OPTIMAL TUNING PARAMETERS OF THE DYNAMIC MATRIX PREDICTIVE CONTROLLER WITH CONSTRAINTS

G. M. DE ALMEIDA, J. L. F. SALLES and J. D. FILHO

UFES - Univ. Federal do Espirito Santo - Departamento de Eng. Elétrica
Av. Fernando Ferrari C. P. 019001 - Vitória, ES, Brasil
gmaialmeida@yahoo.com.br, jleandro@ele.ufes.br, jdenti@ele.ufes.br

Abstract — Dynamic Matrix Control Algorithm is a powerful control method widely applied to industrial processes. The idea of this work is to use the Genetic Algorithms (GA) with the elitism strategy to optimize the tuning parameters of the Dynamic Matrix Controller for SISO (single-input single-output) and MIMO (multi-input multi-output) processes with constraints. A comparison is made between the computational method proposed here with the tuning guidelines described in the literature, showing advantages of the GA method.

Keywords — Genetic Algorithm, Dynamic Matrix Control, Automatic Tuning, Constraints Management.

I. INTRODUCTION

Model Predictive Control (MPC) refers to a class of computer control algorithms that utilize an explicit process model to predict the future response of the plant (Qin and Badgwell, 2003). A variety of processes, ranging from those with simple dynamics to those with long delay times, non-minimum phase zeros, or unstable dynamics, can all be controlled using MPC. MPC integrates optimal, stochastic, multivariable, constrained control with dead time processes to represent time domain control problems (see Camacho and Bordons, 2004; Maciejowski, 2002; and Rossiter, 2003).

The MPC algorithms usually exhibit very good performance and robustness provided that the tuning parameters (prediction and control horizons and move suppression coefficient) have been properly selected. However, the selection of these parameters is challenging because they affect the close loop time response, being able to violate the constraints of the manipulated and controlled variables. In the past, systematic trial-and-error tuning procedures have been proposed (see Maurath et al., 1988; Rawlings and Muske, 1993; and Lee and Yu, 1994). Recently there are some works proposing alternative techniques of adjusting them automatically. In Filali and Wertz (2001) and Almeida et al. (2006) it was used the Genetic Algorithms (GA) to design the tuning parameters of the Generalized Predictive Control (GPC) applied in SISO and time-varying systems without constraints. Another tuning strategy that can be implemented in a computer was proposed by Dougherty and Cooper (2003) which developed easy-to-use tuning guidelines for the Dynamic Matrix Control (DMC) algorithm, one of the most popular MPC algorithms in the industry. DMC uses the step response to model the process, and it is also applied in SISO and MIMO processes that can be approximated by first order plus dead time models.

The main goal of this work is to tune the DMC controller parameters for SISO and MIMO dynamical systems with input and output constraints using GA, in order to optimize the time response specifications (overshoot, rise time and steady state error) and to guarantee feasibility of solutions. It also carries through a comparative study between the method proposed here and the tuning guidelines proposed by Dougherty and Cooper (2003).

This paper is organized as follows: Section 2 describes the classical formulation of the DMC algorithm with constraints and presents the automatic tuning guidelines proposed by Dougherty and Cooper (2003); Section 3 makes an overview of GA and shows the application of GA with elitism strategy for tuning of the DMC parameters; in Section 4 we make the comparison between the proposed method here and the method described in the literature; Section 5 presents the conclusions.

II. DYNAMIC MATRIX CONTROL (DMC)

In this section we formulate the predictive control problem and we present its solution from the DMC algorithm.

Let us consider a linear MIMO dynamic system with \( m \) inputs \((u_l, l=1,...,m)\) and \( n \) outputs \((y_j, j=1,...,n)\), that can be described by the expression:

\[
y_j(k) = \sum_{i=0}^{n} \sum_{q=1}^{m} g_{ij}(k) u_l(k-q)
\]  

(1)

where \( g_i \) is the step response of output \( j \) with respect to the input \( l \), and \( \Delta u(k) = u(k) - u(k-1) \), is the control signal variation. Let us denote by \( \hat{y}_j(k+i) \) the \( i \) step ahead prediction of the output \( j \) from the actual instant \( k \), and \( r_j(k+i) \) is the \( i \) step ahead prediction of the set point with respect to the output \( j \).

The basic idea of DMC is to calculate the future control signals along the control horizon \( h_c \), i.e., determine the sequence \( u(k+i) \), for \( l = 1, 2, ..., m \) and \( i = 0, ..., h_c - 1 \), in such a way that it minimizes the cost function defined by:

\[
\sum_{l=0}^{m} \sum_{i=1}^{h_c} (\hat{y}_j(k+i) - r_j(k+i))^2 + \sum_{l=1}^{m} \sum_{i=1}^{h_c} \lambda_l \Delta u_l^2(k+i)
\]  

(2)
where $\lambda$ is the move suppression parameter and $h_p$ is the prediction horizon. In addition, the following constraints must be satisfied for $i = 1, ..., n$, and $j = 1, ..., n$:

$$
\Delta u^m_i = \frac{\Delta u_i(k + i)}{\Delta u_i^{	ext{max}}, i = 0, \ldots, h_i - 1,}
$$

$$
u^m_i \leq u_i(k + i) \leq u^m_i, i = 0, \ldots, h_i - 1, (3)
$$

$$y^m_j \leq \hat{y}_j(k + i) \leq y^m_j, i = 0, \ldots, h_p.$$

Considering $\Delta u_i(k+i) > 0, i > h_i, i \geq 1$, and assuming that exists an integer number $N_i$ such that $\Delta u_i(k) = 0$ for $k > N_i$, and $l=1, \ldots, m$, the predictive output form the actual instant $k$ is given by the following convolution:

$$
\hat{y}_j(k + i) = \sum_{l=1}^{n} \sum_{q=\min[1, h_q, \ldots, h_q]}^{N_q} g_i(q) \Delta u_i(k + i - q) \quad (4)
$$

Disconnecting in (4) the future control actions ($\Delta u_i(k+i); j > 1$) from the past control actions, and defining:

$$
G_i(i) = \begin{bmatrix} g_i(i) & g_{i+1}(i) & \cdots & g_{n}(i) \end{bmatrix}
$$

$$
f_j(k+i) = \sum_{l=1}^{n} g_i(q) \Delta u_i(k + i - q) \quad (5)
$$

$$\Delta u_i(k+i) = \begin{bmatrix} \Delta u_i(k+i) & \Delta u_i(k+i) & \cdots & \Delta u_i(k+i) \end{bmatrix}\quad (6)
$$

we can represent the expression (4) by:

$$
\hat{y}_j(k + i) = \sum_{q=\min[1, h_q, \ldots, h_q]}^{\infty} G_i(q) \Delta u_i(k + i - q) + f_j(k+i) \quad (6)
$$

where the first term in expression (6) is the forced response, that depends on the present and future control actions, and the second is the free response, that depends on the past control actions.

In the sequel we formulate the control problem as a standard quadratic programming problem with constraint. Let us define the following vectors:

$$
\Delta \Theta = \begin{bmatrix} \Delta u_i(k) & \Delta u_i(k+1) & \cdots & \Delta u_i(k+h_i-1) \end{bmatrix},
$$

$$\hat{y}(k) = \begin{bmatrix} \hat{y}_1(k) & \hat{y}_2(k) & \cdots \hat{y}_n(k) \end{bmatrix},
$$

$$\bar{y} = \begin{bmatrix} \bar{y}(k+1) & \bar{y}(k+2) & \cdots \bar{y}(k+h_j) \end{bmatrix},
$$

$$f(k) = \begin{bmatrix} f_1(k) & f_2(k) & \cdots & f_n(k) \end{bmatrix},
$$

$$r(k) = \begin{bmatrix} r_1(k) & r_2(k) & \cdots & r_n(k) \end{bmatrix},
$$

$$\bar{f} = \begin{bmatrix} \bar{f}(k+1) & \bar{f}(k+2) & \cdots \bar{f}(k+h_p) \end{bmatrix},
$$

$$\bar{r} = \begin{bmatrix} \bar{r}(k+1) & \bar{r}(k+2) & \cdots \bar{r}(k+h_p) \end{bmatrix},
$$

$$G_i(i) = \begin{bmatrix} G_1(i) & G_2(i) & \cdots & G_n(i) \end{bmatrix}.
$$

Using the definitions above, we can represent the $m$ predicted outputs along the prediction horizon $h_p$ by the expression:

$$
\bar{y} = \bar{G} \Delta \Theta + \bar{f} \quad (8)
$$

where $\bar{G}$ has dimension $n h_p \times n h_p$ and is given by:

$$
\bar{G} = \begin{bmatrix} G(1) & 0 & \cdots & 0 \\
G(2) & G(1) & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
G(h_p) & G(h_p-1) & \cdots & G(1) \\
\end{bmatrix}.
$$

Let us denote by $\bar{\Theta}$ a $m \times m$ diagonal matrix with move suppression parameters $h_p, j = 1, \ldots, m$, on its diagonal, and let $\bar{A}$ a $m h_p \times m h_p$ matrix defined by:

$$
\bar{A} = \begin{array}{c c c c c}
\bar{\Theta} & 0 & \cdots & 0 \\
0 & \bar{\Theta} & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & \bar{\Theta} \\
\end{array}.
$$

From (8), we can represent the cost function (2) by the following expression:

$$
\frac{1}{2} \Delta \Theta^T \bar{\Theta} \Delta \Theta + \bar{\Theta} \Delta \Theta + \bar{\Theta} = 2(\bar{G}^T \bar{A} \bar{y} + \bar{G}^T \Delta \Theta + \bar{f}) \quad (9)
$$

where $\bar{H} = 2(\bar{G}^T \bar{A} \bar{y} + \bar{G}^T \Delta \Theta + \bar{f})$, and $\bar{f} = (\bar{f} - \bar{f})^T H$ and $\bar{f} = (\bar{f} - \bar{f})^T (\bar{f} - \bar{f})$.

Applying recursively the fact that $u_i(k+i) = u_i(k+i)+1 + \Delta u_i(k+i-1)$ in the second restriction of (3) and considering (4) in the third inequality of (3), we conclude that (3) is equivalent to:

$$
\Delta u^m_i \leq \Delta u_i(k + i) \leq \Delta u^m_i, i = 0, \ldots, h_i - 1, (10)
$$

$$
u^m_i - u_i(k - 1) \leq \Delta u_i(k) + \Delta u_i(k + 1) + \cdots + \Delta u_i(k + i - 1), i = 0, \ldots, h_i - 1, (10)
$$

$$y^m_j - f_j(k + i) \leq \sum_{q=\min[1, h_q, \ldots, h_q]}^{\infty} G(q) \Delta u_i(k + i - q) \leq y^m_j - f_j(k + i), i = 0, \ldots, h_p.$$

The solution of the Predictive DMC control problem is divided in two step: first we calculate the prediction vector $\Delta \Theta$ by minimizing the quadratic function (9) with $2 m h_p + n h_p$ constraints given in (10). Next apply the current signal $\Delta u_i(k)$ in the input of the processes and we discard the future signals $\Delta u_i(k+i), i = 1, \ldots, h_i - 1$ (this is called receding horizon control technique).

**A. Comments about the DMC parameters**

The relative importance of each one term in the cost function (2) can be reinforced by increasing the move suppression parameter ($\lambda$), the control horizon ($h_i$) and the prediction horizon ($h_p$) (Garcia et al., 1988). However the increment in these parameters affects the time response of the output $y(k)$ (overshoot, rise time and steady state error; Garcia et al., 1988).

An increment in the parameter $\lambda$ decreases the control variables, making the time response slower and softer. The variations in this parameter also affects the robustness of the DMC controller (the increment in the parameter $\lambda$ increases the robustness; Georgeou et al., 1988).
An increment in the control horizon (and consequently in the prediction horizon, since \( h_c \leq h_p \)) makes an improvement in the performance of the DMC controller, increasing the controlled variables, but decreasing the robustness of the controller, (see Georgeou et al., 1988). It is also observed in Georgeou et al. (1988) that the tuning of the parameter \( \lambda_j \) by keeping fixed the control and prediction horizons does not lead to good results; but the contrary case makes the adjustment more feasible. This fact shows that the prediction and the control horizons have a higher adjustment priority than the parameter \( \lambda_j \).

It is also observed that the constraint in the control signal variations (\( \Delta u \)) affects directly the speed of the output response and this constraint has an higher adjustment priority than the parameter \( \lambda_j \) (see Prett and Garcia, 1988). On the other hand, the constraints in the quadratic programming problem increase proportionally with the prediction and control horizons, increasing the complexity of the calculations which favor the obtaining of unfeasible solutions. Thus, it is so important to consider these facts to make the best tuning of the DMC parameters.

**B. Tuning Guidelines**

The methods encountered in the literature for tuning the DMC parameters are mostly based on heuristic rules and trial and error, see Camach and Bordons (2004) and Maciejowski (2002). Only in Dougherty and Cooper (2003) is derived an analytic expression to determine the move suppression parameter \( \lambda_j \) for mono and multivariable systems, in order to obtain an automatic tuning method. The guidelines proposed by Dougherty and Cooper (2003) is described in next steps:

**Step 1.** Approximate the process dynamics of controller output to measured controlled variables pairs of non-integrating sub-processes with the first order plus dead time (FOPDT) model:

\[
y_j(s) = \frac{K_j e^{-(\theta_j)\tau_j}}{\tau_j s + 1}
\]

for \( l = 1, 2, ..., m; j = 1, 2, ..., n \).

**Step 2.** Select the sample time as close as possible to:

\[
T = \min \left[ \max (0.1 \tau_j, \theta_j) \right]
\]

**Step 3.** Compute the prediction horizon, \( h_p \), and the model horizon \( N_c \):

\[
h_p = N_c = \max \left[ \frac{5 \tau_j}{T} + k_y \right]
\]

where \( k_y = \text{Int} \left( \frac{\theta_j}{T} + 1 \right) \).

**Step 4.** Compute the control horizon, \( h_c \):

\[
h_c = \max \left[ \frac{\tau_j}{T} + k_y \right]
\]

**Step 5.** Select the controlled variable weights, \( \gamma^j \), to scale the process unit to be the same.

**Step 6.** Compute the move suppression coefficient, \( \lambda^j \):

\[
\lambda^j = \frac{h_c}{10} \sum_{j=1}^{m} \gamma^j K^2 \left( h_p - k_y - \frac{3 \tau_j}{2} \right) + 2 - \frac{(h_c - 1)}{2}
\]

**Step 7.** Implement DMC using the traditional step response matrix of the actual process and the initial values of the parameters computed in steps 1-6.

**III. GENETIC ALGORITHMS**

Genetic algorithms were first presented by Holland (1975) and are based on the mechanics of natural selection of Darwin and the natural genetics of Mendel. They are a parallel global search technique that emulate natural genetic operators and work on a population representing different parameter vectors whose optimal value with respect to some criterion (fitness) is searched. This technique includes operations such as reproduction, crossover and mutation. These operators work with a number of artificial creatures called generation. By exchanging the information of each individual in a population, GA preserves the best individual and yields higher fitness in each generation such that the performance can be improved. The basic operators of GA are:

- **Crossover:** This operation provides a mechanism to exchange the information of the individuals via probabilistic process. This operation takes two “parents” individuals and produces two “off-springs” which are new individuals whose characteristics are a combination of those of their parents. The operation of crossover can be seen in the Fig. 1.

- **Mutation:** Mutation is an operation where some characteristics of an individual are randomly modified, yielding a new individual. Here, the operation simply consists in changing the value of one bit of the string representing an individual in a random way. This operation is shown in the Fig. 2.

**Reproduction:** Reproduction is the process in which a new generation is obtained by selecting individuals from an existing population, according to their fitness. It takes one or more copies of the individuals with higher fitness values in the next generation, while individuals with lower fitness may have none. Note, however, that reproduction does not generate new individuals but only favors the percentage of fittest individuals in a population of given size (it is called elitism strategy). The Fig. 3 shows the reproduction operator.

![Figures 1, 2, and 3 showing Genetic Algorithm operations](image-url)
A. Convergence of the Genetic Algorithm

In the field of GA and convergence behavior, we could distinguish two large areas of research. The first line of research results from the theoretical investigations by Holland (1975) and Goldgerb (1989) and is based on the schemata theorem and the Walsh functional analysis. However, the global searches of crossover and mutation are not considered possible under the schemata theorem. In addition the Walsh functional analysis has been used to evaluate the “phenomenon of deception”, but it can never analysis the general behavior of GAs. The second line of research treats GAs as a Markov Chain, (see Schafer and Telega, 2007). In Rudolph (1997), it is proved the global convergence of GA algorithm with the elitism strategy and in Schmitt and Rothlauf (2001) it is investigated the determinants which the convergence rate depends on. They showed that it depends mainly on the value of the second largest eigenvalue of the transition matrix of the Markov Chain. However the authors could not explain how this matrix is affected by the GA parameters, like mutation and crossover rates, in order to improve the convergence of GA algorithm.

There are a great deal of discussions of parameter settings and approaches to parameters adaptation in the evolutionary computation literature. In Srinivas and Patnaik (1994) it is heuristically proposed that the population crossover rate and the mutation rate must lie in the intervals [0.5,1.0] and [0.001,0.05], respectively.

B. Tuning by GA

The individual of GA is given by the vector of tuning parameters \( [h_p, h_c, \lambda_1, \ldots, \lambda_m] \). For each one we associate the fitness function given by:

\[
\begin{align*}
&f(x) = \sum_{i=0}^{m} \left( \sum_{j=1}^{n} \left( y_j(k) - r_j(k) \right)^2 \right)^{-1} + \\
&\left[ l + \sum_{i=0}^{m} \left( \Delta u_i(k) \right)^2 \right]^{-1}
\end{align*}
\]

(12)

where \( y_j \) is the output, \( r_j \) is the set point, \( S_i \) is the set of constraints given in (10) and \( I_{\{s<4\}} \) is the indicator function of a set \( A \), i.e., if \( x \in A \) then \( I_{\{x<4\}} = 1 \), otherwise \( I_{\{x<4\}} = 0 \). The first term in (12) is the inverse of the integrated square-error (ISE) and the second term penalizes the unfeasible solutions.

Next the GA program calculates the fitness function of each individual and selects the best parents according to the roulette wheel method. Then, it generates the new individuals by executing the crossover, mutation and reproduction operators. The procedure is summarized in the following steps:

**Step 1:** Define the population size, the maximum generation, the crossover rate and the mutation rate;

**Step 2:** Create an initial population;

**Step 3:** Calculate \( \Delta u(k) \) by solving the Linear Quadratic Programming problem defined by the function (9) and the constraints (10);

**Step 4:** Calculate the fitness of each individual from (12);

**Step 5:** Select the parents individuals from the roulette wheel method;

**Step 6:** Create a new population by applying the genetic operators (crossover, mutation and reproduction);

**Step 7:** Go to step 3 until the maximum generation is reached.

IV. NUMERICAL RESULTS

In this section we compare the Guideline method presented in Section II.B with the tuning method proposed in section III.B. The algorithms were implemented in a Pentium Processor with 2.8 GHZ and 512 MB. We used the MatLab 7.0 and the optimization problem was solved by the Quadriprog routine. We apply the DMC algorithm in SISO and MIMO systems with constraints in the control action \( u(k) \), in the control signal varia-

\[
\begin{align*}
&-1 \leq u_i(k) \leq 1, \\
&-0.1 \leq \Delta u_i(k) \leq 0.1, \\
&0 \leq y_i(k) \leq 1.0
\end{align*}
\]

The population generated by the GA program has 50 individuals and the rates of crossover, mutation and reproduction are equal to 0.8, 0.01 and 0.1, respectively.

A. SISO System

The first example is a first order plus dead time model, represented by the transfer function:

\[
G(s) = \frac{3e^{-10s}}{100s + 1}.
\]

The Table 1 compares the tuning parameters \( (h_p, h_c, \lambda) \) and the total number of constraints (NC) in the quadratic programming problem determined by both methods. We observe from (10) that NC=2mhc+nhp, where \( n \) and \( m \) are the number of output and control signals respectively. The Fig. 4 presents the time responses of signals \( y(t), u(t) \) and \( \Delta u(t) \) by considering the step input and the parameters in Table 1.

<table>
<thead>
<tr>
<th>Method</th>
<th>( h_p )</th>
<th>( h_c )</th>
<th>( \lambda )</th>
<th>NC</th>
</tr>
</thead>
<tbody>
<tr>
<td>GA</td>
<td>19</td>
<td>1</td>
<td>88.13</td>
<td>21</td>
</tr>
<tr>
<td>Guidelines</td>
<td>52</td>
<td>12</td>
<td>340.20</td>
<td>76</td>
</tr>
</tbody>
</table>

Figure 4: Time Responses of signals \( y, u \) and \( \Delta u \)
The Fig. 5 shows the evolution of the mean fitness of the population at each generation. We observe that it converges to the fittest individual at the twentieth generation, which characterizes a global convergence of the Genetic Algorithm with the elitism strategy as we discussed in Section 3.A. The time for convergence of the proposed algorithm in Section 3.B is about 101.3 s. for the SISO system and 119.12 s. for the MIMO system. The half of these processing time is spent by the Quadruprog routine (step 3 in the algorithm of Section 3.B).

**B. MIMO System**

The second example exhibits a multivariable system with two inputs \( m = 2 \) and two outputs \( n = 2 \), represented by the transfer function matrix

\[
\begin{bmatrix}
  y_1(s) \\
  y_2(s)
\end{bmatrix} = \begin{bmatrix}
  12.8e^{-s} & -18.9e^{-3s} \\
  16.7s+1 & 21.0s+1 \\
  6.6se^{-7s} & -19.4e^{-3s} \\
  10.9s+1 & 14.4s+1
\end{bmatrix} \begin{bmatrix}
  u_1(s) \\
  u_2(s)
\end{bmatrix}. \tag{15}
\]

The Table 2 compares the tuning parameters and the total number of constraints (NC) for the MIMO system. The Figs. 6 and 7 show the time response presented by both methods, considering the step input and the tuning parameters described in the Table 2.

We observe in Tables 1 and 2 that the GA method furnishes lesser prediction and control horizons than the Tuning Guidelines. Thus, from the comments in Section 2.A, we conclude that the DMC algorithm tuned by GA is more robust with respect to the variations in the model parameters than the DMC algorithm tuned by the Guidelines method (despite of sometimes the parameter \( \lambda \) furnished by the guidelines method be larger than that given by the GA method). This can be illustrated in Table 3 where we compare the ISE index of the SISO system for both tuning methods when the gain \( K = 3 \) of the model is changed.

**V. CONCLUSIONS**

The comparative study made in this paper, between the two automatic tuning methods, allowed us to conclude important advantages in favor of GA with the elitism strategy. The first is that it generated a lesser number of control and prediction horizons, improving the robustness of the DMC controller without the depreciation of the step time response (overshoot, rise time and steady state error). The second is that the GA method gave a lesser number of constraints in the quadratic programming problem, implying in a lower computational effort to solve the predictive control problem. The third is that the GA method reaches the feasible solutions more easily. In fact, an extreme number of constraints in the DMC algorithm can inhibit its application to real situations where the processing time is crucial, such as those found in the aerospace and robotic areas. In addition, a successful controller for the process industries must maintain the system as close as possible to the con-
strains, without violating then. The advantages exposed here can be extended to the Generalized Predictive Controller tuned by GA.

REFERENCES


Received: November 22, 2007.
Accepted: May 8, 2008.
Recommended by Subject Editor José Guivant.