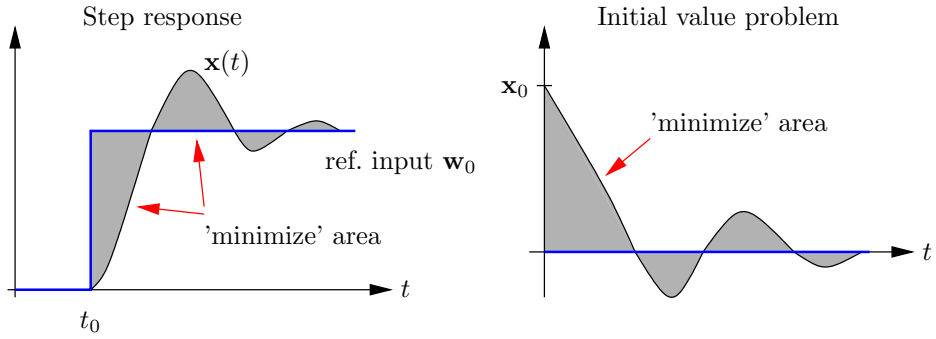
## 3.6 Linear Quadratic Regulator (LQR)

The idea is to introduce and optimize a performance index as depicted in the following figures




For a good controller performance, one would demand for a fast response and little overshooting, hence for minimizing the shaded areas. Meanwhile, one would also want to save the energy used by the controller in implementing control actions. The performance index for the LQR controller is introduced

- for discrete LTI systems:

$$J(\mathbf{x}, \mathbf{u}) = \sum_{k=0}^{N-1} \left( \mathbf{x}[k]^T \mathbf{Q} \mathbf{x}[k] + \mathbf{u}[k]^T \mathbf{R} \mathbf{u}[k] \right) + \mathbf{x}[k]^T \mathbf{Q}_f \mathbf{x}[k] \tag{3.58}$$

- for continuous LTI systems:

$$J(\mathbf{x}, \mathbf{u}) = \int_0^T \left( \mathbf{x}^\top(t) \mathbf{Q} \mathbf{x}(t) + \mathbf{u}^\top(t) \mathbf{R} \mathbf{u}(t) \right) + \mathbf{x}(T)^T \mathbf{Q}_f \mathbf{x}(T) \, \mathrm{d}t \tag{3.59}$$

where $\mathbf{Q}, \mathbf{Q}_f$ are symmetric, positive semidefinite $(n \times n)$ matrices and $\mathbf{R}$ is a symmetric, positive definite $(p \times p)$ matrix. The matrices $\mathbf{Q}, \mathbf{Q}_f$ and $\mathbf{R}$ can be regarded as tuning parameters in order to meet design requirements. While $\mathbf{Q}$ penalizes slow responses and overshoots, $\mathbf{R}$ adds a penalization to steering actuation, and $\mathbf{Q}_f$ puts penalization on the end state of the *horizon*, which consists of steps $[0, N]$ for discrete case or duration $[0, T]$ for the continuous case.

Typical LQR design employs *infinite horizon*, i.e. we replace $N$ by $\infty$ and drop the terms related to $\mathbf{Q}_f$ in (3.58) and (3.59).

### 3.6.1 Solving discrete LQR problems

Solutions to discrete LQR problems are derived using the *dynamic programming principle*, the optimal solution would be obtained recursively backward from the last time step. We present hereby the results for the cases with finite horizon and infinite horizon.

**Discrete LQR with finite horizon** The *cost-to-go* $V_t(\mathbf{z})$, which is the optimal cost depending on the state value $\mathbf{z}$, counted from the *time-to-go* $t$ until the end of the horizon, is $V_t(\mathbf{z}) = \mathbf{z}^T \mathbf{P}_t \mathbf{z}$ with $\mathbf{P}_t$ symmetric, positive definite, obtained using the following algorithm:

1. set $\mathbf{P}_N = \mathbf{Q}_f$
2. for $t = N, \ldots, 1$, compute

$$\mathbf{P}_{t-1} = \mathbf{Q} + \mathbf{A}^T \mathbf{P}_t \mathbf{A} - \mathbf{A}^T \mathbf{P}_t \mathbf{B} \left( \mathbf{R} + \mathbf{B}^T \mathbf{P}_t \mathbf{B} \right)^{-1} \mathbf{B}^T \mathbf{P}_t \mathbf{A} \tag{3.60}$$

3. for $t = 0, \ldots, N-1$, define $\mathbf{K}_t = \left(\mathbf{R} + \mathbf{B}^T\mathbf{P}_{t+1}\mathbf{B}\right)^{-1}\mathbf{B}^T\mathbf{P}_{t+1}\mathbf{A}$. The optimal input for time step $t$ is $\mathbf{u}[t] = -\mathbf{K}_t\mathbf{x}[t]$.

Note that in this linear state feedback controller, the gain $\mathbf{K}_t$ changes over time.

**Discrete LQR with infinite horizon** The performance function to optimize:

$$J(\mathbf{x}, \mathbf{u}) = \sum_{k=0}^{\infty} \left(\mathbf{x}[k]^T\mathbf{Q}\mathbf{x}[k] + \mathbf{u}[k]^T\mathbf{R}\mathbf{u}[k]\right) \tag{3.61}$$

The cost-to-go $V(\mathbf{z})$ does not depend on time, it is associated with current state as $V(\mathbf{z}) = \mathbf{z}^T\mathbf{P}\mathbf{z}$, where $\mathbf{P}$ is a symmetric, positive definite solution of the following matrix equation:

$$\mathbf{P} = \mathbf{Q} + \mathbf{A}^T\mathbf{P}\mathbf{A} - \mathbf{A}^T\mathbf{P}\mathbf{B}\left(\mathbf{R} + \mathbf{B}^T\mathbf{P}\mathbf{B}\right)^{-1}\mathbf{B}^T\mathbf{P}\mathbf{A} \tag{3.62}$$

and the optimal input is a *constant* state feedback controller $\mathbf{u}[k] = -\mathbf{K}\mathbf{x}[k]$ with:

$$\mathbf{K} = \left(\mathbf{R} + \mathbf{B}^T\mathbf{P}\mathbf{B}\right)^{-1}\mathbf{B}^T\mathbf{P}\mathbf{A} \tag{3.63}$$

The equation (3.62) is called the (discrete) Algebraic Riccati Equation (ARE). In MATLAB, we can use the command `dlqr` to solve the discrete ARE and obtain the linear state feedback controller.

### 3.6.2 Solving continuous LQR problems

**Continuous LQR with finite horizon** Summary of the continous LQR solution:

1. set $\mathbf{P}_T = \mathbf{Q}_f$
2. for $t \in [0, T)$, solve the following Riccati differential equation backward in time to get $\mathbf{P}_t$:

$$-\dot{\mathbf{P}}_t = \mathbf{A}^T\mathbf{P}_t + \mathbf{P}_t\mathbf{A} - \mathbf{P}_t\mathbf{B}\mathbf{R}^{-1}\mathbf{B}^T\mathbf{P}_t + \mathbf{Q} \tag{3.64}$$

3. for $t \in [0, T)$, the optimal input is $\mathbf{u}(t) = -\mathbf{K}(t)\mathbf{x}(t)$, with $\mathbf{K}(t) = \mathbf{R}^{-1}\mathbf{B}^T\mathbf{P}_t$.

**Continuous LQR with infinite horizon** The performance function to optimize:

$$J(\mathbf{x}, \mathbf{u}) = \int_0^{\infty} \left(\mathbf{x}^\top(t)\mathbf{Q}\mathbf{x}(t) + \mathbf{u}^\top(t)\mathbf{R}\mathbf{u}(t)\right)\, \mathrm{d}t \tag{3.65}$$

Similar to discrete LQR with infinite horizon, for continuous LTI systems, we can design the optimal controller $\mathbf{u}(t) = -\mathbf{K}(t)\mathbf{x}(t)$ with

$$\mathbf{K} = \mathbf{R}^{-1}\mathbf{B}^T\mathbf{P} \tag{3.66}$$

where $\mathbf{P}$ is the solution of the following (continuous) ARE:

$$\mathbf{A}^T\mathbf{P} + \mathbf{P}\mathbf{A} - \mathbf{P}\mathbf{B}\mathbf{R}^{-1}\mathbf{B}^T\mathbf{P} + \mathbf{Q} = \mathbf{0} \tag{3.67}$$

In MATLAB, we can use the command `lqr` to solve the continuous ARE and obtain the linear state feedback controller.

It is known that if the system is controllable and the pair $(\mathbf{A}, \mathbf{C})$ is *observable* (concept to be introduced in the next chapter) with $\mathbf{C}$ is defined as $\mathbf{Q} = \mathbf{C}^T\mathbf{C}$, then there exist a unique positive definite matrix $\mathbf{P}$ that satisfies the ARE (3.67). Similar result holds for the discrete LQR case.

### 3.6.3 Choice of $Q$ and $R$ Matrices

In this section, some rough ideas on the choices of the **Q** and **R** matrices shall be given.

- As the choice of the **Q** and **R** matrices is crucial for the result, the LQR concept should be regarded more as a mathematical recipe for carrying out the controller design rather than as a self-contained procedure, which comes up with the 'optimal' controller. In practice one would choose certain matrices **Q** and **R**, then compute the controller based on these matrices and compare simulations to given specifications. Eventually, the whole design process has to be repeated with different **Q** and **R** matrices to end up at the desired controller behavior after some iterations.
- The matrix **Q** can be chosen such that $\mathbf{Q} = \mathbf{C}^T\mathbf{C}$, where $\mathbf{y} = \mathbf{C}\mathbf{x}$, so that the objective function includes quadratic terms of outputs and inputs. **R** could be a diagonal matrix.
- Or one can start with diagonal matrices for both **Q** and **R**, and choose

$$\begin{aligned} q_{i,i} &= \frac{1}{\text{Maximum acceptable value for } x_i^2} & i = 1, \ldots, n \\ r_{j,j} &= \frac{1}{\text{Maximum acceptable value for } u_j^2} & j = 1, \ldots, p \end{aligned}$$