CHEMICAL ENGINEERING TRANSACTIONS Volume 21, 2010
Editor J. J. Klemeš, H. L. Lam, P. S. Varbanov
Copyright © 2010, AIDIC Servizi S.r.l., ISBN 978-88-95608-05-1 ISSN 1974-9791
DOI: 10.3303/CET1021087

Exploiting C++ polymorphism for operational optimization of chemical processes

Flavio Manenti[1*], Nadson M. N. Lima[2], Lamia Zuñiga Liñan[2], Simone Colombo[1]

[1]CMIC Dept. "Giulio Natta", Politecnico di Milano, Piazza Leonardo da Vinci 32
20133 Milano, Italy
[2] State University of Campinas (UNICAMP), Dept. of Chemical Processes
flavio.manenti@polimi.it

Object-oriented programming is more and more spreading in engineering and scientific areas for some relevant benefits making it particularly appealing to write complex codes. Nevertheless, the use of such a programming philosophy still encounters large inertia in those areas characterized by a long programming experience and one of them is engineering. This is mainly due to a set of existing models and subroutines wrote in procedural (usually Fortran) language. The present paper is aimed at showing some benefits coming from object-oriented programming applied to the field of process optimization and specifically to the operational levels of supply chain management paradigm such as nonlinear model predictive control (NMPC). Polymorphism is exploited to provide a single solution for different NMPC techniques such as input blocking, offset blocking, and Δ-blocking.

1. Introduction

The optimization of unit operations and chemical processes has recently seen the significant spreading of Nonlinear Model Predictive Control (NMPC) methodology against other control alternatives (Qin and Badgwell, 2003; Bauer and Craig, 2008). Success of NMPC is mainly due to the following reasons: it is intrinsically able to manage nonlinearities in process dynamics and in profits (Manenti and Rovaglio, 2008; Dones et al., 2010); it can be based on first-principles mathematical models and on nonlinear semi-empirical models as well (Lima et al., 2009a; Lima et al., 2009b); it allows solving simultaneously the predictive control and the dynamic optimization (Zavala et al., 2005; Manenti et al., 2010). The basic architecture of NMPC has been wide described in the literature (Morari and Lee, 1999; Rawlings, 2000; Findeisen and Allgower, 2003). The plant provides raw data, which is reconciled (Bagajewicz, 2003; Signor et al., 2010) and sent to NMPC at each sampling time. Specifically, the reconciled data are sent to an optimization routine, which includes an objective function, a dynamic model and, usually, according to the mathematical model type, a numerical integrator to solve specific differential or differential-algebraic systems

(Manenti et al., 2009). In addition, economical and market data have to be provided to the NMPC structure when some economical goals are within the objective function.

2. Alternative Solutions for NMPC

NMPC usually needs a large computational effort when detailed models are adopted on the industrial scale. Actually, the search for the optimal sequence of values for each manipulated variable leads to a constrained multi-dimensional NonLinear Programming (NLP) problem. The dimension of the resulting NLP problem is directly related to the number of degrees of freedom available to control the system in study as well as to number of time intervals within the selected control horizon: $GdL = n_{VM} \cdot N$, where GdL is the total amount of degrees of freedom of NLP, n_{VM} is the number of manipulated variables, and N represents the number of time intervals within the control horizon H_C. NMPC based on complex models with many degrees of freedom may require hard computational efforts with the risk that calculations are not early performed by compromising effectiveness of NMPC itself and its on-line feasibility. To reduce computational efforts some authors (Palavajjhala et al., 1994; Cagienard et al., 2007) proposed the so-called *blocking techniques* described in the following section.

3. NMPC-Blocking Techniques

The term *blocking* points out a set of techniques aimed at reducing the computational effort during the evaluation of the optimal set of manipulated variables. A qualitative picture is given in Figure 1.

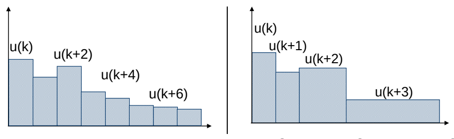

Figure 1: sequence of manipulated variables before (left) and after (right) blocking

3.1 Input Blocking (IB)

Input Blocking (IB) is one of these techniques where some degrees of freedom are intentionally neglected throughout the optimization by keeping them constant. Let us consider a matrix $\mathbf{A}$ containing all control actions for each interval $i = 1,...,N$ of H_C:

$$\mathbf{A} \in \mathbb{R}^{n_{VM} \times N} = \begin{bmatrix} u_{1,0} & u_{1,1} & \cdots & u_{1,N\text{-}1} \\ u_{2,0} & u_{2,1} & \cdots & u_{2,N\text{-}1} \\ \vdots & \vdots & \ddots & \vdots \\ u_{m,0} & u_{m,1} & \cdots & u_{m,N\text{-}1} \end{bmatrix} = \begin{bmatrix} \mathbf{u}_0^T & \mathbf{u}_1^T & \cdots & \mathbf{u}_{N-1}^T \end{bmatrix} \tag{1}$$

where vectors $\mathbf{u}_i$ contain optimal values of all manipulated variables within the $i-th$ time interval. If we consider the transpose matrix $\mathbf{U}$:

$$\mathbf{U} \in \mathbb{R}^{N \times n_{VM}} = \mathbf{A}^T = \begin{bmatrix} u_{1,0} & u_{2,0} & \cdots & u_{m,0} \\ u_{1,1} & u_{2,1} & \cdots & u_{m,1} \\ \vdots & \vdots & \ddots & \vdots \\ u_{1,N-1} & u_{2,N-1} & \cdots & u_{m,N-1} \end{bmatrix} = \begin{bmatrix} \mathbf{u}_0^T & \mathbf{u}_1^T & \dots & \mathbf{u}_{N-1}^T \end{bmatrix}^T \qquad (2)$$

IB technique can be defined as follows:

$$\hat{\mathbf{U}} \in \mathbb{R}^{M \times n_{VM}} = \begin{bmatrix} u_{1,0} & u_{2,0} & \cdots & u_{m,0} \\ u_{1,1} & u_{2,1} & \cdots & u_{m,1} \\ \vdots & \vdots & \ddots & \vdots \\ u_{1,M-1} & u_{2,M-1} & \cdots & u_{m,M-1} \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{u}}_0^T & \hat{\mathbf{u}}_1^T & \dots & \hat{\mathbf{u}}_{M-1}^T \end{bmatrix}^T \qquad (3)$$

where M must be smaller than N to be effective since $N-M$ are the optimal manipulated values of some time intervals removed from the NLP problem; it significantly reduces the NLP dimension. Matrix $\hat{\mathbf{U}}$ is such that $\mathbf{U} = \mathbf{T} \cdot \hat{\mathbf{U}}$ where $\mathbf{T} \in \mathbb{R}^{N \times M}$ is the so-called blocking matrix. Blocking matrix is a Boolean matrix having a single nonzero coefficient per row; it contains relevant information about NLP reduction. IB technique allows the reduction of computational effort by making the NMPC feasible on-line even for complex systems. Unfortunately, it may sometimes lead to instabilities and longer transients; in fact, its application is always a good compromise between computational requirements and solution accuracy.

3.2 Offset Blocking (OB)

A blocking variant is to assign a value to manipulated variables at each interval of H_C: $u_i = kx_i + c_i$, with the external parameter k evaluated by means of an infinite horizon control law and the minimization parameter c_i. Given a law to determine u_i, the optimization problem can be formulated as $\mathbf{C} \in \mathbb{R}^{N \times n_{VM}} = \left[\mathbf{c}_0^T, \mathbf{c}_1^T, ..., \mathbf{c}_{N-1}^T\right]^T$ and OB can be applied to elements of the matrix $\mathbf{C}$: $\hat{\mathbf{C}} \in \mathbb{R}^{M \times n_{VM}} = \left[\hat{\mathbf{c}}_0^T, \hat{\mathbf{c}}_1^T, ..., \hat{\mathbf{c}}_{M-1}^T\right]$ with $M < N$ and $\hat{\mathbf{C}}$ such that $\mathbf{C} = \mathbf{T} \cdot \hat{\mathbf{C}}$, thus the NLP can be formulated as $\min\limits_{\hat{\mathbf{C}}} J(\mathbf{x}, \mathbf{u})$. OB variant offers the advantage of a complete sequence of optimal actions, contrarily to the IB technique, by improving the control flexibility. Nevertheless, some instability problems may arise as per IB technique.

3.3 Δ-Input Blocking (DIB) and Δ-Offset Blocking (DOB)

These blocking techniques limit neither the number of control actions nor the amount of parameters, rather they impose a fixed difference between either a control action and the previous one or a parameter and the previous one. Let us suppose to have $N = 4$, $m = 1$, and $M = 2$:

$$\mathbf{u} = \begin{bmatrix} u_0 & u_1 & u_2 & u_3 \end{bmatrix} \qquad (4)$$

Also, let us suppose to select $\hat{u}_0$ and $\hat{u}_2$ as input:

$$\hat{\mathbf{u}} = \begin{bmatrix} \hat{u}_0 & \hat{u}_2 \end{bmatrix} \qquad (5)$$

Once the deviation $\hat{u}_k - \hat{u}_{k-1}$ is assigned, it is possible to define directly both $\hat{u}_1$ and $\hat{u}_3$ by a linear interpolation. Again, even though the computational effort is significantly reduced, these techniques may take to instabilities and some constrained could be unavoidably unsatisfied (Cagienard et al., 2007).

4 Exploiting Polymorphisms

A single C++ class can solve all the above techniques for NMPC by exploiting polymorphism of object-oriented programming. To develop the class a series of algorithms is necessary to integrate differential systems, to solve NLP problems, to reconcile data, and identify possible outliers. The present research activity is based on the freely downloadable *BzzMath* library (Buzzi-Ferraris, 2010), which provides specific solvers for the aforementioned issues (Buzzi-Ferraris and Manenti, 2010b, 2010c). First of all, raw data acquired from the plant should be treated in order to detect any kind of gross error. An opportune class to reconcile raw data set, which is based on $\mathbf{QR}$ factorization and linear systems solution, can be adopted; for sake of conciseness, data reconciliation is not described here (see Buzzi-Ferraris and Manenti, 2010c; Signor et al., 2010). Reconciled data is then used to initialize MPC structure: the optimizer (Buzzi-Ferraris and Manenti, 2009, 2010a) is called the first time to evaluate the best manipulated variables $\mathbf{u}$ by minimizing the objective function. To do so, all the equality and inequality constraints (including the differential system) have to be evaluated and an opportune differential solver should be invoked. The differential system is then integrated on a specific prediction horizon h_p in order to predict future system behaviour according to different values of $\mathbf{u}$. After an iterative procedure, the optimal vector $\mathbf{u}$ is implemented in the plant and new data are acquired to restart the cycle. The basic NMPC solver can be invoked through the following constructor:

```
BzzModelPredictiveControl NMPC(hp,hc,y0,u0,ad,Dyn,FObj,uL,uU);
```

where `hp` is the prediction horizon; `hc` the control horizon; `y0` the reconciled measures acquired by the field for each MPC call; `u0` the initial values of manipulated variables; `ad` is an integer vector for discriminating between algebraic (`ad[i]=0`) and differential (`ad[i]=1`) equations of the system described in the function `Dyn`; `FObj` is the weighted, normalized, least squares objective function; optionally, `uL` and `uU` are minimum and maximum constraints, respectively, of manipulated variables.
Since NMPC algorithm is practically the same for blocking techniques, object-oriented programming offers the possibility to define additional constructors for the same class and when the user invokes NMPC solver through a specific constructor, the class itself is able to automatically realize which technique the user is requiring for solving his/her NMPC problem. The constructor for blocking techniques needs the definition of the aforementioned Boolean matrix $\mathbf{T}$ and, since all relevant information is included in such a matrix, no else information is required:

```
BzzModelPredictiveControl NMPC(hp,hc,y0,u0,ad,Dyn,FObj,uL,uU,T);
```

Acording to the structure of the matrix $\mathbf{T}$, hence according to the user needs, the object `NMPC` will solve IB, OB, DIB, and DOB problems.

4.1 Class Validation

The class has been validated on the Luyben's Continuous Stirred Tank Reactor (CSTR) (Luyben and Luyben, 1997), where an exothermic reaction A→B takes place. The reactor temperature *Tr* is controled by varying the coolant temperature *Tc* by preserving the inlet flowrate and composition. The case of *Figure 1* shows both the reactor temperature response to a servomechanism problem (right) and the corresponding action imposed by NMPC to *Tc* (left). For the sake of coinciness, only the case of IB technique is checked here by means of different percentage of blocked degrees of freedom. With a large percentage of blocked degrees of freedom, smaller computational times are obtained: the overall simulation of *Figure 1* has taken 433 s with 0 % IB against 36s with 90 % IB on an Intel Core 2 Duo, 2 GHz, 2 GB of RAM. No deviations of IB are appreciated against classical NMPC up to 80% of blocked percentage. In this specific case, instabilities appear when the blocked percentage is significantly large (>80 %).

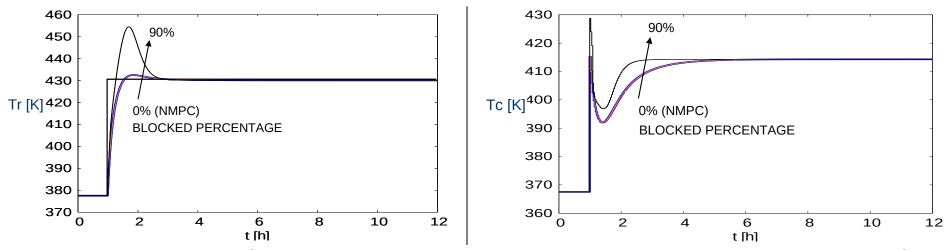

Figure 1: validation of BzzModelPredictiveControl class. IB technique is checked from a blocked percentage of 0% (classical NMPC) to 90 %. Reactor temperature (left) is the controlled variable and coolant temperature (right) is the manipulated variable.

5 Conclusions

Polymorphism of object-oriented programming is a very powerful feature to solve different problems having the same numerical bases. The paper showed the case of operational optimization of chemical processes, specifically the nonlinear model predictive control and some related techniques to improve its performances, by showing the simplicity of implementing different approaches in a single class to tackle different nonlinear programming problems. From this perspective, polymorphism can be successfully exploited even for higher levels of supply chain management.

References

Bagajewicz, M.J. (2003). Data Reconciliation and Instrumentation Upgrade. Overview and Challenges. FOCAPO 2003. 4th International Conference of Computer-Aided Process Operations, Coral Springs, Florida, 103-116.

Bauer, M. and Craig, I.K. (2008). Economic Assessment of Advanced Process Control - A Survey and Framework. Journal of Process Control, 18, 2-18.

Buzzi-Ferraris, G. (2010). BzzMath: Numerical library in C++. Politecnico di Milano, <chem.polimi.it/homes/gbuzzi> (last accessed 12.7.2010)

Buzzi-Ferraris, G. and Manenti, F. (2009). Kinetic models analysis. Chemical Engineering Science, 64(5), 1061-1074.

Buzzi-Ferraris, G. and Manenti, F. (2010a). A Combination of Parallel Computing and Object-Oriented Programming to Improve Optimizer Robustness and Efficiency. Computer Aided Chemical Engineering, 28, 337-342.

Buzzi-Ferraris, G. and Manenti, F. (2010b). Fundamentals and Linear Algebra for the Chemical Engineer: Solving Numerical Problems. Wiley-VCH, Weinheim, Germany.

Buzzi-Ferraris, G. and Manenti, F. (2010c). Interpolation and Regression Models for the Chemical Engineer: Solving Numerical Problems. Wiley-VCH, Weinheim, Germany.

Cagienard, R., Grieder, P., Kerrigan, E.C. and Morari, M. (2007). Move blocking strategies in receding horizon control. Journal of Process Control, 17(6), 563-570.

Dones, I., Manenti, F., Preisig, H.A. and Buzzi-Ferraris, G. (2010). Nonlinear Model Predictive Control: a Self-Adaptive Approach. Industrial & Engineering Chemistry Research, 49(10), 4782-4791.

Findeisen, R. and Allgower, R. (2003). The quasi-infinite horizon approach to nonlinear model predictive control. Nonlinear and Adaptive Control, 281, 89-108.

Lima, N.M.N., Manenti, F., Maciel Filho, R., Embiruçu, M. and Wolf Maciel, M.R. (2009a). Fuzzy Model-Based Predictive Hybrid Control of Polymerization Processes. Industrial & Engineering Chemistry Research, 48(18), 8542–8550.

Lima, N.M.N., Zuñiga, L., Maciel Filho, R., Manenti, F., Manca, D. and Embiruçu, M. (2009b). Dynamic Optimization of MMA and VAc Copolimerization Process. Chemical Engineering Transactions, 17(3), 1383-1388.

Luyben, W.L. and Luyben, M.L. (1997). Essentials of Process Control. McGraw-Hill, New York, USA

Manenti, F., Dones, I., Buzzi-Ferraris, G. and Preisig, H.A. (2009). Efficient Numerical Solver of Partially Structured Differential and Algebraic Equation Systems. Industrial & Engineering Chemistry Research, 48, 9979-9984.

Manenti, F., Lima, N.M.N., Zuñiga, L., Cuoci, A. and Frassoldati, A. (2010). Generalized Classes for Lower Levels of Supply Chain Management: Object-Oriented Approach. Computer Aided Chemical Engineering, 28, 139-144.

Manenti, F. and Rovaglio, M. (2008). Integrated multilevel optimization in large-scale polyethylene terephthalate plants. Industrial and Engineering Chemistry Research, 47(1), 92-104.

Morari, M. and Lee, J.H. (1999). Model predictive control: past, present and future. Computers and Chemical Engineering, 23(4-5), 667-682.

Palavajjhala, S., Motard, R.L. and Joseph, B. (1994). Blocking and Condensing design for quadratic dynamic matrix control using wavelets. Industrial & Engineering Chemistry Research, 33(5), 1159-1173.

Qin, S.J. and Badgwell, T.A. (2003). A survey of industrial model predictive control technology. Control Engineering Practice, 11(7), 733-764.

Rawlings, J.B. (2000). Tutorial Overview of Model Predictive Control. IEEE Control Systems Magazine, 20(3), 38-52.

Signor, S., Manenti, F., Grottoli, M.G., Fabbri, P.and Pierucci, S. (2010). Sulfur Recovery Units: Adaptive Simulation and Model Validation on an Industrial Plant. Industrial & Engineering Chemistry Research, DOI: 10.1021/ie901749t.

Zavala, V.M., Flores-Tlacuahuac, A. and Vivaldo-Lima, E. (2005). Dynamic optimization of a semi-batch reactor for polyurethane production. Chemical Engineering Science, 60(11), 3061-3079.