# Virtual Double Oracle: Principled Reinitialisation for Incremental Nash Equilibrium Computation

Ruilan Wang
London School of Economics
London, United Kingdom
r.wang27@lse.ac.uk

Francisco Aristi Reina
London School of Economics
London, United Kingdom
f.aristi-reina@lse.ac.uk

Katerina Papadaki
London School of Economics
London, United Kingdom
K.P.Papadaki@lse.ac.uk

## ABSTRACT

Online Double Oracle (ODO) computes Nash equilibria in two-player zero-sum games by maintaining a small set of strategies, updating their weights via no-regret online learning, and periodically expanding the set by discovering new best responses. At each expansion step, ODO resets all accumulated information to a uniform distribution—discarding history without theoretical justification.

We propose **Virtual Double Oracle (VDO)** for two-player zero-sum games, which replaces this reset with a principled closed-form initialisation. The key insight is that in matrix games, every strategy's entire loss history compresses into a single observable quantity—its utility against the running average opponent policy. This means we can compute exactly what weight any strategy would have received had it been present from the beginning, even before it was discovered.

VDO provably preserves no-regret guarantees for any exponential-weights inner-loop solver, gives newly discovered best responses a head start proportional to their utility advantage, and achieves a factor $\sqrt{k}$ improvement in exploitability over ODO's $O(\sqrt{k \ln k/T})$, giving $O(\sqrt{\ln k/T})$, where $k$ is the number of strategies in the final population at termination. We further show VDO is one instance of a broader **Flexible Double Oracle (FDO)** framework that unifies the entire Double Oracle family. Experiments on Kuhn poker, Leduc poker, and random matrix games confirm faster convergence; the theoretically derived temperature parameter gives smooth, consistent improvement while empirical tuning yields further gains.

## CCS CONCEPTS

• **Computing methodologies** → **Multi-agent systems**; **Game theory**; *Online learning algorithms.*

## KEYWORDS

Nash equilibrium, double oracle, online learning, multiplicative weights, regret minimisation, game-solving

**ACM Reference Format:**
Ruilan Wang, Francisco Aristi Reina, and Katerina Papadaki. 2026. Virtual Double Oracle: Principled Reinitialisation for Incremental Nash Equilibrium Computation. In *Proc. of the Adaptive and Learning Agents Workshop (ALA 2026), Paphos, Cyprus,*
*https://alaworkshop2026.github.io/, May 25 – 26, 2026*, IFAAMAS, 10 pages.

*Proc. of the Adaptive and Learning Agents Workshop (ALA 2026), Aydeniz, Delgrange, Mohammedalamen, Yang (eds.), May 25 – 26, 2026, Paphos, Cyprus,*
*https://alaworkshop2026.github.io/.* 2026.

## 1 INTRODUCTION

Computing Nash equilibria (NE) is a fundamental challenge in multi-agent systems, with applications spanning security games [12], competitive game playing [9, 14], and multi-agent reinforcement learning [8]. In two-player zero-sum games, while the minimax theorem guarantees existence, computing NE in games with large strategy spaces remains computationally demanding. A key insight shared by the DO family and PSRO is that NE typically have small support [3, 4, 11]: only a handful of pure strategies carry positive probability. This motivates an iterative approach—maintain a small restricted population and expand it with best responses until the population contains the full NE support.

The **Double Oracle (DO)** algorithm [11] formalises this: solve the restricted game to NE, compute each player's best response, expand the population, and repeat. DO converges to the full-game NE without enumerating the entire strategy space [11]. **PSRO** [8] generalises DO to large games via RL best responses and flexible meta-strategy solvers. **Online Double Oracle (ODO)** [4] takes a different route: it runs **Multiplicative Weights Update (MWU)** continuously over the restricted population, expanding it whenever a novel best response to the running time-average opponent strategy is found — without ever solving a subgame to equilibrium. This yields the first convergence *rate* guarantee for the DO family: $O(\sqrt{k \ln k/T})$ exploitability, where $k$ is the size of the final strategy population at termination.

Despite this progress, a design decision in ODO has gone unexamined: whenever a new best response is discovered and added to the population, ODO resets MWU weights to the *uniform* distribution over the enlarged set. This discards all accumulated information and treats the newly added strategy as if it has no history. Uniform re initialization is arbitrary—it implicitly assumes all strategies are equally good, which is false since the new strategy was selected *precisely because* it is the best response to the opponent's current play. Formalizing the initialization dynamics at each restricted game is consequential to further optimize the convergence to achieve that Nash Equilibrium and improve the computational time. We resolve this by asking a simple question: even before a strategy is added to the population, can we estimate how useful it would have been? The answer is yes. Because payoffs in matrix games are bilinear, any strategy's entire history of hypothetical losses compresses into a single quantity — how well it would have done against the average opponent policy observed so far. This means we can compute, at any point, exactly what weight MWU would have assigned to any strategy had it been present from the very beginning.

This gives rise to the *hidden-strategy* view: imagine all strategies that will ever be discovered are present from $t = 1$, but kept hidden from the opponent until revealed as best responses. Hidden

strategies accumulate weights silently by watching opponent play. When a strategy is finally revealed, it enters the population not as a blank slate but with a principled initial weight reflecting everything observed so far. ODO's uniform reset is the special case where this history is ignored; VDO uses it.

We make the following contributions:

- *Virtual Double Oracle (VDO) algorithm.* We propose a principled closed-form reinitialisation for ODO in two-player zero-sum games. The key insight is that bilinearity of payoffs allows every strategy's entire hypothetical loss history to be compressed into a single quantity—its utility against the time-average opponent policy—enabling exact computation of the weight any strategy would have received had it been present from the beginning (Section 4).
- *Theoretical guarantees.* We establish three results (Section 5): (i) no-regret preservation for any multiplicative weights inner-loop solver and any $\lambda \geq 0$ (Theorem 14); (ii) $O(\sqrt{\ln k/T})$ exploitability, a factor $\sqrt{k}$ improvement over ODO's $O(\sqrt{k \ln k/T})$ (Theorem 17); and (iii) virtual weight consistency with a single MWU run over all strategies from $t = 1$ (Lemma 13). The improvement is structural: by treating all strategies as virtually present from the start, VDO reduces to a single MWU instance over $k$ strategies, avoiding the $\sqrt{k}$ penalty ODO incurs from treating each population window independently.
- *Flexible Double Oracle (FDO) framework.* We show that VDO is one instance of a broader two-parameter family, parameterised by the best-response frequency $m$ and the reinitialisation distribution $\pi^{\text{init}}$, that subsumes DO, ODO, Anytime DO (ADO, [10]), Periodic Double Oracle (PDO, [13]), and fictitious play as special cases. Any strictly positive target distribution can be exactly recovered via softmax (Proposition 19), accommodating any oracle-specified initialisation (Section 6).
- *Empirical validation.* Experiments on Kuhn poker, Leduc poker, and random matrix games confirm faster convergence; the theoretically derived temperature parameter $\lambda = \eta\bar{T}$ gives smooth, consistent improvement while empirical tuning yields further gains in games with heterogeneous utility landscapes (Section 7).

Section 2 reviews background; Section 3 discusses related work; Sections 4–5 present VDO and proofs; Section 6 develops FDO; Section 7 reports experiments; Section 8 concludes.

## 2 BACKGROUND

We review the game-theoretic setting, the online learning framework, and the two algorithms that VDO builds upon.

### 2.1 Two-Player Zero-Sum Games

A two-player zero-sum normal-form game is a triple $(S_1, S_2, A)$ where $S_1$, $S_2$ are finite pure strategy sets with $|S_1| = n$, $|S_2| = m$, and $A \in [-1, 1]^{n\times m}$ is player 1's payoff matrix; player 2's payoff is $-A$. A *mixed strategy* is a probability distribution $\pi_i \in \Delta(S_i)$, where $\Delta(S)$ denotes the probability simplex over set $S$. The *utility* of player $i \in \{1, 2\}$ is $u_i : \Delta(S_i) \times \Delta(S_{-i}) \to \mathbb{R}$, where player $i$'s own strategy is always the first argument:

$$u_1(\pi_1, \pi_2) = \pi_1^\top A\pi_2, \qquad u_2(\pi_2, \pi_1) = -\pi_1^\top A\pi_2.$$

For a pure strategy $s \in S_i$ we write $u_i(s, \pi_{-i}) := u_i(e_s, \pi_{-i})$, where $e_s \in \{0, 1\}^{|S_i|}$ is the standard basis vector for $s \in S_i$. Utility is *bilinear* in both arguments.

*Loss convention.* We work throughout in losses rather than utilities. For player $i \in \{1, 2\}$ at round $t$, the *loss vector* $\ell_i^t \in [-1, 1]^{|S_i|}$ assigns to each pure strategy $s \in S_i$ the loss

$$\ell_1^t(s) = -e_s^\top A\pi_2^t \quad (s \in S_1), \qquad \ell_2^t(s) = \pi_1^{t\top} Ae_s \quad (s \in S_2),$$

so that minimising loss equals maximising utility for both players: $\ell_i^t(s) = -u_i(s, \pi_{-i}^t)$. All subsequent algorithmic definitions are stated in terms of $\ell_i^t$ and apply symmetrically to both players.

Definition 1 (Nash Equilibrium). *A strategy profile $(\pi_1^*, \pi_2^*)$ is a* Nash equilibrium *(*NE*) if neither player can benefit from a unilateral deviation:*

$$\pi_1^{*\top} A\pi_2^* \geq \pi_1^\top A\pi_2^* \quad \forall \pi_1 \in \Delta(S_1),$$
$$\pi_1^{*\top} A\pi_2^* \leq \pi_1^{*\top} A\pi_2 \quad \forall \pi_2 \in \Delta(S_2).$$

*By the minimax theorem, every finite zero-sum game has a* NE *with value $v^* = \max_{\pi_1} \min_{\pi_2} \pi_1^\top A\pi_2$.*

Definition 2 (Exploitability). *The* exploitability *of a strategy profile $(\pi_1, \pi_2)$ measures its distance from* NE*:*

$$\text{Exploit}(\pi_1, \pi_2) = \max_{s\in S_1} u_1(s, \pi_2) + \max_{s\in S_2} u_2(s, \pi_1) \;\geq\; 0,$$

*with equality if and only if $(\pi_1, \pi_2)$ is a* NE.

Definition 3 ($\varepsilon$-Nash Equilibrium). *A profile $(\pi_1, \pi_2)$ is an* $\varepsilon$-Nash equilibrium *($\varepsilon$-NE) if* Exploit$(\pi_1, \pi_2) \leq \varepsilon$.

Definition 4 (Best Response). *A* best response *for player $i$ to opponent strategy $\pi_{-i} \in \Delta(S_{-i})$ is any pure strategy $\beta_i \in S_i$ satisfying:*

$$\text{BR}_i(\pi_{-i}) := \arg\max_{s\in S_i} u_i(s, \pi_{-i}).$$

Remark 5. *By the loss convention, $u_i(s, \pi_{-i}) = -\ell_i^t(s)$ at any round $t$ with $\pi_{-i}^t = \pi_{-i}$, so equivalently:*

$$\text{BR}_i(\pi_{-i}) = \arg\min_{s\in S_i} \ell_i^t(s).$$

*This is the form used in Algorithm 2 (OSO) and Algorithm 4 (VDO, Section 4).*

### 2.2 Online Learning and Regret

In online learning, player $i$ selects strategies $\pi_i^1, \pi_i^2, \ldots, \pi_i^T$ over $T$ rounds and observes loss vectors $\ell_i^t \in [-1, 1]^{|S_i|}$ after each round (defined via the loss convention above). The *regret* of player $i$ measures cumulative loss relative to the best fixed strategy in hindsight:

$$\text{Regret}_i(T) = \sum_{t=1}^{T} \langle \pi_i^t, \ell_i^t \rangle - \min_{s\in S_i} \sum_{t=1}^{T} \ell_i^t(s).$$

An algorithm is *no-regret* if $\text{Regret}_i(T)/T \to 0$ as $T \to \infty$. When both players run no-regret algorithms, vanishing regret implies convergence to NE:

Proposition 6 (Regret-Exploitability [6]).

$$\text{Exploit}(\bar{\pi}_1^T, \bar{\pi}_2^T) \leq \frac{\text{Regret}_1(T) + \text{Regret}_2(T)}{T}.$$

**Algorithm 1** Double Oracle (DO) [11]

**Require:** Full pure strategy sets $S_1, S_2$.
1: Initialise $\Pi_1 \subseteq S_1$, $\Pi_2 \subseteq S_2$ (e.g., one strategy each).
2: **repeat**
3: Compute NE $(\pi_1^r, \pi_2^r)$ of the restricted game induced by $(\Pi_1, \Pi_2)$.
4: **for** $i \in \{1, 2\}$ **do**
5: $\beta_i \leftarrow \arg\max_{s \in S_i} u_i(s, \pi^r_{-i})$.
6: $\Pi_i \leftarrow \Pi_i \cup \{\beta_i\}$.
7: **end for**
8: **until** $\beta_i \in \Pi_i$ for both $i$.
9: **return** $(\pi_1^r, \pi_2^r)$.

## 2.3 Multiplicative Weights Update

MWU [1, 6] is the canonical no-regret algorithm for finite action sets. Given a set $\Pi$ of $|\Pi|$ strategies, it maintains weights $w^t \in \mathbb{R}^{|\Pi|}_{>0}$, plays $\pi^t = w^t/\|w^t\|_1$, and upon observing loss $\ell^t \in [-1, 1]^{|\Pi|}$ performs the multiplicative update

$$w^{t+1}(s) = w^t(s) \exp(-\eta\, \ell^t(s)) \quad \forall s \in \Pi.$$

In the two-player setting, each player $i \in \{1, 2\}$ runs MWU independently with $\Pi = \Pi_i$, $\pi^t = \pi_i^t$, and $\ell^t = \ell_i^t$.

Theorem 7 (MWU regret [1]). *Starting from uniform weights $w^1(s) = 1/|\Pi|$ and learning rate $\eta = \sqrt{2 \ln |\Pi|/T}$, MWU achieves:*

$$\frac{1}{T}\left(\sum_{t=1}^{T} \langle \pi^t, \ell^t \rangle - \min_{s \in \Pi} \sum_{t=1}^{T} \ell^t(s)\right) \le \sqrt{\frac{2 \ln |\Pi|}{T}}.$$

The following generalisation to non-uniform initialisation is central to our analysis. It says that if a comparator $s^*$ starts with *larger* initial weight than uniform, the regret against $s^*$ is strictly reduced — precisely the theoretical handle VDO exploits.

Lemma 8 (Non-uniform MWU [2, Lemma 5.1]). *If MWU is run with initial weights $w^1(s) \ge 0$ satisfying $W^1 := \sum_{s \in \Pi} w^1(s) \le 1$, and losses $\ell^t \in [-1, 1]^{|\Pi|}$, then for any comparator $s^* \in \Pi$ with $w^1(s^*) > 0$:*

$$\sum_{t=1}^{T} \langle \pi^t, \ell^t \rangle - \sum_{t=1}^{T} \ell^t(s^*) \;\le\; \frac{1}{\eta} \ln \frac{1}{w^1(s^*)} + \frac{\eta T}{2}.$$

Setting $w^1(s^*) = 1/|\Pi|$ recovers Theorem 7 since $\ln(1/w^1(s^*)) = \ln |\Pi|$. If instead $w^1(s^*) > 1/|\Pi|$, the logarithmic term is strictly smaller.

## 2.4 Double Oracle (DO)

DO (Algorithm 1) maintains restricted sets $\Pi_i \subseteq S_i$, solves the restricted game to NE, then adds each player's best response to the opponent's NE strategy. When no novel best response is found, the restricted-game NE is also a NE of the full game [11], guaranteeing convergence. However, DO provides no convergence rate bounds, and in the worst case must enumerate all pure strategies.

## 2.5 Online Double Oracle (ODO)

ODO runs the above (Algorithm 2) for both players simultaneously, each with their own strategy set $S_i$, population $\Pi_i$, and

**Algorithm 2** Online Single Oracle (OSO) [4]

**Require:** Full strategy set $S$; learning rate $\eta$.
1: Init $\Pi \leftarrow \{s^0\}$ for arbitrary $s^0 \in S$; $w^1(s) \leftarrow 1/|\Pi|$, $\forall s \in \Pi$; $\bar{\ell} \leftarrow \mathbf{0}$; $n_w \leftarrow 0$.
2: **for** $t = 1, 2, \ldots$ **do**
3: **Play:** $\pi^t \leftarrow w^t/\|w^t\|_1$.
4: **Observe:** loss $\ell^t \in [-1, 1]^{|\Pi|}$ from opponent's play.
5: **MWU update:** $w^{t+1}(s) \leftarrow w^t(s) \exp(-\eta\, \ell^t(s))$, $\forall s \in \Pi$.
6: **Window avg:** $n_w \leftarrow n_w + 1$; $\bar{\ell} \leftarrow \bar{\ell} + (\ell^t - \bar{\ell})/n_w$.
7: **BR:** $\beta^* \leftarrow \mathrm{BR}(\bar{\ell}) := \arg\min_{s \in S} \langle e_s, \bar{\ell} \rangle$.
8: **if** $\beta^* \notin \Pi$ **then**
9: $\Pi \leftarrow \Pi \cup \{\beta^*\}$.
10: **Reset:** $w^{t+1}(s) \leftarrow 1/|\Pi|$, $\forall s \in \Pi$; $\bar{\ell} \leftarrow \mathbf{0}$; $n_w \leftarrow 0$. {new window starts}
11: **end if**
12: **end for**
13: **return** $\bar{\pi}^T = (1/T) \sum_{t=1}^{T} \pi^t$.

**Algorithm 3** Online Double Oracle (ODO) [4]

**Require:** Full strategy sets $S_1, S_2$; learning rate $\eta$.
1: Initialise $\Pi_i \leftarrow \{s_i^0\}$ for $i \in \{1, 2\}$.
2: **for** $t = 1, 2, \ldots$ **do**
3: Each player $i$ plays $\pi_i^t$, observes $\ell_i^t$ induced by opponent's play $\pi^t_{-i}$, and updates via OSO (Algorithm 2) with $S = S_i$, $\Pi = \Pi_i$.
4: **end for**
5: **return** $\bar{\pi}_i^T = (1/T) \sum_{t=1}^{T} \pi_i^t$ for $i \in \{1, 2\}$.

weights, against the opponent's played strategies as losses. ODO (Algorithm 3) replaces exact restricted NE solving with MWU-based online learning. At each step, a best response to the window-local average loss is computed and added to $\Pi$ if novel. The iterations are partitioned into *time windows*: window $T_j$ is the maximal interval of steps during which $|\Pi^{(t)}| = j$. We denote the length of window $j$ by $|T_j|$, the cumulative iterations at the start of window $j$ by $\bar{T}_j = \sum_{r=1}^{j-1} |T_r|$, total iterations $T = \sum_{j=1}^{k} |T_j|$, and final population size $k = |\Pi^{(T)}|$.

Theorem 9 (OSO regret [4], Theorem 3). *ODO achieves:*

$$\frac{1}{T}\left(\sum_{t=1}^{T} \langle \pi^t, \ell^t \rangle - \min_{s \in \Pi^{(k)}} \sum_{t=1}^{T} \ell^t(s)\right) \;\le\; \sqrt{\frac{k \ln k}{2T}}.$$

The proof applies MWU's per-window regret bound and sums across $k$ windows via Cauchy–Schwarz. In window $j$, the population has $|\Pi^{(j)}| = j$ strategies, so MWU's bound uses $\ln j$:

$$\sum_{j=1}^{k} \sqrt{\tfrac{|T_j|}{2} \ln j} \overset{\text{C-S}}{\le} \sqrt{\tfrac{T}{2} \cdot \sum_{j=1}^{k} \ln j} \;\le\; \sqrt{\tfrac{Tk \ln k}{2}},$$

where the last step uses $\sum_{j=1}^{k} \ln j \le k \ln k$.

This $\sqrt{k}$ factor — the Cauchy–Schwarz penalty — is precisely what VDO eliminates by treating the full run as a single MWU instance. The ODO paper itself acknowledges this: in a remark on Algorithm 1, the authors note that when prior knowledge is available through historical data, OSO 'can exploit this knowledge

by updating the starting strategy in each time window,' and leave this extension to future work [4]. VDO answers this directly. When both players run OSO, the exploitability bound follows from Proposition 6.

## 3 RELATED WORK

*DO family.* DO [11] and Anytime DO (ADO) [10] solve restricted games exactly but lack rate bounds; ADO guarantees non-increasing exploitability via one-sided restriction. ODO [4] introduced MWU-based solving with $O(\sqrt{k \ln k/T})$. PDO [13] analyses BR computation frequency as an independent design axis, generalising ODO to extensive-form games via CFR. XDO [9] extends DO to extensive-form games by expanding the restricted game at the action level. VDO addresses the previously unanalysed reinitialisation step, orthogonal to all of the above.

*PSRO family.* PSRO [8] uses RL-based best responses with various meta-strategy solvers. Projected Replicator Dynamics, QRE-MSS, and RRD [15] apply softmax-type regularisation. VDO's $\lambda$ plays a mathematically similar role to QRE's rationality parameter, but is applied at *reinitialisation* with a theoretically derived value, not at each step.

*Non-uniform initialisation.* EPSRO [16] carries forward weights via a utility-matching substitute strategy. VDO provides a stronger guarantee: exact virtual weights, not an approximation. Non-uniform MWU initialisation is studied in [2, Lemma 5.1], which our proof invokes directly.

*Regret minimisation theory.* The MWU regret framework originates with [6] and is surveyed by [1]. The present paper studies principled reinitialisation, applicable regardless of which update rule the inner loop uses.

## 4 VIRTUAL DOUBLE ORACLE

This section develops the key insight behind VDO, derives the reinitialisation formula, and presents the algorithm.

### 4.1 The Hidden-Strategy Principle

**Traditional ODO view**: strategies are discovered incrementally; each reinitialisation discards the past.

**Hidden-strategy view**: imagine all $k$ final strategies exist from $t = 1$, but are *hidden* until revealed as best responses. A hidden strategy does not affect what is played (the opponent never sees it), but accumulates virtual losses by observing opponent play.

Definition 10 (Virtual cumulative loss). *For player $i \in \{1, 2\}$, any strategy $s \in S_i$, and any $\tau \geq 1$, the* virtual cumulative loss *is:*

$$L_i^\tau(s) := \sum_{t=1}^{\tau} \ell_i^t(s) = -\sum_{t=1}^{\tau} u_i(s, \pi_{-i}^t).$$

*This quantity is observable from the opponent's play alone; no play of $s$ is required.*

Since $\ell_i^t(s) = -u_i(s, \pi_{-i}^t)$ depends only on the opponent's play at step $t$, the virtual cumulative loss $L_i^\tau(s)$ can be computed for any strategy $s$—even one not currently in the population—from the observed opponent sequence alone.

Lemma 11 (Bilinear collapse). *For player $i \in \{1, 2\}$, any $s \in S_i$, any opponent sequence $\pi_{-i}^1, \ldots, \pi_{-i}^T$, and any $\tau \geq 1$:*

$$L_i^\tau(s) = -\tau \cdot u_i\left(s,\ \bar{\pi}_{-i}^{[1:\tau]}\right), \qquad \bar{\pi}_{-i}^{[1:\tau]} := \frac{1}{\tau}\sum_{t=1}^{\tau} \pi_{-i}^t.$$

Proof. For player 1:

$$L_1^\tau(s) = -\sum_{t=1}^{\tau} e_s^\top A \pi_2^t = -e_s^\top A \sum_{t=1}^{\tau} \pi_2^t = -\tau \cdot u_1\left(s, \bar{\pi}_2^{[1:\tau]}\right).$$

The case $i = 2$ follows by symmetry using $\ell_2^t(s) = \pi_1^{t\top} A e_s$:

$$L_2^\tau(s) = \sum_{t=1}^{\tau} \pi_1^{t\top} A e_s = \left(\sum_{t=1}^{\tau} \pi_1^t\right)^\top A e_s = -\tau \cdot u_2\left(s, \bar{\pi}_1^{[1:\tau]}\right).$$

□

Lemma 11 is the cornerstone of VDO: the full opponent history of $\tau$ policies compresses *exactly* into the utility against one time-average $\bar{\pi}_{-i}^{[1:\tau]}$. This is an algebraic identity holding for any sequence, including adaptive adversaries.

*Why opponent adaptation does not matter.* No-regret guarantees compare against any fixed comparator strategy evaluated on the loss sequence that actually occurred. The virtual losses $L_i^\tau(s)$ are computed against this realised sequence, not against a hypothetical sequence that would have arisen had $s$ actually been played. This is precisely the sense in which regret is a hindsight measure, and it is why the virtual loss computation remains valid even when the opponent adapts.

### 4.2 Virtual MWU Weights and VDO Initialisation

Suppose strategy $s$ joins the population at the start of window $j + 1$ (global step $\bar{T}_{j+1} = \sum_{r=1}^{j} |T_r|$).

Definition 12 (Virtual weight). *For any $\tau \geq 1$ and any $s \in \Pi^{(k)}$, the* virtual weight $w_{\mathcal{V}}^\tau(s)$ *is the weight that MWU with uniform initialisation $w_{\mathcal{V}}^1(s) = 1/k$ over the full final population $\Pi^{(k)}$ would assign to $s$ after $\tau$ steps against the actual opponent sequence.*

Strategy $s$ was just added to the strategy set and its virtual weight is revealed at time $\bar{T}_{j+1}$. By unrolling the MWU update and applying Lemma 11:

$$\begin{aligned} w_{\mathcal{V}}^{\bar{T}_{j+1}}(s) &= \frac{1}{k}\exp\left(-\eta\, L_i^{\bar{T}_{j+1}}(s)\right) \\ &= \frac{1}{k}\exp\left(\eta \bar{T}_{j+1} \cdot u_i\left(s, \bar{\pi}_{-i}^{[1:\bar{T}_{j+1}]}\right)\right), \end{aligned} \tag{1}$$

where the second line applies Lemma 11, using $L_i^{\bar{T}_{j+1}}(s) = -\bar{T}_{j+1} \cdot u_i(s, \bar{\pi}_{-i}^{[1:\bar{T}_{j+1}]})$. **Why utility is essential here.** For strategies already in $\Pi_i$, the losses $\ell_i^t(s)$ have been directly observed at each step $t$. For the newly added best response $\beta^*$, however, no losses were ever observed — it was not in the population. The bilinear collapse (Lemma 11) is precisely what enables the recovery: the entire unobserved loss history compresses exactly into $u_i(\beta^*, \bar{\pi}_{-i}^{[1:\bar{T}_{j+1}]})$, which *is* observable. This is why the reinitialisation formula must be stated in terms of utility, not loss.

Normalising over $\Pi^{(j+1)}$ (the $1/k$ factors cancel) and writing $\lambda = \eta\bar{T}_{j+1}$ gives the **VDO initialisation**:

$$\pi_{\mathrm{VDO}}(s) = \frac{\exp\left(\lambda \cdot u_i\left(s, \bar{\pi}_{-i}^{[1:\bar{T}_{j+1}]}\right)\right)}{\sum_{s' \in \Pi^{(j+1)}} \exp\left(\lambda \cdot u_i\left(s', \bar{\pi}_{-i}^{[1:\bar{T}_{j+1}]}\right)\right)}, \quad \lambda = \eta\bar{T}_{j+1}. \tag{2}$$

*Three important properties of Eq.* (2).

(1) The temperature $\lambda = \eta\bar{T}_{j+1}$ uses the total elapsed steps across all prior windows, not the current window length. This reflects the fact that the virtual MWU has been running since $t = 1$.
(2) The opponent average $\bar{\pi}_{-i}^{[1:\bar{T}_{j+1}]}$ is the *global* time-average from step 1, used for reinitialisation. Best-response computation uses the window-local average $\bar{\ell}_i$ (reset each window), as in OSO. The two quantities play distinct roles: the global average enables virtual weight recovery via utility; the window-local average determines which strategy to add.
(3) The formula applies to all $s \in \Pi^{(j+1)}$, not just the newcomer. Old strategies also receive updated weights reflecting the full virtual history.

Lemma 13 (Virtual weight consistency). *At each reinitialisation starting window $j+1$, the VDO initialisation* (2) *with $\lambda = \eta\bar{T}_{j+1}$ satisfies: for every $s \in \Pi^{(j+1)}$,*

$$\pi_{\mathrm{VDO}}(s) = \frac{w_{\mathcal{V}}^{\bar{T}_{j+1}}(s)}{\sum_{s' \in \Pi^{(j+1)}} w_{\mathcal{V}}^{\bar{T}_{j+1}}(s')},$$

*where $w_{\mathcal{V}}^{\bar{T}_{j+1}}(s)$ is the virtual weight (Definition 12) after $\bar{T}_{j+1}$ steps.*

Proof. From (1), $w_{\mathcal{V}}^{\bar{T}_{j+1}}(s) = \frac{1}{k}\exp(\lambda \cdot u_i(s, \bar{\pi}_{-i}^{[1:\bar{T}_{j+1}]}))$. The $1/k$ factors cancel upon normalisation over $\Pi^{(j+1)}$, yielding exactly $\pi_{\mathrm{VDO}}(s)$ with $\lambda = \eta\bar{T}_{j+1}$. □

Algorithm 4 presents VDO as a two-player algorithm combining both players in a single loop, mirroring ODO's structure (Algorithms 2–3). Each player $i$ maintains its own population $\Pi_i$, weights $w_i^t$, and two running averages: the global average $\bar{\pi}_{-i}^{\mathrm{gl}}$ over the full history from $t = 1$, and the window-local average loss $\bar{\ell}_i$ reset at each reinitialisation.

At each step, each player plays, observes the opponent, computes losses, updates both averages, applies MWU, and computes a best response to $\bar{\ell}_i$ (Lines 4–9), exactly as in OSO. A best response triggers population expansion and reinitialisation only if it is novel.

**The sole difference from ODO** is Line 12. ODO resets $w_i^{t+1}(s) \leftarrow 1/|\Pi_i|$ uniformly. VDO instead sets

$$w_i^{t+1}(s) \leftarrow \exp\left(\eta\bar{T} \cdot u_i\left(s, \bar{\pi}_{-i}^{\mathrm{gl}}\right)\right) \quad \forall s \in \Pi_i.$$

The utility form is *necessary*, not merely a notational convenience. The newly added best response $\beta_i^*$ was not in $\Pi_i$ before this step, so its individual losses $\ell_i^t(\beta_i^*)$ were never observed and cannot be summed directly. The bilinear collapse (Lemma 11) is precisely what makes recovery possible: the entire virtual loss history $L_i^{\bar{T}}(\beta_i^*)$ compresses exactly into the single quantity $-\bar{T} \cdot u_i(\beta_i^*, \bar{\pi}_{-i}^{\mathrm{gl}})$, which is computable from the global average alone. We apply the same utility-based formula uniformly to all $s \in \Pi_i$—including old strategies—to maintain exact consistency with the virtual MWU run (Lemma 13).

**Algorithm 4** Virtual Double Oracle (VDO)

**Require:** Full strategy sets $S_1, S_2$; learning rate $\eta$.
1: Init $\Pi_i \leftarrow \{s_i^0\}$ for $i \in \{1,2\}$; $w_i^1(s) \leftarrow 1, \forall s \in \Pi_i$; $\bar{\pi}_{-i}^{\mathrm{gl}} \leftarrow \mathbf{0}$; $\bar{\ell}_i \leftarrow \mathbf{0}$; $n_w \leftarrow 0$; $\bar{T} \leftarrow 0$.
2: **for** $t = 1, 2, \ldots$ **do**
3:  **for** $i \in \{1,2\}$ **do**
4:  **Play:** $\pi_i^t \leftarrow w_i^t / \|w_i^t\|_1$.
5:  **Observe:** $\pi_{-i}^t$; $\ell_i^t(s) \leftarrow -u_i(s, \pi_{-i}^t), \forall s \in \Pi_i$.
6:  **Global avg:** $\bar{\pi}_{-i}^{\mathrm{gl}} \leftarrow \frac{t-1}{t}\bar{\pi}_{-i}^{\mathrm{gl}} + \frac{1}{t}\pi_{-i}^t$; $\bar{T} \leftarrow t$.
7:  **Window avg:** $n_w \leftarrow n_w + 1$; $\bar{\ell}_i \leftarrow \bar{\ell}_i + (\ell_i^t - \bar{\ell}_i)/n_w$.
8:  **MWU:** $w_i^{t+1}(s) \leftarrow w_i^t(s)\exp(-\eta\,\ell_i^t(s)), \forall s \in \Pi_i$.
9:  **BR:** $\beta_i^* \leftarrow \arg\min_{s \in S_i} \langle e_s, \bar{\ell}_i \rangle$.
10:  **if** $\beta_i^* \notin \Pi_i$ **then**
11:  $\Pi_i \leftarrow \Pi_i \cup \{\beta_i^*\}$.
12:  **VDO reinit:** $w_i^{t+1}(s) \leftarrow \exp\left(\eta\bar{T} \cdot u_i\left(s, \bar{\pi}_{-i}^{\mathrm{gl}}\right)\right), \forall s \in \Pi_i$. {Eq. (2); replaces ODO's uniform reset}
13:  $\bar{\ell}_i \leftarrow \mathbf{0}$; $n_w \leftarrow 0$. {new window starts}
14:  **end if**
15:  **end for**
16: **end for**
17: **return** $\bar{\pi}_i^T = (1/T)\sum_{t=1}^T \pi_i^t$ for $i \in \{1,2\}$.

The global average $\bar{\pi}_{-i}^{\mathrm{gl}}$ is the only additional quantity VDO maintains beyond ODO, requiring $O(|S_{-i}|)$ extra storage and one incremental vector update per step. At each reinitialisation, computing $u_i(s, \bar{\pi}_{-i}^{\mathrm{gl}})$ for all $s \in \Pi_i$ requires a single matrix-vector product, performed once per expansion event.

The BR computation in VDO follows OSO exactly: a best response to the window-local average $\bar{\ell}_i$ is computed at every step, and population expansion occurs whenever a novel BR is found. Flexible BR scheduling is an orthogonal design axis explored in the FDO framework (Section 6).

## 5 THEORETICAL RESULTS

We establish two main results: a general no-regret bound parameterised by $\lambda$ (Theorem 14), and a global exploitability bound showing a $\sqrt{k}$ improvement over ODO (Theorem 17).

### 5.1 No-Regret Preservation

The first main result establishes that VDO preserves the no-regret guarantee for any choice of temperature $\lambda \geq 0$. The bound depends on how much initial weight VDO assigns to the best-in-hindsight strategy at each window: the more weight it receives, the tighter the bound. This is the mechanism by which VDO improves over ODO's uniform initialisation.

Theorem 14 (No-regret preservation). *For any $\lambda \geq 0$, VDO achieves, for each player $i \in \{1,2\}$:*

$$\frac{1}{T}\left(\sum_{t=1}^T \langle \pi^t, \ell^t \rangle - \min_{s \in \Pi^{(k)}} \sum_{t=1}^T \ell^t(s)\right) \leq \frac{1}{\sqrt{2T}}\sqrt{\sum_{j=1}^k \ln \frac{1}{\pi_{\mathrm{VDO},j}(s^*)}},$$

*where* $s^* = \arg\min_{s\in\Pi^{(k)}} \sum_{t=1}^{T} \ell^t(s)$ *is the best strategy in hindsight, and* $\pi_{\mathrm{VDO},j}(s^*)$ *is the initial probability VDO assigns to* $s^*$ *at the start of window* $j$*. Since VDO uses softmax initialisation,* $\pi_{\mathrm{VDO},j}(s^*) > 0$ *for all* $j$ *whenever* $\lambda$ *is finite.*

The proof follows the window-decomposition of [4], applying Lemma 8 within each window with VDO's softmax initialisation, then summing across windows via Cauchy–Schwarz; see Appendix A for the full derivation.

Two immediate consequences clarify the relationship between VDO and ODO.

Corollary 15 (ODO recovery). *Setting* $\lambda = 0$ *gives uniform initialisation* $\pi_{\mathrm{VDO},j}(s) = 1/j$ *for all* $s \in \Pi^{(j)}$*, so* $\ln(1/\pi_{\mathrm{VDO},j}(s^*)) = \ln j$*. Summing:* $\sum_{j=1}^{k} \ln j \le k \ln k$*. The bound becomes* $O(\sqrt{k \ln k / T})$*, recovering Theorem 9 exactly.*

Since $\lambda = 0$ recovers ODO, any $\lambda > 0$ can only help. The next corollary makes this precise for the newly added best response.

Corollary 16 (Best-response head start). *For* $\lambda > 0$*, let* $\beta_i$ *be the best response added at window* $j$ *for player* $i$*. By definition of best response,*

$$u_i\left(\beta_i, \bar{\pi}^{\mathrm{gl}}_{-i}\right) > u_i\left(s, \bar{\pi}^{\mathrm{gl}}_{-i}\right) \quad \forall s \in \Pi^{(j-1)},$$

*so* $\pi_{\mathrm{VDO},j}(\beta_i) > 1/j$*, giving* $\ln(1/\pi_{\mathrm{VDO},j}(\beta_i)) < \ln j$*. The within-window regret against the best-response comparator is* strictly smaller *than under ODO. In other words, the newly added best response enters with a head start proportional to its utility advantage over the existing population.*

## 5.2 Global Exploitability Bound

The second main result shows that the principled reinitialisation of VDO yields a strictly tighter exploitability bound than ODO. The key idea is that the hidden-strategy view (Section 4.1) makes the population conceptually static from $t = 1$, allowing the entire run to be analysed as a single MWU instance over $k$ strategies rather than $k$ separate windows.

Theorem 17 ($O(\sqrt{\ln k / T})$ bound). *VDO with* $\lambda = \eta \bar{T}_{j+1}$ *at each reinitialisation achieves:*

$$\mathrm{Exploit}(\bar{\pi}_1^T, \bar{\pi}_2^T) \;\le\; 2\sqrt{\frac{2\ln k}{T}}.$$

The proof (Appendix C) proceeds in four steps: **(1)** define a virtual MWU instance $\mathcal{V}$ running over all $k$ strategies from $t = 1$ with uniform initial weights $w^1_{\mathcal{V}}(s) = 1/k$; **(2)** show via Lemma 13 that VDO's played distribution at every step equals $\mathcal{V}$'s distribution conditioned on the revealed population $\Pi_i^{(j)}$; **(3)** apply Theorem 7 to the single instance $\mathcal{V}$, giving $\mathrm{Regret}(T) \le \sqrt{2T \ln k}$; **(4)** convert to exploitability via Proposition 6, summing both players' regret bounds and dividing by $T$.

Remark 18 (Why the improvement is structural). *ODO's* $O(\sqrt{k \ln k / T})$ *bound arises from Cauchy–Schwarz applied to*

$$\sum_{j=1}^{k} \sqrt{|T_j| \ln j},$$

*introducing a* $\sqrt{k}$ *factor via* $\sum_{j=1}^{k} \ln j \le k \ln k$ *[4]. VDO avoids this entirely: since the NE support is contained in* $\Pi^{(k)}$ *at termination, the full run is equivalent to a single MWU instance over* $k$ *strategies by Lemma 13, paying only* $\sqrt{\ln k}$*. No window decomposition is needed.*

# 6 THE FLEXIBLE DOUBLE ORACLE FRAMEWORK

The results of Section 5 show that VDO's softmax reinitialisation preserves no-regret guarantees while improving convergence. A natural question is: how special is VDO's particular choice of reinitialisation? In this section we show that VDO is one point in a broader family of algorithms, all sharing the same structure—an inner-loop no-regret solver punctuated by best-response expansions—but differing in two independent design choices: how often to query the BR oracle, and how to reinitialise. We call this the **Flexible Double Oracle (FDO)** framework.

## 6.1 BR Frequency and Regret

VDO queries the BR oracle at every step, matching the canonical OSO. The ODO paper also analyses a *less-frequent* variant [4] in which the BR is checked only when a threshold condition on the window-local regret is met, governed by a schedule $\alpha_j^{t-\bar{T}_j}$. With the specific choice $\alpha_j^{t-\bar{T}_j} = \sqrt{t - \bar{T}_j}$, the bound acquires an additional term:

$$\mathrm{Exploit}(\bar{\pi}_1^T, \bar{\pi}_2^T) \;\le\; 2\sqrt{\frac{k\ln k}{2T}} + \frac{\sqrt{k_1} + \sqrt{k_2}}{\sqrt{T}},$$

where $k_i$ is the final population size for player $i$. The extra term is lower order but non-negligible.

More generally, the PDO framework [13] parameterises BR frequency by a function $m : \{0, \dots, k-1\} \to \mathbb{N}$, where $m(j)$ is the number of inner-loop steps between BR checks in window $j$. The canonical OSO/VDO corresponds to $m(j) = 1$ for all $j$. Analysing VDO under general $m(\cdot)$ and deriving the optimal schedule is left as future work; for the theoretical results of Section 5 we assume $m(j) = 1$.

## 6.2 Softmax as Universal Reinitialisation

We establish that the softmax parameterisation used by VDO is not restrictive: any strictly positive reinitialisation distribution can be represented as a softmax over utilities against some opponent mixture. For a vector $z \in \mathbb{R}^n$ and $\lambda > 0$, the *softmax* is $\mathrm{softmax}(\lambda \cdot z)_s = \exp(\lambda z_s)/\sum_{s'} \exp(\lambda z_{s'})$.

Proposition 19 (Universal reinitialisation). *Let* $\Pi$ *be any finite strategy set,* $\pi^* \in \mathrm{int}(\Delta(\Pi))$ *any strictly positive target distribution, and* $\lambda > 0$*. Assume* $|S_{-i}| \ge |\Pi| - 1$*. Then there exists* $\bar{\pi}^*_{-i} \in \Delta(S_{-i})$ *such that:*

$$\mathrm{softmax}\big(\lambda \cdot u_i(\cdot, \bar{\pi}^*_{-i})\big)_s = \pi^*(s) \quad \forall s \in \Pi.$$

The proof (Appendix B) constructs $\bar{\pi}^*_{-i}$ as the solution to a linear system derived from the softmax log-ratio condition; the system is underdetermined when $|S_{-i}| \ge |\Pi| - 1$ and its solution set intersects $\Delta(S_{-i})$ whenever $\pi^* \in \mathrm{int}(\Delta(\Pi))$.

The practical implication is that any oracle—whether an exact NE solver, an RL agent, or domain-specific prior knowledge—that produces a full-support target $\pi^*$ can be incorporated into FDO by

solving this linear system. Since Theorem 14 requires only $\pi^{\text{init}}(s) > 0$ for all $s$, any interior solution automatically preserves the no-regret guarantee.

### 6.3 The FDO Framework

Definition 20 (FDO). *A **Flexible Double Oracle (FDO)** instance is parameterised by two independent controls:*

(1) ***BR frequency*** $m : \{0, \dots, k-1\} \to \mathbb{N}$*: $m(j)$ is the number of inner-loop steps between BR checks in window $j$.*
(2) ***Reinitialisation distribution****: a full-support distribution $\pi^j_{\text{init}} \in \text{int}(\Delta(\Pi^{(j)}))$ at the start of window $j$, specified directly or via $(\lambda, \bar{\pi}_{-i})$ through Proposition 19.*

*The inner-loop solver is any exponential-weights algorithm run from $\pi^j_{\text{init}}$.*

*No-regret preservation.* Theorem 14 applies to any FDO instance with full-support reinitialisation. For $m(j) = 1$ (the VDO case), the global bound of Theorem 17 also holds. For general $m(j)$, the no-regret guarantee is preserved but the exploitability bound may acquire additional terms depending on $m(\cdot)$, as discussed in Section 6.1.

### 6.4 Classical Algorithms as FDO Special Cases

Table 1 in appendix D shows how the two FDO controls recover the major algorithms in the Double Oracle family.

*Double Oracle (DO) and ADO..* Both solve a restricted game to NE (exact or one-sided), yielding full-support distributions $\pi^r$ and $\pi^{\text{ado}}$ recoverable via Proposition 19. In FDO terms, this corresponds to setting $m = \infty$ (run the inner loop to convergence before expanding) and using the restricted NE as the reinitialisation distribution.

*ODO and PDO..* Both use uniform reinitialisation ($\lambda = 0$). PDO generalises ODO via a non-uniform frequency schedule $m(j)$ [13]; VDO generalises ODO via a non-trivial $\pi^{\text{init}}$. The two generalisations are orthogonal and composable: combining VDO's reinitialisation with PDO's frequency schedule gives a strictly more general algorithm.

*Fictitious play.* $m = 1$ (recompute BR every step), $\lambda \to \infty$ (concentrate all weight on the best response), and $\bar{\pi} = \bar{\pi}^{\text{gl}}_{-}$ (global average). With $\lambda \to \infty$ and $m = 1$, the algorithm plays the pure best response to the global average each step, recovering fictitious play exactly. This illustrates that the two extremes of the $\lambda$ spectrum — $\lambda = 0$ (uniform, ODO) and $\lambda \to \infty$ (point mass, fictitious play) — are both captured by FDO.

*VDO.* The unique FDO instance in which reinitialisation exactly recovers Virtual MWU weights (Lemma 13). It occupies a principled middle ground between ODO's uniform reset and fictitious play's full concentration, with the temperature $\lambda = \eta\bar{T}$ derived from theory rather than chosen heuristically.

### 6.5 Flexible $\lambda$: Theory vs. Practice

Theorem 14 guarantees no-regret for any $\lambda \in [0, \infty)$, giving the practitioner freedom to tune. The theoretically motivated $\lambda = \eta\bar{T}_{i+1}$ ensures virtual weight consistency (Lemma 13), but in practice $\lambda$ controls an exploration–exploitation tradeoff:

- $\lambda = 0$: uniform reinitialisation, recovering ODO.
- $\lambda = \eta\bar{T}$: virtually consistent, theoretically principled default. Gives smooth, consistent empirical improvement.
- **Large** $\lambda$: aggressive initialisation concentrated on high-utility strategies. Fastest initial convergence in experiments but may oscillate in complex games.

An attractive property of the default $\lambda = \eta\bar{T}$ is that it naturally adapts to history: $\bar{T}$ grows with each window, so VDO becomes increasingly confident about which strategies are valuable as history accumulates. Early windows behave conservatively (small $\lambda$, close to uniform); later windows exploit accumulated knowledge (large $\lambda$, concentrated on high-utility strategies). The experiments in Section 7 confirm this: $\lambda = \eta\bar{T}$ gives smooth, consistent improvement across all games tested, while empirical tuning of $\lambda$ yields further gains in games with heterogeneous utility landscapes.

## 7 EXPERIMENTS

### 7.1 Setup

We compare VDO against ODO [4] and warm-start ODO (carry-forward weights) on three game classes using OpenSpiel [7]:

- **Kuhn poker**: small two-player imperfect-information game; NE support size 2.
- **Leduc poker**: larger game with community card; NE support size $\geq 6$.
- **Random** $50\times50$ **matrix games**: i.i.d. $\mathcal{N}(0, 1)$ entries.

All the games are in the strategic form representation. VDO is implemented as a drop-in replacement for ODO's reinitialisation function: the only change is the weight update at population expansion (Lines 9–10 of Algorithm 4 vs. Line 9 of Algorithm 3). All results report exploitability vs. BR oracle calls, averaged over 10 seeds with 95% confidence intervals.

### 7.2 Effect of $\lambda$

Figure 1 compares three choices of the temperature parameter across all three game classes: $\lambda = 0$ (ODO, uniform reinitialisation), $\lambda = \eta\bar{T}$ (VDO, theoretically derived), and $\lambda = 10$ (FDO, fixed empirical value). Key findings:

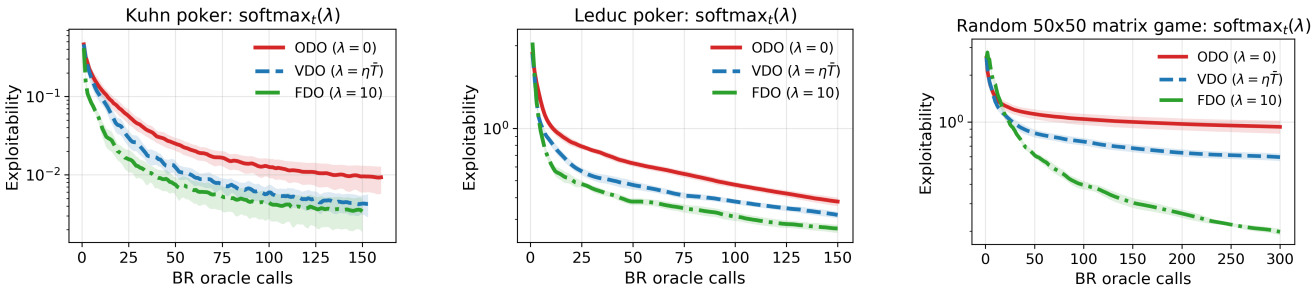
- $\lambda = 0$ (ODO): baseline, slowest convergence across all games.
- $\lambda = \eta\bar{T}$ (VDO): consistently faster convergence with a smooth trajectory, matching the theoretical prediction.
- $\lambda = 10$ (FDO): fastest initial descent in Leduc and random games, but may oscillate in later iterations as strategies cycle.

The theoretically derived $\lambda = \eta\bar{T}$ provides a robust default across all three games, while the fixed $\lambda = 10$ offers further gains in games with more heterogeneous utility landscapes.

### 7.3 Comparison with Baselines

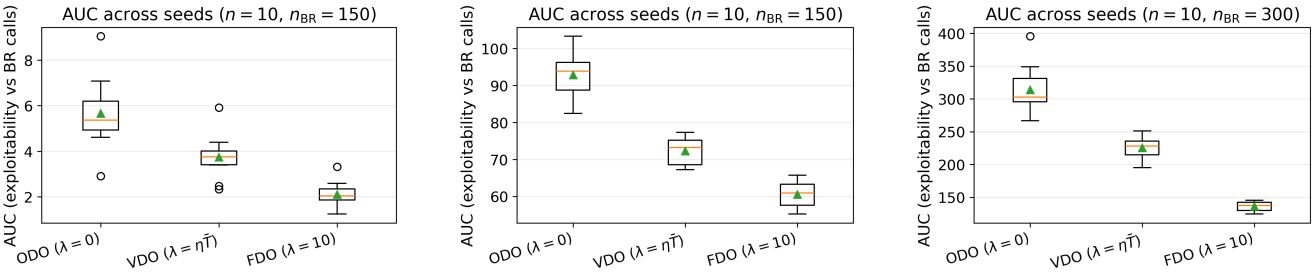
We compare methods using AUC of the exploitability-vs-BR curve, which captures improvements over the entire training trajectory.

Across all three domains, VDO and FDO substantially outperform ODO. Paired statistical tests across seeds (paired $t$-test and Wilcoxon signed-rank) confirm that both VDO and FDO yield significantly lower final exploitability and significantly lower AUC than ODO; FDO is strongest overall.




**(a) Kuhn poker** **(b) Leduc poker** **(c) Random** 50×50

**Figure 1: Effect of $\lambda$ on exploitability versus BR oracle calls. Each panel shows the mean curve over 10 seeds with 95% confidence intervals (shaded). We compare ODO ($\lambda = 0$; solid red), VDO ($\lambda = \eta\bar{T}$; dashed), and FDO ($\lambda = 10$; dash-dotted). Across all three domains, increasing $\lambda$ improves convergence: both VDO and FDO lie below ODO throughout the trajectory, and FDO achieves the lowest exploitability at the same BR budget.**




**(a) Kuhn poker** **(b) Leduc poker** **(c) Random** 50×50

**Figure 2: AUC comparison across seeds (lower is better). Each panel shows the distribution over 10 seeds of the area under the exploitability-vs-BR curve, computed over the shared BR horizon within that game (Kuhn/Leduc: 150 BR calls; Random $50 \times 50$: 300 BR calls). Boxes show the interquartile range with median; whiskers denote 1.5×IQR; markers indicate the mean. VDO and FDO consistently reduce AUC relative to ODO, indicating faster progress throughout the entire trajectory, not only at the endpoint.**

## 7.4 Discussion

Across all three game classes, choosing $\lambda > 0$ yields consistent and statistically significant improvements over ODO ($\lambda = 0$). In Kuhn poker, both VDO and FDO significantly reduce final exploitability (paired $t$-test: VDO $p = 2.9 \times 10^{-3}$, FDO $p = 1.25 \times 10^{-3}$) and AUC (VDO $p = 2.0 \times 10^{-5}$, FDO $p = 5.7 \times 10^{-5}$), with Wilcoxon $p = 0.00195$ in both cases. Similar statistically significant reductions hold for Leduc poker (final: VDO $p = 1.35 \times 10^{-4}$, FDO $p = 2.16 \times 10^{-6}$; AUC: VDO $p = 1.40 \times 10^{-8}$, FDO $p = 3.47 \times 10^{-8}$) and for random $50 \times 50$ matrix games (final: VDO $p = 4.19 \times 10^{-5}$, FDO $p = 7.02 \times 10^{-8}$; AUC: VDO $p = 1.54 \times 10^{-4}$, FDO $p = 1.03 \times 10^{-7}$).

Overall, $\lambda = \eta\bar{T}$ (VDO) is a robust principled default, while a larger tuned value (FDO, here $\lambda = 10$) yields the strongest empirical performance at the same BR budget.

## 8 CONCLUSION

We introduced VDO, resolving the open problem of principled reinitialisation in ODO. The central insight—bilinearity makes virtual loss accumulation exact and compressible into a single time-average quantity—leads to a closed-form initialisation formula (2), no-regret preservation for any $\lambda \geq 0$, and $O(\sqrt{\ln k/T})$ exploitability, a $\sqrt{k}$ improvement over ODO's $O(\sqrt{k \ln k/T})$. VDO is a minimal change to ODO: one reinitialisation line replaced, one running average maintained. The FDO framework places VDO in a broad two-parameter family subsuming the entire DO family, where Proposition 19 enables any oracle to specify the initial distribution without losing no-regret guarantees.

*Future work* includes: extension to extensive-form games via per-information-set virtual values (connecting to XDO [9] and PDO [13]); adaptive $\lambda$ schedules; and composition of VDO's reinitialisation with PDO's frequency schedule.

## REFERENCES


[1] Sanjeev Arora, Elad Hazan, and Satyen Kale. 2012. The Multiplicative Weights Update Method: A Meta-Algorithm and Applications. *Theory of Computing* 8, 1 (2012), 121–164.

[2] Nicolò Cesa-Bianchi and Gábor Lugosi. 2006. *Prediction, Learning, and Games*. Cambridge University Press, Cambridge, UK.

[3] Wojciech M. Czarnecki, Gauthier Gidel, Brendan Tracey, Karl Tuyls, Shayegan Omidshafiei, David Balduzzi, and Max Jaderberg. 2020. Real World Games Look Like Spinning Tops. In *Advances in Neural Information Processing Systems (NeurIPS 2020)*, Vol. 33. 17443–17454.

[4] Le Cong Dinh, Stephen M. McAleer, Zheng Tian, Nicolas Perez-Nieves, Oliver Slumbers, David H. Mguni, Jun Wang, Haitham Bou Ammar, and Yaodong Yang. 2022. Online Double Oracle. *Transactions on Machine Learning Research* (2022). https://openreview.net/forum?id=rrMK6hYNSx

[5] Yoav Freund and Robert E. Schapire. 1997. A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *J. Comput. System Sci.* 55, 1 (1997), 119–139.

[6] Yoav Freund and Robert E. Schapire. 1999. Adaptive Game Playing using Multiplicative Weights. *Games and Economic Behavior* 29, 1-2 (1999), 79–103.

[7] Marc Lanctot, Edward Lockhart, Jean-Baptiste Lespiau, Vinicius Zambaldi, Satyaki Upadhyay, Julien Pérolat, Sriram Srinivasan, Finbarr Timbers, Karl Tuyls, Shayegan Omidshafiei, et al. 2019. OpenSpiel: A Framework for Reinforcement Learning in Games. *arXiv preprint arXiv:1908.09453* (2019).

[8] Marc Lanctot, Vinicius Zambaldi, Audrunas Gruslys, Angeliki Lazaridou, Karl Tuyls, Julien Pérolat, David Silver, and Thore Graepel. 2017. A Unified Game-Theoretic Approach to Multiagent Reinforcement Learning. In *Advances in Neural Information Processing Systems (NeurIPS 2017)*, Vol. 30. 4190–4203.

[9] Stephen McAleer, John B. Lanier, Kevin A. Wang, Pierre Baldi, and Roy Fox. 2021. XDO: A Double Oracle Algorithm for Extensive-Form Games. In *Advances in Neural Information Processing Systems (NeurIPS 2021)*, Vol. 34. 23128–23139.

[10] Stephen McAleer, Kevin Wang, Marc Lanctot, John B. Lanier, Pierre Baldi, and Roy Fox. 2022. Anytime Optimal PSRO for Two-Player Zero-Sum Games. *arXiv preprint arXiv:2201.07700* (2022).

[11] H. Brendan McMahan, Geoffrey J. Gordon, and Avrim Blum. 2003. Planning in the Presence of Cost Functions Controlled by an Adversary. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*. 536–543.

[12] Milind Tambe. 2011. *Security and Game Theory: Algorithms, Deployed Systems, Lessons Learned*. Cambridge University Press, Cambridge, UK.

[13] Xiaohang Tang, Le Cong Dinh, Stephen Marcus McAleer, and Yaodong Yang. 2023. Regret-Minimizing Double Oracle for Extensive-Form Games. In *Proceedings of the 40th International Conference on Machine Learning (ICML 2023) (Proceedings of Machine Learning Research, Vol. 202)*. PMLR, 33599–33615.

[14] Oriol Vinyals, Igor Babuschkin, Wojciech M. Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H. Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. 2019. Grandmaster Level in StarCraft II Using Multi-Agent Reinforcement Learning. *Nature* 575, 7782 (2019), 350–354. https://doi.org/10.1038/s41586-019-1724-z

[15] Yongzhao Wang and Michael P. Wellman. 2023. Regularization for Strategy Exploration in Empirical Game-Theoretic Analysis. *arXiv preprint arXiv:2302.04928* (2023).

[16] Ming Zhou, Jingxiao Chen, Ying Wen, Weinan Zhang, Yaodong Yang, Yong Yu, and Jun Wang. 2022. Efficient Policy Space Response Oracles. *arXiv preprint arXiv:2202.00633* (2022).


## A PROOF OF THEOREM 14

Proof. Apply the window-decomposition of Dinh et al. 2022. Since $s^* \in \Pi^{(i)}$ for all $i \geq i^*$ (the window $s^*$ entered), the total regret decomposes via $\sum_i \min \leq \min \sum$ [4]:

$$\sum_{t=1}^{T} \langle \pi_i^t, \ell_i^t \rangle - \min_s \sum_t \ell_i^t(s) \leq \sum_{i=1}^{k} \left[ \sum_{t \in T_i} \langle \pi_i^t, \ell_i^t \rangle - \min_{s \in \Pi^{(i)}} \sum_{t \in T_i} \ell_i^t(s) \right].$$

Within window $i$, apply Lemma 8 with initial weights $w^1_{(i)}(s) = \pi^{(s)}_{\mathrm{VDO},i}$, which form a valid distribution ($\sum_s \pi^{(s)}_{\mathrm{VDO},i} = 1$). Since $\pi^{(s)}_{\mathrm{VDO},i} = \mathrm{softmax}(\lambda \cdot u(s,\cdot))_s > 0$ for finite $\lambda$:

$$\sum_{t \in T_i} \langle \pi_i^t, \ell_i^t \rangle - \sum_{t \in T_i} \ell_i^t(s^*) \;\leq\; \frac{1}{\eta_i} \ln \frac{1}{\pi^{(s^*)}_{\mathrm{VDO},i}} + \frac{\eta_i |T_i|}{8}.$$

Optimising $\eta_i = \sqrt{8 \ln(1/\pi^{(s^*)}_{\mathrm{VDO},i})/|T_i|}$ per window, then summing and applying Cauchy–Schwarz ($\sum_i a_i b_i \leq \sqrt{\sum_i a_i^2}\sqrt{\sum_i b_i^2}$ with $a_i = \sqrt{|T_i|}$, $b_i = \sqrt{\ln(1/\pi^{(s^*)}_{\mathrm{VDO},i})}$, $\sum_i |T_i| = T$):

$$\sum_{i=1}^{k} \sqrt{\tfrac{|T_i|}{2} \ln \tfrac{1}{\pi^{(s^*)}_{\mathrm{VDO},i}}} \leq \sqrt{\tfrac{T}{2} \sum_{i=1}^{k} \ln \tfrac{1}{\pi^{(s^*)}_{\mathrm{VDO},i}}}.$$

Dividing by $T$ gives the result. □

## B PROOF OF PROPOSITION 19

Proof. Fix any $s_0 \in \Pi$ as a reference strategy. Matching the softmax log-ratios for all $s \neq s_0$ requires:

$$\lambda\,(e_s - e_{s_0})^\top A\,\bar{\pi}^*_{-i} = \ln \pi^*(s) - \ln \pi^*(s_0), \quad \forall s \neq s_0. \tag{3}$$

The softmax denominator cancels in log-ratios, so (3) is a linear system $M\bar{\pi}^*_{-i} = r$, where $M \in \mathbb{R}^{(|\Pi|-1)\times|S_{-i}|}$ has rows $M_s = \lambda(e_s - e_{s_0})^\top A$ and $r_s = \ln \pi^*(s) - \ln \pi^*(s_0)$.

Since $|S_{-i}| \geq |\Pi| - 1$ by assumption, the system is underdetermined (more unknowns than equations). Its solution set is the affine subspace $\{M^\dagger r + (I - M^\dagger M)v : v \in \mathbb{R}^{|S_{-i}|}\}$, which is non-empty (the pseudoinverse gives one solution). Since $\pi^* \in \mathrm{int}(\Delta(\Pi))$, the right-hand side $r$ is finite, and by continuity of the affine map the solution set intersects $\Delta(S_{-i})$. Any such intersection point is a valid $\bar{\pi}^*_{-i}$. □

## C PROOF OF THEOREM 17

Proof. We establish equivalence to a single MWU run over $k$ strategies, then apply the standard regret bound.

**Step 1: Virtual MWU.** Define virtual MWU $\mathcal{V}$ as a single MWU run over $\Pi^{(k)}$ from $t = 1$ with uniform weights $w^1_{\mathcal{V}}(s) = 1/k$, receiving the identical loss sequence as VDO. By Lemma 11 (bilinear collapse) and unrolling the MWU update:

$$w^{\mathcal{V}}_\tau(s) = \frac{1}{k} \exp\!\Big(\eta\tau \cdot u(s, \bar{\pi}_-^{[1:\tau]})\Big) \quad \text{for any } \tau.$$

**Step 2: VDO matches $\mathcal{V}$ on revealed strategies.** By Lemma 13 (virtual weight consistency), VDO initialises window $i$ with weights proportional to $w^{\mathcal{V}}_{\bar{T}_i}(s)$ for $s \in \Pi^{(i)}$. Since both VDO and $\mathcal{V}$ apply identical MWU updates with identical losses within each window:

$$w^{\mathrm{VDO}}_t(s) \propto w^{\mathcal{V}}_t(s), \quad \forall s \in \Pi^{(i)},\ t \in T_i.$$

Thus VDO's played distribution is $\mathcal{V}$'s distribution conditioned on the revealed set $\Pi^{(i)}$. The virtual strategies in $\mathcal{V}$ are used only for analysis and never affect the actual loss sequence, so there is no mismatch between the hidden-strategy view and actual play.

**Step 3: Applying the regret bound.** Since VDO is consistent with $\mathcal{V}$ and $\mathcal{V}$ is a single MWU instance over $k$ strategies receiving the actual opponent losses, applying Theorem 7 with $\eta = \sqrt{2\ln k/T}$:

$$\mathrm{Regret}(T) \;\leq\; \sqrt{2T \ln k}.$$

**Step 4: Two-player conversion.** Both players run VDO against each other's actual play. By the standard regret-to-exploitability conversion [5]:

$$\mathrm{Exploit}(\bar{\pi}^T) \;\leq\; \frac{\mathrm{Regret}_1(T) + \mathrm{Regret}_2(T)}{T} \;\leq\; 2\sqrt{\frac{2\ln k}{T}}.$$

□

## D TABLES

**Table 1: Classical algorithms as special cases of FDO. Controls are window length $m$ and reinitialisation $\pi^{\text{init}}$.**

| Algorithm | Window length $m$ | Reinitialisation $\pi^{\text{init}}$ | Softmax params |
|---|---|---|---|
| DO [11] | $\infty$ (solve to NE each time) | Restricted NE $\pi^r$ | $(\lambda, \bar{\pi})$ s.t. softmax= $\pi^r$ (Prop. 19) |
| ADO [10] | $\infty$ | One-sided restricted NE $\pi^{\text{ado}}$ | $(\lambda, \bar{\pi})$ s.t. softmax= $\pi^{\text{ado}}$ |
| ODO [4] | any fixed $m$ | Uniform: $1/\lvert\Pi^{(j)}\rvert$ | $\lambda = 0$ (any $\bar{\pi}$) |
| PDO [13] | schedule $m(j)$ | Uniform | $\lambda = 0$ |
| Fictitious play | $m = 1$ | Pure BR: $\delta_{\beta^*}$ | $\lambda \to \infty$, $\bar{\pi} = \bar{\pi}^{\text{gl}}_{-}$ |
| VDO (this paper) | any fixed $m$ | Virtual MWU/Replicator | $\lambda = \eta\bar{T}_j$, $\bar{\pi} = \bar{\pi}^{\text{gl}}_{-}$ |
| FDO (general) | any schedule $m(j)$ | Any oracle-specified $\pi^*$ | Recovered via Prop. 19 |