Model-Based Optimizing Control and Estimation using Modelica Models

Lars Imsland Pål Kittilsen Tor Steinar Schei
Cybernetica AS
7038 Trondheim, Norway
{lars.imsland,pal.kittilsen,tor.s.schei}@cybernetica.no

Abstract

This paper reports on experiences from case studies in using Modelica/Dymola models interfaced to control and optimization software, as process models in real time process control applications. Possible applications of the integrated models are in state- and parameter estimation and nonlinear model predictive control. It was found that this approach is clearly possible, providing many advantages over modeling in low-level programming languages. However, some effort is required in making the Modelica models accessible to NMPC software.

Keywords: Nonlinear Model Predictive Control, Online optimization, process control, offshore oil and gas production

1 Introduction

Model Predictive Control (MPC) has become *the* advanced control strategy in the process industries [11]. MPC refers to control strategies which optimize future performance as predicted by a process model, and implement the first part of the calculated control inputs. The optimization/implementation is repeated at regular intervals to achieve robustness through feedback.

Although *linear* MPC (based on linear, typically empirical, process models) is prevalent, it is seen that in many cases, MPC based on nonlinear process models (NMPC), with models derived from first principles and process knowledge, is advantageous or even necessary to achieve better control performance over varying operating conditions (due, for example, to varying product specifications or large process disturbances). In addition to the use of nonlinear process models, another important aspect with NMPC based on models from first principles, is that nonlinear state estimation is an essential part of the control system.

NMPC has received considerable attention in academia, especially in terms of optimization methods [1] and requirements for stability of the resulting closed loop [7]. However, when it comes to industrial application, use of NMPC clearly has an unfulfilled potential, although some applications are being reported, especially in polymerization processes [8, 11].

One important reason for the limited practical use of NMPC, is the substantial time and effort required for developing, validating and maintaining nonlinear process models that are valid over a wide operating range. Importantly, but sometimes overlooked, these models should at the same time be suitable for optimization, in terms of issues such as complexity and smoothness. An important step towards less costly model development is the use of advanced modeling environments, which promotes model structure, model reuse and model maintenance through equation-oriented modeling languages, object orientation and hierarchical composition of sub-models.

Literature reveals some effort towards using advanced process modeling environments in a practical dynamical optimization setting, e.g. [9], where gPROMS are connected to a software environment for dynamic optimization. However, the impression remains that this is very much a developing area.

The use of such models is not limited to NMPC in real-time process control settings. One can envision many types of real-time model-based applications using such models, ranging from data reconciliation, estimation (states, parameters, disturbances, soft-sensing) for monitoring and control, to advisory operator support systems and finally to NMPC. One can argue that a complete NMPC installation involves the other applications mentioned, such that if Modelica models can be used for NMPC, the other applications follows naturally.

The aim of this paper is to discuss requirements, challenges, opportunities, and experiences from using an advanced modeling environment, in particular Dymola/Modelica, for developing models that are used in model-based process control applications.

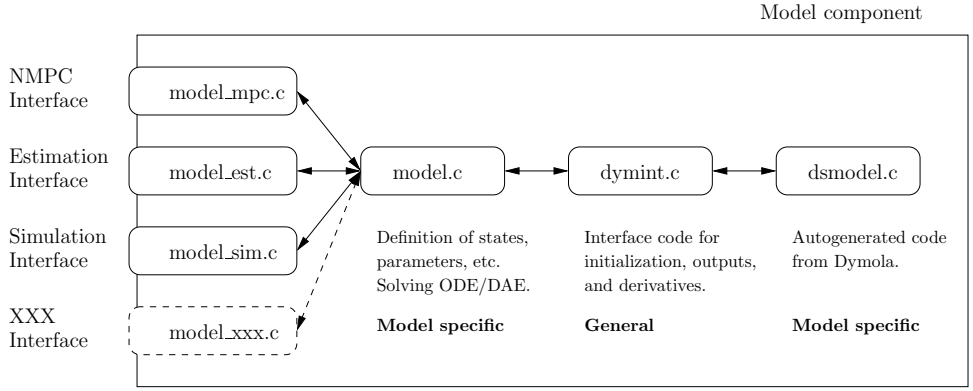

Figure 1: Illustration of model component software structure

The paper is structured as follows: First, we give some remarks on how we have integrated Modelica models (developed using the Dymola tool) into the NMPC tool CYBERNETICA CENIT (for state estimation and optimization). Next, we give some comments on modeling and model types, and give a brief overview over the modeling used in the case examples. In Section 4 we discuss using the Modelica models for state estimation, and give briefly some results obtained using real process data. In Section 5 we discuss optimization in Nonlinear Model Predictive Control, and illustrates with results from a simulation study.

2 Interfacing Modelica/Dymola models with NMPC estimation and optimization software

In this section, we discuss the integration of Modelica models with CYBERNETICA CENIT, a software package for NMPC developed by Cybernetica.

The CYBERNETICA CENIT kernel consists of three components: The NMPC optimization component, the state estimation component, and the model component. The components communicate (with each other and externally) using prespecified interfaces. The two first components are general, while the model component of course is specific for each project. Other CYBERNETICA CENIT modules, for example for offline parameter estimation/optimization for fitting the model to data, also exist and make use of the kernel, but are not considered part of the kernel.

The model component includes discretization (simulation of the model between sample intervals), such that the model is discrete time as seen from the state estimation and NMPC module.

Traditionally, the model component has been coded in C. This has served the purposes well, but for a number of reasons it is desirable to have a more user-friendly way of implementing models, using a high-level modeling language. The overall goal is to reduce the cost of modeling, which is a significant cost factor in a NMPC implementation project. Reasons for the cost reduction include

- Promote reuse of models, also through building of model libraries.
- Better overview of models, ease of implementation and modifications.
- Easier exploitation of modeling effort in other contexts.
- Possibly easier integration of external models (external libraries, customer models, thermodynamics, etc.).

After an investigation of the available alternatives, evaluated against a range of criteria including the issues in the list above, it was found that Modelica was an excellent possible choice for an alternative modeling language. Moreover, the software tool Dymola provided a good Modelica modeling environment, and the opportunity to integrate the models in other software, through the Dymola C-code export option.

With the C-code export, the Modelica model is available in a C-file, dsmodel.c, along with interface functions. Figure 1 illustrates how this C-file can be integrated to form a model component ready to use with CYBERNETICA CENIT.

A distinct advantage of the C-code export offered by Dymola, is that it allows compilation of the total control system including model on any target system equipped with an ANSI C compiler. This is in contrast to systems which base the interface on software component interfaces such as CORBA, and requires (a version of) the modeling environment to run simultaneously.

On the other hand, it might be conceived as a disadvantage that the interface is Dymola specific, and not based on any standard.

Presently, the developed interface only allows obtain-

ing sensitivity information by finite differences, but with the availability of analytical Jacobian from Dymola models, a natural next step, subject for current development, is to include integration of sensitivities in the model component. This can be an advantage both for simulation of the model (see next section), but perhaps more importantly, also for solving the NMPC optimization problem (see Section 5).

3 Modeling and simulation

As mentioned in the previous section, the models developed in Modelica/Dymola must be solved/simulated in the model component. In this section we first give some general remarks on modeling and simulation for NMPC, and thereafter we briefly present the modeling that is done for the case examples in Section 4 and 5.

3.1 Simulating the model

Using equation-based modeling environments such as those based on Modelica, one generally ends up with differential-algebraic equation systems (DAEs). In Dymola, there are implemented algorithms for reformulation (symbolic transformation) of the DAE system such that it from the outside looks like an ODE system,

$$\begin{aligned} \dot{x}(t) &= f(t, x(t), u(t), p), \\ y(t) &= g(t, x(t), u(t), p), \end{aligned} \tag{1}$$

but where the evaluation of the right hand side in general requires the solution of some nonlinear equation systems. The reformulation ensures that these equation systems are as small, and hence as efficiently solved, as possible. However, the solution is based on iterative, local methods, such that it can in general take many iterations to find an acceptable solution, and worse, one is not always guaranteed to find a solution at all. (Although for well-behaved models, one normally finds a solution in few iterations.)

Another issue is that the right hand side might be discontinuous in its arguments. If this is the case, the solvers used to solve the (apparent) ODE above, must be able to handle discontinuities. Moreover, the system will often be stiff, calling for implicit methods with variable step lengths.

Apart from any possible discontinuities, the above issues (DAEs, stiffness, variable step lengths) do not in principle imply any problems using Modelica/Dymola models with an NMPC tool like CYBERNETICA CENIT.

Nevertheless, efficiency and robustness issues may change the picture. Simulation in a NMPC system involves frequent resetting of system parameters (initial states, inputs and estimated parameters), which for DAEs in general requires online re-solving of the nonlinear equation set. For the ODEs exported by Dymola, it leads to frequent re-solving of the 'hidden' nonlinear equation sets.

If we can ensure that the model is a 'real' ODE (without nonlinear equation sets), this is avoided, resulting in increased speed and robustness.

There are no direct help in Dymola to avoid the nonlinear equation sets leading to a DAE system, but the reporting when translating models helps to identify where these nonlinear equations are.

Additionally, ensuring that the model is continuous, means that we can use more efficient solvers that do not have to handle discontinuities.

These issues require more effort during the modeling, and also imply that one often cannot apply other (library, customer) models directly. Nevertheless, the issues are important: In our experience, it is a key aspect of a successful implementation of a NMPC system to find the correct balance between computational complexity of the model/simulation and required model accuracy. Required model accuracy is not easily defined in general, but relates to the specific control objectives of the particular process. In this respect, more complex models are not necessarily more accurate.

When building models from physics, one typically ends up with stiff equation systems, which require implicit solvers with variable step sizes to be solved efficiently. If one chooses to exploit analytical Jacobians in connection with optimization (see also Section 5), this can in principle also be used in the implicit solvers to speed up computation.

3.2 Control-relevant modeling of an offshore oil and gas processing plant

In the North Sea (and on other continental shelfs), oil and gas are produced by drilling wells into the ocean bed. From the wells, a stream of typically oil, gas and water arrive at a surface production facility which main task is to separate the components and make oil and gas ready for export, either through pipelines or by ship. A schematic picture of such an offshore oil and gas processing plant placed on an offshore platform is given in Figure 2. In this case, we have to some extent disregarded water, to concentrate on the oil and gas streams.

In Figure 2, we see that oil and gas enter from two different main sources (each main source is represented by one oil and one gas source) into three separators (the grey ovals). The separators are large tanks which split oil, water and gas. The produced oil is leaving in

the lower right corner of the figure, while the gas enters a compression train from the second and third separator. The compressor train, consisting of five compressors (five stages) compresses the natural gas for re-injection or to export through a pipeline (upper right corner). At the top, there is an additional gas import (from another production platform) with an additional gas compressor. Some of the gas is taken out (top left) as fuel for on board generators.

As can be seen, this is a fairly complex system in terms of numbers of components, however, many of the components are of the same type (mainly separators, compressors, valves, PID controllers, in addition to minor components such as sources, sink, splitter, sensors, etc.), which simplifies overall modeling.

A brief description of some unit models is given below:

- **Separators**: Separators are large tanks which due to their construction, and the different densities of the components, separate water, oil and gas into different process streams. The dynamics of the separator model is based on a mass balance and flash calculations to calculate the split of oil and gas. Based on the separator geometry (and thermodynamics), water and oil levels and gas pressure can be calculated from the component masses.
- **Compressors**: The centrifugal compressor models are static models based on compressor maps (specified by the compressor vendor) of polytropic head vs. volumetric rate, parameterized in compressor speed. The compressor maps are interpolated to yield continuous relations. The compressors are strongly nonlinear, that is, the gain from compressor speed (input) to pressure and volumetric rate are strongly dependent on operating point.
- **Valves**: There are different valve models for liquid and gas flow, both based on basic valve equations. Critical and sub-critical flow are handled. The valve characteristics can be chosen to be either linear or equal percentage via a drop-down menu.

For real-time efficiency reasons, we have made an effort to ensure that we end up with an ODE model. The main manifestation of this, is that we cannot have more than one unit that determines flow between each volume in the model. Therefore, we have introduced semi-physical 'nodes' (the grey round units in Figure 2), and tuned the volumes of these to retain good transient response (for example, by tuning them to be faster than the sample frequency, the exact value is not important in terms of simulation accuracy vs. measurements).

Thermodynamics are important in order to calculate phase transitions between oil and gas. It is also essential to be able to describe the gas' properties over a large span in pressure. Furthermore, the model should have real time capabilities, favoring simple/explicit relations.

For phase equilibrium calculations, correlations of k-values (as function of temperature, pressure and molecular weight) were used together with a simplified representation of the many chemical species found in the real process. Gas density was described by a second-order virial equation, where the model coefficients were fitted to an SRK-equation for the relevant gas composition evaluated for the temperature and pressure range of current interest.

The thermodynamic models have been implemented in the style of the Modelica.Media library in the Modelica Standard Library.

4 State estimation

4.1 State estimation background

Nonlinear state-, disturbance- and parameter estimation are essential for NMPC implementations, but are also important in other settings than purely control-related, such as monitoring and surveillance, and static optimization/RTOs.

Estimation based on Kalman filter algorithms has become tremendously widespread over the last almost 50 years. Other types of estimation algorithms also exist, but are much less used. For nonlinear state estimation, Extended Kalman Filtering (EKF) algorithms should be used. Traditionally, these are based on analytical linearizations, but over the last years, it is seen that using divided differences (or similarly, Unscented Kalman Filtering (UKF) approaches) in many cases provides better performance than linearization-based EKF.

Importantly, the perturbation schemes used in connection with covariance update by divided difference-approaches (including UKFs) obtain information beyond linearization. Thus, for these cases, availability of analytical Jacobians from the model is not necessarily an advantage (unless it speeds up simulation). On the other hand, for estimation schemes based on linearizations (e.g. traditional EKF), or estimation based on numerical optimization (e.g. Moving Horizon Estimation (MHE)-approaches, taking inequality constraints into consideration), analytical Jacobians can be exploited.

CYBERNETICA CENIT has implemented EKFs based

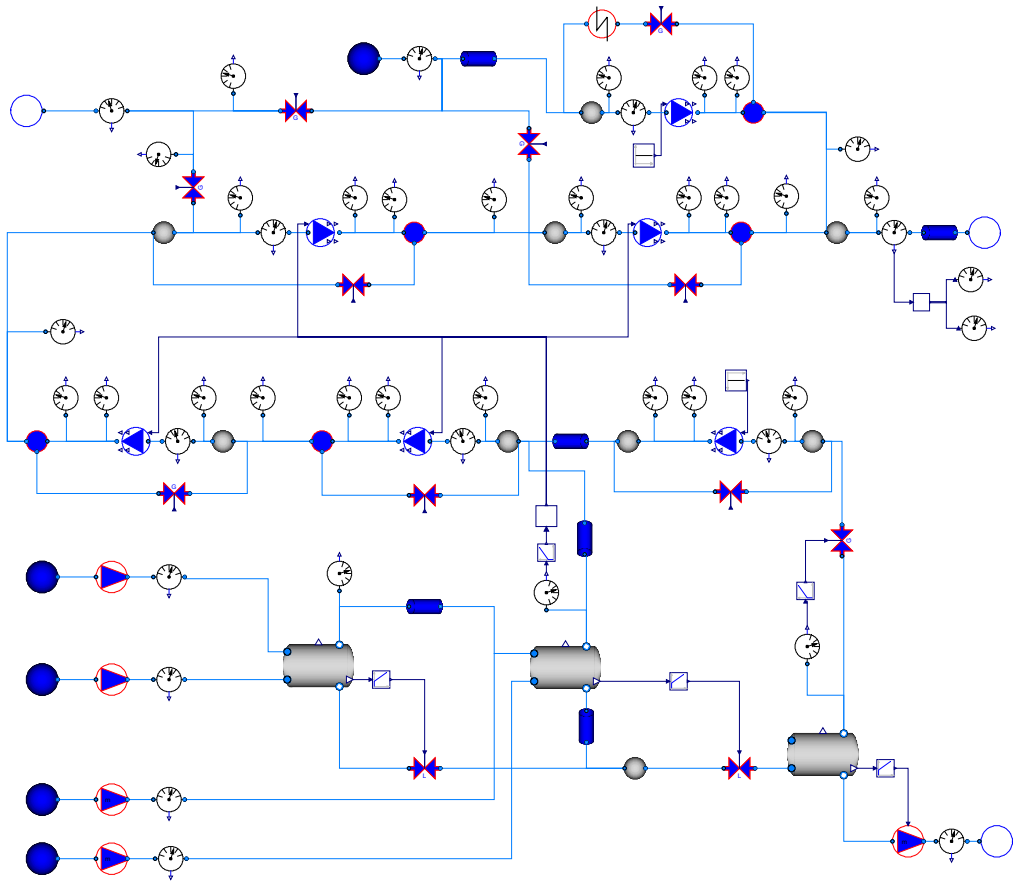

Figure 2: Overview of an offshore oil and gas processing plant, as implemented in Dymola.

on divided differences (both DD1 [13] and DD2 [10], in the notation of [10]), in addition to MHE [12]. For further information and discussion, see also [14].

4.2 Example: State estimation of offshore processing plant

The state and parameter estimation capabilities of CYBERNETICA CENIT (extended Kalman filtering based on finite differences in this case) was used to estimate states and model parameters in a Modelica model of the offshore oil and gas processing plant illustrated in Figure 2. The Modelica model was integrated as a CYBERNETICA CENIT model component as explained in Section 2. Logged data from real operation was used in the test.

The process is fairly well instrumented (a subset of the instrumentation is included in the Modelica model, see Figure 2), but there is no overall reconciliation of the individual measurements nor any overall measurement of key figures. From the individual measurements, most often in engineering units, it is hard to get an overview of the state of the process. With a complete process overview by the help of the model, it is possible to identify the current process state, being an essential basis for taking the correct corrective actions in case of abnormal incidents, and also essential as a starting point for optimization of process operation.

The resulting ODE model of the system was fairly stiff, with modes ranging from around 0.1 seconds to hours, while the sampling time of the process was 1 minute. Therefore, it was absolutely necessary to use an (implicit) ODE solver with varying step lengths. In this case, the CVODE ODE solver[1] was used, with Jacobians found by finite differences. For this model, with 38 states and 35 estimated parameters, the state estimation ran more than 10 times faster than real time.

The state and parameter estimation was successfully tuned and tested on data from several days of operation. An excerpt is shown in Figure 3, where the model initially is simulated 'open loop', and the state estimation is turned on after 60 minutes. The figure demonstrates, for a single compressor stage, how the compressor parameters converge such that the estimated variables match the measured ones.

[1] From the SUNDIALS package, see `http://www.llnl.gov/CASC/sundials/`.

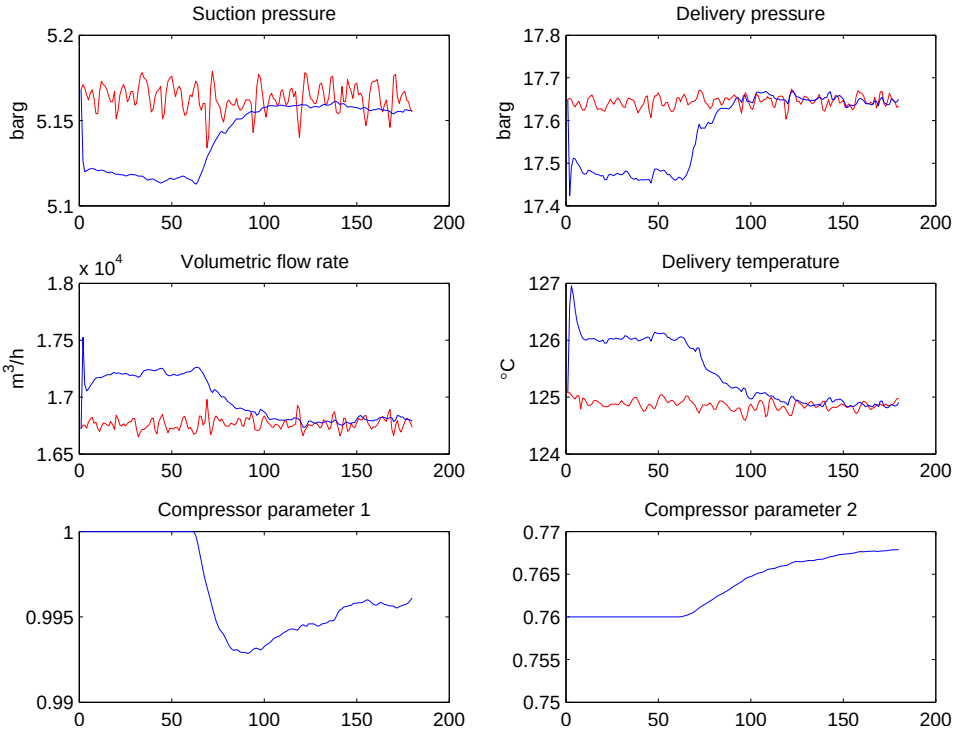

Figure 3: State and parameter estimation of one of the compressor stages. Red lines are real process data, and blue lines are estimated results.

5 Nonlinear Model Predictive Control

5.1 NMPC background

The NMPC optimization problem is a dynamic optimization problem, usually discretized to have a finite number of optimization variables (manipulated variables), that must be solved at regular (sampling) instants. The first part of the optimal solution – the first sample interval, is implemented to the process, before the dynamic optimization problem is resolved before the next sample instant. The optimization problem is using updated process information from a state estimation algorithm.

The optimization problem to be solved at time t, with available state estimate $\hat{x}(t)$, may look something like this, after a piecewise constant parameterization of future manipulated variables (u) over an horizon L:

$$\min_{u_0,u_1,\ldots,u_{L-1}} \sum_{k=0}^{L-1} F(x_{k+1},u_k) \text{ subject to}$$

$$\begin{cases} x_{k+1} - f(x_k,u_k) & = 0,\ k = 0,\ldots,L-1, \\ x_0 & = \hat{x}(t), \\ h_x(x_k) & \geq 0,\ k = 1,\ldots,L, \\ h_u(u_k) & \geq 0,\ k = 0,\ldots,L-1. \end{cases} \quad (2)$$

The discrete-time system $x_{k+1} = f(x_k,u_k)$ is in general obtained by simulation of an ODE (1) over the sample intervals. The functions h_x and h_u represent constraints on states (or controlled variables) and manipulated variables.

In most cases, the (discretized) dynamic optimization problem is solved using numerical algorithms based on sequential quadratic programming (SQP). A SQP method is an iterative method which at each iteration makes a quadratic approximation to the objective function and a linear approximation to the constraints, and solves a QP to find the search direction. Then a linesearch is performed along this search direction to find the next iterate. General SQP solvers may be applied to NMPC optimization, but it is in general very advantageous to use tailor-made SQP algorithms for NMPC applications.

Although a very crucial step in SQP algorithms tailored for NMPC optimization is the linesearch, the main approaches found in the literature are usually categorized by the way they specify the QP for finding the search direction. Arguably, the most common method is the *sequential* approach [5], which at each iteration simulate the model using the current value of the optimization variables $(u_0,u_1,\ldots,u_{L-1})$ to obtain the gradient of the objective function (and possibly the Hessian), thus effectively removing the model equality constraints and the states $x_1,x_2,\ldots,x_L$ as optimization variables. Thereafter a reduced space QP prob-

lem is solved to find the search direction. Conversely, in the *simultaneous* [4, 2] approach, the model is implemented as explicit equality constraints, meaning that the optimization variables are both $u_0, u_1, \ldots, u_{L-1}$ and $x_1, x_2, \ldots, x_L$. The third approach, *multiple shooting* [3, 6], can be viewed as a combination of the two other approaches, where, loosely speaking, the control horizon is divided into some 'sub-horizons' which are solved in a sequential fashion, and equality constraints link the sub-horizons.

There is no general consensus as to which of the above methods is best – probably, it is problem dependent. Note that the two latter approaches allow closer cooperation between the ODE/DAE solvers and SQP optimization than is revealed by the formulation (2).

A sequential approach to dynamic optimization is implemented in CYBERNETICA CENIT. A central issue is how to obtain the necessary sensitivity information for solving the NMPC optimization problem. The two main routes are either by finite differences directly on the objective function, or by integrating ODE/DAE sensitivities along with the model, and calculate NMPC sensitivities based on this. For the latter case, one can exploit the possibility of using analytical Jacobians in Dymola.

Calculating the gradient by finite differences means that many (depending on number of optimization variables) simulations over the control horizon has to be done, which can be time-consuming. Calculating the gradient based on sensitivity integration has the potential to be significantly more efficient, at least for some problems.

A possible problem with forming the NMPC objective function gradient (and possibly Hessian) based on ODE/DAE sensitivities, is that the resulting gradient (and Hessian) is not necessarily a very good approximation to the NMPC objective function to be solved numerically. Consider the following argument: By using finite differences directly on the NMPC objective function (which includes solving the ODE/DAE), we obtain a direct approximation to the gradient of the "numerical" NMPC objective function, which is what we are minimizing numerically. However, by computing the gradient based on ODE/DAE sensitivities, discretization errors will make the computed gradient different from the gradient of the numerical objective function.

Such errors may be important since one for computational complexity reasons is likely to push the accuracy limits for the ODE/DAE solvers.

5.2 Simulation example: NMPC of offshore processing plant

The case used in this section is similar to the one used in the previous section, but is based on (another) production platform. In this case, the focus is on the separation, and the gas compression is not modeled. The process has five different streams of oil and gas, that are to be separated in four separators (a separator train). In contrast to Section 4, the water phase is now explicitly modeled in the separators. The model was tuned to fit data from the real process, but all results shown in this paper are based on simulations.

The process is controlled by level controllers for water and oil, and gas pressure controllers for each separator. This is a standard solution, which works well in many/normal cases. However, in some cases, disturbances in the inlet flows from the inlet pipelines/wells can cause problems for the control of the separators. The levels in the separators will vary, which may cause bad separation and may be detrimental for equipment downstream the separators, due to uneven flow out of the separator train. The purpose for this study is to see if NMPC with state and disturbance estimation, using the level controller setpoints as manipulated variables (MVs), can exploit the buffer capacity in the separators to smooth out the outlet flows of water and oil. The oil is in this particular case entering a distillation column, and the water is entering a glycol regenerator, for regeneration of glycol that is added in the process. Smoother inflow to these units may allow more regular/increased production of the overall process.

There are six manipulated variables: The setpoints for water and oil level controllers in the separators (two of the separators does not separate water, and hence does not have a water level controller). The controlled variables (CVs) are pressures, levels and valve openings for all separators, and rate of change of glycol concentration in one separator.

The resulting model, with 29 states, was not particularly stiff. Therefore, a simple forward Euler ODE solver was used. The NMPC system, including state and disturbance estimation based on finite differences, and NMPC optimization with gradients found by finite differences, ran considerably faster than real time, using a sample interval of 6 s.

Some simulation results with a disturbance, a time-limited increased flow in one of the inflowing pipelines, are shown in Figures 4–6. Figure 4 shows how the NMPC reduces the level controller setpoints in the inlet separator (resulting in increased outflow valve openings, see Figure 6), to let the increased inlet flow (detected by the state and disturbance estimation) be smoothed out over all the separators. Figure 5

demonstrates how the NMPC achieves smoother outflow from the last separator, and that the glycol fraction in the water varies less.

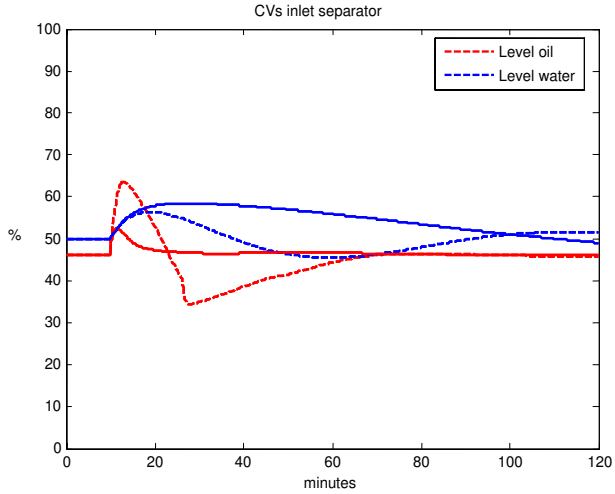

Figure 4: Oil and water levels in the inlet separator, with MPC (solid) and without MPC (dashed).

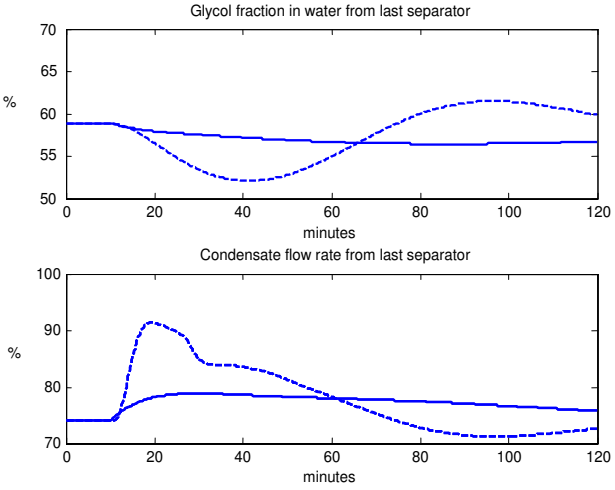

Figure 5: Glycol concentration in glycol/water mixture (top) and oil flow rate (bottom) from the last stage separator, with MPC (solid) and without MPC (dashed).

6 Experiences with using Modelica and Dymola for real time process control applications

In this section, we summarize some of our experiences with using Modelica and Dymola for process control applications.

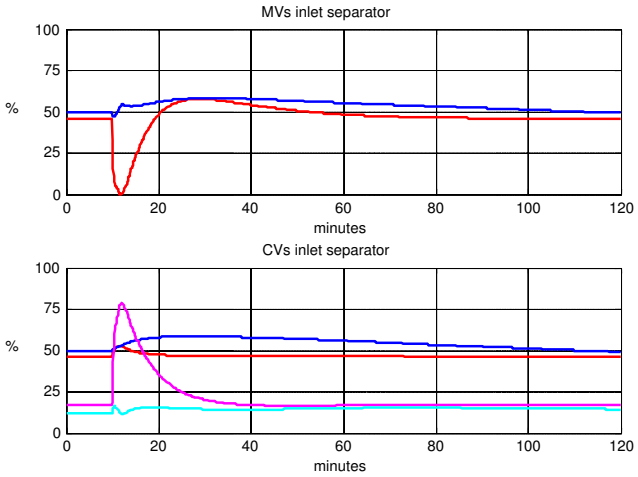

Figure 6: Level controller setpoints (MVs, top) and level and valve openings (CVs, bottom) in inlet separator. Red line is oil, blue line is water, magenta is oil valve opening, cyan is water valve opening.

6.1 Modelica modeling in Dymola

When it comes to modeling, Modelica and Dymola has much to offer over implementing the models in C. Due to the object orientation and the graphical interface it is easy to work on details and at the same time have an overview over the whole model. Using a tool such as Dymola, tasks like manipulation, testing and simulation of the model are convenient.

In this paper, we have used two cases from offshore oil and gas production. We saw some advantages in terms of reuse between these projects, but as we probably will work more in this area, we expect to see further advances at later stages. Using an object-oriented environment like Modelica, makes it easier to develop unit models with more general interfaces, such that they are easier reused. For some of the simple model units, we could use units from the Modelica Standard Library, although in most cases, some modifications were done. By drawing inspiration from Modelica.Media, we had a convenient structure for implementating the thermodynamics.

As with other equation-based modeling systems, debugging models during model development is a challenge in Dymola, and tools to help model debugging would be a benefit. However, by testing unit models thoroughly before aggregating them, many problems can be avoided.

When we have models with nonlinear equations systems (DAEs), we had in some cases problems with initialization of the equation systems, and identifying which variables that were part of the equation system. Of course, when making sure the model was an ODE, these problems were avoided.

6.2 Integration of Modelica/Dymola models in NMPC software

Using the C-code export option of Dymola, it was fairly straightforward to integrate the Modelica models as CYBERNETICA CENIT model components as described in Section 2. However, some modifications must be done to the Modelica model before it is exported, and some information, not directly available from the exported C-code interface functions, must currently be hand-coded into the model component. These issues are discussed below.

A significant part of the effort in constructing the model component based on the structure illustrated in Figure 1, is to generate and keep up to date the referencing/indexing variables in the file model.c. This information is necessary in the NMPC user interface, for instance for tuning of the EKF and the NMPC controller. This is presently coded by hand, but in theory, it should be possible to auto-generate at least large parts of this file based on the model information in the file dsmodel.c. Another possibility might be if Dymola added interface functions/functionality for this.

A NMPC system must exchange the following information dynamically with the model (in addition to states and state derivatives):

- Model inputs: The NMPC must (at least) be able to set the manipulated variables (MVs), the measured disturbances (DVs) and the parameters that are estimated by the EKF.
- Model outputs: Measured variables (for state estimation), and controlled variables (for NMPC).

We have (naturally) chosen to have the MVs and DVs as Modelica inputs. For estimated parameters, there is a choice involving some trade-offs:

- The estimated parameters could be part of the Modelica inputs. The advantages with this is that it is simple to manage in the model component, and that it is possible to calculate analytical Jacobians with respect to these parameters (for use for instance in state estimation using MHE, or in offline parameter estimation). The drawback is that the Modelica unit models must be modified to have these parameters as inputs, which makes it cumbersome to use the same model both for simulation and testing in Dymola, and as model for generating the model component.
- We can access the estimated parameters the same way as all other parameters[2]. This makes 'bookkeeping' of the parameters in the model component (model.c) more involved, and we cannot exploit analytical Jacobians with respect to these parameters. On the other hand, this choice simplifies model maintenance, since we do not have to make new models for estimating parameters.

Presently, our implementation is based on the first choice, which in practice means we must maintain two Modelica models with identical behavior – one for simulation, and one for integration in the model component. This situation is not ideal. One possibility which might rectify the situation, is if Modelica had a variable type that is both parameter and input, and a kind of a 'master switch' that switches the interpretation.

In some cases, it would be an advantage to be able to debug the model code. Due to the structure of the auto-generated code, this is hard.

6.3 Running Modelica/Dymola models in NMPC software

There are some further interesting findings from the case study in Section 5.2. We had this model implemented as a model component in C before we implemented it in Modelica. By using profiling tools, we found that running NMPC with the model component based on the Modelica model, used less than 20% additional time compared to using the pure C model component, where most of the difference must be attributed to Modelica overhead since the models were practically mathematically identical.

However, to get the Modelica-based model to run this fast, we had to implement the Modelica functions used in the Modelica-model in C. Not surprisingly, there is considerable overhead in the implementation of Modelica functions, especially related to indexing of arrays. The possibility to implement Modelica functions in C is supported by the Modelica specification, and implemented in Dymola, and is a considerable practical advantage for real time applications.

7 Conclusions

It is possible to use Modelica/Dymola for modeling for NMPC purposes, with many of the advantages promised by such advanced modeling environments fulfilled. Such environments are helpful in developing complex process models, towards reuse of unit models, and we see potential for increased model value (by extending the application area of the model) and easier customer participation in model development.

However, using Modelica/Dymola models for NMPC has some hurdles. Some effort is required to make a Modelica simulation model ready to be used with

[2] Note that the estimated parameters will be constant in all simulations made, for instance over one sample interval in the EKF, or over the control horizon in the NMPC.

NMPC software. In our experience, two main issues are a) making Modelica model parameters accessible to the NMPC through Modelica inputs and outputs, and b) specifying the structure of state-, input-, output- and parameter vectors (e.g. for NMPC and state estimation tuning).
Finally, we emphasize that process models for NMPC should be developed with the specific task in mind, in terms of issues such as complexity, accuracy and smoothness. In some cases, this means that the model should be an ODE, while models from component-based modeling languages such as Modelica naturally translates into DAEs. It will in general require some effort and compromises for Modelica models to translate into ODEs.

Acknowledgments

We gratefully acknowledge StatoilHydro Research Centre in Porsgrunn, Norway for providing the cases and process data used in this paper.

References

[1] L. Biegler. Efficient solution of dynamic optimization and NMPC problems. In F. Allgöwer and A. Zheng, editors, *Nonlinear Predictive Control*, pages 219–245. Birkhauser, Basel, 2000.

[2] L. T. Biegler, A. M. Cervantes, and A. Wächter. Advances in simultaneous strategies for dynamic process optimization. *Chem. Eng. Sci.*, 57:575–593, 2002.

[3] H. G. Bock, M. Diehl, D. B. Leineweber, and J. P. Schlöder. A direct multiple shooting method for real-time optimization of nonlinear DAE processes. In F. Allgöwer and A. Zheng, editors, *Nonlinear Predictive Control*, volume 26 of *Progress in Systems Theory*, pages 246–267, Basel, 2000. Birkhäuser.

[4] A. M. Cervantes and L. T. Biegler. Large-scale dae optimization using simultaneous nonlinear programming formulations. *AIChE J.*, 44:1038, 1998.

[5] N. M. C. de Oliveira and L. T. Biegler. An extension of newton-type algorithms for nonlinear process control. *Automatica*, 31:281–286, 1995.

[6] M. Diehl, H. G. Bock, and J. P. Schlöder. A real-time iteration scheme for nonlinear optimization in optimal feedback control. *SIAM J. Contr. Optim.*, 43(5):1714–1736, 2005.

[7] R. Findeisen, L. Imsland, F. Allgöwer, and B. A. Foss. State and output feedback nonlinear model predictive control: An overview. *European J. of Control*, 9(2-3):190–206, 2003.

[8] B. A. Foss and T. S. Schei. Putting nonlinear model predictive control into use. In *Assessment and Future Directions Nonlinear Model Predictive Control*, LNCIS 358, pages 407–417. Springer Verlag, 2007.

[9] Y. D. Lang and L. T. Biegler. A software environment for simultaneous dynamic optimization. *Computers & Chemical Engineering*, 31(8):931–942, 2007.

[10] N. K. Poulsen M. Nørgaard and O. Ravn. New developments in state estimation for nonlinear systems. *Automatica*, 36(11):1627–1638, 2000.

[11] S. J. Qin and T. A. Badgwell. A survey of industrial model predictive control technology. *Control Engineering Practice*, 11:733–764, 2003.

[12] C. V. Rao and J. B. Rawlings. Constrained process monitoring: Moving-horizon approach. *AIChE J.*, 48(1):97–109, 2002.

[13] T. S. Schei. A finite-difference method for linearization in nonlinear estimation algorithms. *Automatica*, 33(11):2053–2058, 1997.

[14] T. S. Schei. On-line estimation for process control and optimization applications. In *Proc. 8th International IFAC Symposium on Dynamics and Control of Process Systems (DYCOPS-07)*, Cancún, Mexico, 2007.

[15] T. S. Schei, P. Singstad, and Aa. J. Thunem. Transient simulations of gas-oil-water separation plants. *MIC—Model. Identif. Control*, 12(1):27–46, 1991.