SIMULATION STUDY OF PARALLEL MODEL PREDICTIVE CONTROL

Stefan Behrendt[1], **Peter Dünow**[1], **Bernhard Lampe**[2]

[1]Hochschule Wismar, University of Applied Sciences: Technology, Business and Design, Research Group CEA, Philipp-Müller-Straße, PF1210, 23952 Wismar, Germany
[2]University of Rostock, College of Computer Science and Electrical Engineering, Richard-Wagner-Str. 31 / H.8, 18119 Rostock, Germany

stefan.behrendt@hs-wismar.de (Stefan Behrendt)

Abstract

Over the decades the model predictive control (MPC) concept has proved successfully in controlling plants with complex dynamics. Due to its high computational complexity its usage is being limited to plants with slow dynamics like in the process industry. To address this limitation several techniques have been developed to enlarge the field of MPC to embedded systems with small sampling times. With the advent of multi-core controllers it seems reasonable that parallel algorithms for MPC could lower the computational burden. In this paper the speed-up behaviour of a parallelisation approach on a functional level, in particular the incorporated optimisation, is studied due to a discrete-time simulation with Matlab/Simulink. This approach leads to some extent to super-linear speed-up of more than eight on four workers. The paper pictures the necessary computational basics in MPC and constrained optimisation and explains the mentioned approach in detail. Numerical simulation results are given on a multi-input multi-output (MIMO) control problem and the efficiency of the approach is shown.

Keywords: Model Predictive Control, Parallel Algorithm, Discrete-Time Simulation, Computational Complexity

Presenting Author's Biography

Stefan Behrendt is a Research Associate in the research group Computational Engineering and Automation (CEA) at the Faculty of Engineering, University of Wismar, Germany. He obtained the degree (Honours) in Electrical Engineering in 2006 and M.Eng. in Process Automation in 2009 from University of Wismar. He joined the Ingenieurgesellschaft Auto und Verkehr (IAV) GmbH as a Process Engineer in 2006 and then started the PhD studies under supervision of CEA at the University of Wismar and in cooperation with the University of Rostock in 2007. His research interests are in the areas of mathematical programming, optimisation algorithms and its application in process control engineering. In particular the author is concerned with spark ignition engine control.

1 Introduction

Over the decades the model predictive control (MPC) concept has proved successfully in controlling plants with complex dynamics. Due to its high computational complexity its usage is being limited to plants with slow dynamics like in the process industry (e.g. [1]). Numerous companies developed reliable software for process automation systems (PAS) (e.g. [2]) and programmable logic controllers (PLC) (e.g. [3]).

In contrast, controlling plants with fast dynamics still poses a problem. The fast sampling times necessary prevented the MPC of these plants on standard embedded systems, beside the benefits are already proven [4, 5]. To address this limitation several techniques have been developed to enlarge the field of MPC to embedded systems with small sampling times.

An implementation with sufficient worst case timing for the MPC of a single-input single-output (SISO) system is presented in [6]. It suggests the enhancement of embedded systems by a digital signal co-processor (DSP) for the fast evaluation of the underlying algorithms.

The implementation strategies and the actual implementation on field programmable gate array (FPGA) chips are presented by [7] and [8] respectively. While the former address the specific architecture, like parallelism, explicitly and therefore the greater improvement is expected, the effort of implementing the necessary algorithms in a hardware description language should not be underestimated. To compete against high-potential micro-controllers in terms of computational time is a challenging task.

Another approach is the combination of on-line optimisation and a partial enumeration method [9]. The solutions of the optimisation problem with active constraint sets that appear with highest frequency are computed off-line and stored in a table. This table is searched online for the best control. In case, that meanwhile the online computation an active constraint set does not exist in the table, it is adapted. With this method a significant speed-up is possible. The drawbacks are performance degradation and the memory requirements.

The explicit MPC has gained much attention in the recent years. Therefore the state space is partitioned into polyhedral regions. The control law is formulated as a function of the plant state and the piecewise linear solutions to the control problem with respect to the constraints are calculated [10, 11]. The on-line computational complexity reduces to the selection of the appropriate control law depending on the actual state. Numerous successful applications followed [12, 13, 14]. On the other hand, explicit MPC is limited to low-dimensional plants and short control horizons (for an explanation of this term see section 2) as the number of regions grows exponentially with these parameters. A prohibitively large amount of memory would be necessary, which is addressed e.g. by [15]. Additionally the constraints on the control variables are fixed, which is undesired in some control problems.

With the advent of multi-core controllers for embedded systems [16, 17, 18] and multi-core DSPs [19] it seems reasonable that parallel algorithms for MPC could lower the computational burden. To the knowledge of the author the main developments occur in the field of large-scale and sparse problems [20, 21] and focus on the parallelisation of incorporated operations (e.g. matrix multiplication, factorisation or inversion and solution of system of equations) [22].

In this paper a parallelisation approach on a functional level, in particular the incorporated optimisation, for small-scale problems is presented. Therefore the optimisation is solved in dependency on a number of different initialisations on different cores of a multi-core architecture, which forms an embarrassingly parallel problem. The initialisations are determined by either an heuristic method or an initialisation routine. Because the initialisation is crucial for the computational complexity of the overall algorithm this approach leads to some extent to super-linear speed-up as it will be shown in the remainder of the paper.

The paper is organised as follows. The basics of MPC and algorithms for the solution of the incorporated optimisation are presented in section 2. In section 3 an possible extension to parallel computation is decribed. Section 4 and 5 explain the simulation system and the results of the parallel approach. The paper closes with a conclusion.

2 Model Predictive Control Basics

In this section a short introduction to the MPC fundamentals is presented. For a more comprehensive survey on the theory of MPC the reader is referred to [23] and [24].

The MPC method combines the advantages of predicting the behaviour of the plant, namely the output, and respects constraints on the actuators. Therefore the cost function

$$J(k) = \sum_{i=1}^{H_p} ||\hat{y}(k+i|k) - w(k+i|k)||^2_{q(i)} + \sum_{i=0}^{H_u-1} ||\Delta u(k+i|k)||^2_{r(i)} \quad (1)$$

with the prediction horizon H_p, the control horizon H_u, the weights on the error signal q, the weights on the rate of change of the difference control action r, the predicted output $\hat{y}$, the reference w and the difference control action Δu needs to get minimized with respect to Δu.

The prediction follows from the state equations of the plant

$$x(k+1) = Ax(k) + Bu(k) \quad (2)$$

$$y(k) = Cx(k) \quad (3)$$

with the states $x \in \Re^{n_\mathrm{x}\times 1}$, the input $u \in \Re^{n_\mathrm{u}\times 1}$ and the output $y \in \Re^{n_\mathrm{y}\times 1}$ by

$$\hat{y}(k) = \Psi\hat{x}(k) + \Upsilon u(k-1) + \Theta\Delta u(k)\,. \quad (4)$$

Thereby Ψ, Υ, Θ and $\Delta u(k)$ are in the notation as presented by [23]. With the reference signal $w(k+i|k)$, $i = 1 \ldots H_p$, the control difference over the prediction horizon

$$\varepsilon(k) = \begin{bmatrix} w(k+1|k) \\ w(k+2|k) \\ \vdots \\ w(k+H_p|k) \end{bmatrix} - \Psi\hat{x}(k) - \Upsilon u(k-1) \qquad (5)$$

leads to the cost functional

$$J(k) = \Delta u(k)^{\mathrm{T}} H \Delta u(k) - g^{\mathrm{T}} \Delta u(k) \qquad (6)$$

with

$$g = 2\Theta^{\mathrm{T}} Q \varepsilon(k) \text{ and } H = \Theta^{\mathrm{T}} Q \Theta + R. \qquad (7)$$

By means of the weighting matrices Q and R the resulting control is parametrized.

Additionally constraints on the actuating variables are defined by the linear matrix inequality

$$A\Delta u(k) \leq b \qquad (8)$$

with the well structured coefficient matrix

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 \\ -1 & 0 & -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & -1 & 0 & 0 \\ -1 & 0 & -1 & 0 & -1 & 0 \\ 0 & -1 & 0 & -1 & 0 & -1 \end{bmatrix} \qquad (9)$$

that cumulates associated difference control actions. The size and sparsity pattern of this matrix depends on the number of actuating variables and the chosen control horizon, but does not change its purpose. The specific structure of this matrix is important for the later parallel implementation.

The minimization of the cost function (Eq. 6) under the constraints (Eq. 8)

$$\min_{\Delta u} \left\{ \Delta u^{\mathrm{T}} H \Delta u - g^{\mathrm{T}} \Delta u \ : \ A\Delta u \leq b \right\} \qquad (10)$$

defines a mathematical standard problem that can be solved by quadratic programming (QP). A variety of methods for solving the QP are commonly used [25] and two of them are described in the next sections.

2.1 Hildreths Quadratic Programming Procedure

The procedure presented by HILDRETH [26, 27] belongs to the first algorithms for the solution of constrained quadratic optimisation problems. It solves the dual problem to Eq. 10

$$\min_{\lambda} \left\{ \lambda^{\mathrm{T}} P \lambda + d^{\mathrm{T}} \lambda \ : \ \lambda \geq 0 \right\} \qquad (11)$$

with

$$P = \frac{1}{4} A H^{-1} A^{\mathrm{T}} \qquad (12)$$

$$d = b - \frac{1}{2} A H^{-1} g \qquad (13)$$

which is of the same structure like the primal problem, but of increased dimension due to $A \in \Re^{2n_{\mathrm{u}}H_{\mathrm{u}} \times n_{\mathrm{u}}H_{\mathrm{u}}}$ (every actuating variable is subject to a lower and upper bound) while $H \in \Re^{n_{\mathrm{u}}H_{\mathrm{u}} \times n_{\mathrm{u}}H_{\mathrm{u}}}$. On the other hand, the simplified constraints make it easier to solve.

The unrestricted, optimal solution is given by

$$\lambda^{+} = -\frac{1}{2} P^{-1} d \qquad (14)$$

and by means of the Gauss-Seidel method the restricted, optimal solution is calculated element-wise while preserving the dual feasibility. The explicit formula for the ith element of λ in the mth iteration is

$$w_i^{m+1} = -\frac{1}{p_{ii}} \left(\frac{d_i}{2} + \sum_{j=1}^{i-1} p_{ij} \lambda_j^{m+1} + \sum_{j=i+1}^{2n_{\mathrm{u}}H_{\mathrm{u}}} p_{ij} \lambda_j^{m} \right) \qquad (15)$$

$$\lambda_i^{m+1} = max(0, w_i^{m+1}) \qquad (16)$$

where p_{ij} is the ijth element of the matrix P and the scalar d_i is the ith element of the vector d.

In case of convergence of the method the solution $\lambda^{\star}$ is used to calculate the optimal primal variables by

$$\Delta u^{\star} = \frac{1}{2} H^{-1} \left(g - A^{\mathrm{T}} \lambda^{\star} \right) . \qquad (17)$$

The drawbacks of the algorithm are the slow speed of convergence and the sensitivity to the precision of the used number system in comparison to competing methods like the active set method in the next section.

2.2 Active Set Method

The primal method by FLETCHER [28] identifies iteratively the active constraints set in the solution denoted by

$$I = \{ i \in \{1 \ldots 2n_{\mathrm{u}}H_{\mathrm{u}}\} \ : \ a_i \Delta u = b_i \} \qquad (18)$$

where a_i is the ith row of A and b_i the ith element of b. By definition the matrix $A_{\mathrm{I}} \in \Re^{n_{\mathrm{I}} \times n_{\mathrm{u}}H_{\mathrm{u}}}$ is a matrix, which rows are composed by the vectors a_i with $i \in I$.

The method utilizes the associated LAGRANGE-function to Eq. 6 and Eq. 8

$$L(\Delta u, \lambda) = \Delta u^{\mathrm{T}} H \Delta u - g^{\mathrm{T}} \Delta u + \lambda^{\mathrm{T}} (A\Delta u - b) \qquad (19)$$

where λ are the LAGRANGE multipliers.

By differentiation and setting to zero the equations

$$\frac{\partial L}{\partial \Delta u} = 2H\Delta u - g + A^{\mathrm{T}} \lambda = 0 \qquad (20)$$

$$\frac{\partial L}{\partial \lambda} = A\Delta u - b = 0 \, , \qquad (21)$$

characterize an extremum and the transition to an iterative method that calculates in every iteration m a step p towards the minimum while preserving the primal feasibility by an equality constraint

$$0 = 2H(\Delta u^m + p) - g + A_{\mathrm{I}}^{\mathrm{T}} \lambda_{\mathrm{I}} \qquad (22)$$

$$0 = A_{\mathrm{I}}(\Delta u^m + p) - b_{\mathrm{I}} \qquad (23)$$

follows. Due to eq. 18

$$A_{\mathrm{I}}\Delta u^m = b_{\mathrm{I}} \tag{24}$$

the Eq. 22 and 23 simplify to

$$\begin{bmatrix} 2H & A_{\mathrm{I}}^{\mathrm{T}} \\ A_{\mathrm{I}} & 0_{n_{\mathrm{I}}\times n_{\mathrm{I}}} \end{bmatrix} \begin{bmatrix} p \\ \lambda_{\mathrm{I}} \end{bmatrix} = \begin{bmatrix} -(2H\Delta u^m - g) \\ 0_{n_{\mathrm{I}}\times 1} \end{bmatrix}. \tag{25}$$

In case of violating an inactive restriction

$$A(\Delta u^m + p) - b > 0 \tag{26}$$

a line-search step is performed that prevents this violation and the associated restriction is added to the active set.

The existence of a negative element in λ_{I} implies that the associated restriction needs to be removed from the active set.

Summarised, the task of FLETCHERs method is to iteratively determine the active constraints in the solution. It needs as much iterations as changes to the active set needs to be performed.

3 Parallel Extension of Active Set Method

As stated in the previous section, the number of iterations in the active set method depends on the necessary number of changes to the active set. In case of known active set in the solution $I^\star$, the method needs only one iteration. It is worth noting that the closed solution can be given analytically, too.

$$\Delta u^\star = \frac{1}{2}H^{-1}g + H^{-1}A_{\mathrm{I}^\star}^{\mathrm{T}}\left(A_{\mathrm{I}^\star}H^{-1}A_{\mathrm{I}^\star}^{\mathrm{T}}\right)^{-1} \left(b_{\mathrm{I}^\star} - \frac{1}{2}A_{\mathrm{I}^\star}H^{-1}g\right) \tag{27}$$

Unfortunately the active set in the solution is usually not known in advance. Due to this fact we resort to the technique of estimating the active set. Such guesswork often poses problems, because finding appropriate rules for estimation is a challenging task and the quality of the estimation sometimes increases the computational complexity.

The parallel execution offers the chance of simultaneously solving the quadratic program with different initialisations. The suggested structure is shown in Fig. 1. The first thread solves the problem in the same manner as the sequential implementation would do. Therefore the active set of the solution in the previous time instance is used (known as „warmstart“). This strategy often leads to a decreased number of iterations, but could have contrary effect as well. In the parallel context this initialisation serves as a failsafe mechanism. The computational time of the parallel approach could never exceed the sequential implementation (disregarding the overhead for parallelisation).

The second thread benefits from the estimation of the active set. Because of the huge effort of a reliable heuristic estimation the usage of an extreme assumption like all constraints get activated is preferred. In such

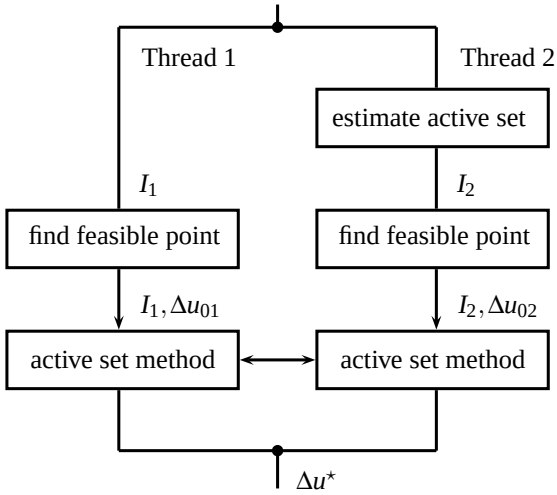

Fig. 1 Structure of the parallel implementation with two workers

conditions the most iterations in the active set method are expected. The drawback is the less usage of this advanced knowledge. Because extreme situations occur infrequently this thread does not often participate in the delivery of the solution. On the other hand, if it is participating the number of iterations and therefore the computational time could be drastically decreased.

A more systematic approach that includes the dynamic behaviour of the plant is the idea of this paper. Despite the slow speed of convergence is HILDRETHs procedure generating a sequence of dual variables that indicate active constraints by positive values. Because the actual solution to the dual problem is not of interest, but the fact that entries are positive already a limited number of iterations in HILDRETHs procedure might be sufficient for a reliable estimation of the active set by

$$I = \{i \in \{1 \ldots 2n_{\mathrm{u}}H_{\mathrm{u}}\} \,:\, \lambda_i > 0\}\,. \tag{28}$$

If the estimated active set equals the active set in the solution $I^\star$ would further iterations in HILDRETHs procedure not improve the result, but increase the computational complexity again. Hence, an optimal value for the iteration count exists that is control problem specific. A related approach that possibly results in suboptimal solutions in described in [29], but does not consider a parallel algorithm.

The calculation of a feasible starting point is necessary for primal methods like the active set method. This process is often referred to as Phase I. Due to the fact, that we want to start with an initial active set the starting point needs to satisfy Eq. 8 as well as Eq. 24. This could easily be done by Eq. 27, but because of the simple structure of the coefficient matrix in the inequality constraint (Eq. 9) a straightforward computation is possible that prevents matrix inverses.

The thread that finishes the optimisation first supplies the optimal difference control value $\Delta u^\star$ and interrupts the other thread. This interruption is the only communication necessary between the workers.

4 Benchmark plant for a class of systems

This paper deals with a special class of systems that consists of a main control variable, a main actuating variable and a number of auxiliary actuating variables. The purpose of the auxiliary variables is to dynamically support the main actuating variable, but statically retain to their references. Therefore zero-order holds are introduced, that allow for the definition of the auxiliary variables as actuating and control variables at the same time. The general system can be described by the discrete-time transfer function

$$G(z) = \begin{bmatrix} G_{11}(z) & G_{12}(z) & G_{13}(z) & \cdots & G_{1n}(z) \\ 0 & z^{-1} & 0 & \cdots & 0 \\ 0 & 0 & z^{-1} & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & z^{-1} \end{bmatrix}. \tag{29}$$

The benchmark plant has been chosen to be of dimension twenty, so $n = 20$ in Eq. 29, to demonstrate the significant reduction in computational time. Each subsystem is stable and of second order that leads to forty states that represent dynamic. The relation between the main control variable and the main actuating variable described by $G_{11}(z)$ is of slower dynamic in comparison to the remaining sub-systems. This encourages the evident active aid by the auxiliary variables for controlling the plant. For clarity the gains of the sub-systems have been selected to be all positive.

Such systems occur e.g. in spark ignition engine control [30, 31, 32]. The main control variable could be the engine speed and the main actuating variable the mass air flow into the cylinder regulated by the throttle. Because of the intake manifold the dynamic is relatively slow. In contrast, the advance angle and exhaust gas recirculation (EGR) are of faster dynamic and serve as auxiliary variables. In the steady state they are retained to not impose deterioration of engine efficiency.

5 Simulation Study

As simulation hardware a standard personal computer architecture with two Intel Xeon E5335 processors that include eight cores in total is used. The operating system is Windows 7 (32 Bit). The simulation software used is Matlab/Simulink by The Mathworks Inc. It provides a comprehensive simulation system with support for e.g. fixed-point models and signal processing algorithms as libraries known as toolboxes as well as continuous and discrete time models.

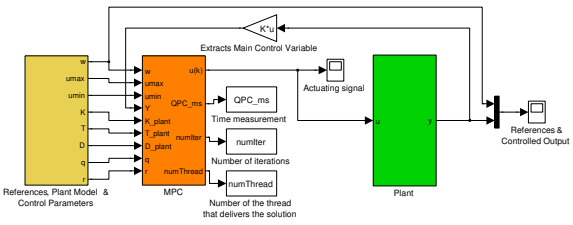

Fig. 2 Simulation model

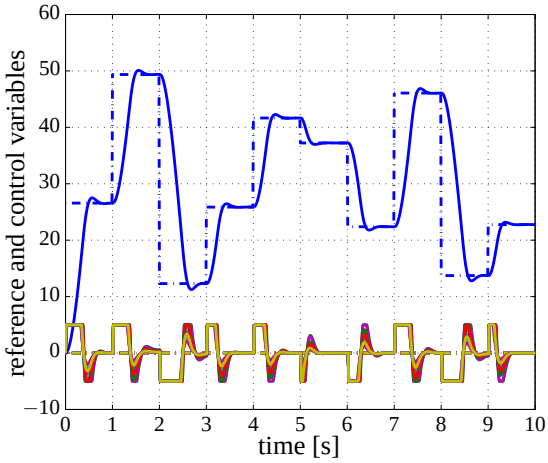

Fig. 3 Control scenario of the plant with references (dash-dot) and control signals (solid)

The simplified model for the evaluation of the proposed algorithm is shown in Fig. 2. The yellow block on the left supplies the references w, the constraints on the actuating variables $u_{\max}$ and $u_{\min}$, the parameters of the plant K, T and D as well as the parameters of the control algorithm q and r (Eq. 1) to the model predictive control block. The parameters of the plant may be changed during run-time that allows for an adaptive control. In the actual study the plant is assumed to be linear, so the parameters are chosen to be static. The control can be parametrised due to changes on q and r during run-time as well.

The model predictive control block (orange) incorporates a Simulink S-Function [33]. It qualifies the user to extend the capabilities of the simulation system. The S-Function block contains hand-written source code in e.g. C, C++ and Fortran and is the analogue to built-in blocks. The integration of new functionality by means of a S-Function follows general rules and it can accommodate continuous, discrete, and hybrid systems. The model predictive control algorithm is written in C to address the planned implementation on embedded systems. The parallelisation of the incorporated optimisation (as explained in section 3) is carried out by OpenMP [34]. The implementation details are omitted, because they are not in the focus of this paper.

The plant block (green) replaces the real plant and is simulated as a discrete-time state space model. A discrete-time representation is chosen that the simulation time is basically determined by the solution to the optimisation problem and not by the ODE solver.

The investigated control cycle is shown in Fig. 3. The control parameters are $H_{\mathrm{p}} = 20$ and $H_{\mathrm{u}} = 5$. The main control variable (blue) shows satisfactory reference tracking and is supported by the auxiliary variables (shown at the bottom of the figure). However, in the stationary operating points their influence is repealed. The main actuating variable is not shown for clarity.

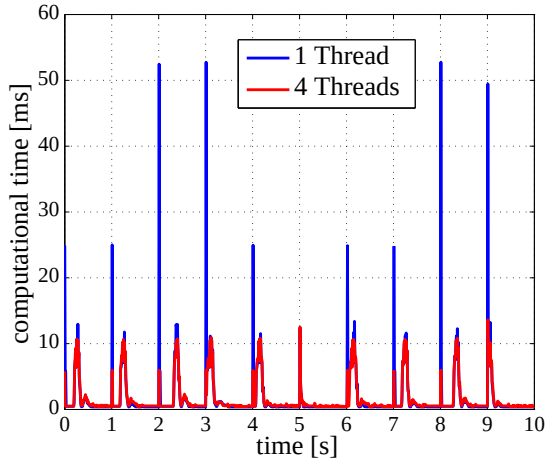

Fig. 4 Computational time for the MPC algorithm for sequential (blue) and parallel (red) implementation with the heuristic initialisation

5.1 Heuristic Method

The inspection of the control cycle reveals the critical time instants in terms of computational time. In case of a changing reference for the main control variable a large number of variables are constrained. Thus, in the time instant of the change many constraints need to get activated that leads to an excessive computational time as shown in Fig. 4. Because of the common characteristic of positive gains the same constraints (upper or lower) are active for all variables.

The number of workers is chosen to be four and are initialised by the assumption of

1. the active set of the previous time instant is valid,
2. the active set is empty,
3. all the constraints associated to the upper bound are active and
4. all the constraints associated to the lower bound are active.

The resulting computational time is shown in Fig. 4 and summarised in Tab. 1. Despite the fact, that the assumptions 3 and 4 are valid in less than one percentage of the control cycle, it is their achievement of computational speed-up of more than eight. Hence, a super-linear speed-up is achieved. While the minimal computational time shows a minor increase due to the parallelisation overhead, the maximum is decreased considerably. In matters of the implementation on embedded systems it is the maximum time that is crucial. The overall time is reduced by twelve percentage. It is interesting to note, that the number of iterations increase. Due to the characteristics of the algorithm a deactivation of few constraints may be more costly than the activation of much constraints. In this case the initialisation with the empty set is more favourable.

Tab. 1 Benchmark data for sequential and parallel implementation with the heuristic initialisation

Algorithm	Computational time [ms]			No. of active set iterations
	min	max	sum	
sequential	0.38	52.76	1814	3890
parallel	0.41	13.62	1592	5339

5.2 Initialisation Routine

The more systematic initialisation by means of HILDRETHs procedure promise a better balance between the workers.

Therefore the workers are initialised by the assumption of

1. the active set of the previous time instant is valid,
2. the active set is empty,
3. that every $\lambda_i > 0$ corresponds to an active constraint and
4. that every $\lambda_i > \frac{\bar{\lambda}}{10}|_{\lambda>0}$ corresponds to an active constraint.

The assumption 4 initialises the active set with the constraints that LAGRANGE multipliers are greater than a tenth of the mean value of the positive multipliers. This accounts for the previously mentioned rule to avoid the deactivation of constraints by disregarding small positive multipliers. They are the potential source of accidental choice due to the early interruption of the procedure. Such a threshold is worked out depending on the process.

The systematic initialisation leads to an involvement of assumption 3 and 4 of approximately fifteen percentage. The improved initialisation is having a further impact on the computational time as shown in Tab. 2 in comparison to Tab. 1. The more HILDRETH iterations are performed the better estimate of the active set in the solution is obtained. This can be judged by the amount of active set iterations that decrease. This conclusion is evident by examining the time instant $t = 6.2$s in Fig. 5.

Tab. 2 Benchmark data for parallel implementation with the initialisation by HILDRETHs procedure

No. of HILDRETH iterations	Computational time [ms]			No. of active set iterations
	min	max	sum	
1	0.41	7.42	1260	2714
3	0.46	7.04	1303	1847
5	0.41	7.33	1265	1717

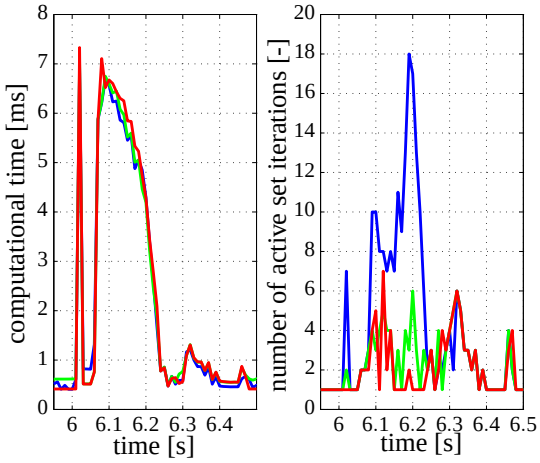

Fig. 5 Comparison of the parallel implementation with the initialisation by HILDRETHs procedure with one (blue), three (green) and five (red) iterations

In contrast, the computational time does not show a significant dependency. The numerical complexity of the HILDRETH iterations outweigh the reduced number of active set iterations. With further increasing number of HILDRETH iterations the computational time would increase, because the initialisation is too costly and the worker with assumption 1 or 2 will solve the problem.

6 Conclusion

In summary a parallel implementation of an active-set algorithm incorporated in the model predictive control scheme for a shared-memory computer is presented . The parallelisation is based on the initialisation of the optimisation algorithm, that takes place by either an heuristic method or an initialisation routine. Moreover, computational results in a simulation environment have been presented that documents the efficiency of the approach for a benchmark control cycle. Within this cycle super-linear speed-up is obtained at particular time instants that allow for a reduction of 85% in the peak computational time and the overall simulation time is reduced by thirty percentage. It should be emphasised that in contrast to other parallelisation studies this approach is much more memory intensive, because several optimisation are running in parallel. Thus, the approach is limited to small-size problems as accounted for the implementation in embedded systems.

Finally, it is noteworthy to point out that the obtained results are promising to address timing problems in associated fields. The solution of overdetermined linear systems is a common task in numerical algebra and may be formulated as quadratic optimisation problem. Hence, the approach may have general interest in other areas such as constrained least square solutions of overdetermined systems that appear e.g. in adaptive techniques.

7 References

[1] Stephen Bassett and Michiel van Wijck. Application of Predictive Control Technology at BP's Crude Oil Terminal at Grangemouth. Technical report, BP Oil Grangemouth Refinery Ltd. and Honeywell Hi-Spec Solution Ltd., 1999.

[2] Aspen Technology, Inc. aspenONE - Advanced Process Control. http://www.aspentech.com.

[3] Siemens AG. SIMATIC PCS 7 APC-Portfolio. http://www.automation.siemens.com, Oktober 2008.

[4] Bart Saerens, Moritz Diehl, Jan Swevers, and Eric Van den Bulck. Model Predictive Control of Automotive Powertrains - First Experimental Results. In *Proceedings of the 47th IEEE Conference on Decision and Control*, pages 5692 –5697, 9-11 December 2008.

[5] Adam Mills, Adrian Wills, and Brett Ninness. Nonlinear model predictive control of an inverted pendulum. In *ACC09*, June 2009.

[6] Adrian Wills, Dale Bates, Andrew Fleming, Brett Ninness, and S.O. Reza Moheimani. Model predictive control applied to constraint handling in active noise and vibration control. *IEEE Transactions on Control Systems Technology*, 16(1):3–12, December 2008.

[7] Geoff Knagge, Adrian Wills, Adam Mills, and Brett Ninness. ASIC and FPGA Implementation Strategies for Model Predictive Control. In *European Control Conference (ECC)*, August 2009.

[8] K.V. Ling, S.P. Yue, and J.M. Maciejowski. A FPGA implementation of model predictive control. In *American Control Conference*, 14-16 June 2006.

[9] Gabriele Pannocchia, James B. Rawlings, and Stephen J. Wright. The partial enumeration method for model predictive control: Algorithm and examples. Technical Report 1, TWMCC – Texas-Wisconsin Modeling and Control Consortium, 22 March 2006.

[10] Alberto Bemporad, Manfred Morari, Vivek Dua, and Efstratios N. Pistikopoulos. The explicit linear quadratic regulator for constrained systems. In *Automatica*, volume 38, pages 3 – 20, 2002.

[11] Petter Tondel, Tor Arne Johansen, and Alberto Bemporad. An algorithm for multi-parametric quadratic programming and explicit mpc solutions. In *Automatica*, volume 39, pages 489 – 497, 2003.

[12] Peter Ortner, Peter Langthaler, José Vicente García Ortiz, and L. del Re. Mpc for a diesel engine air path using an explicit approach for constraint systems. In *Proceedings of the 2006 IEEE International Conference on Control Applications*, 4-6 October 2006.

[13] Gerrit Naus, Roel van den Bleek, Jeroen Ploeg, Bart Scheepers, Rene van de Molengraft, and Maarten Steinbuch. Explicit mpc design and performance evaluation of an acc step-&-go. In *Proceedings of the 2008 American Control Conference*, 11-13 June 2008.

[14] Alicia Arce, Alejandro J. del Real, Carlos Bordons, and Daniel R. Ramirez. Real-Time Implementation of a Constrained MPC for Efficient Airflow Control in a PEM Fuel Cell. IEEE Transactions on Industrial Electronics, Accepted July 2009.

[15] J.A. Rossiter and P. Grieder. Using interpolation to simplify explicit model predictive control. In *Proceeding of the 2004 American Control Conference*, pages 885 – 890, 30 June - 2 July 2004.

[16] Parallax Inc. Propeller. http://www.parallax.com/propeller/.

[17] Infineon Technologies AG. TC1796 (AudoNextGeneration). http://www.infineon.com.

[18] XMOS Ltd. XS1-G4: 4-core processor. http://www.xmos.com.

[19] Texas Instruments Inc. TMS320C647x Multicore DSPs. http://www.ti.com.

[20] J. Gondzio and A. Grothey. *Computational Finance and its Applications II*, chapter Solving Nonlinear Financial Planning Problems with 10^9 Decision Variables on Massively Parallel Architectures. WIT Press, 2006.

[21] J. Gondzio and A. Grothey. Parallel interior point solver for structured quadratic programs: Application to financial planning problems. In *Annals of Operations Research*, volume 152, pages 319 – 339, 2007.

[22] A.E.B. Ruano and H.A. Daniel. Parallel implementation of an adaptive generalized predictive control algorithm. In *European Control Conference*, Brussels, 1-4 July 1997.

[23] Jan Marian Maciejowski. *Predictive Control with Constraints*. Pearson Education Limited, 2002.

[24] E.F. Camancho and C. Bordons. *Model Predictive Control*. Springer, 2004.

[25] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer, second edition edition, 2006.

[26] Clifford Hildreth. A quadratic programming procedure. In *Naval Research Logistics Quarterly*, volume 4, pages 79 – 85, 1957.

[27] Alfredo N. Iusem and Alvaro R. de Pierro. On the convergence properties of hildreths quadratic programming algorithm. In *Mathematical Programming*, volume 47, pages 37 – 51, 1990.

[28] R. Fletcher. *Practical Methods of Optimization*, volume 2: Constrained Optimization. John Wiley & Sons, 1981.

[29] Edwin T. van Donkelaar, Okko H. Bosgra, and Paul M.J. Van den Hof. Constrained model predictive control with on-line input parametrization. In *Proceedings of the 38th Conference on Decision & Control*, December 1999.

[30] C. Fritzsche, P. Dünow, S. Behrendt, P. Seemann, M. Schnaubelt, and M. Schultalbers. Predictive speed and torque control. In *Proceedings of 7. Symposium "Steuerungssysteme für den Antriebsstrang"*, Berlin, Germany, 2009.

[31] C. Fritzsche, H.-P. Dünow, B. Lampe, and M. Schultalbers. Torque coordination of spark ignition engines based on predictive control. In *Proceedings of International Conference on Methods and Models in Automation and Robotics*, 2007.

[32] P. Dünow, K. Lekhadia, M. Köller, and T. Jeinsch. Model based predictive control of spark ignition engine processes. In *Proceedings of International Conference on Methods and Models in Automation and Robotics*, 2005.

[33] The Mathworks Inc. Simulink 7 - Developing S-Functions, March 2010.

[34] OpenMP Architecture Review Board. The OpenMP API specification for parallel programming. http://openmp.org/.