Equality Constrained Linear Optimal Control With Factor Graphs

Shuo Yang[1], Gerry Chen[2], Yetong Zhang[2], Frank Dellaert[2], and Howie Choset[1]

***Abstract*— This paper presents a novel factor graph-based approach to solve the discrete-time finite-horizon Linear Quadratic Regulator problem subject to auxiliary linear equality constraints within and across time steps. We represent such optimal control problems using constrained factor graphs and optimize the factor graphs to obtain the optimal trajectory and the feedback control policies using the variable elimination algorithm with a modified Gram-Schmidt process. We prove that our approach has the same order of computational complexity as the state-of-the-art dynamic programming approach. Furthermore, current dynamic programming approaches can only handle equality constraints between variables at the same time step, but ours can handle equality constraints among any combination of variables at any time step while maintaining linear complexity with respect to trajectory length. Our approach can be used to efficiently generate trajectories and feedback control policies to achieve periodic motion or repetitive manipulation.**

I. Introduction

The Equality Constrained Linear Quadratic Regulator (EC-LQR) is an important extension [1], [2] of the Linear Quadratic Regulator (LQR) [3]. The standard finite-horizon discrete-time LQR problem contains (1) quadratic costs on the state trajectory and the control input trajectory and (2) *system dynamics constraints* which enforce that the current state is determined by a linear function of the previous state and control. In the EC-LQR, *auxiliary constraints* are introduced to enforce additional linear equality relationships on one or more state(s) and/or control(s).

In many important problems, auxiliary constraints do not follow the Markov assumption yet non-Markovian auxiliary constraints are rarely considered in existing EC-LQR approaches. We classify auxiliary constraints in EC-LQR problems into two categories which we term *local constraints* and *cross-time-step constraints*. A local constraint only contains a state and/or control from the same time step. Examples of local constraints include initial and terminal conditions on states, contact constraints, and states along a predefined curve. In contrast, a cross-time-step constraint involves multiple states and controls at different time instances. Such non-Markovian constraints are pervasive in many robotics applications. For example, a legged robot's leg configuration must return to the same state after a period of time during a periodic gait [4]. In optimal allocation with resource constraints [5], the sum of control inputs is constrained to be some constant. Our goal is to solve for both optimal trajectories and optimal feedback control policies to EC-LQR problems with local and cross-time-step constraints in linear time with respect to the trajectory length.

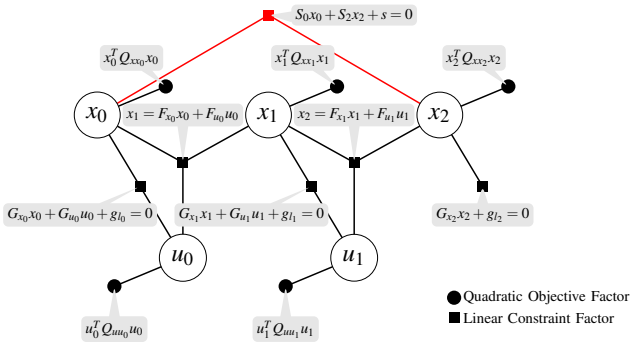


Fig. 1: The factor graph representation of an Equality Constrained Linear Quadratic Regular (EC-LQR) problem. Circles with letters are states or controls. Filled squares and circles represent objectives and constraints that involve the state or controls to which they are connected. The red square represents a cross-time-step constraint.

Reformulating control problems as inference problems [6]–[9] is a growing alternative to common trajectory optimization [10]–[12] and dynamic programming (DP) approaches for optimal control [1], [2], [13] . While trajectory optimization focuses on open-loop trajectories rather than feedback laws and a method using DP to handle cross-time-step constraints has yet to be proposed, control as inference may offer the advantages of both. Factor graphs in particular are a common tool for solving inference problems [14] and have recently been applied to optimal control [15], [16].

In this paper we propose a novel formulation using factor graphs [14] to efficiently solve the EC-LQR problem with both local and cross-time-step constraints in linear time with respect to trajectory length. We demonstrate how to represent the EC-LQR problem as a factor graph, and apply the variable elimination (VE) algorithm [17] on the factor graph to solve for the optimal trajectories and optimal feedback control policies. The flexibility of the factor graph representation allows cross-time-step constraints with arbitrary numbers of variables to be seamlessly handled. As long as the maximum number of variables involved in all constraints is bounded, the computational complexity of our method grows linearly with the trajectory length. The approach in this paper matches the computational complexity of standard dynamic programming techniques [2], but also has the added benefit of handing cross-time-step constraints.

[1] Shuo Yang and Howie Choset are with the Robotics Institute and Department of Mechanical Engineering, Carnegie Mellon University, Pittsburgh. Emails: {shuoyang, choset}@andrew.cmu.edu

[2] Gerry Chen, Yetong Zhang, and Frank Dellaert are with the Institute for Robotics and Intelligent Machines, Georgia Institute of Technology, Atlanta. Emails: {gchen328, yzhang3333, fd27}@gatech.edu

II. Related Work

Trajectory optimization methods typically transcript a problem into a Quadratic Programming (QP) [18] or Non-Linear Programming (NLP) [10] problem which can be efficiently solved to obtain open-loop trajectories of nonlinear systems. Local controllers can be used to track the open-loop trajectories generated [12]. Designing local controllers that obey equality constraints motivates EC-LQR problems.

For EC-LQR problems with just local constraints, DP-based approaches can generate both the optimal trajectories and feedback control policies. Solving standard LQR using DP is well understood in control theory [3]. [12] tackles EC-LQR with state-only local constraints by projecting system dynamics onto the constraint manifold. [1] extends the DP approach by using Karush-Kuhn-Tucker (KKT) conditions [5] to absorb auxiliary constraints into the cost function, but its computation time grows with the cube of the trajectory length for certain auxiliary constraints. [2] solves the EC-LQR with *local* constraints in linear complexity by adding a new auxiliary constraint dubbed "*constraint_to_go*" at each time step during DP steps.

Control as inference, in which a control problem is reformulated and solved as an inference problem, has gained considerable attention [6], [7]. Probabilistic Graphical Models (PGMs) in particular, commonly used for inference, are being applied to optimal control [8], [9] because they describe dependencies amongst variables while maintain sparsity in the graphical representation. So PGMs can solve variable distributions efficiently by exploiting sparsity [19]. The Markov assumption gives optimal control problems a "chain" structure when represented as PGMs allowing linear computational complexity with respect to trajectory length [7], [8], [16], [20], but PGMs can also exploit sparsity for more complex (non-chain) structures which motivates using PGMs for cross-time-step constraints.

Factor graphs, as a type of PGM, has been successfully appied to robot perception and state estimation [14]. Prior works have demonstrated that the variable elimination (VE) algorithm [17] on factor graphs can efficiently reduce the graph matrix representation in order to infer the posterior distribution of random variables in the factor graph. This procedure is called *factor graph optimization*. Moreover, factor graphs can encode constraints [21]. Other than estimation, factor graphs can be used to do motion planning [15], [22]. Standard LQRs are considered in [8], [16] without auxiliary constraints.

III. Problem And Method

In this section we first formulate the standard LQR and EC-LQR problems following the notation used in [2]. Then we solve a standard LQR problem as a factor graph and review relevant concepts related to factor graphs. Next, we solve EC-LQR with local constraints using factor graphs and compare our algorithm to the one proposed by [2], the most recent DP-based approach. Finally, we show how our method handles EC-LQR with cross-time-step constraints.

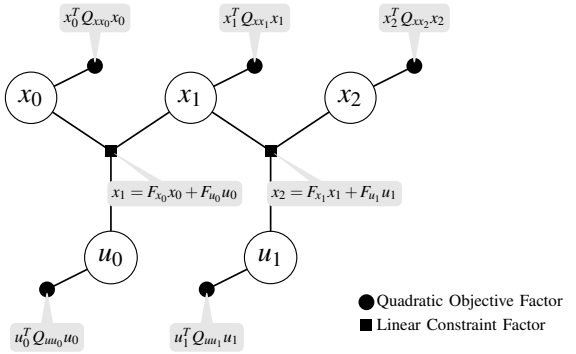


Fig. 2: Factor graph of a standard LQR problem with trajectory length $T = 2$.

A. Problem Formulation

For a robotic system with state $x_t \in \mathbb{R}^n$ and control input $u_t \in \mathbb{R}^m$, we define a state trajectory as $\mathbf{x} = [x_0, x_1, \ldots, x_T]$ and control input trajectory as $\mathbf{u} = [u_0, u_1, \ldots, u_{T-1}]$ where T is the trajectory length. The optimal control input trajectory $\mathbf{u}^*$ and its corresponding state trajectory $\mathbf{x}^*$ are the solution to the constrained linear least squares problem:

$$\min_{\mathbf{u}} \; x_T^T Q_{xx_T} x_T + \sum_{t=0}^{T-1} (x_t^T Q_{xx_t} x_t + u_t^T Q_{uu_t} u_t) \tag{1a}$$

$$\text{s.t.} \quad x_{t+1} = F_{x_t} x_t + F_{u_t} u_t \tag{1b}$$

$$G_{x_t} x_t + G_{u_t} u_t + g_{l_t} = 0, \quad t \in \mathscr{C}_1 \tag{1c}$$

$$G_{x_T} x_T + g_{l_T} = 0 \tag{1d}$$

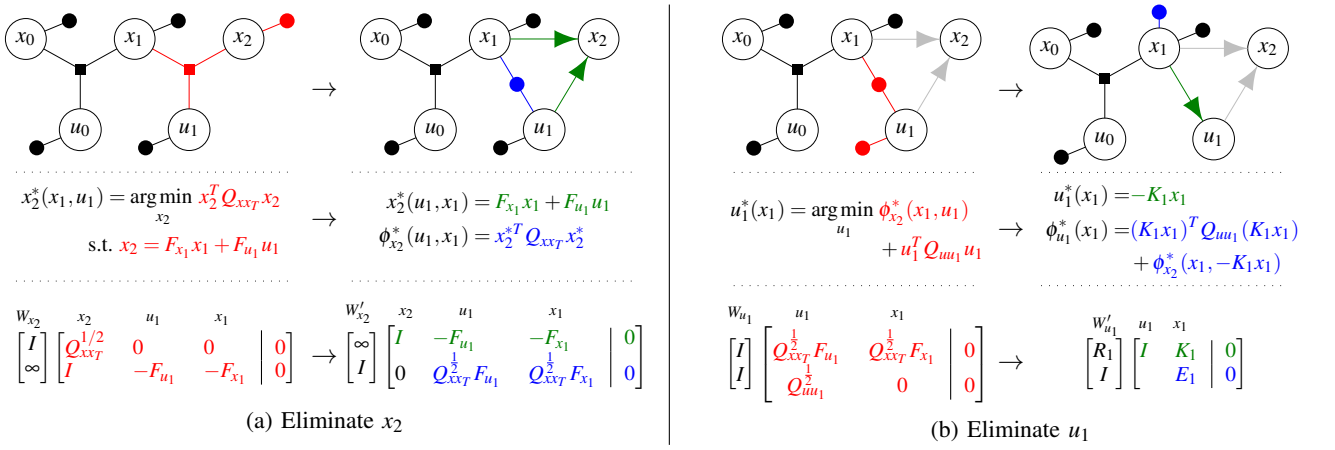
$$\sum_{i \in C_{kx}} S_{xki} x_i + \sum_{j \in C_{ku}} S_{ukj} u_j + s_k = 0, \quad k \in \mathscr{C}_2 \tag{1e}$$

where Q_{xx_T}, Q_{xx_t}, and Q_{uu_t} are positive definite matrices defining the cost function; F_{x_t} and F_{u_t} define the system dynamics at time t, constraints (1c) and (1d) are local auxiliary constraints; and constraint (1e) is a new formulation for cross-time-step constraints. In (1c) and (1d), $G_{x_t} \in \mathbb{R}^{l_t \times n}$, $G_{u_t} \in \mathbb{R}^{l_t \times m}$, and $g_{l_t} \in \mathbb{R}^{l_t}$ form local constraints with constraint dimension l_t; $\mathscr{C}_1$ is the set of time steps where a local constraint applies; and G_{x_T} and g_{l_T} form a local constraint with dimension l_T on the final step. In the cross-time-step constraint (1e), $S_{xki} \in \mathbb{R}^{c_t \times n}$, $S_{ukj} \in \mathbb{R}^{c_t \times m}$, and $s_k \in \mathbb{R}^{c_t}$ form constraints on a set of states x_i and controls u_j for $k \in \mathscr{C}_2$ where $\mathscr{C}_2$ is the set cross-time-step constraint indices.

B. Standard LQR as a Factor Graph

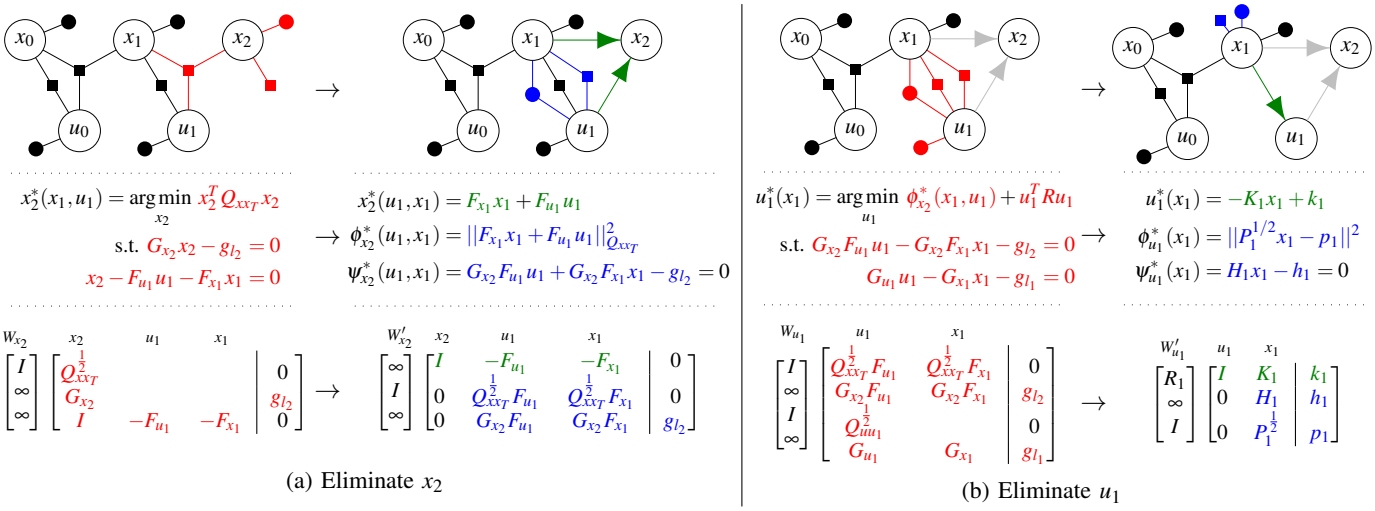
We demonstrate how to represent standard LQR, Problem 1 without constraints (1c), (1d), and (1e), as the factor graph shown in Figure 2 and subsequently obtain the optimal trajectory and the feedback control policy using VE.

Factor graphs can be interpreted as describing either a joint probability distribution with conditional independencies or, as we focus on in this paper, an equivalent least-squares problem derived from minimizing the log-likelihood. A factor graph is a bipartite graph consisting of variables and factors connected by edges, where a factor can be viewed either as



(a) Eliminate x_2 (b) Eliminate u_1

Fig. 3: Two variable eliminations for the LQR problem. Each sub-figure consists of three rows showing three equivalent representations: the factor graph (top), constrained optimization (middle), and modified Gram-Schmidt process on $[A_i|b_i]$ (bottom). The arrows in the factor graphs show variable dependencies. The thin horizontal arrows separate cases before and after elimination. Terms and symbols in the same color correspond to the color-coded variable elimination steps in Section III-B. Note that the matrix factorization representation consists of the weight vector, W_i, next to the sub-matrix $[A_i|b_i]$.



(a) Eliminate x_2 (b) Eliminate u_1

Fig. 4: Two elimination steps for EC-LQR with local constraints. This figure has the same layout as Figure 3.

a joint probability distribution or least squares objective over the variables it is connected to.

We will begin by showing how the probabilistic view of factor graphs is equivalent to a least squares minimization [14]. We construct factor graph to describe probablity distribution of variables $X = [\mathbf{x};\mathbf{u}]$. Factors are conditional probabilities that change the Maximum A Posteriori (MAP) estimation of variables. For Gaussian distributions, the probability distribution for a single objective or constraint factor ϕ_k can be written in matrix form as

$$\phi_k(X_k) \propto \exp\Big\{-\frac{1}{2}\|A_kX_k - b_k\|^2_{\Sigma_k}\Big\}$$

where exp is the exponential function and X_k contains the variables connected to the factor. A_k and b_k are a matrix and a vector respectively having problem-specific values, Σ_k is the covariance of the probability distribution, and $||\cdot||^2_\Sigma := (\cdot)^T\Sigma^{-1}(\cdot)$ denotes the Mahalanobis norm. A_k, b_k, and Σ_k together define the probabilistic distribution of the factor.

The product of all factors defines the posterior distribution of X whose MAP estimate is the solution of the least squares problem [14]:

$$\begin{aligned} X^{MAP} &= \arg\max_X \phi(X) = \arg\min_X -\log\Big(\prod_k \phi_k(X_k)\Big) \\ &= \arg\min_X \sum_k \|A_kX_k - b_k\|^2_{\Sigma_k} = \arg\min_X \|AX - b\|^2_\Sigma \end{aligned} \quad (2)$$

where A, b and Σ stack together all A_k, b_k, and Σ_k respectively such that each factor, ϕ_k, corresponds to a block row in A. Defining the weight matrix $W := \Sigma^{-1}$, X^{MAP} minimizes a weighted least squares expression $(AX-b)^TW(AX-b)$.

The objective factors in Figure 2 are $\phi_{objx}(x_t) \propto \exp\{-\frac{1}{2}\|Q_{xx_t}^{1/2}x_t\|^2\}$ or $\phi_{obju}(u_t) \propto \exp\{-\frac{1}{2}\|Q_{uu_t}^{1/2}u_t\|^2\}$, while the constraint factors are $\phi_{dyn}(x_{t+1},x_t,u_t) \propto \exp\{-\frac{1}{2}\|x_{t+1} - F_{x_t}x_t - F_{u_t}u_t\|^2_{\Sigma_c}\}$ where the covariance $\Sigma_c = 0$ creates infinite

terms in W. When factor graphs have factors with zero covariance, the least squares problem turns into a *constrained* least squares problem which we can solve using e.g. modified Gram-Schmidt [23].

The VE algorithm is a method to solve (2) while exploiting the sparsity of A by solving for one variable at a time. For a variable $\theta_i \in X$, we can identify its *separator* S_i: the set of other variables sharing factors with θ_i. Then we extract sub-matrices A_i, W_i, and sub-vector b_i from the rows of A, W, and b such that $[A_i|b_i]$ contains all factors connected to θ_i. We collect the rows in $[A_i|b_i]$ with finite weights to define objective factor $\phi_i(\theta_i, S_i)$ and rows with infinite weights to define constraint factor $\psi_i(\theta_i, S_i)$. Then we "eliminate" variable θ_i following 3 steps[1]:

Step 1. Identify all the factors adjacent to θ_i to get $[A_i|b_i]$. Split $[A_i|b_i]$ into $\phi_i(\theta_i, S_i)$ and $\psi_i(\theta_i, S_i)$.

Step 2. Solve the (constrained) least squares problem:

$$\theta_i^*(S_i) = \arg\min_{\theta_i} \phi_i(\theta_i, S_i) \text{ s.t. } \psi_i(\theta_i, S_i) = 0$$

using modified Gram-Schmidt or other constrained optimization methods [5, Ch.10]. $\theta_i^*(S_i)$ denotes that θ_i^* is a function of the variables in S_i.

Step 3. Substitute $\theta_i \leftarrow \theta_i^*$ by replacing the factors $\phi_i(\theta_i, S_i)$ and $\psi_i(\theta_i, S_i)$ with $\phi_i^*(S_i) := \phi_i(\theta_i^*, S_i)$ and $\psi_i^*(S_i) := \psi_i(\theta_i^*, S_i)$, respectively, in $[A|b]$.

We follow an *elimination order* [19] to eliminate one variable $\theta_i \in X$ at a time. After all variables are eliminated, the factor matrix A is effectively converted into an upper-triangular matrix R allowing X to be solved by matrix back-substitution. Therefore, one interpretation of the VE algorithm is performing sparse QR factorization on A [14].

To apply VE to the LQR factor graph in Figure 2, we choose the ordering $x_N, u_{N-1}, x_{N-1}, \ldots, x_0$ and execute Steps 1-3 to eliminate each variable. When eliminating a state x_i for the special case of LQR, the constrained least-squares problem in Step 2 is trivially solved as $x_i^*(u_{i-1}, x_{i-1}) = F_u u_{i-1} + F_x x_{i-1}$. Additionally, $\psi_{x_i}^*$ will be empty since $\psi_{x_i}(x_i^*, u_{i-1}, x_{i-1})$ is satisfied for any choice of u_{i-1} and x_{i-1}. Figure 3a shows the factor graphs, corresponding optimization problems, and sub-matrices $[W_i][A_i|b_i]$ before and after eliminating x_2.

The optimal feedback control policy emerges when eliminating a control u_i. The combined constraint factor ψ_{u_i} is empty (since $\psi_{x_{i+1}}^*$ is empty), so Step 2 reduces to an unconstrained minimization problem. To solve it using QR factorization, split the objective $||A_i[u;x]||_2^2 = ||R_i u + T_i x||_2^2 + ||E_i x||_2^2$ using the QR factorization $A_i = Q\begin{bmatrix} R_i & T_i \\ 0 & E_i \end{bmatrix}$ noting that Q is orthogonal and thus doesn't change the norm. Then, $u_i^*(x_i) = -K_i x_i$ where $K_i := R_i^{-1} T_i$ efficiently optimizes the first term and $\phi_{u_i}^*(x_i) = ||E_i x||_2^2$ is the new factor on x. The elimination is shown in Figure 3b.

Furthermore, the *cost_to_go* (or "value function" [24]), which commonly appears in DP-based LQR literature, is visually evident in the (right) factor graph from Figure 3b as the sum of the two unary factors on x_1:

$$cost\_to\_go_1(x_1) = x_1^T Q x_1 + x_1^T E_1^2 x_1.$$

Continuing to eliminate the rest of the variables will reveal the general formula of the *cost_to_go* after applying block-QR elimination to solve for K_i and E_i:

$$cost\_to\_go_i(x_i) = x_i^T (Q_{xx_t} + F_{x_t}^T V_{i+1} F_{x_t} - K_i^T F_{u_i}^T V_{i+1} F_{x_t}) x_i$$

where V_{i+1} comes from $\phi_{u_i}^*(x_i) + x_i^T Q_{xx_t} x_i = x_i^T V_i x_i$.

C. EC-LQR with Local Constraints

The factor graph representation of EC-LQR with only local constraints (1c) and (1d) in Problem 1 is the same as the factor graph in Figure 1 but without the red square marked "cross-time-step constraint". We still use the same elimination order: x_2, u_1, x_1, u_0, x_0 to execute VE.

1) Eliminating a state: The process for eliminating a state needs only consider one more constraint when generating $\psi_{x_i}^*(S_{x_i})$, but solving for x_i remains the same as in standard LQR case. Figure 4a shows the process of eliminating x_2.

2) Eliminating a control: The process for eliminating a control is a constrained minimization with some constraints on u_i derived from $\psi_{x_{i+1}}^*(u_i, x_i)$ and/or $G_{x_i} x_i + G_{u_i} u_i + g_{l_i} = 0$. The elimination procedure is shown in Figure 4b. From the result of eliminating u_1 as shown on the right in Figure 4b, we observe that

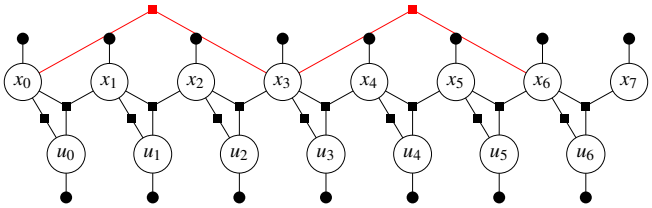
- the optimal control policy $u_1^*(x_1) = -K_1 x_1 + k_1$ falls out,
- $\phi_{u_1}^*(x_1) = ||P_1^{1/2} x_1 - p_1||^2$ corresponds to the $cost\_to\_go(x_1) = x_1^T V_1 x_1 - v_1 x_1$ from [2] where $V_1 = P_1 + Q_{xx_1}$ and $v_1 = 2p_1^T P_1$, and
- $\psi_{u_1}^* = H_1 x_1 - h_1 = 0$ corresponds to the $constraint\_to\_go(x_1) = H_1 x_1 - h_1 = 0$ from [2]

We continue with VE to eliminate the remaining variables similarly. After each u_i is eliminated, we can obtain an optimal control policy, *constraint_to_go*, and *cost_to_go* – all of which being functions of x_i. When the problem is linear and all matrices are invertible or full column rank, the optimal solution is unique. We will show that our method finds the unique optimal solution in Section IV.

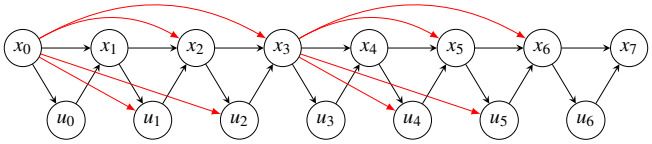
Computational Complexity Analysis

Because Step 1 collects only the factors connected to the variable we seek to eliminate, VE is very efficient and the complexity of eliminating a single variable is independent of the trajectory length. When eliminating one variable, we factorize a matrix, A_i, whose rows consist of all the factors connected to the variable and whose columns correspond to the variable and its separator. Thus, the maximum dimensions of A_i in EC-LQR problem with just local constraints are $3n \times (2n+m)$ when eliminating a state or $(2n+m) \times (n+m)$ when eliminating a control. In the worst case, QR factorization on this matrix has complexity $O(2(3n)^2(2n+m)) = O(36n^3 + 18n^2 m)$ when eliminating a state or $O(2(2n+m)^2(n+m)) = O(8n^3 + 16n^2 m + 10nm^2 + 2m^3)$ when eliminating a control. To obtain the solution from

[1] In the probabilistic form, steps 2 and 3 would come from factoring $\phi_i(\theta_i, S_i)\psi_i(\theta_i, S_i) \propto p(\theta_i|S_i)p(S_i)$. For Gaussian distributions, $\theta_i^*(S_i) = E[p(\theta_i|S_i)]$ and $\phi_i^*(S_i)\psi_i^*(S_i) = p(S_i)$.



(a) Factor graph



(b) The Bayes Net after Variable Elimination

Fig. 5: Example cross-time-step constraint in a factor graph. The bottom figure is a Bayes net showing variable dependencies after VE.

the sparse QR factorization result of A, we apply back substitution whose computation complexity is $O(T \cdot (n^2+m^2))$, so the overall computation complexity of solving the trajectory with length T is $O(T \cdot (\kappa_1 n^3 + \kappa_2 n^2 m + \kappa_3 n m^2 + \kappa_4 m^3))$, which is the same as the state of the art DP approach [2].

D. EC-LQR with Cross-time-step Constraints

The factor graph's ability to add factors on any set of variables allows us to add more general auxiliary constraints and objectives than [2], such as cross-time-step constraints. The VE algorithm for solving EC-LQR with cross-time-step constraints remains exactly the same as in Section III-C. Note that cross-time-step *objectives* could also be handled the same way if desired. Taking Figure 5 as an example, the cross-time-step constraint is $Sx_{n_c+p} - Sx_{n_c} - s = 0$, where S is a selection matrix that selects certain elements from the state, p is the period of the motion cycle, and s is a constant state "advancement" vector. When eliminating x_{n_c+p}, its separator will contain x_{n_c+p-1}, u_{n_c+p-1} and x_{n_c}. After elimination of x_{n_c+p}, the new *constraint_to_go* factor will be connected to not only x_{n_c+p-1} and $_{n_c+p-1}$, but also x_{n_c}. Subsequent elimination steps will generate similar factors so that, after all variables are eliminated, the final feedback controllers for control inputs between x_{n_c+p} and x_{n_c} are functions of two states instead of just the current state. Figure 5b illustrates the result in the form of a Bayes Net [14] where arrows represent the variable dependencies.

Our method maintains linear complexity for cross-time-step constraints. From a complexity analysis point of view, as long as the number of variables involved in all constraints is bounded, then the maximum dimension of the separator for any variable will also be bounded thus bounding the complexity of each variable elimination step.

IV. Experiments

We run simulation experiments to demonstrate the capability of the proposed method. We implement our method using the Georgia Tech Smoothing And Mapping (GTSAM) toolbox [25]. We compare our approach with three baseline methods implemented in MATLAB. Baseline method 1 is [1], Baseline method 2 is [2], and Baseline method 3 is using Matlab's `quadprog` quadratic programming solver (which does not produce an optimal control *policy*). We first present comparison experiments for EC-LQR of a toy system with local constraints. We then show our approach handling cross-time-step constraints on an example system motivated by a single leg hopping robot.

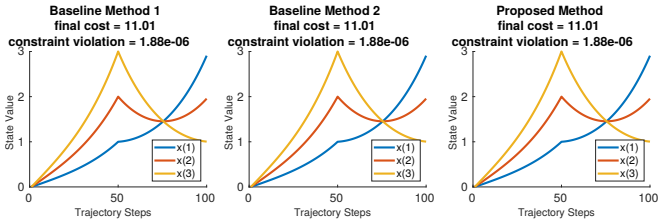


Fig. 6: Optimal trajectory, cost and constraint violation comparison of the three methods for Problem 3. For each method we plot the three dimension of the state x. All 3 methods produce the same result.

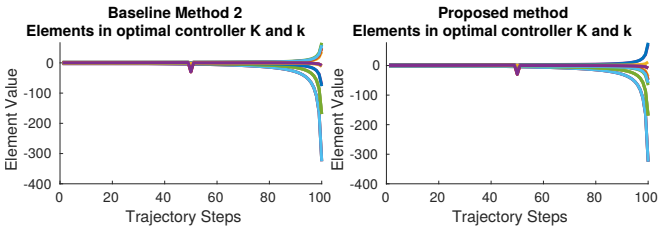


Fig. 7: The plots of feedback control gain matrices from Baseline Method 2 and ours. Each curve represents one element in K_t or k_t with time step t on the horizontal axis.

A. Cost & Constraint Violation Comparison

The first experiment is to find the optimal trajectory for a simple system with $x_i \in \mathbb{R}^3$ and $u_i \in \mathbb{R}^3$ that are subjected to state constraints. The EC-LQR problem is given by:

$$\min_{\mathbf{u}} \; x_T^T Q_{xx_T} x_T + \sum_{t=0}^{T-1} (x_t^T Q_{xx_t} x_n + u_t^T Q_{uu_t} u_t) \tag{3a}$$

$$\text{s.t.} \quad x_{t+1} = F_x x_t + F_u u_t \tag{3b}$$

$$x_0 = [0 \;\; 0 \;\; 0]^T, \quad x_N = [3 \;\; 2 \;\; 1]^T \tag{3c}$$

$$x_{N/2} = [1 \;\; 2 \;\; 3]^T \tag{3d}$$

where $dt = 0.01$, $F_x = I_{3\times3} + I_{3\times3} \cdot dt$, $F_u = I_{3\times3} \cdot dt$, $T = 100$, $Q_{xx_t} = 0.01 \cdot I_{3\times3}$, $Q_{uu_t} = 0.001 \cdot I_{3\times3}$, and $Q_{xx_T} = 500 \cdot I_{3\times3}$.

Figure 6 compares the optimal state trajectories using the three methods. All three methods arrive at the exact same solution, with 0 constraint violation and identical total cost. This is expected because in this problem the optimal solution is unique. The result supports the correctness of our method.

B. Optimal Controller Comparison

In this experiment we show the optimal controller generated by the proposed method is equivalent to that generated by [2]. To demonstrate our method can solve state-only

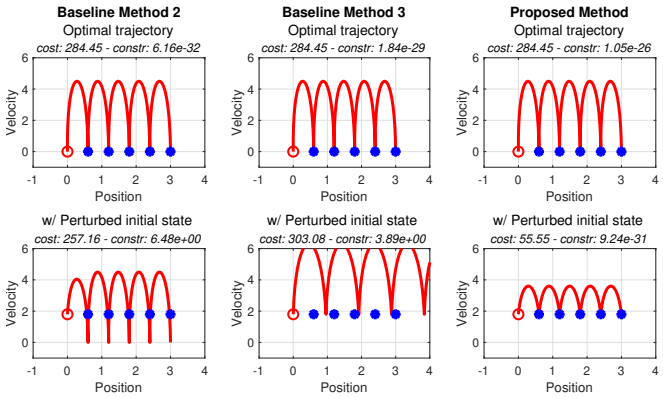


Fig. 8: The state trajectories solving Problem 5 using Baseline method 2 (left), Baseline method 3 (middle), and our proposed method (right) with control sequence/policies applied to the original problem (top) and after perturbing the initial state (bottom). All methods generate the same trajectory to the initial problem, but only ours gives a policy which generates the optimal trajectory for the perturbed problem. "Cost" and "Constr" denote the total objective cost and constraint violation, respectively.

constraints as well as state and control local constraints, we replace the state-only constraint (3d) in Problem 3 to be a constraint that contains both the state and the control as

$$x_{N/2} + u_{N/2} + [1 \quad 2 \quad 3]^T = 0. \tag{4}$$

We solve this problem to get the optimal controllers $u_t = -K_t x_t + k_t$ for the two methods. K_t and k_t are identical between the two methods, as shown in Figure 7.

C. Cross-time-step Constraints

To illustrate an example of how cross-time-step constraints can be used to generate useful trajectories, we use a double integrator system ($x_i = [\text{position}, \text{velocity}], u = \text{acceleration}$) with periodic "step placements". Consider the x-coordinate of a foot of a hopping robot which initially starts in contact with the ground and makes contact with the ground again every 20 time steps. Each contact, it advances forward by 0.6 units and must match the ground velocity (which may be non-zero e.g. on a moving walkway). The problem is:

$$\min_{\mathbf{u}} \; x_T^T Q_{xx_T} x_T + \sum_{t=0}^{T-1} (x_t^T Q_{xx_t} x_n + u_t^T Q_{uu_t} u_t) \tag{5a}$$

$$\text{s.t.} \quad x_{t+1} = \begin{bmatrix} 1 & dt \\ 0 & 1 \end{bmatrix} x_t + \begin{bmatrix} 0 \\ dt \end{bmatrix} u_t, \quad x_0 = [0 \quad 0]^T, \tag{5b}$$

$$x_{n_c+20} - x_{n_c} = \begin{bmatrix} -0.6 \\ 0 \end{bmatrix}, n_c = 0, 20, 40, 60, 80 \tag{5c}$$

The cross-time-step constraints (5c) enforce that contacts must occur at a fixed position relative to and with same velocity as the previous contacts $p = 20$ time steps prior. These create constraint factors between two state variables $p = 20$ time steps apart, as in Figure 5.

Figure 8 shows the solutions to Problem 5 using Baseline 2 [2], Baseline 3 (QP), and our method, as well as the results when using the same controllers with a perturbed initial state $x_0 = [0 \quad 1.8]^T$ (i.e. walking on a moving walkway with velocity 1.8). We apply some modifications to allow for comparison since Baseline 2 cannot natively handle cross-time-step constraints and Baseline 3 cannot generate an optimal policy, but even so, the adjusted baselines do not generate optimal trajectories from perturbed initial state, as shown in Figure 8 (bottom). For baseline method 2, we convert the cross-time-step constraints to same-time-step constraints $x_{n_c} = [0.03 n_c \quad 0]^T$ for $n_c = 0, 20, \ldots$ resulting in incorrect constraints after perturbing the initial state. An alternative would be to introduce 10 additional state dimensions (two for each cross-time-step constraint) analagous to Lagrange multipliers, but we argue that such an approach is not sustainable for online operation and many cross-time-step constraints. For Baseline 3, we re-use the control sequence from Problem 5 for the perturbed case. Our method's control law produces a state trajectory that is optimal and without constraint violation even with a perturbed initial state an shown in Figure 8 (bottom right).

V. Future Work

The proposed method points to a promising direction in generating trajectories for constrained high dimensional robotics systems such as legged robots and manipulators. Just as LQR is a building block of Differential Dynamic Programming (DDP) [26] and iterative LQR [13], linear factor graphs could be the building block of a more general nonlinear optimal control algorithm, but the following need to be explored first:

- Incorporate inequality constraints into the factor graph by converting them to equality constraints e.g. using barrier or penalty functions [27].
- Extend to nonlinear systems by using nonlinear factor graphs [14].
- Address over-constrained "constraints" in VE.
- Leverage incremental factor graph-based solving using Bayes Trees [28] to do efficient replanning.
- Consider stochastic optimal control with uncertain constraints.
- Combine estimation and optimal control factor graphs together to better close the perception-control loop.

VI. Conclusions

In this paper, we proposed solving equality constrained linear quadratic regular problems using factor graphs. We showed that factor graphs can represent linear quadratic optimal control problems with auxiliary constraints by capturing the relationships amongst variables in the form of factors. Variable elimination, an algorithm that exploits matrix sparsity to optimize factor graphs, is used to efficiently solve for the optimal trajectory and the feedback control policy. We demonstrated that our approach can handle more general constraints than traditional dynamic programming approaches. We believe this method has great potential to solve difficult constrained optimal control problems for a number of complex robotics systems.

REFERENCES

[1] A. Sideris and L. A. Rodriguez, "A Riccati approach for constrained linear quadratic optimal control," *International Journal of Control*, vol. 84, no. 2, pp. 370–380, 2011.

[2] F. Laine and C. Tomlin, "Efficient computation of feedback control for equality-constrained LQR," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 6748–6754.

[3] R. E. Kalman, "A new approach to linear filtering and prediction problems," 1960.

[4] F. Farshidian, E. Jelavic, A. Satapathy, M. Giftthaler, and J. Buchli, "Real-time motion planning of legged robots: A model predictive control approach," in *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*. IEEE, 2017, pp. 577–584.

[5] S. Boyd, S. P. Boyd, and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[6] S. Levine, "Reinforcement learning and control as probabilistic inference: Tutorial and review," *arXiv preprint arXiv:1805.00909*, 2018.

[7] M. Toussaint, "Robot trajectory optimization using approximate inference," in *Proceedings of the 26th annual international conference on machine learning*, 2009, pp. 1049–1056.

[8] J. Watson, H. Abdulsamad, and J. Peters, "Stochastic optimal control as approximate input inference," in *Conference on Robot Learning*, 2020, pp. 697–716.

[9] H. J. Kappen, V. Gómez, and M. Opper, "Optimal control as a graphical model inference problem," 2009.

[10] M. Kelly, "An introduction to trajectory optimization: How to do your own direct collocation," *SIAM Review*, vol. 59, no. 4, pp. 849–904, 2017.

[11] H. Dai, A. Valenzuela, and R. Tedrake, "Whole-body motion planning with centroidal dynamics and full kinematics," in *2014 IEEE-RAS International Conference on Humanoid Robots*. IEEE, 2014, pp. 295–302.

[12] M. Posa, S. Kuindersma, and R. Tedrake, "Optimization and stabilization of trajectories for constrained dynamical systems," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 1366–1373.

[13] W. Li and E. Todorov, "Iterative linear quadratic regulator design for nonlinear biological movement systems."

[14] F. Dellaert, M. Kaess *et al.*, "Factor graphs for robot perception," *Foundations and Trends® in Robotics*, vol. 6, no. 1-2, pp. 1–139, 2017.

[15] J. Dong, M. Mukadam, F. Dellaert, and B. Boots, "Motion planning as probabilistic inference using gaussian processes and factor graphs." in *Robotics: Science and Systems*, vol. 12, 2016, p. 4.

[16] G. Chen and Y. Zhang, "LQR control using factor graphs," https://gtsam.org/2019/11/07/lqr-control.html, accessed: 2020-09-13.

[17] J. R. Blair and B. Peyton, "An introduction to chordal graphs and clique trees," in *Graph theory and sparse matrix computation*. Springer, 1993, pp. 1–29.

[18] A. Barclay, P. E. Gill, and J. B. Rosen, "SQP methods and their application to numerical optimal control," in *Variational calculus, optimal control and applications*. Springer, 1998, pp. 207–222.

[19] D. Koller and N. Friedman, *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[20] K. J. Astrom, *Introduction to stochastic control theory*. Elsevier, 1971.

[21] A. Cunningham, M. Paluri, and F. Dellaert, "DDF-SAM: Fully distributed slam using constrained factor graphs," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2010, pp. 3025–3030.

[22] D.-N. Ta, M. Kobilarov, and F. Dellaert, "A factor graph approach to estimation and model predictive control on unmanned aerial vehicles," in *2014 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2014, pp. 181–188.

[23] M. Gulliksson, "On the modified Gram-Schmidt algorithm for weighted and constrained linear least squares problems," *BIT Numerical Mathematics*, vol. 35, no. 4, pp. 453–468, 1995.

[24] D. P. Bertsekas, *Dynamic programming and optimal control*. Athena scientific Belmont, MA, 1995, vol. 1, no. 2.

[25] F. Dellaert, "Factor graphs and GTSAM: A hands-on introduction," Georgia Institute of Technology, Tech. Rep., 2012.

[26] D. Q. Mayne, "Differential dynamic programming–a unified approach to the optimization of dynamic systems," in *Control and Dynamic Systems*. Elsevier, 1973, vol. 10, pp. 179–254.

[27] R. Grandia, F. Farshidian, R. Ranftl, and M. Hutter, "Feedback mpc for torque-controlled legged robots," *arXiv preprint arXiv:1905.06144*, 2019.

[28] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard, and F. Dellaert, "iSAM2: Incremental smoothing and mapping using the Bayes tree," *The International Journal of Robotics Research*, vol. 31, no. 2, pp. 216–235, 2012.