CS258: Introduction to Reinforcement Learning Project 01: Value Based Algorithms

University of California, Riverside

Darshil Devesh Shukla
`dshuk003@ucr.edu`

October 14, 2025

1 GridWorld Results on world1

In this section we report the learning curves and final policies for the four RL methods (On-Policy First-Visit MC Control, SARSA, Q-Learning, and Double Q-Learning), and the training curves and final policies for the two exact methods (Value Iteration, Policy Iteration). Consistent with the assignment, the horizontal axis in all plots denotes the number of training steps/iterations. We also compare the computational efficiency of Value vs. Policy Iteration and discuss agreement between the exact and learned policies.

1.1 Exact Methods

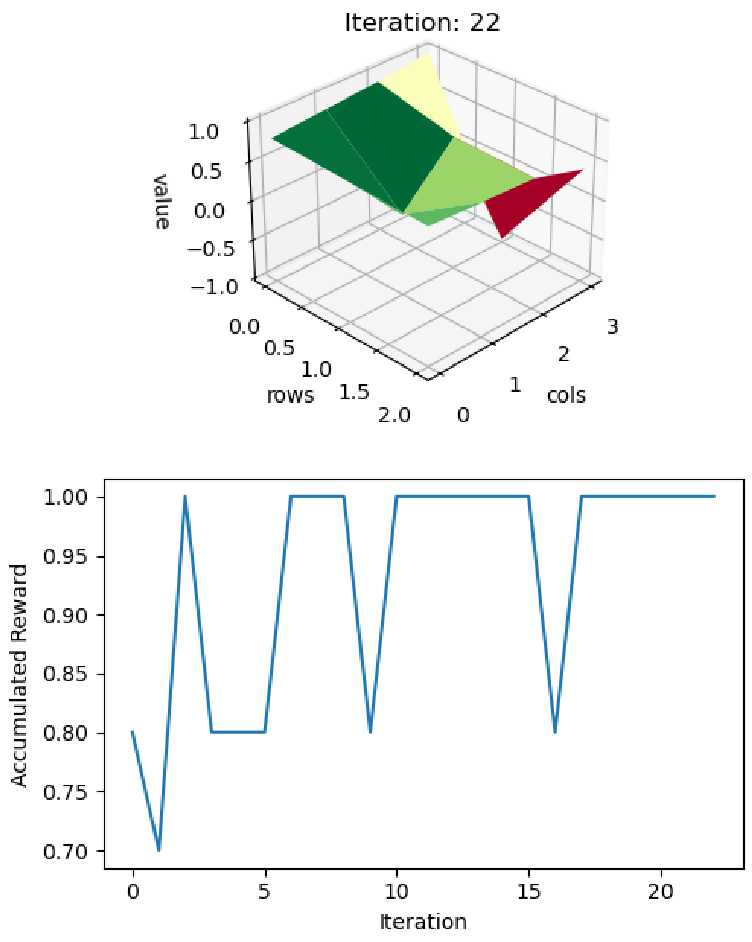


Figure 1: Value Iteration (exact DP). The top panel shows the 3D state-value surface after iterative Bellman optimality updates; the bottom panel shows the accumulated reward vs. iterations. Convergence occurs in a small number of sweeps, producing a stable value gradient toward the goal and the corresponding optimal policy (no exploration parameter applies to exact methods).

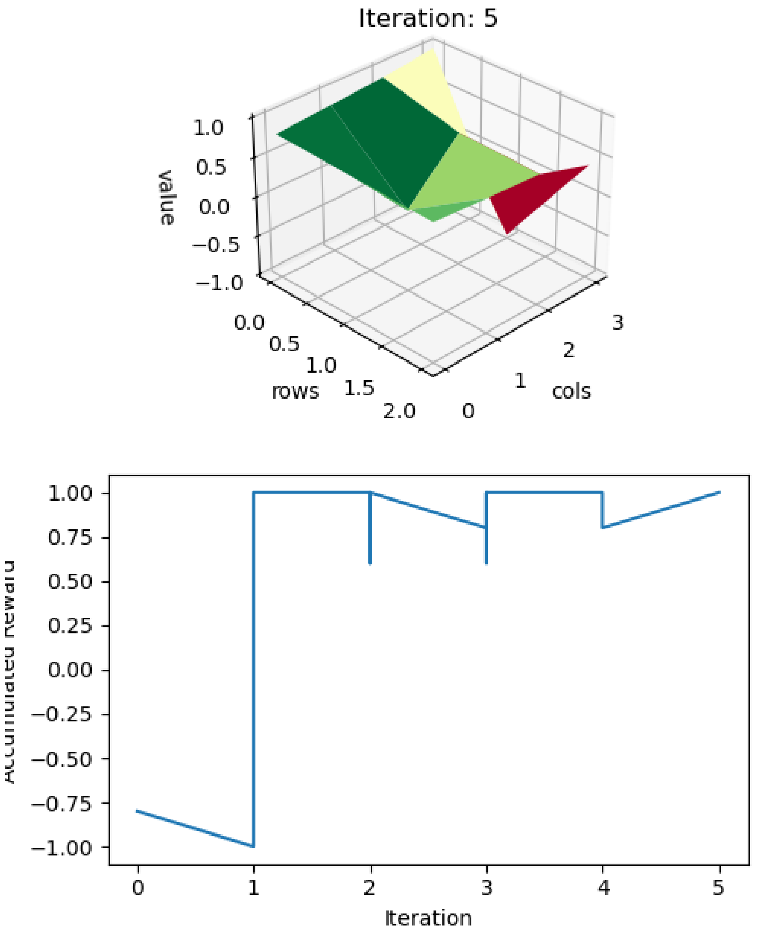
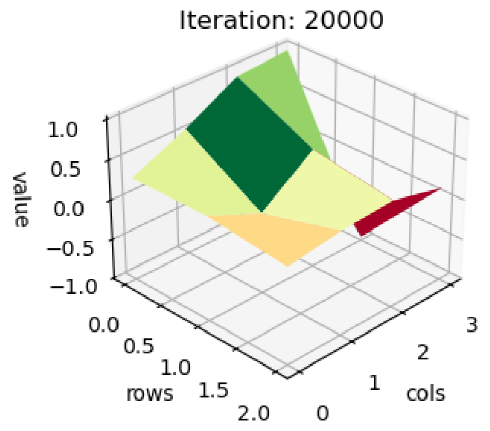


Figure 2: Policy Iteration (exact DP). Alternates full policy evaluation with policy improvement. The value surface stabilizes and the accumulated reward rises to near-optimal levels within only a few global iterations, yielding the optimal policy. (No exploration parameter.)

1.2 RL Methods (Learning Curves + Final Policies)



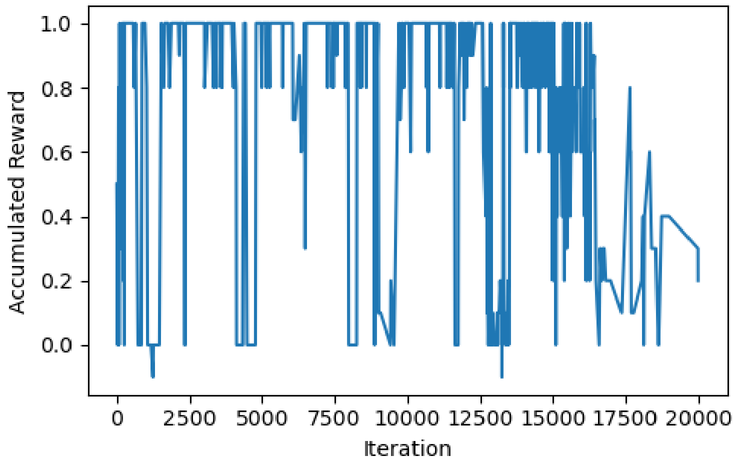
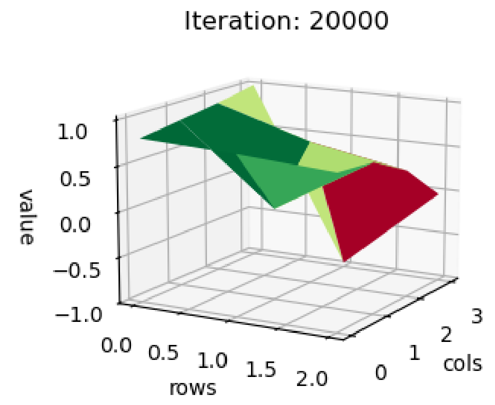


Figure 3: On-Policy First-Visit MC Control. Epsilon strategy: linear decay $\varepsilon : 1.0 \rightarrow 0.05$ over 20,000 steps (GLIE). Updates occur after complete episodes, so the learning curve has higher variance early on, then stabilizes as exploration decays. Final policy closely follows the optimal path.



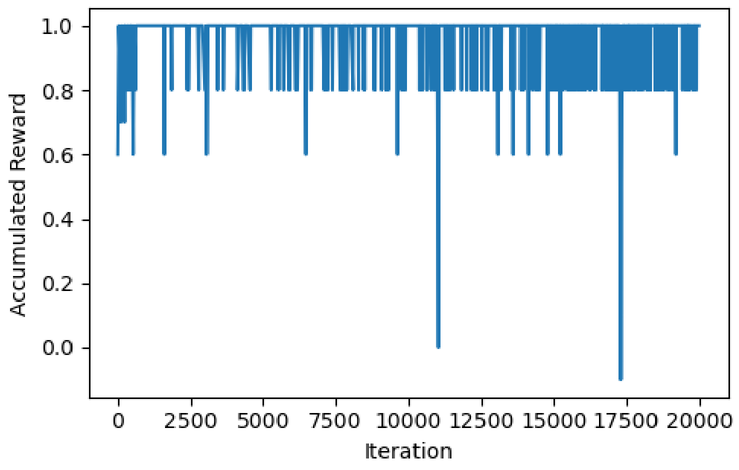
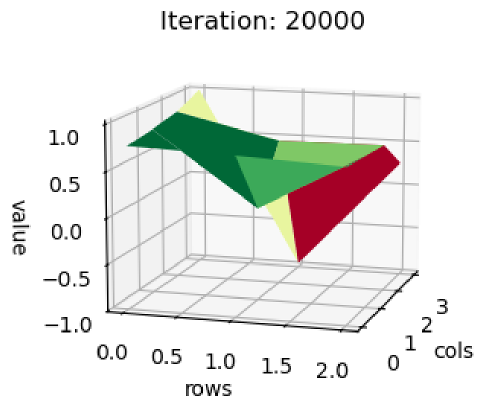


Figure 4: SARSA (on-policy TD). Epsilon strategy: linear decay $\varepsilon : 1.0 \rightarrow 0.05$ over 20,000 steps; $\alpha = 0.1$, $\gamma = 0.9$. The curve exhibits moderate noise during early exploration but steadily approaches an accumulated reward near 1.0. On-policy updates yield a slightly more conservative policy, which is nonetheless optimal in this grid.



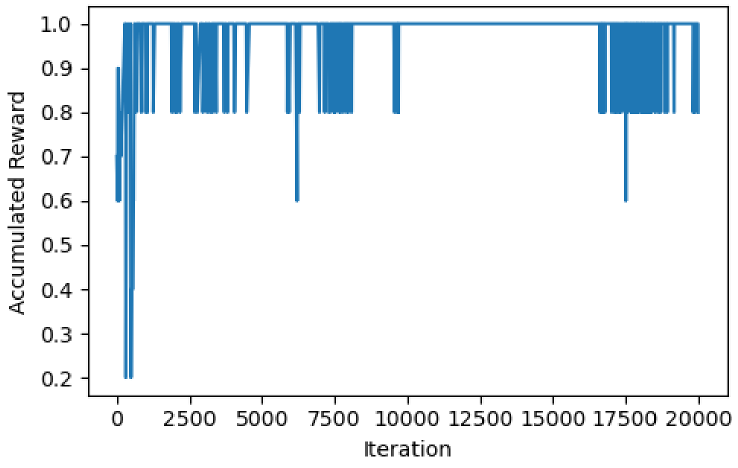
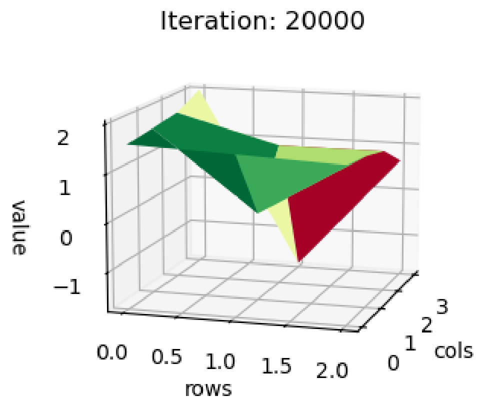


Figure 5: Q-Learning (off-policy TD). Epsilon strategy: linear decay $\varepsilon: 1.0 \to 0.05$ over 20,000 steps; $\alpha = 0.1$, $\gamma = 0.9$. Off-policy greedy targets produce faster convergence to high reward with slightly higher variance during exploration. Final policy matches the optimal policy.



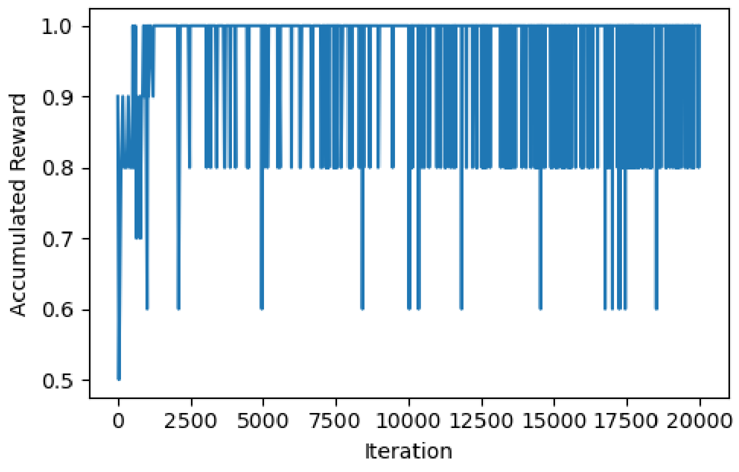


Figure 6: Double Q-Learning (off-policy TD with two Q tables). Epsilon strategy: linear decay $\varepsilon : 1.0 \to 0.05$ over 20,000 steps; $\alpha = 0.1$, $\gamma = 0.9$. Alternating updates between Q_1 and Q_2 mitigate maximization bias, yielding a smoother, more stable learning curve and an optimal final policy.

1.3 Value vs. Policy Iteration: Computational Efficiency

Observation. Policy Iteration reached a stable policy in markedly fewer global iterations than Value Iteration because each PI loop performs a complete policy evaluation followed by a greedy improvement, whereas VI applies partial Bellman optimality updates. Although each PI iteration is computationally heavier (full evaluation sweep), in small discrete MDPs like *world1* the total wall-clock time to convergence is typically lower for PI. Value Iteration, while simpler and cheaper per iteration, required more iterations overall. Empirically, both methods converged rapidly here; PI was slightly faster to a fixed policy.

1.4 Exact vs. RL Policies: Agreement and Differences

The final policies produced by the exact methods (VI, PI) and the learned methods (MC, SARSA, Q-Learning, Double Q-Learning) are in agreement on *world1*: all converge to the same goal-directed path under the same reward/transition model. Minor transient differences arise during training: MC exhibits higher variance because updates are episodic; SARSA is on-policy and thus more conservative near risky transitions; standard Q-Learning can momentarily overestimate action values; Double Q-Learning reduces that bias and stabilizes training. After sufficient steps and with a GLIE ε schedule, the RL methods recover the optimal policy predicted by the exact dynamic-programming solutions.

2 Crawler Results

2.1 Learning Curves and Final Policies

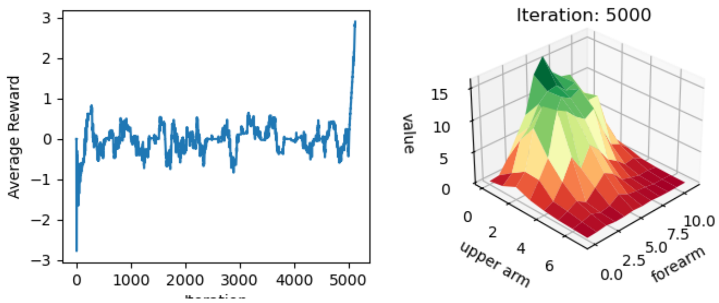


Figure 7: Crawler – Q-Learning. Left: average reward vs. training steps; Right: final value surface (proxy for the learned policy). Epsilon strategy: linear decay $\varepsilon : 1.0 \to 0.05$ over 5,000 steps (GLIE); TD params: $\alpha = 0.1$, $\gamma = 0.95$. The curve shows early variance but steady improvement, culminating in a strong final jump (peak reward ≈ 3) and a high, sharply peaked value surface (best posture concentrated at the optimal upper-arm/forearm configuration).

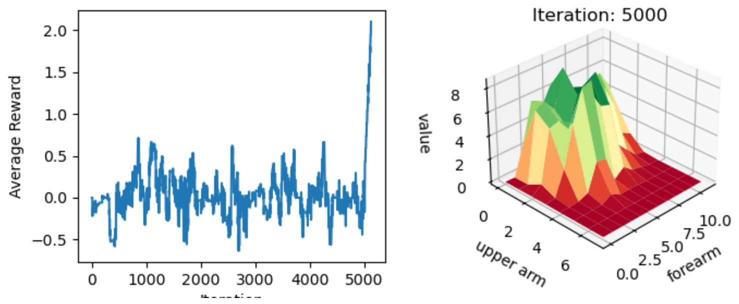


Figure 8: Crawler – SARSA. Left: average reward vs. training steps; Right: final value surface. Epsilon strategy: linear decay $\varepsilon : 1.0 \to 0.05$ over 5,000 steps; TD params: $\alpha = 0.1,\ \gamma = 0.95$. The learning curve exhibits comparable early variance with a smaller final jump (peak reward ≈ 2.1). The value surface is smoother and lower overall than Q-Learning, consistent with a slightly more conservative on-policy solution.

2.2 Comparison and Recommendation

Learning speed. Both methods improve quickly after the initial exploration phase, but **Q-Learning** shows a more decisive climb late in training and reaches higher rewards by step ∼5,000. This is expected: off-policy greedy targets in Q-Learning drive faster exploitation of promising actions, whereas SARSA updates on the executed (exploratory) action and thus learns more conservatively.

Final average reward. From Figures 7–8, Q-Learning attains the higher final reward (peak ≈ 3) versus SARSA (peak ≈ 2.1), and its final value surface is markedly taller and sharper, indicating a stronger preference for the optimal actuation region. Overall, **Q-Learning performs better** in this environment in terms of asymptotic return.

Which algorithm is preferable, and why? In simulation, Q-Learning is preferable for its higher final reward and faster convergence, albeit with slightly higher variance due to off-policy maximization. In the real world (physical crawler), safety and robustness under exploration matter: **SARSA** is often the safer default because its on-policy updates account for exploratory actions, reducing overestimation near risky transitions. A practical compromise is to deploy Double Q-Learning or conservative Q-Learning (lower α, slower ε decay, or action-noise schedules) to curb maximization bias while retaining efficiency.

Takeaway. **Q-Learning** wins on sample efficiency and peak performance in this task; **SARSA** offers steadier, more risk-aware learning. If training on hardware, start with SARSA (or Double Q) and progressively tighten exploration;

in sim-to-real pipelines, pretrain with Q-Learning in simulation, then fine-tune on device with SARSA-style schedules for safety.