Automatic Deployment of Industrial Embedded Model Predictive Control using qpOASES*

D. K. M. Kufoalor[1], B. J. T. Binder[1], H. J. Ferreau[3], L. Imsland[1], T. A. Johansen[1,2], and M. Diehl[4]

Abstract— **Different high-speed quadratic programming (QP) solvers are incorporated into an ANSI C code generation framework for embedded Model Predictive Control (MPC). The controllers developed are based on step response (linear) models and design configurations obtained from SEPTIC, Statoil's software tool for MPC applications. In order to achieve high online computational efficiency, offline computations/preparations are made at the code generation stage, and appropriate problem data are used in the QP solvers. We discuss implementation aspects arising when running an embedded MPC controller on an industrial PLC and present results of hardware-in-the-loop simulation tests for two challenging industrial applications. The results indicate that the online active-set strategy as implemented in the software package qpOASES exhibits superior performance compared to both a tailored interior-point method and a primal-dual first-order method for the step response class of models considered in this paper.**

I. Introduction

Automatic code generation has been introduced to the model predictive control (MPC) community in [1]. Current developments (e.g. [2], [3], [4]) indicate that it has become an essential tool for producing highly efficient solvers tailored for specific MPC problems. Since the problem structure and dimensions are usually known in advance for many practical MPC problems, specific code optimization techniques, offline computations, and other algorithmic preparations that boost the speed of online computations can be employed. This results in customized code that features fewer branches, avoids unnecessary operations and function calls, and will be well suited for further compiler optimization.

A recent development in embedded MPC code customization can be found in [5], where the usefulness of employing target-specific optimization techniques in an MPC algorithm is illustrated. The techniques in [5] boil down to the choice of a specific optimized *micro-kernel* for internal linear algebra routines based on *a priori* knowledge of the target platform hardware architecture. Hence, in embedded MPC applications where a specific target platform is desired, such target-specific techniques can also be incorporated into a code generator.

Automatic code generation is currently used to export customized solver codes from convex optimization software, such as CVXGEN [1] and FiOrdOs [2]. The code generators in both software tools accept configuration data that describes a quadratic programming (QP) problem, and produce solver code that implements interior-point and first-order methods, respectively. Alternative approaches are implemented in FORCES [3] and μAO-MPC [4], where the multistage problem formulation of MPC is chosen as the preferred starting point for the code generators. In this way, the characteristic structure in MPC problems can be well exploited for a particular solution method.

The above developments bring optimization algorithms close to the efficiency level desirable in industrial implementations and engineering. However, the existing MPC code generators depend on MPC formulations typically used in academic studies. An MPC code generator that incorporates design configurations used in many successful industrial MPC tools is reported in [6], [7], [8], with successful embedded implementations. In [7], feasibility studies on the use of tailored QP solvers suitable for a sparse MPC formulation were presented. Specifically, efficient customized interior-point and first-order methods were incorporated into the MPC code generation framework.

This paper presents a new approach used to obtain the embedded controllers for the applications presented in [7], [8]. A new ANSI C embedded variant of the online active set strategy of qpOASES [9], [10] is used for the condensed QP problem resulting from the industrial MPC problems studied in [7], [8]. In this case, the QP solver is not tailored to a specific QP problem data. It rather depends on QP problem data updates from functions produced by the MPC code generator.

Since the main target is embedded real-time applications, it is essential that the generated code is highly portable and uses static memory structures. It is also important that the solver incorporated produces deterministic execution time, low memory footprint, and supports both double and single precision floating point arithmetic. Several other considerations made in the automatic deployment of the embedded controllers are discussed in this paper. A PLC implementation is used in a hardware-in-the-loop simulation study to illustrate the performance of the current approach, compared to the sparse and fully tailored MPC implementations reported in [7], [8].

The paper is organized as follows: Section II introduces

[1] Department of Engineering Cybernetics, Norwegian University of Science and Technology (NTNU), O.S. Bragstads plass 2D N-7491 Trondheim, Norway. {kwame.kufoalor, benjamin.binder, lars.imsland, tor.arne.johansen}@itk.ntnu.no

[2] Center for Autonomous Marine Operations and Systems, NTNU.

[3] ABB Corporate Research, Segelhofstrasse 1K, CH-5405 Baden-Dättwil, Switzerland. joachim.ferreau@ch.abb.com

[4] Institute of Microsystems Engineering (IMTEK) University of Freiburg. moritz.diehl@imtek.uni-freiburg.de

* This work was supported by the Research Council of Norway and Statoil. The research leading to these results has also benefited from collaboration within the European Union's Seventh Framework Programme under EC-GA No. 607957 TEMPO – Training in Embedded Predictive Control and Optimization.

the MPC problem formulation, followed by the QP problems suitable for the solvers considered in this work; Section III briefly discusses embedded QP solvers; Section IV presents the embedded variant of qpOASES; and Section V discusses implementation aspects of embedded MPC. The two industrial MPC applications and the corresponding results obtained from hardware-in-the-loop simulations are described and discussed in sections VI and VII, respectively. Section VIII concludes the paper.

II. MPC PROBLEM FORMULATION

The MPC problem for a plant with a single output y (also known as the controlled variable (CV)) and a single control input u (manipulated variable (MV)), can be formulated as

$$\min \sum_{k=k_0+H_w}^{k_0+H_p} y_e(k)^T \bar{Q}_y y_e(k) + \sum_{k=k_0}^{k_0+H_u-1} \Delta u(k)^T \bar{P} \Delta u(k) + \bar{\rho}_h \bar{\varepsilon}_h + \bar{\rho}_l \bar{\varepsilon}_l \tag{1a}$$

subject to

$$y(k) = y(k|k_0), \qquad k \in \{k_0+H_w, \dots, k_0+H_p\}, \tag{1b}$$
$$\underline{y} - \varepsilon_l \le y(k) \le \bar{y} + \varepsilon_h, \qquad k \in \{k_0+H_w, \dots, k_0+H_p\}, \tag{1c}$$
$$\varepsilon_h \ge 0,\ \varepsilon_l \ge 0, \tag{1d}$$
$$\underline{u} \le u(k) \le \bar{u}, \qquad k \in \{k_0, \dots, k_0+H_u-1\}, \tag{1e}$$
$$\underline{\Delta u} \le \Delta u(k) \le \overline{\Delta u}, \qquad k \in \{k_0, \dots, k_0+H_u-1\}, \tag{1f}$$
$$u(k) = u(k-1) + \Delta u(k), \qquad k \in \{k_0, \dots, k_0+H_u-1\}, \tag{1g}$$

where k_0 is the initial time instant, H_p and H_u are the lengths of prediction and control horizons, respectively, and $H_w > 1$ specifies the number of initial steps in which the deviations from the CV reference $r(k)$, $y_e(k) = y(k) - r(k)$, are not penalized. $\bar{Q}_y \ge 0$, and $\bar{P} > 0$ are control target penalties (or weights), and $\Delta u(k)$ is the change in control input $u(k)$. The *slack* variables $\bar{\varepsilon}_h, \bar{\varepsilon}_l$, weighted by $\bar{\rho}_h, \bar{\rho}_l > 0$, relax the upper and lower constraints of the CV so that infeasibility cannot occur even in case of large disturbances or prediction model errors. Nominal closed-loop stability of the MPC problem can be achieved by an adequate choice of the weights $\bar{Q}_y$, $\bar{P}$ and the horizon lengths H_p and H_u.

The prediction model $y(k|k_0)$ can be derived from a linear single-input single-output (SISO) step response model, which is obtained from the plant by applying a step in the input u and recording the response of the output y. In the following subsections, the MPC problem is transformed into a more compact form and extended to multi-input multi-output (MIMO) systems.

A. General sparse QP

The decision variables in (1) can be stacked into vectors as follows:

$$Y_i = \big(y(k_0+H_w), y(k_0+H_w+1), \dots, y(k_0+H_p)\big),$$
$$U_j = \big(u(k_0), u(k_0+1), \dots, u(k_0+H_u-1)\big),$$
$$\Delta U_j = \big(\Delta u(k_0), \Delta u(k_0+1), \dots, \Delta u(k_0+H_u-1)\big).$$

Let $\mathscr{T}_i = \big(r(k_0+H_w), \dots, r(k_0+H_p)\big)$, $Y_{i,e} = Y_i - \mathscr{T}_i$, $P_j = I_{H_u} \otimes \bar{P}$, and $Q_i = I_{H_p-H_w+1} \otimes \bar{Q}_y$, where $\otimes$ denotes the Kronecker product. The subscript $i = 1, 2, \dots, n_{CV}$, and $j = 1, 2, \dots, n_{MV}$, where n_{CV} and n_{MV} are the number of CVs and MVs in a MIMO system. Define

$$\mathscr{T} = [\mathscr{T}_1^T, \dots, \mathscr{T}_{n_{CV}}^T]^T, \quad Y = [Y_1^T, \dots, Y_{n_{CV}}^T]^T,$$
$$\Delta U = [\Delta U_1^T, \dots, \Delta U_{n_{MV}}^T]^T, \quad U = [U_1^T, \dots, U_{n_{MV}}^T]^T,$$
$$P = \texttt{blkdiag}(P_1, \dots, P_{n_{MV}}), \ Q_y = \texttt{blkdiag}(Q_1, \dots, Q_{n_{CV}}),$$

and $\tilde{u}(k_0-1) = [u_1(k_0-1), \dots, u_{n_{MV}}(k_0-1)]^T$. Using the above definitions, the MPC problem (1) can be converted into a more compact matrix-vector QP problem:

$$\min\ Y^T Q_y Y + \Delta U^T P \Delta U - 2\mathscr{T}^T Q_y Y + \rho_h^T \varepsilon_h + \rho_l^T \varepsilon_l \tag{2a}$$

subject to

$$Y = \Theta \Delta U + \Psi \Delta \tilde{U}(k_0-1) + \Upsilon \tilde{u}(k_0-1) + V(k_0), \tag{2b}$$
$$GY \le g + M_h \varepsilon_h + M_l \varepsilon_l, \ \ \varepsilon_h \ge 0, \varepsilon_l \ge 0, \tag{2c}$$
$$E \Delta U \le e, \tag{2d}$$
$$FU \le f, \tag{2e}$$
$$KU = \Lambda \tilde{u}(k_0-1) + \Delta U, \tag{2f}$$

where the cost function neglects the terms that do not depend on any decision variable. The matrix M_l has 1 at corresponding entries where g represents a lower limit, and 0 at entries where g represents an upper limit, and vice versa for M_h. The size of ε_h and ε_l correspond to the number of CVs with high and low limits, respectively. The QP problem (2) is based on the formulation outlined in [11, §3.2.1], and it is straightforward to deduce all the constraint matrices (E, F, G) and their corresponding vectors (e, f, g), as well as the matrices $K \succ 0$ and Λ.

The prediction model matrices Ψ, Υ, and Θ are composed of the dynamic matrices (step response) of the plant, and the predictions depend on the previous control moves stored in $\Delta\tilde{U}(k_0-1)$ and $\tilde{u}(k_0-1)$. Output feedback and integral control are introduced through the constant disturbance component $V(k_0) = [\mathbf{1}^T v_1, \dots, \mathbf{1}^T v_{n_{CV}}]^T$, defined for each CV as the difference between the latest measurement and the corresponding predicted value. Each element of the vector $\mathbf{1}$ has a value of 1. Note that (2b) can be easily extended to include known disturbances.

By defining a decision vector $\bar{z} = \big(\Delta U, U, Y, \varepsilon_h, \varepsilon_l\big)$, problem (2) can be rewritten in the standard form

$$\min \Big\{ \tfrac{1}{2}\bar{z}^T H_s \bar{z} + \bar{z}^T g_s(k_0) \ \Big|\ \bar{A}_i \bar{z} \le \bar{b}_i,\ A_e \bar{z} = b_e(k_0) \Big\}, \tag{3}$$

where $H_s = H_s^T \succeq 0$, and the vectors $g_s(k_0)$ and $b_e(k_0)$ can be updated at every sampling time. The simple bounds on U, ΔU, ε_h, and ε_h can be moved into a convex set $\mathbb{Z}$, leading to a highly structured, sparse QP problem:

$$\min \Big\{ \tfrac{1}{2}\bar{z}^T H_s \bar{z} + \bar{z}^T g_s(k_0) \Big| A_i \bar{z} \le b_i,\ A_e \bar{z} = b_e(k_0), \bar{z} \in \mathbb{Z} \Big\} \tag{4}$$

B. Condensed QP

The variables Y and U can be eliminated from the sparse QP formulation (2), leading to a condensed QP with the decision vector $z = \big(\Delta U, \varepsilon_h, \varepsilon_l\big)$:

$$\min \Big\{ \tfrac{1}{2} z^T H_d z + z^T g_d(k_0) \ \Big|\ A_d z \le b_d(k_0),\ z_{lb} \le z \le z_{ub} \Big\} \tag{5}$$

It is well-known that this condensed formulation features a significantly lower-dimensional decision vector z, which comes at the expense of less structured (or even fully dense) QP matrices H_d and A_d.

The matrices and vectors in (5) are obtained directly from (2) by considering the following definitions:

$$Y := \Gamma(k_0) + \Theta\Delta U(k_0), \tag{6}$$

$$U := K^{-1}[\Lambda\tilde{u}(k_0-1) + \Delta U(k_0)], \tag{7}$$

$$\gamma := -2\Theta^T Q_y, \tag{8}$$

$$\zeta(k_0) := \mathscr{T}(k_0) - \Gamma(k_0), \tag{9}$$

where

$$\Gamma(k_0) = \Psi\Delta\tilde{U}(k_0-1) + \Upsilon\tilde{u}(k_0-1) + V(k_0). \tag{10}$$

The result is

$$H_d = 2\cdot\texttt{blkdiag}(\bar{H}_d, 0, 0), \tag{11a}$$

$$g_d(k_0) = \begin{bmatrix}\zeta(k_0)^T\gamma^T & \rho_h^T & \rho_l^T\end{bmatrix}^T, \tag{11b}$$

$$A_d = \begin{bmatrix} G\Theta & -M_h & -M_l \\ FK^{-1} & 0 & 0 \end{bmatrix}, \tag{11c}$$

$$b_d(k_0) = \begin{bmatrix} -G\Gamma(k_0) + g \\ -FK^{-1}\Lambda\tilde{u}(k_0-1) \end{bmatrix}, \tag{11d}$$

where $\bar{H}_d = \Theta^T Q_y \Theta + P$.

III. Embedded QP Solvers

One approach to obtaining a high-speed solver is to solve the QP problem explicitly [12]. An explicit solver precomputes (offline) the solution of a parameterized QP problem for a range of parameters over polyhedral partitions. This results in a piecewise affine optimizer function that is stored in a look-up table for fast online evaluation. The number of partitions may grow exponentially with the problem size, rendering the explicit approach viable only for problems with a few constraints. Fast online solvers may therefore be preferable in general.

Existing online approaches are mainly first-order or second-order iterative methods, when classified according to the highest order of information used in finding the solution to the QP problem. First-order methods perform typically many, but cheap, iterations, and can be easily warm-started to attain significant computational speed-up. Also the sparse structure of (4) can be easily exploited by first-order method solvers. In addition, certification properties [13] and the small size of the resulting solver code make first-order methods preferable in some embedded real-time applications [7], [14], [15].

In contrast to first-order methods, second-order methods need fewer, but more expensive, iterations to converge to a solution. Second-order methods can be further grouped into interior-point (IP) and active-set methods, where the iterations of active-set methods are typically more, but cheaper, than those of IP methods. Warm-starting of IP methods is a challenge (see [16] for a good discussion), whereas active-set methods have remarkable warm-start capabilities. The sparse QP problem (3) is suitable for IP methods and some active-set solvers [17] that are specially designed for sparse QP exploitation. However, exploiting sparsity in active-set methods may lead to iterations that are as expensive as IP iterations [18], motivating the use of the dense QP formulation (5) (especially when the control horizon is short).

QP solvers that are fast for small/medium sized problems (up to a few hundreds of variables and constraints) are particularly suitable for embedded MPC since practical examples of embedded control applications with high sampling rates usually translate into small/medium size QP problems.

IV. qpOASES

A. *Online Active Set Strategy*

qpOASES is an open-source implementation of the online active set strategy [9], [19]. It is a primal-dual active set method that introduces a homotopy path from the solution of the current QP problem to that of the next one. When following this path, the optimal set of active constraints is identified online while performing efficient rank-1 updates to maintain the internal matrix factorizations.

Compared to classical primal or dual active-set methods, the online active set strategy offers a couple of advantages that makes it particularly suited for MPC applications:

- no need to find a feasible initial point as all iterates along the homotopy path remain primal and dual feasible;
- improved warm-start capabilities (though problem-dependent) and re-use of matrix factorizations across subsequent QP instances;
- when the solver cannot iterate till convergence, intermediate iterates have a transparent interpretation and at least bounds on control inputs will be satisfied (provided the limits did not change since the previous QP problem was solved exactly).

B. *Features and Limitations*

qpOASES is originally written in C++ following an object-oriented software design. The implementation offers plenty of structure exploiting features and is tailored to QP problems comprising dense matrices as arising when eliminating all states from the MPC formulation as described in Section II-B. It can handle general semi-definite QP problems that may contain general polytopic constraints and has proven to be numerically robust in tackling ill-conditioned QP problems or detecting infeasible problem instances [10]. qpOASES has been used in many academic and industrial applications (also on embedded controller hardware) since its first release.

There are also a couple of situations where qpOASES might not be a good choice, in particular

- if the control horizon is very long favoring a sparse QP formulation (see Section II-A);
- if computations shall be run in parallel (e.g. when using multiple CPUs or a field-programmable gate array);
- if a very simple algorithmic scheme (in terms of size of source code) is mandatory.

Moreover, similar to all existing active-set methods, there are no proven polynomial upper bounds on the number of iterations needed to find an optimal solution. However, the maximum number of iterations observed in practice is almost always rather low (at most linear in the number of variables/constraints) and straight-forward to estimate beforehand by means of simulation studies. This is the case when the optimal active-set at the solution of a sequence of QPs does not change a lot. Furthermore, the online active set strategy can further mitigate this risk by using intermediate iterates if the iteration limit is hit in exceptional cases (the so-called *real-time variant* as mentioned in Section IV-A).

qpOASES is considered a viable option for the industrial case studies of this paper as none of the above-mentioned limitations turns out to be crucial. For running qpOASES on the target hardware, a dedicated embedded variant based on qpOASES 3.0 has been implemented. This variant is a library-free, ANSI C code featuring static memory management and the option to perform all computations with single precision arithmetic.

V. Implementation Aspects

The main considerations that contribute to the successful automatic deployment of the considered embedded MPC controllers are outlined in this section.

A. MPC design

Statoil's Estimation and Prediction Tool for Identification and Control (SEPTIC) [20] is used to obtain the step response models and MPC configurations used as a target specification for the embedded controllers. SEPTIC is a field-proven software that has many features that can be used to achieve high control performance in an MPC application. Besides the use of limits on manipulated variables (MVs) and controlled variables (CVs), a quadratic slack variable implementation is offered to soften constraints on CVs in order to avoid infeasible problems. In addition to weights on control targets, the priority of each control target (including constraints) can be assigned explicitly. In SEPTIC MPC, a sequence of steady-state quadratic programs are solved to respect the specified control targets with as many of the high priority control targets as possible, and the steady-state targets are used as references for the dynamic optimization problem.

In order to reduce the number of decision variables, move blocking is implemented (i.e. MVs are fixed to be constant over several time-steps), and CVs are evaluated on only specified evaluation points on the prediction horizon. SEPTIC calculates the prediction horizon for each CV from the step response models and ensures that the modeled steady-state is reached after the last MV moves. For each CV, one evaluation point is placed automatically at the end of each MV block for all MVs with a model to that CV. After the last MV block, equally distributed evaluation points are configured separately for each CV [20].

B. C code generation for embedded MPC

An ANSI C code generator is developed to use the desired MPC configuration and step response models obtained from SEPTIC (as files), and it generates MPC code suitable for embedded devices. Apart from control target priorities, all the SEPTIC design specifications translate directly into the configuration used by the code generator. As stated in the MPC formulation (1), an 'exact penalty' slack variable implementation is used, where only one slack variable per constraint is needed. This leads to a much faster MPC compared to a quadratic or 1-norm implementation where a separate slack variable for every constraint at every CV evaluation point is normally used [11].

The embedded MPC code generator produces QP solver configuration files (for generating tailored solver code) or QP data source files (for embedding a more general solver such as qpOASES) according to the formulations in sections II-A and II-B. When a sparse QP formulation is most appropriate for a target QP solver, (4) is used, if required by the solver, or if the solver relies on a cheap (fast) projection (on to the constraint set $\mathbb{Z}$) in order to boost its speed (as in the case of some first-order methods). Otherwise, (3) is used to avoid the overhead associated with any solver routine that converts (4) to (3). The QP (5) is used for dense solvers like qpOASES.

An MPC main file is also generated, containing the following function calls:

- load initial conditions,
- read measured data from plant,
- calculate unmeasured disturbances,
- update QP parameters,
- solve QP problem (call QP solver),
- recover eliminated variables (if (5) is used),
- send optimized inputs to plant,
- make a time shift (save data required for next time step).

All the supporting function code (i.e. excluding the QP solver code) is automatically generated by the embedded MPC code generator. Since the problem size and constant data are identified at code generation stage, all the loops involved are either fixed or completely unrolled, providing a flexible trade-off between computational performance and code size.

In the case of the generated QP solver code, the outcome depends on the strategies used in the particular code generator used. Since the embedded varaint of qpOASES only relies on static memory, it is important to specify the maximum number of QP variables (`NVMAX`) and constraints (`NCMAX`) to ensure that an appropriate amount data memory is allocated.

C. Offline preparations for online updates

When the dense QP formulation (5) is used (as in the case of qpOASES), the QP parameters (11b) and (11d) must be computed online. However, most of the data involved is constant, implying that offline computations/preparations can be made in order to speed up the updates online. The same applies to the recovery of eliminated variables (involving equations (6) and (7)) and the calculation of unmeasured disturbances.

Note that the condensed Hessian H_d (11a) and constraint matrix A_d (11c) are constant and still have some sparse structure. This means that some QP solver routines that completely ignore these observations may not be fully efficient. In qpOASES, such a case is identified when the calculation of the matrix-product $A_d z_i$ (where z_i is the current primal iterate) is required (see Algorithm 1 and details in [9, §3.1]). qpOASES provides a `ConstraintProduct` functionality that allows the user to provide a tailored implementation of this calculation. A customized function that efficiently evaluates the product of any constraint at a given primal iterate is therefore automatically generated by the embedded MPC code generator.

D. *Industrial embedded platform: ABB AC500 PLC*

The target platform is the ABB AC500 PM592-ETH PLC. The PLC has a Freescale$^{\text{TM}}$ G2_LE implementation of the MPC603e microprocessor that runs at 400 MHz. The MPC603e is a RISC CPU with a dedicated hardware floating point unit (FPU), which is fully IEEE 754-compliant for both double and single precision. The PLC is also equipped with 4MB RAM for user program memory and 4MB integrated user data memory.

Programming and runtime support is offered for ANSI C89 and C99 code integrated into a PLC software/runtime architecture. The MPC code is therefore implemented in the PLC as a C function block, and the resulting embedded program runs as a dedicated PLC task. A library-free MPC code is implemented since linking against external binaries is not supported.

The GNU `gcc 4.7.0` compiler toolchain is used to compile the C code part of the PLC application. The compiler settings used include `-mcpu=603e` which specifies the architecture type, register usage, instruction scheduling parameters, and allows the compiler to produce optimal code for the CPU. In addition, the compiler optimization level has been set to `-O1`, which is the maximum level that could be used on the given PLC.

E. *Single precision computations*

The computational capabilities of the AC500 PLC for optimization problems were investigated, and a detailed discussion is presented in [5]. The findings indicate that the G2_LE core is unable to attain peak performance for instructions fetch and execution per clock cycle. However, the FPU contains a single-precision multiply-add array that allows the G2_LE core to efficiently implement multiply and fused multiply-add operations, giving single-precision multiply-type instructions an upper hand over equivalent double-precision operations. It will therefore be best to use single-precision if double-precision accuracy is not required (as is the case for most MPC applications). The QP solver tuning options are carefully selected to ensure numerical robustness for single precision arithmetic.

VI. Industrial MPC applications

The following sections present two practical examples where an embedded MPC implementation is preferred. Both examples are industrial multivariable control problems with multiple control targets and constraints. The first example is a case study on Statoil's recently patented subsea compact separation unit [21], [7], and the second considers the use of an electrical submersible pump in an oil well [8], [22].

TABLE I: QP problem sizes for the subsea compact separator

			Constraints		
QP	**Solver**	**Variables**	**Equalities**	**Inequalities**	**Bounds**
Dense(5)	Active-set	24	–	96	24
Sparse(4)	First-order	82	58	60	82
Sparse(3)	Interior-point	82	58	138	–

A. *Subsea compact separation unit*

The subsea compact separation unit separates a multiphase input flow of liquid (oil/water) and gas at two stages. At the first stage, a Gas-Liquid Cylindrical Cyclone (GLCC) separates the liquid and gas coarsely. A phase splitter and a de-liquidizer are then used for finer separation at the second stage. Due to the lack of buffer volumes in the compact separation process, the dynamics are much faster, and disturbance effects are much more significant in the process, compared to most separation techniques.

The main objective is to control the quality (i.e. gas volume fraction) of fluid in the outlets of the separator (Liq_{out} and Gas_{out}). The outlets lead to a compressor and a pump, and it is essential that the gas and liquid contents are kept within their acceptable limits. It is also necessary to control the pressure in the GLCC (P_1) and the de-liquidizer (P_2) around their operational points and within their safety limits, while respecting the physical limits of all control valves. Further details of the process can be found in [21], [7].

B. *Subsea compact separator problem setup*

The embedded MPC for the subsea separator has a total of 4 CVs, each with 10 evaluation points; 1 CV (Liq_{out}) with a high bound; 1 CV (Gas_{out}) with a low bound; and 2 CVs (P_1 and P_2) have both high and low bounds. There are 6 slack variables (1 slack on each CV bound); 3 MVs (control valves), each with 6 move blocking indices, an upper limit, a lower limit, and a rate of change limit. Two measured process disturbances (DVs) are also used in the MPC setup. The required sampling frequency is 1 Hz. However, a faster sampling will enable the controller to further reduce the effect of disturbances on the controlled variables. The resulting QP problem sizes for the three solvers tested are summarized in Table I.

C. *Electric Submersible Pump*

Oil recovery and production rate can be boosted by using Electric Submersible Pumps (ESPs) to create artificial lift. The considered system consists of one oil well with an ESP and a production choke valve.

The main control priority is to ensure good operating conditions for the ESP. Specifically, the following constraints

TABLE II: QP sizes for the electrical submersible pump

			Constraints		
QP	**Solver**	**Variables**	**Equalities**	**Inequalities**	**Bounds**
Dense(5)	Active-set	17	–	78	17
Sparse(4)	First-order	60	43	58	60
Sparse(3)	Interior-point	60	43	105	–

must be respected: a high limit on the electric current (I_{high}), a low limit ($q_{0,\text{low}}$), and an upper limit ($q_{0,\text{high}}$) on the flow (q_0) delivered by the pump. These constraints form part of essential failure prevention measures, without which the lifetime of the ESP will reduce. It is also important to maintain acceptable operating conditions for the well. This entails keeping the inlet pressure within its low limit ($p_{p,in,\text{low}}$), and avoiding backflow into the well by keeping the wellhead pressure higher than the topside pressure.

Other important, but less prioritized, control targets include keeping the ESP inlet pressure at a desired setpoint ($p_{p,in,setp}$), minimizing the electric current consumption of the ESP, and avoiding high wellhead pressure relative to the topside pressure. The ESP motor frequency (f) and the production choke valve opening (z) are the control inputs, and both impose hard constraints that must be kept. Further details can be found in [8], [22]

D. *Electric submersible pump problem setup*

The ESP setup leads to an MPC problem with 4 CVs, each with 8 or 9 evaluation points. There are 2MVs, each with 5 move blocks, an upper limit, a lower limit, and a rate of change limit. There are 7 slack variables (corresponding to each CV limit), and 1 measured process disturbance (DV). A sampling rate of at least 1 Hz is required for the embedded controller. The resulting QP problem sizes are summarized in Table II.

VII. Hardware-in-the-loop simulation results

A. *Test Setup and PLC Implementations*

The test setup consists of the PLC implementation of the embedded MPC in closed-loop with a process simulator developed for the industrial process. Communication between the PLC and the simulator is achieved using Ethernet and an OPC server.

PLC implementations featuring the online active set strategy of qpOASES, the primal-dual interior-point method of CVXGEN [1], and the primal-dual first-order method of [7] (generated using FiOrdOs [2]) are compared. The reader is referred to the references stated for details about the solvers and methods used in this work.

Among existing state-of-the-art first-order methods compared in [7], the primal-dual first-order method turns out as the best choice for the MPC problem used in this work. Both FiOrdOs and CVXGEN generate solvers that use the QP formulations presented in Section II. Other state-of-the-art interior-point solvers such as HPMPC [23] and FORCES [3] require reformulation of the MPC problem (preferably in state-space), and hence more effort to achieve equivalent embedded controllers within the code generation framework used in this paper.

TABLE III: Cold-start HIL test results for 600 time steps of the compact separation process. Subscripts *dp* and *sp* denote double and single precision floating point, respectively.

QP Solver (cold-start)	**Time** (ms) average/max	**Iterations** average/max	**Mean Square Error** $P_1/P_2/Liq_{out}/Gas_{out}$
SEPTIC MPC	–	–	0.01/0.004/3.51/0.23
Active-set$_{dp}$	13.8/18.8	19/27	0.04/0.008/2.73/0.30
Active-set$_{sp}$	10.9/15.2	20/27	0.04/0.008/2.58/0.32
First-order$_{dp}$	114.6/116.4	785/785	0.04/0.008/2.56/0.31
First-order$_{sp}$	102.9/104.7	785/785	0.04/0.008/2.20/0.33
Interior-point$_{dp}$	72.2/84.9	15/18	0.04/0.008/2.68/0.31
Interior-point$_{sp}$	63.8/65.4	18/18	0.04/0.008/2.35/0.31

The differences between the original PC-based SEPTIC MPC implementation and the embedded variants produced using the code generation framework are discussed in [7], [8], [6]. In this paper, the main focus is to compare the embedded implementations, and since control performance is essential in the industrial applications, the best (closed-loop) control performance for each solver method (compared to the target SEPTIC MPC performance) is used as the basis for result comparison. The computed control input is therefore applied immediately it is available from the solver.

B. *QP solver settings for embedded MPC tests*

Several parameter tuning options are available in qpOASES for the online active set strategy. However, the MPC option set of qpOASES provides a parameter collection adequate for most MPC applications. A numerically robust variant of the MPC option set is used for the tests, where `initialStatusBounds` = `ST_LOWER`, `regularizationSteps`= 1, and a `terminationTolerance` $= 10^{-7}$ are used. The maximum number of iterations can also be specified by stating the number of working set recalculations the solver can use. The parameter settings for the primal-dual interior-point method are: `duality_gap` $= 10^{-4}$, `residual_tol` $= 10^{-6}$, `KKT_regularization` $= 10^{-7}$, and `refine_steps` $=$ 1. The only tuning parameters for the primal-dual first-order method are the maximum number of iterations and the primal/dual step size balancing parameter $\eta = 10$.

C. *Results for subsea compact separator*

The hardware-in-the-loop (HIL) test results are shown in tables III and IV, where the same hydrodynamic slugging flow sequence as in [7] is used. The Mean Square Error values are used as the control performance measure.

The priority-based approach of SEPTIC MPC yields a high control performance target for the embedded controllers. Nevertheless, the performance obtained with the embedded controllers are close to the original (PC-based) SEPTIC MPC performance (and well within acceptable performance requirements of the compact separator).

TABLE IV: Warm-start HIL test results for 600 time steps of the compact separation process. Subscripts *dp* and *sp* denote double and single precision floating point, respectively.

QP Solver (warm-start)	**Time** (ms) average/max	**Iterations** average/max	**Mean Square Error** $P_1/P_2/Liq_{out}/Gas_{out}$
SEPTIC MPC	–	–	0.01/0.004/3.51/0.23
Active-set$_{dp}$	4.9/13.1	1/19	0.04/0.008/3.01/0.27
Active-set$_{sp}$	3.4/9.3	1/19	0.04/0.008/2.73/0.31
First-order$_{dp}$	18.2/19.8	100/100	0.02/0.003/2.81/0.28
First-order$_{sp}$	15.3/16.9	100/100	0.02/0.003/2.96/0.31

The MPC based on the online active set strategy of qpOASES yields the fastest embedded MPC controller for the subsea compact separator. When using cold-start, it is about 6 (or 9) times faster than the first-order controller in worst-case (or average), and about 4 (or 6) times faster than the interior-point controller. The warm-start speed-up of the first-order controller is remarkable, but it is still outperformed by the active-set controller. Warm-starting is not supported by the CVXGEN interior-point code, and no further speed-up is reported in this case.

Note that due to the relatively higher accuracy of solution achieved by the second-order methods, the performance of the first-order method approaches that of the second-order methods when the iterations limit of the first-order method is increased. This means that the computational time will increase for the first-order controller if the Mean Square Error values of the active-set controller is the target performance. However, the Mean Square Errors of the first-order controller are much closer to the desired control performance of the original SEPTIC MPC. In this case, the warm-started variant of the first-order controller can be considered as a viable alternative to the fast active-set controller.

The single-precision results show that the interior-point controller attains the most speed-up, where the worst-case cost per iteration reduces by 1.09 ms, followed by the active-set controller with a 0.14 ms reduction, and finally a 0.02 ms reduction in the already cheap first-order iterations. This observation reflects the number of multiply-type operations involved in each approach and the corresponding reduction in the amount memory instructions attained by switching to single-precision. The results suggest that if the compiler is able to attain a good level of single-precision code optimization, the most expensive iterations will gain much from using single-precision arithmetic. Nevertheless, the small reduction achieved on a large number of cheap iterations (as in the case of the first-order controller) can also lead to a total speed-boost for the embedded controller.

The code size (shown in table V) of the FiOrdOs generated first-order implementation is the smallest, and the size of the qpOASES based MPC code is also small compared to that of CVXGEN (which uses massive loop unrolling).

D. Results for electric submersible pump

Since the results in section VII-C are mainly in favor of the active-set and first-order implementations, the results

TABLE V: Program and data sizes for the subsea compact separation application. Double and single precision floating point (FP) codes are denoted by *dp* and *sp*, respectively.

	C/PLC code size (MB)			**Data size** (MB)		
FP	qpOASES	FiOrdOs	CVXGEN	qpOASES	FiOrdOs	CVXGEN
dp	0.64/1.59	0.56/1.35	0.96/2.16	0.29	0.22	0.20
sp	0.63/1.55	0.54/1.33	0.92/2.14	0.16	0.12	0.11

TABLE VI: Cold-start HIL test results for 10 minutes of the electric submersible pump setup. Subscripts *dp* and *sp* denote double and single precision floating point, respectively.

QP Solver (cold-start)	**Time** (ms) average/max	**Iterations** average/max	**Mean Square Error** $I_{\text{high}}/q_{0,\text{low}}/p_{p,in,\text{low}}/p_{p,in,\text{setp}}$
SEPTIC MPC	–	–	0.0006/0.153/3.667/9.647
Active-set$_{dp}$	8.6/15.6	19/31	0.0001/0.153/3.689/8.949
Active-set$_{sp}$	8.5/15.3	20/31	0.0017/0.195/3.228/8.329
First-order$_{dp}$	262.0/263.3	305/305	0.0007/0.161/3.409/8.033
First-order$_{sp}$	128.7/178.1	305/305	0.0007/0.161/3.409/8.033

of both approaches are further compared for the electrical submersible pump application. Both methods also support warm-starting, and the best performance from both implementations can be directly compared. The test results for the embedded controllers using the benchmark scenario defined in [8] are shown in tables VI and VII.

Control performance close to the (PC-based) SEPTIC MPC is achieved for both embedded controllers. Comparing the computational time achieved, the active-set embedded controller outperforms the first-order controller by at least one order of magnitude. Clearly, using a condensed QP formulation for a smaller problem size favors qpOASES even more than in the first case study. In fact, a closer look at the cost per iteration reveals that, for this case, the active-set iterations become as cheap as the first-order iterations.

The results of the single-precision tests and the code/data sizes (shown in table VIII) follow the same observations made for the compact separation unit. Even though the first-order implementation attained good single-precision code optimization for its many iterations, the remarkably cheap iterations of the online active set strategy results in superior performance for the ESP application.

VIII. Conclusions

This paper discussed the development of high-performance embedded MPC controllers. The embedded controllers are based on linear models and design configurations obtained from the field-proven MPC software package SEPTIC. The embedded MPC is automatically deployed through the use of code generation techniques that ensure desired high control performance, and significantly reduce the effort required for the implementation of the embedded MPC. The implementation aspects are also discussed to shed light on the main considerations that contribute to the successful automatic deployment of embedded MPC.

TABLE VII: Warm-start HIL test results for 10 minutes of the electric submersible pump setup. Subscripts *dp* and *sp* denote double and single precision floating point, respectively.

QP Solver (warm-start)	**Time** (ms) average/max	**Iterations** average/max	**Mean Square Error** $I_{\text{high}}/q_{0,\text{low}}/p_{p,in,\text{low}}/p_{p,in,\text{setp}}$
SEPTIC MPC	–	–	0.0006/0.153/3.667/9.647
Active-set$_{dp}$	2.0/7.3	2/17	0.0001/0.153/3.689/8.949
Active-set$_{sp}$	1.6/5.7	2/16	0.0001/0.154/3.683/8.946
First-order$_{dp}$	85.8/87.1	100/100	0.0002/0.154/4.132/10.513
First-order$_{sp}$	58.5/59.8	100/100	0.0002/0.154/4.132/10.516

TABLE VIII: Program and data sizes for the electric submersible pump application. FP represents floating point.

FP	**C/PLC code size** (MB)		**Data size** (MB)	
Precision	qpOASES	FiOrdOs	qpOASES	FiOrdOs
double	0.16/0.37	0.07/0.16	0.11	0.08
single	0.16/0.36	0.07/0.16	0.07	0.05

High-speed QP solvers form the core of the embedded controllers, and much work is made to uncover the performance capabilities of the online active set strategy, as implemented in the qpOASES software package. A comparison study is done through hardware-in-the-loop simulation tests for two challenging industrial applications using an ABB AC500 PLC. The results indicate that the online active set strategy displays superior performance compared to both a tailored interior-point method and a primal-dual first-order method. Although the active-set embedded controller code is neither the simplest nor the lightest, the code sizes and memory footprints produced are low and adequate for the industrial applications studied.

The high performance obtained using the online active set strategy enables even higher sampling rates to be achieved for the considered industrial applications: 100 Hz in worst-case, and the possibility to increase to 300 Hz for the compact separation unit and 500 Hz for the electric submersible pump application. Although 1 Hz is the required sampling rate for both industrial applications, a faster sampling may enable the controllers to further reduce the effect of disturbances on the controlled variables. Moreover, achieving an MPC computational time which is much lower than the required sampling time makes it possible for other essential tasks (e.g. controller reconfiguration and fault-tolerance tasks) to be incorporated into the embedded controller.

ACKNOWLEDGMENT

We are grateful to Gisle Otto Eikrem, Alexey Pavlov, and their colleagues at Statoil for making the industrial application models, SEPTIC MPC tools, and other useful resources available for this study.

REFERENCES

[1] J. Mattingley and S. Boyd, "CVXGEN: A Code Generator for Embedded Convex Optimization," *Optimization and Engineering*, vol. 13, no. 1, pp. 1–27, 2012.

[2] F. Ullmann and S. Richter. (2013–2014) FiOrdOs: Code generation for first-order methods. [Online]. Available: fiordos.ethz.ch

[3] A. Domahidi, A. Zgraggen, M. Zeilinger, M. Morari, and C. Jones, "Efficient Interior Point Methods for Multistage Problems Arising in Receding Horizon Control," in *IEEE Conference on Decision and Control*, Maui, HI, USA, Dec. 2012, pp. 668 – 674.

[4] P. Zometa, M. Kögel, and R. Findeisen, "μAO-MPC: A Free Code Generation Tool for Embedded Real-Time Linear Model Predictive Control," in *2013 American Control Conference (ACC)*. Washington, DC, USA: AACC, June 2013, pp. 5340–5345.

[5] G. Frison, D. K. M. Kufoalor, L. Imsland, and J. B. Jørgensen, "Efficient Implementation of Solvers for Linear Model Predictive Control on Embedded Devices," in *The 2014 IEEE Multi-Conference on Systems and Control*, Antibes/Nice, France, 2014.

[6] V. Aaker, "Embedded MPC for a Subsea Separation Process," Master's thesis, Department of Engineering Cybernetics, Norwegian University of Science and Technology (NTNU), June 2012.

[7] D. K. M. Kufoalor, S. Richter, L. Imsland, T. A. Johansen, M. Morari, and G. O. Eikrem, "Embedded model predictive control on a plc using a primal-dual first-order method for a subsea separation process," in *22nd IEEE Mediterranean Conference on Control and Automation*, Palermo, Italy, 2014.

[8] B. J. T. Binder, D. K. M. Kufoalor, A. Pavlov, and T. A. Johansen, "Embedded Model Predictive Control for an Electric Submersible Pump on a Programmable Logic Controller," in *The 2014 IEEE Multi-Conference on Systems and Control*, Antibes/Nice, France, 2014.

[9] H. J. Ferreau, H. G. Bock, and M. Diehl, "An online active set strategy to overcome the limitations of explicit MPC," *International Journal of Robust Nonlinear Control*, vol. 18, pp. 816–830, July 2008.

[10] H. Ferreau, C. Kirches, A. Potschka, H. Bock, and M. Diehl, "qpOASES: A parametric active-set algorithm for quadratic programming," *Mathematical Programming Computation*, vol. 6, no. 4, pp. 327–363, 2014.

[11] J. M. Maciejowski, *Predictive Control: with constraints*. Pearson and Prentice Hall, 2002.

[12] 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, 2002.

[13] S. Richter, "Computational complexity certification of gradient methods for real-time model predictive control," Ph.D. dissertation, Diss., Eidgenössische Technische Hochschule ETH Zürich, Nr. 20718, 2012.

[14] P. Patrinos and A. Bemporad, "An accelerated dual gradient-projection algorithm for embedded linear model predictive control," *IEEE Transactions on Automatic Control*, vol. 59, no. 1, pp. 18–33, 2014.

[15] J. Jerez, P. Goulart, S. Richter, G. Constantinides, E. Kerrigan, and M. Morari, "Embedded Predictive Control on an FPGA using the Fast Gradient Method," in *European Control Conference*, Zurich, Switzerland, 2013, pp. 3614 – 3620.

[16] A. Shahzad and P. J. Goulart, "A new hot-start interior-point method for model predictive control," in *Proceedings of the 18th IFAC World Congress Milano (Italy)*, vol. 18, no. 1, 2011.

[17] C. M. Maes, "A regularized active-set method for sparse convex quadratic optimization," PhD thesis, Stanford University, Nov. 2010.

[18] H. Ferreau, A. Kozma, and M. Diehl, "A parallel active-set strategy to solve sparse parametric quadratic programs arising in MPC," *Proc. of the 4th IFAC nonlinear model predictive control conference*, 2012.

[19] H. Ferreau, A. Potschka, and C. Kirches. (2007–2014) qpOASES. [Online]. Available: http://www.qpOASES.org/

[20] S. Strand and J. Sagli, "MPC in Statoil – advantages with in-house technology," in *International Symposium on Advanced Control of Chemical Processes (ADCHEM)*, Hong Kong, 2003, pp. 97–103.

[21] J. Høydal, O. Kristiansen, G. O. Eikrem, and K. Fjalestad, "Method and system for fluid separation with an integrated control system," Patent WO2 013 091 719 A1, 06 27, 2013. [Online]. Available: http://www.google.com/patents/WO2013091719A1?cl=en&hl=no

[22] A. Pavlov, D. Krishnamoorthy, K. Fjalestad, E. Aske, and M. Fredriksen, "Modelling and model predictive control of oil wells with electric submersible pumps," in *The 2014 IEEE Multi-Conference on Systems and Control*, Antibes/Nice, France, 2014.

[23] G. Frison, H. Srensen, B. Dammann, and J. Jørgensen, "High-performance small-scale solvers for linear Model Predictive Control," in *2014 European Control Conference (ECC)*, June 2014, pp. 128–133.