Industrial implementation of nonlinear model predictive controllers

Jørgen K. H. Knudsen*, John Bagterp Jørgensen**

* 2-control Aps, Frimodtsvej 11, DK-2900 Hellerup, Denmark
(e-mail: JoeK@2-control.dk)

** Department of Applied Mathematics and Computer Science,
Technical University of Denmark, DK-2800 Lyngby
(e-mail: jbjo@dtu.dk)

Abstract:
This paper describes some important factors for successful implementation and maintenance of nonlinear MPC in an industrial environment. ModelBuilder is a simple modeling language, which uses basic mass and heat balances to describe plant dynamics. The nonlinear ModelBuilder model can effectively be linearized as a set of state space models around given operating points. Based on this set of state space models a linear MPC controller calculates an optimal control strategy. Subsequently the set of state space models are updated based on the predicted future plant trajectory.

Keywords: Advanced process control (APC), Nonlinear model predictive control (NMPC), Model predictive control (MPC), Models, ModelBuilder

1. INTRODUCTION

In order to be successful, an advanced process control application must be commissioned and maintained by process engineers with detailed knowhow and understanding of the process to be controlled. Normally the process engineer does not have a Ph.D degree and is not a programmer used to work in MATLAB or a high level language such as C#. We have to provide tools which can be used by the process engineer, requiring a minimum of programming skills and training, and which can be used occasionally, whenever maintenance of the control system is required.

Furthermore it is important to use simulation of the plant to make the initial tests an tuning of the controller before moving the controller to the real plant and presenting it to the plant operators. If the operators loses confidence to the advanced process control system, it will be virtually impossible to achieve a successful installation.

MPC controllers based on linear models are being implemented successfully in the industry today. Linear models can be defined and entered into the APC control system as state space models or transfer functions.

For MPC using nonlinear models, the situation is much more difficult. Nonlinear models can be developed using high level programming languages as MATLAB or C#. Programming the necessary Jacobians are tedious and error prone. Development of these models requires a MATLAB or a compiler.

ModelBuilder provides a solution for development of nonlinear models. ModelBuilder describes process dynamics at high abstraction level using the mass and energy balances familiar to process engineers. ModelBuilder generates the model without requiring a compiler and finally

Fig. 1. APCSuite modules.

ModelBuilder generates Jacobians using automatically differentiation of the system equations.

ModelBuilder is a part of 2-control’s APCSuite product, shown in Fig. 1. APCBox is the server running the MPC controllers 24/7. APCBox exchanges data and setpoints with the plant PLC system using an OPC connection. APCView in the client program used to configure APCBox and supervise the control loops.

The use of ModelBuilder is demonstrated on a U-Loop reactor for single cell protein production (Olsen et al. (2010)) and finally the generated model is used in a nonlinear controller.

2. THE U-LOOP REACTOR

The U-Loop reactor consist of CSTR and an U shaped plug flow reactor as shown in the left part of Fig. 2. In the U-Loop reactor biomass (single cell protein) grows on
methanol and oxygen. The gases and liquids are mixed efficiently in the plug flow reactor enabling the reaction. After the U-Loop reactor, separator and drying units are used to produce the final single cell protein. This part of the process is not considered in this article.

Inputs to the U-Loop reactor are the methanol, \( F_s \), water, \( F_w \), and oxygen, \( F_o \) flow rates. The output flow from the CSTR reactor is characterized by the concentration of biomass, \( C_x \), the concentration of methanol, \( C_s \), and the concentration of oxygen in the liquid phase, \( C_o \).

The growth rate of the biomass is a highly nonlinear function of the methanol concentration, as shown in Fig. 3, making it a challenging task to control the U-Loop reactor.

3. MODELBUILD

ModelBuilder is used to generate ordinary differential equations, ODE, of the form \( \dot{x} = F(x,u) \) or differential algebraic equations, DAE, of the form \( F(x,u) = 0 \). \( x \) is a vector of states, \( \dot{x} \) is the time derivative of the states and \( u \) a vector of process inputs. In both cases the process outputs are defined as \( y = G(x,u) \).

ModelBuilder source code consists of a declaration section followed by an equations section:

```
// Yield Coefficients
Constant Yso = 0.439;
Constant Ysx = 0.732;
// Molar Weights
Constant Mws = 1.0079*4+12.011+15.9994; // [S = CH3OH][g/mol]
Constant Mwo = 2*15.9994; // [O = O2][g/mol]
Constant Mwx = 12.011+1.8*1.0079+0.5*15.9994+0.2*14.0067; // [X][g/mol]
// Kinetic Parameters
Parameter mumax = 0.37; // [1/hr]
Parameter ks = 0.021; // [kg/m3]
Parameter ki = 0.38; // [kg/m3]
Parameter ko = 6.4e-5; // [kg/m3]
```

Assignments can be expressions. Parameters can be changed from outside the the model.

Declaration of states and inputs:

```
// CSTR states
State Cxcstr = 10.3; // [protein concentration][mol/m3]
State Cscstr = 0.01; // [substrate concentration][mol/m3]
State Cocstr = 0.123; // [oxygen concentration][mol/m3]
Input Fs = 0.00411; // [Substrate input][m3/hr]
Input Fw = 0.1736; // [Water input][m3/hr]
Input Fg = 0.7034; // [gas input][m3/hr]
```

Declaration of variables:

```
// Mixer
Var Fliq; // [Liquid flow][m3/hr]
Var Ftot; // [Total flow][m3/hr]
Var epsilon; // [gas/liquid ratio]
Var v; // [PFR flux][m/hr]
```

The values of the variables must be a function of the states and inputs, as described in the equation section.

Finally the outputs \( y \) can be defined as:

```
Output Cxcstr; // [protein concentration][mol/m3]
Output Cscstr; // [substrate concentration][mol/m3]
Output Cocstr; // [oxygen concentration][mol/m3]
```
The positions in the \( y \) vector follow the declaration sequence.

The equation section is used to specify the relations between inputs, variables and states.

\[
F = F_s + F_w; \quad // \text{Total flow} \\
F_{tot} = F_{liq} + F_{g}; \\
\epsilon = F_{g} / F_{tot}; \\
v = F_{tot} / A_{pfr}; \\
C_{pfrin} = (F_s * C_{xin} + F_{R} * C_{xstr}) / F_{liq}; \\
C_{str} = (F_s * C_{met} + F_{R} * C_{scstr}) / F_{liq}; \\
C_{opfrin} = (F_s * C_{oin} + F_{R} * C_{ocstr}) / F_{liq}; \\
C_{gopfrin} = C_{goin};
\]

The plug flow reactor section of the U-Loop reactor is discretized into \( N \) sections of equal size. In the declaration section the states and variables for these sections can be defined as:

\[
\text{State } C_x[N] = 10.3; \quad // \text{protein concentration} \quad [\text{mol/m}^3] \\
\text{State } C_s[N] = 0.01; \quad // \text{substrate concentration} \quad [\text{mol/m}^3] \\
\text{State } C_o[N] = 0.123; \quad // \text{oxygen concentration} \quad [\text{mol/m}^3] \\
\text{State } C_{g0}[N] = 10.0; \quad // \text{oxygen gas phase concentration} \quad [\text{mol/m}^3]
\]

The declaration \( State \ C_x[N] = 10.3 \) is equivalent to 15 statements.

\[
\text{State } C_x[0] = 10.3; \quad // \text{protein concentration} \quad [\text{mol/m}^3] \\
\text{State } C_x[1] = 10.3; \quad // \text{protein concentration} \quad [\text{mol/m}^3] \\
\text{State } C_x[2] = 10.3; \quad // \text{protein concentration} \quad [\text{mol/m}^3] \\
\text{State } C_x[3] = 10.3; \quad // \text{protein concentration} \quad [\text{mol/m}^3] \\
\text{State } C_x[14] = 10.3; \quad // \text{protein concentration} \quad [\text{mol/m}^3]
\]

In the equations section, the equations can be repeated using the Repeat statement.

\[
\text{Repeat(0 : N - 1)} 
\]

\[
\text{...}
\]

The \textit{Repeat} \((0 : N - 1)\) \textit{...} \textit{repeats the statements in} \{ \textit{...} \} \textit{with the index variable ix set to values from 0 to \( N - 1 \). The Repeat statement can handle up to 3 dimensions, with index variables ix, iy, iz.}

\textit{ModelBuilder} models can be used as ordinary differential equations, ODE, or differential algebraic equation, DAE. The ODE are computationally more efficient than DAE models, but it put some restrictions the equation system. For that reason we standardized the APESuite system to use the DAE formulation. This choice is not presented to the user.
4. THE MPC CONTROLLER

The resulting ModelBuilder model can be used by APCSuite’s MPC controllers. Here the MPC is defined declaring inputs, outputs, targets, limits etc. as shown in Fig. 5.

The default option is to linearize the ModelBuilder model as a state space model around the operations point specified in the ModelBuilder source code, resulting in a conventional linear MPC controller.

Alternatively, the MPC can use the ModelBuilder model to calculate a collection of linear state space models along future trajectories predicted by the MPC controller, as shown in Fig. 4. The state space models are collected in a ModelProvider, which is initialized with state space models corresponding to the operating point given in the ModelBuilder source code.

The plant outputs are read via the OPC connection to the plant PLC, and the current state of the plant is estimated with a Kalman filter.

The MPC algorithm runs as a conventional linear MPC using state space models from the ModelProvider, and calculates setpoints for plant inputs together with the anticipated future trajectory of the plant.

The resulting trajectory scheme was tried on the U-Loop reactor.

After a number of control cycles the MPC controller approaches the nonlinear predictions as shown in Fig. 7. After a while, the predicted trajectory is stable as shown in Fig. 8. Finally, Fig. 9 shows the state after many control cycles just before reaching the final steady state of the process.

In Fig. 10 the MPC controller was started without updating the models. This resulted in an unacceptable oscillatory response, which disappeared after re-starting updating of the models as shown in Fig. 11. This is the type of results, which should be studied with a simulator and not in front of a group of sceptical process operators.
6. CONCLUSION

The ModelBuilder language and the proposed control cycle with a ModelProvider enables process engineers to develop and maintain advanced process controllers for nonlinear processes without having to use MATLAB or compilers.

The relatively simple strategy by using the linear algorithms combined with subsequent updating of the models, provides a fast immediate response to the process and it works for difficult cases as the U-Loop reactor. For unstable systems more complicated strategy based on “multiple-shooting” might be required.

ACKNOWLEDGEMENTS

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 723661, Spire project: “Coordinating Optimisation of Complex Industrial Processes” (www.cocop-spire.eu). This article reflects only the author’s views and the Commission is not responsible for any use that may be made of the information contained therein. This work also received funding in the project EUDP 64013-0558 in the IEA annex for energy efficient process control.

REFERENCES