Fast Evaluation of Quadratic Control-Lyapunov Policy

Yang Wang and Stephen Boyd, *Fellow, IEEE*

***Abstract*—The evaluation of a control-Lyapunov policy, with quadratic Lyapunov function, requires the solution of a quadratic program (QP) at each time step. For small problems this QP can be solved explicitly; for larger problems an online optimization method can be used. For this reason the control-Lyapunov control policy is considered a computationally intensive control law, as opposed to an "analytical" control law, such as conventional linear state feedback, linear quadratic Gaussian control, or $\mathbf{H}_\infty$, too complex or slow to be used in high speed control applications. In this note we show that by precomputing certain quantities, the control-Lyapunov policy can be evaluated extremely efficiently. We will show that when the number of inputs is on the order of the square-root of the state dimension, the cost of evaluating a control-Lyapunov policy is on the same order as the cost of evaluating a simple linear state feedback policy, and less (in order) than the cost of updating a Kalman filter state estimate. To give an idea of the speeds involved, for a problem with 100 states and 10 inputs, the control-Lyapunov policy can be evaluated in around 67 μs, on a 2 GHz AMD processor; the same processor requires 40 μs to carry out a Kalman filter update.**

***Index Terms*—Approximate dynamic programming, model predictive control (MPC), optimization-based control, real-time convex optimization, stochastic control.**

I. Introduction and Background

THIS note concerns the evaluation of a particular type of feedback control policy, called a quadratic control-Lyapunov policy, which requires the solution of convex quadratic program (QP) at each step. Solving a QP is generally considered a serious computational task, but we will see that such control laws can be evaluated very quickly, with a computational complexity often comparable to, or even smaller than, that required for an "analytical" modern control law, or to implement a Kalman filter state estimate. Even for a system with 30 states and 10 actuators, each with lower and upper limits, the policy can be evaluated in around 51 μs on a 2 GHz AMD processor, allowing sample rates exceeding 10 kHz. Our conclusion is that a quadratic control-Lyapunov controller should be considered practical for a broad range of applications, including those with fast sample rates.

In Sections I-A and I-B we briefly describe the constrained linear stochastic control problem and its solution via dynamic programming. This material is used only to motivate the control law we consider, which is described in Sections I-C and I-D. In Section III, we describe methods for fast evaluation of the control law, and in Section IV, we give some experimental results demonstrating fast evaluation.

Manuscript received March 30, 2010; accepted June 24, 2010. Manuscript received in final form June 29, 2010. Date of publication August 16, 2010; date of current version June 17, 2011. Recommended by Associate Editor B. de Jager. The work of Y. Wang was supported by a Rambus Corporation Stanford Graduate Fellowship. This work was supported in part by the Precourt Institute on Energy Efficiency, by NSF Award 0529426, by NASA Award NNX07AEIIA, by AFOSR Award FA9550-06-1-0514, and by AFOSR Award FA9550-06-1-0312

The authors are with the Department of Electrical Engineering, Stanford University, Stanford, CA 94305 USA (e-mail: yw224@stanford.edu; boyd@stanford.edu).

Digital Object Identifier 10.1109/TCST.2010.2056371

A. Linear Stochastic Control Problem

The material of this section is given only to motivate the control-Lyapunov policy; fast evaluation of such a policy, our main focus, does not depend on the problem or assumptions stated here. We consider a discrete-time linear time-invariant system, with dynamics

$$x_{t+1} = Ax_t + Bu_t + w_t, \quad t = 0, 1, \ldots$$

where $x_t \in \mathbf{R}^n$ is the state, $u_t \in \mathbf{R}^m$ is the control input, $w_t \in \mathbf{R}^n$ is the process noise or exogeneous input, $A \in \mathbf{R}^{n\times n}$ is the dynamics matrix, and $B \in \mathbf{R}^{n\times m}$ is the input matrix. We assume that $w_1, w_2, \ldots$ are independent identically distributed (IID) with mean $\bar{w} = \mathbf{E}w_t$. The objective function is the average state cost

$$J = \limsup_{N\to\infty} \frac{1}{N}\mathbf{E}\sum_{t=0}^{N-1} \ell(x_t, u_t)$$

where $\ell : \mathbf{R}^n \times \mathbf{R}^m \to \mathbf{R}$ is the stage cost function. We consider state feedback control policies, i.e.,

$$u_t = \phi(x_t), \quad t = 0, 1, \ldots$$

where $\phi : \mathbf{R}^n \to \mathcal{U} \subseteq \mathbf{R}^m$ is the state feedback function or policy, and $\mathcal{U}$ is a non-empty input constraint set. The stochastic control problem is to find a policy ϕ that minimizes J.

B. Optimal Control Policy

It is well known (see, e.g., [1], [2]) that an optimal feedback function has the form

$$\phi^\star(z) = \operatorname*{argmin}_{v\in\mathcal{U}} \{\ell(z, v) + \mathbf{E}V(Az + Bv + w_t)\} \quad (1)$$

where $V : \mathbf{R}^n \to \mathbf{R}$ is called the value function or Bellman function, and satisfies the Bellman or dynamic programming equation

$$V(z) + J^\star = \min_{v\in\mathcal{U}} \{\ell(z, v) + \mathbf{E}V(Az + Bv + w_t)\}$$

for all $z \in \mathbf{R}^n$, where $J^\star$ is the optimal objective value. (Here we overlook many technical details, including various pathologies that can occur; see, e.g., [1]–[4].)

The value function V, and the optimal control policy $\phi^\star$ can be effectively found in only a few special cases. One well known case is when there are no constraints, i.e., $\mathcal{U} = \mathbf{R}^m$, and the stage cost ℓ is a convex quadratic function. In this case the value

1063-6536/$26.00 © 2010 IEEE

function is convex quadratic, and the optimal state feedback function is affine, i.e., linear plus a constant; the coefficients of these functions can be effectively computed. But in general it is difficult to compute the value function, or the optimal policy, exactly.

C. Control-Lyapunov Policy

A control-Lyapunov policy, also called approximate dynamic programming or horizon-1 model predictive control, is a suboptimal policy inspired by the characterization of the optimal policy given in (1). We replace the value function V with an approximate or surrogate value function $V_{\mathrm{clf}} : \mathbf{R}^n \to \mathbf{R}$, called the control-Lyapunov function [5]–[8]. The policy is given by

$$\phi_{\mathrm{clf}}(z) = \operatorname*{argmin}_{v \in \mathcal{U}} \{\ell(z,v) + \mathbf{E} V_{\mathrm{clf}}(Az + Bv + w_t)\}. \quad (2)$$

When $V_{\mathrm{clf}} = V$, the control-Lyapunov policy is optimal. But the control-Lyapunov policy has been found to provide good control performance even when V_{clf} is a fairly crude approximation of V. Many methods for selecting V_{clf} have been proposed; a brief list of these appears in Section II-B.

To evaluate ϕ_{clf} at z, we must solve an optimization problem, so V_{clf} is chosen in part so that this problem can be solved with reasonable effort. When the minimization problem is not easily solved, the argmin can be computed approximately, by a local optimization method, solving a convex relaxation, or replacing $\ell(z,v)$ with an approximation.

The purpose of this note is not to argue for or against the use of a control-Lyapunov policy, or any particular method for finding a suitable control-Lyapunov function. Instead our focus is on the fast evaluation of a control-Lyapunov function, when $\mathcal{U}$ is a polyhedron and V_{clf} is quadratic, in which case evaluation requires the solution of a QP.

D. Quadratic Control-Lyapunov Function

In this note we focus on the specific case where the stage cost is convex and quadratic, and separable in state and control

$$\ell(z,v) = z^T Q z + 2q^T z + v^T R v + 2r^T v \quad (3)$$

where $Q = Q^T \geq 0$, $R = R^T > 0$, and the input set is a polyhedron

$$\mathcal{U} = \{v \,|\, Fv \leq h\} \quad (4)$$

where $F \in \mathbf{R}^{k \times m}, h \in \mathbf{R}^k$. (Our method however can easily be extended to more general convex stage costs and convex input constraint sets.) A common special case is when $\mathcal{U}$ is a box, i.e.,

$$\mathcal{U} = \{v \,|\, u^{\min} \leq v \leq u^{\max}\} \quad (5)$$

corresponding to $k = 2n$ and

$$F = \begin{bmatrix} I \\ -I \end{bmatrix}, \quad h = \begin{bmatrix} u^{\max} \\ -u^{\min} \end{bmatrix}.$$

We further assume that the control-Lyapunov function is convex and quadratic, with the form

$$V_{\mathrm{clf}}(z) = z^T P z + 2p^T z \quad (6)$$

with $P = P^T \geq 0, p \in \mathbf{R}^n$.

With quadratic stage-cost (3) and quadratic control-Lyapunov function (6), we can explicitly evaluate the expectation appearing in the control-Lyapunov feedback function (2), and write it as

$$\phi_{\mathrm{clf}}(z) = \operatorname*{argmin}_{Fv \leq h} \{v^T (R + B^T P B) v + 2(r + B^T P A z + B^T P \bar{w} + B^T p)^T v\}. \quad (7)$$

To evaluate the quadratic control-Lyapunov policy (7) for a given value of z, we must solve a QP

$$\begin{array}{ll} \text{minimize} & v^T \tilde{R} v + 2\tilde{r}^T v \\ \text{subject to} & Fv \leq h \end{array} \quad (8)$$

where

$$\tilde{R} = R + B^T P B, \quad \tilde{r} = r + B^T P A z + B^T P \bar{w} + B^T p$$

with variable $v \in \mathbf{R}^m$. The data $A, B, R, r, \bar{w}, P, p, F$, and h are known and constant, i.e., do not change in each iteration. The data $z \in \mathbf{R}^n$ (which represents the current system state) changes in each iteration. Note that the QP (8) is always feasible, since we assume (see Section I-A) that the set $\mathcal{U}$ is non-empty.

The data dimensions are n, m, and k, so the computational complexity of solving the QP (8) depends on these. In this note we will show that by precomputing certain quantities (offline), we can reduce the (online) solution of this QP to an $m \times n$ matrix-vector multiply, followed by the solution of a QP with data dimensions m and k only. Coupled with an efficient method for solving this QP, this gives a method for solving the QP (8), i.e., evaluating the quadratic control-Lyapunov policy (7), that is very efficient. For example, we will see that the complexity of evaluating the control-Lyapunov policy grows only linearly with state dimension. As a practical matter, we will see that the control-Lyapunov policy can be evaluated on current processors in time measured in *microseconds*, enabling its use for systems with high sample rates.

II. Related Work

In this section we describe prior work related to our topic, fast evaluation of quadratic control-Lyapunov feedback control policies. None of this material is used in the sequel.

A. Fast Solution of QPs and Fast MPC

In this section we describe various fast methods for solving QPs that arise in control problems. Many of these methods were developed specifically for model predictive control (MPC), which is a popular control method in many applications areas, and goes by other names such as receding horizon control, rolling horizon planning, and dynamic matrix control [9], [10]. In MPC, a QP is solved at each time step to compute a sequence of inputs to apply over a fixed horizon. The first input in this sequence is executed, and at the next time step the process is repeated. The control-Lyapunov policy can be viewed as a special case of MPC when the planning horizon is 1. In particular, all methods for fast evaluation of an MPC policy can be applied to fast evaluation of a control-Lyapunov policy.

There are several methods we can use to solve QPs that arise in MPC. These are roughly divided into two categories: online methods and explicit offline methods. In an online method, an iterative optimization algorithm such an active set method [9], [11], [12] or an interior-point method [13], [14] is used to solve the QP at each time step. Here, there are several techniques we can use to speed up the optimization. The main strategy is to exploit the structure of the QPs that arise in MPC [15]–[21]. The QP can also be solved approximately, since it is often observed that high quality control is obtained even when the QPs are not solved to full accuracy [15], [19], [21]–[26]. In addition, the computation time can be improved via warm-starting, where the optimization algorithm is initialized with the input sequence computed in the previous time period [15], [22], [27], or with an approximation (typically derived from the associated unconstrained linear controller). For example, for a (medium-sized) control problem with $n = 10$ states, $m = 3$ inputs, and a horizon $T = 30$, these methods allow us to solve MPC QPs within *milliseconds* [15], [16].

The second method is to compute the solution of the QP as an explicit function of the current state. This is called multiparametric quadratic programming, and was originally proposed by Bemporad *et al.* [28]. The explicit solution decomposes the state-space into polyhedral regions, and within each region we apply a precomputed affine control law. Online evaluation of the control policy reduces to searching through a lookup table, and applying the appropriate affine function. In the past ten years, there has been a large volume of work on algorithms for efficiently computing the explicit solution (see, e.g., [29]–[31]), as well as methods for reducing complexity of the explicit solution [32]–[34], which is exponential in the size of the QP in the worst case. In addition, many sophisticated methods exist for searching through the lookup table of regions [30], [35], [36]. In cases when the number of regions is small, explicit methods are extremely fast, and have the additional advantages of simplicity and transparency. Indeed, the explicit solution has been successfully applied to a variety of control problems, such as hybrid traction control [37], flow control in supersonic diffusers [38], magnetic actuators in satellites [39], chemical process plants [40] as well as many others.

We will discuss these methods in more detail in the context of evaluating the CLF policy in sections Sections III-A and III-B.

B. Finding a Control-Lyapunov Function

Here we describe some of the many methods that can be used to find a control-Lyapunov function V_{clf} for which evaluating $\phi_{\text{clf}}(z)$ is tractable, and which results in (one hopes) good quality of control, i.e., a small (if not optimal) value of J; see, e.g., [41]–[43]. In recent work, Wang and Boyd developed a tractable method that yields a provable lower bound on $J^\star$, as well as a candidate quadratic control-Lyapunov function [44], [45]. In a different context (finite state and input space) De Farias and Van Roy proposed a method for finding an approximate value function as a linear combination of basis functions, optimizing over the weights by solving a linear program [41].

Approximate dynamic programming (also known as neuro-dynamic programming or reinforcement learning) is another general method for finding a suitable control-Lyapunov function. Here too V_{clf} is taken to be a linear combination of some given basis functions; the coefficients are adjusted through simulation of the system. One method in this area is called temporal difference (TD-λ) learning [46]–[48], which is a stochastic gradient method for adjusting the coefficients in the surrogate value function. Another approach is to use projected value iteration (PVI), which involves applying the Bellman operator followed by projection onto the span of the basis functions [46]. A practical implementation of PVI based on simulating the system is called least squares policy evaluation (LSPE-λ), and has become a popular method for finding surrogate value functions, with provable convergence to the same limit as PVI [2], [46], [49], [50]. Another popular (but slightly different) method is called Q-learning. The Q-learning algorithm attempts to approximate the Q-factor, which is the expression that is minimized at the right-hand side of the Bellman equation. The Q-factors are updated via simulation-based value iteration. One major advantage of Q-learning is that an explicit dynamical system description is not needed; only the ability to simulate the system is required [2], [46], [51], [52]. There are many other variations/improvements on these methods, for example, least squares temporal difference method (LSTD-λ) combines techniques from both TD-λ and LSPE-λ learning [50], [52], [53].

III. Evaluation of Quadratic Control-Lyapunov Policy

In this section we discuss methods for evaluating the quadratic control-Lyapunov policy (7), i.e., solving the QP (8). Our first step is to show that we can reduce this to the evaluation of an affine feedback function, followed by the solution of a QP with problem dimensions involving only m (the number of actuators) and k (the number of constraints).

We first rewrite the control-Lyapunov policy (7) as

$$\phi_{\text{clf}}(z) = \operatorname*{argmin}_{Fv \le h} \{(v-\tilde{u})^T \tilde{R}(v-\tilde{u})\}$$

where

$$\begin{aligned} \tilde{R} &= R + B^T P B \\ \tilde{u} &= Kz + g \\ K &= -(R + B^T P B)^{-1} B^T P A \\ g &= -(R + B^T P B)^{-1}(r + B^T P \bar{w} + B^T p). \end{aligned}$$

The matrices $\tilde{R}$ and K, and the vector g, can be computed offline, i.e., ahead of time. At run-time, we can compute $\tilde{u}$ from the state z from $\tilde{u} = Kz + g$. Note that $\tilde{u} = Kz + g$ is the control-Lyapunov policy for the associated unconstrained stochastic control problem, i.e., the one with the same stage cost and control-Lyapunov function, and $\mathcal{U} = \mathbf{R}^m$.

To evaluate $\phi_{\text{clf}}(z)$ we first form $\tilde{u} = Kz + g$, and then solve the QP

$$\begin{array}{ll} \text{minimize} & (v-\tilde{u})^T \tilde{R}(v-\tilde{u}) \\ \text{subject to} & Fv \le h \end{array} \tag{9}$$

with variable v, so $\phi_{\text{clf}}(z) = v^\star$. This QP has data $\tilde{R} \in \mathbf{R}^{m\times m}$, $F \in \mathbf{R}^{k\times m}$, $h \in \mathbf{R}^k$, and $\tilde{u} \in \mathbf{R}^m$. In particular, none of the data involve the state dimension n. The state dimension n only

comes up in the first step, evaluating $\tilde{u} = Kz+g$, i.e., evaluating an affine control law.

If $\tilde{u}$ satisfies $F\tilde{u} \leq h$, then it is optimal for the QP (9), so $\phi_{\rm clf}(z) = \tilde{u}$, and no further computation is needed. The control-Lyapunov policy can therefore be evaluated as follows.

1) Compute $\tilde{u} = Kz + g$.
2) If $F\tilde{u} \leq h$, $\phi_{\rm clf}(z) = \tilde{u}$. Otherwise, solve the QP (9); set $\phi_{\rm clf}(z) = v^\star$.

The computational effort of the first step is that of a matrix-vector multiply, which costs $2mn$ flops. The complexity of the second step depends on the method used to solve the QP (9), which can only depend on m and k (and, possibly, the sparsity patterns of $\tilde{R}$ and F).

In our computational effort comparisons, we define one flop (floating-point operation) to be one multiplication, subtraction, addition, or division of two floating-point numbers. We will often simplify a flop count to include only the leading terms. For example, if a particular algorithm requires m^2k+m^3+mk+k, we would simplify this to $m^2k + m^3$.

Many methods can be used to solve the QP (9), including active-set methods, various first-order methods (such as projected gradient), interior-point methods, and explicit analytical solution via multiparametric optimization. We will consider here the explicit solution method and interior-point methods, both of which have been used for fast solution of QPs for control purposes; other methods that we do not describe (such as active set methods) can also be used for fast evaluation of a control-Lyapunov policy; see, e.g., [11], [12], [54]. The reason we do not describe active-set methods in detail is because their computational effort is very similar to that of an interior-point method—both methods require solving a similar system of equations in each step. While the worst case complexity of an active-set method is exponential in problem size, for most practical applications the method converges to the optimal solution in surprisingly few iterations. Thus, in the sequel we will focus exclusively on interior-point methods, with the understanding that active set methods can also achieve similar computation times.

We will see in our complexity comparisons that for certain problem sizes the effort of evaluating the control-Lyapunov policy via an interior-point method is comparable to the effort of evaluating a linear state feedback, or Kalman filter update. This shows that for many problems the control-Lyapunov policy can be essentially considered an "analytic" policy.

A. Explicit Solution of QP

The solution of the QP (9) can be shown to be a piecewise linear function of $\tilde{u}$, with the regions determined by the set of active constraints, i.e., the entries of the vector inequality $Fv \leq h$ that hold with equality. When m and k are small enough, we can explicitly compute the coefficients for each region, as well as the inequalities defining the regions. At run-time, we first determine which region $\tilde{u}$ lies in, and then compute the appropriate affine function of $\tilde{u}$.

To see how this works, we first write the necessary and sufficient optimality (KKT) conditions for the QP (9) as

$$2\tilde{R}(v - \tilde{u}) + F^T\lambda = 0, \quad Fv \leq h$$

$$\lambda \geq 0, \quad \lambda_i(f_i^T v - h_i) = 0, \quad i = 1, \ldots, k$$

where f_i^T denotes the ith row of F, and $\lambda \in \mathbf{R}^k$ is a vector of Lagrange multipliers associated with the constraint $Fv \leq h$. For a given feasible v, let $\mathcal{I} \subseteq \{1, \ldots, k\}$ denote the set of indices of the active constraints, i.e., the set of i for which $f_i^T v = h_i$, and let $\backslash\mathcal{I}$ denote its complement. The KKT conditions can then be expressed as

$$\begin{bmatrix} 2\tilde{R} & F_{\mathcal{I}}^T \\ F_{\mathcal{I}} & 0 \end{bmatrix} \begin{bmatrix} v \\ \lambda_{\mathcal{I}} \end{bmatrix} = \begin{bmatrix} 2\tilde{R}\tilde{u} \\ h \end{bmatrix}, \quad F_{\backslash\mathcal{I}} v \leq h_{\backslash\mathcal{I}}, \lambda_{\mathcal{I}} \geq 0. \tag{10}$$

Here $F_{\mathcal{I}}$, $h_{\mathcal{I}}$, and $\lambda_{\mathcal{I}}$ are matrices formed from the rows of F, and entries of h and λ, corresponding to the indices in $\mathcal{I}$. For simplicity we will assume that $F_{\mathcal{I}}$ is full row rank (the explicit method still works when this is not the case, but is slightly more complicated; see, e.g., [28], [30]). This implies that the block matrix in (10) is nonsingular (since $\tilde{R} > 0$), and so for any $\tilde{u}$ there is a unique solution to the linear equations in (10), which moreover is a linear function of the right-hand side. Thus we can write

$$v = L^{(\mathcal{I})}\tilde{u} + l^{(\mathcal{I})}, \quad \lambda = C^{(\mathcal{I})}\tilde{u} + d^{(\mathcal{I})}$$

for appropriate matrices $L^{(\mathcal{I})} \in \mathbf{R}^{m\times m}$, $l^{(\mathcal{I})} \in \mathbf{R}^m$, $C^{(\mathcal{I})} \in \mathbf{R}^{k\times m}$, and $d^{(\mathcal{I})} \in \mathbf{R}^k$. (These matrices can be found using standard linear algebra methods.) So for a fixed set of active constraints, the optimal v is an affine function of $\tilde{u}$. Substituting this expression for v into the other KKT conditions we obtain

$$F_{\backslash\mathcal{I}}\left(L^{(\mathcal{I})}\tilde{u} + l^{(\mathcal{I})}\right) \leq h_{\backslash\mathcal{I}}, \quad C^{(\mathcal{I})}\tilde{u} + d^{(\mathcal{I})} \geq 0 \tag{11}$$

which defines the set of $\tilde{u}$ that are consistent with our active set assumption. Thus, if $\tilde{u}$ satisfies (11), then the solution of the QP is $v^\star = L^{(\mathcal{I})}\tilde{u} + l^{(\mathcal{I})}$. For any value of $\tilde{u}$, the inequalities (11) hold for at least one index set $\mathcal{I}$.

We can solve the QP quickly as follows. Offline, we compute $L^{(\mathcal{I})}$, $l^{(\mathcal{I})}$, $C^{(\mathcal{I})}$, and $d^{(\mathcal{I})}$ for each of the 2^k possible active sets. (We can also carry out a pruning step, in which we analyze the sets of linear inequalities (11), removing those that are inconsistent.) At run-time, we proceed as follows. Given $\tilde{u}$, we search until we find an active index set $\mathcal{I}$ for which the inequalities (11) are satisfied. Then, $v^\star = L^{(\mathcal{I})}\tilde{u} + l^{(\mathcal{I})}$. Checking the inequalities (11) for each $\mathcal{I}$ requires up to $4km + 2m^2$ flops, so searching through the active sets requires at most $2^k(2m(2k + m)) = 2^{k+1}m(2k+m)$ flops (since there are 2^k possible active sets that must be checked in the worst case). Evaluating $v^\star = L^{(\mathcal{I})}\tilde{u}+l^{(\mathcal{I})}$ costs order $2m^2$ flops, so the overall computational effort of evaluating the explicit solution is $2^{k+1}m(2k+m)$ operations in the worst case.

We should emphasize that this is a highly conservative worst-case estimate of the computational effort. For many problem instances the number of active sets we obtain is often much smaller than 2^k. For example, when the constraint set is a box, we have $k = 2m$ inequalities, so the complexity bound above is $5 \cdot 2^{2m+1}m^2$. But for a box, the number of possible active sets is only 3^m, corresponding to the three possible cases for each i: $v_i < u_i^{\min}$, $u_i^{\min} \leq v \leq u_i^{\max}$, and $v_i > u_i^{\max}$. Checking the inequalities (11) requires up to $6m^2$ flops, so the total flop count is no more than $6 \cdot 3^m m^2$, which is smaller than $5 \cdot 2^{2m+1}m^2$.

We should also point out that the algorithm we have described is a simple, but crude implementation. In fact, many methods can be used to speed up evaluation of the explicit QP solution, including more sophisticated search methods, as well as obtaining approximate (suboptimal) solutions with fewer active constraint sets; see, e.g., [10], [30], [32]–[36]. Nevertheless, for m and k larger than ten or so, the explicit method becomes less practical. However, for problems with small m and k, or more accurately, for problems in which the total number of regions is modest, the explicit method is entirely practical.

B. Interior-Point Methods

The QP (9) can also be solved using an interior-point method [13], [14], [55]. In practice interior-point methods require no more than a few tens of iterations to compute the solution of a QP; for control purposes, however, an accurate enough solution can generally be computed in just a few iterations, and certainly fewer than ten or so [15], [16], [18], [19], [21]–[23]. There are several types of interior-point algorithms, including primal barrier, affine scaling, and primal-dual methods. But they are all similar, and involve roughly the same steps.

Some interior-point methods need to be initialized with a strictly feasible primal point. To do this, we find any point v_0 that satisfies $Fv_0 < h$. This step can be done offline, and is often trivial: for example when the constraint set is a box, we can take v_0 as the center of the box. Online, when we are given $\tilde{v}$, we initialize the algorithm with $v^{(0)} = \theta\tilde{v} + (1-\theta)v_0$, where $\theta \in [0,1]$ is chosen so that $Fv^{(0)} < g$. (If $\theta = 1$ works here, $\tilde{v}$ is already optimal.)

The iterations then proceed as follows. At iteration i we compute primal and dual search directions $(\Delta v^{(i)}, \Delta\lambda^{(i)}) \in \mathbf{R}^m \times \mathbf{R}^k$ by solving a system of equations of the form

$$\begin{bmatrix} 2\tilde{R} & F^T \\ F & D^{(i)} \end{bmatrix} \begin{bmatrix} \Delta v^{(i)} \\ \Delta\lambda^{(i)} \end{bmatrix} = -r^{(i)} \tag{12}$$

where $D^{(i)} \in \mathbf{R}^{k\times k}$ is a positive definite diagonal matrix, and $r^{(i)} \in \mathbf{R}^{m+k}$. These depend on the type of interior-point method we use, as well as the current point $(v^{(i)}, \lambda^{(i)})$, and change in each iteration. (In most primal-dual methods, two search directions are computed, using the same coefficient matrix, but two different right-hand sides. Since the factorization of the coefficient matrix dominates the solution time, this does not take significantly more time; see, e.g., [55].) Once the search directions have been computed, we update our primal and dual points as

$$v^{(i+1)} = v^{(i)} + s^{(i)}\Delta v^{(i)}, \quad \lambda^{(i+1)} = \lambda^{(i)} + s^{(i)}\Delta\lambda^{(i)}$$

where $s^{(i)} \in (0,1]$ is a step size that is appropriately chosen. (The details of the line search depend on the particular method.)

The computational effort of the interior-point method is dominated by solving the system of (12); the line searches, and the cost of computing $D^{(i)}$ and $r^{(i)}$ are (relatively) insignificant. By eliminating $\Delta\lambda^{(i)}$, the (12) can be reduced to a system of the form

$$(2\tilde{R} + F^T(D^{(i)})^{-1}F)\Delta v^{(i)} = q^{(i)}$$

where $q^{(i)} \in \mathbf{R}^m$. Forming the matrix $2\tilde{R} + F^T(D^{(i)})^{-1}F$ requires m^2k flops; factoring it and solving the symmetric positive definite system of (12) requires $(1/3)m^3$ flops. Assuming that the interior-point method requires ten iterations, the total computational effort is around $m^2(10k + 3m)$ operations.

For the particular case of box constraints, $F^T(D^{(i)})^{-1}F$ is diagonal, and requires order k flops to form. In this instance, the total computational effort of each step is dominated by the effort of solving the system (12), which costs $(1/3)m^3$ operations. Thus, assuming we take around 10 steps, the overall effort is approximately $3m^3$ flops.

The explicit QP solution method and interior-point methods are more closely related than might first appear. We can interpret the system of (10) arising in the explicit method as a limiting case of the (12) arising in an interior-point method, with D_{ii} converging to 0 or ∞, depending on whether the constraint i is active or not. Thus, we can think of the explicit method as pre-solving the KKT systems for all possible active sets ahead of time; or, we can think of an interior-point method as one that works out what the active set is online, in around ten iterations.

TABLE I
COMPARISON OF COMPUTATIONAL EFFORT FOR SOLVING THE QP (9)

constraints	explicit solution	interior-point method
general	$2^{k+1}m(2k+m)$	$m^2(10k+3m)$
box	$2\cdot 3^{m+1}m^2$	$3m^3$

C. Summary and Complexity Comparison

Table I summarizes the computational effort estimates for both the explicit solution method and the interior-point method for solving the QP (9). Our estimates also distinguish between general linear inequality constraints (4), and box constraints (5). When k is on the same order as m (which is very often the case), the interior-point method effort is a modest multiple of m^3.

Now we consider the computational complexity of evaluating the quadratic control-Lyapunov feedback function. The cost of evaluating the affine feedback function $\tilde{u} = Kz + g$ is $2mn$ flops; the cost of solving the QP (9) is a modest multiple of m^3, assuming k is no more than a small multiple of m. Thus, when m is on the order of $\sqrt{n}$, the cost of solving the QP is order m^3, *the same as evaluating an affine feedback function*. This is quite surprising, since an affine feedback function is among the simplest possible feedback control laws, whereas methods that require solution of a QP, such as a control-Lyapunov or MPC policy, are generally (and incorrectly) considered computationally demanding.

We can also compare the cost of evaluating a quadratic control-Lyapunov policy with the effort of computing a Kalman filter state estimate update. At each step this requires evaluating

$$\hat{x}_t = A\hat{x}_{t-1} + Bu_{t-1} + L_t(y_t - C(A\hat{x}_{t-1} + Bu_{t-1}))$$

where $y_t \in \mathbf{R}^k$ is a vector of outputs, $C \in \mathbf{R}^{k\times n}$ is the output matrix, and L_t is an appropriate filter gain matrix, which requires around $2n^2 + 2nm + 4nk$ flops. When m is on the order of $\sqrt{n}$, we see that the Kalman filter update cost is order m^4, *larger than the cost of evaluating a quadratic control-Lyapunov policy*. Indeed, the cost of computing a Kalman filter update is

comparable with the cost of evaluating the control-Lyapunov policy as long as m is no more than on the order of $n^{2/3}$.

Though not difficult to derive, these results seem remarkable to us. Most control engineers would consider affine state feedback policies and Kalman filters (or the combination, an estimated-state linear feedback policy) to be implementable and practical, even for medium size systems and relatively high sample rates. Many of the same control engineers would think of a control-Lyapunov policy as complicated and slow to evaluate, and not practical for medium size systems and fast sample rates. Our observation is that, when m and k are on the order of $\sqrt{n}$, *a control-Lyapunov feedback function can be evaluated with effort not much more than that required for a simple affine feedback function*, and *less than the effort required to update a Kalman filter*. In particular, it is entirely practical even for medium size systems and fast sample rates.

IV. Implementation Results

To demonstrate fast evaluation of a quadratic control-Lyapunov feedback function we developed a simple C implementation of an interior-point method, using the LAPACK library [56], [57] for the linear algebra operations. The simulations were carried out on a 2 GHz AMD Opteron, running Linux. The particular method we used was primal-dual, with fixed duality parameter κ (as in [15], where, however, a primal barrier method was used). For all of our examples we found that five steps provided more than adequate accuracy for any feedback control purpose. We made no attempt to optimize the code for fast execution, so substantial further reductions in computation time are likely possible. Full source code for our implementation is available from the authors' web sites.

We report the maximum time taken to evaluate the control-Lyapunov policy (including the initial affine policy evaluation $\tilde{v} = Kz + g$) for ten problems with different dimensions. The evaluation time does not depend on the particular data values, but for completeness we explain how the data were generated. The matrices A, B, F, and g were chosen randomly. The matrix P is found from the value function of the associated unconstrained problem, with $Q = I$, $R = I$. We take $p = 0$, and $w_t \sim \mathcal{N}(0, \rho I)$, where ρ was chosen so that the input constraints were active around 50% of the time. For each of our ten problems, we evaluated the control-Lyapunov policy at 10 000 points, by simulating the closed-loop system. We also report the time taken to evaluate an affine policy, as well as the time taken to carry out a Kalman filter update (assuming a number of sensors equal to the number the actuators). It is difficult to accurately measure execution times below 1 μs, so these entries should be interpreted as less than or approximately equal to 1 μs.

Table II shows the maximum time to evaluate the control-Lyapunov policy (t_{clf}) compared with linear state feedback (t_{lin}), and Kalman filter state update (t_{kf}), over a 10 000 time step simulation. The entries in the k column marked with asterisks indicate problems with box constraints, rather than general linear constraints. We should direct the reader to the time unit, which is *microseconds*, not a common time unit when describing the execution time for solving a QP. For a small problem with $n = 15$ states, $m = 5$ box-constrained inputs, the control-Lyapunov policy can be solved in around 35 μs, allowing a sample rate up to 20 kHz or so. For a large example, with $n = 100$ states and $m = 10$ box-constrained inputs, the control-Lyapunov policy can be evaluated in around 67 μs, allowing sample rates up to 10 kHz. Running a control-Lyapunov control policy at 1 kHz seems entirely practical, even on a slower processor, with only a fraction of its time allocated for control law computation.

TABLE II
Maximum Time to Evaluate Control-Lyapunov Policy, Linear State Feedback, and Kalman Filter State Update. (Asterisks (*) Indicate Problems With Box Constraints.)

n	m	k	t_{clf} (μs)	t_{lin} (μs)	t_{kf} (μs)
15	5	*10	35	1	1
15	5	20	70	1	2
50	15	*30	85	3	9
70	5	*10	35	3	20
70	5	20	72	3	21
100	10	*20	67	4	40
100	5	*10	45	3	39
100	20	30	230	4	41
500	25	*50	231	80	2400
1000	30	*60	298	130	8300

We can see that for the middle-sized problems, with $n = 70$ or $n = 100$ states, evaluating the control-Lyapunov policy takes roughly the same time as updating a Kalman filter estimate. For the larger problems, with $n = 500$ or $n = 1000$ states, evaluating the control-Lyapunov policy is far faster than updating a Kalman filter estimate. (We do not claim that problems this large arise in many practical situations; we consider these problems just to illustrate the scaling with state dimension.) All of these computation time are consistent, within a factor or 2 or 3, with our complexity analyses above.

V. Extensions

The same ideas can be applied in more complex situations, a few of which we list here. First, we can use more complex convex control-Lyapunov functions, such as piecewise-linear or piecewise-quadratic; in this case the optimization problem that has to be solved at each step is different (e.g., a quadratically constrained quadratic program (QCQP) in the second case), but is still small and convex, and can be solved quickly for the same reasons and using similar methods. QCQPs also arise in robust formulations of the control-Lyapunov policy, where we add a quadratic constraint that guarantees that the control-Lypunov function decreases at each iteration. These formulations are particularly popular since the resulting policies guarantee stability even if the optimization problem is not solved to high accuracy.

In the same way we can use more complex convex stage cost functions, including piecewise-linear or piecewise-quadratic stage cost functions. We can add equality constraints, as was used in [58] for dynamic portfolio optimization problems. It is also possible to consider the case in which the data A, B, Q, R, and others change in each time step. This occurs for finite-horizon problems, or in controllers for nonlinear systems, where these data are obtained from linearization of the dynamics at each point. In these cases the control-Lyapunov policy can still be evaluated quickly, but the computational

complexity can grow more than linearly with n, so for large n we will no longer have the dramatic speeds seen above.

The same methods can also be applied for problems with constraints on both the inputs as well as the states. In this case, the optimization problem (8) can be infeasible. A common method to deal with this is addition of so-called soft state constraints, in which we add a linear penalty for constraint violations [9], [10].

For these more complex formulations, custom code for an interior-point method can be developed, to obtain high execution speeds. An alternative is to use recently developed automatic code generation techniques, developed by Mattingley and Boyd [16]. Such systems work from a high level description of the problem family to be solved, and automatically generate source code (say, C) and auxiliary files for a custom solver for the particular problem family.

The ability to rapidly evaluate a quadratic control-Lyapunov policy, as described in this note, can be directly leveraged to handle an extension in which the state update has the form $x_{t+1} = f(x_t, u_t)$, where f is not affine in u_t. For such problems we can linearize the state update equation at the current point x_t and some guess of u_t, using a Jacobian or particle method derived linearization. After quickly solving the QP associated with the linearized state update, we can relinearize the state update equation using the new value of u_t; this process can be repeated for a few steps if needed.

REFERENCES

[1] D. Bertsekas, *Dynamic Programming and Optimal Control*. Belmont, MA: Athena Scientific, 2005, vol. 1.

[2] D. Bertsekas, *Dynamic Programming and Optimal Control*. Belmont, MA: Athena Scientific, 2007, vol. 2.

[3] D. Bertsekas and S. Shreve, *Stochastic Optimal Control: The Discrete-time Case*. Belmont, MA: Athena Scientific, 1996.

[4] P. Whittle, *Optimization Over Time*. New York: Wiley, 1982.

[5] E. Sontag, "A Lyapunov-like characterization of asymptotic controllability," *SIAM J. Control Opt.*, vol. 21, no. 3, pp. 462–471, 1983.

[6] R. Freeman and J. Primbs, "Control Lyapunov functions, new ideas from an old source," in *Proc. Conf. Decision Control*, 1996, vol. 4, pp. 3926–3931.

[7] M. Corless and G. Leitmann, "Controller design for uncertain systems via Lyapunov functions," in *Proc. Amer. Control Conf.*, 1988, vol. 3, pp. 2019–2025.

[8] M. Sznaier, R. Suarez, and J. Cloutier, "Suboptimal control of constrained nonlinear systems via receding horizon constrained control Lyapunov functions," *Int. J. Robust Nonlinear Control*, vol. 13, no. 3–4, pp. 247–259, 2003.

[9] J. Maciejowski, *Predictive Control With Constraints*. Englewood Cliffs, NJ: Prentice-Hall, 2002.

[10] A. Bemporad, "Model predictive control design: New trends and tools," in *Proc. IEEE Conf. Decision Control*, 2006, pp. 6678–6683.

[11] L. Wirsching, H. Ferreau, H. Bock, and M. Diehl, "An online active set strategy for fast adjoint based nonlinear model predictive control," in *Proc. 7th Symp. Nonlinear Control Syst.*, 2007, pp. 164–169.

[12] H. Ferreau, H. Bock, and M. Diehl, "An online active set strategy to overcome the limitations of explicit MPC," *Int. J. Robust Nonlinear Control*, vol. 18, no. 8, pp. 816–830, 2008.

[13] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.

[14] Y. Nesterov and A. Nemirovsky, "Interior-point polynomial methods in convex programming," SIAM, Warrendale, PA, 1994.

[15] Y. Wang and S. Boyd, "Fast model predictive control using online optimization," in *Proc. 17th IFAC World Congr.*, 2009, pp. 6974–6997.

[16] J. Mattingley and S. Boyd, "Automatic code generation for real-time convex optimization," in *Convex Optimization in Signal Processing and Communications*, Y. Eldar and D. Palomar, Eds. Cambridge, U.K.: Cambridge Univ. Press, 2009.

[17] P. Goulart, E. Kerrigan, and D. Ralph, "Efficient robust optimization for robust control with constraints," *Math. Program.*, vol. Series A, pp. 1–33, 2007.

[18] S. Wright, "Applying new optimization algorithms to model predictive control," *Chem. Process Control-V*, vol. 93, no. 316, pp. 147–155, 1997.

[19] S. Boyd and B. Wegbreit, "Fast computation of optimal contact forces," *IEEE Trans. Robot.*, vol. 23, no. 6, pp. 1117–1132, Dec. 2007.

[20] L. Biegler, "Efficient solution of dynamic optimization and NMPC problems," in *Nonlinear Model Predictive Control*, F. Allgöwer and A. Zheng, Eds. Berlin, Germany: Birkhauser, 2000, pp. 119–243.

[21] M. Åkerblad and A. Hansson, "Efficient solution of second order cone program for model predictive control," *Int. J. Control*, vol. 77, no. 1, pp. 55–77, Jan. 2004.

[22] A. Wills and W. Heath, "Barrier function based model predictive control," *Automatica*, vol. 40, no. 8, pp. 1415–1422, Aug. 2004.

[23] A. Wills and W. Heath, "A recentered barrier for constrained receding horizon control," in *Proc. Amer. Control Conf.*, May 2002, pp. 4177–4182.

[24] C. Rao, S. Wright, and J. Rawlings, "Application of interior point methods to model predictive control," *J. Opt. Theory Appl.*, vol. 99, no. 3, pp. 723–757, Nov. 2004.

[25] A. Wills, D. Bates, A. Fleming, B. Ninness, and S. Moheimani, "Model predictive control applied to constraint handling in active noise and vibration control," *IEEE Trans. Control Syst. Technol.*, vol. 16, no. 1, pp. 3–12, Jan. 2008.

[26] A. Hansson and S. Boyd, "Robust optimal control of linear discrete-time systems using primal-dual interior-point methods," in *Proc. Amer. Control Conf.*, 1998, vol. 1, pp. 183–187.

[27] E. Yildirim and S. Wright, "Warm-start strategies in interior-point methods for linear programming," *SIAM J. Opt.*, vol. 12, no. 3, pp. 782–810, 2002.

[28] A. Bemporad, M. Morari, V. Dua, and E. Pistikopoulos, "The explicit linear quadratic regulator for constrained systems," *Automatica*, vol. 38, no. 1, pp. 3–20, Jan. 2002.

[29] G. Pannocchia, J. Rawlings, and S. Wright, "Fast, large-scale model predictive control by partial enumeration," *Automatica*, vol. 43, no. 5, pp. 852–860, May 2006.

[30] P. Tøndel, T. Johansen, and A. Bemporad, "An algorithm for multiparametric quadratic programming and explicit MPC solutions," in *Proc. IEEE Conf. Decision Control*, 2001, pp. 1199–1204.

[31] M. Kvasnica, P. Grieder, M. Baotić, and M. Morari, "Multi-parametric toolbox (MPT)," Dec. 2003. [Online]. Available: http://control.ee.ethz.ch/~mpt/

[32] P. Tøndel and T. Johansen, "Complexity reduction in explicit linear model predictive control," presented at the 15th IFAC World Congr. Autom. Control, Barcelona, Spain, Jul. 2002.

[33] M. Zeilinger, C. Jones, and M. Morari, "Real-time suboptimal model predictive control using a combination of explicit MPC and online computation," in *Proc. IEEE Conf. Decision Control*, Dec. 2008, pp. 4718–4723.

[34] C. Christophersen, M. Zeilinger, C. Jones, and M. Morari, "Controller complexity reduction for piecewise affine systems through safe region elimination," in *Proc. IEEE Conf. Decision Control*, Dec. 2007, pp. 4773–4778.

[35] A. Bemporad and C. Filippi, "Suboptimal explicit receding horizon control via approximate multiparametric quadratic programming," *J. Opt. Theory Appl.*, vol. 117, no. 1, pp. 9–38, Nov. 2004.

[36] C. Jones, P. Grieder, and S. Rakovic, "A logarithmic-time solution to the point location problem," *Automatica*, vol. 42, no. 12, pp. 2215–2218, Dec. 2006.

[37] F. Borrelli, A. Bemporad, M. Fodor, and D. Hrovat, "An MPC/hybrid system approach to traction control," *IEEE Trans. Control Syst. Technol.*, vol. 14, no. 3, pp. 541–552, May 2006.

[38] S. Hovland, J. Gravdahl, and K. Willcox, "Explicit model predictive control for large-scale systems via model reduction," *J. Guid., Control Dyn.*, vol. 31, no. 4, pp. 918–926, Jul. 2008.

[39] T. Krogstad, J. Gravdahl, and P. Tøndel, "Explicit model predictive control of a satellite with magnetic torquers," in *Proc. 17th IFAC World Congr.*, 2008, pp. 15250–15255.

[40] A. Grancharova, T. Johansen, and J. Kocijan, "Explicit model predictive control of gas-liquid separation plant via orthogonal search tree partitioning," *Comput. Chem. Eng.*, vol. 28, no. 12, pp. 2481–2491, Nov. 2004.

[41] D. De Farias and B. Van Roy, "The linear programming approach to approximate dynamic programming," *Oper. Res.*, vol. 51, no. 6, pp. 850–865, 2003.

[42] B. Lincoln and A. Rantzer, "Relaxing dynamic programming," *IEEE Trans. Autom. Control*, vol. 51, no. 8, pp. 1249–2006, Aug. 2006.
[43] S. Meyn, *Control Techniques for Complex Networks*. Cambridge, U.K.: Cambridge Univ. Press, 2007.
[44] Y. Wang and S. Boyd, "Performance bounds for linear stochastic control," *Syst. Control Lett.*, vol. 58, no. 3, pp. 178–182, 2009.
[45] Y. Wang and S. Boyd, "Performance bounds and suboptimal policies for linear stochastic control via LMIs," 2009. [Online]. Available: www.stanford.edu/~boyd/papers/gen_ctrl_bnds.html
[46] D. Bertsekas and J. Tsitsiklis, *Neuro-Dynamic Programming*. Belmont, MA: Athena Scientific, 1996.
[47] R. Sutton, "Learning to predict by the methods of temporal differences," *Mach. Learn.*, vol. 3, pp. 9–44, 1988.
[48] J. Tsitsiklis and B. Van Roy, "An analysis of temporal difference learning with function approximation," *IEEE Trans. Autom. Control*, vol. 42, no. 5, pp. 674–690, May 1997.
[49] A. Nedić and D. Bertsekas, "Least-squares policy evaluation algorithms with linear function approximation," *J. Discrete Event Syst.*, vol. 13, pp. 79–110, 2003.
[50] D. Bertsekas, V. Borkar, and A. Nedić, "Improved temporal difference methods with linear function approximation," in *Learning and Approximate Dynamic Programming*. Piscataway, NJ: IEEE Press, 2004.
[51] J. Tsitsiklis, "Asynchronous stochastic approximation and Q-learning," *Mach. Learn.*, vol. 16, pp. 185–202, 1994.
[52] A. Barto, S. Bradtke, and S. Singh, "Real-time learning and control using asynchronous dynamic programming," *Artif. Intell.*, vol. 72, pp. 81–138, 1995.
[53] S. Bradtke and A. Barto, "Linear least-squares algorithms for temporal difference learning," *Mach. Learn.*, vol. 22, pp. 33–57, 1996.
[54] H. Ferreau, H. Bock, and M. Diehl, "An online active set strategy for fast parametric quadratic programming in MPC applications," in *Proc. IFAC Workshop Nonlinear Model Predictive Control for Fast Syst.*, 2006, pp. 21–30.
[55] S. Wright, *Primal-Dual Interior-Point Methods*. Warrendale, PA: SIAM, 1997.
[56] E. Anderson, Z. Bai, J. Dongarra, A. Greenbaum, A. McKenney, J. Du Croz, S. Hammarling, J. Demmel, C. Bischof, and D. Sorensen, "LAPACK: A portable linear algebra library for high-performance computers," in *Proc. Supercomput.*, 1990, pp. 2–11.
[57] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen, *LAPACK Users' Guide*. Warrendale, PA: SIAM, 1999.
[58] J. Skaf and S. Boyd, "Multi-period portfolio optimization with constraints and transaction costs," 2008. [Online]. Available: www.stanford.edu/~boyd/papers/dyn_port_opt.html