Scalability of QP solvers for Embedded Model Predictive Control Applied to a Subsea Petroleum Production System*

B. J. T. Binder[1], D. K. M. Kufoalor[1], T. A. Johansen[1,2]

Abstract— **The performance of two different Quadratic Programming (QP) solvers for embedded Model Predictive Control (MPC), FiOrdOs and qpOASES, is evaluated for a relevant case study from the petroleum industry. Embedded MPC for the considered system is implemented on a PLC (Programmable Logic Controller) using both solvers. The focus is on the computation time and memory requirements of the solvers as the dimensions of the control problem increase. The results show that qpOASES has a superior performance for small systems with respect to computation time. However, qpOASES has a near cubic growth in computation time with respect to the number of system variables, while FiOrdOs only has a near linear growth. FiOrdOs may thus be faster for larger systems. FiOrdOs has a smaller memory footprint than qpOASES for small systems; however, the program size grows faster than with qpOASES, and for the largest system configuration, the program sizes were almost identical. For even larger systems, qpOASES may have smaller program memory requirements than FiOrdOs, though qpOASES requires more data memory for all problem sizes.**

I. INTRODUCTION

Model Predictive Control (MPC) is an optimization-based control strategy especially acknowledged for it's capability of handling multivariable control problems with constraints. MPC is found in a wide range of application areas, including power plants, petroleum refineries, chemicals, food processing, automotive, and aerospace applications, with an increasing number of reported applications [1].

MPC is typically implemented in a PC/server-based environment [1], [2], but there is a growing interest for embedded implementation of MPC. This requires MPC to be implemented on hardware such as microcontrollers, Field Programmable Gate Arrays (FPGAs) or Programmable Logic Controllers (PLCs). A main criticism of MPC has often been that the control method is computationally very demanding, as it is based on repeatedly solving a numerical optimization problem on-line (in real-time). This is a challenge given the limited computational resources typically found in such hardware.

PLCs are commonly used in the industry due to their reliability and robustness properties, and several studies considering implementation of MPC on a PLC have recently been conducted. Implementation aspects of MPC on a PLC were discussed in [3] and [4]. Feasibility of embedded MPC on a PLC was investigated in [5] and [6], with realistic case studies from two different applications within subsea petroleum production; an Electric Submersible Pump (ESP) and a subsea separation unit.

Significant progress has recently been made in the area of embedded MPC. Important contributions include exploiting the MPC problem structure and computational architectures, such as in [7], [8]. This has resulted in a variety of software tools targeting embedded platforms, with efficient and portable software implementations of different Quadratic Programming (QP) solver algorithms, such as FORCES [8], qpOASES [9], FiOrdOs [10], CVXGEN [11], HPMPC [12] and μAO-MPC [13].

Solver scalability tests are performed in most papers where new implementations of QP solver algorithms are presented. However, most of these tests compare similar solver methods (e.g. only first order methods, active set methods or interior point methods). Nevertheless, scalability tests for different solver methods are performed in [14], where all solver algorithms are tested with cold-start. A recent comparison study of different QP solver algorithms for embedded MPC is presented in [15], where the need for proper benchmarking of QP solvers is emphasized, and preliminary benchmarking results are presented based on different performance metrics.

Three different QP solvers (FiOrdOs, qpOASES and CVXGEN) were compared in [16], using the two industrial case studies from [5] and [6]. In this paper, the scalability of two of these solvers (FiOrdOs and qpOASES) is studied, i.e. how the solvers perform in terms of memory requirements and computation time as the problem size increases. The need for a scalable QP solver is motivated by a relevant case study from the petroleum industry; a subsea petroleum production system consisting of a subsea manifold and production wells where ESPs are installed for artificial lift. From a control perspective this is a very scalable system, as one may choose how many wells to include in the controller. Up to four wells are considered in this paper, though up to about 20 wells may produce to a single production manifold in a real system. The focus in this paper is on the performance of the different QP solvers for varying problem sizes. Control of ESPs using MPC was considered in [17] and [6], where the control problem itself is discussed more extensively.

This paper is organized as follows: the considered QP solvers are described in Section II, the case study is presented in Section III, the controller implementation process is outlined in Section IV, results from hardware-in-the-loop simulations are presented and discussed in Section V, and concluding remarks are given in Section VI.

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

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

*This work was supported by the Research Council of Norway and Statoil.

II. EMBEDDED QP SOLVERS

Many code generators are now available to generate portable source code for fast and efficient QP-solvers with modest memory requirements, making embedded implementations practically feasible. Three such solvers were compared in [16]; FiOrdOs, qpOASES and CVXGEN. These solvers are based on three different methods; FiOrdOs on a first order method, qpOASES on an active set method, and CVXGEN on an interior-point method. The latter two are second order methods when classified by the highest order of the information used to solve the QP.

Only FiOrdOs and qpOASES are extensively tested in this paper. As stated in [11], CVXGEN targets small problems, and indeed, CVXGEN was not able to generate a solver for the larger system configurations considered in our case study[1], and was thus not further considered.

FiOrdOs and qpOASES are briefly described in this section.

A. FiOrdOs

FiOrdOs[2], which is implemented as a MATLAB toolbox, generates custom solvers for parametric QP problems, implemented in ANSI C code. The generated solvers are based on first-order methods, which means that only first-order derivatives (gradients) are used to find the solution. In this paper FiOrdOs is used to generate a custom QP solver based on the primal-dual first-order method proposed in [18], which was also used in [6] and [16]. The QP solver is generated based on the sparse formulation of the MPC problem, cf. [16].

Compared to second-order methods, first-order methods perform many, but cheap, iterations when searching for the solution of the QP. In our implementation, the solver runs for a fixed number of iterations. The number of necessary iterations is found empirically, based on hardware-in-the-loop simulations. FiOrdOs can easily be warm-started, to provide an acceptable controller performance with relatively few iterations, which is done in our implementation.

B. qpOASES

qpOASES[3] is an open-source implementation of the Online Active SEt Strategy [9], [19]. This is a primal-dual active set method that introduces a homotopy path from the solution of the previous QP problem to the current one, while the set of active constraints at the optimal solution is identified and updated online. qpOASES is originally written in C++, but a new ANSI C embedded variant is used in this paper, as in [16]. Unlike the solver generated by FiOrdOs, the QP solver code in this case is not tailored to a specific QP. A condensed formulation of the MPC problem (as described in [16]) is used with qpOASES.

[1]CVXGEN was only able to generate a solver for configurations 1 and 2 as described in Section V-B. Extensive results using CVXGEN for problems of small size (comparable to configurations 1 and 2) were presented in [16].

[2]See http://fiordos.ethz.ch

[3]See http://www.qpOASES.org

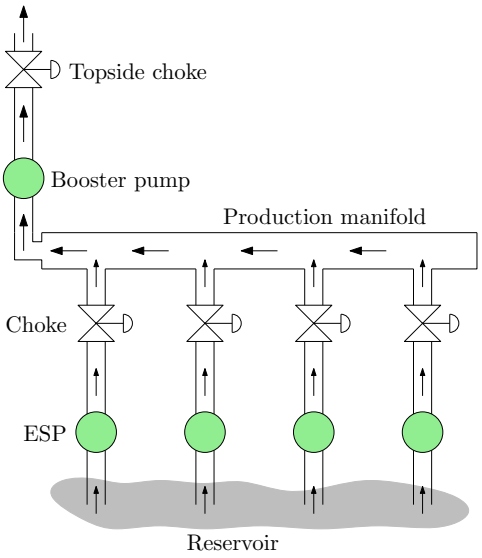

Fig. 1. Subsea petroleum production system

qpOASES performs fewer, but more expensive, iterations compared to the first-order method implemented with FiOrdOs. qpOASES also has strong warm-start capabilities, and efficient solution of the QP problem relies on data from the solution of the previous QP problem. Hardware-in-the-loop simulations show that the number of iterations required to solve the QP problem is decreased dramatically when warm-starting the solver, often by an order of magnitude.

In our implementation, the qpOASES solver is run until it converges to the optimal solution with machine precision, without any bounds on the number of iterations. In a real-time application, an upper bound on the number of iterations can be defined, and an intermediate solution can be implemented as the control input if the bound is reached. Intermediate iterates of the solver have a transparent interpretation, and at least bounds on control inputs will be satisfied (provided these bounds have not changed since the last exact solution of the QP).

III. CASE STUDY

A. The system

The case study considered in this paper is a realistic subsea petroleum production system consisting of (up to) four petroleum production wells producing to a subsea production manifold, and the riser taking the produced fluid to the surface facilities. Electric Submersible Pumps (ESPs) are installed in the wells for artificial lift, as well as a booster pump in the riser. The system is shown in Fig. 1.

ESP is a widely used technique to create artificial lift in petroleum production systems. The ESP is installed inside the well with the purpose of reducing the hydrostatic pressure from the fluid in the well. This provides a boosted production from the well and possibly an increased recovery rate from the reservoir.

B. The control problem

Many factors may cause failure, or contribute to a reduced life-time, of ESP installations, including excessive levels of e.g. thrust forces, vibrations, or electric power consumption of the ESP motor. Failure of the ESP has a huge economic impact, due to both the production loss, and the costs of replacing the equipment. Maintaining good operating conditions for the ESP is thus a main priority in such installations, in addition to production optimization. This highly motivates the use of automatic control, as discussed in [17].

For the considered system, the main control targets are as follows:

- Avoid electric motor burnout by limiting the power consumption of each motor
- Avoid pump failure due to excessive thrust forces by imposing limits on the flow rate in each well and the riser
- Ensure acceptable pressure conditions for each well and the manifold
- Avoid backflow in the system by maintaining a sufficient pressure drop across each choke valve
- Limit (and minimize) the overall electric power consumption of the system
- Optimize the production by maintaining a desired pressure at the inlet of each ESP

Thus, the controlled variables in the system are the current consumption of each electric motor, the flow rate in each well and the riser, the ESP inlet pressure in each well, the manifold pressure, the pressure difference across each choke valve, as well as the total current consumption of the system. The control inputs in the system are the pump speed of each ESP and the booster pump, the production choke opening for each well, and the topside choke opening.

The focus in this paper is on the performance of the different QP solvers for varying problem sizes rather than the control problem itself. The reader is referred to [17] and [6] for a more extensive discussion of the control problem and controller implementation.

C. Simulator

A simulator for the considered system is implemented in MATLAB. The simulator is based on the model of an ESP-lifted well presented in [6], which is based on a model developed by Statoil in [17], [20]. The system model has been augmented to include a model of the production manifold, the booster pump and the riser.

IV. EMBEDDED MPC IMPLEMENTATION

The process of implementing MPC on a PLC using FiOrdOs was described in [6]. In this paper qpOASES is also implemented, but the main steps of the implementation process remain unchanged. As illustrated in Fig. 2, the main steps are:

- Design and configure the controller using SEPTIC
- Generate QP formulation
- Generate solver code for the QP formulation

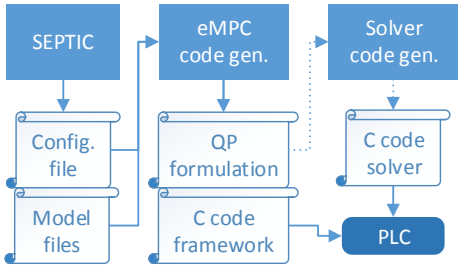

Fig. 2. Implementation process

- Implement on PLC

These steps are further discussed below.

A. SEPTIC

The MPC controller is first designed and configured using SEPTIC (Statoil Estimation and Prediction Tool for Identification and Control), which is Statoil's in-house software tool for MPC [21]. SEPTIC has many built-in features to ease the process of developing and configuring the controller to obtain the desired performance. The model used for prediction in the controller is generated by SEPTIC. Steps in the inputs and disturbances are applied to the simulator, and the response is recorded by SEPTIC. Linear SISO (single input, single output) step response models are then generated for each input/output pair. These models are combined in the controller based on the superposition principle to predict the multivariable system dynamics [21]. Once the models are obtained, the controller is configured and tuned.

B. Generating the QP formulation

The SEPTIC controller is translated into a sparse QP formulation using an automatic code generator developed specifically for this purpose in [22], referred to as the eMPC code generator in the sequel. The eMPC code generator also provides C code files that implement a general framework for embedded MPC implementation.

C. Generating the solver code

An extension of the eMPC code generator has been implemented in MATLAB to obtain the required QP formulations for both FiOrdOs and qpOASES. The sparse QP formulation is slightly reformulated to obtain the desired formulation for FiOrdOs, which is used to generate a custom C code solver for the QP. The sparse QP formulation is also translated into the condensed QP formulation that is used with qpOASES (cf. [16]). As much of the problem data is constant, some offline preprocessing is made to speed up the online updates. The condensed QP formulation, including constant problem data and functions to efficiently update problem data online, is then generated in C code, and later combined with the general qpOASES solver.

D. PLC implementation

The target hardware platform is the PLC model ABB AC500 PM592-ETH, which has support for ANSI C89 and C99, and significant computational power. The software

TABLE I

SYSTEM CONFIGURATIONS

#	Description	MV	CV	DV	Sum
1	1 ESP	2	5	1	8
2	BP and 1 ESP	4	9	1	14
3	BP and 2 ESPs	6	13	1	20
4	BP and 3 ESPs	8	17	1	26
5	BP and 4 ESPs	10	21	1	32

package ABB PS501 Control Builder Plus (version 2.3.0), which is based on the CoDeSys automation platform technology, is used to integrate the MPC controller (C code) into the PLC software/runtime environment.

In the PLC implementation, the C code QP solver is combined with the general functionality provided by the eMPC code generator, as well as functions and data from the solver code generator. Additional functionality, such as warm-starting the solver, and OPC communication via Ethernet, is also implemented.

V. HARDWARE-IN-THE-LOOP SIMULATION RESULTS

The hardware and simulator setup is described in this section, and results from hardware-in-the-loop simulations are presented and discussed.

A. Hardware setup

The ESP simulator is implemented in MATLAB and is run on a Windows PC. The embedded controller is implemented and run on the PLC. Communication between the simulator and the PLC is established via Ethernet and an OPC server.

B. Controller configurations and problem sizes

In order to test the QP-solvers for different problem sizes, five different controller configurations are developed for the system described in Section III. The configurations range from controlling only a single ESP to controlling the entire system with four ESPs and the booster pump (BP). The different configurations and the number of input variables (manipulated variables, MVs), output variables (controlled variables, CVs) and disturbance variables (DVs) in each configuration are shown in Table I.

The number of optimization variables and constraints for each controller configuration using FiOrdOs and qpOASES is given in Table II. Note that FiOrdOs solves the sparse formulation of the MPC problem, while qpOASES solves the condensed formulation, as described in [16]. The condensed formulation has a lot fewer optimization variables, at the expense of less structured (dense) matrices in the QP.

C. Main results

For each of the configurations in Table I, embedded MPC is implemented on the PLC using both FiOrdOs and qpOASES, following the implementation process described in Section IV. Hardware-in-the-loop simulations are performed for each implementation, and the performance in terms of solver run-time, number of solver iterations, and memory usage is recorded.

TABLE II

QP OPTIMIZATION VARIABLES AND CONSTRAINTS

Pumps	QP solver	Variables	Constraints	
			Inequalities	Equalities
1	FiOrdOs	72	61	55
	qpOASES	17	83	-
2	FiOrdOs	133	108	101
	qpOASES	32	148	-
3	FiOrdOs	195	162	147
	qpOASES	48	222	-
4	FiOrdOs	257	216	193
	qpOASES	64	296	-
5	FiOrdOs	319	270	239
	qpOASES	80	370	-

TABLE III

SOLVER PERFORMANCE

(N=NUMBER OF PUMPS, AVG=AVERAGE, WC=WORST CASE)

N	QP solver	Time [ms]		Iterations		Avg. time/ iter. [μs]	Memory [kB]	
		avg	wc	avg	wc		Progr.	Data
1	FiOrdOs	6	7	100	100	63	188	47
	qpOASES	3	12	3	17	818	372	119
2	FiOrdOs	39	40	100	100	393	487	117
	qpOASES	9	73	10	60	2560	641	373
3	FiOrdOs	81	81	100	100	808	944	214
	qpOASES	23	254	8	191	5260	1060	750
4	FiOrdOs	131	133	100	100	1312	1568	338
	qpOASES	52	305	11	88	8850	1639	1241
5	FiOrdOs	192	194	100	100	1923	2344	489
	qpOASES	119	534	19	107	13640	2360	1853

Table III shows the average and worst-case computation time, the average and worst-case number of solver iterations, the average computation time per solver iteration, and the memory usage on the PLC for both program and data memory, for each solver and for each controller configuration.

It should be noted that the current release of FiOrdOs does not provide any functionality for termination conditions in order to solve all problems with the same accuracy. Thus, while qpOASES is run till full convergence of the solver with machine precision, FiOrdOs is run for a fixed number of solver iterations. The number of iterations is set to 100 for all configurations in our case study, which was shown in [6] to provide sufficient performance for one ESP. This may, however, be insufficient for configuration 2-5, as FiOrdOs may be prematurely terminated in order to achieve a performance comparable to qpOASES. Thus, one may expect qpOASES to provide a better controller performance than FiOrdOs for configurations 2-5 with this setup, and the computation times in Table III are thus not directly comparable.

The average computation times required by qpOASES to achieve full convergence, and for FiOrdOs to perform

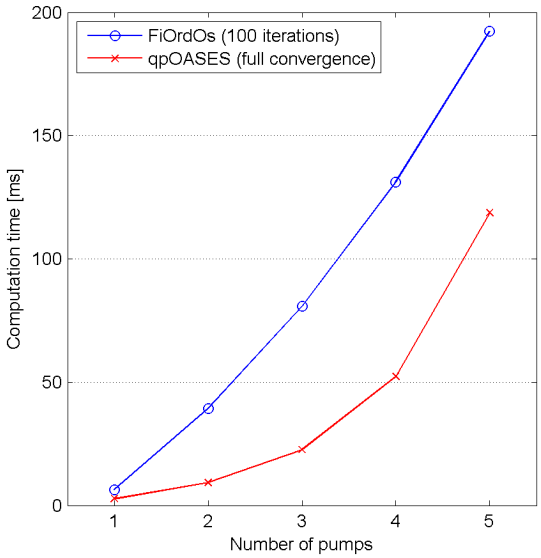

Fig. 3. Average computation time

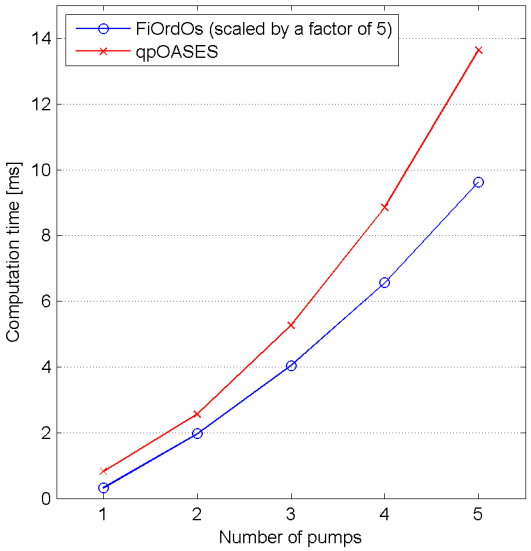

Fig. 5. Average computation time per solver iteration

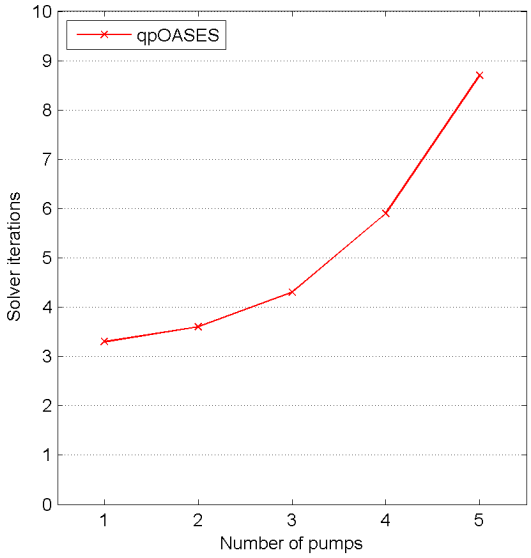

Fig. 4. Average solver iterations, qpOASES

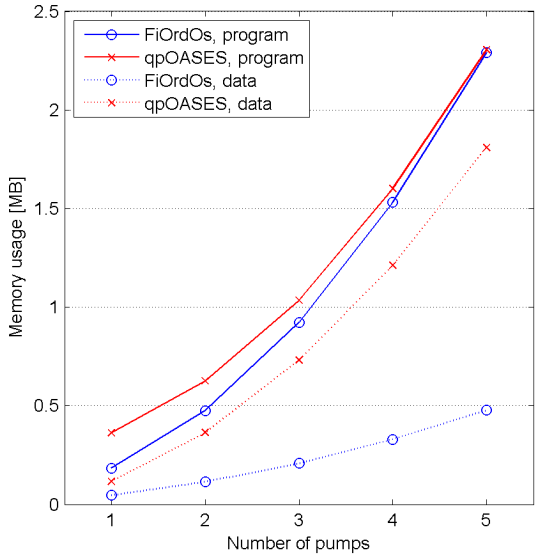

Fig. 6. Memory usage

100 iterations, are shown in Fig. 3. The average number of iterations performed by qpOASES is shown in Fig. 4. The average computation time per iteration for each solver is shown in Fig. 5, where the computations times for FiOrdOs is scaled by a factor of 5 for easier comparison. The memory usage on the PLC for each solver is shown in Fig. 6.

D. Computation time

As shown in Table III and Fig. 3, qpOASES exhibits a superior performance with respect to average computation time for all considered system sizes, even with a modest 100 iterations performed by FiOrdOs.

As seen in Fig. 5, the time required per iteration using FiOrdOs shows a growth between linear and quadratic with respect to problem size, while qpOASES shows a near quadratic growth. This is consistent with the typical asymptotic complexity of first order methods, while solution methods for the condensed problem typically have a cubic growth.

As seen in Fig. 4, the number of iterations performed by qpOASES shows a near cubic growth in problem size, and also the resulting computation time in Fig. 3 shows a near cubic growth, while the computation time using FiOrdOs (with a fixed number of iterations) only shows a growth between linear and quadratic. Depending on the number of iterations required by FiOrdOs, FiOrdOs may thus provide

a superior performance for even larger problems, though the development for larger problems is uncertain.

E. Memory usage

As seen in Fig. 6, FiOrdOs has a lot smaller memory requirement than qpOASES for small problems. For larger problems, the required program memory is quite comparable, with a close to linear growth, though the program memory usage seems to grow a little faster with FiOrdOs than with qpOASES. qpOASES requires more data memory than FiOrdOs for all problem sizes.

VI. CONCLUSION

The results in this paper show that CVXGEN is only suitable for relatively small systems, while both qpOASES and FiOrdOs were able to solve the QP problem for the largest system considered in this paper. The results also show that both qpOASES and FiOrdOs have both advantages and disadvantages for different control problem sizes.

Compared to FiOrdOs, qpOASES has a superior performance with respect to computation time for the considered system sizes, but shows a near cubic growth in computation time with respect to the number of system variables. FiOrdOs requires more computation time for the considered problems, but shows a growth between linear and quadratic with respect to the number of variables with a fixed number of iterations. Depending on the required number of iterations, FiOrdOs may thus be faster for larger systems, though the development for larger systems is uncertain.

FiOrdOs has a smaller memory footprint than qpOASES for small systems; however, the program size grows faster than with qpOASES, and for the largest system considered in this paper, the program sizes are almost identical. For even larger systems, qpOASES may have smaller program memory requirements than FiOrdOs, though qpOASES requires more data memory for all problem sizes. These results are consistent with the results reported in [16].

ACKNOWLEDGMENTS

We thank Alexey Pavlov, Gisle Otto Eikrem and Morten Fredriksen at Statoil, for providing us with system data, SEPTIC and the eMPC code generator, and for useful discussions and support regarding implementation aspects of the simulator and controller.

We also thank Hans Joachim Ferreau at ABB for making the C code version of qpOASES available for our research before it was made publically available.

REFERENCES

[1] S. J. Qin and T. A. Badgwell, "A survey of industrial model predictive control technology," *Control engineering practice*, vol. 11, no. 7, pp. 733–764, 2003.

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

[3] G. Valencia-Palomo and J. A. Rossiter, "Efficient suboptimal parametric solutions to predictive control for PLC applications," *Control Engineering Practice*, vol. 19, no. 7, pp. 732–743, 2011.

[4] B. Huyck, H. J. Ferreau, M. Diehl, J. D. Brabanter, J. F. M. V. Impe, B. D. Moor, and F. Logist, "Towards Online Model Predictive Control on a Programmable Logic Controller: Practical Considerations," *Mathematical Problems in Engineering*, vol. 2012, pp. 1–20, 2012.

[5] 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 *Proc. 22nd IEEE Mediterranean Conf. Control and Automation (MED 2014)*, Palermo, Italy, 2014.

[6] 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 *2014 IEEE Conference on Control Applications (CCA)*, 2014.

[7] Y. Wang and S. Boyd, "Fast model predictive control using online optimization," *IEEE Transactions on Control Systems Technology*, vol. 18, no. 2, pp. 267–278, 2010.

[8] 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.

[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, no. 8, pp. 816–830, 2008.

[10] F. Ullmann, "A Matlab toolbox for C-code generation for first order methods," Master's thesis, Eidgenössische Technische Hochschule Zürich, 2011.

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

[12] G. Frison, H. Sørensen, B. Dammann, and J. Jørgensen, "High-performance small-scale solvers for linear model predictive control," in *2014 European Control Conference (ECC)*, June 2014.

[13] 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, June 2013, pp. 5320–5325.

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

[15] D. Kouzoupis, A. Zanelli, H. Peyrl, and H. J. Ferreau, "Towards proper assessment of QP algorithms for embedded model predictive control," in *2015 European Control Conference (ECC)*, Linz, Austria, July 2015.

[16] D. K. M. Kufoalor, B. J. T. Binder, H. J. Ferreau, L. Imsland, T. A. Johansen, and M. Diehl, "Automatic deployment of industrial embedded model predictive control using qpOASES," in *2015 European Control Conference (ECC)*, Linz, Austria, July 2015.

[17] A. Pavlov, D. Krishnamoorthy, K. Fjalestad, E. Aske, and M. Fredriksen, "Modelling and model predictive control of oil wells with electric submersible pumps," in *Proc. 2014 IEEE Conference on Control Applications (CCA)*, Oct. 2014.

[18] T. Pock and A. Chambolle, "Diagonal preconditioning for first order primal-dual algorithms in convex optimization," in *2011 IEEE International Conference on Computer Vision (ICCV)*, Barcelona, November 2011, pp. 1762–1769.

[19] 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, December 2014.

[20] A. Pavlov and V. Alstad, "Modelling, simulation and automatic control of ESP lifted wells," 2010, Statoil internal report.

[21] S. Strand and J. R. 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.

[22] D. K. M. Kufoalor, V. Aaker, T. A. Johansen, L. Imsland, and G. O. Eikrem, "Automatically generated embedded model predictive control: Moving an industrial PC-based MPC to an embedded platform," *Optimal Control Applications and Methods*, 2015.