Linear Quadratic Regulator(LQR) for Discrete-Time System

Baboo J. Cui

June 10, 2019

Contents

LQR is related to optimal control problem, many problems can be formulated into it. It's one of the fundamental ways to achieve optimal control.

1 Problem Formulation and Direct Solution

Given a discrete LTI system:

$$x[k+1] = Ax[k] + Bu[k], x[0] = x_0$$

given a time horizon $k \in \{0, 1, \ldots, N\}$, where N may be infinity, find the optimal input sequence $U = \{u[0], u[1], \ldots, u[N-1]\}$ that minimize the **cost function**:

$$J(U) = \underbrace{\sum_{k=0}^{N-1} \left(x^T[k]Qx[k] + u^T[k]Ru[k]\right)}_{\text{running cost}} + \underbrace{x^T[N]Q_f x[N]}_{\text{terminal cost}}$$

- **state weight matrix**: $Q = Q^T \succeq 0$
- **control weight matrix**: $R = R^T \succ 0$, indicate that there is no free control input
- **final state weight matrix**: $Q_f = Q_f^T \succeq 0$
- **running cost**: for time horizon from 1 to $N-1$
- **terminal cost**: for time at N
- **infinite case**: N is infinity, in this case, $Q_f = 0$

Here is the direct solution:

- suppose value function at time $t+1$ is **quadratic**: $V_{t+1}(x) = x^T P_{t+1} x$, then value function at time t is also **quadratic**:

  $$V_t(x) = x^T P_t x, \forall x \in \mathbb{R}^n$$

- P_t can be obtained from P_{t+1} according to the **Riccati recursion**:

  $$P_t = Q + A^T P_{t+1} A - A^T P_{t+1} B(R + B^T P_{t+1} B)^{-1} B^T P_{t+1} A$$

- minimum value of J is $J^* = x_0^T P_0 x_0$
- optimal control at time t for the given state $x[t] = x$ is:

  $$u^*[t] = -\underbrace{(R + B^T P_{t+1} B)^{-1} B^T P_{t+1} A}_{\text{Kalman gain } K} x = -K_t x$$

  which is a **linear state feedback** control

Note that it can be **generalized** into time-varying cases by setting A, B, Q, R into is time varying counterparts $A(t), B(t), Q(t), R(t)$. Here $A(t)$ should be used instead of $A(t+1)$.

2 Examples of Implementations

Many problem can be formulated into LQR form, and here are some examples, though they look differently in format.

2.1 Energy Efficient Stabilization

Starting from $x[0] = x_0$, find control sequence U that minimize

$$J(U) = \alpha \sum_{k=0}^{n-1} ||u[k]||^2 + \beta \sum_{k=0}^{N} ||x[k]||^2$$

to make it into LQR form, choose:

- $Q = \beta I$
- $R = \alpha I$
- $Q_f = \beta I$

Note that:

- cost function try to make state trajectory stay close to zero and use the least control energy simultaneously
- α and β determine the emphasis, can be adjusted

Sometimes state cannot be obtained directly, and system **output** y can be used for evaluating running cost. Suppose output equation(Du part can be eliminate) is

$$y = Cx$$

in this case choose

$$Q = \beta C^T C$$

Here is the proof:

$$\begin{aligned}
\beta \sum_{k=0}^{N} ||y[k]||^2 &= \sum_{k=0}^{N} y^T[k]\beta I y[k] \\
&= \sum_{k=0}^{N} (Cx[k])^T \beta I C x[k] \\
&= \sum_{k=0}^{N} x^T[k] C^T \beta I C x[k] = \sum_{k=0}^{N} x^T[k](\beta C^T C)x[k]
\end{aligned}$$

this is a useful conclusion for reformation.

2.2 Minimum Energy Steering

Starting from $x[0] = x_0$, find control sequence U to use least energy to steer the final state to $x[N] = 0$ without lost generosity, the cost is:

$$J(U) = \sum_{k=0}^{N-1} ||u[k]||^2$$

to make it into LQR form, choose:

- $Q = 0$
- $R = I$
- $Q_f = \infty I$

By setting $Q_f \to \infty I$, there is a big penalty if $X[N]$ is far from 0. This won't lead to a analytic solution, but the **approximation** is good enough.

2.3 LQR for Tracking(VIP TOPIC)

Find efficient sequence U for the state to track a given **reference trajectory** x_k^*(may be time-varying):

$$J(U) = \alpha \sum_{k=0}^{N-1} ||u[k]||^2 + \beta \sum_{k=0}^{N} ||x[k] - x_k^*||^2$$

note that $||x[k] - x_k^*||^2$ is not homogeneous quadratic, it should be formulate. It can be expanded(refer math proof in last part) as:

$$\begin{aligned} ||x[k] - x_k^*||^2 &= x^T[k]x[k] - 2x^T[k]x_k^* + (x_k^*)^T x_k^* \\ &= \begin{bmatrix} x^T[k] & 1 \end{bmatrix} \begin{bmatrix} I & x_k^* \\ (x_k^*)^T & (x_k^*)^T x_k^* \end{bmatrix} \begin{bmatrix} x[k] \\ 1 \end{bmatrix} \quad \text{dimension augmentation} \end{aligned}$$

construct new state variable $\tilde{x}[k] = [x[k] \quad 1]^T$, new system dynamic will be:

$$\tilde{x}[k+1] = \begin{bmatrix} A & 0 \\ 0 & 1 \end{bmatrix} \tilde{x}[k] + \begin{bmatrix} B \\ 0 \end{bmatrix} u[k]$$

and the origin cost can be reformed as:

$$J(U) = \alpha \sum_{k=0}^{N-1} ||u[k]||^2 + \beta \sum_{k=0}^{N} \tilde{x}^T[k]\tilde{Q}_k\tilde{x}[k]$$

where

$$\tilde{Q}_k = \begin{bmatrix} I & x_k^* \\ (x_k^*)^T & (x_k^*)^T x_k^* \end{bmatrix}$$

clearly, the system is LTI and the cost function is LTV.

2.4 LQR for System with Perturbation

Suppose system is:

$$x[k+1] = Ax[k] + Bu[k] + w[k]$$

To achieve LQR formulation, new state vector is constructed as:

$$\tilde{x}[k] = [x^T[k] \quad z[k]] \quad \text{dimension augmentation}$$

recall that $x \in \mathbb{R}^n$, and $z[k] \in \mathbb{R}$, set $z[k] = z[k+1] = 1$, new system dynamic will be:

$$\tilde{x}[k+1] = \begin{bmatrix} A & w[k] \\ 0 & 1 \end{bmatrix} \tilde{x}[k] + \begin{bmatrix} B \\ 0 \end{bmatrix} u[k]$$

and system initial condition is $\tilde{x}[0] = [x[0] \quad 1]$. R will be the original one and $\tilde{Q}$ is:

$$\tilde{Q}_k = \begin{bmatrix} Q & 0 \\ 0 & 0 \end{bmatrix}$$

clearly, the system is LTV and the cost function is LTI. In this case, u is not changed, x is augmented.

3 Direct Approach to Solve LQR

LQR can directly be formulated as a least square problem, although this is not recommended, however it offers us very import conclusions.

3.1 Reconstruct the Problem

The system dynamics can be augmented to a big equation:

$$\underbrace{\begin{bmatrix} x[1] \\ x[2] \\ \vdots \\ x[N] \end{bmatrix}}_{\tilde{X}} = \underbrace{\begin{bmatrix} B & 0 & \cdots & \cdots \\ AB & B & 0 & \cdots \\ \vdots & \vdots & \ddots & \cdots \\ A^{N-1}B & A^{N-2}B & \cdots & B \end{bmatrix}}_{\tilde{G}} \underbrace{\begin{bmatrix} u[0] \\ u[1] \\ \vdots \\ u[N-1] \end{bmatrix}}_{\tilde{U}} + \underbrace{\begin{bmatrix} A \\ A^2 \\ \vdots \\ A^N \end{bmatrix}}_{\tilde{H}} x_0$$

Recall that $G\tilde{U}$ is the zero-state response and $\tilde{H}x_0$ is the zero-input response and the cost function can be rewrite as:

$$J(U) = \tilde{X}^T \underbrace{\begin{bmatrix} Q & & & \\ & Q & & \\ & & \ddots & \\ & & & Q_f \end{bmatrix}}_{\tilde{Q}} \tilde{X} + \tilde{U}^T \underbrace{\begin{bmatrix} R & & & \\ & R & & \\ & & \ddots & \\ & & & R \end{bmatrix}}_{\tilde{R}} \tilde{U}$$

And the problem can be written in a compact form as:

$$\begin{aligned} \min \quad & \tilde{X}^T\tilde{Q}\tilde{X} + \tilde{U}^T\tilde{R}\tilde{U} \\ \text{s.t.} \quad & \tilde{X} = \tilde{G}\tilde{U} + \tilde{H}x_0 \end{aligned}$$

3.2 Directly Solve the Reconstructed Problem

There are two ways to solve this problem:

- Lagrange multiplier approach
- plug the equality constraint into cost function to form an unconstrained optimization problem(here we use this way)

By substituting equality constraints into the cost function:

$$\begin{aligned} J(\tilde{U}) &= (\tilde{G}\tilde{U} + \tilde{H}x_0)^T\tilde{Q}(\tilde{G}\tilde{U} + \tilde{H}x_0) + \tilde{U}^T\tilde{R}\tilde{U} \\ &= \tilde{U}^T\tilde{G}^T\tilde{Q}\tilde{G}\tilde{U} + \tilde{U}^T\tilde{G}^T\tilde{Q}\tilde{H}x_0 + x_0\tilde{H}^T\tilde{Q}\tilde{G}\tilde{U} + x_0\tilde{H}^T\tilde{Q}\tilde{H}x_0 + \tilde{U}^T\tilde{R}\tilde{U} \\ &= \tilde{U}^T(\tilde{G}^T\tilde{Q}\tilde{G} + \tilde{R})\tilde{U} + 2\tilde{U}^T\tilde{G}^T\tilde{Q}\tilde{H}x_0 + x_0\tilde{H}^T\tilde{Q}\tilde{H}x_0 \end{aligned}$$

To find the U that minimize J, take the first order derivative:

$$\frac{dJ(\tilde{U})}{d\tilde{U}} = 2(\tilde{G}^T\tilde{Q}\tilde{G} + \tilde{R})\tilde{U} + 2\tilde{G}^T\tilde{Q}\tilde{H}x_0$$

By setting it to 0 can we find the optimal $\tilde{U}$ since it has only one solution:

$$\tilde{U}^* = -(\tilde{G}^T\tilde{Q}\tilde{G} + \tilde{R})^{-1}\tilde{G}^T\tilde{Q}\tilde{H}x_0$$

3.3 Limitations of Direct Approach

- matrix inversion is needed to find optimal control
- matrices dimension increases with time horizon N
- impractical for large N, impossible for infinite time horizon case
- sensitivity of solutions to numerical errors

3.4 Observations from Direct Approach

- easier to solve for shorter time horizon N
- $(N+1)$-horizon solution related to N-horizon solution, iterative solution could be feasible
- optimal control sequence has **linear feedback form**

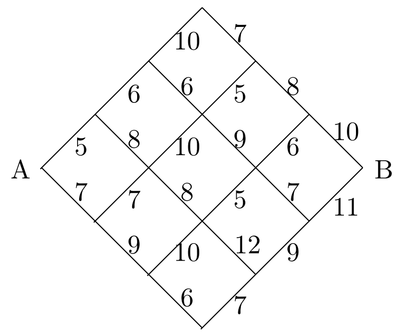
4 Dynamic Programming

4.1 Dynamic programming approach

- reuse results for smaller N to solve for large N case
- each iteration only need to deal with a problem of fixed size

4.2 Motivating Example

Start from point A, try to reach point B, each step only move right and cost labeled on each edge. How to find the least costly path from A to B?



This can be formulated as an optimal control problem, each node may be assigned by a coordinate, specifically:

$$A = (0, 0) \quad B = (3, 3)$$

state $x[k]$ with boundary condition: $x[0] = A$, and $x[6] = B$. Control input is $u[k] = \pm 1$, and system dynamics is

$$x[k+1] = \begin{cases} x[k] + (0, 1) & u[k] = 1 \\ x[k] + (1, 0) & u[k] = -1 \end{cases}$$

Cost to be minimized:

$$\sum_{k=0}^{5} w(x[k], u[k])$$

where w is the edge weight(or edge cost).

4.3 Direct Solution

Enumerate all possible legal from A to B and compare their costs to find the least cost.

- for ℓ-by-ℓ grid, the total number of legal paths is

$$\frac{(2\ell)!}{(\ell!)^2}$$

- grows extremely fast as problem size ℓ increases, beyond exponential bound
- solution impractical for large ℓ
- solution **impossible** when input is **infinite**

4.4 Value Function(VIP)

At any point(state in a more general case) z, the **value function**(optimal cost-to-go) $V(z)$ is the least possible cost to reach terminal(B in motivating example) from z. Note that:

- $V(z)$ can be obtained by solve **shorter** time horizon problems
- original problem can be formulated as to find $V(A)$

So optimal control problem can be transformed into value function problem.

4.5 Principle of Optimality(VIP)

If a least-cost path from A to B is

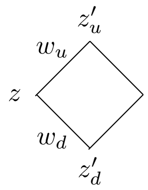
$$x_0^* = A \to x_1^* \to x_2^* \to \cdots \to x_6^* = B,$$

then any truncation of it at the end:

$$x_t^* \to x_{t+1}^* \to \cdots \to x_6^* = B$$

is also a least-cost path from x_t^* to B. As a result:

$$
\begin{aligned}
V(z) &= \min \{w_u + V(z_u'), w_d + V(z_d')\} \\
&= \min_{u \in \pm 1} [w(z, u) + V(z')]
\end{aligned}
$$



- $V(z)$: minimum cost-to-go from current position
- $w(z, u)$: running cost of current step
- $V(z')$: cost-to-go from next state position

And the motivating problem can be solved by **iteration** from final to initial point.

4.6 Advantages of Dynamic Programming

- only need to compute ℓ^2 value functions(P-problem)
- no need to enumerate $\frac{(2\ell)!}{(\ell!)^2}$ paths(avoid NP problem)
- solve an optimization problem of fixed size in each iteration
- even if starting from a different initial position (e.g. due to perturbation), there is no need for re-computation(a family of problems can be solved)

5 Solve LQR Problem by Dynamic Programming

Recall LQR problem formulation: a discrete-time LTI system

$$x[k+1] = Ax[k] + Bu[k], x[0] = x_0$$

Given a time horizon $k \in \{0, 1, ..., N\}$, find the optimal input sequence $U = \{u[0], ..., u[N-1]\}$ that minimizes the cost function

$$J(U) = \sum_{k=0}^{N-1} (x^T[k]Qx[k] + u^T[k]Ru[k]) + x^T[N]Q_f x[N]$$

5.1 Value Function of LQR Problem

The value function at any time $t \in \{0, 1, ..., N\}$ and state $x \in \mathbb{R}^n$ is

$$V_t(x) = \min_{u[t], ..., u[N-1]} \sum_{k=t}^{N-1} (x^T[k]Qx[k] + u^T[k]Ru[k]) + x^T[N]Q_f x[N]$$

with the initial condition $x[t] = x$, namely, **cost-to-go** $V_t(x)$ is optimal cost of the LQR problem over the time horizon $\{t, t+1, ..., N\}$, starting from $x[t] = x$(arbitrary). Note that:

$$V_0(x) = J(U)$$

5.2 Solution of LQR Problem via Value Functions

Preview of results:

- the value function at the final time is quadratic: $V_N(x) = x^T Q_f x$
- the value function at any time t is also **quadratic**: $V_t(x) = x^T P_t x$ for **some** $P_t \succeq 0$,(the proof is in extra part)
- P_t can be obtained from P_{t+1} **recursively**

Solution algorithm(VIP):

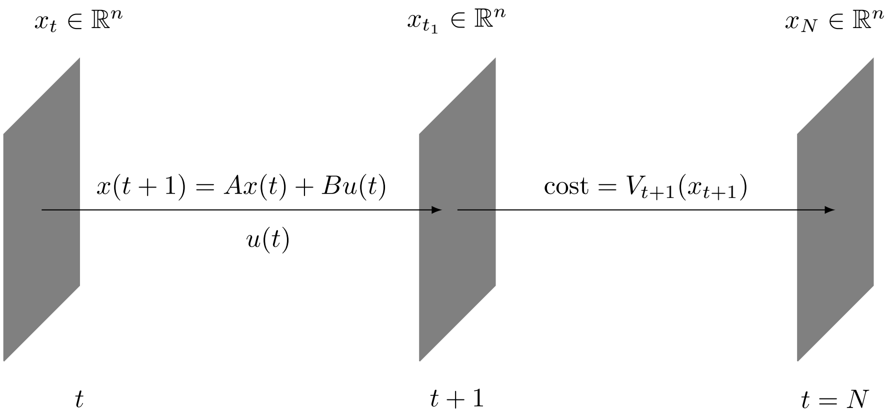
1. start from $P_N = Q_f$ at time $t = N$
2. for $t = \{N-1, N-2, \ldots, 0\}$, compute P_t from P_{t+1} by the above recursion
3. recover optimal control sequence from value functions

5.3 Recursion of Value Functions(VIP)

Hamilton-Jacobi-Bellman(HJB) equation:

$$\begin{aligned} V_t(x) &= \min_{u[t]=v} [x^T Q x + v^T R v + V_{t+1}(Ax + Bv)] \\ &= x^T Q x + \min_{u[t]=v} [v^T R v + V_{t+1}(Ax + Bv)] \end{aligned}$$

Optimality principle: for optimal case, cost-to-go from next state $x[t+1]$, i.e. $V_{t+1}(x[t+1])$, should also be optimal.



And here is the process:

1. $t = N$ **case**: value function is quadratic and can be directly found as:

$$V_N(x) = x^T P_N x = x^T Q_f x, \forall x \in \mathbb{R}^n, \text{where } P_N = Q_f$$

2. $t = N-1$ **case**:

$$\begin{aligned} V_{N-1}(x) &= x^T Q x + \min_v [v^T R v + V_N(Ax + Bv)] \\ &= x^T Q x + \min_v [v^T R v + (Ax + Bv)^T P_N (Ax + Bv)] \end{aligned}$$

First to prove that optimal controller has linear state feed back form:

$$
\begin{aligned}
V_{N-1}(x) &= x^TQx + \min_v [v^TRv + V_N(Ax+Bv)] \\
&= x^TQx + \min_v [v^TRv + (Ax+Bv)^TP_N(Ax+Bv)] \quad \text{quadratic expansion} \\
&= x^TQx + \min_v [v^TRv + x^TA^TP_NAx + 2v^TB^TP_NAx + v^TB^TP_NBv] \quad \text{move out term} \\
&= x^TQx + x^TA^TP_NAx + \min_v [v^TRv + 2v^TB^TP_NAx + v^TB^TP_NBv] \quad \text{combine like terms} \\
&= x^T(Q + A^TP_NA)x + \min_v [2v^TB^TP_NAx + v^T(R + B^TP_NB)v]
\end{aligned}
$$

o find the optimal $u* = v$, take derivative of the terms in min function to v and set to 0:

$$
\begin{aligned}
\frac{\partial}{\partial v}\left(2v^TB^TP_NAx + v^T(R + B^TP_NB)v\right) = 0 \\
2B^TP_NAx + 2(R + B^TP_NB)v^* = 0 \\
v^* = -(R + B^TP_NB)^{-1}B^TP_NAx \quad \text{VIP!}
\end{aligned}
$$

so

$$(v^*)^T = -x^TA^TP_NB(R + B^TP_NB)^{-1}$$

Then to prove that value function at any given time is in quadratic form:

$$
\begin{aligned}
V_{N-1}(x) =& x^TQx + \min_v [v^TRv + V_N(Ax+Bv)] \\
=& x^T(Q + A^TP_NA)x + \min_v [2v^TB^TP_NAx + v^T(R + B^TP_NB)v] \quad \text{substitute } v^* \\
=& x^T(Q + A^TP_NA)x - 2x^TA^TP_NB(R + B^TP_NB)^{-1}B^TP_NAx + \\
& x^TA^TP_NB\underbrace{(R + B^TP_NB)^{-1}(R + B^TP_NB)}_{\text{cancel}}(R + B^TP_NB)^{-1}B^TP_NAx \\
=& x^T\left(Q + A^TP_NA - \underbrace{2A^TP_NB(R + B^TP_NB)^{-1}B^TP_NA + A^TP_NB(R + B^TP_NB)^{-1}B^TP_NA}_{\text{like term}}\right)x \\
=& x^T\left(Q + A^TP_NA - A^TP_NB(R + B^TP_NB)^{-1}B^TP_NA\right)x \quad \text{VIP!}
\end{aligned}
$$

this indicate that:

$$P_t = Q + A^TP_{t+1}A - A^TP_{t+1}B(R + B^TP_{t+1}B)^{-1}B^TP_{t+1}A$$

6 LQR Algorithm and Properties(Detailed)

6.1 Algorithm Summary

1. set $P_N = Q_f$
2. **for** $t = N-1, N-2, ..., 0$, compute the value functions backward in time:

$$P_t = Q + A^TP_{t+1}A - A^TP_{t+1}B(R + B^TP_{t+1}B)^{-1}B^TP_{t+1}A$$

3. return $V_0(x_0)$ as the optimal cost(it can be get before optimal input sequences!)

4. set $x^*[0] = x_0$
5. for $t = 0, 1, ..., N-1$, recover the optimal control and state trajectory forward in time:

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

and

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

6. return u^* and x^* as the optimal control and state sequences

6.2 Remarks

- value function at any time is quadratic (easy numeric representation)
- optimal control strategy is of the state feedback form (though with time-varying gains)
- yield the optimal solutions for **all initial conditions** x_0 and **all initial times** $t_0 \in \{0, 1, ..., N\}$ simultaneously
- easily extended to time-varying dynamics and costs cases

6.3 Steady State Optimal Control

After sufficient number of iterations, if P and K converges, then

- the value function converges to the solution of matrix equation:

$$P_{ss} = Q + A^T P_{ss} A - A^T P_{ss} B (R + B^T P_{ss} B)^{-1} B^T P_{ss} A$$

- The Kalman gain converges to

$$K_{ss} = (R + B^T P_{ss} B)^{-1} B^T P_{ss} A$$

here the subscript ss represents **steady state**.

6.4 Convergence of Riccati Recursion

If (A, B) is stabilizable, then Riccati recursion starting from any P_N:

$$P_t = Q + A^T P_{t+1} A - A^T P_{t+1} B (R + B^T P_{t+1} B)^{-1} B^T P_{t+1} A$$

will converge(in **exponential** order, very fast) to a solution P_{ss} of the **Algebraic Riccati Equation(ARE)**

$$P_{ss} = Q + A^T P_{ss} A - A^T P_{ss} B (R + B^T P_{ss} B)^{-1} B^T P_{ss} A$$

If further $Q = C^T C$ for some C such that (C, A) is detectable, then the ARE has a unique positive semi-definite P_{ss}. Also, in this case by applying the steady-state optimal control with gain

$$K_{ss} = (R + B^T P_{ss} B)^{-1} B^T P_{ss} A$$

the closed-loop system

$$A_{cl} = A - BK_{ss}$$

is stable, which indicate that optimal is sufficient but not necessary for stable.

6.5 Infinite Horizon LQR Problem

In infinite time horizon case, the cost function will be:

$$J(U) = \sum_{k=0}^{\infty} \left(x^T[k]Qx[k] + u^T[k]Ru[k] \right)$$

Note that

- problem invariant to time-shift: same problem faced again and again
- thus, value function is independent of time, with Bellman equation:

$$V(x) = x^T Qx + \min_v \left[v^T Rv + V(Ax + Bv) \right]$$

- infinite value function possible

If (A, B) is stabilizable and (C, A) is detectable where $Q = C^T C$, then the value function $V(x)$ of the infinite horizon problem is

$$V(x) = x^T P_{ss} x$$

where P_{ss} is the unique positive semi-definite solution to the discrete-time ARE and the optimal control is stationary

$$u^*(t) = -K_{ss} x^*(t)$$

7 Example of LQR Implementation

7.1 Direct Implementation Example

Given system dynamic, initial condition and output equation:

$$\begin{aligned} x[k+1] &= \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} x[k] + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u[k], \quad x[0] = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \\ y[k] &= \begin{bmatrix} 1 & 0 \end{bmatrix} x[k] \end{aligned}$$

cost function to be minimized is:

$$J(U) = \sum_{k=0}^{N-1} \|u[k]\|^2 + \rho \sum_{k=0}^{N} \|y[k]\|^2$$

To find solution for time horizon $N = 20$, choose weight matrices:

- state weight matrix: $Q = Q_f = \rho C^T C$
- control weight matrix: $R = 1$
- optimal control sequence has linear state feedback form

The code is as following:

8 Extra

This part offers additional information related to this topic.

8.1 Matlab Functions

- lqrd(): for discrete-time system
- lqr(): for continuous-time system

8.2 Quadratic Expansion

The general length of a vector $x \in \mathbb{R}^n$ is also called the L_2 norm. It is defined as:

$$||x||^2 = x^T x = \sum_{i=1}^{n} x_i^2, \text{ where } x_i \in \mathbb{R}$$

if another vector $y \in \mathbb{R}^n$, the norm of the difference is:

$$\begin{aligned}
||x - y||^2 &= ||y - x||^2 \quad \text{identity property} \\
&= (x - y)^T (x - y) \quad \text{definition} \\
&= x^T x - x^T y - y^T x + y^T y \quad \text{distributive property} \\
&= ||x||^2 - 2x^T y + ||y||^2
\end{aligned}$$

recall that:

$$x^T y = y^T x \quad \text{property of inner product}$$

8.3 Matrix Calculus

Recall some important matrix calculus properties here:

- quadratic derivative:
$$\frac{dx^T Ax}{dx} = (A + A^T)x$$
- linear differentiation:
$$\frac{dx^T A}{dx} = A$$
- inverse and transpose:
$$(A^{-1})^T = (A^T)^{-1}$$