perplexity

How to tune the LQR controller? What do the parameters mean?

To tune an **LQR (Linear Quadratic Regulator) controller**, you adjust the weighting matrices

$$Q$$

and

$$R$$

, which define the trade-offs between state deviations and control effort in the cost function. Here's a detailed explanation of the parameters and how to tune them:

Understanding LQR Parameters

1. **Cost Function**:
   The LQR minimizes the following cost function:

   $$J = \int_0^\infty (\mathbf{x}^\top Q\mathbf{x} + \mathbf{u}^\top R\mathbf{u})dt$$

   - $\mathbf{x}$
     : State vector (e.g., position, velocity, angle, angular velocity).
   - $\mathbf{u}$
     : Control input (e.g., force or torque).
   - Q
     : State weighting matrix (penalizes deviations in states).
   - R
     : Control weighting matrix (penalizes control effort).
2. **Role of**

   $$Q$$

   :
   - Q
     is typically a diagonal matrix where each diagonal element corresponds to a state variable.
   - Larger values in
     Q
     penalize deviations in the corresponding state more heavily.
     - Example: A high penalty on pole angle ensures it stays upright but may result in larger cart movements.
     - Smaller values allow more flexibility for that state variable.
3. **Role of**

   $$R$$

   :

- R is usually a scalar or diagonal matrix that penalizes control effort.
- Larger values in R reduce actuator usage and smooth control actions but may slow down system response.
  - Smaller values make the controller more aggressive, leading to faster stabilization at the cost of higher energy usage.

Steps for Tuning LQR Parameters

1. **Start with Bryson's Rule**:
   - Initialize Q and R based on desired performance:
     - For each state variable, set the corresponding diagonal element of Q as:
       $$q_i = \frac{1}{(\text{maximum acceptable deviation})^2}$$
     - For control effort, set $R = 1/(\text{maximum acceptable input})^2$.
2. **Iterative Adjustment**:
   - Simulate the system with initial values of Q and R.
     - Observe performance metrics such as:
     - Settling time.
     - Overshoot.
     - Control effort.
     - Stability under disturbances.
     - Adjust weights iteratively:
     - Increase elements in Q to prioritize specific states (e.g., pole angle stability).
     - Increase R to reduce control energy or actuator wear.
3. **Trade-Offs**:

- Balancing Q and R involves trade-offs:
  - High penalties in Q improve state tracking but may require excessive control effort.
  - High penalties in R reduce energy usage but may slow down response time or allow larger deviations.

4. **Sensitivity-Based Tuning**[1]:
   - Use sensitivity analysis to determine how changes in Q and R affect closed-loop behavior.
     - Match desired performance metrics (e.g., settling time, damping ratio).

5. **Automated Tuning Methods**:
   - Employ optimization techniques such as genetic algorithms or Bayesian optimization to find optimal values for Q and R [2] [3].
     - Example: Define a fitness function based on performance metrics like RMS error, maximum deviation, or energy consumption.

Practical Tips

- **Disturbance Handling**: If external disturbances (e.g., noise or earthquakes) are significant, prioritize robustness by increasing weights on critical states in Q.
  - **Simulation-Based Testing**: Use simulations to evaluate performance under various scenarios before deploying the controller on real hardware[1:1].
  - **Diagonal vs. Full Matrices**: Start with diagonal matrices for simplicity. If interactions between states are significant, consider off-diagonal terms in Q.

Example Tuning Process

1. Start with initial values:

```
Q = diag([10, 1, 100, 10])  # Penalize pole angle heavily
R = 0.01                    # Allow aggressive control
```

2. Simulate and observe:
   - If pole angle stabilizes but cart moves excessively, increase weight on cart position in Q.
   - If control inputs are too large, increase R.
3. Iterate until desired performance is achieved.

By adjusting these parameters iteratively or using automated techniques like sensitivity analysis or optimization algorithms, you can achieve a well-tuned LQR controller that balances stability, responsiveness, and energy efficiency[2:1] [3:1] [1:2].

⁂

1. http://cse.lab.imtlucca.it/~bemporad/publications/papers/ieeecsl_tuning_lqr.pdf
2. https://www.cs.unm.edu/~afaust/MLPC15_proceedings/MLPC15_paper_Marco.pdf
3. https://engineering.stackexchange.com/questions/26827/how-can-i-tune-the-parameters-of-a-lqr-controller