Implementation of Neural Network-Based Nonlinear Adaptive Model Predictive Control over a Service-Oriented Computer Network

Vincent A. Akpan, Member, IEEE, Ioakeim K. Samaras, and George D. Hassapis, Senior Member, IEEE

Abstract—This paper presents a new neural network-based nonlinear adaptive model predictive control algorithm and its implementation over a service-oriented computer network. The computer network is based on the device profile for web services. At each sampling instant, the algorithm identifies a nonlinear process model using a recurrent neural network. On the basis of the identified model, the nonlinear adaptive model predictive control is updated and the control actions are applied. The network training is performed with data obtained from the prior plant operation under different input disturbances. The proposed nonlinear model identification and nonlinear adaptive model predictive control were applied to the temperature control of a fluidized bed furnace reactor and tested by simulating the reactor operation on the proposed service-oriented computer network. Input step changes and long range output prediction results show good predictive and adaptive control performance. The computation time of the proposed algorithms running on the proposed network architecture was less than the sampling period of the process with a bounded round trip delay. These simulation results indicate that the proposed identification and control algorithms can be of practical use to processes with similar dynamics with the fluidized bed reactor. This is because their realization over a service-oriented computer network which may be the physical platform of their implementation does not introduce delays of such a level that may alter the required sampling time for good control performance.

I. INTRODUCTION

MODEL predictive control (MPC) algorithms have been proven to be very successful in industrial applications over the years [1], [2]. However, improvements are possible if their computations can be reduced with certain simplifications. The majority of the MPC algorithms used in industry are based on a linear mathematical model of the controlled process. These algorithms, although they are better than other conventional algorithms, they are not as much efficient as expected because the characteristics of many industrial applications are highly nonlinear. Methods based on nonlinear models of the processes are preferred but still the application of these methods cannot guarantee stable and accurate control outside the range of the model validity. For this reason adaptive algorithms which would be based on a continuous model updating process and redesign of the MPC strategy before a new control action is applied to the real plant would be the preferred ones. Up to now the development of such algorithms is very much restrained because of their high computation time. However, the recent availability of inexpensive multi-core computers makes us to rethink the possibility of developing adaptive MPC algorithms.

The use of a Multilayer Perceptron (MLP) Recurrent Neural Network (RNN) instead of the actual nonlinear plant model derived from first principles and expressed by ordinary and/or partial differential equations simplifies considerably the computations of adaptive MPC algorithms where data from sensors can readily provide information about the state of the plant.

Due to the fact that service oriented architecture (SOA) [3] has become the state of the art solution for implementing autonomous and interoperable networks the realization of an adaptive predictive control algorithm over such a network would be of practical interest.

The optimization of the RNN model can be based on the Newton’s method [4]. The well-known problems associated with the Newton’s method is that the Hessian matrix occasionally becomes ill-conditioned or singular and is not guaranteed to be positive definite in an open neighbourhood of a minimum.

The aim of this contribution is to present a new RNN-based nonlinear adaptive model predictive control (NAMPC) algorithm which uses the full-Newton method and guarantees the positive definiteness of the second-order Hessian matrix for on-line implementation. The second objective is to demonstrate the on-line implementation of the proposed strategy over a new service-oriented computer network (S-OCN) based on device profile for web services DPWS [6].

The paper is structured as follows: Section II presents the model identification. In Section III the proposed NAMPC is formulated. In Section IV the description of the proposed S-OCN is given. The application of the proposed NAMPC to the temperature control of a fluidized bed furnace reactor and simulation results are presented in Section V followed by conclusions and recommendations in Section VI.

II. Multilayer Perceptron (MLP) Recurrent Neural Network (MLP_RNN)

A. Design of the MLP_RNN

A typical MLP RNN model structure is shown in Fig. 1.
Training this network presents certain difficulties due to the feedback from output to the input. To overcome these difficulties the supervised learning method [7] is used. According to this method first the feedback to the MLP_RNN network is replaced by the actual outputs of the true system. Then the MLP_RNN becomes a feedforward type neural network. Next, the output computed by the MLP_RNN prediction is compared with the prediction of a nonlinear autoregressive (NARX) model [7], [8] of the true system and the error between the predictions of the MLP_RNN and the NARX model is estimated. Then the weights which minimize an index, which is a function of the observed errors and the vector of the parameters of the autoregressive model, are estimated by a NN training method. Given the input-output observations of the true system the MLP NN can be trained as a feedforward NN using our training algorithm proposed in [9]. The above brief description is explained by detailed mathematical notation.

Let the outputs \( y(kT) \) of a multivariable system with an unmeasured output noise \( v(kT) \) at time \( t = kT \) responding to the inputs \( u(kT) \) be given by the following discrete-time transfer function:

\[
y(kT) + A_1 y((k-1)T) + \cdots + A_n y((k-n)T) = B_0 u((k-d)T) + \cdots + B_m u((k-d-m)T) + v(kT) = \sum_{j=1}^{n+m} w_j \cdot u(k-jT) + v(kT)
\]

where \( y(kT) \) is a \( i \times 1 \) vector; \( u(kT) \) is a \( l \times 1 \) vector; \( m \) and \( n \) are the numbers of the past input values and past output values respectively taken over \( NT \) period of time; \( k = 1, 2, \ldots, N \); \( T \) is the sampling period; \( N \) is number of measurements. Introducing the delay operator \( z^{-d} \) with delay \( d = 1 \), and assuming that the system is deterministic, so that making the first few values of \( v(kT) \) non-zeros and setting \( C(z^{-1}) = 1 \), the output noise disturbance model \( \tilde{d}(kT) = C(z^{-1})/A(z^{-1}) v(kT) \) reduces to \( \tilde{d}(kT) = v(kT) \).

So that (1) can be expressed in the following form:

\[
A(z^{-1}) y(kT) = z^{-d} B(z^{-1}) u(kT) + v(kT) = B(z^{-1}) u(kT)
\]

which corresponds to an ARX model [1], [7], [8]; where \( A(z^{-1}) = I + A_1 z^{-1} + \cdots + A_n z^{-n} \) is said to be monic since its first term is an identity matrix and \( B(z^{-1}) = B_0 + B_1 z^{-1} + \cdots + B_m z^{-m} \). Then, the compact form of the one-step ahead model predictor of (2) can be expressed from [8] as:

\[
\hat{y}(kT | \theta) = g[kT, \phi_{\theta_k}(kT, \theta), \theta]
\]

where \( g(\cdot, \cdot) \) is an unknown nonlinear function realized by a NN; \( \theta \) is an unknown vector of \( q \) elements containing the adjustable parameters \( A(z^{-1}) \) and \( B(z^{-1}) \); \( \phi_{\theta_k}(kT, \theta) \) is the regression vector made up from the vectors \( \phi_{\theta_k}(kT) = [u((k-d)T), \ldots, u((k-d-m)T)]^T \) and \( \phi_{\theta_k}(kT) = [y((k-1)T), \ldots, y((k-n)T)]^T \) of the past inputs and outputs respectively; \( v(kT) \) is assumed to be a Gaussian white noise independent of the input. The network output \( \hat{y}(kT) \) in terms of the parameters of Fig. 1 can be expresses as:

\[
\hat{y}(kT) = F_t \left( \sum_{j=1}^{n+m} W_j \cdot f_j(\tilde{b}) + W_{0j} \right)
\]

(4)

where \( j \) is the number of hidden neurons; \( (W_{j,k}) \) and \( W_{0j} \) are the hidden and output weights respectively; \( w_{j,0} \) and \( W_{0j} \) are the hidden and output biases; \( f_\xi(\xi) \) and \( F_\xi(\xi) \) are fast hyperbolic tangent and linear activation functions for the input and output layers respectively; where \( f_\xi(\xi) = 1 - (2/(e^\xi + 1)) \) with \( \xi \) being the activation potential. Here, the term bias is interpreted as a weight acting on an input clamped to 1; the description of weight implies both weights and biases.

B. Formulation of the MLP NN training Problem

Let the true system corresponding to \( \theta = \theta_0 \) be given by

\[
y(kT) = g_0[kT, \phi_{\theta_0}(kT, \theta_0), \theta_0] + v(kT)
\]

(7)

At time \( t = k + 1 \), the \( m \) past inputs and the \( n \) past outputs are available. The predicted model output \( \hat{y}(kT) \) represents the estimate of the true (desired) output. The estimate is subtracted from the true output \( y(kT) \) to produce the error signal \( e(kT) = y(kT) - \hat{y}(kT) \) which is used to adjust the synaptic weights so as to minimize \( e(kT) \). The model identification problem then reduces to searching a parameterized set of model structures \( \hat{\theta} \) to obtain an estimate of \( \theta^* \) through a network training (optimization) method such that \( \hat{\theta} : \theta \rightarrow \theta^* = \theta_0 \).

The MLP NN training can then be formulated as a total square error (TSE) problem which is expressed here by:

\[
J(\theta, Z^N) = \frac{1}{(1/N)} \left( \sum_{k=1}^{N} e(kT) \right)^2 + \theta^T D \theta
\]

(8)

where \( (1/N) \theta^T D \theta \) is a weight decay term [8] with
\[ D = \alpha_s I = [\alpha_1 \alpha_2 I]; \ I \text{ an identity matrix and } \alpha_d \text{ the multiple weight decay parameter: } \alpha_1 \text{ and } \alpha_2 \text{ is for the input-to-hidden and hidden-to-output layer respectively.} \]

Thus, given the input and output measurements pair as:
\[ Z^N = \{ [u(1), y(1), \ldots, u(k), y(k)], k = 1, 2, \ldots, N \}, \]
an estimate of \( \theta^* \) which minimizes (8) can be expressed as:
\[ \theta^* = \arg \min_{\theta} J(\theta, Z^N) \]
which is solved in this work using the adaptive recurrent neural training algorithm (ARNNTA) proposed in [9] (Paper No.: 901 also accepted in this conference).

III. FORMULATION OF THE NEURAL NETWORK-BASED CONSTRAINED NAMPC ALGORITHM

A. The NAMPC Objective Function

The proposed NN-based NAMPC scheme incorporating the NARX model predictor is shown in Fig. 2. A measure of the identified NNARX model accuracy in terms of the modeling error (or residual) is denoted as \( y^*(kT) \). The NAMPC scheme includes also a first-order low-pass digital filter for adjusting the set-points in a way that internal stability and robustness to noise and disturbances is improved. The filter output is given by:
\[ \tilde{\vartheta}(kT) = (B_w(z^{-1})/A_w(z^{-1})) \tilde{n}(kT) \]
where \( \tilde{n}(kT) \) is the desired set-points, \( A_w \) and \( B_w \) are the denominator and numerator polynomials of the filter transfer function, and \( \vartheta(kT) \) the calculated filtered set-points. The reference signal \( R(kT) \) is obtained from:
\[ R(kT) = \vartheta(kT) - \hat{y}(kT) \]
Thus, \( \hat{y}^*(kT) \) should be approximately zero (i.e. \( y(kT) = \hat{y}(kT) \)) to achieve good control system performance.

The idea of the proposed NAMPC strategy is that at each iteration, it minimizes an objective function of the form:
\[ J(kT, U(kT)) = \kappa E(\hat{y}(kT))^2 + \rho U(kT)^2 \]
for a control horizon of \( N_a \) samples, that is:
\[ U(kT) = [u(kT) \ldots u(k + N_a - 1)T]^T \]
subject to the following constraints:
\[ \Delta u((k + \tau)T) = 0, \quad \text{for } N_a \leq \tau \leq N_2 - d \]
\[ y_{\min} \leq \hat{y}(kT) \leq y_{\max}, \quad \text{and } u_{\min} \leq u(kT) \leq u_{\max} \]
where \( \tilde{U}(kT) = [\Delta u(kT) \ldots \Delta u(k + N_a - 1)T]^T \); \( R(kT) = [r((k + N_1)T) \ldots r((k + N_2)T)]^T \)
\[ \hat{y}(k + N_1)T) \]
\[ E(kT) = [r((k + N_1)T) \ldots r((k + N_2)T)]^T \]
\[ \text{and } \hat{y}(k + \eta)T) \]
\[ \text{for } \eta = N_1, \ldots, N_2; \ N_1 \text{ and } \text{N_2 specify the minimum and maximum prediction horizons respectively; here } N_a \text{ and } u(kT) \text{ are the control horizon and control inputs respectively; } \kappa \text{ and } \rho \text{ are two weighting factors penalizing changes in the predicted outputs and the control inputs respectively; and } E(kT) \text{ the prediction error between the desired reference set-point } r(kT) \text{ and predicted output } \hat{y}(kT). \]

B. Extracting the Identified Model and Computing the Output of the System Based on the Model

To compute the output of the system based on the identified model, we assume that: \( N_l = d = 1 \); and Eq. (3) approximates (7), so that \( \theta^* = \theta_0 \) and (3) can be written as:
\[ \hat{y}(kT) = g[kT, \theta_0, (kT, \theta)^T, \theta^*] \]
Also, assuming that the output observation is available up to time \( (k - 1)T \), then the one step-ahead prediction becomes:
\[ \hat{y}(kT | (k - 1)T) = g(\hat{y}(y(k - 1)T), \ldots, y(k - n)T), u(k - d)T), \ldots, u(k - d - m)T) \]
and the \( \eta \)-step ahead prediction is calculated from (16) as:
\[ \hat{y}(k + \eta)T | kT) = g(\hat{y}(k + \eta - 1)T), \ldots, \); \( \hat{y}(k + \eta - \min(\eta, n))T), y(k - 1)T), \ldots, y(k \max(n - \eta, 0))T), \]
\[ \text{In this case, the predictions are substituted for the actual measurements where these do not exist. Next, we calculate (17) based on the (15) in terms of (4), so that (17) becomes:} \]
\[ \hat{y}(k + \eta)T | kT) = \sum_{\eta} W_{j\eta} f_j(\hat{b}(\eta, j)) + W_{j,0} \]
where \( f_j(\chi) \) is given in (6) but here \( \xi = \hat{b}(\eta, j) \) as given in (19) at the bottom of the next page (for convenience).
subject to the constraints in (14) expressed as follows:
\[ U^* = \arg \min J(kT, U(kT)) \]  
(20)
where the global minimizer of (10), \( U^* \in U^{(t)} \subset U \), such that
\[ J(kT, U^*(kT)) \leq J(kT, U^{(t)}(kT)), \quad \forall U^* \in U^{(t)}. \]

The NN-based NAMPc algorithm proposed for solving (20) uses the full Newton [4] method based on the Levenberg-Marquardt [5] algorithm which introduces a regularization parameter, \( \lambda \), to the diagonal of (20) with the following updating rule:
\[ U^* = U^{(t)} + \mu^{(t)} \zeta^{(t)} \]  
(21)
which is applied for updating the sequence of future optimal control signal \( U^* \); \( \mu^{(t)} \) is the current iterate of the control sequence; \( \mu^{(t)} \) the step size; and \( \zeta^{(t)} \) the search direction:
\[ \zeta^{(t)} = -(H[U^{(t)}(kT)] + \lambda^{(t)} I)^{-1}G[U^{(t)}(kT)] \]  
(22)
where \( I \) is a diagonal matrix; \( H[U^{(t)}(kT)] \) and \( G[U^{(t)}(kT)] \) are respectively the Hessian and Jacobian matrices given in (23) at the bottom of this page, where \( \Phi \) is the partial derivatives of (18).

To simplify the computation of \( \Phi \), the control signal is decomposed into the past and future control signals as \( \hat{b}(\eta, j) \) given in (25) at the bottom of this page. The first three sums in (25) depend on future control signals while the last three sums depend on past control signals.

Since the past control signal does not contribute to the output predictions as new measurement becomes available at each time step, \( \Phi \) is computed based on future control signal \( \forall \quad \eta \in [N_1, N_2] \) and \( \forall \quad l \in [0, \min(\eta-1, N_u-1)] \) as:
\[ \Phi = \frac{\partial \gamma((k+\eta)T)}{\partial u((k+l)T)} = \sum_{j=1}^{\min(\eta, N_u)} W_j f' \left( \hat{b}(\eta, j) \right) \frac{\partial \hat{b}((k+\eta)T)}{\partial u((k+l)T)} \]  
(26)
where \( \partial \hat{b} / \partial u \) is given in (27) at the bottom of this page. The second-order derivative is calculated \( \forall \quad \eta \in [N_1, N_2] \), \( \forall \quad p \in [0, l] \), and \( \forall \quad l \in [0, \min(\eta-1, N_u-1)] \) as follows:

\[ \hat{b}(\eta, j) = \sum_{r=1}^{\max(\eta, p)} W_{j r} \left[ \gamma((k+\eta-\tau)T) + \sum_{r=\max(\eta, p)+1}^{\max(\eta, N_u)} W_{j r} \gamma((k+\eta-\tau)T) + \sum_{r=\max(\eta, N_u)+1}^{\max(\eta, N_u-1)} W_{j r} u((k+\eta-\tau)T) + W_{j 0} \right] \]  
(19)
\[ G[U^{(t)}(kT)] = \frac{\partial J(kT, \hat{U}(kT))}{\partial \hat{U}(kT)} \bigg|_{U^{(t)} 
(23)
\[ H[U^{(t)}(kT)] = \frac{\partial^2 J(kT, \hat{U}(kT))}{\partial \hat{U}^2(kT)} \bigg|_{U^{(t)}} = \kappa \left[ \frac{\partial^2 \gamma^2((k+\eta)T)}{\partial \gamma^2(kT)} + \gamma E(kT) \right] + 2 \rho \left[ \frac{\partial^2 \gamma^2((k+\eta)T)}{\partial \gamma^2(kT)} E(kT) \right] + 2 \rho \left[ \frac{\partial^2 \gamma^2((k+\eta)T)}{\partial \gamma^2(kT)} E(kT) \right] \bigg|_{U^{(t)}} \]  
(24)
\[ \hat{b}(\eta, j) = \sum_{r=1}^{\max(\eta, p)} W_{j r} \left[ \gamma((k+\eta-\tau)T) + \sum_{r=\max(\eta, p)+1}^{\max(\eta, N_u)} W_{j r} \gamma((k+\eta-\tau)T) + \sum_{r=\max(\eta, N_u)+1}^{\max(\eta, N_u-1)} W_{j r} u((k+\eta-\tau)T) + W_{j 0} \right] \]  
(19)
\[ \hat{b}(\eta, j) = \sum_{r=1}^{\max(\eta, p)} W_{j r} \left[ \gamma((k+\eta-\tau)T) + \sum_{r=\max(\eta, p)+1}^{\max(\eta, N_u)} W_{j r} \gamma((k+\eta-\tau)T) + \sum_{r=\max(\eta, N_u)+1}^{\max(\eta, N_u-1)} W_{j r} u((k+\eta-\tau)T) + W_{j 0} \right] \]  
(19)

\[ \frac{\partial^2 \gamma((k+\eta)T)}{\partial u((k+l)T)} = \sum_{r=1}^{\max(\eta, p)} W_{j r} \left[ \gamma((k+\eta-\tau)T) + \sum_{r=\max(\eta, p)+1}^{\max(\eta, N_u)} W_{j r} \gamma((k+\eta-\tau)T) + \sum_{r=\max(\eta, N_u)+1}^{\max(\eta, N_u-1)} W_{j r} u((k+\eta-\tau)T) + W_{j 0} \right] \]  
(27)

D. Computing the Optimal Control Signal

Thus, the problem to the full-Newton algorithm reduces to checking for the definiteness of the Hessian matrix using
\[ V^{(t)}(kT) = H[U^{(t)}(kT)] + \lambda^{(t)} I \]  
(29)

with a particular choice of \( \lambda^{(t)} \). Many algorithms [4], [5], [10] have been coded for selecting or adjusting \( \lambda^{(t)} \).

This contribution presents a new algorithm (TABLE I) which first compute (24) and check for positive definiteness. If this condition is not satisfied, the algorithm iteratively selects \( \lambda^{(t)} \) to compute (29) and the algorithm terminates immediately (24) is positive definite.

Also, it is necessary to check whether the search direction is descending using \( \mu^{(t)} \). A new approach is proposed here for adjusting \( \mu^{(t)} \) and \( \zeta^{(t)} \) iteratively based on the trust region method [10]. For this method, let \( \delta^{(t)} \) be the radius of the trust region \( \Upsilon(U^{(t)}) \) around which the current iterate \( U^{(t)} \) can be trusted; so that \( U^* \) can be determined by minimizing the following sub-criterion under constraints:
\[ U^* = \arg \min U^{(t)}(U) \text{ subject to } \| U^{(t)} - U^{(t-1)} \| \leq \delta^{(t)} \]  
(30)

In our approach here, the Cholesky factors \( L_{\mu}^{(t)} \) obtained in TABLE I are used to compute \( \zeta^{(t)} \) using the algorithm in TABLE II and the control inputs \( U^{(t)} \) are updated based on the value of (23). Next, the algorithm of TABLE III is used to compute \( \mu^{(t)} \) as a ratio from (21) with the idea
TABLE I  
ITERATIVE ALGORITHM FOR SELECTING $\lambda$ FOR GUARANTEED POSITIVE DEFINITENESS OF THE HESSIAN MATRIX

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\lambda_a$</td>
<td>Parameter $a$</td>
<td>0.5</td>
</tr>
<tr>
<td>$\lambda_b$</td>
<td>Parameter $b$</td>
<td>1</td>
</tr>
<tr>
<td>$\lambda_c$</td>
<td>Parameter $c$</td>
<td>2</td>
</tr>
<tr>
<td>$\lambda_d$</td>
<td>Parameter $d$</td>
<td>4</td>
</tr>
<tr>
<td>$\lambda_e$</td>
<td>Parameter $e$</td>
<td>6</td>
</tr>
<tr>
<td>$km$</td>
<td>Length of $km$</td>
<td>$\lambda_a, \lambda_b, \lambda_c, \lambda_d, \lambda_e$</td>
</tr>
<tr>
<td>$m$</td>
<td>Size of $V^{(1)}$</td>
<td>$\lambda_a \leq \lambda_b \leq \lambda_c \leq \lambda_d \leq \lambda_e$</td>
</tr>
<tr>
<td>$kl$</td>
<td>Length of $kl$</td>
<td>0</td>
</tr>
<tr>
<td>$p$</td>
<td>Parameter $p$</td>
<td>$N_a + 1$</td>
</tr>
</tbody>
</table>

An iterative algorithm for selecting the Levenberg-Marquardt parameter $\lambda$ for guaranteed positive definiteness of the Hessian matrix.

Initialize parameters: $\lambda_a = 0.5$, $\lambda_b = 1$, $\lambda_c = 2$, $\lambda_d = 4$, $\lambda_e = 6$, $km = [\lambda_a, \lambda_b, \lambda_c, \lambda_d, \lambda_e]$, $kl = length(km)$, $iter = 0$, $m = size(V^{(1)}(kT))$, $L^{(0)}_{ij} = -1$, and $p = 1$.

Compute $L^{(0)}_{ij}$.

for $sm = 1$ to $m$, do

while $iter < kl$ or $L^{(0)}_{ij} < 0$, do

for $kk = 1$ to $kl$, do

for $k = 1$ to $p$, do

$L^{(t)}_{ij} = \left[V^{(t)}_{ij}(kT) - \sum_{i=1}^{i-1} L^{(t)}_{ij}\right]^{2}$

if $L^{(t)}_{ij} < 0$, break end if

for $c = k + 1$ to $p$,

$L^{(t)}_{ij} = \frac{1}{L^{(t)}_{ij}}\left[V^{(t)}_{ij}(kT) - \sum_{i=1}^{i-1} L_{ij}\right]^{2}$

end for, end for

if $L^{(t)}_{ij} < 0$, $\lambda^{(t)} = \lambda km(1, kk)$.

Re-compute (32)

for $k = 1$ to $p$, do

$L^{(t)}_{ij} = \left[V^{(t)}_{ij}(kT) - \sum_{i=1}^{i-1} L^{(t)}_{ij}\right]^{2}$

end for

else

for $k = 1$ to $p$, do

$L^{(t)}_{ij} = \left[V^{(t)}_{ij}(kT) - \sum_{i=1}^{i-1} L^{(t)}_{ij}\right]^{2}$

for $c = k + 1$ to $p$,

$L^{(t)}_{ij} = \frac{1}{L^{(t)}_{ij}}\left[V^{(t)}_{ij}(kT) - \sum_{i=1}^{i-1} L_{ij}\right]^{2}$

end for, end for, end if, end for

$iter = iter + 1$

if iter $> kl$ and $L^{(t)}_{ij} < 0$, break end while

end while iter, end for

end for, end if, end while

end for

TABLE II  
COMPUTING THE SEARCH DIRECTION AND UPDATING THE CRITERION

Compute the search direction and update the criterion using Cholesky factors obtained in TABLE I.

Let $\sigma_{p} \equiv G[U^{(t)}(kT)]?$, relative residual $\sigma_{r} \in (0,1)$, and absolute residual $\sigma_{a} \in (0,1)$.

while $\sigma_{p} \geq \sigma_{a}$, do

$L^{(t)}_{ij} = -G[U^{(t)}(kT)]$ (Forward substitution)

$\zeta^{(t)} = [L^{(t)}_{ij}]^{-1} + (\zeta^{(t)})^{-1} \zeta^{(t)}$ (Backward substitution)

if $G[U^{(t)}(kT)] < \sigma_{r}$, $\sigma_{r} < G[U^{(t)}(kT)] + \sigma_{r}$ (Termination criteria)

end if

Update (21) and evaluate the criterion (13) using $\tilde{U}(t) = \tilde{U}^{(t)}(kT)$.

described in [10]. Finally, the value of $\mu^{(t)}$ is used to control $\lambda^{(t)}$ to obtain the optimal future control signal $U^{*}$.

IV. THE SERVICE-ORIENTED COMPUTER NETWORK

A testbed for evaluating the performance of the developed control algorithm was set up. This is the Service Oriented Computer Network (S-OCN) shown in Fig. 3. A number of computing elements host the mathematical relationships of the controlled variables in terms of the manipulated variables and also act as DPWS (Device Protocol for Web Services) [3] servers. The computed samples of the controlled variables by the mathematical relationships are transmitted to another computer which executes the proposed identification and control algorithms. This computer acts also as DPWS client and transmits the computed values for the manipulated variables by the algorithm to the computing elements which compute the values of the controlled variables. Then, again new samples for the controlled variables are estimated and sent to the control algorithm, closing in this way the control loop.
TABLE IV

<table>
<thead>
<tr>
<th>Constraint parameters</th>
<th>Tri °C</th>
<th>Th °C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial (guessed) control input (u)</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Minimum control inputs (( u_{\text{min}} ))</td>
<td>(-150)</td>
<td>(-200)</td>
</tr>
<tr>
<td>Maximum predicted outputs (( y_{\max} ))</td>
<td>150</td>
<td>200</td>
</tr>
<tr>
<td>Minimum predicted outputs (( y_{\min} ))</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Desired Set-points, Ref. (( u(r) ))</td>
<td>860</td>
<td>1040</td>
</tr>
</tbody>
</table>

TABLE V

<table>
<thead>
<tr>
<th>Tuning Parameters for the NN-Based NAMPC Controller</th>
<th>( N_1 )</th>
<th>( N_2 )</th>
<th>( N_3 )</th>
<th>( \kappa )</th>
<th>( \rho )</th>
<th>( \tau )</th>
<th>( \omega )</th>
<th>( A_0 )</th>
<th>( B_0 )</th>
<th>( \lambda )</th>
<th>( \delta )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tri</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>0.9</td>
<td>0.03</td>
<td>[1 -0.7]</td>
<td>[0.3]</td>
<td>0.1</td>
<td>0.1e-4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Th</td>
<td>1</td>
<td>2</td>
<td>6</td>
<td>1</td>
<td>0.08</td>
<td>[1 -0.7]</td>
<td>[0.3]</td>
<td>0.1</td>
<td>0.1e-6</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

![Fig. 4. NAMPC performance for input step changes on the predicted outputs (a) Tri and (b) Th with respect to the desired set-point (Ref).](image)

![Fig. 5. Long range prediction of (a) Tri and (b) Th with respect to the desired set-point (Ref) for input step change.](image)

V. SIMULATION RESULTS

The proposed control strategy has been applied to the temperature control of a fluidized bed furnace reactor (FBFR) used to evaluate catalyst in a fluid catalytic unit. The process description and model are given in [2]. The control objectives are given in TABLE IV where Tri and Th are the temperatures of the reactor interior and the heater.

The heat supplied (\( Q = 5.04 \) kW) to this model is varied by ±20%. A number of \( N = 1320 \) input output data from the open loop operation of the plant were given. The data were split into 1056 training and 264 validation data. The training data are scaled to zero mean and unit variance. The inputs are the temperatures of the reactor’s interior Tri(\( kT \)), interior reactor wall Tinf(\( kT \)), air gap between the reactor and the heater Threv(\( kT \)), heater Th(\( kT \)), insulator Tins(\( kT \)), and outer reactor metal wall Tmax(\( kT \)) while the outputs are \( \hat{y}(kT) = [\text{Tri}(kT) \text{ Th}(kT)]^T \). The network was trained for 100 epochs with the following selected parameters: \( j=6, i=2, m=2, n=2, \alpha_1=1 \times 10^{10}, \alpha_2=1 \times 10^2; \) and the weights were rescaled afterwards.

The controller is first designed off-line based on \( \theta^* \) and tuned in closed-loop subject to the constraints of (14) given in Table IV and the best design parameters given in Table V were obtained. Then, 20 simulation runs were then performed by using the testbed of Fig. 3 for different values of Q (±15%) with 264 measurements at each iteration.

The Tri and Th output predictions for step changes in the inputs are shown in Fig. 4(a) and (b) respectively while the long range output predictions for a step input is shown in Fig. 5(a) and (b) respectively for Tri and Th. It can be seen in both figures that the proposed NN-based NAMPC tracks the reference signal without any overshoot despite the perturbed heat (Q) supplied to the FBFR process.

VI. CONCLUSIONS

In this work a new neural network-based adaptive model predictive control algorithm was developed. The proposed strategy is applied to the on-line identification and control of the temperatures of a fluidized bed furnace reactor (FBFR) where the control actions are continuously updated at each iteration on the basis of new measurements received over a switched Ethernet, service-oriented computer network. The simulation results have shown good control performances with no overshoot and fast settling times.

REFERENCES