TinyMPC Report

I. Introduction

Objective: Control a system to track a nominal trajectory with constraints.

II. General TVLQR Derivation

A. Nominal Trajectory

Assume we are interested in a time period T, which can be discretized by N intervals of dt. We have:
$\bar{X} = \{\bar{x}_0, \dots, \bar{x}_N\}$ is the nominal state trajectory, $\bar{U} = \{\bar{u}_0, \dots, \bar{u}_{N-1}\}$ is the nominal input trajectory.

These can be obtained from a higher-level planner using sample-based, grid-based, or optimization-based methods. These trajectories can be dynamically feasible, i.e. satisfies (1), or not. REx Lab is often interested in trajectory optimization like ALTRO for this task.

B. Linearized Dynamics

General, nonlinear, discretized dynamics of a system:

$$x_{k+1} = f(x_k, u_k) \tag{1}$$

Define the difference between actual and nominal trajectory as $\delta x_k = x_k - \bar{x}_k, \delta u_k = u_k - \bar{u}_k$.

We locally approximate the nonlinear dynamics with a first-order Taylor expansion about the nominal trajectory

$$x_{k+1} = \bar{x}_{k+1} + \delta x_{k+1} = f(\bar{x}_k + \delta x_k, \bar{u}_k + \delta u_k) \approx f(\bar{x}_k, \bar{u}_k) + \left.\frac{\partial f}{\partial x}\right|_{\bar{x}_k, \bar{u}_k} \delta x_k + \left.\frac{\partial f}{\partial u}\right|_{\bar{x}_k, \bar{u}_k} \delta u_k \tag{2}$$

with

$$A_k \equiv \left.\frac{\partial f}{\partial x}\right|_{\bar{x}_k, \bar{u}_k}, \quad B_k \equiv \left.\frac{\partial f}{\partial u}\right|_{\bar{x}_k, \bar{u}_k}$$

Then,

$$\bar{x}_{k+1} + \delta x_{k+1} = f(\bar{x}_k, \bar{u}_k) + A_k \delta x_k + B_k \delta u_k \tag{3}$$

Now, we have two options to represent state and input, i.e. absolute x_k or delta δx_k. This will lead to two different formulations for our optimal control problem as well.

Delta formulation:

$$\begin{aligned} \delta x_{k+1} &= A_k \delta x_k + B_k \delta u_k + f(\bar{x}_k, \bar{u}_k) - \bar{x}_{k+1} \\ \delta x_{k+1} &= A_k \delta x_k + B_k \delta u_k + f_k \end{aligned} \tag{4}$$

Absolute formulation:

$$\begin{aligned} x_{k+1} &= A_k x_k + B_k u_k + f(\bar{x}_k, \bar{u}_k) - A_k \bar{x}_k - B_k \bar{u}_k \\ x_{k+1} &= A_k x_k + B_k u_k + f_k \end{aligned} \tag{5}$$

In both formulations, we denote f_k as the affine term in the linearized dynamics. In (4), it implies the dynamical infeasibility of the nominal trajectory, otherwise, it is zero. In (5), it includes the dynamics linearization error as well. The dynamics Jacobians A_k and B_k only depend on nominal trajectory therefore the derived system is basically linear time-varying and can be precomputed.

For the ease of representing constraints later, the absolute formulation (5) is preferred and used in this note for the optimal control problem.

C. Cost Function

The tracking linear-quadratic cost (not considering the mixed term and ignoring the constant term)

$$\begin{aligned} J &= \ell_f(x_N) + \sum_{k=1}^{N-1} \ell(x_k, u_k) \\ &= \frac{1}{2}(x_N - \bar{x}_N)^\intercal Q_f (x_N - \bar{x}_N) + \sum_{k=1}^{N-1} \frac{1}{2}(x_k - \bar{x}_k)^\intercal Q_k (x_k - \bar{x}_k) + \frac{1}{2}(u_k - \bar{u}_k)^\intercal R_k (u_k - \bar{u}_k) \\ &= \frac{1}{2} x_N^\intercal Q_f x_N + q_f^\intercal x_N + \sum_{k=1}^{N-1} \frac{1}{2} x_k^\intercal Q_k x_k + \frac{1}{2} u_k^\intercal R_k u_k + q_k^\intercal x_k + r_k^\intercal u_k \end{aligned} \tag{6}$$

where $q_f = -Q_f \bar{x}_N, q_k = -Q_k \bar{x}_k, r_k = -R_k \bar{u}_k$

D. Dynamic Programming Problem

Cost-to-go is supposed to be in the linear-quadratic form

$$V_k(x_k) = \frac{1}{2} x_k^\intercal P_k x_k + p_k^\intercal x_k \tag{7}$$

At terminal state, $P_N = Q_f, p_N = q_f$

Bellman equation:

$$\begin{aligned} V_k &= \min_{u_k} \{\ell(x_k, u_k) + V_{k+1}(f(x_k, u_k))\} \\ &= \min_{u_k} \{Q_k(x_k, u_k)\} \end{aligned} \tag{8}$$

Action-value function:

$$\begin{aligned} Q_k(x_k, u_k) = \frac{1}{2} x_k^\intercal Q_k x_k + \frac{1}{2} u_k^\intercal R_k u_k + q_k^\intercal x_k + r_k^\intercal u_k \\ + \frac{1}{2}(A_k x_k + B_k u_k + f_k)^\intercal P_{k+1}(A_k x_k + B_k u_k + f_k) + p_{k+1}^\intercal (A_k x_k + B_k u_k + f_k) \end{aligned} \tag{9}$$

Group Q_k into linear and quadratic terms:

$$Q_k(x_k, u_k) = \frac{1}{2} \begin{bmatrix} x_k \\ u_k \end{bmatrix}^\intercal \begin{bmatrix} Q_{xx} & Q_{xu} \\ Q_{ux} & Q_{uu} \end{bmatrix} \begin{bmatrix} x_k \\ u_k \end{bmatrix} + \begin{bmatrix} Q_x \\ Q_u \end{bmatrix}^\intercal \begin{bmatrix} x_k \\ u_k \end{bmatrix} \tag{10}$$

$$\begin{aligned} Q_x &= q_k + A_k^\intercal (P_{k+1} f_k + p_{k+1}) \\ Q_u &= r_k + B_k^\intercal (P_{k+1} f_k + p_{k+1}) \\ Q_{xx} &= Q_k + A_k^\intercal P_{k+1} A_k \\ Q_{uu} &= R_k + B_k^\intercal P_{k+1} B_k \\ Q_{ux} &= B_k^\intercal P_{k+1} A_k = Q_{xu}^\intercal \end{aligned} \tag{11}$$

Apply the necessary optimality condition to achieve optimal control:

$$0 = \frac{\partial Q_k}{\partial u_k} = Q_{uu} u_k + Q_{ux} x_k + Q_u \tag{12}$$

Problem (8) has a closed-form solution:

$$\begin{aligned} u_k^* &= -K_k x_k - d_k \\ d_k &= Q_{uu}^{-1} Q_u \\ K_k &= Q_{uu}^{-1} Q_{ux} \end{aligned} \tag{13}$$

Plug these back into (9) to get the cost-to-go:

$$\begin{aligned} p_k &= Q_x + K_k^\intercal Q_{uu} d_k - K_k^\intercal Q_u - Q_{ux}^\intercal d_k \\ P_k &= Q_{xx} + K_k^\intercal Q_{uu} K_k - 2K_k^\intercal Q_{ux} \end{aligned} \tag{14}$$

All of these result in Riccati recursion (backward pass) to calculate cost-to-go and control gains. Because the system is locally linear, we only need to perform one iteration to obtain optimality.

E. Forward Pass

Note that we are using the absolute formulation, the forward pass is as below:

$$u_k = -K_k x_k - d_k \tag{15}$$

$$x_{k+1} = A_k x_k + B_k u_k + f_k \tag{16}$$

III. Constrained Optimal Control Problem

The previous derivation does not consider any constraint. In real-world applications, many types of constraints will negatively affect performance. Some popular ones are box constraints on state and input at each time step, equality constraints on the goal state, and second-order cone constraints.

To handle constraints, we use the augmented Lagrangian method (ALM) which enforces constraints into the cost function. Generally, our tracking problem is convex, and a global solution can be found with only one constrained backward pass. However, multiple iterations are needed to satisfy dual feasibility.

Below is the pseudo-code of our AL-TVLQR algorithm.

Algorithm 1: Augmented Lagrangian TVLQR

```
initialization;
for $i \leftarrow 0$ to maxIters do
    $K, k, P, p \leftarrow$ ConstrainedBackwardPass;
    $X, U \leftarrow$ ForwardPass;
    CalculateConstraints;
    if contraint satisfied then
        return;
    end
    UpdateDuals;
    UpdatePenalty;
end
```

A. Augmented Lagrangian Cost Function

First, consider linear equality and inequality constraints as follows:

$$h_k^x(x) = H_k^x x_k - h_{0k}^x = 0, \tag{17}$$

$$g_k^x(x) = G_k^x x_k - g_{0k}^x \leq 0 \tag{18}$$

$$h_k^u(u) = H_k^u u_k - h_{0k}^u = 0, \tag{19}$$

$$g_k^u(u) = G_k^u u_k - g_{0k}^u \leq 0 \tag{20}$$

Let's look at the constrained backward pass. Basically, the ALM method will add Lagrangian and penalty terms into the original cost function, turning it into an unconstrained optimization problem. Interestingly, these linear-quadratic terms fit nicely into the LQR formulation.

The augmented Lagrangian applies to the action-value function (assuming only constraints on u_k)

$$Q_k(x_k, u_k, \lambda_k, \mu_k) \leftarrow Q_k + \lambda_k^\intercal h_k(u_k) + \frac{1}{2}\rho h_k(u_k)^\intercal h_k(u_k) + \mu_k^\intercal g_k(u_k) + \frac{1}{2} g_k(u_k)^\intercal I_\rho g_k(u_k) \tag{21}$$

Apply the necessary optimality condition to achieve optimal control:

$$0 = \frac{\partial Q_k}{\partial u_k} = Q_{uu}u_k + Q_{ux}x_k + Q_u + \lambda_k^\intercal \frac{\partial h_k}{\partial u_k} + \rho h_k(u_k)^\intercal \frac{\partial h_k}{\partial u_k} + \mu^\intercal \frac{\partial g_k}{\partial u_k} + g_k(u_k)^\intercal I_\rho \frac{\partial g_k}{\partial u_k} \tag{22}$$

Substitute the linear constraint cases (19) and (20) into (22) to obtain:

$$0 = Q_{uu}u_k + Q_{ux}x_k + Q_u + \lambda_k^\intercal H_k + \rho(H_k u_k - h_{0k})^\intercal H_k + \mu_k^\intercal G_k + (G_k u_k - g_{0k})^\intercal I_\rho G_k \tag{23}$$

$$= (Q_{uu} + \rho H_k^\intercal H_k + G_k^\intercal I_\rho G_k)u_k + Q_{ux}x_k + [Q_u + H_k^\intercal(\lambda_k - \rho h_{0k}) + G_k^\intercal(\mu_k - I_\rho g_{0k})] \tag{24}$$

The modification to the backward pass is as follows:

$$\begin{aligned} Q_u &= r_k + B_k^\intercal(P_{k+1}f_k + p_{k+1}) + H_k^\intercal(\lambda_k - \rho h_{0k}) + G_k^\intercal(\mu_k - I_\rho g_{0k}) \\ Q_{uu} &= R_k + B_k^\intercal P_{k+1}B_k + \rho H_k^\intercal H_k + G_k^\intercal I_\rho G_k \end{aligned} \tag{25}$$

Note that the same modification would apply for state constraint Q_x, Q_{xx} and terminal state P_N, p_N. We may have mixed state-input constraints as well.

If the constraints are nonlinear, we can think of linearizing it about the nominal trajectory and get the form (17) **to** (20) **which fit** (22)**.**

B. Dual Updates

Eq. (25) suggests the dual updates as:

$$\lambda_k \leftarrow \lambda_k - \rho h_{0k}, \tag{26}$$

$$\mu_k \leftarrow \max(0, \mu_k - I_\rho g_{0k}) \tag{27}$$

Note the subtle difference between AL here and in iterative LQR. In iLQR, one will naturally approximate the cost with second-order Taylor expansion about the current trajectory, then group the cost in gradient and Hessian terms, not precisely like linear and quadratic terms like ours. Third-rank tensors are ignored. Moreover, one will solve backward pass and forward pass iteratively until convergence so that any mismatch due to nonlinearity can be eliminated. **In** (22)**, constraints have to be in linear form so they can be substituted and grouped into corresponding terms.**

C. Conic Constraints

There is a mathematical background in generalized inequality which starts with the conic combination. Typical inequality like (18) can be seen as a type of cone called non-positive orthant. In fact, the steps of dual update (27) can be interpreted as a projection of the new value into the dual cone (non-positive orthant). Most of the cones we care about are self-dual. This projection operator helps drive the duals back to the constraint manifold.

1) Generalized Inequality: Let's look at the generalized inequality (conic constraint)

$$c(x) \in \mathcal{K} \tag{28}$$

while $c(x)$ has to be linear, i.e. $c(x) = Ax + b$

The conic augmented Lagrangian is defined as

$$\mathcal{L}_\rho = Q(x) + \frac{1}{2\rho}\left(\|(\Pi_\mathcal{K}(\lambda + \rho c(x)\|^2 - \|\lambda\|^2\right) \tag{29}$$

Define $\hat{\lambda} = \lambda + \rho c(x)$, the gradient for the conic penalty term is of the following form:

$$G = \nabla c(x)^\intercal \nabla \Pi_\mathcal{K}(\hat{\lambda})^\intercal \Pi_\mathcal{K}(\hat{\lambda}) \tag{30}$$

The full Hessian for the conic penalty term is of the following form:

$$H = \rho \nabla c(x)^\intercal \nabla \Pi_\mathcal{K}(\hat{\lambda})^\intercal \nabla \Pi_\mathcal{K}(\hat{\lambda}) \nabla c(x) + \rho \nabla c(x)^\intercal \nabla^2 \Pi_\mathcal{K}(\hat{\lambda}) \Pi_\mathcal{K}(\hat{\lambda}) \nabla c(x) + \rho \nabla^2 c(x)^\intercal \nabla \Pi_\mathcal{K}(\hat{\lambda})^\intercal \Pi_\mathcal{K}(\hat{\lambda}) \tag{31}$$

We will only use the first two terms in the Hessian

$$H = \rho \nabla c(x)^\intercal \left[(\nabla \Pi_\mathcal{K}(\hat{\lambda})^\intercal \nabla \Pi_\mathcal{K}(\hat{\lambda}) + \nabla^2 \Pi_\mathcal{K}(\hat{\lambda}) \Pi_\mathcal{K}(\hat{\lambda})\right] \nabla c(x) \tag{32}$$

To make it fit the LQR formulation, we can approximate that conic penalty term to second-order about nominal trajectory (previous solve):

$$\mathcal{L}_\rho = Q(x) + G^T(x-\bar{x}) + \frac{1}{2}(x-\bar{x})^T H (x-\bar{x}) \tag{33}$$

Apply the optimality condition:

$$\frac{\partial \mathcal{L}_\rho}{\partial x} = \frac{\partial Q}{\partial x} + (G^T - \bar{x}^T H) + Hx \tag{34}$$

Now conic Riccati becomes:

$$\begin{aligned} Q_x &= q_k + A_k^\intercal(P_{k+1} f_k + p_{k+1}) + (G_k^T - \bar{x}_k^T H_k) \\ Q_{xx} &= Q_k + A_k^\intercal P_{k+1} A_k + H_k \end{aligned} \tag{35}$$

The dual update for conic constraints becomes:

$$\lambda \leftarrow \Pi_\mathcal{K}(\lambda + \rho c(x)) \tag{36}$$

2) Interpretation: Let's look at the simple case when we have a non-positive orthant cone

$$c(x) \in \mathcal{K}_-, \text{ or } c(x) \leq 0 \tag{37}$$

Simple closed-form expressions for the projection operator exist for several other cones, including the non-positive orthant

$$\Pi_{\mathcal{K}_-}(x) = I^*(x)x \tag{38}$$

where I^* is a mask/selection matrix

$$I^*(x_i) = \begin{cases} 0 & x_i > 0 \\ 1 & x_i \leq 0 \quad \text{(in the cone)} \end{cases} \tag{39}$$

Eq. (29) becomes (sloppy):

$$\begin{aligned} \mathcal{L}_\rho &= f(x) + \frac{1}{2\rho}\left(I^*(\mu + \rho c(x))^\intercal I^*(\mu + \rho c(x)) - \mu^\intercal \mu\right) \\ &= f(x) + \mu^\intercal c(x) + \frac{1}{2} c(x) I_\rho c(x) \end{aligned} \tag{40}$$

3) Second-Order Cone: Second-order cone constraint on controls:

$$g(u) = u \in \mathcal{K}_{SOC} = \{(v,s) \in \mathbb{R} \mid \|v\|_2 - s \leq 0\} \tag{41}$$

where $v = [u_1, \ldots, u_{p-1}]^\intercal$, $s = u_p$, or $\|Au\| \leq c^\intercal u$ with $A = \text{diag}(1,1,\ldots,0)$, $c^\intercal = [0,0,\ldots,1]$

Simple closed-form expressions for the projection operator exist for several other cones, including the second-order cone:

$$\Pi_{\mathcal{K}_{SOC}}(x) = \begin{cases} 0 & \|v\|_2 \leq -s \\ x & \|v\|_2 \leq s \\ \frac{1}{2}(1 + \frac{s}{\|v\|_2})[v^\intercal \ \ \|v\|_2]^\intercal & \|v\|_2 > |s| \end{cases} \tag{42}$$

The conic-augmented Lagrangian is defined as

$$Q_k(x_k, u_k, \mu_k) \leftarrow Q_k + \frac{1}{2\rho}\left(\|(\Pi_\mathcal{K}(\mu_k - \rho g_k(u_k))\|^2 - \|\mu_k\|^2\right) \tag{43}$$

The projection is separated into the vector and scalar parts:

$$\Pi_{\mathcal{K}}(\hat{\lambda}) = \begin{bmatrix} \Pi_v \\ \Pi_s \end{bmatrix} \tag{44}$$

The first two cases are trivial, we will write down the last case:

$$\Pi_{\mathcal{K}}(\hat{\lambda}) = \begin{bmatrix} \frac{1}{2}(v + \frac{sv}{\|v\|}) \\ \frac{1}{2}(\|v\| + s) \end{bmatrix} \quad , \|v\|_2 > |s| \tag{45}$$

The gradient of the SOC projection becomes

$$\nabla \Pi_{\mathcal{K}} = \begin{bmatrix} \frac{\partial \Pi_v}{\partial v} & \frac{\partial \Pi_v}{\partial s} \\ \frac{\partial \Pi_s}{\partial v} & \frac{\partial \Pi_s}{\partial s} \end{bmatrix} \tag{46}$$

where

$$\frac{\partial \Pi_v}{\partial v} = \frac{1}{2}(1 + s\frac{I - \frac{vv^T}{v^Tv}}{\|v\|}) \tag{47}$$

$$\frac{\partial \Pi_v}{\partial s} = \frac{\partial \Pi_s}{\partial v}^T = \frac{v}{2\|v\|} \tag{48}$$

$$\frac{\partial \Pi_s}{\partial s} = \frac{1}{2} \tag{49}$$

The Hessian of the SOC projection is a third-rank tensor so we will use its product with the projection itself (a vector) to get a matrix

$$\nabla^2 \Pi_{\mathcal{K}} \cdot \Pi_{\mathcal{K}} = \partial \left(\begin{bmatrix} \frac{\partial \Pi_v}{\partial v} & \frac{\partial \Pi_v}{\partial s} \\ \frac{\partial \Pi_s}{\partial v} & \frac{\partial \Pi_s}{\partial s} \end{bmatrix} \right) \cdot \begin{bmatrix} \Pi_v \\ \Pi_s \end{bmatrix} = \begin{bmatrix} \frac{\partial \Pi_v}{\partial v \partial v}\Pi_v + \frac{\partial \Pi_s}{\partial v \partial v}\Pi_s & \frac{\partial \Pi_v}{\partial v \partial s}\Pi_v + \frac{\partial \Pi_s}{\partial v \partial s}\Pi_s \\ \frac{\partial \Pi_v}{\partial s \partial v}\Pi_v + \frac{\partial \Pi_s}{\partial s \partial v}\Pi_s & \frac{\partial \Pi_v}{\partial s \partial s}\Pi_v + \frac{\partial \Pi_s}{\partial s \partial s}\Pi_s \end{bmatrix} \tag{50}$$

with

$$\frac{\partial \Pi_v}{\partial v \partial v} = \dots \tag{51}$$

$$\frac{\partial \Pi_s}{\partial v \partial v} = \frac{\partial \Pi_v}{\partial s \partial v}^T = \frac{I - \frac{vv^T}{v^Tv}}{2\|v\|} \tag{52}$$

$$\frac{\partial \Pi_v}{\partial v \partial s} = \frac{I - \frac{vv^T}{v^Tv}}{2\|v\|} \tag{53}$$

$$\frac{\partial \Pi_s}{\partial v \partial s} = \frac{\partial \Pi_s}{\partial s \partial v} = \frac{\partial \Pi_v}{\partial s \partial s} = \frac{\partial \Pi_v}{\partial s \partial s} = \mathbf{0} \tag{54}$$

IV. Fast Conic MPC

As for tracking problems, a linear model is good enough to ensure great performance and constraint satisfaction. Therefore, at each time step (MPC step), considering current feedback as the initial state, we use AL-TVLQR to solve the constrained optimization problem within a horizon of H steps into the future. Then we apply the first control input (gains) to the system.

The key to successful real-time applications is to quickly make an improved (not optimal) decision based on the feedback information. Warm-starting plays an important role in this game. In many cases, people often limit the MPC solve to one to several iterations.

V. Square Root Backward Pass

Backward pass involves a series of linear algebra on matrices, especially inversing Q_{uu}. making errors accumulate. The condition numbers of matrices keep increasing exponentially, making the algorithm vulnerable to numerical issues.

There are several reasons for the numerical illness of the algorithms (gradient vanishing/explosion):

1) Long horizon. This may not be so bad because MPC often uses short horizons.
2) Poorly-scaled dynamics (A and B). The model should be properly derived.
3) Poorly-scaled weight matrices (Q and R). They are often tuned as part of the control design.
4) Machine precision. The single-floating point will be a big problem.