LQR Homework

Problem 1

This is how **state** x and **control** u evolve according to:

$$x[t+1] = Ax[t] + Bu[t]$$

where:

- $x[t] \in \mathbb{R}^n$ is the state vector.
- $u[t] \in \mathbb{R}^m$ is the control input.
- $A \in \mathbb{R}^{n\times n}$ is the state transition matrix.
- $B \in \mathbb{R}^{n\times m}$ is the control matrix.

We seek to determine the optimal control sequence $u[t]$ that minimizes the **quadratic cost function** over a finite horizon T:

$$J = \sum_{t=0}^{T} \left(x[t]^T Q x[t] + u[t]^T R u[t] \right)$$

where:

- $Q \in \mathbb{R}^{n\times n}$ is the **state cost matrix** (positive semi-definite).
- $R \in \mathbb{R}^{m\times m}$ is the **control cost matrix** (positive definite).

The **cost-to-go function** at time t, assuming optimal actions are taken thereafter, is a **quadratic function** of the state:

$$V_t(x) = x[t]^T P[t] x[t].$$

Then we use **Bellman Equation** to derive the recursive equations to compute $P[t]$ and obtain the **optimal control law** $u[t]$.

$$V_t(x) = \min_{u[t]} \left(x[t]^T Q x[t] + u[t]^T R u[t] + V_{t+1}(x[t+1]) \right).$$

Then we substitute the system dynamics:

$$x[t+1] = Ax[t] + Bu[t],$$

we get:

$$V_t(x) = \min_{u[t]} \Big(x[t]^T Q x[t] + u[t]^T R u[t] + (Ax[t] + Bu[t])^T P[t+1](Ax[t] + Bu[t]) \Big).$$

Expanding the quadratic terms:

$$V_t(x) = \min_{u[t]} \Big(x[t]^T Q x[t] + u[t]^T R u[t] + x[t]^T A^T P[t+1] A x[t]$$

$$+ u[t]^T B^T P[t+1] B u[t] + 2x[t]^T A^T P[t+1] B u[t] \Big).$$

Then we differentiate $V_t(x)$ with respect to $u[t]$ and set it to zero to find the optimal $u[t]$:

$$\frac{d}{du}\left(V_t(x)\right) = 0.$$

$$(R + B^T P[t+1] B)u + B^T P[t+1] A x = 0.$$

Solving for u^*:

$$u^* = -(R + B^T P[t+1] B)^{-1} B^T P[t+1] A x.$$

Thus, the **optimal control law** is:

$$u[t] = -K[t]x[t],$$

where the **LQR feedback gain matrix** is:

$$K[t] = (R + B^T P[t+1] B)^{-1} B^T P[t+1] A.$$

Then we substitute u^* into the value function:

$$V_t(x) = x^T Q x + (-Kx)^T R(-Kx) + x^T A^T P[t+1] A x$$

$$+ 2x^T A^T P[t+1] B(-Kx) + (-Kx)^T B^T P[t+1] B(-Kx).$$

We get:

$$V_t(x) = x^T (Q + A^T P[t+1] A - A^T P[t+1] BK - K^T B^T P[t+1] A + K^T B^T P[t+1] BK +$$

The equation simplifies to:

$$P[t] = Q + A^T P[t+1]A - A^T P[t+1]B(R + B^T P[t+1]B)^{-1}B^T P[t+1]A.$$

This is **Riccati Equation**. As you can see we can solve backward using the terminal condition below via dynamic programming:

$$P[T] = Q.$$

Problem 2.1

$$A = \begin{bmatrix} 1 & dt \\ 0 & 1 \end{bmatrix}, B = \begin{bmatrix} 0 \\ dt \end{bmatrix}.$$

Problem 3.1

The following are the parameters of the **undamped pendulum**:

- **Unit length** $l = 1$
- **Unit mass** $m = 1$
- **Gravity** $g = -10$
- **Control input** u (torque applied at the pivot)

The system evolves as:

$$\theta_{t+1} = \theta_t + \dot{\theta}_t dt,$$

$$\dot{\theta}_{t+1} = \dot{\theta}_t + (-\frac{3g}{2l}\sin\theta_t + \frac{3}{ml^2}u)dt.$$

Substituting $g = 10, l = 1, m = 1$, we get:

$$\dot{\theta}_{t+1} = \dot{\theta}_t + (15\sin\theta_t + 3u)dt.$$

Then we linearize the system around $\theta_t = 0$**,** $\dot{\theta}_t = 0$

To linearize, we approximate $\sin\theta_t$ using a **Taylor series expansion** around $\theta_t = 0$:

$$\sin\theta_t = \theta_t - \frac{\theta_t^3}{3!} + \mathcal{O}(\theta_t^5).$$

For small angles, the higher-order terms $\frac{\theta_t^3}{3!}$ and beyond become negligible, leading to the approximation:

$$\sin\theta_t \approx \theta_t.$$

Thus, the second equation simplifies to:

$$\dot{\theta}_{t+1} = \dot{\theta}_t + (15\theta_t + 3u)dt.$$

Rewriting the system in **state-space form**:

$$\begin{bmatrix} \theta_{t+1} \\ \dot{\theta}_{t+1} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 15 & 0 \end{bmatrix} \begin{bmatrix} \theta_t \\ \dot{\theta}_t \end{bmatrix} + \begin{bmatrix} 0 \\ 3 \end{bmatrix} u.$$

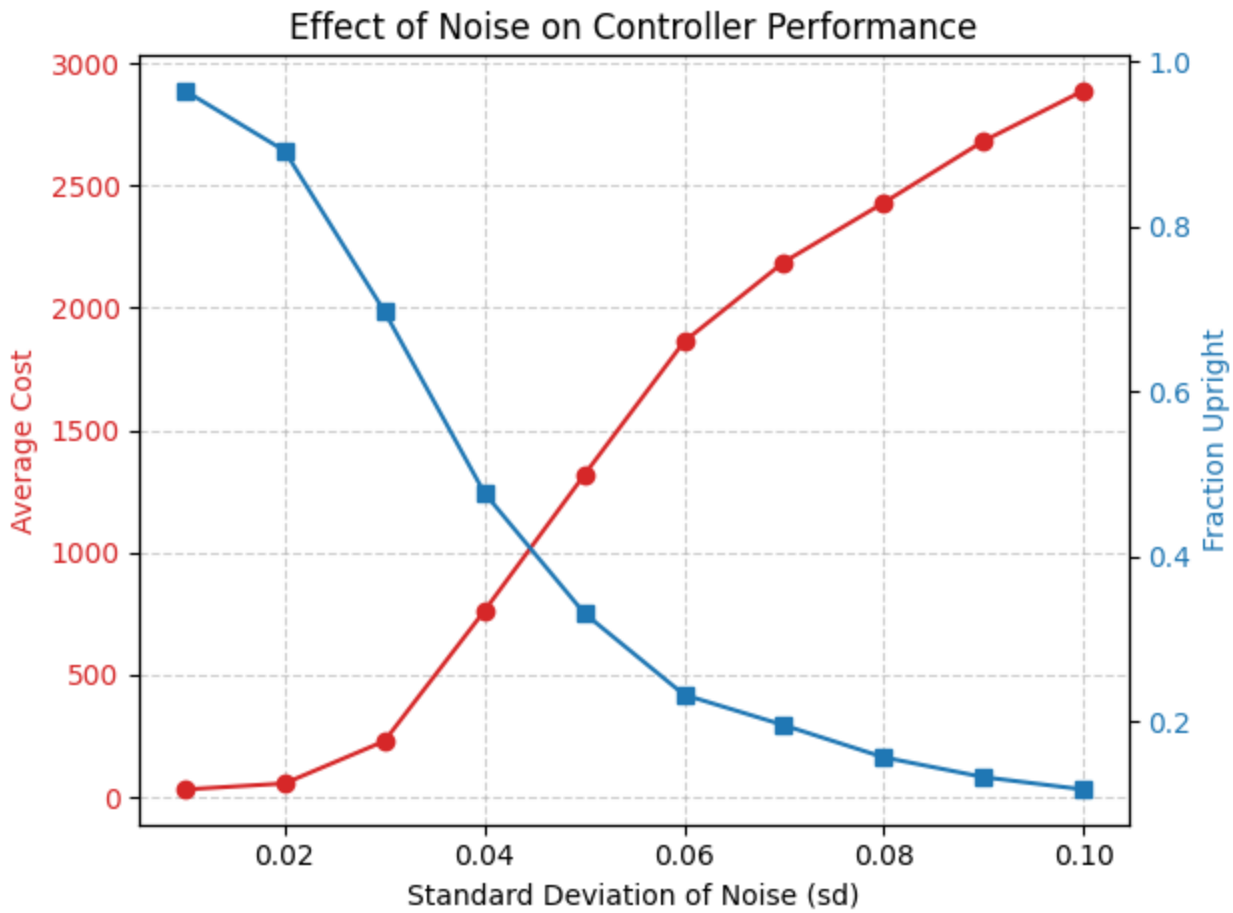
Thus, the **discrete-time linearized system matrices** then are:

$$A = \begin{bmatrix} 1 & dt \\ 15dt & 1 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ 3dt \end{bmatrix}.$$

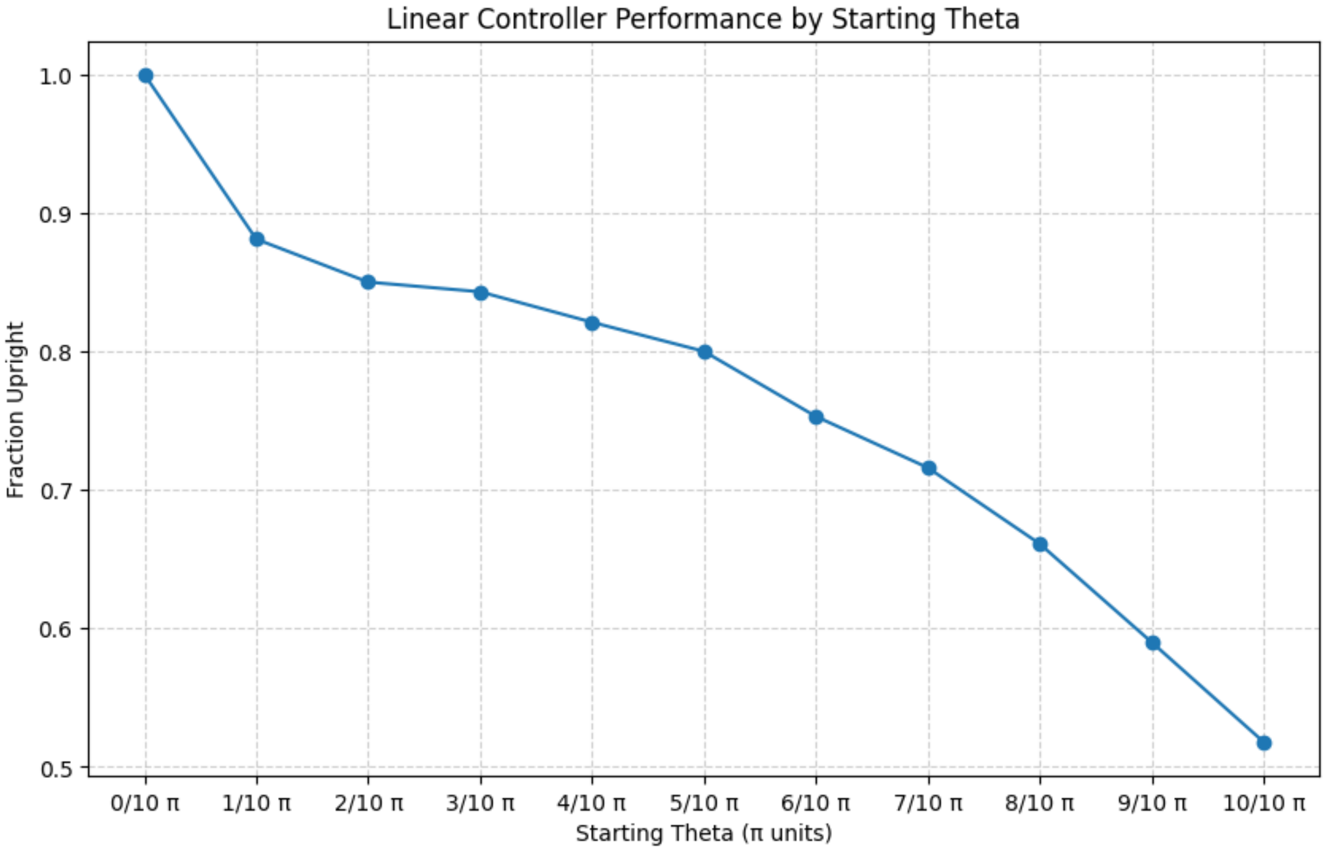
The **Cost Matrices** are:

$$Q = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad R = \begin{bmatrix} 1 \end{bmatrix}.$$

Problem 3.3


Effect of Noise on Controller Performance
Average Cost
Fraction Upright
Standard Deviation of Noise (sd)
3000
2500
2000
1500
1000
500
0
1.0
0.8
0.6
0.4
0.2
0.02
0.04
0.06
0.08
0.10

Problem 3.4


Linear Controller Performance by Starting Theta
Fraction Upright
1.0
0.9
0.8
0.7
0.6
0.5
0/10 π
1/10 π
2/10 π
3/10 π
4/10 π
5/10 π
6/10 π
7/10 π
8/10 π
9/10 π
10/10 π
Starting Theta (π units)

6DOF Homework

Problem 2

The experiments I have done and the validation metrics are shown below:

Experiment	Iteration	cls_accuracy	cls_R_t_accuracy	overall
Basic Model	40000	0.540578	0.046768	0.951857
Cropped Images as Input	46000	0.865199	0.442916	2.559835
6D Rotation Representation & Loss	93000	0.466300	0.027510	0.825309
Multiple Heads for Different Object Classes	47000	0.477304	0.071527	0.944979
Ensemble of All Modifications	65000	0.723521	0.235213	1.687758

My final model is the Cropped Images as Input one which comprises just the basic model architecture and the default training recipe and hyperparameters. The only modification I made is that I just cropped the input images to their boundng boxes.

Hyperparameter	Value	Description
`lr`	1e-4	Learning Rate
`weight_decay`	1e-4	Weight Decay for optimizer
`output_dir`	`runs/basic/`	Output Directory
`data_dir`	`data/ycbv/v1/`	Data Directory
`batch_size`	16	Batch Size
`seed`	2	Random seed

Hyperparameter	Value	Description
`batch_size`	16	Batch Size
`seed`	2	Random seed
`max_iter`	100000	Total Iterations
`val_every`	1000	Iterations interval to validate
`save_every`	50000	Iterations interval to save model
`preload_images`	1	Whether to preload train and val images
`lr_step`	[60000, 80000]	Iterations to reduce learning rate

The plot validation metrics of my best model is shown below: