# A new algorithm for efficient MPC and a comparison with competing schemes

Lars Imsland, Nadav Bar and Bjarne A. Foss

***Abstract*— An approach for constrained predictive control of linear systems (or uncertain systems described by polytopic uncertainty models) is presented. The approach consists of a non-convex offline problem, and an efficient online problem. The offline problem can be considered as a generalization of earlier results, and a solver for the non-convex optimization problem is developed. Two examples, one being a laboratory experiment, compare the approach to existing approaches, revealing both advantages and disadvantages.**

## I. Introduction

Model predictive control (MPC) has gained significant popularity in industry as a tool to optimize system performance while handling constraints explicitly. However, limitations on computational efficiency have restricted the application range. This has lead to a substantial effort to obtain predictive constraint-handling control strategies that have more attractive online computational properties than quadratic programming typically used in traditional linear MPC. In most cases, this is obtained by performing some calculations offline.

Examples of such schemes are explicit MPC as presented in [1], and efficient robust predictive control (ERPC) presented in [2]. Explicit MPC computes offline via multi-parametric programming an explicit solution to the finite horizon MPC problem. In ERPC, the offline part uses the degrees of freedom on the control horizon to find large invariant ellipsoids for an augmented system, while the online part efficiently minimizes control deviation from unconstrained optimal LQ-control, subject to augmented state membership of the precomputed ellipsoid.

Herein, we present a generalization of the offline problem of ERPC, thus we will denote the new approach generalized ERPC, GERPC. Through this generalization, it is possible to obtain significantly larger invariant ellipsoids. Using the information obtained by solving the offline problem, we specify a new efficient online problem. Furthermore, through two examples, one being a laboratory experiment, we compare the merits of GERPC with ERPC and explicit MPC.

## II. Model class and control objective

Consider the following discrete-time linear state-space model subject to input and state constraints

$$x_{k+1} = Ax_k + Bu_k \tag{1a}$$

$$\text{subject to } -\overline{u} < u_k < \overline{u}, \quad x_k \in X, \tag{1b}$$

where the inequalities should be interpreted componentwise. The state and input dimensions are $x \in \mathbb{R}^{n_x}$ and $u \in \mathbb{R}^{n_u}$, and the origin is an equilibrium, $0 < \overline{u}$, and $X \ni 0$ is a closed, convex set. The pair $(A, B)$ is assumed stabilizable. For brevity we have restricted us to linear time-invariant systems, but the generalization to polytopic uncertainty models is straightforward [2], [3].

The control objective will be to minimize (while satisfying constraints) the infinite horizon linear quadratic (LQ) cost function,

$$J_{LQ} = \sum_{i=0}^{\infty} x_{k+i+1}^{\mathsf{T}} Q x_{k+i+1} + u_{k+i}^{\mathsf{T}} R u_{k+i}, \tag{2}$$

where $Q$ and $R$ are positive semi-definite matrices and $x_{k+i+1}$ and $u_{k+i}$ denote predicted values of states and control inputs (subscript $k$ denotes current time). The system is assumed to be prestabilized by a feedback controller $K$, optimal with respect to (2) in the unconstrained case. We will express the degrees of freedom as the perturbation, $c_k$, away from this optimal control, and let the future (predicted) control input be

$$u_i = \begin{cases} -Kx_i + c_i, & i = k, \dots, k + n_c - 1 \\ -Kx_i, & i \geq k + n_c. \end{cases} \tag{3}$$

As a consequence the optimization is carried out in terms of the new free variables $c_i$. The $c_i$'s should be minimized, but must be large enough to prevent constraint violation. Beyond the control horizon $n_c$, we can set $c_i = 0$ assuming the optimal LQ control is feasible onwards. The system equation for (1) with (3) is

$$x_{k+1} = \Phi x_k + Bc_k, \tag{4}$$

where $\Phi = A - BK$.

## III. Efficient constrained sub-optimal control

The proposed controller consists of an offline part and an online part. First we present the offline part, which is a method for constructing enlarged invariant sets for the system (4) by augmenting the state space. In the online part (Section III-B), we use the information obtained through the offline problem to specify an efficient optimizing, constraint-handling controller.

### *A. Augmenting the state space for enlarging the region of attraction*

The unconstrained LQ controller $K$ will typically have a rather small region where it does not hit the constraints (and hence stability is guaranteed). The objective of this section is to enlarge this region of attraction by augmenting the state space, inspired by a similar approach in [2] (see below). Denoting this augmented variable $z = (x, f)$, the augmenting variable $f \in \mathbb{R}^p$ is imposed with the following dynamics:

$$f_{k+1} = Fx_k + Gf_k. \tag{5}$$

This work was supported by the Gas Technology Center NTNU-SINTEF.

The authors are with Department of Engineering Cybernetics, Norwegian University of Science and Technology, 7491 Trondheim, Norway. {Lars.Imsland,barc,Bjarne.Foss}@itk.ntnu.no

Let $c_k$ be computed by $c_k = Df_k$. The overall dynamics is then described by

$$z_{k+1} = \Psi z_k, \quad \Psi = \begin{pmatrix} \Phi & BD \\ F & G \end{pmatrix}. \tag{6}$$

Note that in the special case of $D = I$, $F = 0$ and $G = 0$ we recover the original LQ controller. For $F = 0$, $G = M$ where $M$ has the "time recession" structure

$$M = \begin{pmatrix} 0 & I & 0 & \cdots & 0 \\ 0 & 0 & I & \cdots & 0 \\ & & \ddots & & \\ 0 & \cdots & 0 & 0 & I \\ 0 & \cdots & 0 & 0 & 0 \end{pmatrix}, \tag{7}$$

and $D = [I\ 0\ \cdots\ 0]$ we recover the offline problem of ERPC [2]. In this case the variable $f$ is interpreted as the future $c_i$'s, i.e. $f_k^\mathsf{T} = (c_k^\mathsf{T}, c_{k+1}^\mathsf{T}, \ldots, c_{k+n_c-1}^\mathsf{T})$ thus the dimension is $p = n_u \cdot n_c$. This predictive control interpretation is not so straightforward in the general case of (5), but a connection can still be made. We obtain (in general) an infinite horizon (but note; the control degrees of freedom $f_k$ is still finite) as opposed to the finite horizon given by the time recession matrix (7): For a given $x_k$ and $f_k$, the future $c_{k+i}$'s are given as $c_{k+i} = Df_{k+i}$, $i = 0, 1, \ldots$ where $f_{k+i+1} = Gx_{k+i} + Ff_{k+i}$.

We want to find $D$, $F$ and $G$ that gives the largest possible region of attraction for the original system. We do this by looking for invariant ellipsoids $\mathcal{E}_z := \{z \mid z^\mathsf{T} Q_z^{-1} z \le 1\}$, where the projection of the ellipsoid $\mathcal{E}_z$ onto the state space (see [2]), $\mathcal{E}_{xx} := \{x | x^\mathsf{T}(TQ_zT^\mathsf{T})^{-1}x \le 1\}$ where the matrix $T$ is defined by the coordinate transformation $x = Tz$, should be as large as possible.

The size of $\mathcal{E}_{xx}$, the region of attraction in the $x$-space, is proportional to $\ln\det(TQ_zT^\mathsf{T})$. Thus, the following optimization problem maximizes the region of attraction:

$$\min_{Q_z, F, G} \ln\det(TQ_zT^\mathsf{T})^{-1} \tag{8a}$$

$$\text{subject to} \quad \begin{pmatrix} Q_z & Q_z\Psi^\mathsf{T} \\ \Psi Q_z & Q_z \end{pmatrix} \ge 0 \tag{8b}$$

$$\bar{u}_j^2 - [-K_j^\mathsf{T}\ e_j^\mathsf{T}]Q_z[-K_j^\mathsf{T}\ e_j^\mathsf{T}]^\mathsf{T} \ge 0,\ j = 1, \ldots, n_u \tag{8c}$$

Condition (8b) guarantees invariance of $\mathcal{E}_z$, while (8c) guarantees that the control inside $\mathcal{E}_z$ is feasible. Here $e_j$ is the $j$th column of the identity matrix, and $\bar{u}_j$ and $K_j$ correspond to the $j$th input. With constant $D$, $F$ and $G$ (as in ERPC), this is a convex problem, for which efficient algorithms exist [3]. However, treating $D$, $F$ and $G$ as variables makes the problem non-convex, since they are multiplied with $Q_z$. This increase in complexity can be rewarded by significantly larger ellipsoids, as pointed out in [4] and to be seen in the examples.

State constraints can be handled by adding LMI conditions to (8a). This is omitted for reasons of space limitations, but it should be noted that while the LMI offline problem of ERPC can handle LMIs involving $Q_z$ only, the solver discussed in Section III-C can handle LMIs involving $P_z = Q_z^{-1}$ as well.

### B. Online problem: minimizing cost

Although the augmented part of the autonomous system specifies a valid controller, it is not optimizing (apart from when $f = 0$, when it is the same as the unconstrained LQ controller). In this section we look at how the cost can be minimized (although sub-optimally) while retaining stability. We briefly first present one that is akin to the method used in [2], where the minimizing $f$ is found from the feasible ones (inside the ellipsoid $\mathcal{E}_z$). Then we present a new method, that searches for $f$'s in a larger set; the ones that ensure feasibility at the *next* sample.

In this section, we will assume that the structural constraint $F = 0$ is imposed in (8). This means that $f_{k+1} = Gf_k$, and thus not dependent on the evolution of $x_k$. This assumption is mainly done for simplicity of presentation.

*1) Feasibility now:* As future control flexibility (the $f$) is part of the current augmented state, the ellipsoidal stability constraint can be applied at current time rather than at the end of the control horizon, as is common in other (e.g. QP-based) MPC approaches. This reduces online optimization to minimizing a performance index based on the future degrees of freedom in the input, $J_f$, subject to membership of the precomputed ellipsoid

$$\min_f J_f \quad \text{subject to} \quad z^\mathsf{T} Q_z^{-1} z \le 1, \tag{9}$$

with $z = (x_k, f)$.

Here, $J_f$ penalizes the future control perturbations,

$$J_f = \sum_{i=0}^{\infty} c_{k+i}^\mathsf{T} W c_{k+i} \tag{10}$$

where $W > 0$ is given by

$$W = B^\mathsf{T}PB + R, \quad P = Q + K^\mathsf{T}RK + \Phi^\mathsf{T}P\Phi.$$

In [5] it is shown that $J_f$ and the LQ cost (2) differ by a bias term, thus minimizing the two indices is equivalent. Further, note that by standard arguments, the infinite sum (10) has a limit that is readily computed,

$$\sum_{i=0}^{\infty} c_{k+i}^\mathsf{T} W c_{k+i} = f^\mathsf{T}\Gamma f \tag{11}$$

where $\Gamma$ is the positive definite solution of the discrete Lyapunov equation $G^\mathsf{T}\Gamma G - \Gamma = -D^\mathsf{T}WD$.

This turns the online problem into minimizing a quadratic function subject to one ellipsoidal constraint, which can be solved extremely efficiently using a Newton-Raphson method to determine one Lagrange multiplier [5].

*2) Feasibility at next sample:* The ellipsoidal constraint in (9) leads to sub-optimality. According to [5], this sub-optimality can be reduced by allowing a line search outside the ellipsoid subject to feasibility at the next time instant (i.e., by "scaling" $f$). As this "scaling" can be performed explicitly, it only adds marginally to computational complexity. In view of the improved performance due to the "scaling" of $f$, it is tempting to look for other algorithms that search for $f$ outside $\mathcal{E}_z$ in more general ways, subject to feasibility at the next time instant. The straightforward convex optimization problem that solves this, is

$$\min_f J_f \quad \text{subject to} \quad \begin{cases} z^\mathsf{T}\Psi^\mathsf{T}Q_z^{-1}\Psi z & \le 1 \\ Df & \le \bar{u} + Kx_k \\ -Df & \le \bar{u} - Kx_k \end{cases} \tag{12}$$

where two constraints are added to ensure that the computed control satisfies input constraints. Denoting the optimal solution of (9) by $f^\star_{(9)}$ and the optimal solution of (12) by $f^\star_{(12)}$, it is clear that $J_f(f^\star_{(12)}) \leq J_f(f^\star_{(9)})$. In practice, the eigenvalues of $\Psi$ will be strictly less than 1, in which case the inequality is strict. For the rest of this section, this is assumed.

Apart from the two linear constraints, this optimization problem has the same structure as (9). It is solved relatively fast and reliable by for instance the `fmincon`-algorithm of Matlab. However, using a general optimization routine does not enjoy the same efficiency as using a Newton-Raphson-method to solve (9).

We therefore propose a more efficient method of solving (12), but where the solution might not always be the optimal. Let $f_{feas}$ be a feasible solution of (12). This can be found by solving (9), or it can be obtained from the solution at the previous time-step via (5). Such a $f_{feas}$ exists at the first iteration, if we start inside $\mathcal{E}_{xx}$.

*Algorithm 1:*

i) Solve the optimization problem obtained by removing the linear constraints from (12) using e.g. a Newton-Raphson method. Call the solution $f^\star$. Obtain $f_{feas}$ (e.g. by solving (9)).
ii) Check if $f^\star$ satisfies the linear constraints of (12). If they do, then $f^\star$ is the optimal solution to (12), and we are finished. If they do not, then go to iii).
iii) Pick the $f$ on the line between $f^\star$ and $f_{feas}$ that is closest to $f^\star$ and satisfies the constraints.

The properties of the solution are as follows (the proof is omitted for space reasons):

*Theorem 1:* The $f$ produced by Algorithm 1 guarantees feasibility at next time step, satisfies the control constraints, and when it is not equal to $f^\star$, at least one input is at a constraint. Furthermore $f^\mathsf{T}\Gamma f < f_{feas}^\mathsf{T}\Gamma f_{feas}$, thus $f$ is always a better solution than $f_{feas}$.

The "line-search" for $f$ in Algorithm 1 can be implemented explicitly, and hence very efficiently. If $n_u = 1$ and $f^\star$ exists but violates the input constraints, then it merely amounts to choosing the right input constraint, and $f_{feas}$ is not needed at all.

*3) Algorithm and stability:* The overall approach can be summarized as follows:

*Algorithm 2:*

**Offline:** Find an optimal $K$ for the unconstrained case. Use the optimization problem (8) to find $n_c$, $D$, $G$ and $Q_z$ that give a suitable invariant set.

**Online:** At each time step, find the minimizing $f$ of (9), (12) or Algorithm 1, and implement $u_k = -Kx_k + c_k$, where $c_k = Df$.

The (omitted) proof of stability of Algorithm 2 is similar to that of Algorithm 4.1 of [2], with a slight difference due to the minimization of the "infinite horizon cost" (11).

*Theorem 2 (Closed loop stability):* If for system (1) there exist $K$, $Q_z$, $p$, $D$ and $G$ such that $x_k \in \mathcal{E}_{xx}$, the closed-loop application of Algorithm 2 is feasible and asymptotically stabilizing.

### C. Solving the non-convex offline problem

As mentioned above, the problem (8) to be solved offline is non-convex, and hence hard to solve. However, for a given problem, it only has to be solved once. Also, if specific requirements on the desired region of attraction exists, it might not be necessary to solve the problem to optimality, merely that the region is large enough might suffice.

The non-convexity arises since the Lyapunov matrix is multiplied by the $F$ and $G$ variables. This makes the problem minimizing a convex function subject to bilinear (and linear) matrix inequalities. Similar optimization problems frequently arise in the literature, for instance for the "static output feedback" problem.

We choose to adapt "sequential semidefinite programming" as proposed in [6] for the problem at hand. To do this, we must use a standard trick to transform the bilinear matrix inequality (8b) into a linear matrix inequality but with an additional non-convex inequality constraint. Then, the problem is solved in a manner similar to the "augmented Lagrangian" approach from the SQP world, by augmenting the Lagrangian by weighting the quadratic deviation from the inequality constraint, approximate it by a second-order Taylor expansion and minimize it sequentially by semidefinite programming. See [7] for further details.

Since this is a second-order algorithm, the convergence properties are better than algorithms merely relying on gradient information. Thus, it is a significant advantage that exact expressions for the gradient and Hessian of the augmented Lagrangian are readily computable.

## IV. Examples

In this section we will present two examples, where the performance of the suggested control algorithm will be compared to two control schemes that also are tailored towards efficient predictive control under constraints. We will first briefly review these.

**Efficient robust predictive control**. As mentioned above, the offline problem of ERPC as proposed in [2] is a special (convex) case of the GERPC offline problem. The online problem has the same structure as (9).

**Explicit MPC**. Explicit MPC (referred to as eMPC) refers to the explicit solution of the finite horizon constrained LQR problem as piecewise affine functions (controllers) defined on polytopic partitions of the state space, as derived in [1]. The offline problem of finding these controllers and the corresponding polytopes can be solved as a multi-parametric QP problem, see [8] for a recent algorithm.

The online problem consists of finding which polytope the present state is within. If done with a brute force-approach, this might involve a significant number of arithmetic operations when the number of polytopes is large. However, by using smart data structures to store the problem data and exploiting the structure, it is possible to significantly reduce the online computational effort [9]. Nevertheless, storing all the polytopes and corresponding controllers might require a significant amount of memory.

### A. Double integrator example

Consider the discretized double integrator model [8],

$$A = \begin{pmatrix} 1 & T_s \\ 0 & 1 \end{pmatrix}, \quad B = \begin{pmatrix} T_s^2 \\ T_s \end{pmatrix}$$

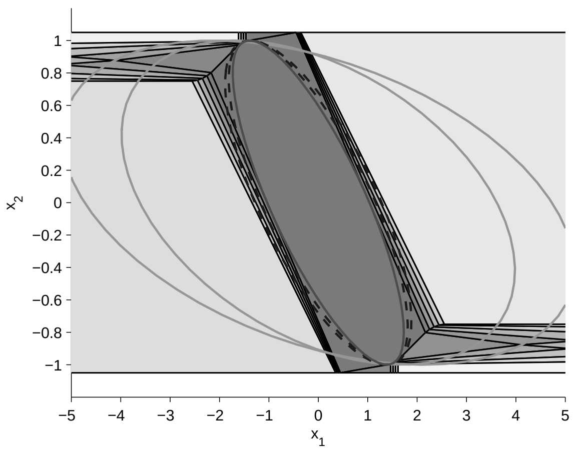

Fig. 1. The shaded area (the polytopes) are the region where the eMPC controller with horizon 5 is defined. The dotted ellipsoids are the region of attraction for ERPC (horizon 2 and 5) and the outer ellipsoids the region of attraction for GERPC (horizon 2 and 5). The innermost ellipsoid is the largest invariant ellipsoid where the LQ controller is unconstrained.

where $T_s = .05s$ is the sampling time. There are constraints on input, $|u| \leq 1$, and on velocity, $|x_2| \leq 1$. The weighting matrices are chosen as $Q = \operatorname{diag}\{1, 0\}$ and $R = 1$.

We will in the following compare the region of attraction, online (and offline) computational demand, online memory use and sub-optimality of GERPC, ERPC and eMPC.

*1) Offline:* **Region of attraction**. The obtained "regions of attractions" for the three approaches are shown in Figure 1. We immediately observe that the computed region is significantly larger for eMPC (which stretches indefinitely in the $x_1$-direction) than for GERPC, which again is larger than for ERPC. For ERPC, increasing the horizon only marginally increased the region of attraction beyond the region of attraction for the LQ-controller. This is not typical in our experience, but must be attributed to the effect of the state constraint. The fixed structure of $G$ in the case of ERPC seemingly makes enlargement most pronounced in one particular direction, and in this case this comes in conflict with the state constraint. This example clearly shows the importance of flexibility in $G$ since the GERPC ellipsoids are both rotated and "fattened" as compared to ERPC.

**Offline computational complexity**. The difference in offline computing time between ERPC and GERPC is large, even though the number of optimization variables are not very different different. The offline problem of ERPC is convex [3], while solving a non-convex optimization problem is NP hard. The fact that one usually does not require optimality helps considerably, and allows the use of local search algorithms with a guaranteed upper complexity limit. These local algorithms are however more complex than the ERPC problem. As for the approach suggested in Section III-C, the convex problem solved at each iteration is about three times as large in number of optimization variables than the corresponding ERPC problem, and also has more constraints.

As mentioned above, the eMPC solution is found using multi-parametric quadratic programming (mpQP). For this type of problem, the computational complexity depends on the number of states $n_x$, the control freedom $n_u \cdot n_c$, and especially the constraints, since the constraints must hold on the entire control horizon. The computational complexity grows exponentially with problem size.

In conclusion, it is probably safe to say that the ERPC offline problem has least complexity for a given system and horizon, while mpQP and the local algorithm in Section III-C are harder to compare. In our experience the mpQP problem is faster than our implementation of the local BMI algorithm for the same control horizon.

*2) Online:* **Memory usage**. The demand on online memory is about the same for ERPC and GERPC (there is a slight difference due to the size of the $\Gamma$ matrix). Therefore, we compare GERPC with eMPC.

eMPC has to store all the piecewise affine controllers online, but in addition it has to store the inequalities that define the polytopes. If $r$ is the number of polytopes, and $s$ is the number of inequalities, the total number of reals to be stored are $r \cdot (n_u \cdot n_x + n_u) + s \cdot (n_x + 1)$. This is an upper bound that does not take into account that the controllers in several regions are the same, and other symmetry effects. GERPC must as a minimum store two symmetric matrices ($\Gamma$ and $Q_z^{-1}$), of dimension $n_x + p$, giving a total number of $2 \cdot \frac{1}{2}(p + n_x)(p + n_x + 1)$ reals to be stored.

The memory requirements for horizons of $n_c = 5$ can be summarized as follows[1]:

| GERPC (reals) | "crude" explicit MPC (reals, controllers plus polytopes) | "smart" explicit MPC (reals plus integers) |
|---|---|---|
| 56 | 182 + 756 | 240 + 255 |

As we can see, the difference is considerable in favor of GERPC, and will be even larger for longer horizons, as the size of the matrices grows quadratically while the number of polytopes grows exponentially. In the rightmost column, we see that by using smart data structures and exploiting problem structure [9], "smart" eMPC considerably reduces memory demand as compared to the straightforward implementation.

**Online computational complexity**. A count of "worst case" floating point operations is shown below for the given example (initial condition at $(-4, 0)$):

| GERPC | "crude" eMPC | "smart" eMPC |
|---|---|---|
| 12896 | 1008 | 44 |

For the Newton-Raphson method used by GERPC (for solving the optimization problem (9)), the worst case number of iterations were 13 in our implementation, with 992 floating point operations (counted with the `flops`-command of Matlab 5) per iteration. Enhancing performance using scaling [5] or Algorithm 1 only adds about 1% to this number.

We see that the "smart" eMPC-controller is extremely quick in this case. However, all these numbers are small

[1]For GERPC, it is advantageous to store the results of some matrix calculations done offline for efficiency reasons. In our implementation, this amounts to two symmetric matrices of dimension $p$ and one of dimension $p + n_x$, which together with $K$ give 82 extra variables to be stored. Furthermore, the memory requirement for GERPC does not take into account "intermediate variables" that can be introduced in the iterations.

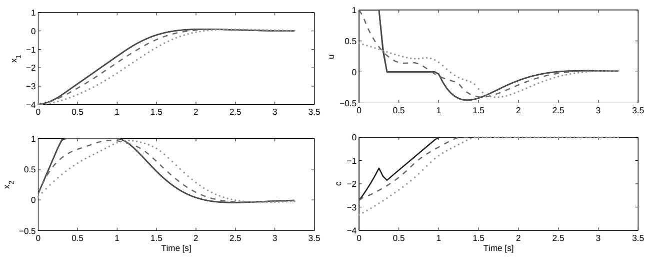

Fig. 2. The states, and control input $u$ and perturbation $c$. Solid line is ihMPC and eMPC (they are barely distinguishable), dotted line is GERPC (with scaling), and dashed line is GERPC with Algorithm 1.

considering present-day computing power. A QP MPC controller (implemented using `quadprog` in Matlab) with horizon 5 uses about 106000 floating point operations.

**Sub-optimality**. Simulations of the system from initial condition $(-4, 0)$ with infinite horizon MPC (ihMPC), eMPC and GERPC with "scaling" and the new algorithm in Section III-B are shown in Figures 2. We would expect that both eMPC and GERPC for initial conditions far from the origin will show sub-optimality. However, for this example it is hard to find initial conditions where eMPC shows significant sub-optimality. GERPC on the other hand, is far from optimal, probably due to the fact that it stays away from the constraints. As we see from the table below, the scaling-technique of [5] reduces sub-optimality slightly, while Algorithm 1 reduces it significantly.

| Initial condition | Cost | | | | |
|---|---|---|---|---|---|
| | GERPC (9) | GERPC w/scaling | GERPC Alg. 1 | eMPC | ihMPC |
| (-4,0) | 832.72 | 831.72 | 677.89 | 609.44 | 609.39 |
| (-2,.6) | 378.24 | 374.57 | 234.63 | 204.46 | 204.46 |

The fact that "scaling" is not as effective as reported in [5] for ERPC, is probably mostly due to the state constraint (state constraints were not considered in the examples in [5]), and that we are able to start further away from the origin due to larger ellipsoidal regions of attraction.

## *B. Lab helicopter example*

This example compares GERPC and ERPC. They were used to control a laboratory helicopter (Quanser 3-DOF Helicopter). Using two "local" PD controllers to decouple pitch and elevation, the following 6 states ($\lambda$ is "travel", $p$ is pitch angle of helicopter and $e$ is elevation angle), 2 inputs (setpoints to pitch and elevation controllers) model is used:

$$\begin{bmatrix}\dot\lambda\\ \ddot\lambda\\ \dot p\\ \ddot p\\ \dot e\\ \ddot e\end{bmatrix} = \begin{bmatrix}0 & 1 & 0 & 0 & 0 & 0\\ 0 & 0 & -0.45 & 0 & 0 & 0\\ 0 & 0 & 0 & 1 & 0 & 0\\ 0 & 0 & -19.8 & -7.28 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 1\\ 0 & 0 & 0 & 0 & -3.09 & -4.98\end{bmatrix}\begin{bmatrix}\lambda\\ \dot\lambda\\ p\\ \dot p\\ e\\ \dot e\end{bmatrix} + \begin{bmatrix}0 & 0\\ 0 & 0\\ 0 & 0\\ 19.8 & 0\\ 0 & 0\\ 0 & 4.98\end{bmatrix}\begin{bmatrix}p_c\\ e_c\end{bmatrix}$$

The weighting matrices are chosen as $Q = \operatorname{diag}\{1,1,1,10,1,10\}$ and $R = \operatorname{diag}\{95,95\}$.

A comparison of the volumes obtained by GERPC and ERPC is shown in the table below. The volume factor $V_f = \sqrt{\det(TQ_zT^\intercal)^{-1}}$ is a measure of the size of the ellipsoids. A three-dimensional projection of the ellipsoids is shown in Figure 3 (most of the other projections showed similar proportions). We clearly see that even for an extremely short control horizon, GERPC is able to obtain significantly larger regions of attractions than ERPC.

| Algorithm | $n_c$ | Offline calc. time | dim $Q_z$ | $V_f[\times 10^3]$ |
|---|---|---|---|---|
| GERPC | 2 | 600s | 10×10 | 55.3 |
| ERPC | 2 | 0.9s | 10×10 | 0.6 |
| ERPC | 5 | 8.9s | 16×16 | 1.0 |
| ERPC | 8 | 48.7s | 22×22 | 1.4 |
| ERPC | 10 | 134s | 26×26 | 1.8 |
| ERPC | 13 | 560s | 32×32 | 2.7 |

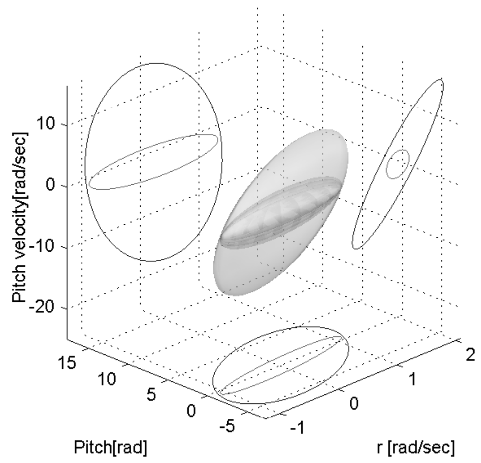

Fig. 3. Projections of 6-dimensional ellipsoids. GERPC with horizon $n_c = 2$ (outer ellipsoid) and ERPC for $n_c = 13$ (the ellipsoids for ERPC for $n_c = 2, 5, 8, 10$ lies inside the $n_c = 13$ ellipsoid).

We have not performed a detailed comparison with eMPC. However, in [8] an example with a similar model of the same dimension is used. There the horizon 4 offline case was solved in 1830 s (not directly comparable to our numbers, since another computer was used) and the number of regions found were 12223.

In order to get a reasonable region of attraction we had to increase $n_c$ for ERPC. For the given system environment, however, the online optimization problem of ERPC with $n_c \geq 10$ was too computationally demanding, and lead to computer exceptions and hence controller failure. It should be noted that the implementation used in this example can be further optimized with respect to efficiency. Figure 4

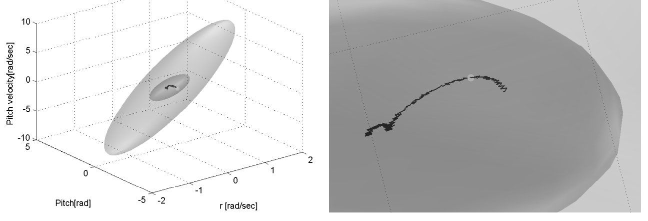

Fig. 4. A projection of two ellipsoids (the GERPC region of attraction and the largest invariant ellipsoid for unconstrained LQ) and one state trajectory, and (right) a (rotated) zoom. The point where $c_k$ becomes zero is marked with a dot.

shows three of the states from a laboratory trial starting

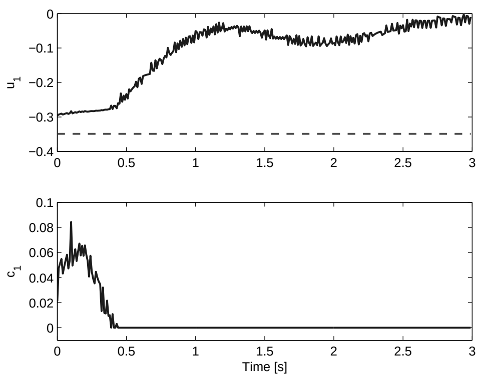

Fig. 5. At top, the first input (the pitch reference, $u_1$) with the constraint (-20° in radians), with the corresponding $c_1$ (cf. (3)).

outside the invariant ellipsoid given by the LQ-controller, and entering it after about 0.4s. This is confirmed by Figure 5, where we see that the "perturbation" to the first control also is zero after 0.45s.

## V. Discussion

**Offline problem**. The examples clearly indicates that the generalized offline problem achieved considerably larger regions of attractions than ERPC. The generalization of the offline problem gives more freedom in "shaping" the ellipsoid, which can be especially helpful in the presence of state constraints as in the first example. Furthermore, we believe that the generalized offline problem in the multiple input case (as in the second example) has more freedom for exploiting possible couplings between the inputs than the ERPC offline problem has, since $G$ is a general matrix as opposed to the "diagonal structure" of the $M$ matrix of ERPC. However, the regions of attraction obtained by GERPC and ERPC will always be a subset of the set where the explicit MPC controller is defined, but this set is not necessarily a region of attraction as discussed below.

The non-convexity of the offline problem makes it harder and less tractable than the convex problem of ERPC. We proposed a local solver of the non-convex BMI problem, which uses the exact Hessian of the optimization criterion. As is common for second-order algorithms, the algorithm must be started reasonably close to a feasible solution - this is achieved by using the ERPC solution as starting point.

**Online problem**. The first example reveals that while GERPC/ERPC requires limited memory online, explicit MPC performs better as far as computational complexity and sub-optimality is concerned. However, it is not hard to imagine examples where the memory requirements of explicit MPC is prohibitive. In this respect, suboptimal implementations of explicit MPC could be an interesting alternative to compare with GERPC.

In the first example, we were able to reduce sub-optimality in GERPC considerably by the new online algorithm suggested in Algorithm 1. Of course, this algorithm could also be used to reduce sub-optimality for ERPC.

Since longer "control horizons" $n_c$ $(= p/n_u)$ might have to be used to obtain a suitable region of attraction for ERPC, the online problem might demand a higher computational load than for GERPC, since the computational complexity grows linearly with $n_c$ [5].

**Stability and robustness**. It is important to keep in mind that GERPC/ERPC gives stability guarantees. On the other hand, it is well known that finite horizon MPC (as implemented by explicit MPC) might enter "blind alleys" if the horizon is not long enough, and thus the fact that the controller is defined for an initial condition does not imply that this initial condition is within the region of attraction. However, the stability of the piecewise affine controller can be checked (conservatively) using e.g. piecewise quadratic Lyapunov functions and LMIs, or one can enforce stability by design, by either finding the horizon length that guarantees stability, or adding stability constraints (at the cost of a more complex controller).

While GERPC/ERPC straightforwardly can handle uncertainty by the use of polytopic models, this is not so straightforward for explicit MPC. Some results in that direction have appeared recently [10], however restricted to 1- and $\infty$-norm type objective functions.

## VI. Conclusion

By considering feasibility through a non-convex optimization problem offline, we are able to pose efficient online optimization algorithms for constrained optimization of linear systems and systems described by polytopic uncertainty models. Compared to ERPC larger regions of attractions were achieved. Further, a less sub-optimal online problem was proposed. The new approach offers few advantages to explicit MPC in terms of online computational efficiency or sub-optimality, but it uses less online memory. In addition, the new approach gives (in the same way as ERPC) stability guarantees, and model uncertainty can be handled.

## References

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

[2] B. Kouvaritakis, J. A. Rossiter, and J. Schuurmans, "Efficient robust predictive control," *IEEE Trans. Aut. Control*, vol. 45, no. 8, pp. 1545–1549, 2000.

[3] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan, *Linear Matrix Inequalities in System and Control Theory*, ser. SIAM Studies in Applied Mathematics. SIAM, 1994, no. 15.

[4] S. Drageset, L. Imsland, and B. A. Foss, "Efficient model predictive control with prediction dynamics," in *Proc. 7th European Control Conference ECC'03*, Cambridge, England, 2003.

[5] B. Kouvaritakis, M. Cannon, and J. A. Rossiter, "Who needs QP for MPC anyway?" *Automatica*, vol. 38, no. 5, pp. 879–884, 2002.

[6] B. Fares, P. Apkarian, and D. Noll, "An augmented Lagrangian method for a class of LMI-constrained problems in robust control theory," *Internat. J. Control*, vol. 74, no. 4, pp. 348–360, 2001.

[7] N. Bar, "Efficient model predictive control using sequential semi-definite programming," Master's thesis, Department of Engineering Cybernetics, NTNU, 2003, http://www.itk.ntnu.no/ansatte/Imsland_Lars_Struen/Barthesis.pdf.

[8] P. Tøndel, T. A. Johansen, and A. Bemporad, "An algorithm for multi-parametric quadratic programming and explicit MPC solutions," *Automatica*, vol. 39, pp. 489–497, 2003.

[9] ——, "Evaluation of piecewise affine control via binary search tree," *Automatica*, vol. 39, pp. 945–950, 2003.

[10] A. Bemporad, F. Borrelli, and M. Morari, "Min-max control of constrained uncertain discrete-time linear systems," *IEEE Trans. Aut. Control*, vol. 48, no. 9, pp. 1600–1606, 2003.