# Riccati Recursion Proof

Mohammed Dawood

October 2025

## 1 Introduction

We are interested in finding a closed form solution of the following problem:

$$J = \sum_{k=0}^{N-1} \left( \tfrac{1}{2} x_k^T Q_k x_k + \tfrac{1}{2} u_k^T R_k u_k + q_k^T x_k + r_k^T u_k \right) + \tfrac{1}{2} x_N^T Q_f x_N + q_f^T x_N.$$

subject to linear dynamics:

$$x_{k+1} = A x_k + B u_k.$$

There are two approaches to find this closed-form solution:

1. Through the Lagrangian
2. Through Dynamic Programming

## 2 The Lagrangian Approach

In a general optimization problem, the Lagrangian is the linear combination objective function and the constraints, each weighted by a corresponding Lagrange multiplier. This helps us to turn the problem from a constrained optimization into an unconstrained optimization problem. By taking the gradient of the Lagrangian and setting it equal to 0, we have found our equations that need to be satisfied to determine Another method, which is more intuitive, is to write the constraint function in terms of one variable, then plugging it into our function that we are interested in minimizing. However, this approach can be cumbersome when dealing with a lot of constraint functions or constraint functions that cannot be easily solved for one variable.

## 2.1 General Lagrangian

Assuming the following problem:

$$\begin{aligned} \min_{x} \quad & f(x) \\ \text{subject to} \quad & g_i(x) = 0, \quad i = 1, 2, \ldots, m, \end{aligned}$$

the Lagrangian is defined as

$$\mathcal{L}(x, \lambda) = f(x) + \sum_{i=1}^{m} \lambda_i g_i(x),$$

where $\lambda_i$ are the Lagrange multipliers

## 2.2 Using Lagrangian to Solve LQR

Now, we will take this definition of the Lagrangian and apply it to our LQR problem. This yields the following.

$$\mathcal{L} = J + G$$

where:

$$J = \sum_{k=0}^{N-1} \left( \tfrac{1}{2} x_k^T Q_k x_k + \tfrac{1}{2} u_k^T R_k u_k + q_k^T x_k + r_k^T u_k \right) + \tfrac{1}{2} x_N^T Q_f x_N + q_f^T x_N.$$

$$G = \sum_{k=0}^{N-1} \lambda_{k+1}^T \left( A x_k + B u_k - x_{k+1} \right)$$

**Note:** we use $\lambda_{k+1}$ since it corresponds to $x_{k+1}$.

This is all equivalent to:

$$\begin{aligned} \mathcal{L} = & \sum_{k=0}^{N-1} \left( \tfrac{1}{2} x_k^T Q_k x_k + \tfrac{1}{2} u_k^T R_k u_k + q_k^T x_k + r_k^T u_k \right) \\ & + \sum_{k=0}^{N-1} \lambda_{k+1}^T \left( A x_k + B u_k - x_{k+1} \right) \\ & + \tfrac{1}{2} x_N^T Q_f x_N + q_f^T x_N \end{aligned}$$

which simplifies to:

$$\begin{aligned} \mathcal{L} = & \sum_{k=0}^{N-1} \left( \tfrac{1}{2} x_k^T Q_k x_k + \tfrac{1}{2} u_k^T R_k u_k + q_k^T x_k + r_k^T u_k + \lambda_{k+1}^T \left( A x_k + B u_k - x_{k+1} \right) \right) \\ & + \tfrac{1}{2} x_N^T Q_f x_N + q_f^T x_N \end{aligned}$$

Clearly, this long equation can seem cumbersome. Thus, in order to further simplify it, we introduce the Hamiltonian.

$$\mathcal{H}_k = \frac{1}{2}x_k^T Q_k x_k + \frac{1}{2}u_k^T R_k u_k + q_k^T x_k + r_k^T u_k + \lambda_{k+1}^T (Ax_k + Bu_k)$$

This simplifies the Lagrangian to:

$$\mathcal{L} = \sum_{k=0}^{N-1} \Big( \mathcal{H}_k - \lambda_{k+1}^T \left(x_{k+1}\right) \Big) + \tfrac{1}{2}x_N^T Q_f x_N + q_f^T x_N$$

Now, to solve the optimization problem, we need to take the gradient of $\mathcal{L}$ w.r.t. $x_k$, $\lambda_{k+1}$, and $u_k$

**Equation 1: Gradient w.r.t.** $x_k$

$$\frac{\partial \mathcal{L}}{\partial x_k} = \frac{\partial \mathcal{H}_k}{\partial x_k} - \lambda_k = 0$$

**Note:** There is an important contribution when taking the derivative w.r.t $x_k$ which comes from the previous term in the Lagrangian: $-\lambda_k^T x_k \implies \frac{\partial(-\lambda_k^T x_k)}{\partial x_k} = -\lambda_k$ Even though $\lambda_k$ is "from the next time step" in the sum, it multiplies $x_k$ in the previous term of the Lagrangian. Thus, the term $\lambda_k$ appears.

Knowing that

$$\frac{\partial \mathcal{H}_k}{\partial x_k} = Q_k x_k + q_k + A^T \lambda_{k+1}$$

Finally yielding:

$$\lambda_k = Q_k x_k + q_k + A^T \lambda_{k+1} \tag{1}$$

**Equation 2: Gradient w.r.t.** $u_k$

$$\frac{\partial \mathcal{L}}{\partial u_k} = \frac{\partial \mathcal{H}_k}{\partial u_k} = 0$$

Knowing that

$$\frac{\partial \mathcal{H}_k}{\partial u_k} = R_k u_k + r_k + B^T \lambda_{k+1}$$

Finally yielding:

$$Ru_k + r_k + B^T \lambda_{k+1} = 0 \tag{2}$$

**Equation 3: Gradient w.r.t. $\lambda_{k+1}$**

$$\frac{\partial \mathcal{L}}{\partial \lambda_{k+1}} = \frac{\partial \mathcal{H}_k}{\partial \lambda_{k+1}} - x_{k+1} = 0$$

Knowing that

$$\frac{\partial \mathcal{H}_k}{\partial \lambda_{k+1}} = Ax_k + Bu_k$$

Finally yielding:

$$x_{k+1} = Ax_k + Bu_k \tag{3}$$

**Equation 4: Expressing $\lambda_k$ as an affine function**

From the structure of Equation (1), we know that $\lambda_k$ is a function of a coefficient multiplied by $x_k$ and a linear term. Moreover, there is a recursive term that will disappear to be a function of a coefficient multiplied by $x_{k+1}$ and a linear term. Let's assume this coefficient is $P_k$ and the linear term is $s_k$. We get

$$\lambda_k = P_k x_k + s_k \tag{4}$$

building on that we get the recursive term to be

$$\lambda_{k+1} = P_{k+1} x_{k+1} + s_{k+1} \tag{5}$$

Equation (5) could be simplified by using Equation (3) to get

$$\lambda_{k+1} = P_{k+1}(Ax_k + Bu_k) + s_{k+1} \tag{6}$$

**Finding Feedback Controls $u_k = -K_k x_k - d_k$**

Now, with the 6 equations we have derived, we are ready to tackle the first part of the problem.
Plugging Equation (6) into Equation (2)

$$R_k u_k + r_k + B^T(P_{k+1}(Ax_k + Bu_k) + s_{k+1}) = 0$$

$$R_k u_k + r_k + B^T P_{k+1} A x_k + B^T P_{k+1} B u_k + B^T s_{k+1} = 0$$

$$R_k u_k + B^T P_{k+1} B u_k = -r_k - B^T P_{k+1} A x_k - B^T s_{k+1}$$

$$(R_k + B^T P_{k+1}) u_k = -(B^T P_{k+1} A) x_k - (r_k + B^T s_{k+1})$$

$$u_k = -(R_k + B^T P_{k+1})^{-1}(B^T P_{k+1} A) x_k - (R_k + B^T P_{k+1})^{-1}(r_k + B^T s_{k+1})$$

which simplifies to

$$u_k = -K_k x_k - d_k \tag{7}$$

where

$$K_k = (R_k + B^T P_{k+1})^{-1}(B^T P_{k+1} A)$$

and

$$d_k = (R_k + B^T P_{k+1})^{-1}(r_k + B^T s_{k+1})$$

**Finding The Remaining Unknown Terms $P_k$ and $s_k$**

We found the equation for our feedback controls in terms of the known terms $R_k$, $B^T$, $A$, and $r_k$. However, we still have two unknowns $P_k$ and $s_k$. To find them, we need to start by plugging in Equation (6) into Equation (1)

$$\lambda_k = Q_k x_k + q_k + A^T(P_{k+1}(Ax_k + Bu_k) + s_{k+1})$$
$$\lambda_k = Q_k x_k + q_k + A^T P_{k+1} A x_k + A^T P_{k+1} B u_k + A^T s_{k+1}$$

Using Equation (4)

$$P_k x_k + s_k = Q_k x_k + q_k + A^T P_{k+1} A x_k + A^T P_{k+1} B u_k + A^T s_{k+1}$$

Using Equation (7)

$$P_k x_k + s_k = Q_k x_k + q_k + A^T P_{k+1} A x_k + A^T P_{k+1} B(-K_k x_k - d_k) + A^T s_{k+1}$$
$$P_k x_k + s_k = Q_k x_k + A^T P_{k+1} A x_k - A^T P_{k+1} B K_k x_k + q_k - A^T P_{k+1} B d_k + A^T s_{k+1}$$
$$(P_k)x_k + (s_k) = (Q_k + A^T P_{k+1} A - A^T P_{k+1} B K_k)x_K + (q_k - A^T P_{k+1} B d_k + A^T s_{k+1})$$

This yields

$$P_k = Q_k + A^T P_{k+1} A - A^T P_{k+1} B K_k$$
$$s_k = q_k - A^T P_{k+1} B d_k + A^T s_{k+1}$$

Simplifying to

$$P_k = Q_k + A^T P_{k+1}(A - BK_k) \tag{8}$$
$$s_k = q_k - A^T(P_{k+1} B d_k - s_{k+1}) \tag{9}$$

**Final Step Boundary Conditions**

So far, we've taken the Lagrangian with respect w.r.t. $\lambda_{k+1}$, $x_k$, and $u_k$. However, we still have one more variable in the Lagrangian we haven't take the derivative with respect to. In this case it's $x_n$. This gives us the boundary conditions at N

$$\frac{\partial \mathcal{L}}{\partial x_n} = -\lambda_n + Q_f x_n + q_f = 0$$
$$\lambda_n = Q_f x_n + q_f$$

Recall from Equation (4)

$$\lambda_k = P_k x_k + s_k$$

Replacing k=n we get

$$\lambda_n = P_n x_n + s_n = Q_f x_n + q_f$$

Thus the boundary conditions at N are:

1. $P_n = Q_f$
2. $s_n = q_f$

### 2.3 Final Solution

From the derivation above, the optimal control law for the discrete-time LQR problem is:

$$u_k = -K_k x_k - d_k \tag{A}$$

where the feedback gain $K_k$ and the feedforward term $d_k$ are given by:

$$K_k = (R_k + B^T P_{k+1} B)^{-1}(B^T P_{k+1} A), \quad d_k = (R_k + B^T P_{k+1} B)^{-1}(r_k + B^T s_{k+1})$$

The matrices $P_k$ and vectors $s_k$ are computed backward in time starting from the terminal boundary conditions at $k = N$:

1. $P_N = Q_f$
2. $s_N = q_f$

The recursion equations for $P_k$ and $s_k$ are:

$$P_k = Q_k + A^T P_{k+1}(A - BK_k), \quad s_k = q_k - A^T(P_{k+1} B d_k - s_{k+1})$$

## 3 Dynamic Programming Approach

Another approach to solve the LQR problem is through dynamic programming. First, it is important to understand the reason behind why dynamic programming works: Bellman's Principle of Optimality

### 3.1 Bellman's Principle of Optimality

The principle states the following:

> "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." — Richard Bellman, *Dynamic Programming* (1957)

To restate Bellman's Principle of Optimality in the context of optimal control: for a given problem, there exists an optimal control trajectory that drives the system from the initial state to the final desired state. Now, if we consider a new problem in which the initial state is any point along this optimal trajectory, the remaining segment of the original trajectory, from that point to the final state, constitutes the optimal control and state trajectory for the new problem.

This has great implications in simplifying our original LQR problem. By knowing what our final state and the terminal cost, we can compute the optimal cost-to-go function and control law for the previous time. We can then repeat this process all the way back to our initial state to find the complete optimal control trajectory.

### 3.2 Cost To Go Function

Now, it is important to introduce the cost-to-go function that will be central to deriving the Riccati Recursion.

$$V_k(x_k) = \min_{u_k} \quad \frac{1}{2}x_k^T Q_k x_k + \frac{1}{2}u_k^T R_k u_k + q_k^T x_k + r_k^T u_k + V_{k+1}(x_{k+1}) \tag{10}$$

**Note:** The cost-to-go function is only a function of $x_k$ and not $u_k$ since we assume optimal controls in $u$. This formulation allows us to determine the state trajectory that yields the minimum possible cumulative cost. Another way to think of it is that our cost-to-go function is the minimization of a certain $f = f(x_k, u_k)$. However, after minimizing w.r.t. $u_k$, we will be left with a function in terms of $x_k$ and $u_k$. Then, when finding the optimal control, we know that $u_k$=$f(x_k)$. Thus, the cost to go function is only a function of $x_k$. This will be clearer as we move forward.

Now, since we have a recursive function, we know that $V_k(x_k)$ is quadratic + affine with respect to $x_k$, so we can represent it as:

$$V_k(x_k) = \frac{1}{2}x_k^T P_k x_k + s_k^T x_k + c_k \tag{11}$$

Logically, this follows:

$$V_{k+1}(x_{k+1}) = \frac{1}{2}x_{k+1}^T P_{k+1} x_{k+1} + s_{k+1}^T x_{k+1} + c_{k+1} \tag{12}$$

**Finding Feedback Controls** $u_k = -K_k x_k - d_k$

Plugging Equations (12) into Equation (10)

$$V_k(x_k) = \min_{u_k} \quad \frac{1}{2}x_k^T Q_k x_k + \frac{1}{2}u_k^T R_k u_k + q_k^T x_k + r_k^T u_k + (\frac{1}{2}x_{k+1}^T P_{k+1} x_{k+1} + s_{k+1}^T x_{k+1} + c_{k+1})$$

To further simplify our problem, we know from our dynamics

$$x_{k+1} = Ax_k + Bu_k$$

So we plug that in instead of $x_{k+1}$:

$$\begin{aligned} V_k(x_k) = \min_{u_k} \quad & \frac{1}{2}x_k^T Q_k x_k + \frac{1}{2}u_k^T R_k u_k + q_k^T x_k + r_k^T u_k \\ & + \frac{1}{2}(Ax_k + Bu_k)^T P_{k+1}(Ax_k + Bu_k) + s_{k+1}^T(Ax_k + Bu_k) + c_{k+1} \end{aligned} \tag{13}$$

Now Let's perform the minimization:

$$\frac{\partial V_k(x_k)}{\partial u_k} = R_k u_k + r_k + B^T P_{k+1}(Ax_k + Bu_k) + B^T s_{k+1} u_k = 0$$

$$\frac{\partial V_k(x_k)}{\partial u_k} = R_k u_k + B^T P_{k+1} B u_k + B^T s_{k+1} + r_k + B^T P_{k+1} A x_k = 0$$

$$\frac{\partial V_k(x_k)}{\partial u_k} = (R_k + B^T P_{k+1} B)u_k + (r_k + B^T s_{k+1}) + (B^T P_{k+1} A)x_k = 0$$

Solving for $u_k$:

$$u_k = -(R_k + B^T P_{k+1} B)^{-1}(B^T P_{k+1} A)x_k - (R_k + B^T P_{k+1} B)^{-1}(B^T s_{k+1} + r_k)$$

which simplifies to

$$u_k = -K_k x_k - d_k \tag{7}$$

where

$$K_k = (R_k + B^T P_{k+1})^{-1}(B^T P_{k+1} A)$$

and

$$d_k = (R_k + B^T P_{k+1})^{-1}(r_k + B^T s_{k+1})$$

**Finding The Remaining Unknown Terms $P_k$ and $s_k$**

To find $P_k$ and $s_k$, we need to first plug in our control law (7) back into our cost-to-go function (13) to find it directly:

$$V_k(x_k) = \frac{1}{2}x_k^T Q_k x_k + \frac{1}{2}(-K_k x_k - d_k)^T R_k(-K_k x_k - d_k) + q_k^T x_k + r_k^T(-K_k x_k - d_k)$$
$$+\frac{1}{2}(Ax_k + B[-K_k x_k - d_k])^T P_{k+1}(Ax_k + B[-K_k x_k - d_k]) + s_{k+1}^T(Ax_k + B[-K_k x_k - d_k]) + c_{k+1}$$

To simplify this problem, we need to tackle it part by part:

- $\frac{1}{2}(-K_k x_k - d_k)^T R_k(-K_k x_k - d_k)$
  $= \frac{1}{2}(a+b)^T R_k(a+b)$
  $= \frac{1}{2}a^T R_k a + a^T R_k b + \frac{1}{2}b^T R_k b$
  $= \frac{1}{2}x_k^T K_k^T R_k K_k x_k + x_k^T K_k^T R_k d_k + \frac{1}{2}d_k^T R_k d_k$
- $r_k^T(-K_k x_k - d_k)$
  $= -r_k^T K_k x_k - r_k^T d_k$
- $\frac{1}{2}(Ax_k + B[-K_k x_k - d_k])^T P_{k+1}(Ax_k + B[-K_k x_k - d_k])$
  $= \frac{1}{2}((A - BK_k)x_k - Bd_k)^T P_{k+1}((A - BK_k)x_k - Bd_k)$
  $= \frac{1}{2}x_k^T(A - BK_k)^T P_{k+1}(A - BK_k)x_k - x_k^T(A - BK_k)^T P_{k+1} B d_k + \frac{1}{2}d_k^T B^T P_{k+1} B d_k$
- $s_{k+1}^T(Ax_k + B[-K_k x_k - d_k])$
  $= s_{k+1}^T((A - BK_k)x_k - Bd_k)$
  $= s_{k+1}^T(A - BK_k)x_k - s_{k+1}^T B d_k$

Thus, the cost-to-go function will simplify to:

$$V_k(x_k) = \frac{1}{2}x_k^T Q_k x_k + \frac{1}{2}x_k^T K_k^T R_k K_k x_k + x_k^T K_k^T R_k d_k + \frac{1}{2}d_k^T R_k d_k$$
$$+ q_k^T x_k - r_k^T K_k x_k - r_k^T d_k$$
$$+\frac{1}{2}x_k^T(A - BK_k)^T P_{k+1}(A - BK_k)x_k - x_k^T(A - BK_k)^T P_{k+1} B d_k + \frac{1}{2}d_k^T B^T P_{k+1} B d_k$$
$$+ s_{k+1}^T(A - BK_k)x_k - s_{k+1}^T B d_k + c_{k+1}$$

$$\begin{aligned}V_k(x_k) = \frac{1}{2}x_k^T(Q_k + K_k^T R_k K_k + (A - BK_k)^T P_{k+1}(A - BK_k))x_k \\ + x_k^T(K_k^T R_k d_k - (A - BK_k)^T P_{k+1} B d_k) \\ + (q_k^T - r_k^T K_K + s_{k+1}^T (A - BK_k))x_k \\ + (-r_k^T d_k + \frac{1}{2} d_k^T B^T P_{k+1} B d_k - s_{k+1}^T B d_k + c_{k+1} + \frac{1}{2} d_k^T R_k d_k)\end{aligned}$$

$$\begin{aligned}V_k(x_k) = \frac{1}{2}x_k^T(Q_k + K_k^T R_k K_k + (A - BK_k)^T P_{k+1}(A - BK_k))x_k \\ + x_k^T(K_k^T R_k d_k - (A - BK_k)^T P_{k+1} B d_k) \\ + x_k^T(q_k - K_k^T r_k + (A - BK_k)^T s_{k+1}) \\ + (-r_k^T d_k + \frac{1}{2} d_k^T B^T P_{k+1} B d_k - s_{k+1}^T B d_k + c_{k+1} + \frac{1}{2} d_k^T R_k d_k)\end{aligned}$$

$$\begin{aligned}V_k(x_k) = \frac{1}{2}x_k^T(Q_k + K_k^T R_k K_k + (A - BK_k)^T P_{k+1}(A - BK_k))x_k \\ + x_k^T(K_k^T R_k d_k - (A - BK_k)^T P_{k+1} B d_k + q_k - K_k^T r_k + (A - BK_k)^T s_{k+1}) \\ + (-r_k^T d_k + \frac{1}{2} d_k^T B^T P_{k+1} B d_k - s_{k+1}^T B d_k + c_{k+1} + \frac{1}{2} d_k^T R_k d_k)\end{aligned}$$

$$\begin{aligned}V_k(x_k) = \frac{1}{2}x_k^T[Q_k + K_k^T R_k K_k + (A - BK_k)^T P_{k+1}(A - BK_k)]x_k \\ + x_k^T[q_k + (A - BK_k)^T(s_{k+1} - P_{k+1} B d_k) + K_k^T(R_k d_k - r_k)] \\ + [-r_k^T d_k + \frac{1}{2} d_k^T B^T P_{k+1} B d_k - s_{k+1}^T B d_k + c_{k+1} + \frac{1}{2} d_k^T R_k d_k] \quad (14)\end{aligned}$$

Now plugging in Equation (11) into Equation (14) will yield:

$$\begin{aligned}\frac{1}{2}x_k^T P_k x_k + s_k^T x_k + c_k = \frac{1}{2}x_k^T[Q_k + K_k^T R_k K_k + (A - BK_k)^T P_{k+1}(A - BK_k)]x_k \\ + x_k^T[q_k + (A - BK_k)^T(s_{k+1} - P_{k+1} B d_k) + K_k^T(R_k d_k - r_k)] \\ + [-r_k^T d_k + \frac{1}{2} d_k^T B^T P_{k+1} B d_k - s_{k+1}^T B d_k + c_{k+1} + \frac{1}{2} d_k^T R_k d_k]\end{aligned}$$

Thus we get:

$$P_k = Q_k + K_k^T R_k K_k + (A - BK_k)^T P_{k+1}(A - BK_k) \quad (15)$$

$$s_k^T = [q_k + (A - BK_k)^T(s_{k+1} - P_{k+1} B d_k) + K_k^T(R_k d_k - r_k)]^T$$

$$s_k = q_k + (A - BK_k)^T(s_{k+1} - P_{k+1} B d_k) + K_k^T(R_k d_k - r_k) \quad (16)$$

**Final Step Boundary Conditions**

We know that final step is at k=N, so plugging that into Equation (11) we get

$$V_n(x_n) = \frac{1}{2}x_n^T P_n x_n + s_n^T x_n + c_n$$

And we know that the final cost-to-go function must be equal to the final cost function:

$$V_n(x_n) = \tfrac{1}{2}x_N^T Q_f x_N + q_f^T x_N.$$

Using both equations above:

$$\frac{1}{2}x_n^T P_n x_n + s_n^T x_n + c_n = \tfrac{1}{2}x_N^T Q_f x_N + q_f^T x_N.$$

Thus the boundary conditions at N are:

1. $P_n = Q_f$
2. $s_n = q_f$

## 3.3 Final Solution

From the derivation above, the optimal control law for the discrete-time LQR problem is:

$$u_k = -K_k x_k - d_k \tag{A}$$

where the feedback gain $K_k$ and the feedforward term $d_k$ are given by:

$$K_k = (R_k + B^T P_{k+1} B)^{-1}(B^T P_{k+1} A), \quad d_k = (R_k + B^T P_{k+1} B)^{-1}(r_k + B^T s_{k+1})$$

The matrices $P_k$ and vectors $s_k$ are computed backward in time starting from the terminal boundary conditions at $k = N$:

1. $P_N = Q_f$
2. $s_N = q_f$

The recursion equations for $P_k$ and $s_k$ are:

$$P_k = Q_k + K_k^T R_k K_k + (A - BK_k)^T P_{k+1}(A - BK_k)$$

$$s_k = q_k + (A - BK_k)^T(s_{k+1} - P_{k+1} B d_k) + K_k^T(R_k d_k - r_k)$$

## 4 Resolving Differences Between Both Methods

The Lagrangian and Dynamic Programming both lead to the same $K_k$ and $d_k$; however, the formulation of $P_k$ and $s_k$ are different. Therefore, in this section, I'll prove that both formulations are equal.

### 4.1 Resolving Differences in $P_k$:

From Lagrangian:

$$P_k = Q_k + A^T P_{k+1}(A - BK_k) \tag{17}$$

From Dynamic Programming:

$$P_k = Q_k + K_k^T R_k K_k + (A - BK_k)^T P_{k+1}(A - BK_k) \tag{18}$$

Let's start first with Equation (18) and simplify it:

$P_k = Q_k + K_k^T R_k K_k + (A - BK_k)^T P_{k+1}(A - BK_k)$
$= Q_k + K_k^T R_k K_k + A^T P_{k+1} A - A^T P_{k+1} BK_k - K_k^T B^T P_{k+1} A + K_k^T B^T P_{k+1} BK_K$
$= Q_k + A^T P_{k+1}(A - BK_k) - K_k^T B^T P_{k+1} A + K_k^T (R_k + B^T P_{k+1} B)K_k$
Using equation (7) and plugging in $K_K$
$= Q_k + A^T P_{k+1}(A - BK_k) - K_k^T B^T P_{k+1} A + K_k^T (R_k + B^T P_{k+1} B)(R_k + B^T P_{k+1} B)^{-1}(B^T P_{k+1} A)$
$= Q_k + A^T P_{k+1}(A - BK_k) - K_k^T B^T P_{k+1} A + K_k^T (B^T P_{k+1} A)$
$= Q_k + A^T P_{k+1}(A - BK_k)$
Thus, Equation (17) and (18) are equal

### 4.2 Resolving Differences in $s_k$:

From Lagrangian:

$$s_k = q_k - A^T(P_{k+1} Bd_k - s_{k+1}) \tag{19}$$

From Dynamic Programming:

$$s_k = q_k + (A - BK_k)^T (s_{k+1} - P_{k+1} Bd_k) + K_k^T (R_k d_k - r_k) \tag{20}$$

Let's start first with Equation (20) and simplify it:

$s_k = q_k + (A - BK_k)^T (s_{k+1} - P_{k+1} Bd_k) + K_k^T (R_k d_k - r_k)$
$= q_k + A^T s_{k+1} - A^T P_{k+1} Bd_k - K_k^T B^T s_{k+1} + K_k^T B^T P_{k+1} d_k + K_k^T R_k d_k - K_k^T r_k$
$= q_k + A^T (s_{k+1} - P_{k+1} Bd_k) + K_k^T (-B^T s_{k+1} + B^T P_{k+1} d_k + R_k d_k - r_k)$
$= q_k - A^T (P_{k+1} Bd_k - s_{k+1}) + K_k^T (-[B^T s_{k+1} + r_k] + [R_k + B^T P_{k+1}]d_k)$
Using equation (7) and plugging in $d_K$
$= q_k - A^T (P_{k+1} Bd_k - s_{k+1}) + K_k^T (-[B^T s_{k+1} + r_k] + [R_k + B^T P_{k+1}][(R_k + B^T P_{k+1})^{-1}(r_k + B^T s_{k+1})])$
$= q_k - A^T (P_{k+1} Bd_k - s_{k+1}) + K_k^T (-[B^T s_{k+1} + r_k] + (r_k + B^T s_{k+1}))$
$= q_k - A^T (P_{k+1} Bd_k - s_{k+1})$
Thus, Equation (19) and (20) are equal