation task of a mixture of 8-Gaussians.

CIFAR-10 DCGAN: We evaluate LEAD-Adam on the task of CIFAR-10 (Krizhevsky & Hinton, 2009) image generation with a non-zero-sum formulation (non-saturating) on a DCGAN architecture similar to Gulrajani et al. (2017). As shown in Table. 1, we compare with several first-order and second order methods and observe that LEAD-Adam outperforms the rest in terms of Fréchet Inception Distance (FID) (Heusel et al., 2017) and Inception score (IS) (Salimans et al., 2016) [3], reaching an FID score of 19.27±0.10 which outperforms OMD (Mertikopoulos et al., 2018) and CGD (Schäfer & Anandkumar, 2019). See Figure 4 that shows the improvement of FID using LEAD-Adam vs vanilla Adam and Figure 7 for a sample of the generated images.

CIFAR-10 ResNet: Furthermore, we evaluate LEAD-Adam on more complex and deep architectures. We adapt the ResNet architecture in SN-GAN Miyato et al. (2018). We compare with several existing results on the task of image generation on CIFAR-10 using ResNets. See Table 1 for a full comparison. Note that state of the art performance in recent work such as Style-GAN based models (Sauer et al., 2022; Kang et al., 2021; Lee et al., 2021) or BigGAN based models (Brock et al., 2018; Lorraine & Duvenaud, 2022) use architectures that are 30 times or more larger than the architecture that we have chosen to test our method on.

We report our results against a properly tuned version of SNGAN that achieves an FID of 12.36. Our method obtains a competitive FID of 10.49. We give a detailed description of these experiments and full detail on the architecture and hyper-parameters in Appendix I. See also Figure 8 for a sample of generated samples on a ResNet using LEAD-Adam.

---

[3] The FID and IS are metrics for evaluating the quality of generated samples of a generative model. Lower FID and higher inception score (IS) correspond to better sample quality.

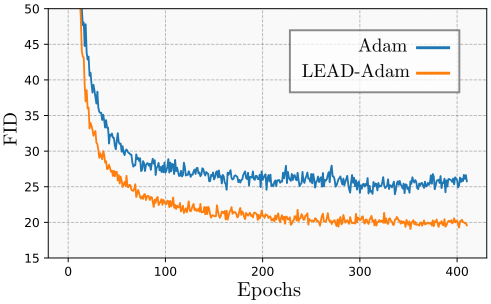


Figure 4: Plot showing the evolution of the FID over 400 epochs for our method (LEAD-Adam) vs vanilla Adam on a DCGAN architecture.

Table 1: Performance of several methods on CIFAR-10 image generation task. The FID and IS is reported over 50k samples unless mentioned otherwise.

DCGAN	**FID** (↓)	**IS** (↑)
Adam (Radford et al., 2015)	24.38 ± 0.13	6.58
LEAD-Adam	19.27 ± 0.10	7.58± 0.11
CGD-WGAN (Schäfer & Anandkumar, 2019)	21.3	7.2
OMD (Daskalakis et al., 2018)	29.6 ± 0.19	5.74 ± 0.1
ResNet		
SNGAN	12.10 ± 0.31	8.58 ± 0.03
LEAD-Adam (ours)	**10.49** ± 0.11	**8.82** ± 0.05
ExtraAdam (Gidel et al., 2018)	16.78 ± 0.21	8.47 ± 0.1
LA-GAN (Chavdarova et al., 2020)	12.67 ± 0.57	8.55 ± 0.04
ODE-GAN (Qin et al., 2020)	11.85 ± 0.21	8.61 ± 0.06
Evaluated with 5k samples		
SN-GAN (DCGAN) (Miyato et al., 2018)	29.3	7.42 ± 0.08
SN-GAN (ResNet) (Miyato et al., 2018)	21.7 ± 0.21	8.22 ± 0.05

8 Related Work

Game Optimization: With increasing interest in games, significant effort is being spent in understanding common issues affecting optimization in this domain. These issues range from convergence to non-Nash equilibrium points, to exhibiting rotational dynamics around the equilibrium which hampers convergence. Authors in Mescheder et al. (2017) discuss how the eigenvalues of the Jacobian govern the local convergence properties of GANs. They argue that the presence of eigenvalues with zero real-part and large imaginary part results in oscillatory behavior. To mitigate this issue, they propose Consensus Optimization (CO). Along similar lines, Balduzzi et al. (2018b); Gemp & Mahadevan (2018); Letcher et al. (2019); Loizou et al. (2020) use the *Hamiltonian* of the gradient vector-field, to improve the convergence in games through disentangling the convergent parts of the dynamics from the rotations. Another line of attack taken in Schäfer & Anandkumar (2019) is to use second-order information as a regularizer of the dynamics and motivate the use of Competitive Gradient Descent (CGD). In Wang et al. (2019), Follow the Ridge (FtR) is proposed. They motivate the use of a second order term for one of the players (follower) as to avoid the rotational dynamics in a sequential formulation of the zero-sum game. See appendix J for full discussion on the comparison of LEAD versus other second-order methods.

Another approach taken by Gidel et al. (2019), demonstrate how applying negative momentum over GDA can improve convergence in min-max games, while also proving a linear rate of convergence in the case of bilinear games. More recently, Zhang & Wang (2021) have shown the suboptimality of negative momentum in specific settings. Furthermore, in (Lorraine & Duvenaud, 2022) authors carry-out an extensive study on the effect of momentum in games and specifically show that complex momentum is optimal in many games ranging from adversarial to non-adversarial settings. Daskalakis et al. (2018) show that extrapolating the next value of the gradient using previous history, aids convergence. In the same spirit, Chavdarova et al. (2020), proposes LookAhead GAN (LA-GAN) and show that the LookAhead algorithm is a compelling candidate in improving convergence in GANs. Gidel et al. (2018) also explores this line of thought by introducing averaging to develop a variant of the extra-gradient algorithm and proposes Extra-Adam-Averaging. Similar to Extra-Adam-Averaging is SN-EMA Yazıcı et al. (2019) which uses the SN-GAN and achieves great performance by applying an exponential moving average on the parameters. More recently, Fiez & Ratliff (2021) study using different time-scales for each player in zero-sum non-convex, non-concave games.

Lastly, in regard to convergence analysis in games, Zhang et al. (2022) study the convergence of alternating gradient descent-ascent for minmax games, Golowich et al. (2020) provide last iterate convergence rate for convex-concave saddle point problems. Nouiehed et al. (2019) propose a multi-step variant of gradient descent-ascent, to show it can find a game's ϵ–first-order stationary point. Additionally, Azizian et al. (2020a) and Ibrahim et al. (2020) provide spectral lower bounds for the rate of convergence in the bilinear setting for an accelerated algorithm developed in Azizian et al. (2020b) for a specific families of bilinear games. Furthermore, Fiez & Ratliff (2020) use Lyapunov analysis to provide convergence guarantees for gradient descent ascent using timescale separation and in Hsieh et al. (2020), authors show that commonly used algorithms for min-max optimization converge to attractors that are not optimal.

Single-objective Optimization and Dynamical Systems: The authors of Su et al. (2014) started a new trend in single-objective optimization by studying the continuous-time dynamics of Nesterov's accelerated method (Nesterov, 2013). Their analysis allowed for a better understanding of the much-celebrated Nesterov's method. In a similar spirit, Wibisono et al. (2016); Wilson et al. (2016) study continuous-time accelerated methods within a Lagrangian framework, while analyzing their stability using Lyapunov analysis. These work show that a family of discrete-time methods can be derived from their corresponding continuous-time formalism using various discretization schemes. Additionally, several recent work (Muehlebach & Jordan, 2019; Bailey & Piliouras, 2019; Maddison et al., 2018; Ryu et al., 2019) cast game optimization algorithms as dynamical systems so to leverage its rich theory, to study the stability and convergence of various continuous-time methods. Nagarajan & Kolter (2017) also analyzes the local stability of GANs as an approximated continuous dynamical system.

9 Conclusion

In this paper, we leverage tools from physics to propose a novel second-order optimization scheme LEAD, to address the issue of rotational dynamics in min-max games. By casting min-max game optimization as a physical system, we use the principle of least action to discover an effective optimization algorithm for this setting. Subsequently, with the use of Lyapunov stability theory and spectral analysis, we prove LEAD to be convergent at a linear rate in bilinear min-max games. We supplement our theoretical analysis with experiments on GANs and toy setups, demonstrating improvements over baseline methods. Specifically for GAN training, we observe that our method outperforms other second-order methods, both in terms of sample quality and computational efficiency. Our analysis underlines the advantages of physical approaches in designing novel optimization algorithms for games as well as for traditional optimization tasks. It is important to note in this regard that our crafted physical system is *a* way to model min-max optimization physically. Alternate schemes to perform such modeling can involve other choices of counter-rotational and dissipative forces which can be explored in future work.

Broader Impact Statement

While our contribution is mostly theoretical, our research has the potential to improve the optimization of multi-agent machine learning models, such as generative adversarial networks (GANs). GANs have been very successful in generating realistic images, music, speech and text, and for improving performance on an array of different real-world tasks. On the other hand, GANs can be misused to generate fake news, fake images, and fake voices. Furthermore, a common problem encountered during GAN training is mode-collapse. This results in GANs being biased in generating certain types of data moreover others, thereby causing data misrepresentation. In this paper, we show that our proposed method can tackle the mode collapse problem by observing improvements over baseline methods. However, we would like to emphasize that practitioners should use our research with caution as the change of dataset and tasks might not prevent the mode collapse problem.

References

Jacob Abernethy, Kevin A Lai, and Andre Wibisono. Last-iterate convergence rates for min-max optimization. *arXiv preprint arXiv:1906.02027*, 2019.

Waïss Azizian, Ioannis Mitliagkas, Simon Lacoste-Julien, and Gauthier Gidel. A tight and unified analysis of gradient-based methods for a whole spectrum of differentiable games. In *International Conference on Artificial Intelligence and Statistics*, pp. 2863–2873, 2020a.

Waïss Azizian, Damien Scieur, Ioannis Mitliagkas, Simon Lacoste-Julien, and Gauthier Gidel. Accelerating smooth games by manipulating spectral shapes. *International Conference on Artificial Intelligence and Statistics*, 2020b.

James P Bailey and Georgios Piliouras. Multi-agent learning in network zero-sum games is a hamiltonian system. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 233–241. International Foundation for Autonomous Agents and Multiagent Systems, 2019.

D Balduzzi, S Racaniere, J Martens, J Foerster, K Tuyls, and T Graepel. *deepmind-symplectic-gradient-adjustment*, 2018a. `https://github.com/deepmind/symplectic-gradient-adjustment/blob/master/Symplectic_Gradient_Adjustment.ipynb`.

D Balduzzi, S Racaniere, J Martens, J Foerster, K Tuyls, and T Graepel. The mechanics of n-player differentiable games. In *ICML*, volume 80, pp. 363–372. JMLR. org, 2018b.

MV Berry and Pragya Shukla. Curl force dynamics: symmetries, chaos and constants of motion. *New Journal of Physics*, 18(6):063018, 2016.

Dimitri P Bertsekas. *Nonlinear programming.* Athena scientific Belmont, 1999.

Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.

Lucian Bu, Robert Babu, Bart De Schutter, et al. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 38 (2):156–172, 2008.

Tatjana Chavdarova, Matteo Pagliardini, Martin Jaggi, and Francois Fleuret. Taming gans with lookahead. *arXiv preprint arXiv:2006.14567*, 2020.

Constantinos Daskalakis, Andrew Ilyas, Vasilis Syrgkanis, and Haoyang Zeng. Training gans with optimism. In *International Conference on Learning Representations*, 2018.

Tanner Fiez and Lillian Ratliff. Gradient descent-ascent provably converges to strict local minmax equilibria with a finite timescale separation. *arXiv preprint arXiv:2009.14820*, 2020.

Tanner Fiez and Lillian J Ratliff. Local convergence analysis of gradient descent ascent with finite timescale separation. In *Proceedings of the International Conference on Learning Representation*, 2021.

Jakob Foerster, Richard Y Chen, Maruan Al-Shedivat, Shimon Whiteson, Pieter Abbeel, and Igor Mordatch. Learning with opponent-learning awareness. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 122–130. International Foundation for Autonomous Agents and Multiagent Systems, 2018.

Ian Gemp and Sridhar Mahadevan. Global convergence to the equilibrium of gans using variational inequalities. *arXiv preprint arXiv:1808.01531*, 2018.

Gauthier Gidel, Hugo Berard, Gaëtan Vignoud, Pascal Vincent, and Simon Lacoste-Julien. A variational inequality perspective on generative adversarial networks. In *International Conference on Learning Representations*, 2018.

Gauthier Gidel, Reyhane Askari Hemmat, Mohammad Pezeshki, Rémi Le Priol, Gabriel Huang, Simon Lacoste-Julien, and Ioannis Mitliagkas. Negative momentum for improved game dynamics. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 1802–1811, 2019.

Noah Golowich, Sarath Pattathil, Constantinos Daskalakis, and Asuman Ozdaglar. Last iterate is slower than averaged iterate in smooth convex-concave saddle point problems. *arXiv preprint arXiv:2002.00057*, 2020.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pp. 2672–2680, 2014.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *Advances in neural information processing systems*, pp. 5767–5777, 2017.

Wolfgang Hahn, Hans H Hosenthien, and H Lehnigk. *Theory and application of Liapunov's direct method*. Prentice-Hall Englewood Cliffs, NJ, 1963.

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*, pp. 6626–6637, 2017.

Ya-Ping Hsieh, Panayotis Mertikopoulos, and Volkan Cevher. The limits of min-max optimization algorithms: convergence to spurious non-critical sets. *arXiv preprint arXiv:2006.09065*, 2020.

Adam Ibrahim, Waïss Azizian, Gauthier Gidel, and Ioannis Mitliagkas. Linear lower bounds and conditioning of differentiable games. In *International conference on machine learning*, 2020.

Minguk Kang, Woohyeon Shim, Minsu Cho, and Jaesik Park. Rebooting acgan: Auxiliary classifier gans with stable training. *Advances in Neural Information Processing Systems*, 34:23505–23518, 2021.

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

GM Korpelevich. The extragradient method for finding saddle points and other problems. *Matecon*, 12: 747–756, 1976.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

LD Landau and EM Lifshitz. *Course of theoretical physics. vol. 1: Mechanics*. Oxford, 1960.

Kwonjoon Lee, Huiwen Chang, Lu Jiang, Han Zhang, Zhuowen Tu, and Ce Liu. Vitgan: Training gans with vision transformers. *arXiv preprint arXiv:2107.04589*, 2021.

Alistair Letcher, David Balduzzi, Sébastien Racaniere, James Martens, Jakob N Foerster, Karl Tuyls, and Thore Graepel. Differentiable game mechanics. *Journal of Machine Learning Research*, 20(84):1–40, 2019.

Nicolas Loizou, Hugo Berard, Alexia Jolicoeur-Martineau, Pascal Vincent, Simon Lacoste-Julien, and Ioannis Mitliagkas. Stochastic hamiltonian gradient methods for smooth games. *ICML*, 2020.

David Acuna Paul Vicol Lorraine, Jonathan P. and David Duvenaud. Complex momentum for optimization in games. In *International Conference on Artificial Intelligence and Statistics*, pp. 7742–7765. PMLR, 2022.

Aleksandr Mikhailovich Lyapunov. The general problem of the stability of motion. *International journal of control*, 55(3):531–534, 1992.

Chris J Maddison, Daniel Paulin, Yee Whye Teh, Brendan O'Donoghue, and Arnaud Doucet. Hamiltonian descent methods. *arXiv preprint arXiv:1809.05042*, 2018.

Eric V Mazumdar, Michael I Jordan, and S Shankar Sastry. On finding local nash equilibria (and only local nash equilibria) in zero-sum games. *arXiv preprint arXiv: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. *arXiv preprint arXiv:1807.02629*, 2018.

Lars Mescheder, Sebastian Nowozin, and Andreas Geiger. The numerics of gans. In *Advances in Neural Information Processing Systems*, pp. 1825–1835, 2017.

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018.

Michael Muehlebach and Michael Jordan. A dynamical systems perspective on nesterov acceleration. In *International Conference on Machine Learning*, pp. 4656–4662, 2019.

Vaishnavh Nagarajan and J Zico Kolter. Gradient descent gan optimization is locally stable. In *Advances in neural information processing systems*, pp. 5585–5595, 2017.

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

Maher Nouiehed, Maziar Sanjabi, Tianjian Huang, Jason D Lee, and Meisam Razaviyayn. Solving a class of non-convex min-max games using iterative first order methods. In *Advances in Neural Information Processing Systems*, pp. 14934–14942, 2019.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. s. 2017.

Boris T Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17, 1964.

Chongli Qin, Yan Wu, Jost Tobias Springenberg, Andrew Brock, Jeff Donahue, Timothy P Lillicrap, and Pushmeet Kohli. Training generative adversarial networks by solving ordinary differential equations. *arXiv preprint arXiv:2010.15040*, 2020.

Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

Ernest K Ryu, Kun Yuan, and Wotao Yin. Ode analysis of stochastic gradient methods with optimism and anchoring for minimax problems and gans. *arXiv preprint arXiv:1905.10899*, 2019.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in neural information processing systems*, pp. 2234–2242, 2016.

Axel Sauer, Katja Schwarz, and Andreas Geiger. Stylegan-xl: Scaling stylegan to large diverse datasets. In *ACM SIGGRAPH 2022 Conference Proceedings*, pp. 1–10, 2022.

Florian Schäfer and Anima Anandkumar. Competitive gradient descent. In *Advances in Neural Information Processing Systems*, pp. 7623–7633, 2019.

Ozan Sener and Vladlen Koltun. Multi-task learning as multi-objective optimization. In *Advances in Neural Information Processing Systems*, pp. 527–538, 2018.

Bin Shi, Simon S Du, Weijie Su, and Michael I Jordan. Acceleration via symplectic discretization of high-resolution differential equations. In *Advances in Neural Information Processing Systems*, pp. 5745–5753, 2019.

Weijie Su, Stephen Boyd, and Emmanuel Candes. A differential equation for modeling nesterov's accelerated gradient method: Theory and insights. In *Advances in Neural Information Processing Systems*, pp. 2510–2518, 2014.

Yuanhao Wang, Guodong Zhang, and Jimmy Ba. On solving minimax optimization locally: A follow-the-ridge approach. In *International Conference on Learning Representations*, 2019.

Andre Wibisono, Ashia C Wilson, and Michael I Jordan. A variational perspective on accelerated methods in optimization. *Proceedings of the National Academy of Sciences*, 113(47):E7351–E7358, 2016.

Ashia C Wilson, Benjamin Recht, and Michael I Jordan. A lyapunov analysis of momentum methods in optimization. *arXiv preprint arXiv:1611.02635*, 2016.

Yasin Yazıcı, Chuan-Sheng Foo, Stefan Winkler, Kim-Hui Yap, Georgios Piliouras, and Vijay Chandrasekhar. The unusual effectiveness of averaging in gan training, 2019.

Guodong Zhang and Yuanhao Wang. On the suboptimality of negative momentum for minimax optimization. In *International Conference on Artificial Intelligence and Statistics*, pp. 2098–2106. PMLR, 2021.

Guodong Zhang, Yuanhao Wang, Laurent Lessard, and Roger B Grosse. Near-optimal local convergence of alternating gradient descent-ascent for minimax optimization. In *International Conference on Artificial Intelligence and Statistics*, pp. 7659–7679. PMLR, 2022.

A Appendix

B Proof of Proposition 1

Proof. The EOMs of the quadratic game in continuous-time (Eq.(10)), can be discretized in using a combination of implicit and explicit update steps as (Shi et al., 2019),

$$x_{k+1} - x_k = \delta v^x_{k+1}, \tag{32a}$$
$$y_{k+1} - y_k = \delta v^y_{k+1}, \tag{32b}$$
$$v^x_{k+1} - v^x_k = -q\delta \nabla_{xy} f\left(x_k, y_k\right) v^y_k - \mu\delta v^x_k - \delta\nabla_x f\left(x_k, y_k\right) \tag{32c}$$
$$v^y_{k+1} - v^y_k = q\delta \nabla_{xy} f\left(x_k, y_k\right) v^x_k - \mu\delta v^y_k + \delta\nabla_y f\left(x_k, y_k\right) \tag{32d}$$

where δ is the discretization step-size. Using Eqns.(32a) and (32b), we can further re-express Eqns. (32c), (32d) as,

$$\begin{aligned} x_{k+1} &= x_k + \beta\Delta x_k - \eta\nabla_x f\left(x_k, y_k\right) - \alpha\nabla_{x,y} f\left(x_k, y_k\right)\Delta y_k \\ y_{k+1} &= y_k + \beta\Delta y_k + \eta\nabla_y f\left(x_k, y_k\right) + \alpha\nabla_{x,y} f\left(x_k, y_k\right)\Delta x_k \end{aligned} \tag{33}$$

where $\Delta x_k = x_k - x_{k-1}$, and,

$$\beta = 1 - \mu\delta,\ \eta = \delta^2,\ \alpha = 2q\delta \tag{34}$$

□

C Continuous-time Convergence Analysis: Quadratic Min-Max Game

Proof. For the class of quadratic min-max games,

$$f\left(\boldsymbol{X}, \boldsymbol{Y}\right) = \frac{h}{2}|\boldsymbol{X}|^2 - \frac{h}{2}|\boldsymbol{Y}|^2 + \boldsymbol{X}^T\mathbb{A}\boldsymbol{Y} \tag{35}$$

where $\boldsymbol{X} \equiv \left(X^1, \cdots, X^n\right), \boldsymbol{Y} \equiv \left(Y^1, \cdots, Y^n\right) \in \mathbb{R}^n$ and $\mathbb{A}_{n\times n}$ is a constant positive-definite matrix, the continuous-time EOMs of Eq.(10) become:

$$\begin{aligned} \ddot{\boldsymbol{X}} &= -\mu\dot{\boldsymbol{X}} - h\boldsymbol{X} - \mathbb{A}\boldsymbol{Y} - q\mathbb{A}\dot{\boldsymbol{Y}} \\ \ddot{\boldsymbol{Y}} &= -\mu\dot{\boldsymbol{Y}} - h\boldsymbol{Y} + \mathbb{A}^T\boldsymbol{X} + q\mathbb{A}^T\dot{\boldsymbol{X}} \end{aligned} \tag{36}$$

We next define our continuous-time Lyapunov function in this case to be,

$$\begin{aligned}\mathcal{E}_t &= \frac{1}{2}\left(\dot{\boldsymbol{X}} + \mu\boldsymbol{X} + \mu\mathbb{A}\boldsymbol{Y}\right)^T\left(\dot{\boldsymbol{X}} + \mu\boldsymbol{X} + \mu\mathbb{A}\boldsymbol{Y}\right) \\ &\quad + \frac{1}{2}\left(\dot{\boldsymbol{Y}} + \mu\boldsymbol{Y} - \mu\mathbb{A}^T\boldsymbol{X}\right)^T\left(\dot{\boldsymbol{X}} + \mu\boldsymbol{Y} - \mu\mathbb{A}^T\boldsymbol{X}\right) \\ &\quad + \frac{1}{2}\left(\dot{\boldsymbol{X}}^T\dot{\boldsymbol{X}} + \dot{\boldsymbol{Y}}^T\dot{\boldsymbol{Y}}\right) + \boldsymbol{X}^T(h + \mathbb{A}\mathbb{A}^T)\boldsymbol{X} + \boldsymbol{Y}^T(h + \mathbb{A}^T\mathbb{A})\boldsymbol{Y} \\ &\geq 0 \;\forall\; t\end{aligned} \tag{37}$$

The time-derivative of $\mathcal{E}_t$ is then given by,

$$\begin{aligned}\dot{\mathcal{E}}_t &= \left(\dot{\boldsymbol{X}} + \mu\boldsymbol{X} + \mu\mathbb{A}\boldsymbol{Y}\right)^T\left(\ddot{\boldsymbol{X}} + \mu\dot{\boldsymbol{X}} + \mu\mathbb{A}\dot{\boldsymbol{Y}}\right) + \left(\dot{\boldsymbol{Y}} + \mu\boldsymbol{Y} - \mu\mathbb{A}^T\boldsymbol{X}\right)^T\left(\ddot{\boldsymbol{Y}} + \mu\dot{\boldsymbol{Y}} - \mu\mathbb{A}^T\dot{\boldsymbol{X}}\right) \\ &\quad + \left(\dot{\boldsymbol{X}}^T\ddot{\boldsymbol{X}} + \dot{\boldsymbol{Y}}^T\ddot{\boldsymbol{Y}}\right) + 2\left(\boldsymbol{X}^T(h + \mathbb{A}\mathbb{A}^T)\dot{\boldsymbol{X}} + \boldsymbol{Y}^T(h + \mathbb{A}^T\mathbb{A})\dot{\boldsymbol{Y}}\right) \\ &= \left(\dot{\boldsymbol{X}}^T + \mu\boldsymbol{X}^T + \mu\boldsymbol{Y}^T\mathbb{A}^T\right)\left(\left(-q + \mu\right)\mathbb{A}\dot{\boldsymbol{Y}} - \mathbb{A}\boldsymbol{Y}\right) + \dot{\boldsymbol{X}}^T\left(-q\mathbb{A}\dot{\boldsymbol{Y}} - \mu\dot{\boldsymbol{X}} - \mathbb{A}\boldsymbol{Y}\right) \\ &\quad + \left(\dot{\boldsymbol{Y}}^T + \mu\boldsymbol{Y}^T - \mu\boldsymbol{X}^T\mathbb{A}\right)\left(\left(q - \mu\right)\mathbb{A}^T\dot{\boldsymbol{X}} + \mathbb{A}^T\boldsymbol{X}\right) + \dot{\boldsymbol{Y}}^T\left(q\mathbb{A}^T\dot{\boldsymbol{X}} - \mu\dot{\boldsymbol{Y}} + \mathbb{A}^T\boldsymbol{X}\right) \\ &\quad + 2\left(\boldsymbol{X}^T(h + \mathbb{A}\mathbb{A}^T)\dot{\boldsymbol{X}} + \boldsymbol{Y}^T(h + \mathbb{A}^T\mathbb{A})\dot{\boldsymbol{Y}}\right) \\ &= \left(\mu\left(q - \mu\right) - 2\right)\left(\boldsymbol{Y}^T\mathbb{A}^T\dot{\boldsymbol{X}} - \boldsymbol{X}^T\mathbb{A}\dot{\boldsymbol{Y}}\right) - \left(\mu\left(q - \mu\right) - 2\right)\left(\boldsymbol{X}^T\mathbb{A}\mathbb{A}^T\dot{\boldsymbol{X}} + \boldsymbol{Y}^T\mathbb{A}^T\mathbb{A}\dot{\boldsymbol{Y}}\right) \\ &\quad - \mu\left(\boldsymbol{X}^T(h + \mathbb{A}\mathbb{A}^T)\boldsymbol{X} + \boldsymbol{Y}^T(h + \mathbb{A}^T\mathbb{A})\boldsymbol{Y}\right) - \mu\left(\dot{\boldsymbol{X}}^T\dot{\boldsymbol{X}} + \dot{\boldsymbol{Y}}^T\dot{\boldsymbol{Y}}\right)\end{aligned} \tag{38}$$

where we have used the fact that $\boldsymbol{X}^T\mathbb{A}\boldsymbol{Y}$ being a scalar thus implying $\boldsymbol{X}^T\mathbb{A}\boldsymbol{Y} = \boldsymbol{Y}^T\mathbb{A}^T\boldsymbol{X}$. If we now set $q = (2/\mu) + \mu$ in the above, then that further leads to,

$$\begin{aligned}\dot{\mathcal{E}}_t &= -\mu\left(\boldsymbol{X}^T(h + \mathbb{A}\mathbb{A}^T)\boldsymbol{X} + \boldsymbol{Y}^T(h + \mathbb{A}^T\mathbb{A})\boldsymbol{Y}\right) - \mu\left(\dot{\boldsymbol{X}}^T\dot{\boldsymbol{X}} + \dot{\boldsymbol{Y}}^T\dot{\boldsymbol{Y}}\right) \\ &= -\mu\left(h||\boldsymbol{X}||^2 + h||\boldsymbol{Y}||^2 + \left|\left|\mathbb{A}^T\boldsymbol{X}\right|\right|^2 + ||\mathbb{A}\boldsymbol{Y}||^2\right) - \mu\left(\left|\left|\dot{\boldsymbol{X}}\right|\right|^2 + \left|\left|\dot{\boldsymbol{Y}}\right|\right|^2\right) \leq 0 \;\forall\; t\end{aligned} \tag{39}$$

exhibiting that the Lyapunov function, Eq.(16) is *asymptotically stable* at all times t.

Next, consider the following expression,

$$\begin{aligned}&-\rho\mathcal{E}_t - \frac{\rho\mu}{2}\left|\left|\boldsymbol{X} - \dot{\boldsymbol{X}}\right|\right|^2 - \frac{\rho\mu}{2}\left|\left|\boldsymbol{Y} - \dot{\boldsymbol{Y}}\right|\right|^2 - \frac{\rho\mu}{2}\left|\left|\dot{\boldsymbol{X}} - \mathbb{A}\boldsymbol{Y}\right|\right|^2 - \frac{\rho\mu}{2}\left|\left|\mathbb{A}^T\boldsymbol{X} + \dot{\boldsymbol{Y}}\right|\right|^2 \\ &\quad = -\rho\mathcal{E}_t - \frac{\rho\mu}{2}\left(||\boldsymbol{X}||^2 + ||\boldsymbol{Y}||^2\right) + \rho\mu\left(\boldsymbol{X}^T\dot{\boldsymbol{X}} + \boldsymbol{Y}^T\dot{\boldsymbol{Y}}\right) - \rho\mu\left(\left|\left|\dot{\boldsymbol{X}}\right|\right|^2 + ||\boldsymbol{Y}||^2\right) \\ &\quad\quad - \rho\mu\left(\boldsymbol{X}^T\mathbb{A}\dot{\boldsymbol{Y}} - \dot{\boldsymbol{X}}^T\mathbb{A}\boldsymbol{Y}\right) - \frac{\rho\mu}{2}\left(\left|\left|\mathbb{A}^T\boldsymbol{X}\right|\right|^2 + ||\mathbb{A}\boldsymbol{Y}||^2\right) \\ &\quad = -\rho\left(1 + \mu\right)\left(\left|\left|\dot{\boldsymbol{X}}\right|\right|^2 + \left|\left|\dot{\boldsymbol{Y}}\right|\right|^2\right) - \frac{\rho}{2}\left(\mu^2 + \mu + 2h\right)\left(||\boldsymbol{X}||^2 + ||\boldsymbol{Y}||^2\right) \\ &\quad\quad - \frac{\rho}{2}\left(\mu^2 + \mu + 2\right)\left(\left|\left|\mathbb{A}^T\boldsymbol{X}\right|\right|^2 + ||\mathbb{A}\boldsymbol{Y}||^2\right) \\ &\quad \leq -\rho\mathcal{E}_t\end{aligned} \tag{40}$$

where ρ is some positive definite constant. This implies that the above expression is negative semi-definite by construction given $\mu \geq 0$. Now, for a general square matrix $\mathbb{A}$, we can perform a singular value decomposition (SVD) as $\mathbb{A} = \mathbb{V}^{\mathrm{T}}\mathbb{S}\mathbb{U}$. Here, $\mathbb{U}$ and $\mathbb{V}$ are the right and left unitaries of $\mathbb{A}$, while $\mathbb{S}$ is a diagonal matrix of singular values (σ_i) of $\mathbb{A}$. Using this decomposition in Eq.(40), then allows us to write,

$$\begin{aligned}
&-\rho\,(1+\mu)\left(\left|\left|\dot{\boldsymbol{X}}\right|\right|^2+\left|\left|\dot{\boldsymbol{Y}}\right|\right|^2\right)-\frac{\rho}{2}\left(\mu^2+\mu+2h\right)\left(||\boldsymbol{X}||^2+||\boldsymbol{Y}||^2\right)\\
&\qquad-\frac{\rho}{2}\left(\mu^2+\mu+2\right)\left(\left|\left|\mathbb{A}^T\boldsymbol{X}\right|\right|^2+||\mathbb{A}\boldsymbol{y}||^2\right)\\
&=-\rho\,(1+\mu)\left(\left|\left|\mathbb{V}\dot{\boldsymbol{X}}\right|\right|^2+\left|\left|\mathbb{U}\dot{\boldsymbol{Y}}\right|\right|^2\right)-\frac{\rho}{2}\left(\mu^2+\mu+2h\right)\left(||\mathbb{V}\boldsymbol{X}||^2+||\mathbb{U}\boldsymbol{Y}||^2\right)\\
&\qquad-\frac{\rho}{2}\left(\mu^2+\mu+2\right)\left(||\mathbb{S}\mathbb{V}\boldsymbol{X}||^2+||\mathbb{S}\mathbb{U}\boldsymbol{Y}||^2\right)\\
&=-\rho\,(1+\mu)\left(\left|\left|\dot{\boldsymbol{\mathcal{X}}}\right|\right|^2+\left|\left|\dot{\boldsymbol{\mathcal{Y}}}\right|\right|^2\right)-\frac{\rho}{2}\left(\mu^2+\mu+2h\right)\left(||\boldsymbol{\mathcal{X}}||^2+||\boldsymbol{\mathcal{Y}}||^2\right)\\
&\qquad-\frac{\rho}{2}\left(\mu^2+\mu+2\right)\left(||\mathbb{S}\boldsymbol{\mathcal{X}}||^2+||\mathbb{S}\boldsymbol{\mathcal{Y}}||^2\right)\\
&=-\sum_{j=1}^{n}\rho\,(1+\mu)\left(\left|\left|\dot{\mathcal{X}}^j\right|\right|^2+\left|\left|\dot{\mathcal{Y}}^j\right|\right|^2\right)\\
&\qquad-\sum_{j=1}^{n}\frac{\rho}{2}\left(\left(1+\sigma_j^2+2h\right)\left(\mu^2+\mu\right)+2\sigma_j^2\right)\left(\left|\left|\mathcal{X}^j\right|\right|^2+\left|\left|\mathcal{Y}^j\right|\right|^2\right)
\end{aligned} \tag{41}$$

where we have made use of the relations $\mathbb{U}^{\mathrm{T}}\mathbb{U}=\mathbb{U}\mathbb{U}^{\mathrm{T}}=\mathbb{I}_n=\mathbb{V}^{\mathrm{T}}\mathbb{V}=\mathbb{V}\mathbb{V}^{\mathrm{T}}$, and additionally performed a basis change, as $\boldsymbol{\mathcal{X}}=\mathbb{V}\boldsymbol{X}$ and $\boldsymbol{\mathcal{Y}}=\mathbb{U}\boldsymbol{Y}$. Now, we know from Eq.(39) that,

$$\begin{aligned}
\dot{\mathcal{E}}_t&=-\mu\left(h||\boldsymbol{X}||^2+h||\boldsymbol{Y}||^2+\left|\left|\mathbb{A}^T\boldsymbol{X}\right|\right|^2+||\mathbb{A}\boldsymbol{Y}||^2\right)-\mu\left(\left|\left|\dot{\boldsymbol{X}}\right|\right|^2+\left|\left|\dot{\boldsymbol{Y}}\right|\right|^2\right)\\
&=-\mu\left(h||\boldsymbol{X}||^2+h||\boldsymbol{Y}||^2+\left|\left|\mathbb{U}^{\mathrm{T}}\mathbb{S}\mathbb{V}\boldsymbol{X}\right|\right|^2+\left|\left|\mathbb{V}^{\mathrm{T}}\mathbb{S}\mathbb{U}\boldsymbol{Y}\right|\right|^2\right)-\mu\left(\left|\left|\mathbb{V}\dot{\boldsymbol{X}}\right|\right|^2+\left|\left|\mathbb{U}\dot{\boldsymbol{Y}}\right|\right|^2\right)\\
&=-\mu\left(h||\boldsymbol{\mathcal{X}}||^2+h||\boldsymbol{\mathcal{Y}}||^2+||\mathbb{S}\boldsymbol{\mathcal{X}}||^2+||\mathbb{S}\boldsymbol{\mathcal{Y}}||^2\right)-\mu\left(\left|\left|\dot{\boldsymbol{\mathcal{X}}}\right|\right|^2+\left|\left|\dot{\boldsymbol{\mathcal{Y}}}\right|\right|^2\right)\\
&=-\sum_{j=1}^{n}\mu\left(\sigma_j^2+h\right)\left(\left|\left|\mathcal{X}^j\right|\right|^2+\left|\left|\mathcal{Y}^j\right|\right|^2\right)-\sum_{j=1}^{n}\mu\left(\left|\left|\dot{\mathcal{X}}^j\right|\right|^2+\left|\left|\dot{\mathcal{Y}}^j\right|\right|^2\right)
\end{aligned} \tag{42}$$

Comparing the above expression with Eq.(41), we note that a choice of ρ as,

$$\rho\leq\min\left\{\frac{\mu}{1+\mu},\ \frac{2\mu(\sigma_{\min}^2+h)}{\left(1+\sigma_{\min}^2+2h\right)\left(\mu^2+\mu\right)+2\sigma_{\min}^2}\right\}\ \forall\ j\in[1,n] \tag{43}$$

implies,

$$\begin{aligned}
\dot{\mathcal{E}}_t&\leq-\rho\mathcal{E}\\
\Rightarrow\mathcal{E}_t&\leq\mathcal{E}_0\exp\left(-\rho t\right)\\
\Rightarrow X^T\left(h+\mathbb{A}\mathbb{A}^T\right)X+Y^T\left(h+\mathbb{A}^T\mathbb{A}\right)Y&\leq\mathcal{E}_0\exp\left(-\rho t\right)\\
\Rightarrow\mathcal{X}^T(h+\mathbb{S}^2)\mathcal{X}+\mathcal{Y}^T(h+\mathbb{S}^2)\mathcal{Y}&\leq\mathcal{E}_0\exp\left(-\rho t\right)\\
\Rightarrow\sum_{j=1}^{n}(h+\sigma_j^2)\left(||\mathcal{X}^j||^2+||\mathcal{Y}^j||^2\right)&\leq\mathcal{E}_0\exp\left(-\rho t\right)\\
\Rightarrow\sum_{j=1}^{n}(h+\sigma_j^2)\left(||X^j||^2+||Y^j||^2\right)&\leq\mathcal{E}_0\exp\left(-\rho t\right)\\
\therefore\ ||\boldsymbol{X}||^2+||\boldsymbol{Y}||^2&\leq\frac{\mathcal{E}_0}{h+\sigma_{\min}^2}\exp\left(-\rho t\right)\ \forall\ j
\end{aligned} \tag{44}$$

□

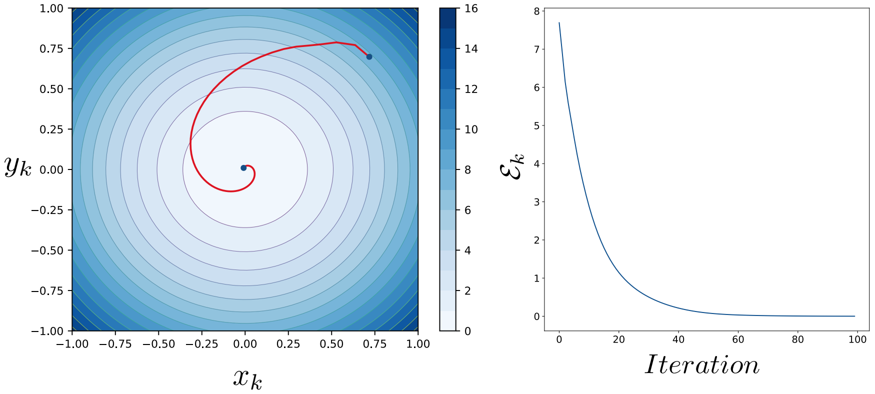


Figure 5: **Left:** Contours of the Lyapunov function $\mathcal{E}_k$, Eq. (37) (black), and convergence trajectory of LEAD (red) in the quadratic min-max game (Eq.(36)) to the Nash equilibrium $(0,0)$. **Right**: The evolution of the discrete-time Lyapunov function of Eq. (37) over iteration, confirming $\mathcal{E}_k - \mathcal{E}_{k-1} \leq 0 \;\forall\; k \in \mathbf{N}$. - *This is colored in Red as the discrete-time Lyapunov has been removed from the text, yet the above diagram only holds in discrete time.*

D Proof of Theorem 2

Theorem. *The eigenvalues of* $\nabla F_{LEAD}(\boldsymbol{\omega}^*)$ *about the Nash equilibrium* $\boldsymbol{\omega}^* = (x^*, y^*)$ *of the quadratic min-max game are,*

$$\mu_{\pm}(\alpha,\beta,\eta) = \frac{1-(\eta+\alpha)\lambda+\beta-\eta h \pm \sqrt{\Delta}}{2} \tag{45}$$

where, $\Delta = \left(1-(\eta+\alpha)\lambda+\beta-\eta h\right)^2 - 4\left(\beta-\alpha\lambda\right)$ *and* $\lambda \in Sp(\text{off-diag}[\nabla \boldsymbol{v}(\boldsymbol{\omega}^*)])$*. Furthermore, for* $h, \eta, |\alpha|, |\beta| << 1$*, we have,*

$$\begin{aligned} \mu_+^{(i)}(\alpha,\beta,\eta) &\approx 1-\eta h + \frac{\left(\eta+\alpha\right)^2\lambda_i^2+\eta^2h^2+\beta^2-2\eta h\beta}{4} \\ &\quad + \lambda_i\left(\frac{\eta+\alpha}{2}\left(\eta h-\beta\right)-\eta\right) \\ \mu_-^{(i)}(\alpha,\beta,\eta) &\approx \beta - \frac{\left(\eta+\alpha\right)^2\lambda_i^2+\eta^2h^2+\beta^2-2\eta h\beta}{4} \\ &\quad + \lambda_i\left(\frac{\eta+\alpha}{2}\left(\beta-\eta h\right)-\alpha\right) \end{aligned} \tag{46}$$

Proof. For the quadratic game equation 35, the Jacobian of the vector field $\boldsymbol{v}$ is given by,

$$\nabla \boldsymbol{v} \equiv \nabla \begin{bmatrix} \nabla_x f(\boldsymbol{x}_t, \boldsymbol{y}_t) \\ -\nabla_y f(\boldsymbol{x}_t, \boldsymbol{y}_t) \end{bmatrix} = \begin{bmatrix} h\mathbb{I}_{2n} & \mathbb{A} \\ -\mathbb{A}^\top & h\mathbb{I}_{2n} \end{bmatrix} \in \mathbb{R}^{2n} \times \mathbb{R}^{2n}. \tag{47}$$

Let us next define a matrix $\mathbb{D}_q$ as,

$$\mathbb{D}_q = \begin{bmatrix} \nabla^2_{xy} f(\boldsymbol{x}, \boldsymbol{y}) & 0 \\ 0 & -\nabla^2_{xy} f(\boldsymbol{x}, \boldsymbol{y}) \end{bmatrix} = \begin{bmatrix} \mathbb{A} & 0 \\ 0 & -\mathbb{A}^\top \end{bmatrix} \in \mathbb{R}^{2n} \times \mathbb{R}^{2n} \tag{48}$$

Consequently, the update rule for LEAD can be written as:

$$\begin{aligned} \begin{bmatrix} \boldsymbol{x}_{t+1} \\ \boldsymbol{y}_{t+1} \end{bmatrix} &= \begin{bmatrix} \boldsymbol{x}_t \\ \boldsymbol{y}_t \end{bmatrix} + \beta \begin{bmatrix} \boldsymbol{x}_t - \boldsymbol{x}_{t-1} \\ \boldsymbol{y}_t - \boldsymbol{y}_{t-1} \end{bmatrix} - \eta \begin{bmatrix} \nabla_x f(\boldsymbol{x}_t, \boldsymbol{y}_t) \\ -\nabla_y f(\boldsymbol{x}_t, \boldsymbol{y}_t) \end{bmatrix} - \alpha \begin{bmatrix} \nabla^2_{xy} f(\boldsymbol{x}_t, \boldsymbol{y}_t)\Delta \boldsymbol{y}_t \\ -\nabla^2_{xy} f(\boldsymbol{x}_t, \boldsymbol{y}_t)\Delta \boldsymbol{x}_t \end{bmatrix} \\ &= \begin{bmatrix} \boldsymbol{x}_t \\ \boldsymbol{y}_t \end{bmatrix} + \beta \begin{bmatrix} \boldsymbol{x}_t - \boldsymbol{x}_{t-1} \\ \boldsymbol{y}_t - \boldsymbol{y}_{t-1} \end{bmatrix} - \eta \boldsymbol{v} - \alpha \mathbb{D}_{\text{q}} \begin{bmatrix} \Delta \boldsymbol{y}_t \\ \Delta \boldsymbol{x}_t \end{bmatrix} \end{aligned} \tag{49}$$

where $\Delta \boldsymbol{y}_t = \boldsymbol{y}_t - \boldsymbol{y}_{t-1}$ and $\Delta \boldsymbol{x}_t = \boldsymbol{x}_t - \boldsymbol{x}_{t-1}$.

Next, by making use of the permutation matrix $\mathbb{P}$,

$$\mathbb{P} := \begin{bmatrix} 0 & \mathbb{I}_n \\ \mathbb{I}_n & 0 \end{bmatrix} \in \mathbb{R}^{2n} \times \mathbb{R}^{2n}$$

we can re-express Eq. equation 49 as,

$$\begin{aligned} \begin{bmatrix} \boldsymbol{\omega}_{t+1} \\ \boldsymbol{\omega}_t \end{bmatrix} &= \begin{bmatrix} \mathbb{I}_{2n} & 0 \\ \mathbb{I}_{2n} & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{\omega}_t \\ \boldsymbol{\omega}_{t-1} \end{bmatrix} + \beta \begin{bmatrix} \mathbb{I}_{2n} & -\mathbb{I}_{2n} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{\omega}_t \\ \boldsymbol{\omega}_{t-1} \end{bmatrix} - \eta \begin{bmatrix} \boldsymbol{v} \\ 0 \end{bmatrix} - \alpha \begin{bmatrix} \mathbb{D}_{\mathrm{q}} & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbb{P} & -\mathbb{P} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{\omega}_t \\ \boldsymbol{\omega}_{t-1} \end{bmatrix} \\ &= \begin{bmatrix} \mathbb{I}_{2n} & 0 \\ \mathbb{I}_{2n} & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{\omega}_t \\ \boldsymbol{\omega}_{t-1} \end{bmatrix} + \beta \begin{bmatrix} \mathbb{I}_{2n} & -\mathbb{I}_{2n} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{\omega}_t \\ \boldsymbol{\omega}_{t-1} \end{bmatrix} - \eta \begin{bmatrix} \boldsymbol{v} \\ 0 \end{bmatrix} - \alpha \begin{bmatrix} \mathbb{D}_{\mathrm{q}}\mathbb{P} & -\mathbb{D}_{\mathrm{q}}\mathbb{P} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{\omega}_t \\ \boldsymbol{\omega}_{t-1} \end{bmatrix} \end{aligned} \tag{50}$$

where $\boldsymbol{\omega}_t \equiv (\boldsymbol{x}_t, \boldsymbol{y}_t)$. Hence, the Jacobian of F_{LEAD} is then given by,

$$\begin{aligned} \nabla F_{\text{LEAD}} &= \begin{bmatrix} \mathbb{I}_{2n} & 0 \\ \mathbb{I}_{2n} & 0 \end{bmatrix} + \beta \begin{bmatrix} \mathbb{I}_{2n} & -\mathbb{I}_{2n} \\ 0 & 0 \end{bmatrix} - \eta \begin{bmatrix} \nabla \boldsymbol{v} & 0 \\ 0 & 0 \end{bmatrix} - \alpha \begin{bmatrix} \mathbb{D}_{\mathrm{q}}\mathbb{P} & -\mathbb{D}_{\mathrm{q}}\mathbb{P} \\ 0 & 0 \end{bmatrix} \\ &= \begin{bmatrix} (1+\beta)\,\mathbb{I}_{2n} - \eta\nabla\boldsymbol{v} - \alpha\mathbb{D}_{\mathrm{q}}\mathbb{P} & -\beta\mathbb{I}_{2n} + \alpha\mathbb{D}_{\mathrm{q}}\mathbb{P} \\ \mathbb{I}_{2n} & 0 \end{bmatrix} \end{aligned} \tag{51}$$

It is to be noted that, for games of the form of Eq. equation 35, we specifically have,

$$\nabla\boldsymbol{v} = \mathbb{D}_{\mathrm{q}}\mathbb{P} + h\mathbb{I}_{2n}$$

and,

$$\text{off-diag}[\nabla\boldsymbol{v}] = \mathbb{D}_{\mathrm{q}}\mathbb{P}$$

Therefore, Eq. equation 51 becomes,

$$\nabla F_{\text{LEAD}} = \begin{bmatrix} (1+\beta-\eta h)\,\mathbb{I}_{2n} - (\eta+\alpha)\,\mathbb{D}_{\mathrm{q}}\mathbb{P} & -\beta\mathbb{I}_{2n} + \alpha\mathbb{D}_{\mathrm{q}}\mathbb{P} \\ \mathbb{I}_{2n} & 0 \end{bmatrix} \tag{52}$$

We next proceed to study the eigenvalues of this matrix which will determine the convergence properties of LEAD around the Nash equilibrium. Using Lemma 1 of (Gidel et al., 2019), we can then write the characteristic polynomial of ∇F_{LEAD} as,

$$\begin{aligned} &\det\left(X\mathbb{I}_{4n} - \nabla F_{\text{LEAD}}\right) = 0 \\ \Rightarrow\ &\det\left(\begin{bmatrix} (X-1)\,\mathbb{I}_{2n} - (\beta-\eta h)\,\mathbb{I}_{2n} + (\eta+\alpha)\,\mathbb{D}_{\mathrm{q}}\mathbb{P} & \beta\mathbb{I}_{2n} - \alpha\mathbb{D}_{\mathrm{q}}\mathbb{P} \\ -\mathbb{I}_{2n} & X\mathbb{I}_{2n} \end{bmatrix}\right) = 0 \\ \Rightarrow\ &\det\left(\left[(X-1)\,(X-\beta)\,\mathbb{I}_{2n} + X\eta h\mathbb{I}_{2n} + (X\eta + X\alpha - \alpha)\,\mathbb{D}_{\mathrm{q}}\mathbb{P}\right]\right) = 0 \\ \Rightarrow\ &\det\left(\left[((X-1)\,(X-\beta) + X\eta h)\,\mathbf{U}\mathbf{U}^{-1} + (X\eta + X\alpha - \alpha)\,\mathbf{U}\Lambda\mathbf{U}^{-1}\right]\right) = 0 \\ \Rightarrow\ &\det\left(\left[((X-1)\,(X-\beta) + X\eta h)\,\mathbb{I}_{2n} + (X\eta + X\alpha - \alpha)\,\Lambda\right]\right) = 0 \\ \Rightarrow\ &\prod_{i=1}^{2n}\left[(X-1)\,(X-\beta) + X\eta h + (X\eta + \alpha\,(X-1))\,\lambda_i\right] = 0 \end{aligned} \tag{53}$$

Where, in the above, we have performed an eigenvalue decomposition of $\mathbb{D}_{\mathrm{q}}\mathbb{P} = \mathbf{U}\Lambda\mathbf{U}^{-1}$. Therefore,

$$\begin{aligned} &X^2 - X\,(1-(\eta+\alpha)\lambda_i + \beta - \eta h) + \beta - \alpha\lambda = 0,\ \ \lambda_i \in \mathrm{Sp}(\mathbb{D}_{\mathrm{q}}\mathbb{P}) \\ &\Rightarrow X^{(i)} \equiv \mu_{\pm}^{(i)} = \frac{1-(\eta+\alpha)\lambda_i + \beta - \eta h \pm \sqrt{\Delta}}{2} \end{aligned} \tag{54}$$

with,

$$\Delta = \left(1-(\eta+\alpha)\,\lambda_i + \beta - \eta h\right)^2 - 4\,(\beta - \alpha\lambda_i) \tag{55}$$

Furthermore for $h, \eta, |\beta|, |\alpha| << 1$, we can approximate the above roots to be,

$$\begin{aligned} \mu_+^{(i)}(\alpha,\beta,\eta) &\approx 1-\eta h+\frac{(\eta+\alpha)^2\lambda_i^2+\eta^2h^2+\beta^2-2\eta h\beta}{4}+\lambda_i\left(\frac{\eta+\alpha}{2}(\eta h-\beta)-\eta\right) \\ \mu_-^{(i)}(\alpha,\beta,\eta) &\approx \beta-\frac{(\eta+\alpha)^2\lambda_i^2+\eta^2h^2+\beta^2-2\eta h\beta}{4}+\lambda_i\left(\frac{\eta+\alpha}{2}(\beta-\eta h)-\alpha\right) \end{aligned} \tag{56}$$

□

E Proof of Proposition 3

Proposition. *For any* $\lambda \in Sp(\textit{off-diag}[\nabla \boldsymbol{v}(\omega^*)])$,

$$\nabla_\alpha \rho(\lambda)\big|_{\alpha=0} < 0 \Leftrightarrow \eta \in \left(0, \frac{2}{\mathrm{Im}(\lambda_{max})}\right), \tag{57}$$

where $\mathrm{Im}(\lambda_{max})$ *is the imaginary component of the largest eigenvalue* λ_{max}*.*

We observe from Proposition 3 above that for $h, \eta, |\alpha|, |\beta| << 1$,

$$\begin{aligned} \rho(\alpha,\eta,\beta) &:= \max\{|\mu_+^{(i)}|^2, |\mu_-^{(i)}|^2\}\ \forall\ i \\ &= \max\{\left|\mu_+^{(i)}\right|^2\}\ \forall\ i \end{aligned} \tag{58}$$

$$\begin{aligned} \therefore \nabla_\alpha\rho\big|_{\alpha=0} &\approx \max\Big\{\frac{\eta^2|\lambda_i|^2-\eta^2h^2-\beta^2}{4}\eta|\lambda_i|^2+\frac{\eta h\beta-(\eta h-\beta)^2}{2}\eta|\lambda_i|^2 \\ &\quad -(1+\beta)\eta|\lambda_i|^2\Big\}\ \forall\ i \\ &\approx \max\Big\{\frac{\eta^3}{4}|\lambda_i|^4-\left(1+\beta+\frac{3\beta^2}{4}\right)\eta|\lambda_i|^2\Big\}\ \forall\ i \\ &< \max\Big\{\left(\frac{\eta^2}{4}|\lambda_i|^2-1\right)\eta|\lambda_i|^2\Big\}\ \forall\ i \end{aligned} \tag{59}$$

where we have retained only terms up to cubic-order in $\eta, |\beta|$ and h. Hence, choosing $\eta \in \left(0, \frac{2}{\mathrm{Im}(\lambda_{\max})}\right)$, ensures:

$$\nabla_\alpha\rho\big|_{\alpha=0} < 0\ \forall\ i, \tag{60}$$

We thus posit, that a choice of a positive α causes the norm of the limiting eigenvalue μ_+ of F_{LEAD} to decrease.

F Proof of Theorem 3

Theorem. *If we set* $\eta = \alpha = \frac{2}{\sigma_{max}(\mathbb{A})}$*, then we have* $\forall\ \epsilon > 0$,

$$\Delta_{t+1} \in \mathcal{O}\left(\left(1-6\frac{2\sigma_{\min}^2+h^2}{\sigma_{\max}^2}-2h\frac{2+\beta}{\sigma_{\max}}+\frac{\beta^2}{2}\right)^t\Delta_0\right) \tag{61}$$

where $\sigma_{max}(\sigma_{min})$ *is the largest (smallest) singular value of* $\mathbb{A}$*,* $\Delta_{t+1} := ||\boldsymbol{\omega}_{t+1}-\boldsymbol{\omega}^*||_2^2+||\boldsymbol{\omega}_t-\boldsymbol{\omega}^*||_2^2$.

Proof: From Eq. (54), we recall that the eigenvalues of $\nabla F_{\text{LEAD}}(\boldsymbol{\omega}^*)$ for the quadratic game are,

$$\mu_\pm^{(i)}(\alpha,\beta,\eta) = \frac{(1-(\alpha+\eta)\lambda_i+\beta-\eta h)}{2}\left(1\pm\sqrt{1-\frac{4(\beta-\eta\lambda_i)}{(1-(\alpha+\eta)\lambda_i+\beta-\eta h)^2}}\right) \tag{62}$$

with $\lambda_i \in \mathrm{Sp}(\text{off-diag}[\nabla \boldsymbol{v}(\omega^*)])$. Now, since in the quadratic-game setting considered, we have,

$$\text{off-diag}[\nabla \boldsymbol{v}(\omega^*)] = \mathbb{D}_{\mathrm{q}}\mathbb{P} = \begin{bmatrix} 0 & \mathbb{A} \\ -\mathbb{A}^T & 0 \end{bmatrix} \tag{63}$$

hence, $\lambda_i = \pm i\sigma_i$ with σ_i being the singular values of $\mathbb{A}$. This, then allows us to write,

$$\mu_{\pm}^{(i)}(\alpha, \beta, \eta) = \frac{(1 - (\alpha + \eta)(\pm i\sigma_i) + \beta)}{2} \left(1 \pm \sqrt{1 - \frac{4\,(\beta - \alpha(\pm i\sigma_i))}{(1 - (\alpha + \eta)(\pm i\sigma_i) + \beta)^2}}\right) \tag{64}$$

Now, according to Proposition 2, the convergence behavior of LEAD is determined as, $\Delta_{t+1} \le \mathcal{O}(\rho + \epsilon)\Delta_t \; \forall \; \epsilon > 0$, where (setting $\eta = \alpha$),

$$\begin{aligned} \rho &:= \max\{|\mu_+^{(i)}|^2, |\mu_-^{(i)}|^2\} \; \forall \; i \\ &= |\mu_+^{(i)}|^2 \; \forall \; i \\ &\approx 1 - 3\eta^2\sigma_{\min}^2 - \frac{\beta^2}{2} - \frac{3}{2}\eta^2 h^2 + (2 + \beta)\,\eta h \\ &\equiv 1 - r_{\mathrm{LEAD}} \end{aligned} \tag{65}$$

Here, $r_{\mathrm{LEAD}} = 3\eta^2\sigma_{\min}^2 - \frac{\beta^2}{2} - \frac{3}{2}\eta^2 h^2 + (2 + \beta)\,\eta h$, is defined to be the *rate of convergence* of LEAD. Furthermore, using the largest learning rate η as prescribed by Proposition 3, in the above, we find,

$$r_{\mathrm{LEAD}} = 6\frac{2\sigma_{\min}^2 - h^2}{\sigma_{\max}^2} + 2h\frac{2 + \beta}{\sigma_{\max}} - \frac{\beta^2}{2} \tag{66}$$

Therefore,

$$\begin{aligned} \Delta_{t+1} &\le \mathcal{O}\left((1 - r_{\mathrm{LEAD}})^t\,\Delta_0\right) \\ &= \mathcal{O}\left(\left(1 - 6\frac{2\sigma_{\min}^2 + h^2}{\sigma_{\max}^2} - 2h\frac{2 + \beta}{\sigma_{\max}} + \frac{\beta^2}{2}\right)^t \Delta_0\right) \end{aligned} \tag{67}$$

where $\Delta_{t+1} := ||\boldsymbol{\omega}_{t+1} - \boldsymbol{\omega}^*||_2^2 + ||\boldsymbol{\omega}_t - \boldsymbol{\omega}^*||_2^2$.

G LEAD-Adam

Since Adam algorithm is commonly used in large-scale experiments, we extend LEAD to be used with the Adam algorithm.

Algorithm 2 Least Action Dynamics Adam (LEAD-Adam)

```
Input: learning rate $\eta$, momentum $\beta$, coupling coefficient $\alpha$.
Initialize: $x_0 \leftarrow x_{init}$, $y_0 \leftarrow y_{init}$, $t \leftarrow 0$, $m_0^x \leftarrow 0$, $v_0^x \leftarrow 0$ $m_0^y \leftarrow 0$, $v_0^y \leftarrow 0$
while not converged do
    $t \leftarrow t + 1$
    $g_x \leftarrow \nabla_x f(x_t, y_t)$
    $g_{xy}\Delta y \leftarrow \nabla_y(g_x)(y_t - y_{t-1})$
    $g_t^x \leftarrow g_{xy}\Delta y + g_x$
    $m_t^x \leftarrow \beta_1.m_{t-1}^x + (1 - \beta_1).g_t^x$
    $v_t^x \leftarrow \beta_2.v_{t-1}^x + (1 - \beta_2).(g_t^x)^2$
    $\hat{m}_t \leftarrow m_t/(1 - \beta_1^t)$
    $\hat{v}_t \leftarrow v_t/(1 - \beta_2^t)$
    $x_{t+1} \leftarrow x_t - \eta\ \hat{m}_t/(\sqrt{\hat{v}_t} + \epsilon)$
    $g_y \leftarrow \nabla_y f(x_{t+1}, y_t)$
    $g_{xy}\Delta x \leftarrow \nabla_x(g_y)(x_{t+1} - x_t)$
    $g_t^y \leftarrow g_{xy}\Delta x + g_y$
    $m_t^y \leftarrow \beta_1.m_{t-1}^y + (1 - \beta_1).g_t^y$
    $v_t^y \leftarrow \beta_2.v_{t-1}^y + (1 - \beta_2).(g_t^y)^2$
    $\hat{m}_t^y \leftarrow m_t^y/(1 - \beta_1^t)$
    $\hat{v}_t^y \leftarrow v_t^y/(1 - \beta_2^t)$
    $y_{t+1} \leftarrow y_t + \eta\ \hat{m}_t^y/(\sqrt{\hat{v}_t^y} + \epsilon)$
end while
return $(x, y)$
```

H 8-Gaussians Generation

We compare our method LEAD-Adam with vanilla-Adam (Kingma & Ba, 2014) on the generation task of a mixture of 8-Gaussians. Standard optimization algorithms such as vanilla-Adam suffer from mode collapse in this simple task, implying the generator cannot produce samples from one or several of the distributions present in the real data. Through Figure 6, we demonstrate that LEAD-Adam fully captures all the modes in the real data in both saturating and non-saturating losses.

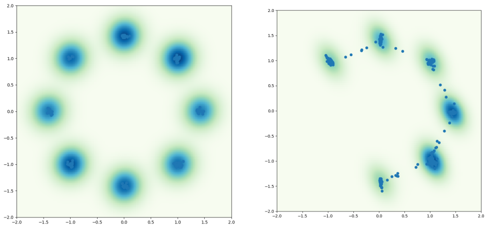
I Experiments and Implementation Details

I.1 Mixture of Eight Gaussians

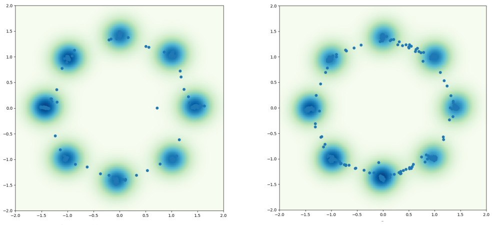
Dataset The real data is generated by 8-Gaussian distributions their mean are uniformly distributed around the unit circle and their variance is 0.05. The code to generate the data is included in the source code.

Architecture The architecture for Generator and Discriminator, each consists of four layers of affine transformation, followed by ReLU non-linearity. The weight initialization is default PyTorch's initialization scheme. See a schematic of the architecture in Table 2.

Other Details We use the Adam (Kingma & Ba, 2014) optimizer on top of our algorithm in the reported results. Furthermore, we use batchsize of 128.



Real Data Adam Non-Saturating Loss



LEAD Saturating Loss LEAD Non-Saturating Loss

Figure 6: Performance of LEAD-Adam on the generation task of 8-Gaussians. All samples are shown after 10k iterations. Samples generated using Adam exhibit mode collapse, while LEAD-Adam does not suffer from this issue.

Table 2: Architecture used for the Mixture of Eight Gaussians.

Generator	**Discriminator**
Input: $z \in \mathbb{R}^{64} \sim \mathcal{N}(0, I)$	*Input:* $x \in \mathbb{R}^{2}$
Linear (64 → 2000)	Linear (2 → 2000)
ReLU	ReLU
Linear (2000 → 2000)	Linear (2000 → 2000)
ReLU	ReLU
Linear (2000 → 2000)	Linear (2000 → 2000)
ReLU	ReLU
Linear (2000 → 2)	Linear (2000 → 1)

I.2 CIFAR 10 DCGAN

Dataset The CIFAR10 dataset is available for download at the following link; `https://www.cs.toronto.edu/~kriz/cifar.html`

Architecture The discriminator has four layers of convolution with LeakyReLU and batch normalization. Also, the generator has four layers of deconvolution with ReLU and batch normalization. See a schematic of the architecture in Table 3.

Table 3: Architecture used for CIFAR-10 DCGAN.

Generator	**Discriminator**
Input: $z \in \mathbb{R}^{100} \sim \mathcal{N}(0, I)$	*Input:* $x \in \mathbb{R}^{3\times32\times32}$
conv. (ker: 4×4, 100 → 1024; stride: 1; pad: 0)	conv. (ker: 4×4, 3 → 256; stride: 2; pad: 1)
Batch Normalization	LeakyReLU
ReLU	conv. (ker: 4×4, 256 → 512; stride: 2; pad: 1)
conv. (ker: 4×4, 1024 → 512; stride: 2; pad: 1)	Batch Normalization
Batch Normalization	LeakyReLU
ReLU	conv. (ker: 4×4, 512 → 1024; stride: 2; pad: 1)
conv. (ker: 4×4, 512 → 256; stride: 2; pad: 1)	Batch Normalization
Batch Normalization	LeakyReLU
ReLU	conv. (ker: 4×4, 1024 → 1; stride: 1; pad: 0)
conv. (ker: 4×4, 256 → 3; stride: 2; pad: 1)	
Tanh	Sigmoid

Other Details For the baseline we use Adam with β_1 set to 0.5 and β_2 set to 0.99. Generator's learning rate is 0.0002 and discriminator's learning rate is 0.0001. The same learning rate and momentum were used to train LEAD model. We also add the mixed derivative term with $\alpha_d = 0.3$ and $\alpha_g = 0.0$.

The baseline is a DCGAN with the standard non-saturating loss (non-zero sum formulation). In our experiments, we compute the FID based on 50,000 samples generated from our model vs 50,000 real samples.

Samples

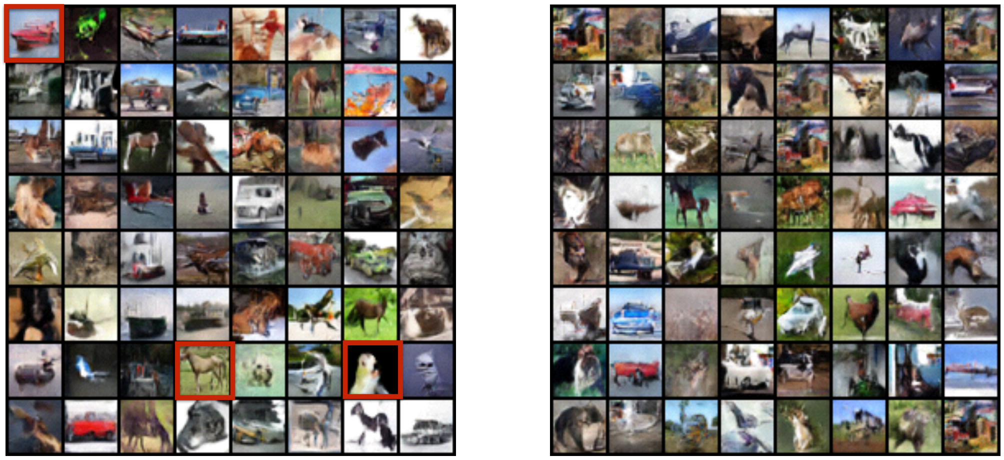


Figure 7: Performance of LEAD on CIFAR-10 image generation task on a DCGAN architecture. **Left**: LEAD achieves FID 19.27. **Right**: Vanilla Adam achieves FID 24.38. LEAD is able to generate better sample qualities from several classes such as ships, horses and birds (red). Best performance is reported after 100 epochs.

I.3 CIFAR 10 ResNet

Dataset The CIFAR10 dataset is available for download at the following link; `https://www.cs.toronto.edu/~kriz/cifar.html`

Architecture See Table 5 for a schematic of the architecture used for the CIFAR10 experiments with ResNet.

Table 4: ResNet blocks used for the ResNet architectures (see Table 5).

Gen–Block
Shortcut:
Upsample(×2)
Residual:
Batch Normalization
ReLU
Upsample(×2)
conv. (ker: 3×3, 256 → 256; stride: 1; pad: 1)
Batch Normalization
ReLU
conv. (ker: 3×3, 256 → 256; stride: 1; pad: 1)

Dis–Block
Shortcut:
downsample
conv. (ker: 1×1, $3_{\ell=1}/128_{\ell\neq 1} \rightarrow 128$; stride: 1)
Spectral Normalization
[AvgPool (ker:2×2, stride:2)], if $\ell \neq 1$
Residual:
[ReLU], if $\ell \neq 1$
conv. (ker: 3×3, $3_{\ell=1}/128_{\ell\neq 1} \rightarrow 128$; stride: 1; pad: 1)
Spectral Normalization
ReLU
conv. (ker: 3×3, 128 → 128; stride: 1; pad: 1)
Spectral Normalization
AvgPool (ker:2×2)

Other Details The baseline is a ResNet with non-saturating loss (non-zero sum formulation). Similar to (Miyato et al., 2018), for every time that the generator is updated, the discriminator is updated 5 times. For both the Baseline SNGAN and LEAD-Adam we use a β_1 of 0.0 and β_2 of 0.9 for Adam. Baseline SNGAN

Table 5: ResNet architectures used for experiments on CIFAR10.

Generator	Discriminator
Input: $z \in \mathbb{R}^{64} \sim \mathcal{N}(0, I)$	*Input:* $x \in \mathbb{R}^{3\times 32\times 32}$
Linear(64 → 4096)	D–ResBlock
G–ResBlock	D–ResBlock
G–ResBlock	D–ResBlock
G–ResBlock	D–ResBlock
Batch Normalization	ReLU
ReLU	AvgPool (ker:8×8)
conv. (ker: 3×3, 256 → 3; stride: 1; pad:1)	Linear(128 → 1)
$Tanh(\cdot)$	Spectral Normalization

uses a learning rate of 0.0002 for both the generator and the discriminator. LEAD-Adam also uses a learning rate of 0.0002 for the generator but 0.0001 for the discriminator. LEAD-Adam uses an α of 0.5 and 0.01 for the generator and the discriminator respectively. Furthermore, we evaluate both the baseline and our method on an exponential moving average of the generator's parameters.

In our experiments, we compute the FID based on 50,000 samples generated from our model vs 50,000 real samples and reported the mean and variance over 5 random runs. We have provided pre-trained models as well as the source code for both LEAD-Adam and Baseline SNGAN in our GitHub repository.

Samples

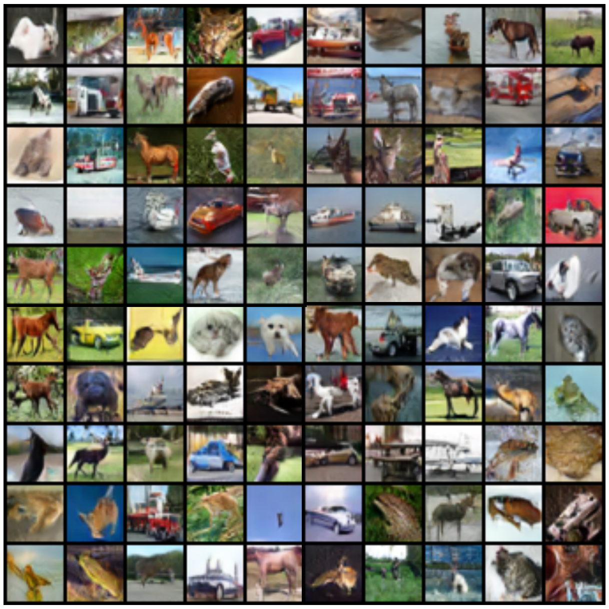


Figure 8: Generated sample of LEAD-Adam on CIFAR-10 after 50k iterations on a ResNet architecture. We achieve an FID score of 10.49 using learning rate $2e-4$ for the generator and the discriminator, α for the generator is set to 0.01 and for the discriminator is set to 0.5.

J Comparison to other methods

In this section we compare our method with several other second order methods in the min-max setting.

Table 6: Comparison of several second-order methods in min-max optimization. Each update rule, corresponding to a particular row, can be constructed by adding cells in that row from Columns 4 to 7 and then multiplying that by the value in Column 1. Furthermore, $\Delta x_{k+1} = x_{k+1} - x_k$, while $\mathcal{C} = \left(\mathrm{I} + \eta^2 \nabla^2_{xy} f \nabla^2_{yx} f\right)$. We compare the update rules of the first player[4] for the following methods: Gradient Descent-Ascent (GDA), Least Action Dynamics (LEAD, ours), Symplectic Gradient Adjustment (SGA), Competitive Gradient Descent (CGD), Consensus Optimization (CO), Follow-the-Ridge (FtR) and Learning with Opponent Learning Awareness (LOLA), in a zero-sum game.

		Coefficient	Momentum	Gradient	Interaction-xy	Interaction-xx
GDA	$\Delta x_{k+1} =$	1	0	$-\eta \nabla_x f$	$-\eta \nabla_x f$	0
LEAD	$\boldsymbol{\Delta \mathbf{x}_{\mathbf{k+1}}} =$	$\mathbf{1}$	$\beta \boldsymbol{\Delta} \mathbf{x_k}$	$-\eta \nabla_{\mathbf{x}} \mathbf{f}$	$-\alpha \nabla^{\mathbf{2}}_{\mathbf{xy}} \mathbf{f} \boldsymbol{\Delta} \mathbf{y_k}$	0
SGA(6)	$\Delta x_{k+1} =$	1	0	$-\eta \nabla_x f$	$-\eta\gamma \nabla^2_{xy} f \nabla_y f$	0
CGD(52)	$\Delta x_{k+1} =$	$\mathcal{C}^{-1}$	0	$-\eta \nabla_x f$	$-\eta^2 \nabla^2_{xy} f \nabla_y f$	0
CO(39)	$\Delta x_{k+1} =$	1	0	$-\eta \nabla_x f$	$-\eta\gamma \nabla^2_{xy} f \nabla_y f$	$-\eta\gamma \nabla^2_{xx} f \nabla_x f$
FtR(56)	$\Delta y_{k+1} =$	1	0	$\eta_y \nabla_y f$	$\eta_x \left(\nabla^2_{yy} f\right)^{-1} \nabla^2_{yx} f \nabla_x f$	0
LOLA(15)	$\Delta x_{k+1} =$	1	0	$-\eta \nabla_x f$	$-2\eta\alpha \nabla_{xy} f \nabla_y f$	0

The distinction of LEAD from SGA and LookAhead, can be understood by considering the 1st-order approximation of $x_{k+1} = x_k - \eta \nabla_x f\left(x_k, y_k + \eta \Delta y_k\right)$, where $\Delta y_k = \eta \nabla_y f\left(x_k + \eta \Delta x, y_k\right)$.

This gives rise to:

$$x_{k+1} = x_k - \eta \nabla_x f\left(x_k, y_k\right) - \eta^2 \nabla^2_{xy} f\left(x_k, y_k\right) \Delta y, \tag{68}$$

$$y_{k+1} = y_k + \eta \nabla_y f\left(x_k, y_k\right) + \eta^2 \nabla^2_{xy} f\left(x_k, y_k\right) \Delta x, \tag{69}$$

with $\Delta x, \Delta y$ corresponding to each player accounting for its opponent's potential next step. However, SGA and LookAhead additonally *model* their opponent as *naive* learners i.e. $\Delta x = -\nabla_x f(x_k, y_k)$, $\Delta y = \nabla_y f(x_k, y_k)$. On the contrary, our method does away with such specific assumptions, instead modeling the opponent based on its most recent move.

Furthermore, there is a resemblance between LEAD and OGDA that we would like to address. The 1st order Taylor expansion of the difference in gradients term of OGDA yields the update (for x):

$$x_{k+1} = x_k - \eta \nabla_x f - \eta^2 \nabla^2_{xy} f \nabla_y f + \eta^2 \nabla^2_{xx} f \nabla_x f, \tag{70}$$

which contains an extra 2nd order term $\nabla^2_{xx} f$ compared to ours. As noted in Schäfer & Anandkumar (2019), the $\nabla^2_{xx} f$ term does not systematically aid in curbing the min-max rotations, rather causing convergence to non-Nash points in some settings. For e.g., let us consider the simple game $f(x, y) = \gamma(x^2 - y^2)$, where x, y, γ are all scalars, with the Nash equilibrium of this game located at $(x^* = 0, y^* = 0)$. For a choice of $\gamma \geq 6$, OGDA fails to converge for any learning rate while methods like LEAD, Gradient Descent Ascent (GDA) and CGD (Schäfer & Anandkumar (2019)) that do not contain the $\nabla_{xx} f(\nabla_{yy} f)$ term do exhibit convergence. See Figure 9 and Schäfer & Anandkumar (2019) for more discussion.

---

[4] For FtR, we provide the update for the second player given the first player performs gradient descent. Also note that in this table SGA is simplified for the two player zero-sum game. Non-zero sum formulation of SGA such as the one used for GANs require the computation of $\mathbb{J}\mathbf{v}, \mathbb{J}^\top \mathbf{v}$.

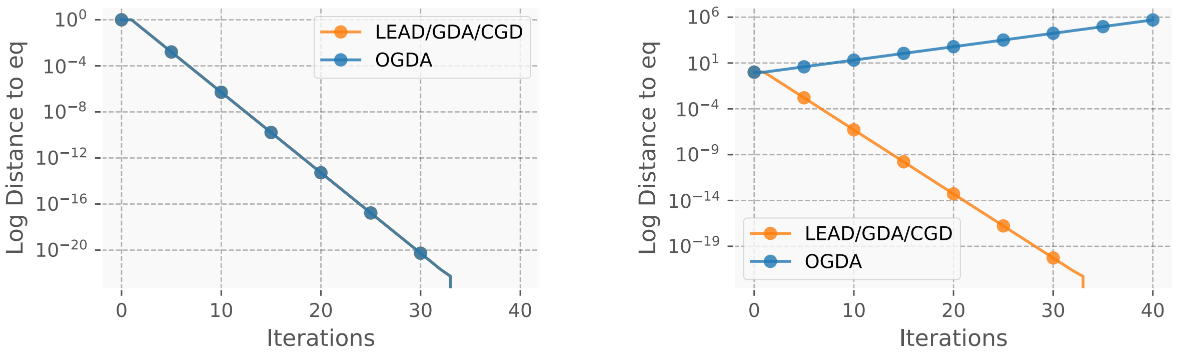


Figure 9: Figure depicting the convergence/divergence of several algorithms on the game of $f(x, y) = \gamma(x^2 - y^2)$ (Nash equilibrium at $x^* = 0, y^* = 0$). **Left**: For $\gamma = 1$, OGDA and LEAD/GDA/CGD (overlaying) are found to converge to the Nash eq. **Right**: For $\gamma = 6$, we find that OGDA fails to converge while LEAD/GDA/CGD (overlaying) converge. We conjecture that the reason behind this observation is the existence of $\nabla^2_{xx} f$ term in the optimization algorithm of OGDA.