Linear Quadraric Model Predictive Control: Notes from Theory to Praxis

Timothy Werner

Abstract. This is LQMPC – Linear-quadratic Model Predictive Control
This paper is still work in progres...

Keywords. MPC, IPM, Riccati

Contents

1 The linear-quadratic OCP

The finite-horizon linear-quadratic optimal control problem is formulated as follows:

$$\begin{aligned}
\underset{x_k,u_k}{\text{minimize}} \quad & \sum_{k=0}^{N} \frac{1}{2} \begin{bmatrix} x_k \\ u_k \\ 1 \end{bmatrix}^\top \begin{bmatrix} Q_k & S_k & q_k \\ S_k^\top & R_k & r_k \\ q_k^\top & r_k^\top & 0 \end{bmatrix} \begin{bmatrix} x_k \\ u_k \\ 1 \end{bmatrix} \\
\text{subject to} \quad & x_0 = \hat{x} \\
& x_{k+1} = A_k x_k + B_k u_k + c_k
\end{aligned}$$

With the states $x_k \in \mathcal{X} \subseteq \mathbb{R}^{n_x}$ and inputs $u_k \in \mathcal{U} \subseteq \mathbb{R}^{n_u}$. The state and input dimensions are n_x and n_u, respectively. The time horizon is $N \in \mathbb{N}_{>0}$ and the initial state is given as $\hat{x} \in \mathcal{X}$. The system dynamics are described by the matrices $A_k \in \mathbb{R}^{n_x \times n_x}$, $B_k \in \mathbb{R}^{n_x \times n_u}$ and the vector $c_k \in \mathbb{R}^{n_x}$. The cost function is defined by the state weight $Q_k \in \mathbb{S}_+^{n_x}$, which is required to be symmetric positive-semi definite. The input weight R_k, which is required to be symmetric and positive-definite on the control subspace $R_k - S_k^\top Q_k^{-1} S_k \in \mathbb{S}_{++}^{n_u \times n_u}$, and the cross-term weight $S_k \in \mathbb{R}^{n_u \times n_x}$. The linear terms of the cost function are given by the vectors $q_k \in \mathbb{R}^{n_x}$ and $r_k \in \mathbb{R}^{n_u}$.

2 Unconstrained linear-quadratic OCP

This section serves to explain the solution of the KKT system, which sits at the core of solving linear-quadratic optimal control problems. Even optimal control problems in general. Let's look at the unconstrained finite-horizon linear-quadratic optimal control problem, which is formulated as follows.

$$\begin{aligned}\underset{x_k,u_k}{\text{minimize}} \quad & \sum_{k=0}^{N} \frac{1}{2} \begin{bmatrix} x_k \\ u_k \\ 1 \end{bmatrix}^\top \begin{bmatrix} Q_k & S_k & q_k \\ S_k^\top & R_k & r_k \\ q_k^\top & r_k^\top & 0 \end{bmatrix} \begin{bmatrix} x_k \\ u_k \\ 1 \end{bmatrix} \\ \text{subject to} \quad & x_0 = \hat{x} \\ & x_{k+1} = A_k x_k + B_k u_k + c_k \end{aligned}$$

Note that *unconstrained* in this case means that there are no *inequality* constraints on states or inputs. The only constraints are the system dynamics and the initial state, but these are primal feasibility constraints, which means they must be satisfied in order to attain a feasible solution. They are thus not a constraint on the problem in the sense of restricting the solution space.

2.1. Solution via Karush-Kuhn-Tucker. In terms of numeric optimization, the OCP above is an equlaity-constrained QP, which is convex and thus has a unique solution. This solution is characterized by the Karush-Kuhn-Tucker (KKT) conditions. To derive the KKT conditions, we first form the Lagrangian by introducing the Lagrange multipliers $\pi_k \in \mathbb{R}^{n_x}$ for the dynamics constraints.

$$\mathcal{L}(x_k, u_k, \pi_k) = \sum_{k=0}^{N-1} l_k(x_k, u_k) + l_N(x_N) + \sum_{k=0}^{N-1} \pi_{k+1}^\top (A_k x_k + B_k u_k + c_k - x_{k+1})$$

With the stage cost

$$l_k(x_k, u_k) = \frac{1}{2} x_k^\top Q_k x_k + \frac{1}{2} u_k^\top R_k u_k + x_k^\top S_k^\top u_k + q_k^\top x_k + r_k^\top u_k$$

and terminal cost

$$l_N(x_N) = \frac{1}{2} x_N^\top Q_N x_N + q_N^\top x_N$$

The KKT conditions are obtained by the stationary conditions, which says that the Lagrangian becomes stationary at the optimal solution with respect to all primal and dual variables. The stationary conditions are given as follows.
Stationary w.r.t. states x_k (for $k = 0, \ldots, N$):

$$\nabla_{x_k} \mathcal{L} = Q_k x_k + S_k u_k + q_k + A_k^\top \pi_{k+1} - \pi_k = 0$$

Stationary w.r.t. inputs u_k (for $k = 0, \ldots, N-1$):

$$\nabla_{u_k} \mathcal{L} = R_k u_k + S_k^\top x_k + r_k + B_k^\top \pi_{k+1} = 0$$

Stationary w.r.t. Lagrange multipliers π_k (for $k = 1, \ldots, N$):

$$\nabla_{\pi_k} \mathcal{L} = A_{k-1} x_{k-1} + B_{k-1} u_{k-1} + c_{k-1} - x_k = 0$$

If we stack the decision variables and Lagrange multipliers into a single vector, we can write the KKT conditions as a large block-tridiagonal linear system. This system obey a sparsity pattern, that can be exploited when solving it. Here is an example showing the structure for $N = 2$.

$$\begin{bmatrix} Q_0 & S_0 & & & & A_0^\top & \\ S_0^\top & R_0 & & & & B_0^\top & \\ & & Q_1 & S_1 & & -I & A_1^\top \\ & & S_1^\top & R_1 & & & B_1^\top \\ & & & & Q_2 & & -I \\ A_0 & B_0 & -I & & & & \\ & & A_1 & B_1 & -I & & \end{bmatrix} \begin{bmatrix} x_0 \\ u_0 \\ x_1 \\ u_1 \\ x_2 \\ \pi_0 \\ \pi_1 \end{bmatrix} = - \begin{bmatrix} q_0 \\ r_0 \\ q_1 \\ r_1 \\ q_N \\ c_0 \\ c_1 \end{bmatrix}$$

The system can be summarized in the following form, which is the standard system arising from optimal control problems.

$$\begin{bmatrix} \mathcal{H} & \mathcal{A}^\top \\ \mathcal{A} & 0 \end{bmatrix} \begin{bmatrix} y \\ \pi \end{bmatrix} = \begin{bmatrix} -h \\ -c \end{bmatrix}$$

where the Hessians $\mathcal{H}$ is block-diagonal and $\mathcal{A}$ is block-bidiagonal.

There are in general three approaches to solve this system. The naive approach would be to feed this system as is to a dense cholesky solver. This works fine, since the hessian is $\mathcal{H} \in \mathbb{S}_+$. But this is is of order O(N3) ? and thus not relevant for practival applications. The other two approaches are worth considering in a bit more detail.

2.1.1 From KKT to Block-Cholesky

The second approach is to notice the spacial sparsity pattern of the hessian $\mathcal{H}$ and realize, the the solution to the system obayes a block-diagonal structure. To show this, let us attempt to solve this system forming the shur complement. Eliminate y from the KKT system

$$y = \mathcal{H}^{-1}(-h - \mathcal{C}^\top \pi)$$

and substitute into $\mathcal{C}y = -c$, we get

$$\mathcal{C}\mathcal{H}^{-1}\mathcal{C}^\top \pi = \mathcal{C}\mathcal{H}^{-1}h - c$$

We now define $Y := \mathcal{C}\mathcal{H}^{-1}\mathcal{C}^\top$ and $\beta := c - \mathcal{C}\mathcal{H}^{-1}h$, to arrive at the system in normal equations form.

$$\begin{aligned} Y\pi &= \beta \\ y &= \mathcal{H}^{-1}(-h - C^\top \pi) \end{aligned}$$

Observing the structure of Y, we see that $\mathcal{H}$ is block-diagonal with the stage hessian blocks

$$H_k = \begin{bmatrix} Q_k & S_k \\ S_k^\top & R_k \end{bmatrix}$$

$\mathcal{C}$ is block-bidiagonal, and thus Y is block-tridiagonal

$$Y = \begin{bmatrix} Y_{00} & Y_{01} & & \\ Y_{01}^\top & Y_{11} & Y_{12} & \\ & Y_{12}^\top & Y_{22} & \ddots \\ & & \ddots & \ddots \end{bmatrix}$$

with $\Phi_k := H_k^{-1}$, the blocks of Y are given as

$$Y_{k,k} = \begin{bmatrix} A_{k-1} & B_{k-1} \end{bmatrix} \Phi_{k-1} \begin{bmatrix} A_{k-1}^\top \\ B_{k-1}^\top \end{bmatrix} + \begin{bmatrix} I & 0 \end{bmatrix} \Phi_k \begin{bmatrix} I \\ 0 \end{bmatrix}$$

$$Y_{k,k+1} = -\begin{bmatrix} I & 0 \end{bmatrix} \Phi_k \begin{bmatrix} A_k^\top \\ B_k^\top \end{bmatrix}$$

The next step is the cholesky factorization of Y

$$Y = L_Y L_Y^\top$$

Due to the block-tridiagonal structure of Y, L_Y is block-lower-bidiagonal:

$$L_Y = \begin{bmatrix} L_{00} & & & \\ L_{10} & L_{11} & & \\ & L_{21} & L_{22} & \\ & & \ddots & \ddots \end{bmatrix}$$

This fact motivates a blockwise factorization with the following recursion

$$L_{kk} L_{kk}^\top = Y_{kk} - L_{k,k-1} L_{k,k-1}^\top$$

$$L_{k+1,k} L_{kk}^\top = Y_{k+1,k}$$

Solve $L_Y z = \beta$ by forward subsditution, solve $L_Y^\top \pi = z$ by backward substitution and finally, recover $y = H^{-1}(-h - C^\top \pi)$.

2.1.2 From KKT to Dynamic Programming

The third approach is to notice, that the problem also has temporal structure. To be fair, the spacial structure we exploited above stems exactly from that temporal structure. The KKT conditions derived above form a system of $2n_x + n_u$ equations per stage, coupled across all time steps. While we could assemble these into a large block-tridiagonal linear system and solve directly, there is a more elegant approach that exploits the temporal structure of the problem.

Key observation: The stationarity condition with respect to x_k can be rewritten as

$$\pi_k = Q_k x_k + S_k u_k + q_k + A_k^\top \pi_{k+1}$$

This reveals that π_k propagates *backward* in time, while the dynamics $x_{k+1} = A_k x_k + B_k u_k + c_k$ propagate *forward*. The coupling between these two directions is what makes the problem non-trivial.

Dynamic programming resolves this by asking: *if we knew the optimal cost-to-go from any state x_k at time k, what would the optimal action be?*

2.2. Dynamic Programming and the Principle of Optimality. Define the **optimal cost-to-go** (or value function) as the minimum cost achievable from state x_k at time k:

$$V_k(x_k) = \min_{u_k,\dots,u_{N-1}} \left\{ \sum_{j=k}^{N-1} l_j(x_j, u_j) + l_N(x_N) \;\middle|\; x_{j+1} = A_j x_j + B_j u_j + c_j \right\}$$

Bellman's **principle of optimality** states that the value function satisfies the recursive relationship:

$$V_k(x_k) = \min_{u_k} \{ l_k(x_k, u_k) + V_{k+1}(A_k x_k + B_k u_k + c_k) \}$$

with terminal condition $V_N(x_N) = l_N(x_N)$.

The crucial insight for linear-quadratic problems is that if V_{k+1} is quadratic in the state, then V_k is also quadratic. Since the terminal cost l_N is quadratic, by induction the value function is quadratic at all stages:

$$V_k(x_k) = \frac{1}{2} x_k^\top P_k x_k + p_k^\top x_k + \rho_k$$

where $P_k \in \mathbb{R}^{n_x \times n_x}$ is symmetric positive semidefinite, $p_k \in \mathbb{R}^{n_x}$, and $\rho_k \in \mathbb{R}$ is a scalar (which we can ignore for optimization purposes).

2.3. Derivation of the Riccati Recursion. Substituting the quadratic form into Bellman's equation and expanding:

$$\begin{aligned} V_k(x_k) = \min_{u_k} \Big\{ &\frac{1}{2} x_k^\top Q_k x_k + \frac{1}{2} u_k^\top R_k u_k + x_k^\top S_k u_k + q_k^\top x_k + r_k^\top u_k \\ &+ \frac{1}{2}(A_k x_k + B_k u_k + c_k)^\top P_{k+1} (A_k x_k + B_k u_k + c_k) \\ &+ p_{k+1}^\top (A_k x_k + B_k u_k + c_k) + \rho_{k+1} \Big\} \end{aligned}$$

The expression inside the minimization is quadratic in u_k. Setting the gradient with respect to u_k to zero:

$$R_k u_k + S_k^\top x_k + r_k + B_k^\top P_{k+1}(A_k x_k + B_k u_k + c_k) + B_k^\top p_{k+1} = 0$$

Solving for u_k:

$$u_k^* = -\underbrace{(R_k + B_k^\top P_{k+1} B_k)^{-1}(S_k^\top + B_k^\top P_{k+1} A_k)}_{K_k} x_k - \underbrace{(R_k + B_k^\top P_{k+1} B_k)^{-1}(r_k + B_k^\top P_{k+1} c_k + B_k^\top p_{k+1})}_{k_k}$$

This gives the optimal control as an **affine state feedback**:

$$\boxed{u_k^* = -K_k x_k - k_k}$$

Substituting u_k^* back into the Bellman equation and matching coefficients with $V_k(x_k) = \frac{1}{2} x_k^\top P_k x_k + p_k^\top x_k + \rho_k$ yields the **Riccati recursion**.