International Journal of New Technology and Research (IJNTR)
ISSN:2454-4116, Volume-4, Issue-3, March 2018 Pages 120-124

Controlling Of Non linear Process Using Neural Network Based Adaptive PID Controller

Mr. R. Sakthi Vel, Mrs. A. Dhanalakshmi, V. Dhivya , S. Keerthana , M. Revathi

***Abstract*— Fault tolerant control of dynamic processes is investigated in this paper using an auto-tuning of Adaptive PID controller. A fault tolerant control scheme is proposed composing an auto-tuning Adaptive PID controller based on an adaptive neural network model. The model is trained online using the Unscented Kalman filter (UKF) algorithm to learn system post-fault dynamics. Based on this model, the Adaptive PID controller adjusts its parameters to compensate the effects of the faults, so that the control performance is improved. The auto-tuning algorithm for the Adaptive PID controller is derived with the Lyapunov method and therefore, the model predicted tracking error is guaranteed to desired point asymptotically.**

***Index Terms*— Adaptive NN models; auto-tuning Adaptive PID; Unscented Kalman filter (UKF); fault tolerant control.**

.

I. INTRODUCTION

With fast increase in difficulty of modern control systems, the importance of the fault tolerant control (FTC) concept and technology has been appreciated and accepted by industry. Control system stability and reliability are not only vital for some projects where stringent safety conditions apply, e.g., nuclear power stations and passenger airplanes, but also essential for significant productions, since most of present industrial plants are complex and often include a number of subsystems which may balance for the effects of sensor faults and element malfunction.

This requires solutions that are very costly in both hardware and development attempt. Therefore, FTC is very important from the viewpoint of safety, as well as reduced production costs. FTC offers the ability to avoid accidental process shut downs from simple faults, e.g. in instrumentation and control loops that could develop into production loss or plant failures. Recently, FTC in most real industrial systems are appreciated by hardware redundancy. For example, the majority-voting scheme is used with redundant sensors to cope with sensor faults [1]. However, due to two main limitations of the hardware redundancy, high cost, and taking more space, solutions using analytical redundancy [1] have been investigated over the last two decades.

There are generally two different approaches using analytical redundancy: 1) passive approaches, and 2) active approaches. Passive approaches use robust control techniques to design closed-loop systems so that it is numb to certain faults, e.g., [2]. In recent times, an elegant design method of passive approach was proposed by Chen *et al.* [3], in which the linear matrix inequality (LMI) method was used to synthesis the consistent controller. Different faults were formulated as constraints in the method and were considered in the optimal design using LMI. While the design example showed the stability and maintenance of suitable system performance, a limitation is that the method is based on an accurate linear state space model and therefore, is not capable of controlling nonlinear processes for which an accurate analytical model is usually not available.

In addition, because the passive approaches consider fault tolerance in only the stage of controller design without taking adaptation when faults occur, the amplitude of the faults that can be allowable is usually limited. Active approaches use online fault adjustment information and reconfigurable controllers. When a fault is identified using analytical or hardware redundancy, the controller is reconfigured to guarantee the post-fault stability and maintain acceptable performance. Active FTC has been explored using different methods including the feedback linearization [4], control law rescheduling [5], and model following control [6].

Reconfigurable control against plant component faults has been studied using state feedback, where the feedback gain matrix was designed using linear quadratic regulation method [7], the pseudo inverse method [8] and eigen structure assignment method [9]. However, these studies were also based on linear models so that they are not suitable for nonlinear processes. Model predictive control (MPC) has been employed in FTC [10], [11], where an adjustable objective function was optimized based on a simple linear model. The model was expected to learn the post-fault dynamics if the amplitude of the fault is not too large. The research utilizing this method is active and the models used are extended to nonlinear models.

II. Structure Of FTC Scheme

The aim of the FTC is to obtain a control variable to force the process to track the preferred trajectory when the process is not subject to any error, while to keep up the system constancy and to make progress from the performance degradation high-speed when fault occurs in the process. Allowing for that element or organic processes are difficult and their mathematical models are usually unfamiliar, an active fault tolerant control approach is developed in this section.

Mr. R. Sakthi Vel, Assistant Professor, Department Of Electronics and Instrumentation Engineering(EIE),Panimalar Engineering College ,Chennai
Mrs. A. Dhanalakshmi, Assistant Professor, Department Of Electronics and Instrumentation Engineering(EIE),Panimalar Engineering College ,Chennai
V. Dhivya, Student, Department Of Electronics and Instrumentation Engineering(EIE) Panimalar Engineering College, Chennai.
S. Keerthana , Student, Department Of Electronics and Instrumentation Engineering(EIE) Panimalar Engineering College, Chennai.
M. Revathi, Student, Department Of Electronics and Instrumentation Engineering(EIE) Panimalar Engineering College, Chennai.

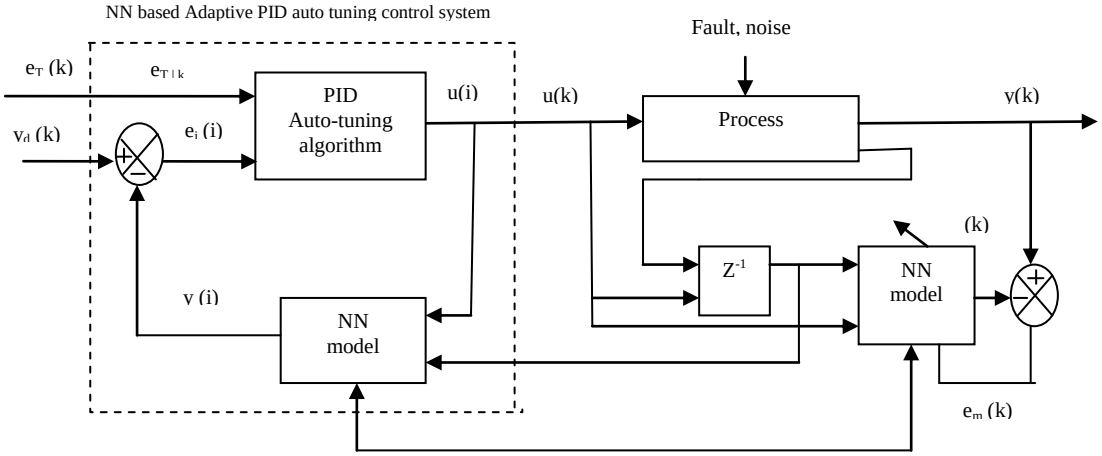

Figure 1. Structure of the NN-based Auto-tuning PID control system.

The control design includes two parts, one is using a NN to model the process and the model is made adaptive to hold the dynamics change caused by the fault, another is an auto-tuning PID controller based on the model. When the model captures the post-fault dynamics, the PID controller is modified to balance the degradation of system stability and performance. The MLP model is online modified with the model calculation error using the EKF algorithm. The modified model is used to calculate process output at next sample time. The prediction is used by the auto-tuning algorithm to derive an optimal control variable.

In Fig.1 PID controller adapts its parameters in the way that the produced control variable will drive the NN model output to track the desired reference. A recursive auto-tuning algorithm derived using the Lyapunov method will make the optimal control that is guaranteed to minimize a squared tracking error. The wide line between the model used for prediction and that to be adapted indicates that the formation and weights are shared between the two models.

III. Adaptive Neural Network Model

A. *NN Model*

To model the system dynamics and also to capture the time-varying dynamics of the process, an adaptive Multi Layer Perceptron (MLP) network model is developed. The NARX model used for this multivariable nonlinear system is :

$$y(k)=g\left(u(k-d-1),\ldots,u(k-d-n_u),\ldots,y(k-1),\ldots,y(k-n_y)\right)+e(k) \quad (1)$$

where $y \in R^p$ *and* $u \in R^m$ are the sampled process output and input vectors, and n_y and n_u are the output and input order, d is the input transmission delay, e is the measurement noise and $g(\bullet)$ is the vector valued nonlinear function. Similarly, the MLP network model used is :

$$\hat{y}(k)=\hat{g}\left(u(k-d-1),\ldots,u(k-d-n_u),\ldots,y(k-1),\ldots,y(k-n_y)\right)+e(k) \quad (2)$$

where $\hat{y} \in R^p$ is the estimated process output by the NN model and $\hat{g}(\bullet)$ is an approximated nonlinear function of $g(\bullet)$. The MLP network with one hidden layer of neurons is implemented. $\hat{y}(k) = W^y h(k)$ (3)

where $W^y \in R^{p\times(q+1)}$ is the weight matrix connecting output and hidden layers, $h(k) \in R^{q+1}$ is the hidden layer output vector where each entry is transformed from the corresponding entry in z(k) by using a sigmoid activation function as :

$$h_i(k) = f\left[z_i(k)\right] = \frac{1}{1+e^{-z_i(k)}}, \quad i = 1, 2, \ldots, q \quad (4)$$

$$z(k) = W^h x(k) \quad (5)$$

where $W^h \in R^{q\times(n+1)}$ is the weight matrix connecting the hidden and input layers, $x(k) \in R^n$ where $n = mn_u + pn_y + 1$ is the network input vector. Thus x(k) and h(k) are defined by :

$$x(k)=\left[u(k-d-1)^T,\ldots,u(k-d-n_u)^T,\ldots,y(k-1)^T,\ldots,y(k-n_y)^T,1\right]^T \quad (6)$$

$$h(k)=\left[\frac{1}{1+e^{-Z_1(K)}},\ldots,\frac{1}{1+e^{-Z_q(K)}},1\right]^T \quad (7)$$

with the last entry introducing bias to the hidden layer.

B. *UKF – Based Training Algorithm*

UKF Training The UKF algorithm for NN training is similar to that of EKF; again, all the connecting weights are organized as a state vector, but now the state is calculated through unscented transformation [12], [13] and propagated analytically through nonlinear system without the need to evaluate the Jacobian matrix. The generic UKF can be summarized as in the following steps.

STEP 1) Initialization

$$w_o = E\left[w_o\right]$$
$$P_O = E\left[(w_o - \hat{w}_o)(w_o - \hat{w}_o)^T\right] \quad (8)$$

STEP 2) Calculation of the sigma points

$$\chi_{o,k-1} = \hat{w}_{k-1}\omega_o = \frac{k}{(L+k)}$$

$$\chi_{i,k-1} = \hat{w}_{k-1} + \left(\sqrt{(L+k)P_{k-1}}\right)_i \qquad \omega_i = \frac{0.5}{(L+k)}$$

$$\chi_{i+L,k-1} = \hat{w}_{k-1} - \left(\sqrt{(L+k)P_{k-1}}\right)_i \qquad \omega_{i+L} = \frac{0.5}{(L+k)}$$

(9)

where i = 1,2,…,L and L is the state dimension. The parameter k is used to control the covariance matrix.

STEP 3) Time Update

$$\chi_{i,k/k-1} = \chi_{i,k-1}$$

$$\hat{w}_{\bar{k}} = \sum_{i=0}^{2L} \omega_i \chi_{i,k|k-1}$$

$$\bar{P}_k = \sum_{i=0}^{2L} \omega_i \left[\chi_{i,k|k-1} - \hat{w}_{\bar{k}}\right]\left[\chi_{i,k|k-1} - \hat{w}_{\bar{k}}\right]^T$$

(10)

$$y_{i,k|k-1} = g\left(\chi_{i,k|k-1}, x_k\right)$$

$$\hat{y}_{\bar{k}} = \sum_{i=0}^{2L} \omega_i \left[y_{i,k|k-1}\right]$$

(11)

STEP 4) Measurement update

$$S_k = \sum_{i=0}^{2L} \omega_i \left[y_{i,k|k-1} - \hat{y}_{\bar{k}}\right]\left[y_{i,k|k-1} - \hat{y}_{\bar{k}}\right]^T + R_k$$

$$G_k = \sum_{i=0}^{2L} \omega_i \left[\chi_{i,k|k-1} - \hat{w}_{\bar{k}}\right]\left[y_{i,k|k-1} - \hat{y}_{\bar{k}}\right]^T$$

$$\hat{w}_k = \hat{w}_k + G_k S_k^{-1}\left(d_k - \hat{y}_k^{-1}\right)$$

$$P_k = P_k^{-1} - \kappa_k S_k \kappa_k^T$$

(12)

C. NN based UKF algorithm

We attempt to improve the algorithm in [14] under the framework of UKF and apply the new method to the problem of nonlinear filtering. The state evolution equation and observation equation are given as

$$x_{k+1} = f\left(x_k\right) + n_k$$

$$y_k = h\left(x_k\right) + v_k \qquad (13)$$

where n_k and v_k represent process noise and observation noise, respectively. We denote ek as the error between true model and the a priori known mathematical model $\hat{f}(x_k)$, namely e(k)=f(k) - $\hat{f}(x_k)$. We can adjust the weighs of the NN through the observation if and only if the observability condition is met. When $e(k) - g(x_k - w_k) \to 0$, the error is well approximated, and the more accurate model becomes the sum of $f(x_k)$and the NN approximation, so we have :

$$x_{k+1} = f\left(x_k\right) + g\left(x_k, w_k\right)$$

$$w_{k+1} = w_k \qquad (14)$$

If we represent the augmented state vestors as $x_k^a = \left[x_K^T \; w_k^T\right]^T$, then the above equations(write the equation no) can be redefined as

$$x_{k+1}^a = \begin{bmatrix} x_{k+1} \\ w_{k+1} \end{bmatrix} = \begin{bmatrix} \hat{f}(x_k) + g(x_k, w_k) \\ w_k \end{bmatrix} \cong f^a\left(x_k, w_k\right)$$

(15)

Because the observation is only related with the state x, the equation $y_k = h\left(x_k\right) + v_k$ remains unchanged. Now the estimation of $\hat{x}_k^a$ based on the new model (15) and the noisy observation $y_k = h\left(x_k\right) + v_k$. Once given

$$\hat{x}_o^a = E\left[x_o^a\right] \quad \text{and} \quad P_o^a = E\left[\left(x_o^a - \hat{x}_o^a\right)\left(x_o^a - \hat{x}_o^a\right)^T\right]$$

(16)

we can get $2L^a + 1$ vectors $\chi_{i,k-1}^a = \left[\left(\chi_{i,k+1}^x\right)^T \left(\chi_{i,k+1}^w\right)^T\right]^T$

(17)

(i=0,1,…,$2L^a$, and L^a is the dimension of x_k^a), according to the equation

$$\chi_{o,k-1} = \hat{w}_{k-1}\omega_o = \frac{k}{(L+k)}$$

$$\chi_{i,k-1} = \hat{w}_{k-1} + \left(\sqrt{(L+k)P_{k-1}}\right)_i \qquad \omega_i = \frac{0.5}{(L+k)}$$

$$\chi_{i+L,k-1} = \hat{w}_{k-1} - \left(\sqrt{(L+k)P_{k-1}}\right)_i \qquad \omega_{i+L} = \frac{0.5}{(L+k)}$$

(18)

and at instant k-1 , then the filtering procedure of NN-aided UKF can be carried out recursively in the similar way ,but now $\chi_{i,k/k-1} = \chi_{i,k-1}$ and $y_{i,k|k-1} = g\left(\chi_{i,k|k-1}, x_k\right)$ are replaced by $f^a\left(\chi_{i,k-1}^x, \chi_{i,k-1}^w\right)$ and $y_{i,k|k-1} = h\left(x_{i,k|k-1}^x\right)$, respectively.

The Procedure of applying the UKF algorithm to the adaptive model is given as follows:

Step 1) Obtain the past process output y_k and past control variable u at sample time k to form NN model input vector x_k in (4).

Step 2) Obtain the current process measurement output y_k which is used as the training target.

Step 3) Update the error covariance matrix $\bar{P}_k$ using (10) & (11).

Step 4) Implement the UKF based training algorithm (13) - (18)

The developed adaptive MLP model is evaluated by modeling the simulated CSTR process and this is described in Section V.

IV. Auto Tuning For Adaptive PID Controller

In this paper, an adaptive neural model is proposed for the multivariable PID controller based on self-learning algorithm. The PID controlled tracking error is evaluated using model prediction and then an optimal set of PID parameters is iteratively approached. Thus, minimum squared tracking

error can be obtained. The discrete-time multivariable PID controller is considered as :

$$u(k) = K_p(k)\, e_T(k) + K_i(k) \sum_{i=1}^{k} e_T(i) + K_d(k) \frac{e_T(k) - e_T(k-1)}{T} \quad (19)$$

where $e_T(k) \in R^p$ is the process tracking error defined as :

$$e_T(k) = y_d(k) - y(k)$$

where $y_d(t) \in R^p$ is the desired trajectory, T is the sampling time, $K_p \in R^{m \times p}$, $K_i \in R^{m \times p}$ and $K_d \in R^{m \times p}$ are PID controller parameter matrices, and m is the number of input. To estimate the optimum PID parameters, $K_p(k)$, $K_i(k)$ and $K_d(k)$, a parameter vector $\theta_{pid}(k) \in R^{3mp}$ is formulated as :

$$\theta_{pid}(k) = \begin{bmatrix} [K_{p_1}(k) \quad K_{i_1}(k) \quad K_{d_1}(k)]^T \\ \vdots \\ [K_{p_m}(k) \quad K_{im}(k) \quad K_{dm}(k)]^T \end{bmatrix} \quad (20)$$

where $k_{p_j}(k)$, $k_{i_j}(k)$ and $k_{d_j}(k)$ denotes the j^{th} row in $K_p(k)$, $K_i(k)$ and $K_d(k)$, j = 1, 2,…, m.

Inorder to obtain the optimal control variable in each sample period k, an iterative algorithm is developed which minimizes the objective function of the predictive tracking error. These tuned PID parameters are used at the end of sample period to produce control variable. Here the sample time is expressed as k and i denotes the iterative step within each sample period. Inorder to avoid confusion with the iterative step i, variable at sample time "(k)" is change to "|k" in the iterative process, such as $\theta_{pid}(k) \rightarrow \theta_{pid|k}$.

On predicting $\theta_{pid}(k)$, $\hat{\theta}_{pid}(i) \in R^{3mp}$ is used to denote the optimum PID parameters in the iterative process. The PID auto-tuning algorithm is defined as :

$$\hat{\theta}_{pid}(i+1) = \hat{\theta}_{pid}(i) + \Delta\hat{\theta}_{pid}(i) = \hat{\theta}_{pid}(i) + K_{pid}(i)\, e_t(i) \quad (21)$$

where $K_{pid}(i)$ is the gain matrix, $e_t(i) \in R^p$ is the NN model tracking error and e_T is the process tracking error.

$$e_t(i) = y_{d|k} - \hat{y}(i)$$
$$\hat{y}(i) = \hat{g}(\hat{u}(i), X_k) \quad (22)$$

where $y_{d|k} \in R^p$ is the desired output at sample time k, $\hat{y}(i)$ is the NN model output in iterative step i within the sample period. $\hat{y}(i)$ is a function of the predicted optimal control variable, $\hat{u}(i) \in R^m$ such that :

$$\hat{u}(i) = K_p(i)\, e_{T|k} + K_i(i)\, E_{pid|k} + K_d(i) \frac{e_{T|k} - e_{T|k-1}}{T} \quad (23)$$

$$E_{pid|k} = E_{pid}(k) = \sum_{j=1}^{k} e_j \quad \text{with} \quad (24)$$

In order to derive the gain matrix in K_{pid} at each iterative step i, such that the convergence of the NN model output to the desired process output is guaranteed, a discrete-time Lyapunov function is chosen as follows:

$$V(i) = \delta e_t(i)^T e_t(i) \quad (25)$$

where δ is a positive constant and thus, V(i) is positive definite. For V(i) calculation, define $e_t(i) = e_t(i-1) + \Delta e_t(i)$. In discrete time operation, $\Delta\hat{\theta}_{pid}(i)$ can be approximated by the derivative of $\hat{\theta}_{pid}(i)$ with respect to i . Thus

$$\Delta e_t(i) = \frac{\partial e_t(i)}{\partial \hat{\theta}_{pid}(i)} \frac{\partial \hat{\theta}_{pid}(i)}{\partial i}$$
$$= \frac{\partial [y_{d\,k} - \hat{y}(i)]}{\partial \hat{\theta}_{pid}(i)} \Delta\hat{\theta}_{pid}(i)$$
$$= \frac{\partial \hat{y}(i)}{\partial \hat{\theta}_{pid}(i)} K_{pid}(i) e_t(i) \quad (26)$$

Then, the increment of the Lyapunov function, $\Delta V(i)$ is expressed as :

$$\Delta V(i) = V(i+1) - V(i)$$
$$= 2\delta\, \Delta e_t(i)^T [e_t(i) + \delta\, \Delta e_t(i)]$$
$$= -2\delta \left[\frac{\partial \hat{y}(i)}{\partial \hat{\theta}_{pid}(i)} K_{pid}(i) e_t(i) \right]^T$$
$$\times \left[e_t(i) - \delta \frac{\partial \hat{y}(i)}{\partial \hat{\theta}_{pid}(i)} K_{pid}(i) e_t(i) \right]$$
$$= -2\delta e_t(i)^T \left[\frac{\partial \hat{y}(i)}{\partial \hat{\theta}_{pid}(i)} K_{pid}(i) \right]^T$$
$$\times \left[I - \delta \frac{\partial \hat{y}(i)}{\partial \hat{\theta}_{pid}(i)} K_{pid}(i) \right] e_t(i) \quad . \quad (27)$$

where the gain matrix $K_{pid}(i)$ is given as :

$$K_{pid}(i) = \frac{1-\delta}{\delta} \left[\frac{\partial \hat{y}(i)}{\partial \hat{\theta}_{pid}(i)} K_{pid}(i) \right]^T \left\{ \frac{\partial \hat{y}(i)}{\partial \hat{\theta}_{pid}(i)} \left[\frac{\partial \hat{y}(i)}{\partial \hat{\theta}_{pid}(i)} \right]^T \right\}^{-1} \quad (28)$$

Then $\Delta V(i)$ becomes, $\Delta V(i) = -2(1-\delta)\, e_t^T(i)\, e_t(i)$.

If $\delta < 1$ is chosen, then $\Delta V(i)$ is always negative. In the design of the gain matrix, $\delta < 1$ is chosen as the learning rate to adjust the self-learning speed of the PID parameters. Due to this, the predictive tracking error will converge to zero.

Thus, when the gain matrix $K_{pid}(i)$ is chosen according to

(28), the PID parameter vector, $\hat{\theta}_{pid}(k)$ will asymptotically converge to the optimal value at each sampling time in the sense of driving the model tracking error to minimum value. The Tracking performance of Auto –tuning PID controller is as shown in figure. 2

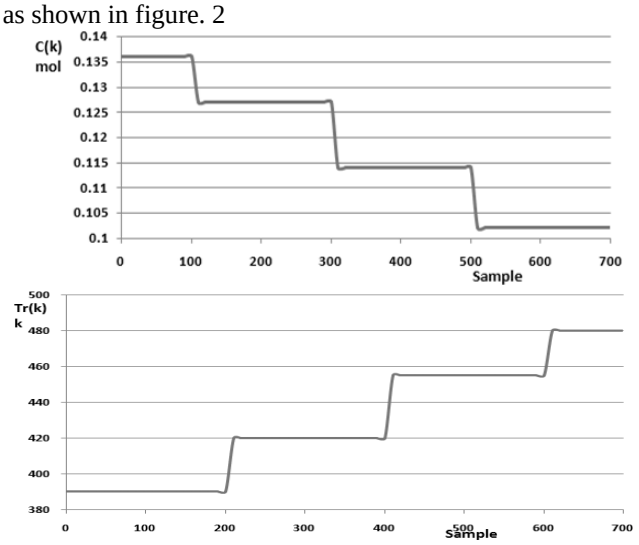

Figure 2. Tracking performance of Auto tuning PID controller

The PID auto-tuning procedure is as follows:

Step 1) For each sample time k, form the NN model input vector X(k) by obtaining the desired trajectory $y_d(k+1)$ the past control variable u (k) and the past process output y (k).

Step 2) Implement the PID auto-tuning algorithm given in (22), (28), and (21) in iterative form to predict the optimum PID parameter vector, $\hat{\theta}_{pid}(k)$. The value at the last sample time $\theta_{pid}(k-1)$ is assigned to be the initial value $\hat{\theta}_{pid}(0)$.

Step 3) Calculate the optimal control variable $\hat{u}(i)$ in (23) by applying the obtained $\hat{\theta}_{pid}(i)$ to the PID controller.

Step 4) Calculate the model output, $\hat{y}(i)$ and model tracking error, $e_t(i)$ in (22) by applying the obtained $\hat{u}(i)$ to the NN model at each iteration step, i.

Step 5) Repeat Step 1 to Step 4 until the NN model tracking error, is less than a pre-specified threshold or a specified bound to the iterative step is reached.

Step 6) Set $\theta_{pid}(k)$ to be equivalent to $\hat{\theta}_{pid}(i_{final})$, and then apply it to the PID controller in the process.

VI. Conclusion

The proposed system employs an adaptive PID controller to compensate the fault effects. The convergence of the predicted tracking error for auto-tuning algorithm is derived with Lyapunov method. In UKF algorithm, the post-fault dynamics can be modeled in time and thus, the degradation in the process tracking performance and in system relative stability is quickly recovered. The adaptive NN model is online trained with measurement process input output data and consequently, the effects of sensor faults will also be modeled. Thus, the sensor faults are not tolerable with the developed method. This leads to a process tracking error of the size of the occurred sensor fault. The research work can be undergone on sensor fault tolerance in processes with unknown dynamics. The potential applications include different industrial processes with multivariable and nonlinear dynamics.

References

[1] R. J. Patton, P. M. Frank, and R. Clark, *Fault Diagnosis in Dynamic Systems : Theory and Application*. Englewood Cliffs, NJ: Prentice-Hall, 2005.

[2] M. Tyler and M. Morari, "Optimal and robust design of integrated control and diagnostic modules," in *Proc. American Control Conf.*, 2008.

[3] J. Chen, R. J. Patton, and Z. Chen, "Active fault tolerant flight controlsystems design using the linear matrix inequality method," *Trans. Inst. Meas. Control*, vol. 21, no. 2/3, pp. 77–84, 2013.

[4] Y. Ochi, "Application of feedback linearization method in a digital restructurable flight control system," *J. Guid. Control Dyn.*, vol. 16, no. 1, pp. 111–117, 2004.

[5] I. Kaminer, A.M. Pascoal, P. P. Khargonekar, and E. E. Coleman, "Avelocity algorithm for the implementation of gain-scheduled controllers," *Automatica*, vol. 31, no. 8, pp. 1185–1192, 2014.

[6] W. D. Morse and K. A. Ossman, "Model-following reconfigurable flight control system for the AFTI/F-16," *J. Guid. Control Dyn.*, vol. 13, no. 6, pp. 969–976, 2014.

[7] D. P. Looze, J. L. Weiss, J. S. Eterno, and N. M. Barrett, "An automatic redesign approach for restructurable control systems," *IEEE Contr. Syst. Mag.*, vol. 5, no. 2, pp. 16–22, May 2009.

[8] Z. Gao and P. J. Antsaklis, "Stability of the pseudo-inverse method for reconfigurable control systems," *Int. J. Control*, vol. 53, pp. 717–729, 2014.

[9] J. Jiang, "Design of reconfigurable control systems using eigenstructure assignments," *Int. J. Control*, vol. 59, no. 2, pp. 395–410,2015.

[10] W. K. Son, O. K.Kwon, and M. E. Lee, "Fault-tolerant model-based predictive control with application to boiler systems," in *Proc. IFAC Symp. SAFEPROCESS*, Hull, U.K., 2007, pp. 1240–1245.

[11] J. M. Maciejowski, "Modeling and predictive control: enabling technologies for reconfiguration," in Proc. IFAC Symp. System Structure *Control*, Oct. 1997, pp. 253–258.

[12] J. Julier, K. Uhlmann, and F. Durrant-Whyte, "A new approach for filtering nonlinear systems," in *Proc. American Control Conf.*, Seattle, WA, Jun. 2015, pp. 1628–1632.

[13] J. Julier and K. Uhlmann, "A new method for the nonlinear transformation of means and covariances in filters and estimators," *IEEE Trans. Autom. Control*, vol. 45, no. 3, pp. 477–482, Mar. 2000.

[14] R. N. Lobbia, S. C. Stubberud, and M. W. Owen, "Adaptive extended Kalman filter using artificial neural networks," *Int. J. Smart Eng. Syst. Des.*, vol. 1, pp. 207–221, 2015.

[15] Ronghui Zhan, J. Wan, " Neural network-aided adaptive unscented Kalman filter for nonlinear state estimation ," IEEE Signal Processing Letters , Vol 13, Issue: 7, July 2006.