

LEAD: Least-Action Dynamics for Min-Max Optimization

Reyhane Askari Hemmat^{* 1 2} Amartya Mitra^{† * 3} Guillaume Lajoie^{1 2 4} Ioannis Mitliagkas^{1 2 4}

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. Existing methods typically employ intuitive, carefully hand-designed mechanisms for controlling such rotations. In this paper, we take a novel approach to address this issue by casting min-max optimization as a physical system. We leverage tools from physics to introduce LEAD (Least-Action Dynamics), a second-order 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 bilinear 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 over baseline methods.

1. Introduction

Much of the advances in traditional machine learning can be attributed to the success of gradient-based methods. Modern machine learning formulations 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. In these *game* settings, best practices and methods developed for single-objective optimization perform poorly (Mescheder et al., 2017; Balduzzi et al., 2018b; Gidel et al., 2019). Notably, they exhibit rotational dynamics about the *Nash Equilibria* (Mescheder et al., 2017), which can slow 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 intuitively introducing additional second-order terms in the optimization algorithm, helps to suppress these rotations, thereby improving convergence. Despite their relative success in many settings, further improvements require a systematic understanding of games dynamics.

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. By likening the gradient-based optimization of two-player (zero-sum) games to the dynamics of a particular physical system, we introduce new forces that help curb the rotations. We then leverage the principle of least action from physics to discover an efficient (ordinary differential equation) ODE for game optimization. Finally, we discretize these continuous-time dynamics to derive our novel second-order game optimization algorithm, *Least Action Dynamics* (LEAD).

Next, by using Lyapunov and spectral analysis, we demonstrate linear convergence of our optimizer (LEAD) in both continuous and discrete-time settings for the bilinear min-max game. In terms of empirical performance, LEAD achieves an FID score of 11.2 on CIFAR-10 image generation, outperforming existing methods 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 games, such as (Balduzzi et al., 2018b; Mescheder et al., 2017; Wang et al., 2019; Mazumdar et al., 2019; Schäfer & Anandkumar, 2019) is its computational complexity. While all these methods, including ours, involve Jacobian (or inverse-Jacobian) vector-product computation, ours rely on computing only *one-block* of the full Jacobian of the gradient vector-field multiplied by a vector. This property makes our method significantly cheaper and comparable to several first-order methods, as we show in §5.1. Furthermore, methods involving Jacobian inverses are intractable in real-world problems such as GANs. In light of this, (Wang et al., 2019; Schäfer & Anandkumar, 2019) propose a conjugate gradient approximation of the inverse to reduce computational

^{*}Equal contribution ¹Mila, ²Université de Montréal,
³University of California, Riverside, ⁴Canada CIFAR AI Chair. Correspondence to: Reyhane Askari Hemmat <reyhane.askari.hemmat@umontreal.ca>, Amartya Mitra <amitr003@ucr.edu>.

[†] Work done while at Mila.

cost for example. Nonetheless, their provided proofs of convergence rely on the expensive exact computation of the inverse. In our experimental comparisons we use the proposed approximation but still observe that these methods are slow for models with many parameters, such as neural networks. We summarize our contributions below:

- In §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 derive our optimization scheme, Least Action Dynamics (LEAD).
- In §4, we use Lyapunov stability theory and spectral analysis to prove a linear convergence of LEAD in continuous and discrete-time settings for bilinear min-max games.
- Finally, in §5, 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. Furthermore, we achieve a competitive FID score of 11.22 for CIFAR-10 on a ResNet architecture.
- The source code for all the experiments is available at https://github.com/ReyhaneAskari/Least_action_dynamics_minmax.

2. Problem Setting

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

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

$$\min_X \max_Y f(X, Y), \quad (1)$$

where $f : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}$, and is assumed to be convex-concave function which is continuous and twice differentiable w.r.t. $X, Y \in \mathbb{R}$. In developing our framework below, we assume X, Y to be scalars. Nevertheless, as we later demonstrate both analytically (Appendix D, E) and empirically, our theoretical analysis are found to hold for the more general case of vectorial X and Y .

3. Optimization Mechanics

In our attempt to find an efficient update scheme or trajectory to optimize the min-max objective $f(X, Y)$, we note from classical physics the following: under the influence of a net force F , the trajectory of motion of a physical object of mass m , is determined by Newton's 2nd Law,

$$m\ddot{X} = F, \quad (2)$$

with the object's coordinate expressed as $X_t \equiv X$. As stated by the *principle of least action*¹ (Landau & Lifshitz, 1960), nature “selects” the particular trajectory of motion obtained from solving Eq.(2), over other possibilities as a quantity called *action* is extremized along it.

Hence, the ability to model our game optimization task in terms of an object moving under a force allows for a discovery of an efficient optimization path through the least action principle, according to Eq.(2). Regarding how such modeling may be performed, we take inspiration from Polyak's heavy-ball momentum (Polyak, 1964) method² in single objective minimization of an objective $f(x)$,

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

which in continuous-time translates to (see Appendix A),

$$m\ddot{X} = -\nabla_X f(X). \quad (4)$$

Comparing Eqns.(4) and (2), we notice that in this case $F = -\nabla_X f(X)$, i.e. $f(X)$ acts 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 $f(X)$ to reach the minimum.

Armed with this observation, we notice that a straightforward extension of Eq.(4) to two-dimensions³ is the following generalization which closely⁴ resembles Gradient Descent-Ascent (GDA),

$$\begin{aligned} m\ddot{X} &= -\nabla_X f(X, Y) \\ m\ddot{Y} &= \nabla_Y f(X, Y). \end{aligned} \quad (5)$$

This system corresponds to the equations of motion (Eq.(2)) of an object moving under a *curl force* (Berry & Shukla, 2016): $\mathbf{F}_{\text{curl}} = (-\nabla_X f, \nabla_Y f)$ in the 2-dimensional $X - Y$ plane. To investigate the nature of the above trajectory Eq.(5), we consider the prototypical min-max objective, $f(X, Y) = XY$. In this bilinear setting though, the object

¹Also referred to as the Principle of Stationary Action.

²Arbitrary momentum coefficient results in incorporating friction in the equivalent physical system.

³Corresponds to *single*-objective minimization w.r.t. two variables.

⁴To be precise, it corresponds to gradient descent-ascent with the momentum of 1.

with mass m , is found to spin away from the origin (Nash Equilibrium) over time. This corroborates with the fact that the GDA with a positive momentum term, resultant from the discretization of Eq.(5), in the bilinear setting is known to be divergent (Gidel et al., 2019).

Drawing insight from the above particular game choice, we note that the system with a curl force (Eq.(5)) by itself does not seem to hold much promise in regards to providing an (efficient) optimization trajectory converging to the Nash Equilibrium. The physical nature of the formulation allows us to select from the extensive set of other physical forces to be systematically added to the right-hand side of Eq.(5) to produce the desired convergence behavior. To this end, we consider the simplest force known to produce rotatory motion on a particle of charge q : a magnetic force,

$$\mathbf{F}_{\text{mag}} = q\dot{\mathbf{X}} \times \mathbf{B} = q\dot{\mathbf{X}} \times (\nabla \times \mathbf{A}). \quad (6)$$

Here, \mathbf{A} is the magnetic *vector potential* generating a magnetic field \mathbf{B} as, $\mathbf{B} = \nabla \times \mathbf{A}$ (Landau et al., 2013). In order to make a prudent choice of \mathbf{F}_{mag} , we observe that desirable counter-rotations are incorporated if \mathbf{A} is itself chosen to be a rotating vector field, similar to the type of \mathbf{F}_{curl} . Specifically, since we are attempting to counteract or negate the rotational effects of \mathbf{F}_{curl} , we choose $\mathbf{A} = -\mathbf{F}_{\text{curl}}$. This results in,

$$\mathbf{F}_{\text{mag}} = \left(-2q(\nabla_{XY}f)\dot{Y}, 2q(\nabla_{XY}f)\dot{X} \right). \quad (7)$$

We add a final ingredient to our system in the form of a (velocity-dependent) frictional force $\mathbf{F}_{\text{fric}} = (\mu\dot{X}, \mu\dot{Y})$, with μ being the friction coefficient. The reason behind adding friction stems from the fact that a) \mathbf{F}_{curl} is known to increase the particle's speed over time (Berry & Shukla, 2016) while, b) \mathbf{F}_{mag} is known to not cause any change to the speed. Hence, under the influence of simply the curl and magnetic forces, our mass m , charge q particle will keep increasing in speed over time, preventing from converging to a point. It is in this light that a dissipative force such as friction comes of use.

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

$$\begin{aligned} m\ddot{X} &= -\mu\dot{X} - \nabla_X f - 2q(\nabla_{XY}f)\dot{Y}, \\ m\ddot{Y} &= -\mu\dot{Y} + \nabla_Y f + 2q(\nabla_{XY}f)\dot{X}. \end{aligned} \quad (8)$$

Without loss of generality, we hereon assume our physical object to be of mass $m = 1$.

3.1. Discretization

With the continuous-time trajectory of Eq.(8) in hand, we now proceed discretize it.

In the following, we make use of both Euler's implicit and explicit discretization schemes to discretize $\dot{X} = V_X$, (δ : discretization step-size, k : iteration step):

$$\begin{aligned} \text{Implicit : } x_{k+1} - x_k &= \delta v_{k+1}^x \\ \text{Explicit : } x_{k+1} - x_k &= \delta v_k^x. \end{aligned} \quad (9)$$

We simply combine the implicit and explicit steps in a system of continuous-time equations and apply a *symplectic* discretization (Shi et al., 2019) to obtain LEAD. Our symplectic discretization, however, does not preserve the symplectic structure of Eq.(8) and hence is not symplectic in the stricter sense. Furthermore, additional optimization schemes can be obtained using other forms of discretization, see Appendix B.

Proposition 1. *The continuous-time EOMs (8) can be discretized in a symplectic way as,*

$$\begin{aligned} x_{k+1} &= x_k + \beta(x_k - x_{k-1}) - \eta\nabla_x f(x_k, y_k) \\ &\quad - \alpha\nabla_{xy} f(x_k, y_k)(y_k - y_{k-1}), \\ y_{k+1} &= y_k + \beta(y_k - y_{k-1}) + \eta\nabla_y f(x_k, y_k) \\ &\quad + \alpha\nabla_{yx} f(x_k, y_k)(x_k - x_{k-1}), \end{aligned} \quad (10)$$

where α, β, η are hyper-parameters dependent on μ, q and δ (Proof in Appendix C).

We refer to these discrete update rules as *Least Action Dynamics (LEAD)*. Algorithm 1 details the pseudo-code of LEAD.

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)
3. Coupling term: $-\nabla_{xy} f(x_k, y_k)(y_k - y_{k-1})$ and $\nabla_{yx} f(x_k, y_k)(x_k - x_{k-1})$: 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.

4. Convergence Analysis on bilinear game

We now study the behavior of our method on the bilinear min-max game,

$$f(\mathbf{X}, \mathbf{Y}) = \mathbf{X}^T \mathbb{A} \mathbf{Y} \quad (11)$$

Algorithm 1 Least Action Dynamics (LEAD)

```

1: Input: learning rate  $\eta$ , momentum  $\beta$ , coupling
   coefficient  $\alpha$ .
2: Initialize:  $x_0 \leftarrow x_{init}$ ,  $y_0 \leftarrow y_{init}$ ,  $t \leftarrow 0$ 
3: while not converged do
4:    $t \leftarrow t + 1$ 
5:    $g_x \leftarrow \nabla_x f(x_t, y_t)$ 
6:    $g_{xy} \Delta y_t \leftarrow \nabla_y(g_x)(y_t - y_{t-1})$ 
7:    $x_{t+1} \leftarrow x_t + \beta(x_t - x_{t-1}) - \eta g_x - \alpha g_{xy} \Delta y_t$ 
8:    $g_y \leftarrow \nabla_y f(x_t, y_t)$ 
9:    $g_{xy} \Delta x_t \leftarrow \nabla_x(g_y)(x_t - x_{t-1})$ 
10:   $y_{t+1} \leftarrow y_t + \beta(y_t - y_{t-1}) + \eta g_y + \alpha g_{xy} \Delta x_t$ 
11: end while
12: return  $(x_{t+1}, y_{t+1})$ 

```

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

$$\mathbf{v} = \begin{bmatrix} \nabla_{\mathbf{X}} f(\mathbf{X}, \mathbf{Y}) \\ -\nabla_{\mathbf{Y}} f(\mathbf{X}, \mathbf{Y}) \end{bmatrix} = \begin{bmatrix} \mathbb{A}\mathbf{Y} \\ -\mathbb{A}^T\mathbf{X} \end{bmatrix}. \quad (12)$$

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). The scalar function $\mathcal{E}_t : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}$, is a Lyapunov function of a continuous-time dynamics if $\forall t$,

- (i) $\mathcal{E}_t(\mathbf{X}, \mathbf{Y}) \geq 0$,
- (ii) $\dot{\mathcal{E}}_t(\mathbf{X}, \mathbf{Y}) \leq 0$

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 bilinear min-max game defined in Eq.(11), Eq.(8) generalizes to,

$$\begin{aligned} \ddot{\mathbf{X}} &= -\mu \dot{\mathbf{X}} - \mathbb{A}\mathbf{Y} - q\mathbb{A}\dot{\mathbf{Y}} \\ \ddot{\mathbf{Y}} &= -\mu \dot{\mathbf{Y}} + \mathbb{A}^T\mathbf{X} + q\mathbb{A}^T\dot{\mathbf{X}}, \end{aligned} \quad (13)$$

where we have redefined $2q \rightarrow q$.

Theorem 1. *For the dynamics of Eq.(13),*

$$\begin{aligned} \mathcal{E}_t &= \frac{1}{2} \left(\dot{\mathbf{X}} + \mu\mathbf{X} + \mu\mathbb{A}\mathbf{Y} \right)^T \left(\dot{\mathbf{X}} + \mu\mathbf{X} + \mu\mathbb{A}\mathbf{Y} \right) \\ &\quad + \frac{1}{2} \left(\dot{\mathbf{Y}} + \mu\mathbf{Y} - \mu\mathbb{A}^T\mathbf{X} \right)^T \left(\dot{\mathbf{Y}} + \mu\mathbf{Y} - \mu\mathbb{A}^T\mathbf{X} \right) \\ &\quad + \frac{1}{2} \left(\dot{\mathbf{X}}^T \dot{\mathbf{X}} + \dot{\mathbf{Y}}^T \dot{\mathbf{Y}} \right) + \mathbf{X}^T \mathbb{A} \mathbb{A}^T \mathbf{X} + \mathbf{Y}^T \mathbb{A}^T \mathbb{A} \mathbf{Y} \end{aligned} \quad (14)$$

is a Lyapunov function of the system. Furthermore, by setting $q = (2/\mu) + \mu$, we find $\dot{\mathcal{E}}_t \leq -\rho\mathcal{E}_t$ for $\rho \leq \min \left\{ \frac{\mu}{1+\mu}, \frac{2\mu\sigma_{max}^2}{(1+\sigma_{max}^2)(\mu^2+\mu)+2\sigma_{max}^2} \right\}$ with σ_{max} being the largest singular value of \mathbb{A} . This thereby ensures linear convergence of Eq.13,

$$\|\mathbf{X}\|^2 + \|\mathbf{Y}\|^2 \leq \frac{\mathcal{E}_0}{\sigma_{min}^2} \exp(-\rho t). \quad (15)$$

Proof in Appendix D and E. See Appendix F for a continuous-time Lyapunov convergence analysis of Eq.(8) on a scalar quadratic game.

4.2. Discrete-Time Analysis

In this Section, we next analyze the convergence behavior of LEAD, Eq.(10) in the case of the bilinear game Eq.(11), using spectral analysis,

$$\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{x}_k + \beta \Delta \mathbf{x}_k - \eta \mathbb{A} \mathbf{y}_k - \alpha \mathbb{A} \Delta \mathbf{y}_k \\ \mathbf{y}_{k+1} &= \mathbf{y}_k + \beta \Delta \mathbf{y}_k + \eta \mathbb{A}^T \mathbf{x}_k + \alpha \mathbb{A}^T \Delta \mathbf{x}_k, \end{aligned} \quad (16)$$

where $\Delta \mathbf{x}_k = \mathbf{x}_k - \mathbf{x}_{k-1}$. (We provide a Lyapunov theory based convergence analysis of the implicit discretized counterpart of the continuous time EOMs of Eq.(8), in Appendix G).

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

$$F_\eta(\omega_t) = \omega_t - \eta \mathbf{v}(\omega_{t-1}).$$

Thus the fixed point ω^* of $F_\eta(\omega_t)$ satisfies $F_\eta(\omega^*) = \omega^*$. Furthermore, at ω^* we have,

$$\nabla F_\eta(\omega^*) = \mathbb{I}_n - \eta \nabla \mathbf{v}(\omega^*), \quad (17)$$

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

$$\text{Sp}(\nabla F_\eta(\omega^*)) = \{1 - \eta \lambda \mid \lambda \in \text{Sp}(\nabla \mathbf{v}(\omega^*))\}. \quad (18)$$

Proposition 2 (Prop. 4.4.1 Bertsekas (1999)). *For the spectral radius $\rho_{max} := \rho\{\nabla F_\eta(\omega^*)\} < 1$, and for some ω_0 in a neighborhood of ω^* , the update operator F ensures linear convergence to ω^* at a rate,*

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

where $\Delta_{t+1} := \|\omega_{t+1} - \omega^*\|_2^2 + \|\omega_t - \omega^*\|_2^2$.

It can be noted that since the eigenvalues of $\nabla \mathbf{v}(\omega^*)$ are purely imaginary in a bilinear game, the update operator in Eq.(17) is not convergent.

Next, we proceed to define the update operator of Eq.(10) as $F_{\text{LEAD}}(\omega_t, \omega_{t-1}) = (\omega_{t+1}, \omega_t)$. Therefore, for the

bilinear game of Eq.(11), the Jacobian of F_{LEAD} has the following form,

$$\nabla F_{\text{LEAD}} = \begin{bmatrix} \mathbb{I}_{2n} + \beta \mathbb{I}_{2n} - (\eta + \alpha) \nabla \mathbf{v} & -\beta \mathbb{I}_{2n} + \alpha \nabla \mathbf{v} \\ \mathbb{I}_{2n} & 0 \end{bmatrix}. \quad (19)$$

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

$$\mu_{\pm}(\alpha, \beta, \eta) = \frac{1 - (\eta + \alpha)\lambda + \beta \pm \sqrt{\Delta}}{2} \quad (20)$$

where, $\Delta = (1 - (\eta + \alpha)\lambda + \beta)^2 - 4(\beta - \alpha\lambda)$ and $\lambda \in \text{Sp}(\nabla \mathbf{v}(\omega^*))$. Furthermore, for $|\alpha| \ll 1$ and $\beta = 0$, we have,

$$\mu_+(\alpha, \eta) \approx 1 - \eta\lambda + \alpha\lambda \left(\frac{\lambda\eta}{1 - \lambda\eta} \right) + \mathcal{O}(\alpha^2) \quad (21)$$

and

$$\mu_-(\alpha, \eta) \approx -\alpha\lambda \left(\frac{1}{1 - \lambda\eta} \right) + \mathcal{O}(\alpha^2). \quad (22)$$

See proof in Appendix H.

Theorem 2 states that the LEAD operator has two eigenvalues μ_+ and μ_- for each $\lambda \in \text{Sp}(\nabla \mathbf{v}(\omega^*))$. μ_+ can be viewed as a shift of the eigenvalues of gradient descent ascent in Eq.(18). Also, for small values of α , μ_+ is the limiting eigenvalue while $\mu_- \approx 0$. See Figure 1 for a schematic description.

In the following proposition, we show that locally, having a positive α decreases the norm of the limiting eigenvalue, $\rho(\alpha, \eta, \lambda) := \max\{|\mu_+|^2, |\mu_-|^2\}$.

Proposition 3. For any $\lambda \in \text{Sp}(\nabla \mathbf{v}(\omega^*))$,

$$\nabla_\alpha \rho(0) < 0 \Leftrightarrow \eta \in \left(0, \frac{1}{\text{Im}(\lambda)} \right), \quad (23)$$

where $\text{Im}(\lambda)$ is the imaginary component of λ .

See proof in Appendix I.

Now that we have established that a small positive value of α improves the rate, in the next theorem, we prove that for a specific choice of positive α and η in the bilinear game Eq.(11), a linear rate of convergence to its Nash equilibrium is attained.

Theorem 3. If we set $\eta = \alpha = \frac{1}{2\sigma_{\max}(\mathbb{A})}$, then we have $\forall \epsilon > 0$,

$$\Delta_{t+1} \in \mathcal{O} \left(\left(\frac{1}{2} + \frac{1}{2} \sqrt{1 - \frac{\sigma_{\min}(\mathbb{A})^2}{\sigma_{\max}(\mathbb{A})^2}} + \epsilon \right)^t \Delta_0 \right) \quad (24)$$

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

Theorem 3 ensures a linear convergence of LEAD in the general bilinear game. See proof in Appendix J.

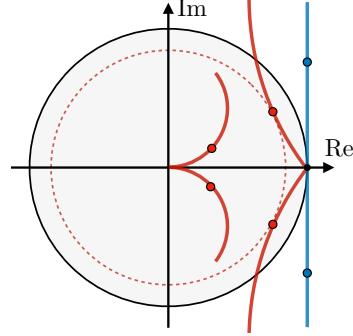


Figure 1. Diagram depicts positioning of the eigenvalues of gradient descent-ascent in blue (Eq.17) and those of LEAD (Eqns.21, 22) in red. Eigenvalues inside the black unit circle imply convergence such that the closer to the origin, the faster the convergence rate (Prop. 2). Every point on solid blue 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 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 which are close to zero for small values of α and hence do not hinder convergence.

5. Experiments

In this Section, we empirically validate the performance of our proposed method LEAD. Furthermore, we implement LEAD-Adam (pseudo-code in Appendix K.1) to be used in our experiments.

5.1. Comparison of Computational Cost

The Jacobian of the gradient vector field $\mathbf{v} = (\nabla_x f(\mathbf{x}, \mathbf{y}), -\nabla_y f(\mathbf{x}, \mathbf{y}))$ is given by,

$$\mathbb{J} = \begin{bmatrix} \nabla_x^2 f(\mathbf{x}, \mathbf{y}) & \nabla_{xy} f(\mathbf{x}, \mathbf{y}) \\ -\nabla_{yx} f(\mathbf{x}, \mathbf{y}) & -\nabla_y^2 f(\mathbf{x}, \mathbf{y}) \end{bmatrix}. \quad (25)$$

Considering player \mathbf{x} , a LEAD update for the same requires the computation of the term $\nabla_{xy} f(\mathbf{x}_k, \mathbf{y}_k)(\mathbf{y}_k - \mathbf{y}_{k-1})$, thereby involving only one block of the full Jacobian \mathbb{J} . On the other hand, the released implementation of Symplectic Gradient Adjustment (SGA) (Balduzzi et al., 2018a), requires the full computation of two Jacobian-vector products $\mathbb{J}\mathbf{v}, \mathbb{J}^\top \mathbf{v}$. Similarly, Competitive Gradient Descent (CGD) (Schäfer & Anandkumar, 2019) involves the computation of $(1 + \eta \nabla_{xy}^2 f(\mathbf{x}_k, \mathbf{y}_k) \nabla_{yx}^2 f(\mathbf{x}_k, \mathbf{y}_k))^{-1}$ along with the Jacobian-vector product $\nabla_{xy}^2 f(\mathbf{x}_k, \mathbf{y}_k) \nabla_y f(\mathbf{x}_k, \mathbf{y}_k)$. While the inverse term is approximated using conjugate gradient method in their implementation, 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 K). 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 implementation⁵.

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 ($\mathbf{y}_k - \mathbf{y}_{k-1}$). Using PyTorch, we do not require to compute ∇f_{xy} and then perform the multiplication. Given ∇f_x the whole term $\nabla f_{xy}(\mathbf{y}_k - \mathbf{y}_{k-1})$, is computed using Pytorch’s Autograd with the computational cost of a single gradient. Thus, LEAD also requires the computation of two gradients for each step.

5.2. Generative Adversarial Networks

8-Gaussians: We first compare 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 Fig. 3, we demonstrate that LEAD-Adam fully captures all the modes in the real data in both saturating and non-saturating losses.

CIFAR-10: We additionally 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)⁶, reaching a score of 19.27 ± 0.10 which outperforms

⁵SGA official DeepMind implementation (non-zero sum setting): https://github.com/deepmind/symplectic-gradient-adjustment/blob/master/Symplectic_Gradient_Adjustment.ipynb

⁶The FID is a metric for evaluating the quality of generated samples of a generative model. Lower FID corresponds to better sample quality.

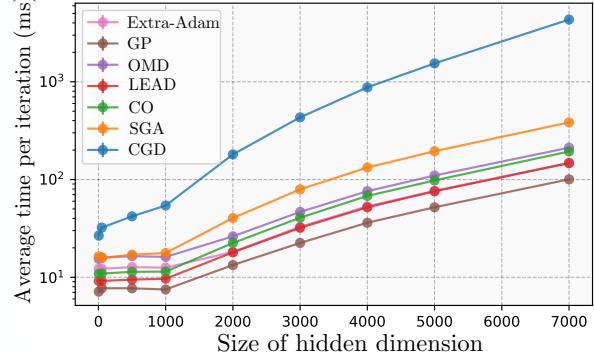


Figure 2. Average computational cost per iteration of several well-known methods for (non-saturating) GAN optimization. The numbers are reported on the 8-Gaussians generation task and averaged over 1000 iterations. Note that the y-axis is log-scale. We compare Competitive Gradient Descent (CGD) (46) (using official CGD optimizer code), Symplectic Gradient Adjustment (SGA) (6), Optimistic Mirror Descent with extra-gradient and gradient penalty (WGAN GP-Extra-OMD) (35), Consensus Optimization (CO) (36), Extra-gradient with Adam (Extra-Adam) (16), WGAN with Gradient Penalty (WGAN GP) (20), Optimistic Mirror Descent (OMD) (12). 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.

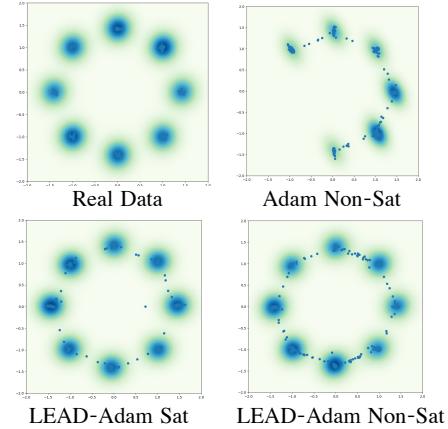


Figure 3. Performance of LEAD-Adam on the generation task of 8-Gaussians. All samples are shown after 10k iterations. Samples generated using Adam exhibit mode collapse, while LEAD-Adam does not suffer from this issue.

OMD (Mertikopoulos et al., 2018) and CGD (Schäfer & Anandkumar, 2019). See also Figure 4.

Furthermore, we evaluate LEAD-Adam on more complex and deep architectures such as the ResNet architecture in (Miyato et al., 2018). We compare with several state of the art results on the task of image generation on CIFAR-

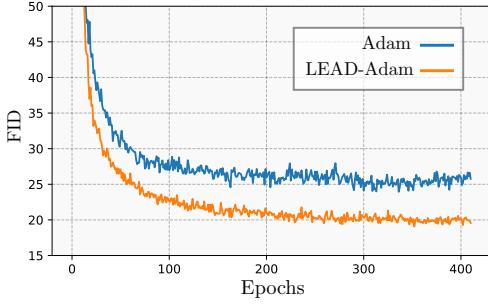


Figure 4. Plot showing the evolution of the FID over 400 epochs for our method (LEAD-Adam) vs vanilla Adam on a DCGAN architecture. Note that the FID is computed over 50k iterations.



Figure 5. Generated sample of LEAD-Adam on CIFAR-10. LEAD-Adam achieves an FID score of 11.22.

10 using ResNets. See Table 1 for a full comparison.

We report our results against a properly tuned version of SNGAN that achieves an FID of 12.36 using the code base of SNGAN PyTorch⁷. Our method obtains a competitive FID score of 11.22.

We give a detailed description of these experiments and full detail on the architecture and hyper-parameters in Appendix K. See also Figure 5 for a sample of generated samples on a ResNet using LEAD.

6. 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

	DCGAN	FID
Adam	24.38 ± 0.13	
LEAD-Adam	19.27 ± 0.10	
CGD (Schäfer & Anandkumar, 2019)	21.3	
OMD (Mertikopoulos et al., 2018)	29.6 ± 0.19	
ResNet		
Improved SNGAN	12.36 ± 0.12	
LEAD-Adam	11.22 ± 0.11	
LA-GAN (Chavdarova et al., 2020)	12.19	
ODE-GAN (Qin et al., 2020)	11.85 ± 0.21	
Evaluated with 5k samples		
SN-GAN (DCGAN) (Miyato et al., 2018)	29.3	
SN-GAN (ResNet) (Miyato et al., 2018)	21.7 ± 0.21	

Table 1. Performance of several methods on CIFAR-10 image generation task. Methods that are not cited are reported using our own implementation where we compute and report the mean and standard-deviation over 5 random runs. The FID is reported over 50k samples unless mentioned otherwise.

which hampers convergence. (Mescheder et al., 2017) provides a discussion on 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 rotational. 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. Table 2 contains a detailed comparison of the update rule of several second order methods. See appendix L 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. (Daskalakis et al., 2018) show that extrapolating the next value of the gradient using previous history, aids

¹¹For 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 $Jv, J^\top v$.

⁷<https://github.com/GongXinyuu/sngan.pytorch>

Method	Interaction Term
GDA	0
LEAD	$-\alpha \nabla_{xy}^2 f \Delta y_k$
SGA ⁽⁶⁾	$-\eta \gamma \nabla_{xy}^2 f \nabla_y f$
CGD ⁽⁴⁶⁾	$-\eta^2 \mathcal{C}^{-1} \nabla_{xy}^2 f \nabla_y f$
CO ⁽³⁶⁾	$-\eta \gamma \nabla_{xy}^2 f \nabla_y f - \eta \gamma \nabla_{xx}^2 f \nabla_x f$
FtR ⁽⁵¹⁾	$\eta_x (\nabla_{yy}^2 f)^{-1} \nabla_{xy}^2 f \nabla_x f$
LOLA ⁽¹⁴⁾	$-2\eta \alpha \nabla_{xy}^2 f \nabla_y f$

Table 2. Interaction term for several second-order methods in min-max optimization. $\Delta y_k = y_k - y_{k-1}$, while $\mathcal{C} = (I + \eta^2 \nabla_{xy}^2 f \nabla_{yx}^2 f)$. We compare the update rules of the first player¹⁰ for the following methods: Gradient Descent-Ascent (GDA), symplectic Least Action Dynamics (LEAD, ours), Symplectic Gradient Adjustment (SGA)¹¹, Competitive Gradient Descent (CGD), Consensus Optimization (CO), Follow-the-Ridge (FtR) and Learning with Opponent Learning Awareness (LOLA), in a two player zero-sum game. Note that the non-zero sum formulation of SGA requires the computation of $\mathbb{J}\mathbf{v}, \mathbb{J}^\top \mathbf{v}$. To see the full update rules, refer to Table 7 in Appendix L.

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. Recently (Tang, 2020) proposes to train mixtures of BigGANs (Brock et al., 2018) to achieve state of the art performance on the task of image generation with GANs on CIFAR-10.

Lastly, in regard to convergence analysis in 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 works 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 & Pilipouras, 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.

7. 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, 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.

8. Acknowledgements

The authors would like to thank Maxime Laborde, Gauthier Gidel, Nicolas Loizou, Hugo Berard, Giancarlo Kerg, Manuela Girotti, Adam Ibrahim, Tatjana Chavdarova, Damien Scieur and Michael Mulligan, for useful discussions and feedback. Reyhane Askari Hemmat would also like to thank Mohammad Pezeshki for his unwavering support.

This work is supported by NSERC Discovery Grants (RGPIN-2018-04821 and RGPIN-2019-06512), FRQNT

Young Investigator Startup Grants (2019-NC-253251 and 2019-NC-257943), a startup grant by IVADO, the Canada CIFAR AI chairs program and a collaborative grant from Samsung Electronics Co., Ltd.

Reyhane Askari Hemmat also acknowledges support of Borealis AI Graduate Fellowship and Microsoft Diversity Award. Amartya Mitra acknowledges support of the internship program at Mila and the UCR Graduate Fellowship.

References

- Abernethy, J., Lai, K. A., and Wibisono, A. Last-iterate convergence rates for min-max optimization. *arXiv preprint arXiv:1906.02027*, 2019.
- Azizian, W., Mitliagkas, I., Lacoste-Julien, S., and Gidel, G. 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.
- Azizian, W., Scieur, D., Mitliagkas, I., Lacoste-Julien, S., and Gidel, G. Accelerating smooth games by manipulating spectral shapes. *International Conference on Artificial Intelligence and Statistics*, 2020b.
- Bailey, J. P. and Piliouras, G. 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.
- Balduzzi, D., Racaniere, S., Martens, J., Foerster, J., Tuyls, K., and Graepel, T. *deepmind-symplectic-gradient-adjustment*, 2018a. https://github.com/deepmind/symplectic-gradient-adjustment/blob/master/Symplectic_Gradient_Adjustment.ipynb.
- Balduzzi, D., Racaniere, S., Martens, J., Foerster, J., Tuyls, K., and Graepel, T. The mechanics of n-player differentiable games. In *ICML*, volume 80, pp. 363–372. JMLR.org, 2018b.
- Berry, M. and Shukla, P. Curl force dynamics: symmetries, chaos and constants of motion. *New Journal of Physics*, 18(6):063018, 2016.
- Bertsekas, D. P. *Nonlinear programming*. Athena scientific Belmont, 1999.
- Brock, A., Donahue, J., and Simonyan, K. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.
- Bu, L., Babu, R., De Schutter, B., 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.
- Chavdarova, T., Pagliardini, M., Jaggi, M., and Fleuret, F. Taming gans with lookahead. *arXiv preprint arXiv:2006.14567*, 2020.
- Daskalakis, C., Ilyas, A., Syrgkanis, V., and Zeng, H. Training gans with optimism. In *International Conference on Learning Representations*, 2018.
- Fiez, T. and Ratliff, L. Gradient descent-ascent provably converges to strict local minmax equilibria with a finite timescale separation. *arXiv preprint arXiv:2009.14820*, 2020.
- Foerster, J., Chen, R. Y., Al-Shedivat, M., Whiteson, S., Abbeel, P., and Mordatch, I. 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.
- Gemp, I. and Mahadevan, S. Global convergence to the equilibrium of gans using variational inequalities. *arXiv preprint arXiv:1808.01531*, 2018.
- Gidel, G., Berard, H., Vignoud, G., Vincent, P., and Lacoste-Julien, S. A variational inequality perspective on generative adversarial networks. In *International Conference on Learning Representations*, 2018.
- Gidel, G., Hemmat, R. A., Pezeshki, M., Le Priol, R., Huang, G., Lacoste-Julien, S., and Mitliagkas, I. Negative momentum for improved game dynamics. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 1802–1811, 2019.
- Golowich, N., Pattathil, S., Daskalakis, C., and Ozdaglar, A. Last iterate is slower than averaged iterate in smooth convex-concave saddle point problems. *arXiv preprint arXiv:2002.00057*, 2020.
- Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. In *Advances in neural information processing systems*, pp. 2672–2680, 2014.
- Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. C. Improved training of wasserstein gans. In *Advances in neural information processing systems*, pp. 5767–5777, 2017.
- Hahn, W., Hosenthien, H. H., and Lehnigk, H. *Theory and application of Liapunov's direct method*. Prentice-Hall Englewood Cliffs, NJ, 1963.

- Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. 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.
- Hsieh, Y.-P., Mertikopoulos, P., and Cevher, V. The limits of min-max optimization algorithms: convergence to spurious non-critical sets. *arXiv preprint arXiv:2006.09065*, 2020.
- Ibrahim, A., Azizian, W., Gidel, G., and Mitliagkas, I. Linear lower bounds and conditioning of differentiable games. In *International conference on machine learning*, 2020.
- Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- Krizhevsky, A. and Hinton, G. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.
- Landau, L. and Lifshitz, E. *Course of theoretical physics. vol. 1: Mechanics*. Oxford, 1960.
- Landau, L. D., Bell, J., Kearsley, M., Pitaevskii, L., Lifshitz, E., and Sykes, J. *Electrodynamics of continuous media*, volume 8. elsevier, 2013.
- Langley, P. Crafting papers on machine learning. In Langley, P. (ed.), *Proceedings of the 17th International Conference on Machine Learning (ICML 2000)*, pp. 1207–1216, Stanford, CA, 2000. Morgan Kaufmann.
- Letcher, A., Balduzzi, D., Racaniere, S., Martens, J., Foerster, J. N., Tuyls, K., and Graepel, T. Differentiable game mechanics. *Journal of Machine Learning Research*, 20(84):1–40, 2019.
- Loizou, N., Berard, H., Jolicoeur-Martineau, A., Vincent, P., Lacoste-Julien, S., and Mitliagkas, I. Stochastic hamiltonian gradient methods for smooth games. *ICML*, 2020.
- Lyapunov, A. M. The general problem of the stability of motion. *International journal of control*, 55(3):531–534, 1992.
- Maddison, C. J., Paulin, D., Teh, Y. W., O’Donoghue, B., and Doucet, A. Hamiltonian descent methods. *arXiv preprint arXiv:1809.05042*, 2018.
- Mazumdar, E. V., Jordan, M. I., and Sastry, S. S. On finding local nash equilibria (and only local nash equilibria) in zero-sum games. *arXiv preprint arXiv:1901.00838*, 2019.
- Mertikopoulos, P., Lecouat, B., Zenati, H., Foo, C.-S., Chandrasekhar, V., and Piliouras, G. Optimistic mirror descent in saddle-point problems: Going the extra (gradient) mile. *arXiv preprint arXiv:1807.02629*, 2018.
- Mescheder, L., Nowozin, S., and Geiger, A. The numerics of gans. In *Advances in Neural Information Processing Systems*, pp. 1825–1835, 2017.
- Miyato, T., Kataoka, T., Koyama, M., and Yoshida, Y. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018.
- Muehlebach, M. and Jordan, M. A dynamical systems perspective on nesterov acceleration. In *International Conference on Machine Learning*, pp. 4656–4662, 2019.
- Nagarajan, V. and Kolter, J. Z. Gradient descent gan optimization is locally stable. In *Advances in neural information processing systems*, pp. 5585–5595, 2017.
- Nesterov, Y. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2013.
- Nouiehed, M., Sanjabi, M., Huang, T., Lee, J. D., and Razaviyayn, M. 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.
- Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. Automatic differentiation in pytorch. 2017.
- Polyak, B. T. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17, 1964.
- Qin, C., Wu, Y., Springenberg, J. T., Brock, A., Donahue, J., Lillicrap, T. P., and Kohli, P. Training generative adversarial networks by solving ordinary differential equations. *arXiv preprint arXiv:2010.15040*, 2020.
- Ryu, E. K., Yuan, K., and Yin, W. Ode analysis of stochastic gradient methods with optimism and anchoring for minimax problems and gans. *arXiv preprint arXiv:1905.10899*, 2019.
- Schäfer, F. and Anandkumar, A. Competitive gradient descent. In *Advances in Neural Information Processing Systems*, pp. 7623–7633, 2019.
- Sener, O. and Koltun, V. Multi-task learning as multi-objective optimization. In *Advances in Neural Information Processing Systems*, pp. 527–538, 2018.

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

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

Tang, S. Lessons learned from the training of gans on artificial datasets. *IEEE Access*, 8:165044–165055, 2020.

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

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

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

Yazıcı, Y., Foo, C.-S., Winkler, S., Yap, K.-H., Piliouras, G., and Chandrasekhar, V. The unusual effectiveness of averaging in gan training, 2019.

Supplementary Materials for LEAD: Least-Action Dynamics for Min-Max Optimization

A. Derivation of Eq. 3

Proof. Polyak's heavy ball method with unit momentum for the minimization of a single objective $f(x)$ is given by,

$$x_{k+1} = x_k + (x_k - x_{k-1}) - \eta \nabla_x f(x_k), \quad (26)$$

where η is the learning rate. One can rewrite this equation as,

$$\frac{(x_{k+\delta} - x_k) - (x_k - x_{k-\delta})}{\delta^2} = -\frac{\eta}{\delta^2} \nabla_x f(x_k), \quad (27)$$

where δ is the discretization step-size. In the limit $\delta, \eta \rightarrow 0$, Eq.(27) then becomes ($x_k \rightarrow X(t) \equiv X$),

$$m \ddot{X} = -\nabla_X f(X) \quad (28)$$

This is equivalent to Newton's 2nd Law of motion (Eq.(2)) of a particle of mass $m = \delta^2/\eta$, if we identify $F = -\nabla_X f(X)$. \square

B. Implicit Discretization

The continuous-time EOMs (8) can be discretized in an implicit way as,

$$\begin{aligned} x_{k+1} &= x_k + \beta_{\text{imp}} \Delta x_k - \eta_{\text{imp}} \nabla_x f(x_{k+1}, y_{k+1}) \\ &\quad - \alpha_{\text{imp}} \nabla_{xy} f(x_{k+1}, y_{k+1}) \Delta y_{k+1}, \\ y_{k+1} &= y_k + \beta_{\text{imp}} \Delta y_k + \eta_{\text{imp}} \nabla_y f(x_{k+1}, y_{k+1}) \\ &\quad + \alpha_{\text{imp}} \nabla_{yx} f(x_{k+1}, y_{k+1}) \Delta x_{k+1}, \end{aligned} \quad (29)$$

where Δx_k is $(x_k - x_{k-1})$ and $\alpha_{\text{imp}}, \beta_{\text{imp}}$ and η_{imp} are hyper-parameters dependent on μ, q and δ . We refer to these discrete update rules as *implicit Least Action Dynamics (iLEAD)*.

Proof. The EOMs of the quadratic game in continuous-time (Eq.(8)), can be discretized in a purely implicit way as ($m = 1$),

$$x_{k+1} - x_k = \delta v_{k+1}^x \quad (30a)$$

$$y_{k+1} - y_k = \delta v_{k+1}^y \quad (30b)$$

$$v_{k+1}^x - v_k^x = -q\delta \nabla_{xy} f(x_{k+1}, y_{k+1}) v_{k+1}^y - \mu\delta v_{k+1}^x - \delta \nabla_x f(x_{k+1}, y_{k+1}) \quad (30c)$$

$$v_{k+1}^y - v_k^y = q\delta \nabla_{xy} f(x_{k+1}, y_{k+1}) v_{k+1}^x - \mu\delta v_{k+1}^y + \delta \nabla_y f(x_{k+1}, y_{k+1}), \quad (30d)$$

where δ is the discretization step-size. Plugging in Eqns.(30b), (30d) in Eqns.(30a), (30c) then give us,

$$\begin{aligned} x_{k+1} &= x_k + \beta_{\text{imp}} \Delta x_k - \eta_{\text{imp}} \nabla_x f(x_{k+1}, y_{k+1}) - \alpha_{\text{imp}} \nabla_{xy} f(x_{k+1}, y_{k+1}) \Delta y_{k+1} \\ y_{k+1} &= y_k + \beta_{\text{imp}} \Delta y_k + \eta_{\text{imp}} \nabla_y f(x_{k+1}, y_{k+1}) + \alpha_{\text{imp}} \nabla_{xy} f(x_{k+1}, y_{k+1}) \Delta x_{k+1} \end{aligned} \quad (31)$$

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

$$\beta_{\text{imp}} = \frac{1}{1 + \mu\delta}, \quad \eta_{\text{imp}} = \frac{\delta^2}{1 + \mu\delta}, \quad \alpha_{\text{imp}} = \frac{q\delta}{1 + \mu\delta}. \quad (32)$$

\square

See more analysis on this discretization in Appendix G.

C. Proof of Proposition 1

Proof. The EOMs of the quadratic game in continuous-time (Eq.(8)), can be discretized in a symplectic (Shi et al., 2019) way as,

$$x_{k+1} - x_k = \delta v_{k+1}^x, \quad (33a)$$

$$y_{k+1} - y_k = \delta v_{k+1}^y, \quad (33b)$$

$$v_{k+1}^x - v_k^x = -q\delta \nabla_{xy} f(x_k, y_k) v_k^y - \mu\delta v_k^x - \delta \nabla_x f(x_k, y_k) \quad (33c)$$

$$v_{k+1}^y - v_k^y = q\delta \nabla_{xy} f(x_k, y_k) v_k^x - \mu\delta v_k^y + \delta \nabla_y f(x_k, y_k) \quad (33d)$$

where δ is the discretization step-size. Now, using Eqns.(33a) and (33b) above, we can further re-express Eqns.(33c), (33d) as,

$$\begin{aligned} x_{k+1} &= x_k + \beta_{\text{sym}} \Delta x_k - \eta_{\text{sym}} \nabla_x f(x_k, y_k) - \alpha_{\text{sym}} \nabla_{x,y} f(x_k, y_k) \Delta y_k \\ y_{k+1} &= y_k + \beta_{\text{sym}} \Delta y_k + \eta_{\text{sym}} \nabla_y f(x_k, y_k) + \alpha_{\text{sym}} \nabla_{x,y} f(x_k, y_k) \Delta x_k \end{aligned} \quad (34)$$

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

$$\beta_{\text{sym}} = 1 - \mu\delta, \quad \eta_{\text{sym}} = \delta^2, \quad \alpha_{\text{sym}} = q\delta \quad (35)$$

In Proposition 1 of the main text, we refer to $\beta_{\text{sym}} \rightarrow \beta$, $\eta_{\text{sym}} \rightarrow \eta$ and $\alpha_{\text{sym}} \rightarrow \alpha$. \square

D. Lyapunov stability of the general bilinear game (Continuous time)

Proof. In the following, we consider a general bilinear game of the form, $f(\mathbf{X}, \mathbf{Y}) = \mathbf{X}^T \mathbb{A} \mathbf{Y}$, where $\mathbf{X} \equiv (X^1, \dots, X^n)$, $\mathbf{Y} \equiv (Y^1, \dots, Y^n)$ are taken to be equi-dimensional parameter vectors, with $\mathbb{A}_{n \times n}$ being a constant positive-definite matrix. For such a game, we can correspondingly generalize our continuous-time EOMs of Eq.(8) as Eq.(13).

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

$$\begin{aligned} \mathcal{E}_t &= \frac{1}{2} \left(\dot{\mathbf{X}} + \mu \mathbf{X} + \mu \mathbb{A} \mathbf{Y} \right)^T \left(\dot{\mathbf{X}} + \mu \mathbf{X} + \mu \mathbb{A} \mathbf{Y} \right) + \frac{1}{2} \left(\dot{\mathbf{Y}} + \mu \mathbf{Y} - \mu \mathbb{A}^T \mathbf{X} \right)^T \left(\dot{\mathbf{Y}} + \mu \mathbf{Y} - \mu \mathbb{A}^T \mathbf{X} \right) \\ &\quad + \frac{1}{2} \left(\dot{\mathbf{X}}^T \dot{\mathbf{X}} + \dot{\mathbf{Y}}^T \dot{\mathbf{Y}} \right) + \mathbf{X}^T \mathbb{A} \mathbb{A}^T \mathbf{X} + \mathbf{Y}^T \mathbb{A} \mathbb{A}^T \mathbf{Y} \\ &\geq 0 \quad \forall t \end{aligned} \quad (36)$$

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

$$\begin{aligned} \dot{\mathcal{E}}_t &= \left(\dot{\mathbf{X}} + \mu \mathbf{X} + \mu \mathbb{A} \mathbf{Y} \right)^T \left(\ddot{\mathbf{X}} + \mu \dot{\mathbf{X}} + \mu \mathbb{A} \dot{\mathbf{Y}} \right) + \left(\dot{\mathbf{Y}} + \mu \mathbf{Y} - \mu \mathbb{A}^T \mathbf{X} \right)^T \left(\ddot{\mathbf{Y}} + \mu \dot{\mathbf{Y}} - \mu \mathbb{A}^T \dot{\mathbf{X}} \right) \\ &\quad + \left(\dot{\mathbf{X}}^T \ddot{\mathbf{X}} + \dot{\mathbf{Y}}^T \ddot{\mathbf{Y}} \right) + 2 \left(\mathbf{X}^T \mathbb{A} \mathbb{A}^T \dot{\mathbf{X}} + \mathbf{Y}^T \mathbb{A} \mathbb{A}^T \dot{\mathbf{Y}} \right) \\ &= \left(\dot{\mathbf{X}}^T + \mu \mathbf{X}^T + \mu \mathbf{Y}^T \mathbb{A}^T \right) \left((-q + \mu) \mathbb{A} \dot{\mathbf{Y}} - \mathbb{A} \mathbf{Y} \right) + \left(\dot{\mathbf{Y}}^T + \mu \mathbf{Y}^T - \mu \mathbf{X}^T \mathbb{A} \right) \left((q - \mu) \mathbb{A}^T \dot{\mathbf{X}} + \mathbb{A}^T \mathbf{X} \right) \\ &\quad + \dot{\mathbf{X}}^T \left(-q \mathbb{A} \dot{\mathbf{Y}} - \mu \dot{\mathbf{X}} - \mathbb{A} \mathbf{Y} \right) + \dot{\mathbf{Y}}^T \left(q \mathbb{A}^T \dot{\mathbf{X}} - \mu \dot{\mathbf{Y}} + \mathbb{A}^T \mathbf{X} \right) + 2 \left(\mathbf{X}^T \mathbb{A} \mathbb{A}^T \dot{\mathbf{X}} + \mathbf{Y}^T \mathbb{A} \mathbb{A}^T \dot{\mathbf{Y}} \right) \\ &= (\mu(q - \mu) - 2) \left(\mathbf{Y}^T \mathbb{A}^T \dot{\mathbf{X}} - \mathbf{X}^T \mathbb{A} \dot{\mathbf{Y}} \right) - (\mu(q - \mu) - 2) \left(\mathbf{X}^T \mathbb{A} \mathbb{A}^T \dot{\mathbf{X}} + \mathbf{Y}^T \mathbb{A} \mathbb{A}^T \dot{\mathbf{Y}} \right) \\ &\quad - \mu \left(\mathbf{X}^T \mathbb{A} \mathbb{A}^T \mathbf{X} + \mathbf{Y}^T \mathbb{A} \mathbb{A}^T \mathbf{Y} \right) - \mu \left(\dot{\mathbf{X}}^T \dot{\mathbf{X}} + \dot{\mathbf{Y}}^T \dot{\mathbf{Y}} \right) \end{aligned} \quad (37)$$

where we have used the fact that $\mathbf{X}^T \mathbb{A} \mathbf{Y}$ being a scalar, implies $\mathbf{X}^T \mathbb{A} \mathbf{Y} = \mathbf{Y}^T \mathbb{A}^T \mathbf{X}$. Setting $q = (2/\mu) + \mu$ in the above, then leads to,

$$\begin{aligned} \dot{\mathcal{E}}_t &= -\mu \left(\mathbf{X}^T \mathbb{A} \mathbb{A}^T \mathbf{X} + \mathbf{Y}^T \mathbb{A} \mathbb{A}^T \mathbf{Y} \right) - \mu \left(\dot{\mathbf{X}}^T \dot{\mathbf{X}} + \dot{\mathbf{Y}}^T \dot{\mathbf{Y}} \right) \\ &= -\mu \left(\|\mathbb{A}^T \mathbf{X}\|^2 + \|\mathbb{A} \mathbf{Y}\|^2 \right) - \mu \left(\|\dot{\mathbf{X}}\|^2 + \|\dot{\mathbf{Y}}\|^2 \right) \leq 0 \quad \forall t \end{aligned} \quad (38)$$

exhibiting that the Lyapunov function, Eq.(14) is *asymptotically stable* at all times t . \square

E. Rate of convergence of the general bilinear game (Continuous time)

Proof. Consider the following expression, where \mathcal{E}_t is our general bilinear game Lyapunov function Eq.(14) :

$$\begin{aligned}
 & -\rho\mathcal{E}_t - \frac{\rho\mu}{2} \left(\|\mathbf{X} - \dot{\mathbf{X}}\|^2 + \|\mathbf{Y} - \dot{\mathbf{Y}}\|^2 \right) - \frac{\rho\mu}{2} \left(\|\dot{\mathbf{X}} - \mathbb{A}\mathbf{Y}\|^2 + \|\mathbb{A}^T\mathbf{X} + \dot{\mathbf{Y}}\|^2 \right) \\
 & = -\rho\mathcal{E}_t - \frac{\rho\mu}{2} \left(\|\mathbf{X}\|^2 + \|\mathbf{Y}\|^2 \right) + \rho\mu \left(\mathbf{X}^T \dot{\mathbf{X}} + \mathbf{Y}^T \dot{\mathbf{Y}} \right) - \rho\mu \left(\|\dot{\mathbf{X}}\|^2 + \|\mathbf{Y}\|^2 \right) \\
 & \quad - \rho\mu \left(\mathbf{X}^T \mathbb{A} \dot{\mathbf{Y}} - \dot{\mathbf{X}}^T \mathbb{A} \mathbf{Y} \right) - \frac{\rho\mu}{2} \left(\|\mathbb{A}^T \mathbf{X}\|^2 + \|\mathbb{A} \mathbf{Y}\|^2 \right) \\
 & = -\rho(1+\mu) \left(\|\dot{\mathbf{X}}\|^2 + \|\dot{\mathbf{Y}}\|^2 \right) - \frac{\rho}{2} (\mu^2 + \mu) \left(\|\mathbf{X}\|^2 + \|\mathbf{Y}\|^2 \right) \\
 & \quad - \frac{\rho}{2} (\mu^2 + \mu + 2) \left(\|\mathbb{A}^T \mathbf{X}\|^2 + \|\mathbb{A} \mathbf{Y}\|^2 \right) \\
 & \leq -\rho\mathcal{E}_t
 \end{aligned} \tag{39}$$

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 $\mathbb{A} = \mathbb{V}^T \mathbb{S} \mathbb{U}$. 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.(39), allows us to write,

$$\begin{aligned}
 & -\rho(1+\mu) \left(\|\dot{\mathbf{X}}\|^2 + \|\dot{\mathbf{Y}}\|^2 \right) - \frac{\rho}{2} (\mu^2 + \mu) \left(\|\mathbf{X}\|^2 + \|\mathbf{Y}\|^2 \right) - \frac{\rho}{2} (\mu^2 + \mu + 2) \left(\|\mathbb{A}^T \mathbf{X}\|^2 + \|\mathbb{A} \mathbf{Y}\|^2 \right) \\
 & = -\rho(1+\mu) \left(\|\mathbb{V} \dot{\mathbf{X}}\|^2 + \|\mathbb{U} \dot{\mathbf{Y}}\|^2 \right) - \frac{\rho}{2} (\mu^2 + \mu) \left(\|\mathbb{V} \mathbf{X}\|^2 + \|\mathbb{U} \mathbf{Y}\|^2 \right) \\
 & \quad - \frac{\rho}{2} (\mu^2 + \mu + 2) \left(\|\mathbb{S} \mathbb{V} \mathbf{X}\|^2 + \|\mathbb{S} \mathbb{U} \mathbf{Y}\|^2 \right) \\
 & = -\rho(1+\mu) \left(\|\dot{\mathcal{X}}\|^2 + \|\dot{\mathcal{Y}}\|^2 \right) - \frac{\rho}{2} (\mu^2 + \mu) \left(\|\mathcal{X}\|^2 + \|\mathcal{Y}\|^2 \right) - \frac{\rho}{2} (\mu^2 + \mu + 2) \left(\|\mathbb{S} \mathcal{X}\|^2 + \|\mathbb{S} \mathcal{Y}\|^2 \right) \\
 & = -\sum_{j=1}^n \rho(1+\mu) \left(\|\dot{\mathcal{X}}^j\|^2 + \|\dot{\mathcal{Y}}^j\|^2 \right) - \sum_{j=1}^n \frac{\rho}{2} ((1+\sigma_j^2)(\mu^2 + \mu) + 2\sigma_j^2) \left(\|\mathcal{X}^j\|^2 + \|\mathcal{Y}^j\|^2 \right)
 \end{aligned} \tag{40}$$

where we have made use of the relations $\mathbb{U}^T \mathbb{U} = \mathbb{U} \mathbb{U}^T = \mathbb{I}_n = \mathbb{V}^T \mathbb{V} = \mathbb{V} \mathbb{V}^T$, and additionally performed a basis change, as $\mathcal{X} = \mathbb{V} \mathbf{X}$ and $\mathcal{Y} = \mathbb{U} \mathbf{Y}$. Now, we know from Eq.(38) that,

$$\begin{aligned}
 \dot{\mathcal{E}}_t & = -\mu \left(\|\mathbb{A}^T \mathcal{X}\|^2 + \|\mathbb{A} \mathcal{Y}\|^2 \right) - \mu \left(\|\dot{\mathcal{X}}\|^2 + \|\dot{\mathcal{Y}}\|^2 \right) \\
 & = -\mu \left(\|\mathbb{U}^T \mathbb{S} \mathbb{V} \mathcal{X}\|^2 + \|\mathbb{V}^T \mathbb{S} \mathbb{U} \mathcal{Y}\|^2 \right) - \mu \left(\|\mathbb{V} \dot{\mathcal{X}}\|^2 + \|\mathbb{U} \dot{\mathcal{Y}}\|^2 \right) \\
 & = -\mu \left(\|\mathbb{S} \mathcal{X}\|^2 + \|\mathbb{S} \mathcal{Y}\|^2 \right) - \mu \left(\|\dot{\mathcal{X}}\|^2 + \|\dot{\mathcal{Y}}\|^2 \right) \\
 & = -\sum_{j=1}^n \mu \sigma_j^2 \left(\|\mathcal{X}^j\|^2 + \|\mathcal{Y}^j\|^2 \right) - \sum_{j=1}^n \mu_r \left(\|\dot{\mathcal{X}}^j\|^2 + \|\dot{\mathcal{Y}}^j\|^2 \right)
 \end{aligned} \tag{41}$$

Comparing the above expression with Eq.(40), we note that a choice of ρ as,

$$\rho \leq \min \left\{ \frac{\mu}{1+\mu}, \frac{2\mu\sigma_j^2}{(1+\sigma_j^2)(\mu^2 + \mu) + 2\sigma_j^2} \right\} \quad \forall j \in [1, n] \tag{42}$$

implies,

$$\begin{aligned}
 \dot{\mathcal{E}}_t &\leq -\rho \mathcal{E} \\
 \Rightarrow \mathcal{E}_t &\leq \mathcal{E}_0 \exp(-\rho t) \\
 \Rightarrow X^T \mathbb{A} \mathbb{A}^T X + Y^T \mathbb{A}^T \mathbb{A} Y &\leq \mathcal{E}_0 \exp(-\rho t) \\
 \Rightarrow \mathcal{X}^T \mathbb{S}^2 \mathcal{X} + \mathcal{Y}^T \mathbb{S}^2 \mathcal{Y} &\leq \mathcal{E}_0 \exp(-\rho t) \\
 \Rightarrow \sum_{j=1}^n \sigma_j^2 (\|\mathcal{X}^j\|^2 + \|\mathcal{Y}^j\|^2) &\leq \mathcal{E}_0 \exp(-\rho t) \\
 \Rightarrow \sum_{j=1}^n \sigma_j^2 (\|X^j\|^2 + \|Y^j\|^2) &\leq \mathcal{E}_0 \exp(-\rho t) \\
 \therefore \|\mathbf{X}\|^2 + \|\mathbf{Y}\|^2 &\leq \frac{\mathcal{E}_0}{\sigma_{\min}^2} \exp(-\rho t) \quad \forall j
 \end{aligned} \tag{43}$$

□

This completes our Proof of convergence (continuous-time) of iLEAD (Eq.(13)) in the general bilinear game $f(\mathbf{X}, \mathbf{Y}) = \mathbf{X}^T \mathbb{A} \mathbf{Y}$.

F. Continuous time Lyapunov analysis on quadratic games

We now study the behavior of our method on the scalar quadratic min-max game,

$$f(X, Y) = \frac{h}{2} X^2 - \frac{h}{2} Y^2 + XY, \tag{44}$$

where h is the strong monotonicity.

Theorem 4. *For the quadratic min-max game of Eq.(44),*

$$\mathcal{E}_t = \frac{1}{2} (\dot{X} + \mu X + \mu Y)^2 + \frac{1}{2} (\dot{Y} + \mu Y - \mu X)^2 + \frac{1}{2} (\dot{X}^2 + \dot{Y}^2) + (1+h)(X^2 + Y^2) \tag{45}$$

is a Lyapunov function for the dynamics of Eq.(8) under the choice $q = (2 + \mu^2)/\mu$, with $\dot{\mathcal{E}}_t \leq -\rho \mathcal{E}_t$ for some positive definite constant ρ dependent on μ and h . Hence,

$$X^2 + Y^2 \leq \frac{\mathcal{E}_0}{1+h} \exp(-\rho t)$$

Thus, the continuous-time dynamics of Eq.(8) for the quadratic game are convergent at a linear rate ρ , to the Nash equilibrium $(0, 0)$.

Proof. For the class of functions in (44), Eq.(8) translates to,

$$\begin{aligned}
 \ddot{X} &= -q\dot{Y} - \mu\dot{X} - hX - Y \\
 \ddot{Y} &= q\dot{X} - \mu\dot{Y} - hY + X
 \end{aligned} \tag{47}$$

The Lyapunov function for the above EOMs is then defined to be,

$$\begin{aligned}
 \mathcal{E}_t &= \frac{1}{2} (\dot{X} + \mu X + \mu Y)^2 + \frac{1}{2} (\dot{Y} + \mu Y - \mu X)^2 + \frac{1}{2} (\dot{X}^2 + \dot{Y}^2) \\
 &\quad + (1+h)(X^2 + Y^2) \\
 &\geq 0 \quad \forall t
 \end{aligned} \tag{48}$$

Next, by Eq.(47), we can compute the time derivative of \mathcal{E}_t as,

$$\begin{aligned}
 \dot{\mathcal{E}}_t &= \dot{X} (\ddot{X} + \mu \dot{X} + \mu \dot{Y}) + \mu X (\ddot{X} + \mu \dot{X} + \mu \dot{Y}) + \mu Y (\ddot{X} + \mu \dot{X} + \mu \dot{Y}) \\
 &\quad + \dot{Y} (\ddot{Y} + \mu \dot{Y} - \mu \dot{X}) + \mu Y (\ddot{Y} + \mu \dot{Y} - \mu \dot{X}) - \mu X (\ddot{Y} + \mu \dot{Y} - \mu \dot{X}) \\
 &\quad + \dot{X} \ddot{X} + \dot{Y} \ddot{Y} + 2(1+h) (\dot{X} \dot{X} + \dot{Y} \dot{Y}) \\
 &= \dot{X} (-q \dot{Y} - hX - Y + \mu \dot{Y}) + \mu X (-q \dot{Y} - hX - Y + \mu \dot{Y}) \\
 &\quad + \mu Y (-q \dot{Y} - hX - Y + \mu \dot{Y}) + \dot{Y} (q \dot{X} - hY + X - \mu \dot{X}) \\
 &\quad + \mu Y (q \dot{X} - hY + X - \mu \dot{X}) - \mu X (q \dot{X} - hY + X - \mu \dot{X}) \\
 &\quad + \dot{X} (-q \dot{Y} - \mu \dot{X} - hX - Y) + \dot{Y} (q \dot{X} - \mu \dot{Y} - hY + X) \\
 &\quad + 2(1+h) (\dot{X} \dot{X} + \dot{Y} \dot{Y}) \\
 &= (2 - \mu q + \mu^2) X \dot{Y} + (2 - \mu q + \mu^2) \dot{Y} Y - \mu (1+h) Y^2 - \mu \dot{X}^2 \\
 &\quad - (2 - \mu q + \mu^2) \dot{X} Y + (2 - \mu q + \mu^2) \dot{X} \dot{X} - \mu (1+h) X^2 - \mu \dot{Y}^2
 \end{aligned} \tag{49}$$

Now, by choosing to set,

$$q = \frac{2 + \mu^2}{\mu} \tag{50}$$

Eq.(49) reduces to,

$$\dot{\mathcal{E}}_t = -\mu (1+h) (X^2 + Y^2) - \mu (\dot{X}^2 + \dot{Y}^2) \leq 0 \quad \forall t \tag{51}$$

as both μ and $h > 0$. Specifically, we observe that for $(X, Y) \neq (0, 0)$,

$$\begin{aligned}
 \dot{\mathcal{E}}_t &< 0, \\
 \therefore \mathcal{E}_t &\rightarrow 0, \text{ as } t \rightarrow \infty, \\
 \text{hence, } X^2 + Y^2 &\leq \mathcal{E}_t \rightarrow 0, \text{ as } t \rightarrow \infty
 \end{aligned} \tag{52}$$

guaranteeing *asymptotic stability* of our algorithm, i.e. convergence to the Nash Equilibrium $(X, Y) = (0, 0)$ as $t \rightarrow \infty$. With this result in hand, let us now consider the following expression,

$$-\rho \mathcal{E}_t - \frac{\rho \mu}{2} (X - \dot{X})^2 - \frac{\rho \mu}{2} (Y - \dot{Y})^2 - \frac{\rho \mu}{2} (\dot{X} - Y)^2 - \frac{\rho \mu}{2} (X + \dot{Y})^2 \tag{53}$$

where ρ is a constant and is determined as,

$$0 \leq \rho \leq \min \left\{ \frac{\mu (1+h)}{\mu^2 + \mu + (1+h)}, \frac{\mu}{1+\mu} \right\} \tag{54}$$

It can then be checked that on expansion, Eq.(53) becomes,

$$\Rightarrow -\rho (\mu^2 + \mu + (1+h)) (X^2 + Y^2) - \rho (1+\mu) (\dot{X}^2 + \dot{Y}^2) \leq -\rho \mathcal{E}_t \tag{55}$$

Now, using Eq.(51) and the condition (54), we can then go on to write,

$$\begin{aligned}
 \dot{\mathcal{E}}_t &= -\mu (1+h) (X^2 + Y^2) - \mu (\dot{X}^2 + \dot{Y}^2) \\
 &\leq -\rho (\mu^2 + \mu + (1+h)) (X^2 + Y^2) - \rho (1+\mu) (\dot{X}^2 + \dot{Y}^2) \\
 &\leq -\rho \mathcal{E}_t
 \end{aligned} \tag{56}$$

On integrating the above, one then gets,

$$\begin{aligned}
 & \Rightarrow \frac{d\mathcal{E}_t}{\mathcal{E}_t} \leq -\rho dt \\
 & \Rightarrow \mathcal{E}_t \leq \mathcal{E}_0 \exp(-\rho t) \\
 \Rightarrow (1+h)(X^2 + Y^2) & \leq \mathcal{E}_t \leq \mathcal{E}_0 \exp(-\rho t) \\
 & \Rightarrow X^2 + Y^2 \leq \frac{\mathcal{E}_0}{1+h} \exp(-\rho t)
 \end{aligned} \tag{57}$$

This exhibits that our continuous-time optimizer in the quadratic min-max game (Eq.(47)), is convergent to the Nash equilibrium $(0, 0)$ at a linear rate ρ , as determined from Eq.(54). \square

G. Lyapunov Analysis and convergence rate for the implicit discretization

To perform Lyapunov analysis on our discrete-time dynamics of iLEAD (Eq.(29)), we note that a function \mathcal{E}_k is a discrete Lyapunov function if $\forall k \in \mathbb{N}$, (i) $\mathcal{E}_k \geq 0$, and (ii) $\mathcal{E}_k - \mathcal{E}_{k-1} \leq 0$.

Theorem 5. *For the scalar quadratic min-max game of Eq.(44),*

$$\begin{aligned}
 \mathcal{E}_k = \frac{1}{2} & \left(v_k^x + 2\sqrt{\frac{\sqrt{5}}{3}} \frac{x_k}{\mu} + 2\sqrt{\frac{\sqrt{5}}{3}} \frac{y_k}{\mu} \right)^2 + \frac{1}{2} \left(v_k^y + 2\sqrt{\frac{\sqrt{5}}{3}} \frac{y_k}{\mu} - 2\sqrt{\frac{\sqrt{5}}{3}} \frac{x_k}{\mu} \right)^2 \\
 & + \frac{1}{2} \left((v_k^x)^2 + (v_k^y)^2 \right) + 2\sqrt{5} \left(1 + \frac{2h}{\sqrt{5}} \right) (x_k^2 + y_k^2)
 \end{aligned} \tag{58}$$

is a discrete Lyapunov function for the dynamics of iLEAD (Eq.(29)) under the choice $q = \sqrt{5} \left(\frac{2+\mu^2}{\mu} \right)$ with $\mathcal{E}_k \leq \left(\frac{C}{C+\delta\mu} \right) \mathcal{E}_{k-1}$, thereby leading iLEAD to converge linearly to the Nash equilibrium $(0, 0)$ as,

$$\boxed{x_k^2 + y_k^2 \leq \frac{\mu^2}{C} \left(\frac{C}{C + \delta\mu} \right)^k \mathcal{E}_0} \tag{59}$$

where $C = \mu^2 (2\sqrt{5} + 4h) + 4\sqrt{5}$, and δ is the discretization step-size

Firstly, we recall that the continuous-time EOMs of the quadratic min-max game (44), can be discretized implicitly as:

$$\begin{aligned}
 x_k - x_{k-1} &= \delta v_k^x \\
 y_k - y_{k-1} &= \delta v_k^y \\
 v_k^x - v_{k-1}^x &= -q\delta v_k^y - \mu\delta v_k^x - h\delta x_k - \delta y_k \\
 v_k^y - v_{k-1}^y &= q\delta v_k^x - \mu\delta v_k^y - h\delta y_k + \delta x_k
 \end{aligned} \tag{60}$$

We next define our discrete-time Lyapunov function to be,

$$\begin{aligned}
 \mathcal{E}_k = \frac{1}{2} & \left(v_k^x + 2\sqrt{\frac{\sqrt{5}}{3}} \frac{x_k}{\mu} + 2\sqrt{\frac{\sqrt{5}}{3}} \frac{y_k}{\mu} \right)^2 + \frac{1}{2} \left(v_k^y + 2\sqrt{\frac{\sqrt{5}}{3}} \frac{y_k}{\mu} - 2\sqrt{\frac{\sqrt{5}}{3}} \frac{x_k}{\mu} \right)^2 \\
 & + \frac{1}{2} \left((v_k^x)^2 + (v_k^y)^2 \right) + 2\sqrt{5} \left(1 + \frac{2h}{\sqrt{5}} \right) (x_k^2 + y_k^2) \\
 \leq \frac{3}{2} & \left((v_k^x)^2 + \frac{4\sqrt{5}}{3\mu^2} x_k^2 + \frac{4\sqrt{5}}{3\mu^2} y_k^2 \right) + \frac{3}{2} \left((v_k^y)^2 + \frac{4\sqrt{5}}{3\mu^2} y_k^2 + \frac{4\sqrt{5}}{3\mu^2} x_k^2 \right) \\
 & + \frac{1}{2} \left((v_k^x)^2 + (v_k^y)^2 \right) + 2\sqrt{5} \left(1 + \frac{2h}{\sqrt{5}} \right) (x_k^2 + y_k^2) \\
 = 2 & \left((v_k^x)^2 + (v_k^y)^2 \right) + 2\sqrt{5} \left(\frac{2}{\mu^2} + 1 + \frac{2h}{\sqrt{5}} \right) (x_k^2 + y_k^2)
 \end{aligned} \tag{61}$$

where we have used the Cauchy–Schwarz inequality (assuming u_j are scalars):

$$\left(\sum_j^n u_j \right)^2 \leq n \left(\sum_j^n u_j^2 \right)$$

Therefore,

$$\begin{aligned} \mathcal{E}_k - \mathcal{E}_{k-1} &\leq 2 \left(2(v_k^x - v_{k-1}^x) v_k^x - (v_k^x - v_{k-1}^x)^2 \right) + 2 \left(2(v_k^y - v_{k-1}^y) v_k^y - (v_k^y - v_{k-1}^y)^2 \right) \\ &\quad + \sqrt{5} \left(\frac{4}{\mu^2} + 2 \left(1 + \frac{2h}{\sqrt{5}} \right) \right) \left(2(x_k - x_{k-1}) x_k - (x_k - x_{k-1})^2 \right) \\ &\quad + \sqrt{5} \left(\frac{4}{\mu^2} + 2 \left(1 + \frac{2h}{\sqrt{5}} \right) \right) \left(2(y_k - y_{k-1}) y_k - (y_k - y_{k-1})^2 \right) \end{aligned} \quad (62)$$

Plugging our implicitly discretized dynamics of Eq.(60) in the above, we then get,

$$\begin{aligned} \mathcal{E}_k - \mathcal{E}_{k-1} &\leq -4q\delta v_k^x v_k^y - 4\mu\delta(v_k^x)^2 - 4h\delta v_k^x x_k - 4\delta y_k v_k^x - 2(q\delta v_k^y + \mu\delta v_k^x + h\delta x_k + \delta y_k)^2 \\ &\quad + 4q\delta v_k^x v_k^y - 4\mu\delta(v_k^y)^2 - 4h\delta v_k^y y_k + 4\delta x_k v_k^y - 2(q\delta v_k^x - \mu\delta v_k^y - h\delta y_k + \delta x_k)^2 \\ &\quad + 2\sqrt{5} \left(\frac{4}{\mu^2} + 2 \left(1 + \frac{2h}{\sqrt{5}} \right) \right) \delta x_k v_k^x - \sqrt{5} \left(\frac{4}{\mu^2} + 2 \left(1 + \frac{2h}{\sqrt{5}} \right) \right) (\delta v_k^x)^2 \\ &\quad + 2\sqrt{5} \left(\frac{4}{\mu^2} + 2 \left(1 + \frac{2h}{\sqrt{5}} \right) \right) \delta y_k v_k^y - \sqrt{5} \left(\frac{4}{\mu^2} + 2 \left(1 + \frac{2h}{\sqrt{5}} \right) \right) (\delta v_k^y)^2 \end{aligned} \quad (63)$$

If we now impose the condition $\mu_r \delta \geq 1$, then that allows us to rewrite Eq.(63) as,

$$\begin{aligned} \mu\delta(\mathcal{E}_k - \mathcal{E}_{k-1}) &\leq -4q\mu\delta^2 v_k^x v_k^y - 4\mu^2\delta^2(v_k^x)^2 - 4h\mu\delta^2 x_k v_k^x - 4\mu\delta^2 y_k v_k^x - 2(q\delta v_k^y + \mu\delta v_k^x + h\delta x_k + \delta y_k)^2 \\ &\quad + 4q\mu\delta^2 v_k^x v_k^y - 4\mu^2\delta^2(v_k^y)^2 - 4h\mu\delta^2 y_k v_k^y + 4\mu\delta^2 x_k v_k^y - 2(q\delta v_k^x - \mu\delta v_k^y - h\delta y_k + \delta x_k)^2 \\ &\quad + \sqrt{5}\delta^2 \left(\frac{4}{\mu^2} + 2 \left(1 + \frac{2h}{\sqrt{5}} \right) \right) (2\mu x_k - v_k^x) v_k^x + \sqrt{5}\delta^2 \left(\frac{4}{\mu^2} + 2 \left(1 + \frac{2h}{\sqrt{5}} \right) \right) (2\mu y_k - v_k^y) v_k^y \end{aligned} \quad (64)$$

We now further choose to set,

$$q = \sqrt{5} \left(\frac{2 + \mu^2}{\mu} \right) \quad (65)$$

leading the $x_k v_k^x$ and $y_k v_k^y$ terms in Eq.(64) to drop off to leave us with,

$$\begin{aligned} \mu\delta(\mathcal{E}_k - \mathcal{E}_{k-1}) &\leq \left(-6\mu^2\delta^2 - 2q^2\delta^2 - \sqrt{5}\delta^2 \left(\frac{4}{\mu^2} + 2 + \frac{4h}{\sqrt{5}} \right) \right) ((v_k^x)^2 + (v_k^y)^2) \\ &\quad + 8\mu\delta^2(x_k v_k^y - y_k v_k^x) - 2\delta^2(1 + h^2)(x_k^2 + y_k^2) \\ &\leq \left(-6\mu^2\delta^2 - 2q^2\delta^2 - \sqrt{5}\delta^2 \left(\frac{4}{\mu^2} + 2 + \frac{4h}{\sqrt{5}} \right) \right) ((v_k^x)^2 + (v_k^y)^2) \\ &\quad + 8\mu\delta^2(x_k v_k^y - y_k v_k^x) - 2\delta^2(x_k^2 + y_k^2) \end{aligned} \quad (66)$$

Next, by adding to the R.H.S of the above, two positive semi-definite terms of the form,

$$(\delta x_k - 4\mu\delta v_k^y)^2 + (\delta y_k + 4\mu\delta v_k^x)^2 = \delta^2(x_k^2 + y_k^2) - 8\mu\delta^2(x_k v_k^y - y_k v_k^x) + 16\mu^2\delta^2((v_k^y)^2 + (v_k^x)^2) \quad (67)$$

give us,

$$\begin{aligned}
 \mu\delta(\mathcal{E}_k - \mathcal{E}_{k-1}) &\leq \delta^2 \left(10\mu^2 - 2q^2 - \sqrt{5} \left(\frac{4}{\mu^2} + 2 + \frac{4h}{\sqrt{5}} \right) \right) \left((v_k^x)^2 + (v_k^y)^2 \right) - \delta^2 (x_k^2 + y_k^2) \\
 &\leq \delta^2 \left(10\mu^2 - 10 \left(\frac{4}{\mu^2} + 4 + \mu^2 \right) - \sqrt{5} \left(\frac{4}{\mu^2} + 2 + \frac{4h}{\sqrt{5}} \right) \right) \left((v_k^x)^2 + (v_k^y)^2 \right) \\
 &\quad - \delta^2 (x_k^2 + y_k^2) \\
 &\leq \delta^2 \left(-\frac{40}{\mu^2} - 40 - \frac{4\sqrt{5}}{\mu^2} - 2\sqrt{5} - 4h \right) \left((v_k^x)^2 + (v_k^y)^2 \right) - \delta^2 (x_k^2 + y_k^2) \\
 \Rightarrow \mathcal{E}_k - \mathcal{E}_{k-1} &\leq -\frac{\delta}{\mu} \left(4\sqrt{5} \left(\frac{2\sqrt{5}+1}{\mu^2} \right) + 2\sqrt{5} (4\sqrt{5}+1) + 4h \right) \left((v_k^x)^2 + (v_k^y)^2 \right) \\
 &\quad - \frac{\delta}{\mu} (x_k^2 + y_k^2) \\
 &\leq -\frac{\delta}{\mu} \left[\left((v_k^x)^2 + (v_k^y)^2 \right) + (x_k^2 + y_k^2) \right]
 \end{aligned} \tag{68}$$

Multiplying both sides of the above expression by,

$$\sqrt{5} \left(\frac{4}{\mu^2} + 2 + \frac{4h}{\sqrt{5}} \right) \tag{69}$$

we get,

$$\begin{aligned}
 \sqrt{5} \left(\frac{4}{\mu^2} + 2 + \frac{4h}{\sqrt{5}} \right) (\mathcal{E}_k - \mathcal{E}_{k-1}) &\leq -\frac{\sqrt{5}\delta}{\mu} \left(\frac{4}{\mu^2} + 2 + \frac{4h}{\sqrt{5}} \right) \left((v_k^x)^2 + (v_k^y)^2 \right) \\
 &\quad - \frac{\sqrt{5}\delta}{\mu} \left(\frac{4}{\mu^2} + 2 + \frac{4h}{\sqrt{5}} \right) (x_k^2 + y_k^2) \\
 &\leq -\frac{\delta}{\mu} \left[2 \left((v_k^x)^2 + (v_k^y)^2 \right) + \sqrt{5} \left(\frac{4}{\mu^2} + 2 + \frac{4h}{\sqrt{5}} \right) (x_k^2 + y_k^2) \right] \\
 &\leq -\frac{\delta}{\mu} \mathcal{E}_k
 \end{aligned} \tag{70}$$

Therefore, we finally have from Eq.(70),

$$\mathcal{E}_k \leq \left(1 - \frac{\delta\mu}{\mu^2(2\sqrt{5}+4h) + \delta\mu + 4\sqrt{5}} \right) \mathcal{E}_{k-1} \tag{71}$$

Therefore, the rate of convergence of iLEAD in the quadratic min-max game is given by,

$$\begin{aligned}
 \mathcal{E}_k &\leq \left(1 - \frac{\delta\mu}{\mu^2(2\sqrt{5}+4h) + \delta\mu + 4\sqrt{5}} \right)^k \mathcal{E}_0 \\
 \Rightarrow x_k^2 + y_k^2 &\leq \frac{1}{\sqrt{5}} \left(2 + \frac{4h}{\sqrt{5}} \right)^{-1} \left(1 - \frac{\delta\mu}{\mu^2(2\sqrt{5}+4h) + \delta\mu + 4\sqrt{5}} \right)^k \mathcal{E}_0 \\
 &\leq \frac{\mu^2}{\mathcal{C} - 4\sqrt{5}} \left(\frac{\mathcal{C}}{\mathcal{C} + \delta\mu} \right)^k \mathcal{E}_0
 \end{aligned} \tag{72}$$

where $\mathcal{C} = \mu^2 (2\sqrt{5} + 4h) + 4\sqrt{5}$. This, completes our Proof on the convergence of iLEAD in the quadratic min-max setting, to the Nash equilibrium $(0, 0)$ as $k \rightarrow \infty$. Fig. 6 shows that the discrete Lyapunov function (58) decreases over time and thus the dynamics in Eq.(10) is convergent for the quadratic game in Eq.(44).

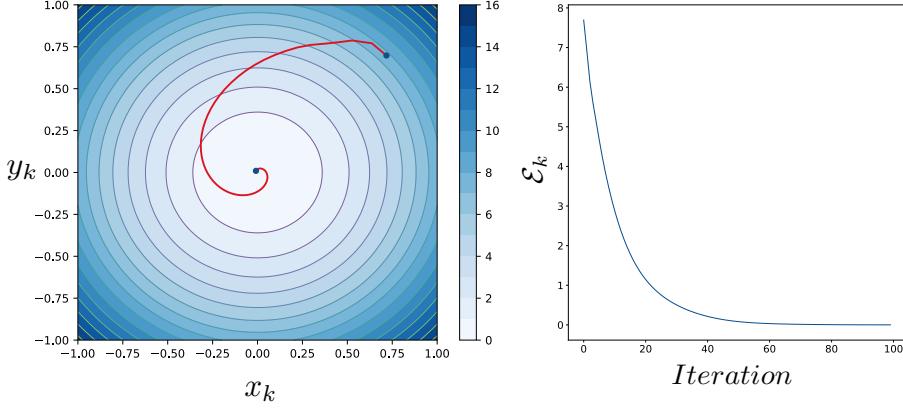


Figure 6. **Left:** Contours of the discrete-time Lyapunov function \mathcal{E}_k , Eq. (58) (black), and convergence trajectory of LEAD (red) in the quadratic min-max game (Eq.(44)) to the Nash equilibrium $(0, 0)$. **Right:** The evolution of the discrete-time Lyapunov function of Eq. (58) over iteration, confirming $\mathcal{E}_k - \mathcal{E}_{k-1} \leq 0 \forall k \in \mathbb{N}$.

H. Proof of Theorem 2

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

$$\mu_{\pm}(\alpha, \beta, \eta) = \frac{1 - (\eta + \alpha)\lambda + \beta \pm \sqrt{\Delta}}{2} \quad (73)$$

where, $\Delta = (1 - (\eta + \alpha)\lambda + \beta)^2 - 4(\beta - \alpha\lambda)$ and $\lambda \in \text{Sp}(\nabla \mathbf{v}(\boldsymbol{\omega}^*))$. Furthermore, for $|\alpha| \ll 1$ and $\beta = 0$, we have,

$$\mu_+(\alpha, \eta) \approx 1 - \eta\lambda + \alpha\lambda\left(\frac{\lambda\eta}{1 - \lambda\eta}\right) + \mathcal{O}(\alpha^2) \quad (74)$$

and

$$\mu_-(\alpha, \eta) \approx -\alpha\lambda\left(\frac{1}{1 - \lambda\eta}\right) + \mathcal{O}(\alpha^2) \quad (75)$$

Proof. For the matrix bilinear game, the Jacobian of \mathbf{v} is given by,

$$\nabla \mathbf{v} \equiv \begin{bmatrix} \nabla_x f(\mathbf{x}_t, \mathbf{y}_t) \\ -\nabla_y f(\mathbf{x}_t, \mathbf{y}_t) \end{bmatrix} = \begin{bmatrix} 0 & \mathbb{A} \\ -\mathbb{A}^\top & 0 \end{bmatrix} \in \mathbb{R}^{2n} \times \mathbb{R}^{2n}. \quad (76)$$

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

$$\mathbb{D} = \begin{bmatrix} \nabla_{xy}^2 f(\mathbf{x}, \mathbf{y}) & 0 \\ 0 & -\nabla_{xy}^2 f(\mathbf{x}, \mathbf{y}) \end{bmatrix}. \quad (77)$$

For the bilinear min-max game, we thus have,

$$\mathbb{D} \equiv \mathbb{D}_q = \begin{bmatrix} \mathbb{A} & 0 \\ 0 & -\mathbb{A}^\top \end{bmatrix} \in \mathbb{R}^{2n} \times \mathbb{R}^{2n}. \quad (78)$$

Consequently, the update rule for LEAD in the case of the bilinear min-max game can be written as:

$$\begin{aligned} \begin{bmatrix} \mathbf{x}_{t+1} \\ \mathbf{y}_{t+1} \end{bmatrix} &= \begin{bmatrix} \mathbf{x}_t \\ \mathbf{y}_t \end{bmatrix} + \beta \begin{bmatrix} \mathbf{x}_t - \mathbf{x}_{t-1} \\ \mathbf{y}_t - \mathbf{y}_{t-1} \end{bmatrix} - \eta \begin{bmatrix} \nabla_x f(\mathbf{x}_t, \mathbf{y}_t) \\ -\nabla_y f(\mathbf{x}_t, \mathbf{y}_t) \end{bmatrix} - \alpha \begin{bmatrix} \nabla_{xy}^2 f(\mathbf{x}_t, \mathbf{y}_t) \Delta \mathbf{y}_t \\ -\nabla_{xy}^2 f(\mathbf{x}_t, \mathbf{y}_t) \Delta \mathbf{x}_t \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{x}_t \\ \mathbf{y}_t \end{bmatrix} + \beta \begin{bmatrix} \mathbf{x}_t - \mathbf{x}_{t-1} \\ \mathbf{y}_t - \mathbf{y}_{t-1} \end{bmatrix} - \eta v - \alpha \mathbb{D}_q \begin{bmatrix} \Delta \mathbf{y}_t \\ \Delta \mathbf{x}_t \end{bmatrix} \end{aligned} \quad (79)$$

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

Let us define the permutation matrix \mathbb{P} ,

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

Thus, Eq. 79 can then be re-expressed as,

$$\begin{aligned} \begin{bmatrix} \omega_{t+1} \\ \omega_t \end{bmatrix} &= \begin{bmatrix} \mathbb{I}_{2n} & 0 \\ \mathbb{I}_{2n} & 0 \end{bmatrix} \begin{bmatrix} \omega_t \\ \omega_{t-1} \end{bmatrix} + \beta \begin{bmatrix} \mathbb{I}_{2n} & -\mathbb{I}_{2n} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \omega_t \\ \omega_{t-1} \end{bmatrix} - \eta \begin{bmatrix} \mathbf{v} \\ 0 \end{bmatrix} \\ &\quad - \alpha \begin{bmatrix} \mathbb{D}_q & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbb{P} & -\mathbb{P} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \omega_t \\ \omega_{t-1} \end{bmatrix} \\ &= \begin{bmatrix} \mathbb{I}_{2n} & 0 \\ \mathbb{I}_{2n} & 0 \end{bmatrix} \begin{bmatrix} \omega_t \\ \omega_{t-1} \end{bmatrix} + \beta \begin{bmatrix} \mathbb{I}_{2n} & -\mathbb{I}_{2n} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \omega_t \\ \omega_{t-1} \end{bmatrix} - \eta \begin{bmatrix} \mathbf{v} \\ 0 \end{bmatrix} \\ &\quad - \alpha \begin{bmatrix} \mathbb{D}_q \mathbb{P} & -\mathbb{D}_q \mathbb{P} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \omega_t \\ \omega_{t-1} \end{bmatrix} \end{aligned} \tag{80}$$

We can now evaluate the Jacobian of F_{LEAD} ,

$$\begin{aligned} \nabla F_{\text{LEAD}} &= \begin{bmatrix} \mathbb{I}_{2n} & 0 \\ \mathbb{I}_{2n} & 0 \end{bmatrix} + \beta \begin{bmatrix} \mathbb{I}_{2n} & -\mathbb{I}_{2n} \\ 0 & 0 \end{bmatrix} - \eta \begin{bmatrix} \nabla \mathbf{v} & 0 \\ 0 & 0 \end{bmatrix} - \alpha \begin{bmatrix} \mathbb{D}_q \mathbb{P} & -\mathbb{D}_q \mathbb{P} \\ 0 & 0 \end{bmatrix} \\ &= \begin{bmatrix} \mathbb{I}_{2n} + \beta \mathbb{I}_{2n} - \eta \nabla \mathbf{v} - \alpha \mathbb{D}_q \mathbb{P} & -\beta \mathbb{I}_{2n} + \alpha \mathbb{D}_q \mathbb{P} \\ \mathbb{I}_{2n} & 0 \end{bmatrix} \end{aligned} \tag{81}$$

It is to be noted that, for the bilinear game we specifically have,

$$\mathbb{D}_q \mathbb{P} = \nabla \mathbf{v}$$

Therefore, Eq. (81) becomes,

$$\nabla F_{\text{LEAD}} = \begin{bmatrix} \mathbb{I}_{2n} + \beta \mathbb{I}_{2n} - (\eta + \alpha) \nabla \mathbf{v} & -\beta \mathbb{I}_{2n} + \alpha \nabla \mathbf{v} \\ \mathbb{I}_{2n} & 0 \end{bmatrix} \tag{82}$$

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,

$$\begin{aligned} \det(X \mathbb{I}_{4n} - \nabla F_{\text{LEAD}}) &= 0 \\ \Rightarrow \det \left(\begin{bmatrix} (X-1) \mathbb{I}_{2n} - \beta \mathbb{I}_{2n} + (\eta + \alpha) \nabla \mathbf{v} & \beta \mathbb{I}_{2n} - \alpha \nabla \mathbf{v} \\ -\mathbb{I}_{2n} & X \mathbb{I}_{2n} \end{bmatrix} \right) &= 0 \\ \Rightarrow \det \left(\left[((X-1)(X-\beta)) \mathbb{I}_{2n} + (X\eta + X\alpha - \alpha) \nabla \mathbf{v} \right] \right) &= 0 \end{aligned} \tag{83}$$

We can now perform an eigen-value decomposition of $\nabla \mathbf{v} = \mathbf{U} \Lambda \mathbf{U}^{-1}$, where Λ is the diagonal matrix of eigenvalues of $\nabla \mathbf{v}$, to write,

$$\begin{aligned} \det(X \mathbb{I}_{4n} - \nabla F_{\text{LEAD}}) &= 0 \\ \Rightarrow \det \left(\left[((X-1)(X-\beta)) \mathbf{U} \mathbf{U}^{-1} + (X\eta + X\alpha - \alpha) \mathbf{U} \Lambda \mathbf{U}^{-1} \right] \right) &= 0 \\ \Rightarrow \det \left(\left[((X-1)(X-\beta)) \mathbb{I}_{2n} + (X\eta + X\alpha - \alpha) \Lambda \right] \right) &= 0 \\ \Rightarrow \prod_{i=1}^{2n} [(X-1)(X-\beta) + (X\eta + X\alpha - \alpha) \lambda_i] &= 0 \end{aligned} \tag{84}$$

with λ_i being the eigen-values of Λ . Therefore,

$$\begin{aligned} X^2 - X (1 - (\eta + \alpha)\lambda_i + \beta) + \beta - \alpha\lambda_i &= 0 \\ \Rightarrow X = \frac{1 - (\eta + \alpha)\lambda_i + \beta \pm \sqrt{\Delta}}{2} \end{aligned} \tag{85}$$

where,

$$\Delta = (1 - (\eta + \alpha) \lambda_i + \beta)^2 - 4(\beta - \alpha \lambda_i) \quad (86)$$

Thus we have,

$$\mu_{\pm}(\alpha, \beta, \eta) = \frac{(1 - (\alpha + \eta)\lambda_i + \beta)}{2} \left(1 \pm \sqrt{1 - \frac{4(\beta - \alpha\lambda_i)}{(1 - (\alpha + \eta)\lambda_i + \beta)^2}} \right) \quad (87)$$

Next, we investigate these eigenvalues for $\beta = 0$ and $|\alpha| \ll 1$ with the help of binomial expansion,

$$\begin{aligned} \mu_+(\alpha, \beta, \eta) &\approx 1 - (\alpha + \eta)\lambda_i + \frac{\alpha\lambda_i}{1 - (\alpha + \eta)\lambda_i} + \mathcal{O}(\alpha^2) \\ &\approx 1 - \eta\lambda_i + \alpha\lambda_i \left(\frac{\eta\lambda_i}{1 - \eta\lambda_i} \right) + \mathcal{O}(\alpha^2) \end{aligned} \quad (88)$$

$$\begin{aligned} \mu_-(\alpha, \beta, \eta) &\approx \frac{(-\alpha\lambda_i)}{1 - (\alpha + \eta)\lambda_i} + \mathcal{O}(\alpha^2) \\ &\approx -\frac{\alpha\lambda_i}{1 - \eta\lambda_i} + \mathcal{O}(\alpha^2) \end{aligned} \quad (89)$$

□

I. Proof of Proposition 3

Proposition. For any $\lambda \in Sp(\nabla \mathbf{v}(\omega^*))$,

$$\nabla_\alpha \rho(0) < 0 \Leftrightarrow \eta \in \left(0, \frac{1}{\text{Im}(\lambda)} \right), \quad (90)$$

where $\text{Im}(\lambda)$ is the imaginary component of λ .

Since $\rho(\alpha, \eta, \lambda) = \max\{|\mu_+|^2, |\mu_-|^2\}$, in the $|\alpha| \ll 1$ regime, we observe from Proposition 3 that $\rho(\alpha, \eta, \lambda) := |\mu_+|^2$. (In the following, we denote $\nabla_\alpha \rho(\alpha = 0) \equiv \nabla_\alpha \rho(0)$ and $\mu_+(\alpha = 0) \equiv \mu_+(0)$.)

$$\begin{aligned} \therefore \nabla_\alpha \rho(0) &= 2\mathcal{R}(\mu_+(0)\bar{\mu}'_+(0)) \quad \mathcal{R} : \text{real part} \\ &= 2\mathcal{R}\left((1 - \eta\lambda)(\frac{\eta\bar{\lambda}^2}{1 - \eta\bar{\lambda}})\right) \\ &= \frac{2\eta}{|1 - \eta\lambda|^2} \mathcal{R}((1 - \eta\lambda)^2 \bar{\lambda}^2) \\ &= \frac{2\eta}{|1 - \eta\lambda|^2} \left(\mathcal{R}(\bar{\lambda}^2) + \eta^2|\lambda|^4 - 2\eta|\lambda|^2\mathcal{R}(\lambda) \right) \end{aligned} \quad (91)$$

where, $\bar{\mu}_+(0), \bar{\lambda}$ correspond to the complex conjugates of $\mu_+(0)$ and λ respectively. Therefore, we observe that the sign of $\nabla_\alpha \rho(\alpha = 0)$ is determined by $\mathcal{R}(\bar{\lambda}^2) + \eta^2|\lambda|^4 - 2\eta|\lambda|^2\mathcal{R}(\lambda)$. Now, for the purely bilinear game, $f(\mathbf{X}, \mathbf{Y}) = \mathbf{X}^T \mathbb{A} \mathbf{Y}$, we have $\forall i, \mathcal{R}(\lambda_i) = 0$, therefore allowing us to write $\lambda = \xi i$. Hence,

$$\nabla_\alpha \rho(0) = \frac{2\eta}{|1 - \eta\lambda|^2} (\eta^2\xi^4 - \xi^2) \quad (92)$$

Choosing $\eta \in \left(0, \frac{1}{\xi} \right)$, we therefore have,

$$\nabla_\alpha \rho(0) < 0 \quad \forall \eta \in \left(0, \frac{1}{\text{Im}(\lambda)} \right), \quad (93)$$

Thus, by having a small positive value α , causes the norm of the limiting eigenvalue μ_+ to decrease.

J. Proof of Theorem 3

Theorem. If we set $\eta = \alpha = \frac{1}{2\sigma_{\max}(\mathbb{A})}$, then we have $\forall \epsilon > 0$,

$$\Delta_{t+1} \in \mathcal{O} \left(\left(\frac{1}{2} + \frac{1}{2} \sqrt{1 - \frac{\sigma_{\min}(\mathbb{A})^2}{\sigma_{\max}(\mathbb{A})^2}} + \epsilon \right)^t \Delta_0 \right) \quad (94)$$

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

Proof: From Eq. (87), we recall that the eigenvalues of $\nabla F_{\text{LEAD}}(\omega^*)$ of the bilinear game $f(\mathbf{X}^T \mathbb{A} \mathbf{Y})$ (for $\beta = 0$),

$$\mu_{\pm}(\alpha, \eta) = \frac{(1 - (\alpha + \eta)\lambda_i)}{2} \left(1 \pm \sqrt{1 + \frac{4\alpha\lambda_i}{(1 - (\alpha + \eta)\lambda_i)^2}} \right) \quad (95)$$

with $\lambda_i \in \text{Sp}(\nabla \mathbf{v}(\omega^*))$. Now, since in the bilinear setting,

$$\nabla \mathbf{v} = \begin{bmatrix} 0 & \mathbb{A} \\ -\mathbb{A}^T & 0 \end{bmatrix} \quad (96)$$

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

$$\mu_{\pm}(\alpha, \eta) = \frac{(1 - (\alpha + \eta)(\pm i\sigma_i))}{2} \left(1 \pm \sqrt{1 + \frac{4\alpha(\pm i\sigma_i)}{(1 - (\alpha + \eta)(\pm i\sigma_i))^2}} \right) \quad (97)$$

Setting $\eta = \alpha = 1/2\sigma_{\max}$, σ_{\max} being the largest singular value of \mathbb{A} , in the above, we then arrive at,

$$|\mu_{\pm}(\sigma_{\max})|^2 = \frac{1}{2} \pm \frac{1}{2} \sqrt{1 - \frac{\sigma_i^2}{\sigma_{\max}^2}} \quad (98)$$

Therefore,

$$\rho = \max\{|\mu_+|^2, |\mu_-|^2\} = \frac{1}{2} + \frac{1}{2} \sqrt{1 - \frac{\sigma_{\min}^2}{\sigma_{\max}^2}} \quad (99)$$

Hence, from Proposition 2 we find,

$$\begin{aligned} \Delta_{t+1} &\leq (\rho + \epsilon) \Delta_t \\ &\leq \mathcal{O} \left(\left(\frac{1}{2} + \frac{1}{2} \sqrt{1 - \frac{\sigma_{\min}^2}{\sigma_{\max}^2}} + \epsilon \right)^t \Delta_0 \right) \end{aligned} \quad (100)$$

where $\Delta_{t+1} := \|\omega_{t+1} - \omega^*\|_2^2 + \|\omega_t - \omega^*\|_2^2$.

K. Experiments Details and implementation

K.1. LEAD-Adam Pseudocode

Algorithm 2 Least Action Dynamics (LEAD)

```

1: Input: learning rate  $\eta$ , momentum  $\beta$ , coupling coefficient  $\alpha$ .
2: Initialize:  $x_0 \leftarrow x_{init}$ ,  $y_0 \leftarrow y_{init}$ ,
    $t \leftarrow 0$ ,
    $m_0^x \leftarrow 0$ ,  $v_0^x \leftarrow 0$   $m_0^y \leftarrow 0$ ,  $v_0^y \leftarrow 0$ 
3: while not converged do
4:    $t \leftarrow t + 1$ 
5:    $g_x \leftarrow \nabla_x f(x_t, y_t)$ 
6:    $g_{xy} \Delta y \leftarrow \nabla_y(g_x)(y_t - y_{t-1})$ 
7:    $g_t^x \leftarrow g_{xy} \Delta y + g_x$ 
8:    $m_t^x \leftarrow \beta_1 \cdot m_{t-1}^x + (1 - \beta_1) \cdot g_t^x$ 
9:    $v_t^x \leftarrow \beta_2 \cdot v_{t-1}^x + (1 - \beta_2) \cdot (g_t^x)^2$ 
10:   $\hat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ 
11:   $\hat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ 
12:   $x_{t+1} \leftarrow x_t - \eta \hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon)$ 
13:   $g_y \leftarrow \nabla_y f(x_{t+1}, y_t)$ 
14:   $g_{xy} \Delta x \leftarrow \nabla_x(g_y)(x_{t+1} - x_t)$ 
15:   $g_t^y \leftarrow g_{xy} \Delta x + g_y$ 
16:   $m_t^y \leftarrow \beta_1 \cdot m_{t-1}^y + (1 - \beta_1) \cdot g_t^y$ 
17:   $v_t^y \leftarrow \beta_2 \cdot v_{t-1}^y + (1 - \beta_2) \cdot (g_t^y)^2$ 
18:   $\hat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ 
19:   $\hat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ 
20:   $y_{t+1} \leftarrow y_t + \eta \hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon)$ 
21: end while
22: return  $(x, y)$ 

```

K.2. Simple Experiment On Quadratics

In this Section, we provide an experimental setting of the quadratic min-max game,

$$f(\mathbf{x}, \mathbf{y}) = \frac{1}{2} \mathbf{x}^T \mathbb{H} \mathbf{x} + \mathbf{x}^T \mathbb{A} \mathbf{y} - \frac{1}{2} \mathbf{y}^T \mathbb{G} \mathbf{y}. \quad (101)$$

where we set, the matrices \mathbb{H} , \mathbb{G} and \mathbb{A} as,

$$\begin{aligned} \mathbb{H} &= \mathbb{G} := \mathbb{R}_{\theta=90^\circ} \Lambda \mathbb{R}_{\theta=90^\circ}^T \\ \mathbb{A} &:= R_{\theta_A} \Lambda \mathbb{R}_{\theta=90^\circ}^T \end{aligned} \quad (102)$$

where,

$$\mathbb{R}_\theta = \begin{bmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{bmatrix}, \quad \Lambda = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \quad (103)$$

We next vary θ_A from 90° (\mathbb{A} fully aligned with \mathbb{H} , \mathbb{G}) to 0° (\mathbb{A} fully unaligned with \mathbb{H} , \mathbb{G}) and compare different methods (Figure 7). For $\theta_A = 90^\circ$, the matrices are simultaneously diagonalizable, implying decoupled dynamics between the different parameters of players \mathbf{x} and \mathbf{y} . As we decrease θ_A , the dynamics become more coupled. We observe that at 0° (fully unaligned), LEAD outperforms Extra-Grad with momentum (the optimal 1st-order method) and all the other 2nd-order methods. We conjecture that superiority of LEAD is the result of the term $\nabla_{xy} f(x, y)(x_k - x_{k-1})$ for player \mathbf{x} and equivalently for player \mathbf{y} . We would like to state that for every angle choice, all the methods are fully tuned (a budget of 1000 hyper-parameters was given to each method).

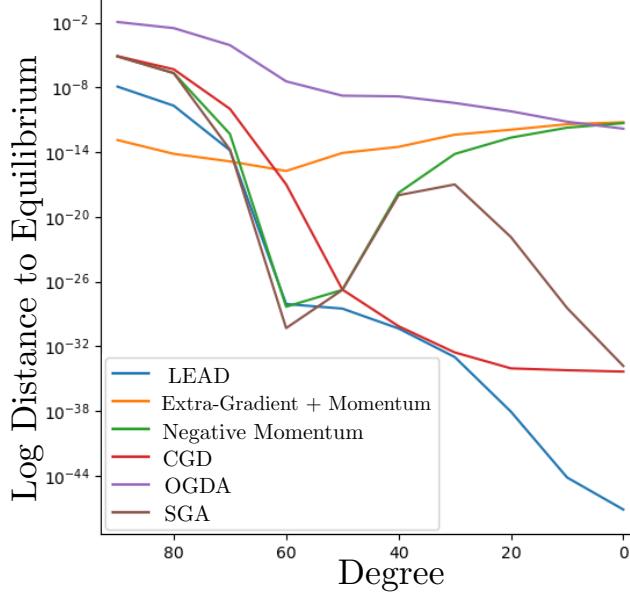


Figure 7. Comparison of the performance of LEAD vs. several other first-order and second-order methods on a variant of the quadratic min-max game. We start with a game where the matrices are all simultaneously diagonalizable and slowly move to the case where they are fully unaligned. We see that LEAD converges faster to the solution of quadratic games whose Jacobian consists of blocks that are not simultaneously diagonalizable.

K.3. 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 3.

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

Table 3. Architecture used for the Mixture of Eight Gaussians.

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.

K.4. 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 4.

Generator	Discriminator
<i>Input:</i> $z \in \mathbb{R}^{100} \sim \mathcal{N}(0, I)$	<i>Input:</i> $x \in \mathbb{R}^{3 \times 32 \times 32}$
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 , 256 → 3; stride: 2; pad: 1)
conv. (ker: 4×4 , 256 → 3; stride: 2; pad: 1)	Tanh
Tanh	Sigmoid

Table 4. Architecture used for CIFAR10 DCGAN.

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.

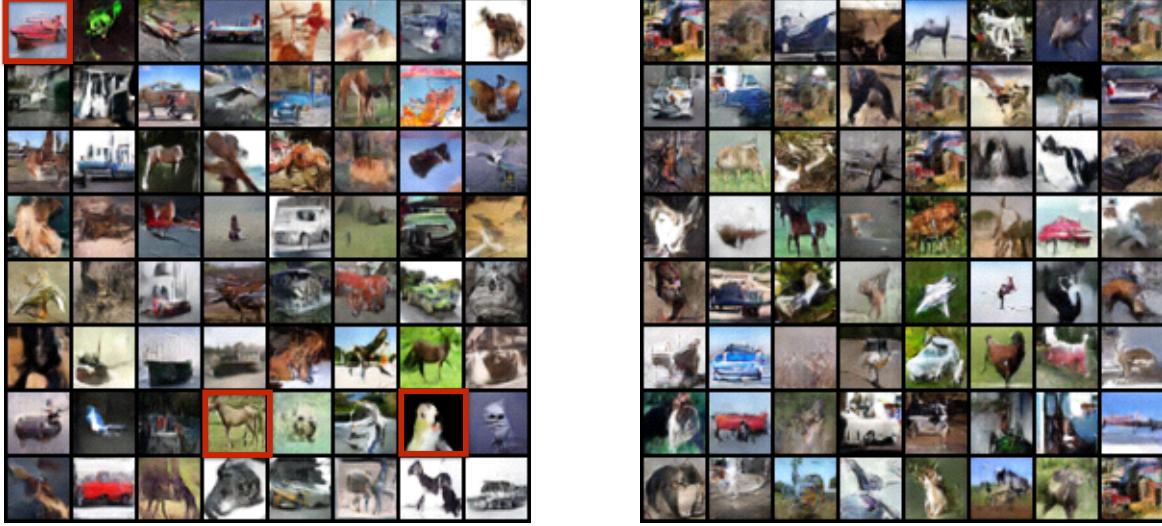


Figure 8. Performance of LEAD on CIFAR-10 image generation task on a DCGAN architecture. **Left:** LEAD achieves FID 19.27. **Right:** Vanilla Adam achieves FID 24.38. LEAD is able to generate better sample qualities from several classes such as ships, horses and birds (red). Best performance is reported after 100 epochs.

K.5. 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 6 for a schematic of the architecture used for the CIFAR10 experiments with ResNet.

		Dis-Block
		<i>Shortcut:</i>
Gen-Block		downsample
<i>Shortcut:</i>		conv. (ker: 1×1 , $3_{\ell=1}/128_{\ell \neq 1} \rightarrow 128$; stride: 1)
Upsample($\times 2$)		Spectral Normalization
<i>Residual:</i>		[AvgPool (ker: 2×2 , stride:2)], if $\ell \neq 1$
Batch Normalization		<i>Residual:</i>
ReLU		[ReLU], if $\ell \neq 1$
Upsample($\times 2$)		conv. (ker: 3×3 , $3_{\ell=1}/128_{\ell \neq 1} \rightarrow 128$; stride: 1; pad: 1)
conv. (ker: 3×3 , $256 \rightarrow 256$; stride: 1; pad: 1)		Spectral Normalization
Batch Normalization		ReLU
ReLU		conv. (ker: 3×3 , $128 \rightarrow 128$; stride: 1; pad: 1)
conv. (ker: 3×3 , $256 \rightarrow 256$; stride: 1; pad: 1)		Spectral Normalization
		AvgPool (ker: 2×2)

Table 5. ResNet blocks used for the ResNet architectures (see Table 6).

Generator	Discriminator
<i>Input:</i> $z \in \mathbb{R}^{64} \sim \mathcal{N}(0, I)$	<i>Input:</i> $x \in \mathbb{R}^{3 \times 32 \times 32}$
Linear($64 \rightarrow 4096$)	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 , $256 \rightarrow 3$; stride: 1; pad:1)	Linear($128 \rightarrow 1$)
$Tanh(\cdot)$	Spectral Normalization

Table 6. ResNet architectures used for experiments on CIFAR10.

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 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 a 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.

L. Comparison to other methods

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

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

¹³For FtR, we have provided the update for the second player given the first player performs gradient descent on f .

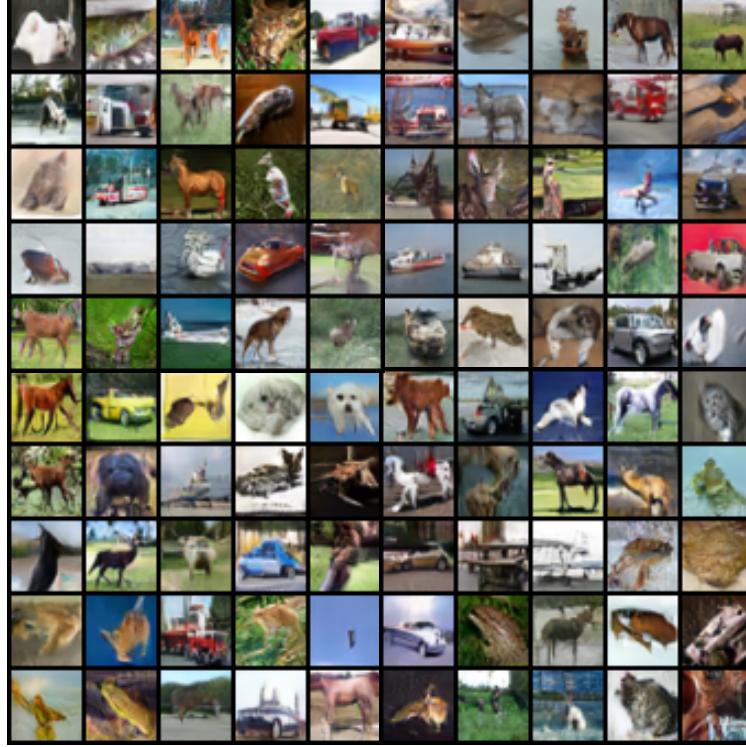


Figure 9. Generated sample of LEAD-Adam on CIFAR-10 after 50k iterations on a ResNet architecture. We achieve an FID score of 11.22 using learning rate $2e - 4$ for the generator and the discriminator, α for the generator is 0.01 and for the discriminator is 0.5.

	Coefficient	Momentum	Gradient	Interaction-xy	Interaction-xx
GDA	$\Delta x_{k+1} =$	1	0	$-\eta \nabla_x f$	$-\eta \nabla_x f$
LEAD	$\Delta \mathbf{x}_{k+1} =$	1	$\beta \Delta \mathbf{x}_k$	$-\eta \nabla_{\mathbf{x}} \mathbf{f}$	$-\alpha \nabla_{\mathbf{x}}^2 \mathbf{f} \Delta \mathbf{y}_k$
SGA ⁽⁶⁾	$\Delta x_{k+1} =$	1	0	$-\eta \nabla_x f$	$-\eta \gamma \nabla_{xy}^2 f \nabla_y f$
CGD ⁽⁴⁶⁾	$\Delta x_{k+1} =$	\mathcal{C}^{-1}	0	$-\eta \nabla_x f$	$-\eta^2 \nabla_{xy}^2 f \nabla_y f$
CO ⁽³⁶⁾	$\Delta x_{k+1} =$	1	0	$-\eta \nabla_x f$	$-\eta \gamma \nabla_{xy}^2 f \nabla_y f$
FtR ⁽⁵¹⁾	$\Delta y_{k+1} =$	1	0	$\eta_y \nabla_y f$	$\eta_x (\nabla_{yy}^2 f)^{-1} \nabla_{yx}^2 f \nabla_x f$
LOLA ⁽¹⁴⁾	$\Delta x_{k+1} =$	1	0	$-\eta \nabla_x f$	$-2\eta \alpha \nabla_{xy} f \nabla_y f$

Table 7. 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 $\mathcal{C} = (I + \eta^2 \nabla_{xy}^2 f \nabla_{yx}^2 f)$. We compare the update rules of the first player¹³ for the following methods: Gradient Descent-Ascent (GDA), symplectic 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.

This gives rise to:

$$x_{k+1} = x_k - \eta \nabla_x f(x_k, y_k) - \eta^2 \nabla_{xy}^2 f(x_k, y_k) \Delta y, \quad (104)$$

$$y_{k+1} = y_k + \eta \nabla_y f(x_k, y_k) + \eta^2 \nabla_{xy}^2 f(x_k, y_k) \Delta x, \quad (105)$$

with $\Delta x, \Delta y$ corresponding to each player accounting for its opponent's potential next step. However, SGA and LookAhead additionally 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):

$$x_{k+1} = x_k - \eta \nabla_x f - \eta^2 \nabla_{xy}^2 f \nabla_y f + \eta^2 \nabla_{xx}^2 f \nabla_x f, \quad (106)$$

which contains an extra 2nd order term $\nabla_{xx}^2 f$ compared to ours. As noted in (Schäfer & Anandkumar, 2019), the $\nabla_{xx}^2 f$ 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 10 and (Schäfer & Anandkumar, 2019) for more discussion.

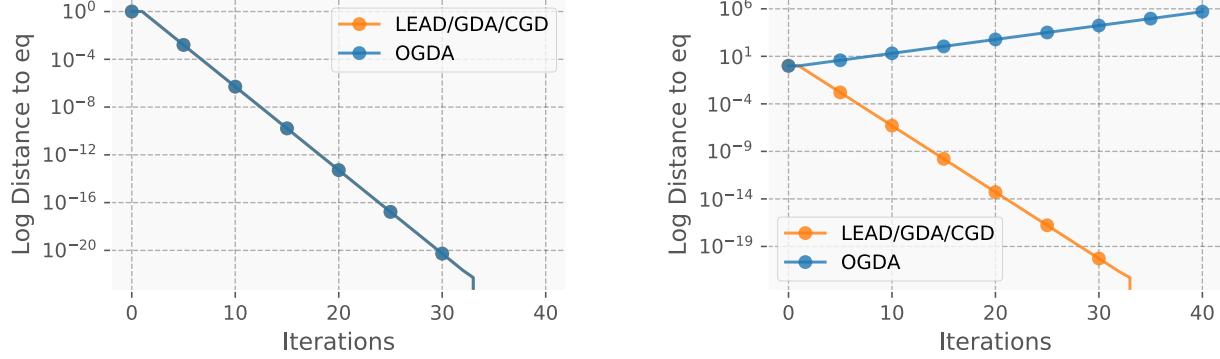


Figure 10. Figure depicting the convergence/divergence of several algorithms on the game of $f(x, y) = \gamma(x^2 - y^2)$ (Nash equilibrium at $x^* = 0, y^* = 0$). **Left:** For $\gamma = 1$, OGDA and LEAD/GDA/CGD (overlays) are found to converge to the Nash eq. **Right:** For $\gamma = 6$, we find that OGDA fails to converge while LEAD/GDA/CGD (overlays) converge. We conjecture that the reason behind this observation is the existence of $\nabla_{xx}^2 f$ term in the optimization algorithm of OGDA.