Problem Set 7: Manipulator Control and Learned Models

6.4210/2 Robotic Manipulation — Fall 2026

Due: November 8, 2026

Out: October 29 **Due:** November 8

Enter the answers to each **Gradescope** block on Gradescope. The **Going further** prompts require no submission. All logarithms are natural. PS7 is worth 52 points for 6.4210 and 57 points for 6.4212; the extra 5 are Part 6.6.

Reading: Manipulator Control, and the model-based part of the Reinforcement Learning notes.

1 Bookwork

Problem 1 Planning with cost-weighted samples [5]

Model predictive path integral control (MPPI) is covered in the model-based part of the Reinforcement Learning notes. Here we use the simple cost-weighted update below, without an importance-sampling correction. A robot starts at $x_0 = 0$ and plans two commands toward a goal at $x = 2$, using the model $\widehat{x}_{t+1} = \widehat{x}_t + u_t$ with $\widehat{x}_0 = x_0$. The predicted cost of $U = (u_0, u_1)$ is

$$S(U) = (\widehat{x}_2 - 2)^2 + \tfrac{1}{2}(u_0^2 + u_1^2).$$

Starting from a nominal sequence $\bar{u}$, consider three feasible candidates:

$$U^{(1)} = (0, 0), \qquad U^{(2)} = (1, 1), \qquad U^{(3)} = (2, 0).$$

Score them using the model, with temperature $\lambda > 0$:

$$w_k = \frac{\exp[-(S_k - S_{\min})/\lambda]}{\sum_j \exp[-(S_j - S_{\min})/\lambda]}, \qquad \bar{u}^{\text{new}} = \bar{u} + \sum_k w_k (U^{(k)} - \bar{u}) = \sum_k w_k U^{(k)},$$

where $S_k = S(U^{(k)})$ and $S_{\min} = \min_k S_k$. Because the weights sum to one, $\bar{u}$ drops out here, and the update is the weighted average of the candidates. Part 6.4 uses the first form.

Gradescope:

1.1 Use the model to compute each candidate's cost: $S_1 =$ **4**, $S_2 =$ **1**, $S_3 =$ **2**. [1]

Solution: *Goal-error costs are $(4, 0, 0)$ and command costs are $(0, 1, 2)$.*

1.2 Let $\lambda = 1$. Compute the normalized weights and updated sequence. Round answers to three decimal places, using unrounded weights to compute the sequence: [2]

$$(w_1, w_2, w_3) = (\textbf{0.035}, \textbf{0.705}, \textbf{0.259}), \qquad \bar{u}^{\text{new}} = (\textbf{1.224}, \textbf{0.705}).$$

Solution: *Normalize $(e^{-3}, 1, e^{-1})$ and average the sequences to obtain $(w_2 + 2w_3, w_2)$.*

1.3 Keep these same candidates and costs. As $\lambda \to 0^+$, the weight of candidate 2 tends to **1**. [1]
As $\lambda \to \infty$, each candidate's weight tends to **0.333** (three decimals).

Solution: *Low temperature favors the cheapest sample; high temperature approaches equal weights.*

1.4 The robot executes $\bar{u}_0^{\text{new}}$, but observes $x_1 = 1.0$ instead of the predicted position $\widehat{x}_1 = \bar{u}_0^{\text{new}}$. What does it do next? Select one. [1]

A. Execute the old second command $\bar{u}_1^{\text{new}}$ without planning again.

B. Replan from the observed $x_1 = 1.0$ and execute only the first action of the new plan.

C. Continue planning from $\widehat{x}_1$, since that is the model's prediction.

D. Execute all three candidate sequences to measure their costs on the robot.

Solution: *The second command helps choose the first and is replaced when the robot replans. Replanning from the observation corrects the starting state, although any error in the transition model remains.*

Problem 2 Force and position control [6]

We model the robot as a point mass moving along a line, as in the "Force and Position Control" exercise in Manipulator Control. Its actuator applies the force u, and the environment applies the contact force f_c:

$$m\ddot{x} = u + f_c, \qquad u = k_p(x_d - x) - k_d\dot{x} - f_d,$$

where $m, k_p, k_d > 0$. Both f_c and its desired value f_d are forces on the robot, positive in the $+x$ direction. With $f_d > 0$, the term $-f_d$ makes the robot push toward $-x$, and a surface on that side pushes back with $f_c > 0$. The desired position x_d and force f_d are constant. Zero error means both $x = x_d$ and $f_c = f_d$. Steady state means $\dot{x} = \ddot{x} = 0$.

Gradescope:

2.1 At steady state, the controller is $u = k_p(x_d - x) - f_d$. Which description is correct? Select one. [1]

A. Feedback for velocity and acceleration.
B. Feedforward for position and velocity.
C. Position feedback plus feedforward for the desired contact force.
D. Acceleration feedforward plus contact-force feedback.

Solution: *The position term depends on measured error. The force term uses the command f_d, without measuring contact-force error.*

2.2 In free space, $f_c = 0$. Which equation describes steady state? Select one. [1]

A. $0 = m(x_d - x) + f_d$.
B. $0 = k_p(x_d - x) - f_d$.
C. $0 = k_p(x_d - x) - f_c$.
D. $0 = k_p\dot{x} + f_d$.

Solution: *Substitute $f_c = 0$ and $\dot{x} = \ddot{x} = 0$ into the closed-loop dynamics.*

2.3 In free space, let $x_d = 0$ and $f_d \neq 0$. Which statement is correct? Select one. [1]

A. Both position and force errors can be zero.
B. A position error allows the actual contact force to equal f_d.
C. Position and force errors cannot both be zero.
D. The system must be unstable.

Solution: *The equilibrium is $x = -f_d/k_p$, while $f_c = 0$. Neither desired quantity is achieved when $f_d \neq 0$.*

2.4 What physical fact prevents a nonzero contact force in free space? Select one. [1]

A. The damping term cancels the position term at rest.
B. The proportional gain becomes zero.
C. There is no surface to provide a reaction force.
D. Steady state requires nonzero acceleration.

Solution: *The actuator can move the robot, but there is no surface to supply a contact force, so $f_c = 0$.*

2.5 A rigid wall now constrains the robot to $x \geq 0$. The robot is in contact at $x = x_d = 0$, and $f_d > 0$. Which statement is correct? Select one. [1]

A. The desired position cannot be maintained.
B. Zero position error and zero force error are possible at steady state.
C. The force must overshoot its desired value.
D. Contact makes the controller unstable for every positive gain.

Solution: *At contact the controller applies $u = -f_d$ toward the wall, and the wall can supply $f_c = f_d$ in the positive direction.*

2.6 Why is zero error possible at this wall? Select one. [1]

A. The wall supplies a reaction force $f_c = f_d$ balancing $u = -f_d$.

B. The wall automatically increases the damping gain.

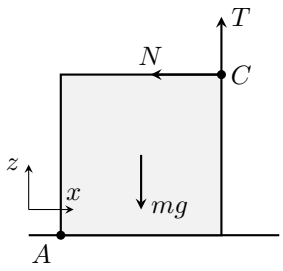
C. Contact increases the proportional gain.

D. The feedforward term cancels gravity in this horizontal model.

Solution: *The wall constrains position while its reaction balances the commanded force.*

Problem 3 Tipping a box without sliding its pivot [4]

A square box of side length s and mass m is initially upright. Its lower left corner A is the ground pivot. A finger contacts its upper right corner C, applying a horizontal force of magnitude N toward the left and an upward tangential force T. Gravity has magnitude mg and acts at the center. Take $N > 0$, $T \geq 0$. The coefficients of static friction are μ_A at the ground and μ_C at the finger. The finger must satisfy $T \leq \mu_C N$. The model uses the initial, axis-aligned approximation of the "Box Flip-up" exercise in Manipulator Control.



Use the tipping-moment test $s(N + T) - smg/2 > 0$. Approximate the ground reaction by horizontal and vertical force balance: $R_x = N$, $R_z = mg - T$, with $N \leq \mu_A R_z$ and $R_z \geq 0$. We only test whether some admissible forces can initiate tipping from the upright pose.

Gradescope:

3.1 The finger friction limit and positive tipping moment imply [1]

$$N > \frac{mg}{a(\mu_C + b)}.$$

Give the integers $a =$ **2** and $b =$ **1**.

Solution: *$mg/2 < N + T \leq (1 + \mu_C)N$, so $N > mg/[2(1 + \mu_C)]$.*

3.2 For $0 < \mu_A < 1$, combine the tipping condition with the ground friction constraint to obtain the necessary upper bound [1]

$$N < \frac{\mu_A mg}{a(b - \mu_A)}.$$

Give $a =$ **2** and $b =$ **1**.

Solution: *Tipping requires $T > mg/2 - N$. Thus $N \leq \mu_A(mg - T) < \mu_A(mg/2 + N)$. Rearranging gives the bound.*

3.3 Let $\mu_A = 0.25$. Some N satisfies both bounds exactly when $\mu_C > k$. Give the integer $k =$ **2**. [1]

Solution: *The lower bound is below the upper bound when $1 - \mu_A < \mu_A(1 + \mu_C)$, that is, $\mu_C > 1/\mu_A - 2 = 2$. Equality is the limiting balance case; the strict tipping condition requires $\mu_C > 2$.*

3.4 Now let $\mu_A = 1$. Which statement is correct under this idealized force model? Select one. [1]

A. Tipping is impossible because the pivot has too much friction.

B. For any $\mu_C \geq 0$, some finite forces can tip the box without sliding the pivot.

C. The finger must apply infinite normal force.

D. Feasibility requires $\mu_C = 2$ exactly.

Solution: *Choose $T = 0$ and $mg/2 < N < mg$. The moment is positive and the ground can resist N because its normal reaction is mg and $\mu_A = 1$. The upper bound derived for $\mu_A < 1$ divides by $1 - \mu_A$, so it does not apply here. The conclusion holds for the initial force model given above.*

Problem 4 Control: choosing feedback and feedforward [4]

For each part, select all statements that are true. Assume ideal, continuous-time models, exact state measurements, and no actuator saturation or delay. A fixed setpoint is constant; a constant-velocity reference has the form $q_d(t) = q_0 + vt$ with $v \neq 0$.

Gradescope:

4.1 A velocity-controlled joint has plant $\dot{q} = u$. [1]

√ **A P controller $u = K_p(q_d - q)$ with $K_p > 0$ drives the error to zero for a fixed setpoint.**

◯ A P controller alone gives zero steady-state error when following a nonzero constant-velocity reference.

√ **Adding integral feedback with suitable gains can eliminate steady-state error for a constant-velocity reference.**

√ **Adding $\dot{q}_d$ as feedforward can eliminate the tracking lag in this ideal model.**

Solution: *With $e = q_d - q$, P control gives $\dot{e} = v - K_p e$, so its ramp-tracking error tends to v/K_p. Integral feedback supplies the missing steady input; feedforward $u = \dot{q}_d + K_p e$ gives $\dot{e} = -K_p e$.*

4.2 A torque-controlled joint lifts a mass against a constant gravity load: $I\ddot{q} = u - \tau_g$, where $I > 0$ and $\tau_g > 0$. There is no separate gravity-feedforward term. Consider fixed-setpoint control. [1]

◯ P control with a finite positive gain guarantees zero steady-state error.

◯ PD control with suitable gains guarantees zero steady-state error.

√ **PID control with suitable stabilizing gains can give zero steady-state error.**

◯ Every choice of positive PID gains is stable.

Solution: *At rest, P and PD require $K_p e = \tau_g$. An integral term can supply the load at zero error. Positive gains alone are insufficient: the characteristic polynomial $Is^3 + K_d s^2 + K_p s + K_i$ also requires $K_d K_p > I K_i$ for stability.*

4.3 Consider direct force control and gravity-compensated stiffness control as described in *Manipulator Control*. [1]

- √ **Direct force control can help with nonprehensile pivoting and close-fitting insertion tasks.**
- √ **Force commands for contact tasks must account for friction constraints.**
- ◯ Direct force control always guarantees that the robot safely comes to rest.
- ◯ The gravity-compensated stiffness controller requires an integral term to cancel gravity.
- √ **The stiffness controller uses a feedforward torque to cancel the modeled gravity load.**
- ◯ Stiffness must always be made large enough to eliminate all position error.
- √ **Lower stiffness can help a robot accommodate an imperfectly known constraint, such as a door's rotation axis.**

Solution: *The correct set is 1, 2, 5, 7. Force control in free space can cause continued acceleration. Stiffness control uses compliance and modeled gravity compensation rather than maximum stiffness.*

4.4 Let $v = J\dot{q}$ map joint velocity to end-effector spatial velocity, and let f be the matching spatial wrench at the same point and in the same frame. Consider the generalized force corresponding to f. [1]

- ◯ The mapping is $\tau = Jf$.
- ◯ The mapping is $\tau = J^{-1}f$.
- √ **The mapping is $\tau = J^T f$.**
- ◯ Operational-space control consists of converting a task trajectory to joint coordinates and applying joint PID control.
- √ **Operational-space inverse-dynamics control expresses the robot dynamics in end-effector coordinates to control task-space motion.**

Solution: *Power consistency gives $f^T v = f^T J\dot{q} = (J^T f)^T \dot{q}$. Operational-space control reasons about task-space dynamics rather than only tracking an inverse-kinematics joint trajectory.*

2 Computational questions

Problem 5 Dragging a book with hybrid force-position control [8]

Friction analysis. A gripper presses down on a book of mass m resting on a horizontal table. Let $F > 0$ be the downward force applied by the gripper. The coefficients of static friction are μ_C at the gripper–book contact and $\mu_A > 0$ at the table–book contact. Assume no vertical acceleration or tipping. We want the book to start sliding on the table while sticking to the gripper. Use the Coulomb model at both contacts. The force analysis follows the "Hybrid Force-Position Control" exercise in Manipulator Control.

Gradescope:

5.1 The condition for some horizontal gripper force to overcome table friction without slipping at the gripper can be written [1]

$$\frac{\mu_C}{\mu_A} > a + \frac{mg}{F}.$$

Give the integer $a =$ **1**.

Solution: *The table normal is $mg + F$. We need $\mu_C F > \mu_A(mg + F)$, which gives $a = 1$ after division by $\mu_A F > 0$.*

5.2 Does lowering μ_C make the motion more feasible? Select one. [1]

A. Yes: it reduces resistance at the table.

B. No: it reduces the tangential force the gripper can transmit without slipping.

C. Yes: it reduces the table's normal load.

D. No: μ_C does not enter the feasibility condition.

Solution: *The gripper needs friction to transmit the horizontal pulling force.*

5.3 Suppose $\mu_C > \mu_A$, and ignore force limits and damage. Does increasing F make the friction condition easier to satisfy? Select one. [1]

A. Yes: although both friction limits increase, the available gripper friction grows faster than the table resistance.

B. No: normal force does not affect horizontal forces.

C. No: increasing table resistance always prevents motion.

D. Yes, but only if μ_A simultaneously decreases.

Solution: *The friction margin is $(\mu_C - \mu_A)F - \mu_A mg$, which increases with F under the stated assumption. If $\mu_C \leq \mu_A$, no finite positive F satisfies the condition.*

Controller implementation (Problem 5, continued).

Notebook: `ps7/hybrid_force_position.ipynb`.

Complete `compute_ctrl` in `ps7/hybrid_force_control.py`. For a Python-only workflow, run `python ps7/hybrid_force_demo.py --verify --simulate` from the course root after completing the function. Install dependencies with `python -m pip install numpy drake manipulation`, or use the course environment. The supplied simulation uses the planar iiwa to drag a book as in that exercise. Use the friction analysis above to interpret the dragging motion.

The function receives the gripper pose components (θ_y, x, z), velocity components (ω_y, v_x, v_z), desired horizontal position x_d, and positive downward force magnitude F_d. Return the planar end-effector *wrench* $w = (\tau_y, f_x, f_z)$, with torque in N m and forces in N. The supplied wrapper maps the wrench to joint torques using $J^T w$, and the station handles gravity compensation.

Use PD feedback to regulate pitch to zero and horizontal position to x_d. Use gains $k_p^\theta = 10$, $k_d^\theta = 3$, $k_p^x = 50$, and $k_d^x = 15$. The desired pitch rate and the desired horizontal velocity in the feedback terms are zero. Command downward force magnitude F_d in world z; positive z points up. There is no position feedback in z.

Run the verification cell, then the supplied 6.5-second simulation with $x_d(t) = 0.65 - 0.125\max(t-2, 0)$ m and $F_d = 10$ N. Inspect the final book position and the minimum finger-tip height printed by the notebook. The book center should end between $x = 0.35$ m and $x = 0.55$ m, and the tip should remain at or above $z = 0.035$ m. A replay can be enabled in the notebook. Submit the friction answers and the fixed numerical verification below to Gradescope.

5.4 Use the fixed verification inputs [5]

$$(\theta_y, x, z) = (0.1, 0.8, 0.05), \qquad (\omega_y, v_x, v_z) = (0.2, -0.1, 0), \qquad x_d = 0.65, \quad F_d = 2.0.$$

Gradescope: Give the commanded wrench components to four decimal places:

$\tau_y =$ ___−1.6000___ N m, $f_x =$ ___−6.0000___ N, $f_z =$ ___−2.0000___ N.

Solution: *$\tau_y = -10(0.1) - 3(0.2) = -1.6$, $f_x = 50(0.65 - 0.8) - 15(-0.1) = -6$, and $f_z = -F_d = -2$.*

Problem 6 Learning what a push does, then planning [25]

Notebook: `ps7/model_based_push.ipynb`. This problem is worth 25 points for 6.4210 and 30 for 6.4212, whose extra Part 6.6 is worth 5. Complete the six functions in `ps7/model_based_math.py`. The supplied transition model and planner are in `model_based_push.py`. We compare two controllers built on learned models with the BC and scalar-residual PPO policies from the PS6 exercise *Improving a cautious policy with residual PPO*, on the same task: the observations, reward, action bounds, initial states, goals and 32-command cutoff are PS6's. Use three decimal places unless stated otherwise, and three significant figures for p-values. Numbers from your own runs (timings, the four-goal table) depend on your machine, so they are not checked against fixed values.

The *heuristic* pushes the block along whichever of its two axes has the larger remaining offset to the goal, asking for that whole offset in one push. It converts a desired displacement e into a command using the nominal relation $\widehat{y} = \ell - 0.02$. The *calibrated heuristic* keeps the same face choice but uses the line fitted in Part 6.1. The neural transition model lets MPPI choose both faces and distances. Both models are fit once and then kept fixed: MPPI replans from each new observation, but the model's weights do not change. Several parts use a supplied trained model, `data/cautious_model_based/dynamics.pt`, in place of your own fit, and everyone reports the same numbers for those parts. A separate set of ten fits, analyzed in the 6.4212-only Part 6.6, measures how much training randomness matters.

Training and planning costs. Fitting a model of how the world responds to commands is called *system identification*. Both models are fit from PS6's 886 demonstration transitions, with no new interaction with the simulator. The supplied neural fit took 1.16 seconds (400 epochs). Each PPO run from PS6 collected 8192–8202 new training commands and took 2.65–2.94 hours, including simulation, updates and monitoring (28.19 hours over ten runs). Both approaches start from the same 16.47 minutes of demonstration collection, and fitting the BC policy that PPO starts from took 0.62 seconds. MPPI pays for planning at run time instead: about 2.37 milliseconds per command in the supplied run. All times in this paragraph are wall-clock times on the machine that produced the reference.

What to run. Fit the two models, then run each controller on the first four PS6 test goals (seeds 9660000–9660003); together this takes about 3–5 minutes. The full 64-goal comparison is supplied, so there is no need to rerun it, and the planner settings stay as given. Once your functions are implemented, the notebook also prints the values for the worked checks in Parts 6.1–4.

6.1 **A fitted correction to the heuristic.** Fit the relation between commanded and realized displacement, then invert it to choose push lengths. [5]

For a push with command length ℓ_i and world unit direction u_i, let p_i, p_i' be the block's planar positions before and after execution. The measured along-push displacement is $y_i = u_i^T(p_i' - p_i)$. Complete `fit_displacement`: construct the design matrix and use a least-squares solver to fit $\widehat{y} = \alpha\ell + \beta$ on the 886 training transitions from PS6's demonstrations. Here α is the dimensionless fitted slope and β is the fitted intercept in meters. Complete `calibrated_length`, which inverts the fit and clips the command to the action bounds.

To see whether fitting improves prediction on new pushes, the notebook plots the fitted line and the nominal relation $\widehat{y} = \ell - 0.02$ against all 221 BC validation transitions, along with the residuals (measured minus predicted displacement). The demonstrations all use commands of 6.4–8.4 cm. A very short command may not move the block at all; the range of such commands is called a *dead zone*. The data do not cover that range, so the fitted line extrapolates through it.

Gradescope: For coefficient order (α, β), which design-matrix row corresponds to a command ℓ_i? Select one. **A.** $[\ell_i, 1]$. B. $[1, \ell_i]$. C. $[\ell_i, \ell_i^2]$.

Using the 886 supplied training transitions, fitted coefficients (five decimals): $\alpha =$ **0.96237**, $\beta =$ **-0.04210** m. On the 221 BC validation transitions, RMSE in millimeters (two decimals): nominal model **25.27**; fitted model **1.47**.

For a separate check, take $\alpha = 0.8$, $\beta = -0.01$ m. Fill in `calibrated_length`'s output after clipping:

Desired displacement e (m)	0.08	0.30	0.002
Command ℓ (m; four decimals)	**0.1125**	**0.1250**	**0.0150**

Can one affine relation exactly represent a flat dead zone followed by a nonzero-slope region? Select one. A. Yes. **B. No.**

Solution: *Solve* $\min_{\alpha,\beta} \|[\ell\ \mathbf{1}](\alpha,\beta)^T - y\|^2$ *using a least-squares solver. Command* $\mathrm{clip}((e-\beta)/\alpha, 0, 0.125)$*, here* 0.1125 *m. A nonpositive fitted slope is a failed model check. A single affine fit cannot reproduce a dead zone followed by motion. The fit is* $\widehat{y} = 0.96237\ell + (-0.04210)$ *m, with held-out RMSE 1.47 mm versus 25.27 mm for the nominal relation; exact reproduction of these rounded coefficients is not the learning objective.*

6.2 **Learning a transition model.** Fit a neural model of configuration changes and command duration for use in multi-step rollouts. [5]

The supplied data contain $(x, a, x', \Delta\tau, \text{event})$ for complete push commands: the physical state before a command, the command, the state afterward, its total duration, and any terminal or invalid-action event. These are PS6's 64 training and 16 validation BC episodes. The first eight validation episodes (119 transitions) choose the network checkpoint; the last eight (102 transitions) are held out for diagnostics.

The model predicts the next physical state and total execution duration. The physical state is the eight-entry vector $x = (p_x, p_y, \phi, f_x, f_y, p_z, \theta_{\text{roll}}, \theta_{\text{pitch}})$: block planar position and yaw, finger planar position, block height, and block roll and pitch. Positions are in world coordinates (meters); angles are in radians. Both bodies have settled before and after a command, so there are no velocities. Supplied code turns states and commands into network inputs and targets y_i (configuration changes and $\log \Delta\tau_i$) and back, handling coordinate transforms and angle wrapping. Invalid commands are handled separately. Durations cover the whole command: approach, push, retraction and settling.

Complete `dynamics_loss`: mean squared error between the network output and $(y_i - \mu)/s$, where division is coordinatewise and μ, s are the supplied per-coordinate training mean and standard deviation (with a minimum scale for each output). The network output is already in normalized coordinates. The notebook fits the network once and keeps the checkpoint with the lowest validation prediction loss. The splits keep whole episodes separate.

Inspect one-step position and duration RMSE and three-step open-loop position RMSE on the diagnostic trajectories. For the latter, start at a recorded state and predict three recorded commands in sequence, feeding each predicted state into the next step without correcting it from Drake. Compare the position errors with the 2.5 cm goal tolerance.

Gradescope: For a two-example, two-output check of `dynamics_loss`, take

$$f_\theta = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}, \quad y = \begin{bmatrix} 2 & 8 \\ 6 & 12 \end{bmatrix}, \quad \mu = (2, 4), \quad s = (2, 4).$$

The scalar mean squared loss over all four entries is **2.250**.

Report the following for the supplied model `data/cautious_model_based/dynamics.pt` on the `diagnostic` split, even after fitting your own. The notebook prints them separately.

One-step position RMSE (mm; two decimals)	**1.18**
One-step duration RMSE (s)	**0.019**
Three-step position RMSE (mm; two decimals)	**4.93**

Is the three-step position RMSE smaller or larger than the 25 mm goal tolerance? Select one. **A. Smaller.** B. Larger.

Select true or false for each statement.

1. Scaling each output separately changes the relative importance of its error in the loss. **A. True.** B. False.
2. A failed control episode can still supply valid state-transition targets. **A. True.** B. False.
3. Small one-step prediction error on the demonstration trajectories guarantees accurate predictions for every action a planner might try. A. True. **B. False.**

Solution: $L = (nq)^{-1} \sum_{i,j} (f_\theta(x_i, a_i)_j - (y_{ij} - \mu_j)/s_j)^2$ *for* q *outputs. Scales weight physical errors. Their statistics use training episodes only. Bad actions still teach their consequences, but a controller may omit states or commands that planning will try. Iterated prediction can compound errors and*

leave the data distribution. On the supplied diagnostic split, one-step position RMSE is 1.18 mm and three-step RMSE is 4.93 mm. These errors come from neither the checkpoint-selection split nor the states MPPI later visits. An average error below tolerance does not guarantee success on every goal.

6.3 **Planning for the same objective.** The supplied MPPI planner looks $H = 2$ pushes ahead, stopping a predicted sequence early only at success or physical failure, and keeps $H = 2$ even near the 32-command cutoff. It enumerates all $4^H = 16$ face sequences and optimizes their continuous distances, predicting each candidate rollout with the learned model. After scoring the optimized sequences, it executes the first command of the best sequence in Drake and replans from the observed state. [5]

Complete `rollout_cost`, which assigns one cost to each predicted sequence. For a predicted horizon ending at H (or earlier at a terminal event), let $\widehat{d}_H$ be the predicted distance to the unchanged goal. Use

$$C = \frac{\widehat{d}_H}{0.15} + 1 - \mathbf{1}\{\text{success}\} + 0.02 \sum_{t<H} \widehat{\Delta\tau}_t + 0.2\, n_{\text{invalid}} + \mathbf{1}\{\text{fall or escape}\}.$$

The wrapper supplies event and validity checks. An invalid command leaves the state unchanged and takes no simulated time. Predicted terminal events are absorbing: the predicted episode stops, with no further motion or cost. Recall PS6's reward: a valid command earns

$$r_t = \frac{d_t - d_{t+1}}{0.15} + \mathbf{1}\{\text{success}\} - 0.02\Delta\tau_t - \mathbf{1}\{\text{fall or escape}\},$$

and an invalid command earns -0.2 without moving anything.

Gradescope: Consider a predicted sequence with initial distance $d_0 = 0.30$ m, final distance $\widehat{d}_H = 0.06$ m, total duration 10 s, and one invalid command. It neither succeeds nor falls/escapes. Its cost is $C =$ **1.800**. Its predicted undiscounted return, the sum of PS6's per-command rewards over the predicted sequence, is $\widehat{G} =$ **1.200**.

Which identity holds at a fixed initial state? Select one.

A. $\widehat{G} = C$.

B. $\widehat{G} = d_0/0.15 + 1 - C$.

C. $\widehat{G} = d_0/0.15 - 1 + C$.

Why does the transition model include finger position? Select one.

A. Success is defined by finger position alone.

B. Finger position specifies the block's mass.

C. The same block pose can require different approach motions and execution times from different finger positions.

Solution: *Progress telescopes, so $\widehat{G} = d_0/0.15 + 1 - C$. Count the success bonus once and charge no further time after a terminal event. At a fixed block pose, different finger positions can have different approach costs. This short-horizon objective is not the optimal full-episode value function.*

6.4 **Cost-weighted MPPI updates.** Use the cost-weighted update in Problem 1, applied to continuous latent distances; discrete faces are enumerated separately. For a fixed face sequence, the planner perturbs a nominal latent-distance sequence $\bar{z}$ (the current guess before the sigmoid transform) by Gaussian noise $\epsilon^{(k)}$, then maps each coordinate to $\ell_t = 0.125\,\text{sigmoid}(z_t)$. Complete `cost_weights` and `update_nominal`, replacing $\bar{u}$ by $\bar{z}$ and $U^{(k)} - \bar{u}$ by $\epsilon^{(k)}$ in that update. The supplied planner handles model rollouts, initial guesses and batching, using 24 candidates per face sequence, three updates and a horizon of two. Function docstrings specify array shapes. Keep the supplied settings; no temperature or horizon sweep is required. [5]

Gradescope: For one face sequence, let $S = (0, \ln 3)$, $\lambda = 1$, and $\bar{z} = (0, 0)$. The two sampled perturbations are $\epsilon^{(1)} = (0.2, -0.4)$ and $\epsilon^{(2)} = (-0.6, 0.8)$.

- Weights: $w_1 =$ **0.750**, $w_2 =$ **0.250**.
- Updated nominal sequence: (**0.000**, **-0.100**).

Which statements are correct? Select all that apply.

- √ **Subtracting $S_{\min}$ preserves the normalized weights and keeps the largest exponential equal to 1, so the normalizing sum cannot underflow to zero.**
- √ **As $\lambda \to 0^+$, a unique lowest-cost candidate gets weight tending to one.**
- √ **As $\lambda \to \infty$, the weights become uniform for fixed finite costs.**
- ◯ Averaging numerical face indices gives a physically meaningful intermediate face.
- √ **Replanning uses the newly observed physical state while keeping the learned model's weights fixed.**

Solution: *The common subtraction preserves normalized weights and makes the cheapest candidate's exponential exactly 1, so the normalizing sum cannot underflow; a very small weight can still round to zero, which is harmless. Small temperature favors low scores; large temperature approaches uniform weights. Faces have no meaningful numerical average. Replanning corrects the predicted state using observations, but does not repair a systematically wrong model. The preset uses uncorrected rollout costs. The continuous path-integral theory gives no optimality guarantee for this hybrid adaptation.*

6.5 **Prediction, control, and cost.** Compare prediction accuracy, control performance, and computation, using the same BC and ten final PPO policies as PS6. [5]

Run the calibrated heuristic and learned-model MPPI on the four goals above; the notebook fixes the planner's random seed for each goal. Look at one replay and at the notebook's table for the two controllers: success, reward, total simulated time, pushes and finger travel, along with your fitting and planning wall times. Four goals are enough to see how each controller behaves but too few for statistical conclusions.

Then analyze the supplied 64-goal comparison on goals 9660000–9660063, which uses the same commands, executor, objective and cutoff for every controller, and check one prediction against what actually happened.

Going further: Using the supplied costs, which method needs new training interaction, and which searches during deployment? Why does comparing their wall times not compare optimizer speed?

Gradescope:

The 64-goal results below use the supplied dynamics model. The notebook's `paired_goal_interval` (5000 resamples, seed 701) resamples goals and averages the ten PPO policies within each goal, so these intervals describe uncertainty over goals for these particular trained controllers.

Reward difference	Mean	95% lower	95% upper
MPPI minus BC	**0.838**	**0.796**	**0.878**
MPPI minus calibrated heuristic	**0.003**	**-0.018**	**0.022**
MPPI minus scalar-residual PPO	**0.359**	**0.333**	**0.383**

Mean reward on these shared 64 goals: BC **1.821**; all-ten scalar-residual PPO mean **2.300**; MPPI **2.659**. Does the interval provide evidence of higher mean reward than BC for these fitted controllers on the goal distribution? Select one. **A. Yes.** B. No. Does it provide evidence of higher mean reward than the calibrated heuristic? Select one. A. Yes. **B. No.** All four methods succeed on all 64 goals, so the differences are in reward and execution time.

Use MPPI minus calibrated heuristic for both signed differences below: mean pushes **0.266**; mean total execution time (s) **-1.220**.

Which explanation is consistent with the supplied traces? Select one.

A. Every push has the same total execution time.

B. Total time depends on approach, push, retract and settling; push count alone does not determine execution cost.

C. Fewer pushes always means less total execution time.

For the first transition in the supplied `data/cautious_model_based/evaluation/mppi.json` (goal 9660000, push 1), use the supplied `dynamics.pt`. Position prediction error $\|\widehat{p}' - p'\|$ (mm; two decimals): **21.20**. Signed duration error, predicted minus actual (s): **-0.556**.

Select true or false for each statement.

1. The 64-goal evaluation represents 64 independently trained dynamics models. A. True. **B. False.**
2. Because PPO collected extra online experience, this comparison does not by itself establish whether MPPI or PPO is the better method when both have the same amount of data. **A. True.** B. False.
3. More search with an inaccurate model can favor commands whose predicted benefits do not occur in Drake. **A. True.** B. False.

Solution: *MPPI minus BC is 0.838 [0.796, 0.878], a substantial reward improvement. Mean simulated execution time falls from 86.88 to 45.89 seconds (47.18%). The paired reward differences are 0.003 [-0.018, 0.022] over calibration and 0.359 [0.333, 0.383] over the all-ten scalar-residual PPO mean. These intervals condition on one model/data realization and the ten fitted PPO policies, and leave out the variability from refitting the model. Both methods start from the same demonstration archive, but PPO's training stage adds new online interaction, so the interaction budgets are unequal. A fair comparison of the two methods would give both the same data, for example by restricting PPO to the demonstrations (which leaves roughly BC) or by also fitting the model to PPO's online experience. PPO's reported training time also includes monitoring, while the model fit reuses demonstrations and MPPI shifts computation to deployment. Their wall-time difference therefore does not measure optimizer speed alone. Search can exploit model errors. MPPI makes more pushes than the calibrated heuristic yet takes less total time, so push count alone does not set execution cost.*

6.6 **Replication over dynamics fits.** 6.4212 only The comparison in Part 6.5 holds one fitted dynamics model fixed. This part asks how much the result depends on that one training run. The archive `data/cautious_model_based/replication` holds ten more fits (seeds 300–309), trained exactly like the supplied model (same data, 400 epochs, checkpoint rule, architecture and optimizer) but from different random initializations and minibatch orders. Each one drives MPPI on the same 64 goals with the same planner seeds. Fitting all ten models took 6.02 seconds, and most of the replication's 13.11 minutes went to running the 640 evaluation episodes. Nothing here needs to be rerun. [5]

Gradescope: Using the notebook's whole-fit analysis (5000 bootstrap resamples, seed 816), report MPPI-minus-BC mean reward gain **0.864**, training-fit SD (four decimals) **0.0138**, and 95% interval [**0.856**, **0.872**]. Here training-fit SD is the sample standard deviation of the ten fits' mean reward gains on the fixed benchmark, not the standard error of their overall mean. How many independent dynamics fits enter this calculation? **10**.

Going further: Look at the two-sided t-tests the notebook prints, against BC and against the calibrated heuristic. Why would treating the 640 episodes as 640 independent training runs answer a different question?

Solution: *The independent ten-fit replication gives mean reward gain 0.864 [0.856, 0.872], training-fit SD 0.0138, and primary two-sided $p = 1.11 \times 10^{-17}$. 640/640 evaluation episodes succeed. The separately labeled affine difference is 0.029 [0.021, 0.037]. The unit for this interval is a whole fit, whereas the preceding goal intervals hold fitted models fixed. Neither estimates variability over new demonstration archives or BC fits.*

The supplied fit used in Part 6.5 happens to be the weakest of the eleven: its gain is 0.838 over BC and 0.003 over the calibrated heuristic, while the ten replication fits range from 0.845 to 0.883 and from 0.010 to 0.048. So Part 6.5 finds no evidence that this particular fit beats the calibrated heuristic, while across fits the gain over the heuristic is small but consistently positive ($t(9) = 6.66$, $p = 9.3 \times 10^{-5}$). The two answers describe different things: one fit's performance across goals, and a typical fit's.

The 640 episodes are ten fits evaluated on the same 64 goals. Episodes from one fit share that fit's model errors, and episodes on one goal share its difficulty, so they are not 640 independent samples of training randomness. Treating them as independent would count each fit 64 times and make the interval far too narrow; the training-randomness question has ten samples, one per fit.