Journal of Process Control, Volume 18, Issue 6, July 2008, Pages 568-581

Nonlinear System Identification for Predictive Control using Continuous Time Recurrent Neural Networks and Automatic Differentiation.

R K Al Seyab Y Cao *

Cranfield University, UK

Keywords: Nonlinear System, System Identification, Predictive Control, Recurrent Neural Network, Automatic Differentiation.

Abstract

In this paper, a continuous time recurrent neural network (CTRNN) is developed to be used in nonlinear model predictive control (NMPC) context. The neural network represented in a general nonlinear state-space form is used to predict the future dynamic behavior of the nonlinear process in real time. An efficient training algorithm for the proposed network is developed using automatic differentiation (AD) techniques. By automatically generating Taylor coefficients, the algorithm not only solves the differentiation equations of the network but also produces the sensitivity for the training problem. The same approach is also used to solve the online optimization problem in the predictive controller. The proposed neural network and the nonlinear predictive controller were tested on an evaporation case study. A good model fitting for the nonlinear plant is obtained using the new method. A comparison with other approaches shows that the new algorithm can considerably reduce network training time and improve solution accuracy. The CTRNN trained is used as an internal model in a predictive controller and results in good performance under different operating conditions.

*To whom correspondence should be addressed (y.cao@cranfield.ac.uk).

1 Introduction

Model Predictive Control (MPC) is proving its continuous success in industrial applications particularly in the presence of constraints and varying operating conditions, thereby allowing processes to operate at the limits of their achievable performance. The basic control strategy in MPC is the selection of a set of future control moves (control horizon) and minimize a cost function based on the desired output trajectory over a prediction horizon with a chosen length. This requires a reasonably accurate internal model, that captures the essential nonlinearities of the process under control, to predict multi-step ahead dynamic behavior [1].

In many reported applications of MPC, a linear model is assumed. However, MPC based on linear models, often results in poor control performance for highly nonlinear processes because of the inadequateness of a linear model to predict dynamic behavior of a nonlinear process. There is therefore, a strong requirement of a good fitting model for NMPC applications.

In many practical applications, a restrict mathematical model based on physical principles is either unknown or too complicated to be used for control. In this case, nonlinear system identification is an inevitable step in a NMPC project. Possibly, it is also the most costly and time consuming part of the project [2]. Therefore, an efficient and effective approach of nonlinear system identification is critical to the success of NMPC.

Unlike linear system identification, there is no uniform way to parameterize general nonlinear dynamic systems. Among existing techniques, the universal approximation properties of neural networks makes them a powerful

tool for modelling nonlinear systems [3]. The structure of neural networks may be classified as feedforward and recurrent. Most of the publications in nonlinear system identification use feedforward neural networks (FFNNs) with backpropagation or its other variations for training, for example [4, 5]. The main drawback of this approach is that it can only provide predictions for a predetermined finite number of steps, in most cases, only one step. This drawback makes such models not well suitable for predictive control, where variable multi-step predictions are desired.

Recurrent neural networks (RNNs) on the other hand are capable of providing long range predictions even in the presence of measurements noise [8]. Therefore, RNN models are better suited for NMPC. RNNs with internal dynamics are adopted in several recent works. Models with such networks are shown [3, 9], to have the capability of capturing various plant nonlinearities. They have been shown more efficient than FFNNs in terms of the number of neurons required to model a dynamic system [10, 11]. In addition, they are more suitable to be represented in state-space format, which is quite commonly used in most control algorithms [12].

In this work, a continuous time version of the recurrent neural networks (CTRNNs) in state-space form is used as the internal model of NMPC. The continuous time RNN brings further advantages and computational efficiency over the discrete formulation even if at the end both are represented on the computer using only discrete values [13]. Using a discrete time RNNs causes a great dependence of the resulting models on the sampling period used in the process and no information is given about the model trajectories between the sampling instants. The sampling period used with CTRNNs, on the other

hand, can be varied without the need for re-training [14, 15].

The main difficulty with recurrent neural networks is their training [13, 16, 17]. Various training strategies have been suggested in the literature, such as the backpropagation method [18], the conjugate gradient method [19], Levenberg-Marquardt optimization [20], or methods based on genetic algorithm (GAs) [21]. To solve the nonlinear optimization problem associated with CTRNN training, the calculation of a large number of dynamic sensitivity equations is required. Depending on the number of sensitivity equations involved, the sensitivity calculation could take more than 90 percent of the total computation time required for solving a training problem. Hence, sensitivity calculation is a bottleneck in training CTRNNs. Ways to find the sensitivity of a dynamic system [22] are: perturbation, sensitivity equations, and adjoint equations. In a perturbation approach, finite difference (FD) is used to approximate derivatives. Hence at least N perturbations to the dynamic system are needed to get the solution of a N-parameter sensitivity problem [22]. Alternatively, sensitivity can also be obtained by simultaneously solving the original ordinary differential equations (ODEs) together with nN sensitivity equations, where n is the number of states [23]. Finally, sensitivity can be calculated by solving n adjoint equations (in reverse direction).

Recently, the automatic differentiation (AD) techniques have been applied to tackle the dynamic optimization problem [24]. In our previous work, [25], a first-order approximation was derived using AD to simplify the dynamic sensitivity equations associated with a NMPC problem so that computation efficiency was improved. In most published work using AD for

dynamic optimization, AD has only been used to generate low (first and/or second) order derivatives. Recently, AD techniques have been used to solve ODEs and sensitivity equations using high-order Taylor series in a NMPC formulation [26]. In this work, the approach of [26] is extended to solve both the CTRNN training and associated NMPC control problems to speed up calculations and to increase efficiency. Both training and NMPC algorithms are applied to an evaporator process [27]. The network training time is significantly reduced by using the new algorithm comparing with other methods. Using the trained CTRNN as its internal model, the NMPC controller gives satisfactory control performance at different operating conditions.

The paper is organized as follows. After the introduction, a CTRNN training algorithm is discussed in section 2. The details of the MPC algorithm are presented in section 3. Section 4 dedicates to the evaporator case study including its nonlinear system identification using CTRNN, the predictive controller design and simulation results. In section 5 some conclusions are drawn from the work.

2 CTRNN Training

2.1 Neural network model

It has been proven that CTRNNs are able to approximate trajectories generated by nonlinear dynamic systems given by:

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

A key to the approximating capabilities of this type of networks is the use of hidden neurons, [10, 28, 29]. There are many types of neural networks from multi-layer perceptrons (MLP) to radial basis functions (RBF), which can be constructed as recurrent networks to approximate the nonlinear system (1). The training algorithm to be discussed is suitable for any kind networks. Hence, the CTRNN to be considered is represented in the following general form.

$$\begin{aligned} \dot{\hat{x}}(t) &= \hat{f}(\hat{x}(t), u(t), \theta) \\ \hat{y}(t) &= C\hat{x}(t) \end{aligned} \tag{2}$$

where $u(t) \in \mathbb{R}^{n_u}$ is the external input, $\hat{y} \in \mathbb{R}^{n_y}$ the network output, $\hat{x} \in \mathbb{R}^{n_{\hat{x}}}$ the network's state vector, $\theta \in \mathbb{R}^{n_\theta}$ the network parameter vector and the

output matrix C is fixed as

$$C = \left[I_{n_y \times n_y}, \emptyset_{n_y \times (n_{\hat{x}} - n_y)} \right] \tag{3}$$

i.e. outputs are the first n_y states of the networks.

A particular example of (2), which will be used for the case study later, is shown in Figure 2, where a MLP network is adopted to construct the recurrent neural network of (2).

2.2 CTRNN sensitivity calculation using AD

The definition of the sensitivity is the variation of the network output against the variation of η, where $\eta \in \mathbb{R}^{n_\eta}$ represents the general parameters, $\eta = \theta$ in training cases, and $\eta = u(t)$ in NMPC, whilst in other cases, η may also include the initial state, $\hat{x}(0)$. Assume the function $\hat{f}$ is d-time continuously differentiable. Then, the sensitivity can be calculated by taking partial derivative for both sides of equations (2):

$$\begin{aligned} \dot{x}_\eta(t) &= f_x x_\eta(t) + f_\eta \\ y_\eta(t) &= C x_\eta(t) \end{aligned} \tag{4}$$

where, $x_\eta := \partial \hat{x} / \partial \eta$, $y_\eta := \partial \hat{y} / \partial \eta$, $f_x := \partial \hat{f} / \partial \hat{x}$, and $f_\eta := \partial \hat{f} / \partial \eta$.

Equation (4) is a linear time-varying system with initial condition, $x_\eta(0) = \partial \hat{x}(0) / \partial \eta$. Generally, system (4) has no analytical solution although it can be represented in a state-transition matrix form [30]. The dynamic sensitivity

function x_η can be calculated using different method as mentioned earlier. Numerically, equation (4) can be solved together with the state equation (2) using a differential equation solver. The total number of differential variables to be solved at each time instant is $n_{\hat{x}} \times (1 + n_\eta)$. Depend on the size of a network, this number of differential variables could growth so large that the calculation causes a significant burden on network training. To tackle this problem, the sensitivity calculation method proposed in [26] is extended for CTRNNs.

To solve differential equations (2) and (4), an integration step has to be determined. Normally, the integration step should be shorter than the sampling period to get accurate results. However, for the approach developed here, the accuracy can be maintained by adjusting the Taylor series order, d. Moreover, for the identification problem, there is no information available between two sampling points to compare integration results if a shorter integration step is adopted. Therefore, for simplicity and efficiency, the integration step is selected to be the same as the sampling period in this work.

Using normalized time, $\tau = t/h$, where h is the sampling period, the right-hand-side of the state equation becomes $z(\hat{x}(\tau), \eta(\tau)) := h\hat{f}(\hat{x}(\tau), \eta(\tau))$ and the solution interval is $0 \leq \tau \leq 1$ for each integration step. Consider $\hat{x}(\tau)$ and $\eta(\tau)$ are given by the truncated Taylor series:

$$\hat{x}(\tau) = \hat{x}_{[0]} + \hat{x}_{[1]}\tau + \cdots + \hat{x}_{[d]}\tau^d \tag{5}$$

$$\eta(\tau) = \eta_{[0]} + \eta_{[1]}\tau + \cdots + \eta_{[s]}\tau^s, \quad s \leq d \tag{6}$$

with coefficients $\hat{x}_{[i]} \in \mathbb{R}^{n_{\hat{x}}}$, and $\eta_{[i]} \in \mathbb{R}^{n_\eta}$ given as follows respectively:

$$\hat{x}_{[i]} = (i!)^{-1} \frac{\partial^i \hat{x}(\tau)}{\partial \tau^i}|_{\tau=0} \tag{7}$$

$$\eta_{[i]} = (i!)^{-1} \frac{\partial^i \eta(\tau)}{\partial \tau^i}|_{\tau=0} \tag{8}$$

Let $v = [\eta_{[0]}^T \cdots \eta_{[s]}^T]^T$, then, $z(\tau) = z(\hat{x}(\tau), v)$ can be expressed by a Taylor expansion:

$$z(\tau) = z_{[0]} + z_{[1]}\tau + \cdots + z_{[d]}\tau^d + \mathscr{O}(\tau^{d+1}) \tag{9}$$

where coefficients $z_{[j]}$ is given as;

$$z_{[j]} = (j!)^{-1} \frac{\partial^j z(\tau)}{\partial \tau^j}|_{\tau=0} \tag{10}$$

From the chain rule, $z_{[j]}$ is uniquely determined by the coefficient vectors, $\hat{x}_{[i]}$ and v with $i \leq j$, *i.e.*

$$z_{[j]} \equiv z_{[j]}(\hat{x}_{[0]}, \hat{x}_{[1]}, \cdots, \hat{x}_{[j]}, v) \tag{11}$$

Nevertheless, inherently, functions $z_{[j]}$ are also d-time continuously differentiable and their derivatives satisfy the identity [31];

$$\frac{\partial z_{[j]}}{\partial \hat{x}_{[i]}} = \frac{\partial z_{[j-i]}}{\partial \hat{x}_{[0]}} := A_{[j-i],\hat{x}} \equiv A_{[j-i],\hat{x}}(\hat{x}_{[0]}, \hat{x}_{[1]}, \cdots, \hat{x}_{[j-i]}, v) \tag{12}$$

$$\frac{\partial z_{[j-i]}}{\partial v} := A_{[j-i],v} \equiv A_{[j-i],v}(\hat{x}_{[0]}, \hat{x}_{[1]}, \cdots, \hat{x}_{[j-i]}, v) \tag{13}$$

where, $A_{[j]x} \in \mathbb{R}^{n_{\hat{x}} \times n_{\hat{x}}}, j = 0, \cdots, d$, and $A_{[j]v} \in \mathbb{R}^{n_{\hat{x}} \times s n_\eta}$, $j = 0, \cdots, d$ are also the Taylor coefficients of the Jacobian path, *i.e.*;

$$\frac{\partial z}{\partial \hat{x}_{[0]}} = A_{[0]x} + A_{[1]x}\tau + \cdots + A_{[d]x}\tau^d + \mathscr{O}\tau^{d+1} \tag{14}$$

$$\frac{\partial z}{\partial v} = A_{[0]v} + A_{[1]v}\tau + \cdots + A_{[d]v}\tau^d + \mathscr{O}\tau^{d+1} \tag{15}$$

AD techniques provide an efficient way to calculate these coefficients vectors, $z_{[j]}$ and matrices $A_{[i]}$ [32]. For example, with the software package, ADOL-C [33, 34], using the forward mode of AD all Taylor coefficient vectors for a given degree, d can be calculated simultaneously, whilst the matrices, $A_{[i]}$ can be obtained using the reverse mode of AD. The run time and memory requirement associated with these calculations grow only as d^2.

Using AD for the CTRNN system (2), the Taylor coefficients of $\hat{x}(\tau)$ can be iteratively determined from $\hat{x}_{[0]}$ and v [26]:

$$\hat{x}_{[k+1]} = \frac{1}{k+1} z_{[k]}(\hat{x}_{[0]}, \cdots, \hat{x}_{[k]}, v), \quad k = 0, \cdots, d-1 \tag{16}$$

$$\hat{y}_{[k]} = C\hat{x}_{[k]}, \quad k = 0, \cdots, d \tag{17}$$

Then, by applying AD to (16), the partial derivatives are obtained and partitioned as;

$$A_{[k]} = \begin{bmatrix} A_{[k]x} & | & A_{[k]v} \end{bmatrix} := \begin{bmatrix} \frac{\partial z_{[k]}}{\partial \hat{x}_{[0]}} & | & \frac{\partial z_{[k]}}{\partial v} \end{bmatrix}, \tag{18}$$

The total derivatives are accumulated from these partial derivatives as fol-

lows:

$$\begin{aligned} B_{[k]} &= \left[B_{[k]x} \;\; | \;\; B_{[k]v} \right] := \left[\frac{d\hat{x}_{[k]}}{d\hat{x}_{[0]}} \;\; | \;\; \frac{d\hat{x}_{[k]}}{dv} \right] \\ &= \frac{1}{k} \left(\; A_{[k-1]} + \sum_{j=1}^{k-1} A_{[k-j-1]x} B_{[j]} \; \right), \qquad k = 1, \cdots, d \end{aligned} \tag{19}$$

Note, $B_{[0]} = \left[\; I \;\; | \;\; B_{[0]v} \right]$, where $B_{[0]v} := \partial \hat{x}_{[0]} / \partial v$. In summary, the solutions of system (2) at $t = h$ are;

$$\hat{x}(h) = \sum_{i=0}^{d} \hat{x}_{[i]}, \qquad\qquad \hat{y}(h) = C\hat{x}(h) \tag{20}$$

whilst their sensitivities to initial value, $\hat{x}_{[0]}$ and coefficients v are,

$$B_x(h) := \frac{d\hat{x}(h)}{d\hat{x}_{[0]}} = \sum_{i=0}^{d} B_{[i]x} = I + \sum_{i=1}^{d} B_{[i]x} \tag{21}$$

$$B_v(h) := \frac{d\hat{x}(h)}{dv} = \sum_{i=0}^{d} B_{[i]v} = B_{[0]v} + \sum_{i=1}^{d} B_{[i]v} \tag{22}$$

$$D_x(h) := \frac{d\hat{y}(h)}{d\hat{x}_{[0]}} = CB_x(h) \tag{23}$$

$$D_v(h) := \frac{d\hat{y}(h)}{dv} = CB_v(h) \tag{24}$$

2.3 Network training algorithm

Training produces the optimal connection weights for the networks by minimizing a quadratic cost function of the errors between the neural network output and the plant output over the entire set of samples. Among many network training algorithms, Levenberge-Marquardt (LM) algorithm [20] is

known to be a robust and fast gradient method because of its second-order converging speed without having to compute the Hessian matrix. For this reason, the LM algorithm is combined with the sensitivity algorithm using AD described above for the dynamic network training.

Firstly, assume the dynamic system (1) is initially at steady-state. By introducing a set of random inputs to the system, the outputs of the plant are collected with the inputs for N sampling points at sampling rate h. Then, the unknown network parameters θ are estimated from the input-output data set by minimizing the sum of squared approximation errors, i.e.

$$\min_{\theta} \ \Phi = \min_{\theta} \ \frac{1}{2} \ \sum_{i=0}^{N} e_i^T e_i \tag{25}$$

where, e_i is the error between the actual plant output and the network output at i-th sampling point which is a function of the model parameter vector given by:

$$e_i \equiv e_i(\theta) = \hat{y}(t_i, \theta) - y(t_i), \quad i = 1, 2, \cdots, N \tag{26}$$

Let:

$$E(\theta) = \begin{bmatrix} e_1^T & \cdots & e_N^T \end{bmatrix}^T \tag{27}$$

The $n_y N \times n_\theta$ Jacobian matrix of E is defined as

$$J(\theta) := \frac{\partial E(\theta)}{\partial \theta} \tag{28}$$

Then, the gradient of Φ is $J(\theta)E(\theta)$, whilst the Hessian of Φ can be approximated as $J^T(\theta)J(\theta)$. The training algorithm based on the nonlinear least square approach of Levenberg Marquandt [20] is:

$$\theta_{k+1} = \theta_k - \left[J(\theta)^T J(\theta) + \mu I\right]^{-1} J(\theta)^T E(\theta) \tag{29}$$

where, θ_{k+1} is an updated vector of weights and biases, θ_k the current weights and biases, and I the identity matrix. When the scalar μ is zero, this is a quasi-Newton approach, using the approximate Hessian matrix, $J^T J$. When μ is large, it is equivalent to a gradient descent method with a small step size. Quasi-Newton method is faster and more efficient when Φ is near the error minimum. In this way, the performance function Φ will always be reduced at each iteration of the algorithm.

The Jacobian matrix can be partitioned into N blokes as:

$$J(\theta) = [J_1^T(\theta) \cdots J_N^T(\theta)]^T \tag{30}$$

where each block is an $n_y \times n_\theta$ matrix as:

$$J_i(\theta) = \frac{\partial e_i(\theta)}{\partial \theta} = \frac{\partial \hat{y}(t_i, \theta)}{\partial \theta} = C\frac{\partial \hat{x}(t_i, \theta)}{\partial \theta} \tag{31}$$

For accurate and fast calculation of the sensitivity equations required for the Jacobian matrix above, the method described in the previous section is adopted here. Since θ is a constant vector, $v = \theta$.

For given v, $\hat{x}(k+1) := \hat{x}(t_{k+1})$, and $\hat{y}(k) := \hat{y}(t_k)$ are iteratively determined from $\hat{x}(0) = [y^T(0), 0_{1\times(n_{\hat{x}}-n_y)}]^T$ using (20). Then the value of

$J_k(\theta) = d\hat{y}(k)/dv$ can be calculated using (19) and (21) – (24) as:

$$
\begin{aligned}
B_v(0) &= 0 = B_{[0]v}(0) && (32)\\
B_x(k) &= I + \sum_{i=1}^{d} B_{[i]x}(k-1) && (33)\\
B_v(k) &= B_v(k-1) + \sum_{i=1}^{d} B_{[i]v}(k-1) = B_{[0]v}(k) && (34)\\
D_v(k) &= CB_v(k) = J_k(\theta) && (35)
\end{aligned}
$$

Hence, with AD, the nonlinear training problem can be efficiently solved using the LM method.

2.4 Model Validation

Many model validity tests for nonlinear models have been developed [35], for example, the Akaike information criterion (AIC), the statistical χ^2 tests, the predicted squared error criterion, and the higher–order correlation tests.

The most common method of validation is to investigate the residual (prediction errors) by cross validation on a *test data set*. Here, validation is done by carrying out a number of tests on correlation functions, including autocorrelation function of the residual and cross-correlation function between controls and residuals. If the identified model based on CTRNN is adequate, the prediction errors should satisfy the following conditions of

high-order correlation tests [36]:

$$R_{ee}(\tau) = E[e(t-\tau)e(t)] = \delta(\tau), \quad \forall\tau \tag{36}$$

$$R_{ue}(\tau) = E[u(t-\tau)e(t)] = 0, \quad \forall\tau \tag{37}$$

where $R_{xz}(\tau)$ indicates the cross-correlation function between $x(t)$ and $z(t)$, e is the model residual. These tests look into the cross-correlation amongst model residuals and inputs. These test are normalized to be within a range of ± 1 so that the tests are independent of signal amplitude and easy to interpret [36]. The significance of the correlation between variables is indicated by a confidence interval. For a sufficiently large data set with length N, the 95% confidence bounds are approximately $\pm 1.96/\sqrt{N}$. If these correlation tests are satisfied (within the confidence limits) then the model residuals are a random sequence and are not predictable from the model inputs. This provides additional evidence of the validity of the identified model.

3 Nonlinear predictive control algorithm

Once the CTRNN has been trained, the network can be used as an internal model of a predictive controller. The recurrent neural network generates prediction of future process outputs over a specified prediction horizon P,

which allows the following performance criterion to be minimized:

$$\begin{aligned}
\min_{\substack{\underline{u}\le u_k\le \overline{u}\\ k=0,\dots,M-1}} \varphi &= \frac{1}{2}\sum_{k=1}^{P} e_{y,k}^T Q e_{y,k} + \sum_{k=1}^{M} \Delta u_k^T R \Delta u_k && (38)\\
\text{s.t. } \dot{\hat{x}}(t) &= \hat{f}(\hat{x}(t), u(t)), \quad t \in [t_0, t_P] && (39)\\
\hat{y}(t) &= C\hat{x}(t) + d(t) && (40)\\
\hat{x}(t_0) &= \hat{x}_0, \quad \hat{x}_k := \hat{x}(t_0 + kh)\\
u_k &:= u(t_k) = u(t), \quad t \in [t_k, t_{k+1}]\\
e_{y,k} &:= \hat{y}_k - r_k, \quad k \in [1, P]\\
\Delta u_k &:= u_{k+1} - u_k, \quad k \in [1, M]\\
u_k &= u_{M-1}, \quad k \in [M, P-1]
\end{aligned}$$

where, M and P are the control and prediction horizons respectively, $Q \in \mathbb{R}^{n_y \times n_y}$ and $R \in \mathbb{R}^{n_{\hat{x}} \times n_{\hat{x}}}$ are the weighting matrices for the output error and the control signal changes respectively, $r_k \in \mathbb{R}^{n_y}$ is the output reference vector at t_k, d is a virtual disturbance estimated at the current time and used to reduce the model-plant mismatch, $\overline{u}$ and $\underline{u}$ are constant vectors determining the input constraints as element-by-element inequalities.

The prediction horizon $[t_0, t_P]$ is divided into P intervals, $t_0, t_1, \cdots, t_P$ with $t_{i+1} = t_i + h_i$ and $\sum_{i=0}^{P-1} h_i = t_P - t_0$. For piecewise constant control, assume the optimal solution to (38) is $u(t) \equiv u(t_k) = u_{[0]}(k)$ for $t_k \le t \le t_{k+1}$, $k = 0, \cdots, P-1$. Then, only the solution in the first interval is to be implemented and whole procedure will be repeated at next sampling instant.

Let $v \in \mathbb{R}^{M \times n_u}$ be defined as $v := [u_{[0]}^T(0) \cdots u_{[0]}^T(M-1)]^T$. Problem (38) is a standard nonlinear programming problem (NLP) which can be solved

by any modern NLP solvers. To efficiently solve the online optimization problem of the predictive controller the same gradient calculation strategy of the NMPC approach proposed by [26] is used.

A simple method is used to estimate the initial value of the model states required to solve the optimization problem at each sample time. In this method, the new states are updated from the old values using the dynamic equation (39). Also, the state estimate error was reduced further by adding the virtual disturbance d to the output. No terminal penalty is used in this work and a good tuning of h, P, M, Q, and R was found enough to ensure the close-loop stability for the case study in different operation conditions.

4 Case Study – An Evaporator Process

This case study is based on the forced-circulation evaporator described by Newell and Lee [27], 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 recirculating 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 rate F_{100} with entry temperature T_{100} and pressure P_{100}. The mixture of feed and recirculating liquor boil 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 recirculating 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 into a condenser

at flow rate F_4 and temperature T_3, where it is condensed by cooling water flowing at rate F_{200}, with entry temperature T_{200} and exist temperature T_{201}. The nominal values of the system variables are given in Table 1, while the first-principle model equations are available in [27].

4.1 System identification using CTRNN

The evaporator system has been adopted as a case study for system identification using CTRNN by a group of researchers [15], where a Genetic Algorithm (GA) based approach was used to train the CTRNN as it was believed that "the implementation of gradient-based training algorithms is computationally expensive". However, only a short period (5 minutes) of data with simple input signals (step changes) was used to train the network and another short period (7.5 minutes) was adopted for model validation. In this work, it is to be demonstrated that with the approach developed above the gradient-based algorithm is not computationally expensive any more comparing with the GA based approach since the new training algorithm is able to handle a much longer period (500 minutes) of data with much more complicated input signals (random pulses) for training and validation.

The evaporator is approximated using a continuous-time recurrent MLP

network with one hidden layer as shown in Figure 2:

$$\begin{aligned} x_h(t) &= \sigma_s\left(W_x\hat{x}(t) + W_u u(t) + b_1\right) \\ \dot{\hat{x}}(t) &= W_2 x_h(t) + b_2 \\ \hat{y}(t) &= C\hat{x}(t) \end{aligned} \tag{41}$$

where, $W_x \in \mathbb{R}^{n_h \times n_{\hat{x}}}$, $W_u \in \mathbb{R}^{n_h \times n_u}$, and $W_2 \in \mathbb{R}^{n_{\hat{x}} \times n_h}$ are connection weights, $b_1 \in \mathbb{R}^{n_h}$ and $b_2 \in \mathbb{R}^{n_{\hat{x}}}$ are bias vectors, whilst each element of the vector $\sigma_s(\cdot) \in \mathbb{R}^{n_h}$ represents the sigmoid-*tanh* function as the neural activation function, *i.e.*

$$\sigma_s(n) = \frac{2}{1 + e^{-2n}} - 1 \tag{42}$$

The parameter vector is $\theta = \begin{bmatrix} \text{vec}(W_x)^T & \text{vec}(W_u)^T & b_1^T & \text{vec}(W_2)^T & b_2^T \end{bmatrix}^T \in \mathbb{R}^{n_\theta}$, where $n_\theta = n_{\hat{x}} \times (n_h + 1) + n_h \times (n_{\hat{x}} + n_u + 1)$. The identification scheme assumes that the plant model equations are unknown and the only available information is the input-output data which is generated through various runs of the first principle model of the plant given by [27]. Two different structures of the CTRNN are studied to model the process. The first network (Network 1) was trained with $n_{\hat{x}} = n_y = 3$, and $n_h = 8$ ($n_\theta = 83$), while the second one (Network 2) was trained with $n_{\hat{x}} = 5$ and $n_h = 8$ ($n_\theta = 117$). The training was carried out repetitively over the data collected within a fixed time interval of 500 minutes and sampled at every 0.2 minutes. The inputs training data was a random pulses with a different amplitude and durations

with the range chosen to cover all the region of operation of the plant (see Figure 3). Another set of data at sampling time 0.05 minutes is randomly generated from the plant to be used for network validation. The output data are corrupted with a normally distributed zero mean noise with variance 5% of the steady state values of the output variables. The initial values of the first n_y network states were chosen equal to the steady state values of the simulated plant outputs while the $(n_{\hat{x}} - n_y)$ remains were equal to zeros.

To demonstrate the CTRNN capability for evaporator model approximation, the simulated plant output and the trained neural networks output are compared in figures 4 and 5. A good model fitting is observed for both networks with approximately similar accuracy with the training data. In terms of model validation, Network 1 is better than Network 2 as shown in Figure 5. This means increase the network state dimension does not necessarily improve the model fitting. Sometimes, networks with high order could include undesirable eigenvalues which may induce an unstable or poor performance. Therefore, Network 1 is chosen as the internal model of the predictive controller for accurate and fast online calculations. Also, the validation results show the capability of the network to approximate the simulated plant output with a sampling time less than that used for training, without the need to re-train the network. In fact, this is one of the most important advantages of CTRNNs over discrete-time recurrent networks.

Also, as a confidence test of the resulting model, the correlation–based model validation results for the CTRNN model can be calculated according to equations (36)–(37) and shown in figures 6 and 7 respectively. The dotted lines in each plot are the 95% confidence bounds ($\pm 1.96/\sqrt{500}$). It can be

seen that only a small number of points are outside the bounds. This demonstrates that the model can be considered as being adequate for modelling this plant.

To solve the training problem, a total $n_{\hat{x}} \times N \times n_\theta = 3 \times 2000 \times 83 = 498000$ sensitivity variables have to be calculated in addition to the original 3 ordinary differential equations (ODEs) of Network 1 while for Network 2, the number of sensitivity is $5 \times 2000 \times 117 = 1170000$. To demonstrate the efficiency of the new algorithm, it is compared with the traditional sensitivity equation integrating approach using a typical numerical ODE solver, the MATLAB function *ode15s*.

To compare computation time associated with a given accuracy, a reference solution is produced by using *ode15s* solver and setting the error tolerance to 10^{-14}. Then with four tolerance settings, (10^{-3}, 10^{-6}, 10^{-8}, 10^{-10}) and four different Taylor series orders (3, 6, 8, 10), computation time and accuracy against the reference solutions using two different approaches are compared in Table 2. A third network (Network 3) with new configurations ($n_x = 4$, $n_h = 15$, and sensitivity variables number $= 4 \times 2000 \times 169 = 1352000$) has been trained and the results are given in Table 2 for comparison. Note that the computation time in Table 2 is the time required to calculate the cost function and the sensitivity variables over one optimization iteration whilst the error term in the same table is the maximum absolute error against the reference solution. The table shows that training algorithm using AD perform better than the traditional sensitivity approach in both efficiency and accuracy. It can be seen that the order of Taylor series plays an important role in error control. Increase the order by a few number, the error

would be reduced by a number of orders magnitude without increasing too much computation time. However, using traditional approaches, significant computation time may have to be traded off for a reduction in computation error. A way to determine an appropriate order of Taylor series for a given error tolerant was suggested in [26]. It is worth to mention that a successful training would require thousands of iterations. If the accuracy of ODE solver is lower, it would require even more iterations to get a converged solution. Therefore, the time comparison listed in Table 2 suggests a massive efficiency improvement in network training achieved by the proposed approach.

All tests are done on a Windows XP PC with an Inetl Pentium-4 processor running at 3.0 GHz. Note that, the proposed algorithm is implemented in C using ADOL-C and interfaced to MATLAB via a *mex* warp.

4.2 Evaporator predictive control

Effective control of the evaporator system using traditional PID controllers was not very successful especially for large setpoint changes [27]. Predictive control was also considered by a number of workers. Linear model predictive control (LMPC) was demonstrated to be not sufficient to fully control this process for an excessive range of variation (see control objectives given bellow)[37] (see Figure 8). A nonlinear MPC strategy based on successive linearization solution to control this process under a large setpoint change condition was proposed by Maciejowski [37]. A good performance was observed after re-linearizing the nonlinear process model after every few steps. However, disturbances have not been considered there. In this paper, the

NMPC algorithms described in section 3 is applied to control the process for setpoint tracking and disturbance rejection tests described as follows;

The control objective of the case study is;

1. Track setpoint ramp changes of X_2 from 25% to 15% and P_2 from 50.5 kPa to 70 kPa.

2. Track setpoint changes as above when unmeasured disturbances, F_1, X_1, T_1 and T_{200} are varied within $\pm 20\%$ 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 a time constant equal to 0.5 minutes and saturation constraints, $0 \leq F_2 \leq 4$, $0 \leq P_{100} \leq 400$ and $0 \leq F_{200} \leq 400$.

To tune control horizon M, prediction horizon P, and sampling time h, initially let $P = M = 1$ min, and $h = 1$ min. By varying M (and assuming $P = M$) from 1 to 15 min, a stable performance is obtained which satisfies all control specifications for $1 \leq M \leq 20$ min. When $M \geq 10$ min, the improvement on the system performance is negligible but computation time increases. Therefore $M = 4$ min is selected. The same steps are used to choose a suitable prediction horizon P, a reasonable range from the minimum value ($P = M = 1$) min to $P = 40$ min has been tested. A stable response without any constraints violation is detected within range $1 \leq P$ min. No performance improvement can be observed when $P \geq 7$ min. Therefore $P = 7$ min is chosen to ensure that both the system stability and satisfactory control performance achieved within a reasonable computation time.

The weighting matrix, $Q = \text{diag}(Q_0, \cdots, Q_0)$, where Q_0 is diagonal and

initially set to be the inverse of the output error bounds. After online tuning, the final values are:

$$Q_0 = \begin{bmatrix} 1000 & 0 & 0 \\ 0 & 500 & 0 \\ 0 & 0 & 200 \end{bmatrix} \tag{43}$$

Also, the input weighting matrix $R = \text{diag}(R_0, \cdots, R_0)$, where R_0 is diagonal and set to I.

By using piecewise constant input, the result NLP problem has $n_u \times M = 12$ degrees of freedom. To solve the NLP problem of the NMPC, a total $(n_{\hat{x}} \times P) \times (n_u \times M) = 252$ sensitivity variables have to calculated in addition to original 3 ODEs of the neural network. In this work, the sensitivity equations are solved using the sensitivity algorithm of [26].

4.3 Simulation Results

Simulation results of all tests above are shown in figures 8 and 9. The efficiency and the stability of the proposed CTRNN based NMPC during setpoint ramp test has been proved in contrast with the LMPC [37] as shown in Figure 8. Also, it can be seen from the results given in Figure 9 that measured outputs follow the setpoints quite well without any input constraints violation in spite of the existence of severe unmeasured disturbances.

To test the controller sensitivity to the sampling time, simulations have also been done by varying h from 0.5 min to 2 min. A stable performance without constraints violation at all tests are also obtained. Knowing that

the recurrent neural network (Network 1) which is trained at $h = 0.2$ min is used as the controller internal model at all the above tests.

A detailed stability analysis for nonlinear model predictive control of the evaporator has been done [38], where using the new stability measure developed, a concrete conclusion had been obtained, *i.e.* the NMPC of the evaporator is asymptotically stable around the nominal steady state for any positive definite state weighting matrix, Q. The work also provided a way to calculate the stability region around the nominal steady state. According to [38], it can be shown that the NMPC described in this work is always stable.

5 Conclusion

This paper demonstrates the reliability of artificial neural networks in process control. An efficient algorithm has been proposed to train continuous-time recurrent neural networks to approximate nonlinear dynamic systems so that the trained network can be used as the internal model for a nonlinear predictive controller. The new training algorithm is based on the efficient Levenberge Marquardt method combined with an efficient and accurate tool: automatic differentiation. The dynamic sensitivity equations and the ODEs of the recurrent neural network are solved accurately and simultaneously via AD. Big time saving to solve sensitivity equations with a higher accuracy are observed using the new algorithm compared with a traditional method. Also, the trained networks with a different model orders show the capability to approximated the multivariable nonlinear plant at different sampling time without the need to re-train the networks. The results show that, the choice

of the network order is also very important to get a good model fitting and stable performance. Based on the identified neural network model, a NMPC controller has been developed. The similar strategy that used in the network training has been used to solve the online optimization problem of the predictive controller. The capability of the new nonlinear identification algorithm and NMPC algorithm are demonstrated via an evaporator case study with satisfactory results.

References

[1] Pearson R. K., "Selecting Nonlinear Model Structures for Computer Control: Review", *Journal of Process Control*, vol. 13, pp. 1-26, 2003.

[2] Zhao H., Guiver J., and Sentoni G., "An Identification Approach to Nonlinear State Space Model for Industrial Multivariable Model Predictive Control", *Proceeding of the American Control Conference, Philadelphia, Pennsylvania*, 1998.

[3] Funahashi K. L. and Nakamura Y., "Approximation of Dynamical Systems by Continous time Recurrent Neural Networks", *Neural Networks*, vol. 6, pp. 183 – 192, 1993.

[4] Temeng H., Schenelle P., and McAvoy T., "Model Predictive Control of an Industrial Packed Reactors using Neural Networks", *J. Proc. Control*, vol. 5(1), pp. 19 – 28, 1995.

[5] Tan Y., and Cauwenberghe A., "Nonlinear One Step Ahead Control using Neural Networks:Control Strategy and Stability Design", *Automatica*, vol. 32(12), pp. 1701 – 1706, 1996.

[6] Miller W. T., Sutton R. S., and Werbos P. J., "Neural Networks for Control", MIT Press, Cambridge, MA, 1990.

[7] Narendra K. S. and Parthasarathy K., "Identification and Control of Dynamical Systems using Neural Networks", *IEEE Trans, Neural Networks*, vol. 1(1), pp. 4 – 26, 1990.

[8] Su H. T. and McAvoy T. J., "Artificial Neural Networks for Nonlinear Process Identification and Control", *Nonlinear Process Control*, M. A. Henson and D. E. Seborg (*eds*) Prentic Hall, NJ, pp. 371 – 428, 1997.

[9] Jin L., Nikiforuk P. , Gupta M., "Approximation of Discrete-Time State-space Trajectories using Dynamic Recurrent Neural Networks", *IEEE Transactions on Automatic Control*, vol. 40(7), pp. 1266-1270, 1995.

[10] Delgado A., Kambhampati C., Warwick K., "Dynamic Recurrent Neural Network for System Identification and Control", *IEE Proc. Control Theory Appl.*, vol. 142(4), pp. 307-314. 1995.

[11] Hush D. R. and Horne B. G., "Progress in Supervised Neural Networks", *IEEE Sig. Process, Mag.*, vol. 1, pp. 8 – 39, 1993.

[12] Zamarreno J. M. and Vega P., "State-Space Neural Network, Properties and Application", *Neural Networks*, vol. 11, pp. 1099 – 1112, 1998.

[13] Pearlmutter B. A., "Gradient Calculations for Dynamic Recurrent Neural Networks: A Survey", *IEEE Transactions on Neural Networks*, 1995.

[14] Kambhampati C., Craddock R. J., Tham M., Warwick K., "Inverse Model Control using Recurrent Networks", *Mathematics and Computers in Simulation*, vol. (51), pp. 181-199, 2000.

[15] Kambhampati C., Garces F., Warwick K., "Approximation of Non-autonomouous Dynamic Systems by Continuous Time Recurrent Neural Networks", *Proceeding of the IEEE-INNS-ENNS International Joint Conference on Neural Networks IJCNN 2000*, vol. 1, pp. 64 – 69, 2000.

[16] Kambhampati C., Manchanda S., Delgado A., Green G., Warwick K., Tham M., "The Relative Order and Inverses of Recurrent Networks", *Automatica*, vol. 32(1), pp. 117-123, 1996.

[17] Prasad V., and Bequette B. W., "Nonlinear System Identification and Model Reduction using Artificial Neural Networks", *Computers and Chemical Engineering*, vol. 27, pp. 1741-1754, 2003.

[18] Rumelhart D. E., Hinton G. E., and Williams R. J., "Learning Internal Representiations by Error Propagation", *in Prallel Distributed Processing, D. E. Rumelhart and J. L. MeClelland, Eds.*, Cambrige MA:MIT Press, 1986.

[19] Leonard J. A. and Kramer M. A., "Improvement of the Backpropagation Algorithm for Training Neural Networks", *Computers and Chemical Eng.*, vol. 14, pp. 337 – 341, 1990.

[20] Marquardt D., "An Algorithm for least-Square Estimation of Nonlinear Parameters" *SIAM J. Appl. Math.*, vol. 11 , pp. 431 – 441 , 1963.

[21] Goldberge D. E., "Genetic Algorithms in Search, Optimization and Machine Learning", Reading, MA:Addision-Wesley, 1989.

[22] Storen S., Hertzberg T., "Obtaining Sensitivity Information in Dynamic Optimization Problems Solved by the Sequential Approach", *Computer and Chemical Engineering*, vol. 23, pp. 807 – 819, 1999.

[23] Schlegel M., Marquardt W., Ehrig R., Nowak U., "Sensitivity Analysis of Linearly Implicit Differential-Algebraic Systems by One-step Extrapolation", *Applied Numerical Mathematics*, vol. 48, pp. 83 – 102, 2004.

[24] Griesse R., Walther A., "Evaluating Gradients in Optimal Control: Continuous Adjoint Versus Automatic Differentiation", *Journal of Optimization Theory and Applications*, vol. 122(1), pp. 63 – 86, 2004.

[25] Cao Y., and A-Seyab R., "Nonlinear Model Predictive Control using Automatic Differentiation", *European Control Conference (ECC 2003)*, Cambridge, UK, 2003, p. in CDROM.

[26] Cao Y.,"A Formulation of nNonlinear Model Predictive Control using Automatic Differentiation", *Journal of Process Control*, vol. 15, pp. 851 – 858, 2005.

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

[28] Garces F., Kambhampati C., and Warwick K., "Dynamic Recurrent Neural Networks for Identification of a Multivariable Nonlinear Evaporator Systems", *Proceedings DYCONS'99*, World Scientific, 1999.

[29] Garces F., Kambhampati C., and Warwick K., "Dynamic Recurrent Neural Networks for Feedback Linearization of a Multivariable Nonlinear Evaporator Systems", *In Review for UKACC International Conference Control*, 2000.

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

[31] B. Christianson, "Reverse Accumulation and Accurate Rounding Error Estimatres for Taylor Series", *Optimization Methods and Software*, vol. 1, pp. 81 – 94, 1992.

[32] Griewank A., "Evauating Derivatives", *SIAM, Philadelphia, PA*, 2000.

[33] Griewank A., David J., Jean U., "ADOL-C: A Package for the Automatic Differentiation of Algorithms written in C/C++", *ACM Transactions on Mathematical Software*, vol. 22(2), pp. 131 – 167, 1996.

[34] Griewank A., "ODE Solving via Automatic Differentiation and Rational Prediction", *in: Griffiths D., Watson G. (Eds.), Numerical Analysis 1995, vol. 344 of Pitman Research Nots in Mathematics Series, Addison-Wesley., Reading, MA*, 1995.

[35] Zhang J., and Morris J., "Recurrent neuro–fuzzy networks for nonlinear process modeling", *IEEE Trans. on Neural Networks*, vol. 10(2), pp. 313–326, 1999.

[36] Billings S. A. and Voon W. S. F., "Correlation based model validity tests for nonlinear models", *Int. J. Control*, vol. 44, pp. 235–244, 1986.

[37] Maciejowski J. M., "Predictive control with constraints", *Prentice Hall*, Harlow, England, 2002.

[38] Chen, W.-H. and Cao, Y., "Stability analysis of constrained nonlinear model predictive control with terminal weighting", submitted to *International Journal of Robust and Nonlinear Control*, 2007.

List of Figures

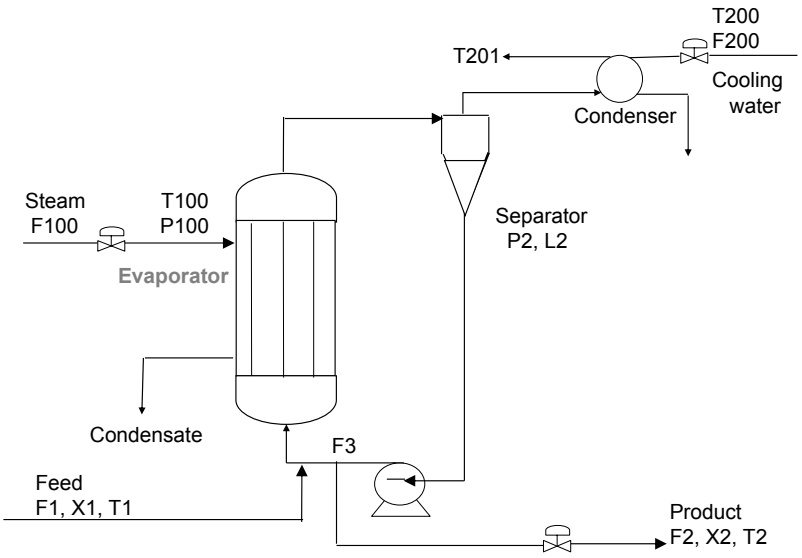

Figure 1: Evaporator System

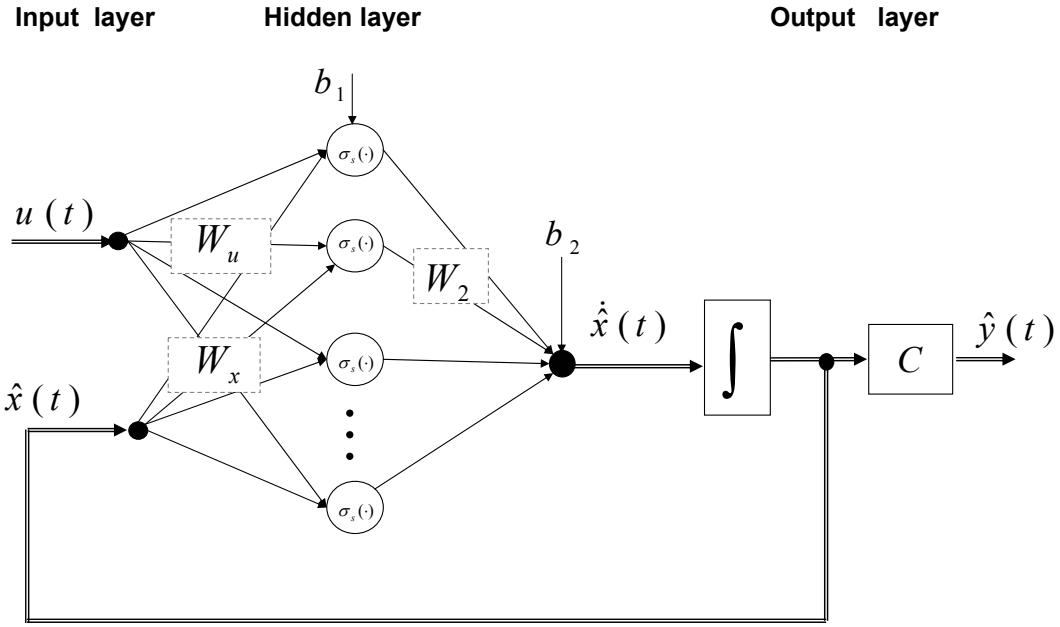

Figure 2: Recurrent Neural Network Structure

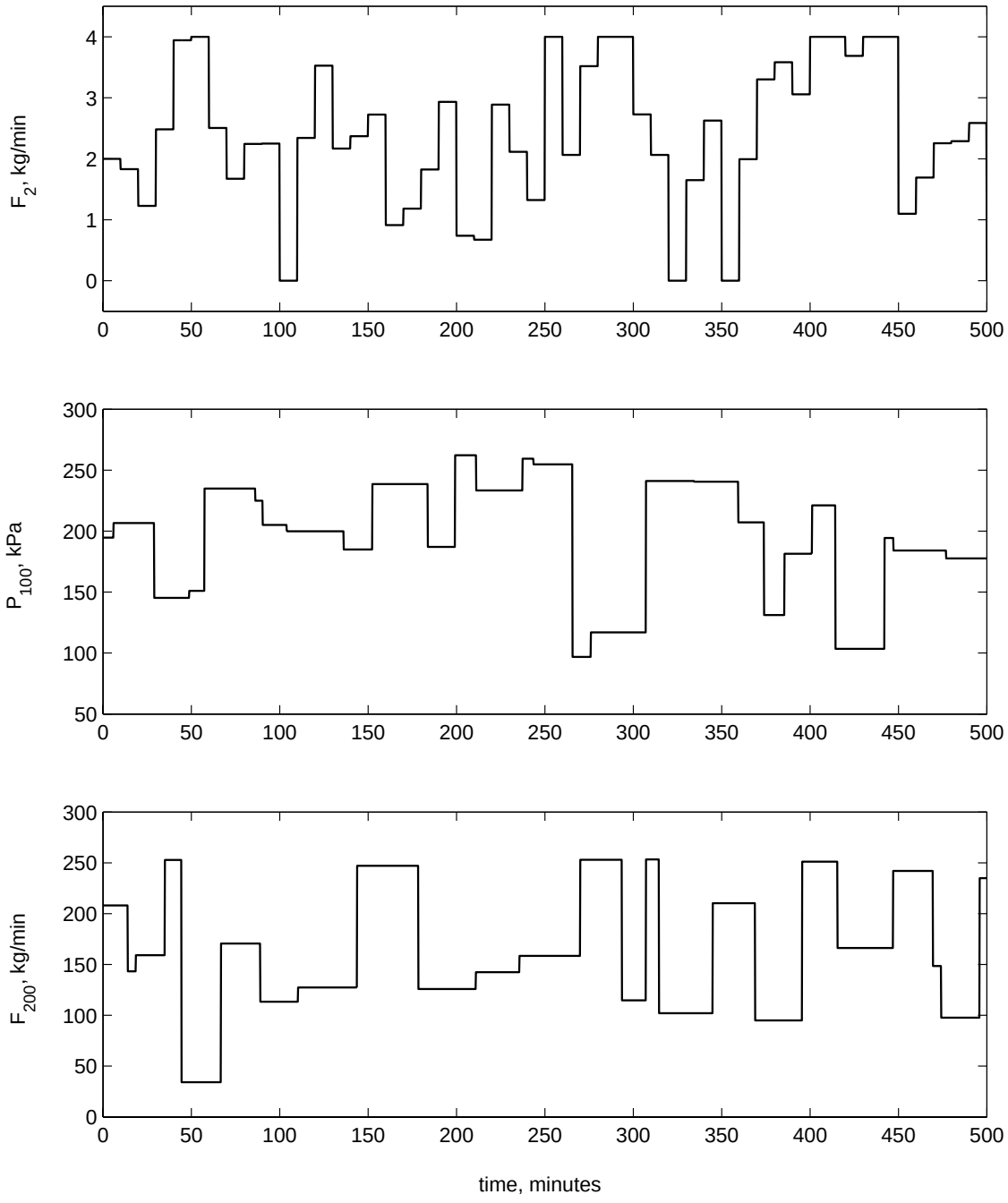

Figure 3: Training data set, Inputs, $\triangle T = 0.2$ min.

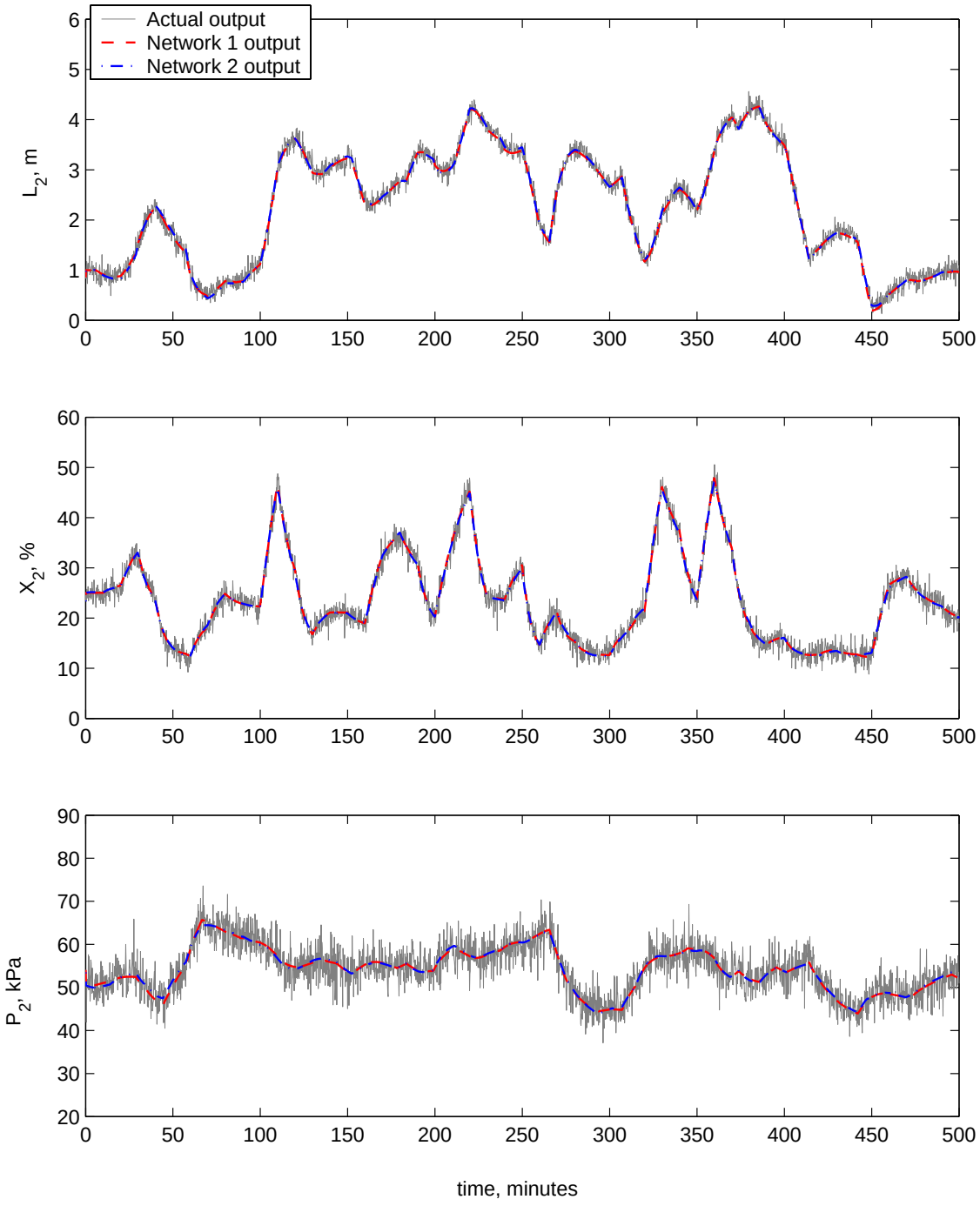

Figure 4: Training data set, Outputs, $\triangle T = 0.2$ min.

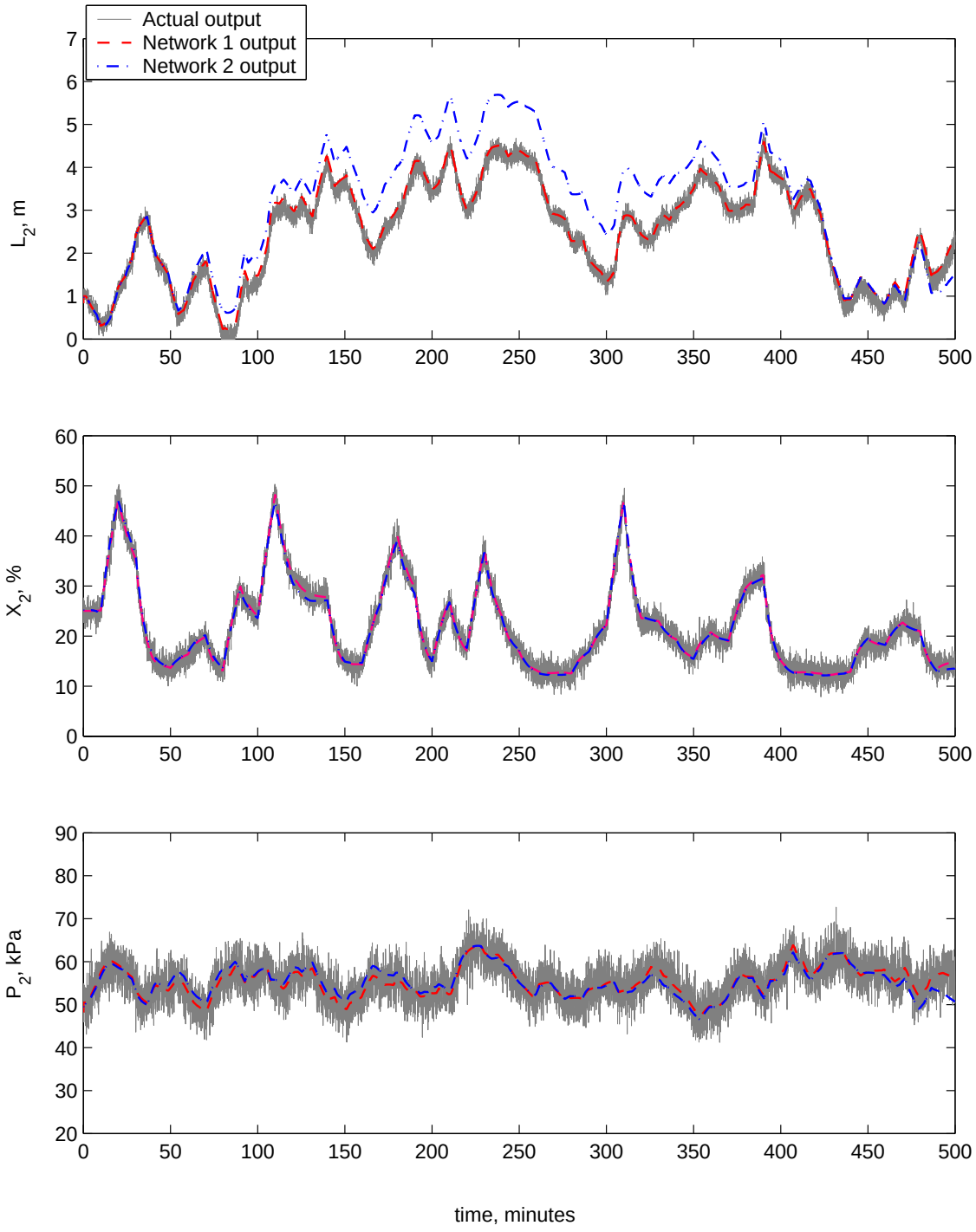

Figure 5: Validation data set, Outputs, $\triangle T = 0.05$ min.

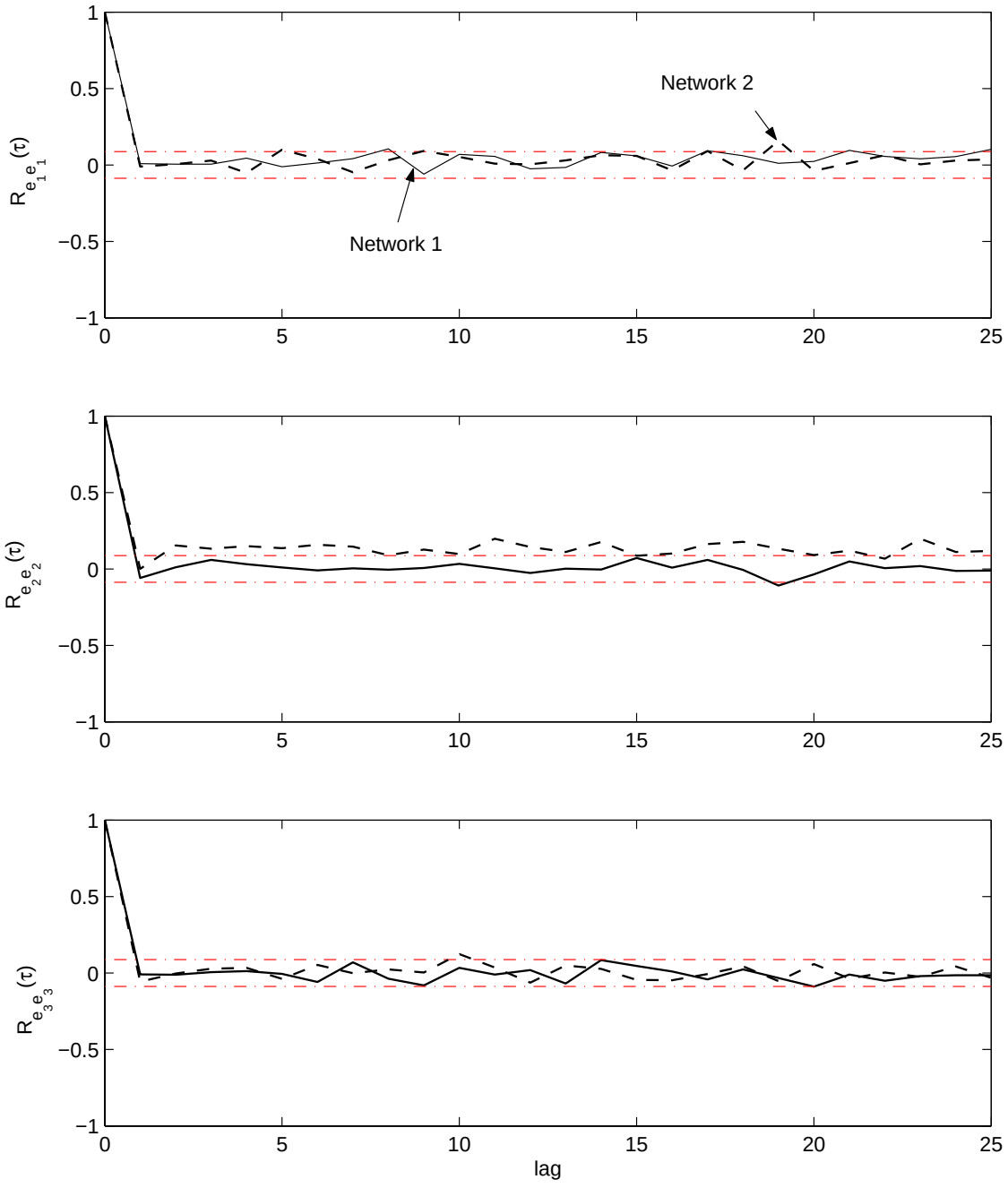

Figure 6: Validation tests/Autocorrelation coeff. for error, $\triangle T = 0.05$ min.

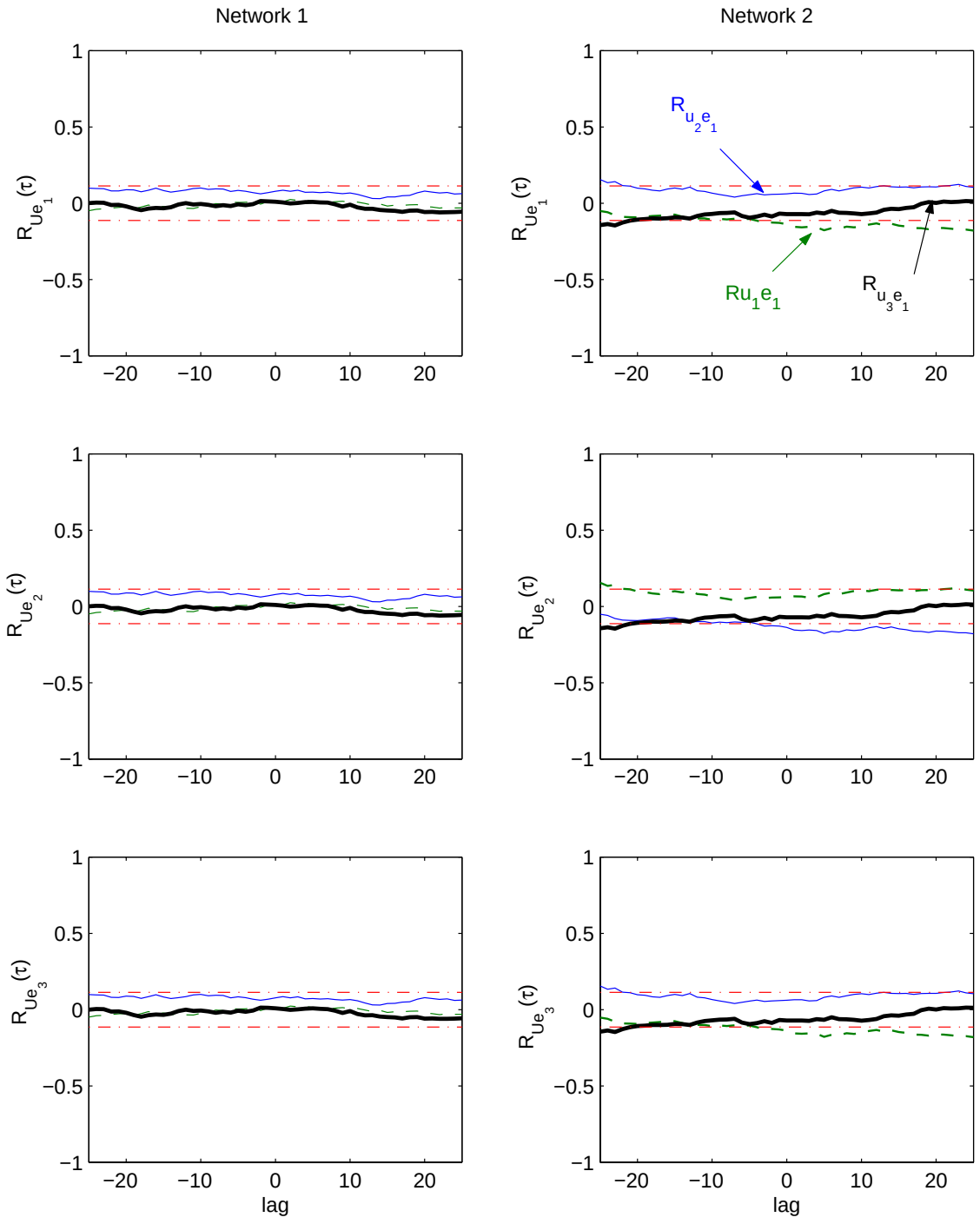

Figure 7: Validation tests/Cross-correlation coef. of U, $\triangle T = 0.05$ min.

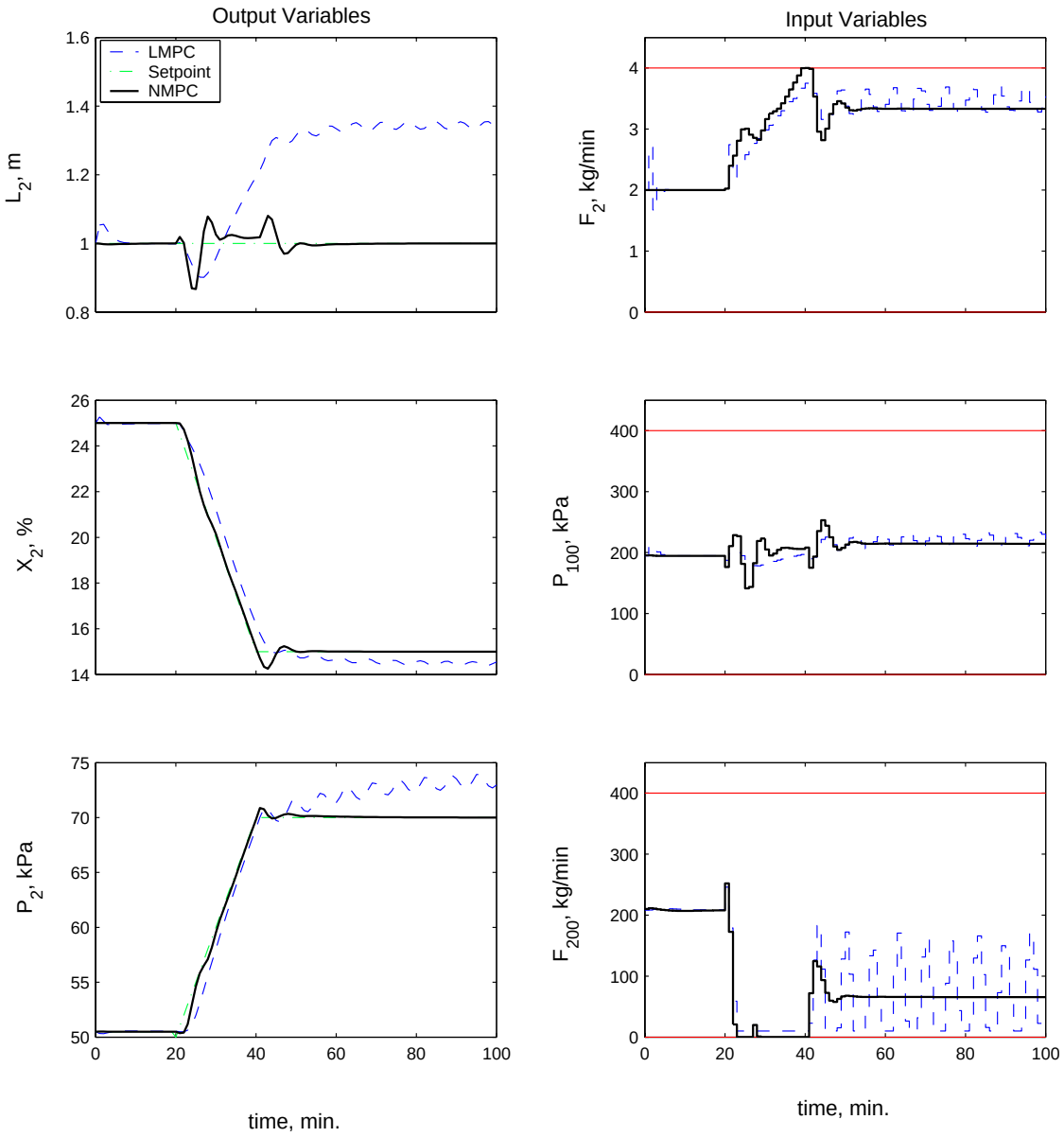

Figure 8: Evaporator performance at setpoints ramp changes using LMPC [37] and the proposed NMPC, $\triangle T = 1$ min.

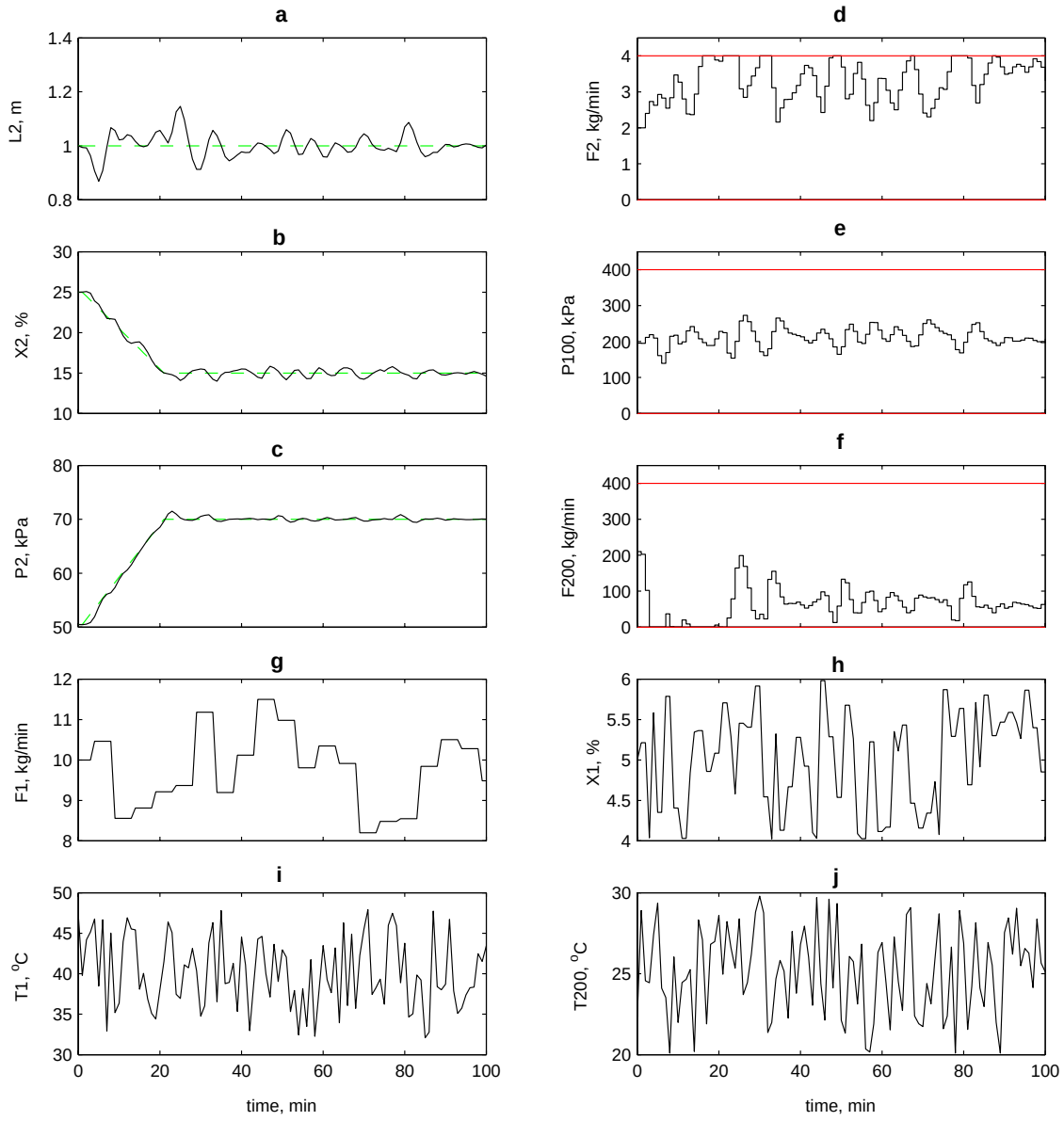

Figure 9: Evaporator performance using the present NMPC at setpoint changes plus random disturbances test. (a)–(c) Measured outputs with setpoints. (d)–(f) Manipulated variables. (g)–(j) Disturbances, $\triangle T = 1$ min.

Table 1: Evaporator Variables and Values

Variables	Description	Nominal value	Units
F_1	Feed flowrate	10	kg/min
F_2	Product flowrate	2.0	kg/min
F_3	Circulating flowrate	50	kg/min
F_4	Vapor flowrate	8.0	kg/min
F_5	Condensate flowrate	8	kg/min
X_1	Feed composition	5.0	%
X_2	Product composition	25	%
T_1	Feed temperature	40.0	%
T_2	Product temperature	84.6	^{o}C
T_3	Vapor temperature	80.6	^{o}C
L_2	Separator level	1.0	m
P_2	Operator pressure	50.5	kPa
F_{100}	Steam flowrate	9.3	kg/min
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/min
T_{200}	Inlet C. W. temperature	25.0	^{o}C
T_{201}	Outlet C. W. temperature	46.1	^{o}C
Q_{200}	Condenser duty	307	kW

Table 2: Computing Time and Accuracy Comparison

Traditional Sensitivity Approach

Tolerance	Network 1		Network 2		Network 3	
	Time, ms	Error	Time, ms	Error	Time, ms	Error
10^{-3}	13.859	0.555	48.328	4.7175	67.641	0.5851
10^{-6}	45.046	0.0153	257.454	0.1351	459.046	0.0135
10^{-8}	69.437	4.0183×10^{-4}	434.547	8.973×10^{-4}	740.688	2.1924×10^{-4}
10^{-10}	77.906	1.1103×10^{-8}	580.125	1.3316×10^{-5}	838.563	9.6209×10^{-9}

ADOL–C Software

Order	Network 1		Network 2		Network 3	
	Time, ms	Error	Time, ms	Error	Time, ms	Error
3	2.609	3.276×10^{-5}	3.11	1.396×10^{-4}	4.157	1.9759×10^{-5}
6	4.031	2.095×10^{-10}	5.281	1.7862×10^{-9}	6.844	7.5623×10^{-9}
8	5.207	1.136×10^{-13}	6.875	1.2301×10^{-12}	9.234	6.7502×10^{-11}
10	6.813	8.881×10^{-16}	8.875	5.6843×10^{-14}	12.609	7.9543×10^{-14}