Problems 1 & 2 — Reinforcement Learning Assignment

Problem 1 — Pick-and-Place Robot as an MDP

Task. Learn a policy that controls a robot arm to pick an object from a table and place it at a target quickly and smoothly.

MDP tuple (S, A, P, R, gamma). Episode starts with the object on the table and the arm at a home pose; ends on successful place, failure (collision/drop), or step limit. Discount gamma = 0.99.

1) State S (1 pt). Minimal info for fast, smooth low-level control and task geometry:

- Joint angles/velocities: q1..qn, dq1..dqn

- Gripper opening g (0..1)

- End-effector pose: p_ee, orientation (quaternion)

- Estimated object pose: p_obj, orientation

- Goal (place) pose: p_goal, orientation

- Optional phase flag: {approach, grasp, lift, move, place}

2) Action A (1 pt). Continuous control for smoothness:

- Joint torques tau1..taun (bounded by hardware)

- Gripper command (binary or continuous)

Note: Velocities would also work; torques give more direct smoothness control.

3) Transition P (1 pt). Determined by robot physics + perception updates (contacts, friction, latency). In simulation: physics engine; on the real robot: true plant. Use domain randomization (mass/friction/latency) for robustness.

4) Reward R — speed + smoothness + safety (3 pts). Let d_pick(t) = distance EE→grasp (pre-grasp); d_place(t) = distance object→goal (post-grasp); tau_t = joint torques; delta_tau_t = tau_t − tau_{t−1} (jerk proxy).

- Before grasp: -alpha1 * d_pick(t)

- After grasp: -alpha2 * d_place(t)

- Time cost: -lambda each step (finish sooner)

- Effort: -beta1 * ||tau_t||^2

- Jerk: -beta2 * ||delta_tau_t||^2

Terminal rewards/penalties:

- Success (goal tolerance, gripper open, object stable): +R_succ

- Collision: -R_coll

- Dropped object: -R_drop

- Timeout: -R_time

Example starting values (tune later): R_succ=100, R_coll=50, R_drop=30, R_time=20, alpha1=2.0, alpha2=2.0, lambda=0.1, beta1=0.001, beta2=0.0005. Reasoning: shaping pulls toward grasp/goal; time cost rewards speed; effort/jerk terms enforce low-jerk, smooth, safe motions.

5) Termination (1 pt). Success: object at goal within position/orientation thresholds, gripper released, object static a few steps. Failure: collision, slip/drop, or step limit T.

6) Initial state distribution (1 pt). Randomize object pose in a tray region, small home-pose jitter, and light sensor noise → better generalization.

7) Assumptions & notes (1 pt). Calibrated kinematics; reachable grasp; perception provides pose estimates. Safety layer on robot (action clipping, joint limits, collision monitor). Train with curriculum: approach → grasp → lift → move → place.

Summary. State = robot + task geometry; Actions = continuous torques + gripper; Reward = accuracy + speed + smoothness with clear terminal signals → fast, safe, low-jerk pick-and-place.

Problem 2 — 2×2 Gridworld (Value Iteration, 2 sweeps)

States S = {s1, s2, s3, s4} laid out as: [s1 s2; s3 s4]. Actions: up, down, left, right. Transitions: deterministic; if a move hits a wall, you stay in place. Rewards: R(s1)=5, R(s2)=10, R(s3)=1, R(s4)=2. Discount gamma = 0.9. Update: V_{k+1}(s) = R(s) + gamma * max_a V_k(s'), where s' is the next state.

Adjacency (next state per action):

State	Next state per action
From s1	up->s1, left->s1, right->s2, down->s3
From s2	up->s2, right->s2, left->s1, down->s4
From s3	down->s3, left->s3, up->s1, right->s4
From s4	down->s4, right->s4, up->s2, left->s3

Iteration 1

Initial values: V0(s1)=0, V0(s2)=0, V0(s3)=0, V0(s4)=0

Update to V1 (all neighbors are 0, so max is 0):

State	V1 update
s1	5 + 0.9*0 = 5
s2	10 + 0.9*0 = 10
s3	1 + 0.9*0 = 1
s4	2 + 0.9*0 = 2

Greedy actions w.r.t. V1 (optional): s1→right; s2→up or right; s3→up; s4→up.

Iteration 2

Use neighbors' V1 values to compute V2:

State	Computation
s1	best among {s1=5, s2=10, s3=1} is 10 => V2(s1) = 5 + 0.9*10 = 14
s2	best among {s2=10, s1=5, s4=2} is 10 => V2(s2) = 10 + 0.9*10 = 19
s3	best among {s3=1, s1=5, s4=2} is 5 => V2(s3) = 1 + 0.9*5 = 5.5
s4	best among {s4=2, s2=10, s3=1} is 10 => V2(s4) = 2 + 0.9*10 = 11

Final after two iterations: V2(s1)=14, V2(s2)=19, V2(s3)=5.5, V2(s4)=11.

Greedy policy w.r.t. V2: s1→right; s2→up or right; s3→up; s4→up.