**Proceedings of the 17th World Congress**
**The International Federation of Automatic Control**
**Seoul, Korea, July 6-11, 2008**

# INTELLIGENT ADAPTIVE DYNAMIC MATRIX CONTROL

**J. Posada*. M. Sanjuan.****

**Mechanical Engineering Department, Universidad del Norte*
*Barranquilla, Colombia ( jposada@ieee.org).*
*** Mechanical Engineering Department, Universidad del Norte*
*Barranquilla, Colombia ( msanjuan@uninorte.edu.co)*

**Abstract:** This paper presents an approach to adapt the suppression and scaling factor from a single input single output (SISO) dynamic matrix controller (DMC) thought a multiobjective optimization algorithm. To optimize, a nonlinear neural network (NN) process model is used, combined with a multiobjective evolutionary algorithm called SPEA II (Strength Pareto Evolutionary Algorithm) to find better controller parameters for the plant each sample time. Also every sample time, a decision over the resultant pareto front from the multiobjective optimization process are taken using a simple decision approach.

## 1. INTRODUCTION

Industrial Model Predictive Controllers were introduced in the 1970s as an industrial alternative for designing controllers based on plant-data based dynamics without fitting a given structure. Two technologies were developed back then, IDCOM (Identification Command) and DMC (Dynamic Matrix Control, Qin and Badgwell (2003), presents a history of their evolution. According to a survey in Japanese Industry (Takatsu and Itoh, 1999) MPC controllers present a combination of high expectation and high technical possibility trend.

DMC controllers are designed using step response from the process about an operating condition. Then a dynamic matrix is generated based on the unit step response vector and the control horizon to be used. The resulting number of rows in the dynamic matrix is called the prediction horizon (Cutler and Ramaker, 1980). Even though no structure is fitted, the dynamic model is linear and stationary (principle of superposition is used in the controller design).

Applications of DMC are presented in boiler level control (Zou, et al., 2004), steam temperature regulation in a 300 MW thermal power plant (Sanchez-Lopez, et al., 2004), intermittently stirred, and thickness control in battery separator plants (Zhang , et al., 2004).

Accounting for process nonlinearities has been a concern ever since DMC was created. Kumar et. al. developed a discrete-time globally linearized control to account for process nonlinearities while solving a quadratic form of the DMC controller (QDMC). Guiamba and Mulholland (2004), presented an Adaptive Linear DMC for processes with integrating behavior using a holistic technique that recognizes residual integration disturbances and matrix parameter variation. Chen and Yea (2003), created a modified QDMC integrating the DMC algorithm with a neural network performing instantaneous linearization of the neural network model at each sampling time.

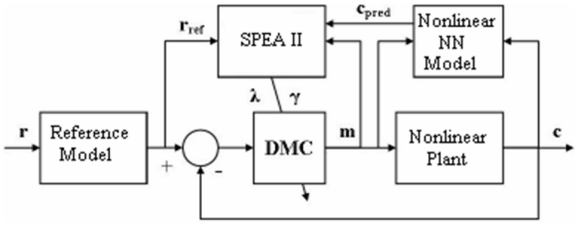


Fig. 1. Proposed adaptive strategy.

This article present an adaptation approach for a SISO DMC using neural networks as process model in the multiobjective evolutionary optimization algorithm to find the best suppressions and scaling factors at each sample time.

## 2. PROPOSED STRATEGY

Fig. 1 shows the proposed adaptive control architecture, which consists of four main components: a reference model, an evolutionary algorithm, a nonlinear neural network model and the dynamic matrix controller (DMC).

The nonlinear model was obtained trough a fast training feed-forward neural network called Random Activation Weights Neural Network (RAWN). This model is updated online trough a recursive least square (RLS) approach.

The reference followed by the DMC was a second order model instead of a step reference which in generally it is not possible to follow for most of industrial processes.

The evolutionary algorithm adapts the suppression and the scaling factor from the DMC control law using the neural network (NN) to predict the future process behavior. Every sample time the algorithm searches for the parameters that minimize the objective functions simultaneously and meet the restrictions imposed by the optimization problem.

The evolutionary algorithm found a Pareto front at each sample time, where there are many possible controllers to choose one, but a decision are made using the procedure that will be explained in section five.

## 3. NONLINEAR MODEL USING NEURAL NETWORKS

### *3.1 Discrete dynamic model structure*

Generally a nonlinear system with multiples inputs and multiple outputs (*MIMO*), with a vector input $\mathbf{u(t)}=[u_1(t)...u_n(t)]$ and an output vector $\mathbf{y(t)}=[y_1(t)...y_m(t)]^T$ can be described by the following function (Rajapakse, et al., 2002):

$$y(t) = f(\varphi(t)) + \nu(t) \tag{1}$$

Where $f=[f_1...f_m]^T$, is a nonlinear vector function. φ(t) is called the regression vector and it is composed by the past values from y(t) and u(t), which are usually called as regressors. The last term ν(t), represent the model error. Typically the function f is nonlinear and unknown and we wish to approximate it from input–output data such that ν(t) is small (Rajapakse, et al., 2002).

For this purpose there are many possible methods, such as higher order polynomials, Takagi-Suegeno or Mandani fuzzy models and neural networks, which uses input-output data and expert knowledge to achieve their job.

In this work a feed-forward neural network are used to map the nonlinear function **f** from the process. Two basic forms for the system model are used when neural networks are utilized in this approach: the Nonlinear Auto Regression with Exogenous inputs (NARX), and the Nonlinear Output Vector (NOE). The differences between them are the elements in the regression vector $\varphi(t)$.

The NARX model uses the past values from inputs and outputs in the regression vector which has the following form:

$$\varphi(t) = \begin{bmatrix} u_1(t-1),...,u_1(t-d_u),...,u_n(t-1),...,u_n(t-d_u), \\ y_1(t-1),...,y_1(t-d_y),...,y_m(t-1),...,y_m(t-d_y) \end{bmatrix}^T \tag{2}$$

where $d_u$ and $d_y$ are the maximum delay considered for every input and output respectively (Narendra and Parthasarathy, 1990)

The NOE model uses the same regression vector from the NARX model, substituting the past output values *y(t-k)* with past predicted outputs *ŷ(t-k)*. The NARX model is only used in the training phase while the NOE model is used to simulate the process in a finite horizon length in the optimization process.

### *3.2 Neural network training and architecture*

The neural network used was a standard feedforward neural network with an input layer, one hidden layer and one output layer. The neural network is called RAWNN because of the training algorithm that uses (Braake, et al., 1995). The activation function used in the hidden layer is a sigmoid like function while in the other two layers a linear activation function were used.

The hidden layer weights are calculated using normal random numbers regularized as indicated by Braake, et al. (1995), using the following regularization formula.

$$W^h = \sqrt{\frac{a}{\max_k \sum_{i=1}^{N_i+1} x_i^2(k)}}\, N(0,1) \tag{3}$$

Where, *a* is the maximum input of the activation function used, $x_i$ *is* the training data for an input, $N_i$ is the number of inputs, and *N(0,1)* is a normal random number generator with mean equal to zero and variance equal to one. The output weights ($W^o$) are calculated using standard least squares problem solution:

$$Z = X \times W^h \tag{4}$$

$$V = f(Z) \tag{5}$$

$$\hat{W}^o = (V_b^T V_b)^{-1} V_b^T Y \tag{6}$$

Where, *X* is the input training data, *f* is the activation function, $V_b$ is equal to *V* except that one column with ones is added to express output bias $b^o$, and *Y* is the output training data. Training data was collected making successive step changes to the plant input, instead of the traditional pseudo-random binary input, in order to use process industry-types identification algorithms.

### *3.3 Online training procedure*

The online adaptation for neural network is required due to process nonlinearities, disturbances, time varying parameters, aging, etc…, that can cause the model obtained with the initial training does not represent the actual behavior of the process. Once the initial training was performed, the weights updating are performed using RLS. For initialization *P* is chosen as

$$P(1) = \left(V_b^T V_b\right)^{-1} \tag{7}$$

The next sample times the weight updating law are

$$\hat{W}^o(k) = \hat{W}^o(k-1) + P(k)V(k)(c(k) - V(k)^T \hat{W}^o(k-1)) \tag{8}$$

with

$$P(k) = P(k-1) - P(k)V(k)[I + [V(k)]^T P(k-1)V(k)]^{-1} .[V(k)]^T P(k-1) \tag{9}$$

## 4. DMC CONTROLLER USING REFERENCE MODEL

### *4.1 Controller description*

The DMC control is a class of predictive control that uses a dynamic matrix *G* to predict the output of the process, over a finite prediction horizon *np,* using a number of finite input movements, *nu* (Cutler and Ramaker, 1980). The vector of the future control variables is obtained from the minimization of the following cost function:

$$J = \sum_{i=1}^{Np} \gamma(i)\left[r(t+i) - y_{pred}\right]^2 + \sum_{i=1}^{Nu} \lambda(i)\Delta m(t+i)^2 \qquad (10)$$

Where $\gamma$ is the scaling factor for error, $\lambda$ is the suppression factor, $r$ is the reference trajectory, $y_{pred}$ is the predicted output using the dynamic matrix, and $\Delta u$ are the predicted movements of the controller.

Using the dynamic matrix the process output can be obtained as follows:

$$y_{pred} = y_{past} + G\Delta m + d \qquad (11)$$

Where the vector $\boldsymbol{y}_{pred}$ is the predicted values for process output, $\boldsymbol{y}_{past}$ is the vector of past predicted values for the process output, $\mathbf{\Delta u}$ is the vector of control movements obtained in the control law, and $\mathbf{d}$ is the vector that computes the perturbations due to model error and measure errors.

The control law obtained using the objective function in (10) is:

$$\Delta m = \left(G^T \Gamma^T \Gamma G + \Lambda^T \Lambda\right)^{-1} G^T \Gamma^T \Gamma\left(r - y_{past} - d\right) \qquad (12)$$

Where $\Gamma$ y $\Lambda$ are the matrices that have in their diagonals the scaling factors and the suppression factors respectively.

The reference model is described by a discrete second order model as shown:

$$R(z) = \frac{1 - 2b\cos(a) + b^2}{z^2 - 2b\cos(a)z + b^2} \qquad (13)$$

with

$$a = T\omega_n\sqrt{1-\xi^2} \qquad (14)$$

$$b = e^{-T\omega_n\xi} \qquad (15)$$

Here $T$ is the sampling time, $\xi$ the damping factor and $\omega_n$ the natural frequency. Using this model, a step response model is obtained to be used as controller reference. The model parameters should be chosen carefully, because it defines the required closed loop response of the process.

### *4.2 Initial Controller tuning*

Tuning a DMC controller implies chose several parameters: the prediction horizon *np*, the control horizon *nu*, the sampling time *T*, the suppression and scaling factors and the step magnitude to obtain the dynamic matrix *G*. The lack of an exact analytical approach to obtain the tuning parameters for a required response, results in model approximations based tuning that have shown good results.

Given a First Order Plus Delay Time (FOPDT) model with transfer function of the form:

$$G_p(s) = \frac{K_p e^{-\theta_p S}}{\tau s + 1} \qquad (16)$$

Shridhar and Coop (1997) shows equations for tuning a DMC controller, which are summarized in Table 1.

The sample time selection needs to be made carefully because a sample time too short increase the computational time that is not always reflected in a better controller and a too long sample time could not capture the systems dynamics and decrement the control quality.

Table1. Tuning equations for dynamic matrix controller

| Parameter | Equation |
|---|---|
| $T$ | $\max(0.1\tau, 0.5\theta_p)$ |
| $k$ | $Int\left(\frac{\theta_p}{T}+1\right)$ |
| $np$ | $Int\left(\frac{5\tau}{T}+k\right)$ |
| $nu$ | $Int\left(\frac{\tau}{T}+k\right)$ |
| $\lambda$ | $\lambda = \frac{nu}{500}\gamma^2 K_p{}^2\left(np - k - \frac{3\tau}{2T} + 2 - \frac{nu-1}{2}\right)$ |

The suppression factors adjust the controller aggressiveness, modifying the closed loop response of the system. If very small values are chosen the controller could be unstable, but in the other way, if very high vales are chosen an excessive slow response are obtained. For that reason, intermediate values are preferred and for start the adaptive strategy the formula shown in Table 1 are used.

## 5. MULTIOBJETIVE EVOLUTIONARY OPTIMIZATION

### *5.1 Multiobjective optimization*

The use of a multiobjective optimization recognizes that in many practical problems, a number of design objectives must be satisfied simultaneously. The optimization problem can be formulated as:

$$\min_{x\in\Omega} F(x) \qquad (17)$$

Where $\mathbf{x} = [x_1, x_2, ..., x_m]$ and $\Omega$ defines the set of *m* free variables, $\mathbf{x}$, subject to any constraints. $\mathbf{F(x)}=[f_1(\mathbf{x}), f_2(\mathbf{x}), ..., f_n(\mathbf{x})]$ is the set of *n* objective functions to be minimized (Chipperfield and Fleming, 1996).

The concept of dominance is crucial in this type of

problems, because its reformulate the concept of optimality of a solution. Given two solutions $\mathbf{x}^1$and $\mathbf{x}^2$, if $f_i(\mathbf{x}^1) <= f_i(\mathbf{x}^2)$ and $f_k(\mathbf{x}^1) < f_k(\mathbf{x}^2)$ para $k < n \wedge k \neq i$, its saids that the solution $\mathbf{x}^1$ dominates $\mathbf{x}^2$. In other words, the value of the objective functions for $\mathbf{x}^1$ are equal, or less in almost one, than the objectives functions for $\mathbf{x}^2$.

Clearly, for the set of functions, **F**(x), there are not an ideal optimal solution that minimize simultaneously all the objectives functions, rather a set of Pareto-optimal solutions for which an improvement in one of the design objectives will lead to a degradation in one or more of the remaining objectives. These solutions are known as nondominated solutions for the multiobjective optimization problem (Chipperfield and Fleming, 1996).

In this paper, an evolutionary approach have been used, using all the advantage of evolutionary algorithms like noise tolerance, time varying parameters and discontinuities on the objective functions and the fact that this algorithms can search in the entire space of feasible solutions thanks to its stochastic nature.

### 5.2 *Multiobjective evolutionary algorithm : SPEA II*

*SPEA* (Strength Pareto Evolutionary Algorithm) is an algorithm that uses the concept of strength of a solution that is proportional to the number of elements that dominate and that it dominate. In the same way, use the elitism, having a population called archive that does not modifies with the variations operator and survives until the next generation to be compared with the modified population (Zitzler, et al., 2001).

In this type of algorithms there are four crucial aspects that depend of each problem specifically: the chromosome coding, the crossover and mutation operators, the objective functions and the restrictions to be evaluated.

**Chromosome coding, crossover and mutation operators***:* for this research a real valued coding is used. Here the chromosome is coded as vector of floating point numbers of the same length as the solution vector, and each element is forced to be within the feasible region. This approach has two advantages; the first one is the precision of the solution because no code or decode has made, and the second is the low computational load because the reduced amount of memory space and the simplicity of operators (Sakawa, 2002)

The crossover operator used was the heuristic crossover (Sakawa, 2002), which has the following form:

$$z = a(w - v) + w \tag{18}$$

where $a$ is a random number between 0 and 1, $z$ is the resultant individual, and the parent $w$ is not worse than $v$; that is **Fitness(w) < Fitness(v)**. For the case of using the SPEA the fitness of nondominated solutions are less than one and the fitness of dominated solutions are greater than one.

The second crossover operator was the arithmetic crossover (Sakawa, 2002), described as follows:

$$z = aw + (1 - a)v \tag{19}$$

Now $a$ is a random number between 0 and 1, $z$ is the resultant individual with parents $w$ and $v$ from the population.

The mutation operator used was the non-uniform mutation [13], which can be described with the following functions:

$$z = \begin{cases} v + [u(v) - v] \times \Gamma(t) & if \quad r_1 < 0.5 \\ v - [v + l(v)] \times \Gamma(t) & if \quad r_1 \geq 0.5 \end{cases} \tag{20}$$

where

$$\Gamma(t) = \left(r_2 \left(1 - t/T\right)\right)^b \tag{21}$$

In equations (20) and (21) $v$ is the individual to be modified, $r_1$ and $r_2$ uniform random numbers between 0 and 1, $b$ is a constant parameter, $t$ is the number of the actual generation, $T$ is the total number of generations, $u(v)$ and $l(v)$ are the upper and lower bounds of the individual and $z$ is the modified individual

**Objective functions and decision making system:** the objective functions to be minimized and restrictions for the optimization procedure are summarized in Table 2.

In Table 2 $e_{pred}$ are calculated using the neural network as process model. The restrictions are used to assure a good search space for the multiobjective optimization tool. The last two restrictions are focus in the control loop stability. The decision making process about the Pareto front needs to be automated because the algorithm needs to run every sampling period. To achieve this goal a rather simple decision making process is implemented using the following function:

$$P_s' \in \langle P_i' \mid i = 1 \ldots N' \rangle \wedge F_s = \left\langle \min\left( \sum_{j=1}^{n} f_{ji} \right) \middle| \, i = 1 \ldots N' \right\rangle \tag{22}$$

where $F_s$ is the sum of the values of every function associated to a point $s$ in the Pareto front, represented by the solution $P'_s$, where $f_{ji}$ is the value from each function evaluated, $N$ ' is the size of archive set in SPEA, and $n$ the number of problem functions.

Table2. Objective functions and restrictions for optimization process

| Objective functions | Restrictions |
|---|---|
| $f_1(\lambda,\gamma) = \sum_{i=1}^{np} e_{pred}^{\,2}$ | $\Delta\lambda_{\min} \leq \Delta\lambda \leq \Delta\lambda_{\max}$<br>$\Delta\gamma_{\min} \leq \Delta\gamma \leq \Delta\gamma_{\max}$ |
| $f_2(\lambda,\gamma) = \sum_{i=1}^{nu} \Delta u^2$ | $\lvert\Delta m\rvert \leq \Delta m_{\max}$<br>$\lambda \geq \lambda_{\min}$<br>$\gamma \geq \gamma_{\min}$ |

## 6. SIMULATIONS AND RESULTS

The process used to apply the strategy is the pH neutralization reactor, which has been chosen largely in the literature to evaluate nonlinear control strategies due to its highly nonlinear behavior. The open-loop system is

illustrated in fig. 2. For the single-input single-output (SISO) analysis, the process input is the signal to the base stream valve, $m_1(t)$, the process output is the pH sensor signal, $c_1(t)$, and the main disturbances are the inlet acid stream flow, $q_1(t)$, and the inlet buffer stream flow, $q_2(t)$. Underlying assumptions are constant densities (due to low molar concentrations), perfect fixing, and constant inlet concentrations.

The initial population of individuals was created using the initial controller tuning and adding a uniform random noise. The parameters for the optimization process are in Table 3.

The FOPDT model and the dynamic matrix, for initial controller tuning was obtained using a -5%CO step change at process input.

To evaluate the effectiveness of the strategy two controllers were used to prove the improvement in the performance. A modified PID with derivative filter and reset windup prevention, and a non adaptive DMC were tuned using the same FOPDT approximation of the process. For DMC the Table 1 was used and for PID controller tables for minimum IAE set point tracking were used (Smith and Corripio, 1997). The final tuning parameters for each controller are found in Table 4.

In Table 5 the result for the tests are shown. It can be seen that the adaptive strategy in the case of set point tracking are better that the others but in case of disturbance rejection the PID shows a better response. The DMC and the adaptive DMC shows a non oscillatory response for set point tracking and disturbance rejection, in the opposite from PID, this can be seen in fig. 3 and fig 4. The inherent noise rejection from the adaptive algorithm without the uses of a filter in the input are greatly appreciable in fig. 4 were the noisy sensor signal instead of the real pH signal are shown.

## 7. CONCLUSIONS

The use of adaptive control strategies is reasonable in highly non-linear processes, where linear or static tuning strategies become either sluggish or unstable as the process moves away from the initial operating condition. This research, as similar research developed for other controllers, demonstrates that adaptation is an alternative that becomes useful and effective for predictive controllers. It also shows that for linear or time-invariant processes the value added by adaptation is negligible. The computational load it is one of the main problems for this strategy if we tray to implement it in a fast real process, but for chemical process like the one used in this research it is not a problem. For another kind of processes, the use of multiobjective evolutionary strategy to calculate the control moves should resolve the problem.

Further research will address the issue of extending the adaptive strategy based on evolutionary learning to the multivariate scenario and the inclusion of a better decision making system for pareto front, also the modification to calculate the control moves using the evolutionary strategy as optimizer.

TABLE3. OPTIMIZATION PARAMETERS FOR SIMULATION

| Spea II parameters | | | | |
|---|---|---|---|---|
| $p_c$ | $p_m$ | $N$ | $N'$ | $T$ |
| 0.8 | 0.3 | 20 | 10 | 10 |

| Constraints parameters | | | | | | |
|---|---|---|---|---|---|---|
| $\Delta\lambda_{min}$ | $\Delta\lambda_{max}$ | $\Delta\gamma_{min}$ | $\Delta\gamma_{max}$ | $\lambda_{min}$ | $\gamma_{min}$ | $\Delta m_{max}$ |
| 10 | 2 | 2 | 10 | 0.2 | 1 | 25%CO |

TABLE4. INITIAL TUNING FOR DMC AND PID CONTROLLERS

| PID tuning | | | |
|---|---|---|---|
| $Kc$ | $t_i$ | $t_d$ | $\alpha$ |
| 6.2674 | 40.9969 | 9.2076 | 0.2 |

| DMC tuning | | | | |
|---|---|---|---|---|
| $T_s$ | $np$ | $nu$ | $\lambda$ | $\gamma$ |
| 11 | 47 | 47 | 0.4238 | 1 |

TABLE5. PERFORMANCE COMPARISON FOR TEST CONTROLLERS

| | IAE | | |
|---|---|---|---|
| **TEST** | **PID** | **DMC** | **ADMC** |
| -5% set point | 493.6 | 539.6 | 466.7 |
| +20% set point | 1429 | 1444 | 1227 |
| Disturbances | 41 | 173.1 | 128.7 |

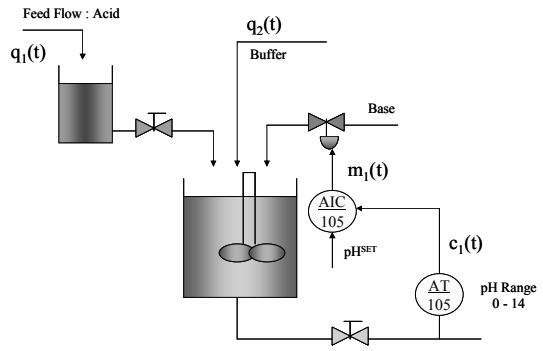


Fig. 2. pH neutralization reactor.

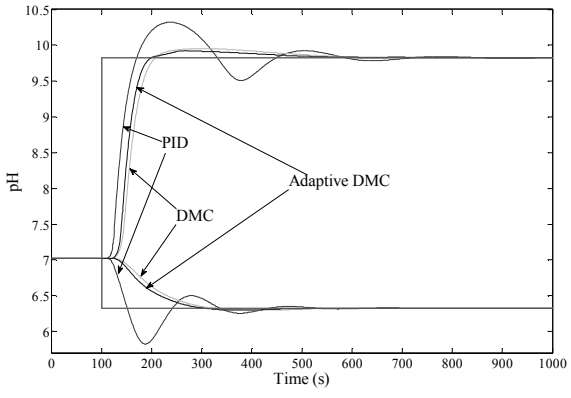


Fig. 3. Process response for set point tracking from various controllers.

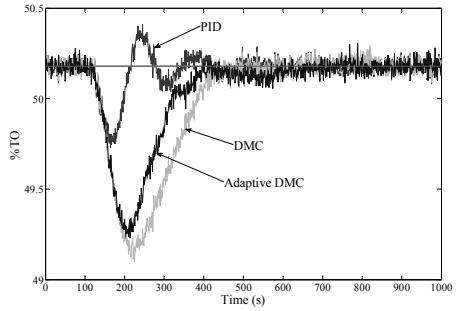


Fig. 4. Process response for a 5% change in acid flow and sensor noise.

REFERENCES

Braake H.A.B, van Can H.J.L and van Straten G. (1995). *Random Activation Weight Neural Net (RAWN) for Fast Non-iterative Training. IFAC Journal of Engineering Applications of Artificial Intelligence,* 8(1), p 71-80**.**

Chen J. and Yea Y. (2003). *Modified QDMC based on instantaneous linearization of neural network models in nonlinear chemical processes. Journal of Chemical Engineering of Japan,* 36, p 198-209**.**

Chipperfield A. and Fleming P. (1996). *Multiobjective Gas Turbine Engine Controller Design Using Genetic Algorithms. IEEE Transaction on Industrial Electronics,* 43(5), p 583-587**.**

Cutler C. and Ramaker B. (1980). *Dynamic Matrix Control – A Computer Control Algorithm. Proceedings of the Joint Automation Control Conference,* 1**,** 6p N WP5-B**.**

Guiamba I. and Mulholland M. (2004). *Adaptive Linear Dynamic Matrix Control applied to an integrating process. Computers and Chemical Engineering,* 28, p 2621-2633**.**

Kumar Jana A., Nath Samanta A. and Ganguly S. (2005). *Globally linearized control system design of a constrained multivariable distillation column. Journal of Process Control,* 15, p 169-181.

Narendra K. and Parthasarathy K. (1990). *Identification and control of dynamical systems using neural networks. IEEE Transactions on Neural Networks,* 1(1), p 4-27.

Rajapakse A., Furuta K. and Kondo S (2002). *Evolutionary Learning of Fuzzy Logic Controllers and their adaptation through Perpetual Evolution. IEEE Trans. on Fuzzy Systems,* 10 (3), p 309-321.

Qin, S. and Badgwell T. (2003). *A Survey of Industrial Model Predictive Control Technology. Control Engineering Practice,* 11, p 733-764.

Sakawa M. (2002). *Genetic Algorithms and fuzzy multiobjective optimization*. *Springer,* Kluwer Academic Pub. Norwell-Massachusetts.

Sanchez-Lopez A., Arroyo-Figueroa G. and Villavicencio-Ramirez A. (2004). *Advanced control algorithms for steam temperature regulation of thermal power plants. International Journal of Electrical Power and Energy System***, 26**, p 779-785.

Shridhar R. and Coop D.J (1997). *Selection of the move suppression coefficients in tuning dynamic matrix control. Proceedings of the American Control Conference***,** Albuquerque-New Mexico, p 729-733.

Smith, C.A. and A. Corripio (1997). *Principles and Practice of Automatic Process Control*. John Wiley and Sons, Inc. New York.

Takatsu H. and Itoh T. (1999). *Future needs for control theory in industry - report of the control technology survey in Japanese industry. Transactions on Control Systems Technology,* **7**, p 298-305.

Zhang Z., Shen Y. and Wang Y. (2004). *Measure and DMC control of battery separator thickness uniformity. Proceedings of the 8th International Conference on Control, Automation, Robotics and Vision,* China, p 851-854.

Zitzler E., Laumanns M. and Thiele L. (2001). *SPEA2: Improving the Strength Pareto Evolutionary Algorithm. Proceedings of the 8th International Conference on Control, TIK-Report 103,* Swiss Federal Institute of Technology (ETH), Zurich, Switzerland.

Zou T, Liu H. and Li. S. (2004). *Dynamic Matrix Control Algorithm in the Boiler Level Integrating Process. Control Theory and Applications,* 21, p 386-390.