A derivative-free constrained predictive controller

BRAHIM TLILI 1, FAOUZI BOUANI 2, MEKKI KSOURI 3

1, 2, 3: Laboratoire d’Analyse et de Commande des Systèmes
1: Ecole Nationale d’Ingénieurs de Tunis, BP 37, 1002 Tunis Belvédère, Tunisie
2: Institut National des Sciences Appliquées et de Technologie de Tunis, Tunis, Tunisie
3: Ministère de la Recherche Scientifique et de la Technologie, et de développement des Compétences, Tunis, Tunisie

Abstract: - This paper deals with a derivative-free constrained predictive control of nonlinear systems. The Nonlinear AutoRegressive Moving Average (NARMA) model is used to characterize the behavior of the plant. Consequently, the optimization problem is no longer convex. The paper presents the optimization of the problem by using an improved version of the Nelder-Mead algorithm which does not require the calculation of the derivative of the criterion and which can converge towards the global minimum. Simulation results are presented to illustrate the effectiveness of the proposed method.

Key-Words: - Predictive control, NARMA model, Nelder-Mead algorithm, penalty function, constraints.

1 Introduction

The increase of complexity in industrial systems has limited the use of regulators based on linear models of the systems. Consequently, several control strategies based on nonlinear models were developed.

The Nonlinear AutoRegressive Moving Average (NARMA) model requires a low number of parameters since it involves a limited number among the old measurements and the cross couples (input-input, input-output and output-output). When a nonlinear model is used, the optimization problem is no longer convex.

Generally, nonlinear predictive controllers are computed via the gradient method. This method leads to a local minimum solution and needs the computation of the derivative of the performance criterion. When the control horizon is high, the computation of the derivative of the criterion becomes difficult and time consuming.

In this paper, we propose a derivative-free nonlinear predictive controller based on the NARMA model. The terms intervening in the model as well as the parameters are estimated with genetic algorithms [3]. The control law is provided by minimizing a performance criterion using the Nelder-Mead method combined with the penalty function. The convergence of this method was proved in low dimensions [7]. Comparing to other direct search methods the simplex method is more easily and faster [4]. In Michal D. and al. [10], one iteration of this method is 10 times faster than Genetic algorithms besides it can converge to the global optimum. Since this optimization method needs only the evaluation of the function to be minimized, a non-standard cost function can be also used.

The paper is organized as fellows. In Section 2, the formulation of the problem is presented. Section 3 is reserved to the algorithm used to compute the control law. In section 4 the control and design are presented. The simulation results are presented in Section 5 and conclusions are given in the last Section.

2 Problem Formulation

The predictive control strategy consists in forecasting the future behavior of the system to be controlled. This is done by considering the future values of the output using the system model [6, 15].

The control law is obtained by minimizing a quadratic criterion, on a finite horizon, related to two terms: the first stands for the future errors of prediction representing the difference between the predicted output of the system and the future reference trajectory. The second is related to control increments. Thus a sequence of future control will be generated by the calculator. Among the sequence of control calculated only the first element \( u(k) \) will be applied to the input of the system, all the other elements of the sequence are omitted and the complete procedure of calculation is repeated at each period of sampling.

2.1 The performance criterion

The performance criterion used is given by:

\[
J = \frac{1}{2} \left[ \sum_{i=1}^{N_y} (y_{i}(k+i) - y_m(k+i))^2 + \lambda \sum_{i=1}^{N_y} (\Delta u(k+i-1))^2 \right]
\]

where \( N_y \) is the prediction horizon of the output, \( N_u \) is the control horizon and \( \lambda \) is the weighting coefficient of the control.
The control law is obtained by minimizing the performance criterion $J$ following the control increments vector $\Delta U$:

$$
\Delta U = [\Delta u(k) \ldots \Delta u(k+N_u-1)]^T \quad (2)
$$

The minimization of the criterion $J$ is done by respecting constraints. The constraints on the control signal and in its increments can be written in the following way:

$$
u_{\text{min}} \leq u(k+j) \leq u_{\text{max}}, \quad j = 0, \ldots, N_u - 1 \quad (3)$$

$$
\Delta u_{\text{min}} \leq \Delta u(k+j) \leq \Delta u_{\text{max}}, \quad j = 0, \ldots, N_u - 1 \quad (4)
$$

The number of constraints is equal to $4N_u$. These constraints can be rewritten as follows:

$$
\Omega = \left\{ \Delta U / fJ(\Delta U) \leq 0, \ i = 1, \ldots, 4N_u \right\} \quad (5)
$$

### 2.2 The NARMA process model

We consider the nonlinear plants that can be described by the following relation:

$$
y(k) = g(\varphi(k)) \quad (6)
$$

where $g$ is a presumed unknown nonlinear function and

$$
\varphi(k) = \left[ y(k-1), \ldots, y(k-n), u(k-1), \ldots, u(k-m) \right].
$$

The integers $m$ and $n$ represent, respectively, the input $u(k)$ and the output $y(k)$ regression orders.

The dynamic structure of an operative NARMA model in a deterministic environment is given by the following discrete mathematical equation [13]:

$$
y_m(k) = \sum_{i=0}^{m} \sum_{j=1}^{n} a_{ij} u(k-j) + \sum_{i=0}^{m} \sum_{j=1}^{n} b_{ij} u(k-i) \cdot \varphi(k-j) + \sum_{i=0}^{m} \sum_{j=1}^{n} c_{ij} y(k-j) \cdot \varphi(k-j) + \sum_{i=0}^{m} \sum_{j=1}^{n} d_{ij} y(k-j) u(k-j) + \sum_{i=0}^{m} \sum_{j=1}^{n} e_{ij} u(k-j) \cdot y(k-j) \cdot \varphi(k-j) + \sum_{i=0}^{m} \sum_{j=1}^{n} f_{ij} y(k-j) \cdot u(k-j) \cdot \varphi(k-j) + \sum_{i=0}^{m} \sum_{j=1}^{n} g_{ij} u(k-j) \cdot y(k-j) \cdot u(k-j) \cdot \varphi(k-j)
$$

where $q$ represents the nonlinearity order of the model, and $\overline{\varphi}$ is the average value.

The output of the model can be written as a product of two vectors [13]:

$$
y_m(k) = \theta^T \phi(k) \quad (8)
$$

$\theta$ and $\phi(k)$ represent, respectively, the observations and the parameters vectors.

$$
\phi(k) = \left[ y(k-1) \ u(k-2) \ \ldots \ y^q(k-n) \right]^T \quad (9)
$$

The number of terms intervening in the expression of the model output $y_m(k)$ is given by the following relation:

$$
r = \frac{(n+m+q)!}{q!(n+m)!} - 1 \quad (11)
$$

Consequently, the number of possible models that one can obtain is:

$$
N_m = 2^r - 1 \quad (12)
$$

For instance, if $(m=n=q=2)$, we will have 16383 possible models. Thus, it is necessary to use a procedure allowing the selection of a model among the total possible models representing the behavior of real system as well as possible. In [3], three different methods were suggested to identify the NARMA-type models. In our case the system model has been identified using one of the three mentioned methods based on the binary genetic algorithm exploiting offline a file of measurements $\{y(k), u(k), k=1, \ldots, N\}$.

### 2.3 The control design

The nonlinear predictive controller is obtained by minimizing the performance criterion (1) in tacking into account the set of constraints (5). Fig. 1 presents the general diagram of the predictor where the NARMA model is used to predict the output of the plant over the prediction horizon $N_p$. Fig. 2 presents the structure of the predictive controller. Usually, the gradient method is used to compute the control law. This method needs the derivative of the criterion and leads to a local minimum solution. Moreover, the computation time increases as the control horizon $N_p$ increases.

In this work, we propose the Nelder-Mead algorithm to compute the control law. This algorithm, still called simplex algorithm, makes it possible to determine the minimum of a function without calculation of the derivative. Moreover, it treats the nonlinear, non convex functions having several minima.
3 Comparisons of the NM Algorithm performances

3.1 The Nelder-Mead method
The Nelder-Mead (NM) algorithm is based on the evaluation of the function to be minimized at the tops of the simplex [5, 11]. By considering a whole of points of the space of research \((x_1, x_2, ..., x_{n+1})\), we calculate the values of the function \((f(x_1), ..., f(x_{n+1}))\). In each iteration of the Nelder-Mead algorithm two transformations are possible:

- Only one point of the current simplex will be modified. The new test point accepted by the algorithm replaces the worst point of the current simplex \((x_{n+1})\) and prepares the new simplex for the following iteration.
- The total \(n\) new points will be calculated and replace the \(n\) current points \(x_i\) \((i = 2, ..., n+1)\) in order to form a new simplex for the following iteration. It is about the shrink step.

The algorithm stops when the difference \((f(x_{n+1}) - f(x_i))\), which indicates the convergence of the algorithm is less than a certain precision \(\varepsilon_0\).

The Nelder-Mead algorithm is summarized by the following stages [2, 5, 11]:

Stage 1: Take the tops \((x_1, x_2, ..., x_{n+1})\) checking:
\[ f(x_1) \leq f(x_2) \leq ... \leq f(x_{n+1}). \]

Stage 2: As long as the difference between \(f(x_{n+1})\) and \(f(x_i)\) is bigger than a certain precision \(\varepsilon_0\):

2.1 Calculate \(\bar{x}\) and \(f_r = f(x(\rho))\) with \(\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i\), is the centroid of the \(n\) best points and \(x(\rho) = \bar{x} + \rho \cdot (\bar{x} - x_{n+1})\).

2.2 Reflect:
if \(f(x_i) \leq f(x_i) < f(x_{n+1})\) then \(x_{n+1} = x(\rho)\), go to step 2.7.

2.3 Expand:
if \(f_r < f(x_i)\) then calculate \(f_e = f(x(\gamma))\) with \(x(\gamma) = \bar{x} + \gamma \cdot (x_r - \bar{x})\).

2.4 Outside contract:
if \(f(x_i) \leq f_e < f(x_{n+1})\) then calculate \(f_o = f(x(y))\) with \(x(y) = \bar{x} + y \cdot (\bar{x} - x_{n+1})\).

if \(f_e \leq f_r\) then \(x_{n+1} = x(y)\), then go to 2.7. Otherwise go to step 2.6.

2.5 Inside contract:

if \(f_i \geq f(x_{n+1})\) then calculate \(f_i = f(x(y))\) with \(x(y) = \bar{x} - \gamma \cdot (\bar{x} - x_{n+1})\).

if \(f_i < f(x_{n+1})\) then \(x_{n+1} = x(y)\) and go to 2.7.

Otherwise, go to step 2.6.

2.6 Shrink:
for \(i = 2, ..., n+1\), calculate \(x_i = x_i + \sigma(x_i - x)\), then calculate \(f(x_i)\).

2.7 Give the tops \((x_1, x_2, ..., x_{n+1})\) checking
\(f(x_1) \leq f(x_2) \leq ... \leq f(x_{n+1})\) and return to the beginning of stage 2.

The standard sequence associated to the coefficients \(\{\rho, \gamma, \sigma\}\) is \(\{1, 2, 0.5, 0.5\}\) [5, 11].

3.2 The Nelder-Mead improvements
Compared to other direct research methods, the NM method is relatively fast. The time computing can be improved following the simplex initialization. Once the number of simplex points is set, their values affect the speed of the algorithm. In [17] one exploit Genetic and NM Algorithms for global optimization, the starting population of the Genetic Algorithm should be avoiding the risk of having a large number of individuals in the same region. On the other hand, because the NM is not a global optimization routine, using multiple starting points is beneficial and helps to ascertain that a highly optimal value will be found [18]. Two other comments on the properties of the NM algorithm are added by M. A. Luersen and R. Le Riche [9]. Firstly, the NM algorithm may fail to converge to a local optimum, which happens in particular when the simplex collapses into a subspace. Secondly, the method may escape a region that would be a basin of attraction for a point wise descent search if the simplex is large enough. Ultimately, if the size of the simplex is small, the algorithm becomes local.

In this section, we present a comparative study of different versions of the NM algorithm using a set of benchmark functions. The list of the benchmark test functions is given in Appendix A. These functions have several local minima and their global minima are known. The versions of the NM algorithm considered are the following:

- Version 1 (NM-V1): in this algorithm, only one initialisation is considered. The values of the simplex initialization are lying on the interval of the research domain.
- Version 2 (NM-V2): in this case, a multiple starting simplex initialization is used. Every simplex is lying on the interval of the research domain.
- Version 3 (NM-V3): in the third version of the Nelder-Mead algorithm, the space research interval \( S \) is divided as a several equal intervals \( s_i, i=1,..,n \), such as \( S = s_1 + s_2 + ... + s_n \). Then, each simplex is initialized in an interval \( s_i \) and the partial solution is kept. When all sub intervals \( s_i \) are gone through, we recuperate the best solution among the partial solutions.

Table 1 shows the results obtained with the three versions of the Nelder Mead algorithm. To evaluate the efficiency of these methods, we retained the following criteria summarizing results from 100 minimizations per function: the rate of successful minimizations (%), the average of time computing optimization and the average of the square error between the best successful point and the known global optimum [16]. The optimisation is considered successful if the square error between the obtained solution and the global minimum is less than \( 10^{-4} \).

<table>
<thead>
<tr>
<th>Test function</th>
<th>Method</th>
<th>Rate of successful minimization (%)</th>
<th>Average of time computing optimization</th>
<th>Average of the square error</th>
</tr>
</thead>
<tbody>
<tr>
<td>( F_1 )</td>
<td>NM-V1</td>
<td>25</td>
<td>0.0547</td>
<td>0.0173</td>
</tr>
<tr>
<td></td>
<td>NM-V2</td>
<td>82</td>
<td>2.7533</td>
<td>6.252e-4</td>
</tr>
<tr>
<td></td>
<td>NM-V3</td>
<td>100</td>
<td>4.7341</td>
<td>1.479e-5</td>
</tr>
<tr>
<td>( F_2 )</td>
<td>NM-V1</td>
<td>8</td>
<td>0.0421</td>
<td>0.161</td>
</tr>
<tr>
<td></td>
<td>NM-V2</td>
<td>94</td>
<td>3.0414</td>
<td>1.596e-5</td>
</tr>
<tr>
<td></td>
<td>NM-V3</td>
<td>100</td>
<td>3.8663</td>
<td>5.041e-7</td>
</tr>
<tr>
<td>( F_3 )</td>
<td>NM-V1</td>
<td>1</td>
<td>0.0620</td>
<td>0.148</td>
</tr>
<tr>
<td></td>
<td>NM-V2</td>
<td>27</td>
<td>0.0681</td>
<td>0.007</td>
</tr>
<tr>
<td></td>
<td>NM-V3</td>
<td>87</td>
<td>0.0742</td>
<td>2.68e-5</td>
</tr>
<tr>
<td>( F_4 )</td>
<td>NM-V1</td>
<td>100</td>
<td>0.0023</td>
<td>4.388e-4</td>
</tr>
<tr>
<td></td>
<td>NM-V2</td>
<td>100</td>
<td>0.0253</td>
<td>4.388e-5</td>
</tr>
<tr>
<td></td>
<td>NM-V3</td>
<td>100</td>
<td>0.0489</td>
<td>1.799e-5</td>
</tr>
</tbody>
</table>

Table 1: Results of NM optimization methods

From this table, we note that the third version of the NM algorithm (NM-V3) presents the best rate of successful minimization and leads to the lowest average of the square error.

4 Control and design

The Nelder-Mead method is designed for unconstrained optimization nonlinear problems. To deal with constraints, we can use an exact penalty function, because methods using penalty functions transform a constrained problem into an unconstrained one [12].

Another method consists in tackling the initial vector \( (x_1,..,x_{n+1}) \), such that the constraints are satisfied. This approach is known as “Constraints First Objective Next, CFON” [1].

4.1 Penalty function algorithm

The constraints are placed into the objective function via a penalty parameter in a way that it penalizes each violation of the constraints. In general, a suitable penalty function must have a positive penalty for infeasible points and no penalty for feasible points [14]. Thus, the penalty function is usually of the form:

\[
\alpha(x) = \sum_{i=1}^{n} \max \{0, f_i(x)\}^p
\]

where \( p \) is a non nil positive integer.

A summary of penalty function method algorithm is given as follows:

**Initialization Step**

Let \( \epsilon > 0 \) be a termination scalar. Choose an initial point \( x_0 \), a penalty parameter \( \nu_1 > 0 \), and a scalar \( \beta > 0 \). Let \( k=1 \), and go to the main step.

**Main Step**

1. Starting with \( x_0 \), solve the following problem: Minimise \( f(x) + \nu_k \alpha(x) \) subject to \( x \in X \).

Let \( x_{k+1} \) be an optimal solution and go to step 2.

2. If \( \nu_k \alpha(x) < \epsilon \), stop; otherwise, let \( \nu_{k+1} = \beta \nu_k \), replace \( k \) by \( k+1 \), and go to step 1.

4.2 The CFON approach

In [1] one comparable approach is used to improve constraint handling for multi-objective genetic algorithms. This approach will reduce the time computing because all points of the start simplex respect all constraints. Although, the penalty function is used with this approach in order to avoid the apparition of non feasible solutions in the inner loops of the Nelder-Mead algorithm.

In order to find the solution of the nonlinear predictive control optimisation problem, we propose the combination of version 3 of the NM algorithm with the penalty function and the CFON approach. At each sample time, we divided the interval of research domain, \([\Delta u_{\min}, \Delta u_{\max}]\), in several intervals. Then, we used the NM algorithm with the penalty function to find the optimal \( \Delta U \) which optimize the performance criterion.

5 Simulation Results

We consider the nonlinear system described by the following equation [8]:

\[
y(k) = \frac{y(k-1)}{1+y(k-1)^2} + u^*(k-1).
\]

For \( (m=n=q=2) \), the model identified by exploiting the binary genetic algorithm is as follows [3]:

\[
y_u(k) = 0.262 - 0.0212 u(k-1) - 0.0065 u(k-2) + 0.359 y(k-1)
+ 0.9991 u(k-1)^2 - 0.2316 u(k-2)^2 + 0.0016 u(k-1) y(k-1) - 0.088 y(k-1)^2
\]

\[(15)\]
In simulations we have added to the system output the disturbance signal \(d(k)\) as follows:

\[
y(k) = \frac{y(k-1)}{1+y^2(k-1)} + u^2(k-1) + d(k). \tag{16}
\]

where:

\[
d(k) = \begin{cases} 
0.1 & \text{for } k \in [120, \ldots, 150] \\
0 & \text{otherwise}
\end{cases} \tag{17}
\]

The parameters of the predictive controller considered in simulation are \(N_u=1\), \(N_y=4\) and \(\lambda = 0.2\). In first simulation, the constraints on the input and constraints on the gradient of the input are taken as follows:

\[
0 \leq u(k) \leq 0.7 \tag{18}
\]

\[
-0.1 \leq \Delta u(k) \leq 0.1 \tag{19}
\]

In the second simulations, only the constraints on the gradient of the control are modified and fixed as follows:

\[
-0.04 \leq \Delta u(k) \leq 0.04 \tag{20}
\]

The evolutions of the set point, the output, the control and the increment of the control are shown in Fig. 4 and Fig. 5.

![Figure 4: The output, the control and the increment of the control for \((\Delta u = 0.1)\)](image)

![Figure 5: Output, control and increment of the control, for \((\Delta u = 0.04)\)](image)

It’s clear from these figures that there’s no violation of input and gradient input constraints even if the constraints in the gradient input are very low. One can also observe that the control signal provided by the controller is capable to reject the disturbance signal \(d(k)\) in a reasonable time.

In these simulations, the space research interval \(S\) is given by:

\[
S = [u(k-1) + \Delta u_{\text{min}}, u(k-1) + \Delta u_{\text{max}}] \tag{21}
\]

This interval is divided in 10 equal sub intervals. The initial simplex is chosen in the sub intervals so the constraints on the input and the gradient of the input are satisfied. The resolution of the optimisation problem is found by using the NM algorithm with the penalty function in each sub interval. When all sub intervals are gone through, we recuperate the best solution among the partial solutions.

In the last simulation, we have considered the same nonlinear predictive controller with constraints on the input and constraints on the gradient of the input are given by relations (18) and (19). We have tested two methods to compute the control law. The first method uses the NM algorithm and the penalty function. The initial simplex is chosen in the following interval:

\[
S = [u_{\text{min}}, u_{\text{max}}] \tag{22}
\]

The second method uses the version three of the NM algorithm (NM-V3) with the penalty function and the CFON approach. In this case, the initial simplex is fixed in the sub intervals. Table 2 gives the time needed by each method to compute the control law. In this case, 20 random simulations are carried out using a Pentium 4-2.4 GHz.

<table>
<thead>
<tr>
<th>Time delay</th>
<th>NM-V3 + Penalty function</th>
<th>NM-V3 + penalty function + CFON approach</th>
</tr>
</thead>
<tbody>
<tr>
<td>(t_{\text{min}})</td>
<td>4.792</td>
<td>1.641</td>
</tr>
<tr>
<td>(t_{\text{max}})</td>
<td>21.057</td>
<td>1.753</td>
</tr>
<tr>
<td>(t_{\text{mean}})</td>
<td>10.324</td>
<td>1.671</td>
</tr>
</tbody>
</table>

Table 2: Time delay depending on simplex initialization.

In table 2, \(t_{\text{min}}, t_{\text{max}}\) and \(t_{\text{mean}}\) indicate, respectively, the minimum time, the maximal time and the average time (in second) put in the 20 random simulations. We note, from table 2 that with the CFON approach the mean of the computing time is less than the time needed by the penalty approach.

6 Conclusion
In this paper, a derivative-free constrained predictive controller of nonlinear systems has been presented. The
NARMA model is used to characterize the behavior of the system. The control law is obtained by minimizing a non-convex criterion. The optimization problem is solved by a method using the Nelder-Mead algorithm combined with the penalty function, which does not require the calculation of the derived of the criterion and which can converge towards the global minimum. The computing time needed each simple time is considerably reduced by choosing a simplex which respects the constraints at the start of the algorithm. As a perspective to this work, we suggest using other derivative-free optimization methods and compare their performances through a practical application.

Appendix A. List of test functions

F₁ (1 variable):
\[ f₁(x) = \sin(x) + \sin(2x/3) \]
Search domain: -3.1 < x < 20.4
1 global minimum: \( x^* = 17.0393 \); \( f₁(x^*) = -1.90596 \).

F₂ (1 variable):
\[ f₂(x) = \sin(x) + \sin(10x/3) + \ln(x) - 0.84x \]
Search domain: 2.7 < x < 7.5
Several local minima: \( f₂(x^*) = 5.19 \); \( f₂(x^*) = -3.8717 \).

F₃ (2 variables):
\[ f₃(x₁,x₂) = x₁² + x₂² - \cos(18x₁) - \cos(18x₂) \]
Search domain: (-1,-1) < (x₁,x₂) < (1,1)
About 50 local minima: 1 global minimum: \( (x₁^*,x₂^*) = (0,0) \); \( f₃(x₁^*,x₂^*) = -2 \).

F₄ (2 variables):
\[ f₄(x₁,x₂) = (x₁² + 1)² + (x₂² + 1)² - 2(x₁ + x₂ + 1)² \]
Search domain: (-3,-3) < (x₁,x₂) < (3,3)
1 global minimum: \( (x₁^*,x₂^*) = (1.3247,1.3247) \); \( f₄(x₁^*,x₂^*) = -11.45851 \).

References:


