Addressing Rotational Learning Dynamics in Multi-Agent Reinforcement Learning

Baraah A. M. Sidahmed
CISPA Helmholtz Center for Information Security
Universität des Saarlandes
baraah.adil@cispa.de

Tatjana Chavdarova
Politecnico di Milano
tatjana.chavdarova@polimi.it

Abstract

Multi-Agent Reinforcement Learning (MARL) has become a versatile tool for tackling complex tasks, as agents learn to cooperate and compete across a wide range of applications. Yet, reproducibility remains a persistent hurdle. We pinpoint one key source of instability: the *rotational* dynamics that naturally arise when agents optimize conflicting objectives—dynamics that standard gradient methods struggle to tame. We reframe MARL approaches using Variational Inequalities (VIs), offering a unified framework to address such issues. Leveraging optimization techniques designed for VIs, we propose a general approach for integrating gradient-based VI methods capable of handling rotational dynamics into existing MARL algorithms. Empirical results demonstrate significant performance improvements across benchmarks. In zero-sum games, *Rock–paper–scissors* and *Matching pennies*, VI methods achieve better convergence to equilibrium strategies, and in the *Multi-Agent Particle Environment: Predator-prey*, they also enhance team coordination. These results underscore the transformative potential of advanced optimization techniques in MARL.

1 Introduction

Multi-Agent Reinforcement Learning (MARL) builds on Reinforcement Learning (RL) by addressing environments where multiple agents interact—either by cooperating or competing—to achieve their objectives. In fact, many real-world problems naturally involve multiple decision-makers, making adaptive strategies essential for navigating these dynamic multi-agent settings. Competition among agents can also foster more efficient and robust learning outcomes. As a result, MARL has found applications across diverse domains, including autonomous driving, robotic coordination, financial markets, and multi-player games, showcasing its ability to address complex challenges [see, for example, Omidshafiei et al., 2017, Vinyals et al., 2017, Spica et al., 2018, Zhou et al., 2021, Bertsekas, 2021].

Despite its potential, advancing and deploying MARL research faces significant challenges. The iterative training process in data-driven MARL is notoriously unstable and often struggles to achieve convergence. Gradient-based optimization methods have difficulty in effectively exploring the joint policy space [Li et al., 2023, Christianos et al., 2021], resulting in suboptimal solutions. Additionally, certain MARL structures also exhibit inherent cycling effects [Zheng et al., 2021], further complicating convergence. Crucially, both actor-critic RL and MARL move beyond standard minimization, instead operating within the framework of two- or multi-player games. The introduction of competitive learning objectives and interaction terms generates unique learning dynamics, where conventional gradient descent (GD) methods may fail to converge even in relatively simple scenarios [Korpelevich, 1976]. Performance is also highly sensitive to seemingly minor factors, such as the initial random seed, making reliable benchmarking difficult. While similar issues exist in single-agent actor-critic

18th European Workshop on Reinforcement Learning (EWRL 2025).

RL–a form of two-player game–they are far more severe in MARL, contributing to what has been called the *MARL reproducibility crisis* [Bettini et al., 2024b]. For example, Gorsane et al. [2022] report considerable performance variability across different seeds in widely used MARL benchmarks, such as the *StarCraft* multi-agent challenge[Samvelyan et al., 2019].

In mathematics and numerical optimization, equilibrium-finding problems can be modeled using several frameworks, most notably the *Variational Inequality* [VIs, Stampacchia, 1964, Facchinei and Pang, 2003] framework (see Section 3 for a formal definition). A key limitation of Gradient Descent (GD) in solving simple VI problems stems from the "rotational component" of their associated vector fields [Mescheder et al., 2018, Balduzzi et al., 2018]. For example, the GD method for the $\min_{\boldsymbol{z}_1 \in \mathbb{R}^{d_1}} \max_{\boldsymbol{z}_2 \in \mathbb{R}^{d_2}} \boldsymbol{z}_1 \cdot \boldsymbol{z}_2$ game, rotates around the equilibrium $(0, 0)$ for infinitesimally small learning rates, and diverges away from it for practical choices of its value. Consequently, GD—along with adaptive variants like *Adam* [Kingma and Ba, 2015]—fails to converge for a broad class of equilibrium-seeking problems.

Recent advances in solving variational inequalities (VIs) have been heavily influenced by challenges observed in training generative adversarial networks [GANs, Goodfellow et al., 2014]. This progress spans both theoretical developments—with new convergence guarantees [e.g., Golowich et al., 2020b, Daskalakis et al., 2020b, Gorbunov et al., 2022]—and practical algorithms for large-scale optimization [Diakonikolas, 2020, Chavdarova et al., 2021]. We provide a detailed discussion of these advances in Appendix A.

MARL problems are modeled with *stochastic games* [Littman, 1994]; refer to Section 3. Three main MARL learning paradigms are commonly used:

- *value-based learning*—focuses on estimating so-called *value functions* (e.g., Q-learning, Deep *Q-Networks* [Mnih et al., 2015]) to learn action-values first and infer a policy based on it,
- *policy-based learning*—directly optimizes the *policy* (e.g., *REINFORCE [Williams, 1992]*) by adjusting action probabilities without explicitly learning the value functions, and
- *actor-critic methods*—combines value-based and policy-based approaches where an actor selects actions, and a critic evaluates them.

Furthermore, MARL can be broadly categorized into *centralized* and *independent* learning approaches. In centralized MARL, a global critic or shared value function leverages information from all agents to guide learning, improving coordination [Sunehag et al., 2017, Lowe et al., 2017, Yu et al., 2021]. In contrast, independent MARL treats each agent as a separate learner, promoting scalability but introducing non-stationarity as agents continuously adapt to each other's evolving policies [Matignon et al., 2012, Foerster et al., 2017]. In this work, we focus on *Centralized Training Decentralized Execution* (CTDE) approaches, specifically the ones with centralized critics. Several of the new algorithms in MARL belong to this category such as [MADDPG, Lowe et al., 2017], [MATD3, Ackermann et al., 2019], [MAPPO, Yu et al., 2021], and [COMA, Foerster et al., 2018].

In summary, this paper explores the following:

Can VI methods counteract rotational dynamics in centralized MARL and enhance algorithmic performance?

To address this question, we primarily focus on the CTDE actor-critic MARL learning paradigm, and build VI approaches leveraging a (combination of) *nested-Lookahead-VI* [Chavdarova et al., 2021] and *Extragradient* [Korpelevich, 1976] methods for iteratively solving variational inequalities (VIs). These methods specifically target the rotational component through contraction in rotational spaces. Our key contributions are:

- We formalize MARL optimization through a variational inequality lens.
- We propose *LA-MARL* (Algorithm 1), a scalable approach for neural network-based agents. While presented for actor-critic systems, the method generalizes to other MARL settings. *LA-MARL* is computationally efficient, making it well-suited for large-scale optimization tasks.
- We evaluate our proposed methods against standard optimization techniques in two zero-sum games—*Rock–paper–scissors* and *Matching pennies*—and in two benchmarks from the *Multi-Agent Particle Environments* [MPE, Lowe et al., 2017].

Our code implementation: `https://anonymous.4open.science/r/VI-marl-1436/README.md`.

As a side contribution, we empirically demonstrate the limitations of reward-based performance metrics in MARL and propose an alternative evaluation approach.

2 Related Works

In the following, we discuss related works that study the optimization in centralized MARL. Our approach primarily builds on two key areas, VIs, and MARL, which we review in Appendix A. The necessary VI/MARL background is presented in Section 3. Works focused on optimization in independent MARL are also discussed in Appendix A.

Convergence. Several works rely on two-player zero-sum Markov games to study the regret of an agent relative to a perfect adversary. For instance, Bai and Jin [2020] introduces self-play algorithms for online learning—the *Value Iteration with Upper/Lower Confidence Bound* (VI-ULCB) and an explore-then-exploit algorithm—and show the respective regret bounds. In addition to the online setting, Xie et al. [2020] also consider the offline setting where they propose using Coarse Correlated Equilibria (CCE) instead of Nash Equilibrium (NE) and derive concentration bounds for CCEs.

For the classical *linear quadratic regulator* (LQR) problem [Kalman, 1960], single-agent policy gradient methods are known to exhibit global convergence [Fazel et al., 2018]. The LQR problem extends to the multi-agent setting through *general-sum linear quadratic (LQ) games*, where multiple agents jointly control a (high-dimensional) linear state process. Unlike the zero-sum case [Bu et al., 2019, Zhang et al., 2021], policy gradient faces significant challenges for general-sum LQ games with n players. Mazumdar et al. [2020] highlight a negative result that contrasts sharply with the corresponding result in the single-agent setting. In particular, policy gradient methods fail to guarantee even local convergence in the deterministic setting [Mazumdar et al., 2020], and additional techniques are required to guarantee convergence [Hambly et al., 2023]. Beyond LQ games, policy gradient struggles with multi-agent settings more broadly. Ma et al. [2021] address gradient descent limitations in multi-agent scenarios, introducing an alternative gradient-based algorithm for finding equilibria in *polymatrix games* [Janovskaja, 1968], which model multi-agent interactions with pairwise competition.

Convergence in MARL is challenging due to complex interactions and non-stationarity among agents. While multi-agent actor-critic methods are widely used [Bettini et al., 2024a], their optimization and convergence properties remain underexplored, making this an open problem.

3 Preliminaries

Notation. We denote *(i)* vectors with small bold letters, *(ii)* sets with curly capital letters, *(iii)* real-valued functions with small letters, and *(iv)* operators $\mathcal{Z} \mapsto \mathcal{Z}$ with capital letters, e.g., F. The notation $[n]$ denotes the set $\{1, \dots, n\}$. In the following, let $\mathcal{Z}$ be a convex and compact set in the Euclidean space, equipped with the inner product $\langle \cdot, \cdot \rangle$. We adopt standard MARL notation to describe the setting, as we discuss next.

MARL problem formulation. *Markov games* [MGs, also known as *stochastic games*, Shapley, 1953, Littman, 1994] generalize *Markov Decision Processes* [MDPs Puterman, 1994] to a multi-agent setting. An MG is defined by the tuple:

$$\left(n, \mathcal{S}, \{\mathcal{A}_i\}_{i=1}^n, p, \{r_i\}_{i=1}^n, \gamma\right), \tag{MG}$$

where n agents interact within an environment characterized by a common state space $\mathcal{S}$. Each agent $i \in [n]$ receives observation $\boldsymbol{o}_i \in \mathcal{O}$ of the current state $\boldsymbol{s} \in \mathcal{S}$ of the environment. In the most general case, agent i's observation $\boldsymbol{o}_i = f(\boldsymbol{s})$, where $f \colon \mathcal{S} \to \mathcal{O}_i$. For instance, f can be an identity or coordinate-selection map with $\mathcal{O}_i \subseteq \mathcal{S}$, or f can be a nontrivial mapping. Based on its policy $\pi_i \colon \mathcal{O}_i \to \mathcal{A}_i$, each agent $i \in [n]$ selects an action $a_i \in \mathcal{A}_i$, where $\mathcal{A}_i$ is its finite action set. The joint actions of all agents are represented as $\boldsymbol{a} \triangleq (a_1, \dots, a_n)$, and the joint action space as $\mathcal{A} \triangleq \mathcal{A}_1 \times \cdots \times \mathcal{A}_n$.

The environment transitions to a new state $\boldsymbol{s}' \in \mathcal{S}$ according to a *transition function* $p \colon \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$, where $\Delta(\mathcal{S})$ is the space of probability distributions over $\mathcal{S}$ (non-negative $|\mathcal{S}|$-dimensional vector summing to 1). The function p specifies the probability distribution of the next state $\boldsymbol{s}'$, given the current state $\boldsymbol{s}$ and the joint action $\boldsymbol{a}$.

Each agent $i \in [n]$ receives a reward r_i, where the reward function $r_i \colon \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ depends on the current state, the joint action, and the resulting next state. The importance of future rewards is governed by the *discount factor* $\gamma \in [0, 1)$.

MGs generalize both MDPs and *repeated games* [Aumann, 1995] by introducing non-stationary dynamics, where agents learn their policies jointly and adaptively. Each agent $i \in [n]$ aims to maximize its expected cumulative reward (return), defined as:

$$v_i^{\pi_i, \boldsymbol{\pi}_{-i}}(\boldsymbol{s}) = \mathbb{E}\Big[\sum_{t=0}^{\infty} \gamma^t r_i(\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{s}_{t+1}) | \boldsymbol{s}_0 \sim \boldsymbol{\rho}, \boldsymbol{a}_t \sim \boldsymbol{\pi}(s_t)\Big], \tag{MA-Return}$$

where $\boldsymbol{\pi} \triangleq (\pi_1, \ldots, \pi_n)$ represents the joint policy of all agents, $\boldsymbol{\pi}_{-i}$ denotes the policies of all agents except agent i, and $\boldsymbol{\rho}$ is the initial state distribution. The interaction among agents introduces challenges such as non-stationarity (due to evolving policies) and complex reward interdependencies, leading to a distinct optimization landscape. Solutions often aim to find *Nash equilibria*, where no agent can improve its return by unilaterally altering its policy.

MADDPG. *Multi-agent deep deterministic policy gradient [MADDPG, Lowe et al., 2017]*, extends *Deep deterministic policy gradient* [DDPG, Lillicrap et al., 2015] to multi-agent setting, leveraging a centralized training with decentralized execution paradigm. Each agent $i \in [n]$ has: (i) *critic* network $Q_i \colon \mathcal{O}_1 \times \cdots \times \mathcal{O}_n \times \mathcal{A} \to \mathbb{R}$, parametrized by $\boldsymbol{w}_i \in \mathbb{R}^{d_i^Q}$: acts as a centralized action-value function, evaluating the expected return of joint actions $\boldsymbol{a}$ in state $\boldsymbol{s}$ and (ii) *actor* network $\mu_i \colon \mathcal{O}_i \to \Delta(\mathcal{A}_i)$, parametrized by $\boldsymbol{\theta}_i \in \mathbb{R}^{d_i^\mu}$: represents the agent's policy, mapping agents' observation of states s to a probability distribution over actions a_i.

For stability during training, MADDPG employs *target* networks, which are delayed versions of the critic and actor networks: *target critic* $\bar{Q}_i$, is parametrized by $\bar{\boldsymbol{w}}_i \in \mathbb{R}^{d_i^Q}$, and *target-actor* $\bar{\mu}_i$, parametrized by $\bar{\boldsymbol{\theta}}_i$. The parameters of the target networks are updated using a soft update mechanism:

$$\bar{\boldsymbol{w}}_i \leftarrow \tau \boldsymbol{w}_i + (1-\tau)\bar{\boldsymbol{w}}_i\,, \quad \text{(Target-Critic)} \qquad \bar{\boldsymbol{\theta}}_i \leftarrow \tau \boldsymbol{\theta}_i + (1-\tau)\bar{\boldsymbol{\theta}}_i\,, \quad \text{(Target-Actor)}$$

where $\tau \in (0, 1]$ is a hyperparameter controlling the update rate of the target networks.

MATD3. *Multi-agent TD3 [Ackermann et al., 2019]* improves upon MADDPG by introducing two key modifications: (i) Dual critics: each agent $i \in [n]$ has two critic networks, $Q_{i,1}$ and $Q_{i,2}$. When calculating the target for Q-learning, the smaller of the two values is used, mitigating overestimation bias. (ii) Delayed actor updates: Actor policies and target networks are updated less frequently, typically after every c critic updates, to improve training stability. Additionally, random noise is added to the target actor's outputs during training to introduce exploration and avoid deterministic policies getting stuck in suboptimal regions. These enhancements make MATD3 more robust in multi-agent settings compared to MADDPG.

Variational Inequality [Stampacchia, 1964, Facchinei and Pang, 2003]. Variational Inequalities (VIs) extend beyond standard minimization problems to encompass a broad range of equilibrium-seeking problems. The connection to such general problems can be understood from the optimality condition for convex functions: a point $\boldsymbol{z}^\star$ is an optimal solution if and only if $\langle \boldsymbol{z} - \boldsymbol{z}^\star, \nabla f(\boldsymbol{z}^\star) \rangle \geq 0, \forall \boldsymbol{z} \in \operatorname{dom} f$. In the framework of VIs, this condition is generalized by replacing the gradient field ∇f with a more general vector field F, allowing for the modeling of a wider class of problems. Formally, the VI goal is to find an equilibrium $\boldsymbol{z}^\star$ from the domain of continuous strategies $\mathcal{Z}$, such that:

$$\langle \boldsymbol{z} - \boldsymbol{z}^\star, F(\boldsymbol{z}^\star) \rangle \geq 0, \quad \forall \boldsymbol{z} \in \mathcal{Z}\,, \tag{VI}$$

where $F \colon \mathcal{Z} \to \mathbb{R}^d$, referred to as the *operator*, is continuous, and $\mathcal{Z}$ is a subset of the Euclidean d-dimensional space $\mathbb{R}^d$. VIs are thus characterized by the tuple $(F, \mathcal{Z})$, denoted herein as VI(F, $\mathcal{Z}$). For a more comprehensive introduction to VIs, including examples and applications, see Appendix B.1.

VI methods. The *gradient descent* method straightforwardly extends for the VI problem as follows:

$$\boldsymbol{z}_{t+1} = \boldsymbol{z}_t - \eta F(\boldsymbol{z}_t)\,, \tag{GD}$$

where t denotes the iteration count, and $\eta \in (0, 1)$ the step size or learning rate.

Extragradient [Korpelevich, 1976] is a modification of GD, which uses a "prediction" step to obtain an extrapolated point $\boldsymbol{z}_{t+\frac{1}{2}}$ using GD: $\boldsymbol{z}_{t+\frac{1}{2}} = \boldsymbol{z}_t - \eta F(\boldsymbol{z}_t)$, and the gradients at the *extrapolated* point are then applied to the *current* iterate $\boldsymbol{z}_t$ as follows:

$$\boldsymbol{z}_{t+1} = \boldsymbol{z}_t - \eta F\Big(\boldsymbol{z}_{t+\frac{1}{2}}\Big)\,. \tag{EG}$$

Unlike gradient descent, EG converges in some simple game instances, such as in games linear in both players [Korpelevich, 1976].

The *nested-Lookahead-VI* (LA) algorithm for VI problems [Alg. 3, Chavdarova et al., 2021], is a general wrapper of a "base" optimizer $B\colon \mathbb{R}^n \to \mathbb{R}^n$ where, after every k iterations with B, $\boldsymbol{z}_{t+1} = B(\boldsymbol{z}_t)$ an averaging step is performed as follows:

$$\boldsymbol{z}_{t+k} \leftarrow \boldsymbol{z}_t + \alpha(\boldsymbol{z}_{t+k} - \boldsymbol{z}_t), \quad \alpha \in [0,1]\,. \tag{LA}$$

For this purpose a copy (snapshot) of the iterate after the averaging step is stored for the next LA update. See Appendix B.1.1 for an alternative view.

This averaging can be applied recursively across multiple levels l, when using LA as base optimizer, typically with $l \in [1,3]$. In Algorithm 6, the parameter $k^{(j)}$ at level $j \in [l]$ is defined as the multiple of $k^{(j-1)}$ from the previous level $j-1$, specifically $k^{(j)} = c_j \cdot k^{(j-1)}$. For $l = 1, k = 2$, LA has connections to EG [Chavdarova et al., 2023], however for higher values of k and l the resulting operator exhibits stronger contraction [Chavdarova et al., 2021, Ha and Kim, 2022], which effectively addresses rotational learning dynamics.

4 VI Perspective & Optimization

In this section, we introduce the operators for multi-agent general policy-based learning, actor-critic methods, and the specific operator corresponding to MADDPG. Following this, we present a broader class of algorithms that incorporate a designated MARL operator and integrate it with LA and/or EG.

4.1 MARL Operators

General MARL. Policy-based learning directly solves the (MA-Return) problem, where agents optimize their policy parameters directly to maximize their return. The operator F_{MAR}, where *MAR* stands for *multi-agent-return*, with $\mathcal{Z} \equiv \mathcal{A}$, corresponds to:

$$F_{\text{MAR}}\Big(\begin{bmatrix}\vdots\\ \boldsymbol{\pi}_i\\ \vdots\end{bmatrix}\Big) \triangleq \begin{bmatrix}\vdots\\ \nabla_{\boldsymbol{\pi}_i} v_i^{\boldsymbol{\pi}_i,\boldsymbol{\pi}_{-i}}\\ \vdots\end{bmatrix} \equiv \begin{bmatrix}\vdots\\ \nabla_{\boldsymbol{\pi}_i}\Big(\mathbb{E}\Big[\sum_{t=0}^{\infty}\gamma^t r_i(\boldsymbol{s}_t,\boldsymbol{a}_t,\boldsymbol{s}_{t+1})|\boldsymbol{s}_0\sim\boldsymbol{\rho},\boldsymbol{a}_t\sim\boldsymbol{\pi}(\boldsymbol{s}_t)\Big]\Big)\\ \vdots\end{bmatrix}. \tag{F_{MAR}}$$

Actor-critic MARL. We denote by $\boldsymbol{x}$ the full state information from which the agents observations $\boldsymbol{o}_i$ are derived. As above, consider a centralized critic network, denoted as the Q–network: $\mathbf{Q}_i^{\boldsymbol{\mu}}(\boldsymbol{x}_t, \boldsymbol{a}_t; \boldsymbol{w}_i)$ and an associated policy network $\boldsymbol{\mu}_i(\boldsymbol{o}_i; \boldsymbol{\theta}_i)$ for each agent $i \in [n]$. Given a batch of experiences $\mathcal{B} = \{(\boldsymbol{x}^j, \boldsymbol{a}^j, \boldsymbol{r}^j, \boldsymbol{x}'^j)\}_{j=1}^{|\mathcal{B}|}$, drawn from a replay buffer $\mathcal{D}$, the objective is to find an equilibrium by solving the (VI) with the operator defined as:

$$F_{\text{MAAC}}\Big(\begin{bmatrix}\vdots\\ \boldsymbol{w}_i\\ \boldsymbol{\theta}_i\\ \vdots\end{bmatrix}\Big) \equiv \begin{bmatrix}\vdots\\ \nabla_{\boldsymbol{w}_i}\Big(\frac{1}{|\mathcal{B}|}\sum_{j=1}^{|\mathcal{B}|}\ell_i^{\boldsymbol{w}}(\cdot;\boldsymbol{w}_i,\boldsymbol{\theta}_i)\Big)\\ \nabla_{\boldsymbol{\theta}_i}\Big(\frac{1}{|\mathcal{B}|}\sum_{j=1}^{|\mathcal{B}|}\ell_i^{\boldsymbol{\theta}}(\cdot;\boldsymbol{w}_i,\boldsymbol{\theta}_i)\Big)\\ \vdots\end{bmatrix}, \tag{F_{MAAC}}$$

where the parameter space is $\mathcal{Z} \equiv \mathbb{R}^d$, with $d = \sum_{i=1}^n (d_i^Q + d_i^\mu)$; and *MAAC* stands for *multi-agent-actor-critic*. Even in the single-agent case ($n = 1$), an inherent game-like interaction exists between the actor and the critic: the update of $\boldsymbol{w}_i$ depends on $\boldsymbol{\theta}_i$, and vice versa. This interplay is fundamental to the optimization dynamics in multi-agent actor-critic frameworks.

MADDPG. As an illustrative example, we fully present the terms in (F_{MAAC}) for MADDPG, deferring the other algorithms to Appendix C. The critic and actor loss functions are defined as:

$$\ell_i^{\boldsymbol{w}}(\cdot;\boldsymbol{w}_i,\boldsymbol{\theta}_i) = \big(y_i - \mathbf{Q}_i^{\boldsymbol{\mu}}(\boldsymbol{x}^j,\boldsymbol{a}^j;\boldsymbol{w}_i)\big)^2, y_i = r_i^j + \gamma \mathbf{Q}_i^{\bar{\boldsymbol{\mu}}}(\boldsymbol{x}'^j,\boldsymbol{a}';\boldsymbol{w}_i')\big|_{\boldsymbol{a}'=\bar{\boldsymbol{\mu}}(\boldsymbol{o}'^j)}\,. \tag{$\ell_{\text{MADDPG}}^{\boldsymbol{w}}$}$$

$$\ell_i^{\boldsymbol{\theta}}(\cdot;\boldsymbol{w}_i,\boldsymbol{\theta}_i)=\boldsymbol{\mu}_i(\boldsymbol{o}_i^j;\boldsymbol{\theta}_i)\nabla_{a_i}\,\mathbf{Q}_i^{\boldsymbol{\mu}}(\boldsymbol{x}^j,a_1^j,\ldots,a_i,\ldots,a_n^j;\boldsymbol{w}_i)\Big|_{a_i=\boldsymbol{\mu}_i(\boldsymbol{o}_i^j)}. \qquad (\ell^{\boldsymbol{\theta}}_{\text{MADDPG}})$$

This formulation captures the interplay between the actor and critic networks in MADDPG, where the critic updates its parameters to minimize the Bellman error, while the actor updates its policy by maximizing the Q-value.

4.2 Proposed Methods

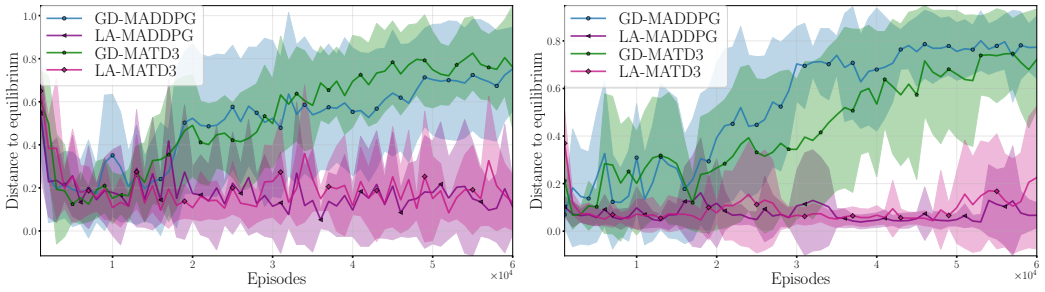
Algorithm 1 LA–MARL Pseudocode.

Input: Environment $\mathcal{E}$, operator F, number of agents n, number of episodes t, action spaces $\{\mathcal{A}_i\}_{i=1}^n$, number of random steps t_{rand}, learning interval t_{learn}, actor networks $\{\boldsymbol{\mu}_i\}_{i=1}^n$ with $\boldsymbol{\theta}\equiv\{\boldsymbol{\theta}_i\}_{i=1}^n$, critic networks $\{Q_i\}_{i=1}^n$ with $\boldsymbol{w}\equiv\{\boldsymbol{w}_i\}_{i=1}^n$, target actor networks $\{\bar{\boldsymbol{\mu}}_i\}_{i=1}^n$ with $\bar{\boldsymbol{\theta}}\equiv\{\bar{\boldsymbol{\theta}}_i\}_{i=1}^n$, target critic networks $\{\bar{Q}_i\}_{i=1}^n$ with $\bar{\boldsymbol{w}}\equiv\{\bar{\boldsymbol{w}}_i\}_{i=1}^n$, learning rates $\eta_{\boldsymbol{\theta}},\eta_{\boldsymbol{w}}$, base optimizer B, discount factor γ, lookahead hyper-parameters $\mathcal{L}\equiv(l,\{k^{(j)}\}_{j=1}^l,\alpha_{\boldsymbol{\theta}},\alpha_{\boldsymbol{w}})$, soft update parameter τ.
Initialize:
  Replay buffer $\mathcal{D}\leftarrow\varnothing$
  Initialize LA parameters: $\boldsymbol{\varphi}\leftarrow\{\boldsymbol{\theta}\}_{\times l},\{\boldsymbol{w}\}_{\times l}$
for all episode $e=1$ to t **do**
  Sample initial state $\boldsymbol{x}$ from $\mathcal{E}$ (with $\boldsymbol{o}\equiv f(\boldsymbol{x})$)
  step $\leftarrow 1$
  repeat
    if step $\leq t_{\text{rand}}$ **then**
      Randomly select actions for each agent i
    else
      Select actions using policy for each agent i
    end if
    Execute $\boldsymbol{a}$, observe rewards and state $(\mathbf{r},\boldsymbol{x}')$
    Store $(\boldsymbol{x},\boldsymbol{a},\mathbf{r},\boldsymbol{x}')$ in replay buffer $\mathcal{D}$
    $\boldsymbol{x}\leftarrow\boldsymbol{x}'$
    if step$\%t_{\text{learn}}==0$ **then**
      Sample a batch $\mathcal{B}$ from $\mathcal{D}$
      Use $\mathcal{B}$ and update to solve VI(F, $\mathbb{R}^d$) using B
      $\bar{\boldsymbol{\theta}}\leftarrow\tau\boldsymbol{\theta}+(1-\tau)\bar{\boldsymbol{\theta}}$ *Update target actor*
      $\bar{\boldsymbol{w}}\leftarrow\tau\boldsymbol{w}+(1-\tau)\bar{\boldsymbol{w}}$ *Update target critic*
    end if
    step $\leftarrow$ step $+1$
  until environment terminates
  NestedLookAhead($n,e,\boldsymbol{\varphi},\mathcal{L}$)
end for
Output: $\boldsymbol{\theta}^{(l-1)}$, $\boldsymbol{w}^{(l-1)}$

To solve the VI problem with an operator corresponding to the MARL algorithm—for instance (F_{MAR}) or (F_{MAAC})—we propose the *LA-MARL*, and *EG-MARL* methods, described in detail in this section.

LA-MARL, Algorithm 1. LA-MARL periodically saves snapshots of all agents' networks (both actor and critic) and averages them with the current networks during training. It operates with a base optimizer (e.g., Adam [Kingma and Ba, 2015], and applies a lookahead averaging step every k intervals. Specifically, the current network parameters $(\boldsymbol{\theta},\boldsymbol{w})$ are updated using their saved snapshots $(\boldsymbol{\theta}^{(j)},\boldsymbol{w}^{(j)})$ through α-averaging (Algorithm 6). The algorithm allows for multiple nested lookahead levels, where higher levels update their corresponding parameters less frequently. All agents apply lookahead updates simultaneously at each step, ensuring consistency across both actor and critic parameters. Extended versions of LA-MARL tailored for MADDPG and MATD3, with more detailed notations, can be found in the appendix (Algorithms 7 and 8).

Generalization and Adaptability. While Algorithm 1 focuses on off-policy actor-critic methods, it serves as a general framework and can be adapted to other MARL learning paradigms: *(i)* Policy Gradient Methods: It can be instantiated with a specific operator—of the form Eq. (F_{MAR})—by



(a) Rock–paper–scissors (b) Matching pennies

Figure 1: **Comparison between *GD-(MADDPG/MATD3)* and *LA-(MADDPG/MATD3)*, on Rock–paper–scissors and Matching pennies.** x-axis: training episodes. y-axis: total distance of agents' policies to the equilibrium policy; averaged over 10 seeds.

setting one set of parameters ($\boldsymbol{w}$ or $\boldsymbol{\theta}$) to ; or *(ii)* On-Policy Learning: modifications include removing the use of target networks.

However, the lookahead method (Eq. LA) must be applied in the joint strategy space of all players. This is crucial because, in multi-agent reinforcement learning (MARL), the adversarial nature of agents' objectives introduces a rotational component in the associated vector field. The averaging steps help mitigate this effect. Particularly, to ensure correct updates, no agent should use parameters that have already been averaged within the same iteration.

(LA-)EG-MARL. For **EG-MARL**, the (EG) update rule is used for both the actor and critic networks and for all agents; refer to Algorithm 2 for a full description. Moreover, Algorithm 1 can also use EG as the base optimizer—represented by B—herein denoted as **LA-EG-MARL**.

Convergence. Under the standard assumption that the operator is *monotone* (see Appendix B.1.1 for the definition), the standard gradient descent method diverges, as shown in prior work [see for instance, Korpelevich, 1976]. This class of operators is broader than—but includes—the case where each agents' and critics' objective functions are convex with respect to their own parameters. In contrast, the LA-EG-MARL methods provably converge for this class of problems [Korpelevich, 1976, Chavdarova et al., 2021, Gorbunov et al., 2022, Pethick et al., 2023]. The key mechanism is that LA increases the contractiveness of the baseline algorithm's fixed-point operator. When applied recursively (nested LA), it further enhances contractiveness, preventing divergence in rotational (competitive) learning dynamics. For more details, see Appendix C.

5 Experiments

5.1 Setup

We use the open-source *PyTorch* implementation of MADDPG [Lowe et al., 2017] and extend it to MATD3 using the same hyperparameters specified in the original papers; detailed in Appendix D. Our experiments cover two zero-sum games—Rock–paper–scissors and Matching Pennies—along with two Multi-Agent Particle Environments (MPE) [Lowe et al., 2017]. We use game implementations from *PettingZoo* [Terry et al., 2021].

Rock–paper–scissors (RPS) and Matching pennies (MP). Widely studied games in multi-agent settings due to their analytically computable mixed Nash equilibria (MNE), which allows for a precise performance measure, and insights into their inherent cyclical behavior [Zhou, 2015, Wang et al., 2014, Srinivasan et al., 2018]. In RPS, two players ($n = 2$) choose among three actions ($m = 3$) per step, with a MNE of $\boldsymbol{\pi}^{\star}_{\text{RPS}}(\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$. Players observe their opponent's previous action before selecting their own, earning $+1$ for a win, 0 for a tie, and -1 for a loss, over $t = 25$ steps per episode. Similarly, MP is a two-player, two-action ($m = 2$) game where one player (*Even*) wins if both actions match, while the other (*Odd*) wins if they differ. The game has a MNE, of $\boldsymbol{\pi}^{\star}_{\text{MP}} = (\frac{1}{2}, \frac{1}{2})$, and runs also for $t = 25$ steps. We compute the squared norm of the learned policy probabilities relative to the equilibrium for both games.

MPE: Predator-Prey and Physical Deception. We evaluate two environments from the Multi-Agent Particle Environments (MPE) benchmark [Lowe et al., 2017]. *Predator-Prey*, has p *good* agents, m

Available at `https://github.com/Git-123-Hub/maddpg-pettingzoo-pytorch/tree/master`.

Table 1: **Competition between agents trained with different algorithms.** Adversary rewards (mean ± std) in Predator-Prey shows that LA exploits GD in direct competition.

#Players	GD vs. GD	GD vs. LA	LA vs. LA	LA vs. GD
$n = 3$	2.99 ± 1.73	$2.14 \pm .91 \downarrow$	5.44 ± 1.27	$7.41 \pm 1.75 \uparrow$
$n = 5$	15.69 ± 7.18	$15.5 \pm 5.32 \downarrow$	14.58 ± 5.45	$22.58 \pm 8.97 \uparrow$

Table 2: ***Equilibrium reached?*** Average adversary win rate for MPE: Physical deception on 100 test environments. *Closer to* 0.5 *is better.*

Method	Adversary Win Rate
Baseline	$0.45 \pm .16$
LA-MADDPG	$0.53 \pm .11$
EG-MADDPG	$0.56 \pm .27$
LA-EG-MADDPG	$\mathbf{0.51 \pm .14}$

adversaries, and l landmarks, where good agents are faster but penalized for being caught or going out of bounds, while adversaries collaborate to capture them. We use $n \equiv (p+m) \in [3, 5]$, and $l = 2$. In *Physical deception*, we have p good agents, one adversary, and p landmarks, with one landmark designated as the *target*. Good agents aim to get close to the target landmark while misleading the adversary, which must infer the target's location. Unlike Predator-Prey, this environment does not involve direct competition for the adversary–its reward depends solely on its own policy. In our experiments, we set $p = 2$.

Methods. We evaluate our methods against the baseline, which is the MARL algorithms (MADDPG or MATD3) with Adam [Kingma and Ba, 2015] as the optimizer B. Throughout the rest of this section, we will refer to baseline as GD-MARL (GD). When referring to LA-based methods, we will indicate the k values for each lookahead level in brackets. For instance, LA $(10, 1000)$ denotes a two-level lookahead where $k^{(1)} = 10$ and $k^{(2)} = 1000$. We denote with *EG* the EG method, and refer to it analogously. Further details on hyperparameters are provided in Appendix D.

5.2 Results

2-player games: RPS & MP. Figures 1(a) and 1(b) illustrate the average distance between learned and equilibrium policies. GD-MARL eventually *diverge*, whereas LA-MARL consistently converges to a near-optimal policy, outperforming the baseline. Both MARL algorithms perform similarly, though MATD3 exhibits lower variance across seeds than MADDPG. For LA-based methods, experiments with different k values indicate that smaller k-values for the innermost LA-averaging yield better performance (see Appendix D.1). The results consistently show performance improvements.

MPE: Predator-prey. Table 1 presents the average rewards of adversaries competing against the good agents. We train agents using GD-MATD3 (baseline) and LA-MATD3 with 5 different seeds each, then evaluate them against one another in 100 test environments. The results show that LA-trained agents not only perform well when matched against other LA-trained agents, but also consistently outperform GD-trained agents in direct competition.

MPE: Physical deception. Table 2 presents the mean and standard deviation of the adversary's win rate, measuring how often it successfully reaches the target. In this setting, equilibrium is achieved when both teams win with equal probability across multiple instances. Given the *cooperative* nature of the game, the baseline performs relatively well, with EG-MADDPG achieving similar performance. However, both LA-MADDPG and LA-EG-MADDPG outperform their respective base optimizers (MADDPG and EG-MADDPG), demonstrating improved stability and effectiveness.

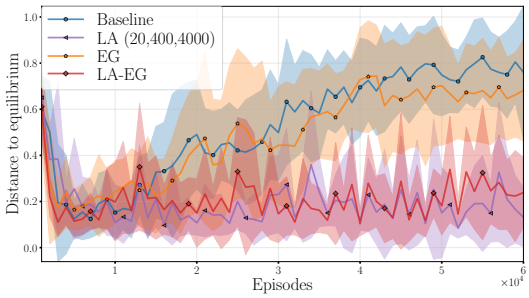
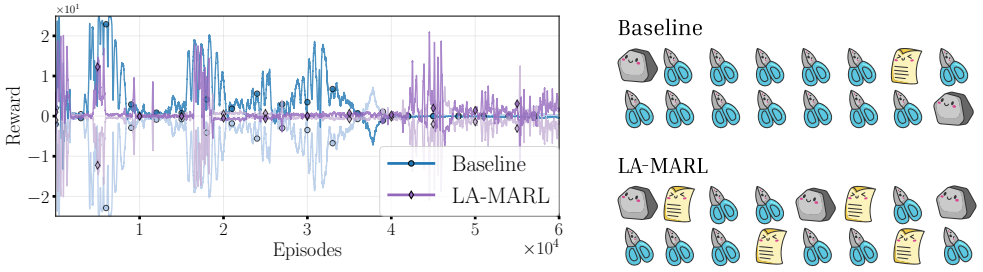


Figure 2: **Comparison between *GD*, *LA*, *EG*, and *LA-EG* optimization methods on the Rock–paper–scissors game.** x-axis: training episodes. y-axis: squared norm of the learned policy probabilities relative to the equilibrium. *EG* uses solely one extrapolation, and thus, as a method, is very close to *GD*; refer to Section 5.2 for a discussion.



(a) Rewards (b) Sampled actions

Figure 3: **Rewards (left) vs. sampled actions from learned policies (right), of (LA-)MADDPG in the Rock–paper–scissors game.** The baseline has saturating rewards (in the last part), however, that is not indicative of the agents' performances. Refer to Section 5.2 for a discussion, and Figure 10 for more detailed plots and larger action samples.

Summary. Overall, our results indicate the following: *(i)* VI-based methods consistently outperform their respective baselines, by effectively handling the rotational dynamics. *(ii)* LA-VI outperforms the other methods.

Comparison among VI methods & insights from GANs. The widely used (EG) for solving VIs is known to converge for monotone VIs. However, in our experiments, EG performs only slightly better than the baseline because it introduces only a minor local adjustment compared to gradient descent (GD). This aligns with expectations: while EG occasionally outperforms GD, its improvements are often marginal. In contrast, Lookahead, introduces a significantly stronger contraction, improving both stability and convergence. As the number of nested levels increases, performance gains become more pronounced—particularly in preventing the last iterate from diverging. However, too many nested levels can lead to overly conservative or slow updates. Based on our experiments, three levels of nested LA yielded the best balance between stability and convergence speed (see Fig. 2 for a comparison of VI methods). Our findings are consistent with those observed in GAN training [Chavdarova et al., 2021], where EG also provides only slight improvements over the baseline, while more contractive methods consistently yield better performance.

These results further confirm that *MARL vector fields in these environments exhibit strong rotational dynamics*. For scenarios with highly competitive reward structures, we recommend using VI methods with greater contractiveness, such as employing multiple levels of LA.

Limitations of Rewards as a Metric in MARL. While saturating rewards are commonly used in MARL, few works challenge their reliability [e.g., Bowling, 2004]. Our results suggest that reward "convergence" *does not necessarily indicate optimal policies*. In multi-agent settings, *rewards can stabilize at a target value even with suboptimal strategies*, leading to misleading evaluations. For example, in Figure 3, baseline agents repeatedly select a subset of actions, resulting in ties that yield a saturating reward but fail to reach equilibrium, making them vulnerable to a more skilled opponent. Conversely, LA-MADDPG agents did not exhibit reward saturation, yet they learned near-optimal policies by randomizing their actions, which aligns with the expected equilibrium. This highlights the need for stronger evaluation metrics in MARL, especially when the true equilibrium is unknown. For further discussion, see Appendix E.5.

6 Discussion

MARL's inherent competitive nature creates complex learning dynamics that standard gradient-based optimization methods—designed for minimization problems—fail to handle effectively. Rather than proposing new learning objectives, this work focuses on a fundamental but often neglected aspect: the optimization process itself.

We address this by adopting a variational inequality (VI) perspective, which provides a unifying framework for MARL learning dynamics. We introduce a general algorithmic framework (Algorithm 1), a computationally efficient VI-based method designed for practical MARL scalability. Our results demonstrate consistent and clear findings that simply replacing the optimizer—while holding all other factors fixed—yields significant improvements. These results demonstrate that optimization methods are a critical yet understudied factor in MARL performance, motivating further research in this direction.

References

J. J. Ackermann, V. Gabler, T. Osa, and M. Sugiyama. Reducing overestimation bias in multi-agent domains using double centralized critics. *ArXiv:1910.01465*, 2019.

G. Arslan and S. Yüksel. Decentralized q-learning for stochastic teams and games. *IEEE Transactions on Automatic Control*, 62:1545–1558, 2015.

R. J. Aumann. *Repeated Games with Incomplete Information*, volume 1 of *MIT Press Books*. The MIT Press, December 1995. ISBN ARRAY(0x6d381400).

W. Azizian, I. Mitliagkas, S. Lacoste-Julien, and G. Gidel. A tight and unified analysis of gradient-based methods for a whole spectrum of differentiable games. In *AISTATS*, pages 2863–2873, 2020.

Y. Bai and C. Jin. Provable self-play algorithms for competitive reinforcement learning. In *ICML*, pages 528–537, 2020.

D. Balduzzi, S. Racaniere, J. Martens, J. Foerster, K. Tuyls, and T. Graepel. The mechanics of n-player differentiable games. In *ICML*, 2018.

A. Bensoussan and J. L. Lions. Contrôle impulsionnel et inéquations quasi variationnelles stationnaires. In *Comptes Rendus Academie Sciences Paris 276*, pages 1279–1284, 1973a.

A. Bensoussan and J. L. Lions. Nouvelle formulation de problèmes de contrôle impulsionnel et applications. In *Comptes Rendus Academie Sciences Paris 276*, pages 1189–1192, 1973b.

A. Bensoussan and J. L. Lions. Nouvelle méthodes en contrôle impulsionnel. In *Applied Mathematics and Optimization*, page 289–312, 1974.

D. Bertsekas. *Rollout, Policy Iteration, and Distributed Reinforcement Learning*. Athena scientific optimization and computation series. Athena Scientific, 2021. ISBN 9781886529076.

M. Bettini, A. Prorok, and V. Moens. Benchmarl: Benchmarking multi-agent reinforcement learning. *Journal of Machine Learning Research*, 25(217):1–10, 2024a.

M. Bettini, A. Prorok, and V. Moens. Benchmarl: Benchmarking multi-agent reinforcement learning. *arXiv:2312.01472*, 2024b.

R. I. Bot, E. R. Csetnek, and P. T. Vuong. The forward-backward-forward method from continuous and discrete perspective for pseudo-monotone variational inequalities in Hilbert spaces. *arXiv:1808.08084*, 2020.

R. I. Bot, E. R. Csetnek, and D.-K. Nguyen. Fast OGDA in continuous and discrete time. *arXiv:2203.10947*, 2022.

M. Bowling. Convergence and no-regret in multiagent learning. In *NIPS*, volume 17. MIT Press, 2004.

J. Bu, L. J. Ratliff, and M. Mesbahi. Global convergence of policy gradient for sequential zero-sum linear quadratic dynamic games. *arXiv:1911.04672*, 2019.

Y. Cai, A. Oikonomou, and W. Zheng. Tight last-iterate convergence of the extragradient method for constrained monotone variational inequalities. *arXiv:2204.09228*, 2022.

T. Chavdarova, G. Gidel, F. Fleuret, and S. Lacoste-Julien. Reducing noise in GAN training with variance reduced extragradient. In *NeurIPS*, 2019.

T. Chavdarova, M. Pagliardini, S. U. Stich, F. Fleuret, and M. Jaggi. Taming GANs with Lookahead-Minmax. In *ICLR*, 2021.

T. Chavdarova, M. I. Jordan, and M. Zampetakis. Last-iterate convergence of saddle point optimizers via high-resolution differential equations. In *Minimax Theory and its Applications*, 2023.

T. Chavdarova, T. Yang, M. Pagliardini, and M. I. Jordan. A primal-dual approach for solving variational inequalities with general-form constraints. In *ICLR*, 2024.

F. Christianos, L. Schäfer, and S. V. Albrecht. Shared experience actor-critic for multi-agent reinforcement learning. *arXiv:2006.07169*, 2021.

R. W. Cottle and G. B. Dantzig. Complementary pivot theory of mathematical programming. *Linear Algebra and its Applications*, 1(1):103–125, 1968. ISSN 0024-3795.

C. Daskalakis and I. Panageas. Last-iterate convergence: Zero-sum games and constrained min-max optimization. In *ITCS*, 2019.

C. Daskalakis, A. Ilyas, V. Syrgkanis, and H. Zeng. Training GANs with optimism. In *ICLR*, 2018.

C. Daskalakis, D. J. Foster, and N. Golowich. Independent policy gradient methods for competitive reinforcement learning. In *NeurIPS*. Curran Associates Inc., 2020a. ISBN 9781713829546.

C. Daskalakis, S. Skoulakis, and M. Zampetakis. The complexity of constrained min-max optimization. *arXiv:2009.09623*, 2020b.

J. Diakonikolas. Halpern iteration for near-optimal and parameter-free monotone inclusion and strong solutions to variational inequalities. In *COLT*, volume 125, 2020.

F. Facchinei and J.-S. Pang. *Finite-dimensional Variational Inequalities and Complementarity Problems*. Springer, 2003.

W. Fan. A comprehensive analysis of game theory on multi-agent reinforcement. *Highlights in Science, Engineering and Technology*, 85:77–88, 03 2024. doi: 10.54097/gv6fpz53.

M. Fazel, R. Ge, S. M. Kakade, and M. Mesbahi. Global convergence of policy gradient methods for the linear quadratic regulator. In *International Conference on Machine Learning*, 2018.

J. Foerster, N. Nardelli, G. Farquhar, T. Afouras, P. H. S. Torr, P. Kohli, and S. Whiteson. Stabilising experience replay for deep multi-agent reinforcement learning. In *ICML*, page 1146–1155. JMLR, 2017.

J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson. Counterfactual multi-agent policy gradients. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

G. Gidel, R. A. Hemmat, M. Pezeshki, R. L. Priol, G. Huang, S. Lacoste-Julien, and I. Mitliagkas. Negative momentum for improved game dynamics. In *AISTATS*, 2019.

N. Golowich, S. Pattathil, and C. Daskalakis. Tight last-iterate convergence rates for no-regret learning in multi-player games. In *NeurIPS*, 2020a.

N. Golowich, S. Pattathil, C. Daskalakis, and A. Ozdaglar. Last iterate is slower than averaged iterate in smooth convex-concave saddle point problems. In *COLT*, pages 1758–1784, 2020b.

I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *NIPS*, 2014.

E. Gorbunov, N. Loizou, and G. Gidel. Extragradient method: $\mathcal{O}(1/K)$ last-iterate convergence for monotone variational inequalities and connections with cocoercivity. In *AISTATS*, 2022.

R. Gorsane, O. Mahjoub, R. de Kock, R. Dubb, S. Singh, and A. Pretorius. Towards a standardised performance evaluation protocol for cooperative marl. *arXiv:2209.10485*, 2022.

Y. Guan, G. Salizzoni, M. Kamgarpour, and T. H. Summers. A policy iteration algorithm for n-player general-sum linear quadratic dynamic games. *arXiv:2410.03106*, 2024.

J. Ha and G. Kim. On convergence of lookahead in smooth games. In *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, volume 151 of *Proceedings of Machine Learning Research*, pages 4659–4684. PMLR, 28–30 Mar 2022.

B. Hambly, R. Xu, and H. Yang. Policy gradient methods find the nash equilibrium in n-player general-sum linear-quadratic games. *Journal of Machine Learning Research*, 24(139):1–56, 2023.

S. Iqbal and F. Sha. Actor-attention-critic for multi-agent reinforcement learning. In *ICML*, 2018.

E. Janovskaja. Equilibrium points in polymatrix games. *Lithuanian Mathematical Journal*, 8(2): 381–384, Apr. 1968. doi: 10.15388/LMJ.1968.20224.

J. Jiang and Z. Lu. I2q: A fully decentralized q-learning algorithm. In *NeurIPS*, volume 35, pages 20469–20481, 2022.

R. E. Kalman. Contributions to the theory of optimal control. *Boletin de la Sociedad Matematica Mexicana*, 5, 1960.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

G. M. Korpelevich. The extragradient method for finding saddle points and other problems. *Matecon*, 1976.

A. Krizhevsky. Learning Multiple Layers of Features from Tiny Images. Master's thesis, 2009.

J. G. Kuba, R. Chen, M. Wen, Y. Wen, F. Sun, J. Wang, and Y. Yang. Trust region policy optimisation in multi-agent reinforcement learning. *arXiv:2109.11251*, 2022.

P. Lancaster and L. Rodman. *Algebraic Riccati Equations*. Oxford Mathematical Monographs. Clarendon Press, 1995. ISBN 9780198537953.

M. Lanctot, J. Schultz, N. Burch, M. O. Smith, D. Hennes, T. Anthony, and J. Perolat. Population-based evaluation in repeated rock-paper-scissors as a benchmark for multiagent reinforcement learning. *arXiv:2303.03196*, 2023.

H. Li and H. He. Multiagent trust region policy optimization. *IEEE transactions on neural networks and learning systems*, PP, 04 2023.

P. Li, J. Hao, H. Tang, Y. Zheng, and X. Fu. Race: improve multi-agent reinforcement learning with representation asymmetry and collaborative evolution. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

S. Li, Y. Wu, X. Cui, H. Dong, F. Fang, and S. Russell. Robust multi-agent reinforcement learning via minimax deep deterministic policy gradient. In *AAAI*, volume 33, pages 4213–4220, 2019.

T. Liang and J. Stokes. Interaction matters: A note on non-asymptotic local convergence of generative adversarial networks. *AISTATS*, 2019.

T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. M. O. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. *CoRR*, abs/1509.02971, 2015.

M. L. Littman. Markov games as a framework for multi-agent reinforcement learning. In *ICML*, ICML'94, page 157–163. Morgan Kaufmann Publishers Inc., 1994. ISBN 1558603352.

R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *NIPS*, 2017.

S. Lu, K. Zhang, T. Chen, T. Başar, and L. Horesh. Decentralized policy gradient descent ascent for safe multi-agent reinforcement learning. *AAAI*, 35(10):8767–8775, 2021. doi: 10.1609/aaai.v35i10.17062.

J. Ma, A. Letcher, F. Schäfer, Y. Shi, and A. Anandkumar. Polymatrix competitive gradient descent. *arXiv:2111.08565*, 2021.

Y. Malitsky. Projected reflected gradient methods for monotone variational inequalities. *SIAM Journal on Optimization*, 25:502–520, 2015.

L. Matignon, G. J. Laurent, and N. Le Fort-Piat. Independent reinforcement learners in cooperative Markov games: a survey regarding coordination problems. *Knowledge Engineering Review*, 27(1): 1–31, Mar. 2012. doi: 10.1017/S026988891200057.

E. Mazumdar, L. J. Ratliff, M. I. Jordan, and S. S. Sastry. Policy-gradient algorithms have no guarantees of convergence in linear quadratic games. In *International Conference on Autonomous Agents and Multiagent Systems*, 2020.

L. Mescheder, A. Geiger, and S. Nowozin. Which Training Methods for GANs do actually Converge? In *ICML*, 2018.

V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015. ISSN 00280836.

D. Monderer and L. S. Shapley. Potential games. *Games and economic behavior*, 1996.

S. Omidshafiei, J. Pazis, C. Amato, J. P. How, and J. Vian. Deep decentralized multi-task multi-agent reinforcement learning under partial observability. In *ICML*, 2017.

T. Pethick, W. Xie, and V. Cevher. Stable nonconvex-nonconcave training via linear interpolation. In *NeurIPS*, 2023.

L. D. Popov. A modification of the arrow–hurwicz method for search of saddle points. *Mathematical Notes of the Academy of Sciences of the USSR, 28(5):845–848*, 1980.

M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., USA, 1st edition, 1994. ISBN 0471619779.

T. Rashid, M. Samvelyan, C. S. de Witt, G. Farquhar, J. Foerster, and S. Whiteson. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. *arXiv:1803.11485*, 2018.

R. T. Rockafellar. Monotone operators associated with saddle-functions and minimax problems. *Nonlinear functional analysis*, 18(part 1):397–407, 1970.

M. Rosca, Y. Wu, B. Dherin, and D. G. T. Barrett. Discretization drift in two-player games. In *ICML*, 2021.

M. Roudneshin, J. Arabneydi, and A. G. Aghdam. Reinforcement learning in nonzero-sum linear quadratic deep structured games: Global convergence of policy optimization. In *IEEE Conference on Decision and Control (CDC)*, page 512–517. IEEE Press, 2020. doi: 10.1109/CDC42340.2020.9303950.

E. K. Ryu, K. Yuan, and W. Yin. Ode analysis of stochastic gradient methods with optimism and anchoring for minimax problems. *arXiv:1905.10899*, 2019.

M. Samvelyan, T. Rashid, C. S. de Witt, G. Farquhar, N. Nardelli, T. G. J. Rudner, C.-M. Hung, P. H. S. Torr, J. Foerster, and S. Whiteson. The starcraft multi-agent challenge. *arXiv:1902.04043*, 2019.

M. O. Sayin, K. Zhang, D. S. Leslie, T. Basar, and A. E. Ozdaglar. Decentralized q-learning in zero-sum markov games. In *NeurIPS*, 2021.

J. Schulman, S. Levine, P. Abbeel, M. I. Jordan, and P. Moritz. Trust region policy optimization. *ArXiv:1502.05477*, 2015.

J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *ArXiv:1707.06347*, 2017.

L. S. Shapley. Stochastic games*. *Proceedings of the National Academy of Sciences*, 39(10): 1095–1100, 1953. doi: 10.1073/pnas.39.10.1095.

K. Son, D. Kim, W. J. Kang, D. E. Hostallero, and Y. Yi. Qtran: Learning to factorize with transformation for cooperative multi-agent reinforcement learning. *arXiv:1905.05408*, 2019.

R. Spica, D. Falanga, E. Cristofalo, E. Montijano, D. Scaramuzza, and M. Schwager. A real-time game theoretic planner for autonomous two-player drone racing. *arXiv:1801.02302*, 2018.

S. Srinivasan, M. Lanctot, V. Zambaldi, J. Perolat, K. Tuyls, R. Munos, and M. Bowling. Actor-critic policy optimization in partially observable multiagent environments. In *NeurIPS*, volume 31, 2018.

G. Stampacchia. Formes bilineaires coercitives sur les ensembles convexes. *Académie des Sciences de Paris*, 258:4413–4416, 1964.

P. Sunehag, G. Lever, A. Gruslys, W. M. Czarnecki, V. F. Zambaldi, M. Jaderberg, M. Lanctot, N. Sonnerat, J. Z. Leibo, K. Tuyls, and T. Graepel. Value-decomposition networks for cooperative multi-agent learning. *ArXiv:1706.05296*, 2017.

J. Terry, B. Black, N. Grammel, M. Jayakumar, A. Hari, R. Sullivan, L. S. Santos, C. Dieffendahl, C. Horsch, R. Perez-Vicente, et al. Pettingzoo: Gym for multi-agent reinforcement learning. *NeurIPS*, 34:15032–15043, 2021.

K. K. Thekumparampil, N. He, and S. Oh. Lifted primal-dual method for bilinearly coupled smooth minimax optimization. In *AISTATS*, 2022.

P. Tseng. On linear convergence of iterative methods for the variational inequality problem. *Journal of Computational and Applied Mathematics*, 60:237–252, 1995. ISSN 0377-0427.

O. Vinyals, T. Ewalds, S. Bartunov, P. Georgiev, A. S. Vezhnevets, M. Yeo, A. Makhzani, H. Küttler, J. Agapiou, J. Schrittwieser, J. Quan, S. Gaffney, S. Petersen, K. Simonyan, T. Schaul, H. van Hasselt, D. Silver, T. Lillicrap, K. Calderone, P. Keet, A. Brunasso, D. Lawrence, A. Ekermo, J. Repp, and R. Tsing. Starcraft ii: A new challenge for reinforcement learning. *arXiv:1708.04782*, 2017.

Z. Wang, B. Xu, and H.-J. Zhou. Social cycling and conditional responses in the rock-paper-scissors game. *Scientific Reports*, 4(1), 2014. ISSN 2045-2322. doi: 10.1038/srep05830.

C.-Y. Wei, Y.-T. Hong, and C.-J. Lu. Online reinforcement learning in stochastic games. In *NIPS*, volume 30. Curran Associates, Inc., 2017.

R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3–4):229–256, 1992. ISSN 0885-6125. doi: 10.1007/BF00992696.

Q. Xie, Y. Chen, Z. Wang, and Z. Yang. Learning zero-sum simultaneous-move markov games using function approximation and correlated equilibrium. In *COLT*, volume 125, pages 3674–3682. PMLR, 09–12 Jul 2020.

T. Yang, M. I. Jordan, and T. Chavdarova. Solving constrained variational inequalities via an interior point method. In *ICLR*, 2023.

Y. Yang and J. Wang. An overview of multi-agent reinforcement learning from game theoretical perspective. *arXiv:2011.00583*, 2021.

C. Yu, A. Velu, E. Vinitsky, Y. Wang, A. M. Bayen, and Y. Wu. The surprising effectiveness of mappo in cooperative, multi-agent games. *ArXiv:2103.01955*, 2021.

C. Yu, A. Velu, E. Vinitsky, J. Gao, Y. Wang, A. Bayen, and Y. Wu. The surprising effectiveness of ppo in cooperative, multi-agent games. *arXiv:2103.01955*, 2022.

K. Zhang, Z. Yang, and T. Basar. Policy optimization provably converges to nash equilibria in zero-sum linear quadratic games. In *NeurIPS*, volume 32, 2019a.

K. Zhang, X. Zhang, B. Hu, and T. Basar. Derivative-free policy optimization for linear risk-sensitive and robust control design: Implicit regularization and sample complexity. In *NeurIPS*, 2021.

M. Zhang, J. Lucas, J. Ba, and G. E. Hinton. Lookahead optimizer: k steps forward, 1 step back. In *NeurIPS*, 2019b.

L. Zheng, T. Fiez, Z. Alumbaugh, B. Chasnov, and L. J. Ratliff. Stackelberg actor-critic: Game-theoretic reinforcement learning algorithms. *arXiv:2109.12286*, 2021.

H.-J. Zhou. The rock–paper–scissors game. *Contemporary Physics*, 57(2):151–163, Mar. 2015. ISSN 1366-5812. doi: 10.1080/00107514.2015.1026556.

M. Zhou, Y. Guan, M. Hayajneh, K. Niu, and C. Abdallah. Game theory and machine learning in uavs-assisted wireless communication networks: A survey. *arXiv:2108.03495*, 2021.

A Extended Related Works Discussion

Our work is primarily grounded in two key areas: Multi-Agent Reinforcement Learning (MARL) and Variational Inequalities (VIs), which we discuss next. Additionally, we extend our discussion on related works on Linear-Quadratic (LQ) games and discuss relevant literature on independent MARL.

Multi-Agent Reinforcement Learning (MARL). Various MARL algorithms have been developed [Lowe et al., 2017, Iqbal and Sha, 2018, Ackermann et al., 2019, Yu et al., 2021], with some extending existing single-agent reinforcement learning (RL) methods [Rashid et al., 2018, Son et al., 2019, Yu et al., 2022, Kuba et al., 2022]. Lowe et al. [2017] extend an actor-critic algorithm to the MARL setting using the *centralized training decentralized execution* framework. In the proposed algorithm, named *multi-agent deep deterministic policy gradient (MADDPG)*, each agent in the game consists of two components: an *actor* and a *critic*. The actor is a policy network that has access only to the local observations of the corresponding agent and is trained to output appropriate actions. The critic is a value network that receives additional information about the policies of other agents and learns to output the Q-value; see Section 3. After a phase of experience collection, a batch is sampled from a replay buffer and used for training the agents. To our knowledge, all deep MARL implementations rely on either stochastic gradient descent or *Adam* optimizer [Kingma and Ba, 2015] to train all networks. Game theory and MARL share many foundational concepts, and several studies explore the relationships between the two fields [Yang and Wang, 2021, Fan, 2024], with some using game-theoretic approaches to model MARL problems [Zheng et al., 2021]. This work proposes incorporating game-theoretic techniques into the optimization process of existing MARL methods to determine if these techniques can enhance MARL optimization.

Li et al. [2019] introduced an algorithm called *M3DDPG*, aimed at enhancing the robustness of learned policies. Specifically, it focuses on making policies resilient to worst-case adversarial perturbations, as well as uncertainties in the environment or the behaviors of other agents.

Variational Inequalities (VIs). VIs were first formulated to understand the equilibrium of a dynamical system [Stampacchia, 1964]. Since then, they have been studied extensively in mathematics, including operational research and network games [see Facchinei and Pang, 2003, and references therein]. More recently, after the shown training difficulties of GANs [Goodfellow et al., 2014]—which are an instance of VIs—an extensive line of works in machine learning studies the convergence of iterative gradient-based methods to solve VIs numerically. Since the last and average iterates can be far apart when solving VIs [see e.g., Chavdarova et al., 2019], these works primarily aimed at obtaining last-iterate convergence for special cases of VIs that are important in applications, including bilinear or strongly monotone games [e.g., Tseng, 1995, Malitsky, 2015, Facchinei and Pang, 2003, Daskalakis et al., 2018, Liang and Stokes, 2019, Gidel et al., 2019, Azizian et al., 2020, Thekumparampil et al., 2022], VIs with cocoercive operators [Diakonikolas, 2020], or monotone operators [Chavdarova et al., 2023, Gorbunov et al., 2022]. Several works *(i)* exploit continuous-time analyses [Ryu et al., 2019, Bot et al., 2020, Rosca et al., 2021, Chavdarova et al., 2023, Bot et al., 2022], *(ii)* establish lower bounds for some VI classes [e.g., Golowich et al., 2020b,a], and *(iii)* study the constrained setting [Daskalakis and Panageas, 2019, Cai et al., 2022, Yang et al., 2023, Chavdarova et al., 2024], among other. Due to the computational complexities involved in training neural networks, iterative methods that rely solely on first-order derivative computation are the most commonly used approaches for solving variational inequalities (VIs). However, standard gradient descent and its momentum-based variants often fail to converge even on simple instances of VIs. As a result, several alternative methods have been developed to address this issue. Some of the most popular first-order methods for solving VIs include the *extragradient* method [Korpelevich, 1976], *optimistic gradient* method [Popov, 1980], *Halpern* method [Diakonikolas, 2020], and (nested) *Lookahead-VI* method [Chavdarova et al., 2021]; these are discussed in detail in Section 3 and Appendix B.1.1. In this work, we primarily focus on the nested Lookahead-VI (LA) method, which has achieved state-of-the-art results on the CIFAR-10 [Krizhevsky, 2009] benchmark for generative adversarial networks [Goodfellow et al., 2014].

General-sum linear quadratic (LQ) games. In LQ games, each agent's action linearly impacts the state process, and their goal is to minimize a quadratic cost function dependent on the state and control actions of both themselves and their opponents. LQ games are widely studied as they admit

global Nash equilibria (NE), which can be analytically computed using coupled algebraic Riccati equations [Lancaster and Rodman, 1995].

Several works establish global convergence for policy gradient methods in zero-sum settings. Zhang et al. [2019a] propose an alternating policy update with projection for deterministic infinite-horizon settings, proving sublinear convergence. Bu et al. [2019] study leader-follower policy gradient in a deterministic setup, and showing sublinear convergence. Zhang et al. [2021] study the sample complexity of policy gradient with alternating policy updates.

For the deterministic n-agent setting, Mazumdar et al. [2020] showed that policy gradient methods fail to guarantee even local convergence. Roudneshin et al. [2020] prove global convergence for policy gradient in a *mean-field* LQ game with infinite horizon and stochastic dynamics. Hambly et al. [2023] show that the *natural policy gradient* method achieves global convergence in finite-horizon general-sum LQ games, provided that a certain condition on an added noise to the system is satisfied. Recently, Guan et al. [2024] proposed a policy iteration method for the infinite horizon setting.

Independent MARL. In independent MARL, each agent learns its policy independently, without direct access to the observations, actions, or rewards of other agents [Matignon et al., 2012, Foerster et al., 2017]. Each agent treats the environment as stationary and ignores the presence of other agents, effectively treating them as part of the environment.

[Daskalakis et al., 2020a] study two-agent zero-sum MARL setting of independent learning algorithms. The authors show that if both players run policy gradient methods jointly, their policies will converge to a min-max equilibrium of the game, as long as their learning rates follow a two-timescale rule. [Arslan and Yüksel, 2015] propose a decentralized Q-learning algorithm for MARL setting where agents have limited information and access solely of their local observations and rewards. Jiang and Lu [2022] proposes a decentralized algorithm. Sayin et al. [2021] explore a decentralized Q-learning algorithm for zero-sum Markov games, where two competing agents learn optimal policies without direct coordination or knowledge of each other's strategies. Each agent relies solely on local observations and rewards, updating their Q-values independently while interacting in a stochastic environment. [Lu et al., 2021] study decentralized cooperative multi-agent setting with coupled safety constraints.

Wei et al. [2017] rely on the framework of average-reward stochastic games to model single player with a perfect adversary, yielding a two-player zero-sum game, in a Markov environment, and study the regret bound.

B Additional Background

In this section, we further discuss VIs, and provide additional background and relevant algorithms.

B.1 VI Discussion

Variational Inequality. We first recall the definition of VIs. A $\mathbf{VI}(F, \mathcal{Z})$ is defined as:

$$\text{find } \boldsymbol{z}^\star \in \mathcal{Z} \quad \text{s.t.} \qquad \langle \boldsymbol{z} - \boldsymbol{z}^\star, F(\boldsymbol{z}^\star) \rangle \geq 0, \quad \forall \boldsymbol{z} \in \mathcal{Z}\,, \tag{VI}$$

where $F \colon \mathcal{Z} \to \mathbb{R}^d$, referred to as the *operator*, is continuous, and $\mathcal{Z}$ is a subset of the Euclidean d-dimensional space $\mathbb{R}^d$.

When $F \equiv \nabla f$ and f is a real-valued function $f \colon \mathcal{Z} \to \mathbb{R}$, the problem VI is equivalent to standard minimization. However, by allowing F to be a more general vector field, VIs also model problems such as finding equilibria in zero-sum and general-sum games [Cottle and Dantzig, 1968, Rockafellar, 1970]. We refer the reader to [Facchinei and Pang, 2003] for an introduction and examples.

To illustrate the relevance of VIs to multi-agent problems, consider the following example. Suppose we have n agents, each with a strategy $\boldsymbol{z}_i \in \mathbb{R}^{d_i}$, and let us denote the joint strategy with

$$\boldsymbol{z} \equiv \begin{bmatrix} \boldsymbol{z}_1 \\ \vdots \\ \boldsymbol{z}_n \end{bmatrix} \in \mathbb{R}^d, \qquad \text{with} \quad d = \sum_{i=1}^{n} d_i\,.$$

Each agent $i \in [n]$ aims to optimize its objective $f_i \colon \mathbb{R}^d \to \mathbb{R}$, which, in the general case, depends on all players' strategies. Then, finding an equilibrium in this game is equivalent to solving a VI where the operator F corresponds to:

$$F_{n\text{-agents}}(\boldsymbol{z}) \equiv \begin{bmatrix} \nabla_{\boldsymbol{z}_1} f_1(\boldsymbol{z}) \\ \vdots \\ \nabla_{\boldsymbol{z}_n} f_n(\boldsymbol{z}) \end{bmatrix}. \tag{$F_{n\text{-agents}}$}$$

An instructive way to understand the difference between non-rotational and rotational learning dynamics is to consider the second-derivative matrix $J \colon R^d \to \mathbb{R}^{d \times d}$ referred herein as the *Jacobian*. For the above ($F_{n\text{-agents}}$) problem the Jacobian is as follows:

$$J_{n\text{-agents}}(\boldsymbol{z}) \equiv \begin{bmatrix} \nabla^2_{\boldsymbol{z}_1^2} f_1(\boldsymbol{z}) & \nabla^2_{\boldsymbol{z}_1 \boldsymbol{z}_2} f_1(\boldsymbol{z}) & \dots & \nabla^2_{\boldsymbol{z}_1 \boldsymbol{z}_n} f_1(\boldsymbol{z}) \\ \vdots & \vdots & \dots & \vdots \\ \nabla^2_{\boldsymbol{z}_n \boldsymbol{z}_1} f_n(\boldsymbol{z}) & \nabla^2_{\boldsymbol{z}_n \boldsymbol{z}_2} f_n(\boldsymbol{z}) & \dots & \nabla^2_{\boldsymbol{z}_n^2} f_n(\boldsymbol{z}) \end{bmatrix}. \tag{$J_{n\text{-agents}}$}$$

Notably, unlike in minimization, where the second-derivative matrix—the so-called *Hessian*—is always symmetric, the Jacobian is not necessarily symmetric. Hence, its eigenvalues may belong to the complex plane. In some cases, the Jacobian of the associated vector field can be decomposed into a symmetric and antisymmetric component [Balduzzi et al., 2018], where each behaves as a *potential* [Monderer and Shapley, 1996] and a *Hamiltonian* (purely rotational) game, resp.

In Section C we will also rely on a more general problem, referred to as the *Quasi Variational Inequality*.

Quasi Variational Inequality. Given a map $F \colon \mathcal{X} \to \mathbb{R}^n$—herein referred as an *operator*—the goal is to:

$$\textit{find} \;\; \boldsymbol{x}^\star \qquad \text{s.t.} \qquad \langle \boldsymbol{x} - \boldsymbol{x}^\star, F(\boldsymbol{x}^\star) \rangle \geq 0, \quad \forall \boldsymbol{x} \in \mathcal{K}(\boldsymbol{x}^\star)\,, \tag{QVI}$$

where $\mathcal{K}(\boldsymbol{x}) \subseteq \mathbb{R}^d$ is a point-to-set mapping from $\mathbb{R}^d$ into subsets of $\mathbb{R}^d$ such that for every $\boldsymbol{x} \in \mathcal{X}$, $\mathcal{K}(\boldsymbol{x}) \subseteq \mathbb{R}^d$ which can be possibly empty.

In other words, the constraint set for QVIs depends on the variable $\boldsymbol{x}$. This contrasts with a standard variational inequality (VI), where the constraint set $\mathcal{K}$ is fixed and does not depend on $\boldsymbol{x}$. QVIs were introduced in a series of works by Bensoussan and Lions [1973a,b, 1974].

B.1.1 VI classes and additional methods

The following VI class is often referred to as the generalized class for VIs to that of convexity in minimization.

Definition B.1 (monotonicity)**.** An operator $F : \mathbb{R}^d \to \mathbb{R}^d$ is *monotone* if $\langle \boldsymbol{z} - \boldsymbol{z}', F(\boldsymbol{z}) - F(\boldsymbol{z}') \rangle \geq 0, \;\; \forall \boldsymbol{z}, \boldsymbol{z}' \in \mathbb{R}^d$. F is μ-strongly monotone if: $\langle \boldsymbol{z} - \boldsymbol{z}', F(\boldsymbol{z}) - F(\boldsymbol{z}') \rangle \geq \mu \|\boldsymbol{z} - \boldsymbol{z}'\|^2$ for all $\boldsymbol{z}, \boldsymbol{z}' \in \mathbb{R}^d$.

The following provides an alternative but equivalent formulation of LA. LA was originally proposed for minimization problems [Zhang et al., 2019b].

LA equivalent formulation. We can equivalently write (LA) as follows. At a step t: *(i)* a copy of the current iterate $\tilde{\boldsymbol{z}}_t$ is made: $\tilde{\boldsymbol{z}}_t \leftarrow \boldsymbol{z}_t$, *(ii)* $\tilde{\boldsymbol{z}}_t$ is updated $k \geq 1$ times using B, yielding $\tilde{\boldsymbol{z}}_{t+k}$, and finally *(iii)* the actual update $\boldsymbol{z}_{t+1}$ is obtained as a *point that lies on a line between* the current $\boldsymbol{z}_t$ iterate and the predicted one $\tilde{\boldsymbol{z}}_{t+k}$:

$$\boldsymbol{z}_{t+1} \leftarrow \boldsymbol{z}_t + \alpha(\tilde{\boldsymbol{z}}_{t+k} - \boldsymbol{z}_t), \quad \alpha \in [0, 1]\,. \tag{LA}$$

In addition to those presented in the main part, we describe the following popular VI method.

Optimistic Gradient Descent (OGD). The update rule of Optimistic Gradient Descent OGD [(OGD) Popov, 1980] is:

$$\boldsymbol{z}_{t+1} = \boldsymbol{z}_t - 2\eta F(\boldsymbol{z}_t) + \eta F(\boldsymbol{z}_{t-1})\,, \tag{OGD}$$

where $\eta \in (0, 1)$ is the learning rate.

B.1.2 Pseudocode for Extragradient

In Algorithm 2 outlines the *Extragradient* optimizer [Korpelevich, 1976], which we employ in EG-MARL. This method uses a gradient-based optimizer to compute the extrapolation iterate, then applies the gradient at the extrapolated point to perform an actual update step. The extragradient optimizer is used to update all agents' actor and critic networks. In our experiments, we use Adam for both the extrapolation and update steps, maintaining the same learning intervals and parameters as in the baseline algorithm.

Algorithm 2 Extragradient optimizer; Can be used as B in algorithm 1.

Input: learning rate η_ψ, initial weights ψ, loss ℓ^ψ, extrapolation steps t
$\psi^{copy} \leftarrow \psi$ *(Save current parameters)*
for $i \in 1, \ldots, t$ **do**
  $\psi = \psi - \eta_\psi \nabla_\psi \ell^\psi(\psi)$ *(Compute the extrapolated ψ)*
end for
$\psi = \psi^{copy} - \eta_\psi \nabla_\psi \ell^\psi(\psi)$ *(update ψ)*
Output: ψ

B.1.3 Pseudocode for Nested Lookahead for a Two-Player Game

For completeness, in Algorithm 3 we give the details of adapted version of the nested Lookahead-Minmax algorithm proposed in [Algorithm 6, Chavdarova et al., 2021] with two-levels.

In the given algorithm, the actor and critic parameters are first updated using a gradient-based optimizer. At interval $k^{(1)}$, backtracking is done between the current parameters and first-level copies (slow weights) and they get updated. At interval $k^{(2)} = c_j k^{(1)}$ backtracking is performed again with second-level copies (slower weights), updating both first- and second-level copies with the averaged version.

Algorithm 3 Pseudocode of Two-Level Nested Lookahead–Minmax. [Chavdarova et al., 2021]

Input: number of episodes t, learning rates $\eta_{\boldsymbol{\theta}}, \eta_{\boldsymbol{w}}$, initial weights $\{\boldsymbol{\theta}, \boldsymbol{\theta}^{(1)}, \boldsymbol{\theta}^{(2)}\}$ and $\{(\boldsymbol{w}, \boldsymbol{w}^{(1)}, \boldsymbol{w}^{(2)})\}$, LA hyperparameters: levels $l = 2$, $(k^{(1)}, k^{(2)})$ and $(\alpha_{\boldsymbol{\theta}}, \alpha_{\boldsymbol{w}})$, losses $\ell^{\boldsymbol{\theta}}$, $\ell^{\boldsymbol{w}}$, real–data distribution p_d, noise–data distribution p_z.
for $r \in 1, \ldots, t$ **do**
  $\boldsymbol{x} \sim p_d$, $\boldsymbol{z} \sim p_z$
  $\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta_{\boldsymbol{w}} \nabla_{\boldsymbol{w}} \ell^{\boldsymbol{w}}(\boldsymbol{w}, \boldsymbol{x}, \boldsymbol{z})$ *(update $\boldsymbol{w}$)*
  $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta_{\boldsymbol{\theta}} \nabla_{\boldsymbol{\theta}} \ell^{\boldsymbol{\theta}}(\boldsymbol{\theta}, \boldsymbol{x}, \boldsymbol{z})$ *(update $\boldsymbol{\theta}$)*
  if $r \% k^{(1)} == 0$ **then**
    $\boldsymbol{w} \leftarrow \boldsymbol{w}^{(1)} + \alpha_{\boldsymbol{w}}(\boldsymbol{w} - \boldsymbol{w}^{(1)})$ *(backtracking on interpolated line $\boldsymbol{w}^{(1)}$, $\boldsymbol{w}$)*
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}^{(1)} + \alpha_{\boldsymbol{\theta}}(\boldsymbol{\theta} - \boldsymbol{\theta}^{(1)})$ *(backtracking on interpolated line $\boldsymbol{\theta}^{(1)}$, $\boldsymbol{\theta}$)*
    $(\boldsymbol{\theta}^{(1)}, \boldsymbol{w}^{(1)}) \leftarrow (\boldsymbol{\theta}, \boldsymbol{w})$ *(update slow checkpoints)*
  end if
  if $r \% k^{(2)} == 0$ **then**
    $\boldsymbol{w} \leftarrow \boldsymbol{w}^{(2)} + \alpha_{\boldsymbol{w}}(\boldsymbol{w} - \boldsymbol{w}^{(2)})$ *(backtracking on interpolated line $\boldsymbol{w}^{(2)}$, $\boldsymbol{w}$)*
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}^{(2)} + \alpha_{\boldsymbol{\theta}}(\boldsymbol{\theta} - \boldsymbol{\theta}^{(2)})$ *(backtracking on interpolated line $\boldsymbol{\theta}^{(2)}$, $\boldsymbol{\theta}$)*
    $(\boldsymbol{\theta}^{(2)}, \boldsymbol{w}^{(2)}) \leftarrow (\boldsymbol{\theta}, \boldsymbol{w})$ *(update super-slow checkpoints)*
    $(\boldsymbol{\theta}_{(1)}, \boldsymbol{w}_{(1)}) \leftarrow (\boldsymbol{\theta}, \boldsymbol{w})$ *(update slow checkpoints)*
  end if
end for
Output: $\boldsymbol{\theta}^{(2)}$, $\boldsymbol{w}^{(2)}$

B.2 MARL algorithms

B.2.1 Details on the MADDPG Algorithm

The MADDPG algorithm [Lowe et al., 2017] is outlined in Algorithm 4. An empty replay buffer $\mathcal{D}$ is initialized to store experiences. In each episode, the environment is reset and agents choose actions

to perform accordingly. After, experiences in the form of *(state, action, reward, next state)* are saved to $\mathcal{D}$.

After a predetermined number of random iterations, learning begins by sampling batches from $\mathcal{D}$. The critic of agent i receives the sampled joint actions $\boldsymbol{a}$ of all agents and the state information of agent i to output the predicted Q-value of agent i. Deep Q-learning [Mnih et al., 2015] is then used to update the critic network; lines 20–21. Then, the agents' policy network is optimized using policy gradient; refer to 23. Finally, following each learning iteration, the target networks are updated towards current actor and critic networks using a fraction τ. Then the process repeats until the end of training.

All networks are optimized using the Adam optimizer [Kingma and Ba, 2015]. Once training is complete, each agent's actor operates independently during execution. This approach is applicable across cooperative, competitive, and mixed environments.

Algorithm 4 Pseudocode for MADDPG [Lowe et al., 2017].

1: **Input:** Environment $\mathcal{E}$, number of agents n, number of episodes t, action spaces $\{\mathcal{A}_i\}_{i=1}^n$, number of random steps t_{rand} before learning, learning interval t_{learn}, actor networks $\{\boldsymbol{\mu}_i\}_{i=1}^n$, with initial weights $\boldsymbol{\theta} \equiv \{\boldsymbol{\theta}_i\}_{i=1}^n$, critic networks $\{\mathbf{Q}_i\}_{i=1}^n$ with initial weights $\boldsymbol{w} \equiv \{\boldsymbol{w}_i\}_{i=1}^n$, learning rates $\eta_{\boldsymbol{\theta}}, \eta_{\boldsymbol{w}}$, optimizer B (e.g., Adam), discount factor γ, soft update parameter τ.
2: **Initialize:**
3: Replay buffer $\mathcal{D} \leftarrow \varnothing$
4: **for all** episode $e \in 1, \ldots, t$ **do**
5: $\boldsymbol{x} \leftarrow$ *Sample*$(\mathcal{E})$ *(sample from environment $\mathcal{E}$)*
6: $step \leftarrow 1$
7: **repeat**
8: **if** $e \leq t_{\text{rand}}$ **then**
9: for each agent i, $a_i \sim \mathcal{A}_i$ *(sample actions randomly)*
10: **else**
11: for each agent i, select action $a_i = \boldsymbol{\mu}_i(\boldsymbol{o}_i) + \mathcal{N}_t$ using current policy and exploration noise
12: **end if**
13: Execute actions $\boldsymbol{a} = (a_1, \ldots, a_n)$, observe rewards $\mathbf{r}$ and new state $\boldsymbol{x}'$ *(apply actions and record results)*
14: replay buffer $\mathcal{D} \leftarrow (\boldsymbol{x}, \boldsymbol{a}, \mathbf{r}, \boldsymbol{x}')$
15: $\boldsymbol{x} \leftarrow \boldsymbol{x}'$
16: *(apply learning step if applicable)*
17: **if** $step \% t_{\text{learn}} = 0$ **then**
18: **for all** agent $i \in 1, \ldots, n$ **do**
19: sample batch $\mathcal{B} : \{(\boldsymbol{x}^j, \boldsymbol{a}^j, \mathbf{r}^j, \boldsymbol{x}'^j)\}_{j=1}^{|\mathcal{B}|}$ from $\mathcal{D}$
20: $y^j \leftarrow r_i^j + \gamma \mathbf{Q}^{\bar{\boldsymbol{\mu}}}(\boldsymbol{x}'^j, a'_1, \ldots, a'_n)$, where $a'_k = \bar{\boldsymbol{\mu}}_k(\boldsymbol{o}'^j_k)$
21: Update critic by minimizing the loss (using optimizer B):
$$\ell(\boldsymbol{w}_i) = \frac{1}{|\mathcal{B}|} \sum_j \left(y^j - \mathbf{Q}_i^{\boldsymbol{\mu}}(\boldsymbol{x}^j, a_1^j, \ldots, a_n^j) \right)^2$$
22: Update actor policy using policy gradient formula and optimizer B
23: $\nabla_{\boldsymbol{\theta}_i} J \approx \frac{1}{|\mathcal{B}|} \sum_j \nabla_{\boldsymbol{\theta}_i} \boldsymbol{\mu}_i(\boldsymbol{o}_i^j) \nabla_{a_i} \mathbf{Q}_i^{\boldsymbol{\mu}}(\boldsymbol{x}^j, a_1^j, \ldots, a_i, \ldots, a_n^j)$, where $a_i = \boldsymbol{\mu}_i(\boldsymbol{o}_i^j)$
24: **end for**
25: **for all** agent $i \in [n]$ **do**
26: $\bar{\boldsymbol{\theta}}_i \leftarrow \tau \boldsymbol{\theta}_i + (1 - \tau) \bar{\boldsymbol{\theta}}_i$ *(update target networks)*
27: $\bar{\boldsymbol{w}}_i \leftarrow \tau \boldsymbol{w}_i + (1 - \tau) \bar{\boldsymbol{w}}_i$
28: **end for**
29: **end if**
30: $step \leftarrow step + 1$
31: **until** environment terminates
32: **end for**
33: **Output:** $\boldsymbol{\theta}$, $\boldsymbol{w}$

B.2.2 MATD3 Algorithm

We provide a psuedo code for MATD3 algorithm from [Ackermann et al., 2019] in algorithm 5. As discussed in the main section, MATD3 was introduced as an improvement to MADDPG and follows a similar structure, except for the learning steps. After sampling a batch from the replay buffer $\mathcal{D}$, both critics of each agent are updated using Deep Q-learning, with the target computed using the minimum of the two critic values (notice the difference in lines 20 and 20of the two algorithms). The actor networks are then updated via policy gradient, using only the Q-value from the first critic; see line 24.

Algorithm 5 Pseudocode for MATD3 [Ackermann et al., 2019].

1: **Input:** Environment $\mathcal{E}$, number of agents n, number of episodes t, action spaces $\{\mathcal{A}_i\}_{i=1}^n$, number of random steps t_{rand} before learning, learning interval t_{learn}, actor networks $\{\boldsymbol{\mu}_i\}_{i=1}^n$, with initial weights $\boldsymbol{\theta} \equiv \{\boldsymbol{\theta}_i\}_{i=1}^n$, both critic networks, $\{\mathbf{Q}_{i,1}, \mathbf{Q}_{i,2}\}_{i=1}^n$ with initial weights $\boldsymbol{w} \equiv \{\boldsymbol{w}_{i,1}, \boldsymbol{w}_{i,2}\}_{i=1}^n$, learning rates $\eta_{\boldsymbol{\theta}}, \eta_{\boldsymbol{w}}$, optimizer B (e.g., Adam), discount factor γ, soft update parameter τ, policy update frequency p.
2: **Initialize:**
3: Replay buffer $\mathcal{D} \leftarrow \varnothing$
4: **for all** episode $e \in 1, \ldots, t$ **do**
5: $\boldsymbol{x} \leftarrow$ *Sample*$(\mathcal{E})$ *(sample from environment $\mathcal{E}$)*
6: $step \leftarrow 1$
7: **repeat**
8: **if** $e \leq t_{\text{rand}}$ **then**
9: for each agent i, $a_i \sim \mathcal{A}_i$ *(sample actions randomly)*
10: **else**
11: for each agent i, select action $a_i = \boldsymbol{\mu}_i(\boldsymbol{o}_i) + \epsilon$ using current policy with some exploration noise
12: **end if**
13: Execute actions $\boldsymbol{a} = (a_1, \ldots, a_n)$, observe rewards $\mathbf{r}$ and new state $\boldsymbol{x}'$ *(apply actions and record results)*
14: replay buffer $\mathcal{D} \leftarrow (\boldsymbol{x}, \boldsymbol{a}, \mathbf{r}, \boldsymbol{x}')$
15: $\boldsymbol{x} \leftarrow \boldsymbol{x}'$
16: *(apply learning step if applicable)*
17: **if** $step \% t_{\text{learn}} = 0$ **then**
18: **for all** agent $i \in [n]$ **do**
19: sample batch $\{(\boldsymbol{x}^j, \boldsymbol{a}^j, \mathbf{r}^j, \boldsymbol{x}'^j)\}_{j=1}^{|\mathcal{B}|}$ from $\mathcal{D}$
20: $y^j \leftarrow r_i^j + \gamma \min_{m=1,2} \mathbf{Q}_{i,m}^{\bar{\boldsymbol{\mu}}}(\boldsymbol{x}'^j, a'_1, \ldots, a'_n)$, where $a'_k = \bar{\boldsymbol{\mu}}_k(\boldsymbol{o}'^j_k) + \epsilon$
21: Update both critics, $m = 1, 2$ by minimizing the loss (using optimizer B):
$$\ell(\boldsymbol{w}_{i,m}) = \frac{1}{|\mathcal{B}|} \sum_j \left(y^j - \mathbf{Q}_{i,m}^{\boldsymbol{\mu}}(\boldsymbol{x}^j, a_1^j, \ldots, a_n^j) \right)^2$$
22: **if** $step \% p = 0$ **then**
23: Update actor policy using policy gradient formula and optimizer B
24: $\nabla_{\boldsymbol{\theta}_i} J \approx \frac{1}{|\mathcal{B}|} \sum_j \nabla_{\boldsymbol{\theta}_i} \boldsymbol{\mu}_i(\boldsymbol{o}_i^j) \nabla_{a_i} \mathbf{Q}_{i,1}^{\boldsymbol{\mu}}(\boldsymbol{x}^j, a_1^j, \ldots, a_i, \ldots, a_n^j)$, where $a_i = \boldsymbol{\mu}_i(\boldsymbol{o}_i^j)$
25: $\bar{\boldsymbol{\theta}}_i \leftarrow \tau \boldsymbol{\theta}_i + (1 - \tau) \bar{\boldsymbol{\theta}}_i$ *(update target networks)*
26: $\bar{\boldsymbol{w}}_{i,m} \leftarrow \tau \boldsymbol{w}_{i,m} + (1 - \tau) \bar{\boldsymbol{w}}_{i,m}$
27: **end if**
28: **end for**
29: **end if**
30: $step \leftarrow step + 1$
31: **until** environment terminates
32: **end for**
33: **Output:** $\boldsymbol{\theta}$, $\boldsymbol{w}$

B.2.3 Counterfactual multi-agent policy gradients (COMA, [Foerster et al., 2018])

COMA is an actor-critic multi-agent algorithm based on the CTDE paradigm, with one centralized critic and n decentralized actors. Additionally, COMA directly addresses the credit assignment

problem in multi-agent settings by: *(i)* computing a counterfactual baseline for each agent $b_i(s, \mathbf{a}_{-i})$, *(ii)* using this baseline to estimate the advantage A_i of the chosen action over all others in $\mathcal{A}_i$, and *(iii)* leveraging this advantage to update individual policies. This ensures that policy updates reflect each agent's true contribution to the overall reward.

B.2.4 Multi-agent Trust Region Policy Optimization (MATRPO, [Li and He, 2023])

Trust Region Policy Optimization [TRPO, Schulman et al., 2015] is a policy optimization method that ensures stable updates by constraining policy changes within a trust region. This constraint is enforced using the KL-divergence, and the update step is computed using natural gradient descent.

Extending TRPO to the cooperative multi-agent setting introduces challenges due to non-stationarity. To address this, MATRPO employs a centralized critic, represented by a central value function $V(\boldsymbol{s})$, which leverages shared information among agents to estimate the Generalized Advantage Estimator (GAE) A_i. The advantage function is then used in the policy gradient update, while ensuring that the KL-divergence constraint is respected, maintaining stable and coordinated learning across agents.

B.2.5 Multi-agent Proximal policy Optimization [MAPPO, Yu et al., 2021]

One of the widely used algorithms in practice is MAPPO, an extension of Proximal Policy Optimization [PPO, Schulman et al., 2017] to the multi-agent setting. Similar to TRPO, PPO ensures that policy updates remain within a small, stable region, but instead of enforcing a KL-divergence constraint, it uses clipping. This clipping mechanism simplifies the optimization process, allowing updates to be performed efficiently using standard gradient ascent methods.

MAPPO is an on-policy algorithm that employs a centralized critic while maintaining decentralized actor networks for each agent. Its critic update follows the same rule as MATRPO, but for the policy update, it optimizes a clipped surrogate objective, which restricts the policy update step size, ensuring stable and efficient learning.

C VI MARL Convergence, Perspectives & Details on the Proposed Algorithms

In this section we extend our discussion on the convergence on VI-MARL operator, then we present the VI operators of additional MARL algorithms within the centralized critic CTDE paradigm. After, we provide detailed versions of Algorithm 1 for MADDPG and MATD3, outlining the full training process when incorporating LA or LA-EG.

C.1 VI MARL Convergence

We first recall the abstract multi-player operator definition from Appendix B.1. Each agent $i \in [n]$ aims to optimize its objective $f_i \colon \mathbb{R}^d \to \mathbb{R}$, which, in the general case, depends on all players' strategies. Then, we have the following operator F:

$$F_{n\text{-agents}}(\boldsymbol{z}) \equiv \begin{bmatrix} \nabla_{\boldsymbol{z}_1} f_1(\boldsymbol{z}) \\ \vdots \\ \nabla_{\boldsymbol{z}_n} f_n(\boldsymbol{z}) \end{bmatrix}, \tag{$F_{n\text{-agents}}$}$$

with the game Jacobian as follows:

$$J_{n\text{-agents}}(\boldsymbol{z}) \equiv \begin{bmatrix} \nabla^2_{\boldsymbol{z}_1^2} f_1(\boldsymbol{z}) & \nabla^2_{\boldsymbol{z}_1 \boldsymbol{z}_2} f_1(\boldsymbol{z}) & \dots & \nabla^2_{\boldsymbol{z}_1 \boldsymbol{z}_n} f_1(\boldsymbol{z}) \\ \vdots & \vdots & \dots & \vdots \\ \nabla^2_{\boldsymbol{z}_n \boldsymbol{z}_1} f_n(\boldsymbol{z}) & \nabla^2_{\boldsymbol{z}_n \boldsymbol{z}_2} f_n(\boldsymbol{z}) & \dots & \nabla^2_{\boldsymbol{z}_n^2} f_n(\boldsymbol{z}) \end{bmatrix}. \tag{$J_{n\text{-agents}}$}$$

More precisely, for multi-agent actor-critic RL we have the following operator:

$$F_{\text{MAAC}}\Bigg(\begin{bmatrix}\vdots\\ \boldsymbol{w}_i\\ \boldsymbol{\theta}_i\\ \vdots\end{bmatrix}\Bigg)\equiv\begin{bmatrix}\vdots\\ \nabla_{\boldsymbol{w}_i}\Big(\frac{1}{|\mathcal{B}|}\sum_{j=1}^{|\mathcal{B}|}\ell_i^{\boldsymbol{w}}(\cdot;\boldsymbol{w}_i,\boldsymbol{\theta})\Big)\\ \nabla_{\boldsymbol{\theta}_i}\Big(\frac{1}{|\mathcal{B}|}\sum_{j=1}^{|\mathcal{B}|}\ell_i^{\boldsymbol{\theta}}(\cdot;\boldsymbol{w}_i,\boldsymbol{\theta}_i)\Big)\\ \vdots\end{bmatrix}, \qquad (F_{\text{MAAC}})$$

where the parameter space is $\mathcal{Z}\equiv\mathbb{R}^d$, with $d=\sum_{i=1}^n(d_i^Q+d_i^\mu)$; and *MAAC* stands for *multi-agent-actor-critic*.

Then, we can notice by computing the Jacobian of the above operator that the eigenvalues are in the complex plane. Applying lookahead results in interpolating the largest eigenvalue (in magnitude) with the point (1,0) in the complex plane, thus reducing the spectral radius of the Jacobian. Furthermore, applying this recursively (nested Lookahead) leads to larger contraction.

To make this more precise, consider the gradient descent operator as a base optimizer

$$T_{GD}\equiv I-\alpha F\,,$$

where α is the step size vector.

Let λ denote the eigenvalue of $J^{base}\triangleq\nabla T_{GD}(\cdot)$ with largest modulus, *i.e.* $\rho(J^{base}(\cdot)=|\lambda|$, let $\boldsymbol{u}$ be its associated eigenvector: $J^{base}\boldsymbol{u}=\lambda\boldsymbol{u}$.

The Jacobian of Lookahead is then:

$$J^{LA}=\nabla F^{LA}(\cdot)=(1-\alpha)I+\alpha(J^{base})^k\,.$$

The power k rotates the eigenvector in the complex plane; see [Chavdarova et al., 2021]. By noticing that:

$$\begin{aligned}J^{LA}\boldsymbol{u}=&((1-\alpha)I+\alpha(J^{base})^k)\boldsymbol{u}\\ =&((1-\alpha)+\alpha\lambda^k)\boldsymbol{u}\,,\end{aligned}$$

we deduce $\boldsymbol{u}$ is an eigenvector of J^{LA} with eigenvalue $1-\alpha+\alpha\lambda^k$. Thus, this is strictly closer to the unit ball in the complex plane, increasing the contractiveness.

C.2 VI MARL Perspectives

In the main text, we introduced the general VI operator for multi-agent actor-critic algorithms (F_{MAAC}) and provided the specific equations for MADDPG in ($\ell_{\text{MADDPG}}^{\boldsymbol{w}}$ & $\ell_{\text{MADDPG}}^{\boldsymbol{\theta}}$), with the operator corresponding to:

$$F_{\text{MADDPG}}\Bigg(\begin{bmatrix}\vdots\\ \boldsymbol{w}_i\\ \boldsymbol{\theta}_i\\ \vdots\end{bmatrix}\Bigg)\equiv\begin{bmatrix}\vdots\\ \nabla_{\boldsymbol{w}_i}\Big(\frac{1}{|\mathcal{B}|}\sum_{j=1}^{|\mathcal{B}|}\Big(r_i^j+\gamma\mathbf{Q}_i^{\bar{\boldsymbol{\mu}}}\,(\boldsymbol{x}'^j,\boldsymbol{a}';\boldsymbol{w}_i')\big|_{\boldsymbol{a}'=\bar{\boldsymbol{\mu}}(\boldsymbol{o}'^j)}-\mathbf{Q}_i^{\boldsymbol{\mu}}(\boldsymbol{x}^j,\boldsymbol{a}^j;\boldsymbol{w}_i)\Big)^2\Big)\\ \nabla_{\boldsymbol{\theta}_i}\Big(\frac{1}{|\mathcal{B}|}\sum_{j=1}^{|\mathcal{B}|}\boldsymbol{\mu}_i(\boldsymbol{o}_i^j;\boldsymbol{\theta}_i)\nabla_{a_i}\,\mathbf{Q}_i^{\boldsymbol{\mu}}(\boldsymbol{x}^j,a_1^j,\ldots,a_i,\ldots,a_n^j;\boldsymbol{w}_i)\big|_{a_i=\boldsymbol{\mu}_i(\boldsymbol{o}_i^j)}\Big)\\ \vdots\end{bmatrix}, \qquad (F_{\text{MADDPG}})$$

where the parameter space is $\mathcal{Z}\equiv\mathbb{R}^d$, with $d=\sum_{i=1}^n(d_i^Q+d_i^\mu)$.

We now show how update equations for several well-known MARL algorithms—that follow the CTDE paradigm with a centralized critic—can be written as a VI. Our VI-based methods can also be applied to these algorithms using the operators below.

For a more general notation, for each agent $i\in[n]$ we assume:

(i) central critic network (one or multiple) that estimates either action value Q–Network($\boldsymbol{s},\boldsymbol{a}$): $\mathbf{Q}_i(\boldsymbol{x}_t,\boldsymbol{a}_t;\boldsymbol{w}_i)$, or state value V–network(s): $\mathbf{V}_i(\boldsymbol{x}_t;\boldsymbol{w}_i)$, and

(ii) a decentralized policy network that can be deterministic $\boldsymbol{\mu}_i(\boldsymbol{o}_i;\boldsymbol{\theta}_i)$ or stochastic $\boldsymbol{\pi}_i(\boldsymbol{o}_i;\boldsymbol{\theta}_i)$, depending on the algorithm.

Given a batch of experiences $\mathcal{B}$: $(\boldsymbol{x}^j, \boldsymbol{a}^j, \mathbf{r}^j, \boldsymbol{x}'^j)$, sampled from a replay buffer ($\mathcal{D}$), we provide the necessary equations and the final operator (F) for each of the following popular MARL algorithms.

C.2.1 MATD3

The VI formulation for MATD3 is very similar to MADDPG, except here, for each agent, we have two critic networks; we write: $\boldsymbol{w}_i \equiv \{\boldsymbol{w}_{i,1}, \boldsymbol{w}_{i,2}\}$. Accordingly, target computation for the critic ($Q_{i,m}$) is calculated by taking the minimum of both critic networks, but only the value of critic 1 is used for the actor (policy network) update. We have:

$$F_{\text{MATD3}}\Bigg(\begin{bmatrix}\vdots\\ \boldsymbol{w}_{i,1}\\ \boldsymbol{w}_{i,2}\\ \boldsymbol{\theta}_i\\ \vdots\end{bmatrix}\Bigg)\equiv$$

$$\begin{bmatrix}
\vdots \\
\nabla_{\boldsymbol{w}_{i,1}}\Big(\frac{1}{|\mathcal{B}|}\sum_{j=1}^{|\mathcal{B}|}\Bigg(\underbrace{r_i^j+\gamma\min_{m\in\{1,2\}}\mathbf{Q}_{i,m}^{\bar{\boldsymbol{\mu}}}\left(\boldsymbol{x}'^j,a_1',\ldots,a_n'\right)\big|_{\boldsymbol{a}'=\bar{\boldsymbol{\mu}}(\boldsymbol{o}'^j)}}_{\text{target } y_i}-\mathbf{Q}_{i,1}^{\boldsymbol{\mu}}(\boldsymbol{x}^j,\boldsymbol{a}^j;\boldsymbol{w}_{i,1})\Bigg)^2\Big) \\
\nabla_{\boldsymbol{w}_{i,2}}\Big(\frac{1}{|\mathcal{B}|}\sum_{j=1}^{|\mathcal{B}|}\Bigg(\underbrace{r_i^j+\gamma\min_{m\in\{1,2\}}\mathbf{Q}_{i,m}^{\bar{\boldsymbol{\mu}}}\left(\boldsymbol{x}'^j,a_1',\ldots,a_n'\right)\big|_{\boldsymbol{a}'=\bar{\boldsymbol{\mu}}(\boldsymbol{o}'^j)}}_{\text{target } y_i}-\mathbf{Q}_{i,2}^{\boldsymbol{\mu}}(\boldsymbol{x}^j,\boldsymbol{a}^j;\boldsymbol{w}_{i,2})\Bigg)^2\Big) \\
\Big(\frac{1}{|\mathcal{B}|}\sum_{j=1}^{|\mathcal{B}|}\nabla_{\boldsymbol{\theta}_i}\boldsymbol{\mu}_i(\boldsymbol{o}_i^j;\boldsymbol{\theta}_i)\nabla_{a_i}\mathbf{Q}_{i,1}^{\boldsymbol{\mu}}(\boldsymbol{x}^j,a_1^j,\ldots,a_i,\ldots,a_n^j)\big|_{a_i=\boldsymbol{\mu}_i(\boldsymbol{o}_i^j)}\Big) \\
\vdots
\end{bmatrix}. \tag{F_{MATD3}}$$

C.2.2 COMA

In COMA, critic is trained using a $TD(\lambda)$ target (y^λ) computed from a target network parameterized by $\bar{\boldsymbol{w}}$ that get updated to main network weights every couple iterations. Given the following Advantage A_i calculations:

$$A_i(\boldsymbol{x},\boldsymbol{a}) = Q(\boldsymbol{x},\boldsymbol{a}) - b_i(\boldsymbol{x},\boldsymbol{a}_{-i})$$
$$b_i(\boldsymbol{x},\boldsymbol{a}_{-i}) = \sum_{a_i}\boldsymbol{\pi}_i(a_i|\boldsymbol{o}_i)Q(\boldsymbol{x},(a_i,\boldsymbol{a}_{-i}))\,,$$

the operator for *COMA* corresponds to:

$$F_{\text{COMA}}\Bigg(\begin{bmatrix}\vdots\\ \boldsymbol{w}_i\\ \boldsymbol{\theta}_i\\ \vdots\end{bmatrix}\Bigg)\equiv\begin{bmatrix}\vdots\\ \nabla_{\boldsymbol{w}_i}\mathbb{E}\left[\left(y_i^\lambda-Q_i(\boldsymbol{x}^j,\boldsymbol{a};\boldsymbol{w}_i)\right)^2\right]\\ \mathbb{E}\left[\nabla_{\boldsymbol{\theta}_i}\sum_i A_i(\boldsymbol{x},\boldsymbol{a})\log\boldsymbol{\pi}_{\boldsymbol{\theta}_i}(a_i|\boldsymbol{o}_i)\right]\\ \vdots\end{bmatrix}. \tag{F_{COMA}}$$

C.2.3 MAPPO

As previously noted, MAPPO can be seen as a simplified version of MATRPO. It shares a similar critic loss with MATRPO but simplifies the actor loss by using a clipped objective instead of a KL

constraint, making the optimization problem more tractable. This allows it to be formulated as a VI, as shown below:

$$\hat{V}_t = (1-\lambda)\sum_{n=1}^{T}\lambda^{n-1}\left(\sum_{k=0}^{n-1}\gamma^k r_{t+k} + \gamma^n \mathbf{V}(\boldsymbol{o}_i')\right),$$

$$F_{\text{MAPPO}}\left(\begin{bmatrix}\vdots\\ \boldsymbol{w}_i\\ \boldsymbol{\theta}_i\\ \vdots\end{bmatrix}\right) \equiv \begin{bmatrix}\vdots\\ \nabla_{\boldsymbol{w}_i}\mathbb{E}\left[\left(\mathbf{V}(\boldsymbol{x};\boldsymbol{w}_i) - \hat{V}_t\right)^2\right]\\ \nabla_{\boldsymbol{\theta}_i}\mathbb{E}\left[\min\left\{\frac{\boldsymbol{\pi}_{\theta_i}(a_i|\boldsymbol{o}_i)}{\boldsymbol{\pi}_{\theta_i^{old}}(a_i|\boldsymbol{o}_i)}A_i^{\theta_i^{old}}, \text{clip}\left(\frac{\boldsymbol{\pi}_{\theta_i}(a_i|\boldsymbol{o}_i)}{\boldsymbol{\pi}_{\theta_i^{old}}(a_i|\boldsymbol{o}_i)}, 1-\epsilon, 1+\epsilon\right)A_i^{\theta_i^{old}}\right\}\right]\\ \vdots\end{bmatrix}. \quad (F_{\text{MAPPO}})$$

C.3 Detailed Algorithms

Herein we provide procedure NestedLookahead called from algorithm 1 to compute the extrapolations and after present two pseudocodes considered as extended versions of the main algorithm in algorithm 1; in which we detail how the lookahead approach can be integrated in the training process of MADDPG and MATD3.

C.3.1 Nested Lookahead algorithm

In algorithm 6 below we share a detailed version of Nested lookahead procedure called from algorithms 1, 7 and 8.

Algorithm 6 Pseudocode for LA-VI, called from Algorithm 1. Updates the parameters in-place.

procedure NESTEDLOOKAHEAD**:**
  Input: #agents n, episode counter e, actor and critic weights and snapshots: $\{(\boldsymbol{\theta}_i, \boldsymbol{\theta}_i^{(1)}, \ldots, \boldsymbol{\theta}_i^{(l)})\}_{i=1}^n$ and $\{(\boldsymbol{w}_i, \boldsymbol{w}_i^{(1)}, \ldots, \boldsymbol{w}_i^{(l)})\}_{i=1}^n$, LA hyperparameters: levels l, $(k^{(1)}, \ldots, k^{(l)})$ and $(\alpha_{\boldsymbol{\theta}}, \alpha_{\boldsymbol{w}})$.
  for all $j \in [l]$ **do**
    if $e \% k^{(j)} == 0$ **then**
      for all agent $i \in [n]$ **do**
        $\boldsymbol{w}_i \leftarrow \boldsymbol{w}_i^{(j)} + \alpha_{\boldsymbol{w}}(\boldsymbol{w}_i - \boldsymbol{w}_i^{(j)})$ *LA j^{th} level*
        $\boldsymbol{\theta}_i \leftarrow \boldsymbol{\theta}_i^{(j)} + \alpha_{\boldsymbol{\theta}}(\boldsymbol{\theta}_i - \boldsymbol{\theta}_i^{(j)})$
        $(\boldsymbol{\theta}_i^{(1)}, \ldots, \boldsymbol{\theta}_i^{(j)}, \boldsymbol{w}_i^{(1)}, \ldots, \boldsymbol{w}_i^{(j)}) \leftarrow (\{\boldsymbol{\theta}_i\}_{\times j}, \{\boldsymbol{w}_i\}_{\times j})$ *Update copies up to j^{th}*
      end for
    end if
  end for
end procedure

C.3.2 Extended version of LA-MADDPG pseudocode

We include an extended version for the LA-MADDPG algorithm without VI notations in algorithm 7.

C.3.3 Extended version of LA-MATD3 pseudocode

We include an extended version for the LA-MATD3 algorithm without VI notations in algorithm 8.

Algorithm 7 Pseudocode for LA–MADDPG: MADDPG with (Nested) Lookahead.

Input: Environment $\mathcal{E}$, number of agents n, number of episodes t, action spaces $\{\mathcal{A}_i\}_{i=1}^n$, number of random steps t_{rand} before learning, learning interval t_{learn}, actor networks $\{\boldsymbol{\mu}_i\}_{i=1}^n$, with initial weights $\boldsymbol{\theta} \equiv \{\boldsymbol{\theta}_i\}_{i=1}^n$, critic networks $\{\mathbf{Q}_i\}_{i=1}^n$ with initial weights $\boldsymbol{w} \equiv \{\boldsymbol{w}_i\}_{i=1}^n$, learning rates $\eta_{\boldsymbol{\theta}}, \eta_{\boldsymbol{w}}$, base optimizer B (e.g., Adam), discount factor γ, lookahead hyperparameters $\mathcal{L} \equiv (l, \{k^{(j)}\}_{j=1}^l, \alpha_{\boldsymbol{\theta}}, \alpha_{\boldsymbol{w}})$, soft update parameter τ.
Initialize:
  Replay buffer $\mathcal{D} \leftarrow \varnothing$
  LA parameters: $\boldsymbol{\varphi} \leftarrow \{\boldsymbol{\theta}\}_{\times l}, \{\boldsymbol{w}\}_{\times l}$ *(store snapshots for LA)*
for all episode $e \in 1, \ldots, t$ **do**
  $\boldsymbol{x} \leftarrow$ *Sample*$(\mathcal{E})$ *(sample from environment $\mathcal{E}$)*
  $step \leftarrow 1$
  repeat
    if $e \leq t_{\text{rand}}$ **then**
      for each agent i, $a_i \sim \mathcal{A}_i$ *(sample actions randomly)*
    else
      for each agent i, select action a_i using current policy and exploration
    end if
    (apply actions and record results)
    Execute actions $\boldsymbol{a} = (a_1, \ldots, a_n)$, observe rewards $\mathbf{r}$ and new state $\boldsymbol{x}'$
    replay buffer $\mathcal{D} \leftarrow (\boldsymbol{x}, \boldsymbol{a}, \mathbf{r}, \boldsymbol{x}')$
    $\boldsymbol{x} \leftarrow \boldsymbol{x}'$
    (apply learning step if applicable)
    if $step \% t_{\text{learn}} = 0$ **then**
      for all agents $i \in 1, \ldots, n$ **do**
        sample batch $\{(\boldsymbol{x}^j, \boldsymbol{a}^j, \mathbf{r}^j, \boldsymbol{x}'^j)\}_{j=1}^{|\mathcal{B}|}$ from $\mathcal{D}$
        $y^j \leftarrow r_i^j + \gamma \mathbf{Q}^{\bar{\boldsymbol{\mu}}}(\boldsymbol{x}'^j, a'_1, \ldots, a'_n)$, where $a'_k = \bar{\boldsymbol{\mu}}_k(\boldsymbol{o}'^j_k)$
        Update critic by minimizing the loss $\ell(\boldsymbol{w}_i) = \frac{1}{|\mathcal{B}|} \sum_j \left(y^j - \mathbf{Q}_i^{\boldsymbol{\mu}}(\boldsymbol{x}^j, a_1^j, \ldots, a_n^j)\right)^2$ using B
        Update actor policy using policy gradient formula B
        $\nabla_{\boldsymbol{\theta}_i} J \approx \frac{1}{|\mathcal{B}|} \sum_j \nabla_{\boldsymbol{\theta}_i} \boldsymbol{\mu}_i(\boldsymbol{o}_i^j) \nabla_{a_i} \mathbf{Q}_i^{\boldsymbol{\mu}}(\boldsymbol{x}^j, a_1^j, \ldots, a_i, \ldots, a_n^j)$, where $a_i = \boldsymbol{\mu}_i(\boldsymbol{o}_i^j)$
      end for
      for all agents $i \in [n]$ **do**
        $\bar{\boldsymbol{\theta}}_i \leftarrow \tau \boldsymbol{\theta}_i + (1-\tau)\bar{\boldsymbol{\theta}}_i$ *(update target networks)*
        $\bar{\boldsymbol{w}}_i \leftarrow \tau \boldsymbol{w}_i + (1-\tau)\bar{\boldsymbol{w}}_i$
      end for
    end if
    $step \leftarrow step + 1$
  until environment terminates
  NESTEDLOOKAHEAD$(n, e, \boldsymbol{\varphi}, \mathcal{L})$
end for
Output: $\boldsymbol{\theta}$, $\boldsymbol{w}$

Algorithm 8 Pseudocode for LA–MATD3: MATD3 with (Nested) Lookahead.

Input: Environment $\mathcal{E}$, number of agents n, number of episodes t, action spaces $\{\mathcal{A}_i\}_{i=1}^n$, number of random steps t_{rand} before learning, learning interval t_{learn}, actor networks $\{\boldsymbol{\mu}_i\}_{i=1}^n$, with initial weights $\boldsymbol{\theta} \equiv \{\boldsymbol{\theta}_i\}_{i=1}^n$, both critic networks, $\{\mathbf{Q}_{i,1}, \mathbf{Q}_{i,2}\}_{i=1}^n$ with initial weights $\boldsymbol{w} \equiv \{\boldsymbol{w}_{i,1}, \boldsymbol{w}_{i,2}\}_{i=1}^n$, learning rates $\eta_{\boldsymbol{\theta}}, \eta_{\boldsymbol{w}}$, base optimizer B (e.g., Adam), discount factor γ, lookahead hyperparameters $\mathcal{L} \equiv (l, \{k^{(j)}\}_{j=1}^l, \alpha_{\boldsymbol{\theta}}, \alpha_{\boldsymbol{w}})$, soft update parameter τ, policy update frequency p.
Initialize:
  Replay buffer $\mathcal{D} \leftarrow \varnothing$
  LA parameters: $\boldsymbol{\varphi} \leftarrow \{\boldsymbol{\theta}\}_{\times l}, \{\boldsymbol{w}\}_{\times l}$ *(store snapshots for LA)*
for all episode $e \in 1, \ldots, t$ **do**
  $\boldsymbol{x} \leftarrow$ *Sample*$(\mathcal{E})$ *(sample from environment $\mathcal{E}$)*
  $step \leftarrow 1$
  repeat
    if $e \leq t_{\text{rand}}$ **then**
      for each agent i, $a_i \sim \mathcal{A}_i$ *(sample actions randomly)*
    else
      for each agent i, select action a_i using current policy and exploration
    end if
    (apply actions and record results)
    Execute actions $\boldsymbol{a} = (a_1, \ldots, a_n)$, observe rewards $\mathbf{r}$ and new state $\boldsymbol{x}'$
    replay buffer $\mathcal{D} \leftarrow (\boldsymbol{x}, \boldsymbol{a}, \mathbf{r}, \boldsymbol{x}')$
    $\boldsymbol{x} \leftarrow \boldsymbol{x}'$
    (apply learning step if applicable)
    if $step \% t_{\text{learn}} = 0$ **then**
      for all agent $i \in [n]$ **do**
        sample batch $\{(\boldsymbol{x}^j, \boldsymbol{a}^j, \mathbf{r}^j, \boldsymbol{x}'^j)\}_{j=1}^{|\mathcal{B}|}$ from $\mathcal{D}$
        $y^j \leftarrow r_i^j + \gamma \min_{m=1,2} \mathbf{Q}_{i,l}^{\bar{\boldsymbol{\mu}}}(\boldsymbol{x}'^j, a'_1, \ldots, a'_n)$, where $a'_k = \bar{\boldsymbol{\mu}}_k(\boldsymbol{o}_k'^j) + \epsilon$
        Update both critics, $m = 1, 2$ by minimizing the loss (using optimizer B):
        $$\ell(\boldsymbol{w}_{i,m}) = \frac{1}{|\mathcal{B}|} \sum_j \left(y^j - \mathbf{Q}_{i,m}^{\boldsymbol{\mu}}(\boldsymbol{x}^j, a_1^j, \ldots, a_n^j) \right)^2$$
        if $step \% p = 0$ **then**
          Update actor policy using policy gradient formula and optimizer B
          $\nabla_{\boldsymbol{\theta}_i} J \approx \frac{1}{|\mathcal{B}|} \sum_j \nabla_{\boldsymbol{\theta}_i} \boldsymbol{\mu}_i(\boldsymbol{o}_i^j) \nabla_{a_i} \mathbf{Q}_{i,1}^{\boldsymbol{\mu}}(\boldsymbol{x}^j, a_1^j, \ldots, a_i, \ldots, a_n^j)$, where $a_i = \boldsymbol{\mu}_i(\boldsymbol{o}_i^j)$
          $\bar{\boldsymbol{\theta}}_i \leftarrow \tau \boldsymbol{\theta}_i + (1 - \tau) \bar{\boldsymbol{\theta}}_i$ *(update target networks)*
          $\bar{\boldsymbol{w}}_{i,m} \leftarrow \tau \boldsymbol{w}_{i,m} + (1 - \tau) \bar{\boldsymbol{w}}_{i,m}$
        end if
      end for
    end if
    $step \leftarrow step + 1$
  until environment terminates
  NestedLookAhead$(n, e, \boldsymbol{\varphi}, \mathcal{L})$
end for
Output: $\boldsymbol{\theta}$, $\boldsymbol{w}$

Table 3: Hyperparameters used for LA-MADDPG experiments.

Name	Description
Adam lr	0.01
Adam β_1	0.9
Adam β_2	0.999
Batch-size	1024
Update ratio τ	0.01
Discount factor γ	0.95
Replay Buffer	1.5×10^6
learning step t_{learn}	100
t_{rand}	1024
Policy update ratio (MATD3) p	2
Noise std (MATD3)	0.2
Noise clip (MATD3)	0.5
Lookahead α	0.5

D Details On The Implementation

We used the configurations and hyperparameters from the original MADDPG paper for our implementation. For completeness, these are listed in Table 3. We ran $t = 60000$ training episodes for all environments, with a maximum of 25 environment steps ($step$) per episode.

In all experiments, we used a 2-layer MLP with 64 units per layer. ReLU activation was applied between layers for both the policy and value networks of all agents.

D.1 Hyperparameter Selection for Lookahead

In this section, we discuss and share guidelines for hyperparameter selection based on our experiments.

Summary.

- We observed two- or three-level of Lookahead outperform single-level Lookahead.
- Each level $j \in [l]$ has different k, denoted here with $k^{(j)}$. These should be selected as multiple of the selected k for the level before, that is, $k^{(j)} = c_j \cdot k^{(j-1)}$, where c_j is positive integer.
- We observed that for the innermost lookahead, small values for $k^{(1)}$, such as smaller than 50, perform better than using large values. For the outer $k^{(j)}, j > 1$ large values work well, such as in the range between $5 - 10$ for the c_j,.
- We typically used $\alpha = 0.5$, and we observed lower values, such as $\alpha = 0.3$, give better performances then $\alpha > 0.5$.

Discussion.

- To give an intuition regarding the above-listed conclusions, small values for $k^{(1)}$ help because the MARL setting is very noisy and the vector field is rotational. If large values are used for k_s, then the algorithm will diverge away. It is known that the combination of noise and rotational vector field can cause methods to diverge away [Chavdarova et al., 2019].
- Relative to the analogous conclusions for GANs [Chavdarova et al., 2021], the differences is that:
  - The better-performing values for $k^{(1)}$ are of a similar range as for Lookahead with GD for GANs; however they are smaller than those used for Lookahead with EG for GANs.

D.2 Compute resources

We ran experiments on Google Colab enterprise using an e2-standard-8 type machine with 100 GB Standard disk (pd-standard).

E Additional Empirical Results

E.1 Rock–paper–scissors: Buffer Structure

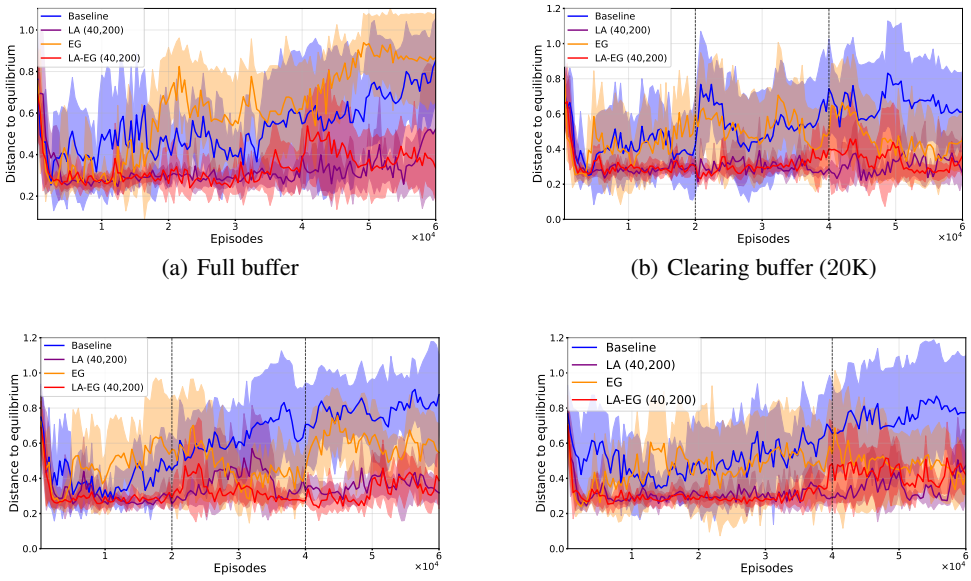
For the Rock–paper–scissors (RPS) game, using a buffer size of 1M wasn't sufficient to store all experiences from the 60K training episodes. We observed a change in algorithm behavior around 40K episodes. To explore the impact of buffer configurations, we experimented with different sizes and structures, as experience storage plays a critical role in multi-agent reinforcement learning.

Full buffer. The buffer is configured to store all experiences from the beginning to the end of training without any loss.

Buffer clearing. In this setup, a smaller buffer is used, and once full, the buffer is cleared completely, and new experiences are stored from the start.

Buffer shifting. Similar to the small buffer setup, but once full, old experiences are replaced by new ones in a first-in-first-out (FIFO) manner.

Results. Figure 4 depicts the results when using different buffer options for the RPS game.



(a) Full buffer (b) Clearing buffer (20K)

(c) Shifting buffer (20K) (d) Shifting buffer (40K)

Figure 4: **Comparison of different buffer configurations (see Appendix E.1) and methods on Rock–paper–scissors game.** x-axis: training episodes. y-axis: 5-seed average norm between the two players' policies and equilibrium policy $(\frac{1}{3}, \frac{1}{3}, \frac{1}{3})^2$. The dotted line indicates the point at which the buffer begins to change, either through shifting or clearing.

E.2 Rock–paper–scissors: Scheduled learning rate

We experimented with gradually decreasing the learning rate (LR) during training to see if it would aid convergence to the optimal policy in RPS. While this approach reduced noise in the results, it also led to increased variance across all methods except for LA-MADDPG.

Figure 5 depicts the average distance to the equilibrium policy over 5 different seeds for each methods, using periodically decreased step sizes.

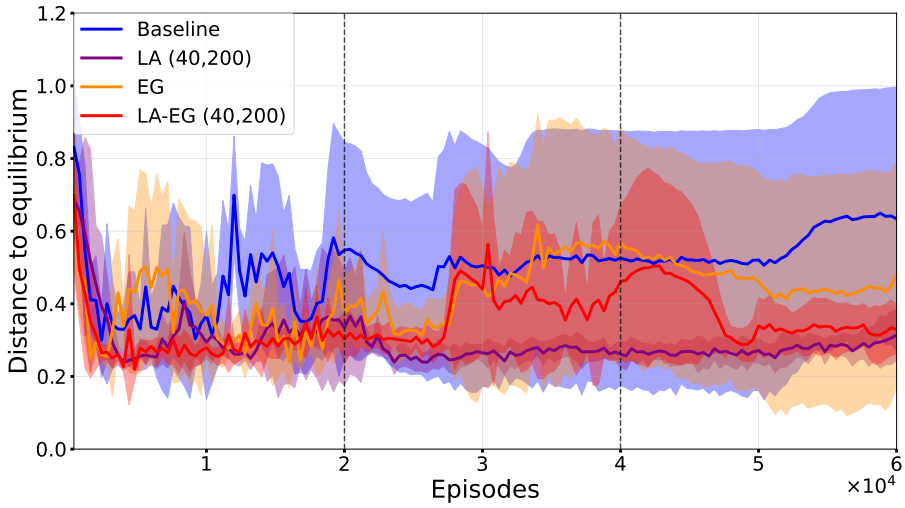


Figure 5: **Compares MADDPG with different *LA-MADDPG* configurations to the baseline MADDPG with (*Adam*) in Rock–paper–scissors with a scheduled learning rate**. x-axis: training episodes. y-axis: 5-seed average norm between the two players' policies and equilibrium policy $(\frac{1}{3}, \frac{1}{3}, \frac{1}{3})^2$. The dotted lines depict the times when the learning rate was decreased by a factor of 10.

E.3 MPE: Predator-prey Full results

We also evaluated the trained models of all methods on an instance of the environment that runs for 50 steps to compare learned policies. We present snapshots from it in Figure 7. Here, you can clearly anticipate the difference between the policies from baseline and our optimization methods. As in the baseline, only one agent will chase at the beginning of episode. Moreover, for the baseline (topmost row), the agents move further away from the landmarks and the good agent, which is suboptimal. This can be noticed from the decreasing agents' size in the figures. While in ours, both adversary agents engage in chasing the good agent until the end.

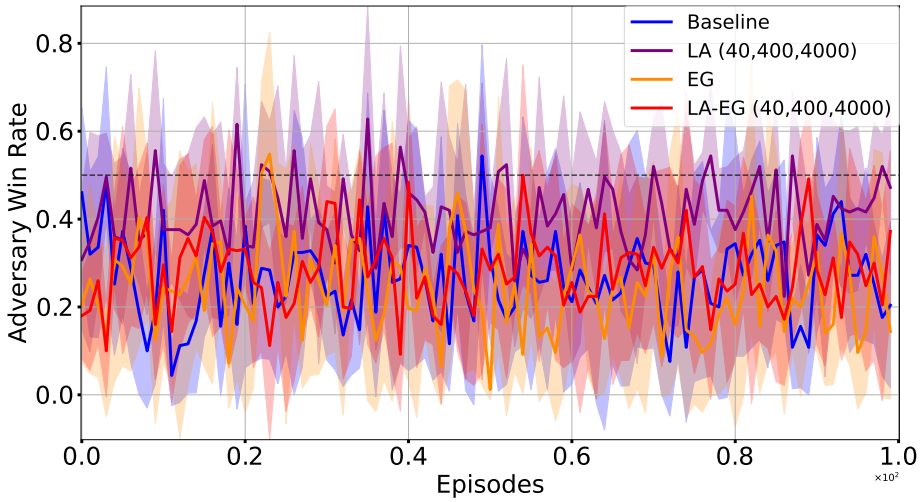


Figure 6: **Comparison on the MPE–Predator-prey game between the *GD-MADDPG*, *LA-MADDPG*, *EG-MADDPG* and *LA-EG-MADDPG* optimization methods, denoted as *Baseline*, *LA*, *EG*, *LA-EG*, resp.** x-axis: evaluation episodes. y-axis: mean adversaries win rate, averaged over 5 runs with different seeds.

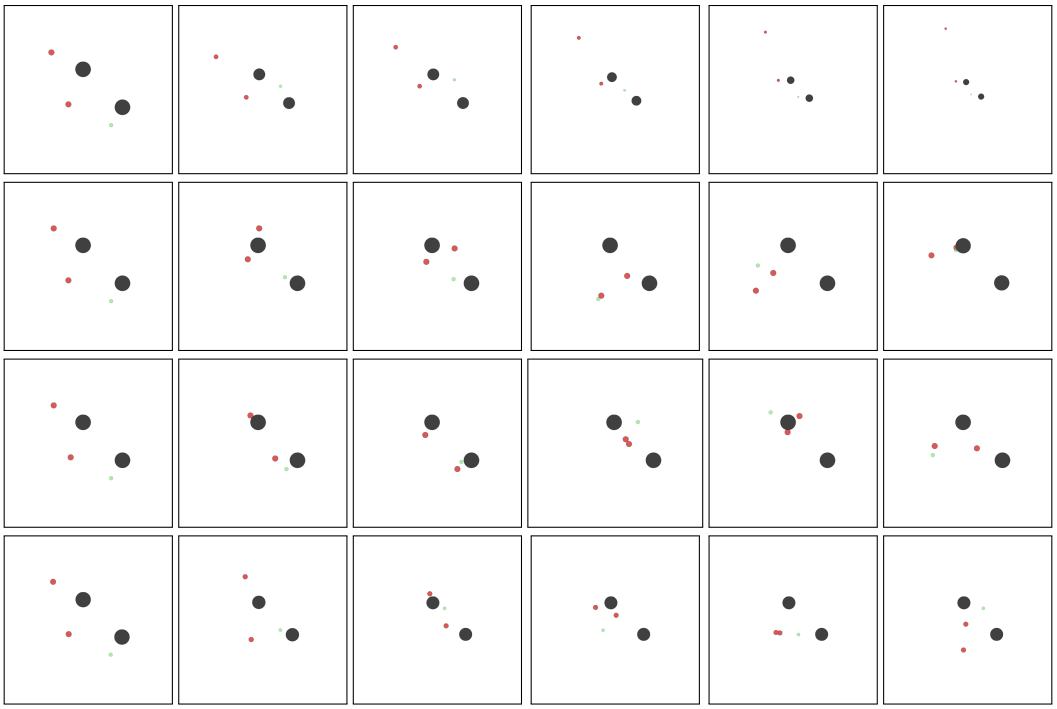


Figure 7: **Agents' trajectories of fully trained models with all considered optimization methods on the same environment seed of MPE: Predator-prey.** Snapshots show the progress of agents as time progresses in a 50 steps long environment. Each row contains snapshots of one method, from top to bottom: *GD-MADDPG*, *LA-MADDPG*, *EG-MADDPG* and *LA-EG-MADDPG*. Big dark circles represent landmarks, small red circles are adversary agents and green one is the good agent.

.

E.4 MPE: Predator-Prey and Physical deception training figures

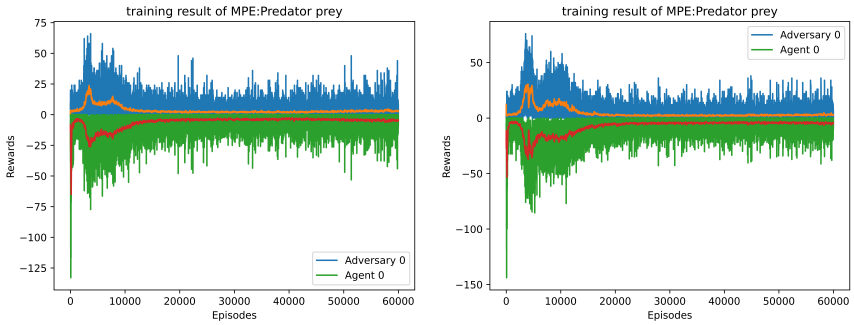
In figures 8(a) and 8(b) we include the rewards achieved during the training of GD-MADDPG and LA-MADDPG resp. for MPE: Predator-prey. The figures show individual rewards for the agent (prey) and one adversary (predator). Blue and green show the individual rewards received at each episode while the orange and red lines are the respective running averages with window size of 100 of those rewards.

Figures 9(a) and 9(b) demonstrate same results but for MPE: Physical deception. In this game, We have two good agents, 'Agent 0 and 1' but since they are both receive same rewards, we only show agent 0.

E.5 On the Rewards as Convergence Metric

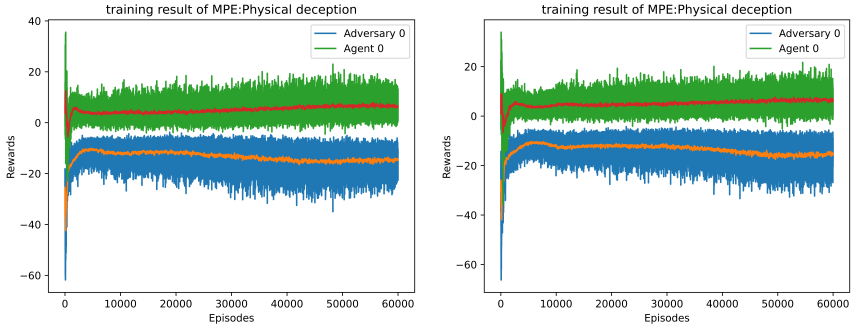
Based on our experiments and findings from the multi-agent literature [Bowling, 2004], we observe that average rewards offer a weaker measure of convergence compared to policy convergence in multi-agent games. This implies that rewards can reach a target value even when the underlying policy is suboptimal. For example, in the Rock–paper–scissors game, the Nash equilibrium policy leads to nearly equal wins for both players, resulting in a total reward of zero. However, this same reward can also be achieved if one player always wins while the other consistently loses, or if both players repeatedly select the same action, leading to a tie. As such, relying solely on rewards during training can be misleading.

Figure 10 (top row) depicts a case with the baseline where, despite rewards converging during training, the agents ultimately learned to play the same action repeatedly, resulting in ties. Although this matched the expected reward, it falls far short of equilibrium and leaves the agents vulnerable to exploitation by more skilled opponents. In contrast, the same figure shows results from LA-MADDPG



(a) GD-MADDPG (b) LA-MADDPG

Figure 8: **The figure shows the learning curves during training of GD-MADDPG and LA-MADDPG for MPE: Predator-Prey.** x-axis: training episodes. y-axis: agents rewards and their moving average with a window size of 100, calculated over 5-seeds over 5 seeds.



(a) GD-MADDPG (b) LA-MADDPG

Figure 9: **The figure shows the learning curves during training of GD-MADDPG and LA-MADDPG for MPE: Physical deception.** x-axis: training episodes. y-axis: agents rewards and their moving average with a window size of 100, calculated over 5-seeds over 5 seeds.

under the same experimental conditions. Notably, while the rewards did not fully converge, the agents learned a near-optimal policy during evaluation, alternating between all three actions as expected. These results also align with the findings shown in Figure 1(a).

We explored the use of gradient norms as a potential metric in these scenarios but found them to be of limited utility, as they provided no clear indication of convergence for either method. We include those results in Figure 11, where we compare the gradient norms of Adam and LA across the networks of different players.

This work highlights the need for more robust evaluation metrics in multi-agent reinforcement learning, a point also emphasized in [Lanctot et al., 2023], as reward-based metrics alone may be inadequate, particularly in situations where the true equilibrium is unknown.

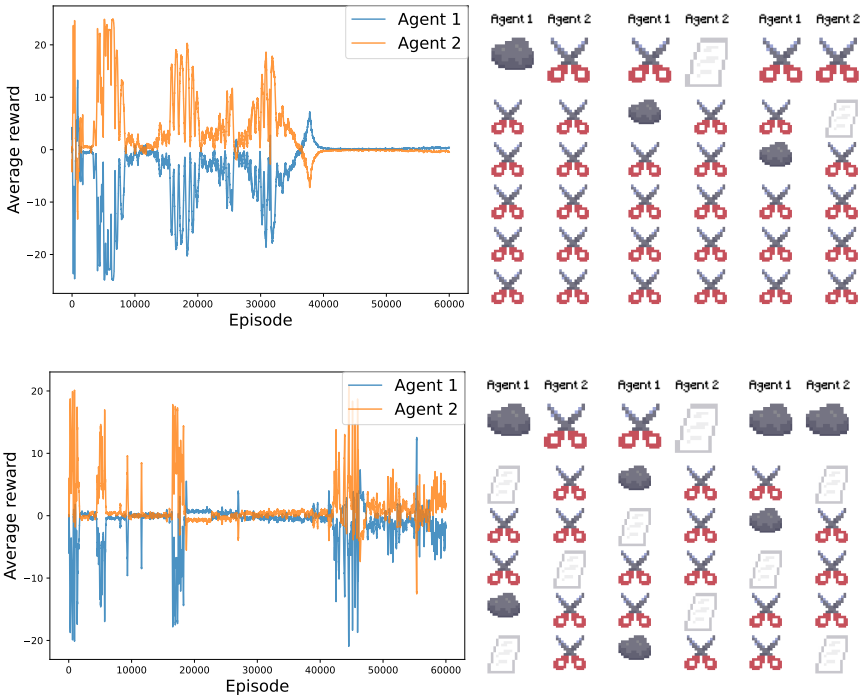


Figure 10: **A more detailed version of Figure 3.** Saturating rewards (left) versus actions of the learned policies at the end (right) in the Rock–paper–scissors game. **Top row:** *GD-MADDPG*; **bottom row:** *LA-MADDPG*. In the left column, blue and orange show the running average of rewards through a window of 100 episodes. In the right column, we depict actions from the respective learned policies evaluation after training is completed, where each row represents what actions players have chosen in one step of the episode. Saturating rewards do not imply good performance, as evidenced by the top row; refer to Section 5.2 for discussion.

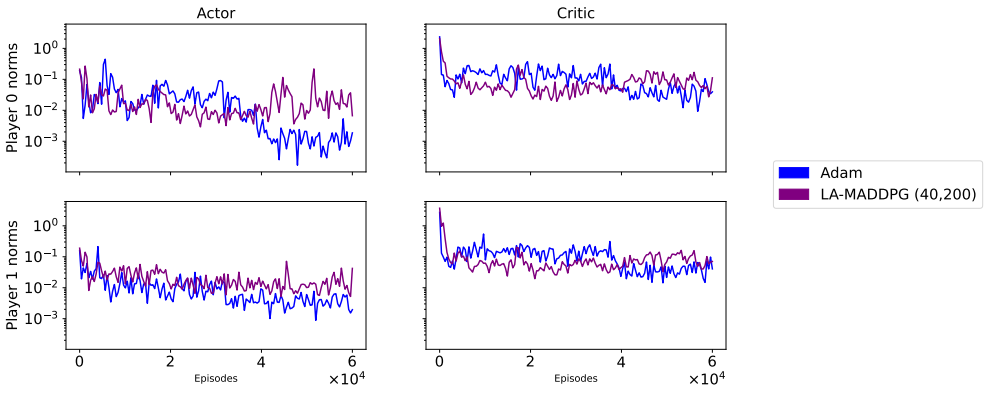


Figure 11: **Gradient norms across training in the *Rock–paper–scissors* game.**