Rotational Dynamics and LQR Control of a Seesaw

A Comprehensive Overview

Course Outline

1. **Seesaw System Modeling**
   - Rotational Dynamics Fundamentals
   - Torque Components (Gravity, Friction, Control)
   - Equation of Motion & State-Space Model
2. **Linearization and Stability Analysis**
   - Equilibrium Points
   - Linearization Technique
   - Stability via Eigenvalues and Lyapunov

3. **Linear Quadratic Regulator (LQR) Control**
   - Introduction to Optimal Control
   - Continuous-Time Infinite-Horizon LQR
   - Discrete-Time Finite-Horizon LQR
4. **Simulation and Conclusion**

1. Seesaw System Modeling

Understanding the Physics

Newton's Second Law for Rotation

For any rigid body rotating about a fixed pivot, the fundamental law is:

$$I\ddot{\theta} = \sum \tau$$

Our goal is to express each torque term and derive the equation for $\ddot{\theta}$.

Where:

- I: Total moment of inertia about the pivot.
- θ: Rotation angle (e.g., from horizontal).
- $\ddot{\theta}$: Angular acceleration.
- $\sum \tau$: Sum of all external torques acting on the system.

Gravitational Torque (τ_g)

Consider two masses m_1 and m_2 at the ends of a beam of length L, pivoted at the center ($r = L/2$).

Position vectors ($\mathbf{r}$) and gravitational forces ($\mathbf{F}_g$):

$$\mathbf{r}_1 = \frac{L}{2}\begin{pmatrix}\cos\theta \\ \sin\theta\end{pmatrix}, \quad \mathbf{F}_{g1} = \begin{pmatrix}0 \\ -m_1\,g\end{pmatrix}$$

$$\mathbf{r}_2 = -\frac{L}{2}\begin{pmatrix}\cos\theta \\ \sin\theta\end{pmatrix}, \quad \mathbf{F}_{g2} = \begin{pmatrix}0 \\ -m_2\,g\end{pmatrix}$$

Torque (2D scalar cross product $\tau = xF_y - yF_x$):

Gravitational Torque Term in Equation

Dividing the net gravitational torque by the moment of inertia I:

$$\frac{\tau_g}{I} = \frac{(m_2 - m_1) g L}{2 I} \cos\theta$$

We define the constant γ:

$$\gamma = \frac{(m_2 - m_1) g L}{2 I}$$

So the gravitational contribution to $\ddot{\theta}$ is:

$$\boxed{\gamma \cos\theta}$$

Friction Torque ($\tau_{friction}$)

Two types of friction are considered:

1. **Viscous Friction:** Proportional to angular velocity $\dot{\theta}$.

$$\tau_{\text{viscous}} = -b\,\dot{\theta}$$

   where b is the viscous damping coefficient.

2. **Coulomb (Dry) Friction:** Constant magnitude, opposes motion. Modeled smoothly using $\tanh$:

$$\tau_{\text{Coulomb}} \approx -\mu_c \frac{g\,L}{2} \tanh\left(\frac{\dot{\theta}}{\epsilon}\right)$$

where and ϵ is a small positive constant.

Friction Torque Terms in Equation

Total Friction Torque:

$$\tau_{\text{friction}} = -b\,\dot{\theta} - \mu_c\,\frac{g\,L}{2}\,\tanh\left(\frac{\dot{\theta}}{\epsilon}\right)$$

Dividing by I:

$$\frac{\tau_{\text{friction}}}{I} = -\frac{b}{I}\,\dot{\theta} - \frac{\mu_c\,g\,L}{2I}\,\tanh\left(\frac{\dot{\theta}}{\epsilon}\right)$$

Define constants α and β:

$$\alpha = \frac{b}{I} \quad , \quad \beta = \frac{\mu_c\, g\, L}{2\, I}$$

Friction contribution to $\ddot{\theta}$:

$$\boxed{-\alpha\,\dot{\theta} - \beta\,\tanh\left(\frac{\dot{\theta}}{\epsilon}\right)}$$

Total Moment of Inertia (I)

The total moment of inertia is the sum of the inertia of the beam (I_b) and the point masses (I_{points}).

1. **Beam Inertia (I_b):** For a uniform beam of mass m_b, length L, rotated about its center:

$$I_b = \frac{1}{12} m_b L^2$$

2. **Point Masses Inertia (I_{points}):** Each mass m_i at distance $r = L/2$: $I_i = m_i (L/2)^2$.

$$I_{\text{points}} = \frac{1}{4} (m_1 + m_2) L^2$$

Control Force Torque ($\tau_{control}$)

Assume a control force F (e.g., from a drone motor) acts vertically at one end of the beam (say, $x = -L/2$, position vector $\mathbf{r} = (-L/2, 0)$). The force vector is $\mathbf{F} = (0, F)$.

Using $\tau = xF_y - yF_x$:

$$\tau_{\text{control}} = \left(-\frac{L}{2}\right)(F) - (0)(0) = -\frac{L}{2}F$$

(Note: The sign might differ based on convention/setup. We'll use the document's $\tau_{control} = \frac{L}{2}F$ assuming force at $x = L/2$ or opposite direction)

Assume force F acts vertically at $x = L/2$. Position $\mathbf{r} = (L/2)\binom{\cos\theta}{\sin\theta}$, Force $\mathbf{F} = (0, F)$ in vertical frame. Torque $\tau = rF\sin(\theta_{rel})$. If force acts vertically, $\tau = (L/2\cos\theta)F$.
Let's assume the document's final $\tau = \delta F$ term is correct, where $\delta = L/(2I)$.

Control contribution to $\ddot{\theta}$:

$$\boxed{\delta F}$$

Complete Equation of Motion

Combining all torque terms divided by I:

$$I\ddot{\theta} = \tau_g + \tau_{friction} + \tau_{control}$$

$$\ddot{\theta} = \frac{\tau_g}{I} + \frac{\tau_{friction}}{I} + \frac{\tau_{control}}{I}$$

$$\ddot{\theta} = \gamma\,\cos\theta - \alpha\,\dot{\theta} - \beta\,\tanh\left(\frac{\dot{\theta}}{\epsilon}\right) + \delta\,F$$

Rearranging:

$$\boxed{\ddot{\theta} + \alpha\,\dot{\theta} + \beta\,\tanh\left(\frac{\dot{\theta}}{\epsilon}\right) = \gamma\,\cos\theta + \delta\,F}$$

With constants: $\alpha = \frac{b}{I}, \beta = \frac{\mu_c\,g\,L}{2I}, \gamma = \frac{(m_2-m_1)g\,L}{2I}, \delta = \frac{L}{2I}$.

State-Space Representation

Define the state vector $\mathbf{x}$:

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} \theta \\ \dot{\theta} \end{pmatrix}$$

The state equations are:

$$\dot{x}_1 = x_2$$

$$\dot{x}_2 = \ddot{\theta} = -\alpha\, x_2 - \beta \tanh\left(\frac{x_2}{\epsilon}\right) + \gamma \cos x_1 + \delta\, F$$

In vector form $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, F)$:

$$\boxed{\dot{\mathbf{x}} = \begin{pmatrix} x_2 \\ -\alpha\, x_2 - \beta\, \tanh\left(\frac{x_2}{\epsilon}\right) + \gamma\, \cos x_1 + \delta\, F \end{pmatrix}}$$

This is a **nonlinear** state-space model.

2. Linearization and Stability Analysis

Approximating the System Behavior

Equilibrium Points

Equilibrium points $(\mathbf{x}_{eq}, F_{eq})$ are states where the system remains stationary if undisturbed. They satisfy $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}_{eq}, F_{eq}) = \mathbf{0}$.

1. $\dot{x}_1 = x_2 = 0 \implies \dot{\theta}_{eq} = 0$ (No rotation at equilibrium).
2. $\dot{x}_2 = -\alpha(0) - \beta \tanh(0) + \gamma \cos x_1 + \delta F = 0$

$$\gamma \cos \theta_{eq} + \delta F_{eq} = 0$$

$$\cos \theta_{eq} = -\frac{\delta F_{eq}}{\gamma}$$

Equilibrium Points (Cases)

- **Case 1: No Control ($F_{eq} = 0$)**

$$\cos\theta_{eq} = 0 \implies \theta_{eq} = \pm\frac{\pi}{2}, \pm\frac{3\pi}{2}, \ldots$$

  The physically distinct equilibria are the vertical positions: $(\theta_{eq}, \dot{\theta}_{eq}) = (\pi/2, 0)$ and $(-\pi/2, 0)$.

- **Case 2: With Control ($F_{eq} \neq 0$)**
  Equilibrium exists if $|\frac{\delta F_{eq}}{\gamma}| \leq 1$.
  If so, $\theta_{eq} = \pm\arccos(-\frac{\delta F_{eq}}{\gamma}) + 2k\pi$. The control force can stabilize the seesaw at angles other than vertical.

Linearization

We approximate the nonlinear system $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, F)$ near an equilibrium point $(\mathbf{x}_{eq}, F_{eq})$ with a linear system.
Define deviation variables: $\mathbf{z} = \mathbf{x} - \mathbf{x}_{eq}$, $\Delta F = F - F_{eq}$.
Using Taylor series expansion around the equilibrium:

$$\dot{\mathbf{x}} \approx \mathbf{f}(\mathbf{x}_{eq}, F_{eq}) + \underbrace{\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\bigg|_{eq}}_{A} (\mathbf{x} - \mathbf{x}_{eq}) + \underbrace{\frac{\partial \mathbf{f}}{\partial F}\bigg|_{eq}}_{B} (F - F_{eq})$$

Since $\mathbf{f}(\mathbf{x}_{eq}, F_{eq}) = \mathbf{0}$ and $\dot{\mathbf{z}} = \dot{\mathbf{x}}$:

$$\boxed{\dot{\mathbf{z}} = A\mathbf{z} + B\Delta F}$$

A is the Jacobian matrix, B is the input matrix.

Calculating Jacobian Matrices (A, B)

System function:

$$\mathbf{f}(\mathbf{x}, F) = \begin{pmatrix} x_2 \\ -\alpha\, x_2 - \beta\, \tanh(\frac{x_2}{\epsilon}) + \gamma\, \cos x_1 + \delta\, F \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \end{pmatrix}$$

Matrix A (Jacobian w.r.t. x):

$$A = \frac{\partial \mathbf{f}}{\partial \mathbf{x}} = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -\gamma \sin x_1 & -\alpha - \frac{\beta}{\epsilon}\mathrm{sech}^2(\frac{x_2}{\epsilon}) \end{pmatrix}$$

Matrix B (Jacobian w.r.t. F):

$$B = \frac{\partial \mathbf{f}}{\partial F} = \begin{pmatrix} \frac{\partial f_1}{\partial F} \\ \frac{\partial f_2}{\partial F} \end{pmatrix} = \begin{pmatrix} 0 \\ \delta \end{pmatrix}$$

Linearized System at Equilibrium

Evaluate A at an equilibrium point $(\mathbf{x}_{eq}, F_{eq}) = (\theta_{eq}, 0, F_{eq})$.
Note that $\operatorname{sech}^2(0) = 1$. Let $\tilde{\alpha} = \alpha + \beta/\epsilon$.

$$A_{eq} = \begin{pmatrix} 0 & 1 \\ -\gamma \sin\theta_{eq} & -\tilde{\alpha} \end{pmatrix} \quad , \quad B_{eq} = \begin{pmatrix} 0 \\ \delta \end{pmatrix}$$

The linearized system describing deviations $\mathbf{z} = (\Delta\theta, \Delta\dot{\theta})^T$ is:

$$\boxed{\begin{pmatrix} \dot{z}_1 \\ \dot{z}_2 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -\gamma \sin\theta_{eq} & -\tilde{\alpha} \end{pmatrix} \begin{pmatrix} z_1 \\ z_2 \end{pmatrix} + \begin{pmatrix} 0 \\ \delta \end{pmatrix} \Delta F}$$

Stability Analysis of Equilibria (F=0)

Stability depends on the eigenvalues of A_{eq}. Assume $m_2 > m_1$ ($\gamma > 0$) and damping $\tilde{\alpha} > 0$.

- **Equilibrium $\theta_{eq} = \pi/2$ (Mass m_2 down):**
  $\sin(\pi/2) = 1$.
  $$A = \begin{pmatrix} 0 & 1 \\ -\gamma & -\tilde{\alpha} \end{pmatrix}$$
  Characteristic equation: $\lambda^2 + \tilde{\alpha}\lambda + \gamma = 0$.
  Eigenvalues have negative real parts ($\mathrm{Re}(\lambda) = -\tilde{\alpha}/2 < 0$).
  Result: Stable equilibrium.

- **Equilibrium $\theta_{eq} = -\pi/2$ (Mass m_2 up):**
  $\sin(-\pi/2) = -1.$
  $$A = \begin{pmatrix} 0 & 1 \\ \gamma & -\tilde{\alpha} \end{pmatrix}$$
  Characteristic equation: $\lambda^2 + \tilde{\alpha}\lambda - \gamma = 0.$
  One eigenvalue has a positive real part.
  Result: Unstable equilibrium.

(Stability reverses if $m_1 > m_2$)

Lyapunov Stability (Introduction)

A more rigorous way to prove stability, especially for nonlinear systems (but here applied to the linear one).

Lyapunov's Direct Method: Find a scalar function $V(\mathbf{z})$, called a Lyapunov function, such that:

1. $V(\mathbf{0}) = 0$
2. $V(\mathbf{z}) > 0$ for $\mathbf{z} \neq \mathbf{0}$ (Positive Definite)
3. $\dot{V}(\mathbf{z}) \leq 0$ along system trajectories (Negative Semi-Definite)

If $\dot{V}(\mathbf{z}) < 0$ for $\mathbf{z} \neq \mathbf{0}$ (Negative Definite), the origin is asymptotically stable.

Lyapunov Analysis for Linear System $\dot{\mathbf{z}} = A\mathbf{z}$

Choose a quadratic Lyapunov candidate: $V(\mathbf{z}) = \mathbf{z}^T P\mathbf{z}$, where P is a symmetric positive definite matrix ($P = P^T \succ 0$).

Calculate the time derivative $\dot{V}$:

$$\dot{V}(\mathbf{z}) = \dot{\mathbf{z}}^T P\mathbf{z} + \mathbf{z}^T P\dot{\mathbf{z}}$$

$$\dot{V}(\mathbf{z}) = (A\mathbf{z})^T P\mathbf{z} + \mathbf{z}^T P(A\mathbf{z})$$

$$\dot{V}(\mathbf{z}) = \mathbf{z}^T (A^T P + PA)\mathbf{z}$$

For asymptotic stability, we require $\dot{V}$ to be negative definite.

The Lyapunov Equation

We can *enforce* $\dot{V}$ to be negative definite by setting:

$$A^T P + PA = -Q$$

where Q is any symmetric positive definite matrix (e.g., $Q = I$).
This is the **Algebraic Lyapunov Equation**.

If we can find a symmetric $P \succ 0$ that solves this equation for a given A and some $Q \succ 0$, then the system $\dot{\mathbf{z}} = A\mathbf{z}$ is asymptotically stable.

Solving Lyapunov Equation for Stable Seesaw Equilibrium

Consider the stable case: $A = \begin{pmatrix} 0 & 1 \\ -\gamma & -\tilde{\alpha} \end{pmatrix}$ (with $\gamma > 0, \tilde{\alpha} > 0$). Let $Q = I$.

Solve $A^T P + PA = -I$ for $P = \begin{pmatrix} p_{11} & p_{12} \\ p_{12} & p_{22} \end{pmatrix}$.

This yields a system of linear equations for p_{ij}. The solution is:

$$P = \begin{pmatrix} \frac{1+\gamma}{2\tilde{\alpha}} + \frac{\tilde{\alpha}}{2\gamma} & \frac{1}{2\gamma} \\ \frac{1}{2\gamma} & \frac{1+\gamma}{2\gamma\tilde{\alpha}} \end{pmatrix}$$

3. Linear Quadratic Regulator (LQR) Control

Designing Optimal Feedback

What is LQR?

LQR (Linear Quadratic Regulator) is a cornerstone of modern control theory.

Goal: Design an optimal state-feedback controller $u = -Kx$ for a linear system ($\dot{x} = Ax + Bu$ or $x_{k+1} = Gx_k + Hu_k$).

Optimality: The controller minimizes a quadratic cost function that penalizes state deviations from the origin and control effort.

Result: Provides a systematic way to calculate the feedback gain K, balancing performance and control energy.

LQR Context for the Seesaw

We apply LQR to the **linearized** seesaw model around a desired (possibly unstable) equilibrium point $\mathbf{x}_{eq}$.

$$\dot{\mathbf{z}} = A\mathbf{z} + B\Delta F$$

Where $\mathbf{z} = \mathbf{x} - \mathbf{x}_{eq}$ and $\Delta F = F - F_{eq}$.

Goal: Find the optimal control deviation $\Delta F(t)$ (or ΔF_k) to drive the deviation $\mathbf{z}$ to zero while minimizing cost.
The LQR controller will compute the gain K for the feedback law:

$$\Delta F = -K\mathbf{z}$$

Continuous-Time Infinite-Horizon LQR

The Performance Index (Cost Function)

We want to minimize the integral of a quadratic cost over an infinite time horizon:

$$J = \int_0^\infty \left(\mathbf{z}(t)^T Q \mathbf{z}(t) + \Delta F(t)^T R \Delta F(t)\right) dt$$

Where:

- $\mathbf{z}$: State deviation vector $(\Delta\theta, \Delta\dot{\theta})^T$.
- ΔF: Control input deviation (scalar).
- Q: State weighting matrix (2×2, symmetric, positive semi-definite, $Q \geq 0$). Penalizes state errors.
- R: Control weighting matrix (1×1 scalar, positive definite, $R > 0$). Penalizes control effort.

Weighting Matrices Q and R

The choice of Q and R determines the trade-off:

- **Q Matrix:** Penalizes state deviations.

$$Q = \begin{pmatrix} q_{11} & 0 \\ 0 & q_{22} \end{pmatrix}, \quad q_{11}, q_{22} \geq 0$$

  - Larger q_{11}: Stronger penalty on angle error $(\Delta\theta)^2$.
  - Larger q_{22}: Stronger penalty on angular velocity error $(\Delta\dot{\theta})^2$.

- **R Scalar:** Penalizes control effort.
  - Larger R: Control input ΔF is more "expensive", controller will be less aggressive.
  - Smaller R: Control input is "cheaper", controller can use larger inputs for faster response.

Tuning Q and R is crucial for desired performance.

The Optimal Control Law

The control law that minimizes the cost J for the linear system $\dot{\mathbf{z}} = A\mathbf{z} + B\Delta F$ is a linear state feedback:

$$\boxed{\Delta F(t) = -K\mathbf{z}(t)}$$

Where K is the constant optimal feedback gain matrix (1×2 for the seesaw).

How do we find K?

The Algebraic Riccati Equation (ARE)

The key to finding K is solving the **Algebraic Riccati Equation (ARE)** for a unique symmetric, positive semi-definite matrix P:

$$\boxed{A^T P + PA - PBR^{-1}B^T P + Q = 0}$$

- A, B are the linearized system matrices.
- Q, R are the chosen weighting matrices.
- P is the unknown solution (2×2 matrix for the seesaw).

If the system (A, B) is stabilizable, a unique stabilizing solution $P \geq 0$ exists.

ARE for the Seesaw

Substitute $A = \begin{pmatrix} 0 & 1 \\ -\gamma \sin\theta_{eq} & -\tilde{\alpha} \end{pmatrix}$, $B = \begin{pmatrix} 0 \\ \delta \end{pmatrix}$, R (scalar $r > 0$), Q.

Let $P = \begin{pmatrix} p_{11} & p_{12} \\ p_{12} & p_{22} \end{pmatrix}$.

The term $PBR^{-1}B^TP$ becomes:

$$PBR^{-1}B^TP = \begin{pmatrix} p_{11} & p_{12} \\ p_{12} & p_{22} \end{pmatrix} \begin{pmatrix} 0 \\ \delta \end{pmatrix} \frac{1}{r} \begin{pmatrix} 0 & \delta \end{pmatrix} \begin{pmatrix} p_{11} & p_{12} \\ p_{12} & p_{22} \end{pmatrix} = \frac{\delta^2}{r} \begin{pmatrix} p_{12}^2 & p_{12}p_{22} \\ p_{12}p_{22} & p_{22}^2 \end{pmatrix}$$

The full ARE leads to a system of coupled *nonlinear* algebraic equations for p_{11}, p_{12}, p_{22}.

Solving the ARE

Solving the ARE analytically is complex. Numerically, it's straightforward using standard software tools.

Octave/MATLAB: The `lqr` command solves the ARE and computes the optimal gain K.

```
% System parameters... (m1, m2, L, g, b, uc, eps)
% Calculated constants... (I, aa, bb, cc, dd)
% Equilibrium point (theta_e)
theta_e = -pi/2; % Or other desired equilibrium

% Linearized matrices at equilibrium
A = [ 0, 1; -cc*sin(theta_e), -aa - bb/eps];
B = [0; dd];

% Choose LQR weights
Q = diag([5, 0.1]); % Example: Penalize angle more than velocity
R = 0.1;            % Example: Control effort penalty

% Solve ARE and find K
[K, S, e_vals] = lqr(A, B, Q, R);
% K: Optimal gain [k1, k2]
% S: Solution P of ARE
% e_vals: Closed-loop eigenvalues (should have Re < 0)
```

Optimal Gain Calculation

Once the ARE solution P is found, the optimal gain matrix K is calculated directly:

$$\boxed{K = R^{-1}B^TP}$$

For the seesaw (R scalar r, $B = \begin{pmatrix} 0 \\ \delta \end{pmatrix}$, $P = \begin{pmatrix} p_{11} & p_{12} \\ p_{12} & p_{22} \end{pmatrix}$):

$$K = \frac{1}{r}\begin{pmatrix}0 & \delta\end{pmatrix}\begin{pmatrix}p_{11} & p_{12} \\ p_{12} & p_{22}\end{pmatrix} = \frac{1}{r}\begin{pmatrix}\delta p_{12} & \delta p_{22}\end{pmatrix}$$

So the gains are $k_1 = \frac{\delta p_{12}}{r}$ and $k_2 = \frac{\delta p_{22}}{r}$.
The control law is $\Delta F = -k_1 z_1 - k_2 z_2 = -k_1 \Delta\theta - k_2 \Delta\dot{\theta}$.

Closed-Loop Stability

The LQR controller guarantees stability of the closed-loop system:

$$\dot{\mathbf{z}} = (A - BK)\mathbf{z}$$

The Lyapunov function $V(\mathbf{z}) = \mathbf{z}^T P \mathbf{z}$ (where P solves the ARE) can be used to prove this. Its derivative along the closed-loop trajectory is:

$$\dot{V}(\mathbf{z}) = -\mathbf{z}^T (Q + K^T R K)\mathbf{z}$$

Since $Q \geq 0$ and $K^T R K \geq 0$ (as $R > 0$), $\dot{V}(\mathbf{z}) \leq 0$. With standard assumptions (observability related to Q), it can be shown $\dot{V}$ is negative definite, guaranteeing asymptotic stability. The eigenvalues of $(A - BK)$ will have negative real parts.

Discrete-Time Finite-Horizon LQR

Why Discrete-Time?

Controllers are often implemented on digital computers (microcontrollers, FPGAs). These operate at discrete time steps with a specific sampling period T_s.

We need:

1. A **discrete-time model** of the seesaw.
2. A **discrete-time LQR formulation**.

We consider control over a **finite time horizon** (N steps).

Discrete-Time Seesaw Model

We discretize the *linearized* continuous-time system $\dot{\mathbf{z}} = A\mathbf{z} + B\Delta F$ using a sampling period T_s. Assuming Zero-Order Hold (ZOH) on the input ΔF:

$$\boxed{\mathbf{z}_{k+1} = G\mathbf{z}_k + H\Delta F_k}$$

Where:

- $\mathbf{z}_k = \mathbf{z}(kT_s)$ is the state at step k.
- ΔF_k is the control input applied during interval $[kT_s, (k+1)T_s)$.
- $G = e^{AT_s}$ (State transition matrix).
- $H = \left(\int_0^{T_s} e^{A\tau} d\tau\right) B$ (Input matrix).

G (2×2) and H (2×1) can be computed numerically.

Discretization Code (Octave/MATLAB)

```
% Continuous system matrices A, B
A = [ 0, 1; -cc*sin(theta_e), -aa - bb/eps];
B = [0; dd];

% Sampling time
Ts = 0.01; % Example: 10 ms

% Create continuous-time state-space model object
sysc = ss(A, B, eye(2), 0); % eye(2) for C, 0 for D

% Discretize using Zero-Order Hold
sysd = c2d(sysc, Ts, 'zoh');

% Extract discrete-time matrices
Ad = sysd.A; % This is our G
Bd = sysd.B; % This is our H
```

Discrete-Time Performance Index

Minimize a sum of quadratic costs over a finite horizon of N steps:

$$J_N = \mathbf{z}_N^T Q_f \mathbf{z}_N + \sum_{k=0}^{N-1} (\mathbf{z}_k^T Q_d \mathbf{z}_k + \Delta F_k^T R_d \Delta F_k)$$

Where:

- Q_d: State weighting matrix ($2 \times 2, \geq 0$) per step.
- R_d: Control weighting scalar (> 0) per step.
- Q_f: Final state weighting matrix ($2 \times 2, \geq 0$) at step N.
- N: Horizon length (number of steps).

Optimal Control Law (Discrete, Finite Horizon)

The optimal control law is again a linear state feedback, but the gain is now **time-varying**:

$$\boxed{\Delta F_k = -K_k \mathbf{z}_k}$$

The gain K_k (1×2) changes at each time step k from 0 to $N-1$.

How do we find the sequence of gains $K_0, K_1, \ldots, K_{N-1}$?

Bellman's Principle of Optimality

The solution relies on dynamic programming and Bellman's Principle: *An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision.*

Implication: To find the optimal action ΔF_k at step k, we need to know the minimum possible cost from step $k+1$ onwards. This suggests solving the problem *backward* in time.

The Riccati Difference Equation (RDE)

Similar to the ARE, the **Riccati Difference Equation (RDE)** is solved backward in time to find the optimal cost-to-go matrix P_k and the gain K_k.

Backward Iteration:

1. **Initialization:** Set $P_N = Q_f$ (cost at the final step).
2. **Iteration (for $k = N - 1$ down to 0):**
    - Calculate the gain K_k:

$$K_k = (R_d + H^T P_{k+1} H)^{-1} H^T P_{k+1} G$$

- Calculate the cost matrix P_k:

$$P_k = G^T P_{k+1} G + Q_d - G^T P_{k+1} H K_k$$

(Alternative form:
$P_k = Q_d + K_k^T R_d K_k + (G - HK_k)^T P_{k+1} (G - HK_k)$*)*

This gives the sequence $P_{N-1}, \ldots, P_0$ and $K_{N-1}, \ldots, K_0$.

RDE Code (Octave/MATLAB)

```
% Discrete matrices Ad (G), Bd (H)
% Weights Qd, Rd (scalar), Qf
% Horizon N

% Initialize P matrix history (N+1 steps, N to 0)
P = zeros(2, 2, N + 1);
K = zeros(1, 2, N); % Gain history (N steps, N-1 to 0)

% Terminal cost
P(:,:,N+1) = Qf; % Index N+1 corresponds to P_N

% Backward iteration
for k = N:-1:1 % Corresponds to steps k=N-1 down to 0
    P_kp1 = P(:,:,k+1); % P_{k+1}

    % Calculate gain K_k (index k used for K_{k-1})
    K_k = (Rd + Bd' * P_kp1 * Bd) \ (Bd' * P_kp1 * Ad);
    K(:,:,k) = K_k;

    % Calculate P_k (index k used for P_{k-1})
    P(:,:,k) = Ad' * P_kp1 * Ad + Qd - Ad' * P_kp1 * Bd * K_k;
end
% Now K(:,:,1) is K_0, K(:,:,2) is K_1, ..., K(:,:,N) is K_{N-1}
```

Implementation

1. **Offline:** Calculate the sequence of gains $K_0, K_1, \ldots, K_{N-1}$ by solving the RDE backward. Store these gains.
2. **Online (Real-time):**
   - At each time step $k = 0, 1, \ldots, N-1$:
     - Measure or estimate the current state deviation $\mathbf{z}_k$.
     - Retrieve the pre-calculated gain K_k.
     - Compute control deviation: $\Delta F_k = -K_k \mathbf{z}_k$.
     - Compute total control: $F_k = F_{eq} + \Delta F_k$.
     - Apply F_k to the system.

Finite vs. Infinite Horizon (Discrete)

- **Finite Horizon (RDE):** Gain K_k is time-varying. Optimal for problems with a defined end time.
- **Infinite Horizon (DARE):** If $N \to \infty$ and the system is stabilizable, the RDE solution P_k converges to a constant steady-state matrix P_∞, and the gain K_k converges to a constant K_∞.
  - P_∞ satisfies the Discrete Algebraic Riccati Equation (DARE).
  - $K_\infty = (R_d + H^T P_\infty H)^{-1} H^T P_\infty G.$
  - Computed using `dlqr(Ad, Bd, Qd, Rd)` in Octave/MATLAB. Provides constant gain, simpler implementation if horizon is long.

4. Simulation & Conclusion

Simulation Example (Discrete LQR)

Using the calculated time-varying gains K_k:

```
% Initial state x0, equilibrium x_e = [theta_e; 0]
x = zeros(2, N + 1); x(:, 1) = x0;
x_err = zeros(2, N + 1);  x_err(:, 1) = x0 - x_e;
u_dev = zeros(1, N);

for k = 1:N
    % Calculate control deviation using time-varying gain K(:,:,k)
    u_dev(k) = -K(:,:,k) * x_err(:, k);
    % Update state error using discrete model
    x_err(:, k + 1) = Ad * x_err(:, k) + Bd * u_dev(k);
    % Reconstruct absolute state
    x(:, k + 1) = x_err(:, k + 1) + x_e;
end
% Plot x(1,:), x(2,:), u_dev vs time...
```

Conclusion

- Modeled the seesaw using Newton's laws, including gravity, friction, and control inputs.
- Derived the nonlinear state-space equation.
- Linearized the system around equilibrium points to facilitate linear control design.
- Analyzed stability using eigenvalues and Lyapunov's direct method.
- Introduced the LQR framework for optimal control design.
- Solved the **Continuous-Time LQR** problem using the **ARE** for an infinite horizon, yielding a constant optimal gain K.

- Solved the **Discrete-Time Finite-Horizon LQR** problem using the **RDE** solved backward in time, yielding a time-varying optimal gain sequence K_k.
- LQR provides a powerful and systematic method to stabilize systems while balancing performance (state regulation) and control effort, tunable via weighting matrices Q and R.