NONLINEAR MODEL PREDICTIVE CONTROL USING AUTOMATIC DIFFERENTIATION

Yi Cao[*], **R. Al-Seyab**

School of Engineering, Cranfield University, UK
[*] Email: y.cao@cranfield.ac.uk. Tel: 01234750111. Fax: 01234750728

Keywords: Nonlinear Model Predictive Control, Dynamic Optimization, Automatic Differentiation, Evaporation Process

Abstract

Although nonlinear model predictive control (NMPC) might be the best choice for a nonlinear plant, it is still not widely used. This is mainly due to the computational burden associated with solving a set of nonlinear differential equations and a nonlinear dynamic optimization problem. In this work, a new NMPC algorithm based on nonlinear least square optimization is proposed. In the new algorithm, the residual Jacobian matrix is efficiently calculated from the model sensitivity functions without extra integrations. Recently developed automatic differentiation techniques are applied to get the sensitivity functions accurately and efficiently. The new algorithm has been applied to an evaporation process with satisfactory results to cope with large setpoint changes, measured and unmeasured severe disturbances and process-model mismatches.

1 Introduction

Model predictive control (MPC) strategies have been well received by industry because they are intuitive and can explicitly handle MIMO systems with input and output constraints. Until recently, industrial applications of MPC have relied on linear dynamic models even though most processes are nonlinear. MPC based on a linear model is acceptable when the process operates at a single setpoint and the primary use of the controller is the rejection of small disturbances[17]. Operating points of modern chemical processes vary over large regions. Such processes cannot be modelled adequately using linear models. These conditions are observed in many cases including change over in continuous processes, tracking problems in startup and batch processes and the control of nonlinear reactors [17, 2]. Because these processes make transition over the nonlinear range of the system, linear MPC often results in poor control performance. To properly control these processes a nonlinear dynamic process model has to be used, rather than a linear convolution model [1]. Recognizing this need, a number of MPC algorithms incorporating nonlinear prediction models (hence called as nonlinear MPC or NMPC) have appeared lately [18, 6].

The objective of NMPC is to select a set of future control moves (control horizon) in order to minimize a function based on a desired output trajectory over a prediction horizon. The introduction of nonlinear models inside the control algorithm does not cause major problems from a theoretical point of view [1]. However, the computation involved to solving the optimization problem at every sampling time can become so intensive, particularly for high-dimensional systems, that it could make on-line applications almost impossible [19]. There exist a number of strategies for tracking the optimal control problem through nonlinear programming (NLP): successive linearization, simultaneous methods, sequential approach, and others [11, 14]. In a successive linearization solution, the Jacobian linearization is performed over the prediction horizon or at a number of time steps in the prediction horizon [3, 21, 12]. Simultaneous solution, transforming the differential equations to algebraic equations which are solved as nonlinear equality constraints in the optimization [7, 16]. In the sequential approach, improved closed-loop performance is achieved as the nonlinear model is used directly in the NMPC calculations. Standard NLP however is not designed to handle dynamic constraints. This limitation can be overcome using a two-stage approach where an optimization routine serves as an outer loop to iteratively select new sets of manipulated variables moves, while an differential equation solver is used to integrate the dynamic equations at each iteration of optimization [11, 1]. This is known as sequential solution because the optimization and integration problems are solved iteratively until the desired accuracy is obtained.

In general, a major part of the optimization calculations involve the evaluation of a function and its derivatives of one kind or the other. The accuracy and speed, with which such derivatives are calculated, is crucial to the success and speedy convergence of the calculation procedure. The gradient information were previously estimated using numerical methods like finite differences but such estimates are subject to truncation error when the differencing intervals are numerically large, and subject to round-off error when they are small. In addition, the run time is unacceptably high, particularly for problems with a large number of independent variables [8]. Automatic or algorithmic differentiation (AD) find the derivatives of a function given in the form of a computer code using the chain rule with no truncation error for any selected outputs with respect to selected inputs with about the same accuracy and efficiency as the function value themselves [10].

In this work, a new NMPC algorithm based a nonlinear least square optimization problem is proposed. The nonlinear least square problem can be efficiently solved by only using residual Jacobian matrix. An efficient algorithm is derived to calculate the residual Jacobian matrix directly from the model sensitivity function without extra integrations. The sensitivity functions can be accurately and efficiently obtained from the state trajectory by using AD techniques. These three features make the new algorithm computationally efficient. To demonstrate the new algorithm, a benchmark example of an industrial evaporator [15] has been chosen as a case study. The process is a multi-input multi-output nonlinear system with a number of constraints. The process is open-loop unstable due to integrating characteristics of the liquid level in the evaporator separator. Effective control of this system using traditional PID controllers was not very successful especially for large setpoint changes. Predictive control was also considered by a number of workers [12, 15]. However, linear MPC was failed to control this process for both the regulating and tracking problems. A NMPC strategy based on a successive linearization solution to control this process under a large setpoint-change condition was proposed in [12]. A good performance was observed after re-linearizing the nonlinear process model for every few steps. However, disturbances have not been considered there. In this work, the new NMPC strategy is applied to control the process for both setpoint tracking and disturbance rejection. The paper is organized as follows: a new NMPC algorithm based on nonlinear least square optimization and using model sensitivity functions to get the residual Jacobian matrix is proposed in section 2. The AD tool applied to get the sensitivity function is described in section 3. Section 4 provides the evaporation process case study to demonstrate the use of the new NMPC algorithm. The paper is concluded in section 5.

2 Nonlinear Model Predictive Control

2.1 Nonlinear least square problem

The nonlinear model predictive control considered is to solve the following nonlinear optimization problem at each sampling point:

$$\min_{\substack{\underline{u}\le u_j\le\overline{u}\\ j=0,\dots,M-1}} \quad \phi=\frac{1}{2}\sum_{k=1}^{P} e_k^T W_k e_k \tag{1}$$

$$\begin{aligned}\text{s.t.}\quad & \dot{x}=f(x,u,d),\ \ t\in\left[t_0,\quad t_P\right] \qquad (2)\\ & x(t_0)=x_0,\ \ x_k:=x(t_k)=x(t_0+kT)\\ & e_k:=x_k-r_k,\ \ k\in\left[1,\quad P\right]\\ & u_j:=u(t_j)=u(t),t\in\left[t_j,\quad t_{j+1}\right]\\ & u_j=u_{M-1},\ \ j\in\left[M,\quad P-1\right]\end{aligned}$$

where T is the sampling period, M and P are control and prediction horizons respectively, $x\in\mathbb{R}^n$, $u\in\mathbb{R}^m$ and $d\in\mathbb{R}^q$ are state, input and disturbance variables respectively, W_k and r_k are the weighting matrix and the reference vector at t_k respectively, $\underline{u}$ and $\overline{u}$ are constant vectors determining the input constraints. The inequalities for input constraints are element-by-element inequalities. Let

$$\begin{aligned} e_k^w &= W_k^{1/2}e_k && (3)\\ E &= [e_1^{wT}\cdots e_P^{wT}]^T && (4)\\ U &= [u_0^T\cdots u_{M-1}^T]^T && (5)\\ \underline{U} &= [\underline{u}^T\cdots\underline{u}^T]^T\in\mathbb{R}^{mM} && (6)\\ \overline{U} &= [\overline{u}^T\cdots\overline{u}^T]^T\in\mathbb{R}^{mM} && (7)\end{aligned}$$

Then the optimization problem of (1) can be restated as a standard nonlinear least square problem:

$$\min_{\underline{U}\le U\le\overline{U}}\phi(U)=\frac{1}{2}E(U)^TE(U) \tag{8}$$

The gradient, $G(U)$ and Hessian matrix, $H(U)$ of (8) have a special structure:

$$\begin{aligned} G(U) &= J^T(U)E(U) && (9)\\ H(U) &= J^T(U)J(U)+Q(U) && (10)\end{aligned}$$

where $J(U)\in\mathbb{R}^{nP\times mM}$ is the Jacobian matrix of residual vector, $E(U)$ and $Q(U)$ is defined as:

$$Q(U)=\sum_{i=1}^{nP}E_i(U)H_i(U) \tag{11}$$

where E_i is the ith element of E and H_i is the Hessian matrix of E_i. The matrix $Q(U)$ has the property that when U approaching optimal solution, residual $\|E(U)\|$ tends to zero then also $Q(U)$ tends to zero. Therefore, an efficient algorithm can be applied to solve this problem [13, 5].

2.2 Jacobian matrix calculation

To efficiently solve the nonlinear least square problem, the Jacobian matrix of E is to be derived in this section. The $nP\times mM$ Jacobian matrix is defined as:

$$J(U)=\left[\frac{\partial E_i}{\partial U_j}\right],\ \ i\in[1,nP],\ \ j\in[1,mM] \tag{12}$$

The Jacobian matrix can be partitioned into $P\times M$ blocks as

$$J(U)=[J_{i,j}],\ \ i\in[1,P],\ \ j\in[1,M] \tag{13}$$

where each block is an $n\times m$ matrix defined as,

$$J_{i,j}=\frac{\partial e_i^W}{\partial u_{j-1}}=W_i^{1/2}\frac{\partial e_i}{\partial u_{j-1}}=W_i^{1/2}\frac{\partial x_i}{\partial u_{j-1}} \tag{14}$$

Since a future input cannot have an effect on a past state, $J_{i,j}=0$ for $i<j$, *i.e.* the Jacobian matrix is low block-triangular.

To solve equation (2) requires information of future disturbances. Since the future disturbances are unknown in many cases, it is assumed in the following calculation that future disturbances are constant as current measurements if disturbances are measured, or as their nominal values if they are not measurable. Taking partial derivative on both sides of (2) gives:

$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial x}{\partial u_{j-1}} = \frac{\partial \dot{x}}{\partial u_{j-1}} = f_x \frac{\partial x}{\partial u_{j-1}} + f_u \frac{\partial u}{\partial u_{j-1}} \tag{15}$$

Equation (15) is a linear time-varying system with initial condition, $\partial x(t_0)/\partial u_{j-1} = 0$. For $j < M$, the input, $\partial u(t)/\partial u_{j-1}$ is an impulse function:

$$\frac{\partial u(t)}{\partial u_{j-1}} = \begin{cases} I, & t \in [t_{j-1}, t_j] \\ 0, & \text{otherwise} \end{cases} \tag{16}$$

For $j = M$, the input is a step function,

$$\frac{\partial u}{\partial u_{M-1}} = \begin{cases} I, & t \geq t_{M-1} \\ 0 & \text{otherwise} \end{cases} \tag{17}$$

Generally, the linear time-varying equation (15) has no analytical solution although it can be represented in a state-transition matrix form [4]. Numerically, equation (15) can be solved together with the state equation (2) using a differential equation solver. The total number of differential variables to be solved in (15) is $n \times m \times M$.

Equation (15) can be simplified by assuming that the sensitivity functions, f_x and f_u are piecewise constant. In this case, within a sampling period, (15) is a linear time-invariant system. Hence, analytical solutions can be obtained. Let $A_k = e^{f_x(t_k)T}$ and $B_k = \int_0^T e^{f_x(t_k)\tau}\,\mathrm{d}\tau f_u(t_k)$, $z_{i,j} = \partial x_i/\partial u_{j-1}$ and $v_{i,j} = \partial u_i/\partial u_{j-1}$, then equation (15) can be discretized as

$$z_{i,j} = A_{i-1}z_{i-1,j} + B_{i-1}v_{i-1,j} \tag{18}$$

For $j < M$, $v_{i-1,j} = 0$ if $i > j$ and $v_{i-1,j} = I$, $z_{i-1,j} = 0$, if $i = j$. Thus,

$$z_{i,j} = \begin{cases} A_{i-1}z_{i-1,j} & \text{for } i > j \\ B_{j-1} & \text{for } i = j \end{cases} \tag{19}$$

Recursively, the following solution can be derived:

$$z_{i,j} = A_{i-1}A_{i-2}\cdots A_j B_{j-1} \tag{20}$$

For $j = M$, $v_{i-1,j} = I$ if $i \geq j$. Denote $\Phi(i,k) = A_i A_{i-1} \cdots A_k$, and $\Phi(i-1,i) = I$, then the solution for this case is

$$z_{i,M} = \sum_{k=M}^{i} \Phi(i-1,k)B_{k-1} \tag{21}$$

2.3 Nonlinear model predictive control algorithm

Based on the algorithm to calculate the residual Jacobian matrix, the proposed nonlinear model predictive control algorithm can be described as follows:

1. Collect current process information, such as measurements, measurable disturbances and setpoint changes.
2. Apply nonlinear least square optimization solver to get a promising guess of next control horizon.
3. The solver calls the objective function to calculate the cost corresponds to the control horizon provided.
4. The cost subroutine calls an ordinary differential equation solver to predict the state trajectory based on the control horizon provided.
5. Based on the prediction trajectory obtained, the cost subroutine applies an AD tool to get the sensitivity function of the trajectory.
6. The residual Jacobian matrix is calculated according to the sensitivity function obtained.
7. The residual vector and residual Jacobian matrix is returned to the optimization solver. If the terminal conditions are not satisfied, the solver updates a new control horizon and the procedural is repeated from step 3. Otherwise, the first point of the control horizon is implemented to the process control system and the procedural is repeated from the step 1.

Remark 1 *Most Newton-type dynamic optimization algorithms involve an inverse integration for co-state or adjoint variables in order to get gradient information. In the proposed algorithm, the gradient information is obtained without such integration. After calculating the state trajectory, only algebraic calculations are involved in the procedural to get residual Jacobian matrix. Therefore, efficiency is greatly improved in the proposed algorithm.*

Remark 2 *For small systems, the sensitivity function might be able to be derived analytically. However, for large systems, particularly for a practical application, it is not a trivial task to get analytic derivative functions. Sometime, it is even not possible. Recently AD techniques have emerged to provide help in this aspect. The proposed algorithm provides a link point where an AD tool can be involved to solve the online dynamic optimization problem.*

Remark 3 *For a feasible problem, the convergence of the algorithm is mainly determined by the nonlinear least square solver used. In this work, the solver provided in MATLAB Optimization Toolbox is applied. It is based on the Levenberg-Marquardt method[13, 5], which uses a scalar to control both search direction and magnitude. When the scalar is zero, the search direction is identical to that of the Gauss-Newton method, whilst as the scalar tends to infinity, search direction tends toward a steepest descent direction. Therefore, in most cases, the value of cost function is non-increasing and the algorithm is convergent.*

Remark 4 *Nonlinear least square is a special case of general nonlinear optimization problems. The main advantage of using the nonlinear least square formulation is its efficiency due to the special structure of its gradient (9) and Hessian (10), whilst the disadvantage is that it cannot directly handel hard* *constraints on output variables. A possible solution is to convert the hard constraints to soft ones by using Lagrange multipliers.*

3 Automatic Differentiation

Automatic differentiation (AD) is the generic name for techniques that use the computational representation of a function to produce analytic values for the derivatives [10]. Some techniques produce code for the derivatives at a general point x by manipulating the function code directly. Other techniques keep a record of the computations made during the evaluation of the function at the specific point x and then review this information to produce a set of derivatives at x. AD uses the chain rule based techniques for evaluating the derivatives with respect to the input variables of function defined by a high-level computer language program. It relies on the fact that all computer programs no matter how complicated use a finite set of elementary (unary e.g. exp(.), sin(.), sqrt(.), *etc*, or binary e.g. +, *, /, *etc*) operations as defined by the programming language. The value of the function computed by the program is simply a composition of these elementary functions, with its partial derivatives are known. Then the chain rule of differential calculus is applied over and over again combining these stepwise derivatives to yield the derivatives of the whole program [20, 10].

There are two basic modes of AD: the forward and reverse modes. The 'forward mode', which is called sometimes 'bottom-up algorithm', computes derivatives of intermediate variables appearing in the process of function computation with respect to a single variable synchronically with computation of the values of intermediate variables themselves. The 'reverse mode', which is called sometimes 'backward mode' or 'top-down algorithm', computes the derivatives in the direction reverse to that of computing the function value. More details about these modes can be found in [20, 10].

A variety of tools exist for AD in standard programming languages. These including ADIFOR, Odysee, ADOL-C, and TAMC. A MATLAB AD toolbox, MAD[9], is used in this work.

4 Evaporation Control Case Study

4.1 Evaporator

This case study is based on the forced-circulation evaporator[15], and shown in Figure 1. In this process, a feed stream enters the process at concentration X_1 and temperature T_1, with flow rate F_1. It is mixed with recalculating liquor, which is pumped through the evaporator at a flow rate F_3. The evaporator itself is a heat exchanger, which is heated by steam flowing at a rate F_{100}, with entry temperature T_{100} and pressure P_{100}. The mixture of feed and recalculating liquor boils inside the heat exchanger, and the resulting mixture of vapour and liquid enters a separator where the liquid level is L_2. The operating pressure inside the evaporator is P_2. Most of the liquid from the separator becomes the recalculating liquor; a small proportion of it is drawn off as product, with concentration X_2, at a flow rate F_2 and temperature T_2. The vapour from the separator flows to a condenser at flow rate F_4 and temperature T_3, where it is condensed by being cooled with water flowing at a rate F_{200}, with enter temperature T_{200} and exit temperature T_{201}. Variable names, description, standard steady state values, and engineering units are listed in Table 1 and model equations are given in Appendix A.

The model has 20 variables and 12 equations. Hence, the process has 8 degrees of freedom, four of which are disturbances, *i.e.* inlet flowrate, F_1, composition, X_1 and temperature, T_1, and cooling water inlet temperature, T_{200} and rest are manipulated variables, *i.e.* outlet flowrate, F_2, steam pressure, P_{100}, cooling water flowrate, F_{200} and circulating flowrate, F_3. In the case study, F_3 is controlled manually, *i.e.* set to a constant. All model equation are differentiable, hence automatic differentiation techniques are applicable to this problem.

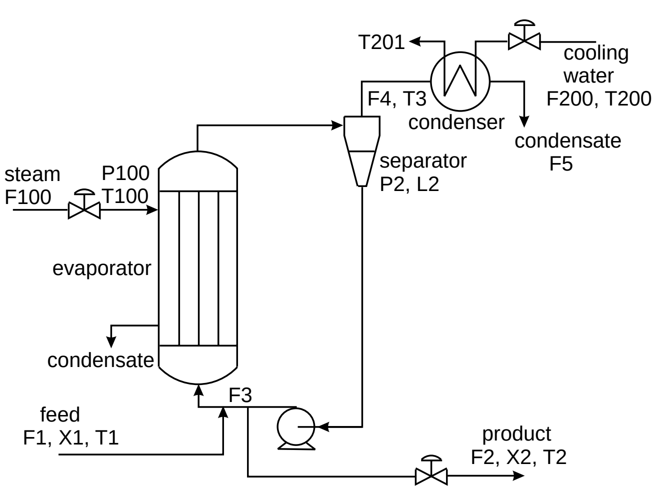

Figure 1: Evaporator System

4.2 Nonlinear model predictive control

The control objective of the case study is to track setpoint changes of X_2 from 25% to 15% and P_2 from 50.5 kPa to 70 kPa when disturbances, F_1, X_1, T_1 and T_{200} are varying within ±20% of their nominal values. The control system is configured with three manipulated variables, F_2, P_{100} and F_{200} and three measurements, L_2, X_2 and P_2. All manipulated variables are subject to a first-order lag with time constant equal to 0.5 min and saturation constraints, $0 \leq F_2 \leq 4$, $0 \leq P_{100} \leq 400$ and $0 \leq F_{200} \leq 400$.

The main performance limitation of the process is the cooling flowrate constraints. At the nominal operation point, the maximum increase rate of P_2 (corresponding to $F_{200} = 0$) is $F_4/4$, which is 2 kPa/min when $X_2 = 25$ % and 1.65 kPa/min when $X_2 = 15$. Therefore, setpoint increase of P_2 is specified within 20 min, *i.e.* increase rate is about 1 kPa/min in order to make the problem feasible.

All disturbances are simulated as a step signal passing through

Table 1: Variables and Optimal Values

Var.	Description	Value	Units
F_1	Feed flowrate	10	kg/mim
F_2	Product flowrate	2	kg/mim
F_3	Circulating flowrate	50	kg/mim
F_4	Vapor flowrate	8	kg/mim
F_5	Condensate flowrate	8	kg/mim
X_1	Feed composition	5	%
X_2	Product composition	25	%
T_1	Feed temperature	40	^{o}C
T_2	Product temperature	84.6	^{o}C
T_3	Vapor temperature	80.6	^{o}C
L_2	Separator level	1	meter
P_2	Operating pressure	50.5	kPa
F_{100}	Steam flowrate	9.3	kg/mim
T_{100}	Steam temperature	119.9	^{o}C
P_{100}	Steam pressure	194.7	kPa
Q_{100}	Heat duty	339	kW
F_{200}	Cooling water flowrate	208	kg/mim
T_{200}	Inlet C.W. temperature	25	^{o}C
T_{201}	Outlet C.W. temperature	46.1	^{o}C
Q_{200}	Condenser duty	307.9	kW

a first-order lag. The amplitudes of step changes are randomly produced within the $\pm 20\%$ range of the nominal values. The changing intervals and time constants of the first-order delays are different for different disturbance variables shown in Table 2.

Table 2: Disturbance model parameters

Disturbance	Interval [min]	Time constant [min]
F_1	5	0.2
X_1	2	0.2
T_1	1	1
T_{200}	1	1

The NMPC is designed with the following parameters: sampling period, $T = 1$ min, $P = 10$ min, $M = 3$ min, $W \equiv \text{diag}[100, 1, 1]$, $\underline{u} = [0, 0, 0]^T$ and $\overline{u} = [4, 400, 400]^T$. It is assumed that F_1 is measurable disturbance whilst other three disturbances are not measured. The nonlinear dynamic model of the process is used as the prediction model for NMPC whilst the actuator lags and disturbances lags are ignored from the prediction. The ignored lags play as process-model mismatches for the NMPC. It has been observed that due to these mismatches, implementing the second point of the control moves calculated is better than using the first one.

Simulation is performed with the above configuration. The results are shown in Figure 2. It can be seen from Figure 2 that measured outputs follow the setpoints quite well (a)–(c) in spite of the existence of severe disturbances (g)–(j). This is achieved without violating the input constraints (d)–(f). Therefore, the NMPC controller is effective and satisfies the performance requirements proposed.

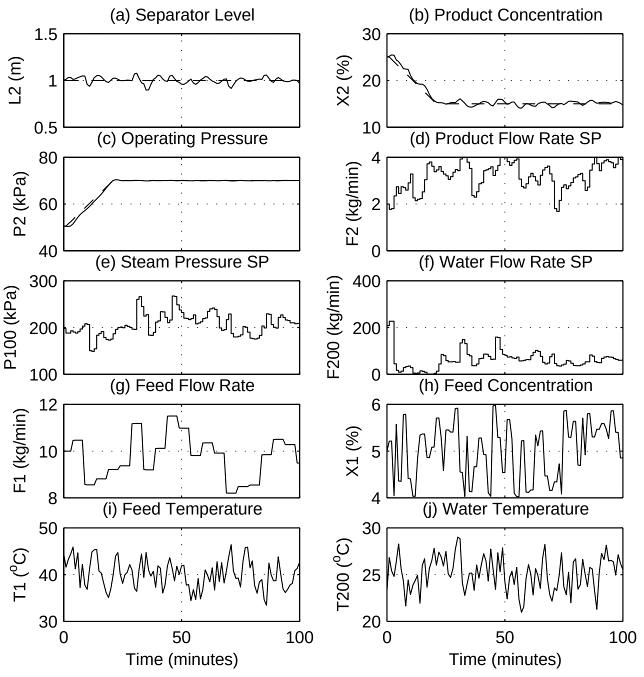

Figure 2: Simulation result. (a)–(c) Measured outputs with setpoints. (d)–(f) Manipulated variables. (g)–(j) Disturbances.

It is worth to mentioned that in the problem formulation, input constraints are explicitly specified as hard constraints, therefore they are not included in the cost function. Constraints on input change rate are also ignored in the objective function because the actuator lags in the model are thought to be sufficient to represent such physical limitation. Since the objective function only contains control errors, the simulation results are almost the bets performance achievable within the hard input constraints.

5 Conclusions

A new algorithm for nonlinear model predictive control is proposed. Based on a nonlinear least square optimization problem, an efficient algorithm to calculate the residual Jacobian matrix is derived. With the new approach, the gradient information can be obtained without further integration of the sensitivity differential equations. The new algorithm also provides a link point where recently developed automatic differentiation techniques can be applied to get derivatives (sensitivity functions) accurately and efficiently. The evaporation case study shows that satisfactory performance is obtained with the controller using the new NMPC algorithm.

References

[1] B.W. Bequtee. Nonlinear control of chemical processes: A review. *Ind. Eng. Chem. Res.*, 30:1391–1413, 1991.

[2] L.T. Biegler. Advance in nonlinear programming concepts for process control. *Journal of Process Control,*

8(5):301–311, 1998.

[3] D.D. Brengle and W.D. Seider. Multistep nonlinear predictive controller. *Ind. Eng. Chem. Res.*, 28:1812–1822, 1989.

[4] C.T. Chen. *Linear System Theory and Design*. Oxford University Press, New York, third edition, 1999.

[5] Jr. Dennis, J.E. Nonlinear least squares. In D. Jacobs, editor, *State of the Art in Numerical Analysis*, pages 269–312, York, England, 1977. Academic Press.

[6] M. Diehl, H. G. Bock, J. P. Schlöder, R. Findeisen, Z. Nagy, and F. Allgöwer. Real-time optimization and nonlinear model predictive control of processes governed by differential-algabric equations. *Journal of Process Control*, 12:577–585, 2002.

[7] J.W. Eaton and J.B. Rawling. Feedback control of nonlinear processes using online optimization techniques. *Comp. Chem. Engng*, 14(4–5):469–479, 1990.

[8] D. Elizondo, B. Cappelaere, and C. Faure. Automatic versus manual model differentiation to compute sensitivities and solve non-linear inverse problems. *Computers and Geosciences*, 28:309–326, 2002.

[9] Shaun Forth. MAD – a MATLAB automatic differentiation toolbox. Eng. System Dep. Cranfield University, 2001.

[10] A. Griewank. *Evaluating Derivatives*. SIAM, Philadelphia, PA, 2000.

[11] M. A. Henson. Nonlinear model predictive control: current states and future directions. *Computer and Chem. Engng.*, 23:187–202, 1998.

[12] J. M. Maciejowski. *Predictive Control with Constraints*. Prentice Hall, Harlow, England, 2002.

[13] D. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *SIAM J. Appl. Math.*, 11:431–441, 1963.

[14] A.M. Morshedi. Universal dynamic matrix control. In *Third International Conference on Chemical Process control*, page 547, New York, 1986.

[15] R.B. Newell and P.L. Lee. *Applied Process Control – A Case Study*. Prentice Hall, Englewood Cliffs, NJ, 1989.

[16] A. Patwardhan, J.B. Rawlings, and T.F. Edgar. Nonlinear model predictive control. *Chem. Engng Comm.*, 87:123–141, 1990.

[17] S. Piche, B. Sayyar-Rodsari, D. Johnson, and M. Gerules. Nonlinear model predictive control using neural networks. *IEEE control system magazine*, 20(3):53–62, 2000.

[18] L.O. Santos, P.A.F.N.A. Afonso, J.A.A.M. Castro, N.M.C. Oliveira, and L.T. Biegler. On-line implementation of nonlinear MPC; an experimental case study. *Control Engineering Practice*, 9(9):847–857, 2001.

[19] P. B. Sistu, R. S. Gopinath, and B. W. Bequette. Computational issues in nonlinear predictive control. *Comput. Chem. Eng.*, 17:361–367, 1993.

[20] A. Verma. An introduction to automatic differentiation. *Current Science*, 78:804–807, 2000.

[21] Y. Zhu and F. Butoyi. Case studies on closed-loop identification for MPC. *Control Engineering Practice*, 10:403–417, 2002.

A Model equations

$$\frac{\mathrm{d}L_2}{\mathrm{d}t} = \frac{F_1 - F_4 - F_2}{20} \tag{22}$$
$$\frac{\mathrm{d}X_2}{\mathrm{d}t} = \frac{F_1X_1 - F_2X_2}{20} \tag{23}$$
$$\frac{\mathrm{d}P_2}{\mathrm{d}t} = \frac{F_4 - F_5}{4} \tag{24}$$
$$T_2 = 0.5616P_2 + 0.3126X_2 + 48.43 \tag{25}$$
$$T_3 = 0.507P_2 + 55.0 \tag{26}$$
$$F_4 = \frac{Q_{100} - 0.07F_1(T_2 - T_1)}{38.5} \tag{27}$$
$$T_{100} = 0.1538P_{100} + 90.0 \tag{28}$$
$$Q_{100} = 0.16(F_1 + F_3)(T_{100} - T_2) \tag{29}$$
$$F_{100} = Q_{100}/36.6 \tag{30}$$
$$Q_{200} = \frac{0.9576F_{200}(T_3 - T_{200})}{0.14F_{200} + 6.84} \tag{31}$$
$$T_{201} = T_{200} + \frac{13.68(T_3 - T_{200})}{0.14F_{200} + 6.84} \tag{32}$$
$$F_5 = \frac{Q_{200}}{38.5} \tag{33}$$