Reinforcement Learning for 1D Adaptive Integration

Team KVS
Kavya Bhalodi (2023101031), Varun Gupta (2023101108),
Shivam Gupta (2023101062)

May 2025

Contents

1 Overview

This document describes a Python-based implementation for 1D adaptive integration using Reinforcement Learning (RL). The implementation uses Proximal Policy Optimization (PPO) to train an agent that adaptively refines intervals over the integration domain for high-accuracy approximation with fewer evaluations.

The agent operates in a custom Gym environment called `EnhancedAdaptiveIntegrationEnv` that models adaptive quadrature as a sequential decision-making problem.

2 Imports and Dependencies

The core libraries include:

- `gymnasium` – custom RL environment for integration.
- `stable_baselines3` – for PPO-based training.
- `scipy.integrate` – high-precision integration reference using `quad`.
- `numpy, matplotlib` – numerical and plotting utilities.

3 Learning Rate Schedule

The function `linear_schedule` defines a learning rate that decays linearly over time, enabling exploration initially and fine-tuning in later stages.

```
def linear_schedule(initial_value, final_value):
    def func(progress_remaining):
        return final_value + progress_remaining * (initial_value -
            final_value)
    return func
```

4 Early Stopping Callback

The `EarlyStopCallback` halts training if improvement stagnates. It monitors:

- Mean episode reward
- Reward improvement over time

5 Environment: EnhancedAdaptiveIntegrationEnv

5.1 Purpose

Defines a Gym environment for 1D numerical integration. The agent selects which interval to split and how, in order to minimize the integral error using a limited number of steps.

5.2 Initialization

Parameters:

- Domain bounds $[a, b]$
- Maximum number of intervals
- Target function $f(x)$
- True integral value via high-accuracy quadrature

5.3 Observation Space

For each interval:

- Location, width
- Function evaluations: endpoints, midpoint, quartiles
- Integration estimates: Trapezoidal, Simpson, Gauss-Legendre
- Error estimates: Richardson extrapolation, difference between methods
- Curvature and variability

5.4 Action Space

A 3D continuous vector:

$$[\texttt{interval\_index}, \texttt{split\_ratio}, \texttt{strategy}]$$

- Interval index: which interval to split
- Split ratio: proportion of where to split
- Strategy: heuristic, adaptive, or error-based

5.5 Integration Method

- **5-point Gauss-Legendre Quadrature:**

$$\int_a^b f(x)dx \approx \sum_{i=1}^{5} w_i f\left(\frac{b-a}{2}x_i + \frac{a+b}{2}\right)$$

- High-order accuracy and used consistently for both estimates and reference.

5.6 Error Estimation

- Richardson extrapolation compares full vs half interval integrations to estimate local error.
- Aids both reward design and feature computation.

5.7 Step Function

At each environment step:

1. Select an interval and compute its refined split point.
2. Add two new sub-intervals.
3. Estimate updated integral.
4. Compute reward based on error reduction.

6 Reward Function

Reward is based on:

$$R = \alpha \cdot (\text{error reduction}) - \beta \cdot (\text{evaluations}) + \gamma \cdot (\text{future benefit})$$

Includes:

- Immediate reward for decreasing error
- Forecasted benefit for exploring complex intervals
- Penalty for excess function evaluations
- Terminal bonus for high final accuracy

7 Training Setup

- PPO algorithm from Stable-Baselines3
- Policy network: 3 hidden layers [256, 256, 128]
- Learning rate: linear decay from 3×10^{-4} to 1×10^{-5}
- Training time: 1 million steps
- Curriculum learning: functions of increasing complexity

8 Conclusion

We implemented a reinforcement learning system for adaptive 1D numerical integration. The agent effectively learns to split intervals based on error, curvature, and variability. Compared to fixed rule-based schemes, our RL approach offers improved adaptability and precision.