I. SCIENTIFIC SESSIONS

I.8. Session

DISTRIBUTED ARCHITECTURES, HIGH PERFORMANCE COMPUTING AND APPLICATIONS

I - 232

International Conference
AUTOMATICS AND INFORMATICS'10
3-7 October 2010, Sofia, Bulgaria

A PARALLEL COMPUTING ALGORITHM FOR DESIGN OF EXPLICIT NONLINEAR MODEL PREDICTIVE CONTROLLERS[1]

A. Grancharova[1], J. Kocijan[2, 3]

[1] Institute of System Engineering and Robotics, Bulgarian Academy of Sciences, Acad G. Bonchev str., Bl.2, P.O.Box 79, Sofia 1113, Bulgaria, e-mail: alexandra.grancharova@abv.bg

[2] Department of Systems and Control, Jozef Stefan Institute, Jamova 39, 1000 Ljubljana, Slovenia e-mail: jus.kocijan@ijs.si

[3] University of Nova Gorica, Centre for Systems and Information Technologies, Vipavska 13, 5000 Nova Gorica, Slovenia

Abstract: Recently, several multi-parametric Nonlinear Programming (mp-NLP) approaches to explicit solution of constrained Nonlinear Model Predictive Control (NMPC) problems have been suggested. The benefits of an explicit solution, in addition to the efficient on-line computations, include also verifiability of the implementation. However, the off-line computational complexity of the explicit NMPC approaches tends to increase rapidly with the number of states. In this paper, a parallel computing algorithm for design of explicit NMPC controllers is proposed, which is based on an approximate mp-NLP approach. The off-line computational efficiency of the mp-NLP approach is improved by allowing the computations to be performed in parallel on multi-core computer architectures.

Key words: Parallel computations, Explicit Model Predictive Control, Multi-parametric Nonlinear Programming.

INTRODUCTION

Nonlinear Model Predictive Control (NMPC) involves the solution at each sampling instant of a finite horizon optimal control problem subject to nonlinear system dynamics and state and input constraints [1]. Several approaches to explicit solution of NMPC problems have been suggested in the literature [2], [3], [4], [5]. The benefits of an explicit solution, in addition to the efficient on-line computations, include also verifiability of the implementation. In [2], [3], [4], approaches for off-line computation of explicit sub-optimal piecewise linear (PWL) predictive controllers for general nonlinear systems with state and input constraints have been developed, based on the multi-parametric Nonlinear Programming (mp-NLP) ideas [6]. It has been shown that for convex mp-NLP problems, it is straightforward to impose tolerances on the level of approximation such that theoretical properties like asymptotic stability of the sub-optimal feedback controller can be ensured [3]. In [4], practical computational methods to handle non-convex mp-NLP problems have been suggested. Algorithms for solving mp-NLP problems, including the non-convex case, are described also in [5].

However, the off-line computational complexity of the explicit MPC tends to increase very rapidly with the number of states and this would restrict the application of the approximate mp-NLP approaches only for systems with a few states. This has led to the development of several methods (e.g. [7], [8]) for complexity reduction of the explicit solution of MPC problems.

Another way for decreasing the off-line computational burden of the explicit MPC approaches is to develop parallel computing algorithms, which will exploit the multi-core computer architectures available nowadays. In this paper, a parallel computing algorithm for design of explicit NMPC controllers is proposed. It uses the Parallel Computing Toolbox for MATLAB [9] and represents a parallel implementation of the approximate mp-NLP approach in [4].

FORMULATION OF NONLINEAR MODEL PREDICTIVE CONTROL PROBLEM

Consider the discrete-time nonlinear system:

$$x(t+1) = f(x(t), u(t)) \qquad (1)$$

$$y(t) = Cx(t) \qquad (2)$$

where $x(t) \in \mathbb{R}^n$, $u(t) \in \mathbb{R}^m$, and $y(t) \in \mathbb{R}^p$ are the state, input and output variable, and $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ is a nonlinear function. It is supposed that a full measurement of the state $x(t)$ is available at the current time t. For the current $x(t)$, the regulation NMPC solves the optimization problem:

Problem P1:

$$V^*(x(t)) = \min_U J(U, x(t)) \qquad (3)$$

subject to $x_{t|t} = x(t)$ and:

$$y_{\min} \le y_{t+k|t} \le y_{\max},\ k = 1, \dots, N \qquad (4)$$

$$u_{\min} \le u_{t+k} \le u_{\max},\ k = 0,1, \dots, N-1 \qquad (5)$$

$$\left\| x_{t+N|t} \right\| \le \delta \qquad (6)$$

[1] This work was financed by the National Science Fund of the Ministry of Education, Youth and Science of Republic of Bulgaria, contract №DO02-94/14.12.2008 and the Slovenian Research Agency, contract №BI-BG/09-10-005 ("Application of Gaussian processes to the modeling and control of complex stochastic systems")

$$x_{t+k+1|t} = f(x_{t+k|t}, u_{t+k}),\ k \ge 0 \quad (7)$$

$$y_{t+k|t} = Cx_{t+k|t},\ k \ge 0 \quad (8)$$

with $U = [u_t, u_{t+1}, ..., u_{t+N-1}]$ and the cost function given by:

$$J(U, x(t)) = \sum_{k=0}^{N-1} \left[\left\| x_{t+k|t} \right\|_Q^2 + \left\| u_{t+k} \right\|_R^2 \right] + \left\| x_{t+N|t} \right\|_P^2 \quad (9)$$

Here, N is a finite horizon and $P, Q, R \succ 0$. From a stability point of view it is desirable to choose δ in (6) as small as possible [10]. However, the feasibility of problem P1 will rely on either δ or N being sufficiently large. A part of the NMPC design will be to address this tradeoff. The problem P1 can be formulated in a compact form as follows:

Problem P2:

$$V^*(x(t)) = \min_U J(U, x(t)) \text{ subject to } G(U, x(t)) \le 0 \quad (10)$$

Problem P2 defines an mp-NLP, since it is NLP in U parameterized by $x(t)$. An optimal solution to this problem is denoted $U^* = [u_t^*, u_{t+1}^*, ..., u_{t+N-1}^*]$ and the control input is chosen according to the receding horizon policy $u(t) = u_t^*$. Define the set of N-step feasible initial states as follows:

$$X_f = \{x \in \mathbb{R}^n \mid G(U, x) \le 0 \text{ for some } U \in \mathbb{R}^{Nm}\} \quad (11)$$

In parametric programming problems one seeks the solution $U^*(x)$ as an explicit function of the parameters x in a set $X \subseteq X_f \subseteq \mathbb{R}^n$ [6].

PARALLEL COMPUTING ALGORITHM FOR DESIGN OF EXPLICIT NMPC

Here, a parallel computing algorithm for design of explicit NMPC controllers is proposed, which represents a parallel implementation of the approximate mp-NLP approach in [4]. Let $X \subset \mathbb{R}^n$ be a hyper-rectangle where we seek to approximate the optimal solution $U^*(x)$ to the problem P2. It is required that the state space partition is orthogonal and can be represented as a $k-d$ tree. The main idea is to construct a feasible PWL approximation $\hat{U}(x)$ to $U^*(x)$ on X, where the constituent affine functions are defined on hyper-rectangles covering X [4]. The computation of an affine state feedback associated to a given region X_i includes the following steps. First, a close-to-global solution of problem P2 is computed at a set of points in X_i. Then, based on the solutions at these points, a local linear approximation $\hat{U}_i(x) = K_i x + g_i$ to the close-to-global solution $U^*(x)$, valid in the whole hyper-rectangle X_i, is determined by applying the following procedure [4]:

Procedure 1:

Consider any hyper-rectangle $X_i \subseteq X$ with a set of points $V_i = \{v_0, v_1, v_2, ..., v_{N_1}\} \subset X_i$. Compute K_i and g_i by solving the following NLP:

$$\min_{K_i, g_i} \sum_{j=0}^{N_1} (J(K_i v_j + g_i, v_j) - V^*(v_j) + \mu \left\| K_i v_j + g_i - U^*(v_j) \right\|_2^2) \quad (12)$$

$$\textit{subject to} \quad G\left(K_i v_j + g_i, v_j\right) \le 0,\ \forall v_j \in V_i \quad (13)$$

In (12), $J(K_i v_j + g_i, v_j)$ is the sub-optimal cost, $V^*(v_j)$ denotes the cost corresponding to the close-to-global solution $U^*(v_j)$, i.e. $V^*(v_j) = J(U^*(v_j), v_j)$, and the parameter $\mu > 0$ is a weighting coefficient. After a state feedback $\hat{U}_i(x) = K_i x + g_i$ has been determined, an estimate $\hat{\varepsilon}_i$ of the maximal cost function approximation error in X_i is computed as follows:

$$\hat{\varepsilon}_i = \max_{j \in \{0,1,2,...,N_1\}} (J(K_i v_j + g_i, v_j) - V^*(v_j)) \quad (14)$$

If $\hat{\varepsilon}_i > \bar{\varepsilon}$, where $\bar{\varepsilon} > 0$ is the specified tolerance of the approximation error, the region X_i is divided and the procedure is repeated for the new regions.

The parallel computations in the algorithm for design of explicit NMPC controllers are implemented by using the function *parfor* (for creating parallel for-loops) of the Parallel Computing Toolbox for Matlab [9]. The basic concept of a *parfor*-loop in MATLAB software is the same as the standard MATLAB *for*-loop: MATLAB executes a series of statements (the loop body) over a range of values. Part of the *parfor* body is executed on the MATLAB client (where the *parfor* is issued) and part is executed in parallel on MATLAB workers [9]. MATLAB workers evaluate the *parfor*-loop iterations in no particular order, and independently of each other. Therefore, in order to create a *parfor*-loop, each iteration of the loop must be independent on all other iterations. *Parfor* divides the loop iterations into groups so that each worker executes some portion of the total number of iterations.

The parallel computing algorithm for design of explicit NMPC is described as follows:

Algorithm 1 Approximate mp-NLP based on parallel computations

Input: Data to problem P2, the parameter μ (used in Procedure 1), the approximation tolerance $\bar{\varepsilon}$.

Output: Partition $\Pi = \{X_1, X_2, ..., X_{N_{regions}}\}$ and associated PWL function $\hat{U} = \{\hat{U}_1, \hat{U}_2, ..., \hat{U}_{N_{regions}}\}$.

```
1.  Initialize the partition to the whole hyper-rectangle, i.e. $X_1 := X$, $\Pi := \{X_1\}$, $N_{regions} := 1$. Mark the hyper-rectangle $X_1$ as unexplored, flag:=1.
2.  while flag=1 do
3.      parfor i=1, 2, … , $N_{regions}$ do
4.          if the hyper-rectangle $X_i \in \Pi$ is unexplored then
5.              Compute a solution to problem P2 at the center point $v_0$ of $X_i$.
6.              if P2 has a feasible solution at $v_0$ then
7.                  Define a set of points $V_0 = \{v_0, v_1, v_2, ..., v_{N_1}\}$ associated to $X_i$.
8.                  Compute a close-to-global solution to problem P2 for $x$ fixed to each of the points $v_i$, $i = 1, 2, ..., N_1$.
9.                  if P2 has a feasible solution at all points $v_i$, $i = 1, 2, ..., N_1$ then
10.                     Compute an affine state feedback $\hat{U}_i(x)$ using Procedure 1, as an approximation to be used in $X_i$.
11.                     if a feasible solution was found then
12.                         Mark $X_i$ feasible. Compute an estimate $\hat{\varepsilon}_i$ of the error bound $\varepsilon_i$ in $X_i$ according to (14).
13.                         If $\hat{\varepsilon}_i > \bar{\varepsilon}$, mark the hyper-rectangle $X_i$ to be split. Otherwise, mark $X_i$ explored.
14.                     else
15.                         Mark $X_i$ infeasible. Compute the size of $X_i$ using some metric. If it is smaller than
```

```
                some given tolerance, mark $X_i$ *explored*. Otherwise, mark $X_i$ *to be split*.
16.             end if
17.           else
18.             Mark $X_i$ *infeasible*. Compute the size of $X_i$ using some metric. If it is smaller than some given tolerance, mark $X_i$ *explored*. Otherwise, mark $X_i$ *to be split*.
19.           end if
20.         else
21.           Mark $X_i$ *infeasible*. Compute the size of $X_i$ using some metric. If it is smaller than some given tolerance, mark $X_i$ *explored*. Otherwise, mark $X_i$ *to be split*.
22.         end if
23.       end if
24.     end parfor
25.     flag:=0
26.     if there are hyper-rectangles in $\Pi$ that are marked to be split then
27.       flag:=1
28.       parfor i=1, 2, … , $N_{regions}$ do
29.         if the hyper-rectangle $X_i \in \Pi$ is marked to be split then
30.           if $X_i$ is feasible then
31.             for j=1, 2, … , n do
32.               Split $X_i$ by a hyperplane through its center and orthogonal to the axis $x_j$. Denote the new hyper-rectangles with $X_{i,1}^j$ and $X_{i,2}^j$.
33.               Compute affine state feedbacks $\hat{U}_{i,1}^j(x)$ and $\hat{U}_{i,2}^j(x)$, valid respectively in $X_{i,1}^j$ and $X_{i,2}^j$, by applying Procedure 1.
34.               Compute estimates $\hat{\varepsilon}_{i,1}^j$ and $\hat{\varepsilon}_{i,2}^j$, respectively of the error bounds $\varepsilon_{i,1}^j$ in $X_{i,1}^j$ and $\varepsilon_{i,2}^j$ in $X_{i,2}^j$ according to (14). Let $\hat{\varepsilon}_i^j = \hat{\varepsilon}_{i,1}^j + \hat{\varepsilon}_{i,2}^j$.
35.             end for
36.             Split $X_i$ by a hyperplane through its center and orthogonal to the axis $x_j$ where $\hat{\varepsilon}_i^j$ is minimal. Mark the new regions $X_{i,1}^j$ and $X_{i,2}^j$ unexplored, remove $X_i$ from $\Pi$, and add $X_{i,1}^j$ and $X_{i,2}^j$ to $\Pi$. $N_{regions} := N_{regions} + 2$.
37.           else
38.             Split $X_i$ into hyper-rectangles $X_{i,1}$, $X_{i,2}$, …, $X_{i,N_s}$ by applying heuristic splitting rules. Mark $X_{i,1}$, $X_{i,2}$, …, $X_{i,N_s}$ unexplored, remove $X_i$ from $\Pi$, add $X_{i,1}$, $X_{i,2}$, …, $X_{i,N_s}$ to $\Pi$. $N_{regions} := N_{regions} + N_s$
39.           end if
40.         end if
41.       end parfor
42.     end if
43. end while
```

The details about the generation of a set of points associated to a given region, the computation of a close-to-global solution to problem P2, and the heuristic rules used to split a region in step 38 of the algorithm can be found in [4].

EXAMPLE

Consider the following 2-nd order compressor model [4] with x_1 being normalized mass flow, x_2 normalized pressure and u normalized mass flow through a close-coupled valve in series with the compressor:

$$\dot{x}_1 = B(\Psi_e(x_1) - x_2 - u) \tag{15}$$

$$\dot{x}_2 = \frac{1}{B}(x_1 - \Phi(x_2)) \tag{16}$$

The following compressor and valve characteristics are used [4]:

$$\Psi_e(x_1) = \psi_{c0} + H\left(1 + 1.5\left(\frac{x_1}{W} - 1\right) - 0.5\left(\frac{x_1}{W} - 1\right)^3\right) \tag{17}$$

$$\Phi(x_2) = \gamma \operatorname{sign}(x_2)\sqrt{|x_2|} \tag{18}$$

with $\gamma = 0.5$, $B = 1$, $H = 0.18$, $\psi_{c0} = 0.3$ and $W = 0.25$.

Like in [4], the control objective is to avoid surge, i.e. stabilize the system. This is formulated as:

$$J(U, x(t)) = \sum_{k=0}^{N-1}\left[\alpha(x_{t+k|t} - x^*)^T(x_{t+k|t} - x^*) + ku_{t+k}^2\right] + Rv^2 + \beta(x_{t+N|t} - x^*)^T(x_{t+N|t} - x^*) \tag{19}$$

with $\alpha, \beta, k, R \ge 0$ and the set-point $x_1^* = 0.40$, $x_2^* = 0.60$ corresponds to an unstable equilibrium point. We have chosen $\alpha = 1$, $\beta = 0$ and $k = 0.08$. The horizon is $T = 12$, which is split into $N = 15$ equal-sized intervals, leading to a piecewise constant control input parameterization. Valve capacity requires the input constraint to hold:

$$0 \le u(t) \le 0.3 \tag{20}$$

The pressure constraint:

$$x_2(t) \ge 0.4 - v \tag{21}$$

avoids operation too far left of the operating point. The variable $v \ge 0$ is a slack variable introduced in order to avoid infeasibility and $R = 8$ is a large weight. The approximation tolerance is chosen as $\bar{\varepsilon}(X_0) = \max(0.005, cV_{\min}^*)$ where $c = 0.03$ and $V_{\min}^* = \min_{x \in X_0} V^*(x)$. The state space to be partitioned is defined by $X = [0, 0.9] \times [0, 0.75]$. The partition of the approximate explicit NMPC controller has 320 regions and is shown in Fig. 1. The performance of the closed-loop system is simulated for initial condition $x(0) = [0.1\ 0.05]^T$ and the resulting response is depicted in Fig. 1.

The off-line computation of the explicit NMPC is performed on a 3 GHz Intel Core 2 Duo processor. The CPU time corresponding to 11 consecutive iterations (steps 3 to 42) of Algorithm 1 is shown in Fig. 2. For comparison in Fig. 2, the CPU time associated to the non-parallel implementation of the partitioning algorithm is also given. In Table 1, the average CPU time necessary to compute a single region of the partition and the total CPU time spent to obtain the partition of the explicit NMPC are given for both the parallel and the non-parallel methods. As it should be expected, the algorithm based on parallel computations performs faster than the non-parallel algorithm (which can be observed from Fig. 2 and Table 1). However, the improvement of computational efficiency is not significant with a *dual-core* processor. The computational efficiency can be improved furtherly on a *multi-core* processor.

It should be noted that a modification of Algorithm 1 was used, where the second *parfor*-loop includes the steps 32, 33, 34 (instead of steps 28 to 41). The reason is that the steps 32, 33, 34 involve the computations of *two* new regions (the region X_i is split by a hyperplane through its center and orthogonal to the axis x_j), which are performed in parallel with the available *dual-core* processor.

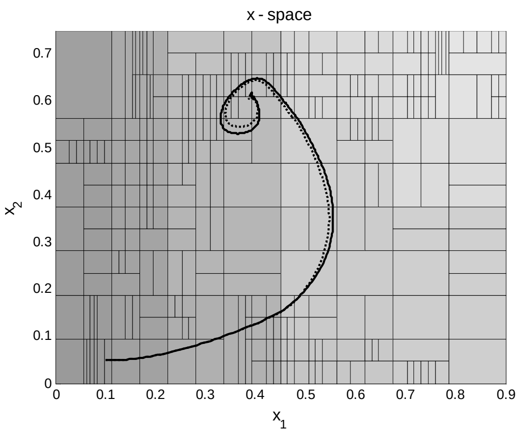

Fig. 1. The state space partition of the approximate explicit NMPC, the approximate (solid curve) and the exact (dotted curve) state trajectories.

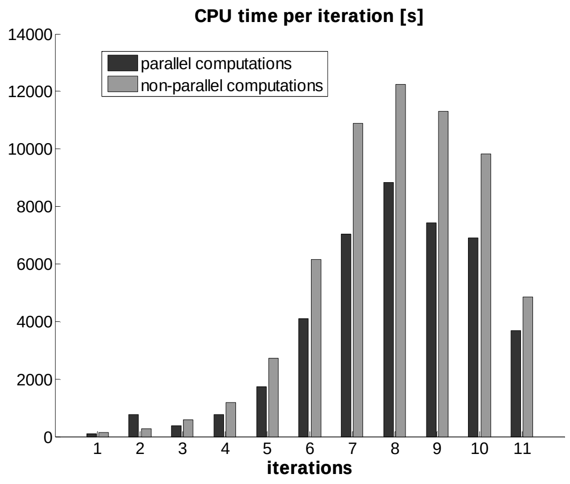

Fig. 2. CPU time corresponding to 11 consecutive iterations of the partitioning algorithm.

Table 1. CPU times for the methods based on parallel and non-parallel computations.

Method	Average CPU time per region [s]	Total CPU time for design of explicit NMPC [s]
Parallel computations	42	45786
Non-parallel computations	62	60190

CONCLUSIONS

In this paper, a parallel computing algorithm for design of explicit NMPC controllers is proposed. It uses the Parallel Computing Toolbox for MATLAB [9] and represents a parallel implementation of the approximate mp-NLP approach [4]. The parallel partitioning algorithm is applied to design an explicit NMPC for compressor surge control. It is shown that the parallel computations improve the off-line computational efficiency of the approximate mp-NLP approach.

REFERENCES

1. Allgöwer F., A. Zheng. Nonlinear Model Predictive Control, Progress in System and Control Theory, vol. 26, Birkhäuser Verlag, Basel, 2000.
2. Johansen T. A. On multi-parametric nonlinear programming and explicit nonlinear model predictive control. Proceedings of IEEE Conference on Decision and Control, Las Vegas, NV, 2002, vol. 3, pp. 2768-2773.
3. Johansen T. A. Approximate explicit receding horizon control of constrained nonlinear systems. Automatica, vol. 40, pp. 293-300, 2004.
4. Grancharova A., T. A. Johansen, and P. Tøndel. Computational aspects of approximate explicit nonlinear model predictive control. In R. Findeisen, F. Allgöwer and L. Biegler, editors, Assessment and Future Directions of Nonlinear Model Predictive Control, Lecture Notes in Control and Information Sciences, vol. 358, Springer-Verlag, Germany, pp. 181-192, 2007.
5. Pistikopoulos E. N., M. C. Georgiadis, and V. Dua. Multi-parametric Programming: theory, algorithms, and applications, Wiley-VCH, 2007.
6. Fiacco A. V. Introduction to sensitivity and stability analysis in nonlinear programming. Orlando, Fl: Academic Press, 1983.
7. Alessio A., A. Bemporad. A survey on explicit model predictive control. In: L. Magni, D. M. Raimondo and F. Allgöwer (Eds.), Nonlinear Model Predictive Control: Towards New Challenging Applications, Lecture Notes in Control and Information Sciences, vol. 384, Springer-Verlag, Germany, 2009, pp. 345-370.
8. Grancharova A., T. A. Johansen. Approaches to explicit nonlinear model predictive control with reduced partition complexity. Proceedings of European Control Conference, Budapest, Hungary, 2009, pp. 2414-2419.
9. Parallel Computing Toolbox^{TM} 4 User's Guide, The MathWorks, Inc., Natick, MA 01760-2098, USA, 2008.
10. Mayne D. Q., J. B. Rawlings, C. V. Rao, and P. O. M. Scokaert. Constrained model predictive control: Stability and optimality. Automatica, vol. 36, pp. 789-814, 2000.