- 小问 3 求解算法：多体动力学 + 强化学习
  - 1. 算法一：深度确定性策略梯度 (Deep Deterministic Policy Gradient, DDPG)
    - 1.1 原理概述
    - 1.2 适用场景
    - 1.3 优缺点分析
  - 2. 算法二：近端策略优化 (Proximal Policy Optimization, PPO)
    - 2.1 原理概述
    - 2.2 适用场景
    - 2.3 优缺点分析
  - 3. 推荐求解方法
    - 推荐算法：近端策略优化 (PPO)

小问 3 求解算法：多体动力学 + 强化学习

针对小问 3 建立的 **多体动力学 + 强化学习协同控制模型**，该问题属于高维连续状态空间和连续动作空间的 **马尔可夫决策过程 (MDP)**。由于涉及到复杂的身体平衡和多关节协同，传统的控制方法（如 PID、MPC）建模难度极大，因此深度强化学习 (Deep RL) 是最佳选择。推荐以下两种主流算法。

---

1. 算法一：深度确定性策略梯度 (Deep Deterministic Policy Gradient, DDPG)

1.1 原理概述

DDPG 是一种基于 **Actor-Critic** 架构的算法，专门用于解决连续动作空间的 RL 问题。

- **Actor (策略网络)**：输入状态 s，确定性地输出具体的动作值 $a = \mu(s)$。
- **Critic (价值网络)**：输入状态 s 和动作 a，评估该动作的 Q 值 $Q(s, a)$。 算法利用经验回放池 (Experience Replay) 打破数据相关性，并使用目标网络 (Target Networks) 软更新技术来保证训练稳定性。

1.2 适用场景

- **连续动作控制**：如机器人关节角度控制、自动驾驶方向盘转角。
- **确定性策略需求**：当需要确定的输入输出映射时。

1.3 优缺点分析

- **优点**：
  - **直接输出连续动作**：无需对动作空间进行离散化。
  - **样本效率较高**：属于异策略 (Off-policy) 算法，可以重复利用历史经验数据。
- **缺点**：
  - **超参数敏感**：学习率、噪声参数等难以调节。
  - **收敛不稳定**：容易受 Q 值高估 (Overestimation) 影响，导致策略崩溃。
  - **探索能力有限**：依赖于在动作上添加随机噪声（如 OU 噪声）进行探索。

---

2. 算法二：近端策略优化 (Proximal Policy Optimization, PPO)

2.1 原理概述

PPO 是目前最流行的 **策略梯度 (Policy Gradient)** 算法之一。它通过限制新策略与旧策略之间的差异（KL 散度或 Clip 操作），防止策略更新步长过大导致性能剧烈下降。 PPO 通常采用 **Actor-Critic** 架构，但在输出动作时，Actor 输出的是动作分布（如高斯分布的均值和方差），从中采样得到动作。

2.2 适用场景

- **复杂的机器人控制**：OpenAI Five、Boston Dynamics 机器人仿真多采用此类算法。
- **需要稳健训练**：在长时间训练中保持性能单调上升。
- **高维状态空间**：处理 30+ 维度的机器人状态游刃有余。

2.3 优缺点分析

- **优点**：

    - **稳定性极强**：Clip 机制保证了更新幅度可控，极少出现训练崩溃。
    - **调参简单**：相比 DDPG，PPO 对超参数不那么敏感，默认参数通常就能跑出不错的结果。
    - **兼顾探索与利用**：通过随机策略采样自然地进行探索。
- **缺点**：
    - **样本效率较低**：属于同策略 (On-policy) 算法，每次更新只能使用当前策略采集的数据，数据利用率不如 DDPG。

---

3. 推荐求解方法

推荐算法：近端策略优化 (PPO)

推荐理由：

1. **收敛稳定性 (首要考虑)**：在数学建模竞赛中，时间有限，没有太多时间反复调试 DDPG 难以收敛的“玄学”参数。PPO 以其鲁棒性著称，能够在较短时间内获得一个可行的控制策略。
2. **动作平滑性**：小问 3 要求“协同控制”且“动作平滑”。PPO 输出的高斯分布可以通过调整方差项逐步收敛，且可以通过在 Reward 中加入控制量惩罚项，更容易训练出平滑的动作轨迹。
3. **业界标准**：PPO 目前是机器人控制领域（如 Isaac Gym, MuJoCo 任务）的 Baseline 算法，具有广泛的开源代码参考（如 Stable Baselines3），容易实现和复现。
4. **处理复杂约束**：对于重心平衡这种复杂的隐式约束，PPO 能够通过 Reward Shaping 很好地学习到保持平衡的策略边界。

求解流程：

1. **环境搭建**：使用 Python (PyBullet 或 MuJoCo) 搭建 11 刚体动力学环境，定义 State（角度、重心）和 Action（关节增量）。
2. **网络构建**：使用 PyTorch 构建 Actor 和 Critic 网络（例如 3 层 MLP）。
3. **训练循环**：
    - 机器人与环境交互，收集一批轨迹数据。
    - 计算优势函数 (Advantage)。
    - 使用 PPO Clip Loss 更新网络参数。
4. **结果验证**：将训练好的 Policy 应用于模型，记录关节曲线，检查是否满足左转 45° 和重心稳定的要求。