A Fast Algorithm for Stochastic Model Predictive Control with Probabilistic Constraints

Minyong Shin and James A. Primbs

Abstract—A fast suboptimal algorithm for finite horizon stochastic linear-quadratic control under probabilistic constraints is presented. This type of control problem is solved repeatedly in stochastic model predictive control. Under the assumption of affine state feedback, the control problem is converted to an equivalent deterministic problem using the mean and covariance matrix as the state. An interior point method is proposed to solve this optimization problem, where the step direction can be quickly computed via a Riccati difference equation. On a two state, two constraint numerical example in this paper, the algorithm is over 200 times faster than a convex formulation that uses a general purpose solver.}

I. INTRODUCTION

In this paper, we propose a fast but suboptimal algorithm for the finite horizon control of a discrete-time linear dynamic system with additive Gaussian noise under a quadratic objective and linear probabilistic constraints. While control problems of this type are of interest in their own right, their use in stochastic model predictive control (MPC) schemes is needed to produce real-time solutions. Stochastic MPC is an important research area in the control community (see for example [8], [2], [1], [4], [3], [12], [13]), and is the primary motivation for this work.

In the MPC setting, the numerical solution of finite horizon LQG problems with constraints has received much attention, especially in the direction of developing a convex formulation. In a series of papers, van Hessem and Bosgra showed that it could be solved as a convex optimization problem by using a Youla parameterization [16] or “innovation feedback” [17]. In related work, Löfberg [9] used disturbance feedback to formulate a similar problem as a convex optimization, but under bounded noise. Thus, it is possible to use general purpose SDP solvers for constrained linear stochastic MPC problems. However, when prediction horizon lengths are long, these convex formulations can grow quickly in size and computation times can be prohibitive for real time implementation.

The purpose of this paper is to present a fast algorithm for the linear-quadratic-Gaussian problem with constraints with the goal that it may increase the applicability of stochastic MPC to larger scale systems and/or those with fast dynamics. Our approach is motivated by the work of Rao et al. [14]. They showed that the Riccati structure present in an interior point method for deterministic linear-quadratic MPC problems could be used to greatly speed up computations. In fact, using this Riccati structure decreases the computational complexity from cubic to linear in the horizon length [7],[19],[5],[18]. This paper shows that an interior point method for the stochastic problem that exploits a Riccati-like structure may also be developed and provide dramatic increases in computation speed. Related work can be found in a robust control context by Hansson [7].

This paper is organized as follows. In section II, we formulate the mean-variance deterministic problem from the finite horizon linear-quadratic-Gaussian control problem. In section III, we propose an Interior Point method for this specific problem and provide a closed form solution for the iterative direction in the form of a Riccati difference equation. In section IV, we summarize the complete algorithm and in section V the simulation of a simple example is implemented and a computation speed comparison with a convex approach using a general SDP solver is provided. Conclusions and future directions are given in section VI.

II. PROBLEM SETUP

We consider the constrained stochastic linear-quadratic finite horizon control problem

\[
\begin{align*}
\min_{u_k} & \quad \mathbb{E} \left( \sum_{k=0}^{N-1} \left( x_k^T Q x_k + u_k^T R u_k \right) + \frac{1}{2} x_N^T Q_f x_N \right) \\
\text{s.t.} & \quad x_{k+1} = A x_k + B u_k + G w_k \\
& \quad \Pr(h_{i,k}^T x_k + h_{i,k}^T u_k \leq g_i) \geq \alpha_i, \quad k = 0 \ldots N - 1 \\
& \quad \Pr(h_{i,N}^T x_N \leq g_i) > \alpha_i \\
& \quad x(0) \sim \mathcal{N}(x_0, \Sigma_0), \quad i = 1 \ldots n
\end{align*}
\]

with horizon \( N \), where (1) is the discrete-time system dynamics with state \( x_k \in \mathbb{R}^n \) at time \( k \), control input \( u_k \in \mathbb{R}^m \), and Gaussian noise \( w_k \in \mathbb{R}^m \) with distribution \( w_k \sim \mathcal{N}(0, \Sigma_w) \). Equation (2) represents \( n \) probabilistic (chance) constraints over the horizon.

Finite horizon control problems of this form are solved repeatedly in MPC applied to stochastic systems. Thus, our goal is to find a fast algorithm to approximately solve this problem. To do so, we make the assumption of time-varying affine feedback, and then convert the problem to a deterministic equivalent, as detailed in the following sections.
A. Feedback Structure

First, we assume a time varying affine feedback structure for the control actions,

$$u_k = K_k (x_k - \bar{x}_k) + \bar{u}_k$$  \hspace{1cm} (3)

where the gain matrices, $K_k$, and the open loop control, $\bar{u}_k$, become the new decision variables.

B. Probabilistic Constraints

Next, under the feedback assumption (3), and by the normality of the noise $w_k$ and the linearity of the state dynamics, we can convert the probabilistic constraints (2) to mean and variance constraints. Suppose we have linear probabilistic constraints of the form

$$\Pr(h^T x \leq g) \geq \alpha$$

where $x \sim N(\bar{x}, \Sigma)$ and $\alpha > 0.95$. This probability constraint can be transformed to the equivalent constraint,

$$h^T \bar{x} + r \sqrt{h^T \Sigma h} \leq g$$  \hspace{1cm} (4)

where $r$ is defined by

$$\sqrt{\frac{2}{\pi}} \int_0^r \exp(-\frac{t^2}{2}) dt = \alpha.$$

Note that this conversion has been commonly used in previous work such as van Hessen and Bosgra [16, 17].

In our algorithm, we will use an equivalent squared version of (4) that requires two constraints.

$$\frac{1}{2} r^2 (h^T \Sigma h) - \frac{1}{2} (g - h^T \bar{x})^2 \leq 0$$  \hspace{1cm} (5)

$$h^T \bar{x} - g \leq 0.$$

By denoting

$$h = \begin{bmatrix} h_k & h_u \end{bmatrix}, \quad \bar{x} = \begin{bmatrix} \bar{x}_k \\ \bar{u}_k \end{bmatrix}, \quad \Sigma = \begin{bmatrix} I & \Sigma_k \\ K_k & I \end{bmatrix}^T$$

where $\bar{x}_k = \mathbb{E}[x_k]$, and $\Sigma_k = \mathbb{E}[(x_k - \bar{x}_k)(x_k - \bar{x}_k)^T]$, the constraint term in (2) involving $h^T \bar{x}_k + h_u^T \bar{u}_k$, can be expressed as

$$r^2 (h_k^T \Sigma_k h_k + K_k^T h_u) - (h_k^T \bar{x}_k + h_u^T \bar{u}_k - g)^2 \leq 0$$

$$h_k^T \bar{x}_k + h_u^T \bar{u}_k - g \leq 0.$$

Due to the absence of the control input, the constraint at the final time step $N$ is

$$r^2 (h_N^T \Sigma_N h_k) - (h_N^T \bar{x}_N - g)^2 \leq 0$$

$$h_N^T \bar{x}_N - g \leq 0.$$

C. “Deterministic” Mean-Variance Formulation

Finally, due to the normality of the noise and the linearity of the system dynamics, we may convert problem cSLQ entirely to a deterministic control problem in terms of the mean and covariance dynamics of the system. That is, under our assumed affine feedback, the mean and variance of the state $x_k$ follow

$$\bar{x}_{k+1} = A\bar{x}_k + B\bar{u}_k$$

$$\Sigma_{k+1} = (A + BK_k) \Sigma_k (A + BK_k)^T + G\Sigma_u G^T.$$

Furthermore, in a straightforward manner, the quadratic objective may be written in terms of the mean and variance. Thus, problem cSLQ under affine feedback may be written equivalently as a constrained deterministic quadratic optimization problem with $n$ quadratic and linear constraints plus 2 system dynamics constraints under the decision variables $\bar{u}_k$ and $K_k$.

$$(\star) \quad \text{Problem mean-var SLQ}$$

$$\min_{\bar{u}, K} \frac{1}{2} \sum_{k=0}^{N-1} \left\{ tr((Q + K_k^T \Sigma_k) \bar{x}_k) + \bar{x}_k^T Q \bar{x}_k + \bar{u}_k^T R \bar{u}_k \right\}$$

$$+ \frac{1}{2} \text{tr}(Q \Sigma_N) + \frac{1}{2} \Sigma_N^T Q \Sigma_N$$

subject to

$$A\bar{x}_k + B\bar{u}_k - \bar{x}_{k+1} = 0$$

$$(A + BK_k) \Sigma_k (A + BK_k)^T + G\Sigma_u G^T - \Sigma_{k+1} = 0$$

$$\left| \begin{array}{c} \frac{1}{2} \bar{r}_k^2 (h_{x,i} + K_k^T h_{u,i})^T \Sigma_k (h_{x,i} + K_k^T h_{u,i}) \\
- \frac{1}{2} (h_{x,i}^T \bar{x}_k + h_{u,i}^T \bar{u}_k - g_i)^2 \leq 0 \\
h_{x,i}^T \bar{x}_k + h_{u,i}^T \bar{u}_k - g_i \leq 0 \end{array} \right. \quad k = 0, \ldots, N - 1$$

$$\left| \begin{array}{c} \frac{1}{2} \bar{r}_k^2 (h_{x,i}^T \Sigma_N h_{x,i}) - \frac{1}{2} (h_{x,i}^T \Sigma_N - g_i)^2 \leq 0 \\
h_{x,i}^T \Sigma_N - g_i \leq 0 \end{array} \right. \quad i = 1, \ldots, n. \hspace{1cm} (6)$$

To solve this constrained optimization problem, we seek an interior point method which can make use of “Riccati” structure in its linearized step. The motivation for this comes from the work of Rao, Wright, and Rawlings [14], who exploit Riccati structure in an interior point method for constrained deterministic MPC problems.

The following sections outline an interior point method for the problem (6) that exploits problem structure to greatly speed up the optimization.

III. STEP DIRECTION FOR INTERIOR POINT METHOD

Roughly speaking, an interior point method is a specialized iterative method to solve the Karush-Kuhn-Tucker (KKT) conditions corresponding to an optimization problem. The interior point method developed in this paper uses three steps at each iteration.

1) Determine the value of a centering parameter (We use Mehrotra’s algorithm [10]).
2) Compute a step direction.
3) Compute the step length.
To keep the presentation brief, we do not provide the details of Mehrrotra’s algorithm. Furthermore, we simply note that computing the step length is done by finding the maximum step such that the updated variables stay in the feasible region. Instead, our presentation focuses on the critical computation of the step direction. It is in this step that the Riccati structure is exploited. Furthermore, this structure appears even in a standard Newton iteration for the KKT conditions. Thus, for clarity, below we detail how to compute the step direction using Riccati structure in a standard Newton iteration. A more complete step-by-step algorithm is given in section IV whereby the interested reader can easily fill in the step length and centering steps from standard references.

The first step is to form the Lagrangian.

A. Lagrangian

We form a Lagrangian corresponding to the optimization problem mean-var SLQ by multiplying Lagrange multipliers (also called dual variables) denoted, $\rho_k$, $Z_k$, $\lambda_i,k$ and $v_i,k$ to the mean and variance dynamics, and the quadratic and linear constraints respectively. This leads to

$$\mathcal{L} = \sum_{k=0}^{N-1} \frac{1}{2} \text{tr}\left((Q + K_i^T R K_k) \Sigma_k\right) + \frac{1}{2} \bar{u}_k^T R \bar{u}_k + \frac{1}{2} \bar{\lambda}_k^T Q \bar{\lambda}_k$$

$$+ \frac{1}{2} \text{tr}\left((A \bar{x}_k + B \bar{u}_k - \bar{s}_{k+1})\right) + \frac{1}{2} \text{tr}\left((A + B K_k) \Sigma_k (A + B K_k)^T + G \Sigma_k G^T - \Sigma_{k+1}\right)$$

$$+ \sum_{i=1}^{n} \left\{ \frac{1}{2} \lambda_i,k \left( r_i^2 (h_{i,x}^T h_{i,u} + K_i^T h_{i,u,i})^T \Sigma_k (h_{i,x} + K_i^T h_{i,u,i})

- (h_{i,x}^T \bar{x}_k + h_{i,u}^T \bar{u}_k - g_i)^2 \right) + v_i,k (h_{i,x}^T h_{i,u} + h_{i,u}^T \bar{u}_k - g_i) \right\}$$

$$+ \frac{1}{2} \text{tr}(Q) \Sigma_N + \frac{1}{2} \bar{\lambda}_k^T Q \bar{\lambda}_N$$

$$+ \sum_{i=1}^{n} \left\{ \frac{1}{2} \lambda_{N,i} \left( r_i^2 (h_{i,x}^T \Sigma_N h_{i,u,i}) - (h_{i,x}^T \bar{x}_N - g_i)^2 \right)

+ v_{N,i} (h_{i,x}^T \bar{x}_N - g_i) \right\}. \quad (7)$$

B. KKT Condition

Next, the KKT conditions come from the first order necessary conditions for a stationary point of the Lagrangian, which is regarded as a function of the primal and dual variables. Define, $H_i^x = h_{i,x} h_{i,x}^T$, $H_i^u = h_{i,u} h_{i,u}^T$, $H_i^{tu} = h_{i,u} h_{i,x}^T$, $H_{iu} = h_{i,u} h_{i,u}^T$, where $i = 1, \ldots, n$. We group the KKT conditions as

$$\begin{cases} 
\bar{x}_{k+1} = \bar{A} \bar{x}_k + B \bar{u}_k \\
\bar{s}_{k+1} = (A + B K_k) \bar{x}_k + G \Sigma_k G^T
\end{cases} \quad (8)$$

$^1$The infeasible IP method we employ uses the Lagrange Multiplier (LM) method which gives the first order necessary condition for optimality. The LM method is valid when the constraints satisfy a qualification. One can show easily that our constraints satisfy the Linear Independence Constraint Qualification.
where the superscript \( j \) represents the \( j \)th iteration value of the variable. The step direction at each iteration of an interior point method is determined by solving the linearized equations (15) - (21) for the \( \Delta \)'s of the variables. It is in these equations that we find exploitable Riccati structure as follows.

First, given \( \lambda_{gj} > 0 \), \( v_{ij} > 0 \), note that equations (20) and (21) may be solved explicitly for \( \Delta t_{ik} \) and \( \Delta s_{ik} \) as

\[
\Delta t_{ik} = -t_{ik}^{j} - \frac{t_{ik}^{j}}{\lambda_{gj}} \Delta \lambda_{ik}, \quad \Delta s_{ik} = -s_{ik}^{j} - \frac{s_{ik}^{j}}{v_{ik}} \Delta v_{ik}.
\]

(22)

Furthermore, substituting (22) into (18), (19) gives

\[
\Delta \lambda_{ik} = -\frac{\lambda_{gj}}{\lambda_{ik}} \left[ G_{i}(X_{i}^{j}) + \nabla G_{i} \cdot \Delta X_{i} \right]
\]

(23)

\[
\Delta v_{ik} = -\frac{v_{ik}}{s_{ik}} \left[ H_{i}(X_{i}^{j}) + \nabla H_{i} \cdot \Delta X_{i} \right].
\]

(24)

Finally, substituting (23), (24) into (15), (16), and (17) leads to a system of equations at time \( k \) that may be written in the general form

\[
\begin{bmatrix}
-1 \dot{A}_{k} \dot{B}_{k} \dot{D}_{k} & \dot{E}_{k} \\
\dot{B}_{k} \dot{E}_{k} & -1 \\
\end{bmatrix}
\begin{bmatrix}
\Delta \lambda_{k-1} \\
\Delta v_{k-1} \\
\end{bmatrix}
= \begin{bmatrix}
\dot{R}_{1,k} \\
\dot{R}_{2,k} \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
\Delta \lambda_{k} \\
\Delta v_{k} \\
\end{bmatrix}
= \begin{bmatrix}
\dot{R}_{1,k} \\
\dot{R}_{2,k} \\
\end{bmatrix}
\]

(25)

The exact expressions for \( \dot{A}_{k}, \dot{B}_{k}, \dot{E}_{k}, \dot{D}_{k}, \dot{E}_{k} \), and the boundary conditions \( \Phi_{N}, \phi_{N} \) can be found in the Appendix.

D. Riccati Difference Equation

In order to construct a recursive solution to (25), we introduce an additional equation which reflects previous calculations from time step \( k+1 \) through time step \( N \).

\[
-\Delta \Gamma_{k} + \Phi_{k+1} \Delta \bar{X}_{k-1} = \phi_{k+1}.
\]

(26)

One may guess the form of this solution from the equation at time step \( N \) and verify it by constructing a recursive solution for \( \Phi_{k}, \phi_{k} \) in terms of \( \Phi_{k+1}, \phi_{k+1} \). Attaching (26) to a part of (25) results in

\[
\begin{bmatrix}
-1 \dot{A}_{k} \dot{B}_{k} \dot{D}_{k} & \dot{E}_{k} \\
\dot{B}_{k} \dot{E}_{k} & -1 \\
\end{bmatrix}
\begin{bmatrix}
\Delta \Gamma_{k-1} \\
\Delta \Gamma_{k} \\
\Delta \bar{X}_{k-1} \\
\Delta \bar{X}_{k} \\
\end{bmatrix}
= \begin{bmatrix}
R_{1,k} \\
R_{2,k} \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
\Delta \Lambda_{k} \\
\Delta \lambda_{k} \\
\end{bmatrix}
= \begin{bmatrix}
\dot{R}_{1,k} \\
\dot{R}_{2,k} \\
\end{bmatrix}
\]

(27)

Matching the solution to (27) with the coefficients in (26) yields a recursive equation of the form of a Riccati difference equation.

\[
\Phi_{k} = \phi_{k} - \dot{B}_{k} \dot{E}_{k}^{-1} \dot{R}_{k} \\
+ (\dot{B}_{k} - \dot{E}_{k} \dot{E}_{k}^{-1} \dot{R}_{k}) (I + \Phi_{k+1} \dot{E}_{k} \dot{E}_{k}^{-1} \dot{R}_{k}^{-1})^{-1} \times \Phi_{k+1} (\dot{B}_{k} - \dot{E}_{k} \dot{E}_{k}^{-1} \dot{R}_{k}) \\
+ \dot{B}_{k} \dot{E}_{k}^{-1} \dot{R}_{k} (I + \Phi_{k+1} \dot{E}_{k} \dot{E}_{k}^{-1} \dot{R}_{k}^{-1})^{-1} \times \left\{ \Phi_{k+1} (R_{3,k} - \dot{E}_{k} \dot{E}_{k}^{-1} \dot{R}_{2,k}) + \phi_{k+1} \right\}.
\]

(28)

Once \( \Phi_{k}, \phi_{k} \) are obtained through backward calculations, other variables such as \( \bar{u}_{k}, \Gamma_{k}, \bar{X}_{k+1} \), can be computed using the following equations through a forward calculation

\[
\Delta \bar{X}_{k+1} = (I + \dot{E}_{k} \dot{E}_{k}^{-1} \dot{E}_{k} T \Phi_{k+1})^{-1} (\dot{B}_{k} - \dot{E}_{k} \dot{E}_{k}^{-1} \dot{R}_{k}) \Delta \bar{X}_{k} \\
- (I + \dot{E}_{k} \dot{E}_{k}^{-1} \dot{E}_{k} T \Phi_{k+1})^{-1} (R_{3,k} - \dot{E}_{k} \dot{E}_{k}^{-1} \dot{R}_{2,k} + \phi_{k+1}).
\]

(29)

\[
\Delta \Gamma_{k} = (I + \Phi_{k+1} \dot{E}_{k} \dot{E}_{k}^{-1} \dot{E}_{k} T)^{-1} \left\{ \Phi_{k+1} (\dot{B}_{k} - \dot{E}_{k} \dot{E}_{k}^{-1} \dot{R}_{k}) \Delta \bar{X}_{k} \\
- \Phi_{k+1} (R_{3,k} - \dot{E}_{k} \dot{E}_{k}^{-1} \dot{R}_{2,k}) - \phi_{k+1} \right\}
\]

(30)

\[
\Delta \bar{u}_{k} = \dot{E}_{k}^{-1} (R_{2,k} - \dot{E}_{k} \Delta \bar{X}_{k} - \dot{E}_{k} T \Delta \Gamma_{k}).
\]

(31)

Thus, the step direction can be computed via Riccati iterations.

IV. Algorithm

For clarity, the previous section detailed the Riccati structure for a standard Newton iteration. However, the same structure applies when centering at the step length step is applied. For completeness, the actual algorithm used to solve the numerical example in this paper is enumerated below. It is an infeasible IP method because our initial path for starting the iterations is the unconstrained LQ solution.

1) Set a duality gap tolerance and solve unconstrained LQR problem.
2) Determine centering parameters (\( \sigma_{k} \in [0,1] \)) using Mehrotra’s algorithm [11].
3) Calculate the residuals of KKT condition \( f_{1,k}, f_{2,k}, \ldots \) using (32)-(35) (See the appendix).
4) Compute the matrices \( \bar{A}_{k}, \ldots, \bar{E}_{k} \) (39) and boundary condition \( \Phi_{N}, \phi_{N} \) (40).
5) Perform backward calculation of \( \Phi_{k}, \phi_{k} \) using the modified Riccati equation (28).
6) Perform forward calculation of \( \Delta \bar{u}_{k}, \Delta \bar{X}_{k} \) (29) and \( \Delta \bar{u}_{k} \), \( \Delta \bar{X}_{k} \) (31).
7) Calculate feasible step length and update variables (See [11]).
8) Repeat procedure from step 2 until the given duality gap condition is satisfied.

V. Numerical Simulation

We simulate three finite horizon cSLQ control problems with 2 states and 2 control inputs: One with state constraint
only, the other with control constraint only, and the last with both constraints. The parameter values for this problem are

\[
A = \begin{bmatrix}
1.02 & -0.1 \\
0.1 & 0.98
\end{bmatrix}, \quad B = \begin{bmatrix}
0.5 & 0.0505 \\
0.0505 & 0.5
\end{bmatrix}, \quad G = \begin{bmatrix}
0.3 & 0 \\
0 & 0.3
\end{bmatrix}, \\
Q = \begin{bmatrix}
10 & 0 \\
0 & 10
\end{bmatrix}, \quad R = \begin{bmatrix}
50 & 0 \\
0 & 50
\end{bmatrix}, \quad Q_f = \begin{bmatrix}
50 & 0 \\
0 & 50
\end{bmatrix}, \\
w_k \sim \mathcal{N}\left(\begin{bmatrix}0 \\ 0\end{bmatrix}, \begin{bmatrix}1 & 0 \\ 0 & 1\end{bmatrix}\right) \text{ iid.}
\]

We use \(\alpha = 95\%\) for the probabilistic constraints (the corresponding \(\gamma\) value is equal to 1.96). Constraint parameters are

\[
h_{x,1} = \begin{bmatrix}
-\frac{1}{\sqrt{5}} \\
-\frac{2}{\sqrt{5}}
\end{bmatrix}, \quad h_{u,1} = \begin{bmatrix}
0 \\
0
\end{bmatrix}, \quad g_1 = 3 \\
h_{x,2} = \begin{bmatrix}
0 \\
0
\end{bmatrix}, \quad h_{u,2} = \begin{bmatrix}
-\frac{0.4}{\sqrt{1.16}} \\
\frac{1}{\sqrt{1.16}}
\end{bmatrix}, \quad g_2 = 0.4.
\]

As shown above, \(h_{x,1}, h_{u,1}, g_1\) are used for the state constraint only simulation and \(h_{x,2}, h_{u,2}, g_2\) are used for the simulation with the control input constraint only. The final scenario uses both constraints simultaneously.

<table>
<thead>
<tr>
<th>Horizon length</th>
<th>IP on cSLQ</th>
<th>SDPT3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CPU time(sec)</td>
<td>Num. of Iteration</td>
</tr>
<tr>
<td>5</td>
<td>0.61</td>
<td>12</td>
</tr>
<tr>
<td>10</td>
<td>0.82</td>
<td>10</td>
</tr>
<tr>
<td>15</td>
<td>1.02</td>
<td>10</td>
</tr>
<tr>
<td>20</td>
<td>1.23</td>
<td>11</td>
</tr>
<tr>
<td>25</td>
<td>1.42</td>
<td>10</td>
</tr>
</tbody>
</table>

**TABLE I**

**METHOD COMPARISON : DIFFERENT HORIZON LENGTH**

<table>
<thead>
<tr>
<th>Horizon length : 10</th>
<th>Num. of states</th>
<th>Num. of inputs</th>
<th>IP on cSLQ</th>
<th>SDPT3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CPU time(sec)</td>
<td></td>
<td>CPU time(sec)</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>0.8</td>
<td>3.0</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>1.7</td>
<td>13.3</td>
<td>7</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>2.2</td>
<td>35.7</td>
<td>8</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td>5.2</td>
<td>382.1</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>7.8</td>
<td>941.1</td>
<td></td>
</tr>
</tbody>
</table>

**TABLE II**

**METHOD COMPARISON : DIFFERENT NUMBER OF STATES AND INPUTS**

Simulation results for a horizon of \(N = 25\) are shown in Figure 1, while Table I provides a comparison of computation time for different horizon lengths between the interior point method developed in this paper and the convex formulation of van Hessem and Bosgra [17]. As expected, the interior point algorithm’s advantage over the convex formulation increases as the horizon length increases. Table II also provides a comparison of the computation time for different sizes of the state and input. To solve the convex SDP optimization problem from van Hessem and Bosgra, we used CVX [6] with the SDPT3 solver [15]. Both tests were implemented with MATLAB code on a computer with a 1.8 GHz Intel Core2 processor.

**VI. CONCLUSION AND FUTURE WORK**

In this paper we develop a fast algorithm for solving probabilistically constrained Linear Quadratic control problems for receding horizon control. The algorithm exploits the recursive Riccati structure of the control problem to significantly improve computation time compared to using general purpose SDP solvers and a convex formulation. This performance advantage should allow one to tackle larger scale problems than those previously possible.

This basic approach can also be applied to linear dynamics with multiplicative noise under mean-variance constraints. Further, if one can extract mean and variance dynamics approximately for a system, this method can be applied to obtain approximate solutions even in the case of nonlinear dynamics.

**APPENDIX**

This appendix contains detailed expressions for the elements appearing in the matrix equation (25). First, define the following quantities corresponding to the equations in the KKT conditions. (Note that we omit the superscript \(i\).

\[
f_{1,k} = A\bar{x}_k + B\bar{u}_k - \bar{x}_{k+1}
\]

\[
f_{2,k} = (A + BK_k)\Sigma_k(A + BK_k)^T + G\Sigma_uG^T - \Sigma_{k+1} \tag{32}
\]

\[
f_{3,k} = Q\bar{x}_k + A^T\rho_k - \rho_{k-1} + \sum_{i=1}^n \left\{ -\lambda_{i,k}(h_{i,x}(h_{i,x}^T\bar{x}_k + h_{i,u}^T\bar{u}_k - g_i)h_{i,x} + h_{i,u}v_{i,k}) \right\} \tag{33}
\]

\[
F_{4,k} = \frac{1}{2} \left\{ Q + K_k^TRK_k + (A + BK_k)^T Z_k(A + BK_k) + \sum_{i=1}^n (\lambda_{i,k}r_i^T(h_{i,x} + K_k^Th_{i,u})(h_{i,x} + K_k^Th_{i,u})^T - Z_{k-1} \right\}
\]
\[
\begin{align*}
    f_{5,k} &= R \tilde{u}_k + B^T \rho_k \\
    &+ \sum_{i=1}^{n} \left\{-\lambda_{i,k} (h_i^T \tilde{x}_k + h_{u,i}^T \tilde{u}_k - g_i) h_{u,i} + h_{u,i} \nu_{i,k} \right\} \\
    F_{6,k} &= \left\{ (R + B^T Z_k B + \sum_{i=1}^{n} \rho_{i,k} H_{i,k}) K_k \right. \\
    &+ \left. (B^T Z_k A + \sum_{i=1}^{n} \lambda_{i,k} H_{i,k} A_k) \right\} \Sigma_k \\
    f_{j,k}^T &= \frac{1}{2} r_i (h_i^T \tilde{x}_k + K_i^T h_{u,i})^T \Sigma_k (h_i^T \tilde{x}_k + K_i^T h_{u,i}) \\
    &- \frac{1}{2} (h_i^T \tilde{x}_k + h_{u,i} \tilde{u}_k - g_i)^T t_i \Sigma_k \\
    f_{j,k}^1 &= k \lambda_{i,k} h_{i,k} h_{i,k}^T, \quad f_{j,k}^2 = k \nu_{i,k} v_i, \quad f_{j,k}^3 = k \nu_{i,k} v_i, \quad f_{j,k}^4 = 0, \quad \nu_{i,k} \geq 0.
\end{align*}
\]

Additionally, define the following
\[
\begin{align*}
    S_{i,k}' &= h_i^T \tilde{x}_k + h_{u,i}^T \tilde{u}_k - g_i, \quad S_{i,N}' = h_i^T \tilde{x}_N + g_i \\
    M_{i,k} &= (h_i + K_i^T h_{u,i}) (h_i + K_i^T h_{u,i})^T, \quad M_{i,N} = H_i \\
    M_{i,k} &= h_{u,i} (h_i + K_i^T h_{u,i})^T \Sigma_k \\
    M_{i,k} &= (R + B^T Z_k B + \sum_{i=1}^{n} \lambda_{i,k} R_i^T H_{i,k}^T) \\
    M_{i,k} &= M_{i,N} K_i + (B^T Z_k A + \sum_{i=1}^{n} \lambda_{i,k} H_{i,u} H_i) \\
    M_{i,k} &= (I + T) \left[ (A + B K_i) \Sigma_k \otimes B \right] \\
    M_{i,k} &= (A + B K_i) \otimes (A + B K_i) \\
    M_{i,k} &= (I + T) \left[ (A + B K_i) \Sigma_k \otimes \Sigma_k \right]
\end{align*}
\]

where \( \otimes \) denotes the kronecker product, and let
\[
\begin{align*}
    a^i_k &= -\lambda_{i,k} - \frac{\lambda_{i,k} \lambda_{i,k}^T}{t_i k} + \frac{v_i, k}{t_i k} h_{u,i} \\
    b^i_k &= \frac{\lambda_{i,k} \lambda_{i,k}^T}{2 t_i k} s_{i,k} \\
    c^i_k &= \lambda_{i,k} \Sigma_k \\
    r_{1,k} &= -f_{3,k} + \sum_{i=1}^{n} \left\{ S_{i,k}' \left( \frac{\lambda_{i,k}}{t_i k} f_{j,k}^4 - \frac{1}{t_i k} (1 - \sigma_i) f_{j,k}^4 \right) \right\} h_{u,i} \\
    r_{2,k} &= -F_{4,k} - \sum_{i=1}^{n} \left\{ \frac{r_i^2}{2} \left( \frac{\lambda_{i,k}}{t_i k} f_{j,k}^4 - \frac{1}{t_i k} (1 - \sigma_i) f_{j,k}^4 \right) \right\} M_{i,k} \\
    r_{3,k} &= -f_{5,k} + \sum_{i=1}^{n} \left\{ S_{i,k}' \left( \frac{\lambda_{i,k}}{t_i k} f_{j,k}^4 - \frac{1}{t_i k} (1 - \sigma_i) f_{j,k}^4 \right) \right\} h_{u,i} \\
    r_{4,k} &= -F_{6,k} - \sum_{i=1}^{n} \left\{ \frac{r_i^2}{2} \left( \frac{\lambda_{i,k}}{t_i k} f_{j,k}^4 - \frac{1}{t_i k} (1 - \sigma_i) f_{j,k}^4 \right) \right\} M_{i,k} \\
    r_{5,k} &= -f_{1,k} \\
    r_{6,k} &= -F_{2,k}.
\end{align*}
\]

Then \( \alpha_k, \beta_k, \gamma_k, \delta_k, \epsilon_k, \rho_k, R_k, S_k, R_{\alpha}, \rho_{\beta} \) and the boundary conditions \( \Phi_N, \Phi_0 \) are given by
\[
\begin{align*}
    \alpha_k &= \left[ Q + \sum_{i=1}^{n} (a^i_k H_i^T) + \sum_{i=1}^{n} b_{i,k} h_{u,i} M_{i,k}^T \right] \\
    \beta_k &= \left[ \sum_{i=1}^{n} (a^i_k M_{i,k}^T h_{u,i}) + \sum_{i=1}^{n} (c_{i,k} M_{i,k} M_{i,k}^T) + M_{i,k} \right]
\end{align*}
\]

REFERENCES


