Verifying stability of approximate explicit MPC
Morten Hovd, Francesco Scibilia, Jan Maciejowski and Sorin Olaru

Abstract—Several authors have proposed algorithms for approximate explicit MPC [1],[2],[3]. These algorithms have in common that they develop a stability criterion for approximate explicit MPC that require the approximate cost function to be within a certain distance from the optimal cost function. In this paper, stability is instead ascertained by considering only the cost function of the approximate MPC. If a region of the state space is found where the cost function is not decreasing, this indicates that an improved approximation (to the optimal control) is required in that region. If the approximate cost function is decreasing everywhere, no further refinement of the approximate MPC is necessary, since stability is guaranteed.

I. INTRODUCTION

Explicit MPC removes the requirement for the online solution of the MPC optimization problem [4], and can therefore make MPC useful for applications where fast sampling is required [2]. Explicit MPC uses multiparametric programming to partition the state space, and finds controllers that are affine in the state for each partition.

However, as pointed out in [4], the optimal partition of the state space may become very complex for large problems (problems with many states and/or a long prediction horizon). Approaches to simplifying the required partitioning of the state space have therefore been proposed [1], [2], [3]. These approaches allow for some degree of sub-optimality, and guarantee stability by ensuring that the approximate cost function is within a pre-derived bound of the exact cost function. Although it is not necessary to know the exact value of the optimal cost function in order to guarantee fulfillment of the stability criterion, the stability bound is only a sufficient condition, and may be conservative.

An alternative approach is provided by Rossiter and Grieder [5], who interpolate between the solution at the boundary of the unconstrained region and the solution on the boundary of the feasible region. They show that this leads to a very small performance loss in many cases. However, they have to introduce a second interpolation between the result of the aforementioned interpolation, and the 'tail' of the solution from the previous timestep in order to guarantee stability. This way of guaranteeing stability is inherently non-robust, since disturbances may render the ‘tail’ of the solution from the previous timestep infeasible.

Nevertheless, the small performance losses found in [5] provides inspiration for the present work. In this work, the state space is partitioned into simplices using Delaunay tessellation. Within each simplex, the controller is defined by linear interpolation between the optimal solution at the vertices - which leads to an affine controller over the simplex. Stability is ensured by refining the tessellation until it can be shown that the approximate cost function is a Lyapunov function. This is done by solving one or more QP problems for each of the resulting simplices.

II. MPC FORMULATION

We consider a linear dynamical discrete-time system

\[ x_{k+1} = Ax_k + Bu_k \]  

where the state vector is of length \( n_x \), and a 'standard' quadratic QP formulation of MPC

\[ \min_{u} \sum_{N-1} \mathbf{J}(u, x_k) = \|x_{k+N}\|_P^2 + \sum_{i=0}^{N-1} \left( \|x_{k+i}\|_Q^2 + \|u_{k+i}\|_R^2 \right) \]

subject to constraints

\[ x_k = \text{given} \]
\[ u_{k+i} \in \mathbb{U} \]
\[ x_{k+i} \in \mathbb{X} \]

The constraints on the inputs and states are assumed to be non-empty polytopes with the origin in the interior. The constraints on the inputs are enforced from the present timestep \((i \geq 0)\), whereas the constraints on the states are enforced from the next timestep \((i \geq 1)\). The model equation (1) is naturally assumed to hold, and the matrices \( Q \geq 0 \) and \( R > 0 \) are tuning matrices, and it is assumed that \((A, B)\) are controllable and \((\sqrt{Q}, A)\) observable. The matrix \( P > 0 \) is then the corresponding solution to the discrete algebraic Riccati equation. This solution to the discrete algebraic Riccati equation also provides a stabilizing linear state feedback controller \( K \). If no constraints are active, the MPC is equivalent to this linear state feedback controller.
Defining

\[ \mathbf{x} = \begin{bmatrix} x_k \\ x_{k+1} \\ \vdots \\ x_{k+N} \end{bmatrix} \]

\[ \dot{\mathbf{Q}} = \text{diag}\{Q, \cdots, Q, P\} \]

\[ \dot{\mathbf{R}} = \text{diag}\{R, \cdots, R\} \]

\[ \hat{\mathbf{A}} = \begin{bmatrix} I \\ A \\ A^2 \\ \vdots \\ A^N \end{bmatrix} \]

\[ \hat{\mathbf{B}} = \begin{bmatrix} 0 & 0 & \cdots & 0 & 0 \\ B & 0 & \cdots & 0 \\ AB & B & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ A^{N-1}B & A^{N-2}B & \cdots & AB & B \end{bmatrix} \]

(5)

the optimization criterion may be reformulated as

\[ J(\mathbf{u}, x_k) = \mathbf{x}^T \dot{\mathbf{Q}} \mathbf{x} + \mathbf{u}^T \dot{\mathbf{R}} \mathbf{u} \]

(6)

with \( x_k \) given. Using the identity

\[ \mathbf{x} = \hat{\mathbf{A}} x_k + \hat{\mathbf{B}} \mathbf{u} \]

(7)

we get

\[ J(\mathbf{u}, x_k) = \|\mathbf{u}\|_H^2 + 2x_k^T F \mathbf{u} + \|x_k\|_Y^2 \]

subject to constraints

\[ x_k \quad \text{given} \]

\[ G \mathbf{u} \leq W - E x_k \]

(9)

where \( H = \hat{\mathbf{B}}^T \dot{\mathbf{Q}} \hat{\mathbf{B}} + \hat{\mathbf{R}} \), \( F = \hat{\mathbf{A}}^T \dot{\mathbf{Q}} \hat{\mathbf{B}} \) and \( Y = \hat{\mathbf{A}}^T \dot{\mathbf{Q}} \hat{\mathbf{A}} \). Note that \( H > 0 \) follows from \( R > 0 \).

The region \( \mathcal{X}_F \) of the state space for which there exists an input sequence \( \mathbf{u} \) fulfilling (9), i.e., the feasible region, can be found by projecting the constraints (9) onto the states \( x \). Similarly, using the theory in [6], it is possible to calculate a set (the so called 'maximal output admissible set (MOAS)', \( \mathcal{M}_0 \)), for which the linear state feedback controller \( \hat{K} \) fulfills all constraints and which is positive invariant for the system (1) when the controller \( K \) is applied. We therefore include in (9) not only the constraints resulting directly from (3) and (4), but add the constraints \( x_{k+N} \in \mathcal{M}_0 \) (see [7] for further details of the MPC formulation). It is noted above that the MPC controller and the state feedback controller \( K \) are equivalent when no constraints are active. Adding the constraints \( x_{k+N} \in \mathcal{M}_0 \) therefore ensures that the system with the MPC controller is closed loop stable provided the initial state is feasible.

III. APPROXIMATE MPC

Bemporad and Filippi [8] propose an algorithm for approximate multiparametric convex programming which partitions the feasible region \( \mathcal{X}_F \) (or whatever subset thereof they are exploring) into simplices. Let \( v_i \) be the \( n_x + 1 \) vertices of a simplex, and \( \mathbf{u}(v_i) \) be the optimal solution to the MPC problem ((8)-(9)) at vertex \( i \).

Given a state \( x_0 \) inside a simplex, define the matrix

\[ M = \begin{bmatrix} v_1 & \cdots & v_{n_x+1} \\ 1 & \cdots & 1 \end{bmatrix} \]

The state \( x_0 \) can be expressed as a linear interpolation between the vertices of the simplex as

\[ x_0 = v_1 \lambda_1 + \cdots + v_{n_x+1} \lambda_{n_x+1} \]

where the interpolation parameters \( \lambda_i \) are found from

\[ \begin{bmatrix} \lambda_1 \\ \vdots \\ \lambda_{n_x+1} \end{bmatrix} = M^{-1} \begin{bmatrix} x_0 \\ 1 \end{bmatrix} \]

(10)

Note that the matrix \( M \) is by construction non-singular and hence invertible, provided the points \( v_1, \cdots, v_{n_x+1} \) define a full-dimensional simplex. Equation (10) and the definition of \( M \) ensure that the \( \lambda_i \)'s sum to one, and they will also be non-negative provided \( x_0 \) is inside the simplex. From (10) observe that the interpolation parameters can be split into a state-dependent part and a state-independent part. That is

\[ \lambda = \lambda_x x_0 + \lambda_0 \]

(11)

where \( \lambda_x \) consists of the \( n_x \) leading columns of \( M^{-1} \) and \( \lambda_0 \) is the last column. Define the matrix

\[ U = \begin{bmatrix} \mathbf{u}(v_1) & \cdots & \mathbf{u}(v_{n_x+1}) \end{bmatrix} \]

and let \( \mathbf{u}_x \) be the approximately-optimal input sequence obtained by linear interpolation between the solutions at the vertices of the simplex. This gives

\[ \mathbf{u}_x = U \lambda = K_x x_0 + k_u \]

(12)

where \( K_x \) and \( k_u \) are constant within a given simplex. Interpolating between the solutions at the vertices of the simplex we therefore obtain an affine state feedback controller that is valid for the whole simplex. In [8] it is shown that this input is a feasible (but obviously not in general optimal) solution to the MPC problem at the state \( x_0 \).

Our interest is in approximate MPC rather than approximate multiparametric programming solution in general. It therefore makes sense to use the optimal controller \( \hat{K} \) inside \( \mathcal{M}_0 \). Instead of calculating the Delaunay tessellation (or triangulation in the 2-dimensional case) of \( \mathcal{X}_F \), we instead calculate the tessellation for \( \mathcal{X}_F \setminus \mathcal{M}_0 \). However, since \( \mathcal{M}_0 \) is in the interior of \( \mathcal{X}_F \), the set difference is clearly non-convex. This non-convex set is therefore partitioned into convex parts using the procedure in [7], and each convex part is subsequently tesselated.

Having thus obtained an initial tessellation and calculated the approximate controller for each resulting simplex, the
approximate cost function can be determined. One can identify which simplex a given state $x_k$ lies within, and identify the corresponding controller parameters $K_a$ and $k_a$ for that simplex. Within the identified simplex the approximate cost function is then given by

$$J_a(u_a, x_k) = \|u_a\|^2 + 2x_k^T F u_a + \|x_k\|^2 + \sum_{i=k} x_k^T [K_a^T H K_a + F K_a + K_a^T F^T + Y] x_k + 2[k_a^T H K_a + k_a^T F^T] x_k + k_a^T H k_a \quad (13)$$

We observe that $J_a$ is continuous (also on the edges of each simplex) and piecewise quadratic in $x_k$. It is underbounded by the optimal cost function, and hence also positive definite. It is therefore a good candidate for a Lyapunov function, if we can show that everywhere in the feasible region (or whatever domain of application the controller is designed for) this Lyapunov function candidate is uniformly decreasing along the resulting closed loop trajectories.

We note also that by using arbitrary many points in the tessellation (not only on the perimeters of the MOAS and the feasible region, but also in between), we can make both the approximate MPC and the approximate cost function arbitrarily close to the optimal control and optimal cost function.

The approach to designing an approximate MPC with guaranteed stability is therefore

1) Start with an initial tessellation, and corresponding initial approximate MPC.
2) Explore the state space to verify that the approximate cost function is decreasing everywhere.
3) If a region is found where the approximate cost function increases, add an extra point where it increases the most, and re-triangulate with this point included.
4) Regions affected by introducing the new point must be explored again to check that the approximate cost function is decreasing, otherwise additional points must be added.
5) Stop when it is verified that the approximate cost function is decreasing everywhere.

IV. EXPLORING THE STATE SPACE

A. Mapping the state from one timestep to the next

Consider Figure 1. Let the heavy solid simplex represent the simplex within which the state resides at the current time step. Applying the approximate control for this simplex, the state for the next timestep must be within the simplex indicated by the dashed lines. Note that this mapping from the current state to the state at the next timestep preserves the simplicial shape due to linearity, although the next timestep simplex may be both rotated and stretched. In ‘extreme cases’ the simplex may collapse to a lower-dimensional simplex (or, in two dimensions, a triangle collapses to a line or to a point).

We also note that the ‘next-timestep-simplex’ can be subdivided into a number of regions, depending on which of the ‘original’ simplices the next-timestep-state is in. For each of these regions, it must be verified that the approximate cost function is actually decreasing. This results in the following procedure:

1) Select one simplex, and identify the corresponding ‘next-timestep-simplex’. Note that as a consequence of the feasible region being invariant under the optimal MPC, and the vertices of the next-timestep-simplex being determined by application of the optimal inputs at the vertices of the present-time simplex, the next-timestep-simplex must lie within the feasible region $X_F$.
2) Select one region within the ‘next-timestep-simplex’.
3) Map this region back to the current timestep. This is illustrated by the shaded regions in Figure 1. The borders of this ‘mapped back’ region will define constraints for the QP problem that needs to be solved to verify decrease of the approximate cost function. The formulation of this QP problem will be further described below.
4) Solve the QP problem. If the cost function is shown to decrease, continue to the next subregion. If the cost function does not decrease, mark the ‘next-timestep-subregion’ as requiring improved control before continuing to the next subregion, and save also the point in the ‘next-timestep-simplex’ where the approximate
cost function increases the most.
5) Repeat for other sub-regions and for other simplices until the whole state space is explored.
6) Solve the (original) MPC problem for the points where this has been shown necessary. Re-tessellate, and return to step 1. Note that simplices that have not been affected by adding additional points will not need to be checked again.
7) End when the whole state space is explored.

B. Optimization problem to verify decrease in the approximate cost function

For each sub-region of the 'next-timestep-simplex', we need to verify reduction of the approximate cost function. From step 3 above we know what part of the current timestep simplex this corresponds to. Both the approximate cost function at the current timestep, \( J_{a,k} \), and the approximate cost function at the next timestep \( J_{a,k+1} \) are given by (13) - provided we use their respective values of the controller parameters \( K_a \) and \( k_a \). Next we use the model (1) and controller (12) to express \( J_{a,k+1} \) in terms of \( x_k \). The result is a little complicated.

\[
J_{a,k+1}(x_k) = \begin{array}{c}
\frac{1}{2} x_k^T \left[ (K_{a,k+1}(A + BK_{a,k}^*))^T H (K_{a,k+1}(A + BK_{a,k}^*)) + (A + BK_{a,k}^*)^T F K_{a,k+1}(A + BK_{a,k}^*) \\
+ (A + BK_{a,k}^*)^T F (K_{a,k+1}(A + BK_{a,k}^*)^T F (A + BK_{a,k}^*) \right] x_k \\
+ 2 \left[ (K_{a,k+1}Bk_{a,k}^* + k_{a,k+1})^T H K_{a,k+1}(A + BK_{a,k}^*) \right] x_k \\
+ \left[ (K_{a,k+1}Bk_{a,k}^* + k_{a,k+1})^T H (K_{a,k+1}Bk_{a,k}^* + k_{a,k+1}) \right] x_k \end{array}
\]

Here we have introduced an extra subscript on \( K_a \) and \( k_a \) to make it clear what timestep they refer to (as these may change from one timestep to the next). Also, the superscript * is introduced, to denote the top block of \( K_a \) and \( k_a \), since only the top blocks determine the input that is (actually) applied at the current timestep. 

Thus, the QP subproblem consists of minimizing \( J_{a,k}(x_k) - J_{a,k+1}(x_k) \) subject to the state \( x_k \) being inside the subregion of the current simplex identified by mapping back from the 'next-timestep-simplex' in step 3 of the procedure above.

C. Numerical issues

In this work we have used Matlab with the Multi-Parametric Toolbox (MPT) [9] for calculations.

1) Lower-dimensional next-timestep-simplices:
MPT seems only to handle full-dimensional polytopes. Simple modifications are therefore necessary when the 'next-timestep-simplex' collapses to a lower dimension, since it is necessary to verify that the cost function decreases also for these cases. This phenomenon was observed for Example 2 below.

2) Indefinite QP problem: The QP problem minimizing \( J_{a,k} - J_{a,k+1} \) is unfortunately not necessarily convex, since the resulting Hessian matrix can be indefinite (or negative definite). Indefinite QP-problems are in general NP-hard, and solvers for these problems are few. However, this paper focuses on problems of modest state dimension, and therefore the resulting QP problems may nevertheless be tractable. In this work, the BMI2NB solver in the YALMIP package [10] is used for non-convex QPs, and it has worked satisfactorily for the problems tested. The BMI2NB document recommends not expecting much from the solver for problems of dimension larger than 10. Other solvers will clearly have different limitations in the maximum problem size that can be handled.

Note that although indefinite QP problems are non-convex, convex lower bounds do exist, and branch-and-bound based solvers (like BMI2NB) can provide guaranteed lower bounds for the solution. Thus, for our application, guarantees for decrease of the approximate cost function can be obtained.

3) Early termination of the indefinite QP solver: For the problem studied here, we are not really interested in finding the minimum of the indefinite QP problem. Rather, we are interested in verifying that the minimum is positive, since that is sufficient to verify decrease of the approximate cost function. There is thus a potential for optimizing IQP solvers for this application. If, for example, a branch-and-bound algorithm is used (which would seem necessary to obtain a guaranteed lower bound on the non-convex optimization criterion) a branch may be cut if the lower bound is found to be positive - it is not necessary to ascertain that the optimum is in another branch before cutting. This may significantly speed up computations. This could be combined with improved lower bounds based on semidefinite programming [11], [12].

V. EXAMPLES

In this section, the approach to verifying stability of approximate MPC is applied to two examples. The first is a rather trivial example included for the purpose of illustration, whereas the second is an example that has been used in previous publications on explicit MPC and is a little more involved.

A. Example 1
Consider the one-state system is given by

\[ x_{k+1} = 1.1 x_k + u_k \]  

and the input is constrained \(-2 \leq u_k \leq 2\). We consider a prediction horizon of 5, and the weights \( Q = 10, R = 0.1 \) are chosen. Solving the Riccati equation, we get \( P \approx 10.1, \)
and the corresponding LQ-optimal controller, which saturates at $x = 1.836$ (only positive state values will be considered, since the problem is symmetrical). We find that with the chosen prediction horizon, the MPC problem is feasible for $x < 8.722$. Between the unconstrained region and the edge of the feasible region the exact solution has 5 regions. We will instead use an approximate solution with only one region in addition to the unconstrained region, i.e., one region for $1.836 < x < 8.722$. When attempting to verify stability using the approximate cost function as a Lyapunov function, we find that for $x_k > 3.487$ $x_{k+1}$ lies within the same region, whereas for $x_k \leq 3.487$ $x_{k+1}$ lies in the unconstrained region. In the latter case, the conditions for stability are trivially fulfilled, whereas in the first case we need to solve a one-dimensional QP. We find that for this case the Hessian is $-8.227$, i.e., the problem is non-convex. However, for this small problem finding the solution is easy, and we find that the minimum value of $J_{a,k}(x_k) - J_{a,k+1}(x_k)$ (for $x_k > 3.487$) is 178.19, occurring at $x_k = 3.487$. Thus, we have proved that the closed loop system is stable with the approximate MPC, using the approximate cost function as a Lyapunov function.

**B. Example 2**

We consider the same system as the one studied in [7].

$$A = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \quad B = \begin{bmatrix} 1 \\ 0.3 \end{bmatrix}$$

with constraints

$$\begin{bmatrix} -2 \\ -5 \end{bmatrix} \leq \begin{bmatrix} u_k \\ x_k \end{bmatrix} \leq \begin{bmatrix} 2 \\ 5 \end{bmatrix}$$

The weight matrices used are $Q = I$ and $R = 1$, whereas the prediction horizon $n = 15$ is used. In [7] it is found that the exact solution requires 101 regions.

Proceeding with the approximate MPC, we obtain an initial partition of the state space with 33 regions (including $M_0$). Although the approximate MPC based on this initial partition does appear to be stable in simulations [7], it is hard to verify this by simulation for all possible initial conditions.

Therefore, proof of closed loop stability is investigated using the approximate cost function as a Lyapunov function. It is found that new regions need to be introduced to be able to prove that the approximate cost function decreases everywhere in the feasible region $X_F$. With a total of 53 regions, we can prove that the system is closed loop stable. The resulting partition of the state space is shown in Fig. 3.

![Fig. 2. The exact and the approximate cost function for Example 1.](image1)

![Fig. 3. Partition of the state space required for proving stability for Example 2.](image2)
in Fig. 3, the controllers for the present timestep are identical. Geyer et al. [13] presents a routine for finding the minimal number of regions to describe a piecewise affine system. Applying this routine to our example results in 17 regions, as shown in Fig. 4. For the exact explicit MPC, we were able to reduce the number of regions only to 53 in this way.

VI. CONCLUSION

This work introduces a novel approach to designing approximate explicit MPC controllers. Most of the existing approaches guarantee closed loop stability with the approximate controller by ensuring that the approximate cost function is within some bound around the optimal cost function. Example 1 illustrates that such bounds on the approximation of the cost function are not necessary for stability. In this work, the approximation is gradually refined until it can be shown that the approximate cost function is a Lyapunov function, thus removing the need for estimating the loss relative to the optimal cost function.

However, there is no guarantee that the approximate solution proposed will be less complex than the exact solution. The exact solution allows for critical regions of more complex shape than the simplex, which may be an advantage, especially in higher dimensions.

REFERENCES