**Pure Mathematics** 理论数学, 2026, 16(3), 191-204
Published Online March 2026 in Hans. https://www.hanspub.org/journal/pm
https://doi.org/10.12677/pm.2026.163082

Hans汉斯

# 零和博弈下的策略惯性正则化多智能体Actor-Critic算法

**陈龙曰，高红伟***

青岛大学数学与统计学院，山东 青岛

收稿日期：2026年1月26日；录用日期：2026年2月27日；发布日期：2026年3月25日

## 摘　要

**随着多智能体系统在复杂物理环境中的广泛应用，如何解决多智能体随机博弈过程中的动力学非平稳性与物理安全约束缺失问题，已成为强化学习领域亟待突破的关键挑战。针对传统算法在随机博弈对抗中易陷入策略震荡以及依赖“试错”机制无法保障系统绝对安全的核心缺陷，本文提出了一种基于策略惯性正则化多智能体Actor-Critic算法。针对随机博弈环境下的纳什均衡收敛难题，本文建立了基于策略惯性正则化的博弈动力学模型。通过在优化目标中引入基于欧氏距离的策略锚点约束，重构了多智能体博弈的优化景观。微分动力学谱分析表明，该正则化机制使得系统雅可比矩阵的特征值实部发生负向平移，将原本不稳定的纳什均衡点转化为局部渐近稳定的，有效抑制了随机博弈中的高频梯度抖动与策略循环现象。在“湿滑网格世界”的数值对比实验中，该机制成功抑制了高频梯度抖动与策略退化现象，使智能体在强随机干扰下仍能习得时间最优路径，验证了改进算法在非平稳环境下的鲁棒收敛能力。**

## 关键词

**多智能体强化学习，随机博弈，策略惯性正则化**

# Policy Inertia Regularization Multi-Agent Actor-Critic Algorithm in Zero-Sum Game

**Longyue Chen, Hongwei Gao***

School of Mathematics and Statistics, Qingdao University, Qingdao Shandong

Received: January 26, 2026; accepted: February 27, 2026; published: March 25, 2026

## Abstract

**With the widespread application of multi-agent systems in complex physical environments,**

*通讯作者。

文章引用: 陈龙曰, 高红伟. 零和博弈下的策略惯性正则化多智能体 Actor-Critic 算法[J]. 理论数学, 2026, 16(3): 191-204. DOI: 10.12677/pm.2026.163082

**addressing the dynamic non-stationarity and lack of physical security constraints in multi-agent stochastic games has become a key challenge in reinforcement learning. To address the core shortcomings of traditional algorithms, such as their susceptibility to policy oscillations in stochastic adversarial games and the inability to guarantee absolute system security through trial-and-error mechanisms, this paper proposes a multi-agent Actor-Critic algorithm based on policy inertia regularization. To solve the Nash equilibrium convergence problem in stochastic game environments, this paper establishes a game dynamics model based on policy inertia regularization. By introducing policy anchor constraints based on Euclidean distance into the optimization objective, the optimization landscape of multi-agent games is reconstructed. Differential dynamics spectrum analysis shows that this regularization mechanism causes a negative shift in the real parts of the eigenvalues of the system's Jacobian matrix, transforming the originally unstable Nash equilibrium point into a locally asymptotically stable one, effectively suppressing high-frequency gradient jitter and policy looping in stochastic games. In a numerical comparison experiment using a "slippery grid world", this mechanism successfully suppressed high-frequency gradient jitter and policy degradation, enabling the agent to learn the time-optimal path even under strong random disturbances. This verifies the robust convergence capability of the improved algorithm in non-stationary environments.**

## Keywords

**Multi-Agent Reinforcement Learning, Stochastic Game, Policy Inertia Regularization**



## 1. 引言

多智能体系统在无人机编队、自动驾驶车队及仓储机器人等领域的应用日益广泛。作为解决多智能体序贯决策问题的核心方法，强化学习通过智能体与环境的交互来优化策略，已在众多领域展现出显著的性能[1] [2]。然而，将强化学习应用于具有竞争性质的随机博弈环境时，仍面临着非平稳性收敛难题。

多智能体博弈动力学的非平稳性与发散问题：随机博弈描述了多个局中人在动态环境中的交互过程，其状态转移概率与奖励函数显式依赖于所有参与者的联合动作[3]。在标准的多智能体 Actor-Critic 算法框架下，各智能体并发更新策略参数导致环境呈现高度非平稳特性[4]。对于单一智能体而言，对手策略的时变性使得目标函数不再满足马尔可夫决策过程的静态假设，导致价值网络的估计误差在贝尔曼迭代中累积放大[5]。特别是在零和博弈中，各方利益的严格对立导致梯度场包含显著的旋转分量。标准的梯度上升算法使得策略参数在单纯形顶点之间产生极限环震荡，系统缺乏耗散梯度场旋转能量的机制，从而无法收敛至纳什均衡点。

针对收敛性的问题，本文提出了一种基于策略惯性正则化的 Actor-Critic 改进算法，通过在原始目标函数中引入基于欧氏距离的策略锚点约束项，重构了多智能体博弈的优化景观。从微分动力系统的角度分析，该正则化项相当于在参数更新的动力学方程中显式引入了一个与参数变化率成正比的耗散力。通过对连续时间动力系统的雅可比矩阵进行谱分析证明，引入惯性系数后，系统特征值的实部发生了负向平移，确保了所有特征值的实部严格小于零。这一数学变换成功地将原本处于临界稳定或不稳定状态的纳什均衡点转化为局部渐近稳定的吸引子，利用 Lyapunov 第一法严格证明了改进算法在博弈均衡点邻域内的收敛稳定性，为解决多智能体学习中的震荡问题提供了坚实的控制理论依据。

多智能体强化学习的核心挑战在于多个决策主体在共享环境中的交互导致了马尔可夫决策过程平稳

性的丧失。早期的研究试图将单智能体算法直接应用于多智能体环境，即独立学习范式。然而 Tan 等人的研究表明，由于其他智能体策略参数的时变特性，环境状态转移概率分布呈现出非平稳漂移，导致独立 Q 学习算法中的经验回放机制失效，无法保证算法收敛至纳什均衡[6]。针对这一问题，集中式训练去中心化执行框架应运而生。Lowe 等人提出的多智能体深度确定性策略梯度算法通过引入集中式价值网络来逼近联合状态动作价值函数，利用全局信息降低了策略梯度估计的方差，在一定程度上缓解了环境非平稳性带来的震荡。Rashid 等人提出的 QMIX 算法则通过引入单调性约束，将联合价值函数分解为个体价值函数[7]，解决了协作环境下的信用分配难题。

尽管集中式价值评估提升了算法的稳定性，但在零和博弈或一般和博弈的微分动力学系统中，单纯的梯度上升更新仍面临严峻的理论瓶颈。在博弈论视角下，各智能体收益函数的对立性使得系统的梯度向量场包含显著的旋转分量。Singh 等人指出，在纳什均衡点附近，标准的梯度动力学可能导致策略参数在单纯形顶点之间形成极限环震荡[8]，系统雅可比矩阵的特征值实部为零或正值，缺乏收敛所需的耗散阻尼。为了解决这一动力学不稳定性，二阶优化方法逐渐受到关注。Foerster 等人提出的 LOLA 算法通过在策略更新中引入对对手学习过程的预测[9]，利用二阶泰勒展开重塑梯度流场，使得智能体能够通过预判对手行为来调整自身策略。Letcher 等人进一步提出了辛梯度调节方法[10]，通过分解哈密顿向量场来分离保守分量与耗散分量，从而设计出能够保证局部渐近稳定的更新规则。

然而，高阶优化方法通常涉及 Hessian 矩阵的计算，在高维神经网络参数空间中计算复杂度过高。因此，基于正则化的其一阶近似方法成为当前研究的热点。Schulman 等人提出的信任域策略优化算法通过限制新旧策略之间的 KL 散度[11]，确保了策略更新的单调改进，但其在多智能体博弈中的直接应用常因联合策略空间的拓扑复杂性而失效。为了增强策略的鲁棒性与探索能力，Haarnoja 等人提出的软 Actor-Critic 算法引入了最大熵正则化项[12]，将优化目标转化为最大化累积回报与策略熵的加权和，在一定程度上平滑了优化景观。近期的研究开始聚焦于策略惯性与锚点机制。Czarnecki 等人探讨了在博弈动力学中引入策略正则化项对纳什均衡收敛性的影响[13]，证明了适当的正则化能够将原本发散的梯度流转化为收敛流。Littman 正式提出了将马尔可夫博弈作为多智能体强化学习的理论框架[14]，为后续随机博弈的研究奠定了基础。

## 2. 零和博弈下的策略惯性正则化多智能体 Actor-Critic 算法

在多智能体系统中，标准的 Actor-Critic 算法虽然提供了一种基于梯度优化的求解范式，但由于环境的非平稳性以及目标函数的非凸性，该类算法在实际应用中往往面临探索能力不足与易陷入局部最优的严峻挑战。仅仅依赖确定性策略或简单的随机噪声，难以在复杂的策略空间中寻找到全局最优的纳什均衡点。针对上述问题，本章提出了一种基于最大熵正则化的多智能体 Actor-Critic 改进算法。该方法通过在优化目标中引入策略熵项，将寻找纳什均衡的问题转化为寻找软纳什均衡的问题[15]，从而在最大化累积期望回报的同时，显式地鼓励智能体保持策略的随机性与探索性。

### 2.1. 标准多智能体 Actor-Critic 算法

本文所基于的基准算法采用了集中式训练去中心化执行的架构。在该框架下，每个智能体 $i \in \{1,\cdots,N\}$ 的状态值函数都由两个神经网络来近似：策略网络(Actor)与价值网络(Critic)。其中，主要通过 TD Learning 算法来更新价值网络(Critic)中的参数以使得其对智能体的动作预测越来越准确；通过策略梯度算法更新价值网络(Critic)使得每个智能体获得越来越高的累计折扣奖励。

策略网络(Actor)：记为 $\pi_i(a_i \mid s_i;\theta_i)$，其中 $\theta_i$ 为神经网络参数。该网络仅接收智能体 $i$ 的局部观测 $s_i$，并输出动作空间上的概率分布。这一设计保证了算法在执行阶段仅需局部信息即可完成决策，满足分布

式控制的要求。

价值网络(Critic)：记为$Q_i\left(\boldsymbol{s},\boldsymbol{a};\boldsymbol{w}_i\right)$，其中$w_i$为神经网络参数，$\boldsymbol{s}=\left(s_1,\cdots,s_N\right)$为联合状态，$\boldsymbol{a}=\left(a_1,\cdots,a_N\right)$为联合动作。Critic 网络利用全局信息来评估当前联合策略在特定状态下的价值。

Critic 更新：

Critic 的目标是准确估计联合策略$\pi$下的动作值函数$Q_i^{\pi}\left(\boldsymbol{s},\boldsymbol{a}\right)$。根据算法设定，在时刻$t$，对于智能体$i$，其 Critic 网络的更新基于时序差分误差。

首先，计算目标 Q 值。根据下一时刻的联合状态$\boldsymbol{s}^{t+1}$，各智能体依据当前策略采样出下一时刻的动作$\tilde{a}_i^{t+1}\sim\pi_i\left(\cdot\mid s_i^{t+1};\theta_i\right)$，构成联合动作$\tilde{\boldsymbol{a}}^{t+1}$。目标值$y_i^t$定义为：

$$y_i^t=r_i^t+\gamma Q_i\left(\boldsymbol{s}^{t+1},\tilde{\boldsymbol{a}}^{t+1};w_i\right) \tag{1}$$

其中，$r_i^t$是环境反馈的即时奖励，$r$是折扣因子。

接着，定义 Critic 的损失函数$L\left(w_i\right)$为 TD 误差的均方误差：

$$L\left(w_i\right)=\frac{1}{2}\left(y_i^t-q_i^t\right)^2 \tag{2}$$

Critic 参数$w_i$通过梯度下降法进行更新：

$$w_i^{t+1}\leftarrow w_i^t-\alpha_i\cdot\delta_i^t\cdot\nabla_{w_i}Q_i\left(\boldsymbol{s}^t,\boldsymbol{a}^t;w_i\right)$$

其中$\delta_i^t=y_i^t-Q_i\left(\boldsymbol{s}^t,\boldsymbol{a}^t;w_i\right)$为 TD 误差，$\alpha_i$为 Critic 的学习率。

Actor 更新：

Actor 的目标是最大化智能体的期望累积折扣回报$J\left(\theta_i\right)=\mathbb{E}\left[\sum_t\gamma^t r_i^t\right]$。基于策略梯度定理，参数$\theta_i$的更新方向由 Critic 提供的价值评估指导。在 Q-based Actor-Critic 算法中，梯度方向近似为策略对数概率的梯度与当前 Q 值的乘积。具体的更新规则为

$$\theta_i^{t+1}\leftarrow\theta_i^t+\beta_i\cdot Q_i\left(\boldsymbol{s}^t,\boldsymbol{a}^t;w_i^t\right)\cdot\nabla_{\theta_i}\log\pi_i\left(a_i^t\mid s_i^t;\theta_i\right)$$

其中$\beta_i$是 Actor 的学习率。该式直观地表明：若某个动作$a_i^t$在当前全局状态下产生了较高的 Q 值(即$Q_i$较大)，则 Actor 会沿着增加该动作概率的方向更新参数$\theta_i$；反之则减小该动作的概率。通过上述 Actor 与 Critic 的交替迭代更新，算法旨在寻找纳什均衡策略$\pi^*$，使得没有任何智能体能通过单方面改变策略获得更高的收益。

## 2.2. 现有方法的局限性分析

标准多智能体 Actor-Critic 算法的理论缺陷源于缺乏对策略更新幅度的有效约束。首先，多智能体并发学习使得环境的状态转移与奖励反馈不再满足平稳性前提，导致价值网络拟合的目标始终处于动态漂移中，破坏了学习过程的连贯性。其次，在博弈动力学层面，各参与者仅依据局部梯度进行的贪婪更新极易进入非收敛的循环轨道，使策略在离散状态之间剧烈震荡而无法稳定在最优混合均衡点。最后，由于网络参数与输出概率分布之间存在高度非线性的映射关系，无约束的参数更新会导致策略分布发生不可控的突变，一旦新策略进入低价值区域，将引发采样质量恶化与价值评估失效的恶性循环，最终导致系统性能崩塌。

## 2.3. 针对传统算法的改进机制分析

标准多智能体 Actor-Critic 算法的理论缺陷源于缺乏对策略更新幅度的有效约束。首先，多智能体并发学习使得环境的状态转移与奖励反馈不再满足平稳性前提，导致价值网络拟合的目标始终处于动态漂

移中，破坏了学习过程的连贯性。其次，在博弈动力学层面，各参与者仅依据局部梯度进行的贪婪更新极易进入非收敛的循环轨道，使策略在离散状态之间剧烈震荡而无法稳定在最优混合均衡点。最后，由于网络参数与输出概率分布之间存在高度非线性的映射关系，无约束的参数更新会导致策略分布发生不可控的突变，一旦新策略进入低价值区域，将引发采样质量恶化与价值评估失效的恶性循环，最终导致系统性能崩塌。

## 2.4. 目标函数构建及参数更新规则

本文提出在原始目标函数中引入基于策略惯性的正则化项。其核心思想是：在追求高回报的同时，强加一个针对策略空间的惯性约束，使得新策略 $\theta_i^t$ 不会过度偏离上一阶段的稳定策略，因此，我们将优化问题转化为一个正则化最大化问题。定义智能 $i$ 的惯性正则化目标函数 $J_{total}\left(\theta_i\right)$ 如下：

$$J_{total}\left(\theta_i\right)=J\left(\theta_i\right)-\lambda_i\cdot\Omega\left(\theta_i^t,\theta_i^{old}\right) \tag{3}$$

其中：

$\theta_i^t$ 表示当前更新的策略参数。

$\theta_i^{old}$ 表示上一阶段的策略参数锚点。

$\lambda_i\geq 0$ 为惯性系数，用于调节探索收益与策略稳定性之间的权衡。

$\Omega\left(\cdot,\cdot\right)$ 为衡量新旧策略差异的正则化项。

智能体 $i$ 的正则项如下所示：

$$\Omega\left(\theta_i^t,\theta_i^{old}\right)=\frac{1}{2}\left\|\theta_i^t-\theta_i^{old}\right\|_2^2 \tag{4}$$

为了利用基于梯度的优化算法对网络参数 $\theta_i$ 进行更新，我们需要推导总目标函数 $J_{total}\left(\theta_i\right)$ 关于参数 $\theta_i$ 的梯度 $\nabla_{\theta_i}J_{total}$。根据公式(3)，总梯度由两部分组成：

$$\nabla_{\theta_i}J_{total}=\nabla_{\theta_i}J\left(\theta_i\right)-\lambda_i\nabla_{\theta_i}\Omega\left(\theta_i^t,\theta_i^{old}\right) \tag{5}$$

1) 原始策略梯度项推导：第一项 $\nabla_{\theta_i}J\left(\theta_i\right)$ 为标准的确定性或随机性策略梯度。在本研究采用的随机高斯策略设定下，利用对数导数技，其梯度推导过程如下：

对等式两边关于参数 $\theta_i$ 求梯度，即得到最终的策略梯度公式：

$$\begin{aligned}\nabla_{\theta_i}J\left(\theta_i\right)&=\sum_{a_i}\left[\pi_i\left(a_i^t\mid s;\theta_i\right)\cdot\nabla_{\theta_i}\log\pi_i\left(a_i^t\mid s;\theta_i\right)\right]\cdot Q_{\pi_i}\left(s,a_i^t;w_i^t\right)\\&=\sum_{a_i}\pi_i\left(a_i^t\mid s;\theta_i\right)\cdot\nabla_{\theta_i}\log\pi_i\left(a_i^t\mid s;\theta_i\right)\cdot Q_{\pi_i}\left(s,a_i^t;w_i^t\right)\\&=\mathbb{E}_{s,\boldsymbol{a}}\left[\nabla_{\theta_i}\log\pi_i\left(a_i\mid s_i;\theta_i\right)\cdot Q_i\left(s,\boldsymbol{a}\right)\right]\end{aligned} \tag{6}$$

应用蒙特卡洛采样，最终梯度可以表示为以下形式：

$$\begin{aligned}\nabla_{\theta_i}J\left(\theta_i\right)&=\mathbb{E}_{s,\boldsymbol{a}}\left[\nabla_{\theta_i}\log\pi_i\left(a_i\mid s_i;\theta_i\right)\cdot Q_i\left(s,\boldsymbol{a}\right)\right]\\&\approx\frac{1}{B}\sum_{k=1}^{B}\nabla_{\theta_i}\log\pi_i\left(a_i^{(k)}\mid s_i^{(k)};\theta_i\right)\cdot Q_i\left(s^{(k)},\boldsymbol{a}^{(k)}\right)\end{aligned} \tag{7}$$

其中：

$\frac{1}{B}\sum_{k=1}^{B}$ 表示我们在训练的一个批次中采样了 $B$ 条数据，用这 $B$ 条数据的平均值来近似真实的梯度。$k$ 表示第 $k$ 个样本数据。

2) 惯性正则项梯度推导：第二项为惯性约束的梯度，将公式(4)对 $\theta_i$ 求导，根据链式法：

$$\nabla_{\theta_i}\Omega\left(\theta_i^t,\theta_i^{old}\right)=\nabla_{\theta_i}\left(\frac{1}{2}\left\|\theta_i-\theta_i^{old}\right\|_2^2\right)=\theta_i^t-\theta_i^{old} \tag{8}$$

3) 最终更新公式：(7)综合(8)与，智能体 $i$ 的 Actor 网络参数更新规则如下：

$$\theta_i^{t+1}\leftarrow\theta_i^t+\beta_i\cdot\left(\mathbb{E}\left[\nabla\log\pi_i\cdot Q_i\right]-\lambda_i\left(\theta_i^t-\theta_i^{old}\right)\right) \tag{9}$$

公式(9)括号内的部分是总梯度，它指示了参数应该朝着什么方向移动才能优化目标函数。

## 2.5. 算法执行流程

本研究提出的改进算法延续了集中式训练去中心化执行的总体架构，但在策略更新阶段引入了惯性锚点机制。整个算法的执行流程可被形式化地描述为初始化、交互采样、价值评估、策略修正与锚点更新五个核心阶段。

1) 初始化阶段：对于系统中的每一个智能体 $i$，首先初始化其策略网络(Actor)参数 $\theta_i$ 与价值网络(Critic)参数 $w_i$。同时，为了在训练初期建立惯性参考系，将初始时刻的策略参数直接赋值给惯性锚点 $\theta_i^{old}$。此外，需设定 Critic 学习率 $\alpha_i$、Actor 学习率 $\beta_i$、折扣因子 $\gamma$ 以及关键的惯性正则化系数 $\lambda$。

2) 分布式交互与采样：在每一个时间步 $t$，智能体与环境进行交互，生成训练数据：

局部观测：每个智能体 $i$ 获取局部观测状态 $s_i^t$。

动作采样：智能体依据当前策略网络采样动作 $a_i^t$：$a_i^t\sim\pi_i\left(\cdot\mid s_i^t;\theta_i^t\right)$

联合执行：形成联合动作向量 $\boldsymbol{a}^t=\left(a_1^t,\cdots,a_N^t\right)$ 并执行。

环境反馈：环境反馈所有智能体的即时奖励 $r_i^t$ 及下一时刻的联合状态 $s^{t+1}$。

3) 集中式价值评估：Critic 网络负责评估当前联合策略的质量，并计算时序差分(TD)误差以驱动价值函数的收敛。

下一步动作采样：为了计算 TD 目标值，需采样下一时刻的联合动作 $\tilde{\boldsymbol{a}}^{t+1}$。对于每个智能体 $i$：$\tilde{a}_i^{t+1}\sim\pi_i\left(\cdot\mid s_i^{t+1};\theta_i^t\right)$

Q 值计算：当前时刻 Q 值估计：$q_i^t=Q_i\left(s^t,\boldsymbol{a}^t;w_i^t\right)$；下一时刻 Q 值估计：$q_i^{t+1}=Q_i\left(s^{t+1},\tilde{\boldsymbol{a}}^{t+1};w_i^t\right)$；

TD 误差计算($\delta_i^t$)：计算当前 Q 值与目标 Q 值(即时奖励 + 折扣后的未来价值)之间的偏差：$\delta_i^t=q_i^t-\left(r_i^t+\gamma\cdot q_i^{t+1}\right)$

Critic 梯度计算($d_{w,i}^t$)：$d_{w,i}^t=\nabla_{w_i}Q_i\left(\boldsymbol{s}^t,\boldsymbol{a}^t;w_i\right)\Big|_{w_i=w_i^t}$

Critic 参数更新：沿着最小化 TD 误差平方的方向更新参数：$w_i^{t+1}\leftarrow w_i^t-\alpha_i\cdot\delta_i^t\cdot d_{w,i}^t$

4) 正则化策略提升：这是算法的核心步骤。Actor 的更新不仅取决于 Critic 提供的价值梯度，还受到惯性锚点的约束，以抑制策略参数的剧烈震荡。

标准策略梯度计算($d_{\theta,i}^t$)：计算最大化动作对数概率的梯度方向：$d_{\theta,i}^t=\nabla_{\theta_i}\log\pi_i\left(a_i^t\mid s_i^t;\theta_i\right)\Big|_{\theta_i=\theta_i^t}$

惯性正则化梯度计算($g_{reg}^t$)：计算当前参数与旧参数(锚点)之间欧氏距离的梯度，该梯度充当参数层面的阻尼力：

$$g_{reg}^t=\nabla_{\theta_i}\left(\frac{1}{2}\lambda\left\|\theta_i^t-\theta_i^{old}\right\|^2\right)=\lambda\left(\theta_i^t-\theta_i^{old}\right)$$

Actor 参数更新：合成总梯度方向进行更新。其中，$q_i^t\cdot d_{\theta,i}^t$ 推动策略向高回报方向移动，而 $-g_{reg}^t$ 将策略拉向旧参数锚点：$\theta_i^{t+1}\leftarrow\theta_i^t+\beta_i\cdot\left(q_i^t\cdot d_{\theta,i}^t-g_{reg}^t\right)$

5) 惯性锚点更新(Anchor Update)在完成单步参数更新后，更新惯性锚点，使其跟随策略的演化，形成滑动窗口式的约束：$\theta_i^{old}\leftarrow\theta_i^t$

上述流程在每一个训练回合中循环执行，直至算法收敛或达到预设的最大迭代次数。

## 2.6. 基于惯性正则化的局部渐近稳定性分析

在多智能体强化学习中，纳什均衡点的收敛稳定性是衡量算法性能的关键指标。特别是在零和博弈场景下，由于博弈动力学的固有旋转特性，传统的基于梯度的优化方法往往难以收敛，而是表现为围绕均衡点的持续震荡。本节将从连续时间动力系的视角，建立博弈动力学的微分方程模型，并严格证明引入策略惯性正则化后，系统雅可比矩阵的谱分布将发生平移，从而保证了算法在纳什均衡点附近的局部渐近稳定性。

综合公式(3)和公式(4)可知，在二人零和博弈中，对于第 $i$ 个智能体，有：

$$J_i^{total} = J_i(\boldsymbol{\theta}) - \frac{1}{2}\cdot\lambda\left\|\theta_i - \theta_i^{old}\right\|^2$$

其中，$J_i^{total}$ 为改进后的第 $i$ 个智能体的惯性正则化联合目标函数，又因为原始目标函数同时依赖于智能体 $i$ 以及对手的策略，所以其原始目标函数中含有两个智能体的策略参数，其中，$\boldsymbol{\theta}$ 表联合策略参数，即所有智能体参数的集合，$\boldsymbol{\theta} = [\theta_1, \theta_2]^\top$。为方便表示，我们将两个智能体的目标函数用联合目标函数形式表示，即 $J(\boldsymbol{\theta}) = \begin{bmatrix} J_1(\theta) \\ J_2(\theta) \end{bmatrix}$。又因为现在主要研究零和博弈，则有 $J_1(\boldsymbol{\theta}) = -J_2(\boldsymbol{\theta})$。下面将进行联合动力方程的推导。

1) 原始部分的梯度：

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = v(\boldsymbol{\theta}) = \begin{bmatrix} \nabla_{\theta_1} J_1(\theta) \\ \nabla_{\theta_2} J_2(\theta) \end{bmatrix} \tag{10}$$

2) 正则化项部分的梯度：$\nabla_{\boldsymbol{\theta}}\left(-\frac{\lambda}{2}\left\|\boldsymbol{\theta} - \boldsymbol{\theta}^{old}\right\|^2\right) = -\frac{\lambda}{2}\cdot 2\left(\boldsymbol{\theta} - \boldsymbol{\theta}^{old}\right) = -\lambda\left(\boldsymbol{\theta} - \boldsymbol{\theta}^{old}\right)$

合并后的总梯度方向：

$$\boldsymbol{g}_{total} = v(\boldsymbol{\theta}) - \lambda\left(\boldsymbol{\theta} - \boldsymbol{\theta}^{old}\right) \tag{11}$$

按照梯度上升规则，参数更新公式为($\eta$ 为学习率/步长)：$\boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^k + \eta\cdot\left(\boldsymbol{v}\left(\boldsymbol{\theta}^k\right) - \lambda\left(\boldsymbol{\theta}^k - \boldsymbol{\theta}^{old}\right)\right)$

进行以下步骤的变换，则有：

$$\frac{\boldsymbol{\theta}^{k+1} - \boldsymbol{\theta}^k}{\eta} = v\left(\boldsymbol{\theta}^k\right) - \lambda\left(\boldsymbol{\theta}^k - \boldsymbol{\theta}^{old}\right) \tag{12}$$

$$\lim_{\eta\to 0}\frac{\boldsymbol{\theta}^{k+1} - \boldsymbol{\theta}^k}{\eta} = \frac{\mathrm{d}\boldsymbol{\theta}}{\mathrm{d}t} = \dot{\boldsymbol{\theta}} \tag{13}$$

综合公式(12)和公式(13)，则有：

$$\dot{\boldsymbol{\theta}} = v(\boldsymbol{\theta}) - \lambda\left(\boldsymbol{\theta} - \boldsymbol{\theta}^{old}\right) \tag{14}$$

因为我们的目的是证明算法在纳什均衡点 $\boldsymbol{\theta}^*$ 附近的局部行为。我们假设系统已经运行到了 $\boldsymbol{\theta}^*$ 的很小的邻域内。所以在公式(14)中我们使用 $\boldsymbol{\theta}^*$ 代替 $\boldsymbol{\theta}^{old}$。则最终的联合动力方程如下所示：

$$\dot{\boldsymbol{\theta}} = v(\boldsymbol{\theta}) - \lambda\left(\boldsymbol{\theta} - \boldsymbol{\theta}^*\right) \tag{15}$$

大量研究表明，深度神经网络的损失曲面在解附近呈现出显著的平坦性特征。Hochreiter & Schmidhuber (1997)最早提出了“平坦极小值”(Flat Minima)概念[16]，指出泛化能力强的解往往位于海森矩阵特征值极小的平坦区域。随后，Goodfellow et al. (2014)通过线性插值实验进一步证实[17]，尽管深层网络具

有高度非线性，其优化路径和局部景观却表现出惊人的平滑性。所以我们做出以下假设。

假设 2.1 设$\boldsymbol{\theta}^*$为原始博弈系统的微分纳什均衡点，$\boldsymbol{\theta}$是其邻域内的任意一点，在此区域内，有智能体 1 与智能体 2 的原始目标函数的二阶导近似为 0，即$\nabla^2_{\theta_1\theta_1}J_1(\boldsymbol{\theta})=0$，$\nabla^2_{\theta_2\theta_2}J_2(\boldsymbol{\theta})=-\nabla^2_{\theta_2\theta_2}J_1(\boldsymbol{\theta})=0$。

定理 2.1 惯性正则化系统的渐近稳定性：对于修正后的连续时间动力系统：$\dot{\boldsymbol{\theta}}=\boldsymbol{v}(\boldsymbol{\theta})-\lambda(\boldsymbol{\theta}-\boldsymbol{\theta}^*)$，若假设 2.1 成立，且惯性系数$\lambda>0$，则该系统在均衡点$\boldsymbol{\theta}^*$处是局部渐近稳定的。

证明：首先在$\boldsymbol{\theta}^*$的邻域内对公式(10)再一次求导，我们可以得到原始目标函数的雅可比矩阵如下所示：

$$\boldsymbol{J}=\nabla\boldsymbol{v}=\begin{bmatrix}\dfrac{\partial\left(\nabla_{\theta_1}J_1(\boldsymbol{\theta})\right)}{\partial\theta_1} & \dfrac{\partial\left(\nabla_{\theta_1}J_1(\boldsymbol{\theta})\right)}{\partial\theta_2}\\ \dfrac{\partial\left(-\nabla_{\theta_2}J_1(\boldsymbol{\theta})\right)}{\partial\theta_1} & \dfrac{\partial\left(-\nabla_{\theta_2}J_1(\boldsymbol{\theta})\right)}{\partial\theta_2}\end{bmatrix}=\begin{bmatrix}\nabla^2_{\theta_1\theta_1}J_1(\boldsymbol{\theta}) & \nabla^2_{\theta_1\theta_2}J_1(\boldsymbol{\theta})\\ -\nabla^2_{\theta_2\theta_1}J_1(\boldsymbol{\theta}) & -\nabla^2_{\theta_2\theta_2}J_1(\boldsymbol{\theta})\end{bmatrix}$$

由于假设 2.1，则有

$$\boldsymbol{J}=\begin{bmatrix}0 & \nabla^2_{\theta_1\theta_2}J_1(\boldsymbol{\theta})\\ -\nabla^2_{\theta_2\theta_1}J_1(\boldsymbol{\theta}) & 0\end{bmatrix}$$

显然，上述矩阵为一个反对称矩阵($\boldsymbol{J}^{\mathrm{T}}=-\boldsymbol{J}$)

设$\mu$是矩阵$\boldsymbol{J}$的特征值，$\boldsymbol{x}$是对应的非零特征向量

$$Jx=\mu x \tag{16}$$

对上式两边取共轭转置：

$$\boldsymbol{x}^H\boldsymbol{J}^{\mathrm{T}}=\bar{\mu}\boldsymbol{x}^H$$

其中，$\bar{\mu}$是$\mu$的复共轭；$\boldsymbol{x}^H$为非零特征向量$\boldsymbol{x}$的共轭转置。

利用反对称性质$\boldsymbol{J}^{\mathrm{T}}=-\boldsymbol{J}$：

$$\boldsymbol{x}^H(-\boldsymbol{J})=\bar{\mu}\boldsymbol{x}^H\Rightarrow-\boldsymbol{x}^HJ=\bar{\mu}\boldsymbol{x}^H \tag{17}$$

将公式(17)两边乘以$\boldsymbol{x}$：

$$-\boldsymbol{x}^H(\boldsymbol{J}\boldsymbol{x})=\bar{\mu}\boldsymbol{x}^H\boldsymbol{x}$$

将公式(16)代入左边的括号：

$$-\boldsymbol{x}^H(\mu\boldsymbol{x})=\bar{\mu}\left(\boldsymbol{x}^H\boldsymbol{x}\right)$$

提取标量后则有：$-\mu\left(\boldsymbol{x}^H\boldsymbol{x}\right)=\bar{\mu}\left(\boldsymbol{x}^H\boldsymbol{x}\right)$

又因为$\|\boldsymbol{x}\|^2=\boldsymbol{x}^H\boldsymbol{x}$是一个正实数，约掉后得到：$-\mu=\bar{\mu}$

设$\mu=a+bi$，($a$，$b$为实数)则有：$-(a+bi)=a-bi$，$-a-bi=a-bi\Rightarrow a=0$

由此证得原始目标函数的雅可比矩阵的特征值$\mu$的实部为 0。

在$\boldsymbol{\theta}^*$的邻域内对公式(15)再一次求导，我们可以得到改进后的目标函数的雅可比矩阵如下所示：

$$\frac{\partial}{\partial\boldsymbol{\theta}}\left(\boldsymbol{v}(\boldsymbol{\theta})-\lambda(\boldsymbol{\theta}-\boldsymbol{\theta}^*)\right)=\frac{\partial\boldsymbol{v}(\boldsymbol{\theta})}{\partial\boldsymbol{\theta}}-\frac{\partial}{\partial\boldsymbol{\theta}}(\lambda\theta)+\frac{\partial}{\partial\boldsymbol{\theta}}\left(\lambda\boldsymbol{\theta}^*\right)=\boldsymbol{J}-\lambda\boldsymbol{I}$$

由公式(16)则有；

$$(\boldsymbol{J}-\lambda\boldsymbol{I})\boldsymbol{x}=\boldsymbol{J}\boldsymbol{x}-\lambda\boldsymbol{I}\boldsymbol{x}=\mu\boldsymbol{x}-\lambda\boldsymbol{x}=(\mu-\lambda)\boldsymbol{x}$$

即修正后的特征值$\mu'$满足$\mu'=\mu-\lambda$。

考察修正后特征值 $\mu'$ 的实部：

$$\mathrm{Re}(\mu') = \mathrm{Re}(\mu - \lambda) = \mathrm{Re}(\mu) - \lambda$$

又因为原始目标函数的雅可比矩阵的特征值 $\mu$ 的实部为 0，则：

$$\mathrm{Re}(\mu') = 0 - \lambda = -\lambda$$

由于惯性系数 $\lambda$ 被设定为严格正实数($\lambda > 0$)，故对于所有特征值 $\mu'_k\ (k = 1, \cdots, d)$，均有：$\mathrm{Re}(\mu'_k) < 0$。根据 yapunov 第一方法，引入惯性正则化后的动力系统在纳什均衡点 $\boldsymbol{\theta}^*$ 处是局部渐近稳定的。

证毕。

## 2.7. 算法收敛性数值验证

为了验证基于惯性正则化的 Actor-Critic 算法在高随机性环境和离散动作空间下的鲁棒性与收敛能力，本节构建了一个名为“湿滑网格世界”的零和博弈场景。该场景模拟了现实世界中执行器误差或环境扰动带来的不确定性，旨在测试算法是否能在高方差梯度下坚持学习最优策略，而非陷入局部极小值。

1) 场景描述：在一个 $N \times N$ 的离散网格中，存在两个智能体：追捕者与逃逸者。

追捕者：目标是在最短的时间内捕获逃逸者(即坐标重合)。

逃逸者：目标是尽可能延长被捕获的时间，或保持与追捕者的距离。

环境特性：地面具有“湿滑”特性，智能体的移动指令无法被 100%精确执行，存在随机滑动的风险。例如，当智能体选择“向上”时，有 80%的概率向上移动，但有 10%的概率向左滑，10%的概率向右滑。若目标位置超出边界，则智能体保持原地不动(撞墙)。

2) 马尔可夫博弈五元组 $(\mathcal{S}, A, P, R, \gamma)$ 定义

状态空间 $\mathcal{S}$：联合状态由两个智能体的二维坐标组成：$\boldsymbol{s}^t = [x_p, y_p, x_e, y_e]$，其中 $\boldsymbol{s}_p^t = [x_p^t, y_p^t]$，$\boldsymbol{s}_e^t = [x_e^t, y_e^t]$ 分别代表追捕者和逃逸者两个智能体的状态，即二者在离散网格中的坐标，$x, y \in \{0, 1, \cdots, N-1\}$。在本实验中，网格大小设为 $N = 6$。

动作空间 $A$：双方均为离散动作空间，包含四个基本方向：$A_i = \{\mathrm{Up}, \mathrm{Down}, \mathrm{Left}, \mathrm{Right}\}$。

状态转移概率 $\mathcal{P}$：对于追捕者和逃逸者中某个智能体，当其选择动作 $a$ 时，实际状态 $\boldsymbol{s}'$ 服从以下分布：

$$P(s' \mid s, a) = \begin{cases} 0.8, & \text{智能体移动到目标位置} \\ 0.1, & \text{智能体移动到与目标方向垂直的左侧位置} \\ 0.1, & \text{智能体移动到与目标方向垂直的右侧位置} \end{cases}$$

奖励函数 $R$ 设计：已知 $\boldsymbol{s}_p^t = [x_p, y_C]$，$\boldsymbol{s}_e^t = [x_e, y_e]$ 分别代表追捕者和逃逸者两个智能体的状态，即 $t$ 时刻两个智能体在网格中的坐标，我们采用曼哈顿距离作为距离度量，以适应离散网格环境的移动特性。对于追捕者，每一时刻 $t$ 的即时奖励 $r_C^t$ 定义如下：

$$r_p^t = \begin{cases} +10, & \text{如果追捕者在} t \text{时刻抓捕到逃逸者} \\ -\frac{1}{2} \cdot \left( \left| x_p^t - x_e^t \right| + \left| y_p^t - y_e^t \right| \right), & \text{其他情况} \end{cases}$$

为了构建严格的对抗环境，逃逸者的奖励函数 $r_M^t$ 被定义为追捕者的相反数：$r_p^t = -r_e^t$，这意味着追捕者的收益严格等于逃逸者的损失。

3) 实验设置

学习率设置为 0.002；折扣因子 $\gamma$ 设置为 0.95；惯性正则化系数 $\lambda$ 设置为 5.0；训练回合数设置为 1000；

回合最大步数设置为 60。

## 2.8. 实验结果与分析

图 1 展示了训练过程中每回合总奖励的变化曲线，该指标综合反映了算法的收敛效能。原始算法的奖励曲线在训练初期迅速下降，并长期停滞在−250 左右的理论下界。这种现象表明智能体陷入了严重的策略退化。由于环境存在 20%的随机转移概率，早期的探索行为常因环境噪声而遭受距离惩罚。在缺乏参数更新约束的情况下，原始梯度算法表现出过度的风险规避倾向，智能体倾向于采取原地不动或撞墙等保守策略以避免即时惩罚，最终因超时而获得最低累积奖励；相比之下，基于惯性正则化的 Actor-Critic 算法的奖励曲线在经历约 100 个回合的探索后显著回升，并稳定收敛于−25 左右。这一数值显著优于基准算法，且接近理论上的平均捕获成本。实验结果证明，惯性正则项成功抑制了由单次随机滑动引起的高频梯度抖动，使得智能体能够基于长期的期望收益坚持执行最优策略，而非受限于短期的随机负反馈。

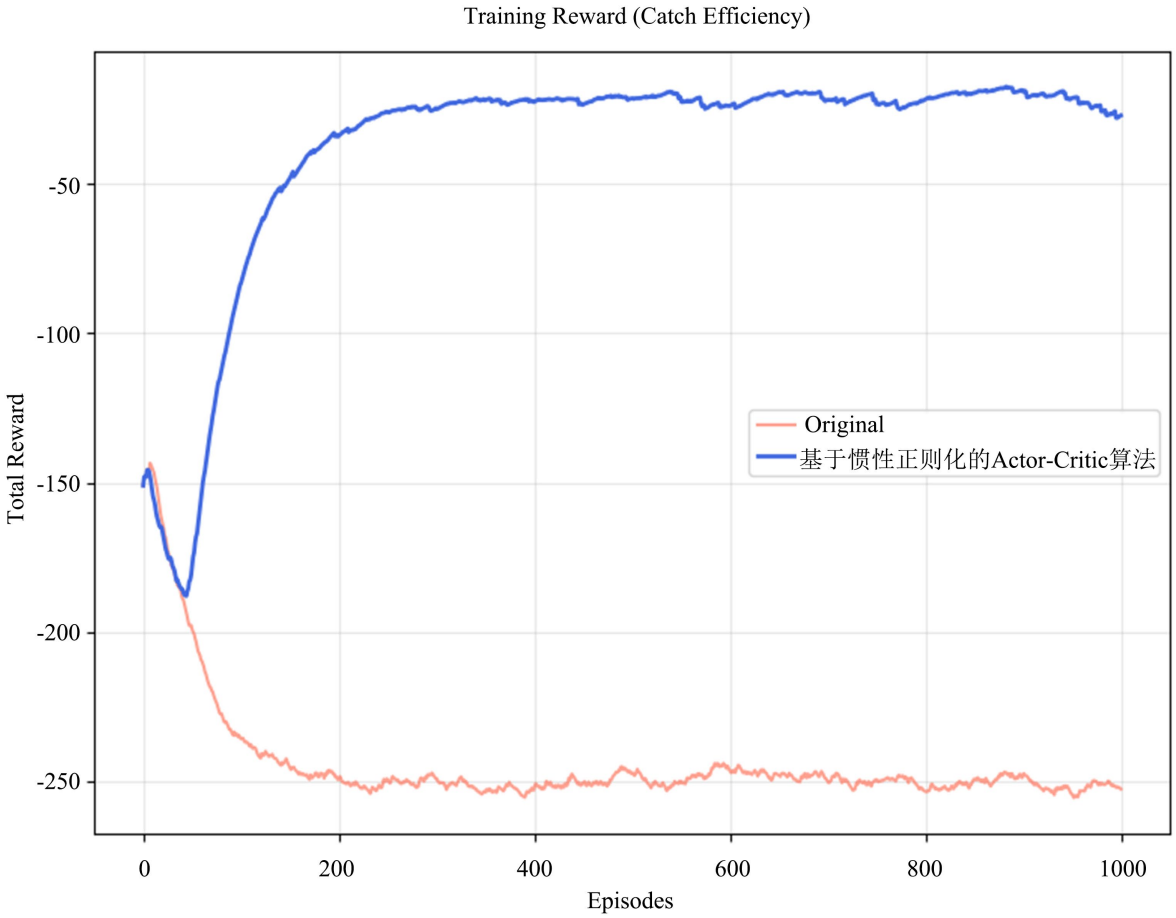


**Figure 1.** Comparison of convergence performance of average cumulative reward per round during the training phase
**图 1.** 训练阶段平均回合累积奖励的收敛性能对比

如图 2 所示，原始算法的成功率长期维持在 0%，证实了其低奖励并非源于路径规划的次优，而是彻底的任务失败，智能体完全丧失了在规定时间内捕获目标的能力。相反，基于惯性正则化的 Actor-Critic 算法的成功率呈现显著上升趋势，最终稳定在 95%至 98%之间。尽管环境具有强随机性，该算法依然实现了极高的任务可靠性，证明其习得的策略具有极强的闭环纠错能力。

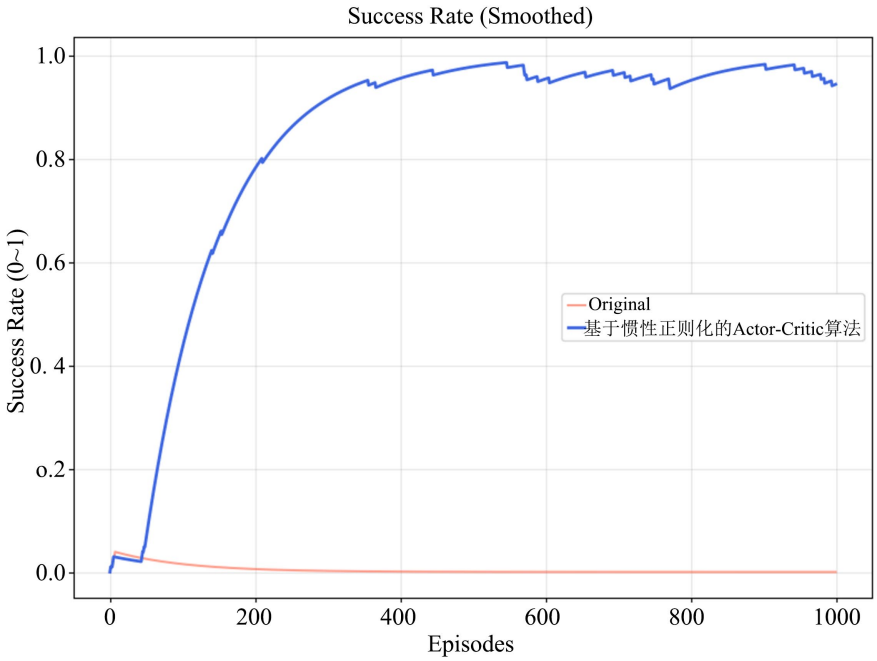


**Figure 2.** Evolution curve of task capture success rate during training
**图 2.** 训练过程中的任务捕获成功率演变曲线

如图 3 所示，原始算法的平均步数始终重合于 60 步的最大限制线，进一步印证了其因无法完成任务而耗尽时间窗口。基于惯性正则化的 Actor-Critic 算法的捕获步数则从初始值迅速下降，最终收敛至 15 至 18 步区间。考虑到网格环境的曼哈顿距离为 10 且存在滑倒干扰，该数值已经逼近随机最短路径的理论下界，表明该算法实现了时间维度上的最优性。

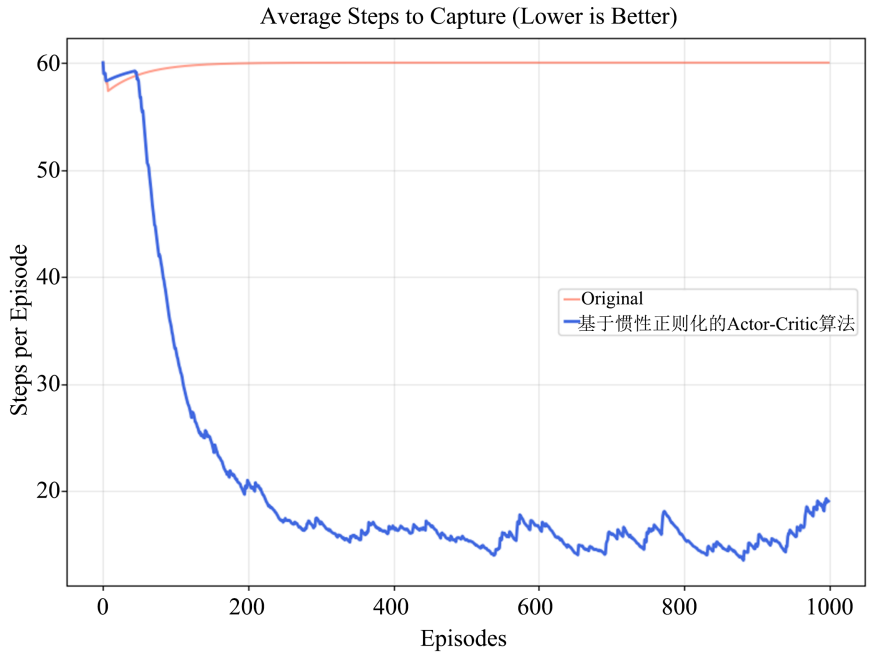


**Figure 3.** Analysis of average time steps and execution efficiency of successful capture rounds
**图 3.** 成功捕获回合的平均时间步数与执行效率分析

图 4 表明在原始算法生成的轨迹中，追捕者智能体的策略表现出显著的局部最优收敛特征。追捕者的状态转移序列主要约束在二维网格的下边界区域，即纵坐标 $y=0$ 的子空间内。追捕者从初始状态(0, 0)出发后，其策略网络 $\pi(a|s)$ 输出的动作概率分布高度偏向于水平位移，导致轨迹仅在 x 轴方向上产生微小的增量，随后陷入停滞。这种现象表明原始算法在处理稀疏奖励或延迟奖励时，价值函数 $V(s)$ 的更新未能有效传播至全局状态空间。智能体仅通过减小 x 轴方向的曼哈顿距离分量来获取局部奖励信号，而未能探索到能够显著降低总势能的纵向移动策略。逃逸者智能体位于状态空间边缘(5, 5)附近，由于追捕者未能构建有效的逼近策略，逃逸者仅需维持在局部区域即可保持较大的状态间距，双方未能形成高水平的对抗博弈，系统陷入次优的稳定平衡态。(横坐标(X 轴)：代表智能体在离散网格中的水平位置坐标，取值范围为[0, 5]。纵坐标(Y 轴)：代表智能体在离散网格中的垂直位置坐标，取值范围为[0, 5]。)

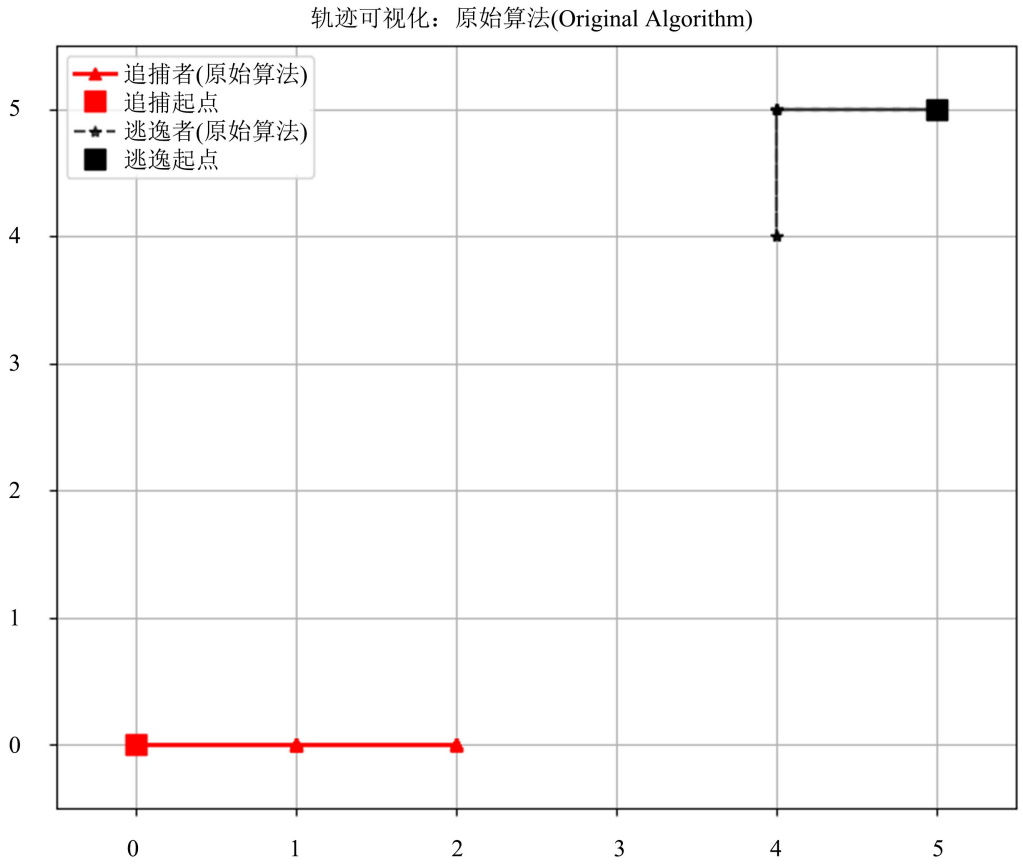


**Figure 4.** Pursuit and escape adversarial trajectory based on the original Actor-Critic algorithm
**图 4.** 基于原始 Actor-Critic 算法的追捕与逃逸对抗轨迹

图 5 显著体现了惯性正则化机制对策略空间探索能力的提升。追捕者智能体的状态转移序列展示出一种分层递进的全局规划策略，其并未受限于初始状态附近的局部梯度，而是优先执行了沿纵轴正方向的高阶位移操作。这种行为模式表明智能体的策略网络已经成功构建了对整个二维状态网格的价值评估映射，能够识别出通过占据网格中心或边界关键点来压缩逃逸者机动空间的战术优势。在追捕者逼近的压力下，逃逸者智能体表现出基于当前状态观测的随机规避行为，试图沿上边界区域进行反向机动以最大化曼哈顿距离。然而，由于追捕者采用了具备前瞻性的拦截路径，逃逸者的可行状态空间被迅速收敛。最终，双方的状态向量在坐标(3, 5)处发生重合，系统的状态距离范数归零。这一收敛过程证明了该方法在解决稀疏

奖励下的长视距规划问题上具有显著的数学优越性。(横坐标(X 轴)：代表智能体在离散网格中的水平位置坐标，取值范围为[0, 5]。纵坐标(Y 轴)：代表智能体在离散网格中的垂直位置坐标，取值范围为[0, 5]。)

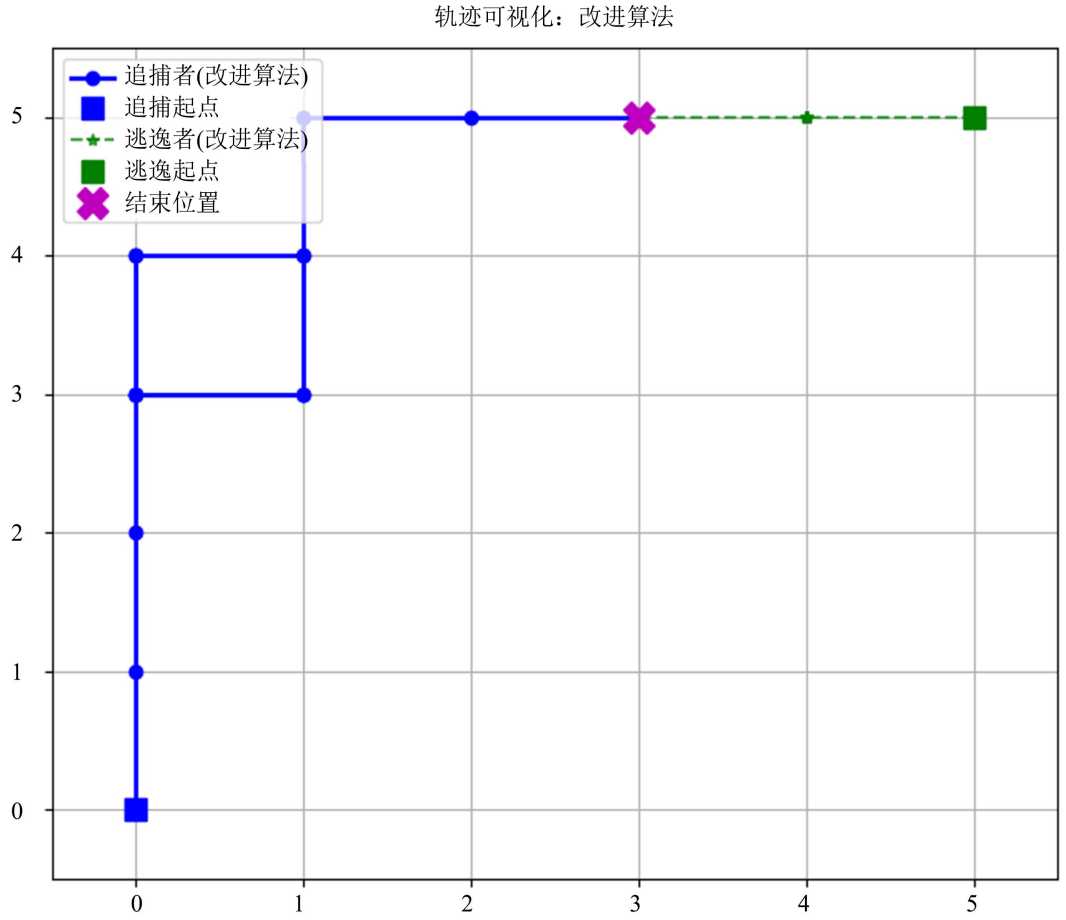


**Figure 5.** Pursuit and escape adversarial trajectories based on the inertial regularization Actor-Critic algorithm
**图 5.** 基于惯性正则化的 Actor-Critic 算法的追捕与逃逸对抗轨迹

## 2.9. 总结

针对原始 Actor-Critic 架构在处理高阶动力学系统与强随机环境时存在的策略震荡与收敛困难问题，本章提出了一种基于惯性的策略正则化改进方案。算法在 Actor 网络的损失函数中引入了惯性惩罚项 $\Omega(\theta)=\frac{\lambda}{2}\|\theta_t-\theta_{old}\|^2$，这限制了参数在单次迭代中的剧烈突变。它确保了策略的演化必须遵循连续性原则，防止了智能体因单次采样偏差或瞬时环境反馈而产生过激的策略调整。

在具有 20%滑动概率的离散网格博弈中，原始算法因无法抵抗单次负反馈样本的干扰，发生了严重的策略退化(如原地停滞)，导致任务成功率为 0%。基于惯性正则化的 Actor-Critic 算法则展现了极强的抗噪能力，保持了策略在宏观方向上的一致性。实验结果表明，改进算法不仅将捕获成功率提升至 98%，且习得了抵抗环境扰动的阶梯状最优路径，证明了其在非确定性环境下的鲁棒性。

## 参考文献

[1] Sutton, R.S. and Barto, A.G. (1998) Reinforcement Learning: An Introduction. Vol. 1, No. 1, MIT Press.

[2] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., *et al*. (2015) Human-Level Control through Deep Reinforcement Learning. *Nature*, **518**, 529-533. https://doi.org/10.1038/nature14236

[3] Shapley, L.S. (1953) Stochastic Games. *Proceedings of the National Academy of Sciences*, **39**, 1095-1100. https://doi.org/10.1073/pnas.39.10.1953

[4] Lowe, R., *et al*. (2017) Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. *NIPS*'17: *Proceedings of the* 31*st International Conference on Neural Information Processing Systems*, Long Beach, 4-9 December 2017, 6382-6393.

[5] Hernandez-Leal, P., *et al*. (2017) A Survey of Learning in Multiagent Environments: Dealing with Non-Stationarity.

[6] Tan, M. (1993) Multi-Agent Reinforcement Learning: Independent vs. Cooperative Agents. *Proceedings of the* 10*th International Conference*, Amherst, 27-29 June 1993, 330-337. https://doi.org/10.1016/b978-1-55860-307-3.50049-6

[7] Rashid, T., *et al*. (2020) Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. *Journal of Machine Learning Research*, **21**, 1-51.

[8] Singh, S., Kearns, M.J. and Mansour, Y. (2000) Nash Convergence of Gradient Dynamics in General-Sum Games. UAI.

[9] Foerster, J.N., *et al*. (2017) Learning with Opponent-Learning Awareness.

[10] Letcher, A., *et al*. (2018) Stable Opponent Shaping in Differentiable Games.

[11] Schulman, J., *et al*. (2015) Trust Region Policy Optimization. *ICML*'15: *Proceedings of the* 32*nd International Conference on Machine Learning*, Volume 37, 1889-1897.

[12] Haarnoja, T., Zhou, A., Abbeel, P. and Levine, S. (2018) Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. *Proceedings of the* 35*th International Conference on Machine Learning*, Stockholm, 10-15 July 2018, 1861-1870.

[13] Czarnecki, W.M., *et al*. (2020) Real World Games Look like Spinning Tops. *NIPS*'20: *Proceedings of the* 34*th International Conference on Neural Information Processing Systems*, Vancouver, 6-12 December 2020, 17443-17454.

[14] Littman, M.L. (1994) Markov Games as a Framework for Multi-Agent Reinforcement Learning. *Proceedings of the* 11*th International Conference*, *Rutgers University*, New Brunswick, 10-13 July 1994, 157-163. https://doi.org/10.1016/b978-1-55860-335-6.50027-1

[15] Ziebart, B.D., *et al*. (2008) Maximum Entropy Inverse Reinforcement Learning. *AAAI*, Volume 8, 1433-1438.

[16] Hochreiter, S. and Schmidhuber, J. (1997) Flat Minima. *Neural Computation*, **9**, 1-42. https://doi.org/10.1162/neco.1997.9.1.1

[17] Goodfellow, I.J., Vinyals, O. and Saxe, A.M. (2014) Qualitatively Characterizing Neural Network Optimization Problems.