华南理工大学

South China University of Technology

硕士学位论文

基于强化学习的数据驱动最优迭代学习控制方法

作　者　姓　名　黎金泽

学　科　专　业　控制科学与工程

指　导　教　师　田森平　教授

所　在　学　院　自动化科学与工程学院

论文提交日期　2024 年 4 月

Data-driven Optimal Iterative Learning Control Method Based on Reinforcement Learning

A Dissertation Submitted for the Degree of Master

Candidate：Li Jinze

Supervisor：Prof. Tian Senping

South China University of Technology

Guangzhou, China

分类号：TP273 **学校代号：10561**

学　号：202120117695

华南理工大学硕士学位论文

基于强化学习的数据驱动最优迭代学习控制方法

作者姓名：黎金泽　　指导教师姓名、职称：田森平　教授

申请学位级别：工学硕士　　学科专业名称：控制科学与工程

研究方向：基于强化学习的迭代学习控制

论文提交日期：2024 年 5 月 28 日　　论文答辩日期：2024 年　6　月　3　日

学位授予单位：华南理工大学　　学位授予日期：　　年　　月　　日

答辩委员会成员：

主席：　仲兆峰

委员：　彭云建、田森平、崔巍、刘伟东

摘　要

迭代学习控制适用于具有重复性动态特性的被控对象，其目标是设计相应的迭代学习控制算法，使得被控系统在整个有限时间内实现对期望轨迹的完全跟踪。迭代学习控制的最优问题可以通过直接或间接近似求解代数 Riccati 方程（Algebraic Riccati Equation, ARE）得到解决，但是这类方法要求系统动态信息完全已知，在实际应用中具有很大的局限性，也与迭代学习控制减少对系统模型参数依赖的初衷产生出入。针对实际应用中无法得到系统精确模型的问题，强化学习提供了仅利用系统输入输出信息实现对迭代学习控制最优问题求解的理论可能。因此，本文以基于模型的最优迭代学习控制算法为研究背景，利用强化学习算法进行改进，以设计一系列减少对模型依赖的数据驱动最优迭代学习控制方案为研究主线，旨在无需预先掌握被控对象的动力学模型参数信息情况下确保系统仍然具备良好的收敛性能，丰富数据驱动迭代学习控制理论的多样性。本文的研究工作贡献如下：

(1) 针对预测最优迭代学习律设计依赖于系统模型参数信息的问题，提出了未知系统动态下的基于 off-policy Q-学习的预测最优迭代学习控制算法，避免了对被控系统动态信息的依赖。随后推导了算法的实现形式并分析了收敛性。最后设计了基于模型和不依赖模型参数信息的预测最优迭代学习控制方法的数值仿真验证算法的有效性。

(2) 针对 off-policy Q-学习算法数据量不足时受限的问题，设计了未知模型参数的基于 on-policy 策略梯度在线学习的参数最优迭代学习控制算法，避免了 Q-学习大量的数据迭代，提高了算法的学习速度。随后利用近端策略优化方法分别设计了算法的行动器和评判器，进一步提升算法的稳定性和收敛速度，并分析算法满足收敛性的条件。最后设计了基于模型的和不依赖模型参数信息的参数最优迭代学习控制方法的数值仿真验证算法的有效性。

(3) 针对 on-policy 策略梯度在线学习动作选择受限和样本效率低泛化能力差的问题，探讨了不依赖系统模型参数的基于 off-policy 策略梯度离线学习的参数最优迭代学习控制算法。通过目标策略与行为策略进行分离，避免训练过程中目标策略的突变对学习过程的影响，提高样本利用率。接着分别设计了算法的“行动器”和“评判器”，给出了算法的实现流程和收敛性条件。最后比较基于模型的和无需预知模型参数信息的参数最优迭代学习控制方法的数值仿真验证算法的有效性。

关键词：迭代学习控制；Q-学习；策略梯度；数据驱动

Abstract

Iterative learning control is applicable to controlled objects with repetitive dynamic characteristics, and the target is to develop corresponding iterative learning control algorithms so as to allow the controlled system to completely track the desired trajectory during the entire finite time. The optimal problem of iterative learning control could be formulated by directly or indirectly approximating the solution of the Algebraic Riccati Equation (ARE), which requires completely known information about the dynamics of the system, having serious restrictions in practical applications, as well as deviating from the initial intention of iterative learning control of diminishing reliance on the parameters of the system model.The reinforcement learning theory gives the possibility of solving the optimal problem for iterative learning control utilising only the information of the system inputs and outputs, when an accuracy model of the system could not be available for the practical application.Therefore, this thesis focuses on the traditional model-based optimal iterative learning control algorithm as the research background, the reinforcement learning algorithm as the improvement method, and the design of a new series of iterative learning control schemes with less reliance on the model as the mainstream of research, aiming to achieve a good convergence performance without pre-knowledge of the dynamics of the controlled object model parameter information, in order to enhance the diversification of the data-driven iterative learning control theory.The contributions of this thesis on research effort are shown as below：

(1) Addressing the problem that the design of the predictive optimal iterative learning law depends on the information of the system model parameters, a data-driven predictive optimal iterative learning control algorithm based on off-policy Q-learning under unknown system dynamics is proposed, avoiding the dependence on the dynamic information of the controlled system. Subsequently, the implementation form of the algorithm is derived as well as the convergence is analysed. Finally, a numerical simulation of the predictive optimal iterative learning control method based on model-free and model-based is designed to verify the effectiveness of the algorithm.

(2) Targeting the problem of limitation of off-policy Q-learning algorithm for lack of data, a parameter optimal iterative learning control algorithm based on on-policy policy gradient online

learning with unknown model parameters is designed, which can improve the learning speed of the algorithm by avoiding a large amount of data iterations in Q-learning. Subsequently, the algorithm under the framework of "Actor-Critic" is given by using the value function approximation baseline and the proximal policy optimisation method, improving the stability of the algorithm and the convergence speed, as well as to analyse the conditions for the algorithm to meet the convergence properties. Finally, a numerical simulations of model-based and model-free parameter optimal iterative learning control methods are presented to verify the effectiveness of the algorithms.

(3) Responding to the problem of limited selection of actions for on-policy policy gradient online learning and poor generalisation ability with low sample efficiency, a parameter-optimal iterative learning control algorithm based on off-policy policy gradient offline learning without any dependence on the parameters of the system model has been investigated. Through separating the ideal policy of optimisation to be performed from the behavioural policy of generating learning data, it could avoid the impact of sudden variations of the ideal policy during the training process on the learning process and improve the sample utilisation rate. Further, the actor and critic of the algorithm are designed, its implementation process and convergence conditions are shown. Finally, the effectiveness of the algorithm is verified by comparing the numerical simulation of the model-based and model-free parameter optimal iterative learning control methods.

Keywords: Iterative learning control; Q-learning; Policy gradient; Data-driven

目 录

插图目录

第一章　绪论

1.1　研究背景和意义

1.1.1　迭代学习控制

自本世纪初中国探月工程立项至今，历经二十年不忘初心砥砺奋进，已圆满完成“绕、落、回”的三步走战略目标。如今，依托中国日趋成熟的航天技术，嫦娥六号工程也在稳步推进，而太空机器人技术以其独特的优势在深空探索中扮演越来越重要的角色。在实际应用中，由于机器人的机械臂与其驱动电机间的连接并非绝对刚性，关节间存在一定的柔韧性，在太空环境下工作的柔性空间机器人会面临诸如燃料消耗不均、载荷难以精确预测以及由柔性引起的运动轨迹追踪误差等诸多不确定性挑战。为有效应对并解决这类控制上的收敛难题，往往可以引入了迭代学习算法来进行优化处理[1]。

迭代学习控制首次提出是日本学者 Uchiyama[2] 在 1978 年研究机械臂控制问题时提出的，然后在 1984 年 Arimoto[3] 等人在国际学术界进一步推广了迭代学习控制理论。迭代学习控制理论与经典控制理论相比更具适应性的特点而备受重视。它能够根据环境的变化进行自适应调整，适用于多样化的环境。与此同时，它对于非线性系统也展现出了强大的适用性，无需事先进行线性化处理便可有效控制。此外，迭代学习控制系统还具备自我调整的能力，能够通过自我监督和反馈不断调整自身行为，以提高控制效果。因此，迭代学习控制理论自其提出以来在接下来的几十年里取得了长足的发展。

迭代学习控制具有如下几个特点。首先迭代学习控制适用于重复性运动的系统，是一种基于获取过去的控制输入和跟踪误差数据中额外的信息经验的试错学习，使的控制器在控制被控对象过程中能不断地优化自己，以得到越来越好的控制效果。其次迭代学习控制的设计初衷[4] 是解决被控系统由于存在未建模的动力学或在实际系统运行中表现出的参数不确定性从而导致被控系统无法尽可能跟踪期望轨迹的问题。最后迭代学习控制器设计简单方便，结构简单易于实现，广泛应用于频繁执行重复工作的工业机械臂[5-7]、直线电机[8]、数控机床[9-11]、半导体晶片生产[12-14] 和车辆编队路径跟踪[15-16] 等领域。

1.1.2　强化学习

强化学习作为一种重要的机器学习方法，与深度学习不同，它更着重于智能体通过与环境的交互来学习如何采取行动以最大化预期奖励，其研究发展与人工智能领域的发

展历程密切关联。早期，强化学习的概念起源于动物学习理论和心理学领域，人们试图理解和模拟生物智能体如何通过与环境的交互来逐步学习并适应不断变化的环境这一过程。然后，在上世纪五六十年代由 Bellman[17] 等人提出的自适应动态规划理论和贝尔曼方程等相关概念，标志着强化学习理论正在系统化、体系化，为强化学习的发展奠定了理论基础。随着计算机科学和人工智能的快速发展，强化学习逐渐成为研究热点。在上世纪八九十年代，学者们开始在强化学习领域进行深入研究，并提出了许多经典的强化学习算法。其中，Sutton 和 Barto 在其著作 [18] 中系统地介绍了强化学习的基本理论、算法和应用，为强化学习在人工智能领域的进一步落地提供了重要的参考。

强化学习目前在包括计算机科学技术领域、心理学领域、经济学领域、数学领域和控制工程领域等有着重要的意义和广泛的应用前景，图1-1展示了强化学习在以上单个领域内及其多个交叉领域下的重要应用。首先，强化学习能够使得机器在未知环境中自主学习与决策，实现了机器的智能化。其次，强化学习具有很强的泛化能力，能够处理各种复杂的环境和任务，在比如游戏[19-21]、机器人控制[22-24]、自动驾驶[25-27] 等领域有广泛的应用。另外，强化学习还为解决许多现实世界的问题提供了新的思路和方法，如储能系统管理[28-30]、优化控制[31-33]、金融交易[34-36] 等实际工程中都有普遍应用。因此，强化学习不仅在理论研究上有着重要意义，而且在实际应用中也有着广泛的应用前景，已然成为了推动人工智能技术发展的重要驱动力之一。

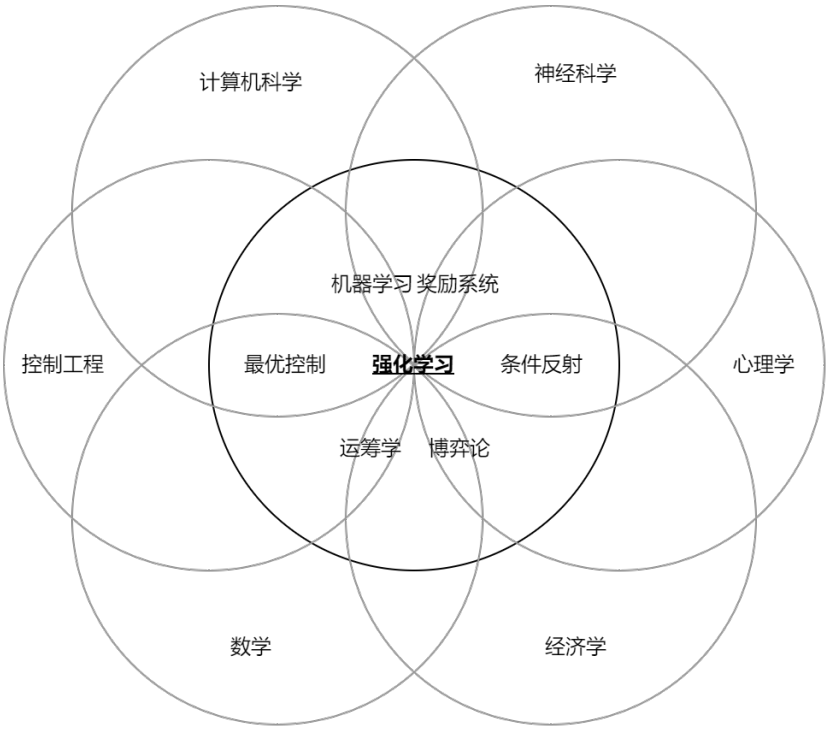

图 1-1 强化学习的应用领域

1.2 国内外研究现状

1.2.1 迭代学习控制的研究现状

迭代学习控制的主要研究方向包括以下几个方面：设计合适的迭代学习法则、探究算法的收敛性特征、研究在存在不确定性条件下的鲁棒性表现、优化学习过程的速度，并且探讨如何有效地应对初始条件的影响。在这些研究分支中，一个重要的研究课题是如何使得系统输出轨迹可以快速收敛于参考信号。

为了解决这一问题，学者们对迭代学习控制的优化理论进行了大量的研究并且提出了许多迭代学习控制设计的方法。这些设计方法根据是否需要已知被控系统动态过程精确的结构和参数被划分为基于模型的方法和模型非依赖的方法。基于模型的迭代学习控制方法需要利用系统的动力学信息才能精确设计学习律，Hatonen[37] 等人基于梯度理论提出了的梯度下降法的迭代学习控制，顾盼盼团队[38] 在梯度理论基础上引入 Nesterov 加速梯度方法研究了一类线性离散时间系统迭代学习控制的快速收敛问题，Togai[39] 通过利用 Gauss-Newton 法最小化跟踪误差求解得到了迭代学习的最优学习律，Amann 研究团队[40] 提出基于范数最优的迭代学习控制算法，并且此基础上使用模型信息考虑“未来”迭代实验的性能指标发表预测最优迭代学习控制律的设计方法[41]。

另一方面，不依赖模型参数的方法一般是利用先前迭代实验的输入和跟踪误差来设计学习律。Arimoto[3] 研究团队提出了 P 型学习律的迭代学习控制。Hatzikos[42] 等人在此基础上对迭代学习算法引入了遗传因子，提出了带遗传因子的 P 型学习律。Tayebi[43] 提出了一种自适应的迭代学习控制方案。还有基于数据驱动[44] 的迭代学习律的设计方法。关于迭代学习律的更多的设计方法和理论可以参考文献 [45]。一般来说，基于模型的迭代学习控制方法通常具有更加快速的收敛特性，但这是建立在已知系统动态过程的精确参数和结构的基础上，这和迭代学习控制设计初衷产生了出入。

这些大量研究均一致表明，迭代学习控制并非孤立的理论体系。通过将迭代学习控制与其他控制理论进行有效融合[46]，可以充分发挥出迭代学习控制与其它控制理论甚至是机器学习理论相结合的优势，使迭代学习控制能够针对不同实际应用场景，灵活运用相应方法解决具体问题。

1.2.2 强化学习的研究现状

强化学习和迭代学习同样是专注于试错学习，从过去的信息中学习经验来调整动作策略，是一种目标导向的学习决策的计算方法，其核心是以指标评估和策略改进为基本

思想实现最优化找到最优动作，设计的控制器能够使系统既能自适应学习又表现出最优性能，本质上强化学习为解决被控系统在未知参数条件下的优化控制问题提供了一种有效的控制器设计框架，特别是强化学习采用自适应动态规划技术, 已经得到了广泛的应用，表现出对很多控制问题和控制理论方法的兼容性。

强化学习理论根据智能体学习对象的不同可以分为基于值函数估计的强化学习算法和基于参数化策略搜索的强化学习算法。基于值函数的强化学习算法包括蒙特卡洛方法、时序差分算法和 Q-学习。蒙特卡洛方法是由 Simon 团队[47] 提出的，其原理是通过多次随机抽样的平均值对值函数进行估计来解决问题，核心在于通过估计状态价值函数或状态价值函数来解决在没有完全了解环境动态特性的情况下如何近似估计一个策略值函数的问题。时序差分算法由美国计算机科学家 Sutton 研究团队[48-49] 提出，结合了动态规划和蒙特卡洛方法的思想，通过在每一步更新中使用即时奖励和后继状态的估计值来学习值函数。而在此基础上的 Q-学习是由英国科学家 Watkins[50] 提出的一种无模型强化学习方法，其核心思想是直接学习动作状态价值函数来实现最优策略的搜索。

基于参数化策略搜索的强化学习算法主要包括策略梯度方法和“行动器-评判器”（Actor-Critic）方法。策略梯度方法是由美国计算机科学家 Williams[51] 团队提出，其原理是直接通过梯度法来更新策略参数化后的策略参数，避免繁琐的迭代寻找最优动作-状态价值函数而是直接求得最优策略。而“行动器-评判器”方法是 Barto[52] 提出的一种结合了值函数和策略梯度的方法，其中行动器表示行动策略，而评判器表示值函数。行动器负责执行行动策略，评判器则评估行动器策略表现的优劣，并给出相应的反馈以引导行动器更新策略。“行动器-评判器”方法既利用了值函数的信息来指导策略更新，又直接更新策略，因此可以更加高效地学习最优策略。

近期强化学习在迭代学习控制领域的应用取得了不错的成果。Lukas 团队在最新成果 [53] 中将一个迭代学习控制问题重新在强化学习框架下重新建立，并且通过马尔科夫决策过程初步建立了关于迭代学习控制基于模型的设计方法。Zhang 研究团队在文献 [54] 中说明强化学习与迭代学习控制有许多相似之处，并能用于解决迭代学习律的设计问题，然后通过仿真表明基于强化学习的迭代学习控制虽然收敛速度依旧慢于基于模型的迭代学习控制方法，但是能够满足高性能跟踪的要求。这为使用先进的强化学习设计新型的迭代学习控制算法开辟了新的可能性。Poot 团队在文献 [55] 提出了一种基于“行动器-评判器”的迭代学习控制方法，并与基于模型带有基函数的范数最优迭代学习控制方法进行了比较；结果表明，它能够在不使用显式模型信息的情况下实现迭代学习

收敛性取得与基于模型方法相同的性能。Shi 团队在文献 [56] 提出了一种在无需任何过程的先验知识条件下利用深度确定性策略梯度算法对非线性最优迭代学习律求解的方法。最近，我们在文献 [57] 提出了一种基于 Q-学习的预测最优迭代学习律设计，旨在减少传统基于模型的预测最优迭代学习控制方法对系统模型参数的依赖。

上述的研究成果充分揭示了迭代学习控制与强化学习之间存在着紧密的关系。这些研究不仅验证了两者具有的强互动性，而且进一步阐明了通过借助强化学习的理念和方法，在确保迭代学习控制系统能实现跟踪误差迅速收敛至零的同时，有望有效地减小甚至解决在设计迭代学习律时过度依赖于系统精确模型参数的问题，从而提升迭代学习控制方法的适应性和实用性。

1.3 本文研究内容

本论文研究课题来源于国家自然科学基金面上项目“广义多智能体系统迭代学习控制的自适应优化方法及应用”(基金号：62173151)。本文以传统基于模型最优迭代学习控制算法为研究背景，结合强化学习算法进行创新改进，旨在设计一种能够显著降低对模型依赖性的新型迭代学习控制方案。本文的研究工作聚焦于离散时间域下，针对传统依赖于系统模型信息的预测最优迭代学习控制问题以及参数最优迭代学习控制问题，分别深入探讨了采用基于值函数估算的强化 Q-学习算法，以及基于策略搜索的策略梯度算法。在此基础上，通过强化学习理论重新设计并优化了迭代学习控制算法，从而有效解决了以往需要精确求解系统动态来设计迭代学习律的问题。

本文的章节结构示意图如图1-2。具体章节安排如下：

第一章为本文绪论，首先分别介绍了迭代学习控制和强化学习的研究背景和研究意义，随后回顾了迭代学习控制理论和强化学习理论的发展历程，详细陈述了这两个领域目前的国内外研究现状。最后，引出了本文的核心研究问题，并简要介绍了各章节的内容安排。

第二章是预备知识，介绍了本文相关的知识和理论基础。这一章分为两部分：第一部分建立了一个基本的迭代学习控制问题，给出了满足迭代学习所需的条件，展示了迭代学习控制方法的具体流程；第二部分对强化学习的马尔科夫决策过程和算法分类进行了概述，并对本文所涉及的强化学习算法进行了基本介绍，还给出了相应算法的流程图。

第三章针对预测最优迭代学习律设计依赖于系统模型参数信息的问题，提出了未知系统动态下的基于 off-policy Q-学习的数据驱动预测最优迭代学习控制算法。首先在马

尔科夫决策框架下提出一种基于模型值迭代的预测最优迭代学习控制算法，通过理论分析证明算法和值迭代方法的收敛性。其次引入 off-policy Q 函数的概念，将迭代学习的 Bellman 方程求解问题转变为 Q 函数的极值的问题，提出了 off-policy Q-学习的数据驱动预测最优迭代学习控制算法并分析了算法的收敛性。最后通过比较基于模型方法和模型参数非依赖的预测最优迭代学习控制方法比较验证算法的有效性和收敛性。

第四章针对第三章提出的 off-policy Q-学习算法在控制领域中存在收敛速度慢，算法结构复杂需要大量的迭代才能收敛的问题，设计了未知模型参数的基于 on-policy 策略梯度的参数最优迭代学习控制算法。本章设计的 on-policy 策略梯度算法不再依赖于价值函数选择动作而是直接学习参数化的动作策略，提高了算法的收敛速度；接着利用价值函数估计基线和近端策略优化方法进一步提升算法的稳定性和收敛速度。还给出了上述两种 on-policy 策略梯度算法的表达形式和实现流程，分析算法满足收敛性的条件。最后比较基于模型的和不依赖模型参数信息的参数最优迭代学习控制方法的仿真实验结果验证算法的有效性和收敛性。

第五章针对第四章 on-policy 策略梯度在线学习存在的动作选择受限于行为策略和样本效率低泛化能力差的问题，探讨了基于 off-policy 策略梯度离线学习的参数最优迭代学习控制算法。本章在第四章的基础上，对目标策略与行为策略进行分离，前者不再被用于选取动作应用于被控系统中，在迭代过程中目标策略的突变将不会令实际系统产生较大的波动。首先推导了本章相关的 off-policy 策略梯度理论，其次分别设计了算法的“行动器”和“评判器”，给出了算法的实现流程和收敛性条件，最后比较基于模型的和无需预知模型参数信息的参数最优迭代学习控制方法的仿真实验结果验证算法的有效性和收敛性。

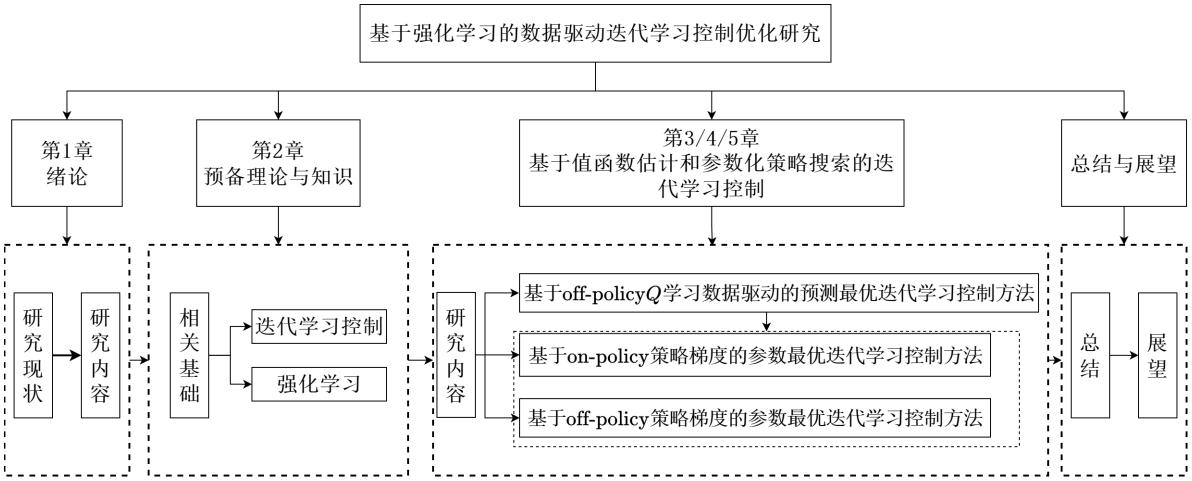

图 1-2 本文章节安排结构图

1.4 本章小结

本章介绍了迭代学习控制和强化学习的研究背景和意义，之后详细探讨了迭代学习控制和强化学习及其交叉领域的国内外研究现状与发展，最后通过简要说明了本文的各章节的研究内容和安排。

第二章　预备知识

2.1　迭代学习控制基本理论

考虑一类离散时间的单输入单输出（SISO）线性时不变（LTI）系统，本文后续章节的被控对象及其状态空间方程表示如下：

$$\begin{aligned} x_k(t+1) &= Ax_k(t) + Bu_k(t) \\ y_k(t) &= Cx_k(t) \end{aligned} \tag{2-1}$$

其中，$t \in [0, N-1]$ 表示离散系统的时间索引，k 是迭代实验的次数；$x_k(t) \in \mathbb{R}^n$，$u_k(t) \in \mathbb{R}$ 和 $y_k(t) \in \mathbb{R}$ 分别是迭代实验索引 k 上的状态、输入和输出；n 是系统的阶数；A、B 和 C 是具有适当维度的实数矩阵。所有迭代实验的初始条件都是相同的，即 $x_k(0) = x_0$，$k = 1, 2, ...,$ 并且在不失去一般性的情况下将迭代的初始值设置为 $x_0 = 0$。假设系统的相对阶数为一，即 $CB \neq 0$。

在系统 (2-1) 中，时间索引取 $t = 0, 1, 2, ..., N-1$，系统模型的输入和输出之间的数学关系可以表示为如下的等效形式：

$$y_k = Gu_k \tag{2-2}$$

其中，输入 u_k 和输出 y_k 的时间序列是 $N \times 1$ 维的列向量，可以表示为：

$$\begin{aligned} u_k &= \Big[u_k(0), u_k(1), \cdots, u_k(N-1)\Big]^T \\ y_k &= \Big[\, y_k(1), y_k(2), \cdots, y_k(N)\Big]^T \end{aligned} \tag{2-3}$$

并且系统的马尔科夫矩阵 G 被定义为：

$$G = \begin{bmatrix} CB & 0 & \cdots & 0 \\ CAB & CB & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ CA^{N-1}B & CA^{N-2}B & \cdots & CB \end{bmatrix} \tag{2-4}$$

其中，在假设条件 $CB \neq 0$ 下矩阵 G 是是可逆的，同时该矩阵是非奇异的。

另外，迭代学习控制的给定期望轨迹 r 描述为：

$$r = \Big[r(1), r(2), \cdots, r(N)\Big]^T \tag{2-5}$$

并且系统的跟踪误差被定义为：

$$e_k = r - y_k \tag{2-6}$$

在每次迭代实验中，迭代学习的控制器旨在使系统以尽可能高的精度尽快跟踪给定的期望轨迹。

为了便于理解，图2-1清晰展示了迭代学习控制方法的基本流程。

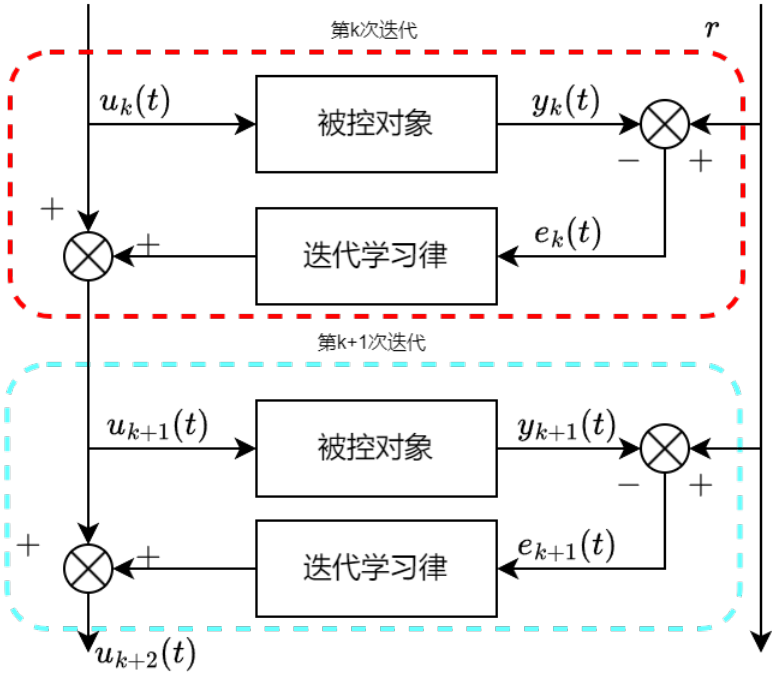

图 2-1 迭代学习控制算法框架图

假设 2.1: [58] 对于任何给定的期望轨迹 $r(t)$，系统总存在期望的控制输入 $u_d(t)$ 使得系统完全跟踪参考轨迹。

由于 $u_d(t)$ 是唯一的，因此控制输入 $u_k(t)$ 到 $u_d(t)$ 的一致收敛意味着跟踪误差 e_k 随着迭代实验次数增加最终收敛到 0。这是一个合理的假设，即该控制目标是可以达到的。

迭代学习控制问题的目标是通过探索迭代学习律：

$$u_{k+1} = f(u_k, e_k) \tag{2-7}$$

最终通过迭代使系统的误差满足如下收敛条件：

$$\lim_{k\to\infty} e_k = 0 \tag{2-8}$$

其中，f 是关于前一次迭代实验的跟踪误差 e_k 和输入 u_k 之间的数学关系。

2.2 马尔科夫决策过程

马尔科夫决策过程（Markov Decision Process，MDP）是一种通过与环境交互试错学习来实现目的的理论框架，为强化学习的研究提供了理论模型[59]，如图 (2-2)。智能体在时间步 t 观察到环境的状态 s_t，然后根据策略 $\boldsymbol{\pi}$ 选择一个动作 a_t，其行为策略通常表示为 $\boldsymbol{\pi}(a|s)$，即在状态 s 下采取动作 a 的概率；智能体执行动作 a_t 后，环境会根据状态转移函数 p 给出下一个状态 s_{t+1} 的概率分布，表示为 $p(s_{t+1}|s_t, a_t)$，同时根据奖励函数 R 给出即时奖励 $R(s_t, a_t)$。在 MDP 中，状态转移函数 p 需要同时满足马尔科夫性条件 (2-9)，即下一个状态 s_{t+1} 的可能值出现的概率只与前一个状态 s_t 和前一个动作 a_t 有关，与更早之前的状态和动作完全无关。

$$p(s_{t+1}|s_t, a_t) = p(s_{t+1}|s_1, a_1, \cdots, s_t, a_t) \tag{2-9}$$

同理，在下一个环境状态下的智能体会继续依旧行为策略选择下一次的动作并获得下一次的环境反馈奖励信号，同时也会进入再下一个状态。上述过程的循环往复就构成了强化学习这一智能体与环境交互式的学习过程。

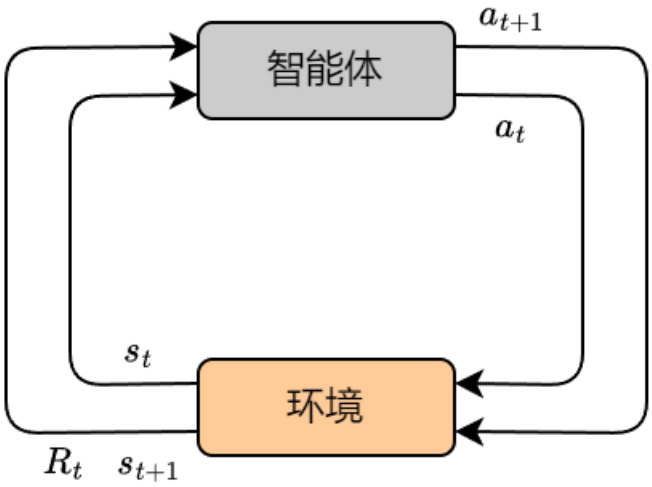

图 2-2 智能体与环境交互过程

2.3 强化学习的算法分类

在第一章中在介绍强化学习的研究现状时将强化学习算法分为基于值函数估计的强化学习算法和基于参数化策略搜索的强化学习算法两类。本文涉及的算法主要包括 Q-学习算法、策略梯度算法、近端策略优化算法和“行动器-评判器”算法，本小节将着重介绍这四个方法寻找最优动作方案的过程，突显它们之间的联系。

2.3.1 Q-学习算法理论

Q-学习算法的重要思想是智能体通过采取不同的动作 a_t 与环境进行互动获得相应的奖励 R_t，以此建立动作-状态价值表 (Q 表)；然后采用时序差分学习的方法更新 Q 表中动作-状态值 (Q 值)，在不断探索中增加对环境的认知。经过多次的迭代后，智能体最终得到 Q 表中已经收敛的 Q 值，通过选取 Q 值最大的动作实现目标得到最优动作策略。Q-学习通过更新动作状态价值函数找到最终的 Q 表，Q 值的更新方法及最优动作策略如式 (2-10) 和 (2-11) 所示。

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha[R_{t+1} + \gamma \max_a Q(s_{t+1},a) - Q(s_t,a_t)] \tag{2-10}$$

$$\boldsymbol{\pi}^*(s) = \arg\max_a Q(a,s) \tag{2-11}$$

其中，$0 < \alpha \leq 1$ 是学习律，$0 < \gamma \leq 1$ 为折扣因子。智能体在 t 时刻处于状态 s_t，动作状态价值函数为 $Q(s_t,a_t)$，采取动作 a_t 后在 $t+1$ 时刻状态转移至 s_{t+1} 同时获得反馈奖励信号 R_{t+1}，此时智能体通过查询已有的 Q 表得到在状态 s_{t+1} 下 Q 值最大的动作 a，以此求出 $Q(s_{t+1},a)$ 并更新 t 时刻下的 $Q(s_t,a_t)$。

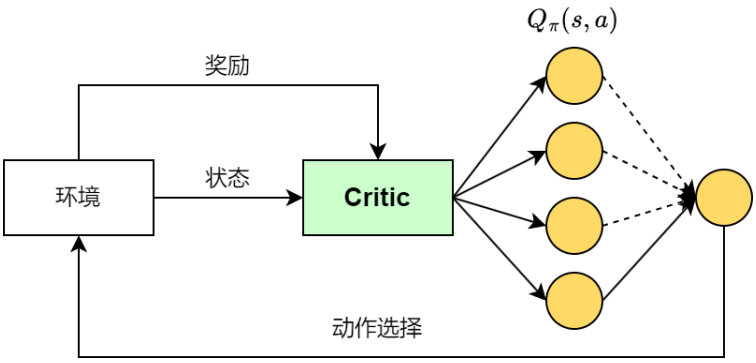

图 2-3 Q-学习算法流程图

图2-3展示了智能体通过模拟执行所有可能的动作来计算 $Q(s,a)$，然后根据某种策略（如贪心策略）选择令 Q 值最大化的最优动作。将 Q-学习视为评价器（Critic）时，$Q(s,a)$ 实际上衡量了在当前状态 s 下执行动作 a 的效果，从而量化了动作选择的优劣性。

2.3.2 策略梯度算法理论

策略梯度算法的主要思路是首先参数化智能体的动作策略函数 $\boldsymbol{\pi}(a_t|s_t)$，根据当前的状态 s_t 及策略参数 $\boldsymbol{\theta}_t$ 生成动作的概率分布 $\boldsymbol{\pi}(a_t|s_t,\boldsymbol{\theta}_t)$，然后智能体以此选择动作得

到计算每个动作对应的某种性能指标 $J(\boldsymbol{\theta}_t)$，最后利用该指标关于策略参数的梯度更新优化策略参数，以提高预期性能指标。通过不断迭代，策略参数达到收敛值，利用该参数的动作策略函数 $\boldsymbol{\pi}(a_t|s_t,\boldsymbol{\theta}^*)$ 可以选取最优动作 a^*。策略梯度算法策略参数 $\boldsymbol{\theta}_t$ 的更新方法如式 (2-12) 所示。

$$\boldsymbol{\theta_{t+1}} = \boldsymbol{\theta_t} + \alpha\nabla J(\boldsymbol{\theta}_t) \tag{2-12}$$

策略梯度算法中策略可以是确定性的（确定性策略）或概率性的（概率性策略）。确定性策略的动作选择可以表示为一个函数 $\boldsymbol{\pi}$ 在状态 s 下输出要执行的具体动作 a，如下式所示：

$$a = \boldsymbol{\pi}(s) \tag{2-13}$$

上式中，如果状态确定为 s_t，那动作的选择将是唯一的，即式 (2-13) 的值为 1。而概率性策略的策略函数 $\boldsymbol{\pi}$ 表示了智能体在状态 s 下采取每个动作 a 的概率分布。

$$\boldsymbol{\pi}(a|s) = p(a|s) \tag{2-14}$$

在实际操作中，通常采用概率最高的动作作为最优动作，即式 (2-15)。

$$a^* = \arg\max_a \boldsymbol{\pi}(a|s_t) \tag{2-15}$$

在分幕式情况下将性能指标定义为该幕的状态价值函数：

$$J(\boldsymbol{\theta}) = v_{\boldsymbol{\pi_\theta}}(s) \tag{2-16}$$

其中，$v_{\boldsymbol{\pi_\theta}}(s)$ 是在策略 $\boldsymbol{\pi_\theta}$ 下真实的价值函数，其由策略参数 $\boldsymbol{\theta}$ 决定。然而仅通过式 (2-16) 计算对策略参数的梯度仍然是具有难度的。在强化学习中，尽管在特定状态下通过策略参数可以直接明了地计算出对动作选择及收益的影响，但由于状态分布受环境动态特性所决定，通常难以精确获知策略对整体状态分布的具体改变，然而，性能指标对策略参数的梯度却恰恰依赖于这一影响，因此，如何有效地估计这个梯度成为关键。

针对这一问题，策略梯度定理[60] 提供了解决方案；下面将给出该定理的内容及部分证明。

定理 2.1: 性能指标 $J(\boldsymbol{\theta})$ 的梯度根据策略梯度定理表达式如下：

$$\nabla J(\boldsymbol{\theta}) \propto \sum_0 \mu(s) \sum_a q_{\boldsymbol{\pi_\theta}}(s,a)\nabla\boldsymbol{\pi}(a|s,\boldsymbol{\theta}) \tag{2-17}$$

其中，该梯度是关于策略参数向量 $\boldsymbol{\theta}$ 的列向量，$\boldsymbol{\pi}$ 表示参数向量对应的策略，符号 $\propto$ 表示“正比于”。μ 是在策略 $\boldsymbol{\pi}$ 下的同轨策略分布，可以被定义为，对于所有状态 $s \in \mathcal{S}$，满足如下定义：

$$\mu(s) = \frac{\eta(s)}{\sum_{s'} \eta(s')} \tag{2-18}$$

$\eta(s)$ 表示从状态 s' 转移到状态 s 中消耗的平均时刻步数，即期望步数。

证明: 根据性能指标的定义 (2-16)，状态价值函数的梯度可以改写为动作价值函数的梯度：

$$\nabla J(\boldsymbol{\theta}) = \nabla v_{\boldsymbol{\pi_\theta}}(s) = \nabla[\boldsymbol{\pi_\theta}(a|s) q_{\boldsymbol{\pi}}(s,a)], s \in S \tag{2-19}$$

按照微积分法则，根据价值函数的定义：

$$\nabla J(\boldsymbol{\theta}) = \sum_a \left[\nabla \boldsymbol{\pi_\theta}(a|s) q_{\boldsymbol{\pi}}(s,a) + \boldsymbol{\pi_\theta}(a|s) \sum_{s'} p(s'|s,a) \nabla v_{\boldsymbol{\pi_\theta}}(s') \right] \tag{2-20}$$

根据价值函数和性能指标的关系，将上式价值函数梯度 $\nabla v_{\boldsymbol{\pi_\theta}}(s')$ 展开：

$$\begin{aligned} \nabla J(\boldsymbol{\theta}) = & \sum_a \left[\nabla \boldsymbol{\pi_\theta}(a|s) q_{\boldsymbol{\pi}}(s,a) + \boldsymbol{\pi_\theta}(a|s) \sum_{s'} p(s'|s,a) \right. \\ & \left. \sum_{s'} \left[\nabla \boldsymbol{\pi_\theta}(a'|s') q_{\boldsymbol{\pi}}(s',a') + \boldsymbol{\pi_\theta}(a'|s') \sum_{s''} p(s''|s',a') \nabla v_{\boldsymbol{\pi_\theta}}(s'') \right] \right] \\ = & \sum_{x \in \mathcal{S}} \sum_{k=0}^{\infty} \Pr(s \to x, k, \boldsymbol{\pi}) \sum_a \nabla \boldsymbol{\pi_\theta}(a|x) q_{\boldsymbol{\pi_\theta}}(x,a) \end{aligned} \tag{2-21}$$

上式中，$\Pr(s \to x, k, \boldsymbol{\pi})$ 的含义是在策略 $\boldsymbol{\pi_\theta}$ 下，状态 s 在 k 步内转移到状态 x 的总概率；同理，$\sum_{k=0}^{\infty} \Pr(s \to x, k, \boldsymbol{\pi})$ 即为状态 s 在 k 步内转移到状态 x 的期望步数，根据定义 (2-18)，上式 (2-21) 可以化简为：

$$\begin{aligned} \nabla J(\boldsymbol{\theta}) = & \sum_{s'} \eta(s') \sum_s \frac{\eta(s)}{\sum_{s'} \eta(s')} \sum_a \nabla \boldsymbol{\pi_\theta}(a|s) q_{\boldsymbol{\pi_\theta}}(s,a) \\ = & \sum_{s'} \eta(s') \sum_s \mu(s) \sum_a \nabla \boldsymbol{\pi_\theta}(a|s) q_{\boldsymbol{\pi_\theta}}(s,a) \end{aligned} \tag{2-22}$$

由于式中关于状态 s' 下 $\eta(s')$ 的求和在状态 s 下是一个常量，所以有：

$$\nabla J(\boldsymbol{\theta}) \propto \sum_s \mu(s) \sum_a q_{\boldsymbol{\pi_\theta}}(s,a) \nabla \boldsymbol{\pi}(a|s, \boldsymbol{\theta}) \tag{2-23}$$

证明完毕。 □

根据定理2.1，策略参数更新式 (2-12) 可以被改写为：

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha \mathbb{E}_{\boldsymbol{\pi}} \left[\sum_a q_{\boldsymbol{\pi}}(S_t, a) \nabla \boldsymbol{\pi}(a|S_t, \boldsymbol{\theta}) \right] \tag{2-24}$$

在强化学习中，对随机变量所有可能动作的价值函数求和等价为求对概率 $\boldsymbol{\pi}(a|s, \boldsymbol{\theta})$ 的期望，对式 (2-24) 引入动作空间 A_t，$\nabla J(\boldsymbol{\theta})$ 改写为：

$$\begin{aligned} \nabla J(\boldsymbol{\theta}) &= \mathbb{E}_{\boldsymbol{\pi}} \left[\sum_a \boldsymbol{\pi}(a|S_t, \boldsymbol{\theta}) q_{\boldsymbol{\pi}}(S_t, a) \frac{\nabla \boldsymbol{\pi}(a|S_t, \boldsymbol{\theta})}{\boldsymbol{\pi}(a|S_t, \boldsymbol{\theta})} \right] \\ &= \mathbb{E}_{\boldsymbol{\pi}} \left[q_{\boldsymbol{\pi}}(S_t, A_t) \frac{\nabla \boldsymbol{\pi}(A_t|S_t, \boldsymbol{\theta})}{\boldsymbol{\pi}(A_t|S_t, \boldsymbol{\theta})} \right] \end{aligned} \tag{2-25}$$

在策略评估方法中，在状态空间 S_t 中所有动作集合 A_t 的回报 G_t 等于其在相同状态下采取同一动作集合的动作价值函数值，满足如下等式：

$$\mathbb{E}_{\boldsymbol{\pi}}[G_t|S_t, A_t] = q_{\boldsymbol{\pi}}(S_t, A_t) \tag{2-26}$$

将式 (2-26) 代入 (2-25) 得到梯度公式

$$\nabla J(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{\pi}} \left[G_t \frac{\nabla \boldsymbol{\pi}(A_t|S_t, \boldsymbol{\theta})}{\boldsymbol{\pi}(A_t|S_t, \boldsymbol{\theta})} \right] \tag{2-27}$$

综上，得到了策略梯度的参数更新式 (2-28) 如下所示。

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha \mathbb{E}_{\boldsymbol{\pi}} \left[G_t \frac{\nabla \boldsymbol{\pi}(A_t|S_t, \boldsymbol{\theta})}{\boldsymbol{\pi}(A_t|S_t, \boldsymbol{\theta})} \right] \tag{2-28}$$

图2-4展示了智能体根据基于策略参数 $\boldsymbol{\theta}$ 的策略函数 $\boldsymbol{\pi}(a|s_t, \boldsymbol{\theta})$ 结合式 (2-15) 选取动作的过程，每次都将选择出更有可能获得高回报的动作。将策略梯度算法视为行动器（Actor）时，智能体通过与环境的交互和参数更新来优化策略参数，以使智能体在环境中获得更高的回报。

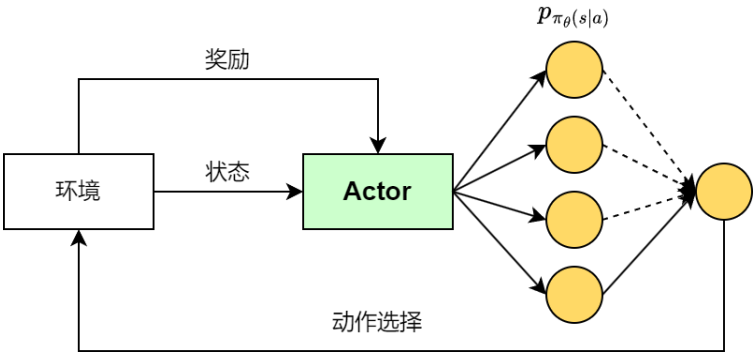

图 2-4 策略梯度算法流程图

2.3.3 近端策略优化

近端策略优化方法本质还是策略梯度算法的一种，主要用于解决策略梯度法更新过程导致策略的大幅波动，进而威胁到训练过程的稳定性和收敛性的问题。为了改进策略梯度算法，文献 [61] 提供了一种名为“近端策略优化”的机制，它通过设定一个策略近端约束，确保在每一次迭代更新过程中，策略的变化幅度保持在一个合理的范围内，从而保证了训练的稳定性。

定义 $r_t(\theta)$ 表示新策略和旧策略之间的重要性比率：

$$r_t(\boldsymbol{\theta}) = \frac{\boldsymbol{\pi}_{\boldsymbol{\theta}}(s_t, a_t)}{\boldsymbol{\pi}_{\boldsymbol{\theta}_{old}}(s_t, a_t)} \tag{2-29}$$

$r_t(\theta)$ 的具体含义为新策略在状态 s_t 下选择动作 a_t 的概率与旧策略在相同状态下选择相同动作的概率的比值，用来量化新策略相对于旧策略的相对改变。根据文献 [62] 保守策略下的性能指标为：

$$L^{CPI}(\boldsymbol{\theta}) = \mathbb{E}_t\left[\frac{\boldsymbol{\pi}_{\boldsymbol{\theta}}(a_t \mid s_t)}{\boldsymbol{\pi}_{\boldsymbol{\theta}_{\text{old}}}(a_t \mid s_t)}\delta_t\right] = \mathbb{E}_t[r_t(\boldsymbol{\theta})\delta_t] \tag{2-30}$$

为了限制新策略和旧策略之间的相对变化，从而控制策略更新的幅度，以确保相对保守不至于引发算法发散的方式更新策略。限制约束函数定义为：

$$\text{clip}(r_t(\boldsymbol{\theta}), 1-\epsilon, 1+\epsilon) \tag{2-31}$$

其中，ϵ 是一个预先设定的超参数，它决定了允许梯度变化的最大幅度。由此可以得到近端策略优化的性能指标函数：

$$L(\boldsymbol{\theta}) = \mathbb{E}_t[\min(r_t(\boldsymbol{\theta})\delta_t, \text{clip}(r_t(\boldsymbol{\theta}), 1-\epsilon, 1+\epsilon)\delta_t)] \tag{2-32}$$

上式中，$L(\boldsymbol{\theta})$ 也是用来更新策略参数 θ 的代理目标函数；δ_t 是优势函数，衡量了在给定状态下采取某一动作相对于平均预期奖励的优势程度，取最小值运算保证了每次梯度的波动幅度。相应的，策略梯度定理中的策略参数更新式 (2-12) 改写为：

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha\nabla L(\boldsymbol{\theta}) \tag{2-33}$$

近端策略优化的目标是在最大化期望累积回报的同时通过引入约束来确保策略更新的稳定性和可控性 (2-32)，限制约束函数 (2-31) 在每次更新中都消除了将 $r_t(\theta)$ 移到区间外 $[1-\epsilon, 1+\epsilon]$ 的可能性。

2.3.4 “行动器-评判器”算法

前面介绍的强化学习算法中，如果只使用 Q-学习寻找最优动作，算法在运行时需要时刻维护一个 Q 表，并且可能因为数据受限收敛缓慢导致估计的 Q 不够准确；而如果只使用策略梯度方法寻找最优策略，算法在面对高度随机的环境时，需要大量采样来估计策略梯度导致训练效率较低。这迫切需要一种方法来克服这两种方法的局限性，使得在训练过程中能够更有效地学习到最优策略。

“行动器-评判器”算法将基于值函数估计的强化学习算法和基于参数化策略搜索的强化学习算法结合起来，经常被运用于改进策略梯度理论中。“行动器-评判器”算法的框架如图2-5所示。“行动器-评判器”框架包含两个部分，一个是行动器（Actor），使用策略梯度算法更新；另一个是评判器（Critic），使用函数逼近的方法估计值函数。对于环境的反馈信号状态 s，行动器会输出选择的动作 a 然后再次与环境交互，交互之后产生即时奖励 R。将状态 s 或状态-动作二元组 (s,a) 输出到评判器中，评判器采用梯度法预测状态价值函数 v 或时序差分的方式估计 Q 函数更新行动器参数，然后循环往复直至智能体选择最优动作。

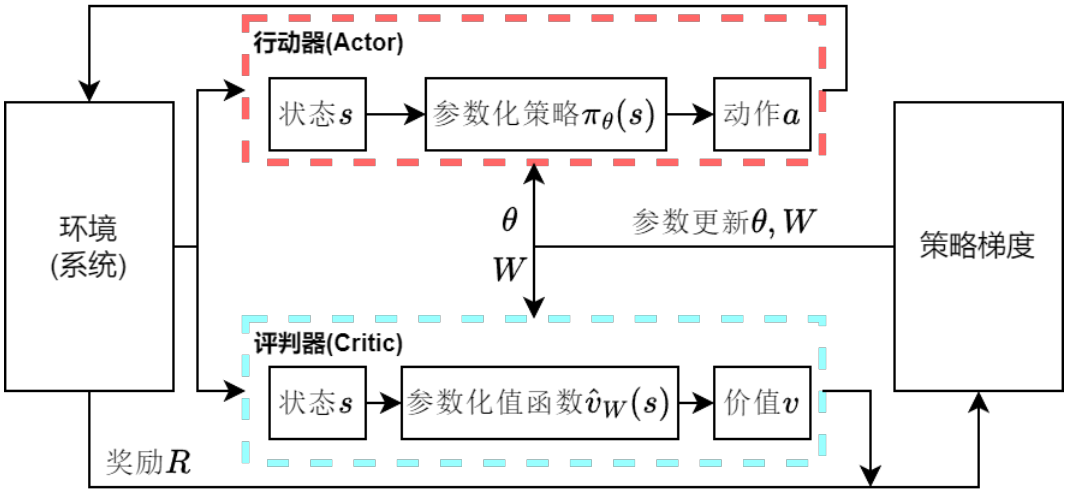

图 2-5 “行动器-评判器”算法框架图

2.4 本章小结

本章首先对迭代学习控制问题进行简要说明，描述迭代学习控制的目标具体的迭代学习流程；然后从强化学习的马尔科夫决策过程和算法分类这两个方面对本文涉及的强化学习算法进行概述，还给出了本文涉及算法的框架图，方便后续章节将强化学习算法的推广和运用于实际问题。

第三章　基于 off-policy Q-学习的预测最优迭代学习控制方法

3.1　引言

在先前的研究中，涉及利用系统模型参数的迭代学习控制通常在实践中相比于传统的数据驱动的迭代学习控制往往能表现出更好的收敛特性，这种良好的收敛特性是依赖于系统的模型信息得到的。然而在实践中，随着日趋复杂的现代产业和工业自动化系统，想要获得这些系统的精确模型可能很昂贵甚至根本难以得到；如何在保证系统拥有不错的收敛特性前提下不依赖于系统模型参数信息设计迭代学习控制方法成为一个挑战。

Q-学习算法鼓励探索未知领域，可以在没有预先知道最优解决方案的情况下，通过执行不同的动作并观察结果得到最优的 Q 函数进而得到最优动作[63]。文献 [54] 对迭代学习控制和强化学习之间的存在的关联性进行了初步研究，发现有许多相似之处，给迭代学习控制和强化学习两个不同领域的交叉研究提供了一些启示文献；另一方面，文献 [64] 的研究成果表明一个迭代学习控制问题可以被描述为强化学习框架下的问题，说明可以利用强化学习的理论解决迭代学习控制未知系统模型信息下使系统跟踪误差快速收敛到的问题，这更加印证了强化学习和迭代学习控制理论之间有很强的互动性。

本章研究了在未知系统动态特性信息下预测最优迭代学习控制问题。首先，利用基于 Δu_{k+1} 的迭代学习误差建立线性二次代价函数，提出了预测最优迭代学习控制问题，并在已知系统动态信息的情况下根据离散代数里卡提方程（DARE）中求得了该问题的解，由此建立了已知系统动态的预测最优算法。其次，提出了在未知系统动态下寻找预测最优迭代学习控制最优解的强化 Q-学习方法，旨在仅利用迭代学习控制的输入和输出数据，开发一种数据驱动的迭代学习算法，以更快的收敛速度实现零误差地完全跟踪期望轨迹。基于性能函数，通过对 Δu_{k+1} 和控制误差的试验，在迭代学习中精心选择一个 Q 函数，使其更新后的迭代学习律性能使得系统快速达到收敛值。第三，证明了算法的收敛性并讨论算法的收敛条件。最后，使用进行数学仿真，验证了所提算法的有效性，并展示了其相对于传统方法的优势。

本章组织框架如下。3.2 节描述了一个基本预测最优迭代学习控制问题框架。3.3 节通过 off-policy$Q$-学习理论，求出在 off-policy 形式下的 Q 函数，进一步推导得出 off-policy 形式下的递归 Bellman 方程，并在此基础上开发了一种基于 off-policy Q-学习

的数据驱动迭代学习算法解决迭代学习控制律设计问题。3.4 节设计了数值仿真，以证明依据 off-policy 设计的学习律的有效性。最后，3.5 节给出对本章内容的小结。

3.2 预测最优迭代学习控制问题

预测最优迭代学习控制最显著的特点是利用当前迭代和未来迭代实验的预测误差来计算当前控制输入，从而使被控系统获得更好的收敛特性。根据文献 [65]，预测最优迭代学习控制基于后续迭代输入和误差的性能指标函数如下：

$$J(u_{k+1}) = \sum_{i=k}^{\infty} \gamma^{i-k}(\|e_{k+i}\|_Q^2 + \gamma\|\Delta u_{k+i+1}\|_R^2) \tag{3-1}$$

其中，$\Delta u_{k+i} = u_{k+i} - u_{k+i-1}$; γ 是衡量在未来迭代学习实验中的误差和输入的重要性变化的权重因子，向量的范数定义为：

$$\begin{aligned} \|e_{k+i}\|_Q^2 &= e_{k+i}^T Q e_{k+i} \\ \|\Delta u_{k+i}\|_R^2 &= \Delta u_{k+i}^T R \Delta u_{k+i} \end{aligned} \tag{3-2}$$

Q 和 R 是对称正定矩阵；当迭代学习控制的学习律为：

$$\Delta u_{k+1} = L e_k \tag{3-3}$$

上式中，L 为学习律增益矩阵，通过上式将基于输入的性能指标函数转换为基于误差的性能指标函数，即：

$$\begin{aligned} J(e_k) =& \sum_{i=k}^{\infty} \gamma^{i-k}(\|e_i\|_Q^2 + \gamma\|\Delta u_{i+1}\|_R^2) \\ =& e_k^T Q e_k + \gamma \Delta u_{k+1}^T R \Delta u_{k+1} + J(e_{k+1}) \end{aligned} \tag{3-4}$$

预测最优迭代学习控制的目标是确定能使性能指标函数达到最小值的当前迭代试验中的最优学习律 Δu_{k+1}，因此，这一设计问题可以转化为性能指标函数 $J_{k+1}(\Delta u_{k+1})$ 最小问题：

$$\begin{aligned} J^*(e_k) &= \min_{\Delta u_{k+1}} J(e_k) \\ \Delta u_{k+1}^* &= \arg\min_{\Delta u} J(e_k) \end{aligned} \tag{3-5}$$

定理 3.1: [65] 预测最优迭代学习控制算法可使跟踪误差规范为零，并满足以下要求收敛 $\exists\, 0 < \beta < 1$

$$\|e_{k+1}\|_Q \leq \beta \|e_k\|_Q \tag{3-6}$$

其中，β 为收敛速率。结果表明，跟踪误差和 Δu_{k+1} 均收敛于零，满足如下条件：

$$\lim_{k\to\infty} e_k = 0,\ \lim_{k\to\infty} \Delta u_{k+1} = 0 \tag{3-7}$$

3.3 马尔科夫决策过程的预测最优迭代学习控制问题

本节将在马尔科夫框架下重新描述预测最优迭代学习控制问题，并依据 Bellman 方程和值迭代方法给出基于模型的迭代学习律设计方法。

3.3.1 马尔科夫框架下的预测最优迭代学习控制问题

预测最优迭代学习控制可以被重新改写为一个马尔科夫决策过程（MDP）。根据文献 [60]，一个 MDP 被定义为 (S, A, p, R)，其中：

• S 是状态空间，状态 s 满足 $s \in S$。在迭代学习中，状态 s 定义为第 k 次迭代学习的误差 $e_k \in \mathbb{R}^N$，N 为实验长度。

• A 是动作空间，动作 a 满足 $a \in A$。动作 a 定义为第 k 次迭代学习的输入增量 $\Delta u_{k+1} \in \mathbb{R}^N$，N 为实验长度。

• $f\left(e_k, \Delta u_{k+1}\right)$ 是状态转移函数。状态转移函数在迭代学习控制理论中满足如下：

$$e_{k+1} = f\left(e_k, \Delta u_{k+1}\right) = e_k - G\Delta u_{k+1} \tag{3-8}$$

• R 是奖励函数。在第 k 次迭代学习状态 e_k 下奖励函数满足如下定义：

$$R(e_k, \Delta u_{k+1}) = \|e_k\|_Q^2 + \gamma\|\Delta u_{k+1}\|_R^2 \tag{3-9}$$

其中，γ 为式 (3-1) 中的权重因子。

MDP 的设计目标是通过最小化状态价值函数寻找一个最优的行为策略（即反馈控制增益 L），为智能体找到一个更优的动作 Δu_{k+1}。状态价值函数定义为：

$$V(e_k) = \sum_{\kappa=k}^{\infty} \gamma^{\kappa-k} R(e_\kappa, \Delta u_{\kappa+1}) \tag{3-10}$$

其中，$\gamma > 0$ 在迭代学习控制中的意义是决定在未来迭代实验中的成本对当前迭代实验的价值影响的折现因子，即 γ 值越大，表示未来的成本越重要。

3.3.2 值迭代求解预测最优迭代学习控制问题

根据最优性理论和奖励函数 (3-3)，式 (3-10) 可改写为：

$$\begin{aligned} V(e_k) =& e_k^T Q e_k + \gamma \Delta u_{k+1}^T R \Delta u_{k+1} + \sum_{i=k+1}^{\infty} \gamma^{i-k-1} R(e_i, \Delta u_{i+1}) \\ =& e_k^T Q e_k + \gamma \Delta u_{k+1}^T R \Delta u_{k+1} + V(e_{k+1}) \end{aligned} \tag{3-11}$$

基于迭代学习律 (3-3) 和状态转移函数 (3-8), 根据对称性状态价值函数满足等式：

$$J(e_k) = e_k^T P e_k \tag{3-12}$$

其中，P 是一个对称正定矩阵。将上式代入状态价值函数 (3-11)，可构造得到 Bellman 方程：

$$e_k^T P e_k = e_k^T Q e_k + \gamma \Delta u_{k+1}^T R \Delta u_{k+1} + \gamma e_{k+1}^T P e_{k+1} \tag{3-13}$$

然后，相应的哈密顿-雅克比方程被定义为如下：

$$H(e_k, \Delta u_{k+1}) = e_k^T Q e_k + \gamma \Delta u_{k+1}^T R \Delta u_{k+1} + \gamma e_{k+1}^T P e_{k+1} - e_k^T P e_k \tag{3-14}$$

根据极小值原理，通过求解：

$$\frac{\partial H(e_k, \Delta u_{k+1})}{\partial \Delta u_{k+1}} = 0 \tag{3-15}$$

可以得到最优迭代学习律 Δu_{k+1}^*。

定理 3.2: (DARE 的唯一解) 当系统的输入输出矩阵 G 是完全已知且非奇异的，系统符合迭代学习控制问题的要求，且每次迭代都能获得对于跟踪误差的完整测量值，此时最优控制学习律 $\Delta u_{k+1}^* = L^* e_k$ 中学习增益矩阵 L^*：

$$L^* = \left(R + G^T P G\right)^{-1} G^T P \tag{3-16}$$

其中，$P = P^T$ 是离散代数黎卡提方程 (3-17) 的唯一解。

$$P = Q + \gamma P - \gamma P G \left(R + G^T P G\right)^{-1} G^T P \tag{3-17}$$

证明: 对偏微分方程 (3-15) 展开求解：

$$\begin{aligned} \frac{\partial H(e_k, \Delta u_{k+1})}{\partial \Delta u_{k+1}} =& 2R\Delta u_{k+1} - 2G^T P(e_k - G\Delta u_{k+1}) \\ =& 0 \end{aligned} \tag{3-18}$$

求解上述偏微分方程即可得到最优迭代学习律：

$$\Delta u_{k+1}^{*}=\left(R+G^{T}PG\right)^{-1}G^{T}Pe_{k} \tag{3-19}$$

将 (3-19) 代入 Bellman 方程 (3-13) 可以求出黎卡提方程 (3-17) 的表达式。证明完毕。□

在实际应用中，由于方程 (3-17) 求解 P 困难进而导致无法求出最优迭代学习律，一般考虑使用值迭代进行求解。值迭代法[66] 具有结构简单易于实现，不需要直接对 Lyapunov 方程进行求解的特点。下面将给出值迭代的预测最优迭代学习控制 (Value iteration for predictive optimal iterative learning control, VI-POILC)，即算法 3-1 的伪代码。

算法 3 - 1: 基于预测最优迭代学习控制的值迭代算法

系统数据: 误差 e_k ，系统输入增量 Δu_{k+1}。

初始化: 选择一可行收敛的迭代学习律 Δu_0、DARE 方程的初始解 P_0 和初始学习增益矩阵 L_0；令 $k=1,2,...,m$，$j=1,2,...,$ 开始迭代。

输出: 代入 $\Delta u_{k+1}^{*}=L^{j+1}e_k$ 求出最优迭代学习律

开始循环 $k=1\rightarrow m$

数据收集 *收集迭代学习运行状态信息误差 e_k 和输入增量 Δu_{k+1}*

k=k+1

结束循环

开始循环 $j=1\rightarrow\infty$

值迭代 *通过如下更新式计算 P^{j+1}：*

$$P^{j+1}=Q+\gamma P^{j}-\gamma P_{k}G\left(R+G^{T}P^{j}G\right)^{-1}G^{T}P^{j} \tag{3-20}$$

策略更新 *计算更新后的学习增益矩阵：*

$$L^{j+1}=\left(R+G^{T}P^{j+1}G\right)^{-1}G^{T}P^{j+1} \tag{3-21}$$

j=j+1

终止更新条件 *若相邻两次的学习增益矩阵 L_k，对任意的 ε，都满足 $\|L^{j+1}-L^{j}\|<\varepsilon$, ε 是某一小正数，停止迭代*

结束循环

值迭代算法中每步迭代中求得的最优动作策略序列 Δv_{k+1} 定义为：

$$\Delta v_{k+1}^{j+1} = \arg\min_{\Delta u_k} \left\{ e_k^T Q e_k + \gamma \Delta u_{k+1}^j R \Delta u_{k+1}^j + V^j(e_{k+1}) \right\} \tag{3-22}$$

相应的值函数更新序列：

$$V^{j+1}(e_{k+1}) = \min_{\Delta u_k} \left\{ e_k^T Q e_k + \gamma \Delta v_{k+1}^j R \Delta v_{k+1}^j + V^j(e_{k+1}) \right\} \tag{3-23}$$

引理 3.1: [67] 如果有任意一迭代学习律序列 Δv_{k+1}，控制策略 Δv_{k+1}^j 定义入 (3-18) 所示。$V(e_{k+1})^j$ 如 (3-19) 式子定义，Λ^j 通过 $\Lambda^{j+1}(e_k) = e_k^T Q e_k + \Delta u_{k+1}^j{}^T(k) R \Delta u_{k+1}^j + \Lambda^j(e_{k+1})$ 迭代得到，假设对于第 $j=0$ 次迭代满足 $0 \le V^0 \le \Lambda^0$ 条件，则 $V^j \le \Lambda^j$。

引理 3.2: [68] 假定对于一个迭代学习控制过程，存在学习律 Δu_{k+1}，那么定义值函数序列如 (3-19)，存在一个上界 Y 使得 $0 \le V^j \le Y$ 。

定理 3.3: (算法 3-1 收敛性) 假定对于一个迭代学习控制过程有学习律 Δu_{k+1}，假设存在可行的控制策略，并且使用值迭代方法求解 $V^j(e_k)$ 和 Δv_{k+1}^j，那么这两个序列将分别收敛于最优值 $V^*(e_k)$ 和 Δv_{k+1}^*

证明: 为方便证明，引入如下新的序列：

$$\xi^{j+1}(e_k) = e_k^T Q e_k + \Delta v_{k+1}^j{}^T(k) R \Delta v_{k+1}^j + \xi^j(e_{k+1}) \tag{3-24}$$

其中，$\xi^0(e_k) = V^0(e_k) = 0$。

第 $j=0$ 次迭代时，代入 (3-23) 和 (3-24) 可得：

$$V^1(e_k) - \xi^0(e_k) = e_k^T Q e_k + \Delta v_{k+1}^1{}^T(k) R \Delta v_{k+1}^1 - 0 \tag{3-25}$$

得出 $V^1(e_k) - \xi^1(e_k) \ge 0$。

同理，第 $j=1$ 次迭代时，再代 (3-11) 入可得：

$$\begin{aligned} V^2(e_k) - \xi^1(e_k) =& e_k^T Q e_k + (\Delta v_{k+1}^2)^T R \Delta v_{k+1}^2 - e_k^T Q e_k - (\Delta v_{k+1}^1)^T R \Delta v_{k+1}^1 \\ =& \Delta v_{k+1}^2{}^T(k) R \Delta v_{k+1}^2 + e_{k+1}^T Q e_{k+1} \\ &+ \Delta v_{k+1}^1{}^T(k) R \Delta v_{k+1}^1 - \Delta v_{k+1}^1{}^T(k) R \Delta v_{k+1}^1 \end{aligned} \tag{3-26}$$

得出 $V^2(e_k) - \xi^2(e_k) \ge 0$。

同理可得，假设对于任意 e_k 都满足条件 $V^j(e_k) - \xi^{j-1}(e_k) \ge 0$，对于第 $j(j \to \infty)$

次迭代时，根据 (3-24) 和 (3-25) 可以得到：

$$V^{j+1}(e_k) - \xi^j(e_k) = V^j(e_k) - \xi^{j-1}(e_k) \geq 0 \tag{3-27}$$

即 $V^{j+1}(e_k) \geq \xi^j(e_k)$。结合引理3.1和引理3.2，容易推得：

$$0 \leq V^j(e_k) \leq \xi^j(e_k) \leq V^{j+1}(e_k) \leq Y \tag{3-28}$$

综上所述，值函数序列 $V^j(e_k)$ 是单调递增且有上界的，所以它是收敛的。接着根据定义 (3-19) 和 (3-22), 得到关于 Δu_{k+1} 的收敛性条件：

$$\lim_{j\to\infty} \Delta u_{k+1}^j = \Delta u_{k+1}^* \tag{3-29}$$

证明完毕。 □

3.4 预测最优迭代学习控制 off-policy Q-学习方法

本节将基于强化 Q-学习框架，提出基于 off-policy Q-学习的预测最优迭代学习控制算法 (Off-policy Q-learning Predictive Optimal Iterative Learning Control, OPQL-POILC)，并证明该算法的收敛性。状态-动作价值函数，又称 Q 函数，意义为在当前状态下根据行为策略执行动作后的期望奖励函数。根据状态-动作价值函数的定义，Q 函数与迭代学习控制的跟踪误差 e_k 和控制更新规律相关联 Δu_{k+1} 定义如下：

$$Q(e_k, \Delta u_{k+1}) = e_k^T Q e_k + \gamma \Delta u_{k+1}^T R \Delta u_{k+1} + \gamma J(e_{k+1}) \tag{3-30}$$

将 (3-12) 代入上式，Q 函数重新表述为：

$$Q(e_k, \Delta u_{k+1}) = e_k^T Q e_k + \gamma \Delta u_{k+1}^T R \Delta u_{k+1} + \gamma e_{k+1}^T P e_{k+1} \tag{3-31}$$

根据状态转移函数 (3-8)，Q 函数可以改写为分块矩阵的形式：

$$\begin{aligned} Q(e_k, \Delta u_{k+1}) &= \begin{bmatrix} e_k \\ \Delta u_{k+1} \end{bmatrix}^T H \begin{bmatrix} e_k \\ \Delta u_{k+1} \end{bmatrix} \\ &= \begin{bmatrix} e_k \\ \Delta u_{k+1} \end{bmatrix}^T \begin{bmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{bmatrix} \begin{bmatrix} e_k \\ \Delta u_{k+1} \end{bmatrix} \end{aligned} \tag{3-32}$$

式中 $H=H^T\in\mathbb{R}^{l\times l}, l=N+N, H$ 分块矩阵为：

$$H=\begin{bmatrix}H_{11} & H_{12}\\ H_{21} & H_{22}\end{bmatrix} \tag{3-33}$$

$$\begin{aligned}H_{11}=&Q+\gamma P, H_{11}\in R^{N\times N}\\ H_{12}=&-\gamma PG, H_{12}\in R^{N\times N}\\ H_{21}=&-\gamma G^TP, H_{21}\in R^{N\times N}\\ H_{22}=&\gamma\left(R+G^TPG\right), H_{22}\in R^{N\times N}\end{aligned}$$

式 (3-32) 是 Q 函数的一般形式，矩阵 H 称为 Q 函数的核心矩阵。

对比式 (3-4) 和式 (3-30)，可以看出矩阵 H 与矩阵 P 可相互转化，因此基于性能指标函数 (3-8) 的优化问题可以转化为对 Q 函数的极小值问题, 通过求解 $\frac{\partial Q(e_k,\Delta u_{k+1})}{\partial\Delta u_{k+1}}=0$ 以及分块矩阵 H 的定义即可以求得最优迭代学习律 Δu_{k+1}^*：

$$\Delta u_{k+1}^*=-(H_{22})^{-1}H_{21}e_k \tag{3-34}$$

为便于 off-policy 理论的应用，这里将状态转移函数 (3-8) 重新改写为：

$$e_{k+1}=\left(I-GL^j\right)e_k-G\left(\Delta u_{k+1}-\Delta u_{k+1}^j\right) \tag{3-35}$$

其中，$j=1,2,...,$ 是 Q-学习的迭代索引，Δu_{k+1} 是根据应用于迭代学习的行为策略执行的行为动作，其作用是与状态转移函数 (3-35) 交互并生成 off-policy Q-学习所需的数据；此外，Δu_{k+1}^j 是 off-policy Q-学习中由不断更新的目标策略采取的目标动作，目标动作的选择满足如等式 (3-36)。

$$\Delta u_{k+1}^j=L^je_k \tag{3-36}$$

根据 Q-学习时序差分的思想，Q 函数 (3-32) 改写为：

$$Q\left(e_k,\Delta u_{k+1}\right)-\gamma e_{k+1}^TPe_{k+1}=e_k^TQe_k+\gamma\Delta u_{k+1}^TR\Delta u_{k+1} \tag{3-37}$$

基于目标最优迭代学习律 Δu_{k+1}^{j}，式 (3-37) 可表示为：

$$\begin{aligned}
&Q^{j+1}\left(e_k,\Delta u_{k+1}^{j}\right)-\gamma\left(e_k-G\Delta u_{k+1}^{j}\right)^{T}P^{j+1}\left(e_k-G\Delta u_{k+1}^{j}\right)\\
=&Q^{j+1}\left(e_k,\Delta u_{k+1}^{j}\right)-\gamma e_k^{T}\left(I-GL^{j}\right)^{T}P^{j+1}\left(I-GL^{j}\right)e_k\\
=&e_k^{T}Qek+\gamma{\Delta u_{k+1}^{j}}^{T}R\Delta u_{k+1}^{j}
\end{aligned}\tag{3-38}$$

沿着状态转移函数 (3-35) 的轨迹，推的 off-policy 形式下的 Q 函数：

$$\begin{aligned}
&Q^{j+1}\left(e_k,\Delta u_{k+1}^{j}\right)-\gamma\left(e_{k+1}+G\Delta u_{k+1}-G\Delta u_{k+1}^{j}\right)^{T}\\
&P^{j+1}\left(e_{k+1}+G\Delta u_{k+1}-G\Delta u_{k+1}^{j}\right)\\
=&Q^{j+1}\left(e_k,\Delta u_{k+1}^{j}\right)-\gamma e_{k+1}^{T}P^{j+1}e_{k+1}\\
&-2\gamma e_{k+1}^{T}p^{j+1}G\left(\Delta\mu_{k+1}-\Delta u_{k+1}^{j}\right)\\
&-\gamma\left(\Delta u_{k+1}-\Delta u_{k+1}^{j}\right)^{T}G^{T}P^{j+1}G\left(\Delta u_{k+1}-\Delta u_{k+1}^{j}\right)\\
=&Q^{j+1}\left(e_k,\Delta u_{k+1}^{j}\right)-\gamma e_{k+1}^{T}P^{j+1}e_{k+1}\\
&-2\gamma\left(\Delta u_{k+1}-L^{j}\Delta u_{k+1}\right)^{T}G^{T}P^{j+1}G\left(\Delta\mu_{k+1}-\Delta u_{k+1}^{j}\right)\\
&-\gamma\left(\Delta u_{k+1}-\Delta u_{k+1}^{j}\right)^{T}G^{T}P^{j+1}G\left(\Delta u_{k+1}-\Delta u_{k+1}^{j}\right)\\
=&e_k^{T}Qek+\gamma{\Delta u_{k+1}^{j}}^{T}R\Delta u_{k+1}^{j}
\end{aligned}\tag{3-39}$$

根据 Q 函数 (3-31)，将关系式 $Q^{j+1}\left(e_k,\Delta u_{k+1}^{j}\right)=J(e_{k+1}^{j+1})=e_k^{T}P^{j+1}e_k$ 代入上式，对等式两侧进行合并同类项，可以整理得到 Bellman 方程：

$$\begin{aligned}
&e_k^{T}P^{j+1}e_k-\gamma e_{k+1}^{T}P^{j+1}e_{k+1}-2\gamma e_k^{T}P^{j+1}G\Delta u_{k+1}\\
&+2\gamma e_k^{T}P^{j+1}G\Delta u_{k+1}^{j}+\gamma\Delta u_{k+1}^{T}G^{T}P^{j+1}G\Delta u_{k+1}\\
&-\gamma{\Delta u_{k+1}^{j}}^{T}G^{T}P^{j+1}G\Delta u_{k+1}^{j}\\
=&e_k^{T}Qek+\gamma{\Delta u_{k+1}^{j}}^{T}R\Delta u_{k+1}^{j}
\end{aligned}\tag{3-40}$$

将 $\Delta u_{k+1}^{j}=L^{j}e_k$ 代入 Bellman 方程 (3-13)，可以求出 Lyapunov 方程：

$$P^{j+1}=Q+\gamma{L^{j}}^{T}RL^{j}+\gamma(I-GL^{j})^{T}P^{j+1}(I-GL^{j})\tag{3-41}$$

将该方程代入 Bellman 方程 (3-40)，化简得到：

$$\begin{aligned}
&e_k^T Q e_k + \gamma e_k^T P^{j+1} - \gamma e_{k+1}^T Q e_{k+1} - \gamma^2 e_{k+1}^T P^{j+1} e_{k+1} \\
&- \gamma^2 \Delta u_{k+2}^j{}^T R \Delta u_{k+2}^j + 2\gamma^2 e_{k+1}^T P^{j+1} G \Delta u_{k+2}^j \\
&- \gamma^2 \Delta u_{k+2}^j{}^T G^T P^{j+1} G \Delta u_{k+2}^j - 2\gamma e_k^T P^{j+1} G \Delta u_{k+1} \\
&+ \gamma \Delta u_{k+1}^T G^T P^{j+1} G \Delta u_{k+1} \\
=&e_k^T Q e k + \gamma \Delta u_{k+1}^j{}^T R \Delta u_{k+1}^j
\end{aligned} \tag{3-42}$$

依据分块矩阵 H 的定义，可以推导出基于跟踪误差数据驱动在 off-policy 下的 Bellman 方程：

$$\begin{aligned}
&e_k^T H_{11}^{j+1} e_k - \gamma e_{k+1}^T H_{11}^{j+1} e_{k+1} + 2 e_k^T H_{12}^{j+1} G \Delta u_{k+1} \\
&- 2\gamma e_{k+1}^T H_{12}^{j+1} G \Delta u_{k+2}^j + \Delta u_{k+1}^T H_{22}^{j+1} \Delta u_{k+1} - \gamma \Delta u_{k+2}^j{}^T H_{22}^{j+1} \Delta u_{k+2}^j \\
=&e_k^T Q e k + \gamma \Delta u_{k+1}^T R \Delta u_{k+1}
\end{aligned} \tag{3-43}$$

其中，$\Delta u_{k+2}^j = L^j e_{k+1}$.

已知 Kronecker 乘积法则：

$$a^T \Pi b = (b^T \otimes a^T) vec(\Pi) \tag{3-44}$$

法则中向量 $a \in \mathbb{R}^{n_a}$，向量 $b \in \mathbb{R}^{n_b}$，矩阵 $\Pi \in \mathbb{R}^{n_a \times n_b}$。根据 Kronecker 乘积法则 (3-44)，Bellman 方程 (3-43) 可以在 Kronecker 乘积的形式下重新表述为：

$$\begin{aligned}
&(e_k^T \otimes e_k^T - \gamma e_{k+1}^T \otimes e_{k+1}^T) vec(H_{11}^{j+1}) \\
&+ (\Delta u_{k+1}^T \otimes e_k^T - \gamma \Delta u_{k+2}^j{}^T \otimes e_{k+1}^T) vec(H_{12}^{j+1}) \\
&+ (\Delta u_{k+1}^T \otimes \Delta u_{k+1}^T - \gamma \Delta u_{k+2}^j{}^T \otimes \Delta u_{k+2}^j{}^T) vec(H_{22}^{j+1}) \\
=&e_k^T Q e k + \gamma \Delta u_{k+1}^T R \Delta u_{k+1}
\end{aligned} \tag{3-45}$$

为了求得 off-policy 下 Bellman 方程中分块矩阵 H 的各个参数块，进而获得第 j 次迭代的学习律，因此需要将式 (3-45) 中的方程进行参数化线性，以便分离参数矩阵。令：

$$\Phi^{j^T} \bar{H}^{j+1} = \Upsilon^j \tag{3-46}$$

其中，向量 $\bar{H}^{j+1} \in \mathbb{R}^{N(2N+1)}$

$$\begin{aligned}\bar{H}^{j+1} &\triangleq \left[\text{vec}\left(H_{11}^{j+1}\right);\text{vec}\left(H_{12}^{j+1}\right);\text{vec}\left(H_{22}^{j+1}\right)\right] \\ &= \left[h_{11}, 2h_{12}, \ldots, 2h_{12N}, h_{22}, 2h_{23}, \ldots, 2h_{22N}, \ldots, h_{2N2N}\right]^T\end{aligned} \tag{3-47}$$

其中，$\bar{H}^{j+1}$ 建立在当前由行为策略和目标策略求得的迭代学习律 Δu_{k+1} 下的第 j+1 次基于跟踪误差数据驱动的 off-policy Q 函数的估计上。$h^{ij}(i;j=1,...,2N)$ 是矩阵 h 的第 i 行第 j 列的元素，数据矩阵 Φ^{j^T} 和数据向量 Υ^j 表示：

$$\Phi^j = \begin{bmatrix} \phi_k^j \\ \phi_{k+1}^j \\ \vdots \\ \phi_{k+L-1}^j \end{bmatrix}^T \quad \Upsilon^j = \begin{bmatrix} f_k^j \\ f_{k+1}^j \\ \vdots \\ f_{k+L-1}^j \end{bmatrix} \tag{3-48}$$

其中，$\Phi^j \in \mathbb{R}^{N(2N+1)\times L}$, $\Upsilon^j \in \mathbb{R}^{L\times 1}$

$$f_k^j = e_k^T Q e k + \gamma \Delta u_{k+1}^T R \Delta u_{k+1} \tag{3-49}$$

并且有 $\phi_k^j = \left[\varphi_1^j, \varphi_2^j, \varphi_3^j\right]$

$$\begin{aligned}\varphi_1^j &= e_k^T \otimes e_k^T - \gamma e_{k+1}^T \otimes e_{k+1}^T \\ \varphi_2^j &= \Delta u_{k+1}^T \otimes e_k^T - \gamma \Delta {u_{k+2}^j}^T \otimes e_{k+1}^T \\ \varphi_3^j &= \Delta u_{k+1}^T H_{22}^{j+1} \Delta u_{k+1} - \gamma \Delta {u_{k+2}^j}^T H_{22}^{j+1} \Delta u_{k+2}^j\end{aligned} \tag{3-50}$$

通过求解线性参数化方程中的向量 $\bar{H}$，可以得到基于跟踪误差数据驱动在 off-policy 下的 Bellman 方程 (3-45) 中的分块矩阵。在当前迭代学习律 Δu_{k+1} 下，基于跟踪误差数据驱动的 off-policy$Q$ 函数的目标迭代学习律可由式 (3-34) 求解。

根据最小二乘法理论，由于方程 (3-46) 存在 $N(2N+1)$ 个独立元素，所以需要至少收集 $N(2N+1)$ 个数据样本以组成数据矩阵 Φ^{j^T} 和数据向量 Υ^j；该方程需要满足如式 (3-51) 矩阵秩的条件。

$$rank(\Phi^j) = N(2N+1) \tag{3-51}$$

下面将给出算法 3-2 的伪代码和算法流程图3-1。

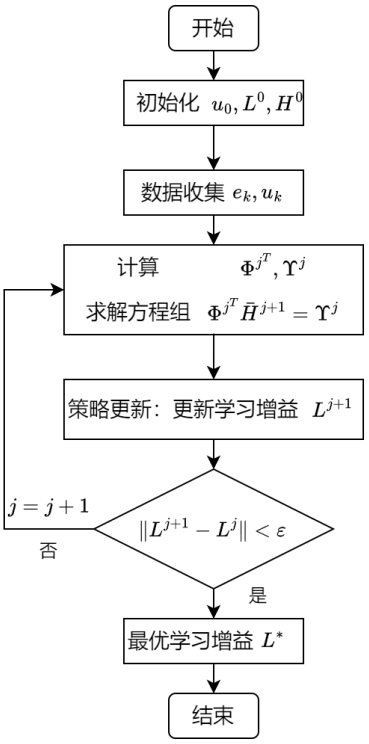

图 3-1 算法 3-2 的流程图

算法 3 - 2: 基于 off-policy Q-学习的预测最优迭代学习控制

系统数据: 误差 e_k，参考轨迹 y_d

初始化: 初始可行的迭代学习律 Δu_{k+1}、初始 Q 函数核心矩阵 H_0 和初始学习目标策略增益矩阵 L_0；令 $k = 1, 2, ..., m, m = N(2N+1)$，$j = 1, 2, ...,$

输出: 最优学习增益 L^*

开始循环 $k = 1 \to m$

> **数据收集** *收集迭代学习运行的状态信息误差 e_k 和计算输入增量 Δu_{k+1}*
>
> k=k+1

结束循环

开始循环 $j = 0 \to \infty$

> **数据准备** *计算行为策略下执行的动作 Δu_{k+2}^j，以及数据集 Φ^{j^T} 和 Υ^j。*
>
> **策略评估** *通过式 (3-46) 求解得到 $\bar{H}$。*
>
> **策略更新** *通过式 (3-36) 更新目标策略，即学习增益矩阵 L^{j+1}。*
>
> j=j+1
>
> **终止更新条件** *若相邻两次的学习增益矩阵 L_j 和 L_{j+1}，对任意的 ε，都满足 $\|L^{j+1} - L^j\| < \varepsilon$, ε 是某一小正数，停止迭代*

结束循环

综上所述，完成了基于 off-policy Q-学习的预测最优迭代学习控制算法（算法 3-2）设计。算法 3-2 依据 off-policy 形式的 Bellman 方程 (3-45) 进行策略评估并求解出 $\bar{H}$，通过 $\bar{H}$ 的各分块矩阵参数，进一步求出在策略更新步骤中的学习增益矩阵 L^{j+1}，目标策略的增益矩阵 L^{j+1} 会在第 $j+1$ 次 Q-学习中被估计。当目标策略随着迭代逼近理想的学习增益 L^* 时，停止更新目标策略 (3-34) 且迭代结束。本章剩余部分给出算法 3-2 的收敛性证明。

定理 3.4: 当 $\Delta u_{k+1}^j = L^j e_k$ 已知的时候，方程 (3-46) 的解等价于基于同轨策略的 Q-学习方程的解。

证明: 由上述推导过程可知，基于跟踪误差数据驱动在 off-policy 下的 Bellman 方程等价于 Q 函数 (3-27)。文献 [69] 中提出了基于同轨策略下 Q-学习的预测最优迭代学习控制算法，研究成果表明基于同轨策略的 Bellman 方程也等价于 Q 函数。因此，当式子 $\Delta u_{k+1}^j = L^j e_k$ 完全已知时，off-policy 下的 Q-学习问题可以转化为基于同轨策略下 Q-学习问题。另一方面，通过参数线性化 Q 函数 (3-27) 的参数，考虑同轨策略下 Q-学习策略评估方程，可以得到同轨策略下 Q-学习策略评估方程

$$(Z_k - \gamma Z_{k+1})\,\bar{H}^{j+1} = e_k^T Q e_k + \gamma \Delta u_{k+1}^T R \Delta u_{k+1} \tag{3-52}$$

其中

$$Z_k = \left[e_k^T, \Delta u_{k+1}^T\right] \otimes \left[e_k^T, \Delta u_{k+1}^T\right] \tag{3-53}$$

不难看出，两种算法其策略评估的核心方程是等价的，因此两种算法得到的控制策略也是等价的。这就完成了证明。 □

定理 3.5: (算法 3-2 收敛性) 当 Q-学习的迭代索引 $j \to \infty$ 基于跟踪误差数据驱动在 off-policy 下 Q-学习算法的学习增益 L^{j+1} 将收敛到最优学习增益 L^*。

证明: 根据文献 [70]，可以证明当 $j \to \infty$ 时，同轨策略下 Q-学习算法将收敛于最优学习增益 L^*；由定理 3.4 可知，off-policy 下的 Bellman 方程与同轨策略下的 Bellman 方程是等价的。因此，算法 3-2 中策略更新的目标策略 L^{j+1} 最终也将随着 j 迭代逼近收敛到理想的控制策略，即最优学习增益 L^*。证明完毕。 □

3.5 数值仿真

在本节中，将针对三个不同的被控系统分别使用四种不同的迭代学习算法进行仿真，系统之间只有一个系统矩阵 a_{21} 的元素不同，用来模拟实际应用中被控对象模型的不确定性。最后，通过对比不同迭代学习控制算法结果，验证了算法 3-1 和算法 3-2 的收敛性和有效性。

考虑下面的 SISO 离散时间系统：

$$\begin{aligned} x_k(t+1) &= \begin{bmatrix} -0.9 & -0.2 \\ a_{21} & 0 \end{bmatrix} x_k(t) + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u_k(t) \\ y_k(t) &= \begin{bmatrix} 0.8 & 0.24 \end{bmatrix} x_k(t) \end{aligned} \tag{3-54}$$

被控系统 (System1、System2 和 System3) 的中元素 a_{21} 分别取 1 , 0.5 和 0.1。系统初始状态 $x_k(0) = [0\ 0]^T$ ，给定期望轨迹：

$$r(t) = 3\sin(0.5t), \quad t \in [0, 5] \tag{3-55}$$

其中，I 为单位矩阵; 参数矩阵 Q 和 R 分别取值为 I 和 50 I; 采样时间 $T_s = 1s$; 权重因子 $\gamma = 0.994$; 初始迭代学习律选取合适的 P 型更新律，如 $\Delta u_{k+1} = 0.8e_k$; 初始学习增益矩阵 L^0 是随机的，满足迭代学习控制的收敛条件; 收集满足条件 (3-41) 数量的数据用于更新 L^{j+1}。

为了验证算法 3-2 的有效性，即在不依赖系统模型参数信息时是否以更快的收敛速度实现零误差地完全跟踪期望轨迹，下面将采取四种不同迭代学习控制方法进行仿真对比：分别是基于模型的算法 3-1(VI-POILC) 和范数最优迭代学习控制 (Norm optimal Iterative Learning Control, NOILC)，还有不依赖系统动力学模型信息的方法 P 型迭代学习控制 (P-ILC) 以及算法 3-2(OPQL-POILC)。根据范数最优迭代学习控制理论[49]，其迭代学习律为：

$$u_{k+1} = u_k + R^{-1}G^T Q e_{k+1} \tag{3-56}$$

其中，矩阵 Q 和 R 分别被赋值为 $3I$ 和 I。当系统参数 $a_{21} = 1$，则可以计算出当前系统的范数最优学习律，并实施迭代学习控制。然而，当系统模型由于不确定性因素导致参数发生变动，例如 a_{21} 变为 0.5 或 0.1 时，由于不完全掌握当前系统动态信息，范数最优学习律无法依据新的系统模型参数做出相应的调整。此时，算法 3-2 以及 P 型迭代学

习控制的优势便得以体现，因为它们无需依赖精确的系统模型参数信息，即使在模型存在不确定性的情况下也能继续有效运行。此外，为了验证算法 3-2 的收敛性，我们将针对上述三种不同的系统模型参数（包括 $a_{21}=1, a_{21}=0.5$ 和 $a_{21}=0.1$），利用算法 3-1 基于每个模型状态选取最优的迭代学习律进行仿真，并通过对比这两种算法在不同模型条件下的收敛特性来获取结论。

在图3-2所展示的结果中，考察了当被控系统的参数 a_{21} 取不同数值（分别为 1 , 0.5 和 0.1）时，算法 3-2 所对应的控制学习增益随迭代过程的变化情况及其收敛特性。随着迭代次数指数 j 的不断增大，算法 3-2 生成的目标策略 L^j 逐步逼近最优策略，并最终稳定收敛到最优学习增益 L^*。这一现象揭示了当系统参数变化时，算法 3-2 能在迭代过程中有效地寻找到适应系统动态特性的最佳控制策略，且收敛性表现良好，确保系统在多次迭代后达到期望的控制效果。

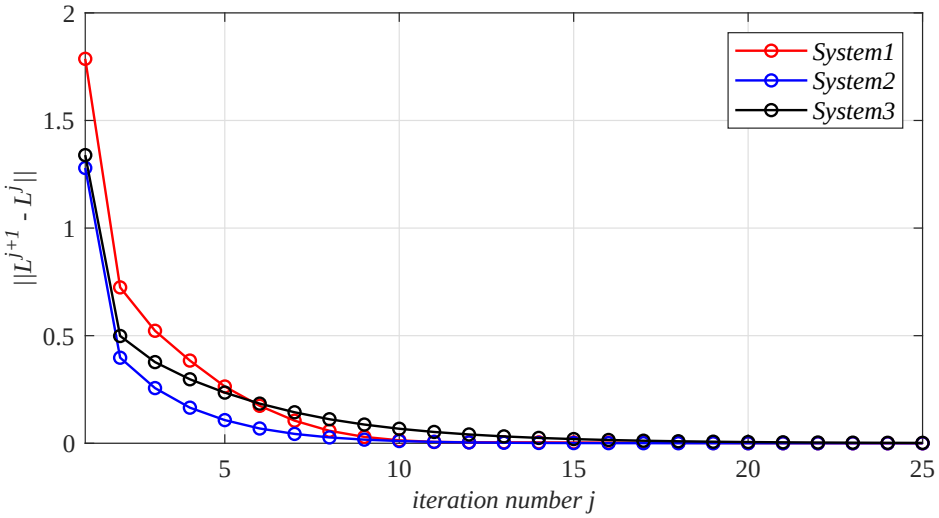

图 3-2 学习增益矩阵 L 的收敛性

从图3-3、图3-4和图3-5中对四种控制策略的纵向对比可以得出，随着系统模型参数的改变，基于模型的范数最优迭代学习控制算法的收敛性能呈现下降趋势，尤其是在系统参数 $a_{21}=0.1$ 时，其需要经过 20 次迭代才逐渐趋于收敛，这表明范数最优迭代学习控制算法在处理变参数系统时的表现并不理想。相反，不依赖系统模型参数信息的算法 3-2 与 P 型迭代学习控制仍能保持相对快速的收敛速度。另外，通过横向对比不同算法下的仿真结果，可以明显观察到，在模型参数变化的情况下，算法 3-2 的收敛性能要优于范数最优迭代学习控制算法，从而有力地证明了算法 3-2 的有效性。与此同时，尽管算法 3-2 相较于算法 3-1 在收敛特性上有所不足，但也进一步证实了依赖模型参数信息

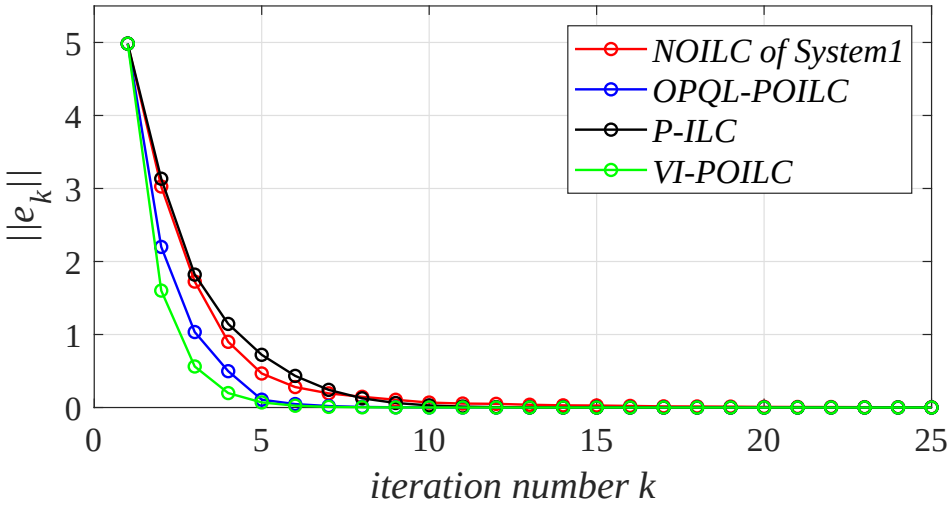

图 3-3 系统 1($a_{21} = 1$) 中四种迭代学习控制方法的误差曲线图

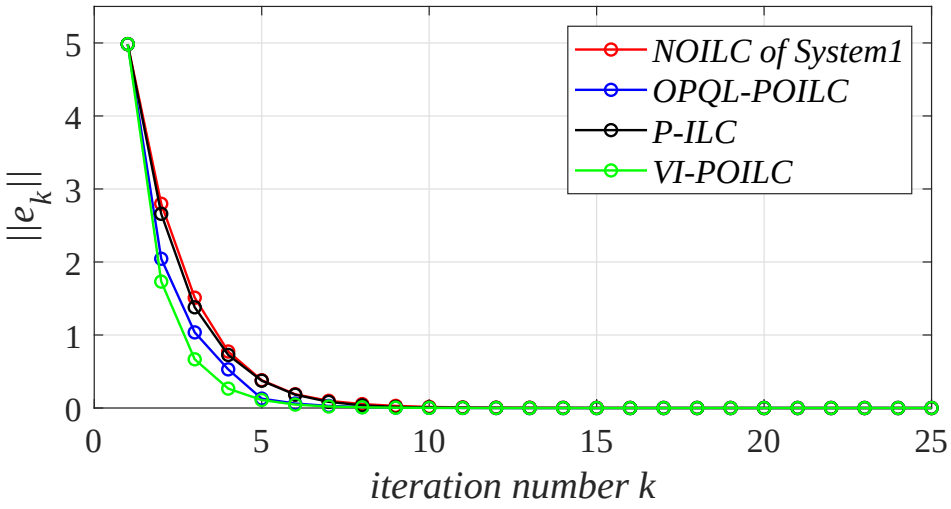

图 3-4 系统 2($a_{21} = 0.5$) 中四种迭代学习控制方法的误差曲线图

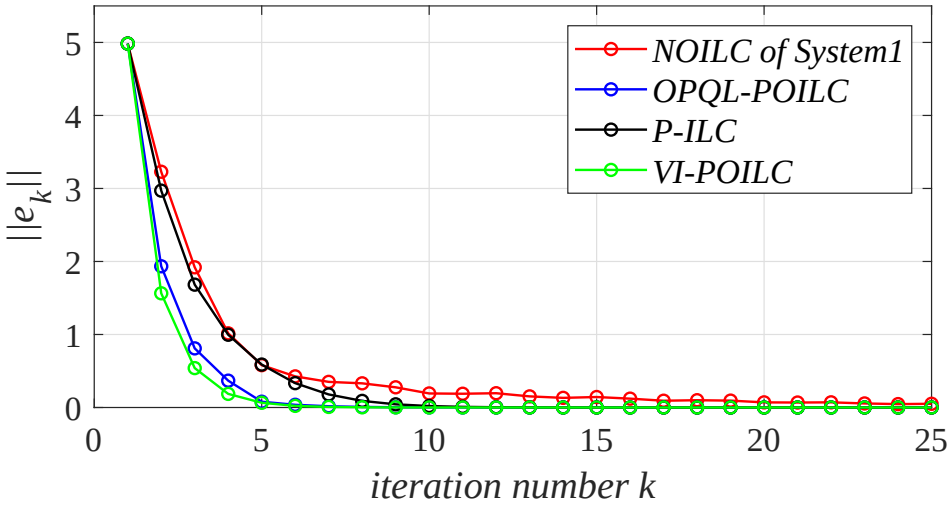

图 3-5 系统 3($a_{21} = 0.1$) 中四种迭代学习控制方法的误差曲线图

的控制方法往往能够比不依赖模型参数信息的方法实现更快的收敛速度这一普遍规律。

3.6 本章小结

针对如何在确保系统具备良好收敛性能的前提下，摆脱对系统精确模型参数的依赖来设计迭代学习控制策略这一问题，本章从迭代学习与强化学习的内在理论联系出发，选取预测最优迭代学习控制作为研究背景，首先提出了一种基于模型的值迭代预测最优迭代学习设计方法，并通过理论分析，证明了值迭代算法的收敛性。随后，在已构建的理论基础上，进一步引入了 Q 函数概念，将迭代学习中的最优性问题转换为寻找 Q 函数的极值问题。结合数学推导，提出了基于数据驱动 off-policy 下 Q -学习的预测最优迭代学习控制算法。算法 3-2 可以在系统模型参数信息完全未知情况下达到对最优学习增益的收敛，并提供收敛性的证明。最后通过数值仿真验证了算法的收敛性和有效性。

第四章　基于 on-policy 策略梯度的参数最优迭代学习控制方法

4.1　引言

在上一章中提出一种基于 Q 函数的预测最优迭代学习控制，该方法是基于动作价值函数的强化学习方法，它首先根据动作价值函数维护一个 Q 表，然后选择表中 Q 值最大的作为动作。然而 Q-学习应用与控制领域中存在收敛速度慢、对动作空间的离散化要求高还有易陷入局部最优解等问题[71]，这要求我们寻求一种更加灵活、高效的控制方法，能够克服 Q-学习的局限性，同时能够直接学习动作策略而非值函数，以提高学习速度和泛化性能。

策略梯度算法不依赖于动作价值函数，也不再依赖于价值函数选择动作, 而是通过直接学习参数化策略 π 的方法，再与环境交互选择相应的动作。策略梯度算法恰好弥补了 Q-学习上述的几个局限问题，这种方法通过直接对策略函数进行优化逼近，使得在连续动作空间或者复杂高维状态空间的环境中依旧能够迅速找到最优策略[72]，不再需要对状态空间或者动作空间进行繁琐的离散化处理。此外，策略梯度算法更有可能避免陷入局部最优解，从而在解决控制问题时，提高了探寻全局最优策略的性能和可靠性。

本章基于参数最优迭代学习控制理论，进一步研究了不使用显式显式模型参数信息的参数最优迭代学习律设计问题。该方法具有结构简单和收敛特性良好的特点。首先，介绍了具有参数 γ 的性能指标函数，阐明了本章参数最优迭代学习控制的背景，并给出了该问题的解。其次，根据强化学习的策略梯度理论来解决数据驱动参数最优迭代学习律设计问题，给出了关于待更新策略 $\boldsymbol{\pi}$ 的参数化方法，提出了基于 on-policy 在线学习策略梯度的参数最优迭代学习控制方法，给出了算法满足收敛性的条件。然后，为了改进算法的稳定性和收敛速度，利用近端策略优化的理论，并引入价值函数预测作为基线，在“行动器-评判器”的算法框架下提出了本章第二个算法，即基于近端策略优化策略梯度的参数最优迭代学习控制算法。最后，通过数值仿真比较了不同方法的性能，证明了所提算法的有效性。

4.2 节描述了一个高性能跟踪问题，介绍了一个参数最优迭代学习控制问题框架。4.3 节在策略梯度方法上，提出了基于 on-policy 策略梯度方法的参数最优迭代学习控制算法，介绍了如何利用策略梯度方法来解决参数更新的问题，并讨论了算法的收敛性和

优势。4.4 节基于近端策略优化和价值函数预测，在“行动器-评判器”的算法框架下提出了一种改进的参数最优迭代学习控制算法，描述了如何引入近端策略优化方法和添加基线来提高算法的稳定性和收敛速度，并讨论了算法的有效性。4.5 节中，我们通过仿真实验验证了所提出算法的有效性，并与基于模型的参数最优迭代学习控制进行了性能比较。最后，4.6 节对本章内容进行总结。

4.2 高性能跟踪控制：参数最优迭代学习控制

在本节内容中，将详细介绍高性能跟踪控制的一种解决方案——参数最优迭代学习控制，进一步回顾如何通过该算法来有效地求解出最优迭代学习律；然后延续上一章节的研究思路，将迭代学习问题重新在马尔科夫框架下建立以，方便后续与强化学习算法的结合。

4.2.1 参数最优迭代学习控制

迭代学习控制在实际应用中通过与系统模型不断交互，根据过去的信息中学习经验来调整学习律，最终系统输出收敛于参考轨迹，实现高性能跟踪控制这一目的。为了解决上述问题，研究人员已经在文献中提出了许多迭代学习控制设计算法。在本节中，将参数最优迭代学习控制算法作为研究背景，该算法的学习律:

$$u_{k+1} = u_k + \gamma_{k+1} e_k \tag{4-1}$$

其中，学习增益 γ_{k+1} 的选择是通过最小化以下性能指标函数 (4-2) 来进行的。

$$J_{k+1}(\gamma_{k+1}) = \|e_{k+1}\|^2 + \omega \gamma_{k+1}^2 \tag{4-2}$$

其中，$\omega > 0$ 是学习增益 γ 一个小的加权标量。如果已知系统输入输出映射矩阵 G，则基于上式定义下的迭代学习解为:

$$\gamma_{k+1}^* = \frac{e_k^T G e_k}{\omega + e_k^T G^T G e_k} \tag{4-3}$$

该算法具有很好的收敛特性，可以加快迭代学习的收敛速度。根据文献 [73]，参数最优迭代学习控制的收敛性满足下定理。

定理 4.1: 根据式 (4-1) 和式 (4-2) 所定义的参数最优迭代学习控制在跟踪误差范数上满足单调收敛条件:

$$\|e_{k+1}\| \leq \|e_k\|, \; k = 0, 1, 2, ... \tag{4-4}$$

学习增益最终也会趋近于 0，故有：

$$\lim_{k\to\infty} \gamma_{k+1}^{*} = 0 \tag{4-5}$$

因此当系统输入输出矩阵 G 满足 $G + G^T > 0$ 条件，参数最优迭代学习将达到完全跟踪，即 $\lim_{k\to\infty} e_k = 0$ 。

通过式 (4-3) 可以看出，该算法在系统模型参数已知时，可以计算出具有良好收敛性的的最优控制学习增益; 然而，当系统模型不可用时，如何寻找参数最优迭代学习控制的最优增益将是一个挑战，这将在下节中探讨。

4.2.2 马尔科夫框架下的参数最优迭代学习控制

根据第二章的预备知识，一个参数最优迭代学习控制过程被重新构建为一个 MDP，定义如下：

• S 是状态空间，状态 s 满足 $s \in S$。在迭代学习中，状态是第 k 次迭代学习的误差 $e_k \in \mathbb{R}^N$ N 为实验长度。

• A 是动作空间，动作 a 满足 $a \in A$。强化学习中动作 a 定义为第 k 次迭代学习的输入增量 γ_{k+1}。

• $f\left(e_k, \Delta u_{k+1}\right)$ 是状态转移函数。状态转移函数在本章迭代学习控制理论中依旧满足如下：

$$e_{k+1} = f\left(e_k, \Delta u_{k+1}\right) = e_k - G\Delta u_{k+1} \tag{4-6}$$

• R_k 是奖励函数。在第 k 次迭代学习状态 e_k 下奖励函数满足如下定义

$$\begin{aligned} R(e_k) =& \|e_{k+1}\|^2 + \omega\gamma_{k+1}^2 \\ =& e_k^T(I - \gamma_{k+1}G^T)(I - \gamma_{k+1}G)e_k + \omega\gamma_{k+1}^2 \end{aligned} \tag{4-7}$$

• G_k 是折扣回报函数。G_k 表示在迭代域内 k 的折扣回报，是指代理在执行一次动作之后所能获得的未来奖励的总和。定义如下：

$$G_k = R_k + \eta R_{k+1} + \eta^2 R_{k+2} + \ldots = \sum_{k=0}^{\infty} \eta^k R_k \tag{4-8}$$

其中，$0 \leq \eta \leq 1$ ，是折扣因子，表示了 η 未来奖励的重要性衰减率，通常情况下，较大的 η 表示更加重视未来奖励，而较小的 γ 则相对更加重视即时奖励。在参数最优迭代学习控制问题中,$\eta = 0$。

本章中强化学习的目标是通过通过最小化折扣回报函数，进而寻找最优的策略 $\boldsymbol{\pi}^*$，根据当前状态采取动作，即 $\gamma_{k+1}=\boldsymbol{\pi}^*(e_k)$。在下一小节中，将讨论如何衡量策略 $\boldsymbol{\pi}$ 和如何对策略进行优化。

4.3 基于 on-policy 策略梯度的参数最优迭代学习控制

本节将对参数最优迭代学习控制方法引入第二章的策略梯度算法，首先给出了策略参数化的方法，推导了参数最优迭代学习控制下的策略梯度更新式，然后提出一种不依赖系统模型参数信息的策略梯度参数最优迭代学习控制，并给出了算法收敛性满足的条件。

在参数最优迭代学习控制中，因为动作空间是连续的，并且是一个实数集，因此在本章中不直接计算每一个动作的概率，而是通过概率密度分布函数来衡量策略，通过对参数化概率密度分布函数的学习，最终得到最优策略。参数化策略函数 $\boldsymbol{\pi}(\gamma_{k+1}|e_k,\boldsymbol{\theta_k}=\boldsymbol{\theta})$ 可以用正态分布表示：

$$\boldsymbol{\pi}(\gamma_{k+1}|e_k,\boldsymbol{\theta})=\frac{1}{\sqrt{2\pi}\sigma(e_k,\boldsymbol{\theta})}\exp\left(-\frac{[\gamma_{k+1}-\mu(e_k,\boldsymbol{\theta})]^2}{2\sigma^2(e_k,\boldsymbol{\theta})}\right) \tag{4-9}$$

其中，$\mu(e_k,\boldsymbol{\theta})\in\mathbb{R}$ 表示在状态 e_k 下策略 $\boldsymbol{\pi}$ 的均值，$\sigma(e_k,\boldsymbol{\theta})\in\mathbb{R}^+$ 是在相同状态下策略 $\boldsymbol{\pi}$ 的标准差，$\boldsymbol{\theta}$ 是策略的参数向量，可以划分为两个部分 $\boldsymbol{\theta}=[\theta_\mu,\theta_\sigma]^T$，一部分用来近似均值 μ，另一部分用来近似标准差 σ。

将均值通过线性函数来近似具有结构简单且有利于策略梯度方法的进一步推导的特点，并提供良好的收敛保证；根据强化学习的线性特征构造理论，可以将待近似值表示为一类基函数的线性组合形式，这些基函数常用包括多项式基、傅立叶基、径向基等。在本章中，关于均值 μ 的线性构造方法的基函数定义为：

$$\mu(\theta_\mu,e_k)=\theta_\mu^T z_\mu=\theta_\mu^T[z_{\mu,1}(e_k),z_{\mu,2}(e_k),...,z_{\mu,h}(e_k)] \tag{4-10}$$

其中，$h\geq 1$，表示近似基函数向量的维度，基函数向量的元素 $\{z_{\mu,i}(e_k)|i=1,2,...h\}$ 定义为：

$$z_{\mu,i}(e_k)=\exp(-\frac{a_i\|e_k\|^2}{2}) \tag{4-11}$$

a_i 是一个放缩系数。对于标准差 σ 的参数化，可以将 $e_k^Te_k$ 作为衡量期望 μ 与最优参数

之间的距离，期望较小的误差标准差也应该较小，故 σ 可选为：

$$\sigma(e_k,\theta_\sigma)=\exp(-\frac{\theta_\sigma}{\|e_k\|^2})=\exp(\theta_\sigma z_\sigma(e_k)) \tag{4-12}$$

其中，$\theta_\sigma>0$，$z_\sigma(e_k)$ 是对 σ 近似的函数逼近器，$z_\sigma(e_k)$ 满足如等式 (4-13) 所示。

$$z_\sigma(e_k)=-\frac{1}{\|e_k\|^2} \tag{4-13}$$

在参数最优迭代学习控制中，基于参数化策略函数 (4-9)，定义衡量策略参数 $\boldsymbol{\theta}$ 优劣性的性能指标为：

$$J(\boldsymbol{\pi}(\gamma_{k+1}|e_k,\boldsymbol{\theta}))=\mathbb{E}_{\boldsymbol{\pi}}[R_k(\boldsymbol{\theta})]=\mathbb{E}_{\boldsymbol{\pi}}[G_k(\boldsymbol{\theta})] \tag{4-14}$$

相应的，上式关于策略参数的梯度根据第二章的式 (2-27) 实例化为：

$$\nabla J(\boldsymbol{\theta})=\mathbb{E}_{\boldsymbol{\pi}}\left[G_k\frac{\nabla\boldsymbol{\pi}(\gamma_{k+1}|e_k,\boldsymbol{\theta})}{\boldsymbol{\pi}(\gamma_{k+1}|e_k,\boldsymbol{\theta})}\right] \tag{4-15}$$

在本章中，策略函数 (4-9) 对于给定的状态 e_k 下都会输出每个动作 γ_{k+1} 的概率分布，我们可以通过选取每次迭代实验具有最高概率的动作 γ_{k+1} 来设计参数最优迭代学习律 (4-1)。根据式 (2-15) 选取最优动作，上式梯度可以化简为：

$$\nabla J(\boldsymbol{\theta})=G_k\frac{\nabla\boldsymbol{\pi}(\gamma_{k+1}|e_k,\boldsymbol{\theta})}{\boldsymbol{\pi}(\gamma_{k+1}|e_k,\boldsymbol{\theta})} \tag{4-16}$$

本章的目标是通过求解最小化奖励函数来得到最优的策略参数 $\boldsymbol{\theta}$，基于上式策略参数更新式 (2-28) 将被重新改写为梯度下降的形式：

$$\boldsymbol{\theta}_{k+1}=\boldsymbol{\theta}_k-\alpha G_k\frac{\nabla\boldsymbol{\pi}(\gamma_{k+1}|e_k,\boldsymbol{\theta})}{\boldsymbol{\pi}(\gamma_{k+1}|e_k,\boldsymbol{\theta})} \tag{4-17}$$

存在等式 $\nabla\ln x=\frac{\nabla x}{x}$ 成立，上式可以用一种紧凑的形式表达：

$$\boldsymbol{\theta}_{k+1}=\boldsymbol{\theta}_k-\alpha_k G_k\nabla\ln\boldsymbol{\pi}(\gamma_{k+1}|e_k,\boldsymbol{\theta}) \tag{4-18}$$

式 (4-18) 为本节迭代学习控制下策略梯度更新策略参数的方法。其中，$\alpha_k>0$ 是步长，$\nabla\ln\boldsymbol{\pi}(\gamma_{k+1}|e_k,\boldsymbol{\theta})$ 表示策略 $\boldsymbol{\pi}$ 对策略参数 $\boldsymbol{\theta}$ 的梯度；基于上述逼近器的选择，梯度可以通过以下式计算：

$$\begin{aligned}\nabla_{\theta_\mu}\ln\boldsymbol{\pi}(\gamma_{k+1}|e_k,\theta_\mu)&=\frac{\gamma_{k+1}-\mu(e_k,\theta_\mu)}{\sigma(e_k,\theta_\sigma)^2}z_\mu(e_k)\\ \nabla_{\theta_\sigma}\ln\boldsymbol{\pi}(\gamma_{k+1}|e_k,\theta_\sigma)&=\left(\frac{(\gamma_{k+1}-\mu(e_k,\theta_\mu))^2}{\sigma(e_k,\theta_\sigma)^2}-1\right)z_\sigma(e_k)\end{aligned} \tag{4-19}$$

由此，得到了 on-policy 策略梯度的参数最优迭代学习控制算法，即算法 4-1，下面给出算法的伪代码。

算法 4 - 1: on-policy 策略梯度的参数最优迭代学习控制算法

初始化: 初始输入 u_0，策略参数向量 $\boldsymbol{\theta}_0$，最大迭代试验数 m ，开始迭代。

输出: 下一次迭代实验的输入 u_{k+1}

开始循环 $k=1 \rightarrow m$

将 u_k 输入到系统中进行一次迭代实验，收集迭代学习运行状态信息误差 e_k

计算策略参数 $\mu(e_k,\theta_u)$ 和 $\sigma(e_k,\theta_\sigma)$

根据式 (4-9) 选择动作 γ_{k+1}

代入参数最优迭代学习律 (4-1) 更新下次迭代实验输入

通过更新式 (4-18) 和式 (4-19) 求出当前状态下的策略梯度更新策略参数 $\boldsymbol{\theta}_{k+1}$

k=k+1

结束循环

根据文献 [74]，算法 4-1 达到收敛时需要满足如下定理4.2。

定理 4.2: 假设奖励函数 R、策略 $\boldsymbol{\pi}(\boldsymbol{\theta})$ 和式 (4-18) 中的学习步长序列 $\{\alpha_k\}_{k=0}^{\infty}$ 分别满足以下条件:

(1) 奖励函数 R_k 在任何状态和行为下都是有界的，即 $R(e,\gamma)\leq B_R$

(2) 参数化策略 $\boldsymbol{\pi}(\boldsymbol{\theta})$ 是一个关于策略参数 $\boldsymbol{\theta}$ 的可微的策略，并且其梯度 $\nabla \ln \boldsymbol{\pi}(\gamma_{k+1}|e_k,\boldsymbol{\theta})$ 也是有界的，还同时符合 L-Lipschitz 条件。即对于任意的 $e,\gamma,\theta,\theta_p,\theta_q$，都存在常数 $\boldsymbol{B},\boldsymbol{L}$，满足数学关系 $\|\nabla \ln \boldsymbol{\pi}(\gamma|e,\theta)\| \leq \boldsymbol{B}$ 和 $\|\nabla_\theta \ln \boldsymbol{\pi}(\gamma|e,\theta_p)-\nabla_\theta \ln \boldsymbol{\pi}(\gamma|e,\theta_q)\|\leq \boldsymbol{L}\|\theta_p-\theta_q\|$。

(3) 学习步长序列 $\{\alpha_k\}_{k=0}^{\infty}$ 满足条件 $\lim_{k\rightarrow\infty}\{\alpha_k\}=0$ 、$\sum_{k=0}^{\infty}\alpha_k=\infty$ 和 $\sum_{k=0}^{\infty}\alpha_k^2<\infty$。

然后，性能指标 $J(k)_{k=0}^{\infty}$ 从任意 $\boldsymbol{\theta}_0$ 开始，遵循策略梯度参数更新公式 (4-29)，最终性能指标收敛，满足 $\lim_{k\rightarrow\infty}\frac{\partial \boldsymbol{J}(\theta_k)}{\partial\theta}=0$。

4.4 基于“行动器-评判器”框架下 on-policy 策略梯度方法

本节在“行动器-评判器”框架下设计参数最优迭代学习控制算法。首先通过对式 (4-15) 引入基线改进策略梯度算法，随后利用随机梯度法逼近状态价值函数设计评判器；

接着引入近端策略优化设计算法的行动器，利用重要性比率以限制每次策略参数更新都不会使策略变化过大，最后分析了参数最优迭代学习控制算法的实现流程。

4.4.1 评判器设计

在参数最优迭代学习控制中，根据策略梯度定理2.1，式 (2-24) 中改写为如下等式：

$$\nabla J(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{\pi}} \left[\sum_{\gamma_{k+1}} q_{\boldsymbol{\pi}}(\gamma_{k+1}|e_k, \boldsymbol{\theta}) \nabla \boldsymbol{\pi}(\gamma_{k+1}|e_k, \boldsymbol{\theta}) \right] \tag{4-20}$$

将值函数基线 $b(e_k)$ 引入上式 (4-20) 中进行推广：

$$\nabla J(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{\pi}} \left\{ \sum_{\gamma_{k+1}} [q_{\boldsymbol{\pi}}(\gamma_{k+1}|e_k, \boldsymbol{\theta}) - b(e_k)] \nabla \boldsymbol{\pi}(\gamma_{k+1}|e_k, \boldsymbol{\theta}) \right\} \tag{4-21}$$

式 (4-21) 的基线 $b(e_k)$ 可以是任意的函数，唯一需要满足的条件是基线的值与动作 γ_{k+1} 的变化无关，此时存在如下等式 (4-22) 成立。

$$\sum_{\gamma_{k+1}} b(e_k) \nabla \boldsymbol{\pi}(\gamma_{k+1}|e_k, \boldsymbol{\theta}) = b(e_k) \nabla \sum_{\gamma_{k+1}} \boldsymbol{\pi}(\gamma_{k+1}|e_k, \boldsymbol{\theta}) = b(e_k) \nabla 1 = 0 \tag{4-22}$$

这样式 (4-15) 可以改写为一个包含值函数基线 $b(e_k)$ 的形式：

$$\begin{aligned} \nabla J(\boldsymbol{\theta}) =& \mathbb{E}_{\boldsymbol{\pi}} \left\{ [G_k - b(e_k)] \frac{\nabla \boldsymbol{\pi}(\gamma_{k+1}|e_k, \boldsymbol{\theta})}{\boldsymbol{\pi}(\gamma_{k+1}|e_k, \boldsymbol{\theta})} \right\} \\ =& [G_k - b(e_k)] \frac{\nabla \boldsymbol{\pi}(\gamma_{k+1}|e_k, \boldsymbol{\theta})}{\boldsymbol{\pi}(\gamma_{k+1}|e_k, \boldsymbol{\theta})} \end{aligned} \tag{4-23}$$

在赌博机算法中[75]，基线是一个值，但是在其马尔科夫决策过程下基线应该随着状态变化而改变，例如在某些状态下所有的动作价值都比较大，这时候需要一个较大的基线对动作加以区分；相反，当所有动作价值都较小时基线也随之减小。所以选择一个良好的基线可以降低梯度算法的方差并且加快学习速度。在策略梯度的强化学习中，近似状态价值函数可以作为基线，并且也可以作为评判器评估与环境交互的动作好坏，其中常用的方法是通过随机梯度下降的函数逼近方法对价值函数进行预测。

使用随机梯度法首先要明确预测目标。在价值函数预测中，我们更关注状态 e_k 的近似状态价值函数 $\hat{v}(e_k, \mathbf{w})$ 与真实状态价值函数的误差 $v(e_k, \mathbf{w})$ 的误差，显然可以通过均方价值误差定义预测目标 $\overline{\mathrm{VE}}$：

$$\overline{\mathrm{VE}}\,(\mathbf{w}) = \sum_{n} \frac{1}{n} \left[v_{\boldsymbol{\pi}}(e_k) - \hat{v}(e_k, \mathbf{w}) \right]^2 \tag{4-24}$$

其中，$\mathbf{w}$ 是评判器的权值向量，n 为样本容量，$\frac{1}{n}$ 表示对不同状态下误差的重视程度，

通过上式可以对近似和真实状态价值函数的差异大小进行度量。通过求解预测目标 $\overline{\mathrm{VE}}$ 的梯度，权值向量每次迭代将沿着缩小预测目标的方向逼近，逐步减小近似预测价值函数的误差。评判器的权值向量更新如式 (4-25) 所示。

$$\begin{aligned}\mathbf{w}_{k+1} &= \mathbf{w}_k - \frac{1}{2}\alpha\nabla\left[v_{\boldsymbol{\pi}}\left(e_k\right) - \hat{v}\left(e_k, \mathbf{w}_k\right)\right]^2 \\ &= \mathbf{w}_k + \alpha\left[v_{\boldsymbol{\pi}}\left(e_k\right) - \hat{v}\left(e_k, \mathbf{w}_k\right)\right]\nabla\hat{v}\left(e_k, \mathbf{w}_k\right)\end{aligned} \tag{4-25}$$

其中，$\alpha > 0$ 是步长参数。$\nabla\hat{v}\left(e_k, \mathbf{w}_k\right)$ 是列向量，表示近似价值函数关于权值向量 $\mathbf{w}$ 对应分量的偏导数。

为了保证随机梯度法的权值向量最终收敛，近似状态价值函数的学习律 α 需要满足定理5.1中第三点的条件，即设步长参数序列 $\{\alpha_k\}$ 是一个正数序列，当序列满足条件 $\sum_{k=1}^{\infty}\alpha_k = \infty$ 和 $\sum_{k=1}^{\infty}\alpha_k^2 < \infty$，则对于几乎所有样本路径，随机梯度下降算法保证收敛。文献 [76] 关于随机梯度法提供了更详细的证明。

在实际应用中，状态价值函数往往是未知的，或者得到的值是带有噪声的扰动的状态价值函数；这种情况下不再使用式 (4-25) 进行权值向量更新，而是通过 U_k 近似取代 $v_{\boldsymbol{\pi}}\left(e_k\right)$ 进行更新。如果 U_k 是一个关于状态价值函数的无偏估计，那么当步长序列 $\{\alpha_k\}$ 满足定理4.2，权值 $\mathbf{w}_k$ 会收敛到局部最优解。根据折扣回报函数 G_t 和状态价值函数 $v_{\boldsymbol{\pi}}(e_k)$ 的定义，存在下式 (4-26) 成立。

$$v_{\boldsymbol{\pi}}(e_k) = \mathbb{E}_{\boldsymbol{\pi}}[U_k|e_k] = \mathbb{E}_{\boldsymbol{\pi}}[G_k|e_k] \tag{4-26}$$

此时评判器权值向量更新式 (4-25) 可以改写为：

$$\mathbf{w}_{k+1} = \mathbf{w}_k + \alpha_{\mathbf{w}}\left[G_k - \hat{v}\left(e_k, \mathbf{w}_k\right)\right]\nabla\hat{v}\left(e_k, \mathbf{w}_k\right) \tag{4-27}$$

观察式 (4-27)，近似状态价值函数 $\hat{v}(e_k, \mathbf{w})$ 仍然未知。在随机梯度法中只需要对其选择相应的近似构造方法就可以对真实价值函数进行预测逼近。在参数最优迭代学习控制中，对状态价值函数的近似方法可以采用线性构造的方法，即近似状态价值函数是权值向量 $\mathbf{w}$ 的线性函数。

根据状态价值函数的定义式 (4-26)，参数最优迭代学习控制的状态价值函数为：

$$v_{\boldsymbol{\pi}}(e_k) = \mathbb{E}_{\boldsymbol{\pi}}[G_k|e_k] = {e_k}^T W e_k + \omega\gamma_{k+1}^2 \tag{4-28}$$

其中，矩阵 W 为奖励函数 (4-7) 中 e_k 项的二次型矩阵。根据克罗内克积法则 (3-44) 可以对二次型参数线性化，并通过梯度法逐步更新该参数。近似的状态价值函数可以重新

改写为：

$$\hat{v}_{\boldsymbol{\pi}}(e_k,\gamma_{k+1},\mathbf{w}_k)=vec(\hat{W})(e_k\otimes e_k)+\omega\gamma_{k+1}^2=\mathbf{w}_k(e_k\otimes e_k)+\omega\gamma_{k+1}^2 \tag{4-29}$$

式中，$\mathbf{w}_k$ 是关于 $vec(\hat{W})=[h_{11},h_{12},\ldots,h_{1N},h_{22},h_{23},\ldots,h_{2N},\ldots,h_{NN}]$ 的估计权值向量，e_k 是第 k 次迭代学习实验的误差列向量。

状态价值函数 (4-29) 虽然引入了动作 γ_{k+1}，但是本章的动作 γ_{k+1} 根据式 (2-15) 来选择，即满足等式 (4-30)。

$$b(e_k,\gamma_{k+1})\nabla\sum_{\gamma_{k+1}}\max_{\gamma_{k+1}}\boldsymbol{\pi}(\gamma_{k+1}|e_k,\boldsymbol{\theta})=b(e_k,\gamma_{k+1})\nabla p=0 \tag{4-30}$$

p 为概率值。上式表明引入近似状态价值函数 $\hat{v}_{\boldsymbol{\pi}}(e_k,\gamma_{k+1},\mathbf{w}_k)$ 即不会使策略梯度的更新值发生变化，还可以减少策略梯度的估计方差，使得策略梯度方法更加稳定。

综上，基于随机梯度的线性状态价值函数逼近的评判器就完成了设计，通过估计动作的价值函数，帮助行动器更快学习到最优策略参数 $\boldsymbol{\theta}^*$。评判器的权值向量迭代式最终如式 (4-31) 所示。

$$\mathbf{w}_{k+1}=\mathbf{w}_k+\alpha_{\mathbf{w}}[G_k-\mathbf{w}_k(e_k\otimes e_k)-\omega\gamma_{k+1}^2](e_k\otimes e_k) \tag{4-31}$$

$\alpha_{\mathbf{w}}$ 表示价值函数预测的步长参数。

4.4.2 行动器设计

设计行动器的目标是通过最小化价值回报函数 G_k，找到一个最优的动作策略 $\boldsymbol{\pi}(\cdot)$，从而解决参数最优迭代学习控制问题。根据算法 4-1，自然可以通过 on-policy 策略梯度算法来设计行动器学习策略参数 $\boldsymbol{\theta}$，但是该方法在学习过程中策略参数的大范围波动极易导致算法的发散和稳定性变差。为了解决这些问题，下面将提出一种基于近端策略优化的策略梯度方法作为行动器运用于参数最优迭代学习控制中。

基于本文第二章近端策略优化的预备知识，近端策略优化的性能指标函数的梯度 (2-32) 在迭代学习中实例化为：

$$\begin{aligned}\nabla L(\boldsymbol{\theta})=&\mathbb{E}_k[\nabla\boldsymbol{\pi}(\gamma_{k+1}|e_k,\boldsymbol{\theta}_k)\min(r_k(\boldsymbol{\theta})\delta_k,\text{clip}(r_k(\boldsymbol{\theta}),1-\epsilon,1+\epsilon)\delta_k)]\\=&\mathbb{E}_k[\frac{\nabla\boldsymbol{\pi}(\gamma_{k+1}|e_k,\boldsymbol{\theta}_k)}{\boldsymbol{\pi}(\gamma_{k+1}|e_k,\boldsymbol{\theta}_{k-1})}\frac{\boldsymbol{\pi}(\gamma_{k+1}|e_k,\boldsymbol{\theta}_k)}{\boldsymbol{\pi}(\gamma_{k+1}|e_k,\boldsymbol{\theta}_k)}\min(r_k(\boldsymbol{\theta})\delta_k,\\&\text{clip}(r_k(\boldsymbol{\theta}),1-\epsilon,1+\epsilon)\delta_k)]\end{aligned} \tag{4-32}$$

其中，$r_k(\boldsymbol{\theta})$ 是新旧策略的重要性比率。根据重要性比率的定义式 (2-29)，参数最优迭

代学习控制下的策略梯度的策略比率为：

$$r_k(\boldsymbol{\theta}) = \frac{\boldsymbol{\pi}(\gamma_{k+1}|e_k, \boldsymbol{\theta}_k)}{\boldsymbol{\pi}(\gamma_{k+1}|e_k, \boldsymbol{\theta}_{k-1})} \tag{4-33}$$

$\boldsymbol{\theta}_k$ 是当前迭代实验策略参数，$\boldsymbol{\theta}_{k-1}$ 是上一次迭代的策略参数。根据式 (4-18) 的紧凑形式处理和，式 (4-32) 可以化简为：

$$\begin{aligned} \nabla L(\boldsymbol{\theta}) &= \min(\text{clip}(r_k(\boldsymbol{\theta}), 1-\epsilon, 1+\epsilon), r_k(\boldsymbol{\theta}))\mathbb{E}_k[\frac{\nabla \boldsymbol{\pi}(\gamma_{k+1}|e_k, \boldsymbol{\theta}_k)}{\boldsymbol{\pi}(\gamma_{k+1}|e_k, \boldsymbol{\theta}_k)}\delta_k] \\ &= \min(\text{clip}(r_k(\boldsymbol{\theta}), 1-\epsilon, 1+\epsilon), r_k(\boldsymbol{\theta}))\mathbb{E}_k[\nabla \ln(\boldsymbol{\pi}(\gamma_{k+1}|e_k, \boldsymbol{\theta}_k))\delta_k] \end{aligned} \tag{4-34}$$

式中函数 clip$(\cdot)$ 和 $r_k(\theta)$ 的结果都是一个值，δ_k 是优势函数，即迭代学习中的折扣回报函数 G_k，引入评判器的近似状态价值函数 (4-29) 作为基线后如式 (4-35) 所示。

$$\delta_k = G_k - \mathbf{w}_k(e_k \otimes e_k) - \omega\gamma_{k+1}^2 \tag{4-35}$$

为了控制算法中策略更新的幅度，引入了参数 β_k，定义为 $r_k(\boldsymbol{\theta})$ 和该值在输入限制约束函数输出的最小值，则式 (4-34) 改写为：

$$\nabla L(\boldsymbol{\theta}) = \beta_k \mathbb{E}[\nabla \ln(\boldsymbol{\pi}(\gamma_{k+1}|e_k, \boldsymbol{\theta}_k))\delta_k] \tag{4-36}$$

其中，$\beta_k = \min(\text{clip}(r_k(\boldsymbol{\theta}), 1-\epsilon, 1+\epsilon), r_k(\boldsymbol{\theta}_k))$，$\epsilon$ 是一个超参数，用于限制重要性比率的变化范围。策略参数迭代式 (2-33) 在迭代学习中实例化为

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \alpha_{\boldsymbol{\theta}}\beta_k \mathbb{E}[\nabla \ln(\boldsymbol{\pi}(\gamma_{k+1}|e_k, \boldsymbol{\theta}_k))\delta_k] \tag{4-37}$$

$\alpha_{\boldsymbol{\theta}}$ 是策略参数更新的步长参数。基于 (2-15) 的动作选择策略，得到最终 on-policy 近端策略优化的策略梯度算法参数更新如式 (4-38) 所示。

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \alpha_{\boldsymbol{\theta}}(G_k - \hat{v}_{\boldsymbol{\pi}}(e_k, \gamma_{k+1}, \mathbf{w}_k))\beta_k \nabla \ln(\boldsymbol{\pi}(\gamma_{k+1}|e_k, \boldsymbol{\theta}_k)) \tag{4-38}$$

综上，行动器通过采用参数化函数形式重新构建策略 $\boldsymbol{\pi}$，旨在最小化式回报函数 (4-8)，进而借助参数化策略有效选择和优化下一个动作 γ_{k+1}。这一优化过程同时还借助于评判器提供的状态价值函数预测，对策略参数 $\boldsymbol{\theta}_k$ 的循环迭代进行调整。

4.4.3 基于近端策略优化 on-policy 策略梯度方法的参数最优迭代学习控制

接下来，基于行动器和评判器参数更新式 (4-38) 和 (4-31)，通过本章算法 4-1 的参数化策略函数 (4-9) 及参数化的均值 (4-10) 和方差 (4-12)，即可在线学习优化策略参数

$\boldsymbol{\theta}_k$。经过多次迭代后最终将收敛于最优的策略参数 $\boldsymbol{\theta}^*$，相应的通过 (2-15) 即可求解参数最优迭代学习控制的最优动作序列 γ_{k+1}。

现在提出本章第二个基于 on-policy 策略梯度理论的算法，即带有价值函数预测的近端策略优化梯度方法的最优迭代学习控制算法 4-2，该算法与算法 4-1 都是通过在线学习最优策略参数旨在建立一种不依赖于系统模型参数实现参数最优迭代学习控制的方法；不同之处在于算法 4-2 针对算法 4-1 的局限性进行了改进，这将在数值仿真中将予以展示。下面给出了算法 4-2 在“行动器-评判器”框架 (图2-5) 下的伪代码和流程图见图4-1。

算法 4 - 2: 基于近端策略优化 on-policy 策略梯度方法的参数最优迭代学习控制

初始化: 初始输入 u_0，策略参数向量 $\boldsymbol{\theta}_0$，权值向量 $\mathbf{w}_0$，最大迭代试验数 m，步长参数 $\alpha_{\boldsymbol{\theta}}, \alpha_{\mathbf{w}}$，超参数 ϵ，开始迭代。

输出: 下一次迭代实验的输入 u_{k+1}

开始循环 $k = 1 \rightarrow m$

> **环境交互** *将 u_k 输入到系统中进行一次迭代实验，收集迭代学习运行状态信息误差 e_k*
>
> **行动器策略更新** *计算策略参数 $\mu(e_k, \theta_u)$ 和 $\sigma(e_k, \theta_\sigma)$，代入式 (4-9) 选择动作 γ_{k+1}，通过参数最优迭代学习律 (4-1) 更新下次迭代实验输入 u_{k+1}*
>
> **评判器评估** *通过式 (4-29) 求得近似价值函数 $\hat{v}(e_k, \gamma_{k+1}, \mathbf{w}_k)$*
>
> **真实回报** *通过式 (4-8) 求得折扣回报函数 G_k*
>
> **评判器参数更新** *递推迭代式 (4-31) 更新近似价值函数参数 $\mathbf{w}_{k+1}$*
>
> **行动器参数更新** *计算重要性比率求得 β_k，代入式 (4-38) 更新策略参数 $\boldsymbol{\theta}_{k+1}$*
>
> k=k+1

结束循环

算法 4-2 显示了决定迭代学习参数 γ_k 的策略 $\boldsymbol{\pi}$ 在不依赖于系统模型参数信息下在线学习求解近端策略优化性能指标函数最小值的过程。在每一次迭代学习的实验中，待优化的迭代学习律可以通过策略 $\boldsymbol{\pi}(\gamma_{k+1}|e_k, \boldsymbol{\theta}_k)$ 选择动作 γ_{k+1} 求得，通过状态价值函数预测以及策略梯度不断逼近更优的权值向量 $\mathbf{w}_k$ 和策略参数 $\boldsymbol{\theta}_k$，而动作策略的好坏以及近似状态价值函数的性能取决于权重向量和动作策略参数的优劣；也就是说算法 4-2 是在线学习最优策略参数进而确定最优策略然后选择最优动作最终得到参数最优的迭代

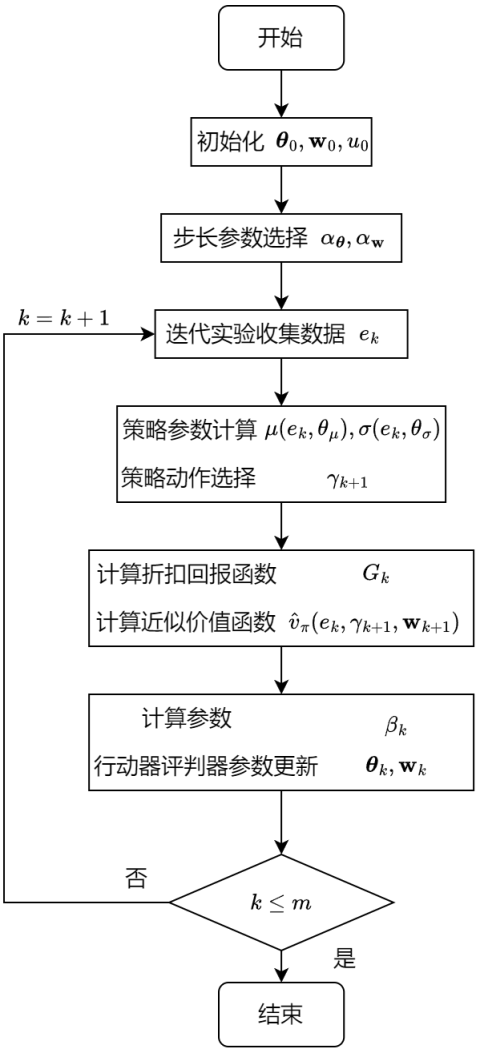

图 4-1 算法 4-2 的流程图

学习过程。在下一节中，将对本章提出的算法设计实验，然后展示实验结果来比较数据驱动的方法和基于模型的方法性能。

4.5 数值仿真

在本节中，将使用三轴龙门式机器人模型作为被控对象来设计仿真实验。仿真实验分别使用参数最优迭代学习控制 (POILC)、算法 4-1(PG-POILC) 和算法 4-2(PPO-POILC) 分别对该系统进行迭代学习控制，其中参数最优迭代学习控制分为使用精确模型的情况 (accurate) 和不准确模型参数 (inaccurate) 的情况。通过比较以上算法的误差曲线、不同迭代次数实验的跟踪曲线、基于策略梯度算法的策略参数 θ_μ 范数曲线、近似状态价值函数权值估计向量 **w** 和损失函数的参数值曲线，验证了算法 4-1 和算法 4-2 的有效性和收敛性。

根据文献 [77] 中的多轴系统系统建模，考虑下面的 SISO 离散时间状态空间方程：

$$\begin{aligned} x_k(t+1) &= \begin{bmatrix} 0.99 & -0.0156 & 0.1559 & -0.002 \\ 0.250 & 0 & 0 & 0 \\ 0 & 0.125 & 0 & 0 \\ 0 & 0 & 0.0039 & 0 \end{bmatrix} x_k(t) + \begin{bmatrix} 0.0078 \\ 0 \\ 0 \\ 0 \end{bmatrix} u_k(t) \\ y_k(t) &= \begin{bmatrix} 0.0013 & -0.00001 & -0.00092 & 0.0039 \end{bmatrix} x_k(t) \end{aligned} \tag{4-39}$$

而参数最优迭代学习律使用了不准确模型参数的系统状态空间方程如下：

$$\begin{aligned} x'_k(t+1) &= \begin{bmatrix} 0.99 & -0.0154 & 0.1559 \\ 0.250 & 0 & 0 \\ 0 & 0.125 & 0 \end{bmatrix} x'_k(t) + \begin{bmatrix} 0.125 \\ 0 \\ 0 \end{bmatrix} u_k(t) \\ y_k(t) &= \begin{bmatrix} 0.0897 & -0.0379 & -0.0348 \end{bmatrix} x'_k(t) \end{aligned} \tag{4-40}$$

这两个系统的仿真实验时间长度均为 $0.25s$，实验的采样时间 T_s 都是 $0.01s$，系统初始状态 $x_k(0) = \mathbf{0}$。迭代学习的期望轨迹：

$$r(t) = 2\sin(8\pi t), \quad t \in [0, 0.25] \tag{4-41}$$

本节中仿真实验通过 200 次迭代学习实验来展示所提出的算法性能，且每次迭代学习的初始输入条件为零。算法其余涉及的参数选择设置如下: 不论是否依赖于系统模型参数信息，POILC、PG-POILC 和 PPO-POILC 三个算法的权值 ω 为 5×10^{-12}。对于策略梯度的方法 PG-POILC 和 PPO-POILC，这两个算法都将策略参数 μ 和 σ 利用函数近似器 (4-24) 和 (4-26) 计算，其中为了方便策略的探索和近似，参数序列 $a_i(i = 1, 2, ..., 10)$ 选为 $(10^0, 10^{-1}, 10^1, 10^{-2}, 10^2, ...)$，两个算法的梯度迭代式 (4-29) 和 (4-56) 的学习步长参数都预设为 1.6，然后逐渐减小以符合算法的稳定性条件；二者的参数向量的初始值满足 $\theta_\mu = 0.00002 \times \mathbf{1}$，其中 $\mathbf{1}$ 是所有元素全为数值 1 的列向量；θ_σ 为 0.001。对于带有价值函数近似和近端策略优化的 PPO-POILC 算法，超参数 ϵ 设置为 0.25 来用于控制更新策略时的剪切量，价值近似权重向量初始化为 $\mathbf{0}$，其价值近似梯度更新的步长初始参数为 2.5×10^{-6}。

首先 POILC、PG-POILC 和 PPO-POILC 三个算法的前 100 次迭代实验的误差范数曲线如图4-2，图例的 $POILC(inaccurate)$ 表示基于系统模型 (4-40) 设计的参数最优迭代学习律作用在模型 (4-39) 的误差范数曲线；其次为了更加直观展示策略梯度法的有效

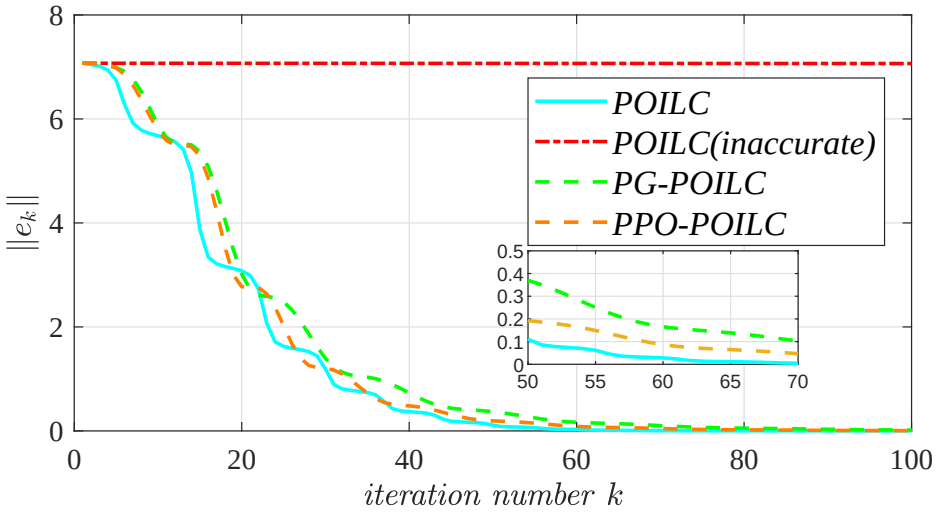

图 4-2 前 100 次试验中三种算法的误差范数收敛性

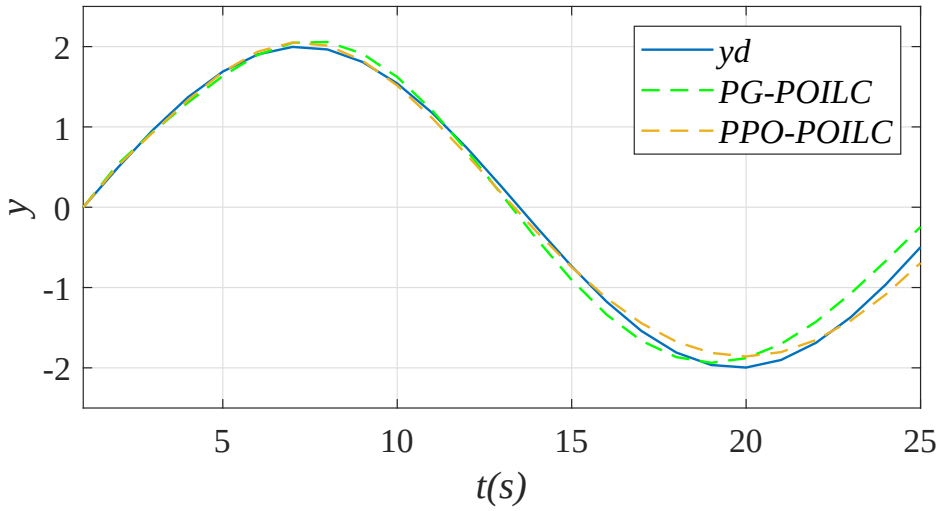

图 4-3 第 40 次迭代实验时的系统输出轨迹

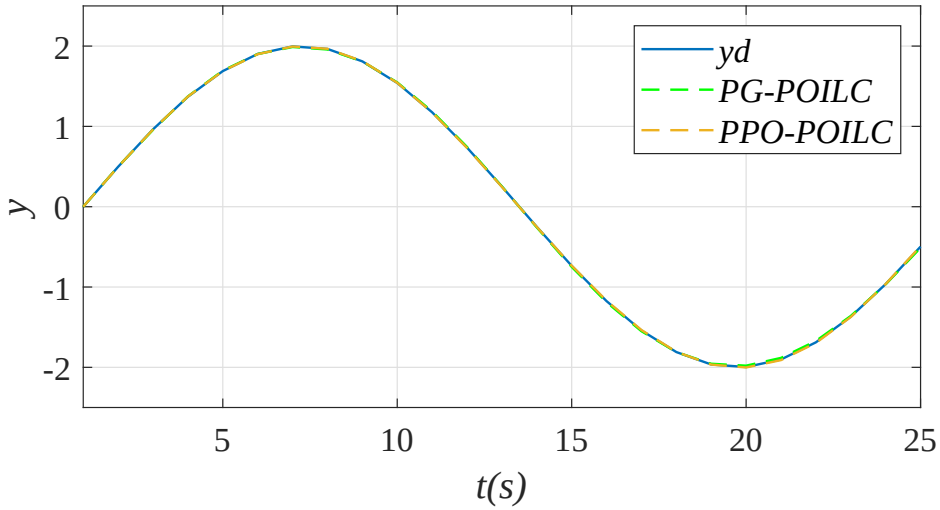

图 4-4 第 80 次迭代实验时的系统输出轨迹

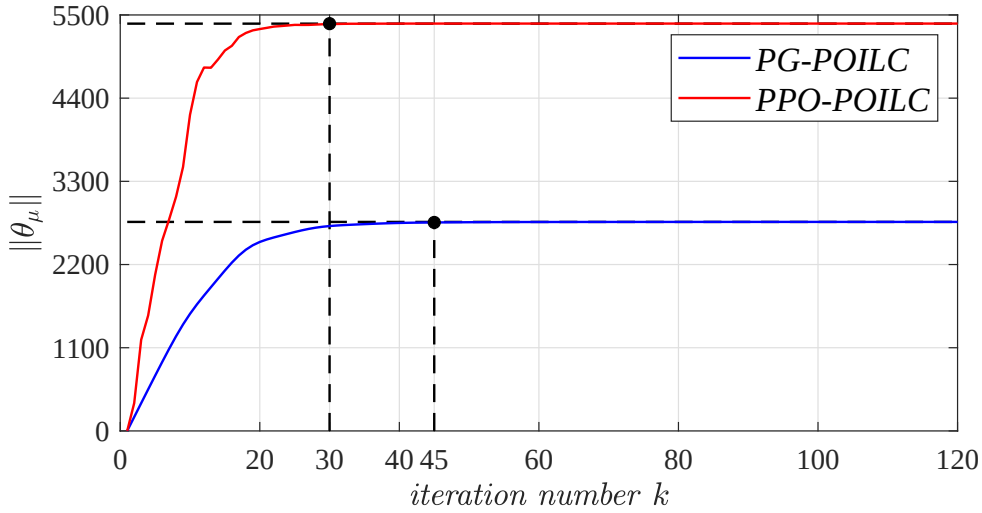

图 4-5 算法 4-1 和算法 4-2 的策略参数 θ_μ 的范数

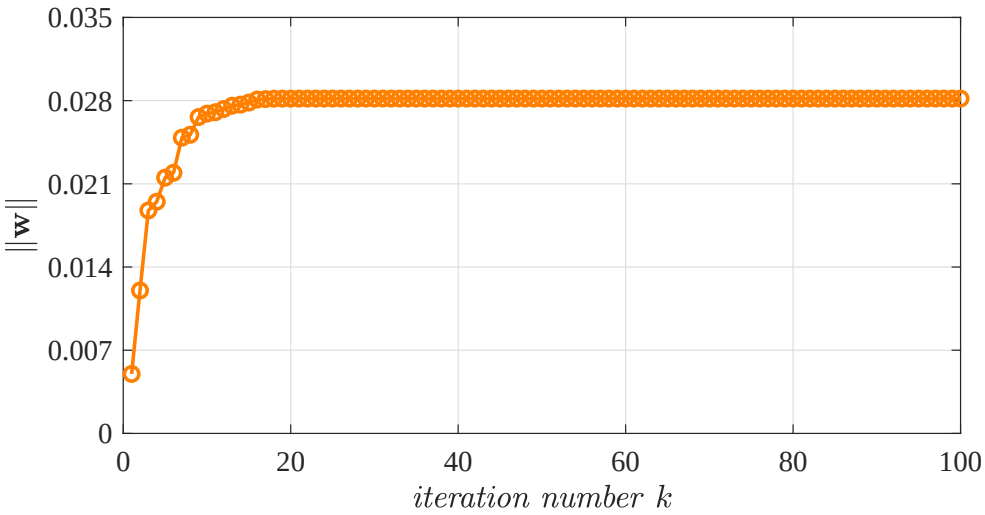

图 4-6 算法 4-2 近似状态价值函数权值估计向量范数

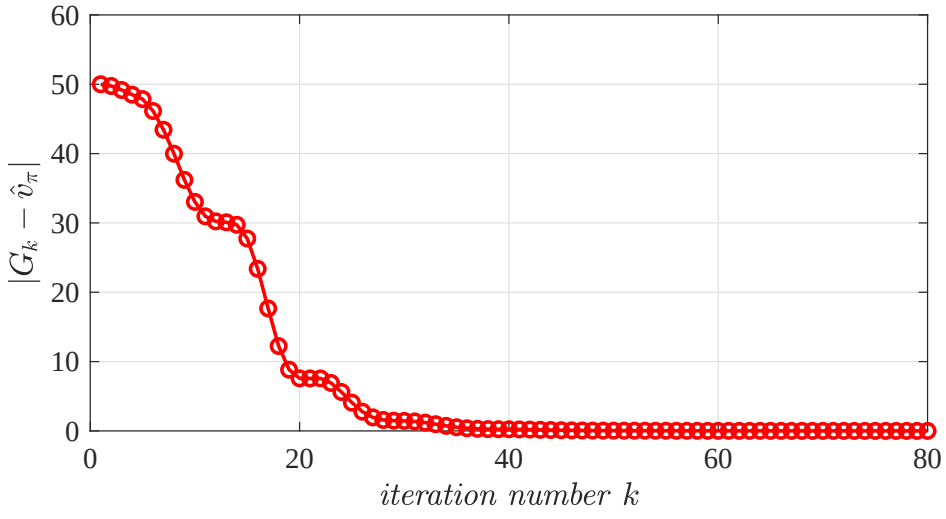

图 4-7 算法 4-2 状态价值函数的损失曲线

性，图4-3和图4-4分别为 40 次迭代实验 80 次迭代实验时的系统输出轨迹和期望轨迹曲线；接着图4-5展示了两种算法的策略参数 θ_μ 在策略优化学习过程中的收敛性及收敛速度比较，图中的两条横向虚线表示算法各自的策略参数的最终取值；最后为了进一步展示算法 4-2 中近端策略优化梯度和状态价值函数近似的收敛性和稳定性，图4-6展示了算法 4-2 中评判器关于状态价值函数的估计权值向量收敛曲线，而图4-7反映了其价值函数的损失曲线，用来衡量评判估计的价值函数与实际的奖励（或回报）之间的差异。

观察图4-2，图中的放大图展示了第 50 次至 70 次迭代实验误差范数曲线。当已知系统动态模型时，基于模型的 POILC 算法性能最好，在第 60-65 次之间迭代实验使得系统达到了收敛，这也符合基于模型的迭代学习方法的普遍特性，即基于模型的方法的收敛特性相比于不依赖于系统动力学参数信息的方法往往取得更好的效果；但是当系统模型参数未知或不准确时，其收敛性能明显下降，这也是此类方法的局限性。

通过观察图4-2的 PG-POILC 和 PPO-POILC 的误差范数曲线，说明两个算法都能有效地降低跟踪误差范数，与性能最优的 POILC 相比可以表明这一类基于策略梯度的迭代学习方法可以在保证系统取得良好收敛特性的前提下有效降低了对系统模型的依赖性；图4-3和图4-4系统轨迹和期望轨迹比较也印证了两个算法的有效性。

通过观察图4-5的两条策略参数 θ_μ 曲线可知，算法 4-1 和算法 4-2 的策略参数最终都能稳定收敛到一个固定值（图中各自的横向虚线），验证了本章算法的收敛性；将两条曲线互相比较，算法 4-2 的曲线在第 30 次迭代实验左右达到稳定值，而算法 4-1 的在第 45 次迭代实验才收敛于稳定值，表明 PPO-POILC 算法相比于 PG-POILC 算法的收敛速度更快，能更快地学习到优化后且稳健的动作策略，体现了近端策略优化和价值函数近似的有效性。

最后基于近似价值函数权值图4-6和价值函数损失曲线图4-7，仿真结果显示近似价值函数的权值向量范数从 0 开始逐渐增大，并最终趋向于稳定某一个值，相应的价值损失函数的值也逐渐减小最后也趋向于稳定。这表明模型的学习过程取得了进展，也同时意味着算法在训练过程中可能逐渐收敛到了一个相对稳定的策略，这样的稳定性反映了算法对系统环境的充分探索，使得模型能够更准确地估计状态的价值。然而，同轨策略在线学习的方法在面对高速系统有限的训练数据和有限的实时计算内存仍然难以解决问题，并且在算法的训练过程中由于待优化策略本身就是行为策略也因此容易受到自身扰动的影响。

4.6 本章小结

本章延续上一章的研究问题，如何在保证系统在迭代学习中依旧取得良好的收敛特性的前提下减少迭代学习律的设计对系统模型的依赖性。在第三章中研究了强化学习基于值函数的算法在预测最优迭代学习控制中的应用，本章以参数最优迭代学习控制问题为背景，首先介绍了参数最优迭代学习控制并在强化学习框架下重新构建。接着描述并引入了基于策略梯度算法的强化学习，通过参数化策略 π，定义关于策略参数的性能指标进行估计，并且在线探索最优参数策略，提出了基于 on-policy 策略梯度方法的参数最优迭代学习控制，并验证了策略梯度的收敛性。然后为了提高算法稳定性，利用近端策略优化和价值函数近似的方法改进策略梯度理论，并在“行动器-评判器”的框架下提出了基于价值函数预测近端策略优化梯度的参数最优迭代学习控制。最后在多轴系统下的数值仿真表明，这两种算法都能解决迭代学习控制律的设计问题，实现跟踪误差范数的快速收敛，验证了算法的有效性。

第五章　基于 off-policy 策略梯度的参数最优迭代学习控制方法

5.1　引言

在上一章中研究了一类基于 on-policy 的策略梯度算法，并且使用近端策略优化等理论加以改进，最终通过同轨策略在线整定的方法得到了最优迭代学习律。但是在在线学习的同轨策略梯度方法中，动作选择受限于当前采取的行为策略，因此在在线学习中所使用的数据也只能依赖于当前策略生成，这样使得样本效率较低和算法的探索性变差，需要通过与环境的交互获得更多的样本数据才能迭代获得良好的策略；而在线学习的方式也使得算法在面对解决一些数据量有限、实时计算内存受限的高速系统问题仍然面临挑战。

相较于 on-policy 策略梯度算法，off-policy 策略梯度方法由于学习过程中待优化的目标策略和同环境交互生成训练数据的行为策略分离，从而不受当前策略的限制，所以样本效率得到了提高，并且能更充分地利用已有的经验，这也使得该算法具有更强的探索性，能够更加灵活地探索环境和动作空间。

本章在延续上一章参数最优迭代学习控制的研究背景基础上，继续研究开发一种离线学习下的迭代学习律设计方法，旨在实现在不依赖于系统模型参数的情况下，获得与使用基于模型的参数最优迭代学习控制算法相近的收敛速度。首先将行为策略和目标策略进行分离，将策略梯度方法推广到 off-policy 形式下。然后基于前一章马尔科夫框架下的参数最优迭代学习控制和价值函数近似方法，结合 off-policy 策略梯度理论，依旧在“行动器-评判器”的算法框架下提出了一种基于 off-policy 策略梯度的参数最优迭代学习控制算法。最后，通过数值仿真与基于模型的参数最优迭代学习控制对比，验证了所提算法的有效性，并展示了该算法不依赖模型参数的优势。

5.2 节介绍了 off-policy 策略梯度理论，描述了算法的流程及如何在离线下更新学习策略参数。5.3 节在该理论下，提出了离线学习的 off-policy 策略梯度的参数最优迭代学习控制，并展示了算法的运行过程并讨论了算法的有效性。5.4 节给出了仿真实验，通过将目标策略、行为策略下的迭代学习控制与参数最优迭代学习控制比较，验证了 off-policy 策略梯度在迭代学习中的有效性。5.5 节总结了本章的内容。

5.2 off-policy 策略梯度理论

本节将从第四章的 on-policy 策略梯度理论出发，通过将行为策略和目标策略分离，最终将策略梯度理论推广到离轨策略下，得到关于目标策略参数 θ 的迭代式。

根据文献 [78]，off-policy 策略参数梯度迭代式如下所示：

$$\theta_{t+1} \approx \theta_t - \alpha \nabla_\theta J(\theta_t) \tag{5-1}$$

其中，θ 为目标策略参数，t 为更新迭代次数，关于性能指标对策略参数 θ 的梯度为：

$$\nabla_\theta J(\theta_t) \approx \mathbf{g}(\theta) = \sum_{s \in S} d^b(s) \sum_{a \in A} \nabla_\theta \boldsymbol{\pi}(a|s) Q^{\boldsymbol{\pi}}(s,a) \tag{5-2}$$

$d^b(s)$ 表示在行为策略 $b(\cdot)$ 下的状态分布，S 和 A 分别为状态空间和动作空间，s 和 a 分别为状态和动作；现在通过行为策略与环境的交互数据将策略梯度推广到离轨策略下，将上式方程重写为期望的形式：

$$\begin{aligned}
\mathbf{g}(\theta) &= \mathbb{E}\left[\sum_{a \in A} \nabla_\theta \boldsymbol{\pi}(a|s) Q^{\boldsymbol{\pi}}(s,a) \middle| s \sim d^b\right] \\
&= \mathbb{E}\left[\sum_{a \in A} b(a|s) \frac{\boldsymbol{\pi}(a|s)}{b(a|s)} \frac{\nabla_\theta \boldsymbol{\pi}(a|s)}{\boldsymbol{\pi}(a|s)} Q^{\boldsymbol{\pi}}(s,a) \middle| s \sim d^b\right] \\
&= \mathbb{E}\left[\rho(s,a) \frac{\nabla_\theta \boldsymbol{\pi}(a|s)}{\boldsymbol{\pi}(a|s)} \middle| s \sim d^b, a \sim b(\cdot|s)\right] \\
&= \mathbb{E}_b\left[\rho(s_t,a_t) \frac{\nabla_\theta \boldsymbol{\pi}(a_t|s_t)}{\boldsymbol{\pi}(a_t|s_t)} Q^{\boldsymbol{\pi}}(s_t,a_t)\right]
\end{aligned} \tag{5-3}$$

其中，$\rho(s_t,a_t) = \frac{\boldsymbol{\pi}(a_t|s_t)}{b(a_t|s_t)}$ 描述的是在相同状态 s_t 下目标策略 $\boldsymbol{\pi}(s_t)$ 和行为策略 $b(s_t)$ 选取相同动作 a_t 的采样比率；$E_b[\cdot]$ 含义是从行为策略的稳态分布中采样得到的所有随机变量的期望，这样可以将所有采样数据都视为来自行为策略的稳态分布，简化了离轨策略梯度的推导和分析。

基于上述方程 (5-3) 得到新的关于目标策略参数的梯度，可以得到 off-policy 策略梯度方法的策略参数学习迭代式：

$$\theta_{t+1} = \theta_t - \alpha_\theta \rho(s_t,a_t) \frac{\nabla \boldsymbol{\pi}(a_t|s_t)}{\boldsymbol{\pi}(a_t|s_t)} Q^{\boldsymbol{\pi}}(s_t,a_t) \tag{5-4}$$

进行紧凑形式处理方程 (5-1) 被重新表述为：

$$\theta_{t+1} = \theta_t - \alpha_\theta \rho(s_t,a_t) \nabla_\theta \ln \boldsymbol{\pi}(a_t|s_t,\theta_t) Q^{\boldsymbol{\pi}}(s_t,a_t) \tag{5-5}$$

这样就得到了 off-policy 的策略梯度方法。

相比于 on-policy 的策略梯度方法，离轨策略下的方法依旧需要满足策略梯度定理2.1。为了保证算法的收敛性，首先，重要性采样比率 ρ 的期望值必须保持有界，这避免了算法在参数更新时受到过大的扰动；其次，选择合适的学习率 α_θ 进行学习，这控制了参数的变化幅度，以确保算法的稳定性；然后，策略函数 $\boldsymbol{\pi}$ 满足 Lipschitz 连续性条件，这有助于防止梯度过大或过小的问题，加速算法的收敛速度；最后，值函数 $Q^{\boldsymbol{\pi}}$ 严格有界，降低数值计算的不稳定性，更容易地控制算法的学习过程。文献 [79] 详细证明了 off-policy 策略梯度的收敛性及所需要满足的条件。

5.3 基于“行动器-评判器”框架下 off-policy 策略梯度方法

在本节中，仍然延续第四章中 4.1.2 节马尔科夫决策过程下的参数最优迭代学习控制作为研究对象，基于离轨策略理论分离行为策略和目标策略并引入 off-policy 策略梯度的迭代索引 $j = 1, 2, ..., m-1$，然后分别通过对行动器和评判器的设计逼近最优策略参数和真实价值函数，最终完成数据驱动参数最优迭代学习律的设计。

5.3.1 评判器设计

通过观察方程 (5-5)，为了减少梯度估计的方差和提高算法的效率，显然依旧自然联想到给价值函数 $Q^{\boldsymbol{\pi}}$ 添加基线，同时也可以继续根据上一章的随机梯度法对状态价值函数进行估计。利用随机梯度法对梯度估计存在方差较高、更新不稳定性和收敛速度较慢的问题，而神经网络能够通过学习特征表示，降低梯度估计的方差，并且具备更强的表达能力，从而更有效地逼近状态价值函数，提高算法的效率。因此，下面将给出一种基于单层神经网络梯度下降法的状态价值函数的近似设计方法。

神经网络具有普遍近似的特性可以用于对价值函数近似。在参数最优迭代学习控制中，定义在状态 e_j 下利用神经网络逼近的价值函数为：

$$v_{\boldsymbol{\pi}}(e_j, \gamma_{j+1}, \mathbf{w}_j) = \mathbf{w}_j^T \varphi(e_j) + \omega\gamma_{j+1}^2 + \eta(j) \tag{5-6}$$

其中，$\mathbf{w}_j$ 代表神经网络的权重向量，$\varphi(\cdot)$ 是神经网络的激活函数向量，$\eta(j)$ 代表神经网络的逼近误差。假设当前单层神经网络第 j 次强化学习的价值函数估计值为 $\hat{v}_{\boldsymbol{\pi}}(e_j, \gamma_{j+1}, \mathbf{w}_j)$，则上式的神经网络逼近值为：

$$\hat{v}_{\boldsymbol{\pi}}(e_j, \gamma_{j+1}, \mathbf{w}_j) = \mathbf{w}^T{}_j \varphi(e_j) + \omega\gamma_{j+1}^2 \tag{5-7}$$

损失函数可以选择均方误差定义：

$$E(\mathbf{w}_j)=\frac{1}{2n}\sum_{i=0}^{n}(G_j^i-\hat{v}_{\boldsymbol{\pi}}^i(e_j^i,\gamma_{j+1}^i,\mathbf{w}_j))^2 \tag{5-8}$$

式中，n 为样本容量，表示第 j 次评判器参数更新中参与学习逼近的样本数量。神经网络通过最小化损失函数来调整其内部参数，使得预测结果与真实观测值之间的差距尽可能小；这一优化过程通常使用梯度更新法，它利用损失函数关于权值参数的梯度信息，沿着梯度方向更新参数，以逐步降低损失函数值。下面给出损失函数 E 关于权值向量的梯度和神经网络权值估计的迭代式：

$$\nabla E(\mathbf{w}_j)=-\sum_{i=0}^{n}(G_j^i-\hat{v}_{\boldsymbol{\pi}}^i(e_j^i,\gamma_{j+1}^i,\mathbf{w}_j))\varphi(e_j^i) \tag{5-9}$$

$$\mathbf{w}_{j+1}=\mathbf{w}_j-\alpha_{\mathbf{w}}\sum_{i=0}^{n}(G_j^i-\mathbf{w}_j^T\varphi(e_k^i)-\omega(\gamma_{j+1}^i)^2)\varphi(e_j^i) \tag{5-10}$$

其中，$\alpha_{\mathbf{w}}>0$ 是评判器的学习律。

5.3.2 行动器设计

行动器设计的目标是找到一个最优的目标策略 $\boldsymbol{\pi}(\cdot)$，从而通过最小化价值回报函数 (4-8) 寻找出参数最优迭代学习控制问题的解。与前一章相似，通过将目标策略参数化，可以选择连续动作 γ，并利用行动器的值函数估计来调整这些参数，从而引入更多学习信号以适应不同的环境，提高了算法的收敛速度。

为了便于动作值 γ 的选择，对于每个状态空间中的状态 $e_j\in S$，参数化目标策略 $\boldsymbol{\pi}(e_j)$ 由连续动作空间上的一维高斯分布的概率密度函数来近似。近似目标策略 $\boldsymbol{\pi}(\gamma_{j+1}|e_j,\theta_j=\theta)$ 由下式给出：

$$\boldsymbol{\pi}(\gamma_{j+1}|e_j,\theta)=\frac{1}{\sqrt{2\pi}\sigma(e_j,\theta)}\exp\left(-\frac{[\gamma_{j+1}-\mu(e_j,\theta)]^2}{2\sigma^2}\right) \tag{5-11}$$

其中，$\mu(e_j,\theta)\in\mathbb{R}$ 表示在状态 e_j 下目标策略 $\boldsymbol{\pi}$ 的均值；与 on-policy 的策略参数化不同，$\sigma\in\mathbb{R}^+$ 在本章 off-policy 策略参数化中是相对于目标策略参数 θ 不变的标准差；相应的，off-policy 下的策略参数 θ 将降维成只关于均值 μ 的逼近权值向量 θ_μ。

相比于 on-policy 的策略梯度算法，本章选择在 off-policy 策略梯度方法中固定方差而不进行参数化，主要原因在于 off-policy 策略梯度算法中，目标策略参数的梯度方程 (5-3) 中目标策略和行为策略的采样比率 $\rho(s_t,a_t)$ 会放大目标策略参数化方差不稳定的

影响。这进一步导致算法的不稳定性增加，甚至可能引起学习过程出现发散的情况。相反，采用固定方差的方法在提高算法稳定性和泛化能力方面具有显著优势。通过保持方差固定，避免了由于方差参数的不稳定性而引起的算法不稳定情况，并且有助于算法更好地适应待观察的数据，从而增强了算法的泛化能力。

在本章节中，概率密度函数 (5-11) 的均值 μ 的估计依旧可以通过以下函数近似器：

$$\mu(\theta, e_j) = \theta^T z_\mu = \theta^T [z_{\mu,1}(e_j), z_{\mu,2}(e_j), ..., z_{\mu,h}(e_j)] \tag{5-12}$$

其中，$h \geq 1$，表示近似基函数向量的维度，基函数向量的元素 $\{z_{\mu,i}(e_j)|i = 1, 2, ...h\}$ 定义为：

$$z_{\mu,i}(e_j) = \exp(-\frac{a_i \|e_j\|^2}{2}) \tag{5-13}$$

a_i 是一个放缩系数。

接下来通过调整策略参数，使得关于策略参数的性能指标最小化。性能指标相对于目标策略参数的梯度描述了该参数的最速下降方向，并且可以沿着该方向用于更新目标策略参数。但是性能指标的梯度不能直接计算，因为性能指标是未知的，并且取决于行为策略下的状态和选择的动作，这两者都同时取决于行为策略。此时，可以根据 off-policy 的策略梯度定理 (5-3) 对梯度进行采样，即根据行为策略访问的每个状态和选择的动作都可以用于估计性能指标关于目标策略参数的梯度。

根据 off-policy 的策略梯度方法的参数更新方程 (5-5)，可以得到参数最优迭代学习控制的目标策略参数的学习迭代式：

$$\theta_{j+1} = \theta_j - \alpha_\theta \rho(e_j, \gamma_{j+1}) \nabla_\theta \ln \boldsymbol{\pi}(\gamma_{j+1}|e_j, \theta_j) Q^{\boldsymbol{\pi}}(e_j, \gamma_{k+1}) \tag{5-14}$$

将上式引入评判器的值函数估计作为价值函数的基线：

$$\theta_{j+1} = \theta_j - \alpha_\theta \rho(e_j, \gamma_{j+1}) \nabla_\theta \ln \boldsymbol{\pi}(\gamma_{j+1}|e_j, \theta_j)(G_j - \hat{v}_{\boldsymbol{\pi}}(e_j, \gamma_{j+1}, \mathbf{w}_j)) \tag{5-15}$$

其中，α_θ 是行动器目标策略参数的学习率。上式关于高斯分布中参数 μ 的梯度可以通过以下式子计算得到：

$$\nabla_\theta \ln \boldsymbol{\pi}(\gamma_{j+1}|e_j, \theta) = \frac{\gamma_{j+1} - \mu(e_j, \theta)}{\sigma^2} z_\mu(e_j) \tag{5-16}$$

至此便可以通过式 (5-15) 和 (5-16) 对目标策略参数 θ 进行估计，完成关于 off-policy 策略梯度行动器的设计。

5.3.3 基于 off-policy 策略梯度方法的参数最优迭代学习控制

接下来，基于行动器和评判器参数更新式 (5-15) 和 (5-10)，即可得到最优的目标策略参数 θ^*，然后依据形如 (5-12) 和 (5-13) 的参数化方法，即：

$$\begin{aligned}\mu(\theta^*, e_k) =& \theta^{*T}[z_{\mu,1}(e_k), z_{\mu,2}(e_k), ..., z_{\mu,h}(e_k)] \\ z_{\mu,i}(e_k) =& \exp(-\frac{a_i\|e_k\|^2}{2})\end{aligned} \tag{5-17}$$

其中，k 是第 k 次迭代学习实验的索引，而参数最优迭代学习控制的动作 γ_{k+1} 由最优策略参数在状态 e_k 下，通过式 (5-11) 计算得到的最优目标策略 $\boldsymbol{\pi}^*(\gamma_{k+1}|e_k, \theta^*)$ 决定。

现在提出带有价值函数预测基于 off-policy 策略梯度方法的最优迭代学习控制算法-算法 5-1，该算法旨在不依赖于系统模型参数的情况下通过离线学习最优目标策略实现参数最优迭代学习控制。本节剩余部分将呈现算法 5-1 在“行动器-评判器”框架下的伪代码和流程图见图5-1，并给出满足算法收敛性的条件。

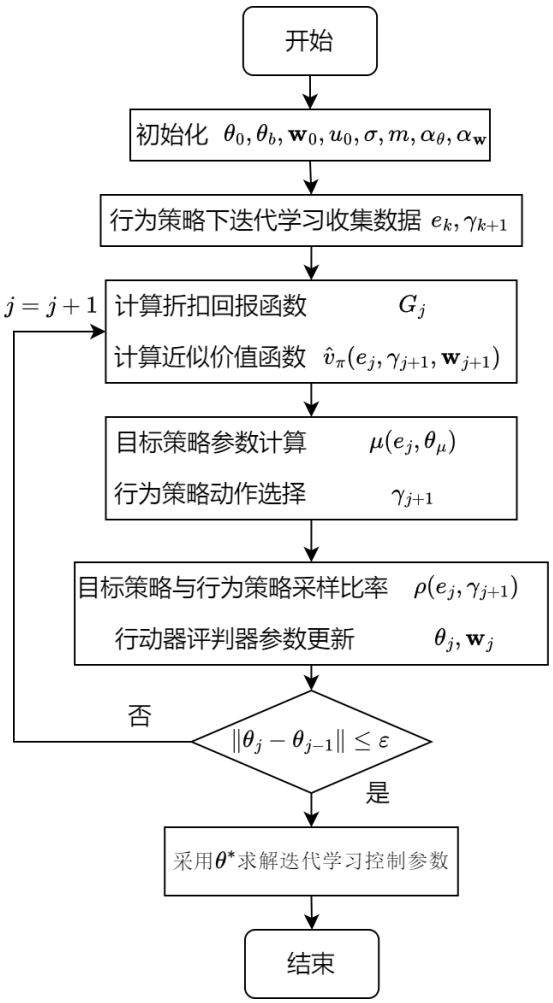

图 5-1 算法 5-1 的流程图

算法 5 - 1: off-policy 策略梯度方法的参数最优迭代学习控制

初始化: 初始输入 u_0，目标策略参数向量 θ_0，行为策略参数向量 θ_b，行为策略和目标策略的高斯分布标准差 σ，权值向量 $\mathbf{w}_0$，最大迭代试验数 m，步长参数 $\alpha_\theta, \alpha_{\mathbf{w}}$，开始迭代。

输出: 迭代学习的最优目标策略参数 θ^*

开始循环 $k = 1 \to m$

环境交互 *将 u_k 输入到系统中进行一次迭代实验，并根据行为策略参数 θ_b 计算当前状态 e_k 的行为策略选择动作 γ_{k+1} 并代入迭代学习律 (4-1) 计算 u_{k+1}*

数据收集 *收集迭代学习在行为策略 $b(\cdot)$ 下的状态误差 e_k 和动作 γ_{k+1}*

k=k+1

结束循环

开始循环 $j = 1 \to m-1$

评判器评估 *将行为策略下第 k 次迭代实验的状态信息和动作信息分别代入式 (5-7) 和 (4-8) 求得评判器神经网络对价值函数的估计 $\hat{v}(e_j, \gamma_{j+1}, \mathbf{w}_j)$ 和折扣回报函数 G_j*

行动器策略更新 *根据式 (5-12) 和 (5-13) 计算目标策略分布的参数 $\mu(e_j, \theta_j)$*

采样比率 *计算相同状态 e_j 下目标策略 $\boldsymbol{\pi}(e_j)$ 和行为策略 $b(e_j)$ 选取相同动作 γ_{j+1} 的采样比率 $\rho(e_j, \gamma_{j+1})$*

评判器参数更新 *递推迭代式 (5-10) 更新近似价值函数参数 $\mathbf{w}_j$*

行动器参数更新 *代入式 (5-15) 更新策略参数 θ_{j+1}*

j=j+1

终止更新条件 *若相邻的目标策略参数 θ_j 和 θ_{j-1}，对任意的 ε，都满足 $\|\theta_j - \theta_{j-1}\| < \varepsilon$, ε 是某一小正数，停止迭代*

结束循环

定理 5.1: 假设折扣回报函数 G、近似状态价值函数 $\hat{v}_{\boldsymbol{\pi}}$、策略 $\boldsymbol{\pi}(\theta)$、式 (5-10) 和 (5-15) 中的学习步长序列 $\{\alpha_j\}_{j=0}^{\infty}$ 分别满足以下条件:

(1) 折扣回报函数 G、近似状态价值函数 $\hat{v}_{\boldsymbol{\pi}}$ 在任何状态和行为下都是有界的，即 $G(e,\gamma) \le B_R, \hat{v}_{\boldsymbol{\pi}}(e,\gamma) \le B_R$

(2) 重要性采样比率 $\rho(e,\gamma)$ 在行为策略下任意状态和行为组合处的值是有界的，即 $\rho(e,\gamma) \le B_R$。

(3) 参数化目标策略 $\boldsymbol{\pi}(\theta)$ 是一个关于目标策略参数 θ 的可微的策略，并且其梯度 $\nabla \ln \boldsymbol{\pi}(\gamma_{j+1}|e_j,\theta)$ 也是有界的，还同时符合 L-Lipschitz 条件。即对于任意的 $e,\gamma,\theta,\theta_p,\theta_q$，都存在常数 $\boldsymbol{B},\boldsymbol{L}$，满足数学关系 $\|\nabla \ln \boldsymbol{\pi}(\gamma|e,\theta)\| \leq \boldsymbol{B}$ 和 $\|\nabla_\theta \ln \boldsymbol{\pi}(\gamma|e,\theta_p) - \nabla_\theta \ln \boldsymbol{\pi}(\gamma|e,\theta_q)\| \leq \boldsymbol{L}\|\theta_p - \theta_q\|$。

(4) 行动器和评判器迭代更新式的学习步长序列 $\{\alpha_j\}_{j=0}^{\infty}$ 满足条件 $\lim_{j\to\infty}\{\alpha_j\} = 0$、$\sum_{j=0}^{\infty}\alpha_j = \infty$ 和 $\sum_{j=0\infty}^{\infty}\alpha_j^2 < \infty$。

这样性能指标 $J(j)_{j=0}^{\infty}$ 将从任意的 θ_0 沿着梯度方向 $\mathbf{g}(\theta)$ 开始，遵循 off-policy 下的策略梯度参数更新公式 (5-15)，最终性能指标收敛，满足 $\lim_{j\to\infty}\frac{\partial \boldsymbol{J}(\theta_j)}{\partial\theta} = 0$。

算法 5-1 描述了在 off-policy 策略梯度下，学习并优化参数最优迭代学习控制目标策略参数 θ 的过程，其中通过离线学习求解性能指标函数的最小值得到目标策略 $\boldsymbol{\pi}$。首先，我们将行为策略和目标策略划分，并通过行为策略与系统的交互获得全部状态信息 e_j 和动作信息 γ_{j+1}。随后，我们利用评判器神经网络近似状态价值函数和 off-policy 策略梯度理论来更新目标策略和近似价值函数。最终，我们获得了关于该迭代学习控制的最优目标策略参数 θ^* 和估计权值向量 $\mathbf{w}_j$，并且将最优目标策略参数应用于迭代学习中求解不同误差状态 e_k 下的最优参数 γ_{k+1}^*。由于该算法是在离轨策略下离线进行的，因此显著减小了行为策略对目标策略和状态价值函数性能的影响，也拓宽了算法的适用场景。在接下来的部分，我们将介绍本章提出的仿真实验设计，并通过实验结果展示数据驱动方法与基于模型方法的性能的优劣。

5.4 数值仿真

在本节中，通过将参数最优迭代学习控制 (POILC) 和 off-policy 策略梯度方法的参数最优迭代学习控制中的行为策略 (Behavior Policy) 及其目标策略 (Idea Policy) 分别应用于三轴龙门式机器人模型作为被控对象设计仿真实验验证算法 5-1 的有效性。此处的三轴龙门式机器人模型状态空间方程以及参考轨迹与第四章仿真实验中一致。

本章中仿真实验设计如流程图5-1所示一致，首先利用行为策略为算法 5-1 收集数据，然后根据数据更新学习最优目标策略参数 θ^*，最后将其应用于机器人模型系统进行参数最优迭代学习控制，依旧通过 200 次迭代学习来实验所提出的算法性能，初始输入条件为零。与第四章的一类 on-poicy 策略梯度算法不同，因为不再对策略的标准差参数化和提升目标策略的泛化能力，行为策略和目标策略的标准差 $\sigma = 1000$；行动器的梯

度迭代式 (5-15) 的学习律 $\alpha_\theta = 0.005$；评判器的激活函数为：

$$f(x) = \frac{1}{1+e^{-x}} \tag{5-18}$$

根据梯度下降优化算法理论，评判器的步长参选择满足下式：

$$\alpha_{\mathbf{w}} = 0.00005 \times \varphi(e_j)^T \varphi(e_j) \tag{5-19}$$

行为策略与系统交互进行迭代学习每次实验的行为策略参数 θ_b 由第四章的算法 4-1 在 (4-18) 中的步长参数 $\alpha_k = 0.3$ 迭代 7 次所得到，具体取值为：

$$\begin{aligned}\theta_b = [&0.2, 0.2, 0.2044, 0.2, 520.3169, 0.2, 586.0613, 0.2, 586.7592, 0.2, 586.7662,\\ &0.2, 586.7662, 0.2, 586.7662, 0.2, 586.7662, 0.2, 586.7662, 0.2]^T\end{aligned} \tag{5-20}$$

其余涉及的参数选择设置取值不变。

首先，仿真实验比较了 POILC、算法 5-1 的目标策略和行为策略这三个算法分别应用于参数最优迭代学习控制的前 100 次迭代实验的误差范数曲线，见图5-2。其次，与上一章仿真相似，绘制了目标策略的参数 θ_μ 范数曲线图5-3以展示算法 5-1 中 off-policy 策略梯度在训练过程的收敛性和稳定性。然后，为了更加全面地展示算法 5-1 的收敛性和有效性，不仅绘制了图5-4用来展示了算法 5-1 中评判器单层神经网络关于状态价值函数的估计权值向量范数的曲线，还绘制了图5-5展示了算法 5-1 中价值函数损失的收敛曲线，用以评价估计的状态价值函数与实际奖励（或回报）之间的差距。

由图5-2及其第 50 次至 70 次迭代实验放大图可以看出，当已知系统动态模型时，基于模型的 POILC 算法性能最好，在第 60-65 次之间迭代实验使得系统达到了收敛，这与前一章的仿真实验结果相符，即基于模型的方法的收敛特性相比于不依赖于系统动力学参数信息的方法往往取得更好的效果。

通过观察图5-2的行为策略和目标策略下的误差范数曲线，说明 off-policy 离线学习的策略梯度算法可以对目标策略进行优化；其次对比 POILC、PPO-POILC 和目标策略下的误差范数曲线，表明算法 5-1 的迭代学习控制过程的收敛特性已经达到近似最优，佐证了 off-policy 离轨策略下的参数最优迭代学习算法可以在降低系统对模型的依赖情况下依旧能满足参数最优迭代学习的收敛特性性要求，验证了算法 5-1 的有效性。

图5-3描述了目标策略参数 θ_μ 在 off-policy 策略梯度更新中的学习情况。结果表明算法 5-1 目标策略参数随着策略梯度更新最终都能稳定收敛到一个固定值，验证了算法 5-1 的收敛性；图中曲线稳步上升最终趋向于固定值，佐证算法 5-1 训练过程的稳定性。

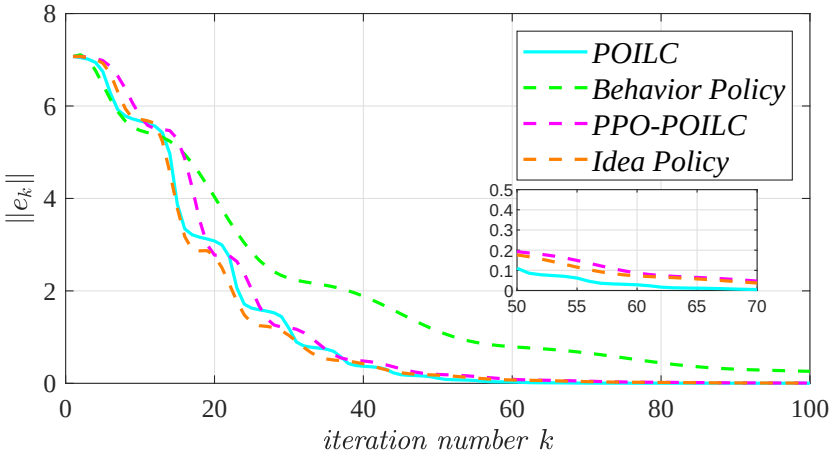

图 5-2 前 100 次迭代实验的误差范数收敛图

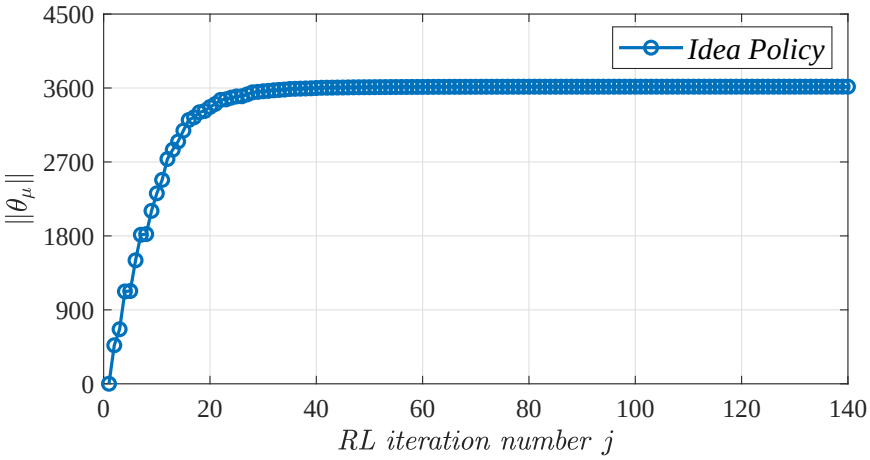

图 5-3 算法 5-1 策略参数 θ_μ 的范数

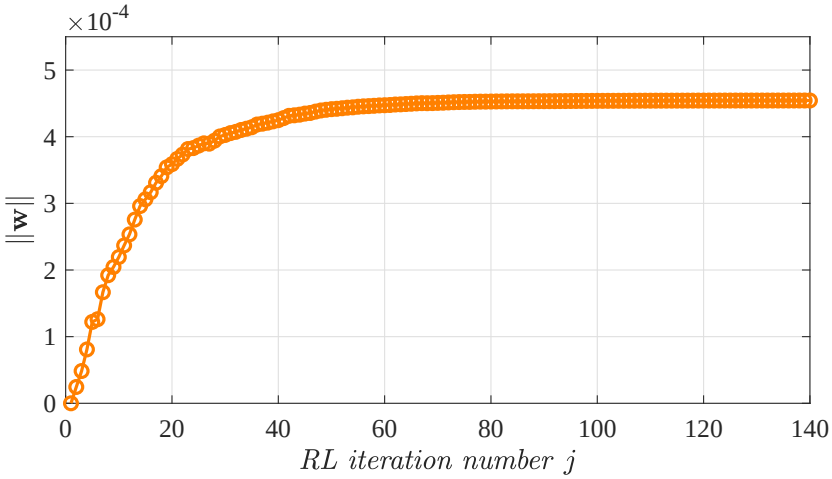

图 5-4 算法 5-1 近似状态价值函数权值向量范数

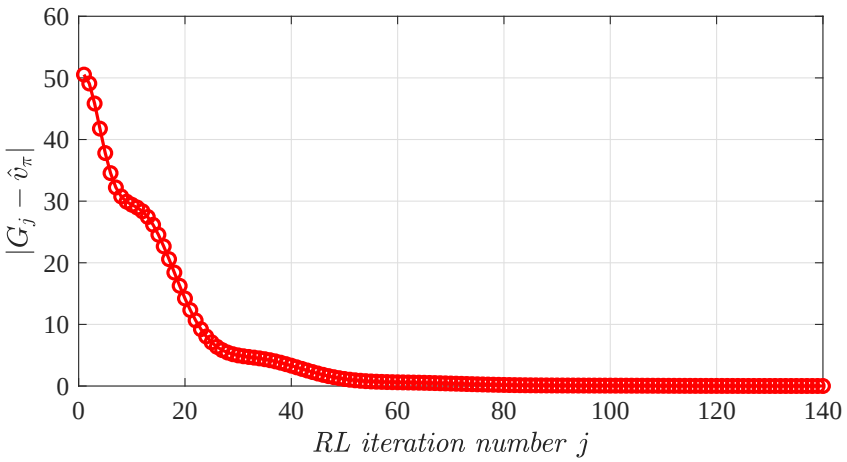

图 5-5 算法 5-1 价值函数的损失曲线

参考近似价值函数权重图5-4与价值函数损失曲线图5-5所示的仿真结果，可以观察到近似价值函数所对应的权值向量的范数从由初始值 0 逐步递增至某一稳定数值的过程，与此同时价值损失函数的值则表现出逐渐下降并趋于稳定的趋势。这一系列现象说明了模型在学习进程中取得了实质性的进步，并且意味着算法在训练环节有较大可能性已收敛至一种较为稳健的目标策略。这种收敛性反映出了算法对系统环境进行了全面的探索，从而使评判器具备了更高精度的状态价值估算能力。但是，尽管呈现出明显的收敛迹象，因为参数最优迭代学习控制在强化学习的折扣回报函数4-8存在“短视”的局限性，几乎只考虑最小化当下的累积回报，因此尚不能断言已经找到全局最优策略，因为利用这种折扣回报函数有可能导致收敛到了局部最优解。

5.5 本章小结

本章仍然研究如何在减少迭代学习律设计算法对模型参数依赖的条件下依旧使得被控系统在迭代学习过程的收敛特性达到最优迭代学习控制的收敛性要求这一问题。在第四章中研究了 on-policy 的策略梯度应用于参数最优迭代学习控制过程的算法，然而这种方法是 on-policy 下在线学习的。虽然这种算法具有实时性高且学习过程中稳定性强的优势，但是在高速系统中在线获取有效的训练数据有时可能会很困难，并且可能由于在线计算资源有限无法满足在线学习计算复杂度的要求。

本章依旧以参数最优迭代学习控制问题为背景，首先介绍了 off-policy 策略梯度理论，并对理论需要满足的条件作了简要阐述。接着在“行动器-评判器”的算法结构下，利用单层神经网络设计了评判器的近似状态价值函数估计过程；接着通过固定高斯分布的标准差，对目标策略进行参数化，并给出了 off-policy 策略梯度下目标策略的参数

更新迭代式；然后提出了一种离线学习的 off-policy 策略梯度方法的参数最优迭代学习控制算法，给出了算法收敛性所需满足的条件。最后，在多轴系统下进行数值仿真，并绘制误差收敛曲线图，以比较基于模型、同轨和离轨方法的收敛效果。结果表明，算法 5-1 可以解决了迭代学习控制律的设计问题，实现了跟踪误差范数的快速收敛，从而验证了该算法的有效性。

总结与展望

为实现被控系统对参考轨迹的快速收敛完全跟踪，一类传统基于模型的最优迭代学习控制理论设计方法通常基于系统动力学的高精度模型参数信息，而在实际应用中高精度的系统模型通常非常昂贵，甚至根本不可能获得，进而阻碍了最优迭代学习控制理论的发展。本文以解决最优迭代学习控制方法对系统模型参数信息依赖的问题为出发点，以基于模型的最优迭代学习控制算法为研究背景，运用强化学习算法作为改进研究方法，主要探索了一种新的减少对模型参数信息依赖的最优迭代学习控制方案。具体研究成果及创新点概括如下：

(1) 针对预测最优迭代学习律设计依赖于系统模型参数信息的问题，基于 off-policy Q-学习理论将最优性问题转化为 Q 函数的最值问题，减小了对系统模型参数信息的依赖，提出了 off-policy Q-学习的数据驱动形式的迭代学习算法，并给出算法相应的实现流程和收敛条件。

(2) 基于价值函数 Q-学习算法存在诸如收敛速度慢和对动作空间离散化要求高的局限，亟待能够直接学习动作策略的方法。本文以基于模型的参数最优迭代学习控制为研究背景，基于 on-policy 策略梯度理论提出数据驱动的参数最优迭代学习控制算法，并应用价值函数估计基线和近端策略优化方法在“行动器-评判器”算法框架下给出改进的算法。这些算法都不依赖于显式模型参数信息且都能满足迭代学习快速收敛的要求。

(3) on-policy 策略梯度算法的目标策略也是用于生成数据的行为策略，虽然该类算法收敛，但在学习过程中目标策略的突变将会对实际系统运行产生较大的波动。另外 on-policy 在线学习的方法习对数据的利用率较低。针对这两种限制，本文基于 off-policy 策略梯度离线学习方法，对目标策略与行为策略进行分离，在一段学习周期内不改变行为策略，避免目标策略波动的影响，同时采样的数据可以多次使用提高了数据利用率。最后在“行动器-评判器”算法框架下给出算法实现流程并结合仿真实验验证算法有效性。

本文的后续工作展开可以从以下几个角度考虑：

(1) 本文的策略梯度算法中，参数最优迭代学习控制问题所考虑的折扣回报函数存在“短视”的问题，这将更容易导致在强化学习的过程中收敛于局部次优解，因为它无法全面考虑长期累积的奖励和未来的影响。

(2) 尽管基于强化学习的算法使得迭代学习控制可以取得了良好的收敛性能，但是

关于收敛速度的量化工作还有待研究。通过量化迭代学习控制中的收敛速度，相应地改进已有的算法参数，以进一步提高强化学习在迭代学习控制中的效果。

(3) 目前研究的最优迭代学习控制理论性能指标结构简单，往往只考虑收敛的最优性。通过深度强化学习诸如 DQN、DDPG、TRPO 等算法，可以更加灵活地定义奖励函数，使得迭代学习过程能够更好地适应不同的任务和环境，从而提高最优迭代学习算法的性能和适用性。

(4) 目前关于强化学习和迭代学习控制交叉领域内其期望轨迹在一个迭代学习过程中是固定不变的，针对变期望轨迹的不依赖系统模型参数的迭代学习理论仍然有待研究。

(5) 本文研究的被控对象均为无时滞系统，下一步将考虑设计一种时滞系统模型非依赖的迭代学习控制方法使工程中常见的时滞系统完全跟踪参考轨迹。

(6) 本文研究的被控对象均为单智能体系统，而对于多智能体系统甚至是广义多智能体迭代学习一致性控制的研究有待进一步探索。

参考文献

[1] Riaz S, Hui L, Aldemir M S, et al. A future concern of iterative learning control: A survey[J]. Journal of Statistics and Management Systems, 2021, 24(6): 1301-1322.

[2] Uchiyama M. Formation of high-speed motion pattern of a mechanical arm by trial[J]. Transactions of the Society of Instrument and Control Engineers, 1978, 14(6): 706-712.

[3] Arimoto S, Kawamura S, Miyazaki F. Bettering operation of robots by learning[J]. Journal of Robotic systems, 1984, 1(2): 123-140.

[4] Boming S. On iterative learning control[D]. Singapore: National University of Singapore, 1996.

[5] Meng T, He W. Iterative learning control of a robotic arm experiment platform with input constraint[J]. IEEE Transactions on Industrial Electronics, 2017, 65(1): 664-672.

[6] Xing X, Liu J. Modeling and robust adaptive iterative learning control of a vehicle-based flexible manipulator with uncertainties[J]. International Journal of Robust and Nonlinear Control, 2019, 29(8): 2385-2405.

[7] Wu J, Zhang B, Wang L, et al. An iterative learning method for realizing accurate dynamic feedforward control of an industrial hybrid robot[J]. Science China Technological Sciences, 2021, 64(6): 1177-1188.

[8] Lee T, Tan K K, Lim S, et al. Iterative learning control of permanent magnet linear motor with relay automatic tuning[J]. Mechatronics, 2000, 10(1-2): 169-190.

[9] 臧汗青. 基于实时轮廓误差估计的三轴运动平台迭代学习控制[D]. 沈阳: 沈阳工业大学, 2022.

[10] 梁建智, 邱彪, 陈宇燕, 等. 基于数据驱动的数控机床自适应迭代学习控制[J]. 机床与液压, 2021, 49: 50-54.

[11] 王慧霞. 基于迭代学习与交叉耦合的数控机床进给伺服系统运动控制[D]. 兰州: 兰州理工大学, 2021.

[12] Choi J Y, Do H M. A learning approach of wafer temperature control in a rapid thermal processing system[J]. IEEE transactions on Semiconductor Manufacturing, 2001, 14(1): 1-10.

[13] Li M, Chen T, Cheng R, et al. Dual-loop iterative learning control with application to an ultraprecision wafer stage[J]. IEEE Transactions on Industrial Electronics, 2021, 69(11): 11590-11599.

[14] Ren J C, Liu D, Wan Y. Model-free adaptive iterative learning control method for the czochralski silicon monocrystalline batch process[J]. IEEE Transactions on Semiconductor Manufacturing, 2021, 34(3): 398-407.

[15] Zuo Z, Yang X, Li Z, et al. MPC-based cooperative control strategy of path planning and trajectory tracking for intelligent vehicles[J]. IEEE Transactions on Intelligent Vehicles, 2020, 6(3): 513-522.

[16] Hou C, Li X, Wang H, et al. Fuzzy linear extended states observer-based iteration learning fault-tolerant control for autonomous underwater vehicle trajectory-tracking system[J]. IET Control Theory & Applications, 2023, 17(3): 270-283.

[17] Bellman R, Kalaba R. Dynamic programming and adaptive processes: mathematical foundation[J]. IRE Transactions on Automatic Control, 1960(1): 5-10.

[18] Sutton R S, Barto A G. Reinforcement learning: An introduction[J]. Robotica, 1999, 17(2): 229-235.

[19] Szita I. Reinforcement learning in games[M]. Berlin, Germany: Springer, 2012: 539-577.

[20] Ye D, Chen G, Zhang W, et al. Towards playing full moba games with deep reinforcement learning[J]. Advances in Neural Information Processing Systems, 2020, 33: 621-632.

[21] Wang Y, Dong L, Sun C. Cooperative control for multi-player pursuit-evasion games with reinforcement learning[J]. Neurocomputing, 2020, 412: 101-114.

[22] Ibarz J, Tan J, Finn C, et al. How to train your robot with deep reinforcement learning: lessons we have learned[J]. The International Journal of Robotics Research, 2021, 40(4-5): 698-721.

[23] Salvato E, Fenu G, Medvet E, et al. Crossing the reality gap: A survey on sim-to-real transferability of robot controllers in reinforcement learning[J]. IEEE Access, 2021, 9: 153171-153187.

[24] Zhu K, Zhang T. Deep reinforcement learning based mobile robot navigation: A review[J]. Tsinghua Science and Technology, 2021, 26(5): 674-691.

[25] Kiran B R, Sobh I, Talpaert V, et al. Deep reinforcement learning for autonomous driving: A survey[J]. IEEE Transactions on Intelligent Transportation Systems, 2021, 23(6): 4909-4926.

[26] Yan Z, Kreidieh A R, Vinitsky E, et al. Unified automatic control of vehicular systems with reinforcement learning[J]. IEEE Transactions on Automation Science and Engineering, 2022, 20(2): 789-804.

[27] Wu Y, Liao S, Liu X, et al. Deep reinforcement learning on autonomous driving policy with auxiliary critic network[J]. IEEE transactions on neural networks and learning systems, 2021, 34(7): 3680-3690.

[28] 陈帅. 基于强化学习的微电网能量管理与调度[D]. 北京: 北京科技大学, 2023.

[29] 许德州. 基于 Soft Actor-Critic 算法的电动汽车复合储能系统能量管理策略研究[D]. 北京: 中国矿业大学, 2022.

[30] Liu Y, Zhang D, Gooi H B. Optimization strategy based on deep reinforcement learning for home energy management[J]. CSEE Journal of Power and Energy Systems, 2020, 6(3): 572-582.

[31] Wang N, Gao Y, Zhao H, et al. Reinforcement learning-based optimal tracking control of an unknown unmanned surface vehicle[J]. IEEE Transactions on Neural Networks and Learning Systems, 2020, 32(7): 3034-3045.

[32] Li Y, Gao W, Huang S, et al. Data-driven optimal control strategy for virtual synchronous generator via deep reinforcement learning approach[J]. Journal of Modern Power Systems and Clean Energy, 2021, 9(4): 919-929.

[33] Guo L, Zhao H. Online adaptive optimal control algorithm based on synchronous integral reinforcement learning with explorations[J]. Neurocomputing, 2023, 520: 250-261.

[34] 毕京瑞. 基于深度强化学习的数字货币量化交易系统[D]. 广州: 广东财经大学, 2022.

[35] Sun S, Wang R, An B. Reinforcement learning for quantitative trading[J]. ACM Transactions on Intelligent Systems and Technology, 2023, 14(3): 1-29.

[36] Hambly B, Xu R, Yang H. Recent advances in reinforcement learning in finance[J]. Mathematical Finance, 2023, 33(3): 437-503.

[37] Owens D, Hatonen J, Daley S. Robust monotone gradient-based discrete-time iterative learning control[J]. International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal, 2009, 19(6): 634-661.

[38] Gu P, Tian S, Chen Y. Iterative learning control based on Nesterov accelerated gradient method[J]. IEEE Access, 2019, 7: 115836-115842.

[39] Togai M, Yamano O. Analysis and design of an optimal learning control scheme for industrial robots: A discrete system approach[A]. Proceedings of the 24th IEEE Conference on Decision and Control[C]. Fort Lauderdale, USA: IEEE, 1985: 1399-1404.

[40] Chu B, Owens D. Accelerated predictive norm-optimal iterative learning control[J]. Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering, 2011, 225(6): 744-759.

[41] Amann N, Owens D H, Rogers E. Iterative learning control using optimal feedback and feedforward actions[J]. International Journal of Control, 1996, 65(2): 277-293.

[42] Hatzikos V, Hätönen J, Owens D. Genetic algorithms in norm-optimal linear and nonlinear iterative learning control[J]. International Journal of control, 2004, 77(2): 188-197.

[43] Tayebi A. Adaptive iterative learning control for robot manipulators[J]. Automatica, 2004, 40(7): 1195-1203.

[44] Jiang Z, Chu B. Norm Optimal Iterative Learning Control: A Data-Driven Approach[J]. IFAC-PapersOnLine, 2022, 55(12): 482-487.

[45] Tharayil D, Alleyne A G. A survey of iterative learning control: A learning-based method for highperformance tracking control[J]. IEEE Control systems magazine, 2006, 26(3): 96-114.

[46] Tao Y H, Chao C H. How online tacit knowledge learning influences traditional classroom learning? A case study[J]. Journal of Statistics and Management Systems, 2013, 16(4-5): 293-308.

[47] Simon H A. Organizations and markets[J]. Journal of economic perspectives, 1991, 5(2): 25-44.

[48] Sutton R S. Learning to predict by the methods of temporal differences[J]. Machine learning, 1988, 3: 9-44.

[49] Singh S P, Sutton R S. Reinforcement learning with replacing eligibility traces[J]. Machine learning, 1996, 22(1): 123-158.

[50] Watkins C J, Dayan P. Q-learning[J]. Machine learning, 1992, 8: 279-292.

[51] Williams R J. Simple statistical gradient-following algorithms for connectionist reinforcement learning[J]. Machine learning, 1992, 8: 229-256.

[52] Barto A G, Sutton R S, Anderson C W. Looking back on the actor–critic architecture[J]. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2020, 51(1): 40-50.

[53] Fröhlich L. Data-Efficient Controller Tuning and Reinforcement Learning[D]. Zurich: ETH Zurich, 2022.

[54] Zhang Y, Chu B, Shu Z. A preliminary study on the relationship between iterative learning control and reinforcement learning[J]. IFAC-PapersOnLine, 2019, 52(29): 314-319.

[55] Poot M, Portegies J, Oomen T. On the role of models in learning control: Actor-critic iterative learning control[J]. IFAC-PapersOnLine, 2020, 53(2): 1450-1455.

[56] Shi J, Wen K, Xu X, et al. Design of Nonlinear Iterative Learning Control Based on Deep Reinforcement Learning Algorithm[A]. Proceedings of the 10th Data Driven Control and Learning Systems Conference[C]. Suzhou, China: IEEE, 2021: 722-727.

[57] Li J, Tian S, Peng Y, et al. Data-driven-based Predictive Optimal for a class of Iterative Learning Control by Q-learning method[A]. Proceedings of the 12th Data Driven Control and Learning Systems Conference[C]. 1214-1220.

[58] Lan T, Lin H, Li B. Kernel-based auto-associative P-type iterative learning control strategy[J]. Journal of Systems Engineering and Electronics, 2020, 31(2): 383-392.

[59] Low E S, Ong P, Cheah K C. Solving the optimal path planning of a mobile robot using improved Q-learning[J]. Robotics and Autonomous Systems, 2019, 115: 143-161.

[60] Sutton R S, Barto A G. Reinforcement learning: An introduction[M]. Cambridge, USA: MIT press, 2018: 317-335.

[61] Gu Y, Cheng Y, Chen C P, et al. Proximal policy optimization with policy feedback[J]. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2021, 52(7): 4600-4610.

[62] Van Hezewijk L, Dellaert N, Van Woensel T, et al. Using the proximal policy optimisation algorithm for solving the stochastic capacitated lot sizing problem[J]. International Journal of Production Research, 2023, 61(6): 1955-1978.

[63] Dayan P, Watkins C. Q-learning[J]. Machine learning, 1992, 8(3): 279-292.

[64] Moore K L. Iterative learning control: An expository overview[M]. Boston, USA: Birkhäuser Boston, 1999: 151-214.

[65] Amann, Notker and Owens, David H and Rogers, Eric. Predictive optimal iterative learning control[J]. International Journal of Control, 1998, 69(2): 203-226.

[66] Dai P, Weld D S, Goldsmith J, et al. Topological value iteration algorithms[J]. Journal of Artificial Intelligence Research, 2011, 42: 181-209.

[67] Bradtke S, Ydstie B, Barto A. Adaptive linear quadratic control using policy iteration[A]. Proceedings of 1994 American Control Conference[C]. Baltimore, USA: IEEE, 1994: 3475-3479.

[68] 陈永康. 基于迭代强化学习的多智能体系统分布式最优一致控制[D]. 南京: 南京邮电大学, 2023.

[69] Zhang Y, Chu B, Shu Z. Model-free predictive optimal iterative learning control using reinforcement learning[A]. Proceedings of 2022 American Control Conference[C]. Atlanta, USA: IEEE, 2022: 3279-3284.

[70] Cai Q, Yang Z, Lee J D, et al. Neural temporal difference and q learning provably converge to global optima[J]. Mathematics of Operations Research, 2024, 49(1): 619-651.

[71] Jang B, Kim M, Harerimana G, et al. Q-learning algorithms: A comprehensive classification and applications[J]. IEEE access, 2019, 7: 133653-133667.

[72] Agarwal A, Kakade S M, Lee J D, et al. On the theory of policy gradient methods: Optimality, approximation, and distribution shift[J]. Journal of Machine Learning Research, 2021, 22(98): 1-76.

[73] Owens D H, Feng K. Parameter optimization in iterative learning control[J]. International Journal of Control, 2003, 76(11): 1059-1069.

[74] Zhang Y, Chu B, Shu Z. Parameter Optimal Iterative Learning Control Design: from Model-based, Data-driven to Reinforcement Learning[J]. IFAC-PapersOnLine, 2022, 55(12): 494-499.

[75] Kaelbling L, Littman M, Moore A. Reinforcement learning: A survey[J]. Journal of Artificial Intelligence Research, 1996, 4: 237-285.

[76] Robbins H, Monro S. A stochastic approximation method[J]. The annals of mathematical statistics, 1951: 400-407.

[77] Ratcliffe J D. Iterative learning control implemented on a multi-axis system[D]. Southampton: University of Southampton, 2005.

[78] Degris T, White M, Sutton R S. Off-policy actor-critic[A]. Proceedings of the 29th International Coference on International Conference on Machine Learning[C]. Edinburgh, Scotland: PMLR, 2012: 179-186.

[79] Xu T, Yang Z, Wang Z, et al. Doubly robust off-policy actor-critic: Convergence and optimality[A]. Proceedings of the 38th International Conference on Machine Learning[C]. PMLR, 2021: 11581-11591.