Probabilistic Iterative LQR for Short Time Horizon MPC

Teguh Santoso Lembono and Sylvain Calinon

***Abstract*— Optimal control is often used in robotics for planning a trajectory to achieve some desired behavior, as expressed by the cost function. Most works in optimal control focus on finding a single optimal trajectory, which is then typically tracked by another controller. In this work, we instead consider trajectory distribution as the solution of an optimal control problem, resulting in better tracking performance and a more stable controller. A Gaussian distribution is first obtained from an iterative Linear Quadratic Regulator (iLQR) solver. A short horizon Model Predictive Control (MPC) is then used to track this distribution. We show that tracking the distribution is more cost-efficient and robust as compared to tracking the mean or using iLQR feedback control. The proposed method is validated with kinematic control of 7-DoF Panda manipulator and dynamic control of 6-DoF quadcopter in simulation.**

I. Introduction

Optimal control is a versatile problem formulation that has a large number of applications. With the increasing computational power, it can be used in more complex systems with high degrees of freedom. While the term *control* suggests that the formulation is used to find an optimal control input of a given problem, it is often used also for planning, i.e., to find the state trajectory that minimizes the cost function, typically for a long time horizon to anticipate future events. For example, optimal control is used for planning multiple quadcopters trajectories in a constrained space [1], biped walking generation [2] [3], centroidal dynamics trajectory [4] [5], whole-body motion planning [6] [7], and visual servoing [8]. The planned trajectory is usually tracked by some other controller, e.g., PID controller, or shorter time horizon MPC [8]. The formulation is convenient as one can derive the cost function from the desired behavior and obtain the corresponding optimal state and control trajectory.

However, most optimal control approaches focus on finding only a single optimal output. When the optimal control problem (OCP) is only one part of a multi-step process, which is often the case when it is used for planning, this can be an important limitation. For example, the optimal trajectory may not be feasible for the subsequent step due to the presence of a new obstacle or model errors. In this work, we consider a probabilistic formulation of OCP that allows us to obtain not only a single output, but a probability distribution of the output that minimizes the OCP's cost.

Among many algorithm variants to solve OCP, iterative Linear Quadratic Regulator (iLQR) [9] is often used in robotics due to its computational efficiency. By iteratively approximating the cost function and the dynamics as quadratic and linear, respectively, an LQR subproblem is solved at each step. It has been used for high dimensional systems such as quadruped and humanoid robots [7]. We show that a probabilistic solution of iLQR can be obtained efficiently by using the information provided by a standard iLQR solver.

The authors are with the Idiap Research Institute, Martigny, Switzerland, and with EPFL, Lausanne, Switzerland.

This work was supported by the European Commission under the EU H2020 collaborative project MEMMO (Memory of Motion, `http://www.memmo-project.eu/`), Grant Agreement No. 780684.

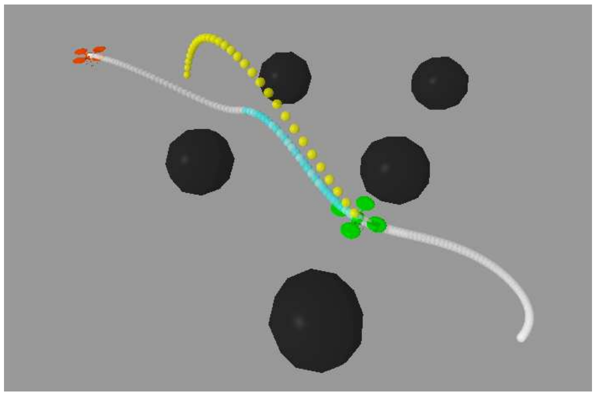

Fig. 1. Tracking an iLQR trajectory of a quadcopter. The current position, the goal position, the initial planned trajectory, and the obstacles are shown in green, red, white, and black, respectively. At the current state, an upward velocity disturbance is introduced to the system. For MPC$_{\text{marg}}$, the short horizon OCP reference trajectory (shown in cyan) remains along the planned trajectory because it does not depend on the current state. For MPC$_{\text{cond}}$, the reference trajectory is calculated by conditioning the trajectory distribution on the current state. Since the disturbance adds an upward velocity, the reference trajectory adjusts accordingly, as shown in yellow.

The probabilistic treatment of OCP is discussed by Kappen *et al.* [10], who formulate it as minimizing Kullback-Leibler (KL) divergence. The optimal control solution is the product of the free dynamics and the exponentiated cost (i.e., the exponent of the cost function is considered as an unnormalized probability distribution). Toussaint [11] shows that using an approximate inference method (similar to expectation propagation) to solve an optimal control problem results in an algorithm that is similar to iterative Linear Quadratic Gaussian (iLQG). These works, however, concentrate on improving the solver's efficiency, and not many works actually use the probabilistic solution, except in Guided Policy Search (GPS) where the probabilistic distribution of the iLQR solution is used to provide off-policy samples for reinforcement learning [12].

In this work, we propose to use the probability distribution in the context of a tracking controller. Instead of tracking only the optimal solution, we use a short horizon Model Predictive Control (MPC) to track the trajectory distribution. We show that it improves the tracking performance significantly, with lower cost and better stability. Given a disturbance, a controller tracking only the mean will require the controller to react stiffly to perturbations in all directions, while a controller tracking the distribution can react more intelligently, as it knows in which direction it can move without increasing the cost function too much.

In short, our contribution is twofold. First, we show how to obtain a probability distribution of iLQR solution as a Gaussian distribution using the terms available from a standard iLQR solver. Then we propose a tracking strategy with adaptive gains to follow this distribution using a short time horizon MPC controller, and show that it improves the tracking performance in terms of the total cost and stability, as compared to tracking only the mean.

The outline of the paper is as follows. In Section II we give some background on the connection between quadratic costs and Gaussian distribution, and how this relates to finding the probability distribution of LQR. In Section III, we extend this approach to iLQR and show how to track the resulting distribution using a short time horizon MPC. In Section IV, we evaluate the method qualitatively on two different systems: manipulator and quadcopter moving around obstacles, and compare the proposed controller against baselines. Section V concludes the paper.

II. Background

A. *Optimal Control Problem (OCP)*

A general discrete OCP consists of a cost function

$$C(\boldsymbol{x},\boldsymbol{u}) = \sum_{t=0}^{T-1} c_t(\boldsymbol{x}_t,\boldsymbol{u}_t) + c_T(\boldsymbol{x}_T,\boldsymbol{u}_T), \tag{1}$$

subject to the dynamics

$$\boldsymbol{x}_{t+1} = \boldsymbol{f}(\boldsymbol{x}_t,\boldsymbol{u}_t). \tag{2}$$

OCP may also have equality and inequality constraints. The objective of solving OCP is to find the sequence of state and control trajectories $(\boldsymbol{x}^*,\boldsymbol{u}^*)$ that minimizes the cost function while respecting the dynamics. In robotics, given the system's high degree of freedom and complexity, most researchers rely on numerical optimization to solve the problem. One of the popular methods is iterative LQR [9].

B. *Quadratic Cost and Product of Gaussians*

A quadratic cost can be viewed probabilistically as corresponding to a Gaussian distribution. Given the quadratic cost

$$C(\boldsymbol{x}) = (\boldsymbol{x}-\bar{\boldsymbol{x}})^\top \boldsymbol{W}(\boldsymbol{x}-\bar{\boldsymbol{x}}), \tag{3}$$

the optimal solution $\boldsymbol{x}^* = \bar{\boldsymbol{x}}$ does not contain much information about the cost function itself. Instead, we can view $\boldsymbol{x}$ as a random variable with a Gaussian probability, i.e., $p(\boldsymbol{x}) = \mathcal{N}(\bar{\boldsymbol{x}}, \boldsymbol{W}^{-1})$ where $\bar{\boldsymbol{x}}$ and $\boldsymbol{W}^{-1}$ are the mean and the covariance of the Gaussian, respectively. The negative log-likelihood of this Gaussian distribution is equivalent to (3) up to a constant factor. According to $p(\boldsymbol{x})$, $\boldsymbol{x}$ has the highest probability at $\bar{\boldsymbol{x}}$, and $\boldsymbol{W}^{-1}$ gives the directional information on how this probability changes as we move away from $\bar{\boldsymbol{x}}$. The point having the lowest cost in (3) is therefore associated with the point having the highest probability.

Similarly, an objective function composed of several quadratic terms

$$\hat{\boldsymbol{\mu}} = \arg\min_{\boldsymbol{x}} \sum_{k=1}^{K} (\boldsymbol{x}-\boldsymbol{\mu}_k)^\top \boldsymbol{W}_k (\boldsymbol{x}-\boldsymbol{\mu}_k) \tag{4}$$

can be seen as a product of Gaussians $\prod_{k=1}^{K} \mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{W}_k^{-1})$, with centers $\boldsymbol{\mu}_k$ and covariance matrices $\boldsymbol{W}_k^{-1}$. The Gaussian $\mathcal{N}(\hat{\boldsymbol{\mu}}, \hat{\boldsymbol{W}}^{-1})$ resulting from this product has parameters

$$\hat{\boldsymbol{\mu}} = \left(\sum_{k=1}^{K} \boldsymbol{W}_k\right)^{-1} \left(\sum_{k=1}^{K} \boldsymbol{W}_k \boldsymbol{\mu}_k\right), \quad \hat{\boldsymbol{W}} = \sum_{k=1}^{K} \boldsymbol{W}_k.$$

$\hat{\boldsymbol{\mu}}$ and $\hat{\boldsymbol{W}}$ are the same as the solution of (4) and its Hessian, respectively. Viewing the quadratic cost probabilistically allows us to capture more information about the cost function in the form of the covariance matrix $\hat{\boldsymbol{W}}^{-1}$.

C. *Probabilistic Solution of Time-Varying Finite Horizon Linear Quadratic Regulator (LQR)*

A time-varying finite horizon LQR problem is a subclass of OCP with time-varying linear dynamics

$$\boldsymbol{x}_{t+1} = \boldsymbol{A}_t \boldsymbol{x}_t + \boldsymbol{B}_t \boldsymbol{u}_t,$$

and quadratic costs

$$C(\boldsymbol{x},\boldsymbol{u}) = \sum_{t=0}^{T-1} (\boldsymbol{x}_t^\top \boldsymbol{Q}_t \boldsymbol{x}_t + \boldsymbol{u}_t^\top \boldsymbol{R}_t \boldsymbol{u}_t) + \boldsymbol{x}_T^\top \boldsymbol{Q}_T \boldsymbol{x}_T.$$

For such class of problems, the solution can be obtained analytically. We focus here on the batch least-squares solution of LQR. Each $\boldsymbol{x}_t$ can be written in terms of $\boldsymbol{x}_0$,

$$\boldsymbol{x}_1 = \boldsymbol{A}_0 \boldsymbol{x}_0 + \boldsymbol{B}_0 \boldsymbol{u}_0,$$
$$\boldsymbol{x}_2 = \boldsymbol{A}_1 \boldsymbol{x}_1 + \boldsymbol{B}_1 \boldsymbol{u}_1 = \boldsymbol{A}_1 \boldsymbol{A}_0 \boldsymbol{x}_0 + \boldsymbol{A}_1 \boldsymbol{B}_0 \boldsymbol{u}_0 + \boldsymbol{B}_1 \boldsymbol{u}_1,$$

and so on until $\boldsymbol{x}_T$. We can then stack all $\boldsymbol{x}_t$ and $\boldsymbol{u}_t$ and get the batch equation

$$\boldsymbol{x} = \boldsymbol{S}_x \boldsymbol{x}_0 + \boldsymbol{S}_u \boldsymbol{u}, \tag{5}$$

where $\boldsymbol{x} = (\boldsymbol{x}_0^\top, \boldsymbol{x}_1^\top, \cdots, \boldsymbol{x}_T^\top)^\top$, $\boldsymbol{u} = (\boldsymbol{u}_0^\top, \boldsymbol{u}_1^\top, \cdots, \boldsymbol{u}_{T-1}^\top)^\top$, and

$$\boldsymbol{S_x} = \begin{bmatrix} \boldsymbol{I} \\ \boldsymbol{A}_0 \\ \boldsymbol{A}_1 \boldsymbol{A}_0 \\ \vdots \\ \prod_{t=0}^{T-1} \boldsymbol{A}_{T-t} \end{bmatrix}, \boldsymbol{S_u} = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{B}_0 & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{A}_1 \boldsymbol{B}_0 & \boldsymbol{B}_1 & \cdots & \boldsymbol{0} \\ \vdots & \vdots & \ddots & \vdots \\ \prod_{t=1}^{T-1} \boldsymbol{A}_{T-t} \boldsymbol{B}_0 & \cdots & \cdots & \boldsymbol{B}_{T-1} \end{bmatrix}.$$

We can then write the cost function as

$$C(\boldsymbol{x},\boldsymbol{u}) = \boldsymbol{x}^\top \boldsymbol{Q}_s \boldsymbol{x} + \boldsymbol{u}^\top \boldsymbol{R}_s \boldsymbol{u}, \tag{6}$$

where $\boldsymbol{Q}_s = \text{blockdiag}(\boldsymbol{Q}_0, \boldsymbol{Q}_1, \ldots, \boldsymbol{Q}_T)$ and $\boldsymbol{R}_s = \text{blockdiag}(\boldsymbol{R}_0, \boldsymbol{R}_1, \ldots, \boldsymbol{R}_{T-1})$ are block diagonal matrices. Substituting (5) to (6), we obtain

$$\begin{aligned} C(\boldsymbol{x},\boldsymbol{u}) =& \boldsymbol{x}^\top \boldsymbol{Q}_s \boldsymbol{x} + \boldsymbol{u}^\top \boldsymbol{R}_s \boldsymbol{u} \\ =& (\boldsymbol{S}_x \boldsymbol{x}_0 + \boldsymbol{S}_u \boldsymbol{u})^\top \boldsymbol{Q}_s (\boldsymbol{S}_x \boldsymbol{x}_0 + \boldsymbol{S}_u \boldsymbol{u}) + \boldsymbol{u}^\top \boldsymbol{R}_s \boldsymbol{u} \\ =& \boldsymbol{u}^\top (\boldsymbol{S}_u^\top \boldsymbol{Q}_s \boldsymbol{S}_u + \boldsymbol{R}_s) \boldsymbol{u} + 2 \boldsymbol{u}^\top \boldsymbol{S}_u^\top \boldsymbol{Q}_s \boldsymbol{S}_x \boldsymbol{x}_0 \\ &+ \boldsymbol{x}_0^\top \boldsymbol{S}_x^\top \boldsymbol{Q}_s \boldsymbol{S}_x \boldsymbol{x}_0. \end{aligned} \tag{7}$$

Note that the cost is quadratic in $\boldsymbol{u}$. As discussed in Section II-B, we can view this probabilistically and obtain the probability distribution of $\boldsymbol{u}$ as a Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{u}}, \boldsymbol{\Sigma}_{\boldsymbol{u}})$, where

$$\boldsymbol{\mu}_{\boldsymbol{u}} = -(\boldsymbol{S}_{\boldsymbol{u}}^\top \boldsymbol{Q}_s \boldsymbol{S}_{\boldsymbol{u}} + \boldsymbol{R}_s)^{-1} \boldsymbol{S}_{\boldsymbol{u}}^\top \boldsymbol{Q}_s \boldsymbol{S}_{\boldsymbol{x}} \boldsymbol{x}_0, \quad (8)$$
$$\boldsymbol{\Sigma}_{\boldsymbol{u}} = (\boldsymbol{S}_{\boldsymbol{u}}^\top \boldsymbol{Q}_s \boldsymbol{S}_{\boldsymbol{u}} + \boldsymbol{R}_s)^{-1}. \quad (9)$$

The mean is obtained as the optimum of the cost function in (7) by setting the gradient equal to zero, while the covariance matrix is the inverse of the cost function's Hessian.

This distribution tells us that $\boldsymbol{u}$ has the highest probability at the mean $\boldsymbol{\mu}_{\boldsymbol{u}}$ (corresponding to the point with the lowest cost), and $\boldsymbol{\Sigma}_{\boldsymbol{u}}$ explains how the probability changes along any direction. With this distribution, we know how to move away from the mean while avoiding significant increase in the cost function. We can also obtain the distribution of the state trajectory, since $\boldsymbol{x}$ is a linear transformation of $\boldsymbol{u}$ according to (5), namely

$$p(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x} | \boldsymbol{S}_{\boldsymbol{x}} \boldsymbol{x}_0 + \boldsymbol{S}_{\boldsymbol{u}} \boldsymbol{\mu}_{\boldsymbol{u}}, \boldsymbol{S}_{\boldsymbol{u}} \boldsymbol{\Sigma}_{\boldsymbol{u}}^{-1} \boldsymbol{S}_{\boldsymbol{u}}^\top). \quad (10)$$

More details about the probability distribution of LQR can be found in [13], [14].

III. Method

In this section, we first describe how to obtain the solution of an iLQR problem as a Gaussian distribution. The key insight is to note that each step of iLQR solves an LQR subproblem, of which we can find the probability distribution of the solution. We then show how to track this distribution using a short time horizon MPC.

A. Probabilistic Solution of iLQR

1) iLQR solution: iLQR can be used to solve more general problems than LQR involving non-quadratic cost functions and nonlinear dynamics. Starting from an initial guess $(\boldsymbol{x}_0, \boldsymbol{u}_0)$, iLQR iteratively refines this guess by making a simpler approximation of the OCP at each step. Let us consider the current guess $(\boldsymbol{x}^k, \boldsymbol{u}^k)$, where k is the iteration index. Given the general cost function in (1), we can approximate it as a quadratic function around $(\boldsymbol{x}^k, \boldsymbol{u}^k)$,

$$c_t(\delta\boldsymbol{x}_t, \delta\boldsymbol{u}_t) = \frac{1}{2} \begin{bmatrix} \delta\boldsymbol{x}_t \\ \delta\boldsymbol{u}_t \end{bmatrix}^\top \begin{bmatrix} \boldsymbol{c}_{\boldsymbol{xx},t} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{c}_{\boldsymbol{uu},t} \end{bmatrix} \begin{bmatrix} \delta\boldsymbol{x}_t \\ \delta\boldsymbol{u}_t \end{bmatrix} + \begin{bmatrix} \boldsymbol{c}_{\boldsymbol{x},t} & \boldsymbol{c}_{\boldsymbol{u},t} \end{bmatrix} \begin{bmatrix} \delta\boldsymbol{x}_t \\ \delta\boldsymbol{u}_t \end{bmatrix}, \quad (11)$$

where $\boldsymbol{c}_{\boldsymbol{x},t}, \boldsymbol{c}_{\boldsymbol{u},t}, \boldsymbol{c}_{\boldsymbol{xx},t},$ and $\boldsymbol{c}_{\boldsymbol{uu},t}$ are the cost function's first and second order derivatives with respect to $\boldsymbol{x}$ and $\boldsymbol{u}$. We omit here the cross-derivatives $\boldsymbol{c}_{\boldsymbol{xu},t}$ for simplifying the derivations and the notations, but a similar derivation works when $\boldsymbol{c}_{\boldsymbol{xu},t}$ is not zero.

Similarly, we can approximate the dynamics in (2) using the linear approximation

$$\delta\boldsymbol{x}_t = \boldsymbol{A}_t \delta\boldsymbol{x}_t + \boldsymbol{B}_t \delta\boldsymbol{u}_t, \quad (12)$$

where $\boldsymbol{A}_t$ and $\boldsymbol{B}_t$ are the derivatives of the dynamics $\boldsymbol{f}(\boldsymbol{x}_t, \boldsymbol{u}_t)$ with respect to $\boldsymbol{x}_t$ and $\boldsymbol{u}_t$, respectively. The derivatives are evaluated at the current guess $(\boldsymbol{x}^k, \boldsymbol{u}^k)$. If the dynamics is approximated as quadratic instead of linear, it is referred to as Differential Dynamic Programming (DDP) [15].

At this stage, we have quadratic costs and linear dynamics as functions of $\delta\boldsymbol{x}$ and $\delta\boldsymbol{u}$. This is therefore a time-varying LQR problem, of which the variables of interest are $\delta\boldsymbol{x}$ and $\delta\boldsymbol{u}$. We can solve this either by the batch least-squares solution or dynamic programming, and obtain the optimal $\delta\boldsymbol{x}^*$ and $\delta\boldsymbol{u}^*$. Although in standard LQR the two methods give the same outputs, in iLQR they will be different due to the dynamics rollout step, i.e. the forward pass. When solving the LQR subproblem using dynamic programming, at each time step we calculate the resulting $\boldsymbol{u}_t$, and the forward pass is calculated using the actual nonlinear dynamics. In the batch least-squares solution, on the other hand, the forward pass is calculated using the approximated linear dynamics, so the two will give slightly different solutions.

The $\delta\boldsymbol{u}$ calculated by solving the LQR subproblem is a directional step to improve the current guess $(\boldsymbol{x}^k, \boldsymbol{u}^k)$. Typically, a line search is performed to find the optimum step length to move in this direction (see [7]). With the given optimum step length α we calculate the new $\boldsymbol{u}$ as $\boldsymbol{u}^{k+1} = \boldsymbol{u}^k + \alpha\delta\boldsymbol{u}^k$. By performing dynamics rollout using this new $\boldsymbol{u}^{k+1}$ we obtain the new state trajectory $\boldsymbol{x}^{k+1}$. The line search guarantees that $(\boldsymbol{x}^{k+1}, \boldsymbol{u}^{k+1})$ has lower cost than the previous guess. We can then make another approximation around the new guess to obtain a new LQR problem and improve the solution. This is iterated until convergence. Besides obtaining the optimal solution $(\boldsymbol{x}^*, \boldsymbol{u}^*)$, we also obtain the time-dependent feedback gain $\boldsymbol{K}_t$ to be used for feedback control in the proximity of $(\boldsymbol{x}^*, \boldsymbol{u}^*)$.

2) Obtaining the iLQR distribution: Let us assume that we have reached convergence and obtain the optimal solution as $(\boldsymbol{x}^*, \boldsymbol{u}^*)$. If we again approximate the cost function and the dynamics to obtain a new LQR subproblem around this solution, its solution would be $\delta\boldsymbol{u}^* = \boldsymbol{0}$, because the optimization gradient is zero at local optima, and the cost function cannot be reduced further. However, as discussed in Section II-B, the optimal solution $\delta\boldsymbol{u}^* = \boldsymbol{0}$ does not contain much information about the cost function and the underlying optimization problem. Since this is an LQR problem, we can obtain not only the optimal solution but also the distribution of the solution, as discussed in Section II-C.

We consider the quadratic cost functions and the linear dynamics at the final iteration $k = K$. As explained in Section II-C, we can compute the probability distribution of $\delta\boldsymbol{u}$ as $p(\delta\boldsymbol{u}) = \mathcal{N}(\boldsymbol{0}, \Sigma_{\delta\boldsymbol{u}^*})$, where $\Sigma_{\delta\boldsymbol{u}^*}$ is calculated using (9). The precision matrices $\boldsymbol{Q}_s$ and $\boldsymbol{R}_s$ are computed from the cost derivatives, i.e.

$$\boldsymbol{Q}_s = \text{blockdiag}(\boldsymbol{c}_{\boldsymbol{xx},0}, \boldsymbol{c}_{\boldsymbol{xx},1}, \cdots, \boldsymbol{c}_{\boldsymbol{xx},T}),$$
$$\boldsymbol{R}_s = \text{blockdiag}(\boldsymbol{c}_{\boldsymbol{uu},0}, \boldsymbol{c}_{\boldsymbol{uu},1}, \cdots, \boldsymbol{c}_{\boldsymbol{uu},T-1}).$$

Now the mean of this $p(\delta\boldsymbol{u})$ is a zero vector, corresponding to an optimal solution at convergence. The covariance $\boldsymbol{\Sigma}_{\delta\boldsymbol{u}^*}$ tells us how to move away from the mean while keeping a low cost. Since $\boldsymbol{u} = \boldsymbol{u}^* + \delta\boldsymbol{u}$, we obtain $p(\boldsymbol{u}) = \mathcal{N}(\boldsymbol{u}^*, \boldsymbol{\Sigma}_{\boldsymbol{u}^*})$ where $\boldsymbol{\Sigma}_{\boldsymbol{u}^*} = \boldsymbol{\Sigma}_{\delta\boldsymbol{u}^*}$. That is, the distribution of $\boldsymbol{u}$ is centered around the optimal solution $\boldsymbol{u}^*$, with the same covariance as $\delta\boldsymbol{u}$.

Since $\delta\boldsymbol{x} = \boldsymbol{S}_{\boldsymbol{x}}\delta\boldsymbol{x}_0 + \boldsymbol{S}_{\boldsymbol{u}}\delta\boldsymbol{u}$, the probability distribution of $\delta\boldsymbol{x}$ can be computed as

$$p(\delta\boldsymbol{x}) = \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma}_{\delta\boldsymbol{x}^*}), \quad \text{with} \quad \boldsymbol{\Sigma}_{\delta\boldsymbol{x}^*} = \boldsymbol{S}_{\boldsymbol{u}}\boldsymbol{\Sigma}_{\delta\boldsymbol{u}^*}\boldsymbol{S}_{\boldsymbol{u}}^\top. \quad (13)$$

Furthermore, $\boldsymbol{x} = \boldsymbol{x}^* + \delta\boldsymbol{x}$, so $p(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}^*, \boldsymbol{\Sigma}_{\boldsymbol{x}^*})$, where $\boldsymbol{\Sigma}_{\boldsymbol{x}^*} = \boldsymbol{\Sigma}_{\delta\boldsymbol{x}^*}$. Note that the probability distribution is computed based on the terms that are available from a standard iLQR solver, i.e. the derivatives of the costs and the dynamics. Indeed, we can just run a standard solver until convergence, and extract all the dynamics and cost derivatives to construct the trajectory distribution $\mathcal{N}(\boldsymbol{x}^*, \boldsymbol{\Sigma}_{\boldsymbol{x}^*})$.

The probability distribution is obtained by approximating the cost function and the dynamics. Technically, a Gaussian distribution propagated by nonlinear dynamics would not remain as Gaussian distribution. We obtain a final Gaussian distribution of the overall trajectory because of the linear approximation of the dynamics at each iLQR step, similar to what is done in Extended Kalman Filter [16]. Therefore, this distribution only holds locally near the optimal solution $(\boldsymbol{x}^*, \boldsymbol{u}^*)$. Nevertheless, it still contains important information on the local behavior of the cost function and the dynamics around this optimal solution. We will show that this information will be beneficial for the next step, i.e., tracking the optimal trajectory.

B. Tracking Distribution using Short Time Horizon MPC

From the previous section, we obtain the probability distribution of the state trajectory $p(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}^*, \boldsymbol{\Sigma}_{\boldsymbol{x}^*})$. Most optimal control approaches only consider the optimal solution $\boldsymbol{x}^*$. As discussed in Section I, the OCP is often an intermediary step to generate a trajectory, which is then tracked by using another controller. This controller can be a simple PID controller or a short time horizon MPC [8] that tracks $\boldsymbol{x}^*$. We argue that tracking only the optimal solution is suboptimal because it does not contain sufficient information about the underlying cost functions. When the system faces disturbances, the controller will force the system to go back to the optimal solution, although the disturbance may be acceptable according to the desired behavior.

Consider as an example the problem of controlling a bicopter to reach a goal position. The mean and the samples from the trajectory distribution are shown in Fig. 2c. The main objective is the position of the bicopter at the final time step to be at the goal. This results in a wide trajectory distribution in-between, signifying that it is acceptable to deviate from the mean in the middle of the trajectory. When there is a disturbance, a controller tracking the mean will force the system to come back to the mean, although this is not necessary according to our cost function (which reflects the desired behavior). In contrast, if the controller knows about the distribution, it knows when a disturbance is acceptable and hence does not apply strong correction, following a *minimal intervention principle* [13], [17], [18].

To track the distribution $p(\boldsymbol{x})$, we propose to use a short time horizon MPC. At each time step, we solve an OCP with horizon T_s, which is much shorter than the long horizon T used to plan the trajectory in Section III-A. We solve this short horizon OCP with iLQR just as we do for the long horizon OCP, but in practice we can use any OCP solver for this part. The cost function

$$c_t(\boldsymbol{x}_t, \boldsymbol{u}_t) = (\boldsymbol{x}_t - \bar{\boldsymbol{x}}_t)^\top \boldsymbol{Q}_t(\boldsymbol{x}_t - \bar{\boldsymbol{x}}_t) + l(\boldsymbol{x}_t, \boldsymbol{u}_t) + \boldsymbol{u}_t^\top \boldsymbol{R}\boldsymbol{u}_t \quad (14)$$

is designed to track a reference trajectory, where $\bar{\boldsymbol{x}}_t$ is the reference state at time t, $l(\boldsymbol{x}_t, \boldsymbol{u}_t)$ is the collision cost, and the precision matrices $\boldsymbol{Q}_t$ are designed to correspond to the probability distribution of $\bar{\boldsymbol{x}}_t$. If $\bar{\boldsymbol{x}}_t$ has a large variance, we do not want to track this state too precisely, so $\boldsymbol{Q}_t$ should be small, and vice-versa.

How to relate $\bar{\boldsymbol{x}}_t$ and $\boldsymbol{Q}_t$ to the trajectory distribution $\mathcal{N}(\boldsymbol{x}^*, \boldsymbol{\Sigma}_{\boldsymbol{x}^*})$ that we find earlier? One way is to use the marginal distribution $p_m(\boldsymbol{x}_t) = \mathcal{N}(\boldsymbol{x}_t^*, \boldsymbol{\Sigma}_{\boldsymbol{x}^*,t})$, where $\boldsymbol{x}_t^*$ is the component of $\boldsymbol{x}^*$ at time t, and $\boldsymbol{\Sigma}_{\boldsymbol{x}^*,t}$ is the corresponding block matrix, so that $\bar{\boldsymbol{x}}_t = \boldsymbol{x}_t^*$ and $\boldsymbol{Q}_t = \boldsymbol{\Sigma}_{\boldsymbol{x}^*,t}^{-1}$. However, when we do this, we neglect the correlation between the different time steps in $\boldsymbol{\Sigma}_{\boldsymbol{x}^*}$. Instead, we can use the conditional distribution based on the current state.

At time step $t = \tau$, we observe the current state at $\boldsymbol{x}_\tau$. We first extract the probability $p(\boldsymbol{x}_{\tau:\tau+T_s})$ from $p(\boldsymbol{x})$, and write this in partition format

$$p\begin{pmatrix} \boldsymbol{x}_\tau \\ \boldsymbol{x}_{\tau+1:\tau+T_s} \end{pmatrix} = \mathcal{N}\left(\begin{pmatrix} \boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2 \end{pmatrix}, \begin{pmatrix} \boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{12} \\ \boldsymbol{\Sigma}_{21} & \boldsymbol{\Sigma}_{22} \end{pmatrix}\right).$$

We can then condition on $\boldsymbol{x}_\tau$ to obtain $p_c(\boldsymbol{x}_{\tau+1:\tau+T_s}|\boldsymbol{x}_\tau) = \mathcal{N}(\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)$ where

$$\boldsymbol{\mu}_c = \boldsymbol{\mu}_2 + \boldsymbol{\Sigma}_{21}\boldsymbol{\Sigma}_{11}(\boldsymbol{x}_\tau - \boldsymbol{\mu}_1), \quad \boldsymbol{\Sigma}_c = \boldsymbol{\Sigma}_{22} - \boldsymbol{\Sigma}_{21}\boldsymbol{\Sigma}_{11}^{-1}\boldsymbol{\Sigma}_{12}.$$

The conditional distribution $p_c(\boldsymbol{x}_t|\boldsymbol{x}_\tau) = \mathcal{N}(\boldsymbol{\mu}_{c,t}, \boldsymbol{\Sigma}_{c,t})$ can then be obtained from $p_c(\boldsymbol{x}_{\tau+1:\tau+T_s}|\boldsymbol{x}_\tau)$ for $t = \tau + 1$ to $\tau + T_s$. $\boldsymbol{\mu}_{c,t}$ is the t_{th} element of $\boldsymbol{\mu}_c$, with the corresponding block diagonal $\boldsymbol{\Sigma}_{c,t}$. This can be used to set $\bar{\boldsymbol{x}}_t = \boldsymbol{\mu}_{c,t}$ and $\boldsymbol{Q}_t = \boldsymbol{\Sigma}_{c,t}^{-1}$ in (14).

We demonstrate in the next section that formulating the tracking controller to follow the reference trajectory distribution will improve the tracking performance. The complete algorithm is given in Table 1.

It is also possible to use the long horizon OCP in MPC fashion. However, the long horizon requires longer computational time, making it difficult to be used in real-time. The computation time is linear with respect to the time horizon T. Formulating the controller as a short time horizon MPC to track the distribution speeds up the computational time significantly, while still being able to consider the future events using the distribution. With shorter computational time, the short horizon MPC can better adapt to real time changes such as new obstacles along the trajectory. We can also add hard constraints to the OCP, e.g., using the augmented Lagrangian method [19].

Algorithm 1 Tracking iLQR distribution

```
1:  Solve the long horizon ($T$) OCP by iLQR;
2:  Calculate $p(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}^*, \boldsymbol{\Sigma}_{\boldsymbol{x}^*})$;
3:  for $t = \tau < T$ do
4:      Estimate the current state $\boldsymbol{x}_\tau$;
5:      Extract the probability $p(\boldsymbol{x}_{\tau:\tau+T_s})$ from $p(\boldsymbol{x})$;
6:      Compute the conditional probability $p_c(\boldsymbol{x}_{\tau+1:\tau+T_s}|\boldsymbol{x}_\tau)$;
7:      Extract the ref. traj. distribution $p_c(\boldsymbol{x}_t|\boldsymbol{x}_\tau)$ from
        $p_c(\boldsymbol{x}_{\tau+1:\tau+T_s}|\boldsymbol{x}_\tau)$ for $t = \tau+1$ to $\tau+T_s$;
8:      Construct the cost function of the short horizon ($T_s$) OCP
        (Eq. 14);
9:      Solve the short horizon ($T_s$) OCP using iLQR;
10:     Execute the first control input $\boldsymbol{u}_t$;
11: end for
```

IV. Experiments

Fig. 2 shows the probability distribution on several systems, i.e., inverted pendulum (1 DoF), unicycle (3 DoF), and bicopter (3 DoF). The system description can be found in [17], [15], [20]. The distribution is determined by both the dynamics and the cost function. As we put a high cost at $t = T$ to reach the goal and low cost at $t < T$, the distribution is narrow around the goal and quite wide in the middle. Thus the tracking controller knows that deviation from the mean is more tolerable in the middle of the trajectory.

We quantitatively evaluated the proposed algorithm on robotic manipulator (Panda) and quadcopter, with the task to move from an initial configuration to a goal location while avoiding obstacles. Following Algorithm 1, a long horizon iLQR is first solved to obtain the trajectory distribution $p(\boldsymbol{x})$ that reaches the goal. The cost function is defined as

$$C(\boldsymbol{x}, \boldsymbol{u}) = \sum_{t=0}^{T-1} \Big((\boldsymbol{x}_t - \boldsymbol{x}_{\text{goal}})^\top \boldsymbol{Q} (\boldsymbol{x}_t - \boldsymbol{x}_{\text{goal}}) + \boldsymbol{u}_t^\top \boldsymbol{R} \boldsymbol{u}_t + l(\boldsymbol{x}_t, \boldsymbol{u}_t) \Big) + (\boldsymbol{x}_T - \boldsymbol{x}_{\text{goal}})^\top \boldsymbol{Q}_T (\boldsymbol{x}_T - \boldsymbol{x}_{\text{goal}}), \quad (15)$$

where $\boldsymbol{Q}$ and $\boldsymbol{Q}_T$ are the precision matrices, $\boldsymbol{Q}$ being much smaller than $\boldsymbol{Q}_T$. $l(\boldsymbol{x}_t, \boldsymbol{u}_t)$ is a collision avoidance cost formulated as a potential field that is active only when the quadcopter is in collision. After running the iLQR solver until convergence, we can compute the state trajectory distribution $p(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}^*, \boldsymbol{\Sigma}_{\boldsymbol{x}^*})$. Section IV-A describes various tracking algorithms to be compared, with the results discussed in Section IV-B.

A. Tracking Algorithms

To illustrate the benefit of tracking the trajectory distribution, we compare four different algorithms:

1) iLQR Feedback Control (iLQR$_{feed}$): iLQR$_{\text{feed}}$ tracks the mean trajectory $\boldsymbol{x}^*$ using the feedforward control input $\boldsymbol{u}^*$ and the feedback term $\boldsymbol{K}_t$,

$$\boldsymbol{u}_t = \boldsymbol{u}_t^* + \boldsymbol{K}_t(\boldsymbol{x}_t - \boldsymbol{x}_t^*),$$

where $\boldsymbol{x}_t$ is the current observed state. While this requires very little computation, the feedback gain is only good around the planned trajectory $(\boldsymbol{x}^*, \boldsymbol{u}^*)$, and it can be unstable for large disturbance.

2) Short time horizon MPC tracking the mean trajectory (MPC$_{mean}$): MPC$_{\text{mean}}$ solves an OCP problem at each time step with the cost function in (14). The reference state is obtained from the planned trajectory, i.e., $\bar{\boldsymbol{x}}_t = \boldsymbol{x}_t^*$, whereas the precision matrix $\boldsymbol{Q}_t$ is set to be the same as $\boldsymbol{Q}_T$ in (15), i.e., a high-gain tracking.

3) Short time horizon MPC tracking the marginal trajectory distribution (MPC$_{marg}$): MPC$_{\text{marg}}$ solves an OCP problem at each time step with the cost function in (14). $\bar{\boldsymbol{x}}_t$ and $\boldsymbol{Q}_t$ are set according to the marginal distribution $p_m(\boldsymbol{x}_t)$ as described in Section III-B.

4) Short time horizon MPC tracking the conditional trajectory distribution (MPC$_{cond}$): MPC$_{\text{cond}}$ solves an OCP problem at each time step with the cost function in (14). $\bar{\boldsymbol{x}}_t$ and $\boldsymbol{Q}_t$ are set according to the conditional distribution $p_c(\boldsymbol{x}_t|\boldsymbol{x}_\tau)$ as described in Section III-B.

B. Tracking Comparison

We run the experiments on two systems, the 7-DoF Panda manipulator and 6-DoF quadcopter. The manipulator task is kinematic control to reach a desired end-effector pose where the state $\boldsymbol{x} \in \mathbb{R}^{14}$ consists of the joint angles and velocities, and the control $\boldsymbol{u} \in \mathbb{R}^7$ is the joint acceleration command. The dynamics of the system is therefore linear (i.e., double integrator), but the cost is non-quadratic as it involves the end effector pose. The quadcopter task is dynamic control to reach a desired goal location where the state $\boldsymbol{x} \in \mathbb{R}^{12}$ consists of the position, orientation, and its corresponding velocities, while the control $\boldsymbol{u} \in \mathbb{R}^4$ consists of the four propellers thrusts. For both systems, the horizon is set to be $T = 150$ and $T_s = 30$ for the long and short horizon, respectively, with 50 ms interval. The long horizon is set to be long enough to reach the goal at the end of the trajectory, while the short horizon is set to be as short as possible while still managing to obtain good performance. As the computation time of DDP is linear with respect to the horizon length, the iteration time for short horizon DDP is around 5 times faster than the long horizon DDP. The cost function is defined in (15). Fig. 1 shows an example of the optimal iLQR trajectory $\boldsymbol{x}^*$ for the quadcopter, shown in white.

After solving the long horizon iLQR to obtain the trajectory distribution $p(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}^*, \boldsymbol{\Sigma}_{\boldsymbol{x}^*})$, we track the trajectory using the algorithms in Section IV-A. During tracking we introduce external velocity disturbance to the system, and evaluate how well the algorithms overcome the disturbance with varying magnitude. We consider two types of disturbance: *impulse* and *time-varying* disturbance, each of which has three levels of disturbance: *small, medium, and large*. For the impulse disturbance, we introduce external velocity disturbance for a short time, i.e., during 2 time steps, with the magnitude of $(0.5, 1.5, 3.)$ for manipulator and $(0.2, 0.7, 1.5)$ for quadcopter (the units are m/s and rad/s for linear and angular velocity, respectively) in random direction. For the time-varying disturbance, we introduce time-varying velocity disturbance between $t = 30$ and $t = 100$ with the magnitude of $(0.1, 0.3, .6)$ for manipulator and $(0.07, 0.2, 0.4)$ for quadcopter. These numbers are chosen to specifically

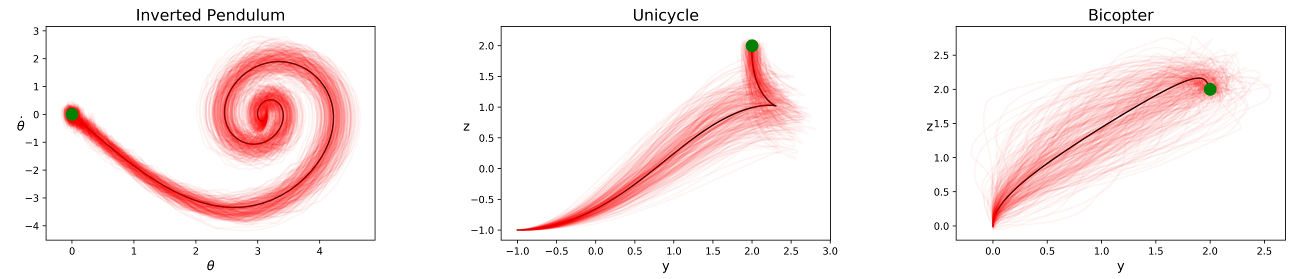

Fig. 2. Trajectory distribution for different systems, shown at selected axes. The mean trajectory is shown as a black line, and the trajectory samples drawn from the distribution are shown as red thin lines. The goal is shown in green.

demonstrate the different relative performance among the controllers at each level of disturbance, as will be discussed shortly after this. After each task completion, we evaluate the cost of the resulting state and control trajectories by using the original cost function in (15). For each disturbance level, we run N=50 experiments with disturbance in random direction. As the costs vary greatly between each experiment, we normalize the cost by the minimum cost achieved at each experiment, so a cost of value 1.0 means that the method achieves the best cost for that particular experiment. The mean and standard deviation of the normalized cost for each method is given in Table I and II.

The results from both tables give the same conclusions, i.e., the relative performance of the controllers are the same for both impulse and time-varying disturbance. For the Panda experiment, iLQR$_{\text{feed}}$ has the best performance for all disturbance levels. However, it only performs best at small disturbance for the quadcopter. As the disturbance increases, iLQR$_{\text{feed}}$ becomes less reliable and the cost increases. We observe that iLQR$_{\text{feed}}$ often becomes very unstable at large disturbance, resulting in diverging movements (these samples were not considered in the results of Table I and II). Large disturbance move the system far from the planned trajectory where the feedback gain is no longer valid. This is especially true for underactuated systems such as bicopter and quadcopter where each control is not directly associated with a particular state, because the optimal feedback gain changes its sign (not only magnitude) according to the current system state. Note that the standard deviation of iLQR$_{\text{feed}}$ cost at large disturbance in Table I and II for quadcopter is very high. On the other hand, the kinematic control of manipulator involves a linear dynamic and fully actuated system, so the feedback gain remains good at large disturbance.

The three remaining controllers are more stable even at large disturbance. MPC$_{\text{mean}}$ has the largest cost among the three because it tries to track the mean trajectory precisely without knowing the underlying cost function and the desired behavior. MPC$_{\text{marg}}$ performs better because it takes into account the variance of the planned trajectory, but it ignores the correlations between different time steps. The smallest cost is achieved by MPC$_{\text{cond}}$ because it computes the future reference trajectory by considering the current state, i.e., by computing the conditional distribution. This enables the controller to adapt to the disturbance better. In practice, it means that the controller exerts less control to overcome the disturbance while still achieving the objective.

TABLE I
TRACKING PERFORMANCE COST COMPARISON (IMPULSE DISTURBANCE)

System	**Method**	**Disturbance**		
		Small	**Medium**	**Large**
Panda	iLQR$_{\text{feed}}$	**1.00 ± 0.0**	**1.00 ± 0.0**	**1.00 ± 0.0**
	MPC$_{\text{mean}}$	1.02 ± 0.0	1.03 ± 0.0	1.05 ± 0.0
	MPC$_{\text{marg}}$	1.02 ± 0.0	1.03 ± 0.0	1.07 ± 0.0
	MPC$_{\text{cond}}$	1.01 ± 0.0	1.01 ± 0.0	1.01 ± 0.0
Quadcopter	iLQR$_{\text{feed}}$	**1.00 ± 0.0**	1.10 ± 0.3	1.99 ± 2.9
	MPC$_{\text{mean}}$	1.48 ± 0.1	2.58 ± 1.0	3.41 ± 1.2
	MPC$_{\text{marg}}$	1.26 ± 0.0	1.23 ± 0.1	1.25 ± 0.2
	MPC$_{\text{cond}}$	1.08 ± 0.0	**1.05 ± 0.0**	**1.20 ± 0.4**

TABLE II
TRACKING PERFORMANCE COST COMPARISON (TIME-VARYING DISTURBANCE)

System	**Method**	**Disturbance**		
		Small	**Medium**	**Large**
Panda	iLQR$_{\text{feed}}$	**1.00 ± 0.00**	**1.00 ± 0.0**	**1.00 ± 0.0**
	MPC$_{\text{mean}}$	1.02 ± 0.00	1.03 ± 0.0	1.05 ± 0.0
	MPC$_{\text{marg}}$	1.02 ± 0.00	1.03 ± 0.0	1.06 ± 0.0
	MPC$_{\text{cond}}$	1.01 ± 0.00	1.01 ± 0.0	1.01 ± 0.0
Quadcopter	iLQR$_{\text{feed}}$	**1.01 ± 0.06**	1.33 ± 0.7	2.60 ± 4.4
	MPC$_{\text{mean}}$	1.54 ± 0.18	3.06 ± 1.3	4.49 ± 2.0
	MPC$_{\text{marg}}$	1.26 ± 0.03	1.25 ± 0.1	1.26 ± 0.2
	MPC$_{\text{cond}}$	1.07 ± 0.02	**1.05 ± 0.1**	**1.20 ± 0.6**

Fig. 1 illustrates the difference with the quadcopter experiment. The planned trajectory $\boldsymbol{x}^*$ is shown in white. At the current state, an upward velocity disturbance is introduced to the system. Since MPC$_{\text{mean}}$ and MPC$_{\text{marg}}$ do not consider the current state when computing the reference trajectory, the disturbance does not affect its reference trajectory (shown in cyan), which is always along the planned trajectory $\boldsymbol{x}^*$. On the other hand, the reference trajectory for MPC$_{\text{cond}}$ is computed by conditioning on the current state. The algorithm knows that the quadcopter is moving with an additional upward velocity, and it adjusted the reference trajectory accordingly (shown in yellow). While MPC$_{\text{mean}}$ and MPC$_{\text{marg}}$ force the quadcopter to go back to the mean, MPC$_{\text{cond}}$ use the information from the distribution more effectively to move according to the desired behavior. Note that the cost function in (15) dictates that the important task is to reach the goal at the end, while the path in-between is less important. This is well represented by the trajectory distribution.

However, the conditional distribution is also obtained from local approximation. This means that for very large disturbance, it can still result in undesired behavior, such as producing a reference trajectory that has a high cost. We indeed find that $\mathrm{MPC}_{\mathrm{marg}}$ is more stable than $\mathrm{MPC}_{\mathrm{cond}}$ at very large disturbance. We can see in Table I and II that the normalized cost of $\mathrm{MPC}_{\mathrm{cond}}$ increases to almost the same as $\mathrm{MPC}_{\mathrm{marg}}$ at the large disturbance level. At even larger disturbance, we observe that $\mathrm{MPC}_{\mathrm{marg}}$ performs the best and most stably. However, the local approximation does not affect $\mathrm{MPC}_{\mathrm{cond}}$ as much as $\mathrm{iLQR}_{\mathrm{feed}}$, because the conditional distribution is only used as the reference trajectory, while the underlying controller in $\mathrm{MPC}_{\mathrm{cond}}$ still considers the actual dynamics around the current state to compute the control command during the tracking. In contrast, for $\mathrm{iLQR}_{\mathrm{feed}}$, the local approximation from the planning step directly determines the controller gain, which remains unchanged during tracking. This poor approximation results in unstable controllers at large disturbance, especially for underactuated and highly nonlinear systems such as quadcopter.

C. Discussion

In this work, we show an example of kinematic control of a serial manipulator with non-quadratic cost functions. We do not use dynamic control for this system because the derivative of the dynamics (the matrix $\boldsymbol{A}_t$) is often unstable, i.e., some of its eigenvalues are outside the unit circle. Since computing the distribution involves the multiplication of the matrices $\prod_{t=1}^{T-1} \boldsymbol{A}_{T-t}$, the unstable eigenvalues causes the resulting matrix to be numerically poor and the distribution cannot be computed. More research still needs to be done to handle this issue. Future works will also consider moving obstacles scenario, as well as real robot implementation.

The proposed framework can also be extended to control systems with both state and control constraints. Augmented Lagrangian iLQR (AL-iLQR) is discussed in [19] to handle such systems. Note that each iteration in AL-iLQR still solves an LQR problem, so we can still obtain the distribution as we do here. The resulting distribution would take the constraints into consideration, i.e., having higher probability when the trajectory satisfies the constraints.

V. Conclusion and Future Work

We have shown that by obtaining the distribution of solution from iLQR and tracking this distribution, the resulting controller is more cost-efficient and robust to disturbance [1]. The tracking performance is shown to be better than tracking only the mean trajectory or using the iLQR feedback controller. The latter is very unstable when moving far from the planned trajectory due to large disturbance, especially for underactuated systems such as quadcopter. The iLQR distribution can be calculated using the information available from a standard iLQR solver. The method can also be extended to constrained OCP methods such as Augmented Lagrangian iLQR [19].

[1]The implementation codes are available at `https://github.com/teguhSL/optimal_control_distribution`

References

[1] D. R. Robinson, R. T. Mar, K. Estabridis, and G. Hewer, "An efficient algorithm for optimal trajectory generation for heterogeneous multi-agent systems in non-convex environments," *IEEE Robotics and Automation Letters (RA-L)*, vol. 3, no. 2, pp. 1215–1222, 2018.

[2] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, and H. Hirukawa, "Biped walking pattern generation by using preview control of zero-moment point," in *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, vol. 2, 2003, pp. 1620–1626.

[3] S. Caron and A. Kheddar, "Multi-contact walking pattern generation based on model preview control of 3d com accelerations," in *Proc. IEEE Intl Conf. on Humanoid Robots (Humanoids)*, 2016, pp. 550–557.

[4] B. Ponton, A. Herzog, A. Del Prete, S. Schaal, and L. Righetti, "On time optimization of centroidal momentum dynamics," in *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, 2018, pp. 1–7.

[5] A. W. Winkler, C. D. Bellicoso, M. Hutter, and J. Buchli, "Gait and trajectory optimization for legged systems through phase-based end-effector parameterization," *IEEE Robotics and Automation Letters (RA-L)*, vol. 3, no. 3, pp. 1560–1567, 2018.

[6] R. Budhiraja, J. Carpentier, C. Mastalli, and N. Mansard, "Differential dynamic programming for multi-phase rigid contact dynamics," in *Proc. IEEE Intl Conf. on Humanoid Robots (Humanoids)*, 2018.

[7] C. Mastalli, R. Budhiraja, W. Merkt, G. Saurel, B. Hammoud, M. Naveau, J. Carpentier, L. Righetti, S. Vijayakumar, and N. Mansard, "Crocoddyl: An Efficient and Versatile Framework for Multi-Contact Optimal Control," in *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, 2020.

[8] A. Paolillo, T. S. Lembono, and S. Calinon, "A memory of motion for visual predictive control tasks," in *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, 2020, pp. 9014–9020.

[9] W. Li and E. Todorov, "Iterative linear quadratic regulator design for nonlinear biological movement systems." in *ICINCO (1)*, 2004, pp. 222–229.

[10] H. J. Kappen, V. Gómez, and M. Opper, "Optimal control as a graphical model inference problem," *Machine learning*, vol. 87, no. 2, pp. 159–182, 2012.

[11] K. Rawlik, M. Toussaint, and S. Vijayakumar, "On stochastic optimal control and reinforcement learning by approximate inference," in *Twenty-third international joint conference on artificial intelligence*, 2013.

[12] S. Levine and V. Koltun, "Guided policy search," in *Proc. Intl Conf. on Machine Learning (ICML)*, 2013, pp. 1–9.

[13] S. Calinon, "Stochastic learning and control in multiple coordinate systems," in *Intl Workshop on Human-Friendly Robotics*, 2016.

[14] S. Calinon and D. Lee, "Learning control," in *Humanoid Robotics: a Reference*, P. Vadakkepat and A. Goswami, Eds. Springer, 2019, pp. 1261–1312.

[15] Y. Tassa, T. Erez, and W. D. Smart, "Receding horizon differential dynamic programming," in *Advances in Neural Information Processing Systems (NIPS)*, 2008, pp. 1465–1472.

[16] M. I. Ribeiro, "Kalman and extended kalman filters: Concept, derivation and properties," *Institute for Systems and Robotics*, vol. 43, p. 46, 2004.

[17] E. Todorov and M. I. Jordan, "A minimal intervention principle for coordinated movement," in *Advances in Neural Information Processing Systems (NIPS)*, 2002, pp. 27–34.

[18] S. Calinon, D. Bruno, and D. G. Caldwell, "A task-parameterized probabilistic model with minimal intervention control," in *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, 2014, pp. 3339–3344.

[19] T. A. Howell, B. E. Jackson, and Z. Manchester, "ALTRO: A fast solver for constrained trajectory optimization," in *Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS)*, 2019, pp. 7674–7679.

[20] N. Mansard, A. DelPrete, M. Geisert, S. Tonneau, and O. Stasse, "Using a memory of motion to efficiently warm-start a nonlinear predictive controller," in *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, 2018, pp. 2986–2993.