PRACTICAL NON-LINEAR MODEL PREDICTIVE CONTROL PNMPC: ALGORITHM IMPLEMENTATIONS

Bryant Picon Yang*, Agustinho Plucenio†

*Dpto. de Automação e Sistemas
Universidade Federal de Santa Catarina
88040-900
Florianópolis, SC, Brasil

†Firenze-Consultoria em Engenharia e Informatica
Av. Lédio João Martins, 201-203 - 88102-000 KOBRASOL São José, SC, Brasil

Emails: bryant.bruce@posgrad.ufsc.br, brplucenio@firenzeconsult.com,

Abstract— Most dynamic processes exhibit nonlinear relationships between controlled and manipulated variables. Depending on the application, the MPC algorithm is often chosen between a non-linear or a linear type. This paper presents different implementations of the PNMPC-Practical Non-linear Model Predictive Control algorithm. It is shown that the PNMPC algorithm may be written to handle the non-linearity at different levels. For the sake of simplicity, the algorithms are implemented to control the level of a conical tank. The main purpose of the controller is to manipulate the position of the outlet valve in order to follow the reference. The performance and the computational cost of six different implementations of PNMPC controller are shown. The results of the controllers are compared and analyzed by the system response verification, the algorithm runtime and driver performance.

Keywords— Nonlinear predictive control, Computational cost, Conical tank, Level control, SISO system

Resumo— A maioria dos processos dinâmicos exibem relações não lineares entre variáveis controladas e manipuladas. Dependendo da aplicação, o algoritmo MPC é muitas vezes escolhido entre um tipo linear ou não linear. Este trabalho apresenta varias maneiras de implementar o algoritmo PNMPC. Mostra-se que o algoritmo PNMPC pode ser escrito para lidar com a não-linearidade em diferentes níveis. Por uma questão de simplicidade, os algoritmos são implementados para controlar o nível de um tanque cônico. A principal finalidade do controlador é manipular a abertura da válvula de saída, a fim de seguir a referência. O desempenho e o custo computacional de seis implementações diferentes de controlador PNMPC são exibidos. Os resultados dos controladores são comparados e analisados pelo sistema de verificação de resposta, o tempo de execução do algoritmo e o desempenho do atuador.

Palavras-chave— Controle preditivo não linear, Custo computacional, Tanque cônico, Controle de nível, Sistema SISO

1 Introduction

The Model Predictive Control (MPC) is one of the most advanced modern control techniques, being probably the one that has been the most successful in industrial applications especially in the oil industry (García, 1989).

The history of advanced control begins in the early 1960's with the work of Kalman as well as co-author Ricatti and others, (Qina and Badgwell, 2003). But it was only after a set of papers presented in the late 1970's that MPC controllers became interesting for industrial applications

In 1978, Richalet described the applications of "Control Models Predictive Heuristic" in 1979 Shell Engineers Cutler and Ramaker as well as Pratt and Gillette outlined "Dynamic Control Matrix" (DMC) and reported applications for a catalytic tab fluid. In both algorithms, an explicit dynamic model of the plant has been used to predict the effect of future control actions on the output, hence the name "Model Predictive Control".

The main causes of this success is due to the ability of the MPC to control a wide range of processes, from those having a relatively simple dynamic to others more complex, for example: Systems with delay, multivariables, unstable, as well non-linear dynamic among others. The MPC also allows to obtain solutions, notwithstanding the existence of constraints in the variables (Normey and Camacho, 2007).

The majority of the real systems have non-linear dynamics, however, when the process operational range is small, the dynamics can be satisfactorily approximated by linear models. This approach has motivated many MPC techniques using linear models such as: Dinamix Matrix Control-DMC, (C.R. Cutler and B.L. Ramaker, 1988); Model Algorithm Control-MAC,(Richalet, Rault, Testud and Papon, 1976); Generalized Predictive Control-GPC, (Clarke, Mothadi and Tuffs, 1987)) (Plucenio et al., 2007).

The main advantage of linear MPC is the low cost to obtain linear models in the development of the project compared to non-linear approaches. Other advantages are lesser difficulties to solve the associated optimization problems. On the other hand, when the processes have a very strong non-linear dynamic or when the operation range is wide, then it might be necessary to consider

the non-linear model in the control algorithm, to maintain the desired performance for the closed-loop system.

When choosing a predictive control algorithm, the initial decision is to choose between two algorithm systems, either the linear or the non-linear systems. In many cases we have good non-linear process models. Unfortunately it is common practice to get a linear representation of the model in a single point of operation and use this representation to obtain the dynamic matrix coefficients and the free response. The original PNMPC technique uses a real-time linearization of the input-output variables. This linearization is performed in a two step process and it obtains almost a second-order approximation. Depending on the non-linearity process to be controlled, there are several ways to implement the PNMPC algorithm. The algorithm choice will depend on the available processing time and quality of response to be obtained. The PNMPC can be implemented as a linear controller. Due to the way it treats the prediction error, the linear version of PNMPC will be similar to a GPC.

This paper is organized as follows: section 2 presents the description of the mathematical model of the conical tank. The theoretical principles of non-linear control strategy are presented in section 3. The description of the six alternative versions of the implementation of the PNMPC are shown in section 4. And section 5 presents the results of implementations described in section 4.

2 Mathematical Model Description of the Conical Tank

To demonstrate the application of the Practical MPC algorithm for non-linear systems (PNMPC), a process with a non-linear dynamics was considered. Furthermore and in order to simplify the demonstration, the level control in a conical tank (SISO system) was also used. The process is shown in Figure 1, where the manipulated variable is the output valve. The input flow will act as a disturbance in the system. The mass balance

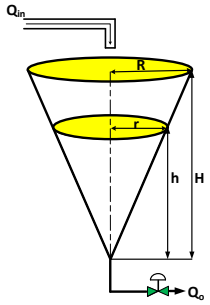


Figure 1: conical tank

inside the tank is given by the equation:

$$Q_{in} - Q_o = A\frac{dh}{dt} \tag{1}$$

where:

Q_{in}	input flow;
Q_o	output flow;
R	maximum radius of the conical tank;
r	radius of the conical tank at steady state;
H	maximum height of the conical tank;
h	height of the conical tank at steady state.

Whereas the cross-section area of the conical tank is given by:

$$A = \pi\left[r - (R-r)\left(\frac{h}{H}\right)\right]^2 \tag{2}$$

and the output flow is represented by the equation:

$$Q_o = C_v U_c \sqrt{h} \tag{3}$$

where U_c is the control action of the outlet valve and C_v is a constant for the valve.

Substituting equations (2) and (3) in (1),we have the following model of the conical tank:

$$\frac{dh}{dt} = \frac{Q_{in} - C_v U_c \sqrt{h}}{\pi\left[r - (R-r)\left(\frac{h}{H}\right)\right]^2} \tag{4}$$

3 PNMPC control

The PNMPC technique was developed by (Plucenio, 2010) and is based on algorithms that use the vector representation of predictions over the horizon p, $\tilde{\mathbf{Y}}$, as a vector function with m changes in the control action $\Delta\mathbf{u}$, according to the equation (5). The PNMPC is a technique that offers an alternative to solve the problem of linear optimization using conventional optimizers as QP quadratic programming or linear programming solution.

$$\tilde{\mathbf{Y}} = \mathbf{F} + \mathbf{G}\Delta\mathbf{u} \tag{5}$$

The PNMPC differs from other MPC techniques mainly because it uses linearized models of the system trajectory. It is assumed that the predictions $\tilde{\mathbf{Y}}$ depend only on past entries $\overleftarrow{u}$, past outputs $\overleftarrow{y}$ and the future control increments $\Delta\mathbf{u}$.

$$\tilde{\mathbf{Y}} = f\left(\overleftarrow{y}, \overleftarrow{u}, \Delta\mathbf{u}\right) \tag{6}$$

The vector of predictions is rewritten as:

$$\tilde{\mathbf{Y}} = \mathbf{F} + \mathbf{G}_{PNMPC}\Delta\mathbf{u} \tag{7}$$

where

$$\mathbf{F} = f\left(\overleftarrow{y}, \overleftarrow{u}\right) \tag{8}$$

$$\mathbf{G}_{PNMPC} = \frac{\partial\tilde{\mathbf{Y}}}{\partial\Delta\mathbf{u}} \tag{9}$$

The matrix $\mathbf{G}_{PNMPC}$ is the Jacobian of $\tilde{\mathbf{Y}}$ relative to the control increments and is obtained numerically as the vector of free response $\mathbf{F}$. A prediction algorithm computes the vector $\tilde{\mathbf{Y}}$ with

the p predictions when it provides the values of the past inputs and outputs, and the vector with the m future input increments $\Delta\mathbf{u}$.

If we call the set of past inputs $\overleftarrow{u}$ and the current and past output $\overleftarrow{y}$, predictions can be rewritten in the following expression.

$$\begin{aligned}
&\tilde{y}(k+1) = f(\overleftarrow{y}, \overleftarrow{u}, \Delta u(k)) \\
&\tilde{y}(k+2) = f(\overleftarrow{y}, \overleftarrow{u}, \Delta u(k), \Delta u(k+1)) \\
&\vdots \\
&\tilde{y}(k+p) = f(\overleftarrow{y}, \overleftarrow{u}, \Delta u(k), \ldots, \Delta u(k+m-1))
\end{aligned} \tag{10}$$

The following compact representation is obtained:

$$\tilde{\mathbf{Y}} = \mathbf{F} + \mathbf{G}_{PNMPC}\Delta\mathbf{u} \tag{11}$$

Whereas that $\mathbf{F}$ is the vector of predictions that would be obtained for $\Delta\mathbf{u} = 0$ and $\mathbf{G}_{PNMPC}$ is the Jacobian of the predicted outputs with respect to the vector of increments of control signal $\Delta\mathbf{u}$. $\mathbf{G}_{PNMPC}$ is a generalized dynamic matrix that can be used for linear systems and non-linear systems, provided they are continuous and differentiable. The expression (11) for a SISO system can be re-written as:

$$\begin{bmatrix} \tilde{y}(k+1) \\ \tilde{y}(k+2) \\ \vdots \\ \tilde{y}(k+p) \end{bmatrix} = \begin{bmatrix} f_1(\overleftarrow{y}, \overleftarrow{u}) \\ f_2(\overleftarrow{y}, \overleftarrow{u}) \\ \vdots \\ f_p(\overleftarrow{y}, \overleftarrow{u}) \end{bmatrix} + \tag{12}$$

$$\begin{bmatrix} \frac{\partial\tilde{y}(k+1)}{\partial\Delta u(k)} & 0 & \cdots & 0 \\ \frac{\partial\tilde{y}(k+2)}{\partial\Delta u(k)} & \frac{\partial\tilde{y}(k+2)}{\partial\Delta u(k+1)} & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial\tilde{y}(k+p)}{\partial\Delta u(k)} & \frac{\partial\tilde{y}(k+p)}{\partial\Delta u(k+1)} & \cdots & \frac{\partial\tilde{y}(k+p)}{\partial\Delta u(k+m-1)} \end{bmatrix} \begin{bmatrix} \Delta u(k) \\ \Delta u(k+1) \\ \vdots \\ \Delta u(k+m-1) \end{bmatrix}$$

The lower triangular form of the Jacobian matrix is due to the causality of the system and absence of direct coupling between input and output. This means $\frac{\partial\tilde{y}(k+j)}{\partial\Delta u(k+i)} = 0 \quad \text{for} \quad i \geq j$. The partial derivative $\frac{\partial\tilde{y}_{k+j}}{\partial\Delta u(k)}$ it is defined as:

$$\frac{\partial\tilde{y}_{k+j}}{\partial\Delta u(k)} = \lim_{\Delta u_k \to 0} \frac{\tilde{y}_{k+j}(u_{k-1} + \Delta u_k) - \tilde{y}_{k+j}(u_{k-1})}{\Delta u_k} \tag{13}$$

For a SISO system, for example, on the first instant of sampling, the following procedure for obtaining the numerical value of $\mathbf{F}$ and $\mathbf{G}_{PNMPC}$ is performed:

1. Get the vector $\tilde{\mathbf{Y}}_p^0$ (dimension $p \times 1$) executing the prediction model with the past inputs and outputs and $\Delta\mathbf{u} = [0\ 0 \ldots 0]^T$. $\mathbf{F} = \tilde{\mathbf{Y}}_p^0$.
2. Computes the first column of the matrix $\mathbf{G}_{PNMPC}$. Obtaining the vector $\tilde{\mathbf{Y}}_p^1$ (dimension $p \times 1$) executing the model with the past inputs and outputs and considering $\Delta\mathbf{u} = [\epsilon\ 0 \ldots 0]^T$, where ϵ is a very small value, $\frac{u_{k-1}}{1000}$, for example. $\mathbf{G}_{PNMPC}(:,1) = \frac{\tilde{\mathbf{Y}}_p^1 - \tilde{\mathbf{Y}}_p^0}{\varepsilon}$.
3. Computes the second column of the matrix $\mathbf{G}_{PNMPC}$. Obtain the vector $\tilde{\mathbf{Y}}_p^2$ (dimension $p \times 1$) executing the model with the past inputs and outputs and considering $\Delta\mathbf{u} = [0\ \epsilon \ldots 0]^T$. $\mathbf{G}_{PNMPC}(:,2) = \frac{\tilde{\mathbf{Y}}_p^2 - \tilde{\mathbf{Y}}_p^0}{\varepsilon}$.
4. Continues with the calculation of the others columns of the matrix $\mathbf{G}_{PNMPC}$ to the last column, where the vector $\tilde{\mathbf{Y}}_p^m$ is obtained executing the model with the past inputs and outputs $\Delta\mathbf{u} = [0\ 0 \ldots \epsilon]^T$.$\mathbf{G}_{PNMPC}(:,m) = \frac{\tilde{\mathbf{Y}}_\mathbf{p}^\mathbf{m} - \tilde{\mathbf{Y}}_\mathbf{p}^\mathbf{0}}{\varepsilon}$.

For the application of PNMPC technique in multivariable systems it is necessary to concatenate the prediction vectors for each variable $\tilde{\mathbf{Y}}$ and the control increments $\Delta\mathbf{u}$. Thus, a system with ni entries and no outputs have the following vectors:

$$\begin{aligned}
\tilde{\mathbf{Y}} &= [\tilde{\mathbf{Y}}_1 \tilde{\mathbf{Y}}_2 \ldots \tilde{\mathbf{Y}}_{pno}]^T \\
\Delta\mathbf{u} &= [\Delta\mathbf{u}_1 \Delta\mathbf{u}_2 \ldots \Delta\mathbf{u}_{mni}]^T
\end{aligned} \tag{14}$$

To obtain $\mathbf{F}$ and $\mathbf{G}_{PNMPC}$ the procedure is then as above. The difference is that now the matrix $\mathbf{G}_{PNMPC}$ have $ni \times no$ computed blocks as in the SISO case for each pair of input and output. Obviously, as in linear MPC, weak relationships output-input may be disregarded to avoid noise propagation. The control action is obtained by minimizing the cost function J. Using equation (7) as a way of expressing the predictions, the quadratic objective function from section 2.2.2 can be re-written as follows:

$$\begin{aligned}
&\min_{\Delta\mathbf{u}} J \\
&\text{subject to} \\
&J = (\tilde{\mathbf{Y}} - \mathbf{W})^T\mathbf{R}(\tilde{\mathbf{Y}} - \mathbf{W}) + \Delta\mathbf{u}^T\mathbf{Q}\Delta\mathbf{u}, \\
&\tilde{\mathbf{Y}} = \mathbf{F} + \mathbf{G}_{PNMPC}
\end{aligned} \tag{15}$$

where $\mathbf{W}$ represents the vector of future reference, $\mathbf{R}$ the error weighting matrix and $\mathbf{Q}$ the control weighting matrix. The minimization of the cost function for the case without constraints can be obtained by equating to zero the gradient of the cost function and in the case with constraints the solutions is obtained by solving a QP.

Thus, in terms of optimization, the PNMPC has a complexity similar to the DMC or the GPC algorithm. The predictive control techniques applied to linear systems use a mechanism that consists in correct the predictions with the error between the measured value of the controlled variable and its predicted value at time k. This technique applied to linear systems guaranteed zero error in steady stated, because the increase in the control effort required for error correction at the instant k will be the same in future instants due to the linear relationship between the input and output, so the gain is constant. The application of this procedure to the PNMPC technique applied to non-linear systems does not guarantee null error in steady state, because the gain varies with the operating point of the system. The procedure to correct the predictions presented in (Plucenio, 2010), consists in adding to each prediction the integral of the filtered prediction error. Figure 2 shows how the correction factor for each controlled variable can be obtained. A correction factor

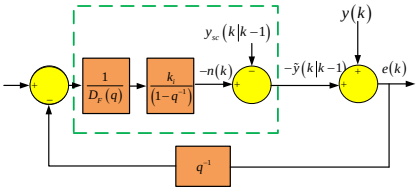


Figure 2: Block diagram of the correction factor

is added to the prediction so that the error between the measured and the corrected value of prediction to

zero. The correction factor is obtained filtering and integrating the error as shown in Figure 2. Considering the controlled variable $y(t)$, if the level error has a linear approximation, the transfer function that relates the error and the difference between the measured and the predicted values of the variable without correction can be obtained as shown in (16). It must be noted that the error $e(z)$ is defined as the difference between the measured value and the corrected predicted value, whereas $(y(k) - \tilde{y}_{sc}(k))$ corresponds to the error between the measured variable and the predicted value without correction.

$$\frac{e(z)}{(y(k) - \tilde{y}_{sc}(k))} = \frac{z^2 - (1 + f_d)z + f_d}{z^2 - (1 + f_d - k_i)z + f_d} \quad (16)$$

Assuming the closed loop system is stable, we can apply the Final Value Theorem to verify that the presence of $(1-z^{-1})$ in the numerator of the transfer function (16) guarantees null error in steady state for an input of type step. The values of k_i and f_d are chosen by comparing the denominator of the transfer function (16) with the desired polynomial $p_d(z) = z^2 - 2az + a^2$. The value of a has an impact on the speed at which the error responds to disturbances. After the value of a is determined, f_d and k_i can be obtained through equations (17) and (18).

$$f_d = a^2 \quad (17)$$

$$k_i = 1 + a^2 - 2a \quad (18)$$

4 Implementation of PNMPC: Alternatives

The PNMPC presented in the previous section can be implemented in different ways. The goal of this paper is to study the performance and the computational cost of the PNMPC controller, by using the following: six different versions of the PNMPC implementation, as well as simplifications of the complete original technique to reduce the processing time. The way of calculating the predictions can be modified to convert the PNMPC into a linear MPC controller. Six versions of the PNMPC will be applied to the conical tank. The goal is to control the level of the tank through outlet valve manipulation.

4.1 1st version: PNMPC complete

Following the PNMPC theory from (Plucenio, 2010), the first PNMPC proposed in this paper considers the non-linear free response and calculation of the matrix $\mathbf{G}$ in two steps. The first procedure of the algorithm is to calculate at a given iteration the variation of initial control ($\Delta\mathbf{u}_{ini}$), which is calculated according to the existing matrix $\mathbf{G}$, that is, the matrix $\mathbf{G}_{previous}$. Obtaining the initial control variation, the future control actions are calculated according to the equation:

$$\mathbf{U}_{m\times 1} = \mathbf{H}_{m\times 1} u\,(k-1) + \mathbf{T}^I_{m\times m}\Delta\mathbf{u}_{inim\times 1} \quad (19)$$

where:

m	control horizon;
$\mathbf{H}$	vector of ones;
$u(k-1)$	previous control action;
$\mathbf{T}^I$	integrating Toeplitz matrix;
$\Delta\mathbf{u}_{ini}$	initial control variation.

With the control action calculated, an initial predicted output $\tilde{\mathbf{Y}}^0_p$ is obtained. A control increment ε is added to the control vector $\mathbf{U}$ and a new output prediction ($\tilde{\mathbf{Y}}$) is obtained. This value ε will scroll through the $\Delta\mathbf{u}$ vector in order to calculate the matrix $\mathbf{G}_{PNMPC}$ as described in (Plucenio, 2010) and shown in equation (20).

$$\mathbf{G}_{PNMPC}\,(:, m) = \frac{\tilde{\mathbf{Y}}^{\mathbf{m}}_{\mathbf{p}} - \tilde{\mathbf{Y}}^{\mathbf{0}}_{\mathbf{p}}}{\varepsilon} \quad (20)$$

In fact, the calculation of the derivative of this new control signal $\mathbf{U}$, serves to compensate the non-linearity of the system. Having the matrix $\mathbf{G}(k)$ at the current time and the matrix $\mathbf{G}(k-1)$ calculated in the previous instant $\mathbf{G}_{previous}$, it is possible calculate in two steps the intermediate matrix $\overline{\mathbf{G}}_{PNMPC}$ by the following equation:

$$\overline{\mathbf{G}}_{PNMPC} = 0.5\mathbf{G}(k-1) + 0.5\mathbf{G}(k) \quad (21)$$

The algorithm implemented for this version is detailed in the flow diagram in Figure 3:

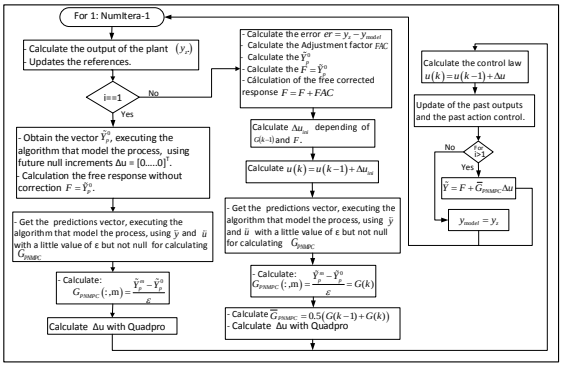


Figure 3: PNMPC complete algorithm flow diagram

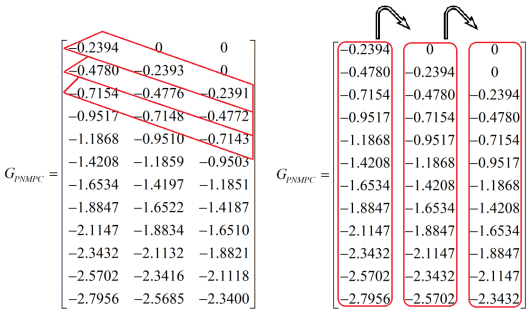
4.2 2nd version: PNMPC without calculation $\Delta\mathbf{u}_{ini}$

For processes with a strong non-linear characteristic it is recommended to choose the complete PNMPC technique (first version), but if the process presents a smooth non-linearity it may be possible to reduce the computational cost by eliminating the calculation of $\Delta\mathbf{u}_{ini}$. In this version it is proposed the calculation of the $\mathbf{G}_{PNMPC}$ matrix in one step. Therefore, it is not necessary to calculate $\Delta\mathbf{u}_{ini}$. In order to calculate the control action needed to obtain $\tilde{\mathbf{Y}}^0_p$ the value of the control variation of the previous iteration is used. This $\tilde{\mathbf{Y}}^0_p$ is then calculating the matrix $\mathbf{G}_{PNMPC}$, as shown in section three. Once the calculation of $\Delta\mathbf{u}_{ini}$ is eliminated, the precision with which the PNMPC calculates the $\mathbf{G}_{PNMPC}$ decreases, this can be explained as the $\Delta\mathbf{u}_{ini}$ has a direct influence in the calculation of the $\tilde{\mathbf{Y}}^0_p$, as it is directly involved in the calculation of the matrix $\mathbf{G}_{PNMPC}$.

4.3 3rd version: PNMPC getting only the first column of $\mathbf{G}$

As already described in the literature, the matrix $\mathbf{G}$ the one that is responsible to save the values of the dynamics process (Camacho and C., 2007) and

the $\mathbf{G}_{PNMPC}$ is the non-linear version of the dynamic matrix used in the linear predictive control (Plucenio, 2010). One of the possibilities to reduce the computational cost is to calculate only the first column of the matrix $\mathbf{G}$ by the normal procedure of PNMPC and mount the matrix $\mathbf{G}_{PNMPC}$ considering the system time-invariant, this procedure is commonly performed in linear systems (De Keyser, 1998). Thus the non-linear optimization problem can be transformed into a sequence of linear optimization problems. For this it is necessary to check how much the diagonal elements of the matrix $\mathbf{G}$ vary. In the case where the variations are not significant, one can opt for the technique used in linear systems; only calculate the first column of the matrix $\mathbf{G}$ and displaced until a lower triangular matrix is obtained, that means treat the non-linear process as if it were linear. In the case of a non-linear process, it can be seen that the diagonal elements of the matrix $\mathbf{G}$ present significant variations between the elements of the diagonal, however, in the case of a linear process, the diagonals elements of the matrix $\mathbf{G}$ do not present significant variations. To check the degree of the non-linearity in the study case, Figure 4(a) shows the variations in the diagonals elements of the matrix $\mathbf{G}$ calculated by the Jacobian matrix ($\mathbf{G}_{PNMPC}$). As this matrix is recalculated on



(a) Non-linear dynamics matrix (b) Linear dynamics matrix

Figure 4: Dynamic matrix $\mathbf{G}_{PNMPC}$

each iteration, the data shown in Figure 4(a) represent only a particular step, the iteration k = 35. As shown, the diagonal elements of the matrix $\mathbf{G}_{PNMPC}$ present variations, but these variations are small, so they could be disregarded. Applying the third version of the PNMPC algorithm proposed in this paper, where only the first column of the matrix $\mathbf{G}$ is calculated, so that the new matrix $\mathbf{G}$ for iteration k = 35 shown in Figure 4(b). In the Figure 4(b) the first column of the matrix $\mathbf{G}$ is shifted m times, to form a lower triangular matrix, where m is the horizon control. With this modification, the computational cost for the calculation of $\mathbf{G}$ decreases reasonably.

4.4 4th version: PNMPC similar to 3rd version, but now the matrix $\mathbf{G}$ is recalculated only from time to time.

This version is a very interesting and flexible option. The process is treated as if it were linear and only the first column of $\mathbf{G}$ is calculated. However, the matrix $\mathbf{G}$ is not obtained at each iteration, it is recalculated every n sampling times. An operator with knowledge of the process and the non-linearity degree can adjust the value of n in each application. In the case study of this paper, as shown in the previous version, the non-linearity degree of the process is not significant, so the recalculation of the matrix $\mathbf{G}$ does not need to be performed at each iteration. For the testes was used a recalculation every 25 samples. The choice of this time was made taking care not to lose the dynamics of the process.

4.5 5th version: PNMPC looking for free response time to time, using the Cutler idea (Recursive DMC) to the other instants

This version seeks to reduce the computational cost in processes where the non-linearity is not significant, inserting in the PNMPC technique the calculation of free response as is done in the recursive DMC proposed by Cutler. To (Cutler and Ramaker 1988), as shown in (Plucenio et al., 2014), the open-loop predictions for time $t+k$, having information in t and $t-1$ are shown in the equations (22) and (23) respectively:

$$y_0\left(t+k\,|t\right) = \sum_{i=k+1}^{\infty} g_i \Delta u\left(t+k-i\right) \tag{22}$$

$$y_0\left(t+k\,|t-1\right) = \sum_{i=k+2}^{\infty} g_i \Delta u\left(t+k-i\right) \tag{23}$$

The difference between the open-loop prediction in $t+k$ having t, and in $t+k$ having $t-1$, it is only the new control action which is not known in $\Delta\mathbf{u}(t-1)$. Subtracting the equations (22) and (23) has the equation (24):

$$y_0\left(t+k\,|t\right) - y_0\left(t+k\,|t-1\right) = g_{k+1}\Delta u\left(t-1\right) + \sum_{i=k+2}^{\infty} g_i \Delta u\left(t+k-i\right) - \sum_{i=k+2}^{\infty} g_i \Delta u\left(t+k-i\right) \tag{24}$$

which leads to the equation (25):

$$y_0\left(t+k\,|t\right) = g_{k+1}\Delta u\left(t-1\right) + y_0\left(t+k\,|t-1\right) \tag{25}$$

The recursive DMC calculation is executed as follows: Keep in memory a vector with N elements: $\mathbf{Y}_{free} = \left[y_0\left(t\,|t-1\right), ..., y_0\left(t+N-1\,|t-1\right)\right]^T$. It's elements are the free responses given the known control actions at the moment $t-1$. When starting the DMC controller at the moment t_0, and as the system is in steady state, it is considered that the open-loop predictions are constant and equal to a value of $y(t_0)$. When starting the DMC algorithm at the moment t, the vector of the free response must be updated because the change of control action $\Delta\mathbf{u}(t-1)$ it is now known. The vector $\mathbf{Y}_{free}$ of the equation (26) is updated.

$$\mathbf{Y}_{free} = \mathbf{Y}_{free} + \begin{bmatrix} g_1 \\ g_2 \\ \vdots \\ g_N \end{bmatrix} \Delta\mathbf{u}\left(t-1\right) \tag{26}$$

After getting the new control signal $u(t)$, it is necessary to 'displace' the values within the vector $\mathbf{Y}_{free}$. At time $t+1$ the free responses in $t+1$ until $t+N$ is obtained based on the data until t, where N is called

the model horizon. So the first element is discarded because it is the prediction concerning to instant t that has passed. The problem is that making this displacement, the last element ($y(t+N|t)$) is unknown. Therefore, in the stable case $y_0(t+N|t) \cong y_0(t+N-1|t)$, the new vector $\mathbf{Y}_{free}$ will be given by the equation (27):

$$\mathbf{Y}_{free} = \begin{bmatrix} y_0(t+1|t) \\ y_0(t+2|t) \\ \vdots \\ y_0(t+N-1|t) \\ y_0(t+N-1|t) \end{bmatrix} \quad (27)$$

The condition $y_0(t+N|t) \cong y_0(t+N-1|t)$ is shown as:

$$y_0(t+N-1|t) = \sum_{i=N}^{\infty} g_i \Delta u(t+N-1-i)$$

$$y_0(t+N|t) = \sum_{i=N+1}^{\infty} g_i \Delta u(t+N-i) = \sum_{i=N}^{\infty} g_{i+1} \Delta u(t+N-1-i)$$

$$y_0(t+N|t) - y_0(t+N-1|t) = \sum_{i=N}^{\infty} (g_{i+1} - g_i) \Delta u(t+N-1-i)$$

As $g_{i+1} - g_i \cong 0, \forall i > N$, it has $y_0(t+N|t) \cong y_0(t+N-1|t)$, as was able to demonstrate.

Thus, the calculation of free response considered in the DMC Recursive is given by the equation:

$$\mathbf{f} = \begin{bmatrix} y_0(t+1|t) \\ \vdots \\ y_0(t+p|t) \end{bmatrix} + \mathbf{1}_{p\times 1}(y(t) - y_0(t|t)) \quad (28)$$

Where the equation of the model error is:

$$e = y(t) - y_0(t|t) \quad (29)$$

As shown in (Plucenio et al., 2014), the representation of predictions considering the calculation of the free response to the DMC Recursive is shown in the equation (30):

$$\tilde{\mathbf{Y}} = \mathbf{f} + \mathbf{G}\Delta\mathbf{u} \quad (30)$$

where:

- $\tilde{\mathbf{Y}}$ predicted output with dimension $p \times 1$;
- $\mathbf{f}$ free corected response with dimension $p \times 1$;
- $\mathbf{G}$ Jacobian matrix of $\tilde{\mathbf{Y}}$ with dimension $p \times m$;
- $\Delta\mathbf{u}$ vector with dimension $m \times 1$ containing the increments of the control action.

Using this form of implementation, the necessary calculations to obtain the free response become more simple, not requiring the storage of past control increments. This procedure only changes the way to calculate the free response, the rest of the control algorithm is not modified.

4.6 6th version: PNMPC reassessing the matrix $\mathbf{G}$ *and free response according to the prediction error*

The idea of this version is calculate the absolute value of the prediction error, with the objective to fix a range tolerable error in which it is considered that the values obtained are acceptable, and it is not necessary to recalculate the $\mathbf{G}$ matrix and the free response for each iteration.

Table 1: Tuning parameters for the PNMPC controllers

Param	Definition	Value
p	prediction horizon	12
m	control horizon	3
λ	control-weighting factor	3
γ	reference tracking weighting factor	1
α	reference filter	0.25
ϵ	value for variation in control	0.01
a	filter parameter for prediction error	0.15

Table 2: Comparative table of the execution times for the PNMPC versions

Execution	1st version	2nd version	3rt version	4th version	5th version	6th version
1	40.2191	38.1917	25.7825	19.3745	29.4440	27.5302
2	39.5854	38.2922	25.6580	19.1152	29.9883	27.6980
3	38.8933	38.4277	25.6750	19.1802	29.9960	27.6653
4	38.9484	38.1932	25.6655	19.1637	29.2337	27.5607
5	39.1778	38.3610	25.8829	19.0723	29.0735	27.4466
6	39.0862	38.5138	25.6190	19.2308	29.0478	27.6603
7	39.1187	38.6370	25.7077	19.2133	29.0250	27.4830
8	39.1515	38.5807	25.7749	19.1650	29.0478	27.7040
9	39.1868	38.7472	25.6909	19.2164	29.0875	27.6850
10	38.9841	38.4975	25.7159	19.0694	29.4870	27.7875
Tot.time[s]	392.3513	384.4420	257.1723	191.8008	293.4296	276.2206
Averaget.[s]	39.2351	38.4442	25.7172	19.1800	29.3429	27.6220

5 Results

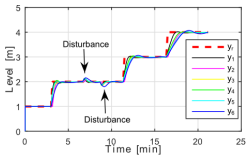
This section presents the results obtained in the different versions implemented with the PNMPC technique. As the main objective is to evaluate the computational cost involved in the implementation of the PNMPC technique, it will be shown the runtime obtained in the calculation of the algorithm for each version of the PNMPC applied in the case study on the nonlinear process of the level control of the conical tank. All simulations were performed in MatLab 2013, on a PC with A8 quad-core processor, AMD with 1.9 GHz and 6GB of RAM, with Windows 8. To first obtain an accurate approximation of the model, the differential equation describing the model was integrated using Ode45 function of MatLab. The tuning parameters used for the PNMPC controllers are the same for all versions for comparison purposes, these values are shown in Table 1. The sample time for all versions is 10 second for reference greater than 1 meter and 1 second for reference less than 1 meter . Table 2 shows runtime the values for each version of the PNMPC technique algorithm implemented in section 4.

To estimate the execution time of each algorithm, each version was executed 10 times and the final time was calculated as the average of 10 executions. The time is measured in seconds and the results are shown in Table 2.

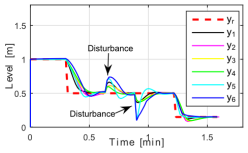
Figure 5 shows the results for two cases. The first for reference greater than 1 meter (Figure 5(a), 5(c) and 5(e)) and the second test for reference less than 1 meter (Figures 5(b), 5(d) and 5(f)).

6 Conclusions

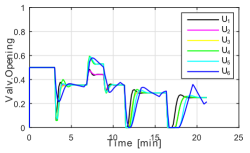
This paper presented six different versions of the PNMPC technique. With little modifications of the complete technique, either in the calculation of the free response or in assembling the matrix G_{PNMPC}, the processing time is reduced. With these modifications the PNMPC can be simplified and be converted into a linear MPC controller. As shown in the results in



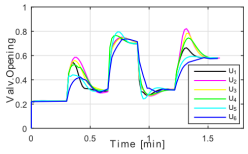
(a) Outputs for ref>1m



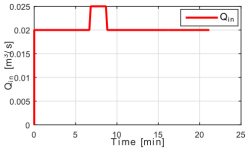
(b) Outputs for ref<1m



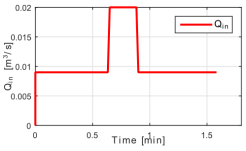
(c) Signals control for ref>1m



(d) Signals control for ref<1m



(e) Disturbance for ref>1m



(f) Disturbance for ref<1m

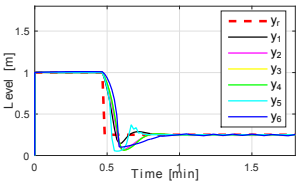
Figure 5: Outputs references, control signals, disturbance

Table 3: Performance index of the PNMPC versions

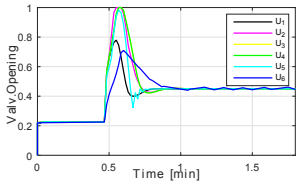
Performance index	V_1	V_2	V_3	V_4	V_5	V_6
MSE	0.0031	0.0040	0.0063	0.0085	0.0100	0.0096
IAE	2.58	3.40	4.56	4.77	5.89	5.07

Figures 5, the six versions implemented can follow reference and reject disturbances in the both tests (reference greater than 1 meter and reference less than 1 meter), however, the PNMPC can also reach the references and reject disturbances more quickly. As it is shown in Table 2 the computational cost is higher due to the complexity of calculation the PNMPC requires. On the other hand, versions 3, 4, 5 and 6 which where were part of the non-linearity process are soft; these controllers do manage to follow the reference and reject disturbance, however, more slowly as compared to the complete PNMPC, which has a lower computational cost.

Figure 6 shows that, for significant height variations, the six versions have a good reference tracking, where the complete PNMPC reacts faster than the linear controllers remembering that the computational cost is greater. Comparing the results, it can be observed that the six implemented algorithms have a very similar response, although the complete PNMPC algorithm provides the best results, where all can react properly to the disturbance, and where, the main difference between them being the runtime. This can be attributed to the "$soft$" non-linearity of the process. Thus, a study of the behavior process can lead to an implementation with gains in terms of processing time without a significantly compromise of the performance. Table 3 shows the comparison of the performance index of MSE (mean squared error) and IAE (integral of the absolute magnitude of the error) for PNMPC versions.



(a) Outputs for significant height variation



(b) Signals control for significant height variation

Figure 6: Outputs and signal control for significant height variation

Acknowledgments

The authors wish to extend his gratitude for the financial support he received from PRH-34 The Agency National of Petroleum (ANP), oil, Gas, and Biofuels.

References

Camacho and C., B. (2007). *Model Predictive Control*, segunda edn.

De Keyser, R. (1998). A gentle introduction to model based predictive control, *PADI2 International Conference on Control Engineering and Signal Processing* .

García, Carlos E.; Prett, D. M. (1989). Model predictive control: Theory and practice-a survey, *International Federation of Automation Control* **25**(3): 14.

Normey, J. E. and Camacho, E. F. (2007). *Control of Dead time processes*.

Plucenio, A. (2010). *Desenvolvimento de técnicas de controle não linear para elevação de fluidos multifásicos*, Dissertação de doutorado-engenharia de automação e sistemas, Centro Tecnológico, Universidade Federal de Santa Catarina, Florianópolis".

Plucenio, A., Campos, M., Vasconcellos, L. and De Gomes, M. (2014). Model predictive control algorithm with stability step, *Congresso Brasileiro de Automática* .

Plucenio, A., J., N., D., P. and A., B. (2007). Controle preditivo não linear na indústria do petróleo e gás, *PDPETRO* .

Qina, S. J. and Badgwell, T. A. (2003). A survey of industrial model predictive control technology, *Control Engineering Practice 11* **11**: 733–764.